UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Entorno de virtualización de bajo coste y alta disponibilidad utilizando tecnología Xen

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Enrique Parra Carrascosa
Tutor: Floreal Acebrón Linuesa

Curso 2019-2020

Resum

La virtualització es un camp d'estudi que a dia d'avuí gaudeix de molta fama en les infraestructures de TI. Això és tant a causa de l'estalvi de costos que permet la seua utilització com a la reducció de temps que han experimentat els desplegaments d'infraestructura. Aquests fets fan que el seu estudi i implantació continuen avançant i es presenten cada vegada més possibilitats que cobrisquen segments diferents d'usuaris amb necessitats diferents.

Dit això, aquest projecte proposarà una nova plataforma de virtualització de baix cost dirigida a un segment d'usuaris que no puguen o desitgen invertir en l'adquisició d'altres productes comercials. Aquesta plataforma de virtualització proporcionarà característiques d'alta disponibilitat sobre els sistemes virtuals i la pila de programari utilitzada estarà formada íntegrament per aplicacions de codi obert.

Paraules clau: xen, gluster, virtualització, alta disponibilitat, Linux

Resumen

La virtualización es un campo de estudio que a día de hoy goza de mucha fama en las insfraestructuras de TI. Esto es tanto debido al ahorro de costes que permite su utilización como a la reducción de tiempo que han experimentado los despliegues de infraestructura. Estos hechos hacen que su estudio e implantación sigan avanzando y se presenten cada vez más posibilidades que cubran segmentos diferentes de usuarios con necesidades diferentes.

Dicho esto, este proyecto va a proponer una nueva plataforma de virtualización de bajo coste dirigida a un segmento de usuarios que no puedan o deseen invertir en la adquisición de otros productos comerciales. Esta plataforma de virtualización proporcionará características de alta disponibilidad sobre los sistemas virtuales y la pila de software utilizada estará formada en su totalidad por aplicaciones de código abierto.

Palabras clave: xen, gluster, virtualización, alta disponibilidad, Linux

Abstract

Virtualization is a field of study that is now very well known in IT infrastructures. This is due both to the cost savings that its use allows and to the reduction in time that infrastructure deployments have experienced. These facts keep your study and implementation moving forward and present more and more possibilities that cover different segments of users with different needs.

That said, this project will propose a new low-cost virtualization platform aimed at a segment of users who are unable or unwilling to invest in the acquisition of other commercial products. This virtualization platform provides high availability features on the virtual systems and the software stack used is made up entirely of open source applications.

Key words: xen, gluster, virtualization, high availability, Linux

Índice general

Índice de figuras

CAPÍTULO 1
Glosario

CPU: la unidad central de procesamiento, o CPU de sus siglas en inglés (Central Processing Unit), es el componente de un ordenador que interpreta las instrucciones.

Anfitrión: en el contexto de virtualización de servidores, este el es nombre que recibe el sistema que se encarga de administrar los recursos físicos y gestionarlos entre los diferentes sistemas invitados.

Invitado o huésped: en el contexto de virtualización de servidores, este es el nombre que recibe el sistema virtualizado que se ejecuta sobrer un sistema anfitrión.

BIOS: del inglés *Basic Input Output System*, es un software que localiza y reconoce los dispositivos necesarios para empezar con el arranque del sistema operativo.

IaaS: del inglés *Infraestructure as a Service*, denomina un modelo de negocio por el cual un proveedor proporciona acceso a servicios de infraestructura.

PaaS: del inglés *Platform as a Service*, denomina un modelo de negocio por el cual un proveedor proporciona acceso a servicios que serán utilizados por aplicaciones.

SaaS: del inglés *Software as a Service*, denomina un modelo de negocio por el cual un proveedor proporciona acceso a aplicaciones.

TI: Abrr. Tecnologías de la Información. Es la aplicación de computadores y equipos de telecomunicaciones al procesamiento de datos.

Bonding: tecnología propia de los sistemas UNIX que permite configurar redundancia en los interfaces de red.

IP: las siglas se refieren a *Internet Protocol*. Protocolo de comunicación de datos clasificado, según el modelo OSI, en la capa de red. La mayoría de veces este término también se refiere a dirección IP, que es un número que identifica la interfaz de un dispositivo que conoce el protocolo IP.

UPS: del inglés, *Uninterruptible Power Supply*, se refiere a un sistema de alimentación que gracias a sus abaterias es capaz de alimentar elementos eléctricos durante un tiempo limitado.

Blade: computadora especialmente diseãda para ahorrar espacio en un centro de datos.

SAN: del inglés *Storage Area Network*. Red especialmente dedicada a servicios de almacenamiento.

Código abierto: también conocido como software libre, se refiere a un modelo de desarrollo de software basado en la colaboración abierta. Se caracteriza por facilitar el acceso al código fuente de los progamas.

CAPÍTULO 2
Introducción

Vivimos en un mundo en el que la competitividad empresarial es cada vez mayor, por lo que las empresas buscan reducir costes con la intención de maximizar los beneficios y generalmente una de las partidas de costes que más se busca reducir es la inversión en TI.

Por ello, en la actualidad son muy populares las tecnologías de virtualización, cuyo empleo permite reducir el número de sistemas físicos dentro de un parque de servidores, así como sus costes asociados.

La virtualización en la actualidad tiene un papel relevante dentro de las infraestructuras de TI. Cada vez son más populares los paradigmas como IaaS, PaaS, etc. los cuáles tienen su origen a partir del concepto de virtualización. Por lo tanto, se puede decir que la popularización de la virtualización ha revolucionado por completo el mundo de las TI.

Este trabajo aportará una solución más al gran abanico de posibilidades que existen en el mundo de la virtualización, enfocándose en poder ofrecer al potencial usuario de la solución una alternativa con el mínimo coste posible en el mercado sin renunciar a las prestaciones básicas que otras plataformas pueden proporcionar por un coste mayor.

2.1 Motivación

Como se ha mencionado anteriormente, la virtualización es una tecnología ampliamente utilizada en los sistemas actuales y que seguirá utilizándose durante un largo tiempo. Así pues, es un campo dentro de la tecnología que nos despierta un profundo interés y sobre el que deseamos estudiar en profundidad, causa principal por la que se ha elegido este trabajo.

Dentro de la virtualización existen varias soluciones comerciales ampliamente extendidas pero con un coste asociado a su adquisición que no siempre puede ser asumido por algunos segmentos de usuarios, como por ejemplo pequeñas empresas que acaban de fundar su negocio y no pueden realizar una inversión muy grande en el departamento de TI. Este segmento de usuarios puede utilizar tecnologías de virtualización con un menor coste de adquisición, como es el caso de la tecnología Xen.

Además de la posibilidad de virtualización, con el ahorro de costes intrínseco a la utilización de dicha tecnología, se propone dotar de tolerancia a fallos la infraestructura sobre la que se ejecutan las aplicaciones, reduciendo así los tiempos de pérdida de servicio de las diversas aplicaciones que se ejecuten sobre la infraestructura.

Uniendo las dos características mencionadas, junto con la utilización de software libre (otra cultura que nos despierta un gran interés) pueden dar lugar a crear una solución

robusta, respaldada por una comunidad cada vez más numerosa y que a su vez pueda ser tomada como base de otros proyectos y mejorada.

2.2 Objetivos

Los objetivos a conseguir mediante la realización del presente trabajo son los que figuran a continuación:

1. El primer objetivo es presentar una solución software completa de virtualización de sistemas operativos vuyo valor de adquisición sea el mínimo posible para que sea una solución competitiva. Por esta razón de coste se utilizarán tecnologías de código abierto.
2. El segundo objetivo es ofrecer alta disponibilida en la capa de virtualización de sistemas operativos, ya que esto impactará positivamente en las aplicaciones que se ejecuten reduciendo los tiempos de indisponibilidad del servicio.
3. Además la solución ofrecerá una operativa sencilla a los usuarios con la intención de minimizar los costes de administración de la infraestructura.
4. Por último, adquirir conocimientos sobre tecnologías de virtualización que cada vez están más extendidas en el mundo empresarial y que son aptitudes necesarias para los nuevos administradores y arquitectos de sistemas.

2.3 Estructura de la memoria

La memoria se ha estructurado en base a un enfoque genérico de la información y posteriormente a un enfoque mas específico. Así pues, en el capítulo 3 se plantea el estado del arte en el entorno de virtualización de servidores, haciendo referencia primeramente a la historia de la virtualización y a continuación, se presentan las tecnologías que mejor aplican al trabajo expuesto en este documento.

En el capítulo 4 se realiza un análisis del problema descomponiéndolo en partes y explicando las características que debe proporcionar el producto final para cumplir con los objetivos.

En el capítulo 5 se diseña la solución aportada teniendo como base los criterios expuestos en el capítulo anterior. En el contenido de este capítulo también se describen la diferentes tecnologías utilizadas para obtener el producto final.

En los capítulos 6, 7 y 8 se presentan varios aspectos de la implementación de la solución aportada en este trabajo. Podemos decir que en el capítulo 6 se expone con detalle la implementación realizada del producto para que el lector sea capaz de replicarlo. Además, en el capítulo 7 se proporciona una serie de tests realizados sobre la infraestructura para validar su correcto funcionamiento. En el capítulo 8 se describen algunas operaciones de administración a realizar sobre el producto para su uso diario.

Para finalizar, en el capítulo 9 se proporcionan las diferentes conclusiones a las que se ha llegado después de realizar este trabajo y una serie de ampliaciones futuras y mejoras a realizar sore el producto presentado.

CAPÍTULO 3
Estado del arte

La virtualizacón es una tecnología que permite la abstracción, a través de software, de un recurso tecnológico, ya sea de tipo físico o lógico. Por lo tanto, se puede virtualizar un disco, una tarjeta de red, un sistema operativo, e incluso un servidor completo.

Respecto a su objetivo, se pueden distinguir varios tipos de virtualización:

- Virtualización de servidores: se refiere a la virtualización que permite la ejecución de varios sistemas operativos sobre un servidor. Esta categoría será mejor desarrollada más adelante en el presente documento, ya que es el tipo de virtualización que se utiliza en este trabajo.
- Virtualización de escritorio: se refiere a la virtualización que permite la ejecución de varios sistemas operativos de escritorio sobre un mismo ordenador, pudiendo ser este un PC de escritorio o un servidor. Dentro de este tipo de virtualizacón se podría distinguir si los escritorios que se ejecutan son locales o si están alojados en un servidor central (VDI).
- Virtualización de red: la virtualización de red utiliza software para crear una capa de abstracción sobre los dispositivos de red, tales como enrutadores i conmutadores. Esta técnica permite simplificar su trabajo a los administradores de red. (por ejemplo VMWare NSX)
- Virtualización de almacenamiento: la virtualización de almacenamiento permite el acceso a todos los dispositivos de almacenamiento que forman parte de una red como si fueran un único elemento. Esto permite facilitar el aprovisionamiento de almacenamiento a las máquinas virtuales y un uso más eficiente de todos los recursos disponibles en la red.
- Virtualización de datos: consiste en crear una capa software entre las aplicaciones que acceden a los datos y el almacenamiento que lo contiene de tal forma que las aplicaciones sean capaces de acceder a los datos sin importar el formato, el origen o la localización donde estén almacenados.
- Virtualización de aplicaciones: se ejecuta software sin instalarlo directamente sobre un sistema operativo.
- Virtualización de centros de datos: permite abstraer la mayor parte del hardware que forma parte de un centro de procesamiento de datos, proporcionando al administrador la facultad de dividir un único centro físico en varios lógicos. Esta división puede utilizarse para dar servicios totalmente aislados a múltiples clientes, dando lugar a un concepto conocido com infraestructura como servicio (IaaS de sus siglas en inglés).

- Virtualización de CPU: este tipo de virtualización es la base tecnológica fundamental que hace posible los hipervisores y las máquinas virtuales posibles. En un principio esta virtualizacón se realizaba por software, pero a día de hoy los procesadores incluyen instrucciones que permiten una gran mejora de rendimiento en la virtualización.

3.1 Historia de la virtualización

El concepto de virtualización surgió en los años 50, época en la que los computadores, conocidos como *mainframes*, tenían el tamaño similar al de una habitación. Cabe destacar su elevado coste de adquisición, por lo que múltiples usuarios tenían acceso a un mismo computador a través de terminales "tontas". Esta situación forzaba a los usuarios a compartir los mismos dispositivos de almacenamiento y de CPU, con lo que los recursos se aprovechaban mejor, aumentando así el retorno de la inversión realizada.

Casi dos décadas después, en el anõ 1967, IBM desarrolló el primer hipervisor y en 1968 una segunda versión de éste con la capacidad de compartir la memoria entre máquinas virtuales, proporcionando su propio espacio de memoria a cada usuario. No fue hasta el anõ 1970 cuando se lanzó al mercado un sistema operativo llamado "VM", que permitía a los administradores de los *mainframe* modelo System/370 utilizar varios sistemas virtuales en un mismo nodo físico. Este sistema operativo llevó el concepto de los años 50 de compartir los recursos de un único computador a otro nivel, permitiendo convivir varios entornos de computación distintos en un único entorno físico.

Aunque se han realizado numerosas mejoras, la tecnología de la década de los 70 sigue utilizándose actualmente en los *mainframe* IBM Z. En Octubre del año 2000, coincidiendo con las presentación de la nueva arquitectura Z de 64 bits, el producto se renombró a z/VM. Actualmente está disponible la versión 7.1 de z/VM, tal como se puede ver en las últimas noticias proporcionadas por IBM [1].

A finales de la década de los noventa, en el año 1998, nace la empresa VMware, Inc.[33]. Esta empresa nace con la intención de mejorar los sistemas informáticos y de este modo, no tardó en lanzar al mercado la primera versión de Workstation, su solución de virtualización para escritorios. Este hecho fue un verdadero éxito, ya que permitía a un desarrollador ejecutar varias máquinas virtuales sobre un mismo PC. Cabe destacar que la solución de virtualización de WMware, Inc. se desarrolla para utilizarse sobre computadores basados en la arquitectura x86[28], lo que es una arquitectura totalmente diferente a la virtualización que hasta esta fecha utilizaba IBM en sus *mainframe*.

Posteriormente, en el año 2000, aparece la característica de jaulas en FreeBSD 4.0[7]. Este nuevo tipo de virtualización se basa en el concepto de aislar procesos cambiando su directorio raíz en el sistema de ficheros, permitiendo aislar un proceso en un espacio de directorios independiente proporcionando seguridad entre procesos. Las jaulas son la base de lo que actualmente se llaman contenedores.

Más tarde, en el año 2002, VMware, Inc. lanza su primer hipervisor, ESX Server 1.5, que permite consolidar servidores obre un menor número de dispositivos físicos, optimizando así el rendimiento de dichos sistemas y la administración de TI. La aparición de este producto, sumado a la mejora de las prestaciones de los procesadores en esta misma época, como por ejemplo la aparición de la tecnología *Hyper-Threading* de Intel[10], supuso la popularización del uso de tecnologías de virtualización a todos los niveles. Finalmente, en diciembre del año 2003 salió al mercado VMware Virtual Center 1.0[34],

[1] La URL de consulta es https://www.vm.ibm.com/news/

que incluía la capacidad de VMotion. Esto es la posibilidad de migrar sistemas virtuales entre diferentes servidores físicos sin que este se vea afectado.

Coincidiendo en el tiempo con la solución de VMware, se desarrollaba en los laboratorios de la Universidad de Cambridge un proyecto llamado "Xenoserver"[20], cuyo objetivo era crear una infraestructura pública para computación distribuida dentro de una amplia área. Una de las ramas de dicho proyecto era la creación de un hipervisor que albergara varios sistemas operativos sobre un servidor que utilizara la arquitectura x86, y cuyo nombre fue *Xen virtual machine monitor*. La presentación del nuevo hipervisor[3] se realizó durante el SOSP 2003[2]. Dicho hipervisor fue liberado bajo los términos de la licencia GPL[3]

También en 2003 se anunció por primera vez en lanzamiento público del emulador QEMU[24]. Este software se liberó bajo licencia GPL y permitía la virtualización tanto de sistemas dentro de un escritorio como la emulación de otras arquitecturas diferentes a la del sistema anfitrión.

Este nuevo hipervisor es capaz de utilizar una técnica llamada *paravirtualización* cuyo objetivo es el de reducir el sobrecoste asociado a la traducción de algunas operaciones realizadas por el sistema invitado, permitiendo optimizar al máximo los recursos del sistema anfitrión. La contrapartida es la necesidad de modificación del sistema operativo invitado para poder soportar este método de virtualización, mayoritariamente a través de manejadores de dispositivos, ya que los dispositivos físicos ofrecidos al invitado por parte del hipervisor no son los originales a causa de la abstracción que se realiza.

