Institut Puig Castellar
Santa Coloma de Gramenet

Consolidación de servidores con oVirt

Memoria

CFGS Administración de Sistemas Informáticos y Redes

Enrique Villarreal

Curso **2015-2016**
3 de junio de 2016

Resumen

En este proyecto se explorarán herramientas de gestión del *datacenter*, es decir, herramientas que integran la gestión de máquinas virtuales, almacenamiento, y red al mismo tiempo. El principal objetivo del proyecto es conseguir una mayor disponibilidad y garantizar la robustez de los servicios que actualmente se ofrecen en el Institut La Bastida.

Partiendo del modelo de funcionamiento actual, se realizará un estudio concretando los argumentos para migrar del modelo actual a uno orientado a la virtualización, respaldando la decisión de implementar la herramienta escogida.

Una vez implementada, se realizarán las pruebas pertinentes para garantizar los servicios mencionados, particularmente en las áreas de migración de máquinas virtuales / alta disponibilidad, y copias de seguridad.

Palabras clave: virtualización, servidores, linux

Abstract

In this project, I will explore datacenter management tools, that integrate server, storage, and network management in the same package. The main objective of this project is to achieve higher reliability and secure the services that are being currently usedat INS La Bastida.

Starting from the actual architecutre, a study will be conducted outlining arguments to switch from the current model to a virtualitzation-oriented one, backing up the chosen tool.

Once it is implemented, tests will be carried out, focusing primarily on virtual machine migration / high availability, and backups.

Keywords: virtualization, servers, linux

Índice general

Índice de figuras

0.1. Introducción

0.1.1. Contexto y justificación

La virtualización no es nada nuevo. Tiene sus orígenes en la respuesta de IBM al pryecto MAC (originalmente *Mathematics and Computation*, renombrado más tarde a *Multiple Access Computer*), que pretendía soportar más de un usuario de forma simultánea. Aunque los sistemas de hoy en día soportan cargas de trabajo multiusuario sin ningún problema, en aquellos tiempos cada usuario realizaba su computación en lotes secuenciales.

Para hacer realidad ése proyecto, el MIT necesitaba un hardware más potente del cual disponían, y se acercó a diferentes fabricantes. El proceso culminó con la elección de GE como proveedor, ya que IBM no veía futuro en *mainframes* concurrentes. Ésta oportunidad perdida sirvió para que IBM reaccionara, que a raíz de ello inició a desarrollar el CP-40, que acabaría llevando al CP-67, el primer mainframe que soportaba virtualización.

De todas formas, la virtualización aplicada a servicio pasó a un segundo plano a causa de la evolución meteórica de la informática durante los años 80 y 90, aademás del relativo bajo precio de servidores basados en la arquitectura x86. Con el paso del tiempo, fueron surgiendo diferentes soluciones de virtualización de escritorio, pero el modelo de computación distribuida se estableció como el estándar de la industria.

Actualmente, los servidores de La Bastida están todos implementados en un único servidor físico. Aunque esto funciona bien, constituye un *SPOF* que en caso de fallar, dejaría al centro sin servicio. Éste proyecto, entonces, tiene por objetivo explorar el uso de la virtualización para mantener a flote los servicios en caso de que el servidor que los provee falle.

0.1.2. Listado de objetivos

Los objetivos principales del proyecto son:

- Estudiar el uso de la virtualización como medio para asegurar servicios
- Estudiar herramientas de gestió de virtualització y del *datacenter*.
- Asegurar los servicios del IES La Bastida mediante máquinas virtuales.
- Obtener la posibilidad de migrar máquinas virtuales.
- Automatizar las copias de seguridad de las máquinas virtuales.
- Simplificar la gestión de los servicios del instituto.

0.1.3. Metodología

El enfoque del proyecto se basará en dos partes principales: el estudio del arte, en el cual se explorará la arquitectura actual de los servicois del instituto y s eestudiará la virtualización como medio para ofrecer servicios, y la implementación de la herramienta propiamente dicha.

Éste estudio del arte proporcionará argumentos defendiedno un modelo en el cual se emplean máquinas virtuales en lugar de servidores físicos adicionales, y explorará también las diferentes opciones de las cuales disponemos para llevar a cabo éste proyecto.

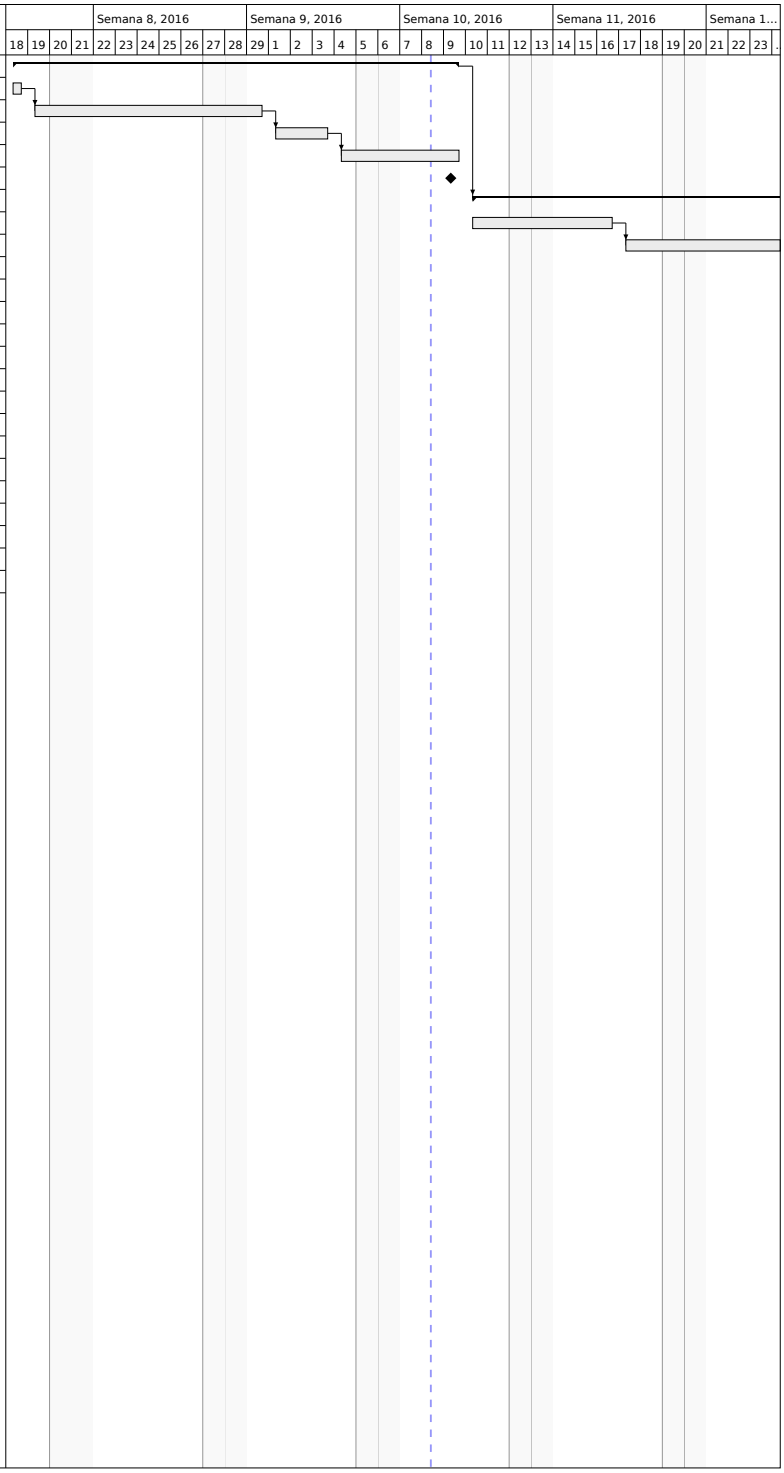
En principio, sólo se implementará una opción, siendo la memoria una descripción del uso de la misma, sin ser necesariamente un manual de usuario. Si el tiempo alcanza, y no hay problemas que me obliguen a modificar éste plan, se tratará de estudiar herramientas adicionales, ofreciendo una pequeña comparativa con la herramienta escogida para el proyecto.

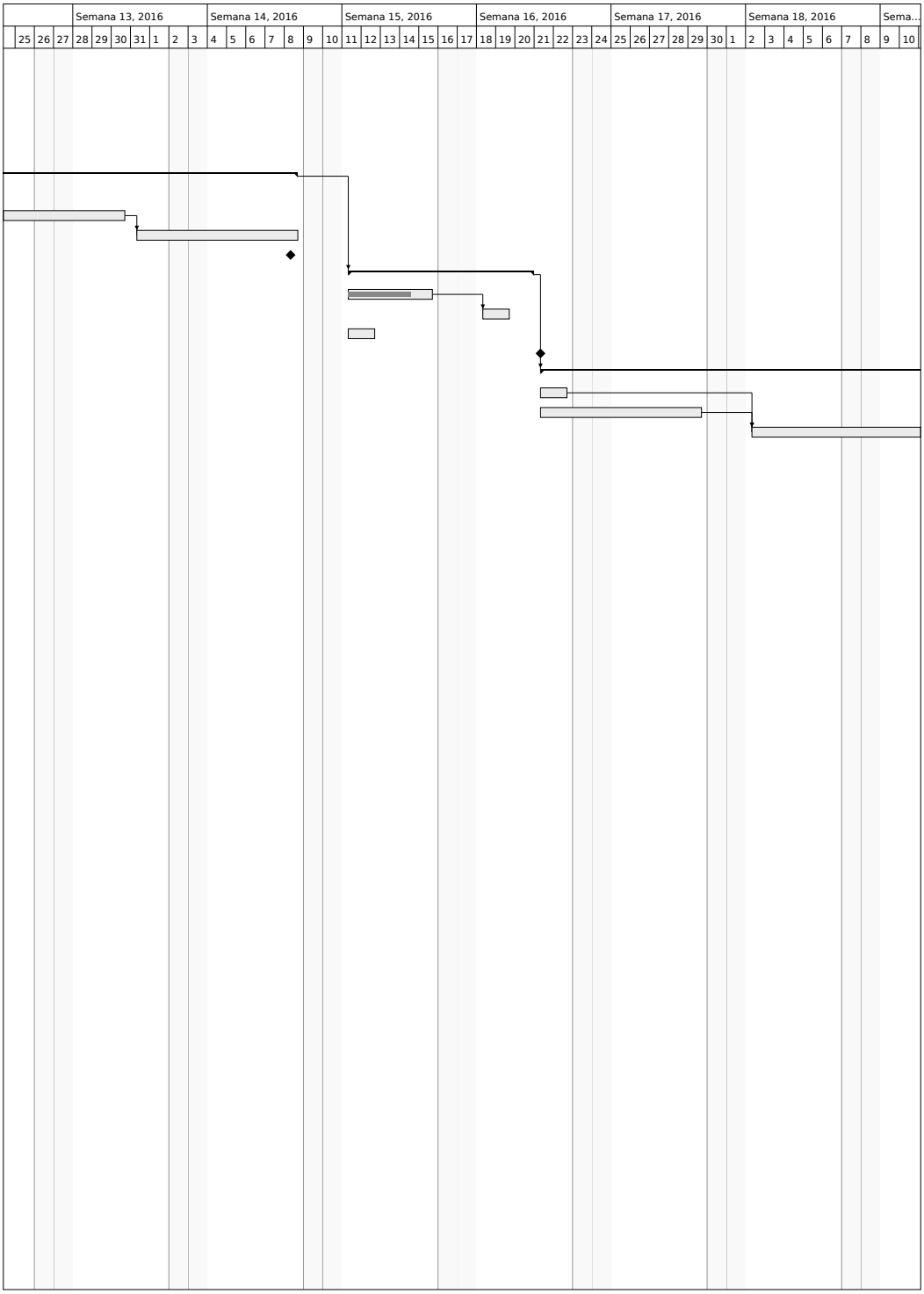
0.1.4. Planificación del proyecto

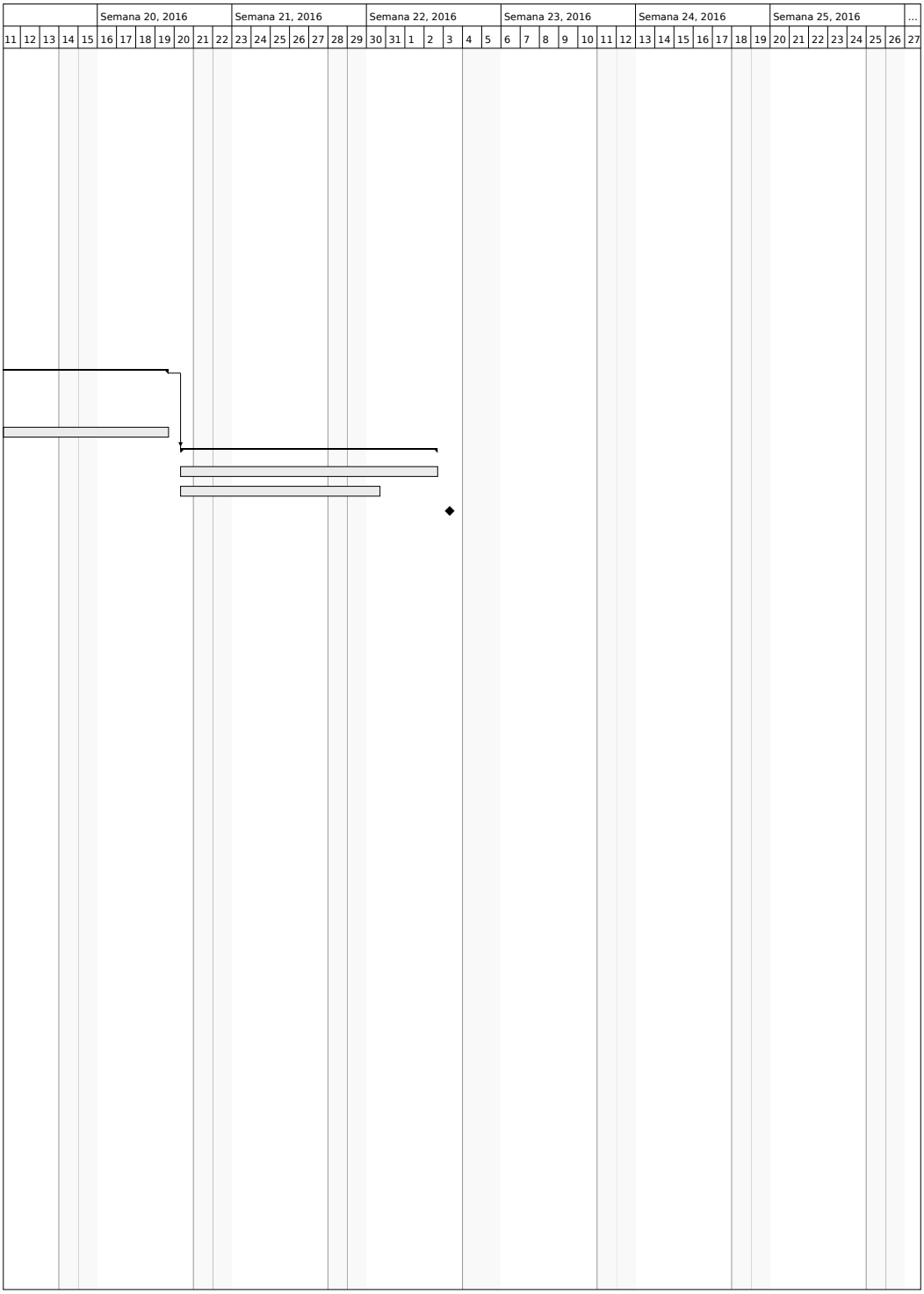
0.1.4.1. Lista de tareas

WBS	Nombre	Inicio	Fin	Trabajo	Duración	Desperdicio	Coste	Asignado a	% Completado
1	**Definición del proyecto**	**feb 18**	**mar 9**	**15d**	**15d**	**1d**	**0**		0
1.1	Elección del tema	feb 18	feb 18	1d	1d	1d	0		0
1.2	Definición del tema	feb 19	feb 29	7d	7d	1d	0		0
1.3	Identificación de tareas	mar 1	mar 3	3d	3d	1d	0		0
1.4	Análisis de riesgos	mar 4	mar 9	4d	4d	1d	0		0
1.5	Entrega #1	mar 9	mar 9	N/D	N/D	2d	0		0
2	**Estado del arte**	**mar 10**	**abr 8**	**22d**	**22d**		**0**		0
2.1	Toma de requisitos	mar 10	mar 16	5d	5d	1d	0		0
2.2	Análisis de arquitectura	mar 17	mar 30	10d	10d	1d	0		0
2.3	Descripción de componentes	mar 31	abr 8	7d	7d		0		0
2.4	Entrega #2	abr 8	abr 8	N/D	N/D	1d	0		0
3	**Configuración de la arquitectura**	**abr 11**	**abr 21**	**9d**	**8d**		**0**		0
3.1	Configuración del Motor	abr 11	abr 15	5d	5d		0		75
3.2	Configuración NFS	abr 18	abr 19	2d	2d	1d	0		0
3.3	Configuración hosts	abr 11	abr 12	2d	2d	6d	0		0
3.4	Entrega #3	abr 21	abr 21	N/D	N/D		0		0
4	**Máquinas virtuales**	**abr 21**	**may 19**	**23d**	**21d**		**0**		0
4.1	Servicios de red	abr 21	abr 22	2d	2d	5d	0		0
4.2	Servicios web	abr 21	abr 29	7d	7d		0		0
4.3	Proxmox VE	may 2	may 19	14d	14d		0		0
5	**Tests**	**may 20**	**jun 2**	**17d**	**10d**		**0**		0
5.1	Backups	may 20	jun 2	10d	10d		0		0
5.2	Migraciones	may 20	may 30	7d	7d	3d	0		0
6	Entrega final	jun 3	jun 3	N/D	N/D		0		0

Nombre	Trab...
Definición del proyecto	**15d**
Elección del tema	1d
Definición del tema	7d
Identificación de tareas	3d
Análisis de riesgos	4d
Entrega #1	
Estado del arte	**22d**
Toma de requisitos	5d
Análisis de arquitectura	10d
Descripción de componentes	7d
Entrega #2	
Configuración de la arquitectu	**9d**
Configuración del Motor	5d
Configuración NFS	2d
Configuración hosts	2d
Entrega #3	
Máquinas virtuales	**23d**
Servicios de red	2d
Servicios web	7d
Proxmox VE	14d
Tests	**17d**
Backups	10d
Migraciones	7d
Entrega final	




Semana 13, 2016
Semana 14, 2016
Semana 15, 2016
Semana 16, 2016
Semana 17, 2016
Semana 18, 2016
Sema...
25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10


Semana 20, 2016
Semana 21, 2016
Semana 22, 2016
Semana 23, 2016
Semana 24, 2016
Semana 25, 2016
...
11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

0.1.4.2. Planificación horaria

Debido a que el hardware necesario para la realización del proyecto se encuentra en el IES La Bastida (Nota: aunque en el anexo hay documentada una manera de acceder a la consola de administración a través de un proxy SOCKS, una retahíla de errores y descalabros a nivel físico han hecho muy difícil el trabajo de manera remota.) únicamente dispondré de tres horas durante los días lectivos para la realización del proyecto. El resto de horas del MP14 se dedicarán a investigar documentación, redactado de la memoria, y tareas relacionadas con la memoria y la presentación del proyecto.

A eso de Mayo de 2016, entré a trabajar para Coopux Systems, lo que me impide continuar cualquier tipo de trabajo en el proyecto. Por suerte, oVirt ya había sido probado y documentado, así que el impacto en el proyecto es relativamente pequeño, limitándose a la imposibilidad de documentar otras herramientas, y profundizar en la migración de máquinas virtuales.

0.1.5. Descripción de los capítulos

1. **Estado del arte:** En éste capítulo se analizará la historia y el estado actual de la virtualización, se estudiará la arquitectura actual de La Bastida, y mi propuesta para el cambio.
2. **Implementación de oVirt:** En éste capítulo, se documentará el proceso de instalación de oVirt.
3. **Trabajando con VMs en oVirt:** En éste capítulo se documentará brevemente algunas de las tareas esenciales a la hora de trabajar con máquinas virtuales
4. **Conclusiones:** En éste capítulo se resumirán mis conclusiones y opiniones personales acerca del proyecto.

Capítulo 1

Estado del arte, requisitos y arquitectura

1.1. Virtualización de servidores

Como ya dije antes, la virtualización aplicada a servidores no es nada que haya surgido los últimos años. Pero cuando IBM comenzó a trabajar con los primeros *mainframes* concurrentes, su propósito era meramente ofrecer a sus usuarios una manera de trabajar al mismo tiempo en un mundo que, en aquellos tiempos, era puramente secuencial.

Hoy en día la virtualización ha ido evolucionando gradualmente, resultando en diferentes técnicas y métodos que permiten sustituir de manera eficaz y flexible las máquinas físicas. Cada método funciona de una manera diferente, con sus ventajas y desventajas, que iré explorando brevemente en las siguientes secciones:

1.1.1. Cómo opera el instituto actualmente

Antes de discutir los pros y los contras de los diferentes métodos de virtualizacion, veamos cómo se ofrecen los servicios actualmente: en una máquina personalizada con las siguientes características:

- **CPU:** Intel Xeon E-1220 @ 3.10 GHz
- **RAM:** 8 GB

Considerando los servicios ofrecidos (Un par de servidores web con la página web del centro y un Owncloud, y un servidor DHCP.), ésta máquina está lo suficientemente preparada para dichos servicios, pero opino que se pueden ofrecer de manera más eficiente. A continuación, listaré los que a mi juicio son los principales detrimentos de éste sistema:

- **Aprovechamiento mínimo de recursos:** Aunque ésto se vea atenuado de cierto modo en este caso, los recursos de una máquina raramente se aprovechan del todo, ya que un servicio no hará uso de todo el hardware disponible.
- **Ningún tipo de aislamiento entre servicios:** Debido a que todos los servicios se están ejecutando en el mismo sistema operativo sin ningún mecanismo de aislamiento, cualquier problema grave en cualquiera de los servicios podría comprometer la disponibilidad de los otros servicios.
- **Costos de expansión:** Expandir ésta arquitectura puede resultar muy costoso, tanto en materia de hardware, electricidad, y acondicionamiento de la sala de servidores.

1.1.2. Ventajas de la virtualización

Incluso si éstos no parecen tan malos, todavía son suficientemente crtíicos para considerar un cambio a un sistema basado en máquinas virtuales. Algunas de las ventajas son:

- **Aislamiento de servicios:** Las máquinas virtuales son, en esencia, ordenadores enteros emulados dentro de una máquina física, así que el resultado es el mismo que antes: Estamos ejecutando cada servicio dentro de su propia máquina. Esto asegura que un servicio que falle no puede tumbar otros servicios, asumiendo que éstos se estén ejecutando en una máquina virtual diferente.
- **Ahorro:** Ésta tranquilamente podría ser la razón principal. No sólo baja el coste del hardware adicional, pero también resulta en menos mantenimiento, menor coestes eléctricos, y otras tareas relacionadas con el espacio de trabajo.
- **Un ciclo de trabajo más rápido:** Comparad el tiempo que tardáis en crear una nueva máquina virtual, y en preparar un ordenador físico al punto que es sea capaz de ofrecer un servicio. Ésto nos permite usar el sistema de virtualización tanto para servicios en producción, y experimentos y pruebas variadas.
- **Menor tiempo de indisponibilidad:** Combinando técnicas de ahorro de energía y aseguramiento eléctrico como SAIs, y el uso de un sistema de virtualización que lo soporte, las máquinas virtuales pueden ser migradas en caliente entre diferentes hosts, permitiendo la realización de tareas de mantenimiento sin que los usuarios se den cuenta.
- **Recuperación acelerada en caso de desastre:** Las máquinas virtuales pueden ser respaldadas a ficheros, y re-importadas en un servidor diferente en caso de emergencia. Herramientas como oVirt pueden incluso reiniciar máquinas fallidas en otro host diferente si el que previamente virtualizaba la máquina falla.

¡Importante!

Pese a todo ésto, sigue siendo muy importante el uso de sistemas de alimentación ininterrumpida para asegurar el equipo de red y los ordenadores y minimizar los SPOF.

1.1.3. Métodos de virtualización

Después de haberle echado una ojeada a los pros y contras de la virtualización frente al modelo distribuido, veamos de qué maneras podemos crear y utilizar máquinas virtuales:

1.1.3.1. Virtualització de hardware completa

La virtualización de hardware completa es un método en el cual la máquina virtual resultante es una simulación entera de todo el hardware de un ordenador físico, incluyendo elementos como el juego de instrucciones de la CPU, interrupciones, y otros elementos que permiten que los programas se ejecuten sobre el *bare metal*.

La virtualización completa en x86 fue inicialmente desarollada por VMware en 1998, mediante técnicas de traducción binaria, y ejecucción directa. Para entender mejor éste aspecto, es necesario conocer un poquito de la arquitectura x86, en concreto los **anillos de protección** [1]: una serie de "niveles", que definen y aíslan los privilegios de los que cierto código dispone. Típicamente, las aplicaciones de usuario se ejecutan sobre el nivel 3, el menos privilegiado, y código referente a la gestión de la memoria, o cualquier otras funciones del sistema operativo, en el nivel 0, el más privilegiado.

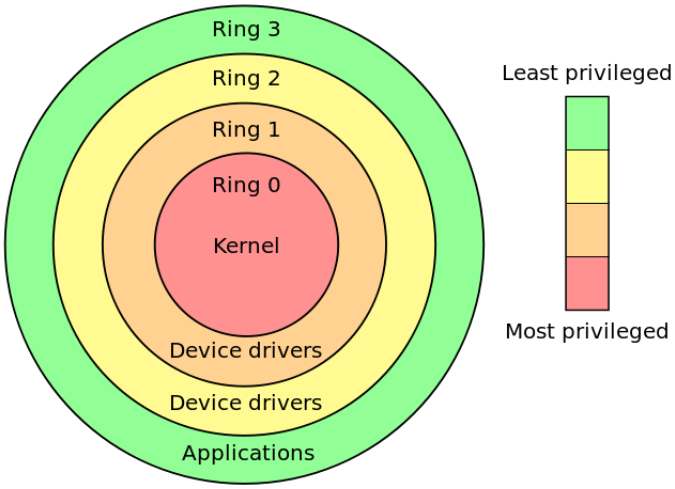


Figura 1.1: Anillos de protección, por Hertzsprung en Wikipedia

Entonces, '?en qué consiste la técnica que VMware utilizó en un primer momento para conseguir la virtualización de sistemas x86? Explicándolo de manera sencilla, consiste en ejecutar el sistema virtualizado en el anillo 1, y traduciendo el código no-virtualizable, a instrucciones que obtienen el mismo resultado en el hardware virtual. Dado el hecho de que el código de usuario resulta igual, éste se ejecuta directamente en el procesador para obtener más rendimiento.

Todo ésto en conjunto resulta en el sistema operativo completamente abstraído del hardware físico. A parte del aislamiento que así se obtiene, este método de virtualización no requiere ningún tipo de modificación en el sistema operativo para funcionar.

Algunos ejemplos de plataformas de virtualización completa son:

- Parallels Workstation (Mac OS X)
- VMware Workstation (Windows, GNU/Linux)
- Oracle VM VirtualBox (Windows, Mac OS X, GNU/Linux)

1.1.3.2. Paravirtualització

La paravirtualización es una técnica de emulación consistente en ofrecer una interfaz por software, que aunque no sea lo mismo que el hardware real, resulta bastante similar. El propósito de dicha interfaz es reducir el tiempo que el host virtualizado gasta realizando operaciones que resultan más complicadas de realizar en un entorno virtualizado comparado a un entorno normal. Esta técnica requiere que el sistema operativo virtualizado se haya portado a la API de paravirtualización, ya que un sistema en desconocimiento de ésta interfaz no puede hacerse funcionar bajo un software de paravirtualización.

En GNU/Linux la paravirtualitzación está disponible a partir de la versión 2.6.23 del kernel, y KVM puede utilizar la API VirtIO para virtualizar GNU/Linux, OpenBSD, FreeBSD, NetBSD, Plan 9 y Windows.

1.1.3.3. Virtualització asistida por hardware

La virtualización asistida por hardware se diferencia de la virtualización completa por el hecho de que consigue un mayor rendimiento gracias a la ayuda del hardware físico de la máquina anfitriona, principalmente el procesador. Ésto se basa en las siguientes teconologías:

- En procesadores Intel, **Intel VT-x** (introducida por primera vez en los modelos 662 y 672 de Pentium 4 el 20 de noviembre de 2005), es la tecnología de virtualización x86 asistida por hardware ofrecida por Intel. Aunque a día de hoy todas las CPUs de Intel a excepción de modelos de bajo consumo y móviles soportan ésta teconología, en GNU/Linux se puede comprobar el soporte de la CPU actual mediante el siguiente comando: `cat /proc/cpuinfo | grep vmx`. A lo largo del tiempo Intel ha ido añadiendo nuevas funcionalidades a VT-x, siendo las ma importantes la adición

de EPT para la emulación de tablas de página con la arquitectura Nehalem en 2008, el arranque del procesaodr lógico en modo real con la arquitectura Westmere en 2010, y el *shadowing* de VMCS con Haswell en 2013.

- En procesadores AMD, **AMD-V** es la tecnología de virtualización asistida por hardware de AMD, introducida por primera vez con los Athlon 64, 64 FX y 64 X2 en 2006.

Todo esto suena muy bien, pero cuando éstas tecnologías surgieron inicialmente, no ofrecían ventajas tangibles sobre la virtualización por software. Además, una estrategia exclusivamente de virtualización asistida por software implica bastantes trucos de implementación en la máquina virtual, provocando gran *overhead* de CPU, limitando la eficiencia de la máquina virtual. Ésto se puede solventar mediante el uso de drivers paravirtualizados.

1.1.3.4. Virtualitzación a nivel de sistema operativo

También denominada virtualización de contenedores, este méodo permite al núcleo de un sistema operativo, ejecutar múltiples *userspaces* por sí solo.

Este método no impone *overhead*[1], dado que la interfaz de llamadas al sistema, o cualquier hardware, no necesita ser emulado. Por otro lado, ésto comporta que sólo se pueda virtualizar el mismo sistema operativo que la máquina anfitriona: mientras que cualquier sistema GNU/Linux puede emular otras distribuciones GNU/Linux, Windows y Mac OS X se ven limitados a emularse a ellos mismos.

Hoy en día las opciones más conocidas son **Docker**, y **LXC**.

1.1.4. Arquitectura de oVirt

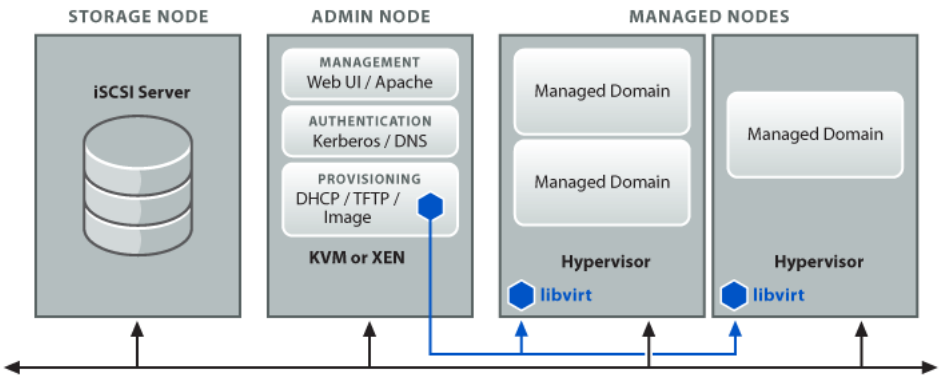


Figura 1.2: Arquitectura de oVirt

oVirt consta de una arquitectura relativamente simple, en la cual encontramos al menos **dos** máquinas:

- Un nodo de control: Esta es la máquina donde se instala el oVirt Engine. Desde aquí realizaremos todas tareas de administración y mantenimiento del *datacenter*: despliegue y migración de máquinas virtuales monitorización de los hosts, configuración del almacenamiento, y más.
- Hosts de virtualización: Aquí es dónde se ejecutarán las máquinas virtuales. Éstos han de disponer de una CPU con las extensiones de virtualización mencionadas en la sección anterior.

La imagen especifica también un tercer nodo: el de almacenamiento. Aquí es donde se guardan tanto las imágenes ISO de instalación de sistemas operativos, como las imágenes de discos duros en uso por las máquinas virtuales. Éste puede ser tanto un host dedicado como en la imagen, o integrarse en el nodo de administración. oVirt soporta diferentes soluciones de almacenamiento en red, como NFS, iSCSI o GlusterFS.

Como se ve en la imagen, el nodo de administración y los hosts de virtualización utilizan la API **libvirt**, para realizar todo el trabajo con máquinas virtuales en materia de creación, arranque y control

[1]Véase el glosario

del almacenamiento. Respecto a la forma de virtualizar, podemos decir que libvirt y KVM virtualizan la máquina virtual por completo, ayudándose de la paravirtualización y las extensiones de virtualización presentes en el procesador.

1.2. Requisitos

1.2.1. Hardware

Los siguientes son las características recomendadas para instalaciones pequeñas y medias, de los diferentes host que forman parte de la arquitectura de oVirt:

- oVirt Engine
  - CPU de dos núcleos.
  - 4 GB de RAM.
  - 25 GB de espacio libre en el disco duro.
  - Interfaz de red de 1 Gbps.
- Hosts de virtualización
  - CPU de dos núcleos con extensiones de virtualización (Intel VT-x / AMD-V)
  - 10 GB de RAM
  - 10 GB de espacio libre en el disco duro.
  - Interfaz de red de 1 Gbps.

Las máquinas utilizadas para los host de virtualización en éste proyecto, por razones estrictamente monetarias, son un par de Acer Aspire X, con las siguientes especificaciones:

- Intel Pentium J29000 (Quad Core, 2,41 GHz)
- 4 GB de RAM
- 500 GB HD
- Interfaz Gigabit Ethernet

Obviamente, éstos quedan muy por debajo de los recomendados incluso para instalaciones pequeñas, por lo que éste proyecto no debe considerarse apto para ser llevado a cabo en producción, al menos en su estado actual.

1.2.2. Software

La siguiente lista muestra los sistemas operativos que oVirt soport a la hora de ser virtualizados:

- Windows (XP, 7, 8, 2003, 2008, 2012)
- RHEL 5, 6 o 7
- CentOS 5, 6 o 7
- Fedora 16 - 23.
- Ubuntu 12.04 +
- openSUSE 12.x+

Capítulo 2

Implementando oVirt

2.1. Instalación del motor

Una vez instalado el sistema operativo en el nodo de control (Fedora, RHEL, o CentOS 7)[1], hace falta actualizar los paquetes del sistema:

Actualizando paquetes

```
yum -y update
```

El siguiente paso para instalar oVirt es subscribirnos a su repositorio. Instalar el siguiente RPM es similar a añadir un PPA en distribuciones basadas en Ubuntu en el hecho de que actualizaciones a éste paquete deberían aparecer al hacer más `yum update` en el futuro.

Instalando oVirt

```
yum install http://plain.resources.ovirt.org/pub/yum-repo/ovirt-release35.rpm
yum -y install ovirt-engine
```

Cuando el paquete se haya instalado, ejecutaremos el instalador del motor, que nos hará una serie de preguntas. El siguiente es un ejemplo del mismo instalador, con las respuestas por defecto envueltas entre corchetes [2].

Instalación del motor

```
  engine-setup
[ INFO  ] Stage: Initializing
   [ INFO  ] Stage: Environment setup
           Configuration files: ['/etc/ovirt-engine-setup.conf.d/10-packaging.
               conf']
           Log file: /var/log/ovirt-engine/setup/ovirt-engine-setup
               -20140310163840.log
           Version: otopi-1.2.0_rc2 (otopi-1.2.0-0.7.rc2.fc19)
   [ INFO  ] Stage: Environment packages setup
   [ INFO  ] Stage: Programs detection
   [ INFO  ] Stage: Environment setup
   [ INFO  ] Stage: Environment customization

           --== PRODUCT OPTIONS ==--
```

[1] Léase anexo para el proceso de instalación de CentOS

```
          --== PACKAGES ==--

[ INFO  ] Checking for product updates...
[ INFO  ] No product updates found

          --== NETWORK CONFIGURATION ==--

          Host fully qualified DNS name of this server [server.name]: example.
              ovirt.org
          Setup can automatically configure the firewall on this system.
          Note: automatic configuration of the firewall may overwrite current
              settings.
          Do you want Setup to configure the firewall? (Yes, No) [Yes]:
[ INFO  ] firewalld will be configured as firewall manager.

          --== DATABASE CONFIGURATION ==--

          Where is the Engine database located? (Local, Remote) [Local]:
          Setup can configure the local postgresql server automatically for
              the engine to run. This may conflict with existing applications.
          Would you like Setup to automatically configure postgresql and
              create Engine database, or prefer to perform that manually? (
              Automatic, Manual) [Automatic]:

          --== OVIRT ENGINE CONFIGURATION ==--

          Application mode (Both, Virt, Gluster) [Both]:
          Default storage type: (NFS, FC, ISCSI, POSIXFS) [NFS]:
          Engine admin password:
          Confirm engine admin password:

          --== PKI CONFIGURATION ==--

          Organization name for certificate [ovirt.org]:

          --== APACHE CONFIGURATION ==--

          Setup can configure apache to use SSL using a certificate issued
              from the internal CA.

          Do you wish Setup to configure that, or prefer to perform that
              manually? (Automatic, Manual) [Automatic]:
          Setup can configure the default page of the web server to present
              the application home page. This may conflict with existing
              applications.
          Do you wish to set the application as the default page of the web
              server? (Yes, No) [Yes]:

          --== SYSTEM CONFIGURATION ==--

          Configure WebSocket Proxy on this machine? (Yes, No) [Yes]:
          Configure an NFS share on this server to be used as an ISO Domain? (
              Yes, No) [Yes]:
          Local ISO domain path [/var/lib/exports/iso-20140310143916]:
          Local ISO domain ACL - note that the default will restrict access to
               example.ovirt.org only, for security reasons [example.ovirt.org
              (rw)]:
```

```
          Local ISO domain name [ISO_DOMAIN]:

          --== MISC CONFIGURATION ==--

          --== END OF CONFIGURATION ==--

       [ INFO  ] Stage: Setup validation

                   --== CONFIGURATION PREVIEW ==--

          Engine database name                    : engine
          Engine database secured connection      : False
          Engine database host                    : localhost
          Engine database user name               : engine
          Engine database host name validation    : False
          Engine database port                    : 5432
          NFS setup                               : True
          PKI organization                        : ovirt.org
          Application mode                        : both
          Firewall manager                        : firewalld
          Update Firewall                         : True
          Configure WebSocket Proxy               : True
          Host FQDN                               : example.ovirt.org
          NFS export ACL                          : 0.0.0.0/0.0.0.0(rw)
          NFS mount point                         : /var/lib/exports/iso
              -20140310143916
          Datacenter storage type                 : nfs
          Configure local Engine database         : True
          Set application as default page         : True
          Configure Apache SSL                    : True
          Please confirm installation settings (OK, Cancel) [OK]:
  [ INFO  ] Stage: Transaction setup
  [ INFO  ] Stopping engine service
  [ INFO  ] Stopping websocket-proxy service
  [ INFO  ] Stage: Misc configuration
  [ INFO  ] Stage: Package installation
  [ INFO  ] Stage: Misc configuration
  [ INFO  ] Creating PostgreSQL 'engine' database
  [ INFO  ] Configuring PostgreSQL
  [ INFO  ] Creating Engine database schema
  [ INFO  ] Creating CA
  [ INFO  ] Configuring WebSocket Proxy
  [ INFO  ] Generating post install configuration file '/etc/ovirt-engine-
      setup.conf.d/20-setup-ovirt-post.conf'
  [ INFO  ] Stage: Transaction commit
  [ INFO  ] Stage: Closing up

          --== SUMMARY ==--
          `         SSH fingerprint: `<SSH_FINGERPRINT> `         Internal CA:
               `<CA_FINGERPRINT>
          Web access is enabled at: `             `[`http://example.ovirt.org
              :80/ovirt-engine`](http://example.ovirt.org:80/ovirt-engine) `
                          `[`https://example.ovirt.org:443/ovirt-engine`](
              https://example.ovirt.org:443/ovirt-engine)
          Please use the user "admin" and password specified in order to login
               into oVirt Engine
```

```
          --== END OF SUMMARY ==--

  [ INFO  ] Starting engine service
  [ INFO  ] Restarting httpd
  [ INFO  ] Restarting nfs services
  [ INFO  ] Generating answer file '/var/lib/ovirt-engine/setup/answers
      /20140310163837-setup.conf'
  [ INFO  ] Stage: Clean up
          Log file is located at /var/log/ovirt-engine/setup/ovirt-engine-
              setup-20140310163604.log
  [ INFO  ] Stage: Pre-termination
  [ INFO  ] Stage: Termination
  [ INFO  ] Execution of setup completed successfully

    **** Installation completed successfully ******
```

Nota

```
Configuraré los \emph{exports} NFS a mano, así que omitiré la parte del
    instalador que configura un dominio ISO.
```

Una vez hecho ésto, ya estaremos listos para acceder a la interfaz web del motor, con la contraseña que le hemos dicho al instalador.

2.2. Instalación de los Host

A la hora de montar los hosts de virtualización, disponemos de varias opciones: podemos instalar el hipervisor propio de oVirt Node, que una vez configurado se añade automáticamente en el motor, o instalar el paquete encima de un CentOS o Fedora ya instalado. Ésta será la opción que documentaré.

Dicho ésto, para instalar el paquete, primero deberíamos asegurarnos que el sistema está actualizado:

Actualizando paquetes

```
yum -y update
```

Y instalamos oVirt:

Instalando paquetes

```
yum localinstall http://plain.resources.ovirt.org/pub/yum-repo/ovirt-
    release36.rpm
```

Cuando ya esté instalado, podemos añadirlo manualmente a oVirt, en la pestaña Hosts. Cuando cliquemos en *New*, aparecerá el siguiente diálogo:

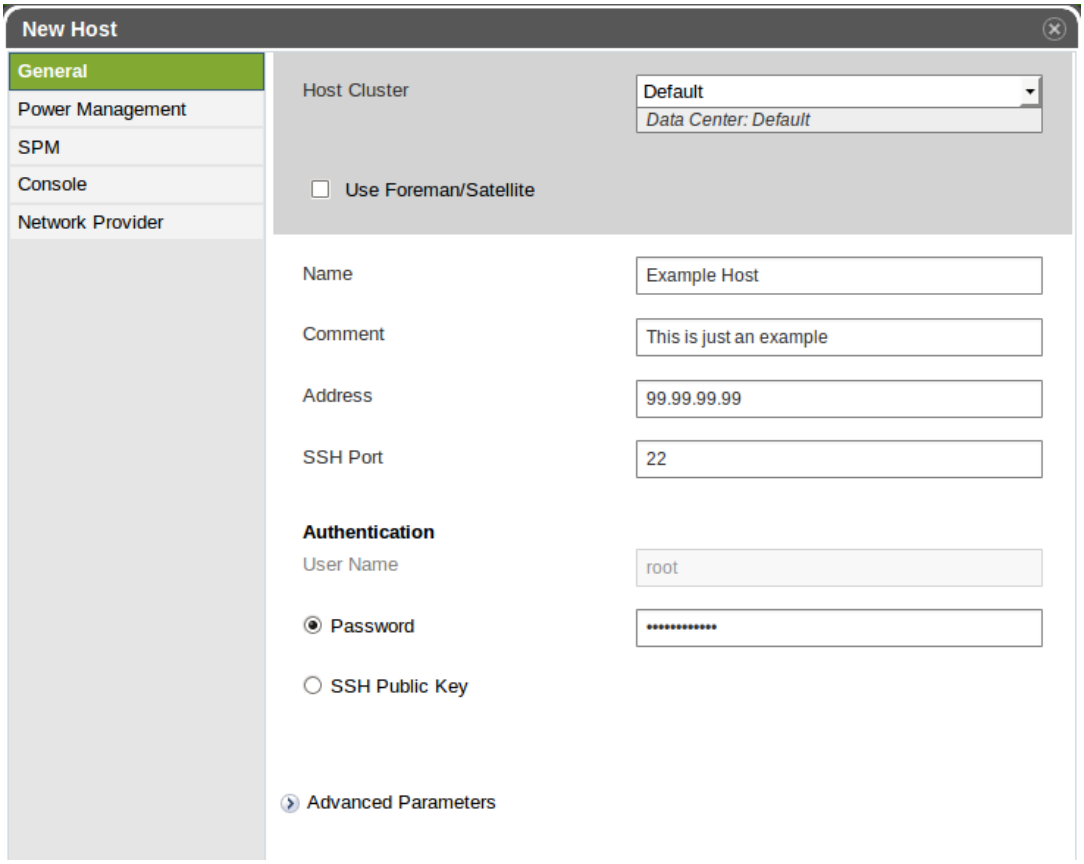


Figura 2.1: Añadiendo el host

Una vez añadido, oVirt descargará e instalará ciertos paquetes. Podemos ver el progreso de ésta tarea en la parte inferior de la pantalla, en las pestañas Events y Tasks.

2.3. Configuración del almacenamiento

Como ya mencioné cuando describía la arquitectura de oVirt, el almacenamiento viene proporcionado por un sistema en red. Las opciones incluyen soluciones como GlusterFS, iSCSI, o FCP, pero nosotros usaremos NFS. Hay que tener en cuenta, que el tipo de almacenamiento que se use debe coincidir con el tipo por defecto que se seleccionó en el instalador.

Entonces, para configurar los *exports* que usaremos en oVirt, instalamos el servidor, y lo configuramos acordemente [2]:

Configurando NFS

```
  yum -y install nfs-utils
  vi /etc/exports
---------------------------
/srv/ovirt/export  *(rw,sync,no_subtree_check,all_squash,anonuid=36,anongid=36)
/srv/ovirt/data    *(rw,sync,no_subtree_check,all_squash,anonuid=36,anongid=36)
/srv/ovirt/iso     *(rw,sync,no_subtree_check,all_squash,anonuid=36,anongid=36)
/usr/ovirt/disks   *(rw,sync,no_subtree_check,all_squash,anonuid=36,anongid=36)
```

Una vez hecho ésto, debemos añadirlos en el siguiente orden en la interfaz de administración web:

1. Export

[2] Ésta configuración sirve sólo de ejemplo: léase la documentación de NFS

2. Data
3. iso

2.3.1. Cargando las ISOs

Cuando ya tenemos el almacenamiento en red configurado, una particularidad de oVirt es que no podemos abocar directamente al directorio /srv/ovirt/iso las imágenes de instalación, sino que debemos usar la herramienta `engine-iso-uploader` para subirlas al datacenter, y puedan ser usadas para posteriormente instalar sistemas operativos en las máquinas virtuales. A continuación muestro un ejemplo de dicho comando:

Subiendo ISOs

```
engine-iso-uploader -i nombre_del_dominio_en_ovirt upload imagen.iso
```

2.3.2. Creando los discos

La creación de los discos puede hacerse tanto al momento de crear la máquina virtual, o cuando el usuario quiera. De todas maneras el proceso es el mismo, detallado en la siguiente captura:

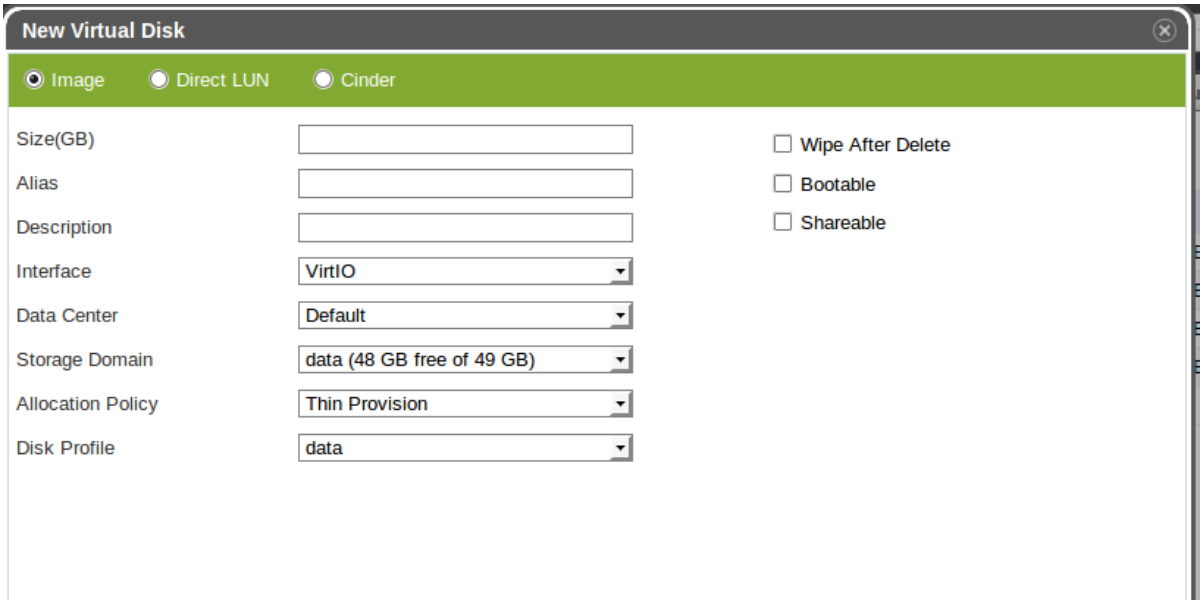


Figura 2.2: Creando discos

2.3.3. Creando interfaces de red

Antes de proceder finalmente a crear una máquina virtual, debemos crear la interfaz de red.

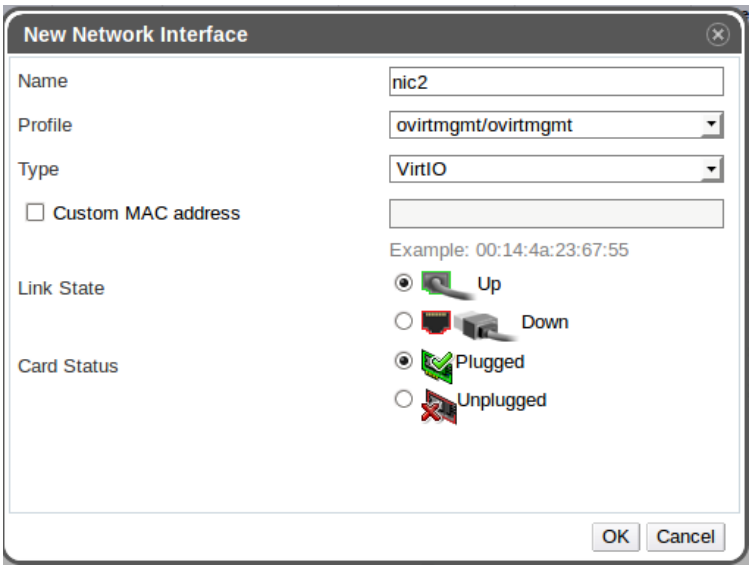


Figura 2.3: Creando interfaces de red

Acorde a la captura, podemos escoger varios tipos de interfaces de red:

- **VirtIO:** La opción más rápida y ligera, si el sistema virtualizado lo soporta
- **Dos modelos de tarjetas físicas reales:** Su uso supone más *overhead*, ya que tienen que virtualizarse las tarjetas por completo.
- **NIC Passthrough:** Otorga de forma exclusiva una interfaz de red de la máquina anfitriona al sistema virtualizado.

2.4. Creando máquinas virtuales

La creación de las máquinas virtuales es bastante simple. Se puede acceder al diálogo de creación de máquinas virtuales mediante el menú Virtual Machines >New. Las siguientes capturas muestran algunas de las opciones de las que disponemos a la hora de crearlas:

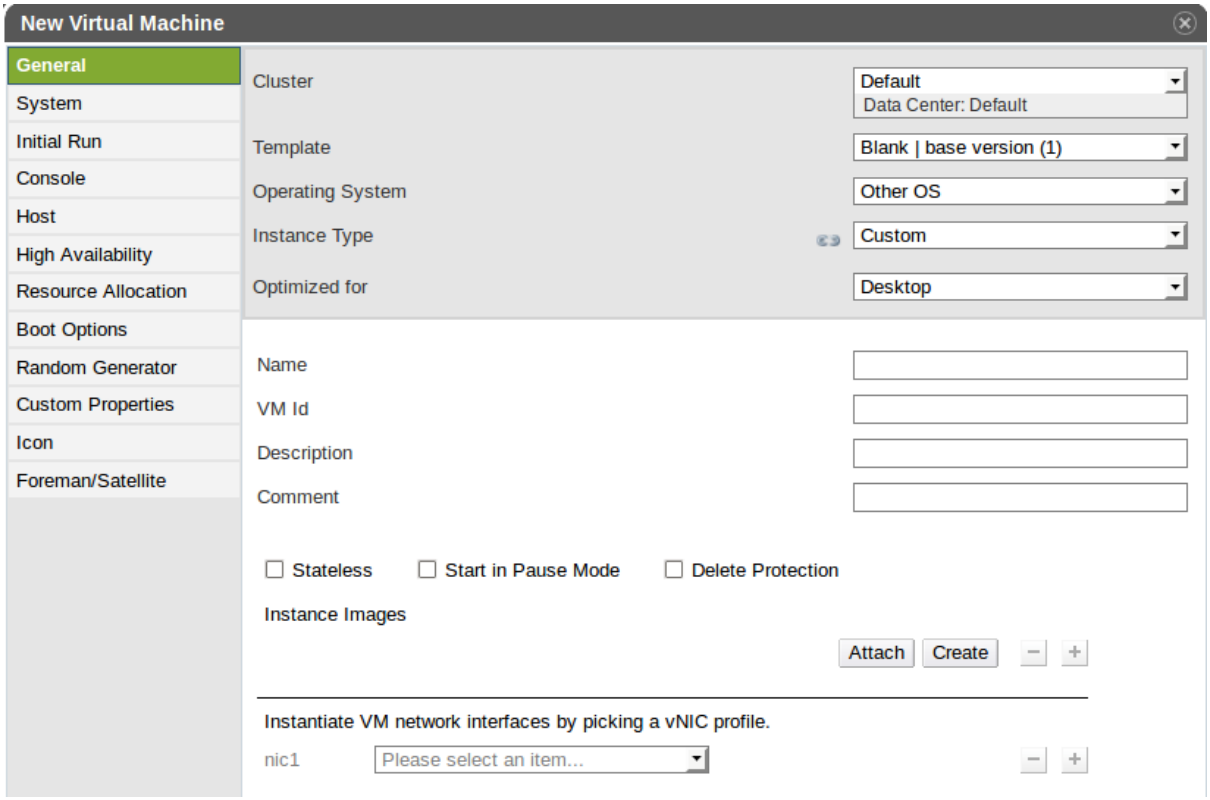


Figura 2.4: Creando máquinas virtuales

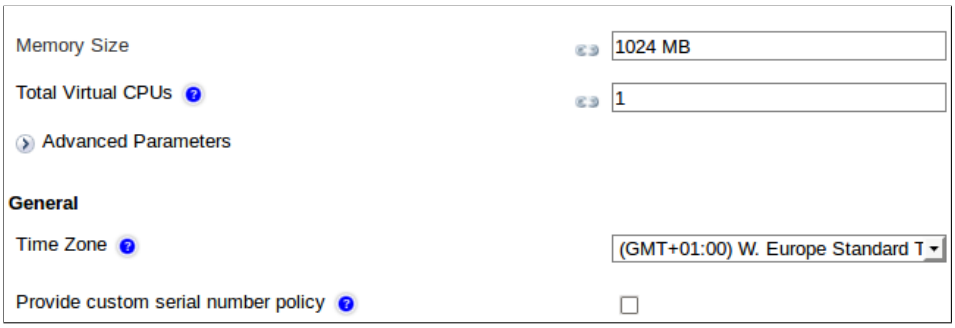


Figura 2.5: Opciones de sistema

Figura 2.6: Opciones de alta disponibilidad

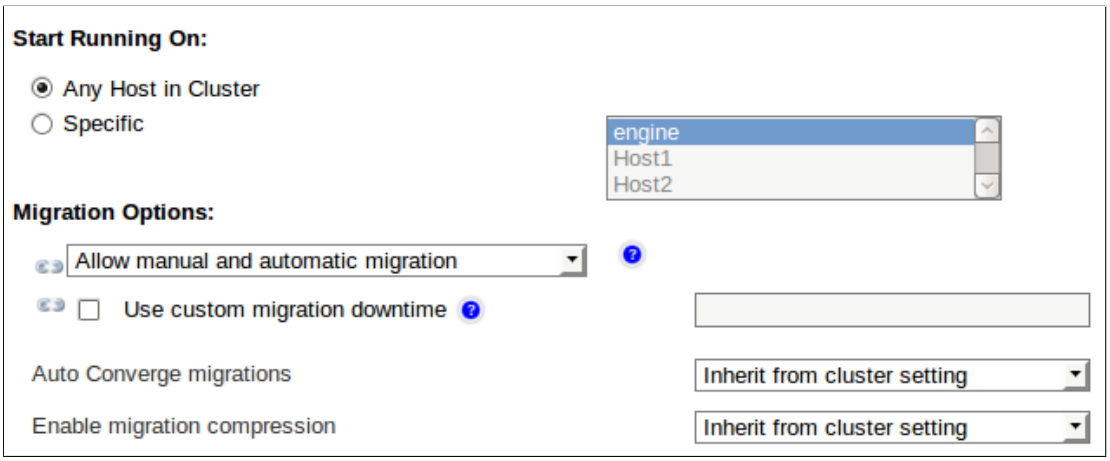


Figura 2.7: Opciones de host

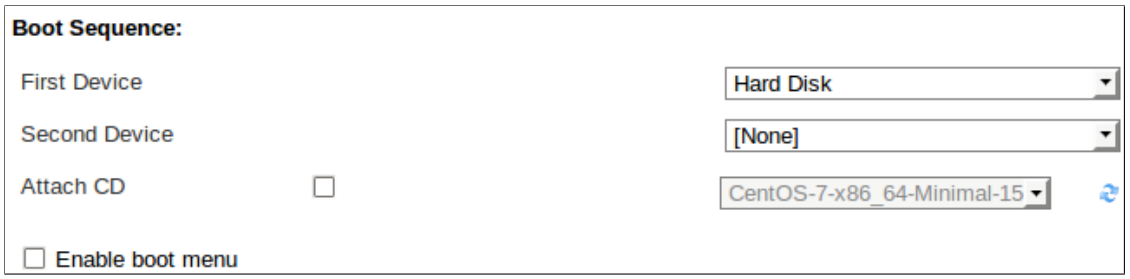


Figura 2.8: Arranque

2.4.1. oVirt Guest Agent

Una vez instaladas las máquinas, debemos instalar el oVirt Guest Agent. Ésta herramienta permite al sistema operativo virtualizado comunicarse con el nodo maestro, ofreciéndole información tal como la dirección IP, el *FQDN* y estadísticas de uso de recursos. El proceso varía dependiendo del sistema operativo, pero a modo de ejemplo, el proceso en Ubuntu es el siguiente:

1. Editar el fichero `/etc/apt/sources.list.d/ovirt-guest-agent.list` añadiendo el siguiente contenido:

   /etc/apt/sources.list.d/ovirt-guest-agent.list

   ```
   deb http://download.opensuse.org/repositories/home:/evilissimo:/
       ubuntu:/14.04/xUbuntu_14.04/ /
   ```

2. Añadir la *Release key*:

   Añadiendo la release key

   ```
   wget http://download.opensuse.org/repositories/home:/evilissimo:/
       ubuntu:/14.04/xUbuntu_14.04//Release.key
   sudo apt-key add - < Release.key
   ```

3. Y finalmente, actualizamos las listas de software e instalamos el paquete:

   Instalando

   ```
   sudo apt-get update && sudo apt-get install ovirt-guest-agent
   ```

Una vez esté instalado, el servicio se autoejecutará en cada arranque de la máquina virtual. Si ahora vamos a la interfaz web, deberíamos ver la información de la siguiente manera:

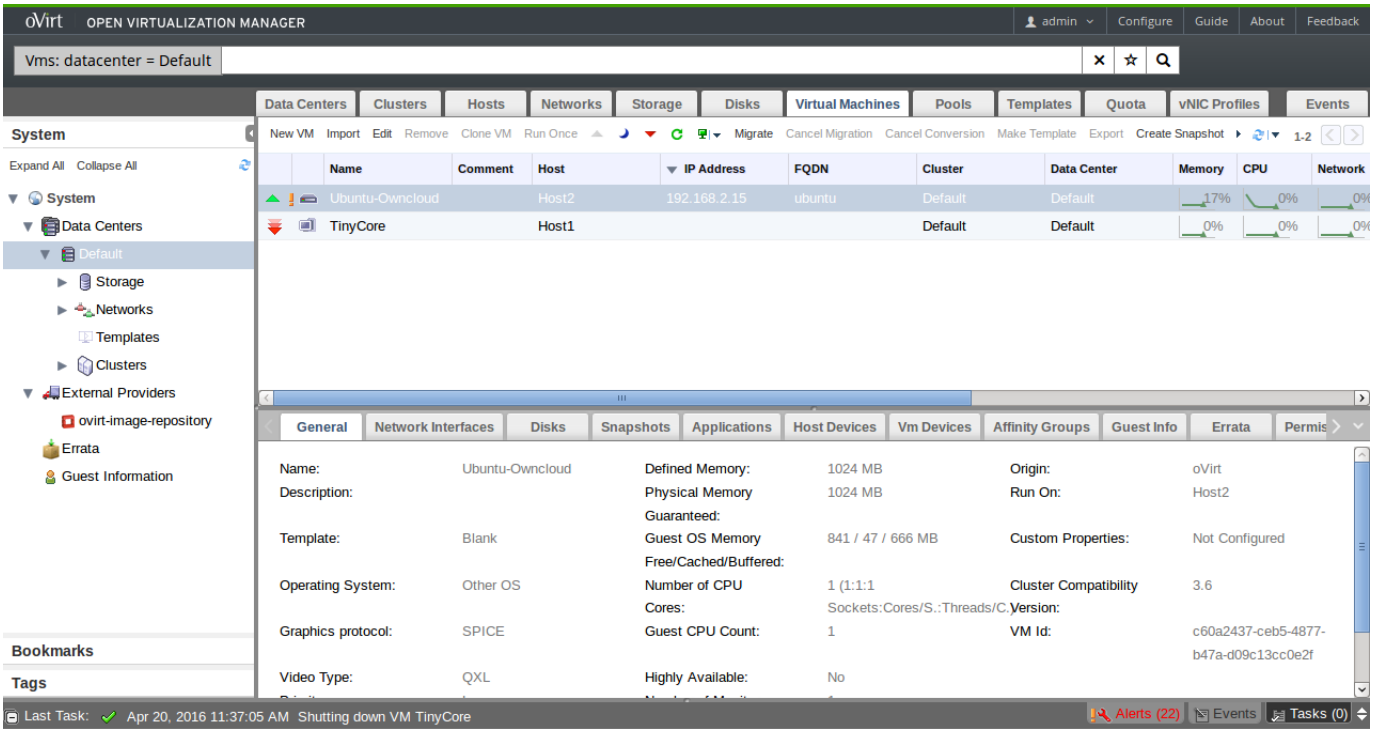


Figura 2.9: Lista de máquinas virtuales

Capítulo 3

Trabajando con VMs en oVirt

3.1. Barra de herramientas

Si volvemos a la lista de las máquinas, veremos una lista de herramientas en la parte superior. La siguiente captura los muestra numerados, con una lista explicando su función:

1. **New VM:** Crea una nueva máquina virtual. Éste botón abre el diálogo que vimos en la sección anterior.
2. **Import:** Importa una máquina virtual.
3. **Edit:** Edita las propiedades de la máquina.
4. **Remove:** Borra la máquina virtual. Por razones obvias, ésto sólo se puede hacer si la máquina está apagada.
5. **Clone VM:** Clona la máquina virtual.
6. **Run Once:** Ejecuta una única vez.
7. **Flecha verde:** Ejecuta una única vez.
8. **Luna azul:** Suspende la ejecución de la máquina.
9. **Flecha roja:** Envía señal de apagado a la máquina virtual.
10. **Reiniciar:** Envía señal de reinicio a la máquina virtual.
11. **Consola:** Accede a la consola.
12. **Migrate:** Migra la máquina virtual.
13. **Cancel Migration:** Cancela la migración en curso.
14. **Cancel Conversion:** Cancela la conversion en curso.
15. **Make Template:** Crea una plantilla a partir del estado actual de la máquina virtual.
16. **Export:** Exporta la máquina virtual.
17. **Create Snapshot:** Crea una instantánea de la máquina virtual.
18. **Refresh:** Refresca la lista de máquinas virtuales de forma manual.
19. Cambia la tasa de refresco de la lista de máquinas virtuales.

3.2. Accediendo a consola

3.3. Exportando o importando máquinas virtuales

3.3.1. Exportando máquinas

Para exportar antes una máquina virtual, deberemos importar, tal y como vimos en el apartado de instalación de oVirt, otro dominio de almacenamiento más, en este caso con función **export**. Cuando esté importado y activo, podemos proceder a exportar la máquina virtual:

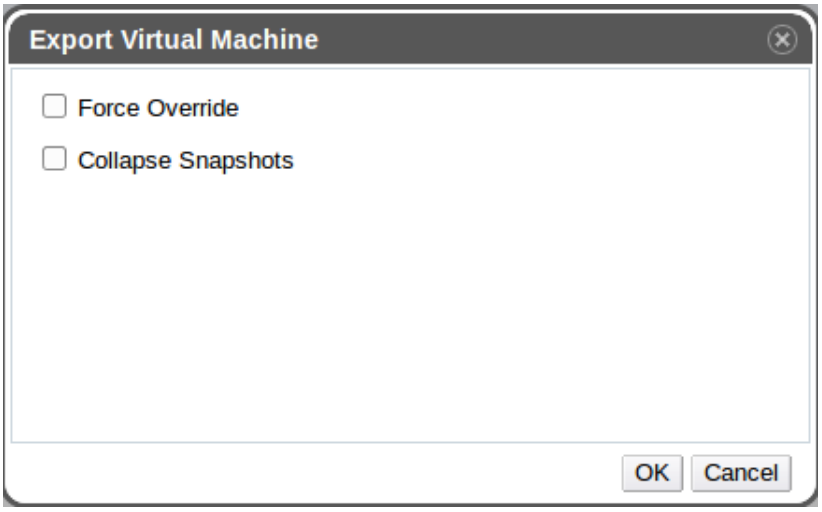


Figura 3.1: Exportando la máquina virtual.

Ésto servirá para poder mover una máquina virtual entre *datacenters*. Es importante remarcar que un dominio export sólo puede estar activo en un clúster a la vez.

Cuando hayamos exportado la máquina a un dominio de almacenamiento, podemos comprimirlo y pasar el archivo resultantes para que otros usuarios de oVirt puedan importar el .ovf resultante a sus clústers.

3.3.2. Importando máquinas

Para importar máquinas virtuales, tenemos diferentes opciones:

- VMware
- **Dominio de export:** Carga las máquinas de un dominio de almacenamiento, especificado durante el proceso de exportado.
- **Virtual Appliance (fichero OVA):** