Balanceo de carga de aplicaciones Web con Apache mod_proxy, empleando almacenamiento compartido basado en iSCSI

IAGO SEARA VICENTE - (39492835H)
XIAN JIMÉNEZ NIETO - (53192437S)
DIEGO BARREIRA BARREIRA - (53193549T)
LAURA ALEJANDRA VARGAS ALARCÓN - (58002322W)

ÍNDICE

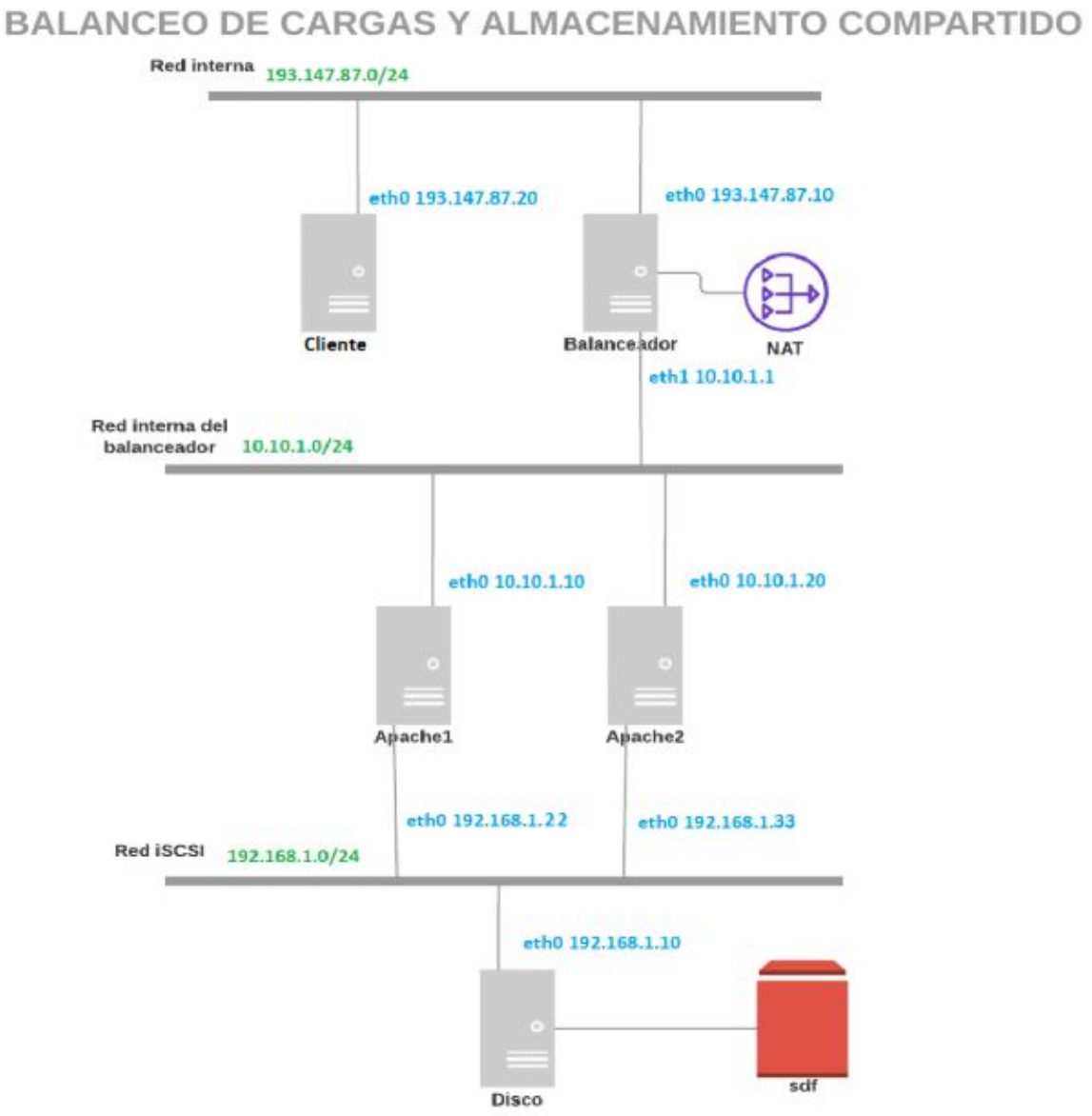
Introducción

➔ El objetivo de este proyecto es por un lado desplegar un cluster de balanceo de cargas y por otro que todos los servidores web del cluster ofrezcan el mismo contenido almacenado en un dispositivo compartido.