Poco después de la aparición de los hipervisores para arquitectura x86 ya mencionados, y más concretamente en el año 2005, intel desarrolla una extensión sobre el juego de instrucciones de la arquitectura x86. Dicha extensión se conoce por el nombre clave *Vanderpool* en su origen, y después se renombra como Intel VT. Estas nuevas instrucciones específicas de virtualización permiten optimizar el rendimiento de los sistemas de virtualización que las utilicen y aumentar las seguridad entre las máquinas virtuales. Más tarde AMD también lanzará una extensión en sus procesadores que recibirá el nombre de AMD-V.

Este mismo año, en el mes de marzo, aparece la primera versión del sistema operativo Solaris 10 [26] para arquitecturas x86 y SPARC. Dicha versión incluye la capacidad de utilizar un sistema de virtualización llamado *Solaris Zones*[4]. Esta tecnología permite aislar entornos de ejecución compartiendo una única instancia del sistema operativo, lo que permite optimizar los recursos del sistema al no proporcionar una capa de abstracción como sí hacen otros tipos de virtualización. También se incluye un mecanismo de control de recursos del sistema que impide la interferencia en el uso de estos entre los diferentes entornos de ejecución.

En el año 2006 se anunció por primera vez KVM[12], tecnología que permite al kernel de linux actuar como un hipervisor, ejecutando cada una de las máquinas virtuales como un proceso regular de Linux. Cabe destacar que es una tecnología que poco después obtendrá mucha relevancia debido a la adquisición por parte de la compañia Red Hat[22] , y su inclusión un producto que tomará el nombre de Red Hat Virtualization.

También 2006 marcó el inicio de una gran revolución: Amazon Web Services comenzó a trabajar con la idea novedosa de proporcionar servicios de infraestructura de TI en forma de servicios web[2]. Amazon Web Services fue el origen de lo que a día de hoy se conoce como la nube, y que es el máximo exponente en cuanto a virtualización se refiere.

[2] 19th ACM Symposium on Operating Systems Principles, October 19-22, 2003
[3] GNU General Public License, *http://www.gnu.org/licenses/gpl.html*
[4] Anteriormente ya se había uncluído en versiones Beta y para desarrolladores

La gran innovación de este servicio fue la idea de vender directamente los sistemas ya virtualizados.

Un año después llega otro hito importante en el mundo de la virtualizacón de escritorio: el lanzamiento de la primera versión de Virtualbox[31]. Virtualbox es un software de virtualización de escritorio que nació como un producto comercial que después fue liberado bajo licencia GPL. Su popularización llegó por ser un software de virtualización que ofrece un buen rendimiento y que ofrece características parecidas a la solución VMware Workstation.

Otros hipervisores que vieron su nacimiento durante la primera década del tercer milenio son Logical Domain (LDoms)[5][35] para arquitectura SPARC en el año 2007 y Hyper-V, el hipervisor lanzado por Microsoft, en el 2008[8].

Actualmente está cambiando el paradigma de la virtualización y se está llevando al máximo exponente: la gestión de cualquier recurso físico disponible se delega en un orquestador que decidirá qué sistemas virtualizar, sobre qué sistemas y qué recursos se le proporcionarán. Se pueden observar dos ejemplos claros de este nuevo paradigma: Openshift y Openstack.

Además de lo comentado, la virtualización únicamente no se ciñe a virtualizar sistemas operativos sobre hardware: los sistemas base que se utilizan para configurar infraestructuras de nube pueden ser adquiridos a otras compañias que ya proporcionan un sistema virtual, como es el caso de Windows Azure o Amazon AWS.

3.2 Virtualización de servidores

Actualmente la virtualización de servidores es el tipo de virtualización más extendido, ya que ha sido una tecnología muy bien adoptada en el entorno empresarial por las siguientes razones:

1. Reducción de capital: la virtualización hace posible ejecutar varios servidores virtuales sobre un mismo servidor físico, lo que disminuye el número de servidores físicos necesarios para la ejecución de los diferentes servicios, reduciendo el capital a invertir en hardware.

2. Reducción de costes operativos: al menguar el número de servidores físicos también se reducen los costes asociados a la gestión de estos, como por ejemplo los costes de cableado, suministro eléctrico, refrigeración, etc.

3. Simplificación de la gestión del parque informático: al disminuir el número de servidores físicos también disminuye la complejidad de gestión para los administradores de sistema.

4. Reducción en el tiempo de despliegue de nuevos servicios: uno de los factores que intervienen en el tiempo de implantación de nuevos servicios es la compra y provisionamiento de nuevo hardware. Si el hardware existe es un tiempo que ya no es necesario aguardar.

5. Optimización de recursos: en un ecosistema de servidores físicos cada aplicación requiere uno para su funcionamiento, y todos los recursos que no utilice se quedan desaprovechados. En un ecosistema de virtualización los recursos no utilizados por una aplicación pueden ser utilizados por otra, con lo que se optimiza la utilización de dichos recursos.

[5] Más tarde será renombrado como Oracle VM for SPARC

6. Minimización de pérdida de servicio: la virtualización facilita aumentar el número de servidores, con lo que se pueden implementar servidores redundantes para cada uno de los servicios con un menor coste.
7. Solución ecológica: optimizando los recursos disponibles y reduciendo el número de servidores físicos se reducen también el consumo eléctrico tanto en hardware como en refrigeración, generando así menores emisiones de CO2.

Dentro de la virtualización de servidores se pueden distinguir varios modos, según la técnica que utilicen:

- Virtualización completa o nativa: el software de virtualización ofrece una interfaz completa emulada de hardware al sistema invitado, permitiendo que dicho sistema pueda ejecutarse sin ninguna modificación. Este tipo de virtualización permite ejecutar cualquier tipo de sistema operativo sobre el huésped.
- *Paravirtualización*: este tipo de virtualización se basa en la virtualización nativa, con la diferencia que no emula la interfaz hardware sino que los sistemas invitados interactúan directamente con el hardware. Esto es posible gracias a las extensiones de virtualización implementadas en los procesadores. Para poder utilizar este tipo de virtualización los sistemas invitados deben ser modificados, por lo que deben estar preparados para esta circunstancia.
- Virtualización software o emulación: la emulación del hardware se realiza a nivel de sistema operativo, permitiendo que hayan diferentes instancias del mismo sistema operativo. Este tipo de virtualización es la más óptima, ya que no debe realizar traducciones en el acceso al hardware, pero también es la menos flexible, ya que en los sistemas virtualizados solo se permite el mismo sistema operativo que el anfitrión.

3.3 Hipervisores

Como se ha mencionado anteriormente, el concepto de hipervisor nació con la capacidad de virtualización de los primeros *mainframe* de IBM que hacían uso de esta tecnología.

Los diferentes sistemas virtualizados intentan acceder a los mismos recursos hardware, por lo que se necesita un mecanismo que administre dichos recursos, y esa es la función principal del hipervisor: administrar el acceso a los recursos de cada uno de los sistemas virtuales que intenta acceder a ellos.

Según el tipo de virtualización que llevan a cabo se pueden distinguir dos tipos de hipervisores, los que se ejecutan directamente sobre el hardware y los que se ejecutan sobre un sistema operativo.

3.3.1. Hipervisor de tipo 1

Un hipervisor de tipo 1, también conocido como nativo o *bare metal*, se caracterica por ser software que se ejecuta directamente el hardware. Por lo tanto, se encarga directamente de administrar los recursos de los que se disponen y proporcionarlos a los diferentes entornos virtuales que se ejecutan sobre el mismo sistema físico.

La figura 3.1 muestra una arquitectuta estándar de un hipervisor tipo 1. Se observa que el hipervisor se ejecuta directamente sobre el hardware del entorno físico, y gestiona el acceso a éste de cada uno de los sistemas invitados que se ejecutan.

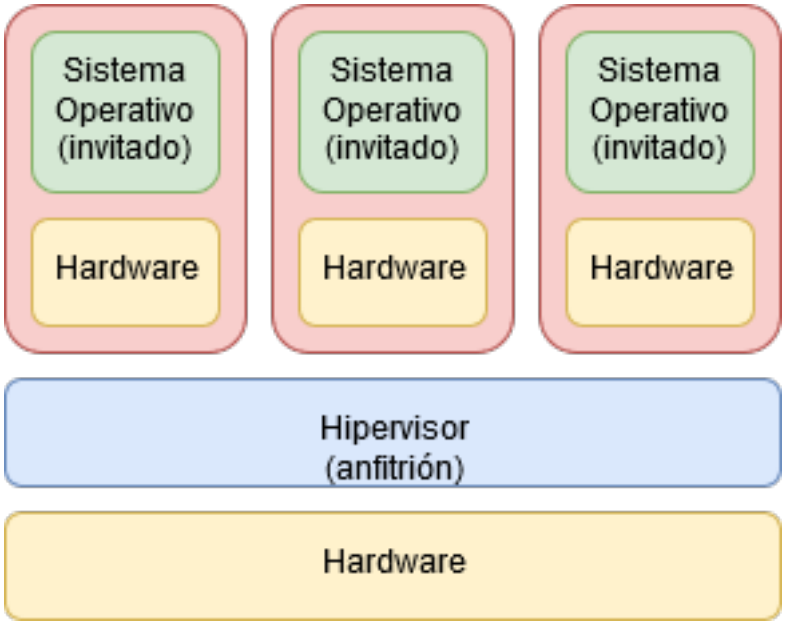

Figura 3.1: Arquitectura estándar de un hipervisor de tipo 1.

En este tipo de hipervisor es el que mayormente se utiliza en el segmento de servidores ya que la sobrecarga debido a la virtualización es menor que con los hipervisores de tipo 2 y esto provoca una mejor optimización de los recursos disponibles.

3.3.2. Hipervisor de tipo 2

Un hipervisor de tipo 2, también conocido como *hosted* (alojado), se caracteriza por ser software que se ejecuta sobre un sistema operativo. Dicho software se ejecuta como cualquier otro programa sobre un sistema operativo

La figura 3.2 representa el esquema de la arquitectura estándar de un hipervisor de tipo 2. Se puede apreciar que el hipervisor se ejecuta como un proceso más del sistema operativo anfitrión (sobre el que se pueden ejecutar otras aplicaciones de manera normal). Este proceso crea una capa de abstracción del hardware, presentando al sistema operativo huésped un hardware virtual.

3.4 Alta disponibilidad en plataformas de virtualización

En la actualidad prácticamente todas las soluciones destinadas a la virtualización de servidores cuentan con una serie de mecanismos que se encargan de proporcionar alta disponibilidad a los diferentes sistemas virtualizados. La función de estos es la de proporcionar tolerancia a fallos en la capa física proporcionando mecanismos que permitan, por una parte la recuperación automática de un sistema virtualizado ante un fallo de hardware sobre el sistema físico donde se esté ejecutando y, por otra, la posibilidad de migrar sistemas virtualizados de un sistema físico a otro para permitir su apagado por diferente motivos, como por ejemplo, por mantenimiento.

Las tecnologías de alta disponibilidad en la virtualización de servidores han surgido como un producto externo a los hipervisores y por lo tanto generalmente no forman parte del cuerpo de la solución. Esto no significa que los hipervisores no se hayan tenido que adaptar a la utilización de dichos productos, cabe destacar la característica de migración de sistemas virtualizados.

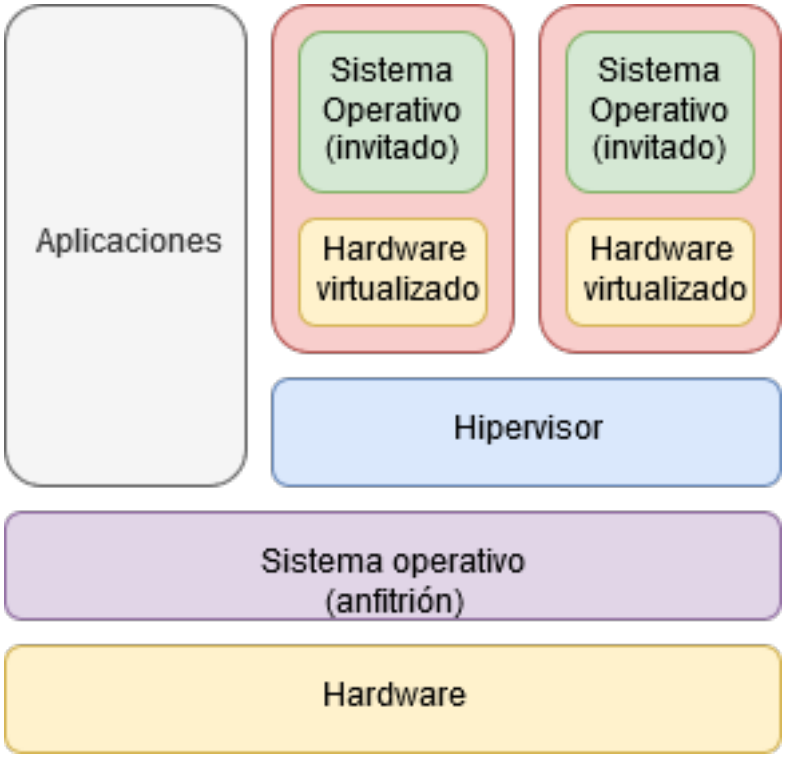

Figura 3.2: Arquitectura estándar de un hipervisor de tipo 2.

Algunas de las tecnologías punteras que proporcionan la funcionalidad de alta disponibilidad son:

- vCenter Server: es la tecnología de alta disponibilidad que incorporan los productos de VMware, e incluye el component vMotion, que es el encargado de migrar sistemas virtualizados. Vio la luz en el año 2003 y desde entonces ha evolucionado para incorporar características como Storage vMotion, que proporciona la posibilidad de migrar el almacenamiento virtual entre varios orígenes.
- Citrix XenCerter: al igual que vCenter, XenCenter proporciona una interfaz de administración y supervisión sobre una infraestructura formada por hipervisores Xen. Aunque como hipervisor utiliza Xen, que es software libre, las herramientas que proporcionan la alta disponibilidad son propietarias de Citrix. El componente que se encarga de la migración de sistemas virtuales es XenMotion.
- Red Hat Virtualization Manager: es el software de administración central de una infraestructura de virtualización mantenido por Red Hat. Integra las capacidades de monitorización y alta disponibilidad de los diferentes componentes de la infraestructura.

3.5 Propuesta

Después de estudiar el panorama actual de la virtualización y las posibles opciones de configuración existentes se observa que hay varias tecnologías capaces de ofrecer virtualización de servidores a nivel de un centro de datos, y las más apropiadas para ofrecer el servicio planteado son los hipervisores de tipo 1 por ofrecer una mejor optimización de lor recuros. Dentro de este tipo de servidores se pueden encontrar opciones propietarias y de código abierto.

Por otra parte, la posibilidad de dotar de alta disponibilidad a una infraestructura de virtualización reside en software mayoritariamente propietario, o en el caso de ser de código abierto (como Red Hat Virtualization Manager) es necesaria la adquisicón de una suscripción para poder tener actualizaciones. Estas opciones son impracticables en el caso de tener bajo presupuesto dedicado a la infraestructura de TI.

Por lo tanto, en este trabajo se va a proponer la configuración de una infraestructura de virtualización que sea asumible para un tipo de usuarios que tengan un bajo presupuesto para invertir en tecnologías de TI. Para poder cumplir con esta premisa, la infraestructura utilizará tecnologías de código abierto en su totalidad donde el desarrollo, actualizaciones y soporte provengan de una comunidad de voluntarios que, al mismo tiempo, se retroalimente con nuevos usuarios que aporten valor a dicha comunidad. La pila de software utilizada deberá ser capaz de aportar características de alta disponibilidad sin tener que hacer una inversión inicial en su adquisición.

CAPÍTULO 4
Análisis del problema

En el presente trabajo se parte de la proposición de crear un entorno de virtualización de bajo coste. Dicho entorno debe cumplir una serie de requisitos para que sea atractivo comercialmente con la intención de que el trabajo realizado sea utilizado por el mayor número de usuarios posible.

Como el problema planteado consta de cierta complejidad se va a descomponer en problemas más pequeños con la finalidad de encontrar una solución de manera más fácil.

4.1 Hardware requerido

La solución a aplicar debería poder ejecutarse sobre cualquier tipo de hardware que el usuario pueda aportar, hecho que podría aumentar el número de potenciales usuarios del producto desarrollado en este trabajo, pero se ha decidido limitar el número mínimo de arquitecturas soportadas a las más populares actualmente: x86, x86_64 y ARM. Des esta manera no conocemos ningún producto de virtualización que se puede ejecutar sobre todas las arquitecturas de procesador existentes.

Esta premisa limita el abanico de software de virtualización que pueda utilizarse, aunque no demasiado ya que se han incluído las arquitecturas más comunes.

