CEPH COMO SISTEMA DE ALMACENAMIENTO

Ceph como sistema de almacenamiento.

I.E.S Gonzalo Nazareno

Proyecto Final del Ciclo Superior de Administración de Sistemas Informatico en Red.

Francisco Javier Giménez Vallejo

Ceph como sistema de almacenamiento.

Introducción a Ceph.

Para empezar hay que definir un poco que es Ceph.

Un sistema de almacenamiento distribuido y diseñados bajo la licencia Gpl como software libre.

Permite almacenamiento de objetos, ofrecer dispositivos de bloques para almacenamiento (muy enfocado a cloud), y por último desplegar o implantar un sistema de archivos (que esta aún en desarrollo).

La base de el cluster de almacenamiento es RADOS (Reliable Autonomic Distributed Object Store). Usa CRUSH (Controlled Replication Under Scalable Hashing) como algoritmo que optimiza la ubicación de los datos. La interfaz REST es proporciona por Ceph Object Gateway (RGW) y los discos virtuales por Ceph Block Device (RBD).

Todo esto permite implementar un cluster de almacenamiento, para ello tenemos que tener claro los siguientes conceptos:

OSDs: El Demonio Osd de Ceph almacena datos, gestiona su replicación y recuperación y ofrece información de monitorización. Un cluster de almacenamiento Ceph requiere al menos dos Osds.

MONITORS: Un Monitor Ceph mantiene un mapeo del estado del cluster, incluyendo el mapeo del monitor, el mapeo del Osd entre otros.

MDSs: El servidor de meta-datos de Ceph almacena meta-datos necesarios del sistema de archivos.

Ceph usa el algoritmo CRUSH para distribuir los datos en en cluster de almacenamiento, de modo que el algoritmo se encarga de calcular en que nodo y con que demonio Osd se almacenarán los datos. Dicho algoritmo posibilita que el cluster sea escalable, el balanceo de información y la recuperación dinámica de los datos.

Además hay que especificar que Ceph solo esta soportado por sistemas operativos basados en el kernel de Linux.

Y Por último el hardware necesario no tienen grandes exigencias, por lo cual se usa un hardware común para la implementación de un cluster.

¿Que vamos a montar, y sobre que?.

Vamos a montar cluster en un entorno de prueba de Ceph, para ellos vamos a usar las

siguientes maquinas virtuales con sistema operativo Debian que crearemos en un cloud con OpenStack:

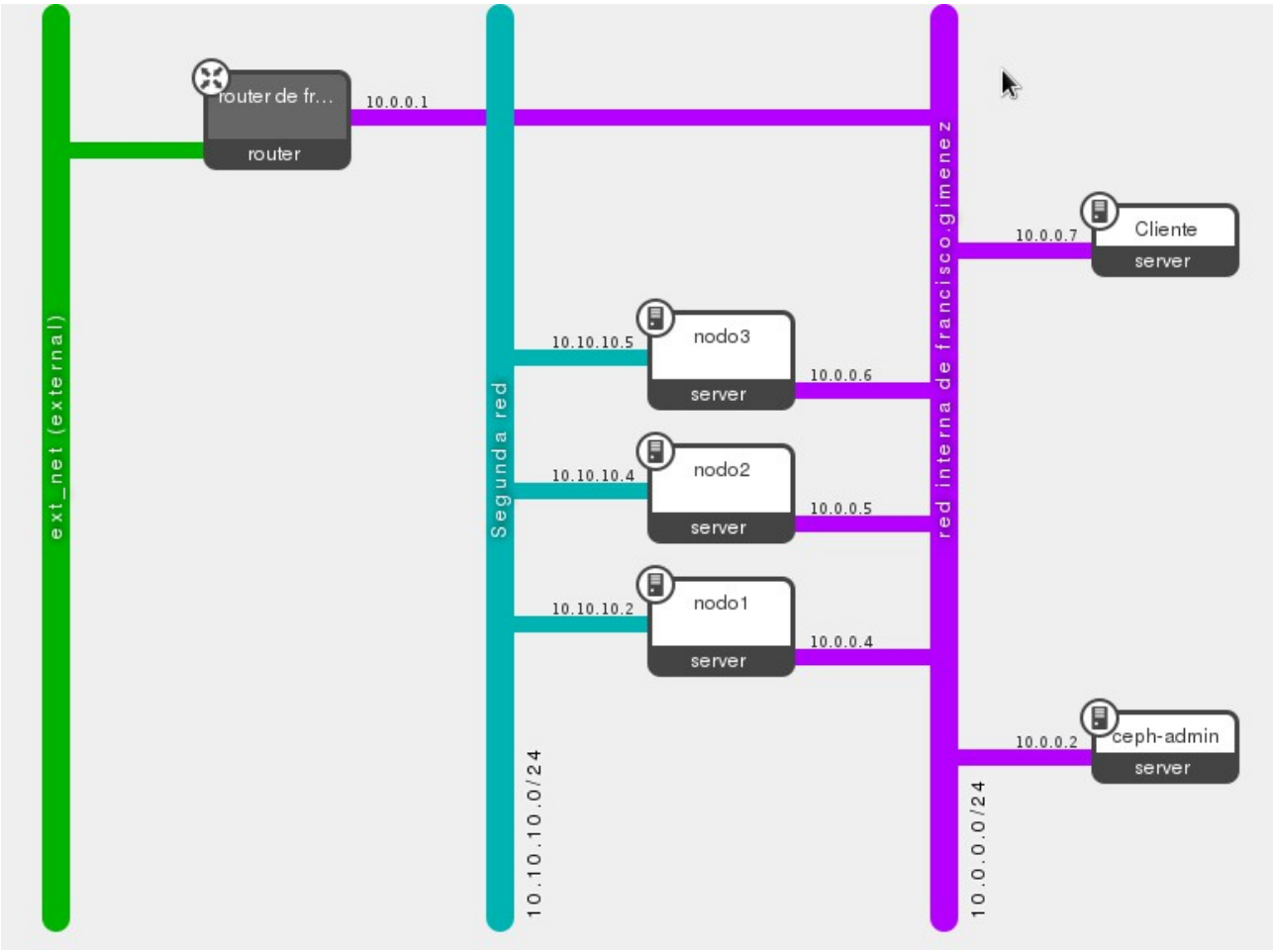
- ceph-admin : Nodo desde el que desplegaremos Ceph a los demás nodos, y admás sera monitor del cluster. Ip 10.0.0.2
- nodo1 : nodo con módulo Osd y volumen de 10Gb. IP 10.0.0.4 - 10.10.10.2
- Nodo2 : nodo con módulo Osd y volumen de 10Gb. IP 10.0.0.5 - 10.10.10.4
- Nodo3 : nodo con módulo Osd y volumen de 10Gb. IP 10.0.0.6 - 10.10.10.5
- Cliente : IP 10.0.0.7

Todas la maquinas tienen un único core con 512Mb de Ram y un disco de 10Gb y un volumen de otros 10 Gb.