➔ Para el balanceo de cargas emplearemos dos módulos de apache: mod_proxy y mod_proxy_balancer. El primero nos aportará un servidor proxy inverso (reverse server) que básicamente se comporta como un intermediario entre una red externa (Internet) y una red privada (cliente), consiguiendo así controlar el tráfico de datos entrante, poner a disposición varios servidores bajo el mismo URL, distribuir solicitudes de manera simultánea por diferentes servidores, etc. Aunque todo esto proporciona muchas funciones, la que nos interesa en este caso es la de balanceo de cargas, que al combinarlo con el módulo mod_proxy_balancer genera un equilibrio de cargas entre los distintos servidores empleados.

➔ Por otro lado, para ofrecer el mismo contenido almacenado en un dispositivo compartido usaremos una SAN iSCSI, de esta tecnología hablaremos en el punto: Descripción de la herramientas y tecnología empleadas.

Escenario planteado



Este esquema de aquí arriba representa el escenario con el que vamos a trabajar. Contamos con tres redes: red interna (compartida por el cliente y el balanceador, este a su vez conectado a una red externa NAT), red interna del balanceador (en esta se encuentran el balanceador y dos servidores) y por último la red iSCSI (donde tenemos Disco que es el dispositivo compartido por los servidores y los servidores). Para simular estas redes emplearemos cinco máquinas virtuales, una por cada servidor.

La creación de estas máquinas la hemos llevado a cabo con ayuda de las imágenes proporcionadas en los apuntes de las prácticas 2 y 6, añadiendoles a estas algunas configuraciones a mayores que explicaremos más adelante.

Descripción de las herramientas y tecnologías empleadas

Balanceo de cargas

- **Apache2**: servidor web HTTP de código abierto para distintas plataformas (Unix, Windows, Macintosh...). Este puede ser configurado de distintas formas y puede montarse en cualquier equipo ya que es compatible con prácticamente todos los sistemas operativos.

- **Módulo de apache mod_proxy**: encargado de gestionar y redirigir las conexiones. Cabe aclarar que un proxy es un servidor que hace de intermediario entre un cliente y otro servidor. Por ejemplo, si un cliente quiere acceder a una web, primero el servidor proxy recibe esa petición y se la traslada al servidor que soporta esa URL, así este servidor no sabrá la identidad del cliente que realizó esa petición.

- **Módulo de apache mod_proxy_balancer**: proporciona un equilibrio de carga entre ambos servidores.

- **Módulos adicionales:** Los principales módulos empleados fueron los siguientes:
  - #a2enmod proxy.
  - #a2enmod proxy_balancer.

- **Alternativas:** podríamos usar Varnish o NGINX. Esta primera está orientada exclusivamente a HTTP, a diferencia de apache mod_proxy que también soporta FTP y SMTP. Es muy usada actualmente, por páginas web bien conocidas. NGINX es un servidor HTTP gratuito, a diferencia de los otros no depende de hilos para manejar las solicitudes, utiliza una arquitectura asíncrona (basada en eventos). Al igual que apache estas dos alternativas son software libre.

Dispositivo de almacenamiento compartido

- **SAN**: Red de almacenamiento conectada a las redes de comunicación de unos servidores. Puede tratarse tanto de una red de fibra o, lo que usaremos en este caso, SCSI (un equipo de interconexión dedicado y elementos de almacenamiento de red). La gran ventaja es que permite compartir datos entre varios equipos de la red sin afectar al rendimiento pues el tráfico de SAN está separado del de los usuarios. Si quisiéramos abaratar aún más el coste del almacenamiento y trabajar con recursos podríamos decantarnos por utilizar un almacenamiento de tipo NAS, que transfiere ficheros sobre la red de datos de la empresa (y no bloques como en una SAN) utilizando un protocolo llamado NFS.

- **iSCSI:** es una extensión de SCSI, que no es otra cosa que un protocolo para comunicación de dispositivos. En lo referido a almacenamiento, es una herramienta

diseñada para compartir el almacenamiento, sigue el modelo Cliente-Servidor. Así permite a un initiator (cliente) conectarse a un target (servidor).

- **Elementos de iSCSI:**

  - **Initiators**: gestionan la conexión con los targets y hacen posible la disposición de los dispositivos de bloques expuestos mediante iSCSI en los equipos cliente. Estos dispositivos de bloques tienen como identificador: LUN (número de la unidad lógica) o si está implementado en hardware HBA (adaptador de bus anfitrión).
    - Para modificar los initiators instalamos tgt (apt-get install tgt) y los utilizaremos con tgtadm

  - **Targets**: componentes responsable de hacer accesibles los dispositivos de bloques expuestos mediante iSCSI. Ya dicho anteriormente a cada dispositivo de bloques se le hace referencia mediante su identificador (único para cada uno), LUN y a cada target se cuenta a su vez con identificador único para cada uno, iqn (iqn.[fecha:yyyy-mm].[nombre del equipo invertido]:[identificador local del target]).
    - Para modificar los targets instalamos open-iscsi (apt-get install open-iscsi) y utilizaremos el comando iscsiadm

Común

En ambos casos hemos empleado ***VirtualBox 5.2***, que nos ha servido para simular y representar el escenario que conforman las máquinas de las figuras.

Desarrollo del trabajo

Pasos seguidos y configuraciones empleadas

Hemos empezado descargando el script de la práctica 6 y le hemos añadido la maquina discos. Para que funcionara correctamente hemos modificado el script, para ello le hemos añadido a las máquinas apache1 y apache2 una interfaz y unas IPs que pertenezcan a la red de la maquina discos.

Red iSCSI

Creamos el target (Servidor que expone los bloques), le asignamos un iqn y un identificador interno (tid=1):

root@discos:~# tgtadm --lld iscsi --mode target --op new --tid=1 --targetname iqn.2019-12.discos:compartido

Añadimos el LUN (dispositivo de bloque a exponer):

root@discos:~# tgtadm --lld iscsi --mode logicalunit --op new --tid 1 --lun 1 --backing-store /dev/sdf

root@discos:~# tgtadm --lld iscsi --mode target --op show

```
Target 1: iqn.2019-12.discos:compartido
  System information:
    Driver: iscsi
    State: ready
  I_T nexus information:
  LUN information:
    LUN: 0
      Type: controller
      ...
    LUN: 1
      Type: disk
      SCSI ID: IET     00010001
      SCSI SN: beaf11
      Size: 105 MB, Block size: 512
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: rdwr
      Backing store path: /dev/sdf
      Backing store flags:
  Account information:
  ACL information:
```

Restringimos el acceso al target de los servidores Apache1(192.168.100.22) y Apache2(192.168.10.33):

root@discos:~# tgtadm --lld iscsi --mode target --op bind --tid 1 --initiator-address 192.168.100.22

root@discos:~# tgtadm --lld iscsi --mode target --op bind --tid 1 --initiator-address 192.168.100.33

Definimos usuarios y contraseñas controlando el acceso mediante CHAP

root@discos:~# tgtadm --lld iscsi --mode account --op new --user usuario --password password

Ahora vinculamos el usuario con el target

root@discos:~# tgtadm --lld iscsi --mode account --op bind --tid 1 --user usuario

```
Target 1: iqn.2019-12.discos:compartido
  System information:
    Driver: iscsi
    State: ready
```

```
I_T nexus information:
LUN information:
  LUN: 0
       .....
  LUN: 1
    Type: disk
    SCSI ID: IET     00010001
    SCSI SN: beaf11
    Size: 105 MB, Block size: 512
    Online: Yes
    Removable media: No
    Prevent removal: No
    Readonly: No
    SWP: No
    Thin-provisioning: No
    Backing store type: rdwr
    Backing store path: /dev/sdf
    Backing store flags:
Account information:
  cda
ACL information:
  192.168.100.22
  192.168.100.33
```

Procedemos a vincular los **initiators**

root@apache1:~# echo "InitiatorName="`iscsi-iname` > /etc/iscsi/initiatorname.iscsi

root@apache2:~# echo "InitiatorName="`iscsi-iname` > /etc/iscsi/initiatorname.iscsi

