Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Nombre del Centro

Trabajo Fin de Grado

INTRODUCCIÓN A LA VIRTUALIZACIÓN

Alumno: David López Chica

Tutor: Prof. D. Víctor Manuel Rivas Santos
Dpto: Lenguajes y Sistemas Informáticos

Septiembre, 2016

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don **VICTOR MANUEL RIVAS SANTOS**, tutor del Proyecto Fin de Carrera titulado: **Introducción a la Virtualización**, que presenta **DAVID LÓPEZ CHICA**, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, SEPTIEMBRE de 2016

El alumno: Los tutores:

DAVID LÓPEZ CHICA VICTOR MANUEL RIVAS SANTOS

Contenido

1. Introducción

Son las ocho de la mañana. Como un día cualquiera, los empleados de esta empresa entran a trabajar. Las herramientas de trabajo que emplean dependen de varios servidores que hospedan diferentes servicios y gestionan la comunicación y el desarrollo de la empresa. Pero hoy es diferente. Uno de los servidores más críticos para la actividad empresarial presenta una avería inesperada.

Automáticamente, una serie de mecanismos entran en acción. El servicio que gestionaba el servidor, es migrado a un diferente sistema y la actividad continua. Ninguno de los empleados de la empresa ha notado nada, el servidor averiado es dado de baja y se planifica su sustitución.

¿Qué ha sucedido? La empresa está empleando servicios virtualizados. Debido a que estos servicios son independientes al servidor físico en el que se ejecuta, es posible migrarlo a un nuevo sistema rápidamente y resumir su operación. Todo gracias a procesos de recuperación de errores, facilidad de migración y sistemas redundantes que sólo es posible mediante el uso de la virtualización.

2. Qué es la virtualización

VMWARE: "La virtualización es una tecnología probada de software que permite ejecutar múltiples sistemas operativos y aplicaciones simultáneamente en un mismo servidor.".

Tech Target: "Virtualización es la creación de una versión virtual (en lugar de real) de algo, como un sistema operativo, un servidor, un dispositivo de almacenamiento o recursos de red.".

Virtualización, es la tecnología que nos permite crear sistemas (ya sea servidores, equipos de trabajo, dispositivos de almacenamiento, etc.) no atados a un servidor físico en específico. Esto nos proporciona varios beneficios, como son:

- **Consolidación de servicios**: Ejecutar distintos sistemas en un mismo servidor físico al mismo tiempo.
- **Migración**: Almacenar un sistema en un simple fichero, transportar a otro lugar, realizar copias de seguridad y ejecutar en diferentes servidores físicos.
- **Detección y recuperación de errores**: En caso de que un sistema deje de funcionar correctamente, detectar el problema y generar una nueva instancia que continue el trabajo.
- **Sistemas redundantes**: Permitir la comunicación entre distintas instancias del mismo servicio, para en caso de error de la instancia principal, retomar la actividad sin contratiempos.
- **Balance de carga**: Contando con varias instancias del mismo servicio, repartir la carga de trabajo entre distintos servidores.
- **Emulación**: Podemos emular diferente hardware, para poder ejecutar programas y servicios que no funcionarían en equipos corrientes.

En la situación que presentamos en el inicio de este trabajo, gracias a los sistemas redundantes y a la facilidad de migrar el servicio, un error inesperado que podría haber detenido la empresa durante un rango de varias horas hasta incluso semanas, dependiendo de la complejidad del sistema, es reducido a simples minutos.

Existen muchas más situaciones donde el uso de la virtualización es útil. Suponiendo que una empresa necesita de diferentes servicios hospedados en sus servidores, el caso común es que estos servicios se ejecutan de manera separada en diferentes servidores. Contando con servidores de correo electrónico, autentificación de usuarios, o aquellos que contienen servicios críticos para la actividad de la empresa, sucede que algunos de estos servidores cuenta con un uso bajo de sus recursos, ocupan espacio, gastan electrícidad y cuesta dinero y tiempo para mantenerlos en buen funcionamiento. En este caso, la virtualización puede consolidar todos estos distintos servicios en un menor número de servidores, distribuir el uso de recursos y aislar los procesos en diferentes instancias. De esta manera, conseguimos un uso mucho más eficiente de los recursos, y facilitamos la administración de los servicios.

Esta idea, concuerda con los ideales de lo que se conoce como **Green IT**, **Green Computing**, o en español, **Tecnologías Verdes**, que se refiere a la práctica de un uso eficiente de la tecnología, reduciendo el impacto ambiental, en forma de un menor empleo de electricidad. Hablaremos de ello con más detalle en un apartado posterior.

La virtualización no sólo permite consolidar servicios, sino también el almacenamiento de datos. De esta manera, se trivializa el acceso al espacio de almacenamiento desde diferentes servidores a un sistema dinámico de almacenamiento conocido como **Storage Pool**. De la misma manera que distribuímos los recursos hardware entre diferentes servicios, podemos distribuir el almacenamiento en función de la necesidad de cada servidor virtual que empleemos. Añadir más capacidad de almacenamiento se limita a añadir más discos duros a nuestro sistema de almacenamiento, que nuestros servicios accederán mediante recursos en red.

Junto a la consolidación de servidores, y la consolidación del espacio de almacenamiento, claramente reducimos en gran medida el uso de electricidad y espacio que estos sistemas necesitan, pero además, encontramos otra gran ventaja. La tarea de administrar estos servidores se facilita enormemente. Existen gran cantidad de herramientas para monitorizar sistemas virtualizados. La detección de errores, el mantenimiento del hardware, instalación de parches y software actualizado,

la creación de copias de seguridad, y la facilidad para automatizar tareas y crear sistemas redundantes, ha revolucionado la profesión del administrador de sistemas.

Con la virtualización...

- No es necesario instalar controladores de hardware para distintos servidores.
- Podemos mover un servicio virtualizado a un servidor físico diferente sin pausar su ejecución, también conocido como Migration on the Fly.
- Debido al punto anterior, podemos realizar mantenimiento de un servidor físico, moviendo los servicios virtualizados que esté ejecutando a un diferente servidor.
- Mayor facilidad para escalar los recursos disponibles, simplemente añadiendo más servidores, discos, memoria...
- Podemos monitorizar todos nuestros servidores desde una misma interfaz.
- En días de mayor actividad, podemos aumentar el límite de recursos que un servidor virtualizado puede emplear. Así mismo, en días menos ocupados, podemos reducir estos recursos y mantener los servidores físicos en desuso apagados.

3. Evolución de la virtualización a lo largo de la historia

En los años 60, IBM (**International Business Machines Corporation**) contaba con una gran variedad de sistemas, que evolucionaba radicalmente por generaciones. Esto creaba problemas debido al gran número de cambios que se producian en cada sistema. Esto, junto a la limitación que que los ordenadores de aquella época no eran capaces de hacer más de una tarea al mismo tiempo, provocó que IBM empezará a desarrollar sistemas retrocompatibles, siendo la computadora central S/360 su primer prototipo.

ILUSTRACIÓN 1: INGENIEROS EMPLEANDO LA IBM/360

Estos nuevos sistemas eran diseñados con la capacidad de ser empleados por un único usuario, que satisfacía las necesidades de la época. Esto cambió en 1963, cuando el MIT (**Instituto Tecnológico de Massachussetts**) anunció el **proyecto MAC** (**Mathematics and Computation**, posteriomente renombrado a **Multiple Access Computer**), destinado a distintos campos de investigación sobre ordenadores. Debido a este proyecto, se crearon nuevas necesidades para ordenadores capaces de dar soporte a varios usuarios simultáneos.

El MIT solicitó a empresas como IBM y GE (**General Electric**) el desarrollo de ordenadores que cumplieran sus demandas. IBM rechazó esta propuesta, debido a

que pensaban que no existia suficiente demanda para un producto así, pero se equivocaban. Compañias como Bell Labs también comenzaron a requerir sistemas multiusuario.

En respuesta a estas demandas, IBM diseñó el sistema **CP-40**. No fue un producto comercializado, pero originó lo que posteriormente se convertiría en el sistema **CP-67**, el primer sistema comercializado que empleaba tecnologías de virtualización para la computadora central IBM S/360.

CP (**Control Program**), era parte del sistema operativo **CP/CMS** (**Console Monitor System**). Su función en la computadora central era crear máquinas virtuales con las que los usuarios interactuaban independientemente.

La interacción del usuario con el sistema era una característica novedosa para IBM. Originalmente, los sistemas que desarrollaba IBM se limitaban a introducir un programa, que realizaría sus cálculos y devolvería los resultados, sin ningun método para interactuar con el proceso. Esto cambia con CP/CMS.

Por otro lado, el MIT en colaboración con GE y Bell Labs, afrontaron este problema, creando el sistema **MultiCS**, que en el futuro evolucionaría al sistema **Unix**. MultiCS repartia los recursos del sistema (tales como la memoria) entre los distintos usuarios[1]. En comparación con **MultiCS**, **CP** mediante sus máquinas virtuales permitía a sus usuarios tener su propio sistema operativo, lo cual simplificaba su funcionamiento.

La ventaja principal de las máquinas virtuales, frente a un sistema de repartición de recursos era que las máquinas virtuales permitian compartir los recursos de manera dinámica en función de los requisitos de cada usuario, en lugar de repartir equitativamente dichos recursos. Además, esto permitia aislar a los usuarios, ya que cada uno de ellos usaba su propio sistema operativo, independientemente de los demás.

IBM fue la primera compañia en implantar el concepto de Máquina Virtual en productos comercializados como fueron las computadoras centrales. Pero por otra parte, la mayoría de las compañias no empleaban computadoras centrales.

En 1987, Insignia Solutions enseñó al público lo que sería el prototipo de un emulador software llamado **SoftPC**. Este emulador permitia ejecutar aplicaciones DOS (**Disk Operating System**) en sistemas Unix, que eran mucho más baratos que aquellos que podían ejecutar las aplicaciones DOS nativamente. Esto fue el comienzo de lo que a día de hoy conocemos como **virtualización software**.

Insignia Solutions no se límito a ofrecer SoftPC a los usuarios de Unix. En 1989, se lanzó la versión **Mac OS** de SoftPC, ampliando el número de plataformas que soportaban el emulador. Además, no sólo era capaz de emular programas DOS, sino que se añadió la posibilidad de ejecutar programas Windows. A estas versiones de SoftPC se les denominó **SoftWindows**.

El éxito de SoftPC demostró a las empresas la gran demanda que existía para productos así. Apple, en 1997, creó Virtual PC, que comercializó a través de la compañia Connectix. Virtual PC permitia a los usuarios de Macintosh, ejecutar el sistema operativo Windows, a diferencia de SoftPC que permitia ejecutar aplicaciones.

En 1998, **VMware** fue fundada, y un año después comercializaron las primeras versiones de **VMware Workstation**. Más tarde, en 2001, lanzaron **GSX Server** y **ESX Server**. Ambos productos estaban destinados a la creación de máquinas virtuales, pero existia una importante diferencia: ESX Server no necesitaba de ningún sistema operativo sobre el cual funcionar, a diferencia de GSX Server, que empleaba Windows inicialmente. Son el origen de lo que conocemos como **hipervisores** de **Tipo 1** y **Tipo 2**.

VMware desde sus inicios, creció rápidamente como empresa, desarrollando más productos destinados a la virtualización. Por otro lado, empresas como Microsoft ampliaron sus esfuerzos para satisfacer esta nueva demanda. En 2003, Connectix fue comprada por Microsoft, y se empezaron a lanzar productos como **Microsoft Virtual PC** ó **Microsoft Virtual Server**. **Citrix** adquiere **XenSource** en 2007, lo que pronto pasaría a llamarse **Xenserver**.

3.1 Tabla cronológica

Año 1963	El MIT anuncia el proyecto MAC.
Año 1964	IBM anuncia la computadora central S/360, empleando el sistema CP-40/CMS.
Año 1965	En un esfuerzo en crear sistemas multiusuario, el MIT, GE, y Bell Labs comienzan el desarrollo de MultiCS, el precursor de Unix.
Año 1967	IBM introduce el S/360-67 (CP-67/CMS), comercializando por primera vez un sistema que empleaba técnicas de virtualización.
Año 1969	La primera versión comercializada de MultiCS sale a la luz.
Año 1987	Insignia introduce SoftPC, el primer emulador hardware capaz de ejecutar software DOS.
Año 1989	Sale a la venta la versión Mac de SoftPC y se añade la posibilidad de ejecutar software Windows.
Año 1997	Apple, a través de Connectix, comercializa VirtualPC.
Año 1998	VMware es fundada, la primera empresa con el objetivo principal de desarrollar productos de virtualización.
Año 1999	VMware introduce la primera versión comercial de VMware Workstation.
Año 2001	VMware GSX Server y VMware ESX Server salen a la luz.
Año 2003	Connectix es adquirida por Microsoft. XenSource lanza la primera versión de Xen.
Año 2007	Citrix adquiere Xensource.

3.2 Referencias

1. Robert C. Daley y Jack B. Dennis: «Virtual Memory, Processes, and Sharing in Multics».

4. Tipos de virtualización

Existen distintos tipos de virtualización que podemos emplear, dependiendo del objetivo que deseamos conseguir o los recursos virtualizados. Desde sistemas mediante emulación hardware, a consolidar el espacio de almacenamiento disponible, implementando una interfaz virtual que comunica los sistemas con los dispositivos de almacenamiento, veremos con más detalle los distintos técnicas existentes.

Antes de explicar las diferentes técnicas de virtualización, es necesario definir los tipos de hipervisor existentes. El **hipervisor** es una pieza de software que implementa las distintas técnicas de virtualización, coordinando el sistema huésped con las máquinas virtuales. Dependiendo de que manera es instalado el hipervisor, existirán dos tipos a considerar:

- **Hipervisor tipo 1** o **baremetal**: Son aquellos que se instalan directamente, sobre el equipo hardware, sin necesidad de un sistema operativo previamente instalado.
- **Hipervisor tipo 2** o **hosted**: Se instalan sobre un sistema operativo previamente instalado, como una pieza de software adicional a las ya instaladas en el sistema.

4.1 Empaquetado de aplicaciones

El empaquetado de aplicaciones se refiere al uso de virtualización para aislar una aplicación, dotándola de un conjunto de recursos virtuales con las que funcionar. De esta manera, la aplicación es incapaz de modificar sectores críticos u otras aplicaciones, favoreciendo la seguridad y la estabilidad del sistema. El método por el cual se realiza el empaquetado de aplicaciones varía según la implementación.

Por ejemplo, Docker implementa el empaquetado de aplicaciones mediante el uso de contenedores. El sistema operativo, la aplicación, dependencias, etcétera, son contenedores independientes y reutilizables, que junto a una imagen, unen y forman el sistema de archivos y recursos para el funcionamiento de la aplicación.

Este método emplea de un sistema operativo húesped, que es capaz de ejecutar multitud de instancias, y reutiliza de forma eficiente aquellas dependencias que son usadas por varias aplicaciones. Hablaremos de Docker en más detalle más adelante.

4.2 Emulación Hardware

La **emulación hardware** es una de las formas más conocidas de la virtualización hoy en día. Se trata de simular una interfaz hardware mediante software, con la que las máquinas virtuales se comunican, en lugar del hardware real. Dicho método necesita de un sistema huésped, con un software de virtualización llamado **hipervisor**. El hipervisor gestiona las llamadas al sistema que realizan las máquinas virtuales mediante la interfaz hardware virtualizada, y coordina el acceso al hardware real.

Las máquinas virtuales cuentan con su propio sistema operativo, sistema de ficheros, etcétera, y emplean una interfaz hardware estandarizada que el hipervisor presenta como un sistema real. Además, una de las ventajas de este método de virtualización es que las máquinas virtuales pueden ser guardadas en un archivo imagen y ser trasladadas a otro sistema o respaldadas en una copia aparte, otorgando una gran flexibilidad a la hora de implementar sistemas.

Además, existe una gran facilidad de escalar el modelo de virtualización en red. Empleando multitud de sistemas que cuentan con un hipervisor instalado es posible contar con un monitor central que supervise la actividad de los hipervisores, permitiendo la migración sin interrupción de las máquinas virtuales entre diferentes hipervisores. Por ejemplo, un producto así es **VMware vCenter Server** mediante su implementación llamada **vMotion**.

Uno de los asuntos a tratar es cómo el hipervisor coordina las llamadas al sistema que realizan las máquinas virtuales. Para ello debe modificar en tiempo de ejecución, las llamadas que el sistema operativo realiza al hardware, y en su lugar mandarlas al hipervisor, que resuelve la llamada empleando el hardware real en el cual está el hipervisor instalado. Se emplea para ello la **recompilación binaria**.

Esta técnica emula la arquitectura para la cual el sistema está instalado, usualmente, la arquitectura **x86** y más recientemente la **x86_64**, ya que son las más

populares. Otras arquitecturas menos comunes cuentan con menor soporte debido a una menor demanda.

Puesto que las máquinas virtuales funcionan completamente aisladas unas de otras, es totalmente posible contar con varios sistemas completamente diferentes, juntos en un mismo hipervisor. Debido a ello, este método de virtualización cuenta con multitud de casos de uso, como por ejemplo, testeo de nuevas aplicaciones en diferentes entornos, o consolidación de servicios en un mismo servidor.

Pero no todo son ventajas. La emulación hardware sufre de problemas como una reducción del rendimiento del sistema, debido a que los sistemas para emplear sus recursos deben coordinarse con el hipervisor. En general, esta pérdida es poco notable, alrededor de un 10%[2], e incluso menor en hardware más reciente.

Otra desventaja es la disponibilidad de controladores para hardware más reciente. Debido a que el hipervisor emplea una interfaz estándar para el hardware, debe ser capaz de emplear al completo la capacidad del hardware instalado. Esto presenta un problema cuando el hipervisor no cuenta con controladores actualizados para el hardware instalado, imposibilitando el uso.

4.3 Virtualización a nivel de SO

La virtualización a nivel de SO (**Sistema Operativo**) sigue un método similar a la emulación hardware. Se instala un software de virtualización que coordina los recursos de la máquina huésped, con las llamadas al sistema que uno o varias máquinas virtuales realizan. La diferencia radica que en lugar de presentar una interfaz hardware virtualizada, las máquinas virtuales usan los recursos reales del sistema mediante un espacio de nombres único, que produce identificadores enlazados al sistema de la máquina huésped.

Esta virtualización puede ser completa o parcial. Mientras que la virtualización completa, aisla completamente una máquina virtual en su propio espacio de nombres, el método parcial permite virtualizar sólo ciertos recursos, como por ejemplo, la memoria o el sistema de archivos. La utilidad **chroot** de Linux es un caso de virtualización parcial del sistema de archivos.

Como en la emulación hardware, las máquinas virtuales están aisladas unas de otras. A diferencia, existe una gran limitación con la virtualización a nivel de sistema operativo. Ya que las máquinas virtuales emplean una serie de librerias que comunican con el sistema huésped, dichas máquinas están limitadas a emplear el mismo sistema operativo que el sistema huésped. Un entorno Linux sólo puede virtualizar sistemas Linux.

Por otro lado, existen un buen número de ventajas:

- La virtualización de recursos reduce en menor medida el rendimiento del sistema, en comparación a la emulación hardware.
- Existe la posibilidad de virtualizar parte de los recursos, sólo reduciendo el rendimiento en el uso de dichos recursos.
- Técnicamente, el sistema húesped y las máquinas virtuales de dicho sistema emplean la misma instancia del sistema operativo, por lo tanto, sólo es necesario una licencia, reduciendo costes.

En resumen, si deseamos emplear virtualización y nos encontramos en el caso de que contamos con una serie de sistemas homogéneos, la virtualización a nivel de SO es una de las mejores elecciones posibles actualmente..

4.4 Paravirtualización

La paravirtualización es otro de los métodos que se emplea para virtualizar un sistema. Similar a la emulación hardware, la paravirtualización permite a las máquinas virtuales comunicarse con el hardware real mediante la comunicación con un hipervisor.

Mientras que en la emulación hardware el hipervisor presentaba una interfaz virtualizada del hardware, la paravirtualización permite a las máquinas virtuales emplear el hardware de manera coordinada. Las máquinas virtuales, en este caso, son conscientes de la existencia del hipervisor con el cual se coordinan, en lugar de que este intercepte las llamadas al sistema mediante la técnica de recompilación binaria que vimos anteriormente.

Para que esto sea posible, los sistemas operativos de las máquinas virtuales deben ser modificados previamente antes de ser instalados en el hipervisor. Esto es

relativamente fácil con aquellos sistemas de código abierto, que permiten recompilar el kernel con las modificaciones necesarias. Por esta razón, no es posible usar la paravirtualización en sistemas operativos como Microsoft Windows de manera completamente libre.

