Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Proyecto de Servidor WEB en clúster con alta disponibilidad y distribución de carga. Herramienta de virtualización KVM

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Sergio García Lanza

Tutor: Floreal Acebrón Linuesa

2017-2018

Resumen

El presente TFG ha consistido en la configuración de un clúster para tener la alta disponibilidad de un servidor web, todo ello con software libre. Para ello se ha utilizado la herramienta de virtualización KVM de Linux y el sistema operativo Ubuntu Server. El funcionamiento principal consiste en recibir peticiones web por parte de los usuarios, estas peticiones son administradas por los dos nodos principales que reparten la carga a los servidores web. Estos dos nodos principales están en continua comunicación para saber el estado de cada uno de ellos en todo momento y así conseguir la alta disponibilidad.

Palabras clave: Clúster, Servidor Web, KVM, Linux, Ubuntu Server, Corosync, Pacemaker, LVS.

Abstract

The present TFG has consisted in the configuration of a cluster to have the first availability of a web server, all with free software. For this we have used the Linux KVM virtualization tool and the Ubuntu Server operating system. The main operation is to receive requests from users, these requests are administered by the two nodes. These two main nodes are in continuous communication to know the status of each of them at all times and thus obtain maximum availability.

Keywords: Cluster, Web Server, KVM, Linux, Ubuntu Server, Corosync, Pacemaker, LVS.

Acrónimos

PC	Personal Computer
RAM	Random Access Memory
CPU	Central Process Unit
VM	Virtual Machine
SO	Sistema Operativo
HDD	Hard Disk Drive
HP	High Performance
HA	High Availability
HT	High Throughput
NAS	Network Access Server
NFS	Network File System
MB	Megabyte
GB	Gigabyte
RAID	Redundant Array of Inexpensive Disks
NAT	Network Address Translate
DHCP	Dynamic Host Configuration Protocol
RHEL	Red Hat Enterprise Linux
USB	Universal Serial Bus
STP	Spanning Tree Protocol
KVM	Kernel-based Virtual Machine
SPOF	Single point of failure

Tabla de contenidos

Tabla de Imágenes

1. Objetivos

Se tiene como objetivo principal en este TFG el conseguir tener un clúster de alta disponibilidad utilizando herramientas de software libre bajo licencias GNU, para ello se utilizará la herramienta de virtualización KVM (Kernel-based Virtual Machine) de Linux y como sistema operativo para los diferentes nodos Ubuntu Server. Una vez implementado y configurado el proyecto conseguiremos tener un clúster de servidores, todos ellos virtuales.

2. Motivación

La virtualización fue introducida a finales de 1990 en forma de máquinas virtuales en las PC. Sin embargo, el verdadero ímpetu para implantar la virtualización de servidores fue por la demanda de los clientes.

En los últimos años existe la tendencia tecnológica, por parte de las empresas, de apostar por la virtualización de sus equipos. Esto en gran parte es debido a que permite que varios ordenadores se puedan utilizar de forma independiente dentro de un mismo equipo físico. Todo esto permite a las empresas reducir costes ya que no debe obtener tanto equipo físico, por tanto, se reduce también el costo en energía y por otro lado esto también permite no tener limitaciones de espacio físico para alojar todos estos equipos.

Por otro lado, contar con un equipo más rápido, como un clúster, permite a las empresas adaptar sus necesidades y tener buen rendimiento en cuanto al tráfico o uso de aplicaciones alojadas. Hay que tener en cuenta que no todas las aplicaciones se pueden beneficiar de un clúster, pero las aplicaciones que sí se pueden beneficiar mejoran tiempos de respuesta incluso teniendo un número de peticiones mayor debido a que se pueden ir añadiendo elementos de cálculo a medida que aumenta la carga.

Hoy en día Linux está en un sin fin de dispositivos a nuestro alrededor, es el sistema operativo más usado en los servidores y esto es debido a una serie de razones que permiten a las empresas tener una mejor estabilidad ya que no presenta fallos durante años, una seguridad debido a la velocidad de reacción ante fallos, una flexibilidad debido a que puede configurarse y ajustarse tanto como sea necesario y todo esto unido a una reducción de costos incluso con versiones empresariales con soporte.

3. Estado del arte

3.1. Virtualización

La virtualización de servidores, sistemas operativos y redes juega un papel fundamental tanto para los administradores de sistemas como para los expertos en seguridad informática. Para los administradores de sistemas, es una herramienta perfecta para separar unos servicios de otros y ganar en seguridad, ya que si un hacker compromete un servicio no tendrá acceso a todo el sistema. [1]

La virtualización es la tecnología que permite crear múltiples entornos simulados o recursos dedicados desde un solo sistema de hardware físico. El software llamado "hipervisor" se conecta directamente con el hardware y permite dividir un sistema en entornos separados, distintos y seguros, conocidos como "máquinas virtuales" (VM). Estas VM dependen de la capacidad del hipervisor de separar los recursos de la máquina del hardware y distribuirlos adecuadamente. [2]

A la máquina física original equipada con el hipervisor se le llama "host", y las VM que utilizan estos recursos se llaman "guests" o “máquinas virtuales” (VM). Estas VM tratan a los recursos informáticos, como la CPU, la memoria y el almacenamiento, como un hangar de recursos que pueden trasladarse con facilidad. Los operadores pueden controlar las instancias virtuales de la CPU, la memoria, el almacenamiento y otros recursos para que las VM reciban los recursos que necesitan cuando los necesiten. [2]

Ventajas de la Virtualización:

- Reducción de costes tanto de espacio como de consumo.
- Incorporación de nuevos recursos de manera rápida y sencilla.
- Mejoras en los procesos de clonación y copias de seguridad.
- La administración es centralizada.
- Migración en caliente de la VM de un servidor físico a otro sin necesidad de realizar paradas.

Desventajas de la Virtualización:

- Coste inicial mayor debido a la necesidad de hardware de altas prestaciones.
- El servidor físico es un punto crítico debido a que en él están alojadas todas las VM y si sufriera cualquier anomalía afectaría a todas las VM.

Para subsanar esto deberíamos tener otro servidor físico que soportara todas las VM como respaldo, esto incrementaría el coste.

- Los sistemas operativos no tienen el mismo rendimiento en una VM que si son instalados en una máquina real.

3.1.1. Tipos de Virtualización

❖ **Virtualización de almacenamiento:**

Consiste en la virtualización del almacenamiento, esto reduce problemas generados por las grandes cantidades de datos que se manejan evitando así los habituales cuellos de botella cuando la demanda de datos es grande. Se busca así, con este tipo de virtualización, una mayor efectividad ya que tanto los discos como las memorias flash están separadas de los servidores en máquinas de mayor eficiencia.

❖ **Virtualización de Redes:**

Consiste en la virtualización de redes con redes virtuales y te permite tener lo mejor de ambos mundos. Para ello se reproduce totalmente la red mediante software y estas redes virtuales cuentan con las mismas características y garantías que las redes físicas, con las ventajas operativas y la independencia del hardware que ofrece la virtualización. Por lo tanto, aumenta el rendimiento en seguridad y en eficiencia, no teniendo los lazos que atan a una red física, pero respetando toda la funcionalidad. Es una de las opciones más demandadas y que mejores resultados proporciona sin importar el tipo de empresa. ([3] y [4])

❖ **Virtualización de Servidores:**

Consiste en la virtualización de servidores en máquinas virtuales proporcionando así la oportunidad de sacar el máximo partido a la instalación y permitiendo ejecutar varios sistemas operativos en un único servidor físico. Este tipo de virtualización mejora la eficiencia general y reduce los costos, así como también permite desplegar las cargas de trabajo más rápido, mejorar el rendimiento de las aplicaciones y aumentar la disponibilidad. [5]

❖ **Virtualización de escritorios:**

La virtualización de escritorios beneficia al informático teniendo controlado en un mismo lugar sin ningún tipo de complicación la presencia de escritorios a distancia que podrán utilizar otros miembros de la empresa sin importar cuál sea su situación. Reduce los costes y aumenta el servicio al ofrecer las aplicaciones y los escritorios virtualizados de forma rápida y sencilla a las

sucursales, a los empleados externos y que se encuentran en otros países, y a los trabajadores móviles. ([3] y [5])

En cuanto al tipo de virtualización por el que se ha optado en este TFG es la virtualización de servidores. Dentro de este tipo de virtualización existe unos subtipos los cuales dependiendo de su grado de complejidad son:

- **Virtualización completa:**

La máquina virtual simula un hardware específico para poder arrancar de forma aislada. Se puede llegar a tener varias instancias a la vez.

- **Virtualización parcial:**

La máquina virtual simula múltiples instancias del entorno subyacente del hardware. Particularmente se comparte el espacio de direcciones que permite alojar y compartir recursos o procesos. Dado que se están utilizando el mismo espacio de direcciones utilizando esta tecnología no se permite tener instancias separadas de las máquinas.

- **Virtualización Semi-parcial:**

La virtualización se lleva a cabo mediante la partición de un servidor físico al nivel del sistema operativo, permitiendo ejecutar múltiples servidores virtuales aislados sobre un único servidor físico.

Por lo tanto, podemos concluir, para ser más exactos, que la virtualización por la que se ha optado en este TFG es una virtualización de servidores completa.

3.2. Clúster de Servidores

Un clúster de servidores es un conjunto de servidores que se instalan para que trabajen como si fuera uno solo. Estos servidores se unen mediante una red de alta velocidad de tal manera, que el conjunto de los mismos se ve como un único servidor mucho más potente que si se tratase de un solo servidor.

Una de sus principales características es que los equipos que lo integran no tienen por qué tener las mismas características tanto a nivel de hardware como de sistema operativo.

Con este tipo de sistemas, clúster de servidores, se consigue obtener cuatro servicios principales, aunque dependerá del tipo de clúster que

utilicemos de que obtengamos los diferentes servicios. Estos cuatro servicios son:

- Alta disponibilidad
- Alto rendimiento
- Balanceo de carga
- Escalabilidad

3.2.1. Tipos de Clúster

Se pueden realizar diferentes clasificaciones de los diferentes tipos de clúster en función de diferentes conceptos, pero todas estas clasificaciones están relacionadas con los servicios que hemos nombrado en el punto anterior. Por lo tanto, refiriéndonos a los servicios anteriores tenemos tres tipos de clúster:

- **HP:** clúster de alto rendimiento, son sistemas que ejecutan tareas con gran capacidad de cálculo o que necesitan el uso de grandes cantidades de memoria. El tiempo para ejecutar las tareas suele ser elevado debido a la complejidad del programa y los recursos del clúster son utilizados casi en exclusiva para estas.

- **HA:** clúster de alta disponibilidad, son sistemas que buscan aportar disponibilidad y confiabilidad de los servicios que ofrecen. En este tipo de clúster lo que se hace es tener por duplicado el hardware, por lo tanto, si falla algún componente del clúster responderá el otro componente exactamente igual y que realiza el mismo trabajo, de esta manera se garantiza la disponibilidad del sistema, también incorporan software de detección y recuperación ante fallos, con objeto de hacer más confiable el sistema.

- **HT:** clúster de alta eficiencia, son sistemas que buscan el poder ejecutar el mayor número de tareas en el menor tiempo posible, estas tareas siempre deben ser individuales y que no tengan dependencias de datos entre ellas.

3.3. Sistemas Operativos

Un sistema operativo es el programa principal del ordenador a través del cual se permite la interacción entre la máquina y el usuario. A parte también realiza funciones como las de coordinar la interacción entre el *hardware* (elementos físicos de la máquina) y el *software* (programas) o gestionar la memoria y archivos que se encuentran en el ordenador.

3.3.1. Comparativa Linux y Windows Server

A la hora de elegir el sistema operativo para un servidor hay que tener en cuenta cuales son las necesidades del proyecto para decantarnos por el sistema operativo. A continuación, detallaremos las ventajas e inconvenientes que poseen cada uno de los dos sistemas operativos más presentes en servidores hoy en día.

3.3.1.1. Linux

Es un sistema operativo gratuito y de software libre. Linux es el núcleo del sistema operativo que por sí mismo no funciona, al ser software libre cualquier usuario puede modificar su código fuente.

Hoy en día existen muchas distribuciones de Linux, variantes de Linux, algunas de la más comunes son RHEL, Ubuntu, Debian o CentOS. Cada una de ellas tiene sus particularidades, pero al partir de Linux, todas siguen siendo de código abierto.

Ventajas de Linux:

- Seguridad: los ataques a estos sistemas operativos son menos frecuentes, a parte solo el administrador o *"root"* tiene permisos para realizar cambios importantes a nivel de sistema.
- Estabilidad: este SO no requiere de una desfragmentación del disco, por lo tanto, le permite ser un sistema más rápido y robusto. Una de sus características es la de procesar gran cantidad de procesos al mismo tiempo.
- Reducción de costos: como hemos indicado anteriormente es un sistema gratuito, por lo tanto, su implantación también lo es. Si bien es cierto algunas distribuciones cobran por soporte técnico, pero es más barato que Windows.
- Comunidad: Linux cuenta con una comunidad de desarrolladores y usuarios que aporta gran cantidad de documentación de manera desinteresara y que es de gran ayuda.

Desventajas de Linux:

- Compatibilidad: algunos elementos del hardware, así como algunos programas no son compatibles con Linux.
- Soporte especializado: la gran mayoría de las distribuciones de Linux no cuenta con un soporte técnico profesional en caso de que tengamos problemas. La distribución RHEL sí que tiene soporte profesional, pero con un costo, no siempre superior al de Windows.
- Difícil para usuarios novatos: hay ciertas tareas como actualizaciones o configuraciones que pueden ser difíciles para usuarios novatos, prácticamente todo este tipo de tareas se realizan a través de comandos.

3.3.1.2. Windows Server

Es un sistema operativo de Microsoft, de software privado y que por lo tanto su código no es público y las actualizaciones dependen de la empresa. De este también nos encontramos con diferentes versiones, aunque son bastantes más escasas que las de Linux.

Ventajas de Windows Server:

- Soporte: cuenta con un soporte técnico profesional siempre disponible ante cualquier problema, pero siempre con un coste superior al de Linux.
- Actualizaciones: estas están automatizadas y son fáciles de instalar.
- Compatibilidad: cuenta con los controladores más funcionales y actualizados, por lo tanto, cualquier instalación o cambio de un controlador resulta más sencilla.
- Amable con el usuario: es uno de los objetivos principales de Microsoft. Su manejo intuitivo lo hace más práctico y por lo tanto más accesible para mayor cantidad de usuarios.

3.3.1.3. Conclusión

Como hemos observado cada sistema operativo tiene sus pros y sus contras. Si, por un lado, buscamos una seguridad y estabilidad, todo esto añadido a su bajo costo, nos decantaremos por distribuciones de Linux, si, por el contrario, buscamos tener un soporte técnico profesional garantizado o una mayor facilidad de uso, Windows Server será mejor opción.

No obstante, cabe destacar que de la lista Top500, donde se muestran los mejores supercomputadores del mundo, el 99,6 % de ellos utilizan distribuciones Linux. Por lo tanto, observamos que, a la hora de administrar un supercomputador o un servidor, en prácticamente todas las ocasiones se administra a través de distribuciones de Linux.

En nuestro caso para la virtualización de los servidores del cluster que se presenta en este TFG nos hemos decantado por la distribución de Linux, Ubuntu Server.