Habilitamos la autenticación mediante CHAP y establecemos las credenciales de acceso en /etc/iscsi/iscsid.conf (en **ambos** initiators)

root@apachex:~# nano /etc/iscsi/iscsid.conf

```
# *************
# CHAP Settings
# *************

node.session.auth.authmethod = CHAP
node.session.auth.username = cda
node.session.auth.password = cdapass

# ********
# Timeouts
# ********
#
```

Reiniciar en ambos Initiators el daemon iscsi en ambos servidores

root@apache1:~# systemctl restart iscsid.service

root@apache2:~# systemctl restart iscsid.service

Comprobamos la lista iqn de targets y a su vez la actualizamos:

root@apache1:~# iscsiadm --mode discovery --type sendtargets --portal 192.168.100.11

192.168.100.11:3260,1 iqn.2019-12.discos:compartido

Conectamos el target seleccionado de la máquina discos:

root@apache1:~# iscsiadm --m node --targetname iqn.2019-12.discos:compartido --portal 192.168.100.11 --login

root@apache2:~# iscsiadm --m node --targetname iqn.2019-12.discos:compartido --portal 192.168.100.11 --login

root@discos:~# tgtadm --lld iscsi --mode target --op show

```
Target 1: iqn.2019-12..discos:pruebas
    System information:
        Driver: iscsi
        State: ready
    I_T nexus information:
        I_T nexus: 1
            Initiator: iqn.1993-08.org.debian:01:bd191abcfb83 alias: base
            Connection: 0
                IP Address: 192.168.100.33
        I_T nexus: 2
            Initiator: iqn.1993-08.org.debian:01:bd191abcfb83 alias: base
            Connection: 0
                IP Address: 192.168.100.22
    LUN information:
        LUN: 0
                ...
        LUN: 1
            Type: disk
            SCSI ID: IET     00010001
            SCSI SN: beaf11
            Size: 105 MB, Block size: 512
            Prevent removal: No
            Readonly: No
            SWP: No
            Thin-provisioning: No
            Backing store type: rdwr
            Backing store path: /dev/sdf
            Backing store flags:
    Account information:
        cda
    ACL information:
        192.168.100.22
        192.168.100.33
```

```
root@apache1:~# parted -l
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 17,2GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  17,2GB  17,2GB  primary  ext3         boot


Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system     Flags
 1      1049kB  1073MB  1072MB  primary  linux-swap(v1)


Model: IET VIRTUAL-DISK (scsi)
Disk /dev/sdc: 105MB
Sector size (logical/physical): 512B/512B
Partition Table: loop
Disk Flags:

Number  Start  End    Size   File system  Flags
 1      0,00B  105MB  105MB  ext3
```

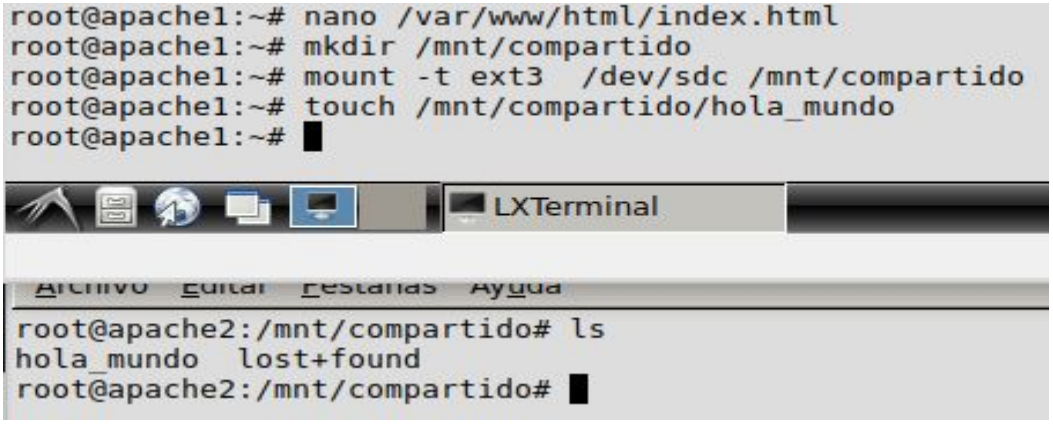
Creamos un nuevo punto de montaje en /mnt/compartido en **ambos** servidores Apache, lo formateamos a un .ext3 y lo montamos con /dev/sdc:

root@apache1:~# mkfs.ext3 /dev/sdc
root@apache1:~# mkdir /mnt/compartido
root@apache1:~# mount -t ext3 /dev/sdc /mnt/compartido

