INFRAESTRUCTURA LAMP DE ALTA DISPONIBILIDAD

1-Objetivo

El objeto de este proyecto es la creación y documentación de un cluster de alta disponibilidad que ofrezca un servicio web (**Wordpress**), montado sobre una infraestructura virtual basada en VirtualBox.
Para el mismo, se utilizarán la siguentes herramientas:

Sistema base:

Linux – Como sistema operativo de las máquinas que soportarán el portal
Apache – Como servidor Web encargado de servir nuestras páginas
Mysql – Será el sistema de base de datos a utilizar
PHP – El sistema que montaremos necesitará este intérprete para funcionar.

Herramientas de Alta disponibilidad:

Varnish - Servirá como sistema de balanceo entre los nodos Frontend y como sistema de cacheo.
DRBD - Es un sistema de almacenamiento en red ideal para clusters Mysql
GlusterFs – Otro sistema de almacenamiento en red, en este caso, muy óptimo para almacenar ficheros.
Heartbeat – Sistema que utilizaremos para vigilar la disponibilidad de los nodos y los servicios implicados.

1.1-Cluster de alta disponibilidad

Un cluster de alta disponibilidad es un conjunto de dos o más máquinas que se caracterizan por mantener una serie de servicios compartidos y por estar constantemente monitorizándose entre sí. Podemos dividirlo en dos clases:

Alta disponibilidad de infraestructura: Si se produce un fallo de hardware en alguna de las máquinas del cluster, el software de alta disponibilidad es capaz de arrancar automáticamente los servicios en cualquiera de las otras máquinas del cluster (**failover**). Y cuando la máquina que ha fallado se recupera, los servicios son nuevamente migrados a la máquina original (**failback**). Esta capacidad de recuperación automática de servicios nos **garantiza la alta disponibilidad** de los servicios ofrecidos por el cluster, minimizando así la percepción del fallo por parte de los usuarios.

Alta disponibilidad de aplicación: Si se produce un fallo del hardware o de las aplicaciones de alguna de las máquinas del cluster, el software de alta disponibilidad es capaz de **arrancar automáticamente los servicios que han fallado** en cualquiera de las otras máquinas del cluster. Y cuando la máquina que ha fallado se recupera, los servicios son nuevamente migrados a la máquina original. Esta capacidad de recuperación automática de servicios nos **garantiza la integridad** de la información, ya que no hay pérdida de datos, y además evita molestias a los usuarios, que no tienen por qué notar que se ha producido un problema.

1.1.0-LAMP

LAMP es el acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

Linux, el sistema operativo; En algunos casos también se refiere a LDAP.

Apache, el servidor web;

MySQL/MariaDB, el gestor de bases de datos;

Perl, PHP, o Python, los lenguajes de programación.

La combinación de estas tecnologías es usada principalmente para definir la infraestructura del servidor web.

1.1.1-Linux

Linux es un sistema operativo de núcleo **libre** (también suele referirse al núcleo como **kernel**) basado en Unix.4 Es uno de los principales ejemplos de **software libre y de código abierto**. Está licenciado bajo la GPL v2 y está desarrollado por colaboradores de todo el mundo. El desarrollo del día a día tiene lugar en la Linux Kernel Mailing List Archive
El núcleo Linux fue concebido por el entonces estudiante de ciencias de la computación finlandés **Linus Torvalds** en 1991. Linux consiguió rápidamente desarrolladores y usuarios que adoptaron códigos de otros proyectos de software libre para usarlo con el nuevo sistema operativo. El núcleo Linux ha recibido **contribuciones de miles de programadores de todo el mundo**. Normalmente Linux se utiliza junto a un empaquetado de software, llamado distribución GNU/Linux y servidores.

1.1.2-Apache

Apache es el servidor web por excelencia, su c**onfigurabilidad, robustez y estabilidad** hacen que cada vez millones de servidores reiteren su confianza en este programa.
Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, **Apache, es el servidor HTTP más usado**. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft3).

La mayoría de las **vulnerabilidades** de la seguridad descubiertas y resueltas tan **sólo pueden ser aprovechadas por usuarios locales** y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

1.1.3-MySql

MySQL es un **sistema de gestión de bases de datos relacional**, multihilo y multiusuario con más de seis millones de instalaciones. Desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productosprivativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

1.1.4-PHP

PHP es un **lenguaje de programación** de uso general de código del lado del servidor originalmente **diseñado para el desarrollo web de contenido dinámico**. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El **código es interpretado por un servidor web** con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. PHP puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

1.1.5-Varnish

Varnish Cache es un **acelerador de aplicaciones web**, también conocido como caché de proxy HTTP inversa. Se instala delante de cualquier servidor HTTP y se configura para almacenar en caché una copia del recurso solicitado. Ideado para **aumentar el rendimiento** de las aplicaciones web.

Una de las características clave, además de su rendimiento, es la flexibilidad de su configuración de lenguaje VCL. VCL permite determinar las políticas a tomar sobre las peticiones de entrada. En esta política se puede decidir qué contenido desea servir, desde donde se desea obtener el contenido y la forma en que la solicitud o la respuesta debe ser alterada.

Aunque en el planteamiento del proyecto se pretendía utilizar **HaProxy**, finalmente he decidido utilizar esta solución, ya que junto con **Heartbeat** hace las funciones de balanceo que necesitamos pero además es un excelente sistema de cacheo que se integra a la perfección con **Wordpress**, lo que acelerará la correcta visualización de nuestro portal.

1.1.6-DRBD

El Distributed Replicated Block Device o Dispositivo de Bloque Replicado Distribuido, es una **solución de almacenamiento replicada** basada en software que refeja el contenido de dispositivos de bloque (discos duros, particiones, volúmenes lógicos, etc) entre servidores.

Con sincronicidad o sin ella. Con reflejo sincróno, una operación es notificada de que ha sido completada por escrito, sólo después de que haya sido llevada a cabo en ambos sistemas del ordenador. El reflejo asíncrono implica que las operaciones serán notificadas por escrito cuando la escritura sea completada de modo local, pero antes de que la escritura se haya propagado al sistema peer.

En DRBD se puede **conseguir un cluster sobre casi todo lo que pueda replicar en el disco**. En nuestro caso específico, cuando queremos "clusterizar" sólo la base de datos, pero también podemos replicar cualquier otra cosa.

El DRBD es un módulo de kernel basado en RAID-1/TCP, muy sencillo de configurar y realmente rápido y sometido a pruebas de error.

Aunque en el planteamiento del proyecto se pretendía utilizar **NFS**, finalmente he decidido utilizar esta solución para el almacenamiento de la base de datos, ya que es un sistema específicamente preparado para replicación de datos entre distintas máquinas e integrado perfectamente con **Heartbeat.**

1.1.6-GlusterFS

El Sistema de Archivos Gluster, Gluster File System o GlusterFS, es un **sistema de archivos multiescalable para NAS** desarrollado inicialmente por Gluster Inc. Este permite agregar varios servidores de archivos sobre Ethernet o interconexines Infiniband RDMA en un entorno de archivos de red en paralelo. El diseño de GlusterFS **se basa en la utilización del espacio de usuario** y de esta manera **no compromete el rendimiento**. Se puede encontrar en una gran variedad de entornos y aplicaciones como computación en la nube, ciencias biomédicas y almacenamiento de archivos. El GlusterFS está licenciado bajo la licencia GNU General Public License versión 3.

Aunque en el planteamiento del proyecto se pretendía utilizar **NFS**, finalmente he decidido utilizar esta solución para el almacenamiento de los ficheros compartidos, ya que su fantástico rendimiento en este caso junto con Apache, mejora notablemente el funcionamiento de nuestro Wordpress