4. Descripción de la Solución

Tenemos como máquina anfitriona, es decir, la máquina física donde se van a virtualizar todas las máquinas que componen el clúster, un portátil **HP** modelo **dv6-6b13ss** con las siguientes características:

- SO: Ubuntu 16.04 LTS
- CPU: Intel Core i7-2670QM CPU 2.2GHz x 8
- RAM: 16 GB
- HDD: 1 TB HD

Se ha optado por el sistema operativo Ubuntu ya que es la distribución de Linux más utilizada y amigable para su configuración. A la hora de obtener soporte o ayuda para ciertos bloqueos o dudas, dispone de gran cantidad de foros oficiales donde tanto desarrolladores como otros usuarios suben documentación que es de gran ayuda y de fácil y rápida obtención.

4.1. Instalación de Ubuntu en la máquina física

Empezaremos descargándonos la imagen del sistema operativo de la página web oficial de Ubuntu (https://www.ubuntu.com/download/desktop). Una vez hemos accedido a la página web seleccionaremos la versión deseada, en nuestro caso es la versión 16.04 LTS.

Una vez tenemos la imagen de Ubuntu descargada la montamos en un pendrive USB. Insertamos el pendrive USB en la máquina y realizamos la instalación.

A la hora de la instalación seleccionamos el idioma y la zona horaria, ponemos el nombre a la máquina, indicamos el usuario y contraseña y todo lo demás lo dejamos por defecto. Si fuera necesario configurar algo más, posteriormente se indicará, por lo tanto, la configuración inicial de la máquina seria:

- Nombre de la máquina: admin-PC
- Usuario: administrador
- Idioma: español

- Zona Horaria: Europa/Madrid

Después de unos minutos la instalación ha finalizado, por lo tanto, ya podemos iniciar sesión con nuestro usuario y contraseña. Antes de nada, configuramos nuestra red wifi para que la máquina tenga conexión a Internet.

Una vez se ha establecido la conexión lo primero que haremos es actualizar nuestro sistema operativo:

```
$ sudo apt-get update
```

Tras haber realizado la actualización del sistema operativo reiniciamos la máquina ya que puede que alguna de las actualizaciones instaladas lo requieran. Insertamos usuario y contraseña y ya tenemos la máquina física preparada.

4.2. Instalación de KVM

Comenzaremos instalando KVM, solución para implementar virtualizaciones completas con Linux y que por lo tanto nos va a permitir ejecutar nuestras máquinas virtuales del clúster.

Al instalar KVM en nuestra máquina física conseguimos que Linux pase a ser nuestro hipervisor, a parte como KVM forma parte del *kernel* de Linux cuenta con todos los componentes que cualquier hipervisor necesita a nivel de sistema operativo.

Antes de instalar KVM debemos comprobar que nuestro hardware ofrezca soporte para la virtualización, por lo tanto, ejecutaremos:

```
$ egrep -c '(svm|vmx)' /proc/cpuinfo
```

Si al ejecutar el comando anterior nos devuelve un '0' es que nuestro hardware no tiene soporte para virtualización, en nuestro caso nos devolvió un '8' y por lo tanto procedemos a instalar KVM.

Para instalar KVM junto con los componentes necesarios para aprovisionar a las máquinas virtuales ejecutaremos desde la línea de comandos de nuestra máquina física el siguiente comando:

```
$ sudo apt-get install qemu-kvm libvirt-bin
virtinst kvm virt-viewer -y
```

Después de que termine la instalación de KVM con los componentes necesitamos añadir nuestro usuario de la máquina física al grupo kvm para

poder estar habilitados para usar KVM y cargamos el driver del *kernel* de Intel, por lo tanto, ejecutamos:

```
$ sudo adduser administrador kvm
$ sudo modprobe kvm-intel
```

Una vez que se haya completado lo anterior, debemos verificar que los módulos KVM se hayan cargado para poder hacer un uso completo de KVM, ejecutamos:

```
$ lsmod | grep kvm
```

Después de finalizar la instalación de KVM, y antes de ponernos a montar el entorno de nuestro clúster, tenemos que realizar una última configuración en nuestra máquina anfitrión para que las máquinas de nuestro clúster tengan acceso a internet. Para ello realizamos los siguientes pasos en nuestra máquina anfitrión:

- Modificamos el siguiente fichero añadiendo las líneas:

```
$ nano /etc/network/interfaces
```

```
...
auto kvmbr0
iface kvmbr0 inet dhcp
      bridge_ports eno1
      bridge_stp off
```

Añadiendo estas líneas en el fichero estamos creando la interfaz virtual (kvmbr0) y le hemos definido como puente la interfaz física de la máquina anfitrión eno1. Con la línea "*bridge_stp off*" desactivamos el protocolo STP (*Spanning Tree Protocol*) el cual nos permite correr múltiples *bridges*.

4.3. Estructura y Modelo del Clúster

Una vez tenemos a punto la máquina física procedemos a montar lo que propiamente el clúster. Para ello hemos instalado Ubuntu 16.04 LTS y KVM en la máquina física y a partir de procedemos a montar el clúster virtualizado de alta disponibilidad.

A continuación, os mostramos la estructura del clúster, elementos que lo componen y cómo van a estar conectados entre sí:

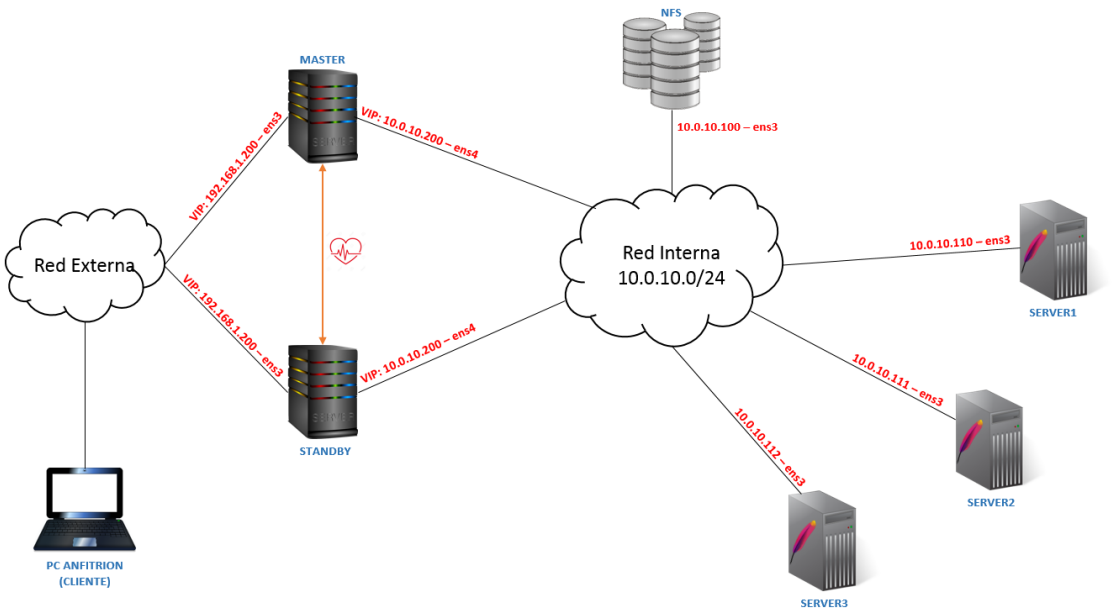

Imagen 1. Estructura del Clúster de Alta Disponibilidad

Como podemos observar en la *Imagen 1*, el clúster que se monta en este TFG consta de varios servidores, cada uno de ellos con funciones diferentes y muy específicas. Todos estos servidores están virtualizados a través de KVM. Como también podemos observar hay una red interna para conectar los diferentes componentes del clúster, a esta red solo tendrán acceso los servidores del clúster y no será accesible desde ningún otro sitio.

A continuación, vamos a detallar la función de cada uno de los elementos que componen este clúster:

- **Nodos Master y Standby:**

Son los nodos principales del clúster. Estos dos servidores son exactamente iguales, es decir, que el nodo **Standby** es una copia exacta del nodo **Master**. Forman entre los dos un sistema activo-pasivo (master-standby) y esto está hecho así para conseguir la alta disponibilidad. Por lo tanto, podemos decir que son la parte *front-end* del sistema y normalmente el nodo activo será el **Master.**

Este nodo se encargará de recibir las peticiones de los clientes y distribuir la carga entre los servidores que componen el *back-end* del sistema. Como podemos observar en la imagen estos nodos tienen tres conexiones de red, una de ellas mantendrá la conexión con el resto de los elementos del clúster, las otras dos son conexiones entre los dos, una de estas conexiones se utilizará para que estos dos nodos se intercambien el estado de cada uno de ellos, es decir, a través de esta red se detectaran si un nodo se ha caído y por lo tanto el otro tomara el relevo, con esto se consigue la alta disponibilidad. La

tercera red se utilizará como respaldo de la anterior y por si es necesario utilizarse para cualquier otra cosa que no sea la funcionalidad anterior.

Por otra parte, el nodo **Standby**, como hemos descrito anteriormente es una copia exacta del nodo **Master**, estará a la espera y escuchando en todo momento al nodo **Master**, solo cuando el nodo **Master** se caiga este tomará el control y realizara la misma función.

- **Nodo NFS:**

Este nodo es el que conforma el servicio de almacenamiento del clúster. Este servicio de almacenamiento es el encargado de servir, a través de la red, las páginas web a los servidores **Apache**. Por lo tanto, cuando un usuario haga una petición de una página web, el nodo **Master** le pedirá a uno de los servidores **Apache** que le sirva dicha página y este servidor deberá comunicarse con el servicio de almacenamiento para recoger esa página web y servirla.

El nodo **NFS** tiene montado internamente en la máquina un *RAID 6*, este no penaliza el rendimiento de lectura, que nos interesa, y a su vez soporta que dos discos dejen de funcionar a la vez.

- **Nodos Server1, Server2 y Server3:**

Estos tres nodos son tres servidores web reales que se encargan de servir las páginas web que les piden los nodos **Master** y **Standby.** Para ello tienen instalado cada uno de ellos un servidor apache que atiende las peticiones que les solicita el nodo de *front-end* que este activo, **Master** o **Standby**.

Estos servidores se comunican tanto con los servidores de *front-end* como con el servidor de almacenamiento distribuido a través de la interna del clúster.

A continuación, vamos a detallar las diferentes redes que componen el clúster y explicar su función principal:

- **Red Externa:**

Esta red es desde la cual se simulan los accesos al clúster desde el exterior, es decir, desde Internet. Esta red permite la comunicación entre la máquina física y las máquinas virtuales del clúster, por lo tanto, a través de esta red y un navegador instalado en la máquina física realizaremos las peticiones al clúster.

- **Red Interna:**

Esta red será la común a todo el clúster, es decir será la red interna del clúster a la que todos los componentes del mismo tendrán acceso. A través de esta red los nodos del *front-end* (**Master** o **Standby**) se comunicarán con los nodos **Apache** del clúster haciéndoles llegar las peticiones web, también a través de esta red estos últimos, nodos **Apache**, se comunicarán con el servidor de almacenamiento distribuido. Los nodos de *front-end* también se podrán comunicar con el servidor de almacenamiento.

4.4. Creación red virtual interna

Antes de comenzar con la instalación y configuración de las máquinas virtuales que comprenderán nuestro clúster, vamos a crear nuestra red interna, a la cual estarán conectados todos los nodos del clúster. Para nuestra red virtual interna simplemente tenemos que tener en cuenta que todas las direcciones serán estáticas, es decir, que no se asignaran a través de DHCP y que posteriormente habrá que configurarla de manera manual en cada uno de los nodos.

Para la creación de la red nos ayudaremos de la herramienta de Ubuntu "Virtual Machine Manager" y los pasos a seguir son:

- **Paso 1**: en la pantalla principal hacemos botón derecho sobre la barra "QEMU/KVM":

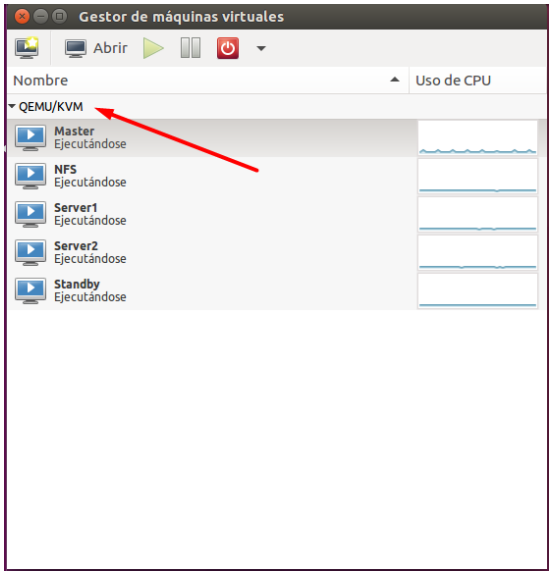

Imagen 2. Pantalla Inicio Virtual Machine Manager

- **Paso 2**: despliega un menú en el cual seleccionamos "Detalles":

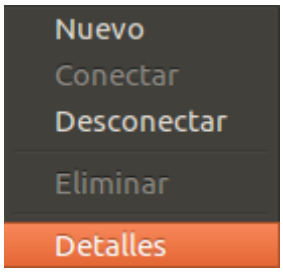

Imagen 3. Menú Desplegable

- **Paso 3**: abre una pantalla con todas las redes que tenemos creadas hasta el momento. Le daremos al botón "+" para crear una nueva red:

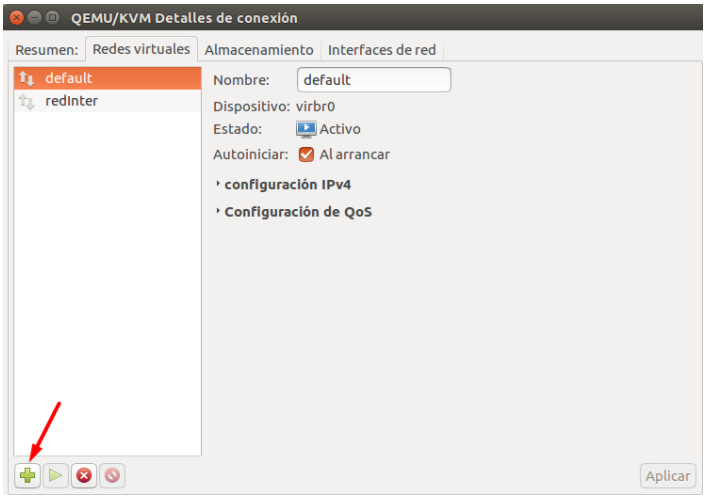

Imagen 4. Pantalla Detalle de conexión

- **Paso 4**: nos abrirá la primera pantalla del asistente para crear una red donde introducimos el nombre de la red (redInter) y le damos al botón "Adelante":

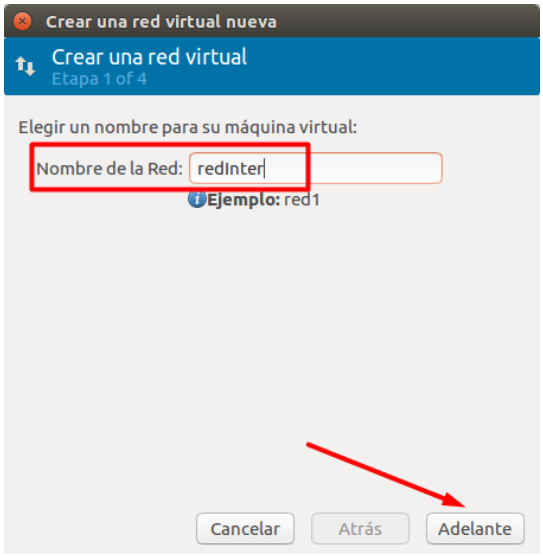

Imagen 5. Pantalla 1/4 - Asistente Creación de una red

- **Paso 5**: en la segunda pantalla del asistente introducimos el espacio de dirección de red que deseamos, en nuestro caso será 10.0.10.0/24, desmarcamos todo y le daremos al botón "Adelante":