En lo que se refiere a la utilización de almacenamiento, aunque se podrían explorar otras posibilidades, se prefiere la utilización del almacenamiento local de los servidores con la intención de abaratar costes. Esta situación podría evitar la adquisición de una solución de almacenamiento externa que sin duda aumentaría el coste asociado a la adquisición de la infraestructura.

4.2 Plataforma de virtualización

Uno de los objectivos del proyecto es proporcionar un servicio IaaS, ofreciendo al usuario sistemas operativos virtualizados sobre los que pueda ejecutar sus aplicaciones. Para que este producto llegue al mayor número posible de usuarios el abanico de sistemas operativos huésped y anfitrión debería ser lo más amplio posible sobre todo los que se van a utilizar como huésped (Linux, BSD, Windows, etc).

La plataforma de virtualización elegida debe ser capaz de aprovechar al máximo los recursos hardware disponibles, por lo que la optimización es un factor importante a tener en cuenta en la elección del producto elegido.

Otro factor importante a tener en cuenta en la elección de la plataforma de virtualización debe ser la seguridad, pues es necesario reducir al mínimo los privilegios de cada uno de los sistemas virtualizados para que un incidente sobre uno de ellos no se pueda propagar al resto.

4.3 Alta disponibilidad

La solución de alta disponibilidad elegida debe ser capaz de dotar de tolerancia a fallos en el hardware a los diferentes sistemas operativos virtualizados, así como proporcionar mecanismos donde las ventanas de mantenimiento hardware no impacten sobre el servicio prestado.

La solución elegida deberá tener en cuenta las posibles situaciones de fallo que se pueden producir y reaccionar de manera previsible ante ellas.

Por último, se debe proporcionar una solución de almacenamiento compartido de tipo distribuido y que, como se ha mencionado anteriormente, sea capaz de aprovechar los recursos locales de almacenamiento que contienen cada uno de los nodos.

CAPÍTULO 5
Diseño de la solución

Una vez analizados los diferentes requisitos que debe cumplir la solución obtenida y producto final de este trabajo se procede al diseño de la solución final.

5.1 Hardware utilizado

El hardware disponible para llevar al cabo el proyecto presentado en este trabajo son tres servidores modelo Primergy RX200 S5 del fabricante Fujitsu y cuentan con las siguientes características:

- Dos procesadores Intel Xeon modelo E5520, cada uno con cuatro cores y tecnología *Hyperthreading* ofreciendo un total de 16 hilos ejecución.
- 16 GB de memoria RAM.
- 2 discos SAS de 300GB configurados con RAID 1.
- 6 interfaces ethernet 1Gb (10 uno de ellos), configurados como dos interfaces en la placa base y una tarjeta (dos tarjetas el tercer nodo) de cuatro puertos.
- Interfaz iRmc de administración remota

A nivel de red, se dispone de dos switches Cisco Catalyst 2960, de 24 puertos cada uno, para poder dotar la red de tolerancia a fallos.

En la figura 5.1 se muestra un mapa de interconexión donde figuran todos los elementos hardware que forman parte de la plataforma. En él se indican las IP asignadas a cada nodo, así como de la interfaz remota de administración.

5.2 Pila de virtualización (Xen Project)

El software de virtualización que se va a utilizar en esta implementación va a ser el hipervisor Xen. Es un hipervisor de código libre que actualmente está impulsado por la Linux Foundation. Este hipervisor se encuentra en pleno desarrollo y actualmente tiene una amplia adopción en el mundo empresarial. Como ejemplo de esta última afirmación se puede citar la empresa Citrix, cuyos productos de virtualización están basados utilizando esta tecnología.

Finalmente, y como complemento a la tecnología mencionada, se utilizará la API libvirt. Esta API proporciona una capa de administración común al usuario independientemente del hipervisor que se utilice como gestor de recursos, y facilitará la tarea de

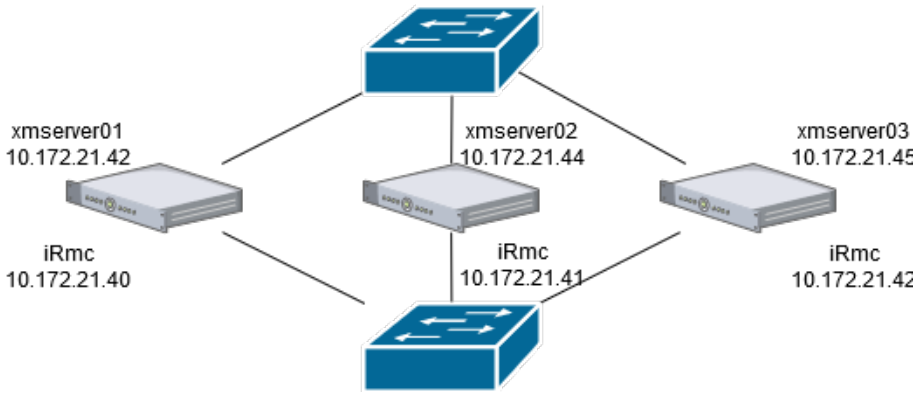

Figura 5.1: Mapa de interconexión de red de la plataforma.

integración del hipervisor Xen con la tecnología de alta disponibilidad que se utilizará en el proyecto presentado por este trabajo.

Aunque la tecnología de hipervisor Xen permite varios sistemas operativos como dominio privilegiado se va a optar por la utilización de Linux como tal. Es el sistema operativo más extendido en este tipo de configuración, razón por la que existe mayor documentación al respecto y mayor soporte por parte de la comunidad desarrolladores. Concretamente se va a utilizar CentOS 7 como sistema base ya que es una distribución que cuenta con el respaldo de la multinacional Red Hat, puntera en soluciones Linux para empresas.

5.2.1. Hipervisor

El hipervisor Xen es un software que se ejecuta directamente sobre el hardware reemplazando el sistema operativo (tipo 1), permitiendo así la ejecución simultánea de varios sistemas operativos. Dicho hipervisor tiene soporte para arquitecturas x86, x86-64, Itanium, PowerPC y ARM permitiendo ejecutarse sobre una gran cantidad de dispositivos. A su vez, soporta la virtualización de sistemas Windows, Linux, BSD y solaris, entre otros.

El hipervisor Xen se liberó bajo licencia GPL en el año 2004, y relativamente poco después se libera la versión 2.0. Al mismo tiempo se fundaba la empresa XenSource, que respaldaba el proyecto, y poco después fue comprada por Citrix, pero el hipervisor quedando como una solucion de código abierto sobre la que después se crearían otros productos. Desde 2013 el proyecto Xen quedó bajo el amparo de *The Linux Foundation*, creando así la marca *Xen Project* para diferenciarse entre los diferentes productos que no son libres.

Las ventajas de utilizar Xen como hipervisor dentro de una infraestructura de virtualización son las siguientes:

- Neutralidad de sistema operativo: una de las características claves en la utilización de xen como hipervisor es la opción de poder elegir entre diferentes sistemas operativos para que actúen como dom0 (Linux, Solaris, BSD, etc). Además, esta separación entre el hipervisor y el dominio de control permite que hipervisor más liviano al que no se sobrecarga con acciones que no sean procesar las peticiones de los sistemas invitados.

- Seguridad: un aspecto crítico de un hipervisor es el de proporcionar seguridad para que los diferentes sistemas virtualizados no interfieran entre ellos. Para ello se

siguen algunos métodos como el aislamiento de sistemas huésped, acceso privilegiado solo para el *dominio 0* (es el único capaz de contactar con el hipervisor), código pequeño (reduciendo las áreas de ataque) y separación del sistema operativo (al separar el hipervisor, este no puede ser utilizado para atacar un sistema operativo).

- Rendimiento: al separar el sistema operativo del hipervisor se consigue que el hipervisor no tenga que ejecutar las tareas normales de un sistema operativo, evitando así el utilizar recursos computacionales innecesarios. Por otra parte, la tecnología de *paravirtualización* consigue que los huéspedes obtengan mejores prestaciones en el acceso al hardware.

5.2.2. Arquitectura

Los diferentes elementos que forman parte de la arquitectura del hipervisor Xen son los siguientes:

- Hipervisor Xen: como se ha mencionado anteriormente, el hipervisor es la capa que se ejecuta directamente sobre el hardware. Este elemento es el responsable de administrar el uso de CPU y memoria de los diferentes sitemas virtuales que se ejecuten sobre el mismo anfitrión. El hipervisor no es capaz de gestionar otro tipo de elementos como los dispositivos de red o el acceso disco, ya que estas tareas recaen sobre un sistema invitado especial que se denomina *dominio 0*.

- Invitado de tipo Dominio 0 (dom0): este invitado de tipo especial es un kernel de Linux modificado y es el único sistema virtual que tiene acceso especial tanto a los recursos de Entrada/Salida como para interactuar con otros sistemas virtuales (Domain U) que se ejecuten sobre el mismo hardware. Todos los entornos de virtualización que utilizan Xen requieren que esté ejecutándose un *dominio 0* antes que cualquier otro sistema virtualizado.

  En el *dominio 0* se incluyen dos manejadores para poder procesar las diferentes peticiones de disco y red que provienen de los otros sistemas virtuales (dominio U), el manejador de servicio de red y el manejador de servicio de disco. Por una parte, el manejador de servicio de disco comunica directamente con el disco local, mediante accesos tanto de lectura como de escritura, para satisfacer la peticiones que provengan de los dominios virtuales. Por otra parte, el manejador de servicio de red realiza las mismas funciones pero sobre los dispositivos de red.

- Invitado de tipo Dominio U: los sistemas virtualizados utilizando un hipervisor de Xen, y que no son de *dominio 0* (un único sistema por servidor físico) pertenecen a este tipo de invitado. Son sistemas no provilegiados que no pueden acceder directamente al hardware, por lo que para poder utilizar dispositivos de entrada/salida deben hacerlo a través del *dominio 0*. Se pueden distinguir dos tipos de según su modo de virtualización.

  1. El primer tipo, los huéspedes PV, utilizan *paravirtualización*, por lo que solo pueden utilizarse sistemas operativos que soporten las modificaciones necesarias (como por ejemplo Linux o FreeBSD). Este tipo de invitados saben que son sitemas virtualizados y utilizan manejadores especiales para el acceso al hardware.
  2. Respecto al segundo tipo, los huéspedes HVM[1], utilizan el modo de virtualización completa, soportando entonces más variedad de sistemas operativos

[1] Del ingés *Hardware Virtual Machine*.

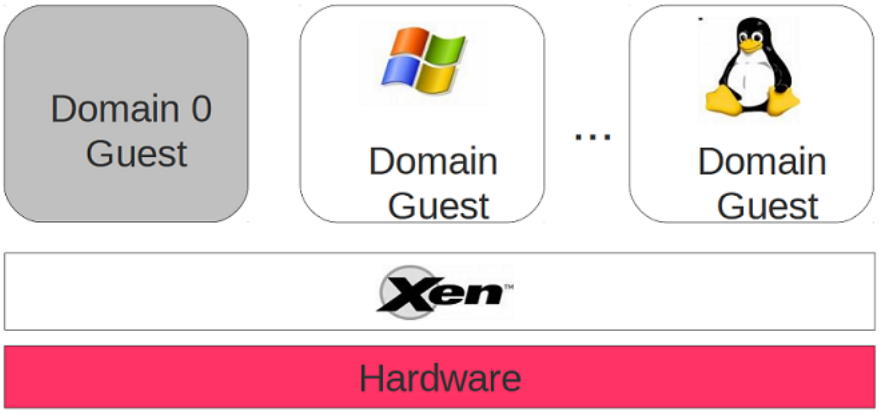

Figura 5.2: Arquitectura de virtualización de Xen. Extraída de [36]

como Windows. En el momento de iniciar este tipo de invitados se ejecuta un servicio especial para cada uno de ellos, Qemu-dm. Este tipo de dominios deben iniciarse como cualquier servidor físico, por lo que Xen incluye un *firmware* especial que simula la BIOS.

La figura 5.2 representa un esquema de la arquitectura de Xen. En ella se pueden identificar los distintos componentes que se han descrito anteriormente.

5.2.3. Administración y control

La administración y control del hipervisor Xen se realiza a través de una serie de demonios y librerías en el dom0. En esta comunicación intervienen:

- xend: el servicio xend se considera el administrador de sistema en un entorno Xen. Utiliza la librería libxenctrl para controlar el hipervisor. Este servicio se controla a través de una interfaz XML RPC utilizando el comando xl (xm en versiones anteriores a Xen 4.0).
- xl: es la herramienta de consola que permite interactuar con el servicio xend. Es la herramienta principal de control y administración de los diferentes sistemas virtualizados.
- xenstored: este servicio mantiene un registro de información con todos los enlaces entre el dom0 y el resto de huéspedes. Estos enlaces son los canales mediante los cuales se comunican los diferentes servidores virtuales con el hardware.
- libxenctrl: es una librería escrita en C que proporciona la capacidad al servicio xend de comunicarse con el hipervisor a través del *domain 0*.
- Firmware virtual de Xen: se trata de una BIOS virtual que se inserta en cada uno de los dominios que utilizan virtualización completa (huéspedes HVM) con la intención de que cada uno de los sistemas virtuales reciba las operaciones estándar que espera.

5.2.4. libvirt

La pila de software de virtualización se va a completar con la utilización de libvirt. Libvirt es una colección de software que proporciona una interfaz genérica de administración tanto de máquinas virtuales como de sus recursos independientemente de la ten-

cología de virtualización utilizada. Este software incluye una librería API, un servicio (libvirtd) y una utilidad de comandos (virsh).

Las características más importantes de libvirt son las siguientes:

- Administración de máquinas virtuales: proporciona las operaciones comunes a realizar sobre este tipo de sistemas (arrancar, parar, migrar, ...). Además permite añadir dispositivos en caliente como pueden ser discos, memoria, CPU, etc.
- Conexión remota: el servicio libvirtd permite conectarse desde otros dispositivos que no seah el sistema local, proporcionando una interfaz de administración remota.
- Gestión de almacenamiento: el servicio libvtird puede crear imágenes de varios formatos (vmdk, qcow2, raw, ...), utilizar NFS, crear volúmenes de LVM, gestionar discos iSCSI, ... Puede realizar estas acciones y muchas más.
- Administración de interfaces de red: el servicio libvirtd tiene capacidad para administrar interfaces de red, tanto físicas como lgícas.

5.3 Alta disponibilidad

La alta disponibilidad en la plataforma va a ser proporcionada por la pila de software recomendada por Red Hat en su distribución de Linux, ya que es una solución de clusterización consolidada entre el mundo empresarial, y proporciona confianza, escalabilidad y disponibilidad a los servicio y aplicaciones que se ejecutan dentro de su entorno.

Un clúster consiste en varios sistemas, llamados nodos, que trabajan juntos para llevar a cabo una tarea, dando lugar a una tipología variada: se pueden diferenciar clústeres de almacenamiento, alta disponibilidad, balanceo de carga y alto rendimiento. Por ser el que más se ajusta al proyecto actual el enfoque se va a centrar en el tipo de alta disponibilidad.

Un clúster de alta disponibilidad proporciona servicios que son capaces de tolerar fallos en los nodos que lo conforman, así como permitir la evacuación de dichos servicios entre nodos, reduciendo así drásticamente la indisponibilidad de las aplicaciones.

5.3.1. Pacemaker

Pacemaker es un software de gestión de recursos en un clúster de alta disponibilidad. Dentro de él se integran una serie de componentes que se encargan de gestionar la membresía de los nodos a un clúster, scripts que administran los servicios y subsistemas que monitorizan los diferentes recursos. Los diferentes componentes que forman parte de su arquitectura son:

- Cluster Information Base (CIB): es el servicio de información de Pacemaker. Utiliza XML internamente para distribuir y sincronizar la información de configuración y estado que contiene el nodo elegido como coordinador[2] a los demás nodos del clúster.
- Cluster Resource Management Daemon (CRMd): es el servicio encargado de administrar los diferentes recursos del clúster. Pacemaker delega sobre este servicio las acciones a realizar sobre cada uno de los recursos.

[2]DC, por sus siglas en inglés (Designated Coordinator)

- Shoot the Other Node in the Head (STONITH): es un recurso del clúster que se encarga de procesar las solicitudes de vallado por parte del clúster, asegurando el apagado de los nodos y eliminándolos del clúster para asegurar la integridad de los datos. Está configurado dentro del servicio CIB y se monitoriza como cualquier otro recurso del clúster.

- Corosync: es el componente (y servicio) que se encarga de controlar la membresía y la comunicación entre los diferentes miembros del clúster. Es capaz de proporcionar capacidad de comunicación a las aplicaciones que deben coordinarse a través de los diferentes miembros del clúster. Además se encarga de gestionar el *quorum*.