Microsoft adopta la paravirtualización de sus sistemas operativos mediante **Hyper-V**, su software de virtualización más reciente (evolución de Microsoft Virtual PC). A partir de Windows 7, Microsoft Windows es capaz de emplear técnicas de paravirtualización (llamadas **Enlightenments**) que comunican especificamente con el hipervisor de Hyper-V. Debido a que, en origen, estas modificaciones de Windows sólo son soportadas por Hyper-V, otros programas de virtualización ajenos a Microsoft, emulan el comportamiento de HyperV para dar soporte a Windows.

Las ventajas de la paravirtualización resultan aparentes. Debido a que permiten el acceso al hardware real del sistema huésped, no existe una limitación de los controladores que las máquinas virtuales pueden emplear, como existia en la emulación hardware. Además, el hipervisor se limita unicamente a coordinar los accesos al hardware, por lo que apenas afecta al rendimiento del sistema.

La paravirtualización también es capaz de emplear técnicas de memoria compartida, permitiendo una comunicación más rápida entre la máquina huésped y las máquinas virtuales. Trabajos recientes demuestran la ganancia en rendimiento empleando esta técnica[3], y las últimas versiones de Xen cuentan con soporte a dicha función, mediante el protocolo **vchan**[4].

Por si fuera poco, Intel y AMD, en los últimos 10 años, han introducido en sus chips, optimizaciones destinadas a la paravirtualización. Esto implica que algunas de las tareas que anteriormente realizaba el hipervisor, ahora pueden ser llevadas a cabo de una forma mucho más rápida por los actuales microprocesadores.

En resumen, la paravirtualización es una de las técnicas más novedosas de virtualización, rápida y con multitud de soporte.

4.5 Virtualización del almacenamiento

En los apartados anteriores hemos visto los diferentes tipos de virtualización que se pueden emplear en sistemas, con el objetivo de consolidar un número de sistemas

en una misma máquina compartiendo recursos, o bien, poder crear de forma rápida entornos de prueba sin necesidad de adquirir previamente una nueva máquina con una configuración hardware exacta.

En este apartado, podremos estudiar las distintas técnicas para consolidar uno de los recursos cuya demanda ha crecido exponencialmente en los últimos años: el espacio de almacenamiento. Con los siguientes métodos de virtualización, podremos crear una infraestructura de un tamaño más reducido que satisfaga las necesidades de nuestros sistemas.

4.5.1 Virtualización basada en el sistema huésped

La virtualización basada en el host se basa en el empleo de **administradores de volúmenes lógicos**, para crear un espacio de almacenamiento dinámico en el sistema. Esto se implementa virtualizando en el sistema unidades de almacenamiento que físicamente están formadas por conjuntos de sectores de uno o varios discos duros.

Para ello, el administrador de volúmenes lógicos organiza los sectores físicos de cada disco duro en varios conjuntos de extensiones físicas de espacio. Estas extensiones se mapean en extensiones lógicas, y un conjunto concatenado de extensiones lógicas constituye un volúmen lógico. Ampliar o reducir un volúmen lógico se trivializa a asignar más o menos extensiones lógicas a dicho volúmen.

Esto cuenta con un número de ventajas, como son:

- Una mayor libertad a la hora de organizar las unidades de almacenamiento. Por ejemplo, podemos crear un único volúmen, a partir de varios discos físicos.
- Dependiendo de la configuración de volúmen que empleamos, podemos implementar distintos niveles de un RAID tradicional (**redundant array of independent disks**), obteniendo funciones y ventajas como son:
- Aumento de la velocidad de escritura y lectura mediante operaciones concurrentes distribuyendo los datos entre distintos discos.
- Recuperación de errores, mediante replicación de datos o bits de paridad.

Sin embargo, este tipo de virtualización del almacenamiento cuenta con la limitación de que sólo es posible implementarla a nivel de sistema, imposibilitando la centralización del almacenamiento de varios sistemas.

4.5.2 Virtualización basada en el controlador de almacenamiento

Este tipo de virtualización consiste en el uso de controladores de almacenamiento que virtualizan el acceso al espacio disponible de los discos conectados al controlador, mientras se presenta al sistema al cual está conectado como una unidad de almacenamiento corriente. El controlador se ocupa de mantener una comunicación transparente entre el sistema y los discos, mientras permite funcionalidades como, por ejemplo, el **hot swapping**, que se refiere a la capacidad de añadir o extraer discos del controlador, sin necesidad de detener el sistema previamente.

Además, este método de virtualización es independiente del sistema operativo empleado, o las aplicaciones que hacen uso del almacenamiento, siendo incluso posible combinarlo con el uso de volúmenes lógicos, visto en la técnica de virtualización vista anteriormente.

Como desventaja, depende completamente de la implementación que realizan los fabricantes de dicha tecnología, pudiendo existir una variedad de limitaciones introducidas en la práctica.

4.5.3 Virtualización basada en sistemas en red

Hasta ahora, los métodos que hemos visto están limitados a soporte para un único sistema. Si lo que deseamos es consolidar todo el espacio de almacenamiento disponible para que varios sistemas puedan hacer uso de ello, necesitaremos un enfoque basado en red.

Una SAN (**Storage Area Network**, o en español, red de área de almacenamiento) consiste en una red dedicada (normalmente fibra debido a su alta velocidad) que comunica a los sistemas que emplean esta red con los sistemas de almacenamiento. Esta red permite interconectar sistemas independientemente del hardware, sistema operativo o incluso su localización física, a un espacio centralizado que hace uso de todos los dispositivos de almacenamiento disponibles.

Cuenta con todas las ventajas mencionadas en otros tipos de virtualización (replicación de datos, operaciones de escritura y lectura a mayor velocidad, mayor libertad para organizar el acceso a los dispositivos, etcétera), añadiendo además la posibilidad de poder monitorizar y administrar el uso del almacenamiento.

Algunas de las posibilidades existentes con la virtualización basada en sistemas en red son:

- Administrar los permisos que cada sistema tiene sobre el almacenamiento, como por ejemplo, que discos puede leer un sistema, en que carpetas escribir...
- Crear una política de uso que restringe cuanto espacio usado puede emplear un sistema en concreto.
- Permitir desconectar un sistema vulnerable de la red, impidiendo que un atacante pueda obtener o modificar la información almacenada.
- Si la red se extiende a varias sedes, crear réplicas de la información en cada sede, para un acceso más rápido a la información almacenada, similar a técnicas empleadas en bases de datos.
- Emplear un sistema de creación de copias de seguridad independiente, como por ejemplo, una máquina de unidades de cinta magnética que cuenta con una mayor compresión que las copias almacenadas en discos duros.

4.6 Referencias

2. Albert Reuther, Peter Michaleas, Andrew Prout, y Jeremy Kepne: «Database Systems on Virtual Machines: How Much do You Lose».

3. Jacob Faber Kloster, Jesper Kristensen y Arne Mejlholm: «On the Feasibility of Memory Sharing».

4. MirageOS Blog: «Vchan: Low-latency inter-VM communication channels».

5. Software de virtualización

Hoy en día, existe numerosas alternativas que implementan técnicas de virtualización para resolver problemas como reducir el espacio y electricidad empleadas en centros de procesamiento de datos. Dichas técnicas de virtualización permiten emular configuraciones específicas de hardware sin necesidad de obtener dicho hardware.

A continuación describimos las aplicaciones más utilizadas actualmente a la hora de afrontar un proyecto de virtualización de sistemas.

5.1 VMware

VMware posee un amplio catálogo de productos destinados a virtualización, no sólo limitandose a virtualización de sistemas mediante hipervisor, sino incluyendo aquellos que implementan virtualización del almacenamiento o virtualización de red. En este apartado describiremos sus productos más importantes.

5.1.1 VMware vSphere

VMware vSphere es sin ninguna duda el producto principal de VMware. Se trata de una suite de productos destinado a la virtualización de sistemas, mediante un hipervisor de tipo 1 o **bare-metal**. De esta manera, el rendimiento de las máquinas virtuales residentes es mayor.

Fácil de escalar y administrar, emplea **VMware ESXi** como hipervisor a instalar en los sistemas donde residen las máquinas virtuales, mientras **VMware vCenter** se emplea en aquellos sistemas que realizarán la administración y coordinación de las máquinas virtuales.

5.1.2 VMware ESXi

Se trata del hipervisor que VMware vSphere emplea en las máquinas huésped. Se trata de un hipervisor de tamaño reducido, lo cual consigue limitando el número de controladores que contiene, dando soporte únicamente a sistemas hardware especializados. Incluye además, su propio kernel (llamado **VMkernel**) basado en el kernel de Linux que actúa como base para el hipervisor.

Para configurar el hipervisor, tenemos dos alternativas. Por un lado, desde el propio sistema que contiene el hipervisor, podemos emplear las herramientas RCLI que emplean una consola de comandos para realizar una configuración básica. Por otro lado podemos administrar el hipervisor desde vCenter Server o vSphere Client, permitiendo una configuración completa del sistema.

Debido al reducido tamaño del hipervisor, y la posibilidad de emplear almacenamiento en red, el sistema huésped apenas necesita capacidad de almacenamiento local, sólo el necesario para contener al hipervisor. Es por ello que existe hardware especializado que contiene de fábrica versiones integradas de VMware ESXi, reduciendo el despligue de estos sistemas a simplemente conectarlos y configurarlos.

5.1.3 VMware vCenter Server

VMware vCenter trata del software de administración y monitorización de servidores empleado para configurar y coordinar aquellos sistemas que cuentan con el hipervisor ESXi instalado. Se instala sobre un equipo servidor que ejecute el sistema operativo Windows. Dicho servidor requiere de una base de datos sobre la que funcionar, aunque puede usar una integrada sólo recomendable para entornos pequeños.

5.1.4 VMware vSphere Client

Para administrar nuestro entorno virtual necesitamos un software cliente que nos permita conectarnos a nuestro vCenter Server o bien, a alguno de los servidores con ESXi. Para ello existe **vSphere Client**, un pequeño software que podemos instalar en cualquier escritorio Windows.

5.1.5 VMware Web Access

En el caso de que no pudieramos utilizar vSphere Client, o simplemente no deseamos instalarlo, existe la posibilidad de acceder a los sistemas ESXi o vCenter Server empleando **Web Access**, un servicio que podemos activar en dichos sistemas. Por lo tanto, existe la posibilidad de administrar un entorno vSphere en sistemas no compatibles con vSphere Client.

5.1.6 VMware Workstation

VMware también cuenta con una alternativa de virtualización, que emplea hipervisores de tipo 2 o **host-based**. VMware Workstation permite crear máquinas virtuales con una interfaz hardware emulada (como ya vimos en el apartado de Emulación Hardware).

5.2 Xen Project

Creado originalmente por la compañia XenSource Inc., que posteriormente fue adquirida por Citrix Systems. Xen Project es un hipervisor que implementa diferentes tipos de virtualización, desde emulación hardware a paravirtualizacion, así como combinaciones de ambos.

Xen Project puede instalarse como un hipervisor de tipo 1 llamado **Citrix XenServer** y dirigido a entornos empresariales. O bien como un hipervisor de tipo 2, soportado por numerosas distribuciones de Linux.

Para soportar técnicas de paravirtualización, Xen Project ejecuta inicialmente una máquina virtual privilegiada con acceso directo a la interfaz hardware. Esta maquina virtual coordina el acceso al hardware y a sus propias librerias para otras máquinas virtuales paravirtualizadas que hayamos creado, ya explicado en el apartado de paravirtualización.

5.3 Microsoft Hyper-V

Es el software de virtualización desarrollado por Microsoft, para la virtualización de sistemas. Integrado en Windows Server 2008 y posteriores, emplea la emulación hardware para crear máquinas virtuales con el SO Windows, aunque también es capaz de instalar algunas de las principales distribuciones de Linux.

Es capaz de emplear técnicas de paravirtualización en máquinas virtuales cuyo sistemas operativos cuenten con las modificaciones para ello, llamadas **Enlightenments**. De esta manera, el rendimiento de los sistemas virtualizados se maximiza.

5.4 Oracle VirtualBox

Se trata del software de virtualización desarrollado por Oracle Corporation. Como muchos otros, emplea técnicas de emulación hardware para la virtualización de sistemas en máquinas virtuales. Como diferencia, **VirtualBox** cuenta con soporte para plataformas que cuenten con sistemas operativos instalados como Windows, Linux, OS X o Solaris.

Su soporte no se limita a las plataformas en las que puede instalarse. Tambien soporta numerosos SO en máquinas virtuales como Haiku, o incluso OS X si nuestro sistema huésped es un Mac.

5.5 Docker

Docker es una aplicación de virtualización destinada al empaquetamiento de aplicaciones. Mediante un sistema de contenedores, cuyo contenido varía desde librerias y otras dependencias así como herramientas y código necesario para la aplicación empaquetada. De esta manera se obtiene un entorno único y estático para la aplicación empaquetada.

Docker sólo es capaz de empaquetar aplicaciones destinadas para Linux, mediante el uso de un kernel de Linux modificado y un uso de espacio de nombres único para cada aplicación empaquetada.

Además, existe la versión **Docker for Windows**, que nos permite ejecutar aplicaciones ya empaquetadas mediante el sistema de containerización, efectivamente soportando el uso de aplicaciones Linux en Windows.

5.6 Linux VServer

Linux VServer emplea virtualización a nivel de sistema operativo. Para ello, se modifica el kernel de Linux de manera que sea capaz de soportar varios sistemas al mismo tiempo. Cada sistema corre bajo una máquina virtual que se compone del kernel compartido de Linux, junto a un subconjunto de recursos únicos a cada máquina virtual, todo bajo un espacio de nombres diferente a cada sistema.

De esta manera, cada máquina virtual contiene sus propios procesos aislados de los demás, mientras que el kernel coordina el uso de los recursos de la máquina,

asignando prioridades y una cola para las llamadas al sistema. Esta implementación de virtualización además permite asignar una cuota de uso para los recursos hardware, limitando el tiempo de cómputo usable, uso de memoria, etcétera, evitando que una de las máquinas sea capaz de sobreutilizar el sistema, inutilizando el resto de las máquinas hospedadas.

6. Ventajas y desventajas de la virtualización

Previamente hemos enumerado algunas de las ventajas y desventajas de diferentes técnicas de virtualización. No todas estas características se aplican, y dependen mucho de la implementación empleada, pero podemos obtener a grandes rasgos una enumeración de las ventajas que podemos obtener, así como ciertas desventajas existentes en entornos virtualizados, más concretamente con la licenciación.

6.1 Ventajas

6.1.1 Consolidación de recursos

Prácticamente todas las técnicas de virtualización nos ofrece la posibilidad de reducir el número de sistemas físicos empleados para nuestro entorno de trabajo. Ya sea instalando un número de máquinas virtuales en un mismo sístema, en lugar de en diferentes equipos, así como consolidar el espacio de almacenamiento a uno o varios equipos especializados, como controladores RAID, sistemas SAN o sistemas NAS, previamente explicados.

De esta manera, el espacio físico que nuestro entorno necesita disminuye en gran manera. Con ello, también reducimos la factura de la electricidad, y aprovechamos eficientemente el hardware del que disponemos. En lugar de la antigua idea obsoleta de un servidor por cada servicio, que se justificaba en la necesidad de aislar los distintos servicios para un mantenimiento más sencillo, empleamos servidores que contienen todos estos distintos servicios en una misma máquina, y aislamos los servicios mediante técnicas de virtualización.

En resumen, mantenemos el necesario aislamiento de servicios mientras reducimos el número de servidores utilizados, reduciendo el espacio y la electricidad necesaria para mantener estos sistemas. Un menor número de sistemas reduce además la potencia necesaria del sistema de aire acondicionado para mantener estos servidores en una temperatura segura, reduciendo aún más el uso de energia. Conseguimos un entorno mucho más eficiente y respetuoso con el medioambiente, abrazando la filosofía de las **Tecnologías Verdes**.

6.1.2 Escalabilidad asequible

Si previamente deseábamos implementar un nuevo servicio, era necesario calcular los requerimientos, comprando el hardware que más se ajustase. En ocasiones, el uso de estos servicios excede nuestras expectativas y es necesario adquirir capacidad extra. Esto desarrolla en dos situaciones:

- Consideramos a la hora de comprar el equipo necesario, la posibilidad de en un futuro, nuestro servicio requiera de capacidad extra, por lo que compramos equipo con más potencia de lo necesario. Esta capacidad extra no se emplea inicialmente, por lo que estamos desaprovechando el servidor.
- Compramos equipo ajustado a los requerimientos actuales, pero corremos el riesgo de que los requerimientos en un futuro sean mayores, y no seamos capaces de satisfacer las nuevas necesidades, por lo que debemos desechar el sistema actual y comprar equipamiento más potente.

En resumen, nos encontramos con un dilema que desafía nuestra capacidad de previsión. En esta situación, si contamos con el uso de técnicas de virtualización, nuestro problema se limita a comprar sistemas capaces de hospedar numerosas máquinas virtuales. El uso de recursos se puede distribuir entre las distintas máquinas virtuales (que denominamos **balance de carga**), por lo que no existe la palabra desuso en el diccionario de la virtualización.

En caso de que nuestro sistema huésped no tenga la capacidad necesaria para soportar todos los servicios que queremos implementar, la solución se limita a comprar un segundo equipo, distribuyendo las máquinas virtuales entre ambos equipos. Existe software que centraliza la administración de varios sistemas huésped, coordinando el uso de recursos de cada sistema, migrando máquinas virtuales a diferentes sistemas, implementando replicación de servicios, eliminando problemas como los puntos únicos de fallo (como sería el caso de que nuestro único equipo huésped se averiase, deteniendo todos nuestros servicios de manera catastrófica).

6.1.3 Resistencia ante errores

Contamos con una serie de técnicas de virtualización que nos protege ante posibles puntos únicos de fallo, así como algunos inconvenientes que se presentan cuando un sistema falla.

- **Migración de servicios entre diferentes servidores**. Esto es, la posibilidad de mover una máquina virtual de un sistema huésped a otro diferente, de una manera rápida y reduciendo el tiempo fuera de servicio necesario para dicho cambio.
- **Replicación de servicios**. Esto es, la posibilidad de sincronizar dos o más maquinas virtuales que implementan el mismo servicio, reflejando el estado de la memoria de la máquina virtual en uso. Esto nos permite migrar el servicio a otro sistema inmediatamente, ya sea por decisión nuestra, o en caso de que el sistema principal falle, con lo que una de las máquinas virtuales reflejadas toma el control del servicio en pocos segundos.
- **Formato simple para almacenar máquinas virtuales, en fichero imagen**. De esta manera, el transporte de las máquinas virtuales, y la creación de copias de seguridad se simplifica.

6.1.4 Mayor compatibilidad

Con la posibilidad de emular hardware, la necesidad de comprar equipo con diferentes combinaciones de hardware para dar soporte a distintos sistemas operativos o servicios, se reduce a emular dicho hardware en un equipo estándar. Presentamos además interfaces hardware estandarizadas que facilitan la migración de las máquinas virtuales implementadas a sistemas completamente diferentes, sin necesidad de reinstalar controladores, o alterar la configuración de nuestro sistema virtualizado.

6.1.5 Despliegue rápido de servicios

Anteriormente, si deseabamos desplegar un servicio nuevo en nuestro entorno de trabajo, era necesario disponer del equipo necesario, instalar el sistema operativo base, instalar el servicio, configurarlo y ponerlo en servicio.

Con un entorno virtualizado, el tiempo necesario para implementar un servicio se reduce de manera significativa. Por ejemplo, nos podemos limitar a clonar la imagen de una máquina virtual con el sistema operativo deseado ya instalado y configurado, instalar nuestro servicio y la configuración necesaria y la puesta en servicio. Los pasos previos a la instalación del servicio son completamente reducidos a copiar una imagen.

Además, una vez el servicio está configurado, podemos clonar la imagen de dicho servicio ya configurado, por lo que si debemos implementar este servicio en varios equipos, simplemente debemos copiar la imagen a dichos equipos, sin necesidad de repetir la misma rutina varias veces.

Con esta ventaja, se nos abre un nuevo mundo de posibilidades. Podemos crear entornos de prueba en pocos minutos, con la configuración hardware (proporcionada por la emulación hardware) y software (mediante una imagen virtual) que deseamos. Podemos crear instantáneas del estado de una máquina virtual, y dado el caso de que una de nuestras modificaciones volviera el sistema inestable, volver al estado anterior en menos de un minuto. Desechar la máquina virtual está al alcance de unos pocos clics de ratón.

6.2 Desventajas

No todo es perfecto en el mundo de la virtualización. Existen diferentes obstáculos que pueden frenar la capacidad que el uso de virtualización ofrece.

6.2.1 Dificultad en el cambio de panorama

Crear un entorno virtual desde cero es simple. Obtenemos el equipo necesario, creamos y configuramos máquinas virtuales donde implementamos los servicios que necesitamos en nuestro entorno de trabajo. El problema radica cuando nuestra situación actual cuenta con un entorno de producción de varios sistemas físicos con sus servicios implementados.