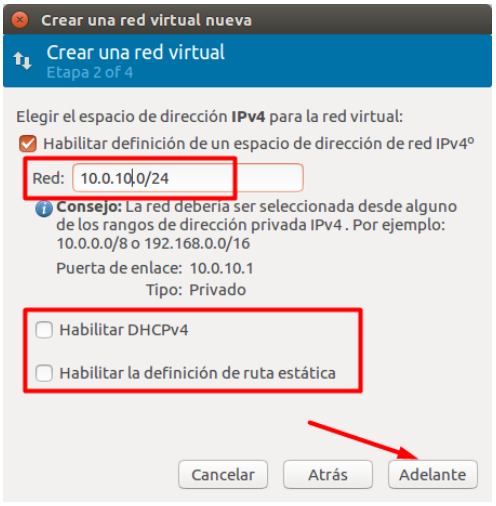

Imagen 6. Pantalla 2/4 - Asistente Creación de una red

- **Paso 6**: en la tercera pantalla del asistente dejamos todo como viene por defecto y le damos al botón "Adelante":

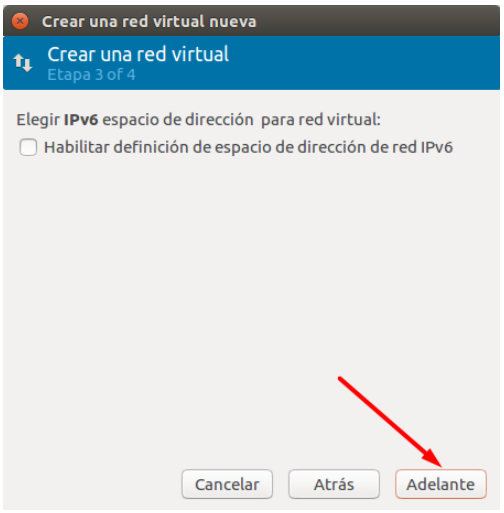

Imagen 7. Pantalla 3/4 - Asistente Creación de una red

- **Paso 7**: en la cuarta y última pantalla del asistente seleccionaremos "Red virtual aislada", deseleccionamos todo lo demás y le damos al botón "Finalizar":

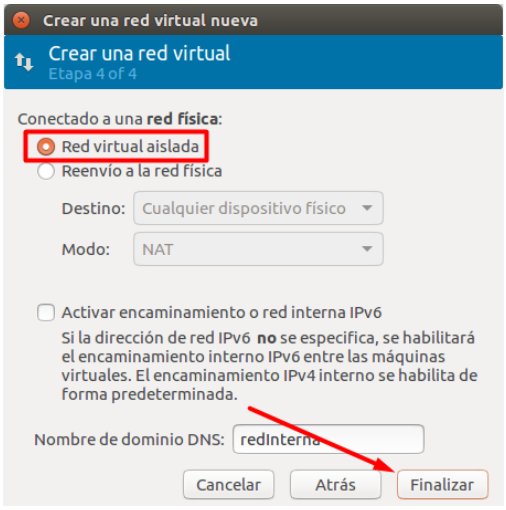

Imagen 8. Pantalla 4/4 - Asistente Creación de una red

4.5. Instalación y configuración de las máquinas virtuales

Comenzamos instalando el nodo de almacenamiento (NFS) el cual ofrecerá el espacio de almacenamiento al resto de los servidores del clúster.

El servidor NFS nos permite tener una máquina dedicada al almacenamiento. Son utilizados en entornos de red de área local para sistemas de archivos distribuidos. Este sistema nos permite que distintas máquinas conectadas a una misma red puedan acceder a ficheros remotos como si fueran locales. Este protocolo está incluido por defecto en la mayoría de las distribuciones de Linux.

El sistema NFS está compuesto por al menos dos partes, un servidor y uno o varios clientes los cuales accederán de forma remota a los ficheros y datos que están alojados en el servidor. Esto posibilita que las máquinas clientes no precisen de tanto almacenamiento local debido a que los datos están centralizados en el servidor NFS, aparte de datos también se pueden compartir dispositivos como unidades ZIP, CD-ROM, etc.

Las características de la máquina virtual del nodo de almacenamiento serán las siguientes:

- SO: Ubuntu Server 16.04 LTS
- VCPU: 2 cores
- RAM: 512 MB
- HDD: 10GB
- RED1: conectada a la red interna 10.0.10.0/24

Para comenzar con la instalación nos ayudamos de la herramienta de Ubuntu "Virtual Machine Manager" la cual es un administrador de máquinas virtuales de escritorio. Arrancaremos la máquina virtual, seleccionaremos el archivo ".iso" ubicado en el escritorio de la máquina física y el cual contiene la distribución Ubuntu que vamos a instalar. Para ello se realizan los siguientes pasos:

- **Paso 1:** clicamos en el botón para crear una nueva máquina virtual.

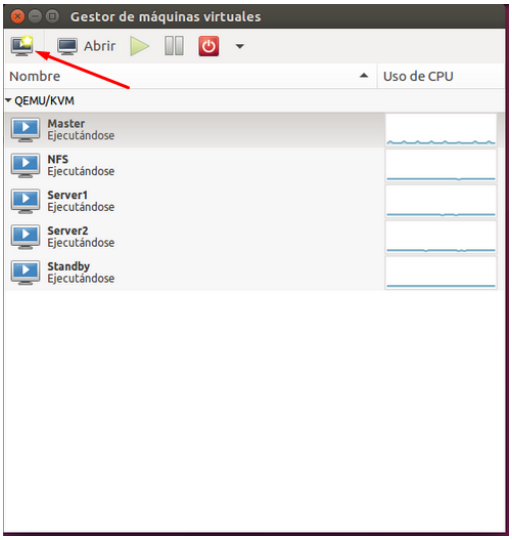

Imagen 9. Pantalla Inicial del Administrador de máquinas virtuales (Virtual Machine Manager)

- **Paso 2:** seleccionamos desde donde cargaremos la ".iso" y le damos al botón Adelante.

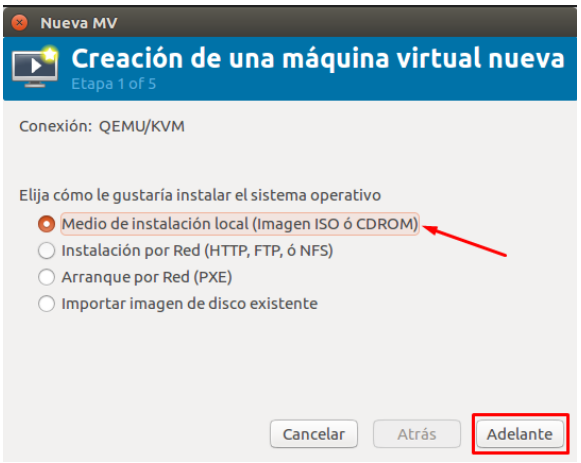

Imagen 10. Asistente de creación de máquinas virtuales. Pantalla 1/5

- **Paso 3:** indicamos el directorio donde se encuentra la ".iso" y le damos al botón Adelante.

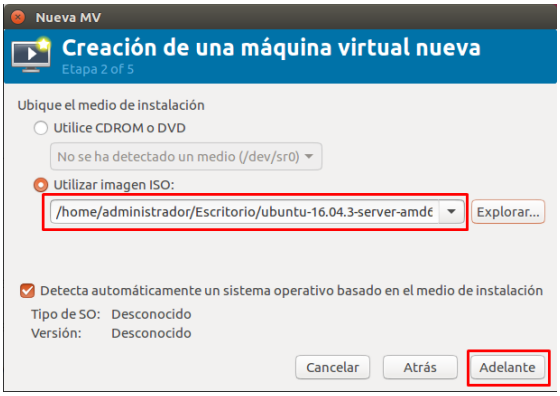

Imagen 11. Asistente de creación de máquinas virtuales. Pantalla 2/5

- **Paso 4:** indicamos la RAM y los *cores* que va a tener la máquina y le damos al botón Adelante.

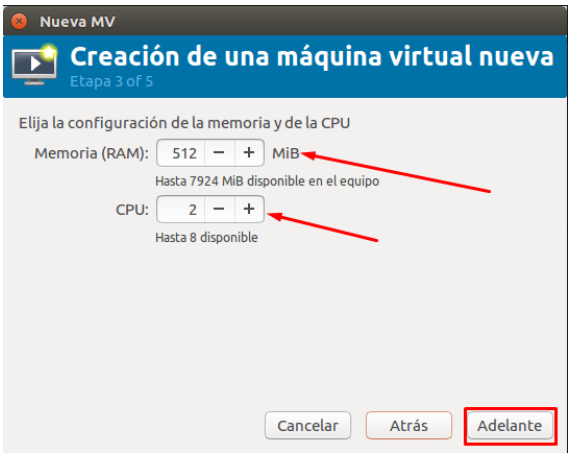

Imagen 12. Asistente de creación de máquinas virtuales. Pantalla 3/5

- **Paso 5:** indicamos la capacidad del disco duro y le damos al botón Adelante.

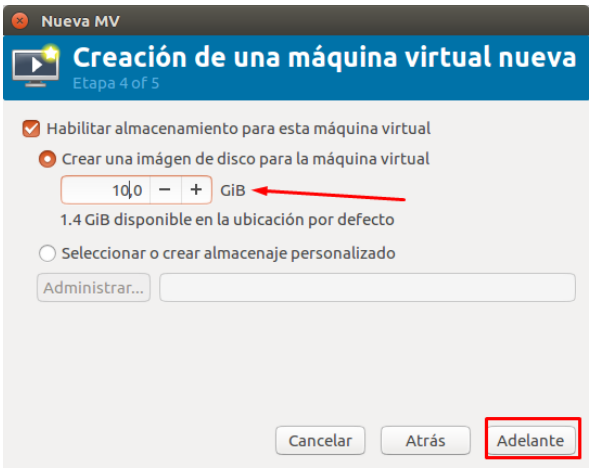

Imagen 13. Asistente de creación de máquinas virtuales. Pantalla 4/5

- **Paso 6:** indicamos el nombre de la máquina, seleccionamos a que red ira conectada la interfaz de red y le damos al botón Finalizar.

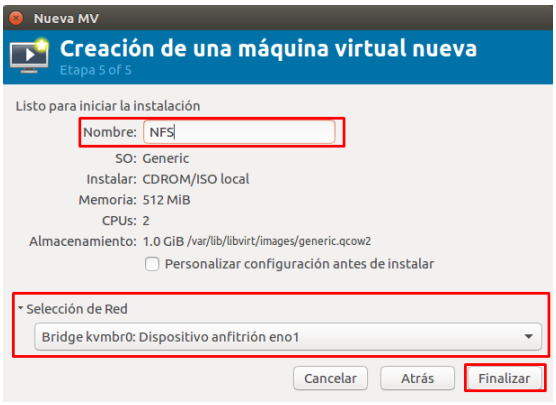

Imagen 14. Asistente de creación de máquinas virtuales. Pantalla 5/5

También podemos crear las máquinas virtuales a través de la línea de comandos, para crear la máquina anterior (NFS) ejecutaríamos en el equipo anfitrión:

```
$ virt-install --connect qemu:///system -name NFS --
memory 512 -disk
path=/var/lib/libvirt/images/NFS.qcow2,size=10
--vcpus=2 -c /home/administrador/escritorio/ubuntu-
16.04.3-server-amd64.iso --vnc --os-type linux --
network bridge=kvmbr0 -noautoconsole --hvm --keymap
es
```

Vamos contestando a las diferentes preguntas de instalación del SO relativas al idioma, zona horaria, etc., y comenzaremos con la instalación de Ubuntu Server.

Los siguientes pasos a seguir son:

- Nombre de la máquina: “nas”
- Usuario y contraseña: “administrador” y “administrador”
- Se realizará un particionado de disco manual, la configuración del mismo será la siguiente:
  - **/dev/sd1** → Sistema, punto de montaje “/”. 3GB y formato ext4.
  - **/dev/sd2** → Almacenamiento local, punto de montaje “/home”. 6GB y formato ext4.
  - **/dev/sd3** → Swap. Tamaño el espacio restante y formato “área de intercambio”.
- Dejaremos la configuración automática de red y sin proxy.
- Sin actualizaciones automáticas.
- Seleccionaremos que instale OpenSSH Server, así no tendremos que instalarlo posteriormente.
- Instalaremos el cargador de arranque en el GRUB de MBR.

Cuando haya finalizado la instalación reiniciaremos el sistema y tendremos nuestro SO instalado en la máquina virtual.

Ahora que ya tenemos la máquina virtual creada, la arrancaremos e iniciaremos sesión con el usuario y contraseña elegidos en la instalación, en nuestro caso es "administrador" en ambos. Una vez logados cambiaremos la contraseña del superusuario "root" con el siguiente comando, como contraseña hemos elegido "root":

```
$ sudo passwd root
```

Indicaremos la contraseña deseada para "root" en nuestro caso será la misma que el usuario, es decir contraseña = root.

Seguidamente actualizaremos el sistema e instalaremos los paquetes requeridos para el servidor NFS, para ello ejecutaremos:

```
# Comando para actualizar el sistema
    $ apt-get update
# Comando para instalar el servidor NFS
    $ apt-get install rpcbind nfs-kernel-server
```

Una vez actualizado e instalados los paquetes del servidor NFS, comprobamos que se ha instalado correctamente el servidor ssh que indicamos en los ajustes de instalación, para comprobarlo podemos ejecutar:

```
$ apt-get install openssh-server
```

Con el comando anterior estamos solicitando su instalación, y si este ya está instalado nos lo indicara que ya está en su versión más reciente. También podemos ejecutar el siguiente comando que nos indica si el servidor está en ejecución:

```
$ service ssh status
```

Una vez hemos comprobado que tenemos instalado nuestro servidor ssh pasamos a su configuración, para ello editamos su archivo de configuración, para ello ejecutaremos el siguiente comando:

```
$ nano /etc/ssh/sshd_config
```

Editáramos en este fichero la siguiente línea para que el servidor ssh acepte conexiones remotas a la cuenta root:

```
...
PermitRootLogin yes
...
```

Una vez editado, reiniciamos el servicio para que los cambios se hagan efectivos:

```
$ service ssh restart
```

Pasamos a configurar la red. El nodo de almacenamiento estará conectado a la red interna del clúster, por lo tanto, le asignaremos de manera estática la dirección 10.0.10.100/255.255.255.0, como puerta de enlace 10.0.10.1 (esta es la dirección IP que le asignaremos al nodo MASTER más adelante) y como servidor de nombres el nodo MASTER (10.0.10.1), para ello editamos el siguiente fichero:

```
$ nano /etc/network/interfaces
```

Editáramos en este fichero la siguiente línea:

```
...
auto ens3
iface ens3 inet static
      address 10.0.10.100
      netmask 255.255.255.0
      gateway 10.0.10.1
      mtu 9000
      dns-nameservers 10.0.10.1
```

Pasamos a configurar el servidor NFS, para ello exportaremos el directorio "/home" del nodo de almacenamiento para que tenga la apariencia del directorio "/nfs" para todas las máquinas del clúster que estén en la red interna (10.0.10.0/24) con permisos de lectura y escritura. Ejecutaremos los siguientes comandos y editaremos los ficheros con las siguientes líneas:

```
$ nano /etc/fstab
```

```
...
/home   /nfs   none   bind   0   0
```

```
$ nano /etc/exports
```

```
...
/nfs
10.0.10.0/24(fsid=0,rw,sync,no_subtree_check,no_root_squash)
```

Lo montamos manualmente, reiniciamos el servicio y comprobamos que no tenemos ningún error con los siguientes comandos:

```
# Comando crear el directorio
     $ mkdir /nfs
# Comando montar el directorio
     $ mount /nfs
# Comando para reiniciar el servicio NFS
     $ service nfs-kernel-server restart
```

Cuando hemos terminado de configurar la máquina la apagamos, se puede ejecutar en comando **porweroff** o bien ayudarnos del gestor de máquinas virtuales "Virtual Machine Manager" dándole al botón de apagado. Una vez apagada la máquina conectaremos el interfaz de red a la red interna creada (10.0.10.0/24) y volveremos a encender la máquina.