Comprobamos creando un archivo desde apache1 y accediendo a él desde apache2

root@apache1:~# touch /mnt/compartido/hola_mundo

root@apache2:~# cd /mnt/compartido
root@apache2:~#ls
hola_mundo lost+found



Red de balanceo de cargas

Una vez hemos comprobado que el dispositivo de almacenamiento compartido iSCSI funciona correctamente comenzaremos con la implementación del balanceo de cargas mod_proxy.

Hemos empezado modificando el index.html tanto de apache1 como de apache2 para poder diferenciarlos, de tal forma que hemos modificado el fichero:

root@apache1:~# nano /var/www/html/index.html

Resultado:

```
<html>
<body>
<h1>Pagina por defecto de apache</h1>

<p>Contenido estático, contenido estático</p>
<p>Contenido estático, contenido estático</p>

<h1>Servido por APACHE_UNO </h1>
</body></html>
```

En apache2 sustituimos la línea 9 por “<h1>Servido por APACHE_DOS<h1>

A continuación, hemos activado en el balanceador los módulos necesarios, con el comando a2enmod:

root@balanceador:~# a2enmod proxy

root@balanceador:~# a2enmod proxy_balancer

root@balanceador:~# a2enmod proxy_http (Proporciona las características utilizadas para la representación de las solicitudes HTTP y HTTPS)

root@balanceador:~# a2enmod xml2enc (Extendiende una ayuda internacionalizada de los módulos filtros basados en libxml12)

root@balanceador:~# a2enmod lbmethod_byrequests (proporciona el método de balanceo de carga byrequests)

Reiniciamos el servicio:

root@balanceador:~# service apache2 restart

Modificamos el archivo /etc/apache2/sites-enabled/000-default.conf, donde se guarda la configuración del sitio web almacenado, y modificamos lo siguiente, para activar el

root@balanceador:~# nano /etc/apache2/sites-enabled/000-default.conf

Resultado:

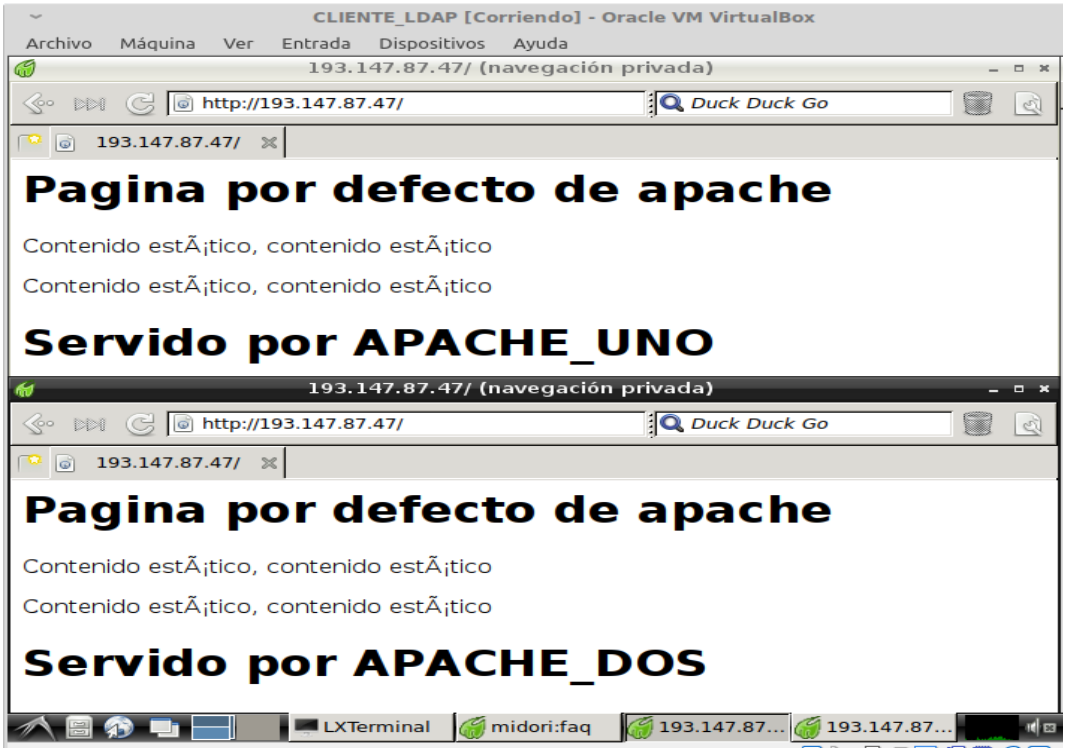
```
Fichero: /etc/apache2/sites-enabled/000-default.conf

        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf

        <Proxy *>
                Order deny,allow
        Allow from all
        </Proxy>

        ProxyPass / balancer://cluster
        <Proxy balancer://cluster>
                BalancerMember http://10.10.10.11
                BalancerMember http://10.10.10.22
        </Proxy>
</VirtualHost>
```