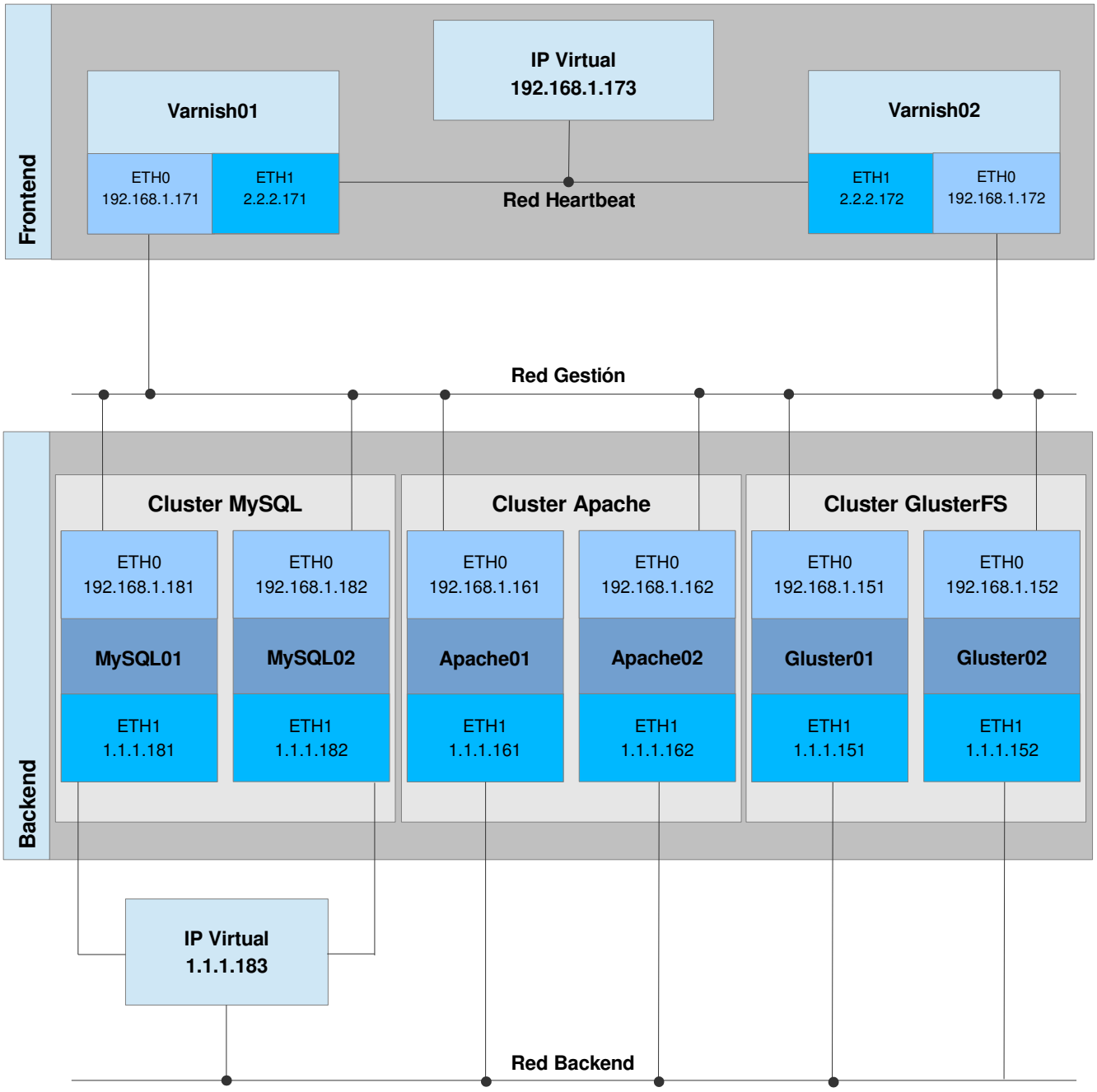
1.1.7-Heartbeat

Heartbeat es un **demonio que proporciona infraestructura de cluster a sus clientes**. Esto permite a los mismos conocer a tiempo real la presencia o ausencia de sus parejas de servidores y mensajearse de manera sencilla con ellos. **Se pueden monitorizar servicios** y, por tanto, en el caso de que la máquina “dueña” del servicio caiga, su compañera lo levantara en su lugar para evitar una perdida del mismo.

Utilizaremos este demonio para vigilar en nuestro entorno los procesos de **Varnish y Mysql**.

2.0-Esquema

El esquema de la instalación es el siguiente:



2.1-Máquinas Linux e infraestructura de red

Para comenzar, partimos de una imágen Debian minima como maqueta de todas las máquinas.
Dicha imágen consta de **128 Mb RAM** y un **disco duro de 8Gb**. En algunos casos ha sido necesario añadir un disco más (en las máquinas de GlusterFs y de Mysql) y en la mayoría de ellos una tarjeta de red adicional para manejar las redes creadas por los demonios de heartbeat y las tareas de gestión internas de las máquinas.

Los motivos de esta simple configuración de hardware son diversos, pero los principales son:

- **No se necesita mas potencia** para nuestra práctica de laboratorio.
- La idea original es **arrancar todas las máquinas simultaneamente** en un equipo con 4 Gb de Ram

La solución funciona perfectamente gracias a la potencia del sistema de cacheo (Varnish) que evita la mayor parte de las peticiones a los backends.

Se ha utilizado VirtualBox como solución de virtualización por su sencillez y gran potencia.

La parte de red también es sencilla. Se a utilizado el rango **192.168.1.0** como red de gestión y de acceso a las máquinas y por otro lado se han desplegado una serie de redes internas que son utilizadas por los distintos demonios que corren en las máquinas.

1.1.1.0 – Utilizada por todas las máquinas del backend, por los demonios Heartbeat, DRBD y GlusterFS.
2.2.2.0 – Utilizada por el heartbeat en el Frontend de las máquinas de Varnish.

2.2-Configuración del cluster de Almacenamiento

Como mencionábamos anteriormente, utilizaremos la solución GlusterFS como sistema de almacenamiento para alojar los ficheros de nuestra aplicación.
GlusterFS agrupa los distintos dispositivos de almacenamiento de la red **como si fueran un único bloque**. Es flexible, modular, escalable y además muy sencillo de configurar.

Esta solución nos brinda dos maneras principales de manejar los datos:

- Datos distribuidos
- Datos replicados

Para nuestro proyecto, utilizaremos el método de **datos replicados**, ya que sólo tenemos dos máquinas y queremos alta disponibilidad.

Para acceder a los datos, podemos también utilizar distintos métodos:

- **Gluster Native Client** (Recomendado) - Permite alta concurrencia, rendimiento y un gran control de failover. Es necesario instalar el cliente en las máquinas que necesitan acceder a los datos.
- **NFS** – Utilizando NFS v3 es posible acceder a los volúmenes de Gluster. Está comprobado que la mayoría de lso sistemas operativos funcionan correctamente utilizando este sistema.
- **CIFS** – También es posible utilizar este método y facilitar el acceso por Samba.

En nuestro caso, y por ser el más recomendado por la comunidad, utilizaremos el **Gluster Native Client.**

2.2.1-Configuración del cluster de Almacenamiento – Tareas previas

Para poder trabajar con los hostnames, se añaden las siguientes entradas al fichero /etc/hosts de ambos nodos de Gluster

```
127.0.0.1    localhost
127.0.1.1    gluster02
1.1.1.151    gluster01
1.1.1.152    gluster02
```

También se utiliza una segunda tarjeta para crear la red por la que se conectaran nuestros clientes, para ello se añade dicha tarjeta desde VirtualBox y se configura de la siguiente forma en cada máquina:

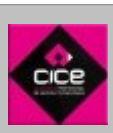
/etc/network/interfaces - gluster01

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.1.151
netmask 255.255.255.0
gateway 192.168.1.1

auto eth1
iface eth1 inet static
address 1.1.1.151
netmask 255.255.255.0
```

/etc/network/interfaces - gluster02

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.1.152
netmask 255.255.255.0
gateway 192.168.1.1

auto eth1
iface eth1 inet static
address 1.1.1.152
netmask 255.255.255.0
```

Después se reinicia la máquina (o los servicios de red) para que se apliquen los cambios.



Posteriormente, se crea el disco duro que utilizaremos como “recurso compartido”. Para ello se añade a ambas máquinas desde VirtualBox y lo se configura de la siguiente manera.

En primer lugar, se crea la partición:

```
root@gluster01:~# fdisk /dev/sdb
El dispositivo no contiene una tabla de particiones DOS válida ni una etiqueta de disco Sun o SGI
Building a new DOS disklabel with disk identifier 0xe2868d78.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Atención: el indicador 0x0000 inválido de la tabla de particiones 4 se corregirá mediante w(rite)

Orden (m para obtener ayuda): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Número de partición (1-4, valor predeterminado 1): 1
Primer sector (2048-16777215, valor predeterminado 2048):
Se está utilizando el valor predeterminado 2048
Last sector, +sectores or +size{K,M,G} (2048-16777215, valor predeterminado 16777215):
Se está utilizando el valor predeterminado 16777215

Orden (m para obtener ayuda): w
¡Se ha modificado la tabla de particiones!

Llamando a ioctl() para volver a leer la tabla de particiones.
Se están sincronizando los discos.
```

Posteriormente, se formatea la nueva partición creada (utilizaremos ext3, que es compatible con Gluster):

```
root@gluster01:~# mkfs.ext3 /dev/sdb1
mke2fs 1.42.5 (29-Jul-2012)
Etiqueta del sistema de ficheros=
OS type: Linux
Tamaño del bloque=4096 (bitácora=2)
Tamaño del fragmento=4096 (bitácora=2)
Stride=0 blocks, Stripe width=0 blocks
524288 inodes, 2096896 blocks
104844 blocks (5.00%) reserved for the super user
Primer bloque de datos=0
Número máximo de bloques del sistema de ficheros=2147483648
64 bloque de grupos
32768 bloques por grupo, 32768 fragmentos por grupo
8192 nodos-i por grupo
Respaldo del superbloque guardado en los bloques:
       32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: hecho
Escribiendo las tablas de nodos-i: hecho
Creating journal (32768 blocks): hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho
```

*Es necesario realizar los mismos pasos en ambos nodos.

2.2.2-Configuración del cluster de Almacenamiento – Instalación del servidor GlusterFS

En primer lugar, es necesario instalar el servidor en ambos nodos:

```
root@gluster01:apt-get install glusterfs-server
```

```
root@gluster02:apt-get install glusterfs-server
```

Tras la instalación se comprueba que gluster se ha iniciado y que está corriendo.

```
root@gluster01:~# service glusterfs-server status
[ ok ] glusterd service is running with pid 1896.
```

Se crea un pool de confianza entre ambos servidores. Para ello se ejecuta lo siguiente.

```
root@gluster01:~# gluster peer probe gluster02
Probe successful
```

Creamos el directorio donde montaremos el disco con los datos a replicar en ambos nodos.

```
root@gluster01:~# mkdir -p /var/export
root@gluster01:~# ssh root@gluster02 -C "mkdir -p /var/export"
```

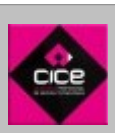
Se ejecuta la siguiente sentencia para crear el volumen que llamaremos "storage"

```
root@gluster01:~# gluster volume create repl-vol storage 2 \
gluster01:/var/export gluster02:/var/export
Creation of volume storage has been successful. Please start the volume to access data.
```

Arrancamos el volumen creado y comprobamos que está correcto:

```
root@gluster01:~# gluster volume start storage
root@gluster01:~# Starting volume storage has been successful
```

```
root@gluster01:~# gluster volume info storage

Volume Name: storage
Type: Replicate
Status: Started
Number of Bricks: 2
Transport-type: tcp
Bricks:
Brick1: gluster01:/var/export
Brick2: gluster02:/var/export
Options Reconfigured:
network.ping-timeout: 5
```

2.3 – Configuración del cluster de base de datos

Ahora explicaré como montar el cluster de base de datos. Las aplicaciones necesarias son las siguientes

- **Mysql Server** – El servidor de base de datos que contendrá las tablas de nuestro Wordpress
- **DRBD** – El sistema que utilizaremos apara compartir nuestro almacenamiento en red.
- **Heartbeat** – Con este demonio daremos alta disponibilidad al servicio.

En primer lugar, será necesario añadir una disco en ambos nodos para almacenar la base de datos. Dicho disco estará montado en red utilizando DRBD con una **IP Virtual al estilo NAS**. Será un cluster **activo** / **pasivo** en el que el nodo "maestro" tendrá los datos en su posesión durante todo el tiempo. En el caso de que **Heartbeat** detecte que que este nodo deja de responder, se apropiará del cluster declarándose a sí mismo como maestro, levantando la ip virtual en lugar del antiguo nodo y montando la unidad compartida en su filesystem.

Para ello necesitamos que nuestras dos máquinas posean una tarjeta extra de red y una nueva unidad de disco cada una.

2.3.1 – Configuración del cluster de base de datos – Tareas previas

Al igual que hicimos anteriormente, para poder trabajar con los hostnames, se añaden las siguientes entradas al fichero /etc/hosts de ambos nodos de Gluster

```
127.0.0.1    localhost
127.0.1.1    mysql01
1.1.1.181    mysql01
1.1.1.182    mysql02
```

También se utiliza una segunda tarjeta para crear la red por la que se funcionará heartbeat, para ello se añade dicha tarjeta desde VirtualBox y se configura de la siguiente forma en cada máquina:

/etc/network/interfaces - mysql01

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.1.181
netmask 255.255.255.0
gateway 192.168.1.1

auto eth1
iface eth1 inet static
address 1.1.1.181
netmask 255.255.255.0
```

/etc/network/interfaces - mysql02

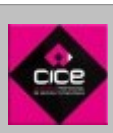
```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.1.182
netmask 255.255.255.0
gateway 192.168.1.1

auto eth1
iface eth1 inet static
address 1.1.1.182
netmask 255.255.255.0
```

Posteriormente se reinicia la máquina (o los servicios de red) para que se apliquen los cambios.

Se crea el disco duro que utilizaremos para almacenar la base de datos. Para ello se añade a ambas máquinas desde VirtualBox y lo se configura de la siguiente manera.

En primer lugar, se crea la partición:

```
root@mysql01:~# fdisk /dev/sdb
El dispositivo no contiene una tabla de particiones DOS válida ni una etiqueta de disco Sun o SGI
Building a new DOS disklabel with disk identifier 0xe2868d78.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Atención: el indicador 0x0000 inválido de la tabla de particiones 4 se corregirá mediante w(rite)

Orden (m para obtener ayuda): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Número de partición (1-4, valor predeterminado 1): 1
Primer sector (2048-16777215, valor predeterminado 2048):
Se está utilizando el valor predeterminado 2048
Last sector, +sectores or +size{K,M,G} (2048-16777215, valor predeterminado 16777215):
Se está utilizando el valor predeterminado 16777215

Orden (m para obtener ayuda): w
¡Se ha modificado la tabla de particiones!

Llamando a ioctl() para volver a leer la tabla de particiones.
Se están sincronizando los discos.
```

Posteriormente, se formatea la nueva partición creada (utilizaremos ext3, que es compatible con Gluster):

```
root@mysql01:~# mkfs.ext3 /dev/sdb1
mke2fs 1.42.5 (29-Jul-2012)
Etiqueta del sistema de ficheros=
OS type: Linux
Tamaño del bloque=4096 (bitácora=2)
Tamaño del fragmento=4096 (bitácora=2)
Stride=0 blocks, Stripe width=0 blocks
524288 inodes, 2096896 blocks
104844 blocks (5.00%) reserved for the super user
Primer bloque de datos=0
Número máximo de bloques del sistema de ficheros=2147483648
64 bloque de grupos
32768 bloques por grupo, 32768 fragmentos por grupo
8192 nodos-i por grupo
Respaldo del superbloque guardado en los bloques:
       32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: hecho
Escribiendo las tablas de nodos-i: hecho
Creating journal (32768 blocks): hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho
```

*Es necesario realizar los mismos pasos en ambos nodos.

2.3.2 – Configuración del cluster de base de datos – Instalación de componentes

En primer lugar se instala el servidor de MySQL. Para ello, y de la manera habitual en entornos Debian, utilizamos de nuevo el comando apt-get. Se ejecuta en ambos nodos.

```
root@mysql01:~# apt-get install mysql-server
```

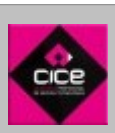
Durante la instalación se solicita una password para el usuario root. En nuestro caso **cice2013**
Una vez terminada la instalación, instalamos el software drbd y heartbeat en ambos nodos.

```
root@mysql01:apt-get install drbd8-utils drbdlinks heartbeat
```

Los paquetes que instalamos son los siguientes:

- drbd8-utils – El “core” de DRBD
- drbdlinks – Plugin extra que permite manejar links simbólicos
- heartbeat – demonio heartbeat

Una vez instalados, se procede en primer lugar con la configuración de DRBD. Dicha configuración comienza con la parametrización del fichero /etc/drbd.conf .En el caso de Debian, dicho fichero hace un include a otro llamado global_common.conf ubicado en /etc/drbd.d que será el que nosotros modifiquemos.



2.3.2.1 – Configuración del cluster de base de datos – DRBD

/etc/drbd.d/global_common.conf

```
global {
    usage-count no;
}
common {
  protocol C;
  syncer {
        rate 10M;
        al-extents 257;
 }
}
resource db {
  startup {
    become-primary-on mysql01;
  }
  disk {
    on-io-error   detach;
  }
  net {
    cram-hmac-alg sha1;
    shared-secret "cice13";
  }
  on mysql01 {
    device    /dev/drbd1;
    disk      /dev/sdb1;
    address   1.1.1.181:7788;
    meta-disk internal;
  }
  on mysql02 {
    device    /dev/drbd1;
    disk      /dev/sdb1;
    address   1.1.1.182:7788;
    meta-disk internal;
  }
}
```

En este fichero ubicado en ambas máquinas se definen los nodos del cluster y la manera en la que funcionará el mismo, así como su nombre, etc...

Bloque Global – Opciones Globales

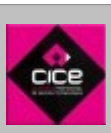
- usage-count – Cuenta el numero de descargas.

Bloque Common

- Protocol C – Hace las operaciones de manera síncrona
- rate – Modo de transmisión (velocidad)
- al-extents – (activity log extents) Define la el tamaño del area donde se escriben los metadatos.

Bloque resource (db – Nombre de cluster)

- become-primary – Nodo principal por defecto
- on-io-error – Acción a tomar en caso de error de escrituro o lectura
- cram-hmac-alg – Algoritmo de encriptación
- shared-secret – Clave compartida
- device - Nombre del disco virtual
- disk - Nombre del disco físico
- address – Ip del nodo
- meta-disk – Tipo de disco de metadatos que se utilizará.



Una vez parametrizado dicho fichero en ambos nodos, se crea el recurso compartido. Está acción se realiza mediante el comando drbdadm de la siguiente manera:

```
root@mysql01:# drbdadm create-md db
Writing meta data...
initializing activity log
NOT initialized bitmap
New drbd meta data block successfully created.

            --== Creating metadata ==--
As with nodes, we count the total number of devices mirrored by DRBD at
at http://usage.drbd.org.