En el siguiente paso instalamos y realizamos la configuración básica del nodo MASTER. La instalación del nodo MASTER es idéntica a la del nodo NFS realizando unos pequeños cambios tanto en la red como en la configuración, las características de la máquina virtual del nodo MASTER serán las siguientes:

- SO: Ubuntu Server 16.04 LTS
- VCPU: 2 cores
- RAM: 512 MB
- HDD: 10GB
- RED1: conectada al bridge con el host anfitrión, IP por DHCP
- RED2: conectada a la red interna 10.0.10.0/24

Los pasos a seguir para la creación de la máquina virtual y durante la instalación del SO son idénticos al nodo NFS, contestando a las diferentes preguntas relativas al idioma, zona horaria, etc., y comenzaremos con la instalación de Ubuntu Server. Para crear la máquina por línea de comandos en este caso ejecutaríamos en el equipo anfitrión:

```
$ virt-install --connect qemu:///system –name Master
--memory 512 –disk
path=/var/lib/libvirt/images/Master.qcow2,size=10
--vcpus=2 -c /home/administrador/escritorio/ubuntu-
16.04.3-server-amd64.iso --vnc --os-type linux --
network bridge=kvmbr0 –network bridge=kvmbr0 –
noautoconsole --hvm --keymap es
```

Los siguientes pasos a seguir son:

- Nombre de la máquina: "master"
- Usuario y contraseña: "administrador" y "administrador"
- Haremos el particionado de disco el manual, la configuración del mismo será la siguiente:
  - **/dev/sd1** → Sistema, punto de montaje "/". 3GB y formato ext4.

    - **/dev/sd2** → Almacenamiento local, punto de montaje "/home". 6GB y formato ext4.
    - **/dev/sd3** → Swap. Tamaño el espacio restante y formato "área de intercambio".
- Dejaremos la configuración automática de red y sin proxy.
- Sin actualizaciones automáticas.
- Seleccionaremos que instale OpenSSH Server, así no tendremos que instalarlo posteriormente.
- Instalaremos el cargador de arranque en el GRUB de MBR.

Cuando haya finalizado la instalación reiniciaremos el sistema y tendremos nuestro SO instalado en la máquina virtual.

Ahora que ya tenemos la máquina virtual creada, la arrancaremos e iniciaremos sesión con el usuario y contraseña elegidos en la instalación, en nuestro caso es "administrador" en ambos. Una vez logados cambiaremos la contraseña del superusuario "root" con el siguiente comando, como contraseña hemos elegido "root":

```
$ sudo passwd root
```

Primeramente, comprobaremos que se ha instalado correctamente el servidor ssh, con el siguiente comando solicitamos su instalación y si ya lo está nos indicara que ya está en su última versión.

```
$ apt-get install openssh-server
```

También podríamos utilizar el siguiente comando para comprobar el estado del servidor:

```
$ service ssh status
```

Una vez comprobado que tenemos instalado el servidor ssh vamos a configurarlo para que acepte conexiones remotas a la cuenta de root.

```
$ nano /etc/ssh/sshd_config
```

Editáramos en este fichero la siguiente línea y reiniciamos:

```
...
# PermitRootLogin without-password
PermitRootLogin yes
...
```

```
$ service ssh restart
```

El siguiente paso es configurar la red del nodo MASTER, la interfaz "ens3", la cual está conectada al exterior, se configura para que arranque por DHCP, la interfaz "ens4" le asignaremos una IP fija, del rango de direcciones de la red interna (10.0.10.0/24), para que se comunique con las demás máquinas del clúster, para ello ejecutamos:

```
$ nano /etc/network/interfaces
```

Editáremos el fichero dejándolo de la siguiente manera:

```
auto lo
iface lo inet loopback
auto ens3
iface ens3 inet dhcp
auto ens4
iface ens4 inet static
      address 10.0.10.1
      netmask 255.255.255.0
      mtu 9000
```

Para que los cambios surjan efecto reiniciamos el servicio de red:

```
$ service networking restart
```

El siguiente paso es generar la clave pública para tener el acceso por ssh, esta clave la generamos en el nodo MASTER, la copiamos en el archivo que contiene las claves autorizadas y posteriormente las copiaremos a los diferentes servidores que comprenden el clúster, para ello ejecutaremos los siguientes comandos:

```
# Comando para generar la clave
     $ ssh-keygen

# Comando para copiar la clave al archivo
     $ cp /root/.ssh/id_rsa.pub /root/.ssh/authorized_keys
```

Para la resolución de nombres debemos modificar el fichero **hosts**, añadiendo en las parejas de IP/nombres. Por lo tanto, editaremos el fichero y su contenido será el siguiente:

```
$ nano /etc/hosts
```

```
# NFS SERVER

10.0.10.100   nas nfsserver nas.cluster nfsserver.cluster



# MASTER - STANDBY
```

```
10.0.10.1   master master.cluster

10.0.10.2   standby standby.cluster



# GRANJA

10.0.10.110   server1 server1.cluster

10.0.10.111   server2 server2.cluster

10.0.10.112   server3 server3.cluster
```

Una vez tenemos configurado el fichero **hosts** ya podemos copiar las claves que hemos generado en el nodo MASTER para ssh al nodo de almacenamiento NFS:

```
# Nos conectamos por ssh a "nas" y creamos el
directorio
     $ ssh nas "mkdir /root/.ssh"
# Copiamos la clave de MASTER a NFS(nas)
     $ scp /root/.ssh/id_rsa.pub nas:/root/.ssh/authorized_keys
```

También copiamos el fichero **hosts**:

```
     $ scp /etc/hosts nas:/etc
```

El siguiente paso es instalar los paquetes correspondientes al cliente NFS:

```
     $ apt-get install nfs-common
```

Una vez instalados los paquetes de NFS cliente procedemos a configurarlo, para ello montamos el sistema de archivo nfs en el arranque creando una entrada en el fichero **fstab**:

```
     $ nano /etc/fstab
```

Añadimos la siguiente línea al final del fichero:

```
...
nas:/nfs /nfs nfs auto,rsize=8192,wsize=8192 0 0
```

Finalmente lo montamos manualmente:

```
# Creamos el directorio
     $ mkdir /nfs
# Montamos el directorio
```

```
$ mount /nfs
```

Ahora hacemos la configuración para que los servidores del clúster tengan acceso al exterior a través del nodo MASTER, para ello en el nodo MASTER activamos *ip forwarding* y configuramos el cortafuegos. Editamos el siguiente archivo dejando su contenido de la siguiente manera:

```
$ nano /etc/rc.local
```

```
...
sysctl -w net.ipv4.ip_forward=1
iptables -P FORWARD ACCEP
iptables --table nat -A POSTROUTING -o ens3 -j MASQUERADE
exit 0
```

Con la última entrada en el fichero anterior estamos indicando en la tabla NAT (`--table nat`), que vamos a modificar los paquetes antes de que salgan del equipo (`-A POSTROUTING`) por la interfaz (`-o ens3`) y cambiando la dirección IP de origen por la que tenga la interfaz de salida (`-j MASQUERADE`).

Una vez modificado y guardado el fichero le daremos permisos de ejecución y lo lanzaremos manualmente, solo la primera vez para no tener que reiniciar la máquina y los cambios surjan efecto:

```
$ chmod +x /etc/rc.local
$ /etc/rec.local
```

Para comprobar que la activación de *ip_forwarding* y la configuración del cortafuegos esta correcta, nos conectamos a la máquina NFS (nas) y realizamos un ping a la IP del MASTER y posteriormente realizamos otro ping a la IP de Google, si la configuración es correcta ambos pings deben recibir respuesta:

```
$ ping 10.0.10.1
$ ping 8.8.8.8
```

Una vez tenemos la configuración básica del nodo de almacenamiento NFS y del nodo MASTER pasamos a crear los servidores Apache que nos ofrecerán las webs que tengamos alojadas en NFS. Para ello realizamos la instalación y configuración del nodo SERVER1, luego clonaremos la imagen de esta máquina para generar los nodos SERVER2 y SERVER3.

Vamos a instalar y realizar la configuración básica del nodo SERVER1. La instalación del nodo SERVER1 es idéntica a la del nodo NFS o nodo MASTER realizando unos pequeños cambios, las características de la máquina virtual del nodo SERVER1 serán las siguientes:

- SO: Ubuntu Server 16.04 LTS
- VCPU: 2 cores
- RAM: 512 MB
- HDD: 5GB
- RED1: conectada a la red interna 10.0.10.0/24

Los pasos a seguir durante la creación de la máquina virtual e instalación del SO son idénticos al nodo NFS o MASTER, contestando a las diferentes preguntas relativas al idioma, zona horaria, etc., y comenzaremos con la instalación de Ubuntu Server. Para crear la máquina por línea de comandos en este caso ejecutaríamos en el equipo anfitrión:

```
$ virt-install --connect qemu:///system –name Server1
--memory 512 –disk
path=/var/lib/libvirt/images/Server1.qcow2,size=5
--vcpus=2 -c /home/administrador/escritorio/ubuntu-
16.04.3-server-amd64.iso --vnc --os-type linux --
network bridge=kvmbr0 –noautoconsole --hvm --keymap
es
```

Los siguientes pasos a seguir son:

- Nombre de la máquina: “server1”
- Usuario y contraseña: “administrador” y “administrador”
- Haremos partición del disco guiado utilizando todo el disco.
- Dejaremos la configuración automática de red y sin proxy.
- Sin actualizaciones automáticas.
- Seleccionaremos que instale OpenSSH Server, así no tendremos que instalarlo posteriormente.
- Instalaremos el cargador de arranque en el GRUB de MBR.

Cuando haya finalizado la instalación reiniciaremos el sistema y tendremos nuestro SO instalado en la máquina virtual.

Como hemos hecho en los nodos anteriores realizamos la configuración lo primero que hacemos es cambiar la contraseña del usuario root, como contraseña hemos elegido “root”:

```
$ sudo passwd root
```

Comprobamos que se ha instalado correctamente el servidor ssh, con el siguiente comando solicitamos su instalación y si ya lo está nos indicara que ya está en su última versión.

```
$ apt-get install openssh-server
```

Después configuramos el servidor ssh para que acepte conexiones remotas a la cuenta de root.

```
$ nano /etc/ssh/sshd_config
```

Editáramos en este fichero la siguiente línea y reiniciamos:

```
...
# PermitRootLogin without-password
PermitRootLogin yes
...
```

```
$ service ssh restart
```

Ahora configuramos la red para que el nodo SERVER1 tenga IP estática y este en la red interna del clúster (10.0.10.0/24), para ello editamos el siguiente fichero dejando su contenido así:

```
$ nano /etc/network/interfaces
```

Editáremos el fichero dejándolo de la siguiente manera:

```
auto ens3
iface ens3 inet static
      address 10.0.10.110
      netmask 255.255.255.0
      gateway 10.0.10.1
      mtu 9000
      dns-nameservers 10.0.10.1
```

Para que los cambios surjan efecto reiniciamos el servicio de red:

```
$ service networking restart
```

Una vez tenemos la red del nodo SERVER1 configurada nos conectamos en la máquina MASTER, ahora vamos a copiar la clave pública del nodo MASTER al nodo SERVER1 y también nos copiaremos el fichero **hosts**, para ello ejecutaremos en MASTER los siguientes comandos:

```
# Copiamos la clave pública
     $ scp /root/.ssh/id_rsa.pub
     server1:/root/.ssh/authorized_keys
# Copiamos el fichero hosts
     $ scp /etc/hosts server1:/etc
```

Volvemos a la máquina SERVER1 para terminar de instalar los paquetes que nos falta. Primeramente, instalamos el cliente NFS y configuramos para que monte el sistema de archivos nfs en el arranque:

```
# Instalamos cliente NFS
     $ apt-get install nfs-common
# Insertamos línea en /etc/fstab
     $ echo "nas:/nfs /nfs nfs
     auto,rsize=8192,wsize=8192 0 0" >> /etc/fstab
```

Una vez hemos instalado y configurado el cliente NFS ya podemos montarlo manualmente:

```
# Creamos el directorio
     $ mkdir /nfs
# Montamos el directorio
     $ mount /nfs
```

Una vez finalizada la configuración del nodo SERVER1 nos guardamos una imagen de la máquina que luego posteriormente utilizaremos para clonar los otros dos servidores Apache del clúster.

Nos vamos a centrar ahora en el nodo de almacenamiento NFS, otro de los puntos débiles que podemos encontrarnos es que los datos, al tenerlos centralizados en el nodo NFS, se puedan perder con facilidad. Para solucionar este problema hemos decidido instalar un RAID software, los beneficios de su configuración dependiendo del nivel que se escoja son:

- Mayor integridad
- Tolerancia frente a fallos
- Tasas de transferencia
- Capacidad

En nuestro caso para el clúster del TFG hemos decidido instalar un RAID 6, este nos protege los datos tanto si se dañan uno o dos discos, se distribuyen los datos a nivel de bloques y la información de paridad entre todos los discos que comprendan el RAID. Si fallaran más de dos unidades de disco los datos se tendrían que restaurar a partir de la copia de seguridad. En este nivel de RAID necesitamos al menos 4 unidades de disco y como máximo 18, para nuestro nodo de almacenamiento NFS instalaremos 5 unidades de disco.

En primer lugar, vamos a añadir los 5 discos físicos a la máquina virtual NFS para poder realizar posteriormente la configuración del RAID, para ello nos ayudaremos de la herramienta de Ubuntu "Virtual Machine Manager" para añadir los discos a NFS. Teniendo la máquina virtual apagada le instalamos 5 discos de 100MB de capacidad cada uno y los conectamos al controlador SATA.

Una vez instalados los discos, iniciamos la máquina NFS y hacemos login con el usuario root. Comprobaremos que los discos están conectados con el siguiente comando:

```
     $ fdisk -l
```

Comprobados que tenemos los nuevos discos ***/dev/sdb***, ***/dev/sdc***, ***/dev/sdd***, ***/dev/sde*** *y* ***/dev/sdf*** crearemos una partición de cada uno de ellos del tipo "fd" (Linux raid autodetect). Para ello realizaremos los siguientes pasos con cada uno de los discos, se realiza el ejemplo de los comandos a ejecutar con el disco ***/dev/sdb***:

- Ejecutamos el siguiente comando para realizar la partición del disco:

```
$ fdisk /dev/sdb
```

- Nos sale la siguiente línea e introducimos el valor "n" y pulsamos Entrar:

```
...
Command (m for help): n
```

- Nos dice si será partición extendida o partición primaria introduciremos el valor "p" indicando que es primaria.
- Nos vuelve a salir la misma línea de antes y en este caso introduciremos el valor "t" y pulsamos Entrar:

```
...
Command (m for help): t
```

- Seguidamente nos indica la partición seleccionada y nos pide que introduzcamos el valor de código hexadecimal, le introduciremos el valor "fd" (Linux raid autodetect) y pulsamos Entrar:

```
...
Se ha seleccionado la partición 1
Código hexadecimal (escriba L para ver los códigos): fd
```

- Para finalizar nos vuelve a salir la línea indicada inicialmente e introduciremos el valor "w", nos mostrara un mensaje como el siguiente y nos devuelve el *prompt*:

```
...
Command (m for help): w
```

```
# Mensaje que nos muestra

¡Se ha modificado la tabla de particiones!

Llamando a ioctl() para volver a leer la tabla de
particiones.
```

```
Se están sincronizando los discos.
```

Una vez finalizadas hemos obtenido las siguientes particiones de cada disco con las cuales ya podremos montarnos nuestro RAID: ***/dev/sdb1***, ***/dev/sdc1***, ***/dev/sdd1***, ***/dev/sde1*** *y* ***/dev/sdf1.***

Procedemos a instalar los paquetes necesarios para poder montar el RAID software, en nuestro caso utilizaremos la herramienta **mdadm**:

```
$ apt-get install mdadm
```

Usamos la herramienta instalada para montar nuestro dispositivo RAID 6 (**md**) con los 5 discos que hemos añadido a nuestro nodo NFS:

```
$ mdadm --create /dev/md0 --level=6 --raid-devices=5
/dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1 /dev/sdf1
```

Al no haber especificado ninguna opción más en el comando la distribución para los bloques de paridad será por defecto *left-symetric*, para cambiar esta distribución se usa la opción "**`--layout=`**". Tampoco hemos indicado el tamaño de bloque (*stripe unit*) por lo que se quedara el valor por defecto 512KB, para cambiar el tamaño se usa la opción "**`--chunk=`**".