La configuración de Pacemaker se realiza a través de la herramienta de línea de comandos pcs. También existe la alternativa de una interfaz web que es capaz de realizar las mismas acciones que la herramienta de línea de comandos.

5.3.2. Quorum

Con la intención de mantener la integridad y la disponibilidad, los sistemas en clúster utilizan un concepto conocido como *quorum* para prevenir la corrupción y la pérdida de datos mediante el *split-brain*, un fenómeno en el que los nodos del cluster pierden la comunicación pero cada parte continúa trabajando como si fueran clústeres separados, procesando los mismos datos y pudiendo causar corrupción. Un cluster tiene *quorum* cuando más de la mitad de los nodos están disponibles. Para mitigar las probabilidades de corrupción o pérdida de datos Pacemaker, por defecto, detiene todos los recursos si un cluster no tiene *quorum*.

El *quorum* se establece utilizando un sistema de votos. Cuando un nodo del cluster no funciona como debiera o pierde la comunicación con el resto del cluster, la mayor parte de los nodos que están funcionando pueden votar para aislar y, en caso necesario, vallar dicho nodo para seguir ofreciendo el servicio.

El soporte de *quorum* en Pacemaker lo proporciona un plugin de corosync llamado *votequorum*, que permite a los administradores configurar un cluster con un número determinado de votos a cada uno de los sistemas y asegurando que solo con una mayoría de votos presentes se permiten las operaciones del clúster. En una situación donde no hay mayoriía (un clúster de 2 nodos) es posible modificar el comportamiento de este componente para que se utilice una política de desempate en la que el nodo de mayor peso es el que posee un identificador menor.

5.3.3. Vallado

Como se ha indiciado anteriormente, un nodo dentro de un clúster puede empezar a comportarse de un modo anómalo en un momento determinado, por lo que el clúster debe proporcionar mecanismos que eviten cualquier tipo de corrupción en los datos. Este hecho se puede mitigar utilizando una política de vallado.

El vallado es la desconexión de un nodo del almacenamiento compartido, cortando las solicitudes de entrada/salida y asegurando la integridad de los datos. Esta desconexión se realiza a través de un mecanismo que recibe el nombre de STONITH, el cual debe asegurarse que la desconexión se ha realizado satisfactoriamente.

Hay diferentes métodos de vallado utilizados por STONITH para asegurar la desconexión de un nodo: UPS, PDU, control de blades, interfaces de administración remota, etc.

5.3.4. Recursos

Un recurso de clúster es una instancia de programa, datos o aplicación que es administrado por el servicio de clúster. Para realizar la gestión de los recursos se utiliza un tipo de objeto llamado agente, cuya función es crear una capa de abstracción sobre los recursos con la finalidad de proporcionar al clúster una interfaz estándar de gestión.

Los diferentes recursos son monitorizados por el clúster a través de los agentes disponibles. Para esta acción se utiliza el mecanismo proporcionado por el agente correspondiente, aunque es posible especificar una operación en concreto.

El clúster permite definir dependencias en los recursos con la finalidad de determinar su comportamiento. Se pueden definir tres clases de dependencias:

- Dependencias de ubicación: este tipo de dependencia determina en qué nodos puede ejecutarse un recurso.
- Dependencias de orden: una dependencia de orden establece el orden en el que se deben ejecutar los recursos.
- Dependencias de ubicación con otros recuros: este tipo de dependencias definen si los recursos pueden ejecutarse en el mismo nodo que otro.

Se pueden definir grupos de recursos, cuya finalidad es servir como atajo para definir una serie de dependencias en las que un conjunto de recursos debe ejecutarse en un mismo nodo, arrancar en un orden concreto y parar en el orden inverso.

5.4 Almacenamiento

Al dotar a la infraestructura de alta disponibilidad es indispensable proporcionar un almacenamiento compartido a todos los nodos que forman parte delclúster. Históricamente dicho almacenamiento compartido se ha proporcionado a través de redes de área de almacenamientoa (ya sean a través de canales de fibra o mediante iSCSI), pero este paradigma comporta los costes de crear dicha red.

Debido a uno de los objectivos del proyecto, el de bajo coste de la infraestructura, no se puede permitir el utilizar una red de este tipo, así que la alternativa es la utilización de tecnologías novedosas que utilizan un paradigma de almacenamiento distribuido. En este caso en concreto la tecnología escogida ha sido Gluster.

Gluster es un sistema de ficheros distribuído y escalable que permite incorporar recursos de almacenamiento de varios servidores en un un solo espacio. Entre sus características principales se encuentran la compatibilidad de la mayor parte del hardware existenten en el mercado, la posibilidad de acceder a los datos a través de los protocolos NFS y SMB, y que es de código abierto. Gluster está presente en miles de empresas a lo largo y ancho del mundo, y forma parte del producto comercial Red Hat Gluster Storage, de Red Hat, lo que asegura un contínuo desarrollo de mejoras.

Se apunta que en la solución a implementar se utilizarán volúmenes de tipo replicado con el número de réplicas configurado en tres. Aunque más adelante en este capítulo se presentan las diferentes opciones de configuración, se adelanta que teniendo tres réplicas de los datos es la forma más fiable de tenerlos disponible, además de prevenir un suceso llamado *split brain*.

5.4.1. Volúmenes

En la tecnología de Gluster el objeto que exporta el servicio y que será utilizado por un cliente es un volumen, y dichos volúmenes están formados por una colección de *bricks*. Un *brick* es la unidad básica de alamacenamiento en Gluster, representado por la exportación de un directorio en un servidor dentro de un grupo de almacenamiento de confianza.

Ya se ha comentado que los volúmenes están formados por *bricks*, y según la disposición de estos se pueden distinguir varios tipos de volúmenes:

- Volumen distribuido: es el tipo de volumen por defecto que ofrece Gluster. Los ficheros se distribuyen en varios *bricks* de manera que cada fichero está almacenado en un solo *brick*, por lo que no se ofrece ningún tipo de redundancia.
- Volumen replicado: este tipo de volumen soluciona el problema de la fiabilidad del tipo anterior, replicaándose los datos en cada uno de los *bricks*. El número de replicas lo decide el administrador en el momento de crear el volumen. La desventaja de este tipo de volúmenes es el espacio de almacenamiento consumido debido a la replicación de los datos.
- Volumen distribuido y replicado: en este tipo de volumen los ficheros se distribuyen en grupos de *bricks* replicados. El número de *bricks* a utilizar debe ser múltiplo del número de réplicas. Este tipo de volúmenes se utiliza cuando se requiere tanto la alta disponibilidad y fiabilidad del almacenamiento como la posibilidad de escalar.
- Volumen segmentado: el tipo de volumen segmentado trata de repartir la carga del acceso al almacenamiento entre los diferentes *bricks* que conforman el volumen dividiendo los datos en segmentos iguales y después guardando cada segmento en un *brick*. Esto permite conseguir un mayor rendimiento en el acceso al almacenamiento, pero no se proporciona redundancia.
- Volumen distribuido y segmentado: se basa en el tipo de volumen segmentado pero distribuyendo los ficheros en mas *bricks*. El número de *bricks* debe ser múltiplo del número de segmentos.

En la figura 5.3 se muestran esquemas de los diferentes tipos de configuración de volúmenes que se pueden realizar utilizando la tecnología Gluster.

5.4.2. *Split Brain*

Como se ha expuesto anteriormente en la sección de alta disponibilidad, el problema del *split brain* se produce cuando una pérdida de conectividad en el clúster hace que los nodos actúen como clústeres separados, pudiendo provocar corrupción sobre los datos. Tambiés se ha visto anteriormente que la forma de prevenir esta situación es implementar un sistema de votos que decida qué nodos pertenecen al clúster legítimo y cuáles no.

En el contexto de Gluster, este problema se puede solucionar de dos maneras: utilizando volúmenes con tres réplicas o utilizando un volumen árbitro.

- Triple réplica: en este tipo de configuración deben estar disponibles dos de las 3 réplicas para que el volumen siga estando en modo lectura/escritura.
- Volumen árbitro: este tipo de configuración ofrece la misma protección que un escenario de triple réplica pero no se requiere invertir en el espacio necesario para mantener tres réplicas de los datos. El árbitro es un tercer volumen configurado como si fuera una tercera réplica de los datos pero que solo contiene la información

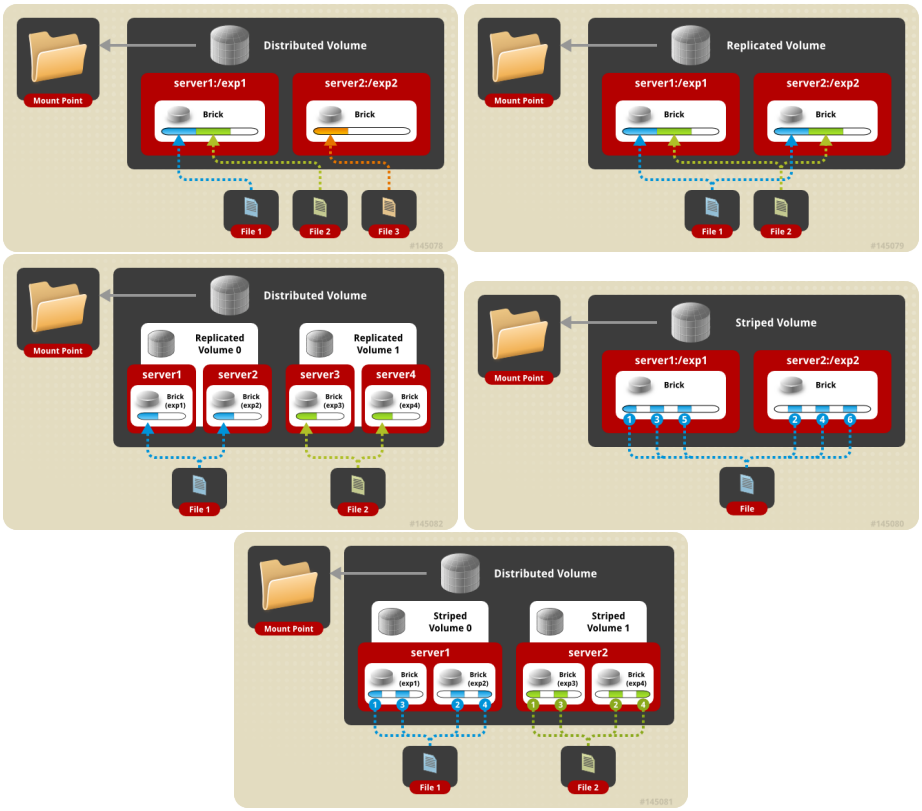

Figura 5.3: Esquemas de arquitectura de voúmenes en Gluster. De arriba a abajo y de izquierda a derecha: volumen distribuido, volumen replicado, volumen distribuido y replicado, volumen segmentado y volumen distribuido y segmentado.

de los ficheros que se guardan, no los datos, ahorrando así ese espacio de almacenamiento.

Sea cual sea el método elegido ambos utilizan la opción *quorum*. El cliente de *quorum* actúa de forma que si una de las réplicas no cumple con las condiciones idóneas entonces pasa a estado de solo lectura, independientemente que las otras réplicas sigan teniendo permisos es critura o no.

CAPÍTULO 6
Desarrollo e implantación de la solución propuesta

En esta sección se va a tratar la implementación de la solución paso a paso con el máximo nivel de detalle posible con la finalidad que el proceso pueda ser reproducido.

Antes de empezar con el proceso detallado se van a establecer algunas consideraciones:

1. En el entorno de laboratorio utilizado para desarrollar la solución propuesta no se dispone de un servidor DNS propio, por lo que la gestión de nombres de hosts se va a realizar sobre el fichero de hosts de los sistemas operativos (en linux el fichero */etc/hosts*).

2. Como no se dispode de un espacio de nombres propio, el dominio a utilizar durante el desarrollo de la solución propuesta será *example.org*.

3. El detalle de los pasos realizados para la consecución de los objetivos parte desde que ya se encuentra un sistema operativo instalado en los nodos que van a formar parte del clúster.

4. Como información, la instalación realizada sobre los nodos se ha realizado a través de CD-ROM, y las opciones de instalación se han dejado por defecto excepto las siguientes:

   - Selección de software: Instalación mínima.
   - Durante la instalación se ha creado el usuario xen.

6.1 Configuración inicial del sistema

Una vez instalado el sistema operativo se realiza la configuración inicial de cada uno de los nodos que van a conformar el clúster. Primero se configura la red en el primer nodo paso a paso:

```
[root@localhost ~]# nmcli general hostname xmserver01
```

El primer paso es configurar el nombre de host, que en este caso se ha hecho utilizando la interfaz de comandos del servicio *NetworkManager*. A continuación se limpia la configuración de red de las interfaces que van a formar parte del *bonding*. En el caso del primer nodo las interfaces *ens1f0* y *enp7s0f1*:

```
[root@xmserver01 ~]# nmcli con del ens1f0
[root@xmserver01 ~]# nmcli con del enp7s0f1
```

El siguiente paso es configurar el interfaz de *bonding* y asignarle la configuración adecuada del protocolo IP:

```
[root@xmserver01 ~]# nmcli con add type bond ifname bond0 con-name \
  bond0 autoconnect yes bond.options "mode=active-backup,miimon=100"
[root@xmserver01 ~]# nmcli con add type ethernet ifname ens1f0 \
  con-name ens1f0 master bond0
[root@xmserver01 ~]# nmcli con add type ethernet ifname enp7s0f1 \
  con-name enp7s0f1 master bond0
[root@xmserver01 ~]# nmcli con modify bond0 ipv4.method manual \
  ipv4.addr 10.172.21.43/24 ipv4.gateway 10.172.21.119
[root@xmserver01 ~]# nmcli con modify bond0 ipv6.method ignore
```

Una vez realizadas todas las operaciones se reinicia el servicio de red para que se aplique la configuración:

```
[root@xmserver01 ~]# systemctl restart network
```

A continuación se muestra la configuración aplicada sobre los otros nodos del clúster. Nótese que en el nodo 3 las interfaces de red utilizadas son diferentes:

```
[root@localhost ~]# nmcli general hostname xmserver02
[root@xmserver02 ~]# nmcli con del ens1f0
[root@xmserver02 ~]# nmcli con del enp7s0f1
[root@xmserver02 ~]# nmcli con add type bond ifname bond0 con-name bond0 \
  autoconnect yes bond.options "mode=active-backup,miimon=100"
[root@xmserver02 ~]# nmcli con add type ethernet ifname ens1f0 con-name \
  ens1f0 master bond0
[root@xmserver02 ~]# nmcli con add type ethernet ifname enp7s0f1 con-name \
  enp7s0f1 master bond0
[root@xmserver02 ~]# nmcli con modify bond0 ipv4.method manual ipv4.addr \
  10.172.21.44/24 ipv4.gateway 10.172.21.119
[root@xmserver02 ~]# nmcli con modify bond0 ipv6.method ignore
[root@xmserver02 ~]# systemctl restart network

[root@localhost ~]# nmcli general hostname xmserver03
[root@xmserver03 ~]# nmcli con del ens1f0
[root@xmserver03 ~]# nmcli con del enp4s0f3
[root@xmserver03 ~]# nmcli con add type bond ifname bond0 con-name bond0 \
  autoconnect yes bond.options "mode=active-backup,miimon=100"
[root@xmserver03 ~]# nmcli con add type ethernet ifname ens1f0 con-name \
  ens1f0 master bond0
[root@xmserver03 ~]# nmcli con add type ethernet ifname enp4s0f3 con-name \
  enp4s0f3 master bond0
[root@xmserver03 ~]# nmcli con modify bond0 ipv4.method manual ipv4.addr \
  10.172.21.45/24 ipv4.gateway 10.172.21.119
[root@xmserver03 ~]# nmcli con modify bond0 ipv6.method ignore
[root@xmserver03 ~]# systemctl restart network
```

Una vez queda configurada la red hacemos se modifican las contraseñas de los usuarios creados durante la instalación:

```
[root@xmserver01 ~]# passwd
[root@xmserver01 ~]# passwd xen

[root@xmserver02 ~]# passwd
[root@xmserver02 ~]# passwd xen

[root@xmserver03 ~]# passwd
[root@xmserver03 ~]# passwd xen
```

Y se desactiva el acceso por ssh del usuario root, debido a una serie de recomendaciones de seguridad para bastionado de sistemas Linux.

```
[root@xmserver01 ~]# sed -i -e 's/\^#\(PermitRootLogin \)yes/\1no/' \
  /etc/ssh/sshd_config
[root@xmserver01 ~]# systemctl restart sshd

[root@xmserver02 ~]# sed -i -e 's/\^#\(PermitRootLogin \)yes/\1no/' \
  /etc/ssh/sshd_config
[root@xmserver02 ~]# systemctl restart sshd

[root@xmserver03 ~]# sed -i -e 's/\^#\(PermitRootLogin \)yes/\1no/' \
  /etc/ssh/sshd_config
[root@xmserver03 ~]# systemctl restart sshd
```

Otro factor a tener en cuenta es la sincronización del tiempo, ya que dentro de un clúster todos los nodos deben estar sincronizados. Por lo tanto, en cada uno de los nodos se va a realizar la configuración del demonio ntp:

```
[root@xmserver01 ~]# yum install ntp
[root@xmserver01 ~]# systemctl enable ntpd
[root@xmserver01 ~]# systemctl start ntpd
```

Con estas acciones los sistemas físicos ya están preparados para poder empezar la parte relevante del presente trabajo de fin de grado: la configuración de la virtualización propiamente dicha.