Realizar la migración a sistemas virtualizados requiere tiempo e inversión económica, así como la necesidad de detener los servicios que estamos trasladando a una máquina virtual, requiriendo un tiempo de mantenimiento durante el cual realizar la migración.

La dificultad de iniciar un proyecto de virtualización es frenada por administraciones que no ven justificada la necesidad de cambiar un sistema que ya funciona, por uno teóricamente más eficiente. Es necesario por tanto, un esfuerzo en explicar las ventajas que la virtualización nos aporta.

La reducción de la factura eléctrica es uno de los mayores argumentos que podemos emplear, siendo notable en entornos con un gran número de servidores.

Reducir las horas de trabajo necesarias para el mantenimiento de sistemas, que podemos emplear en su lugar a innovación de sistemas y otras tareas más importantes, es otro buen argumento. En resumen, justificar el cambio en términos de inversión y horas de trabajo mejor empleadas, reduciendo costes y aumentando la productividad.

6.2.2 El problema de licencias

Previamente, si deseábamos usar un producto software, adquiriamos una licencia que nos permitia emplear dicho software en un número limitados de equipos, dependiendo de las condiciones de dicha licencia. Actualmente, instalamos software en máquinas virtuales hospedadas en uno o varios sistemas físicos.

Esto presenta un nuevo paronama que los desarrolladores de software no terminan de encajar. Por ejemplo, suponemos que deseamos instalar una base de datos Oracle que funcione sobre un único sistema. Normalmente el coste de la licencia depende del número de núcleos que dicho sistema emplea. Suponiendo que empleamos una máquina virtual a un sistema físico con 4 núcleos, sólo deberiamos costear una licencia para dicha configuración.

En la práctica, esto no sucede así. Debido a la posibilidad de migrar la máquina virtual a diferentes sistemas, con diferentes configuraciónes hardware, sobretodo en el número de núcleos disponibles, o funcionalidades como la replicación de servicios, que ejecutaría varias instancias de esta base de datos concurrentemente, dificultan la capacidad de calcular el coste de las licencias, así como las condiciones a las que nos debemos someter para el uso del software.

Por ejemplo, Oracle y su concesión de licencias en un entorno virtualizado requiere pagar en función del número de sistemas huésped que ejecutan o pueden ejecutar máquinas virtuales con software Oracle, así como el número de procesadores en dichos sistemas. Esto significa, que si queremos contar con una máquina virtual con software Oracle, y contar con la posibilidad de migrar dicha máquina a un segundo sistema en el caso de que el primer sistema falle, debemos pagar una licencia para dos sistemas huésped, así como pagar el número de procesadores de ambos sistemas[5].

Esto nos presenta con necesidades artificiales de limitar los sistemas físicos autorizados a hospedar software Oracle, para satisfacer las condiciones de uso de la licencia. Este problema no se limita a Oracle, y se extiende a otros desarrolladores de software diferentes.

6.3 Referencias

5. VMware: «Understanding Oracle Certification, Support and Licensing for VMware Environments».

7. Proyecto de virtualización con VMware Workstation

Para experimentar las diferentes técnicas de virtualización, desarrollaremos un proyecto de virtualización con tres máquinas virtuales que coexistiran en un mismo equipo. Para ello, emplearemos una versión de prueba gratuita de VMware Workstation, catalogado como hipervisor de tipo 2.

Para cada máquina virtual, se asignará inicialmente un núcleo de procesador, 1024 MB de memoria RAM, y un disco de 20 GBs. Debido a las características de cada máquina virtual, existirán excepciones a esta configuración que se documentarán debidamente.

Las tres máquinas virtuales que implementaremos serán las siguientes:

- Un **servidor dedicado de Don't Starve Together**, un videojuego online desarrollado por Klei Entertainment. El sistema operativo base empleado será **GNU/Linux**, distribución **Arch**.
- Un **servidor web Nginx**. El sistema operativo es **GNU/Linux**, distribución **CentOS 7**.
- Un **servidor NAS con OpenMediaVault**. El sistema operativo es **GNU/Linux**, distribución **Debian 7 Wheezy**.

Las razones por las que se ha elegido las diferentes distribuciones para estos servicios son:

- **Arch** es una distribución minimalista de GNU/Linux. Reduciendo el software empleado al estrictamente necesario, permitimos que la máquina virtual pueda dedicar la totalidad de sus recursos al servicio implementado, en este caso, un servidor dedicado de videojuegos. Los jugadores de videojuegos online son un sector muy sensible a fallos y ralentizaciones del servidor, por lo que es importante que se asigne los suficientes recursos y se empleen eficientemente.

- **CentOS** es una de las distribuciones de GNU/Linux centradas en un entorno empresarial. Basado en RHEL (**Red Hat Enterprise Linux**), ofrece las mismas características de forma gratuita. Existe especial atención en el apartado seguridad, que es crítico para los servidores web, expuestos a posibles accesos no autorizados por parte de usuarios maliciosos.
- La mayor parte de software NAS ofrece distribuciones de Linux ya preparadas con el software necesario instalado. En el caso de OpenMediaVault, se ofrece una imagen de **Debian 7 Wheezy** preconfigurada para ahorrar tiempo y crear un entorno estándar para los usuarios de dicho software, que facilita el soporte.

Existen diferencias en los recursos asignados a cada máquina virtual, documentados a continuación:

- **Servidor Arch**: Para hospedar el servidor dedicado de DST, es necesario un mínimo de 2048 MBs de RAM para ofrecer todas las características que el videojuego ofrece. No obstante, es posible emplear únicamente los 1024 MBs originales de memoria si limitamos el servidor a una única instancia. Existen más detalles sobre este problema en la documentación correspondiente.
- **Servidor NAS**: Debido a la naturaleza del servidor NAS, es razonable que empleemos mayor capacidad de almacenamiento para dicho servicio. Concretamente, es necesario un segundo disco para guardar los distintos archivos que almacenemos en el servidor. Por otra parte, podemos limitar el primer disco donde se instalará OpenMediaVault a 2 GBs de almacenamiento.

7.1 Manual de uso básico de VMware Workstation

Podemos descargar la versión de prueba de VMware Workstation en su página web (http://www.vmware.com/products/workstation.html). La instalación es sencilla, y sólo debemos hacer una observación sobre el requisito de licencias. Ya que en principio sólo deseamos emplear la versión de prueba, no es necesario que introduzcamos una licencia, limitándonos a usar el producto durante 30 días.

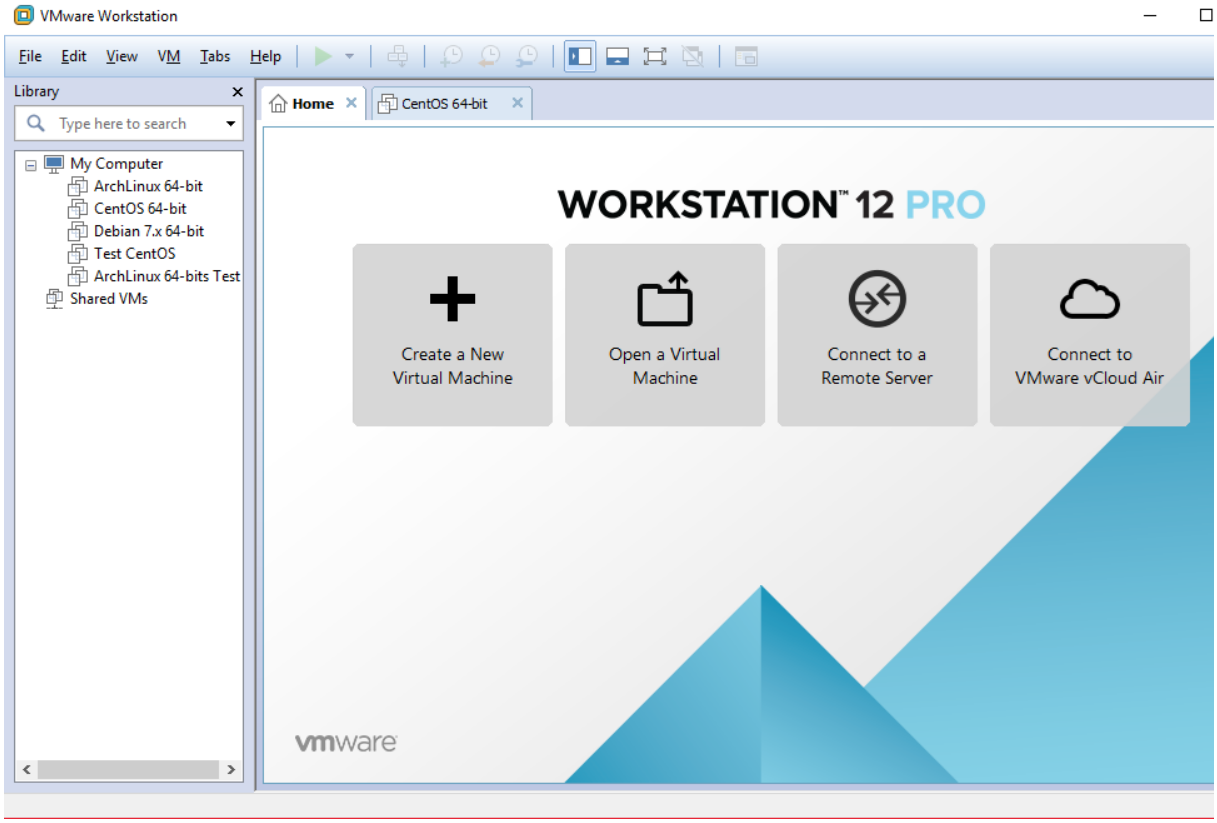


ILUSTRACIÓN 2: INTERFAZ PRINCIPAL DE VMWARE WORKSTATION

Crearemos una máquina virtual, presionando sobre "Create a New Virtual Machine" en la interfaz, o bien, accediendo al menú File. Podremos elegir a continuación una configuración típica o personalizada. Mientras que la configuración típica nos ahorra tiempo, no nos permite editar la asignación de memoria de la máquina virtual, entre otros detalles, por lo que emplearemos una configuración personalizada.

Detallamos los pasos a seguir:

1) Se nos pedirá elegir hasta que versión de Workstation soportar. Elegiremos la última versión (Workstation 12.0) para poder emplear todas las funcionalidades ofrecidas.

2) Necesitamos indicar donde se encuentran los archivos de instalación del sistema operativo que contendrá la máquina virtual.

3) Debemos indicar que sistema operativo empleará la máquina virtual, para adaptar los siguientes pasos de configuración al entorno elegido. Para Debian y CentOS, elegiremos las opciones correspondientes, para Arch, elegiremos la opción "Other Linux 3.x kernel 64-bit".

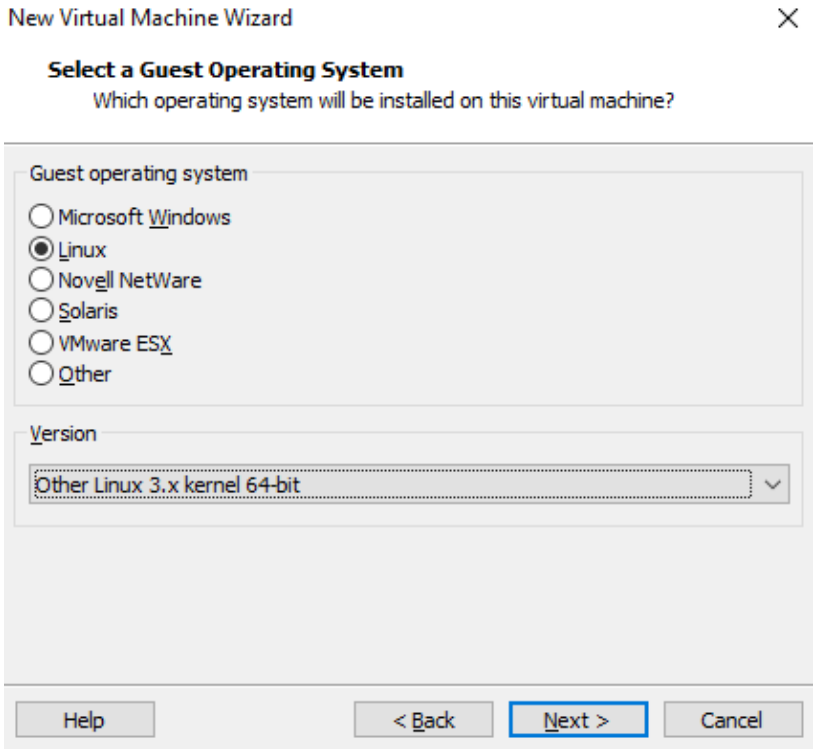


ILUSTRACIÓN 3: ELECCIÓN DEL SISTEMA OPERATIVO

4) Elegiremos el nombre de la máquina virtual y la carpeta donde se guardarán los archivos asociados (discos virtuales principalmente).

5) A continuación elegimos el número de procesadores y núcleos asignados a la máquina virtual. Para la configuración que estamos empleando en este proyecto, todas las máquinas contarán con un único núcleo y procesador.

6) Asignamos la memoria que la máquina virtual empleará. La configuración para este proyecto es de 1024 MBs, excepto para Arch que empleará 2048 MBs.

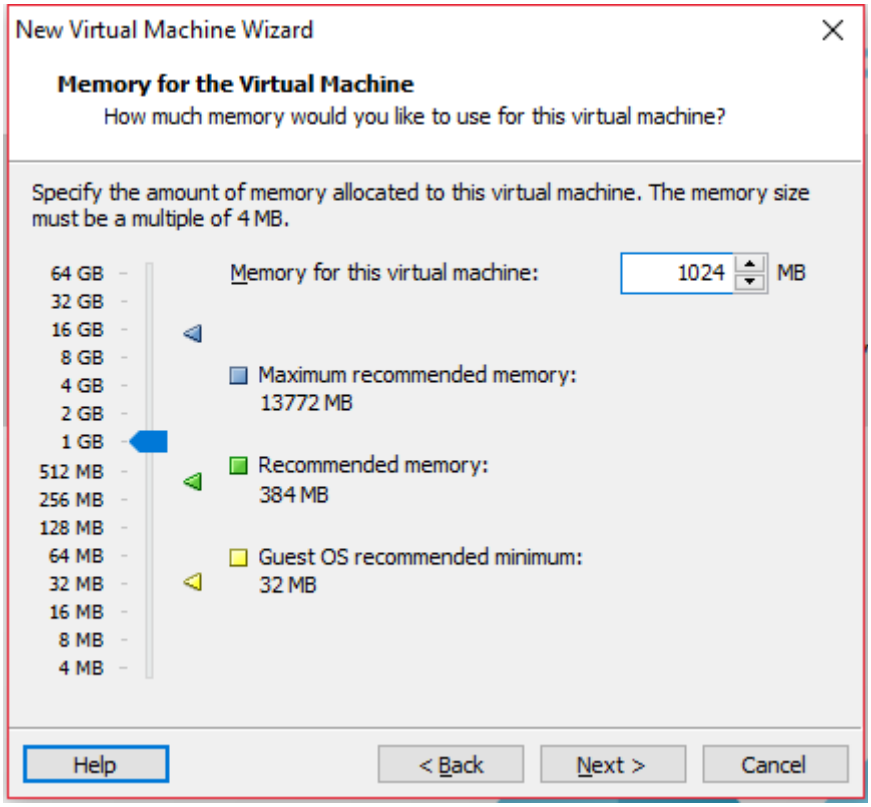


ILUSTRACIÓN 4: ASIGNACIÓN DE MEMORIA PARA LA MÁQUINA VIRTUAL

7) A continuación configuramos el tipo de red que empleará la máquina virtual. Debido a que se trata de servidores accesibles a través de Internet, está en nuestro interés poder configurarlos como si se trataran de sistemas individualmente y conectados a la red local, con su propia IP. Por tanto, elegiremos la opción "Use bridged networking".

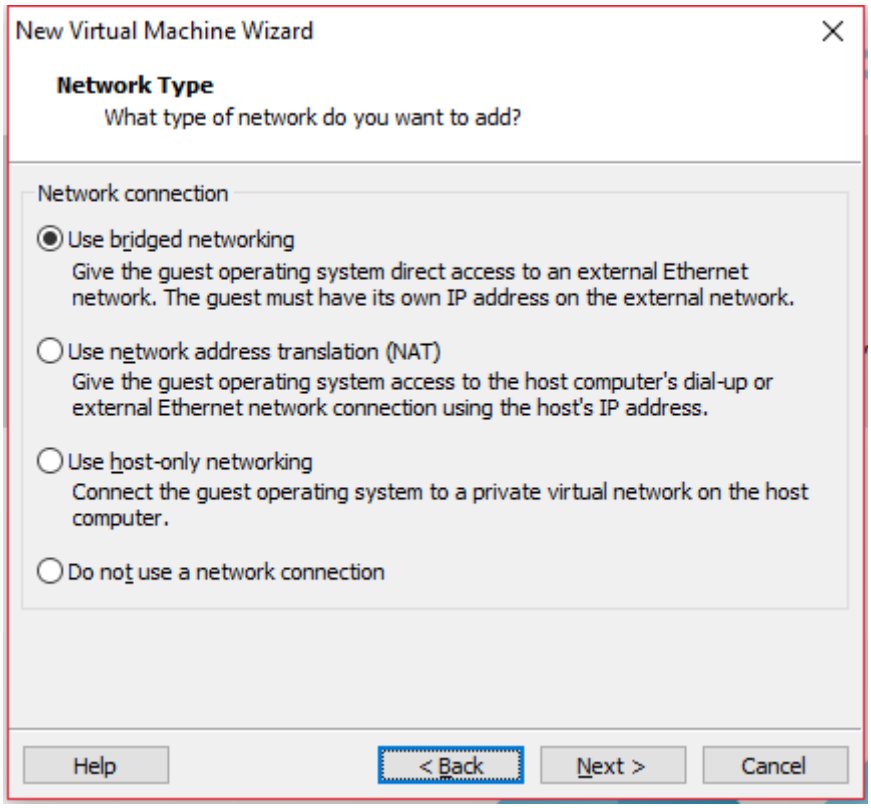


ILUSTRACIÓN 5: ELECCIÓN DE LA INTERFAZ RED A UTILIZAR

8) Seleccionamos LSI Logic como controlador I/O para nuestras máquinas virtuales. Es la opción de mayor rendimiento, y está soportada por las tres máquinas virtuales que emplearemos.

9) En el tipo de disco a emplear, elegiremos la opción recomendada SCSI. Podremos crear discos adicionales posteriormente. Esta opción se refiere al disco inicial donde se instalará el sistema operativo.

10) Se nos preguntará si deseamos crear un disco virtual o elegir uno previamente creado. También tenemos la opción de emplear un disco conectado al sistema huésped (que es una opción a considerar en un entorno en el cual contamos con controladores de almacenamiento en red). En nuestro caso, crearemos un nuevo disco virtual para cada máquina, que se almacenará en un fichero en el sistema huésped.

11) Debemos elegir cuanta capacidad asignar al disco virtual, y si deseamos almacenarlo en uno o varios ficheros. Elegiremos una capacidad de 20 GBs para las máquinas Arch y CentOS, y 2 GBs para Debian. Además, marcaremos la opción de asignar toda la capacidad del disco inmediatamente, y almacenarlo en un único fichero.

12) Elegimos el nombre del fichero donde se almacenará el disco virtual, así como la ruta. La opción por defecto es aceptable.

13) Finalmente, se nos presenta una vista resumen de la configuración que hemos elegido para la máquina virtual, así como la posibilidad de personalizar mas detalladamente algunas opciones hardware. Finalizaremos la configuración.

14) Existe un paso adicional para la máquina virtual Debian. Una vez finalizada la configuración, haremos clic sobre la máquina virtual, que nos presentará una vista resumen del estado de la máquina virtual. Haremos clic sobre la opción "Edit virtual machine settings". Podremos observar el hardware instalado en dicha máquina virtual. Añadiremos un nuevo disco duro a la máquina mediante la opción "Add...". El disco duro a crear será de tipo SCSI, con una capacidad de 18 GBs y la opción Independent marcada. No existen más pasos adicionales.

15) Una vez realizado estos pasos con esta serie de pasos para cada máquina virtual, podremos iniciarlas seleccionando la máquina virtual, y presionando sobre la opción "Power on this virtual machine". Los detalles sobre la instalación y configuración de cada máquina virtual podrán consultarse en la documentación proporcionada a continuación.

8. Don't Starve Together - Introducción

Desarrollado por Klei Entertainment, DST es la versión online de su anterior juego Don't Starve. Varios jugadores podrán compartir la misma experiencia de supervivencia en un mundo hostil. La experiencia multijugador se implementa mediante servidores dedicados que permiten un número limitado de usuarios jugando en el mismo mundo.

A continuación veremos como instalar y configurar un servidor dedicado de DST, de manera que hasta 10 jugadores puedan jugar en él.