The counter works anonymously. It creates a random number to identify
the device and sends that random number, along with
DRBDs version number, to usage.drbd.org.

http://usage.drbd.org/cgi-bin/insert_usage.pl?nu=14515620191698488957&amp;ru=118619300892

* If you wish to opt out entirely, simply enter 'no'.
* To continue, just press [RETURN]

success
```

En este punto es necesario iniciar el servicio en ambos nodos. Lo haremos de la siguiente manera.

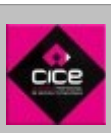
```
root@mysql01:#service drbd start
Starting DRBD resources: [
db
Found valid meta data in the expected location, 2146758656 bytes into /dev/sdb1.
d(db) s(db) n(db) ]..........
```

Se convierte el nodo primario en principal y se sincroniza.

```
root@mysql01:# drbdadm -- --overwrite-data-of-peer primary db
```

Podemos ver el estado de la sincronización haciendo un cat al fichero /proc/drbd

```
root@mysql01:# cat /proc/drbd
version: 8.3.8 (api:88/proto:86-94)
GIT-hash: d78846e52224fd00562f7c225bcc25b2d422321d build by mockbuild@builder10.centos.org

 1: cs:c ro:Secondary/Primary ds:Inconsistent/UpToDate C r----
    ns:0 nr:189440 dw:189440 dr:0 al:0 bm:11 lo:0 pe:0 ua:0 ap:0 ep:1 wo:b oos:1906908
       [###.................] synced:  9.2% (1906908/7656700)K queue_delay: 0.1 ms
       finish: 0:04:38 speed: 6,820 (6,532) want: 6,144 K/sec
```

El siguiente paso es darle formato a la partición de DRBD y montarla en el primer nodo.

```
root@mysql01:# mkfs.ext3 /dev/drbd1
mke2fs 1.39 (29-May-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
262144 inodes, 524087 blocks
26204 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=8255636
16 block groups
32768 blocks per group, 32768 fragments per group
16384 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912

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

This filesystem will be automatically checked every 39 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

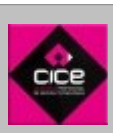
Se crea el directorio en el primer nodo donde ubicar el recurso y se monta en el mismo.

```
root@mysql01:# mkdir /db
root@mysql01:# mount /dev/drbd1 /db
```

En el nodo secundario esta vez, se formatea el recurso y se crea el directorio de montaje.

```
root@mysql02:# mkfs.ext3 /dev/drbd1
mke2fs 1.39 (29-May-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
262144 inodes, 524087 blocks
26204 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=8255636
16 block groups
32768 blocks per group, 32768 fragments per group
16384 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912

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

This filesystem will be automatically checked every 39 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

```
root@mysql02:# mkdir /db
```

2.3.2.2 – Configuración del cluster de base de datos – Heartbeat

- La configuración de heartbeat consta de 3 ficheros principales, estos son:
- **/etc/heartbeat/ha.cf** – Donde se define el comprortamiento y los nodos del cluster
- **/etc/heartbeat/haresources** – Donde se define la red, la ip virtual y los servicios a vigilar
- **/etc/heartbeat/authkeys** – Donde se define el tipo de enctiptación y la clave compartida

/etc/heartbeat/ha.cf

```
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility local0
keepalive 2
deadtime 30
warntime 10
initdead 120
udpport 694 # If you have multiple HA setup in same network.. use different ports
bcast eth1 # Linux
auto_failback on # This will failback to machine1 after it comes back
apiauth ipfail gid=haclient uid=hacluster
node mysql01
node mysql02
```

Aqui definimos los siguientes parámetros:

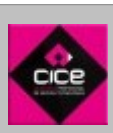
- **debugfile** – Ubicación del fichero de Debug
- **logfile** – Ubicación del fichero de log
- **logfacility** – Tipo de log a escribir
- **keepalive** – Especifica cada cuanto Heartbeat enviará paquetes para comprobar la disponibilidad
- **deadtime** – Tiempo en el que Hearbeat confirmará que un nodo ha caído.
- **warntime** – Tiempo que esperará Heartbeat para avisar cuando un nodo falle.
- **initdead** – Tiempo máximo que Heartbeat esperará a que un nodo arranque.
- **udpport** – Puerto que utilizarán ambos nodos para comunicarse.
- **bcast** – Dispositivo de red a utilizar y modo broadcast
- **auto_failback on** – Heartbeat tratará de recuperar el servicio si lo detecta caído
- **node** – Nombre de cada nodo implicado en el cluster.

/etc/heartbeat/haresources

```
mysql01 IPaddr::1.1.1.183/24/eth1 drbddisk::db Filesystem::/dev/drbd1::/db::ext4 mysql
```

Este fichero también debe ser igual en ambos nodos y se definen los siguientes parámetros:

- Nodo principal
- Dirección ip Virtual que se aplicará al cluster
- Nombre del disco drbd
- Disco Virtual
- Directorio Físico
- Servicio a vigilar



/etc/heartbeat/authkeys

```
auth 1
1 sha1 cice13
```

En este último, se declara el tipo de autenticación, la encriptación y la clave compartida. Debe tener permisos 600 para poder funcionar bien.

A continuación, se copian los fichero necesarios para poder tener la base de datos en el nuevo recurso y que ambos nodos puedan manejarla. Creamos un enlace símbolico a la nueva ubicación para “engañar al servidor” como si estos estuvieran localmente. Dichos ficheros son todos los que se encuentran ubicados en el directorio /var/lib/mysql

```
root@mysql01:# cp  -pr /var/lib/mysql /db/
root@mysql01:# ln -s /db/mysql /var/lib/mysql
```

Por último, modificamos el fichero de configuración de mysql (my.cf) para indicarle el nuevo directorio de datos

/etc/mysql/my.cf

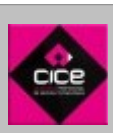
```
[mysqld]
datadir=/db/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Default to using old password format for compatibility with mysql 3.x
# clients (those using the mysqlclient10 compatibility package).