Para comprobar el estado del dispositivo (md) podemos listar el siguiente fichero, hay que tener en cuenta que en este fichero el tamaño de bloque se muestra en bits:

```
$ cat /proc/mdstat
```

Ahora le instalamos a nuestro dispositivo (md0) el sistema de ficheros elegido, en nuestro caso hemos elegido ext4. Seguidamente lo montamos:

```
# Instalamos sistema de ficheros
    $ mkfs.ext4 /dev/md0
# Creamos el directorio
    $ mkdir /mnt/raid
# Montamos el directorio
    $ mount /dev/md0 /mnt/raid
```

Para finalizar hay que registrar el RAID que hemos creado y actualizamos el *initial ramdisk* con los últimos cambios, para ello editaremos el siguiente fichero y actualizaremos mediante los siguientes comandos:

```
# Editamos el fichero
    $ mdadm --detail --scan >> /etc/mdadm/mdadm.conf
# Actualizamos initial ramdisk
    $ update-initramfs -u
```

Para que el RAID se monte de manera automática en el arranque de la máquina NFS modificamos el siguiente fichero añadiendo la siguiente línea:

```
$ nano /etc/fstab
```

```
...
# filesystem    mountpoint    fstype    flags    dump fsck
  /dev/md0      /mnt/raid     ext4     defaults    0    0
```

También exportaremos el directorio donde hemos montado el RAID, para ello modificaremos el siguiente fichero y reiniciamos el servicio para que los cambios surjan efecto:

```
$ nano /etc/exports
```

```
...
/mnt/raid 10.0.10.0/24(rw,no_root_squash,sync,no_subtree_check)
```

```
$ service nfs-kernel-server restart
```

Para que se pueda utilizar el RAID que hemos montado en el nodo NFS en los demás nodos del clúster hay que configurar el acceso, para ello en los otros dos nodos que hasta el momento tenemos instalados, MASTER y SERVER1, realizaremos la siguiente configuración, esta configuración es idéntica para ambos nodos:

```
# Creamos el directorio
    $ mkdir /mnt/raid
# Actualizamos initial ramdisk
    $ mount –t nfs 10.0.10.100:/mnt/raid /mnt/raid
```

También añadiremos una entrada al siguiente fichero para que se monte automáticamente cuando se inician los nodos, para ello realizamos:

```
$ nano /etc/fstab
```

```
...
# filesystem    mountpoint   fstype    flags      dump fsck
nas:/mnt/raid   /mnt/raid    nfs    auto,_netdev   0    0
```

Por último y antes de instalar y configurar las herramientas necesarias para ofrecer alta disponibilidad vamos a instalar en el nodo SERVER1 los paquetes correspondientes a Apache2, se ha elegido este servidor por ser uno de los más utilizados. Para instalar los paquetes: nos logamos en el nodo SERVER1 con el usuario “root” y ejecutamos:

```
$ apt-get install apache2
```

Una vez instalados los paquetes de Apache2 realizaremos las configuraciones pertinentes para ubicar la localización de los contenidos que van a servir todos ellos, esta estará ubicada en el nodo de almacenamiento NFS. Para ello en la máquina SERVER1 modificaremos los siguientes ficheros dejando sus contenidos de la siguiente manera:

```
$ nano /etc/apache2/sites-enabled/000-default.conf
```

```
...
DocumentRoot /nfs/www
...
```

```
$ nano /etc/apache2/apache2.conf
```

```
...
<Directory /nfs/www/>
      Options Includes
      AllowOverride All
      Require all granted
</Directory>
...
```

Ahora crearemos una página web dinámica que ubicaremos en el nodo de almacenamiento NFS. Esta página web nos servirá posteriormente para realizar las pruebas y saber fecha, hora y servidor que nos está sirviendo la página web en cada momento, para ello necesitaremos instalar PHP en nuestra máquina SERVER1. Ejecutaremos:

```
$ apt-get install php libapache2-mod-php php-
mcrypt php-mysql
```

Nos posicionamos en el directorio compartido y crearemos la página web de prueba sencilla:

```
# Nos posicionamos en el directorio
     $ cd /nfs/www
# Creamos y editamos el fichero (página web)
     $ nano index.php
```

El contenido de la página web será el siguiente:

```
<html>
<body>
Página de prueba para TFG Sergio García Lanza
<?php
echo "<br>";
echo "Hoy es " . date ("d/m/Y") . " son las " . date
("h:i:s") "<br>";
echo "Soy el servidor " . $_SERVER['SERVER_ADDR'] . "<br>";
?>
</body>
</html>
```

Finalmente reiniciaremos el servicio de Apache2 para que se carguen los cambios:

```
$ service apache2 restart
```

Ahora que ya tenemos instalado apache y php en el nodo SERVER1, vamos a clonar la máquina para tener tres servidores Apache como indicábamos al inicio del punto 4.3. Para ello debemos tener la máquina SERVER1 apagada y nos ayudamos de la herramienta de Ubuntu "Virtual Machine Manager" para realizar la clonación. Realizamos dos copias exactas de la máquina SERVER1 las cuales se llaman SERVER2 y SERVER3. Los pasos a seguir son:

- **Paso 1**: con la máquina virtual apagada haremos botón derecho del ratón sobre la máquina que queremos clonar.

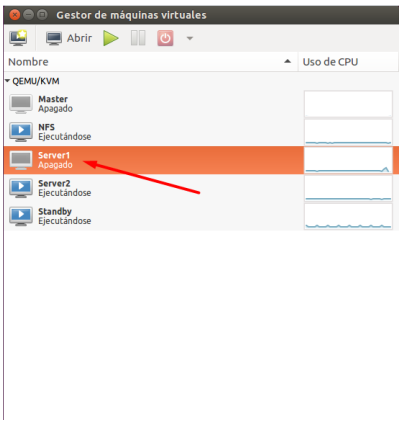

Imagen 15. Pantalla Inicio Virtual Machine Manager

- **Paso 2**: nos despliega un menú en el cual seleccionaremos "clonar":

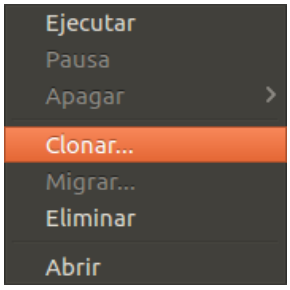

Imagen 16. Menú Desplegable

- **Paso 3**: nos mostrara la pantalla con el detalle de la máquina que vamos a clonar. Le introduciremos el nombre de la nueva máquina, en nuestro caso SERVER2 y SERVER3 y le daremos al botón clonar.

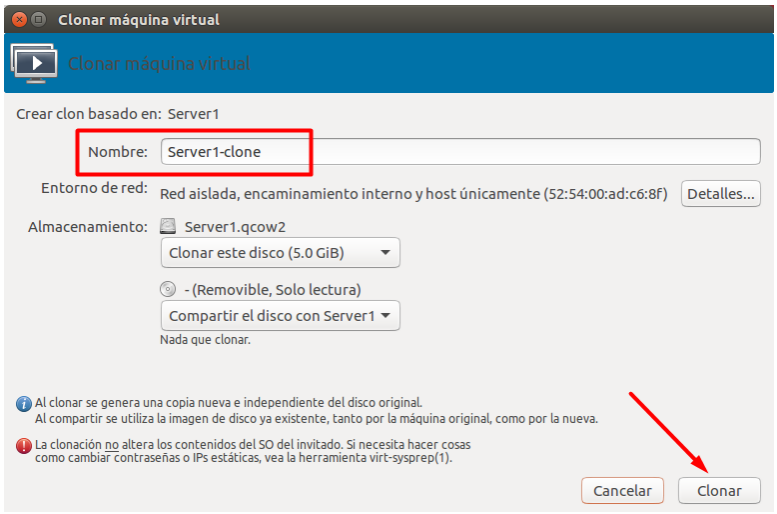

Imagen 17. Pantalla Detalle de la máquina a clonar (SERVER2 y SERVER3)

Una vez tenemos las dos máquinas clonadas deberemos realizar los cambios pertinentes de la interfaz de red y nombre de la máquina, para ello ejecutaremos en las dos máquinas, SERVER2 y SERVER3, con el usuario "root" los siguientes pasos:

- Configurar las interfaces de red:

```
$ nano /etc/network/interfaces
```

Para el nodo SERVER2 el fichero queda de la siguiente manera:

```
auto ens3
iface ens3 inet static
      address 10.0.10.111
      netmask 255.255.255.0
      gateway 10.0.10.1
      mtu 9000
      dns-nameservers 10.0.10.1
```

Para el nodo SERVER3 el fichero queda de la siguiente manera:

```
auto ens3
iface ens3 inet static
      address 10.0.10.112
      netmask 255.255.255.0
      gateway 10.0.10.1
      mtu 9000
      dns-nameservers 10.0.10.1
```

- Configurar el nombre de la máquina (hostname):

```
$ nano /etc/hostname
```

Para el nodo SERVER2:

```
server2
```

Para el nodo SERVER3:

```
server3
```

4.6. Configuración de Alta Disponibilidad

4.6.1. Instalación y configuración de LVS

Una de las mayores preocupaciones en la actualidad es el volumen de tráfico y peticiones que deben manejar las empresas basadas en la web, cada vez es más elevado. Estas empresas están en la obligación de tener entornos que sean fácilmente escalables y que dispongan de una alta disponibilidad.

La solución es tener clusters de servidores interconectados entre ellos y tener un mecanismo de reparto de carga. Por lo tanto, el entorno es altamente escalable ya que, si la carga aumenta, el administrador aumenta el número de nodos que sirven las peticiones y es altamente disponible ya que dispone de varios nodos sirviendo peticiones. Para que el servicio dejase de funcionar tendrían que caer todos los nodos a la vez.

La solución que hemos elegido es instalar un balanceador de carga en el nodo MASTER, que es la máquina accesible por los clientes. Los dos balanceadores de carga de código abierto, condición indispensable en este TFG, más usados son LVS (*Linux Virtual Server*) y *HAproxy*.

Para el presente TFG se ha elegido LVS ya que está implementado en el *kernel* de Linux. LVS implementa un balanceo de carga en el nivel 4 de modelo *OSI*, esto consiste en que las sesiones *TCP* y *UDP* sean repartidas entre los servidores. Su uso es balancear el tráfico *HTTP* y *HTTPS*.

El balanceo de la carga con LVS se puede hacer de tres formas diferentes:

- **NAT**: la máquina principal recibe la petición, esta reescribe los paquetes sustituyendo la dirección IP y lo mando a uno de los servidores WEB de la granja. Esta es también la encargada de servir la respuesta de servidor WEB al cliente.
- **IP Tunneling**: la máquina principal únicamente se hace cargo de las peticiones de los clientes y enviarlas a los servidores. Los servidores WEB son los que después responderán a los clientes.
- **Direct Routing**: prácticamente igual que la anterior, pero sin utilizar *Tunneling*.

Para implementar nuestra solución nos hemos decantado por la opción de LVS-NAT ya que es el modo más fácil de configurar. Para ello usaremos como "director" el nodo MASTER y como servidores web SERVER1, SERVER2 y SERVER3.

Por tanto, las direcciones IP implicadas en la configuración de LVS son:

	Nombre	**Dirección IP**	
Director	MASTER	DIP: 10.0.10.1 (ens4)	VIP: 192.168.1.135 (ens3)
Servidores Reales	SERVER1 SERVER2 SERVER3	RIP: 10.0.10.110 (ens3) 10.0.10.111 (ens3) 10.0.10.112 (ens3)	

En primer lugar, asignaremos una dirección IP estática para la interfaz de red del nodo MASTER que se conecta con el exterior ya que hasta el momento la obtenía por DHCP, para ello nos logamos en el nodo MASTER con el usuario "root" y editaremos el siguiente fichero dejando así su contenido:

```
$ nano /etc/network/interfaces
```

```
auto lo
iface lo inet loopback

auto ens3
iface ens3 inet static
      address 192.168.1.135
      netmask 255.255.255.0
      gateway 192.168.1.1
      dns-nameservers 192.168.1.1
      mtu 9000
```

```
auto ens4
iface ens4 inet static
      address 10.0.10.1
      netmask 255.255.255.0
      mtu 9000
```

Para que los cambios surjan efecto reiniciamos el servicio de red:

```
$ service networking restart
```

Una vez configurada la interfaz de red del nodo MASTER instalamos los paquetes correspondientes al programa que gestiona LVS en el mismo nodo:

```
$ apt-get install ipvsadm
```

Una vez instalado el programa editaremos el fichero de configuración dejando su contenido de la siguiente manera:

```
$ nano /etc/default/ipvsadm
```

```
...
AUTO="true"
...
DAEMON="master"
...
IFACE="ens4"
...
```

Modificando esta líneas indicamos que se carguen las reglas IPVS en el arranque (`AUTO="true"`), que el nodo MASTER lo configuramos como maestro (`DAEMON="master"`) y que la interfaz multicast de IPVS es ens4 (`IFACE="ens4"`).

Seguidamente ejecutaremos:

```
$ dpkg-reconfigure ipvsadm
```

Configuramos LVS añadiendo un servicio virtual para la IP (192.168.1.135) y puerto 80 con método de planificación *Round Robin*.

```
$ ipvsadm -A -t 192.168.1.135:80 -s rr
```

Posteriormente redirigimos los paquetes que se van a recibir en la IP a la dirección de los servidores WEB reales que tenemos dentro de nuestro clúster, para ellos ejecutaremos:

```
$ ipvsadm -a -t 192.168.1.135:80 -r 10.0.10.110:80 -m
$ ipvsadm -a -t 192.168.1.135:80 -r 10.0.10.111:80 -m
```

```
$ ipvsadm -a -t 192.168.1.135:80 -r 10.0.10.112:80 -m
```

Con la opción **–m** estamos indicando que el método de retransmisión de los paquetes es NAT.