8.1 Instalación – Tabla de contenidos

- Obtención del software: Arch Linux
- Instalación de Arch Linux
  - Particionamiento del disco duro
  - Formateo de la partición
  - Instalando el sistema
  - Configuración básica
  - Instalación del gestor de arranque: GRUB
  - Post-instalación: Configuración de teclado permanente
  - Post-instalación: Creación de usuarios
- Instalación del escritorio: xfce
- Instalación de SteamCMD
  - Prerrequisitos
  - Instalación
- Implementación del servidor de DST
  - Instalación
  - Configuración
    - Token de validación
    - Steam Auth
    - Configuración de instancias
  - Ejecución

8.2 Obtención del software: Arch Linux

Descargaremos una imagen actualizada de **Arch Linux** en **https://www.archlinux.org/download/**. La imagen no incluye todo el software que necesitaremos, por lo que la máquina virtual donde se instalará, requerirá de una conexión a Internet.

8.3 Instalación de Arch Linux

Al iniciar la máquina virtual con la imagen que hemos descargado, el gestor de arranque nos pedirá con cual arquitectura arrancar **Arch Linux**. Elegiremos **x86_64** salvo casos aislados.

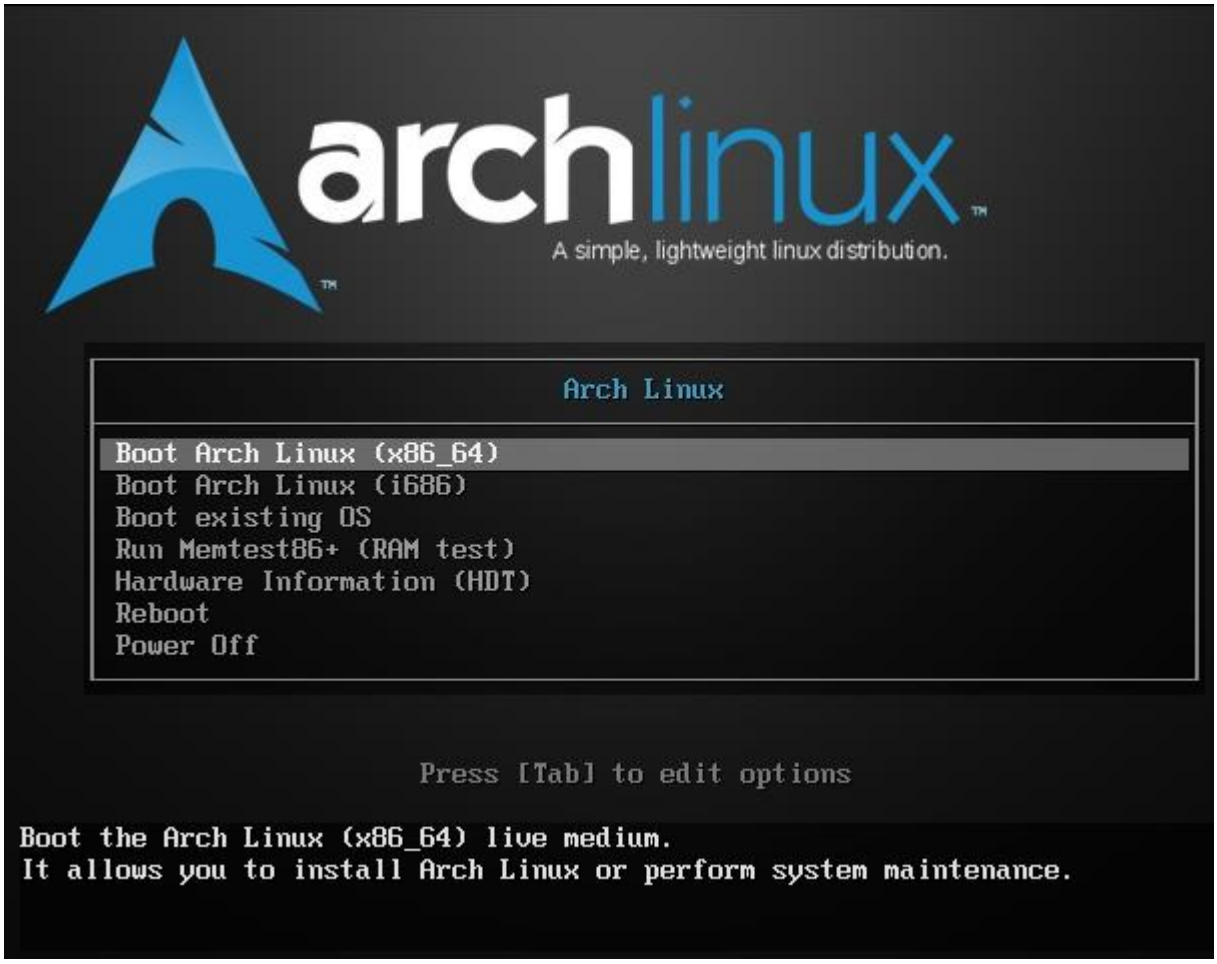


ILUSTRACIÓN 6: GESTOR DE ARRANQUE

Nada más arrancar nos encontraremos en la cuenta de superusuario. A partir de aquí seguiremos los pasos que describimos a continuación para preparar las particiones donde instalar **Arch Linux**, configurar un gestor de arranque, e instalar el software que necesitaremos para trabajar con el servidor.

Antes de empezar, emplearemos la orden `loadkeys es` para configurar la distribución de teclado a español.

8.3.1 Particionamiento del disco duro

Necesitaremos preparar una partición para instalar **Arch Linux**. Puesto que el sistema se trata de una máquina virtual, habremos creado un disco virtual con el espacio que deseamos dedicar al servidor, por lo tanto crearemos una partición única que ocupe todo el disco disponible.

Si utilizamos la orden `fdisk -l`, podremos observar que no tenemos ninguna partición creada, y en su lugar aparecerá el único disco disponible, en */dev/sda*. Para particionar el disco, emplearemos la orden `fdisk /dev/sda`, con lo que nos aparecerán las opciones disponibles con dicho disco.

Empleando el argumento **n**, empezaremos a crear la partición. Si empleamos las opciones por defecto que se nos ofrece, se creará una partición primaria que ocupará todo el disco duro, que es lo que deseamos realizar.

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p):

Using default response p.
Partition number (1-4, default 1):
First sector (2048-33554431, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-33554431, default 33554431)

Created a new partition 1 of type 'Linux' and of size 16 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

ILUSTRACIÓN 7: PARTICIONANDO EL DISCO VIRTUAL

Una vez creada la partición, empleamos la opción **w** para guardar la nueva tabla de particiones, y cerraremos **fdisk** presionando **Ctrl+C**.

8.3.2 Formateo de la partición

Necesitaremos formatear la nueva partición a ext4. Para ello, primero observaremos cómo se llama la partición empleando la orden lsblk.

```
root@archiso ~ # lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0    16G  0 disk
└─sda1   8:1    0    16G  0 part
sr0     11:0    1   742M  0 rom  /run/archiso/bootmnt
loop0    7:0    0 320.9M  1 loop /run/archiso/sfs/airootfs
```

ILUSTRACIÓN 8: TABLA DE DISCOS Y PARTICIONES

En nuestro caso, la nueva partición se encuentra en /dev/sda1. La formatearemos empleando la orden mkfs.ext4 /dev/sda1.

```
root@archiso ~ # mkfs.ext4 /dev/sda1
mke2fs 1.43.1 (08-Jun-2016)
Creating filesystem with 4194048 4k blocks and 1048576 inodes
Filesystem UUID: a50d5491-8747-4e3f-b059-7471f53c3e01
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654
        4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

ILUSTRACIÓN 9: FORMATEANDO LA PARTICIÓN

Montaremos la nueva partición con la orden mount -t ext4 /dev/sda1 /mnt.

8.3.3 Instalando el sistema

Descargaremos e instalaremos el software básico para iniciar Arch Linux con la orden `pacstrap /mnt base`. La operación necesitará de varios minutos.

Generaremos el archivo **fstab** (que contiene las particiones que se han de montar y su ubicación en el sistema de archivos) con la orden `genfstab -p /mnt >> /mnt/etc/fstab` y pasaremos a realizar cambios en la instalación, realizando **chroot** con `arch-chroot /mnt`.

8.3.4 Configuración básica

Empezaremos por configurar la zona horaria en la que nos encontramos, en este caso, la hora de Madrid. Para ello, usamos la orden `ln -s /usr/share/zoneinfo/Europe/Madrid /etc/localtime`.

Aparte, para el funcionamiento correcto del software que instalaremos, necesitamos configurar Arch Linux para que sea capaz de emplear el idioma español e inglés (por compatibilidad). Para ello modificaremos el archivo */etc/locale.gen* mediante la orden `nano /etc/locale.gen` y descomentaremos las líneas que contienen *en_US.UTF-8* y *es_ES.UTF-8*. Presionando **Ctrl+X** se nos preguntará si queremos salir guardando los cambios. Finalmente empleamos la orden `locale-gen`.

Debemos elegir un nombre de host, que insertaremos en */etc/hostname* y en */etc/hosts*. En esta ocasión emplearemos **serverTFG** como nombre de host. Lo insertaremos en */etc/hostname* mediante la orden `echo serverTFG >> /etc/hostname` y manualmente en */etc/hosts* con la orden `nano /etc/hosts`. Debemos insertar el mismo nombre de host, para ámbas líneas, como se muestra en la Ilustración 10.

```
  nano 2.6.2                                    File: /etc/hosts

#
# /etc/hosts: static lookup table for host names
#

#<ip-address>   <hostname.domain.org>   <hostname>
127.0.0.1       localhost.localdomain   localhost       serverTFG
::1             localhost.localdomain   localhost       serverTFG

# End of file
```

ILUSTRACIÓN 10: ARCHIVO /ETC/HOSTS

Configuraremos la red para que se emplee DHCP para asignar una IP al sistema. Primero necesitaremos saber como se llama la interfaz con la que accedemos a Internet. Para ello, usamos la orden `ip link`.

```
[root@archiso /]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default
1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT grou
ult qlen 1000
    link/ether 00:0c:29:79:1b:f5 brd ff:ff:ff:ff:ff:ff
```

ILUSTRACIÓN 11: INTERFACES DE RED

En nuestro caso, se trata de la interfaz **ens33**. Habilitaremos el servicio DHCP para dicha interfaz mediante la orden `systemctl enable dhcpcd@ens33.service`.

En siguiente lugar, configuramos una contraseña para la cuenta de superusuario, mediante la orden `passwd`.

8.3.5 Instalación del gestor de arranque: GRUB

Finalmente, necesitamos instalar **GRUB**. Arch Linux emplea **Pacman** como gestor de paquetes, con lo que descargaremos **GRUB** con la orden `pacman -Syu grub`. Aparte, instalaremos el gestor de arranque en nuestro disco con la orden `grub-install /dev/sda` y crearemos una configuración estándar con la orden `grub-mkconfig -o /boot/grub/grub.cfg`.

8.3.6 Post-instalación: Configuración de teclado permanente

Una vez seguido los pasos anteriores, reiniciaremos el sistema con la orden `reboot`, para confirmar que todo funciona correctamente. Algo que notaremos al volver a iniciar el sistema, es que el teclado ha vuelto a su configuración inglesa, por lo que usaremos la orden `localectl set-keymap --no-convert es` para configurarlo permanentemente en español.

8.3.7 Post-instalación: Creación de usuarios

Vamos a seguir una de las prácticas más recomendadas en Linux. Usar un usuario diferente al superusuario. Para ello, crearemos al usuario con el comando `useradd server -m -G wheel`, y configuraremos una contraseña con `passwd server`. A partir de ahora, usaremos este nuevo usuario para iniciar sesión.

Para poder usar permisos de superusuario, desde este usuario, necesitaremos instalar **sudo**. Para ello, usamos la orden `pacman -Syu sudo`. Necesitaremos habilitar el grupo **wheel** para permisos de superusuario. Con la orden `visudo`, se abrirá la configuración de **sudo** con el editor **vi**.

Vi es un editor de texto poco convencional, y cuenta con ciertas dificultades para usuarios nóveles. Por suerte, sólo haremos cambios menores. Buscaremos la linea

donde aparece `# %wheel ALL=(ALL) ALL` y la descomentaremos, dejándolo como aparece en la Ilustración 12. Para cerrar **vi** y guardar los cambios, escribimos **:x** y pulsamos **Enter**.

```
##
## User privilege specification
##
root ALL=(ALL) ALL

## Uncomment to allow members of group wheel to execute any command
%wheel ALL=(ALL) ALL

## Same thing without a password
# %wheel ALL=(ALL) NOPASSWD: ALL

## Uncomment to allow members of group sudo to execute any command
# %sudo ALL=(ALL) ALL

## Uncomment to allow any user to run sudo if they know the password
## of the user they are running the command as (root by default).
# Defaults targetpw  # Ask for the password of the target user
# ALL ALL=(ALL) ALL  # WARNING: only use this together with 'Defaults targe
```

ILUSTRACIÓN 12: ARCHIVO SUDOERS

Una vez listo, cerraremos la sesión de superusuario (`exit`) e iniciaremos sesión en el nuevo usuario. Para usar ordenes que necesitan permisos de superusuario, simplemente añadiremos `sudo` al inicio de la orden. Este pequeño cambio se reflejará en la documentación posterior.

8.4 Instalación del escritorio: xfce

Descargaremos el software (`sudo pacman -Syu xfce4 xorg`), con todas las opciones por defecto. Para que **xfce** se ejecute al iniciar sesión, copiaremos un esqueleto del archivo *.bash_profile*, con la orden `cp /etc/skel/.bash_profile ~/.bash_profile` y lo editaremos para añadir la siguiente linea: "`[ -z "$DISPLAY" -a "$(fgconsole)" -eq 1 ] && exec startxfce4`". El archivo debe quedar como se muestra en la Ilustración 13.