1. Le decimos por medio de la etiqueta, quiénes pueden ingresar, y quienes no.
2. El nombre de nuestro grupo o cluster balanceador y los nodos colaboradores involucrados.
3. La ruta de acceso a nuestro cluster balanceador, en este caso es la raíz, por tal motivo "/"
4. Para que las peticiones sean balanceadas, deberán solicitarse a http://localhost/.

Tras haber hecho estas configuraciones volvemos a reiniciar el servidor Apache para que los cambios surjan efecto:

root@balanceador:~# service apache2 restart

Procedemos a comprobar el trabajo realizado:
Nos conectamos desde la máquina cliente (IP: 193.147.87.33), abrimos dos buscadores web (con "xstart" iniciamos el entorno gráfico) y nos conectamos a la ip pública del clúster
(IP: 193.147.87.47) y reiniciamos en uno hasta comprobar que cambia de servidor (una de las ventanas es aconsejable que esté en modo incógnito o habrá que borrar las cookies.



Conclusiones

- **Problemas encontrados**

  A lo largo de la realización del proyecto nos hemos encontrado con numerables problemas que hemos conseguido ir solventando poco a poco. Entre ellos cabe destacar la configuración del entorno ya que al ser la tarea inicial no nos permitía continuar con el resto. Finalmente optamos por generar y configurar parte de las máquinas manualmente modificando los scripts de las prácticas.
  Otro de los mayores inconvenientes que hemos afrontado es encontrar una fuente de información útil para realizar la configuración del balanceador de carga, como por ejemplo, la configuración de la redirección dentro del clúster.

- **Conclusión final**

  En lo referente a encontrar posibles mejoras del sistema, podríamos incluir la utilización de más servidores en caso de que el número de usuarios fuese muy elevado puesto que nuestros dos servidores colapsarían llegado a un límite de sesiones de acceso.

Bibliografía

Intro

(1) https://www.ionos.es/digitalguide/servidores/know-how/que-es-un-servidor-proxy-inverso/

(2) https://documentation.help/httpd-2.4-es/mod_proxy_balancer.html

Escenario planteado

(3) https://ccia.esei.uvigo.es/docencia/CDA/1920/practicas/ejercicio-haproxy/

(4) https://ccia.esei.uvigo.es/docencia/CDA/1920/practicas/ejercicio-iscsi/

Herramientas y tecnologías

(5) https://www.virtualbox.org/wiki/Download_Old_Builds_5_2

(6) https://es.wikipedia.org/wiki/Servidor_proxy

(7) http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=820

(8) https://es.wikipedia.org/wiki/Varnish_Cache

(9) https://www.nginx.com/resources/wiki/

(10) https://es.ccm.net/contents/638-san-red-de-area-de-almacenamiento

(11) http://director-it.com/index.php/es/ssoluciones/data-center-cloud-virtualizacion/almacenamiento/119-diferencia-entre-san-y-nas.html

Desarrollo del trabajo

(12) https://ccia.esei.uvigo.es/docencia/CDA/1920/practicas/ejercicio-iscsi/

(13) https://ccia.esei.uvigo.es/docencia/CDA/1920/practicas/ejercicio-haproxy/

(14) https://marromang.wordpress.com/2018/02/26/servidor-proxy-cache-proxy-inverso-y-balanceador-de-carga/