Instalación de CEPH.

Pasos Previos.

Redes.



Se va a utilizar una red pública (10.0.0.0/24) donde estarán conectados todas la máquinas del cluster como los futuros clientes que se puedan usar. Vamos a usar una red solo para los Osd del cluster (10.10.10.0/24).

Dns.

Si disponemos de un servidor DNS incluiremos las máquinas para que se resuelvan sus nombres, en el caso de no disponer de el modificaremos el archivo /etc/hosts incluyendo las ip y nombre de máquina de cada una de las que compondrán el cluster..

```
10.0.0.2 ceph-admin
10.0.0.4 nodo1
10.0.0.5 nodo2
10.0.0.6 nodo3
```

Par de claves.

Vamos a Crear un usuario genérico que vamos a llamar ceph, a dicho usuario lo incluiremos en sudoers en cada una de las máquinas, además generaremos par de claves para ssh tanto para el usuario ceph como para el usuario root (al par de claves le dejaremos la frase de paso den blanco).

```
$ ssh-keygen
$ ssh-copy-id ceph@ceph-node2
```

Para que funcione tenemos que repasar la configuración del servidor ssh en el fichero /etc/ssh/sshd_config, donde comprobaremos que se permite el acceso a root si pasword y que el fichero en el que se guarda la clave pública sea el que se especifica en la configuración de ssh

Los Repositorios.

Podemos usar varios métodos a la hora de instalar Ceph, ya sea descargar el código y compilarlo, crear paquetes de instalación y por último utilizar los repositorios oficiales de Ceph. Vamos a explicar los pasos que realizamos a la hora de instalar Ceph con apt y desde los repositorios.

Lo primero que tenemos que hacer es incluir en apt la clave que nos permitirá acceder a los repositorios:

```
wget -q -O- 'https://ceph.com/git/?p=ceph.git;a=blob_plain;f=keys/release.asc' | sudo apt-key add -
```

Ahora actualizamos el fichero sourcelist de la aplicación apt, podemos incluir un fichero nuevo ceph.list que tendrá la información del la url del repositorio en el archivo /etc/apt/sources.list.d/ceph.list.

```
echo deb http://ceph.com/debian-emperor/ $(lsb_release -sc) main | sudo tee
```

```
/etc/apt/sources.list.d/ceph.list
```

O en su defecto modificar directamente el fichero etc/apt/sources.list incluyendo la siguiente linea:

```
deb http://ceph.com/debian-emperor/ wheezy main
```

Podemos especificar la distribución de Ceph que queremos usar, en nuestro caso vamos a usar la última estable que es EMPEROR, simplemente tendremos que escribir el nombre de la distribución que deseemos sustituyéndolo por la cadena de testo emperor.

Se produjo un error al actualizar los paquetes del repositorio que he solucionado con el comando:

```
# apt-get clean
# apt-get update
```

Creamos el Cluster de almacenamiento

Vamos a instalar en el nodo ceph-admin los paquetes necesarios de Ceph para poder desplegarlos posteriormente en los demás nodos, este paquete nos sirve como herramienta de orquestación tipo ansible, chef o pupet.

```
# apt-get install ceph-deploy
```

Ahora iniciamos el cluster y especificamos el primer nodo, como no hemos dado nombre al cluster ceph lo nombra de forma automática. Se generan las claves del cluster y se crea el fichero de configuración.

```
# ceph-deploy new ceph-admin
```

Instalamos los paquetes de ceph en el nodo ceph-admin, de este modo podemos instalar la aplicación de forma remota en cada nodo sin tener que hacerlo de forma local.

```
# ceph-deploy install ceph-admin
```

Especificamos en nuestro cluster el primer nodo que hará de monitor, posteriormente podremos agregar tantos como necesite nuestra infraestructura.

```
# ceph-deploy mon create ceph-admin
```

Sincronizamos el conjunto de claves generando los ficheros con las keyrings, si realizamos un listado de los archivos del directorio donde se guardan los archivos de configuración veremos los archivo generados:

```
# ceph-deploy gatherkeys ceph-admin
root@ceph-admin:/home/ceph/cluster1# ls -l
total 116
-rw-r--r-- 1 root root    72 May  3 23:16 ceph.bootstrap-mds.keyring
-rw-r--r-- 1 root root    72 May  3 23:16 ceph.bootstrap-osd.keyring
-rw-r--r-- 1 root root    64 May  3 23:16 ceph.client.admin.keyring
-rw-r--r-- 1 root root   228 May  3 23:09 ceph.conf
-rw-r--r-- 1 ceph ceph 73535 May  4 11:13 ceph.log
```

```
-rw-r--r-- 1 root root    73 May  3 23:09 ceph.mon.keyring
-rw-r--r-- 1 root root   255 May  3 22:33 cluster.conf
-rw-r--r-- 1 root root  6859 May  3 22:33 cluster.log
-rw-r--r-- 1 root root    73 May  3 22:33 cluster.mon.keyring
-rw-r--r-- 1 root root  1752 May  3 23:10 release.asc
```

Ahora agregamos el resto de nodos al cluster y instalamos ceph.

```
# ceph-deploy new nodo1 nodo2 nodo3
# ceph-deploy install nodo1 nodo2 nodo3
```

a continuación vemos los discos de el nodo1.

```
# ceph-deploy disk list nodo1
```

Preparamos el disco para que ceph lo utilice.

```
# ceph-deploy disk zap nodo1:vdb
```

Generamos un nodo osd con el dispositivo vdb.

```
# ceph-deploy --overwrite-conf osd prepare nodo1:vdb
```