```
#
# ~/.bash_profile
#

[[ -f ~/.bashrc ]] && . ~/.bashrc
[ -z "$DISPLAY" -a "$(fgconsole)" -eq 1 ] && exec startxfce4
```

ILUSTRACIÓN 13: ARCHIVO .BASH_PROFILE

Reiniciaremos nuevamente el sistema (`reboot`), y esta vez, al iniciar sesión, nos encontraremos con el escritorio que **xfce** tiene preparado para nosotros. Se nos preguntará por una configuración inicial, que en nuestro caso, usaremos la de defecto. Si todo ha salido correctamente, nuestro sistema debe mostrarse como en la Ilustración 14.

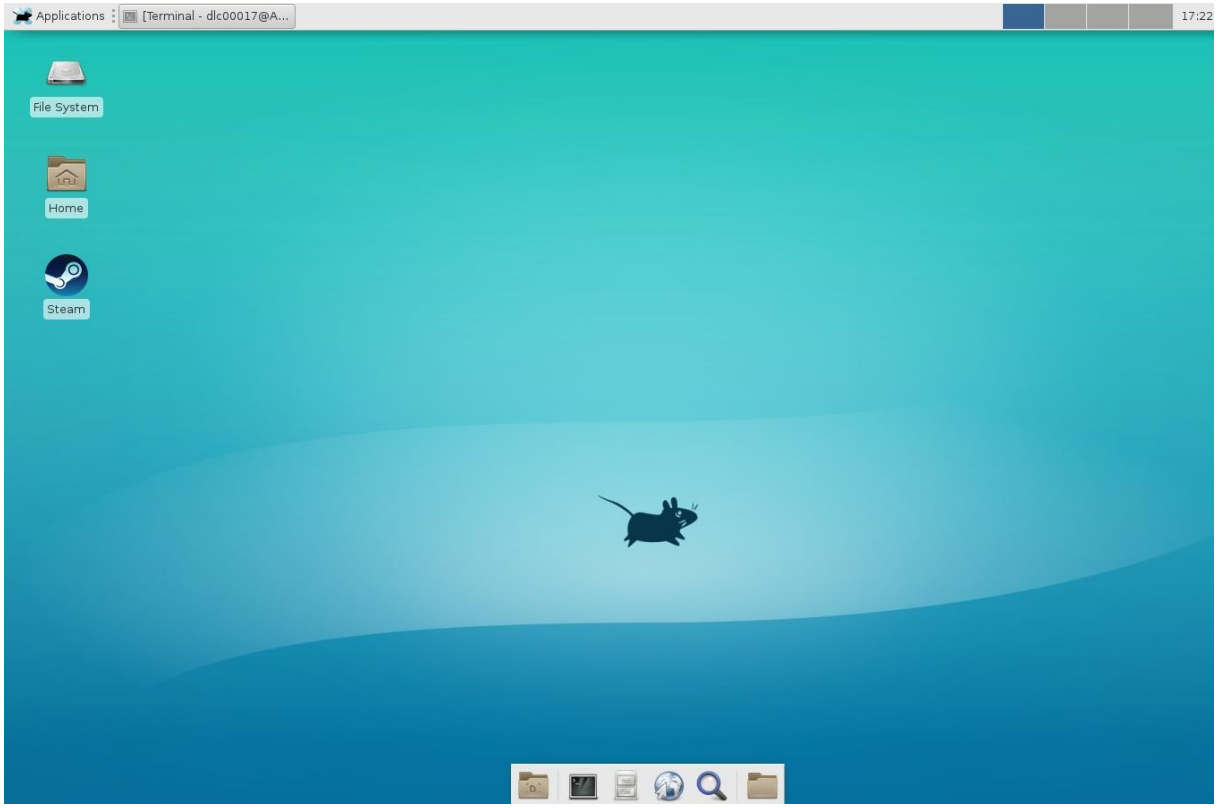


ILUSTRACIÓN 14: ESCRITORIO XFCE

Realizaremos unos pequeños ajustes a la configuración. Podemos acceder al menú de configuración, presionando en la esquina superior izquierda, sobre

Applications, y eligiendo *Settings*. En primer lugar, nos iremos al submenú de *Keyboard*, y en la ventana que se nos abrirá, presionaremos sobre la pestaña *Layout*.

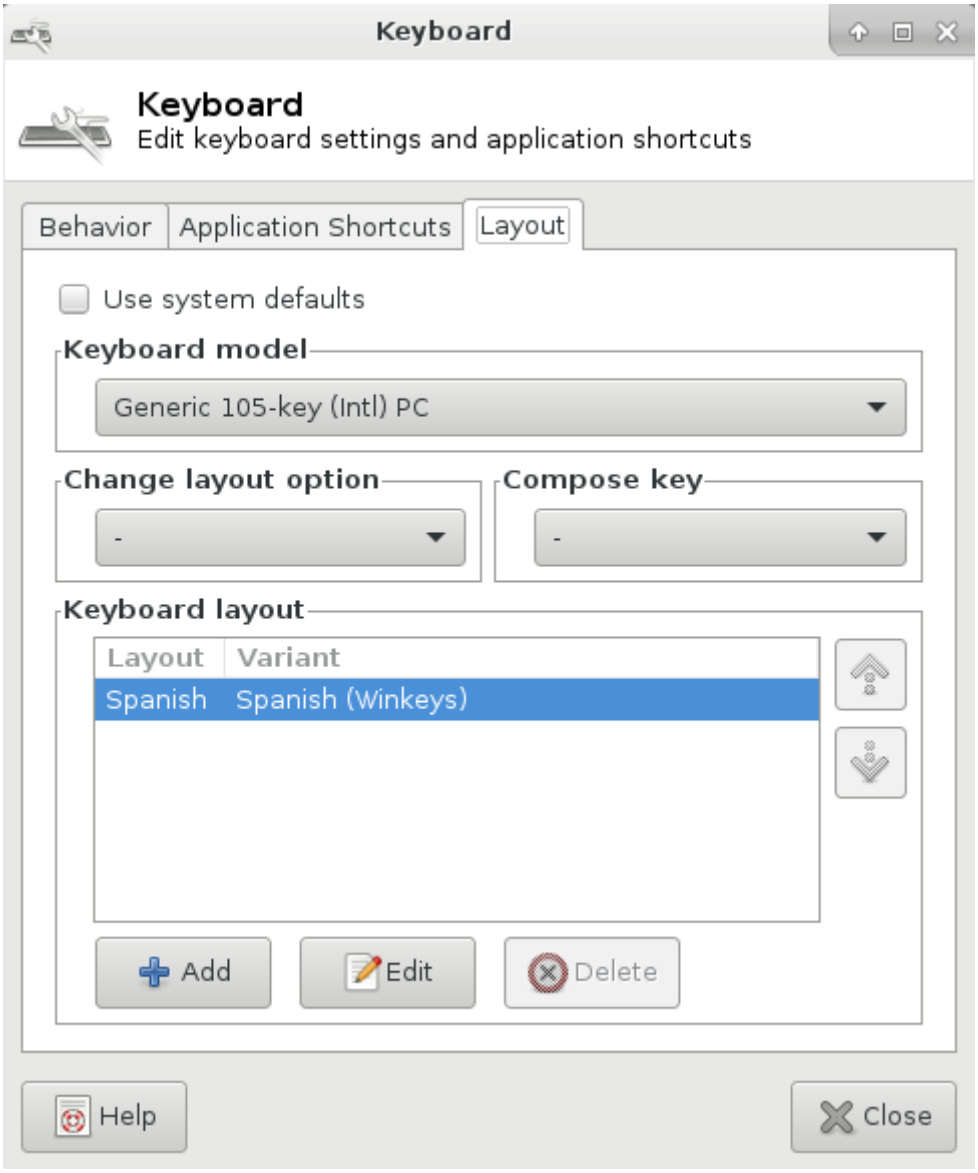


ILUSTRACIÓN 15: CONFIGURACIÓN DEL TECLADO

Queremos configurar un teclado español para **xfce**, por lo que desmarcaremos la casilla *Use system defaults*, y en *Keyboard layout*, editaremos la opción por defecto que existe, y la intercambiaremos por *Spanish (Winkeys)*.

8.5 Instalación de SteamCMD

El servidor de DST necesita Steam para autentificar y poder ejecutarse. Para ello, vamos a usar una versión minimalista de Steam, llamada **SteamCMD**, que se

encuentra en la AUR (Archlinux User Repository) de Arch, concretamente en https://aur.archlinux.org/packages/steamcmd.

8.5.1 Prerrequisitos

En primer lugar, vamos a habilitar el repositorio **Multilib** en **pacman**, ya que uno de los prerrequisitos se encuentra en dicho repositorio. Para ello editaremos el archivo **pacman.conf** (`sudo nano /etc/pacman.conf`) y descomentaremos las líneas de **Multilib**, dejando el archivo tal como aparece en la Ilustración 16.

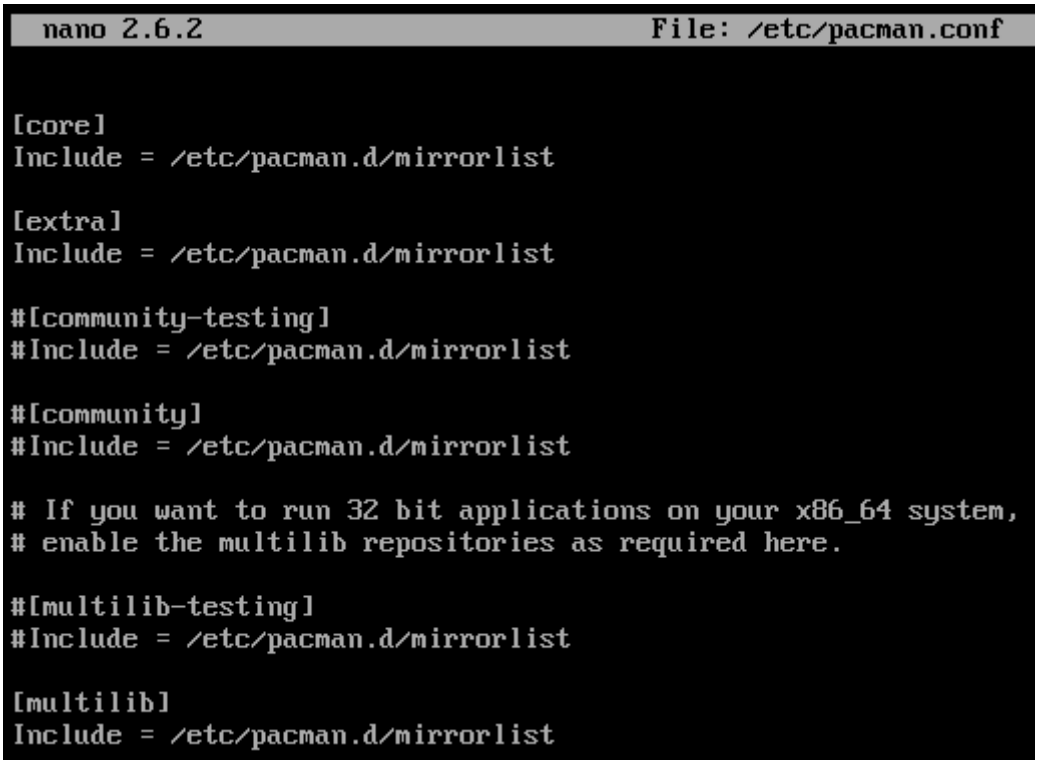


ILUSTRACIÓN 16: ARCHIVO PACMAN.CONF

Una vez realizado los cambios, vamos a instalar las herramientas de desarrollo necesarias para instalar paquetes de la AUR. Para ello, usamos la orden `sudo pacman -Syu --needed base-devel`.

Adiccionalmente, accederemos al AUR de **Arch Linux** para descargar unas librerías necesarias para el servidor de DST. Crearemos una carpeta llamada **builds** en nuestra carpeta de usuario, donde descargaremos los paquetes procedentes del AUR. Para ello, escribimos la orden `mkdir ~/builds` y posteriormente nos ubicamos en ella con `cd ~/builds`.

Descargaremos **lib32-rtmpdump** (`curl -L -O https://aur.archlinux.org/cgit/aur.git/snapshot/lib32-rtmpdump.tar.gz`) y lo descomprimimos (`tar -xvf lib32-rtmpdump.tar.gz`). Accedemos a la nueva carpeta creada **lib32-rtmpdump** (`cd lib32-rtmpdump`) y lo instalamos (`makepkg -sri`). Una vez realizado la instalación, volvemos a la carpeta anterior (`cd ..`).

Realizaremos los mismos pasos con **lib32-libcurl-compat**. Descargamos (`curl -L -O https://aur.archlinux.org/cgit/aur.git/snapshot/lib32-libcurl-compat.tar.gz`), descomprimimos (`tar -xvf lib32-libcurl-compat.tar.gz`), nos movemos a la nueva carpeta (`cd lib32-libcurl-compat`) e instalamos (`makepkg -sri`). Finalmente, volvemos a la carpeta **builds** (`cd ..`).

8.5.2 Instalación

Con los pasos anteriores, tenemos lo necesario para instalar **SteamCMD**.

Mientras nos encontramos en la carpeta **builds**, descargamos el paquete de SteamCMD (`curl -L -O https://aur.archlinux.org/cgit/aur.git/snapshot/steamcmd.tar.gz`) y lo descomprimimos (`tar -xvf steamcmd.tar.gz`). Se habrá creado una carpeta llamada **steamcmd**, por lo que nos reubicaremos en ella (`cd steamcmd`). Finalmente instalamos el paquete con la orden `makepkg -sri`. Las instrucciones que aparecen al finalizar la instalación, se nos pedirá ejecutar **steamcmd** sin uso de argumentos, por lo que lo hacemos inmediatamente (`sudo steamcmd`).

8.6 Implementación del servidor de DST

8.6.1 Instalación

Una vez instalado SteamCMD, lo ejecutaremos (`sudo steamcmd`) e iniciaremos sesión con una cuenta pública (`Steam> login anonymous`). Configuraremos dónde instalar los archivos del servidor (`Steam> force_install_dir /home/server/server_dst`) y descargaremos los archivos (`Steam> app_update 343050 validate`).

8.6.2 Configuración

Para este servidor, crearemos dos instancias, **Overworld** y **Caves**, que hacen referencia a los dos mapas que el juego emplea. Para ello, tendremos que realizar una configuración para cada instancia. Adicionalmente, el servidor de DST emplea un **token de validación**, sólo obtenible mediante el juego original, para poder iniciar los servidores.

Toda la configuración se realiza en la carpeta *~/dst/.klei/DoNotStarveTogether*. Antes de escribir la configuración, es necesario que ejecutemos el servidor, para crear una configuración por defecto, y comprobar que todo funciona correctamente.

Para ello, crearemos unos scripts shell que iniciarán el servidor con los parámetros necesarios. Accederemos al lugar donde se encuentran los archivos del servidor (`cd ~/server_dst/bin`) y crearemos dos archivos de texto: *start-overworld.sh* y *start-caves.sh*.

```
  nano 2.6.2                File: start-overworld.sh

./dontstarve_dedicated_server_nullrenderer -console -cluster TFGSe
-shard Master
```

ILUSTRACIÓN 17: ARCHIVO START-OVERWORLD.SH

```
  nano 2.6.2                  File: start-caves.sh

./dontstarve_dedicated_server_nullrenderer -console -cluster TFGSe
-shard Caves
```

ILUSTRACIÓN 18: ARCHIVO START-CAVES.SH

En dos terminales diferentes, nos localizaremos en la carpeta correcta (`cd ~/server_dst/bin`) y ejecutaremos ambos scripts (`sh start-overworld.sh`) y (`sh start-caves.sh`). En ambos terminales, podremos leer lo siguiente:

Necesitaremos el token de validación, obteniéndolo aparte en un sistema que ejecute el juego original. Siguiendo las instrucciones, obtendremos una clave de texto que posteriormente emplearemos.

ILUSTRACIÓN 19: TOKEN DE VALIDACIÓN REQUERIDO

Detendremos ambos servidores pulsando **Ctrl+C** en la terminal.

8.6.2.1 Token de validación

Si hemos seguido las instrucciones que aparecían en la Ilustración 19, debemos disponer de una clave de texto. Esta clave de texto debe ser introducida en el archivo *cluster_token.txt*, localizado en *~/.klei/DoNotStarveTogether/TFGServer*. Inicialmente, el archivo no existirá, por lo que nos moveremos a la carpeta (`cd ~/.klei/DoNotStarveTogether/TFGServer`) e introduciremos directamente el token (`echo EL_TOKEN_QUE_HEMOS_OBTENIDO > cluster_token.txt`).

8.6.2.2 Steam Auth

Otro requisito para poder ejecutar el servidor de DST, es que Steam esté en ejecución. Una excepción es si empleamos SteamCMD. En ese caso, sólo basta hacer un enlace simbólico. Para ello, debemos eliminar la librería que DST emplea inicialmente (`rm ~/server_dst/bin/lib32/steamclient.so`) y la sustituiremos (`ln -s /usr/share/steamcmd/steamclient.so ~/server_dst/bin/lib32/steamclient.so`).

8.6.2.3 Configuración de instancias

Cada instancia del servidor emplea una configuración específica, que indica el nombre del servidor, puerto utilizado, número de jugadores, descripción, y otros detalles propios del juego. Debemos crear la configuración deseada para cada instancia, más una compartida por ambas.

Los detalles de configuración se encuentran en la ruta *~/.klei/DoNotStarveTogether/TFGServer/*. Dentro de esta ruta encontraremos la configuración compartida de ambas instancias en *./cluster.ini*, mientras que la configuración de la instancia **Master** y la instancia **Caves** emplean *./Master/server.ini* y *./Caves/server.ini* respectivamente.

Emplearemos una configuración éstandar. A continuación se muestra la configuración que debemos insertar.

```
  nano 2.6.2                    File: cluster.ini

[GAMEPLAY]
game_mode = survival
max_players = 10
pvp = false
pause_when_empty = true

[NETWORK]
cluster_name = TFGServer
cluster_description = Servidor de DST
cluster_password = UJA
cluster_intention = social
autosaver_enabled = true
enable_vote_kick = false

[MISC]
console_enabled = true

[SHARD]
shard_enabled = true
bind_ip = 127.0.0.1
master_ip = 127.0.0.1
master_port = 11001
cluster_key = dst
```

ILUSTRACIÓN 20: ARCHIVO CLUSTER.INI

```
  nano 2.6.2                          File: server.ini

[NETWORK]
server_port = 11000


[SHARD]
is_master = false
name = Caves
id = 370435798


[STEAM]
master_server_port = 12348
authentication_port = 12347
```

ILUSTRACIÓN 21: ARCHIVO CAVES/SERVER.INI

```
  nano 2.6.2                          File: server.ini

[NETWORK]
server_port = 10999


[SHARD]
is_master = true


[STEAM]
master_server_port = 12346
authentication_port = 12345
```

ILUSTRACIÓN 22: ARCHIVO MASTER/SERVER.INI

Además de los detalles de configuración, cada carpeta de cada instancia debe contener un archivo *worldgenoverride.lua*, que contiene la configuración con la que se debe generar el mundo de juego. Debido a la extensión de dichos archivos, el código de ambos archivos estará disponible en los anexos al final de este trabajo.

8.6.3 Ejecución

Una vez realizado los pasos anteriores, tenemos todo preparado para iniciar el servidor y permitir jugadores al mismo. Para iniciar el servidor, deberemos abrir dos

terminales que ejecuten los scripts *start-overworld.sh* (`sh ~/server_dst/bin/start-overworld.sh`) y posteriormente *start-caves.sh* (`sh ~/server_dst/bin/start_caves.sh`).

Para entrar al servidor, desde el exterior de nuestra red local, los jugadores tendrán que emplear nuestra clave pública, y el **puerto 10999**. Desde el interior de la red local, usaremos el comando `ip addr show` para visualizar la IP que posee el servidor.

9. Nginx – Introducción

Nginx es un servidor web, libre, de código abierto y multiplataforma, que comenzó a desarrollarse en 2002. Es actualmente el segundo servidor web más utilizado, por detrás de Apache. Basado en una arquitectura dirigida a eventos, permite ahorrar recursos reduciendo la memoria empleada por cada usuario.

Uno de sus mayores fuertes es su alta escabilidad, que permite servir web a un mayor número de clientes concurrentes con un menor impacto en el rendimiento. A diferencia de Apache, es más ligero y eficiente, pero cuenta con un menor número de funcionalidades, por lo cual es la mejor elección para proyectos web que confíen en un diseño simple pero dirigido a un mayor número de usuarios.

Veremos a continuación cómo instalar y configurar Nginx en un servidor CentOS, incluyendo un ejemplo web sencillo que nos permitirá verificar un funcionamiento correcto del software.

9.1 Instalación – Tabla de contenidos

9.2 Obtención del software CentOS

Nuestra intención es instalar un servidor web Nginx en CentOS 7. Para instalar CentOS, podemos emplear tanto la imagen DVD, como la imagen NetInstall, que descargará los componentes que elijamos automáticamente durante la instalación. Todas las imagenes están disponibles en la página web de CentOS, concretamente en https://www.centos.org/download/.

ILUSTRACIÓN 23: IMÁGENES CENTOS DISPONIBLES

9.3 Instalación del sistema operativo: CentOS

El proceso de instalación en la máquina virtual es sencillo. Se nos pedirá el idioma que deseamos emplear, localización para fecha y hora (que sincronizará automáticamente si tenemos conexión a Internet), perfil de seguridad (empleamos el perfil general para sistemas de propósito general, que contiene una serie de reglas estándar para servidores), y la elección de software a incluir (elegimos Servidor web básico, que incluirá una serie de herramientas básicas para administración web).

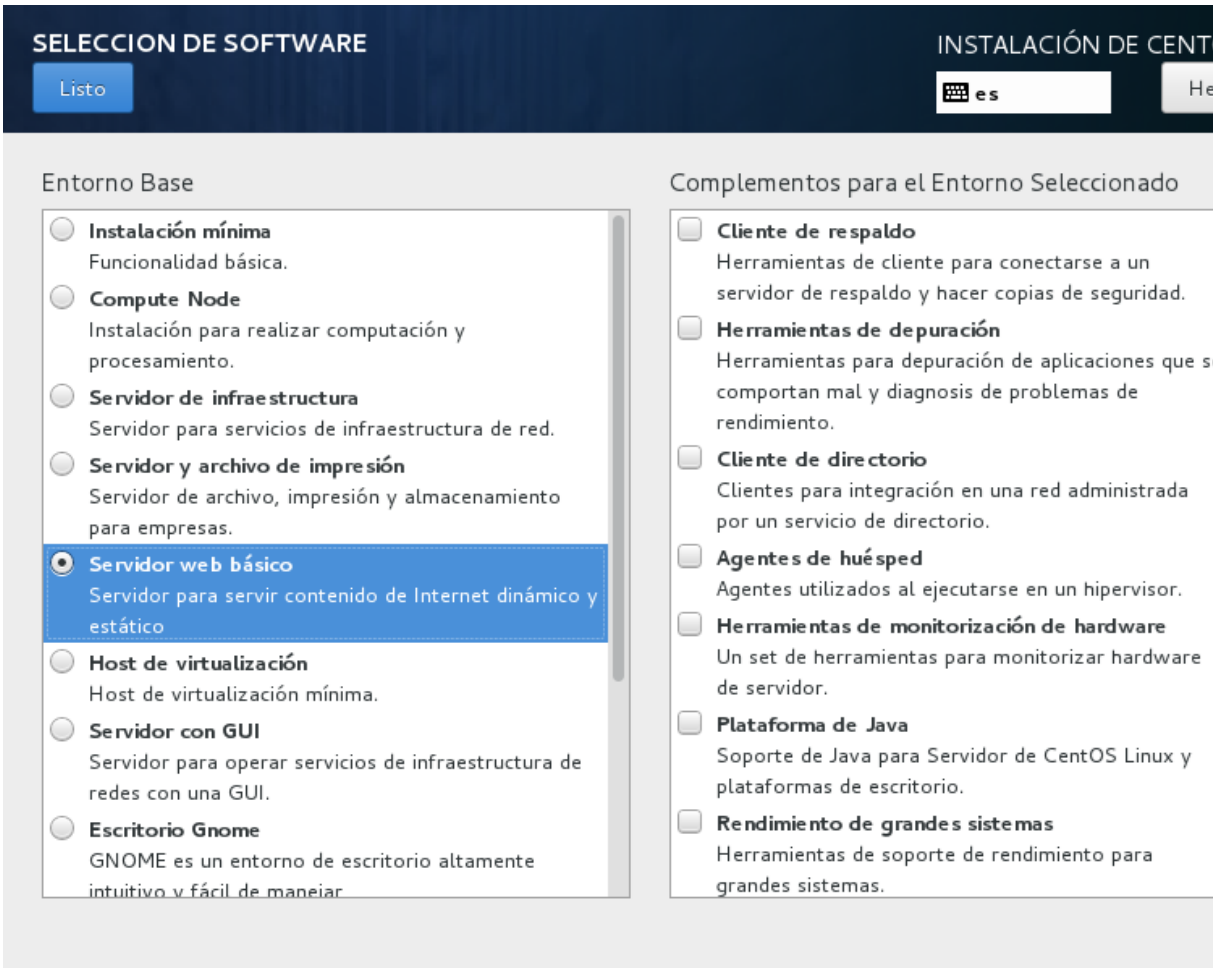


ILUSTRACIÓN 24: SELECCIÓN DEL SOFTWARE A INSTALAR

Existen una serie de complementos que podemos incluir en la instalación, como monitorización de hardware, depurador, etc. Para este sistema no los incluiremos.

También necesitaremos indicar en que partición deseamos instalar el sistema (hemos creado un disco duro de 20 GB para la instalación), configuración de red, contraseña del usuario root, y nuestra propia cuenta de usuario.

Adicionalmente, si empleamos la imagen NetInstall, necesitaremos configurar un repositorio del cual descargar el software necesario. Para CentOS 7, empleamos el mirror oficial: http://mirror.centos.org/centos/7/os/x86_64/

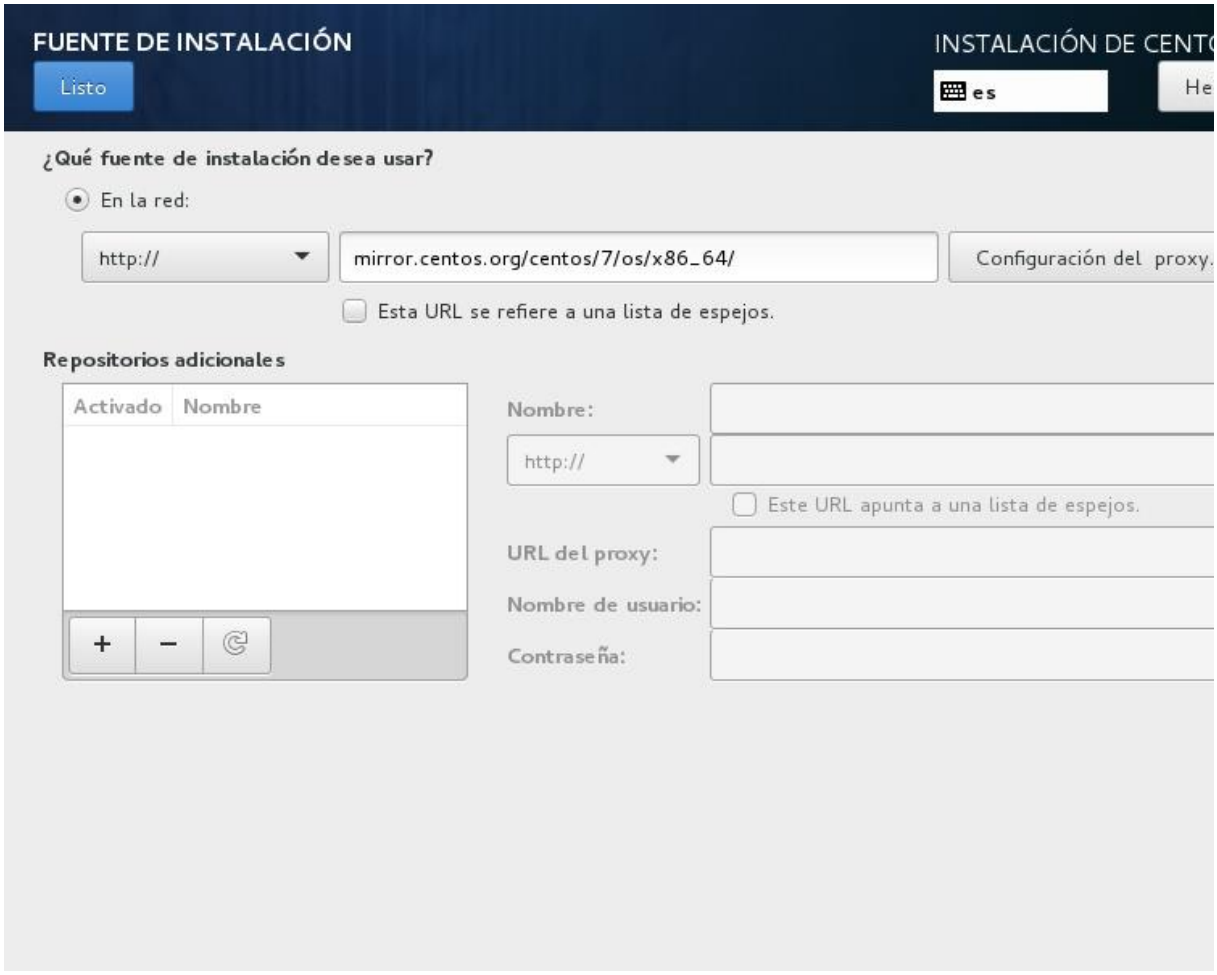


ILUSTRACIÓN 25: REPOSITORIO A UTILIZAR EN LA INSTALACIÓN

Una vez instalado el SO y arrancada la máquina, sólo tendremos la consola para trabajar a partir de ahora. Opcionalmente podemos instalar GNOME, que desde CentOS es una tarea relativamente fácil.

9.4 Instalación del escritorio: GNOME

Para instalar GNOME, primero iniciaremos sesión en el usuario root. Si ya nos hemos autentificado con nuestra cuenta personal, simplemente deberemos escribir `su`, y se nos pedirá nuestra contraseña de root.

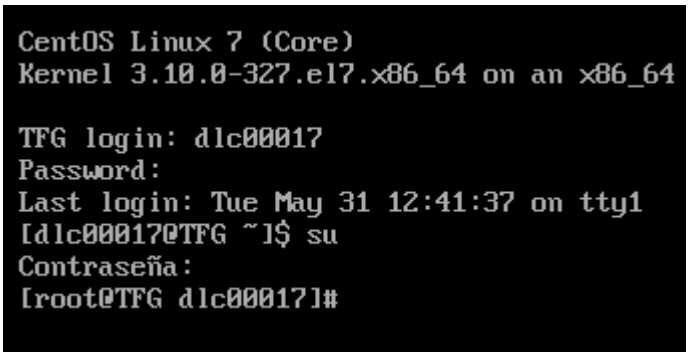


ILUSTRACIÓN 26: INICIO DE SESIÓN DEL SUPERUSUARIO

Una vez en la cuenta de root, debemos escribir `yum -y groups install "GNOME Desktop"`. Esto descargará todo el software necesario para arrancar GNOME. La tarea necesitará de 5 a 10 minutos para descargar el software e instalarlo.

ILUSTRACIÓN 27: GNOME FINALMENTE INSTALADO

Una vez instalado, lo configuraremos para que arranque por defecto con el sistema, mediante el comando `systemctl set-default graphical.target`. Reiniciaremos la máquina para poder observar los cambios con `reboot`.

9.5 Reemplazando Apache por Nginx

Nos falta reemplazar Apache (que viene incluído por defecto), por Nginx. Para ello, primero debemos desactivar el servicio de Apache. Abriremos una terminal en el escritorio (menú aplicaciones, buscamos la aplicación Terminal"), y nos autentificaremos con la cuenta root empleando `su`.

Detenemos el servicio con `service httpd stop`, y lo desactivaremos del ciclo de arranque con `systemctl disable httpd`. Con ello, pasamos a instalar Nginx.

Para instalar Nginx, necesitaremos activar el paquete de software EPEL (Extra Packages for Enterprise Linux). Simplemente debemos escribir `yum install epel-release`. A continuación, podremos instalar Nginx con `yum -y install Nginx`. Finalmente ejecutamos Nginx (`service Nginx start`), y lo configuramos para que arranque con el sistema automáticamente (`systemctl enable Nginx`).

9.6 Creando y configurando nuestro sitio web

A continuación, detallaremos una serie de pasos para servir una página web de ejemplo mediante la instalación de Nginx que hemos preparado.

9.6.1 Creación de usuarios

Necesitaremos un usuario del sistema, distinto al superusuario, que posea los derechos de los archivos del sitio web. En caso de un ataque al servicio web, el daño se limitaría a dicho servicio y no se extendería más allá de los permisos que el usuario posee.

Para ello, utilizaremos un usuario llamado **server**. Empleando la cuenta de superusuario (`su`), crearemos dicho usuario mediante la orden `adduser server` y le configuraremos una contraseña mediante `passwd server`.

9.6.2 Directorio web

Seguidamente, crearemos un directorio donde se hospedarán los archivos del sitio web. Suponiendo que el dominio de nuestro sitio web se llama **nginxTFG.es**, usariamos la orden `mkdir -p /var/www/nginxTFG.es/public_html` para crear el directorio.

Dentro de este directorio, haremos una página de inicio básica para nuestro sitio web. Empleando el editor de texto que deseemos, crearemos el archivo *index.html*, en la ruta creada anteriormente, y crearemos una página HTML básica como la expuesta a continuación (Ilustración 28).

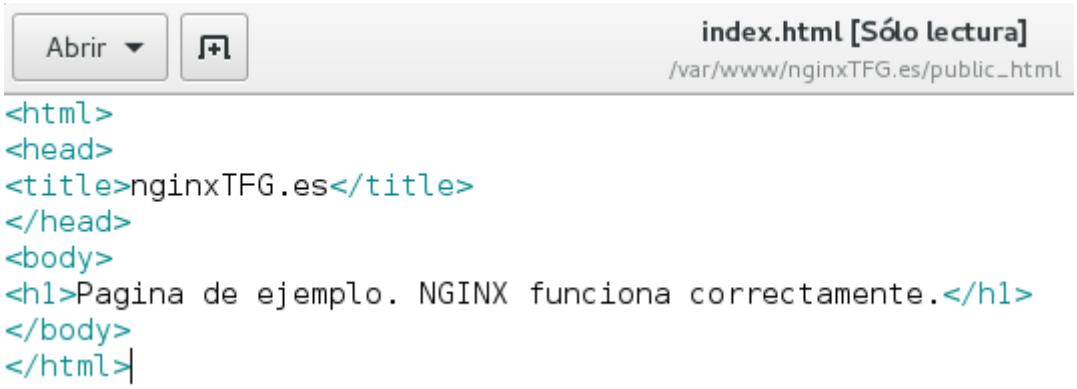


ILUSTRACIÓN 28: ARCHIVO INDEX.HTML

Daremos los permisos de propietario al usuario **server**, para el directorio que hemos creado. Para ello, empleamos la orden `chown -R server:server /var/www/nginxTFG.es/public_html`. Además, configuraremos los permisos necesarios con `chmod 755 /var/www/nginxTFG.es/public_html`. De esta manera, todos los usuarios cuentan con permisos de lectura sobre el sitio web.

9.6.3 Configurando Nginx

Nginx emplea una configuración modular por cada sitio web en servicio. Necesitaremos crear directorios donde se localizarán los archivos de configuración: *sites-available* y *sites-enabled*. El primero guarda la configuración de todos nuestros sitios web, mientras el segundo, sólo cuenta con aquellos sitios web que estamos activamente sirviendo a Internet. De esta manera, añadiendo y eliminando un énlace simbólico, somos capaz de activar y desactivar un sitio web.

Por consiguiente, creamos la carpeta *sites-available* (`mkdir /etc/nginx/sites-available`) y *sites-enabled* (`mkdir /etc/nginx/sites-enabled`). Además, debemos configurar Nginx para que emplee estos directorios. Para ello, debemos editar el archivo */etc/nginx/nginx.conf*.

A la configuración por defecto, dentro del bloque de configuración http, deberemos añadir la siguiente linea: `include /etc/nginx/sites-enabled/*.conf;`

ILUSTRACIÓN 29: ARCHIVO NGINX.CONF

Finalmente, nos falta por crear la configuración específica para nuestro sitio web. Empleando el directorio *sites-available*, crearemos un archivo llamado *nginxTFG.es.conf*, propio de nuestro sitio web.

En dicho archivo, debemos especificar opciones básicas como el puerto de escucha, nombre del sitio web, directorio de trabajo, etcétera. Para nuestra página de ejemplo, sólo emplearemos opciones básicas.

Nuestro archivo de configuración debe tener un aspecto como el que se muestra en la Ilustración 30.

Abrir nginxTFG.es.conf [Sólo lectura]
/etc/nginx/sites-available

```
server {
   listen 80;
   server_name nginxTFG.es www.nginxTFG.es;
   location / {
      root /var/www/nginxTFG.es/public_html;
      index index.html index.htm;
      try_files $uri $uri/ =404;
   }
   error_page 500 502 503 504 /50x.html;
   location = /50x.html {
      root html;
   }
}
```

ILUSTRACIÓN 30: ARCHIVO NGINXTFG.ES.CONF

Como hemos detallado antes, la configuración empleada cuenta con el puerto de escucha, dominios de nuestro sitio web, directorio de trabajo y el tratamiento de errores. Aún nos falta por activar, en último lugar, el sitio web.

Emplearemos un enlace simbólico de la configuración que hemos creado, al directorio *sites-enabled*. Usamos la orden `ln -s /etc/nginx/sites-available/nginxTFG.es.conf /etc/nginx/sites-enabled/nginxTFG.es.conf`.

Finalmente, reiniciamos Nginx para que recargue la configuración y nuestro sitio web esté disponible (`service nginx restart`).

9.6.4 Visualizando nuestro sitio web

Si deseamos probar que nuestro sitio web funciona correctamente, necesitaremos que el dominio que hemos empleado, apunte a la dirección IP de nuestro servidor. Ya que nos limitamos a realizar una prueba, visitaremos el sitio desde dentro de la red local del servidor, en un ordenador aparte.

Con Nginx, no podemos visitar el sitio web empleando únicamente la dirección IP del servidor, sino que debemos emplear el dominio nginxTFG.es. Para ello, modificaremos el archivo *hosts*, del sistema que estemos empleando para visitar el sitio web.

Dependiendo del sistema operativo empleado, la localización del archivo *hosts* es diferente.

- **Windows**: *%WINDIR%\System32\drivers\etc\hosts*
- **GNU/Linux**: */etc/hosts*
- **Mac OS X**: */private/etc/hosts*

A dicho archivo, simplemente necesitamos añadir una linea, indicando la IP y dominios. En nuestro caso, suponiendo que la IP de nuestro servidor sea 10.0.0.20, escribiriamos lo siguiente:

```
10.0.0.20   www.nginxTFG.es nginxTFG.es
```

Hecho esto, podemos visitar nuestro sitio web.

ILUSTRACIÓN 31: SITIO WEB DE EJEMPLO

10. NAS – Introducción

NAS (Network Attached Storage) es una tecnología empleada para el uso de almacenamiento en lugares remotos, accesible mediante una amplia variedad de protocolos (FTP, NFS, CIFS, etc...). En resumen, es un espacio de almacenamiento de archivos, accesible desde cualquier parte del mundo.

Aparte de almacenar, podemos emplear un NAS para tareas más cotidianas como la reproducción de video (películas, series, videos educativos, documentales...) o audio durante la transmisión, sin necesidad de descargar el video previamente. De la misma manera, podemos compartir de una manera más sencilla y eficiente nuestros archivos a amigos y familiares.

A continuación, falicitaremos la documentación necesaria para instalar un servidor NAS es una máquina virtual de Vmware, y habilitar uno de los múltiples protocolos para comunicar con el NAS.

10.1 Instalación – Tabla de contenidos

- Obtención del software
- Instalación básica
- Configuración
  - Creación de un certificado SSL
  - Configuración FTP
  - Almacenamiento y permisos

10.2 Obtención del software

Hemos empleado la imagen ISO que OpenMediaVault proporciona para instalar Debian 7 Wheezy, junto a su solución NAS ya incluida. De esta manera, el proceso de preparar el servidor se reduce a instalar la ISO en una máquina virtual (MV), y realizar la configuración.

La imagen ISO podemos descargarla en https://sourceforge.net/projects/openmediavault/files/, donde encontramos imágenes ISO para diferentes versiones y arquitecturas. En nuestro caso, instalamos la última versión no-beta (2.1), para una arquitectura de 64 bits.

10.3 Instalación básica

La instalación de la imagen ISO se realiza fácilmente, consistiendo en la configuración de la cuenta superusuario, además de configuración de fecha y hora.

Una vez instalado, no necesitamos realizar ningún paso adicional en la propia máquina virtual. El NAS se iniciará y será accesible nada más arrancar la MV. Podemos acceder al NAS desde su interfaz web (ubicado en el puerto 80 del servidor), autentificarnos con el usuario y contraseña por defecto, cambiarla a una diferente, y configurar el NAS a nuestro antojo.

- Usuario: admin
- Contraseña: openmediavault

10.4 Configuración

Para este sistema, hemos configurado un simple servidor FTP sobre TLS, y hemos restringido el acceso a un sólo usuario y contraseña que hemos creado.

Como primer paso, habilitaremos el servidor FTP desde la interfaz web. Nada más acceder con el usuario y contraseña, podemos encontrarnos esta ventana (Ilustración 32) de servicios donde rápidamente habilitar el servicio FTP.

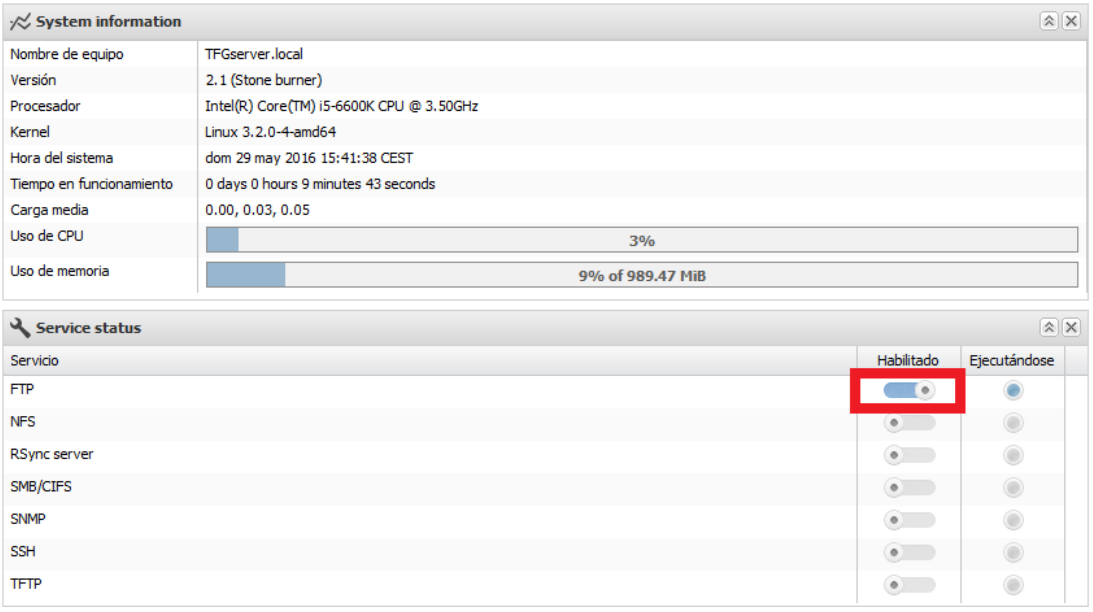


ILUSTRACIÓN 32: ESTADO NAS Y SERVICIOS

Lo habilitaremos y lo configuraremos posteriormente. Antes deberíamos cambiar el usuario y contraseña por defecto. En la página inicial, tendremos a la izquierda un menú (Ilustración 33) dónde configurar distintos apartados del NAS.

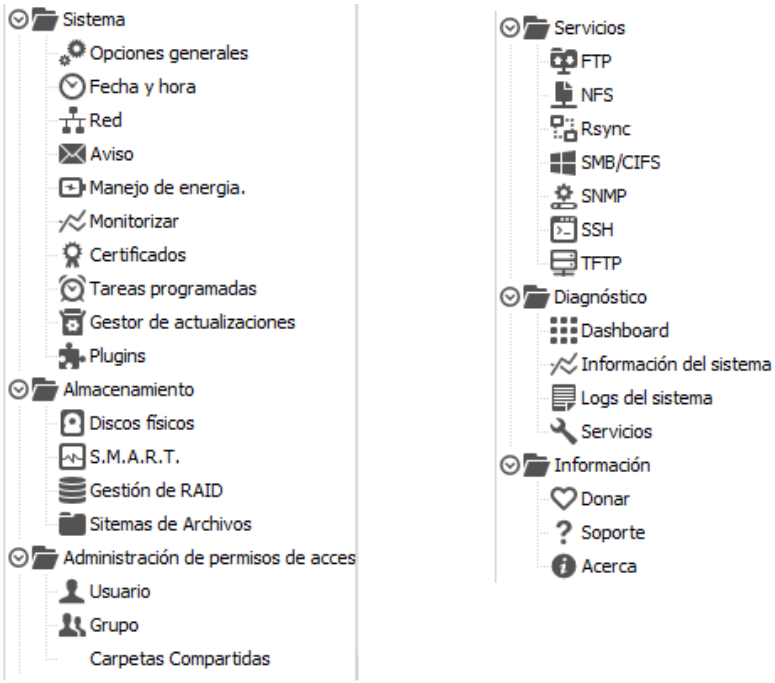


ILUSTRACIÓN 33: MENÚ DE OPCIONES

Para cambiar la contraseña, iremos a opciones generales, que nos presentará dos apartados. Uno para configurar el acceso a la interfaz web (puerto, caducidad de la sesión, uso de SSL/TLS...). Y otro para configurar la contraseña.

Idealmente la interfaz web sólo debería ser accesible desde la intranet y sólo haríamos visible los puertos necesarios para FTP sobre TLS. Pero para este ejemplo, habilitaremos SSL para la interfaz web.

10.4.1 Creación de un certificado SSL

Tanto el acceso por HTTPS como FTP sobre TLS necesitarán certificados para emplear esta función. Tenemos la opción de crear certificados self-signed desde el propio servidor NAS, accediendo al menú de Certificados. En el menú tendremos dos apartados: SSH y SSL. Iremos a la opción de SSL y usaremos la opción de crear certificado.

Adicionalmente podemos importar uno previo, en el caso de que no queramos usar un certificado self-signed, o lo tengamos creado previamente.

Para crear el certificado, sólo necesitaremos elegir la longitud de clave, periodo de validez y un nombre. Adicionalmente podemos añadir más detalles como el nombre de la organización y localización, pero no son obligatorios.

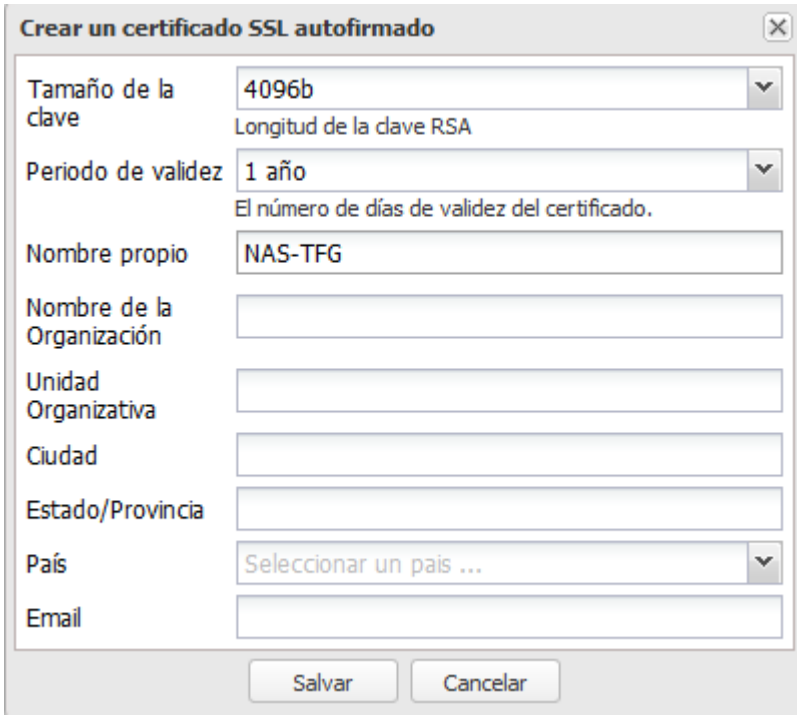


ILUSTRACIÓN 34: CREACIÓN DE UN CERTIFICADO SSL

Una vez creado, volvemos al apartado de Opciones Generales, habilitamos SSL/TLS, y seleccionamos emplear este certificado.

10.4.2 Configuración FTP

Configuraremos el servidor FTP, accediendo a su opción en el menú.

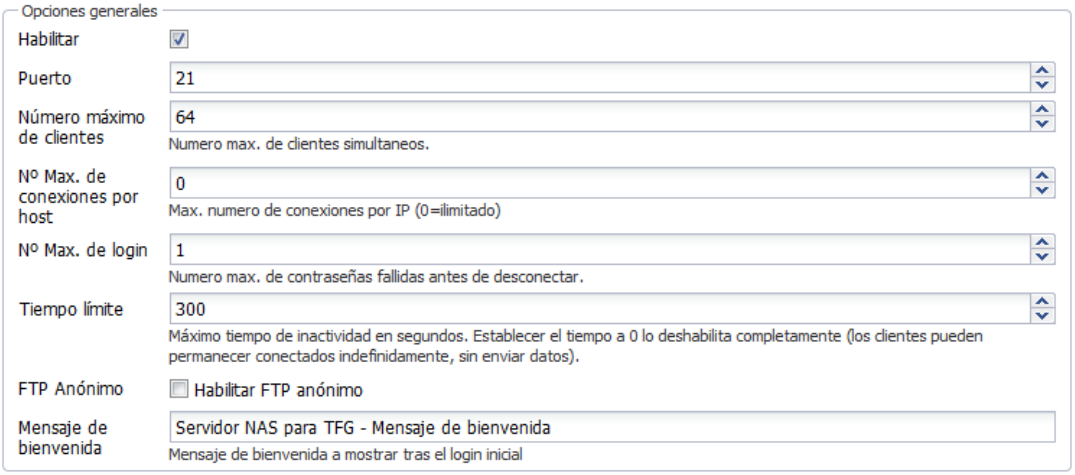


ILUSTRACIÓN 35: CONFIGURACIÓN FTP BÁSICA

Podremos habilitar el servicio si no lo hemos habilitado anteriormente, y configurar algunas opciones básicas. Necesitaremos obligatoriamente abrir el puerto 21 en el firewall para permitir clientes en el exterior acceder a nuestro FTP. Dependiendo del uso que le vayamos a dar, y los recursos disponibles para el NAS, configuraremos un número máximo de conexiones. Adiccionalmente, disponemos de una configuración más avanzada (Ilustración 36).

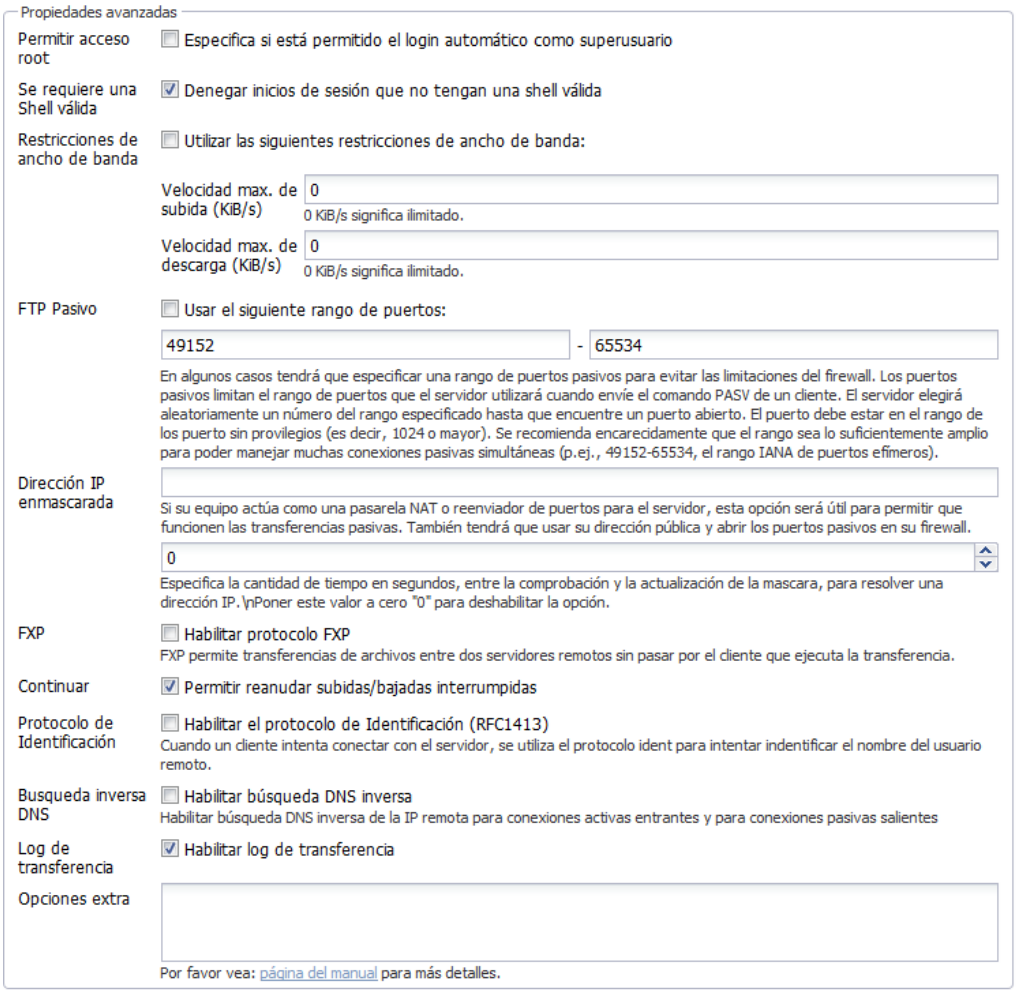


ILUSTRACIÓN 36: CONFIGURACIÓN FTP AVANZADA

Las opciones que más nos interesan son las restricciones de ancho de banda y el FTP pasivo. La restricción de ancho de banda nos permitirá asignar un límite máximo para los recursos de red. De esta manera podemos controlar la carga máxima que nuestros sistemas pueden soportar, sin que ello afecte negativamente al resto de máquinas virtuales.

El FTP pasivo es necesario para que aquellos clientes que estén detras de un firewall puedan acceder a los ficheros del NAS. En este caso, nos interesaría crear un pequeño rango de puertos, similar al número de clientes máximos que pueden conectarse a nuestro servidor, y abrir dichos puertos en el firewall. Sólo los puertos del servicio FTP deben ser los únicos abiertos que accedan a este sistema, de esta manera limitamos la superficie de ataques que el sistema puede recibir desde el exterior.

Para terminar, necesitamos configurar FTP sobre TLS, accediendo a una pestaña disponible en la parte superior.

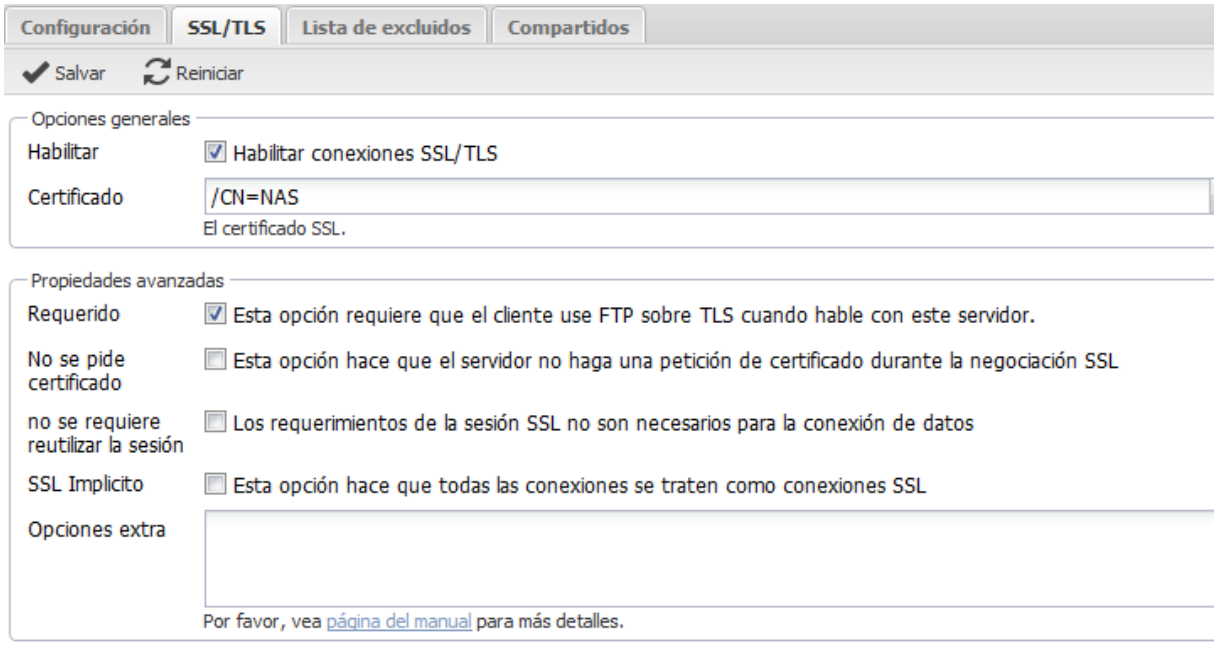


ILUSTRACIÓN 37: CONFIGURACIÓN SSL/TLS PARA FTP

Habilitamos el uso de SSL/TLS, elegimos un certificado (preferiblemente uno diferente al usado para la interfaz web), y requerimos el uso de TLS para todas las conexiones con el servicio de FTP.

10.4.3 Almacenamiento y permisos

Nos queda por configurar usuarios y carpetas. Empezaremos por configurar el espacio de almacenamiento a usar, seguido de la gestión de usuarios, y finalizando con la configuración de carpetas y privilegios.

Primero es necesario configurar dónde se guardarán nuestros archivos. OpenMediaVault necesita discos aparte para el almacenamiento. Idealmente usaríamos varios discos en una configuración RAID, dependiendo del uso que vayamos a darle y la confiabilidad que necesitamos. Para este ejemplo, hemos usado un disco virtual y lo hemos conectado a la máquina virtual.

Accederemos a Almacenamiento > Discos físicos, y localizaremos el disco que acabamos de conectar.

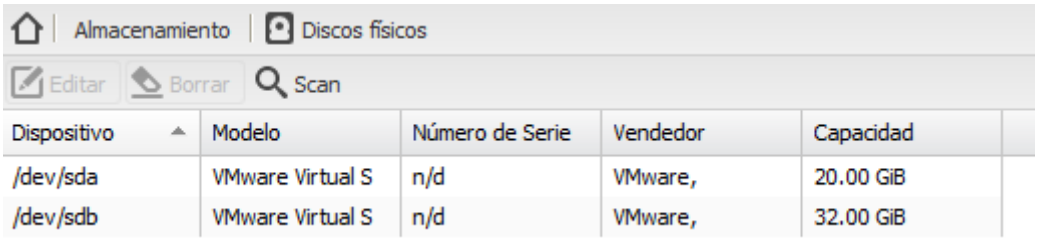


ILUSTRACIÓN 38: DISCOS NAS

Podemos ver que nos detecta dos discos. El primero pertenece al sistema, y no podemos emplearlo para almacenamiento de archivos. El particionamiento no está permitido. El segundo es el que acabamos de conectar y podemos emplear para almacenar nuestros archivos.

Accediendo a Almacenamiento > Sistemas de Archivos, podemos activar el segundo disco para almacenamiento. Lo montaremos y pasaremos a crear los usuarios y carpetas para el servidor FTP.

Empezaremos creando los usuarios. En Administración de permisos de usuario > Usuarios, podremos crear los usuarios para los servicios de NAS, aunque sólo los emplearemos para FTP. El proceso es similar a la creación de usuarios en Linux.

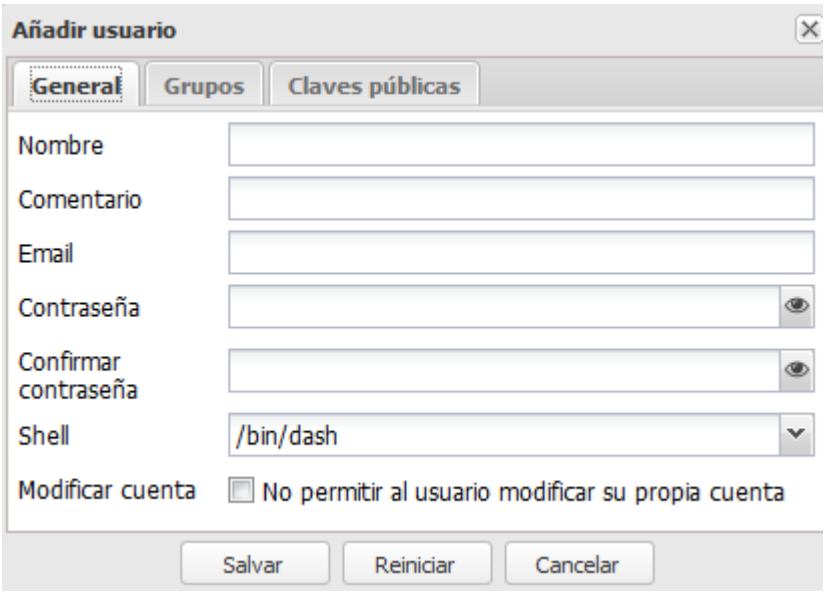


ILUSTRACIÓN 39: CREACIÓN DE USUARIOS

Si deseamos una configuración mínima, sólo es necesario que asignemos un nombre y contraseña, el resto es opcional. Tampoco es necesario que editemos la asignación a grupos, sobretodo si el usuario que creamos está unicamente destinado al uso del servicio FTP.

Por otro lado, una opción más recomendable es crear dos grupos, userFTP y adminFTP, y durante la creación de usuarios, asignar los usuarios al grupo al que pertenece. Esto nos permite más facilidad a la hora de asignar permisos a la carpeta, pudiendo asignar permisos a los grupos, o a un usuario en concreto.

Para este ejemplo, hemos creado el usuario dlc00017, perteneciente al grupo de adminFTP (con permisos de escritura y lectura).

A continuación, creamos las carpetas. Puesto que para este sistema, es imposible escribir en la raíz del sistema de archivos (ya que entraría en conflicto con el resto de servicios NAS que tambien emplean el mismo sistema de archivos), hemos creado una simple carpeta FTP. Para ello debemos ir a Administración de permisos de acceso > Carpetas Compartidas.

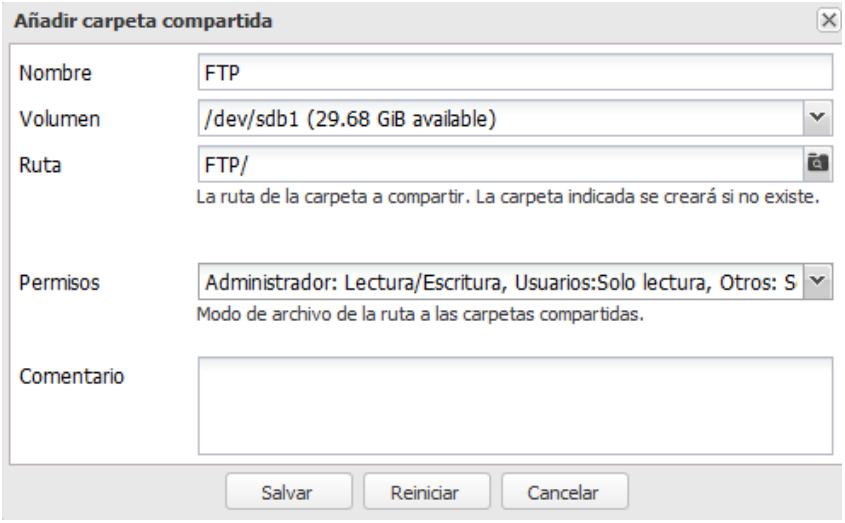


ILUSTRACIÓN 40: CARPETAS COMPARTIDAS

Necesitamos elegir un nombre de carpeta, el volumen a usar (nuestro segundo disco), una ruta para la carpeta, y los permisos. Los permisos a los que hace referencia son a los propios de los grupos de Linux, por los que nuestros grupos adminFTP y

userFTP no tienen permisos aún. Una vez creada la carpeta, seleccionándola, tendremos disponible la opción de privilegios.

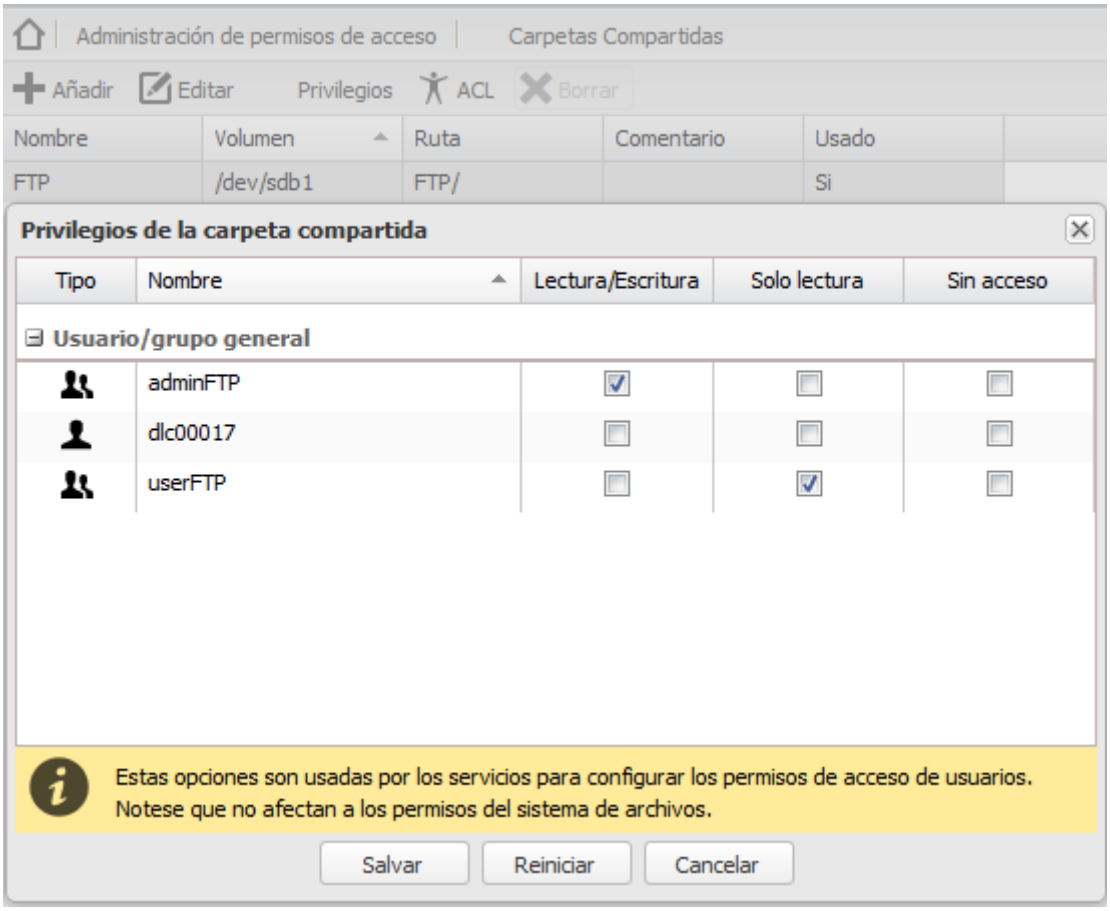


ILUSTRACIÓN 41: CONFIGURACIÓN DE PRIVILEGIOS

Asignaríamos los permisos tal como planteábamos anteriormente. De esta manera, la carpeta FTP permite a los administradores cargar y descargar archivos, mientras que los usuarios sólo pueden descargar.

En último lugar, habilitamos esta carpeta para su uso en el servicio FTP. Para ello volvemos al menú FTP, y accedemos a la opción de Compartidos. Una vez añadida la carpeta, aplicamos y guardamos los cambios. Nuestro servicio FTP está finalmente en funcionamiento.

11. Bibliografía

[1] Shackleford, Dave: *Virtualization Security: Protecting Virtualized Environments (1)*. Sybex, 2012.

[2] Luke S. Crawford, Chris Takemura: *Book of Xen: A Practical Guide for the System Administrator (1)*. No Starch Press, US, 2009.

[3] Bernard Golden: *Virtualization For Dummies®*. John Wiley & Sons, 2007.

[4] Sander van Vugt: *VMware Workstation - No Experience Necessary*. Packt Publishing, 2013.

[5] Al Muller, Seburn Wilson: *Virtualization with VMware ESX Server*. Syngress, 2005.

[6] Ryan Troy, Matthew Helmke: *VMware Cookbook, 2nd Edition*. O'Reilly Media, Inc., 2012.

[7] VMware: *Virtualization* [Online]. Disponible en: http://www.vmware.com/latam/solutions/virtualization.html

[8] Chris Dean*: How to install and configure NGINX on CentOS 7* [Online]. 2015. Disponible en: https://www.godaddy.com/garage/tech/config/how-to-install-and-configure-nginx-on-centos-7/

[9] Sean P. Conroy: *History of Virtualization* [Online]. 2011. Disponible en: http://www.everythingvm.com/content/history-virtualization

[10] Jon Brodkin: *With long history of virtualization behind it, IBM looks to the future* [Online]. 2009. Disponible en: http://www.networkworld.com/article/2254433/virtualization/with-long-history-of-virtualization-behind-it--ibm-looks-to-the-future.html

[11] Awodele Oludele, Emmanuel C. Ogu, Kuyoro ‘Shade, Umezuruike Chinecherem: *On the Evolution of Virtualization and Cloud Computing: A Review* [Online]. 2014. Disponible en: http://pubs.sciepub.com/jcsa/2/3/1/

[12] Kent Roberts, Richard Norwood: *Different Types of Virtualization* [Online]. 2013. Disponible en: https://www.superb.net/blog/2013/02/28/different-types-of-virtualization/

[13] Jukka Kouletsis: *The Basics of Application Packaging: Best Practices for Enabling Reduced Software Management Costs* [Online]. 2005. Disponible en: http://www.dell.com/downloads/global/power/ps4q05-20050175-Kouletsis.pdf

[14] Andy Patrizio: *Virtualization vs. Containers: What You Need to Know* [Online]. 2015. Disponible en: http://www.datamation.com/data-center/virtualization-vs.-containers-what-you-need-to-know.html

[15] Akshay Karle: *Operating System Containers vs. Application Containers* [Online]. 2015. Disponible en: https://blog.risingstack.com/operating-system-containers-vs-application-containers/

[16] Wilco van Bragt: *An Introduction to OS Virtualization* [Online]. 2008. Disponible en: http://www.virtualizationadmin.com/articles-tutorials/general-virtualization-articles/introduction-os-virtualization-part1.html

[17] Docker: *Understanding Docker* [Online]. Disponible en: https://docs.docker.com/engine/understanding-docker/

[18] Albert Reuther, Peter Michaleas, Andrew Prout, Jeremy Kepner: *HPC-VMs: Virtual Machines in High Performance Computing Systems* [Online]. 2012. Disponible en: https://www.ll.mit.edu/mission/cybersec/publications/publication-files/full_papers/2012_09_10_Reuther_HPEC_FP.pdf

[19] Oren Laadan, Jason Nieh: *Operating System Virtualization: Practice and Experience* [Online]. 2010. Disponible en: http://www.cs.columbia.edu/~orenl/papers/systor2010_osvirt.pdf

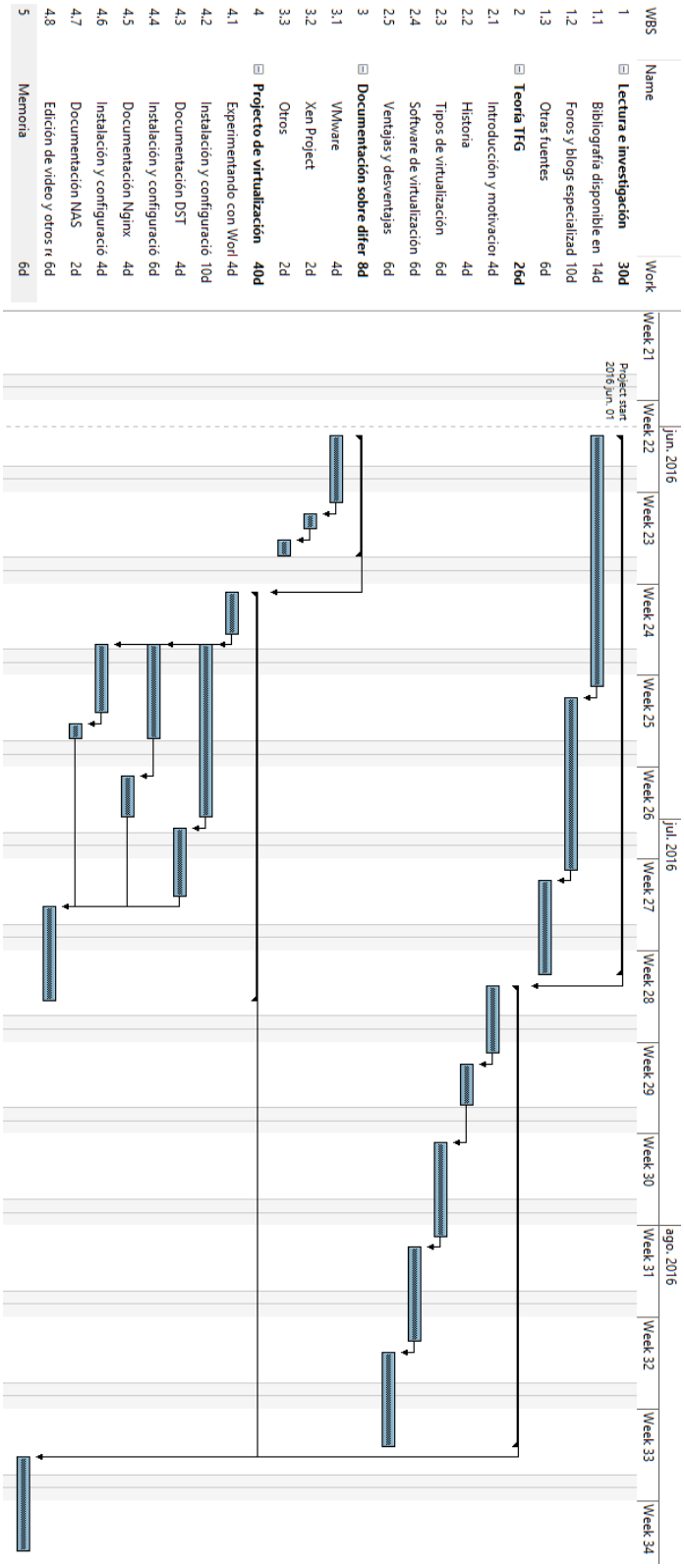
[20] Hasan Fayyad-Kazan, Luc Perneel, Martin Timmerman: *A Performance Comparison Between Enlightenment and Emulation in Microsoft Hyper-V* [Online]. 2013. Disponible en: https://globaljournals.org/GJCST_Volume13/4-A-Performance-Comparison.pdf

[21] Frank Bunn, Nik Simpson, Robert Peglar, Gene Nagle: *Storage Virtualization* [Online]. 2003. Disponible en: http://www.snia.org/sites/default/files/sniavirt.pdf

[22] Multicians: *History of MultiCS* [Online]. 2016. Disponible en: http://www.multicians.org/history.html

12. Diagrama de Gantt

Diagrama de Gantt indicando el reparto de tiempo dedicado a la realización de este trabajo.



13. Anexo I: Master/worldgenoverride.lua

```
return {
override_enabled = true,
preset = "SURVIVAL_TOGETHER",       -- "SURVIVAL_TOGETHER", "MOD_MISSING",
"SURVIVAL_TOGETHER_CLASSIC", "SURVIVAL_DEFAULT_PLUS", "COMPLETE_DARKNESS",
"DST_CAVE", "DST_CAVE_PLUS"
overrides = {
-- MISC
task_set = "default",        -- "classic", "default", "cave_default"
start_location = "default",       -- "caves", "default", "plus", "darkness"
world_size = "default",        -- "small", "medium", "default", "huge"
branching = "default",       -- "never", "least", "default", "most"
loop = "default",       -- "never", "default", "always"
autumn = "default",       -- "noseason", "veryshortseason", "shortseason",
"default", "longseason", "verylongseason", "random"
winter = "default",       -- "noseason", "veryshortseason", "shortseason",
"default", "longseason", "verylongseason", "random"
spring = "default",       -- "noseason", "veryshortseason", "shortseason",
"default", "longseason", "verylongseason", "random"
summer = "default",       -- "noseason", "veryshortseason", "shortseason",
"default", "longseason", "verylongseason", "random"
season_start = "default",         -- "default", "winter", "spring", "summer",
"autumnorspring", "winterorsummer", "random"
day = "default",        -- "default", "longday", "longdusk", "longnight", "noday",
"nodusk", "nonight", "onlyday", "onlydusk", "onlynight"
weather = "default",         -- "never", "rare", "default", "often", "always"
earthquakes = "default",       -- "never", "rare", "default", "often", "always"
lightning = "default",       -- "never", "rare", "default", "often", "always"
frograin = "default",        -- "never", "rare", "default", "often", "always"
wildfires = "default",       -- "never", "rare", "default", "often", "always"
touchstone = "default",        -- "never", "rare", "default", "often", "always"
regrowth = "default",        -- "veryslow", "slow", "default", "fast", "veryfast"
cavelight = "default",       -- "veryslow", "slow", "default", "fast", "veryfast"
boons = "default",        -- "never", "rare", "default", "often", "always"
prefabswaps_start = "default",         -- "classic", "default", "highly random"
prefabswaps = "default",       -- "none", "few", "default", "many", "max"

-- RESOURCES
flowers = "default",         -- "never", "rare", "default", "often", "always"
grass = "default",        -- "never", "rare", "default", "often", "always"
sapling = "default",         -- "never", "rare", "default", "often", "always"
marshbush = "default",       -- "never", "rare", "default", "often", "always"
tumbleweed = "default",        -- "never", "rare", "default", "often", "always"
reeds = "default",        -- "never", "rare", "default", "often", "always"
trees = "default",        -- "never", "rare", "default", "often", "always"
flint = "default",        -- "never", "rare", "default", "often", "always"
rock = "default",       -- "never", "rare", "default", "often", "always"
rock_ice = "default",        -- "never", "rare", "default", "often", "always"
meteorspawner = "default",        -- "never", "rare", "default", "often", "always"
meteorshowers = "default",        -- "never", "rare", "default", "often", "always"
mushtree = "default",        -- "never", "rare", "default", "often", "always"
fern = "default",       -- "never", "rare", "default", "often", "always"
flower_cave = "default",       -- "never", "rare", "default", "often", "always"
wormlights = "default",        -- "never", "rare", "default", "often", "always"

-- UNPREPARED
berrybush = "default",       -- "never", "rare", "default", "often", "always"
carrot = "default",       -- "never", "rare", "default", "often", "always"
```

```
mushroom = "default",       -- "never", "rare", "default", "often", "always"
cactus = "default",       -- "never", "rare", "default", "often", "always"
banana = "default",       -- "never", "rare", "default", "often", "always"
lichen = "default",       -- "never", "rare", "default", "often", "always"

-- ANIMALS
rabbits = "default",        -- "never", "rare", "default", "often", "always"
moles = "default",       -- "never", "rare", "default", "often", "always"
butterfly = "default",      -- "never", "rare", "default", "often", "always"
birds = "default",       -- "never", "rare", "default", "often", "always"
buzzard = "default",        -- "never", "rare", "default", "often", "always"
catcoon = "default",        -- "never", "rare", "default", "often", "always"
perd = "default",      -- "never", "rare", "default", "often", "always"
pigs = "default",      -- "never", "rare", "default", "often", "always"
lightninggoat = "default",        -- "never", "rare", "default", "often", "always"
beefalo = "default",        -- "never", "rare", "default", "often", "always"
beefaloheat = "default",       -- "never", "rare", "default", "often", "always"
hunt = "default",      -- "never", "rare", "default", "often", "always"
alternatehunt = "default",        -- "never", "rare", "default", "often", "always"
penguins = "default",       -- "never", "rare", "default", "often", "always"
cave_ponds = "default",        -- "never", "rare", "default", "often", "always"
ponds = "default",       -- "never", "rare", "default", "often", "always"
bees = "default",      -- "never", "rare", "default", "often", "always"
angrybees = "default",      -- "never", "rare", "default", "often", "always"
tallbirds = "default",      -- "never", "rare", "default", "often", "always"
slurper = "default",        -- "never", "rare", "default", "often", "always"
bunnymen = "default",       -- "never", "rare", "default", "often", "always"
slurtles = "default",       -- "never", "rare", "default", "often", "always"
rocky = "default",       -- "never", "rare", "default", "often", "always"
monkey = "default",      -- "never", "rare", "default", "often", "always"

-- MONSTERS
spiders = "default",        -- "never", "rare", "default", "often", "always"
cave_spiders = "default",         -- "never", "rare", "default", "often", "always"
hounds = "default",       -- "never", "rare", "default", "often", "always"
houndmound = "default",        -- "never", "rare", "default", "often", "always"
merm = "default",      -- "never", "rare", "default", "often", "always"
tentacles = "default",      -- "never", "rare", "default", "often", "always"
chess = "default",       -- "never", "rare", "default", "often", "always"
lureplants = "default",        -- "never", "rare", "default", "often", "always"
walrus = "default",      -- "never", "rare", "default", "often", "always"
liefs = "default",       -- "never", "rare", "default", "often", "always"
deciduousmonster = "default",       -- "never", "rare", "default", "often",
"always"
krampus = "default",        -- "never", "rare", "default", "often", "always"
bearger = "default",        -- "never", "rare", "default", "often", "always"
deerclops = "default",      -- "never", "rare", "default", "often", "always"
goosemoose = "default",        -- "never", "rare", "default", "often", "always"
dragonfly = "default",      -- "never", "rare", "default", "often", "always"
bats = "default",      -- "never", "rare", "default", "often", "always"
fissure = "default",        -- "never", "rare", "default", "often", "always"
worms = "default",       -- "never", "rare", "default", "often", "always"
},
}
```

14. Anexo II: Caves/worldgenoverride.lua

```
return {
override_enabled = true,
preset = "DST_CAVE",                -- "SURVIVAL_TOGETHER", "MOD_MISSING",
"SURVIVAL_TOGETHER_CLASSIC", "SURVIVAL_DEFAULT_PLUS", "COMPLETE_DARKNESS",
"DST_CAVE", "DST_CAVE_PLUS"
overrides = {
-- MISC
task_set = "cave_default",      -- "classic", "default", "cave_default"
start_location = "default",       -- "caves", "default", "plus", "darkness"
world_size = "default",         -- "small", "medium", "default", "huge"
branching = "default",       -- "never", "least", "default", "most"
loop = "default",       -- "never", "default", "always"
autumn = "default",       -- "noseason", "veryshortseason", "shortseason",
"default", "longseason", "verylongseason", "random"
winter = "default",       -- "noseason", "veryshortseason", "shortseason",
"default", "longseason", "verylongseason", "random"
spring = "default",       -- "noseason", "veryshortseason", "shortseason",
"default", "longseason", "verylongseason", "random"
summer = "default",       -- "noseason", "veryshortseason", "shortseason",
"default", "longseason", "verylongseason", "random"
season_start = "default",         -- "default", "winter", "spring", "summer",
"autumnorspring", "winterorsummer", "random"
day = "default",        -- "default", "longday", "longdusk", "longnight", "noday",
"nodusk", "nonight", "onlyday", "onlydusk", "onlynight"
weather = "default",         -- "never", "rare", "default", "often", "always"
earthquakes = "default",       -- "never", "rare", "default", "often", "always"
lightning = "default",            -- "never", "rare", "default", "often", "always"
frograin = "default",             -- "never", "rare", "default", "often", "always"
wildfires = "default",       -- "never", "rare", "default", "often", "always"
touchstone = "default",        -- "never", "rare", "default", "often", "always"
regrowth = "default",       -- "veryslow", "slow", "default", "fast", "veryfast"
cavelight = "default",       -- "veryslow", "slow", "default", "fast", "veryfast"
boons = "default",        -- "never", "rare", "default", "often", "always"
prefabswaps_start = "default",              -- "classic", "default", "highly
random"
prefabswaps = "default",            -- "none", "few", "default", "many", "max"

-- RESOURCES
flowers = "default",         -- "never", "rare", "default", "often", "always"
grass = "default",        -- "never", "rare", "default", "often", "always"
sapling = "default",         -- "never", "rare", "default", "often", "always"
marshbush = "default",       -- "never", "rare", "default", "often", "always"
tumbleweed = "default",        -- "never", "rare", "default", "often", "always"
reeds = "default",        -- "never", "rare", "default", "often", "always"
trees = "default",        -- "never", "rare", "default", "often", "always"
flint = "default",        -- "never", "rare", "default", "often", "always"
rock = "default",       -- "never", "rare", "default", "often", "always"
rock_ice = "default",        -- "never", "rare", "default", "often", "always"
meteorspawner = "default",        -- "never", "rare", "default", "often", "always"
meteorshowers = "default",        -- "never", "rare", "default", "often", "always"
mushtree = "default",        -- "never", "rare", "default", "often", "always"
fern = "default",       -- "never", "rare", "default", "often", "always"
flower_cave = "default",       -- "never", "rare", "default", "often", "always"
wormlights = "default",        -- "never", "rare", "default", "often", "always"

-- UNPREPARED
berrybush = "default",       -- "never", "rare", "default", "often", "always"
```

```
carrot = "default",       -- "never", "rare", "default", "often", "always"
mushroom = "default",        -- "never", "rare", "default", "often", "always"
cactus = "default",       -- "never", "rare", "default", "often", "always"
banana = "default",       -- "never", "rare", "default", "often", "always"
lichen = "default",       -- "never", "rare", "default", "often", "always"

-- ANIMALS
rabbits = "default",         -- "never", "rare", "default", "often", "always"
moles = "default",        -- "never", "rare", "default", "often", "always"
butterfly = "default",       -- "never", "rare", "default", "often", "always"
birds = "default",        -- "never", "rare", "default", "often", "always"
buzzard = "default",         -- "never", "rare", "default", "often", "always"
catcoon = "default",         -- "never", "rare", "default", "often", "always"
perd = "default",       -- "never", "rare", "default", "often", "always"
pigs = "default",       -- "never", "rare", "default", "often", "always"
lightninggoat = "default",        -- "never", "rare", "default", "often", "always"
beefalo = "default",         -- "never", "rare", "default", "often", "always"
beefaloheat = "default",        -- "never", "rare", "default", "often", "always"
hunt = "default",       -- "never", "rare", "default", "often", "always"
alternatehunt = "default",        -- "never", "rare", "default", "often", "always"
penguins = "default",             -- "never", "rare", "default", "often", "always"
cave_ponds = "default",         -- "never", "rare", "default", "often", "always"
ponds = "default",        -- "never", "rare", "default", "often", "always"
bees = "default",       -- "never", "rare", "default", "often", "always"
angrybees = "default",            -- "never", "rare", "default", "often", "always"
tallbirds = "default",            -- "never", "rare", "default", "often", "always"
slurper = "default",         -- "never", "rare", "default", "often", "always"
bunnymen = "default",        -- "never", "rare", "default", "often", "always"
slurtles = "default",        -- "never", "rare", "default", "often", "always"
rocky = "default",        -- "never", "rare", "default", "often", "always"
monkey = "default",       -- "never", "rare", "default", "often", "always"

-- MONSTERS
spiders = "default",         -- "never", "rare", "default", "often", "always"
cave_spiders = "default",         -- "never", "rare", "default", "often", "always"
hounds = "default",       -- "never", "rare", "default", "often", "always"
houndmound = "default",         -- "never", "rare", "default", "often", "always"
merm = "default",       -- "never", "rare", "default", "often", "always"
tentacles = "default",       -- "never", "rare", "default", "often", "always"
chess = "default",        -- "never", "rare", "default", "often", "always"
lureplants = "default",         -- "never", "rare", "default", "often", "always"
walrus = "default",       -- "never", "rare", "default", "often", "always"
liefs = "default",        -- "never", "rare", "default", "often", "always"
deciduousmonster = "default",        -- "never", "rare", "default", "often",
"always"
krampus = "default",         -- "never", "rare", "default", "often", "always"
bearger = "default",         -- "never", "rare", "default", "often", "always"
deerclops = "default",            -- "never", "rare", "default", "often", "always"
goosemoose = "default",         -- "never", "rare", "default", "often", "always"
dragonfly = "default",            -- "never", "rare", "default", "often", "always"
bats = "default",       -- "never", "rare", "default", "often", "always"
fissure = "default",         -- "never", "rare", "default", "often", "always"
worms = "default",        -- "never", "rare", "default", "often", "always"
},
}
```