...
```

También, en el mismo fichero, configuramos el siguiente parámetro para permitir el acceso a los datos desde las máquinas del exterior.

```
bind-address            = 0.0.0.0
```

Se quita el demonio del inicio para permitir que heartbeat lo controle y se reinicia MySQL

```
root@mysql01:# update-rc.d mysql disable
root@mysql01:# service mysql restart
```

2.4 – Configuración de Apache

En este punto se montan las dos máquinas que proporcionarán el servicio Web. Debemos tener en cuenta que estás máquinas tendrán acceso a los servicios creados anteriormente por lo que necesitaremos lo siguiente:

- **Cliente GlusterFS** – Para acceder a los ficheros compartidos en red.
- **Acceso a la ip Virtual del cluster de Mysql** – Para poder trabajar con la base de datos.

2.4.1 – Configuración de Apache – Tareas previas

Como "extras de hardware" en esta máquina únicamente necesitaremos una tarjeta de red adicional para poder acceder al resto de los recursos. Lo haremos como en las ocasiones anteriores, configurando el fichero "interfaces" de la siguiente manera. Así pues, esta es la configuración correcta para el nodo 1 de apache.

/etc/network/interfaces - apache01

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.1.161
netmask 255.255.255.0
gateway 192.168.1.1

auto eth1
iface eth1 inet static
address 1.1.1.161
netmask 255.255.255.0
```

Y está debería ser la configuración del nodo 2.

/etc/network/interfaces - apache02

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.1.162
netmask 255.255.255.0
gateway 192.168.1.1

auto eth1
iface eth1 inet static
address 1.1.1.162
netmask 255.255.255.0
```

Configuramos el fichero /etc/hosts en ambos nodos.

/etc/hosts - apache01

```
127.0.0.1     localhost
127.0.1.1     apache1.localhost.localdomain   apache01
1.1.1.151     gluster01
1.1.1.152     gluster02
1.1.1.183     apache
```

/etc/hosts - apache02

```
127.0.0.1     localhost
127.0.1.1     apache2.localhost.localdomain   apache02
1.1.1.151     gluster01
1.1.1.152     gluster02
1.1.1.183     apache
```

2.4.2 – Configuración de Apache – Instalación de componentes

En primer lugar se instala el cliente de glusterfs en ambos nodos de la siguiente manera.

```
root@apache01:# apt-get install glusterfs-client
```

Se monta, con el siguiente comando, el recurso compartido creado anteriormente.

```
root@apache01:# mount.glusterfs gluster01:storage /var/www/
```

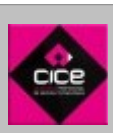
Se revisa el fichero /etc/fstab para comprobar que la unidad se montará al iniciar el sistema.

/etc/fstab

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=d3c22d35-ad51-49ff-a9a3-c160815d9fb3 /           ext4    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=d4eed404-55ba-45b2-a639-8c0200c30f95 none        swap    sw              0        0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto  0       0
gluster01:/storage/var/www/ glusterfsdefaults,_netdev,log-level=WARNING,log-file=/var/log/gluster.log 0 0
```

Ahora que ya tenemos acceso al almacenamiento, instalamos Apache con sus módulos necesarios para poder interpretar el lenguaje PHP y poder acceder a bases de datos mysql.

```
root@apache01:# apt-get install apache2 php5 php5-mysql
```

Una vez instalados los paquetes, se edita el fichero /etc/apache2/sites-enabled/000-default, el virtualhost que contiene la configuración de nuestra web.

/etc/apache2/sites-enabled/000-default

```
<VirtualHost *:80>
   DocumentRoot "/var/www/html"
   ServerName proyecto_cice.com
   ServerAlias www.proyecto_cice.com
   Options -Indexes
</VirtualHost>
```

Tambien es conveniente añadir al final del fichero de configuración un par de reglas básicas de seguridad.

/etc/apache2/apache2.conf

```
ServerSignature Off
ServerTokens Prod
```

Se comprueba desde ambos nodos la conectividad con la ip virtual de MySQL.

```
root@apache01:~# ping 1.1.1.183
PING 1.1.1.183 (1.1.1.183) 56(84) bytes of data.
64 bytes from 1.1.1.183: icmp_req=1 ttl=64 time=2.97 ms
64 bytes from 1.1.1.183: icmp_req=2 ttl=64 time=0.379 ms
64 bytes from 1.1.1.183: icmp_req=3 ttl=64 time=0.466 ms
...
```

Y se reinicia el servicio de apache.

```
root@apache01:~# service apache2 restart
```

2.5 – Configuración de Varnish

En este punto vamos a proceder a instalar los servicios en las máquinas frontEnd. Para funcionar, varnish necesita únicamente el fichero /etc/varnish/default.vcl que contiene la configuración de funcionamiento del cacheo así como los distintos nodos de apache que componen el cluster. Se pueden crear millones de reglas para filtrar el contenido HTTP y cachearlo ya que el lenguaje VCL en el que está basado este fichero es muy rico para ello.

Por otro lado utilizaremos de nuevo heartbeat para monitorizar los dos nodos de varnish.

2.5.1 – Configuración de Varnish – Tareas previas

/etc/hosts varnish01

```
127.0.0.1   localhost
127.0.1.1   varnish01.localhost.localdomain  varnish01

2.2.2.171 varnish01
2.2.2.172 varnish02

192.168.1.161 apache01
192.168.1.162 apache02
```

/etc/hosts varnish02

```
127.0.0.1   localhost
127.0.1.1   varnish02.localhost.localdomain  varnish02

2.2.2.171 varnish01
2.2.2.172 varnish02

192.168.1.161 apache01
192.168.1.162 apache02
```

2.5.2 – Configuración de Varnish – Configuración del demonio Varnish

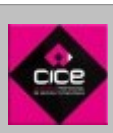
En primer lugar instalamos el demonio Varnish en ambos nodos.

```
root@varnish01:~# apt-get install varnish
```

Modificamos el fichero de configuración. En nuestro caso, existe una optimización para Wordpress que he descargado de la Web oficial de Varnish que quizás no es relevante en este momento ya que varía dependiendo de la aplicación instalada ya que cambian los objetos a cachear. Por ello en este documento, únicamente incluyo la configuración básica de varnish que es la que en realidad hace que funcione, excluyendo de esta manera la parte mas irrelevante

/etc/varnish/default.vcl varnish01 y varnish02

```
backend apache01 {
 .host = "192.168.1.161";
 .port = "80";
}
backend apache02 {
 .host = "192.168.1.162";
 .port = "80";
}
director balanceo round-robin {
 {
 .backend = apache01;
 }
 {
 .backend = apache02;
 }
}
sub vcl_recv {
 set req.backend = balanceo;
}
```

Por último, reiniciamos el servicio en ambos nodos. Cada vez que se reinicia el mismo, se vacía la caché de Varnish.

```
root@varnish01:~# service varnish restart
```

2.5.2 – Configuración de Varnish – Configuración de Heartbeat

Como hicimos en el caso anterior, instalamos y configuramos los tres ficheros de Heartbeat.

- **/etc/heartbeat/ha.cf**
- **/etc/heartbeat/haresources**
- **/etc/heartbeat/authkeys**

/etc/heartbeat/ha.cf

```
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility local0
keepalive 2
deadtime 30
warntime 10
initdead 120
udpport 694 # If you have multiple HA setup in same network.. use different ports
bcast eth1 # Linux
auto_failback on # This will failback to machine1 after it comes back
apiauth ipfail gid=haclient uid=hacluster
node varnish01
node varnish02
```

/etc/heartbeat/haresources

```
varnish01 192.168.1.173/24 varnish
```

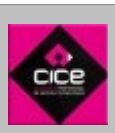
/etc/heartbeat/authkeys

```
auth 1
1 crc
```

Recordad que éste último fichero ebe tener permisos 600 para poder funcionar bien.

Reiniciamos el servicio en ambos nodos.

```
root@heartbeat01:~# service heartbeat restart
```

3.0 – Arranque de las máquinas y acceso a la aplicación.

En este punto ya tenemos nuestra infraestructura de HA creada. Para comprobar que funciona correctamente, he instalado una instancia de Wordpress en la misma.

Wordpress es un CMS que requiere PHP y base de datos mysql para funcionar. Como único punto destacable en este documento en su instalación, tuve que crear una base de datos y un usuario para la misma en nuestro sistema de base de datos. Para ello utilicé el siguiente script.

```
CREATE DATABASE cice;
CREATE USER 'cice'@'1.1.1.161' IDENTIFIED BY 'cice2013';
CREATE USER 'cice'@'1.1.1.162' IDENTIFIED BY 'cice2013';
CREATE USER 'cice'@'apache01' IDENTIFIED BY 'cice2013';
CREATE USER 'cice'@'apache02' IDENTIFIED BY 'cice2013';
GRANT ALL PRIVILEGES ON cice.* TO 'cice'@'1.1.1.161';
GRANT ALL PRIVILEGES ON cice.* TO 'cice'@'1.1.1.162';
GRANT ALL PRIVILEGES ON cice.* TO 'cice'@'apache01';
GRANT ALL PRIVILEGES ON cice.* TO 'cice'@'apache02';
FLUSH PRIVILEGES;
```

Se instaló la aplicación según el manual en http://www.wordpress.com

Orden de arranque.

Para arrancar el entorno de manera correcta, debe de hacerse de forma ordenada para que no haya problemas a la hora de arrancar los servicios por lo que el orden debe ser el siguiente. Cuando un pool de máquinas esté arrancado, procederemos a arrancar el siguiente.

1. Cluster GlusterFS
2. Cluster Mysql
3. Cluster Apache
4. Cluster Varnish

Acceso a la aplicación.

El acceso a la aplicación se realiza mediente la ip virtual de nuestro balanceador Varnish **192.168.1.173**.

Credenciales.

En todos los casos la contraseña de root de todos los demás servicios es: **cice2013**

El usuario de las máquinas y de base de datos: **cice**

Se ha securizado ssh para que no se pueda acceder directamente como root, teniendo que entrar primeramento con el usuario cice y despúes saltar a root.

4.0 – Referencias.

http://es.wikipedia.org
http://www.linux-es.org/
http://httpd.apache.org/
http://www.mysql.com/
http://www.gluster.org/
http://www.drbd.org/
http://www.linux-ha.org
https://www.varnish-cache.org/
http://blog.exceliance.fr/2012/08/25/haproxy-varnish-and-the-single-hostname-website/
http://how-to.linuxcareer.com/configuration-of-high-availability-storage-server-using-glusterfs
http://keithscode.com/blog/23-running-mysql-on-a-small-128mb-vps.html
http://blog.irwan.name/?p=118
http://www.alcancelibre.org/staticpages/index.php/como-cluster-heartbeat-centos
http://www.cyberciti.biz/tips/linux-unix-bsd-openssh-server-best-practices.html