Activamos el nodo y el dispositivo. Realizamos la misma operación con los tres nodos restantes.

```
# ceph-deploy osd activate nodo1:vdb1
```

Podemos ver el estado del cluster.

```
# ceph status
    cluster 6a77e271-0104-4119-b03d-b03f9eff8fab
     health HEALTH_OK
     monmap e1: 1 mons at {ceph-admin=10.0.0.2:6789/0}, election epoch 1, quorum 0 ceph-
admin
     osdmap e13: 3 osds: 3 up, 3 in
      pgmap v23: 192 pgs, 3 pools, 0 bytes data, 0 objects
            105 MB used, 15221 MB / 15326 MB avail
                 192 active+clean
```

Configuración de un Cliente

Una vez creado el cluster configuramos un cliente que se conectará al cluster y usará los recursos que este le ofrece.

```
root@ceph-admin:/home/ceph# ceph-deploy install cliente
[ceph_deploy.conf][DEBUG ] found configuration file at: /root/.cephdeploy.conf
[ceph_deploy.cli][INFO  ] Invoked (1.5.1): /usr/bin/ceph-deploy install cliente
[ceph_deploy.install][DEBUG ] Installing stable version emperor on cluster ceph hosts
cliente
[ceph_deploy.install][DEBUG ] Detecting platform for host cliente ...
root@cliente's password:
[cliente][DEBUG ] connected to host: cliente
[cliente][DEBUG ] detect platform information from remote host
[cliente][DEBUG ] detect machine type
```

```
[ceph_deploy.install][INFO  ] Distro info: debian 7.2 wheezy
[cliente][INFO  ] installing ceph on cliente
[cliente][INFO  ] Running command: env DEBIAN_FRONTEND=noninteractive apt-get -q install
--assume-yes ca-certificates
...
...
[cliente][DEBUG ] Setting up ceph-mds (0.72.2-1~bpo70+1) ...
[cliente][DEBUG ] Setting up dmsetup (2:1.02.74-8) ...
[cliente][DEBUG ] update-initramfs: deferring update (trigger activated)
[cliente][DEBUG ] Processing triggers for initramfs-tools ...
[cliente][DEBUG ] update-initramfs: Generating /boot/initrd.img-3.2.0-4-amd64
[cliente][INFO  ] Running command: ceph --version
[cliente][DEBUG ] ceph version 0.72.2 (a913ded2ff138aefb8cb84d347d72164099cfd60)
Unhandled exception in thread started by
sys.excepthook is missing
lost sys.stderr
```

Una vez realizada la instalción le copiamos la configuración y las claves al cliente.

```
root@ceph-admin:/home/ceph/cluster1# ceph-deploy admin cliente
[ceph_deploy.conf][DEBUG ] found configuration file at: /root/.cephdeploy.conf
[ceph_deploy.cli][INFO  ] Invoked (1.5.1): /usr/bin/ceph-deploy admin cliente
[ceph_deploy.admin][DEBUG ] Pushing admin keys and conf to cliente
root@cliente's password:
[cliente][DEBUG ] connected to host: cliente
[cliente][DEBUG ] detect platform information from remote host
[cliente][DEBUG ] detect machine type
[cliente][DEBUG ] get remote short hostname
[cliente][DEBUG ] write cluster configuration to /etc/ceph/{cluster}.conf
root@ceph-admin:/home/ceph/cluster1#
```

Ya podemos crear un dipositivo de bloques desde el cliente, para ello empezamos cargando el módulo rdb, posteriormente creamos un dispositivo de bloques de 500Mb

```
root@cliente:/etc/ceph# modprobe rbd
root@cliente:/etc/ceph# rbd create block1-ceph-cliente --size 500
root@cliente:/etc/ceph# rbd map block1-ceph-cliente
root@cliente:/etc/ceph# rbd showmapped
id pool image               snap device
0  rbd  block1-ceph-cliente -    /dev/rbd0
```

```
root@cliente:/etc/ceph# mkfs.ext4 /dev/rbd/rbd/block1-ceph-cliente
mke2fs 1.42.5 (29-Jul-2012)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=4096 blocks, Stripe width=4096 blocks
128016 inodes, 512000 blocks
25600 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67633152
63 block groups
8192 blocks per group, 8192 fragments per group
2032 inodes per group
Superblock backups stored on blocks:
       8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
root@cliente:/etc/ceph# mkdir /mnt/rdb1
root@cliente:/etc/ceph# mount /dev/rbd/rbd/block1-ceph-cliente /mnt/rdb1/
root@cliente:/etc/ceph# df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          9.9G  1.3G  8.1G  14% /
udev             10M     0   10M   0% /dev
tmpfs            50M  128K   50M   1% /run
/dev/vda1       9.9G  1.3G  8.1G  14% /
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           100M     0  100M   0% /run/shm
/dev/rbd0       485M   11M  449M   3% /mnt/rdb1
```

Comandos.

Vamos a ver algunos comandos útiles del manejo de dispositivos de bloques RBD (Rados Block Device).

La sintaxis es la siguiente:

rdb [Opciones...] [Parámetros...] [Comandos...]

rbd [-c *ceph.conf*] [-m *monaddr*] [-p | --pool *pool*] [–size *size*] [–order *bits*] [*command* ...]

Opciones

-c ceph.conf, --conf ceph.conf
Usa el archivo ceph.conf en lugar del asignado por defecto /etc/ceph/ceph.conf al determinar la dirección del monitor principal.

-m monaddress[:port]
Conectarse a un monitor especifico (en lugar del especificado en ceph.conf).

-p pool, --pool pool
Interactua con un pool específico.

--no-progress
No muestra información de progreso.

Parámetros

--image-format format.
especifica el formato final del objeto. El valor por defecto es 1.

format 1 – Usa el formato original para una nueva imagen rbd. Este formato es soportado por todas las versiones de librbd y el modulo rbd del kernel, pero no soporta nuevas características como la clonación.

format 2 – Se usa el segundo formato rbd que soporta librbd pero no el módulo rbd del kernel. Así se consigue la clonación y es más propenso a soportar nuevas características.

--size size-in-mb

Especifica el tamaño (en megabytes) de una imagen rbd.

--stripe-unit size-in-bytes

Especifica la unidad de stripe en bytes.