Para consolidar esta configuración, es decir, para que las reglas que acabamos de definir se carguen en el arranque al iniciar el sistema debemos guárdalas en el siguiente fichero ejecutando:

```
$ ipvsadm-save > /etc/ipvsadm.rules
```

Por lo tanto, el contenido del fichero quedara:

```
-A -t 192.168.1.135:http -s rr
-a -t 192.168.1.135:http -r server1.cluster:http -m -w 1
-a -t 192.168.1.135:http -r server2.cluster:http -m -w 1
-a -t 192.168.1.135:http -r server3.cluster:http -m -w 1
```

Si queremos que en un futuro la dirección de la interfaz ens3 del nodo MASTER sea dinámica y se le asigne a través de DHCP dejaremos el contenido del fichero anterior de la siguiente manera para no tener problemas de redirección:

```
$ nano /etc/ipvsadm.rules
```

```
-A -t master:http -s rr
-a -t master:http -r server1.cluster:http -m -w 1
-a -t master:http -r server2.cluster:http -m -w 1
-a -t master:http -r server3.cluster:http -m -w 1
```

4.6.2. Instalación y configuración de Pacemaker, Corosync y ldirectord

Llegados a este punto tenemos un servidor web instalado y configurado, pero nos falta, para completar el TFG, que este sea altamente disponible. Si el nodo MASTER dejara de funcionar nuestro servidor web dejaría de estarlo también ya que las peticiones de los clientes no podrían ser servidas.

Para eliminar este punto único de fallo añadiremos redundancia, para ello añadiremos un nuevo nodo (Standby) a nuestro clúster que asumirá las mismas funciones que el nodo MASTER cuando este caído. Utilizaremos una configuración activo/pasivo siendo el nodo MASTER el encargado de servir todas las peticiones de los clientes repartiéndolas entre los servidores del clúster

y solo en caso de que el servicio deje estar operativo en él asumiría esta función en nodo STANDBY.

Para permitir que este nodo STANDBY entre en escena cuando se detecte la caída del nodo MASTER, es necesario la instalación de software para controlarlo automáticamente. El software que hemos elegido para realizar esta funcionalidad es **Corosync** y **Pacemaker** ya que son de código libre, premisa de nuestro TFG, y estos paquetes tienen el apoyo de la comunidad Red Hat y son los únicos que recomiendan para lograr la alta disponibilidad.

- **Pacemaker**: es el encargado de gestionar los recursos del clúster. Para nuestro clúster *Pacemaker* va a gestionar dos direcciones IP virtuales, una para la red interna y otra para la red externa y también gestionará *ldirectord* que es un software que se utiliza para monitorizar todos los nodos del clúster y pasarle esta información a LVS.
- **Corosync**: es el encargado de controlar los nodos que están activos en el clúster. Esto lo consigue enviando periódicamente unos mensajes, se conocen como latidos. Por lo tanto, este software se instalará en los nodos MASTER y STANDBY y se configurará para que avise a *Pacemaker* de la caída de alguno de ellos.

Las direcciones implicadas en nuestro clúster serán las siguientes:

	Nombre	**Dirección IP**	
Director MASTER	MASTER	DIP:192.168.1.135 (ens3) DIP: 10.0.10.1 (ens4)	VIP: 192.168.1.200 VIP: 10.0.10.200
Director STANDBY	STANDBY	DIP:192.168.1.139 (ens3) DIP: 10.0.10.2 (ens4)	VIP: 192.168.1.200 VIP: 10.0.10.200
Servidores Reales	SERVER1 SERVER2 SERVER3	RIP: 10.0.10.110 (ens3) 10.0.10.111 (ens3) 10.0.10.112 (ens3)	

Como observamos en la tabla anterior, utilizaremos una VIP (192.168.1.200) para la red externa, es decir, esta dirección estará asociada a las interfaces ens3 del servidor MASTER o STANDBY, dependiendo de cuál de ellos esté activo en ese momento, y es la dirección a través de la cual los clientes accederán a los servicios web. Por otra parte, al tener configurado LVS-NAT necesitamos del mismo modo un VIP (10.0.10.200) para la red interna ya que los servidores web responden las peticiones de los clientes a través de los nodos MASTER o STANDBY, por lo tanto, esta VIP "interna" se asociará a las interfaces ens4 de ambos nodos.

Una vez explicado cómo conseguiremos la alta disponibilidad, procedemos a la instalación y configuración, primeramente, crearemos y configuraremos el nodo STANDBY. Nos ayudaremos de la herramienta de Ubuntu "Virtual Machine Manager" para clonar la máquina MASTER y así crear la máquina STANDBY que serán idénticas. Los pasos a seguir son:

- **Paso 1**: con la máquina virtual apagada pulsaremos el botón derecho del ratón sobre la máquina que queremos clonar.

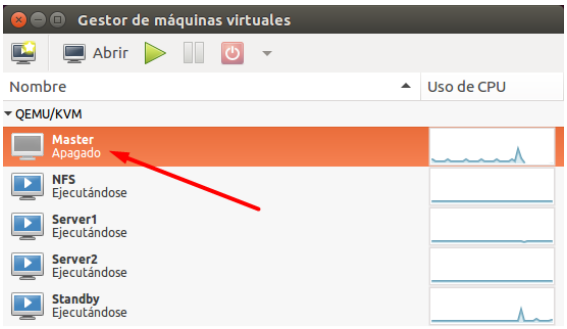

Imagen 18. Pantalla Inicio Virtual Machine Manager

- **Paso 2**: nos despliega un menú en el cual seleccionaremos "clonar":

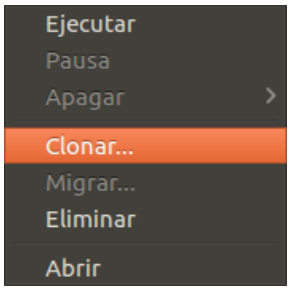

Imagen 19. Menú Desplegable

- **Paso 3**: nos mostrará la pantalla con el detalle de la máquina que vamos a clonar. Le introduciremos el nombre de la nueva máquina, en nuestro caso STANDBY y le daremos al botón clonar.

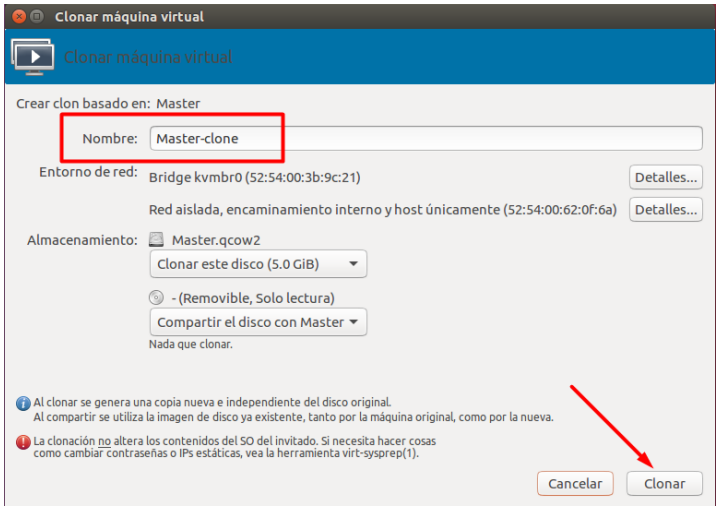

Imagen 20. Pantalla Detalle de la máquina a clonar (STANDBY)

Una vez tenemos el nodo STANDBY, nos logamos en ella con el usuario "root" y cambiaremos las configuraciones de red y nombre de la máquina, para ello realizaremos los siguientes pasos:

- Configurar las interfaces de red:

```
$ nano /etc/network/interfaces
```

```
auto lo
iface lo inet loopback

auto ens3
iface ens3 inet static
      address 192.168.1.139
      netmask 255.255.255.0
      gateway 192.168.1.1
      dns-nameservers 192.168.1.1
      mtu 9000

auto ens4
iface ens4 inet static
      address 10.0.10.2
      netmask 255.255.255.0
      mtu 9000
```

- Configurar el nombre de la máquina (hostname):

```
$ nano /etc/hostname
```

```
standby
```

Una vez hemos terminado de configurar el nodo STANDBY pasamos a la instalación de las herramientas *Pacemaker, Corosync* y *ldirectord* que nos van a brindar la alta disponibilidad de nuestro servidor web. Para ello instalaremos en los dos nodos MASTER y STANDBY los paquetes necesarios para tener estas herramientas. Ejecutamos en ambos nodos los siguiente:

```
$ apt-get install corosync pacemaker ldirectord
```

Después de tener los paquetes instalados en ambos nodos editaremos el siguiente fichero para que *Corosync* se inicie en el arranque. La modificación la hacemos en ambos nodos:

```
$ nano /etc/default/corosync
```

```
...
START=yes
...
```

Ahora generamos las claves de *Corosync*, estas las generamos en el nodo MASTER y posteriormente las copiaremos al nodo STANDBY. Cuando creemos la clave y antes de copiarla al nodo STANDBY nos aseguramos de que el propietario es **root:root** y que los permisos son **400**. A la hora de generar la clave generamos entropía (pulsar teclas al azar) para que haya suficiente y el proceso sea menos costoso:

```
# Generamos la clave
    $ corosync-keygen
# Nos posicionamos en el directorio
    $ cd /etc/corosync
# Comprobamos propietario y permisos (Son correctos)
    $ ls -lrt
# Copiamos la clave a STANDBY
    $ scp authkey standby:/etc/corosync
```

Después de que hayamos generado la clave y que la hayamos copiado al nodo STANDBY, editaremos en ambos nodos el siguiente fichero para indicar la red de nuestros directores y arrancaremos *Corosync*. Una vez arrancado esperaremos unos segundos para que los dos nodos, MASTER y STANDBY, se sincronicen:

```
$ nano /etc/corosync/corosync.conf
```

```
...
bindnetaddr=10.0.10.0
...
```

```
$ service corosync start
```

Llegados a este punto ya tenemos configurado *Corosync* en nuestros nodos directores, MASTER y STANDBY, ahora pasamos a configurar *ldirectord*. Vamos a deshabilitar los scripts de inicio tanto de *ldirectord* como de *ipvsadm*, puesto que no hacen falta. También paramos *ipvsadm* porque ahora será *Pacemaker* quien lanzará a *ldirectord* en el nodo que en ese momento este activo y, a su vez, *ldirectord* lanzará *ipvsadm* con los parámetros adecuados. Estas modificaciones las realizamos en los dos nodos:

```
# Deshabilitamos el script de ldirectord
     $ update-rc.d –f ldirectord remove
# Deshabilitamos el script de ipvsadm
     $ update-rc.d –f ipvsadm remove
# Paramos el servicio de ipvsadm
     $ service ipvsadm stop
```

Vamos a editar el fichero de configuración de *ldirectord*. Para ello descomprimimos una plantilla del fichero y la copiaremos al directorio de configuración. Ejecutaremos estos pasos en ambos nodos:

```
# Vamos al directorio de la plantilla
     $ cd /usr/share/doc/ldirectord/examples
# Descomprimimos el fichero
     $ gzip –d ldirectord.zip
# Copiamos la plantilla al directorio
     $ cp ldirectord /etc/ha.d/ldirectord.cf
```

Editaremos el fichero y su contenido será el siguiente:

```
$ nano /etc/ha.d/ldirectord.cf
```

```
checktimeout=3
checkinterval=10
autoreload=yes
quiescent=yes

virtual=192.168.1.200:80
        real=10.0.10.110:80 masq
        real=10.0.10.111:80 masq
        real=10.0.10.112:80 masq
        fallback=127.0.0.1:80 masq
        service=http
        scheduler=rr
        protocol=tcp
        checktype=connect
```

Voy a comentar brevemente que significa cada línea del fichero que hemos introducido:

- `checktimeout:` si se excede del tiempo de respuesta introducido en segundos (3s), el nodo se declara muerto.
- `checkinterval:` cada x tiempo (indicado en nuestro caso 10s) se produce la monitorización de los servidores.
- `autoreload:` indica que *ldirectord* va a chequear continuamente el fichero de configuración y si su contenido cambia lo actualiza y recarga.
- `quiescent:` indica que si los servidores han caído no se retiran de la tabla de *LVS*, se marcan con peso 0 para que no reciban peticiones.
- `virtual=192.168.1.200:80:` define la VIP y un Puerto.
- `real=10.0.10.110:80 masq:` define los servidores reales y un número de puerto, "masq" indica que están en modo NAT.
- `fallback=127.0.0.1:80 masq:` indica la dirección a la que se reenviaran todas las peticiones si todos los servidores reales están caídos.
- `service:` servicio que se chequea (en nuestro caso HTTP).
- `scheduler:` planificador usado ("rr" es LVS).
- `protocol:` protocolo a usar (en nuestro caso TCP).
- `checktype=connect:` indica que intentará establecer una conexión TCP/IP con el servidor y si pasado el checktimeout no lo consigue lo considerará caído.

En este punto tendremos configurados tanto *Corosync* como *ldirectord*, para terminar de configurar nuestra alta disponibilidad configuraremos *Pacemaker*, para ello iniciamos el servicio en ambos nodos, MASTER y STANDBY:

```
$ service pacemaker start
```

Como ya hemos indicado al inicio de este punto los recursos que va a gestionar *Pacemaker* son: una IP virtual (VIP) para la red externa que será 192.168.1.200 la cual será un alias de la máquina que este activa en cada momento (MASTER o STANDBY) y por la que accederán los clientes, otra IP virtual (VIP) para la red interna que será 10.0.10.200 la cual será un alias de la máquina que este activa en ese momento (MASTER o STANDBY) y por la cual los servidores reales de la granja responderán a los clientes y como último recurso que gestionara *Pacemaker* será el servicio de distribución de carga *ldirectord* que es el encargado de lanzar *LVS* configurando en cada momento para que distribuya la carga a los servidores reales que estén activos.

Para gestionar *Pacemaker* nos ayudaremos de la utilidad "crm" (*Cluster Resource Manager*). Para ello antes de editar el fichero de configuración indicaremos que nuestro editor por defecto será *nano*:

```
$ crm options editor nano
```

Tanto el comando anterior como todos los siguientes los ejecutaremos en el nodo MASTER ya que va a ser el director principal, es decir, siempre funcionara él a no ser que este caído. Una vez hemos definido nuestro editor editaremos el fichero de configuración con el siguiente contenido:

```
$ crm configure edit
```

```
node $id=xxxxxx master
node $id=xxxxxx standby
primitive ip1 ocf:heartbeat:IPaddr2 \
params ip=192.168.1.200 nic:ens3 cidr_netmask=24 \
      broadcast=192.168.1.255
primitive ip2 ocf:heartbeat:IPaddr2 \
params ip=10.0.10.200 nic:ens4 cidr_netmask=24 \
      broadcast=10.0.10.255
primitive ldirectord1 ocf:heartbeat:ldirectord \
      params configfile=/etc/ha.d/ldirectord.cf \
      op monitor interval=15s timeout=20s \
      meta migration-threshold=10 target-role=Started
group group1 ip1 ip2 ldirectord1
order ip_before_lvs inf: ip1:start ip2:start
ldirectord:start
property $id=cib-bootstrap-options \
      dc-version=xxxxxx \
      cluster-infrastructure=corosync \
      stonith-enable=false \
      no-quorum-policy=ignore \
```

Como podemos observar en el contenido del fichero anterior hemos añadido las líneas que están en negrita, con la sentencia *primitive* hemos añadido cada uno de los recursos que debe controla *Pacemaker*, VIP externa, VIP interna y *ldirectord*, con la sentencia *group* lo que hacemos es crear un grupo y meter a los tres recursos para que se ejecuten a la vez en el mismo nodo y con la sentencia *order* establecemos la secuencia de arranque entre ellos. Añadiendo las dos últimas sentencias que hay en el fichero indicamos que no se usa ningún dispositivo de *stonith* y que no se aplica el concepto de *quorum*.