6.2 Instalación y configuración de la virtualización

En este apartado se va a detallar el proceso de instalación y configuración de la capa de virtualización elegida, Xen.

Se empieza instalando los paquetes necesarios:

```
[root@xmserver01 ~]# yum install centos-release-xen-412
[root@xmserver01 ~]# yum update
[root@xmserver01 ~]# yum install xen
[root@xmserver01 ~]# reboot

[root@xmserver02 ~]# yum install centos-release-xen-412
[root@xmserver02 ~]# yum update
[root@xmserver02 ~]# yum install xen
[root@xmserver02 ~]# reboot

[root@xmserver03 ~]# yum install centos-release-xen-412
[root@xmserver03 ~]# yum update
[root@xmserver03 ~]# yum install xen
[root@xmserver03 ~]# reboot
```

Primero se instala el paquete que contiene los repositorios de Xen, y a continuación se instala el paquete "xen", que instalará el hipervisor y todas sus dependencias. Se puede observar una traza del proceso de instalación del software en la figura 6.1. A continuación se reiniciará el host para que se cargue el hipervisor.

Al reiniciar el nodo aparecerá una nueva opción en el arranque del sistema (figura 6.2) que permitirá arrancar el sistema utilizando el hipervisor de Xen. Tras el reinicio del sistema el hipervisor de Xen está ejecutándose sobre el hardware, y desde este punto toda la administración del host se realiza sobre el huésped *Dom0*. Se puede comprobar el funcionamiento del hipervisor ejecutando el *xl info*, que nos mostrará información sobre el hipervisor en ejecución. Puede verse un ejemplo en la figura 6.3

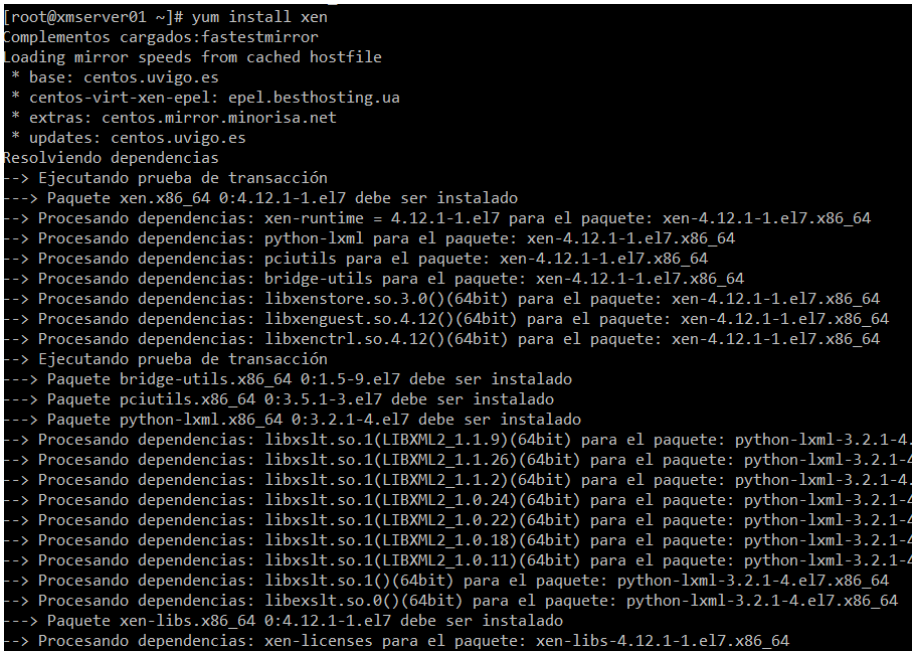

Figura 6.1: Captura de la instalación del software de Xen.

El siguiente paso es configurar un interfaz de red de tipo puente que pueda ser utilizado por los diversos sistemas virtuales que se ejecuten sobre el host. Para ello, se configura un enlace con el interfaz de red que había antes configurado.

```
[root@xmserver01 ~]# nmcli con add type bridge con-name xenbr0 ifname xenbr0
[root@xmserver01 ~]# nmcli con mod xenbr0 bridge.stp no
[root@xmserver01 ~]# nmcli con mod xenbr0 bridge.hello-time 0
[root@xmserver01 ~]# nmcli con modify bond0 connection.master xenbr0 \
  connection.slave-type bridge
[root@xmserver01 ~]# nmcli con modify xenbr0 ipv4.method manual ipv4.addr \
  10.172.21.43/24 ipv4.gateway 10.172.21.119
[root@xmserver01 ~]# systemctl restart network

[root@xmserver02 ~]# nmcli con add type bridge con-name xenbr0 ifname xenbr0
[root@xmserver02 ~]# nmcli con mod xenbr0 bridge.stp no
[root@xmserver02 ~]# nmcli con mod xenbr0 bridge.hello-time 0
[root@xmserver02 ~]# nmcli con modify bond0 connection.master xenbr0 \
  connection.slave-type bridge
[root@xmserver02 ~]# nmcli con modify xenbr0 ipv4.method manual ipv4.addr \
  10.172.21.44/24 ipv4.gateway 10.172.21.119
[root@xmserver02 ~]# systemctl restart network

[root@xmserver03 ~]# nmcli con add type bridge con-name xenbr0 ifname xenbr0
[root@xmserver03 ~]# nmcli con mod xenbr0 bridge.stp no
[root@xmserver03 ~]# nmcli con mod xenbr0 bridge.hello-time 0
[root@xmserver03 ~]# nmcli con modify bond0 connection.master xenbr0 \
  connection.slave-type bridge
[root@xmserver03 ~]# nmcli con modify xenbr0 ipv4.method manual ipv4.addr \
  10.172.21.45/24 ipv4.gateway 10.172.21.119
[root@xmserver03 ~]# systemctl restart network
```

Al configurarse la interfaz puente se observa que una de las opciones desactiva el protocolo STP en dicha interfaz. Esta recomendación figura en wl wiki del proyecto Xen, y se desactiva porque puede causar problemas al levantar nuevas direcciones IP y también se evita la inundación de tráfico de control emitida por este procotolo.

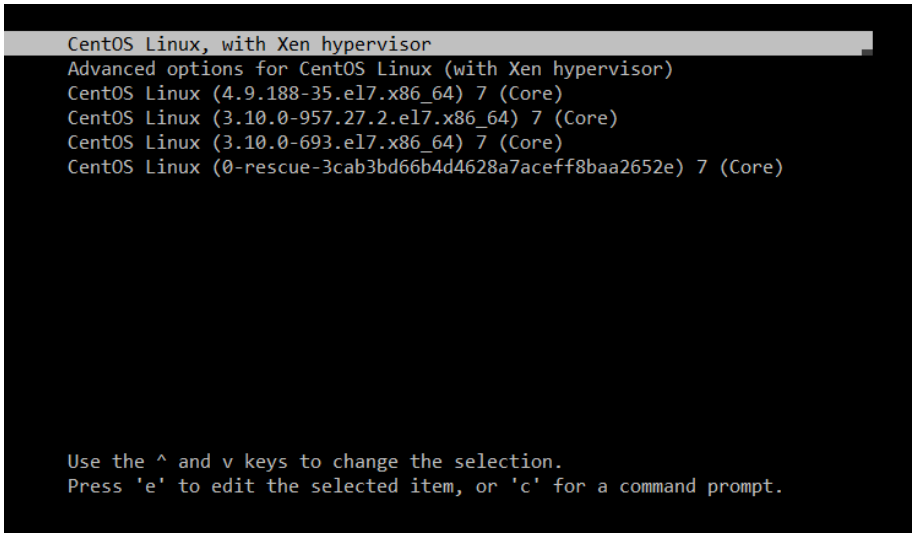

Figura 6.2: Arranque del hipervisor.

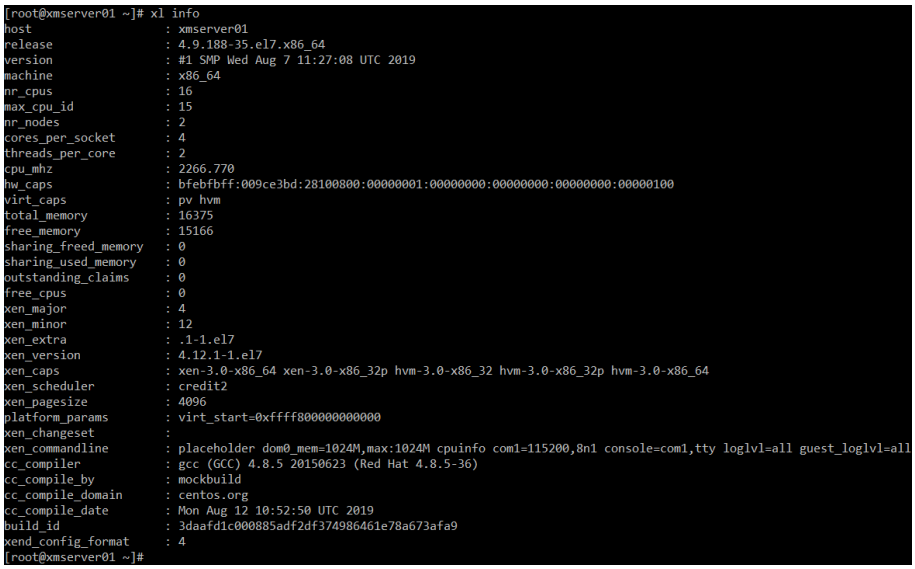

Figura 6.3: Información proporcionada por el hipervisor Xen.

6.3 Instalación y configuración de la interfaz libvirt

La instalación del software necesario se realiza mediante la herramienta *yum*.

```
[root@xmserver01 ~]# yum install libvirt libvirt-daemon-xen virt-install \
  virt-viewer

[root@xmserver02 ~]# yum install libvirt libvirt-daemon-xen virt-install \
  virt-viewer

[root@xmserver03 ~]# yum install libvirt libvirt-daemon-xen virt-install \
  virt-viewer
```

Una vez instalado el software, se ha detectado un error en la localización de in binario al ejectuar *virt-install*, por lo que es necesario crear un enlace simbólico.

```
[root@xmserver01 ~]# ln -s /usr/lib64/xen/bin/qemu-system-i386 \
  /usr/lib/xen/bin/qemu-system-i386

[root@xmserver02 ~]# ln -s /usr/lib64/xen/bin/qemu-system-i386 \
```

```
  /usr/lib/xen/bin/qemu-system-i386

[root@xmserver03 ~]# ln -s /usr/lib64/xen/bin/qemu-system-i386 \
  /usr/lib/xen/bin/qemu-system-i386
```

Una vez instalado el software, ya se podría utilizar la librería libvirt para gestionar los diferentes sistemas virtualizados, pero el producto final tiene funciones de alta disponibilidad, por lo que los sistemas virtualizados deben poder migrarse de un anfitrión a otro. Para poder utilizar esta funcionalidad la librería libvirt utiliza autenticación, y como la autenticación no debe ser interactiva, por la gestión automática del clúster, dicha autenticación se realizará mediante certificados. Por lo tanto, es necesario crear una infraestructura de clave pública que identifique a los nodos entre ellos.

Se empieza creando las claves pública y privada de la autoridad de certificación. Por su naturaleza, este proceso se realizará en un solo nodo y las claves creadas se distribuirán a los demás.

```
[root@xmserver01 ~]# mkdir EPC_CA
[root@xmserver01 ~]# cd EPC_CA/
[root@xmserver01 EPC_CA]# openssl genrsa -des3 -out epc_ca.key 4096
[root@xmserver01 EPC_CA]# openssl req -x509 -new -nodes -key epc_ca.key \
  -sha256 -days 3650 -out epc_ca.pem
```

De la ejecución de los comandos anteriores se crean un archivo con la clave privada de la autoridad de certificación (*epc_ca.key*) y su clave pública (*epc_ca.pem*). Con estos certificados se firmarán los certificados de los clientes y así podrán ser validados.

Ahora se generan las claves privadas de cada nodo y las peticiones de firma para que sean validados por la autoridad de certificación creada anteriormente. Una vez firmadas esas peticiones se obtiene la clave pública de los diferentes certificados.

```
[root@xmserver01 EPC_CA]# openssl genrsa -out xmserver01.example.org.key 2048
[root@xmserver01 EPC_CA]# openssl req -new -key xmserver01.example.org.key \
  -out xmserver01.example.org.csr
[root@xmserver01 EPC_CA]# openssl genrsa -out xmserver02.example.org.key 2048
[root@xmserver01 EPC_CA]# openssl req -new -key xmserver02.example.org.key \
  -out xmserver02.example.org.csr
[root@xmserver01 EPC_CA]# openssl genrsa -out xmserver03.example.org.key 2048
[root@xmserver01 EPC_CA]# openssl req -new -key xmserver03.example.org.key \
  -out xmserver03.example.org.csr
[root@xmserver01 EPC_CA]# x509 -req -in xmserver01.example.org.csr \
  -CA epc_ca.pem -CAkey epc_ca.key -CAcreateserial \
  -out xmserver01.example.org.pem -days 365 -sha256
[root@xmserver01 EPC_CA]# x509 -req -in xmserver02.example.org.csr \
  -CA epc_ca.pem -CAkey epc_ca.key -CAcreateserial \
  -out xmserver02.example.org.pem -days 365 -sha256
[root@xmserver01 EPC_CA]# x509 -req -in xmserver03.example.org.csr \
  -CA epc_ca.pem -CAkey epc_ca.key -CAcreateserial \
  -out xmserver03.example.org.pem -days 365 -sha256
```

A continuación se envían a cada nodo los ficheros que se corresponden con la clave pública de la autoridad de certificación y las claves pública y privada de ese mismo nodo, y cada uno de esos ficheros de ubicará en el directorio correspondiente dentro de la configuración de libvirt.

```
[root@xmserver01 EPC_CA]# scp epc_ca.pem xen@xmserver01:
[root@xmserver01 EPC_CA]# scp xmserver01.example.org.key xen@xmserver01:
[root@xmserver01 EPC_CA]# scp xmserver01.example.org.pem xen@xmserver01:
[root@xmserver01 EPC_CA]# scp epc_ca.pem xen@xmserver02:
[root@xmserver01 EPC_CA]# scp xmserver02.example.org.key xen@xmserver02:
[root@xmserver01 EPC_CA]# scp xmserver02.example.org.pem xen@xmserver02:
```

```
[root@xmserver01 EPC_CA]# scp epc_ca.pem xen@xmserver03:
[root@xmserver01 EPC_CA]# scp xmserver03.example.org.key xen@xmserver03:
[root@xmserver01 EPC_CA]# scp xmserver03.example.org.pem xen@xmserver03:

[root@xmserver01 ~]# mkdir -p /etc/pki/libvirt/private
[root@xmserver01 ~]# mv /home/xen/epc_ca.pem /etc/pki/CA/cacert.pem
[root@xmserver01 ~]# chown root:root /etc/pki/CA/cacert.pem
[root@xmserver01 ~]# chmod 644 /etc/pki/CA/cacert.pem
[root@xmserver01 ~]# cp /home/xen/xmserver01.example.org.pem \
  /etc/pki/libvirt/
[root@xmserver01 ~]# cp /home/xen/xmserver01.example.org.key \
  /etc/pki/libvirt/private/
[root@xmserver01 ~]# chown -R root:qemu \
  /etc/pki/libvirt/xmserver01.example.org.pem /etc/pki/libvirt/private
[root@xmserver01 ~]# chmod 750 /etc/pki/libvirt/private
[root@xmserver01 ~]# chmod 640 \
  /etc/pki/libvirt/private/xmserver01.example.org.key

[root@xmserver02 ~]# mkdir -p /etc/pki/libvirt/private
[root@xmserver02 ~]# mv /home/xen/epc_ca.pem /etc/pki/CA/cacert.pem
[root@xmserver02 ~]# chown root:root /etc/pki/CA/cacert.pem
[root@xmserver02 ~]# chmod 644 /etc/pki/CA/cacert.pem
[root@xmserver02 ~]# cp /home/xen/xmserver01.example.org.pem \
  /etc/pki/libvirt/
[root@xmserver02 ~]# cp /home/xen/xmserver01.example.org.key \
  /etc/pki/libvirt/private/
[root@xmserver02 ~]# chown -R root:qemu \
  /etc/pki/libvirt/xmserver01.example.org.pem /etc/pki/libvirt/private
[root@xmserver02 ~]# chmod 750 /etc/pki/libvirt/private
[root@xmserver02 ~]# chmod 640 \
  /etc/pki/libvirt/private/xmserver01.example.org.key

[root@xmserver02 ~]# mkdir -p /etc/pki/libvirt/private
[root@xmserver02 ~]# mv /home/xen/epc_ca.pem /etc/pki/CA/cacert.pem
[root@xmserver02 ~]# chown root:root /etc/pki/CA/cacert.pem
[root@xmserver02 ~]# chmod 644 /etc/pki/CA/cacert.pem
[root@xmserver02 ~]# cp /home/xen/xmserver01.example.org.pem \
  /etc/pki/libvirt/
[root@xmserver02 ~]# cp /home/xen/xmserver01.example.org.key \
  /etc/pki/libvirt/private/
[root@xmserver02 ~]# chown -R root:qemu \
  /etc/pki/libvirt/xmserver01.example.org.pem /etc/pki/libvirt/private
[root@xmserver02 ~]# chmod 750 /etc/pki/libvirt/private
[root@xmserver02 ~]# chmod 640 \
  /etc/pki/libvirt/private/xmserver01.example.org.key
```