--stripe-count num

Especifica el número de objetos para stripe.

--snap snap

Aplica un nombre al snaphot.

--id username

Especifica un nombre de usuario.

--keyfile filename

Especifica el fichero que contiene las claves a usar en al comando. Si no se específica se usará el client.admin por defecto.

--keyring filename

Especifica el fichero keyring que contiene las claves para el usuario especificado. Si no se buscara el kayring por defecto.

--format format

Especifica el formato de salida (por defecto: plain, json, xml)

--pretty-format

Genera un archivo con formato xml o json tabulado para una lectura más sencilla.

Comandos

ls [-l | –long] [pool-name]

Muestra un lista de las imágenes rbd en el rbd_directorio(pool). Con la opción -l, muestra snapshots, y incluye más información como el tamaño, padre (si es clonado), formato, etc...

info [image-name]

Muestra información de la imagen especificada.

create [image-name]

Crea una nueva imagen rbd. Se puede especificar el tamaño –size, los argumentos

-stripe-unit y -stripe-count son opcionales.

clone [parent-snapname] [image-name]
Creará un clon de un snapshot padre. El hijo mantendrá dependencias con e snapshot padre, por tanto este debe estar protegido, requiere formato 2.

flatten [image-name]
Realiza una clonación el la que se copian todos los bloques de datos compartidos de tal modo que el hijo es independiente del padre. El snapshot padre puede ser desbloqueado y borrado si no quedan más dependencias, requiere formato 2.

children [image-name]
Lista los clones del snaphot especificado, requiere formato 2.
Requiere image format 2.

resize [image-name] [–allow-shrink]
Redimensiona una imagen rbd.
rm [image-name]
Borra una imagen rbd.

export [image-name] [dest-path]
Exporta una imagen a un destino especificado.

import [path] [dest-image]
Crea una imagen importando los datos de un origen especificado.

export-diff [image-name] [dest-path] [–from-snap snapname]

Exporta una diferencia incremental de una imagen dado un snapshot inicial.

import-diff [src-path] [image-name]
Importa una diferencia incremental de una imagen dado un snapshot inicial.

diff [image-name] [–from-snap snapname]
Devuelve un listado de tamaño de los cambios en una imagen dado una snapshot de inicio.

cp [src-image] [dest-image]
Copia el contenido de una imagen en otra de destino que han de tener el mismo tamaño, formato ...

mv [src-image] [dest-image]
Renombra una imagen.

snap ls [image-name]
Devuelve una lista de snapshots en una imagen especificada.

snap create [image-name]

Crea un nuevo snapshot.

snap rollback [image-name]
Devuelve una imagen al estado en el que se creo un snapshot.

snap rm [image-name]
Borra un snapshot especificado.

snap purge [image-name]
Elimina todos los snapshots de una imagen.

snap protect [image-name]
Proteger un snapshot del borrado, de modo que se pueden realizar clones de él. L protección implica que existen clones dependientes del snapshot.

snap unprotect [image-name]
Desbloquea un snapshot para el borrado.
map [image-name] [-o | –options map-options] [–read-only]
Mapea una imagen específica a un dispositivo de bloques vía modulo rbd del kernel.

unmap [device-path]
Desmapea el dispositivo de bloques que fue mapeado vía modulo rbd del kernel.

showmapped
Muestra las imágenes mapeadas vía modulo rbd del kernel.

Ejemplos de comandos.

Unos ejemplos de comandos.

Listar dispositivos de bloques: rbd ls

```
root@ceph-admin:/home/ceph/cluster1# rbd ls
block1-ceph-cliente
```

Información de un dispositivo de bloques: rbd –image block-cliente info

```
root@ceph-admin:/home/ceph/cluster1# rbd --image block1-ceph-cliente info
rbd image 'block1-ceph-cliente':
       size 500 MB in 125 objects
       order 22 (4096 kB objects)
       block_name_prefix: rb.0.107a.74b0dc51
       format: 1
```

Redimensionar dispositivo de bloques: rbd --resize block-cliente info --size 1024

```
root@ceph-admin:/home/ceph/cluster1# rbd resize --image block1-ceph-cliente --size 1024
Resizing image: 100% complete...done.
root@ceph-admin:/home/ceph/cluster1# rbd --image block1-ceph-cliente infor
bd image 'block1-ceph-cliente':
```

```
        size 1024 MB in 256 objects
        order 22 (4096 kB objects)
        block_name_prefix: rb.0.107a.74b0dc51
        format: 1
```

Hemos redimensionado la imagen o dispositivo de bloques ahora vamos a redimensionar el sistema de archivos y comprobamos que se ha realizado correctamente.

```
root@cliente:/home# e2fsck -f /dev/rbd0
e2fsck 1.42.5 (29-Jul-2012)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/rbd0: 12/128016 files (0.0% non-contiguous), 26667/512000 blocks

root@cliente:/home# resize2fs /dev/rbd/rbd/block1-ceph-cliente 1024M
resize2fs 1.42.5 (29-Jul-2012)
Resizing the filesystem on /dev/rbd/rbd/block1-ceph-cliente to 1048576 (1k) blocks.
The filesystem on /dev/rbd/rbd/block1-ceph-cliente is now 1048576 blocks long.

root@cliente:/home# mount /dev/rbd/rbd/block1-ceph-cliente /mnt/rdb1/

root@cliente:/home# df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          9.9G  1.3G  8.1G  14% /
udev             10M     0   10M   0% /dev
tmpfs            50M  128K   50M   1% /run
/dev/vda1       9.9G  1.3G  8.1G  14% /
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           100M     0  100M   0% /run/shm
/dev/rbd0       992M   11M  931M   2% /mnt/rdb1
```

Como almacena Ceph información en el Cluster.

Para empezar debemos familiarizarnos con algunos conceptos, como son *object*, *rep size*, *pg* y *pool*.

“Object” (objetos), son la menor unidad de almacenamiento, cada cosa se almacena en forma de objetos, es por esto por lo que un cluster Ceph es conocido también como cluster de almacenamiento de objetos. Los objetos son mapeados a PG, y luego se distribuyen las copias en los distintos OSD.

“rep size” (tamaño de réplicas), número de replicas que tendrá un objeto.