Para terminar con la configuración de nuestro clúster de alta disponibilidad deberemos indicar en los nodos SERVER1, SERVER2, SERVER3 y NFS el nuevo *gateway* y DNS que tienes, para ello modificaremos el fichero `/etc/network/interfaces` de los cuatro nodos y modificaremos las siguientes líneas dejándolas así:

```
...
gateway 10.0.10.200
...
dns-nameservers 10.0.10.200
```

5. Verificación de la solución

Llegados a este punto tenemos instalado y configurado un servidor web en clúster altamente disponible. En esta sección vamos a proceder a realizar las pruebas para verificar que lo que hemos explicado e instalado funciona adecuadamente.

Las pruebas que vamos a realizar nos permitirán verificar tanto el rendimiento de nuestro servidor web como la alta disponibilidad y el balanceo de carga, todas ellas son importantes para confirmar todas las afirmaciones que hemos mencionado durante el proceso de implementación del clúster.

Primeramente, comenzaremos con las pruebas de alta disponibilidad y balanceo de carga, para todas estas pruebas, al ser máquinas virtuales, emularemos la caída de un nodo apagando la máquina virtual, para ello realizamos las siguientes pruebas:

- **Prueba 1:** simulamos que todos los nodos están funcionando correctamente. Realizamos varias peticiones a través de nuestro equipo anfitrión a la dirección VIP externa (http://192.168.1.200/index.php).

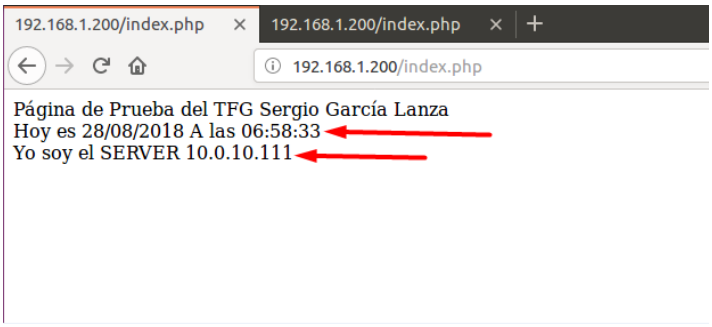

Imagen 21. Captura de Pantalla Firefox Anfitrión con todos los nodos 1/2

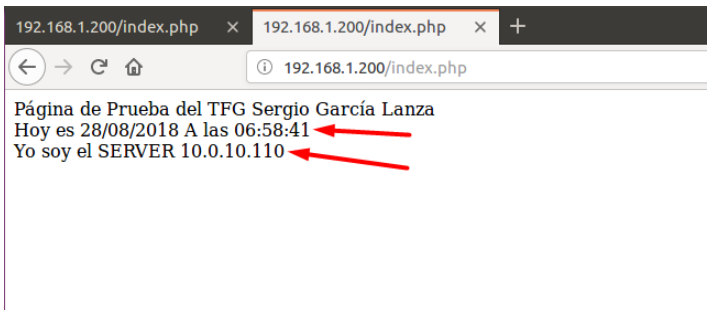

Imagen 22. Captura de Pantalla Firefox Anfitrión con todos los nodos 2/2

Como podemos ver en las imágenes anteriores, podemos verificar que el clúster está funcionando correctamente. También se observa que la primera petición la sirve el servidor con la IP 10.0.10.111 y que la segunda petición la sirve el servidor con la IP 10.0.10.110, por lo tanto, fijándonos en el día y la hora a las que han sido servidas las dos peticiones, verificamos que el balanceo de carga en el nodo MASTER también funciona correctamente.

- **Prueba 2:** simulamos que se ha caído el nodo MASTER. Realizamos varias peticiones a través de nuestro equipo anfitrión a la dirección VIP externa (http://192.168.1.200/index.php).

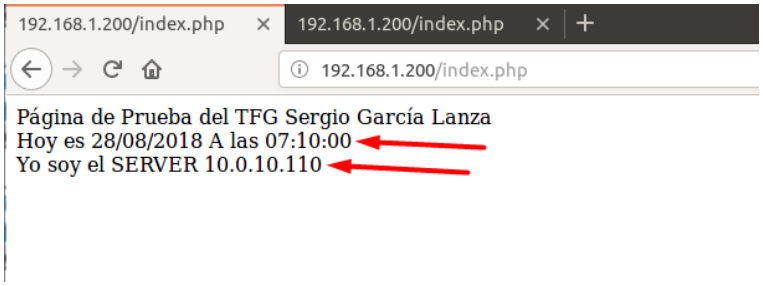

Imagen 23. Captura de Pantalla Firefox Anfitrión con el nodo MASTER caído 1/2

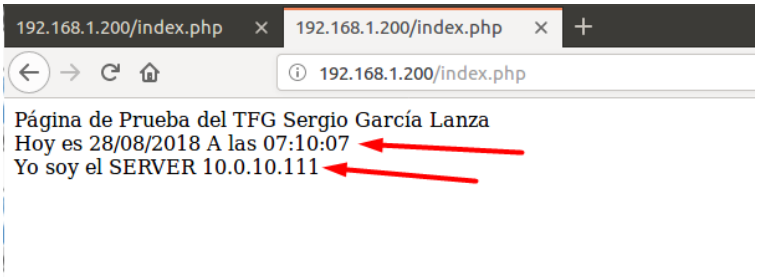

Imagen 24. Captura de Pantalla Firefox Anfitrión con el nodo MASTER caído 2/2

Como podemos ver en las imágenes anteriores y sabiendo que el nodo MASTER está apagado, está caído, el clúster sigue funcionando correctamente por lo tanto verificamos que está funcionando la alta disponibilidad, también se observa que la primera petición nos la sirve el servidor con la IP 10.0.10.111 y que la segunda petición nos la sirve el servidor con la IP 10.0.10.110, por lo tanto, fijándonos en el día y la hora a las que han sido servidas las dos peticiones, verificamos que el balanceo de carga en el nodo STANDBY también funciona correctamente.

- **Prueba 3:** simulamos la caída de los nodos SERVER1 y SERVER3. Realizamos varias peticiones a través de nuestro equipo anfitrión a la dirección VIP externa (http://192.168.1.200/index.php).

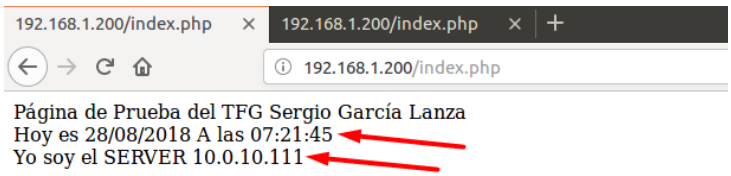

Imagen 25. Captura de Pantalla Firefox Anfitrión con los nodos SERVER1 y SERVER3 caídos 1/2

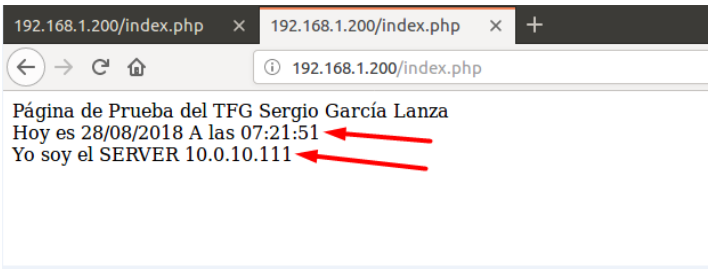

Imagen 26. Captura de Pantalla Firefox Anfitrión con los nodos SERVER1 y SERVER3 caídos 2/2

Como vemos en las imágenes anteriores los nodos SERVER1 y SERVER3 están apagados, y el clúster sigue funcionando correctamente. En este caso al estar solo el nodo SERVER2 en funcionamiento siempre nos sirve la página web él con la IP 10.0.10.11. Podemos verificar que funciona la alta disponibilidad y que el balanceador de carga solo le envía las peticiones al nodo activo, en este caso SERVER2.

- **Prueba 4:** vamos a realizar la prueba más importante en el clúster, esta consiste en comprobar si es capaz de controlar que todos los recursos del mismo estén operativos en todo momento. El grupo de recursos debe levantarse inicialmente y de manera automática en el nodo MASTER si este está operativo, pero si un recurso falla más de tres veces en el nodo, el clúster automáticamente debe mover este recurso al otro nodo director, en nuestro caso es el nodo STANDBY.

  Por lo tanto, para realizar la prueba paramos tres veces el recurso *ldirectord* en el nodo MASTER. Cuando el contador de fallos llegue a tres, significa que hemos parado tres veces el recurso y, por lo tanto, deben levantarse todos los recursos en el nodo STANDBY.

Para realizar la prueba ejecutaremos los siguientes pasos en el nodo MASTER:

```
# Listamos el proceso de ldirectord
     $ ps -ef | grep ldirectord
# Matamos el proceso listado en el comando anterior
     $ kill -9 numero_proceso
```

Una vez realizado el paso anterior tres veces comprobamos, con el siguiente comando sobre el nodo MASTER, que los recursos se han levantado en el nodo STANDBY.

```
# Listamos recursos activos y sobre que nodos
     $ crm_mon
```

Por lo tanto, podemos concluir que la prueba de tolerancia a fallos también a funcionando correctamente.

Debemos destacar que el clúster tiene un SPOF (punto único de fallo), este se trata del servidor de almacenamiento (NFS). Este se podría solucionar replicando el servidor de almacenamiento y montando un RAID 1 por red entre ambos servidores. Esto puede lograrse con el sistema de ficheros *GlusterFS* o con *DRBD*.

Vamos a realizar pruebas para verificar el rendimiento del servidor web. Para ello nos ayudaremos del *benchmark* con el nombre "Apache Benchmark" el cual es un programa corporativo que evalúa servidores web. Nos hemos decantado por este programa debido a que nuestros servidores son Apache2 y por lo tanto al pertenecer a la comunidad Apache pensamos que los resultados serán más favorables.

Este programa simula el envió de un número de peticiones al servidor web que le indicaremos nosotros y obtendremos una métrica sobre el comportamiento que el servidor ha tenido. También le podemos indicar el nivel de concurrencia, así simularemos la presencia de varios usuarios realizando peticiones a la vez.

Las pruebas que vamos a realizar están relacionadas con el número de nodos que estarán activos. Realizaremos una primera medición con un nodo activo y luego con dos para finalmente comparar los resultados.

Para realizar las pruebas comenzamos instalando en el equipo anfitrión, desde el cual se realizarán las peticiones, los paquetes correspondientes al programa *Apache Benchmark*, para ello ejecutaremos siempre con el usuario "root":

```
     $ apt-get install apache2-utils
```

Realizamos la primera medición con un solo nodo activo, es decir tenemos activo el nodo SERVER1, por lo tanto, desde el equipo anfitrión logados con el usuario "root" lanzamos:

```
$ ab -n 100000 -c 10 http://192.168.1.200/index.php
```

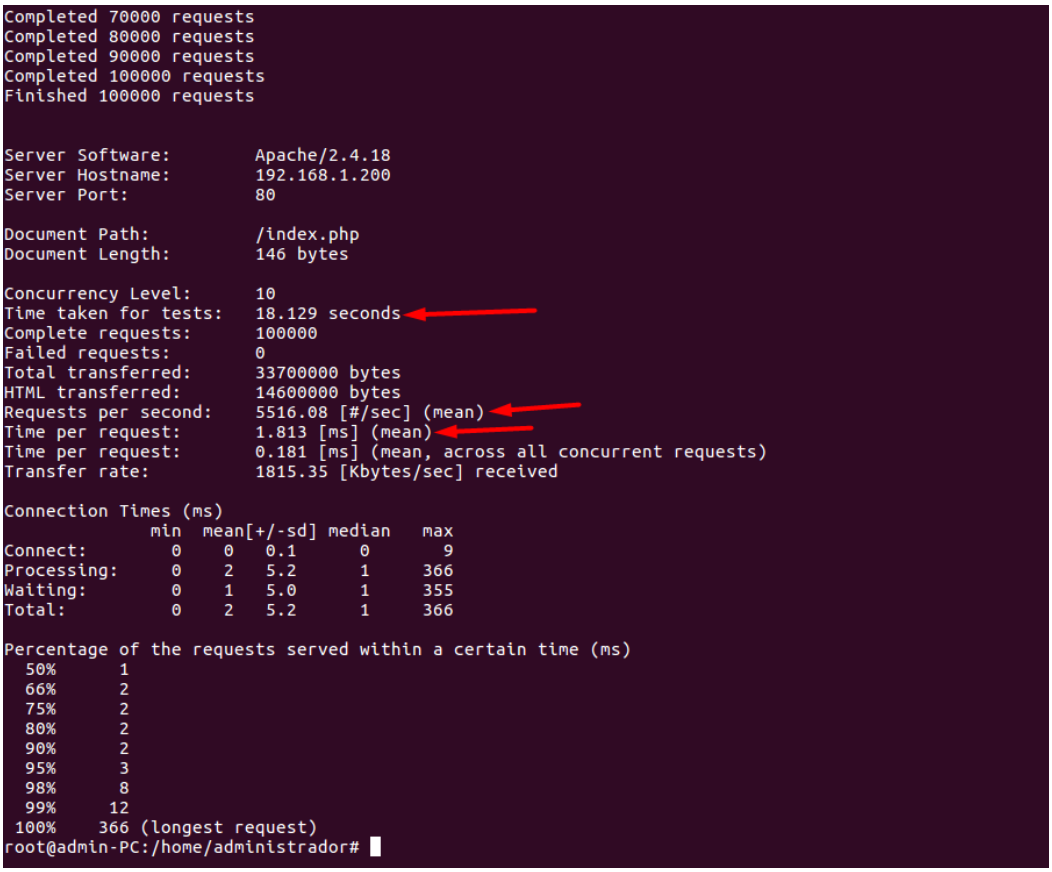

Imagen 27. Captura de Pantalla Línea de Comandos del Anfitrión, ApacheBench y un nodo (SERVER1)

Como podemos observar en la imagen con un solo nodo operativo el servidor web ha tardado 18,2 segundos aproximadamente en servirnos las 100000 peticiones, que el tiempo de respuesta es de 1,8 ms y que la tasa de peticiones atendidas es de 5516 peticiones/s aproximadamente.

Realizamos una segunda medición con dos nodos activos. Están los nodos SERVER1 y SERVER2 en funcionamiento. Desde el equipo anfitrión volvemos a ejecutar el mismo comando:

```
$ ab -n 100000 -c 10 http://192.168.1.200/index.php
```

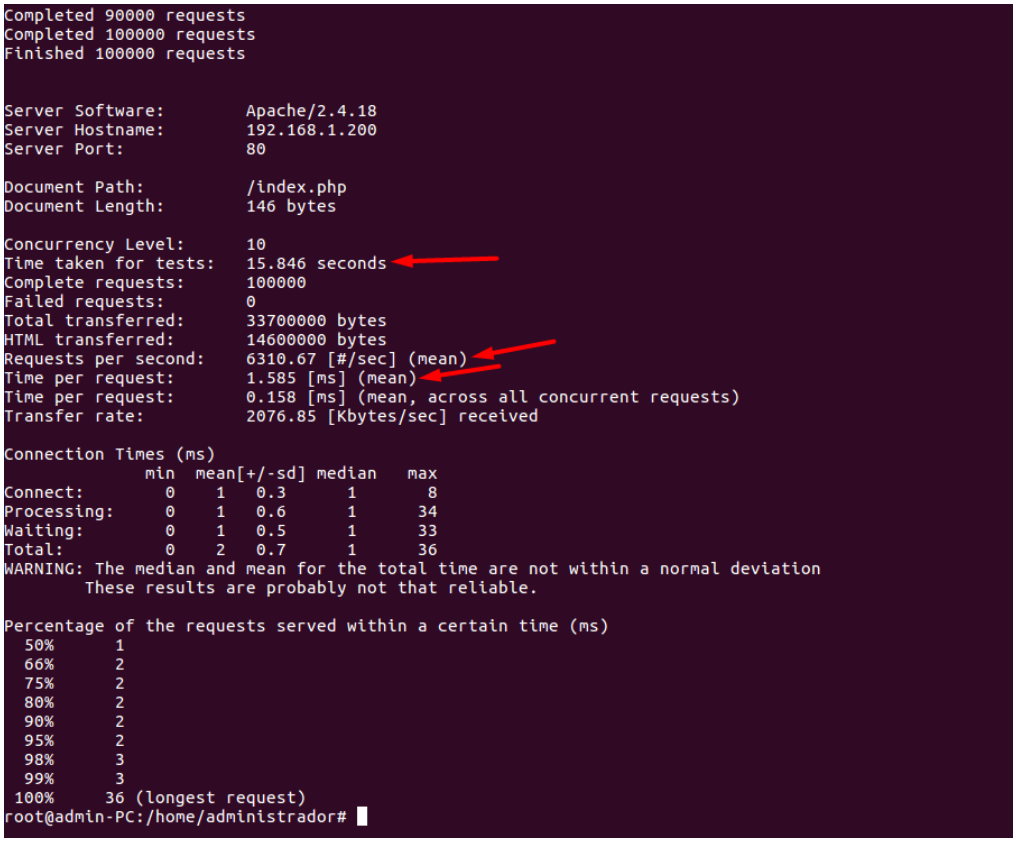

Imagen 28. Captura de Pantalla Línea de Comandos del Anfitrión, ApacheBench y dos nodos (SERVER1 y SERVER2)

Como vemos en esta segunda imagen el servidor web ha tardado 15.85 segundos aproximadamente en servirnos las mismas 100000 peticiones. El tiempo de respuesta en este caso es de 1,58 ms y la tasa de peticiones atendidas es de 6310 peticiones/s aproximadamente.

Nº NODOS	PETICIONES	TIEMPO (segundos)	PETICIONES/s
1	100000	18,2	5516
2	100000	15,85	6310

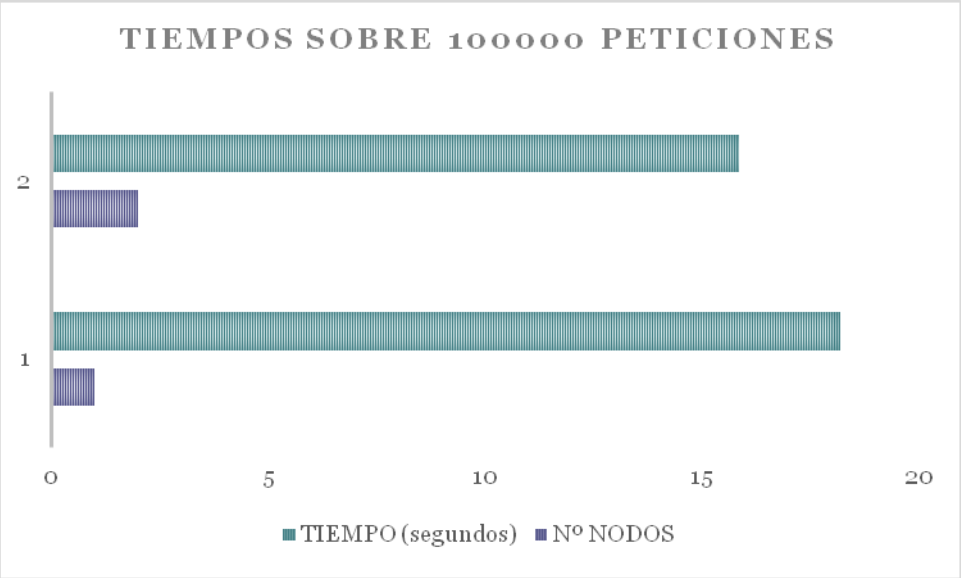

Imagen 29. Gráfico Comparativo (Nº Nodos/Tiempos de Respuesta)

Podemos concluir viendo el grafico de las pruebas realizadas que conforme aumenta el número de nodos, habiendo las mismas peticiones, el tiempo disminuye sustantivamente, por lo tanto, cuantos más nodos haya activos más significativa será la disminución del tiempo.

Para finalizar nuestras pruebas de rendimiento del servidor web vamos a realizar un último test con el programa **Siege**. Este es un programa bajo la licencia de GNU/Linux que funciona diferente al *benchmark* de Apache.

En este caso, con el programa **Siege**, se va a medir la cantidad de peticiones que es capaz de servir nuestro servidor web en un espacio de tiempo determinado. Por lo tanto, nuestras pruebas van a consistir en ver cuántas peticiones sirve nuestro servidor web en 20 segundos cuando hay un nodo activo y cuando hay dos nodos activos.

Comenzamos instalando en nuestro equipo anfitrión el programa, para ello ejecutaremos el siguiente comando:

```
$ apt-get install siege
```

Realizamos la primera medición con un solo nodo activo, es decir tenemos activo el nodo SERVER1, por lo tanto, desde el equipo anfitrión logados con el usuario "root" lanzamos:

```
$ siege -c 10 -t20S -b http://192.168.1.200/index.php
```

Con `-c 10` indicamos que vamos a realizar la prueba con 10 usuarios haciendo peticiones de manera concurrente, con `-t20S` indicamos que la

prueba se realiza durante 20 segundos y con `-b` la dirección sobre la que hacemos las peticiones.

Una vez terminada la prueba los resultados son:

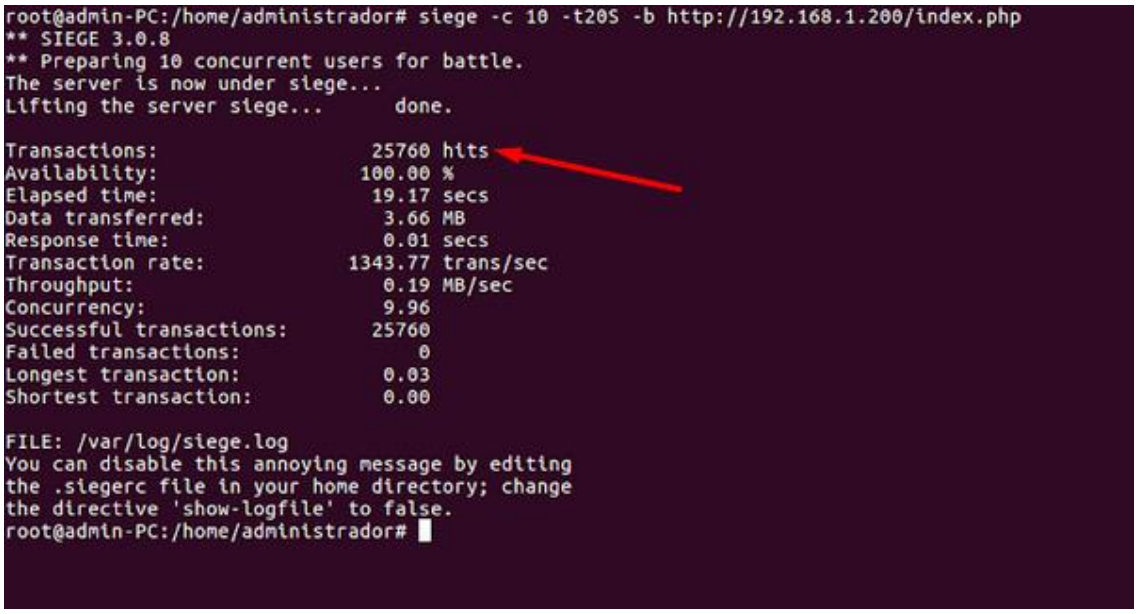

Imagen 30. Captura de Pantalla Línea de Comandos del Anfitrión, Siege y un nodo (SERVER1)

Como podemos observar en la imagen con un nodo activo se han servido 25760 peticiones en 20 segundos.

Ahora pasamos a realizar una segunda medición con dos nodos activos. Están los nodos SERVER1 y SERVER2 en funcionamiento. Desde el equipo anfitrión volvemos a ejecutar el mismo comando:

```
$ siege -c 10 -t20S -b http://192.168.1.200/index.php
```

Una vez terminada la prueba los resultados son:

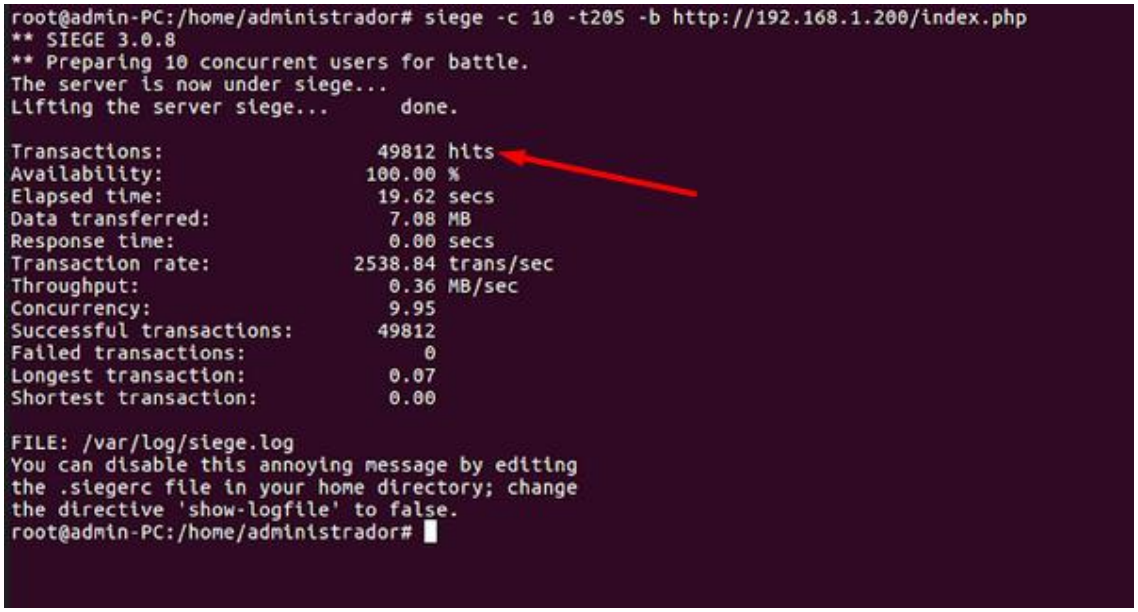

Imagen 31. Captura de Pantalla Línea de Comandos del Anfitrión, Siege y dos nodos (SERVER1 y SERVER2)

Como podemos observar en la imagen con un nodo activo se han servido 49812 peticiones en 20 segundos.

Nº NODOS	PETICIONES	TIEMPO (segundos)
1	25760	20
2	49812	20

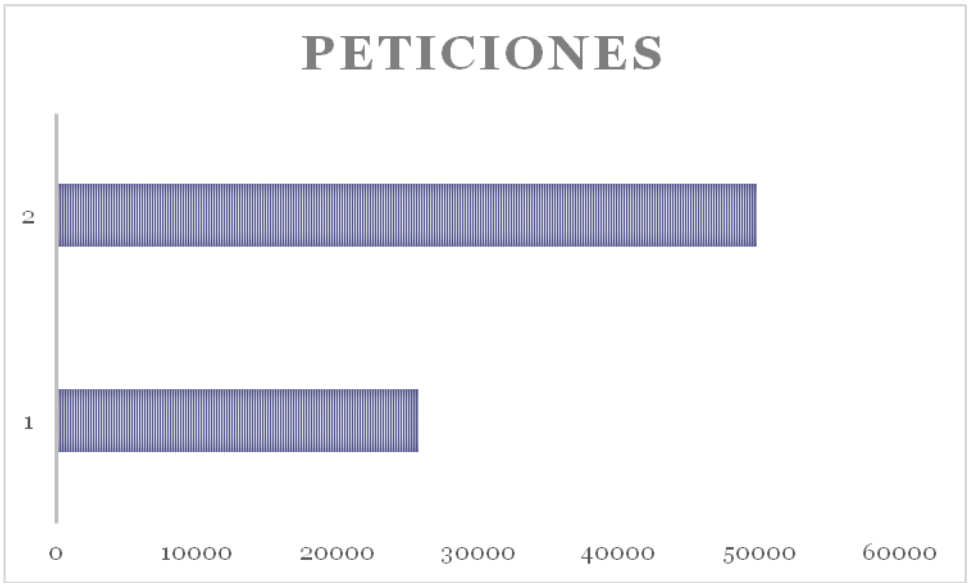

Imagen 32. Gráfico Comparativo (Nº Nodos/Peticiones Resueltas)

Observando el grafico comparativo podemos concluir que cuanto mayor es el número de nodos mayor es el número de peticiones que podrá servir nuestro servidor web

6. Conclusiones

Como podemos observar en este TFG, se puede construir un servidor web mediante un clúster como una buena alternativa coste/prestaciones. En este caso se ha construido un clúster basado en máquinas virtuales con el sistema operativo Ubuntu 16.04 Server.

Se ha conseguido implementar este clúster con las herramientas soportadas por la comunidad Red Hat, todas ellas de código abierto, el cual era otro de los fines de este TFG para abaratar el coste. Se ha probado el sistema simulando varios escenarios provocando fallos intencionadamente y el servidor web ha seguido funcionando correctamente; sirviendo las peticiones de los clientes, por lo tanto, esto nos indica que tenemos un servicio web altamente disponible. También hemos puesto a prueba el rendimiento de nuestro servidor web ayudándonos de la herramienta *Apache Benchmark* y se ha comprobado que los tiempos de respuesta son razonables y que no se cae el servicio.

Como posibles mejoras al proyecto se podría planear de agregar un nuevo nodo de almacenamiento para tener replicados los datos y que en caso de caída del nodo de almacenamiento siguiera sirviendo su réplica. También podríamos añadir una BBDD de alta disponibilidad ya que es prácticamente indispensable para un servidor web.

En general ha sido todo un reto el tener que investigar e implementar y configurar todas las tecnologías utilizadas y ha sido divertido y muy interesante.

7. Bibliografía

[1] Sitio Web Conceptos de Virtualización: http://www.solucionstecnologiques.net/tag/vmware/

[2] Sitio Web de SO Redhat: https://www.redhat.com/es

[3] Sitio Web Sobre Virtualización: http://www.apser.es/blog/2015/12/17/los-principales-tipos-de-virtualizacion/

[4] Sitio Web Vmware: https://www.vmware.com/es/solutions/virtualization.html

[5] Sitio Web: https://www.administracionderedes.com/

[6] Sitio Web Instalación KVM: http://proyectosbeta.net/2016/05/instalar-kvm-en-ubuntu-16-04-lts/

[7] Sitio Web Configuración KVM: https://help.ubuntu.com/community/KVM

[8] Sitio Web Instalación de Corosync y Pacemaker: https://aula128.wordpress.com/2015/02/28/alta-disponibilidad-como-configurar-un-cluster-ha-linux-con-corosync-y-pacemaker-con-recurso-apache2/

[9] Sitio Web Configuración de un RAID: https://www.uatek.com/proyecto/2009/03/15/configuracion-de-volumen-raid-por-software-en-linux/

[10] Sitio Web Instalación de PHP: https://www.digitalocean.com/community/tutorials/como-instalar-linux-apache-mysql-php-lamp-en-ubuntu-16-04-es

[11] Sitio Web Configuración ldirectord: http://manpages.ubuntu.com/manpages/xenial/man8/ldirectord.8.html

[12] Sitio Web Apache Benchmark: http://manpages.ubuntu.com/manpages/xenial/man1/ab.1.html

[13] Sitio Web Apache Benchmark: https://blog.desdelinux.net/apache-benchmark-gnuplot-medir-rendimiento-de-servidor-web/