Después de colocar los diferentes certificados en los lugares indicados se modifica la configuración del servicio libvirtd para que se adecúe a la nueva configuración. Esto es, se modifica el fichero de configuración */etc/libvirt/libvirtd.conf* y se establecen los siguientes valores en las variables correspondientes, adecuándose los diferentes nombres de ficheros según el host que se está configurando:

```
key_file = "/etc/pki/libvirt/private/xmserver01.example.org.key"
cert_file =  "/etc/pki/libvirt/xmserver01.example.org.pem"
ca_file = "/etc/pki/CA/cacert.pem"
```

Y en el mismo archivo se configuran los clientes autorizados seguún la propiedad DN de los certificados utilizados por cada cliente. En el caso de ejemplo mostrado el valor y la propiedad quedarían de la siguiente forma (cabe indicar que el símbolo ? representa un caracter comodín):

```
tls_allowed_dn_list = ["C=ES,ST=Valencia,L=Valencia,O=EPC,OU=Servers,CN=
    xmserver0?.example.org,EMAIL=enparcar@ei.upv.es"]
```

Se continúa configurando el servicio libvirtd para que pueda atender conexiones remotas. Esto se realiza modificando el fichero */etc/sysconfig/libvirtd* para que la variable LIBVIRTD_ARGS quede como sigue:

```
LIBVIRTD_ARGS="--listen"
```

Como la infraestructura descrita utiliza un sistema de clúster para dotar de alta disponibilidad a los sistemas virtuales los ficheros de configuración de cada uno de ellos debe residir sobre un directorio compartido. Como dichos ficheros de configuración residen en el directorio */etc/libvirt/libxl* (esto es en el caso de utilizar Xen como hipervisor), la solución adoptada es crear un enlace simbólico al directorio compartido destinado a albergar los archivos de configuarción.

```
[root@xmserver01 ~]# rm -r /etc/libvirt/libxl
[root@xmserver01 ~]# ln -s /datos/xenconf/guests /etc/libvirt/libxl

[root@xmserver02 ~]# rm -r /etc/libvirt/libxl
[root@xmserver02 ~]# ln -s /datos/xenconf/guests /etc/libvirt/libxl

[root@xmserver03 ~]# rm -r /etc/libvirt/libxl
[root@xmserver03 ~]# ln -s /datos/xenconf/guests /etc/libvirt/libxl
```

Por último solo queda reiniciar el servicio para que aplique toda la configuración.

```
[root@xmserver01 ~]# systemctl restart libvirtd

[root@xmserver02 ~]# systemctl restart libvirtd

[root@xmserver03 ~]# systemctl restart libvirtd
```

6.4 Instalación y configuración del almacenamiento

Al igual que se ha hecho en el apartado anterior, el primer paso es instalar los paquetes necesarios para ejecutar el servicio de Gluster.

```
[root@xmserver01 ~]# yum install centos-release-gluster6
[root@xmserver01 ~]# yum install glusterfs gluster-cli glusterfs-libs \
  glusterfs-server

[root@xmserver02 ~]# yum install centos-release-gluster6
[root@xmserver02 ~]# yum install glusterfs gluster-cli glusterfs-libs \
  glusterfs-server

[root@xmserver03 ~]# yum install centos-release-gluster6
[root@xmserver03 ~]# yum install glusterfs gluster-cli glusterfs-libs \
  glusterfs-server
```

Una vez instalado el software se continúa configurando los bloques que el servicio va a utilizar. En la plataforma se van a crear dos volúmenes, uno para datos y otro para configuración, por lo que se deben crear dos bloques en cada nono. Esta tarea, aunque solo se va a reflejar el proceso de uno de los nodos, se debe ejecutar en todos los nodos que van a formar parte de la plataforma.

```
[root@xmserver01 ~]# lvcreate -n lvbrick1 -L 30G rootvg
  Logical volume "lvbrick1" created.
[root@xmserver01 ~]# lvcreate -n lvbrick2 -l 100%FREE rootvg
  Logical volume "lvbrick2" created.
[root@xmserver01 ~]# mkfs.xfs /dev/rootvg/lvbrick1
```

```
...
[root@xmserver01 ~]# mkfs.xfs /dev/rootvg/lvbrick2
...
```

Hasta aquí se han creado dos volúmenes lógicos que sobre el grupo de volúmenes rootvg: lvbrick1, con un tamaño de 30Gb y que formará parte del volumen de Gluster destinado a la configuración de las máquinas virtuales y lvbrick2, con el resto del tamaño que queda libre, que formar
a parte del volumen de Gluster destinado a los datos. Una vez creados los volúmenes se ha creado un sistemas de ficheros sobre cada uno de ellos.

A continuación se incluyen los puntos de montaje donde se montarán los volúmenos lógicos creados anteriormente en el archivo */etc/fstab* para que se monten durante el arranque del sistema y se montan utilizando el comando *mount*.

```
[root@xmserver01 ~]# cat /etc/fstab
...
/dev/mapper/rootvg-lvbrick1     /gluster/brick1 xfs     defaults     0 0
/dev/mapper/rootvg-lvbrick2     /gluster/brick2 xfs     defaults     0 0
[root@xmserver01 ~]# mkdir -p /gluster/brick1
[root@xmserver01 ~]# mkdir -p /gluster/brick2
[root@xmserver01 ~]# mount -a
[root@xmserver01 ~]# df -h /gluster/brick1 /gluster/brick2
root@minimal:~# df -h
Filesystem                    Size  Used Avail Use% Mounted on
/dev/mapper/rootvg-lvbrick1    30G   33M   30G   1% /gluster/brick1
/dev/mapper/rootvg-lvbrick2   170G   33M  170G   1% /gluster/brick2
```

Una vez creados los sistemas de ficheros que funcionarán como bloques de Gluster en cada uno de los nodos, se continúa habilitando el arranque automático del servicio de Gluster y se arranca:

```
[root@xmserver01 ~]# systemctl enable glusterd
Created symlink from /etc/systemd/system/multi-user.target.wants/glusterd.
    service to /usr/lib/systemd/system/glusterd.service.
[root@xmserver01 ~]# systemctl start glusterd

[root@xmserver02 ~]# systemctl enable glusterd
Created symlink from /etc/systemd/system/multi-user.target.wants/glusterd.
    service to /usr/lib/systemd/system/glusterd.service.
[root@xmserver02 ~]# systemctl start glusterd

[root@xmserver03 ~]# systemctl enable glusterd
Created symlink from /etc/systemd/system/multi-user.target.wants/glusterd.
    service to /usr/lib/systemd/system/glusterd.service.
[root@xmserver03 ~]# systemctl start glusterd
```

Como se confía en la conectividad entre los tres servidores se adaptan las reglas del cortafuegos para que cada uno de ellos permita cualquier conexión de de cualquier otro nodo que forme parte de la infraestructura de virtualización. Esta acción también simplifica la configuración de la alta disponibilidad (que se realizará en un capítulo posterior).

```
[root@xmserver01 ~]# firewall-cmd --add-rich-rule='rule family="ipv4" \
  source address="10.172.21.43" accept'
[root@xmserver01 ~]# firewall-cmd --add-rich-rule='rule family="ipv4" \
  source address="10.172.21.44" accept'
[root@xmserver01 ~]# firewall-cmd --add-rich-rule='rule family="ipv4" \
  source address="10.172.21.45" accept'
[root@xmserver01 ~]# firewall-cmd --runtime-to-permanent

[root@xmserver02 ~]# firewall-cmd --add-rich-rule='rule family="ipv4" \
  source address="10.172.21.43" accept'
```

```
[root@xmserver02 ~]# firewall-cmd --add-rich-rule='rule family="ipv4" \
  source address="10.172.21.44" accept'
[root@xmserver02 ~]# firewall-cmd --add-rich-rule='rule family="ipv4" \
  source address="10.172.21.45" accept'
[root@xmserver02 ~]# firewall-cmd --runtime-to-permanent

[root@xmserver03 ~]# firewall-cmd --add-rich-rule='rule family="ipv4" \
  source address="10.172.21.43" accept'
[root@xmserver03 ~]# firewall-cmd --add-rich-rule='rule family="ipv4" \
  source address="10.172.21.44" accept'
[root@xmserver03 ~]# firewall-cmd --add-rich-rule='rule family="ipv4" \
  source address="10.172.21.45" accept'
[root@xmserver03 ~]# firewall-cmd --runtime-to-permanent
```

El siguiente paso es conectar los nodos para crear la confianza entre los tres servidores. Este paso solo es necesario ejecutarse en uno de los nodos.

```
[root@xmserver01 ~]# gluster peer probe xmserver02.example.org
[root@xmserver01 ~]# gluster peer probe xmserver03.example.org
[root@xmserver01 ~]# gluster peer status
```

Se puede observar un ejemplo de confianza en la figura 6.4

```
[root@xmserver01 ~]# gluster peer status
Number of Peers: 2

Hostname: xmserver02
Uuid: c86081eb-e715-44bf-bb9b-8cf061c7541d
State: Peer in Cluster (Connected)

Hostname: xmserver03
Uuid: 95309c5c-64c4-4ed4-ab43-afe5f6d9404d
State: Peer in Cluster (Connected)
[root@xmserver01 ~]#
```

Figura 6.4: Estado de la confianza entre los nodos del grupo de confianza de Gluster.

Una vez configurado el grupo de confianza ya se pueden crear los volúmenes necesarios (solo desde uno de los nodos).

```
[root@xmserver01 ~]# gluster volume create glxenconf replica 3 \
  xmserver01:/gluster/brick1/data \
  xmserver02:/gluster/brick1/data \
  xmserver03:/gluster/brick1/data
volume create: glxenconf: success: please start the volume to access data
[root@xmserver01 ~]# gluster volume create glxendata replica 3 \
  xmserver01:/gluster/brick2/data \
  xmserver02:/gluster/brick2/data \
  xmserver03:/gluster/brick2/data
volume create: glxendata: success: please start the volume to access data
[root@xmserver01 ~]# gluster volume start glxenconf
volume start: glxenconf: success
[root@xmserver01 ~]# gluster volume start glxendata
volume start: glxendata: success
```

En la figura 6.5 se puede observar un ejemplo del estado de los volúmenes creados.

Con esto ya se ha terminado de realizar la configuración de los volúmenes de Gluster. En este caso no se montan dichos volúmenes en el árbol de directorios porque dicho control se va a delegar a los servicios de cluster. Esto asegurará la monitorización de los

```
[root@xmserver02 ~]# gluster volume status
Status of volume: glxenconf
Gluster process                                 TCP Port  RDMA Port  Online  Pid
------------------------------------------------------------------------------
Brick xmserver01:/gluster/brick1/data           49152     0          Y       1833
Brick xmserver02:/gluster/brick1/data           49152     0          Y       1911
Brick xmserver03:/gluster/brick1/data           49152     0          Y       1984
Self-heal Daemon on localhost                   N/A       N/A        Y       1981
Self-heal Daemon on xmserver01                  N/A       N/A        Y       1893
Self-heal Daemon on xmserver03                  N/A       N/A        Y       2015

Task Status of Volume glxenconf
------------------------------------------------------------------------------
There are no active volume tasks

Status of volume: glxendata
Gluster process                                 TCP Port  RDMA Port  Online  Pid
------------------------------------------------------------------------------
Brick xmserver01:/gluster/brick2/data           49153     0          Y       1865
Brick xmserver02:/gluster/brick2/data           49153     0          Y       1959
Brick xmserver03:/gluster/brick2/data           49153     0          Y       2001
Self-heal Daemon on localhost                   N/A       N/A        Y       1981
Self-heal Daemon on xmserver01                  N/A       N/A        Y       1893
Self-heal Daemon on xmserver03                  N/A       N/A        Y       2015

Task Status of Volume glxendata
------------------------------------------------------------------------------
There are no active volume tasks

[root@xmserver02 ~]#
```

Figura 6.5: Información proporcionada por el hipervisor Xen.

diferentes puntos de montaje y asegurará que estén disponibles antes de la ejcución de los diferentes sistemas virtuales.

6.5 Instalación y configuración de la alta disponibilidad

El primer paso en la configuración de la alta disponibilidad es instalar el software de clúster.

```
[root@xmserver01 ~]# yum install pcs pacemaker fence-agents-all

[root@xmserver02 ~]# yum install pcs pacemaker fence-agents-all

[root@xmserver03 ~]# yum install pcs pacemaker fence-agents-all
```

Durante la instalación de los paquetes se ha creado el usuario hacluster, que será el administrador del cluster. Por lo tanto, se debe establecer una contraseña para dicho usuario.

```
[root@xmserver01 ~]# passwd hacluster

[root@xmserver02 ~]# passwd hacluster

[root@xmserver03 ~]# passwd hacluster
```

Una vez establecida la contraseña del usuario se levantan los servicios de clúster en todos los nodos y se configura el inicio automático durante el arranque.

```
[root@xmserver01 ~]# systemctl enable pcsd
[root@xmserver01 ~]# systemctl start pcsd

[root@xmserver02 ~]# systemctl enable pcsd
[root@xmserver02 ~]# systemctl start pcsd

[root@xmserver03 ~]# systemctl enable pcsd
[root@xmserver03 ~]# systemctl start pcsd
```

A continuación se van a establecer las relaciones de confianza entre los nodos y se inicializará un clúster que recibirá el nombre de *xencluster*. Esta operación solo se realiza desde uno de ellos.

```
[root@xmserver01 ~]# pcs cluster auth xmserver01 xmserver02 xmserver03
[root@xmserver01 ~]# pcs cluster setup --start --name xencluster \
  xmserver01 xmserver02 xmserver03
```

Una vez creado el clúster se configura para que se inicie automáticamente con el servidor.

```
[root@xmserver01 ~]# pcs cluster enable --all
```

Una vez se ha creado el clúster el siguiente paso es crear los recursos que van a ser administrados. Cabe tener en cuenta que los comandos que afenten al clúster solo se lanzan en uno de los nodos, ya que directamente afectan a todo el clúster. Se empezará creando los recursos de vallado.

```
[root@xmserver01 ~]# pcs stonith create fence_xm01 fence_ipmilan \
  ipaddr=10.172.21.40 login=****** passwd=****** pcmk_host_list=xmserver01
[root@xmserver01 ~]# pcs stonith create fence_xm02 fence_ipmilan \
  ipaddr=10.172.21.41 login=****** passwd=****** pcmk_host_list=xmserver02
[root@xmserver01 ~]# pcs stonith create fence_xm03 fence_ipmilan \
  ipaddr=10.172.21.42 login=****** passwd=****** pcmk_host_list=xmserver03
```

En la infraestructura de la que se dispone se ha configurado el vallado utilizando el agente *fence_ipmilan*. Este agente utiliza la interfaz IPMI para interactuar con la interfaz de administración del hardware, y se ha configurado para que se valide utilizando usuario y contraseña.

Anteriormente se ha indicado que los sistemas de ficheros compartidos no se configurarían de forma habitual porque serían controlados por el clúster. A continuación se procederá a configurar dichos recursos de forma que estén disponibles en todos los nodos del clúster.

```
[root@xmserver01 ~]# pcs resource create xencfg Filesystem \
  device="localhost:/glxenconf"  directory="/datos/xenconf" \
  fstype="glusterfs"
[root@xmserver01 ~]# pcs resource clone xencfg
[root@xmserver01 ~]# pcs resource create xendata Filesystem \
  device="localhost:/glxendata"  directory="/datos/xendata" \
  fstype="glusterfs"
[root@xmserver01 ~]# pcs resource clone xendata
```

En la figura 6.6 se muestra un ejemplo de estado del clúster que se obtiene ejecutando *pcs status*. Se puede observar que los recursos de almacenamiento (xencfg y xendata) están ejecutándose correctamente en los tres nodos, y los recursos de tipo STONITH correspondientes a cada nodo (que se encargarán de parar ese nodo en concreto) se están ejecutando en uno que no sea él mismo. Además se observan dos recursos más de tipo *VirtualDomain* que se verá más adelante en este documento.

6.6 Primer sistema virtual

Una vez instalada y configurada toda la pila de software, realizado durante las secciones anteriores de este capítulo, es momento de crear el primer sistema virtualizado sobre la infraestructura. Para ello se utilizará libvirt por ser la librería que utilizará el clúster para gestionar los sistemas virtuales, además de incorporar un script que facilita enormemente la creación e instalación de un sistema virtualizado.

```
[root@xmserver01 ~]# pcs status
Cluster name: xencluster
Stack: corosync
Current DC: xmserver03 (version 1.1.19-8.el7_6.4-c3c624ea3d) - partition with quorum
Last updated: Thu Nov 28 11:31:36 2019
Last change: Thu Nov 28 11:31:33 2019 by hacluster via crmd on xmserver03

3 nodes configured
11 resources configured

Online: [ xmserver01 xmserver02 xmserver03 ]

Full list of resources:

 fvm    (ocf::heartbeat:VirtualDomain): Started xmserver01
 winvm  (ocf::heartbeat:VirtualDomain): Started xmserver02 (Monitoring)
 fence_xm01     (stonith:fence_ipmilan):        Started xmserver03
 fence_xm02     (stonith:fence_ipmilan):        Started xmserver03
 fence_xm03     (stonith:fence_ipmilan):        Started xmserver02
 Clone Set: xencfg-clone [xencfg]
     Started: [ xmserver01 xmserver02 xmserver03 ]
 Clone Set: xendata-clone [xendata]
     Started: [ xmserver01 xmserver02 xmserver03 ]

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
[root@xmserver01 ~]#
```

Figura 6.6: Ejemplo de salida del comando *pcs status*

El primer paso es crear el disco virtual que pertenecerá al sistema mediante el comando *dd*, ya que el script a utilizar no lo crea automáticamente. Se creará un disco virtual de 40GB.

```
[root@xmserver02 ~]# dd if=/dev/zero of=/datos/xendata/winvm.img \
  bs=1M count=40960
```

A continuación se creará e instalará el sistema virtualizado. Utilizando el comando *virt-install* se configurará un sistema virtual que recibirá el nombre de "winvm", con una memoria asignada de 4GB, 2 CPU, disco el que se ha creado anteriormente de 40GB, tipo de hipervisor xen, tipo de virtualización completa y se utilizará la imagen ISO ubicada en */datos/xenconf/isos/Windows.iso* para instalar el sistema.

```
[root@xmserver02 ~]# virt-install --name winvm --memory 4096 \
  --vcpus 2 --disk /datos/xendata/winvm.img --virt-type xen -v \
  --cdrom /datos/xenconf/isos/Windows.iso --network bridge=xenbr0
```

Al ejecutar el comando se crea el nuevo dominio huésped y aparece una ventana cuyo contenido es su consola. En la figura 6.7 se observa el inicio de la instalación de un sistema windows que se corresponde con el sistema virtualizado creado recientemente.

Una vez instalado el sistema se pasará a integrarlo como un recurso dentro del clúster. Se empieza por apagar el sistema, ya sea realizando la operación de apagado dentro del huésped o mediante órdenes desde el *dom0*. Una forma de comprobar que el sitema se ha detenido es mediante la interfaz estándar de administración *virsh*. Si al listar los dominios activos no aparece, entonces está detenido.

Una vez detenido el dominio se procederá a crear el recurso dentro del clúster mediante su herramienta de administración.

```
[root@xmserver02 ~]# pcs resource create winvm VirtualDomain \
  hypervisor="xen:///system" config="/datos/xenconf/guests/winvm.xml" \
  migration_transport=tls remoteuri="xen://%n.example.org/system" \
  op start timeout="300s" op stop timeout="300s" op monitor \
  timeout="30s" interval="10" meta allow-migrate="true" priority="100" \
  op migrate_from interval="0" timeout="300s" op migrate_to interval="0" \
  timeout="300s"
```

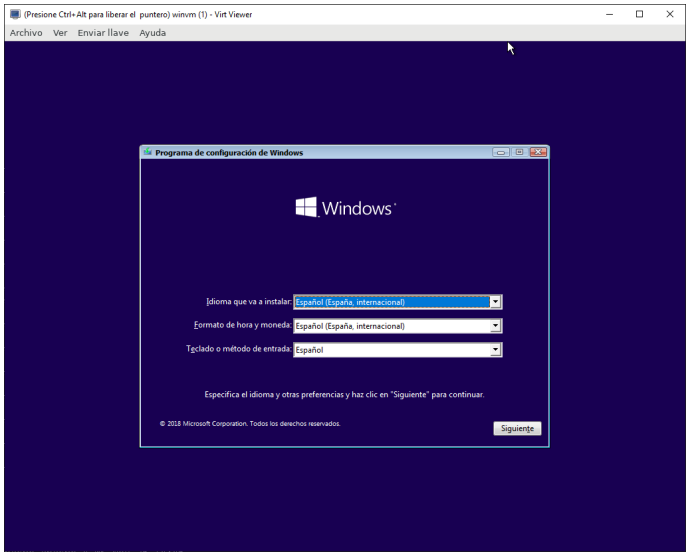

Figura 6.7: Contenido de la consola de un sistema virtualizado

Después de la creación del recurso, automáticamente se inicia el sistema virtual y queda incluído dentro de la alta disponibilidad. En la figura 6.8 se muestra el estado de los recursos del clúster después de la creación de este último. Como el sistema está arrancado también será accesible su consola.

```
[root@xmserver02 ~]# pcs status
Cluster name: xencluster
Stack: corosync
Current DC: xmserver01 (version 1.1.19-8.el7_6.4-c3c624ea3d) - partition with quorum
Last updated: Thu Nov 28 14:38:37 2019
Last change: Thu Nov 28 14:37:51 2019 by hacluster via crmd on xmserver01

3 nodes configured
10 resources configured

Online: [ xmserver01 xmserver02 xmserver03 ]

Full list of resources:

 winvm  (ocf::heartbeat:VirtualDomain): Started xmserver02
 fence_xm01     (stonith:fence_ipmilan):        Started xmserver03
 fence_xm02     (stonith:fence_ipmilan):        Started xmserver01
 fence_xm03     (stonith:fence_ipmilan):        Started xmserver02
 Clone Set: xencfg-clone [xencfg]
     Started: [ xmserver01 xmserver02 xmserver03 ]
 Clone Set: xendata-clone [xendata]
     Started: [ xmserver01 xmserver02 xmserver03 ]

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
[root@xmserver02 ~]#
```

Figura 6.8: Estado de los recursos del clúster tras la creación del nuevo sistema virtual

Para finalizar, el sistema virtual necesita el almacenamiento compartido para poder ejecutarse. Por lo tanto, se van a configurar dependencias entre el recurso que se acaba de crear y los recursos que proporcionan el almacenamiento compartido

```
[root@xmserver02 ~]# pcs constraint order set xencfg-clone xendata-clone \
  require-all=true winvm
```

CAPÍTULO 7
Pruebas de validación sobre la solución propuesta

En este capítulo se van a referir una serie de pruebas realizadas sobre los diferentes elementos de la infraestructura que van a validar el correcto funcionamiento de cada uno de los componentes.

7.1 Pruebas de la infraestructura de virtualización

Una vez instalada y configurada la capa de virtualización se va a realizar la prueba de crear un sistema virtualizado tanto utilizando virtualización completa como *paravirtualización*.

Para concluir esta sección, se comprobará la migración en caliente de los sistemas virtualizados

7.1.1. Creación de un sistema utilizando virtualización completa

Mediante la librería libvirt se crea un sitema virtualizado de ejemplo. En este caso el sistema virtualizado contendrá Windows 20012 R2 como sistema operativo.

Se utiliza la utilidad *virt-install* para crear sl sitema e iniciar la instalación del mismo

```
[root@xmserver02 ~]# virt-install --name dc1 --memory 4096 --vcpus 2 \
  --disk /datos/xendata/dc1.img --virt-type xen -v --cdrom \
  /datos/xenconf/isos/windows_server_2012r2.iso --network bridge=xenbr0
```

Como se ha instalado la utilidad virt-viewer anteriormente, después de proceder con la creación del sistema virtualizado aparecerá una ventana que mostrará la consola del sistema permitiendo así, la instalación del sistema.

Una vez finalizada la instalación del sistema se podrá acceder a la consola del sistema a través del comando *virt-viewer*. Un ejemplo de puede observar en la figura 6.7 del capítulo anterior.

7.1.2. Creación de un sistema utilizando *paravirtualización*

Esta vez se va a proceder a la instalación de un sistema utilizando *paravirtualización*, por lo que el sistema operativo huésped será Linux. Antes de proceder, para facilitar la tarea de instalación, se ha configurado un repositorio accesible a través del protocolo HTTP.

```
[root@xmserver03 ~]# virt-install --name www1 --memory 2048 --vcpus 2 \
  --virt-type xen -p --nographics --disk /datos/xendata/www1.img \
  --network bridge=xenbr0 -l http://10.172.21.46/CentOS -x \
  "text console=com1 utf8 console=hvc0 inst.ip=dhcp
  inst.repo=http://10.172.21.46/CentOS
  inst.ks=http://10.172.21.46/CentOS/ks.cfg"
```

En el caso de ser un sistema *paravirtualizado*, la herramienta conecta con la consola serie del sistema virtualizado, y mostrará una terminal serie para realizar la instalación. En el caso contemplado se ha realizado una instalación desatendida utilizando un archivo de configuración de *kickstart*, por lo que no se solicitará nada al usuario.

Una vez finalizada la instalación y reiniciado el sistema, se accederá a la consola del sistema utilizando la orden *virsh console <domain>*, con los que se presentará la terminal del sistema.

7.1.3. Migración entre nodos

La prueba siguiente trata de migrar el sistema virtualizado de un nodo a otro (en este caso manualmente ya que el sistema de alta disponibilidad aún está configurado).

Con el sistema arrancado y una interfaz de red configurada, se prueba la conectividad desde un cliente externo a la plataforma y se observa si hay pérdida de paquetes.

```
[root@xmserver02 ~]# virsh migrate --live lb1 \
  xen://xmserver01.example.org/system
```

En la figura 7.1 se observa una traza del comando *ping* ejecutándose durante la migración. Como se puede observar, aunque los tiempo se resienten durante la migración, no hay pérdida de paquetes, permitiendo la migración en caliente de los sistemas. Este componente será clave para el clúster.

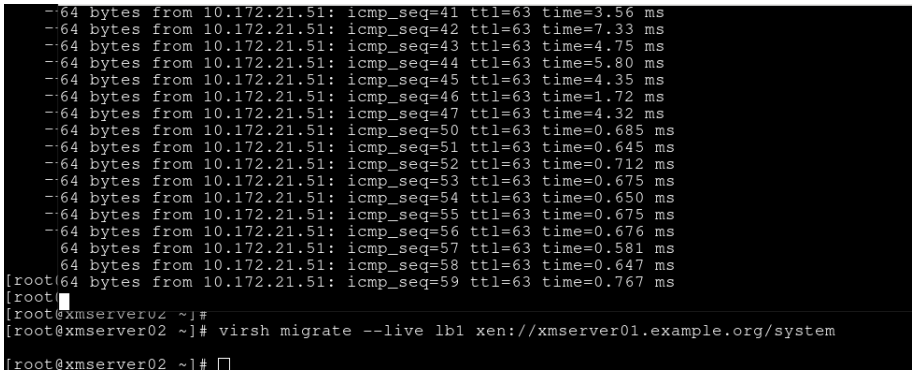

Figura 7.1: Estado de los recursos del clúster tras la creación del nuevo sistema virtual

7.2 Pruebas del almacenamiento compartido

El almacenamiento compartido debe proporcionar mecanismos de protección ante fallos hardware. En el caso que se presenta, en el que existen 3 nodos, el sistema debería ser capaz de tolerar la caída de uno de ellos sin que hayan consecuencias.

Se empieza comprobando que en verdad el almacenamiento es compartido. Por lo se crea un archivo en un nodo y se ve desde otro que se ha creado. Esta acción se puede

observar en la figura 7.2. Los dos nodos tienen un sistema de ficheros montado en /mnt y que el fichero que se crea en uno aparece en el otro.

Figura 7.2: Comprobación de un volumen compartido de Gluster.

Como segunda prueba, se comprueba la alta disponibilidad del servicio. El estado del volumen debe reflejar la caída simulada del nodo xmserver01 y continuar ofreciendo el servicio en los otros dos nodos. Como se ve en la figura 7.3 funciona correctamente.

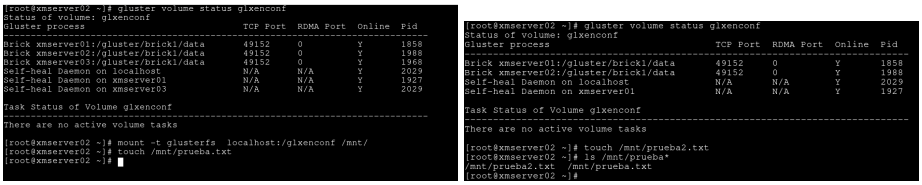

Figura 7.3: Comprobación de alta disponibilidad de un volumen de Gluster.

7.3 Pruebas de alta disponibilidad

Por último, se reflejan las pruebas de alta disponibilidad. En este apartado se comprueba la pila de software entera. Por lo tanto, uno de los sistemas virtuales se va a migrar entre nodos y se va a simular la caída de uno de ellos.

El primer paso es hacer la prueba de migrar un sistema en caliente. Hay un sistema ya configurado en el cúster llamado www1, un sistema *paravirtualizado*, y que como se puede observar en la figura 7.4 se migra correctamente. En las estadísticas totales de ping aparece que no ha habido pérdida de ningún paquete.

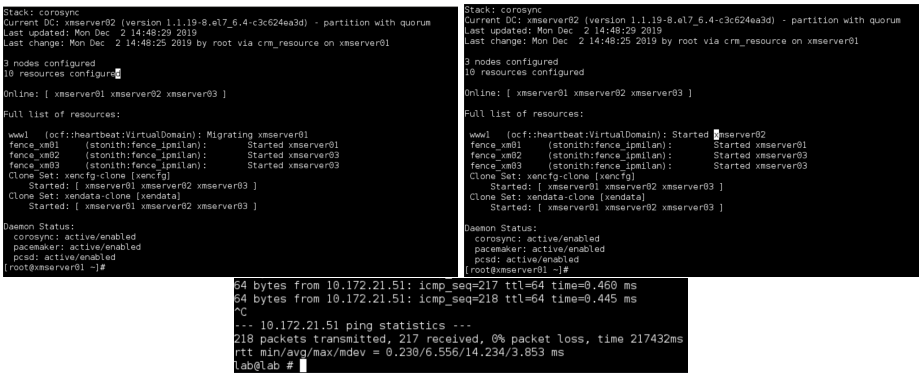

Figura 7.4: Muestra de migración de sistemas mediante clúster.

Como colofón final a este capítulo, se comprueba la reacción del clúster ante la caída del nodo que está ejecutando el sistema virtual. En la figura 7.5 se aprecia que el sistema virtual se detiene en el nodo que ha fallado. Una vez el clúster se da cuenta que un nodo ha fallado procede al vallado de este nodo, con lo que se asegura que no interferirá con las operaciones que realicen el resto de nodos del clúster. A continuación se arranca automáticamente en otro nodo el sistema virtual que se ejecutaba en el nodo caído.

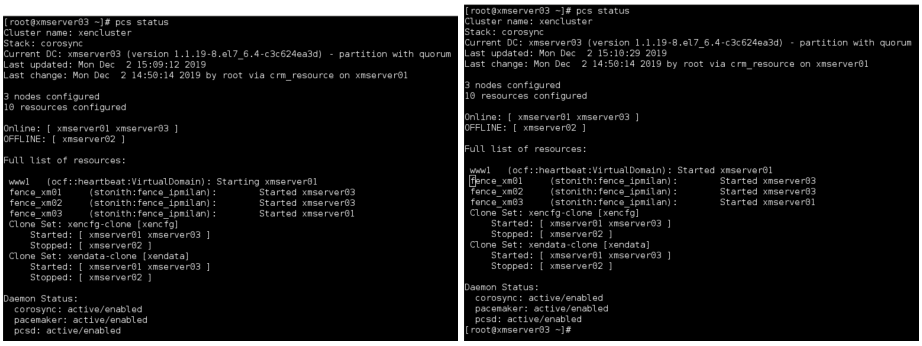

Figura 7.5: Muestra de caída de un nodo del clúster.

La recuperación automática del sistema virtualizado es relativamente rápida debido a que no necesita una intervención manual. Este mismo tiempo de recuperación se puede aplicar a otro tipo de servicios como, por ejemplo, un servidor web.

Después de realizar todas las pruebas descritas en este capítulo se puede asegurar que la plataforma desarrollada en este trabajo proporciona alta disponibilidad a los servicios de sistemas virtualizados, tal como se menciona en los objetivos del proyecto.

CAPÍTULO 8
Operación sobre la infraestructura

En el presente capítulo se van a describir una serie de operaciones básicas a realizar sobre la plataforma de virtualización. La finalidad de este capítulo es que el posible lector de este documento adquiera conocimientos de como gestionar la plataforma desarrollada.

8.1 Creación de una máquina virtual

En los pasos de creación de una máquina virtual se distinguen dos situaciones: si la técnica utilizada es virtualización completa o *paravirtualización*.

El primer paso es crear una imagen de disco del tamaño deseado mediante la herramienta *dd*. Si se desea crear una imagen de 40GB de tamaño se debe ejecutar el siguiente comando:

```
[root@xmserver02 ~]# dd if=/dev/zero of=/datos/xendata/winvm.img \
  bs=1M count=40960
```

En el caso de utilizar virtualización completa se debe especificar el parámetro v", y se puede utilizar una imagen ISO com fuente de instalación. Un ejemplo puede ser el siguiente:

```
[root@xmserver02 ~]# virt-install --name winvm --memory 4096 \
  --vcpus 2 --disk /datos/xendata/winvm.img --virt-type xen -v \
  --cdrom /datos/xenconf/isos/Windows.iso --network bridge=xenbr0
```

Los parámetros utilizados en el comando son los siguientes:

- –name winvm: especifica el nombre del sistema virtualizado. No tiene por qué coincidir con el nombre del sistema operativo que se ejecuta, aunque es recomendable.
- –memory 4096: cantidad de memoria asignada al dominio, expresada en MB. En el ejemplo se asignarán 4GB.
- –vcpus 2: cantidad de CPU virtuales de las que dispondrá el domino.
- –disk /datos/xendata/winvm.img: imagen de disco asignada. Previamente se debe haber creado.
- –virt-type xen: se indica que el hipervisor utilizado es Xen.
- -v: indica a libvirt que utilice virtualización completa.

- –cdrom /datos/xenconf/isos/Windows.iso: imagen ISO utilizada para instalar el sistema huésped.
- –network bridge=xenbr0: se configura una interfaz de red en modo *bridge* sobre la interfaz xenbr del *dom0*.

Si se desea que el sistema virtualizado utilice *paravirtualización* la creación del dominio es como sigue:

```
[root@xmserver03 ~]# virt-install --name www1 --memory 2048 --vcpus 2 \
  --virt-type xen -p --nographics --disk /datos/xendata/www1.img \
  --network bridge=xenbr0 -l http://10.172.21.46/CentOS -x \
  "text console=com1 utf8 console=hvc0 inst.ip=dhcp
  inst.repo=http://10.172.21.46/CentOS
  inst.ks=http://10.172.21.46/CentOS/ks.cfg"
```

Casi todas las opciones utilizadas anteriormente coinciden en la creación de este nuevo dominio. Las que se diferencia son las siguientes:

Una vez configurada instalado el sistema operativo sobre la máquina virtual se procede a dejarla apagada, ya sea a través del sistema operativo o mediante el sistema de virtualización. Es necesario que la máquina virtual se encuentre en este estado antes de agregar el recurso al clúster.

Se crea el recurso del clúster con el siguiente comando:

```
[root@xmserver01 ~]# pcs resource create www1 VirtualDomain \
  hypervisor="xen:///system" config="/datos/xenconf/guests/www1.xml" \
  migration_transport=tls remoteuri="xen://%n.example.org/system" \
  op start timeout="300s" op stop timeout="300s" op monitor \
  timeout="30s" interval="10" meta allow-migrate="true" priority="100" \
  op migrate_from interval="0" timeout="300s" op migrate_to interval="0" \
  timeout="300s"
```

La primera línea indica al clúster que añada el recurso www1, de tipo VirtualDomain al clúster. El resto de parámetros indican:

- hypervisor="xen:///system": el tipo de hipervisor a utilizar por libvirt.
- config=/datos/xenconf/guests/www1.xml": archivo de configuración del sistema virtualizado.
- migration_transport=tls: se utilizará tls en la comunicación entre los nodos en caso de migrar el servicio.
- remoteuri="xen:// %n.example.org/system": en el caso de migración de sistema, la URI de acceso al hipervisor. En el ejemplo mostrado %n es un campo que reemplaza por el nombre del nodo destino.
- El resto de opciones configuran otros pormenores del agente.

Después de crear el recurso en el clúster solo queda establecer relaciones con otros recursos que sean necesarios. En la plataforma todos los sistemas virtuales necesitan que los discos compartidos estén montados, por lo que se añadirán las dependencias:

```
[root@xmserver01 ~]# pcs constraint order set xencfg-clone xendata-clone \
  require-all=true www1
```

8.2 Eliminación de una máquina virtual

La eliminación de un sistema virtualizado se realiza en tres pasos: eliminar el recurso del clúster, eliminar el archivo de definición del dominio y eliminar el archivo de imagen.

Un ejemplo de comando para eliminar el recurso del clúster es el siguiene:

```
[root@xmserver01 ~]# pcs resource delete <recurso>
```

Donde recurso es el nombre del recurso (generalmente el nombre de la máquina virtual).

A continuación se eliminarán el archivo de configuración y el de imagen utilizados por el sistema:

```
[root@xmserver01 ~]# rm /datos/xenconf/guests/www1.xml \
  /datos/xendata/www1.img
```

8.3 Migración de una máquina virtual

Se recuerda que la gestión de las máquinas virtuales depende del clúser, por lo que para migrar un sistema virtualizado se ejecuta el comando de clśter que mueve recursos:

```
[root@xmserver01 ~]# pcs resource move <domain> <node>
```

Donde *domain* es el nombre del recurso y *node* el nodo destino de la migración del recurso.

8.4 Arrancar y parar una máquina virtual

Aunque hay diferentes formas de apagar una máquina virtual (desde el sistema huésped, desde la librería de vortualización) es muy importante que no se utilicen, ya que es el clúster quien debe parar y arrancar los diferentes sistemas virtuales. Si no se hace así el clúster puede interpretar un estado que no es correcto y ejecutar acciones que no debería.

Dicho esto, los comandos a utilizar para arrancar o parar recursos son los siguientes:

```
[root@xmserver01 ~]# pcs resource enable <domain>
[root@xmserver01 ~]# pcs resource disable <domain>
```

En ambos casos, *domain* en el nombre del recurso de clúster.

CAPÍTULO 9
Conclusión

Una vez expuesto el estado del arte y desarrollada la propuesta de plataforma de virtualización, tan solo queda reflejar las reflexiones y conclusiones.

Haciendo referencia a los objetivos, consideramos que se han cumplido íntegramente, ya que en el estado del arte se ofrecen diferentes productos de virtualización como punto de partida. Es un ejemplo la pila de software utilizada por los productos de VMware. Además, también se ofrecen ventajas, inconvenientes y estrategias que resuelven los diferentes problemas que se han detectado en la fase inicial de investigación.

Por último, en lo que se refiere a la propuesta, consideramos que es evidente que se cumple, ya que en el capítulo dedicado a los tests de funcionamiento se ha comprobado que la plataforma de virtualización diseñada cumple uno por uno todos los objetivos planteados. Cabe destacar que todos los objetivos son reales, concretos y alcanzables.

Con todo esto, también podemos destacar los puntos débiles de esta investigación. Por un lado se debe señalar que una parte de la tecnología utilizada en este trabajo de fin de grado no tiene actualmente la penetración en el mercado que antaño ha tenido. Podemos señalar como ejemplo el hipervisor de Xen, núcleo central de este trabajo. No obstante, la propuesta se ha puesto en práctica en su totalidad, quedando demostrada su eficacia en las pruebas de funcionamiento realizadas.

Así pues, consideramos que nuestra aportación sirve como punto de partida para poder implementar trabajos futuros que complementen la plataforma de virtualización desarrollada.

Para finalizar, podemos afirmar que elaborar este trabajo de fin de grado ha sido muy enriquecedor. La búsqueda de información realizada para poder desarrollar el estado del arte ha sido muy cuantiosa, por tanto se han podido adquirir muchos conocimientos que hasta este momento eran desconocidos. También investigar de primera mano las diferentes técnicas y arquitecturas de virtualización utilizadas por varios fabricantes hace que tengamos a nuestro alcance diferentes tecnologías que podemos utilizar en nuestro futuro como profesionales.

En conclusión, solo nos queda remarcar que este trabajo puede ser tomado como base de otros proyectos más ambiciosos y que aporten más funcionalidad al conjunto.

9.1 Trabajo futuros

Aunque en el documento se ha dedicado un capítulo a los procedimientos de administración a efectuar sobre la plataforma desarrollada sería unteresante crear un manual para los usuarios de la plataforma (con operaciones a realizar sobre los sistemas virtua-

les) y otro para administradores de la plataforma (con operaciones a realizar sobre la configuración de la plataforma).

Otra tarea pendiente de realizar es la creación de una interfaz web desde la que se pueda administrar los recursos que forman parte de la plataforma. Aunque en un principio era mi intención desarrollarla, el esfuerzo necesario para ello sumado a todo el trabajo ya realizado era excesivo, por lo que lo he considerado como futura ampliación.

Aunque teóricamente está soportado, no he tenido ocasión de realizar el montaje de la plataforma propuesta sobre una arquitectura ARM. Esto podría considerarse como otra ampliación futura al trabajo, y documentar aquellas variaciones que surjan en los manuales de usuario y administrador antes propuestos.

En este trabajo se trata de crear una infraestructura con sistemas virtualizados, pero no se ha tratado el tema de consolidación de servidores físicos sobre la plataforma de virtualización. Sería una posible ampliació que abriría el camino a la utilización de otro software de código abierto o posiblemente a su desarrollo.

Por último, otra posible ampliación a este trabajo sería implementación de la misma plataforma pero utilizando otros hipervisores, como por ejemplo KVM. Teóricamente esto sería posible ya que cada uno de los componentes que la forman son independientes entre sí, además que la interfaz libvirt está desarrollada bajo esta premisa. Esto podría hacer que llegara a un mayour número de usuarios que no quieren cambiar de hipervisor, y posiblemente llegar a otras arquitecturas de procesador como puede ser SPARC.

Bibliografía

[1] A Brief History of Cloud Computing Recuperado de https://www.ibm.com/cloud/blog/cloud-computing-history

[2] Acerca de AWS Reduperado de https://aws.amazon.com/es/about-aws/

[3] Barham, P et al. Xen and the Art of Virtualizationa 19th ACM Symposium on Operating Systems Principles, The Sagamore, New York, 2003

[4] Bitner, B. & Greenlee, S. (s.f.) *z/VM A Brief Review of Its 40 Year History* [Mensaje en un blog]. Recuperado de http://www.vm.ibm.com/vm40hist.pdf

[5] Documentación de Citrix Hypervisor 8.0 Reduperado de https://docs.citrix.com/es-es/citrix-hypervisor

[6] Drewanz, D. & Grimmer, L. (Diciembre 2012). *The Role of Oracle Solaris Zones and Linux Containers in a Virtualization Strategy* [Mensaje en un blog]. Recuperado de https://www.oracle.com/technical-resources/articles/it-infrastructure/admin-zones-containers-virtualization.html

[7] FreeBSD 4 Release notes Recuperado de https://www.freebsd.org/releases/4.0R/notes.html

[8] Howard, J. (26 de junio de 2008) *Hyper-V RTM announcement. Available today from the Microsoft Download Centre* [Mensaje en un blog]. Recuperado de https://blogs.technet.microsoft.com/jhoward/2008/06/26/hyper-v-rtm-announcement-available-today-from-the-microsoft-download-centre/

[9] Hypervisors Recuperado de https://www.ibm.com/cloud/learn/hypervisors

[10] Intel Timeline. A history of Innovation. Recuperado de https://www.intel.com/content/www/us/en/history/historic-timeline.html.

[11] Jones, M. (31 de mayo de 2009) *La anatomía de un hipervisor Linux* [Mensaje en un blog]. Recuperado de https://www.ibm.com/developerworks/ssa/library/l-hypervisor/index.html

[12] KVM: Kernel-based Virtual Machine Recuperado de http://lkml.iu.edu/hypermail/linux/kernel/0610.2/1369.html

[13] Levins, S. *Red Hat Enterprise Linux 7 High Availability Add-On Overview* Red HatCustomer Content Services, (s.f.)

[14] Levins, S. *Red Hat Enterprise Linux 7 Documentation. High availability add-on administration* Red HatCustomer Content Services, (s.f.)

[15] Levins, S. *Red Hat Enterprise Linux 7 Documentation. High availability add-on reference* Red HatCustomer Content Services, (s.f.)

[16] Libvirt FAQ Reduperado de https://wiki.libvirt.org/page/FAQ

[17] New to Xen Guide Recuperado de http://www-archive.xenproject.org/files/Marketing/NewtoXenGuide.pdf

[18] Oracle Corporation *Introducción a los entornos de virtualización de Oracle Solaris 11.1* (s.f.)

[19] Oracle Corporation *Oracle VM Concepts Guide for release 3.4* (s.f.)

[20] Página principal del proyecto Xenoserver. Recuperado de https://www.cl.cam.ac.uk/research/srg/netos/projects/archive/xeno/

[21] QEMU version 4.1.0 User Documentation Recuperado de https://qemu.weilnetz.de/doc/qemu-doc.html

[22] Red Hat Advances Virtualization Leadership with Qumranet, Inc. Acquisition Reduperado de https://www.redhat.com/en/about/press-releases/qumranet

[23] Redacción de Baquía. (14 de mayo de 2013) *Las tres formas de virtualizar un servidor* [Mensaje en un blog]. Recuperado de https://www.baquia.com/emprendedores/2013-05-13-las-tres-formas-de-virtualizar-un-servidor

[24] Release de QEMU Recuperado de https://www.winehq.org/pipermail/wine-devel/2003-March/015577.html

[25] Shah, A. (2 de noviembre de 2016) *Ten years of KVM* [Mensaje en un blog]. Recuperado de https://lwn.net/Articles/705160/

[26] Solaris 10 What's New Recuperado de https://docs.oracle.com/cd/E19253-01/817-0547/chapter1-1a/index.html

[27] The FreeBSD Documentation Project *FreeBSD Handbook* (s.f.)

[28] The Xen virtual machine monitor. Recuperado de https://www.cl.cam.ac.uk/research/srg/netos/projects/archive/xen/architecture.html

[29] Torres, G. (9 de julio de 2012). *Everything You Need to Know About the Intel Virtualization Technology* [Mensaje en un blog]. Recuperado de https://www.hardwaresecrets.com/everything-you-need-to-know-about-the-intel-virtualization-technology/2/

[30] Virtual machines Recuperado de https://www.ibm.com/cloud/learn/virtual-machines

[31] Virtualbox changelog. Obtenido desde Internet Archive. Recuperado de https://web.archive.org/web/20070214064527/http://www.virtualbox.org/wiki/Changelog

[32] Virtualization Recuperado de https://www.ibm.com/cloud/learn/virtualization-a-complete-guide

[33] VMware timeline Recuperado de https://www.vmware.com/timeline.html

[34] VMware VirtualCenter 1.0 Release Notes Recuperado de https://www.vmware.com/support/vc/doc/releasenotes_vc.html

[35] What's New in Oracle VM Server for SPARC Software Recuperado de https://www.oracle.com/technetwork/server-storage/vm/documentation/sparc-whatsnew-330281.html

[36] Xen Project *How Does Xen Work?* (s.f.)

[37] Xen Project *Why Xen?* (s.f.)

[38] Xen Project *What is Xen Hypervisor?* (s.f.)

[39] z/VM Overview Recuperado de https://www.vm.ibm.com/overview/

[40] ¿Qué es KVM? Recuperado de https://www.redhat.com/es/topics/virtualization/what-is-KVM