“PG” (Placement group o grupod de ubicación), en un cluster Ceph se relacionan los objetos con los pg. Los pg contienen los objetos de los que se harán copias distribuidas en los distintos OSD mejorando de este modo su fiabilidad.

“Pools” (piscina, pozo, fuente) son el grupo lógico de almacenamiento de objetos. Están compuestos de pg. Cuando se crean las pools se ha de proporcionar el número de pg que contendrá cada una, el “rep size” (numero de replicas de los objetos). En el caso de

que no se definan tomarán los valores por defecto.

Vamos a crear pool-1 con un número de pg de 128.

```
root@ceph-admin:/home/ceph/cluster1# ceph osd pool create pool-1 128
pool 'pool-1' created
```

Listamos los pool que están definidos.

```
root@ceph-admin:/home/ceph/cluster1# ceph osd lspools
0 data,1 metadata,2 rbd,3 pool-1,
```

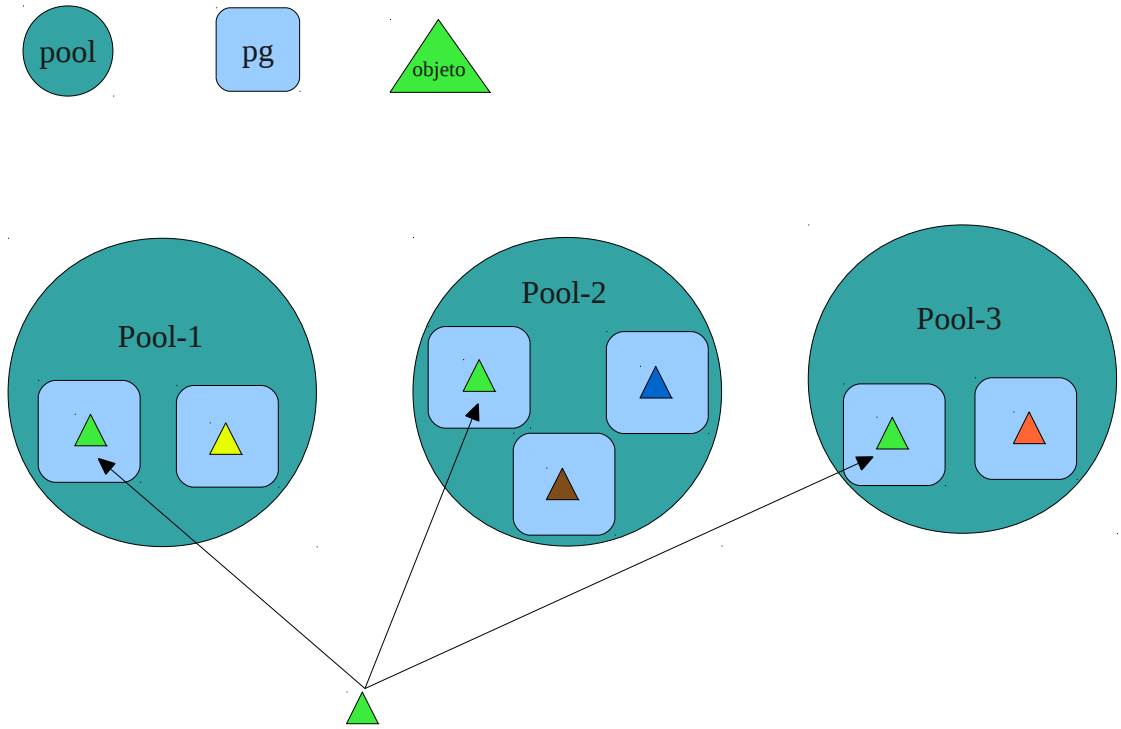
Volcamos información de pool-1 donde vemos rep size y pg_num.

```
root@ceph-admin:/home/ceph/cluster1# ceph osd dump | grep -i pool-1
pool 3 'pool-1' rep size 2 min_size 1 crush_ruleset 0 object_hash rjenkins pg_num 128
pgp_num 128 last_change 35 owner 0
```

Cambiamos pg_num a 3.

```
root@ceph-admin:/home/ceph/cluster1# ceph osd pool set pool-1 size 3
set pool 3 size to 3
root@ceph-admin:/home/ceph/cluster1# ceph osd dump | grep -i pool-1
pool 3 'pool-1' rep size 3 min_size 1 crush_ruleset 0 object_hash rjenkins pg_num 128
pgp_num 128 last_change 37 owner 0
```

Veamos una representación de como se almacenaría el objeto .



Vemos que el objeto en cuestión se almacena en 3 veces como hemos definido en el campo “rep size” en distintos pools ubicados en osd distintos. Por lo tanto cada objeto de pool-1 se replicaran tres veces en tres osd distintos.

Creamos un objeto object-A.

```
root@ceph-admin:/home/ceph# dd if=/dev/zero of=object-A bs=10M count=1
1+0 records in
1+0 records out
10485760 bytes (10 MB) copied, 0.0408887 s, 256 MB/s
```

Ubicamos dicho objeto en el pool-1.

```
root@ceph-admin:/home/ceph# rados -p pool-1 put object-A  object-A
root@ceph-admin:/home/ceph# rados -p pool-1 ls
object-A
```

Vemos la información del mapeo del objeto.

```
root@ceph-admin:/home/ceph#  ceph osd map pool-1 object-A
osdmap e38 pool 'pool-1' (3) object 'object-A' -> pg 3.b301e3e8 (3.68) -> up [0,1,2]
acting [0,1,2]
```

De esta información podemos distinguir los siguientes datos:

- Versión del osdmap “osdmap e38”.
- Nombre del pool con id 3 “pool 'pool-1' (3)”
- Nombre del objeto “object 'object-A'”
- Pg al que pertenece el objeto “pg 3.b301e3e8 (3.68)”
- Osd en los que se almacena el objeto “up [0,1,2] acting [0,1,2]”

Nos desplazamos a la dirección donde se ha almacenado el objeto en el osd y listamos el contenido.

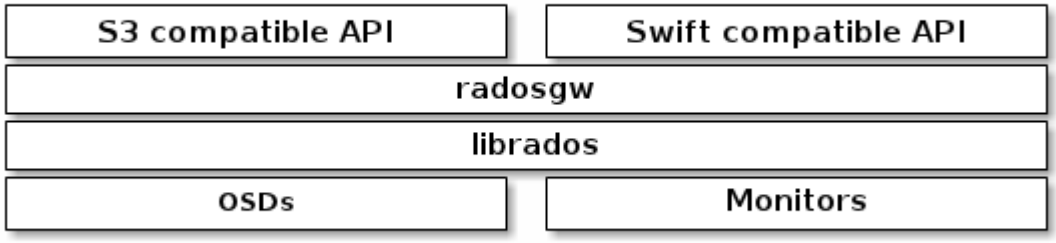
```
root@nodo1:/home/ceph# cd /var/lib/ceph/osd/ceph-0/current/
root@nodo1:/var/lib/ceph/osd/ceph-0/current# ls -la |grep 3.68
drwxr-xr-x   2 root root    39 May 25 14:11 3.68_head
root@nodo1:/var/lib/ceph/osd/ceph-0/current# cd 3.68_head/
root@nodo1:/var/lib/ceph/osd/ceph-0/current/3.68_head# ls -la
total 16396
drwxr-xr-x   2 root root       39 May 25 14:11 .
drwxr-xr-x 324 root root     8192 May 25 13:43 ..
-rw-r--r--   1 root root 10485760 May 25 14:11 object-A__head_B301E3E8__3
```

Ceph Object Gateway

Ceph object gateway es una interfaz de almacenamiento de objetos construida sobre librgw para proporcionar una interfaz de acceso al cluster de almacenamiento.

Hasta ahora proporciona compatibilidad con dos interfaces que son S3 (Amazon) y Swift (Open Stack).

Ceph object gateway usa el demonio de almacenamiento de objetos radosgw que es un módulo Fastcgi para interactuar con el cluster de almacenamiento.



Instalación de ceph object gateway

Lo primero que tenemos que hacer es actualizar nuestros repositorios, aunque en los repositorios de Debian encontramos los paquetes de apache y Fastcgi, en los repositorios de Ceph podemos encontrar una versión que es compatible con 100-continue HTTP1 , que es la que es la que se usa en la documentación oficial de Ceph.

```
root@ceph-admin:/etc/apt/sources.list.d# wget -q -O-
https://raw.github.com/ceph/ceph/master/keys/autobuild.asc | sudo apt-key add -

OK
```

```
root@ceph-admin:/home/ceph# echo deb http://gitbuilder.ceph.com/libapache-mod-fastcgi-
deb-$(lsb_release -sc)-x86_64-basic/ref/master $(lsb_release -sc) main | sudo tee
/etc/apt/sources.list.d/ceph-fastcgi.list
deb http://gitbuilder.ceph.com/libapache-mod-fastcgi-deb-wheezy-x86_64-basic/ref/master
wheezy main
```

```
root@ceph-admin:/etc/apt/sources.list.d# echo deb http://gitbuilder.ceph.com/apache2-deb-
$(lsb_release -sc)-x86_64-basic/ref/master $(lsb_release -sc) main | sudo tee
/etc/apt/sources.list.d/ceph-apache.list
deb http://gitbuilder.ceph.com/apache2-deb-wheezy-x86_64-basic/ref/master wheezy main
```

Actualizamos y instalamos los paquetes necesario de apache y Fastcgi.

```
root@ceph-admin:/home/ceph/cluster1# apt-get update && sudo apt-get install apache2
libapache2-mod-fastcgi
```

Vemos FQDN del equipo, para especificarlo en el archivo de configuración de apache.

```
root@ceph-admin:/home/ceph/cluster1# hostname -f
ceph-admin.novalocal
root@ceph-admin:/home/ceph/cluster1# nano /etc/apache2/apache2.conf
ServerName ceph-admin.novalocal
```

Iniciamos los módulos rewrite y fastcgi para reiniciar posteriormente el servicio apache.

```
root@ceph-admin:/home/ceph/cluster1# a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  service apache2 restart
root@ceph-admin:/home/ceph/cluster1# a2enmod fastcgi
Module fastcgi already enabled
root@ceph-admin:/home/ceph/cluster1# service apache2 restart
[ ok ] Restarting web server: apache2 ... waiting .
```

A continuacio instalamos el paquete radosgw

```
root@ceph-admin:/home/ceph/cluster1# apt-get install radosgw
```

Configuramos radosgw

Ahora generamos el Keyring para radosgw.

```
root@ceph-admin:/home/ceph/cluster1#           ceph-authtool           --create-keyring
/etc/ceph/ceph.client.radosgw.keyring
creating /etc/ceph/ceph.client.radosgw.keyring
root@ceph-admin:/home/ceph/cluster1# chmod +r /etc/ceph/ceph.client.radosgw.keyring
```

Creamos el usuario para el Ceph Object Gateway. Usaremos el nombre `gateway` tras `client.radosgw.`

```
root@ceph-admin:/home/ceph/cluster1# ceph-authtool /etc/ceph/ceph.client.radosgw.keyring
-n client.radosgw.gateway –gen-key
```

Le asignamos propiedades a la clave, nuestro caso rwx en los osd y rw en monitores,

```
root@ceph-admin:/home/ceph/cluster1# ceph-authtool -n client.radosgw.gateway --cap osd
'allow rwx' --cap mon 'allow rw' /etc/ceph/ceph.client.radosgw.keyring
```

Agregamso la clave creada a nuestro cluster.

```
root@ceph-admin:/home/ceph/cluster1# ceph -k /etc/ceph/ceph.client.admin.keyring auth add
client.radosgw.gateway -i /etc/ceph/ceph.client.radosgw.keyring
added key for client.radosgw.gateway
```

Distribuimos la clave en los nodos.

```
sudo scp /etc/ceph/ceph.client.radosgw.keyring  ceph@nodo1:/home/ceph
ssh nodo1
sudo mv ceph.client.radosgw.keyring /etc/ceph/ceph.client.radosgw.keyring
sudo scp /etc/ceph/ceph.client.radosgw.keyring  ceph@nodo2:/home/ceph
ssh nodo2
sudo mv ceph.client.radosgw.keyring /etc/ceph/ceph.client.radosgw.keyring
sudo scp /etc/ceph/ceph.client.radosgw.keyring  ceph@nodo3:/home/ceph
ssh nodo3
sudo mv ceph.client.radosgw.keyring /etc/ceph/ceph.client.radosgw.keyring
```

Editamos el fichero de configuracio de deph para incluir:

```
[client.radosgw.gateway]
```

```
host = ceph-admin
keyring = /etc/ceph/ceph.client.radosgw.keyring
rgw socket path = /var/run/ceph/ceph.radosgw.gateway.fastcgi.sock
log file = /var/log/ceph/client.radosgw.gateway.log
```

Agregamos el siguiente script en el archivo s3gw.fcgi

```
root@ceph-admin:/home/ceph/cluster1# nano /var/www/s3gw.fcgi
#!/bin/sh
exec /usr/bin/radosgw -c /etc/ceph/ceph.conf -n client.radosgw.gateway
```

Damos permiso de ejecución a s3gw.fcgi y cambiamos el propietario al usuario ceph.

```
root@ceph-admin:/home/ceph/cluster1# chmod +x /var/www/s3gw.fcgi
root@ceph-admin:/home/ceph/cluster1# chown ceph:ceph /var/www/s3gw.fcgi
```

Creamos el virtual host rgw.conf.

```
root@ceph-admin:/home/ceph/cluster1# nano /etc/apache2/sites-available/rgw.conf

FastCgiExternalServer /var/www/s3gw.fcgi -socket
/var/run/ceph/ceph.radosgw.gateway.fastcgi.sock

<VirtualHost *:80>

        ServerName ceph-admin.novalocal
        ServerAdmin fjjaviergv@yahoo.es
        DocumentRoot /var/www
        RewriteEngine On
        RewriteRule  ^/(.*) /s3gw.fcgi?%{QUERY_STRING} [E=HTTP_AUTHORIZATION:%
{HTTP:Authorization},L]

        <IfModule mod_fastcgi.c>
        <Directory /var/www>
                        Options +ExecCGI
                        AllowOverride All
                        SetHandler fastcgi-script
                        Order allow,deny
                        Allow from all
                        AuthBasicAuthoritative Off
                </Directory>
        </IfModule>

        AllowEncodedSlashes On
        ErrorLog /var/log/apache2/error.log
        CustomLog /var/log/apache2/access.log combined
        ServerSignature Off

</VirtualHost>
```

Lo activamos y lo hacemos que sea la pagina por defecto de apache.

```
root@ceph-admin:/home/ceph/cluster1# sudo a2ensite rgw.conf
Enabling site rgw.conf.
To activate the new configuration, you need to run:
  service apache2 reload
root@ceph-admin:/home/ceph/cluster1# a2dissite default
Site default disabled.
To activate the new configuration, you need to run:
```

```
  service apache2 reload
```

Iniciamos el servicio radosgw.

```
root@ceph-admin:/home/ceph/cluster1# /etc/init.d/radosgw start
Starting client.radosgw.gateway...
2014-06-01   17:01:25.036451   7f2f9272e780   -1   WARNING:   libcurl   doesn't   support
curl_multi_wait()
2014-06-01  17:01:25.036458  7f2f9272e780  -1  WARNING:  cross  zone  /  region  transfer
performance may be affected
```

Realizamos una comprobación para ver que esta funcionando correctamente, para ello ejecutamos una petición GET anónima a la url de radosgw y vemos el fichero que devuelve.

```
javi@debian:~$ wget http://172.22.196.80
--2014-06-01 16:58:17--  http://172.22.196.80/
Conectando con 172.22.196.80:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: no especificado [application/xml]
Grabando a: “index.html”

    [ <=>                                   ] 214         --.-K/s   en 0s

2014-06-01 16:58:19 (6,32 MB/s) - “index.html” guardado [214]

javi@debian:~$nano index.html
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
        <Owner>
                <ID>anonymous</ID>
                <DisplayName></DisplayName>
        </Owner>
<Buckets></Buckets>
</ListAllMyBucketsResult>
```

Creación de usuarios y claves

Creamos un usuairo para Radosgw S3 y Swift, para S3 creamos el usuario “usuario” y para Swift el subusuario “usuario:swift”.

Para S3

```
root@ceph-admin:/home/ceph/cluster1# radosgw-admin user create --uid=usuario --display-
name=usuario
{ "user_id": "usuario",
  "display_name": "usuario",
  "email": "",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [],
  "keys": [
        { "user": "usuario",
          "access_key": "ASBARC78L1HVYZ7DPU3X",
          "secret_key": "ItIot8Jq4kd0IYZfb9bYty0LR5OS86k1fyXui3Q9"}],
  "swift_keys": [],
  "caps": [],
```

```
  "op_mask": "read, write, delete",
  "default_placement": "",
  "placement_tags": [],
  "bucket_quota": { "enabled": false,
      "max_size_kb": -1,
      "max_objects": -1}}
```

Para Swift

```
root@ceph-admin:/home/ceph/cluster1#   radosgw-admin   subuser   create   --uid=usuario
--subuser=usuario:swift --access=full
{ "user_id": "usuario",
  "display_name": "usuario",
  "email": "",
  "suspended": 0,
  "max_buckets": 1000,
 "auid": 0,
  "subusers": [
     { "id": "usuario:swift",
         "permissions": "full-control"}],
  "keys": [
       { "user": "usuario",
         "access_key": "ASBARC78L1HVYZ7DPU3X",
         "secret_key": "ItIot8Jq4kd0IYZfb9bYty0LR50S86k1fyXui3Q9"}],
  "swift_keys": [],
  "caps": [],
  "op_mask": "read, write, delete",
  "default_placement": "",
  "placement_tags": [],
  "bucket_quota": { "enabled": false,
      "max_size_kb": -1,
      "max_objects": -1}}
```

Creamos la clave para swift.

```
radosgw-admin key create --subuser=usuario:swift --key-type=swift –gen-secret

root@ceph-admin:/home/ceph/cluster1# radosgw-admin user info --uid=usuario
{ "user_id": "usuario",
  "display_name": "usuario",
  "email": "",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [
    { "id": "usuario:swift",
         "permissions": "full-control"}],
  "keys": [
       { "user": "usuario",
         "access_key": "ASBARC78L1HVYZ7DPU3X",
         "secret_key": "ItIot8Jq4kd0IYZfb9bYty0LR50S86k1fyXui3Q9"}],
  "swift_keys": [
       { "user": "usuario:swift",
         "secret_key": "0ZuWUJifolz+qpbnvXI1lQEJN++AhGzFexF0Il4z"}],
  "caps": [],
  "op_mask": "read, write, delete",
  "default_placement": "",
  "placement_tags": [],
  "bucket_quota": { "enabled": false,
      "max_size_kb": -1,
      "max_objects": -1}}
```

Ahora en el equipo que vamos a usar como cliente instalmos el cliente swift.

Insatalación y uso de el cliente Swift

```
 root@cliente:~#pip install python-swiftclient
```

Realizamos una consulta para ver el estado. Para realizar la consulta usamos la siguiente sintaxis: swift -A [url] -U [usuario] -k [clave] [comandos].

- -A para solicitar tokken con la url especificada.
- -U para especificar el usuario.
- -K la clave del usuario.
- Por último los comandos de swift que queramos usar.

```
root@cliente:~#    swift    -A    http://172.22.196.80/auth    -U    usuario:swift    -K
0ZuWUJifolz+qpbnvXI1lQEJN++AhGzFexF0Il4z stat --info
       Account: v1
    Containers: 0
       Objects: 0
         Bytes: 0
          Vary: Accept-Encoding
        Server: Apache/2.2.22 (Debian)
X-Account-Bytes-Used-Actual: 0
  Content-Type: text/plain; charset=utf-8
```

Creamos un contenedor.

```
root@cliente:~#    swift    -A    http://172.22.196.80/auth    -U    usuario:swift    -K
0ZuWUJifolz+qpbnvXI1lQEJN++AhGzFexF0Il4z post contenedor1
```

Ahora creamos un objeto, en concreto un archivo que vamos a llamar objeto1.

```
dd if=/dev/zero of=objeto-1 bs=10M count=1
1+0 records in
1+0 records out
10485760 bytes (10 MB) copied, 0.028108 s, 373 MB/s
```

Subimos el objeto al contenedor que hemos creado previamente.

```
root@cliente:~#    swift    -A    http://172.22.196.80/auth    -U    usuario:swift    -K
0ZuWUJifolz+qpbnvXI1lQEJN++AhGzFexF0Il4z upload contenedor1 objeto-1
objecto-1
```

Vemos ahora el estado.

```
root@cliente:~#    swift    -A    http://172.22.196.80/auth    -U    usuario:swift    -K
0ZuWUJifolz+qpbnvXI1lQEJN++AhGzFexF0Il4z stat --info
       Account: v1
    Containers: 1
       Objects: 1
         Bytes: 10485760
          Vary: Accept-Encoding
```

```
        Server: Apache/2.2.22 (Debian)
X-Account-Bytes-Used-Actual: 10485760
  Content-Type: text/plain; charset=utf-8
```

Vemos una lista de contenedores y objetos almacenados.

```
root@cliente:~#    swift    -A    http://172.22.196.80/auth    -U    usuario:swift    -K
0ZuWUJifolz+qpbnvXI1lQEJN++AhGzFexF0Il4z list
contenedor1
root@cliente:~#    swift    -A    http://172.22.196.80/auth    -U    usuario:swift    -K
0ZuWUJifolz+qpbnvXI1lQEJN++AhGzFexF0Il4z list contenedor1
objeto-1
```

Para descargar un objeto ejecutamos la siguiente sentencia.

```
root@cliente:~#    swift    -A    http://172.22.196.80/auth    -U    usuario:swift    -K
0ZuWUJifolz+qpbnvXI1lQEJN++AhGzFexF0Il4z download contenedor1 objeto-1
objecto-1 [auth 2.080s, headers 3.260s, total 3.818s, 6.036 MB/s]
```

Para ver mas opciones o la ayuda del comando swift ejecutamos swift --help

Complementos para Ceph (dashboard)

Existen algunos complementos para Ceph que se pueden agregar a nuestra implementación. Algunos de estos complemetnos son Dashboards que nos daran una interfaz gráfica para monitorizar nuestra infrestructura. A continuación dejo dos enlaces en Github donde nosmuestra como realizar la instalción de el software:

https://github.com/Crapworks/ceph-dash

https://github.com/krakendash/krakendash

Conclusión

Ceph es un sistema de almacenamiento diseñado para el uso con gran cantidad de datos, esta muy enfocado para el uso con Big Data como con cloud (en la actualidad soporta las api de Swift (OpenStack) y Amazon (S3)), donde permite tener almacenado las imágenes, instancias o datos con un sistema redundante si que llegue a ser muy complejo su configuración.

Además las ultimas versiones incluye un orquestador (ceph-deploy) que facilita mucho la instalación y la configuración de los monitores y osd.

Una de las grandes ventajas que tiene es la flexibilidad y elasticidad del sistema de almacenamiento para redimensionarse, ya que podemos aumentar la capacidad de almacenamiento de un modo sencillo, simplemente necesitamos incluir tantos servidores nuevos como necesitamos. Esta característica hace que se un alternativa a

las redes de almacenamiento SAN (más rígidas y costosas a la hora de aumentar su capacidad de almacenamiento).

Gracias a la mejora de la tecnología de redes es posible usar redes ethernet 10G para la comunicación entre osd y monitores de modo que no se penalice la velocidad de transferencia de los datos acercándose a las conexiones fiberchanel que se usan en redes SAN.

Definitivamente podemos decir que Ceph es una alternativa real a los grandes sistemas de almacenamiento, permitiendo tener implementada una solución que nos ofrezca las mismas características que una red SAN por un coste menor, siendo además más fácil, sencillo y económico aumentar nuestra infraestructura para satisfacer futuras necesidades de almacenado.

Agradecimientos.

He de dar a gracias a mi familia que durante estos dos , han estado a mi lado y me han servido de apoyo en los momentos más difíciles. Sin ellos no podría haberlos conseguido.

No quiero olvidarme de ninguno de los profesores del ciclo, que de un modo u otro han sabido motivarme e ilusionarme durante estos dos años.

Ha todos muchas gracias.

Bibliografía.

http://ceph.com

http://docs.openstack.org/

http://docs.openstack.org/user-guide/content/

http://s3tools.org/s3cmd

http://iesgn.github.io/cloud/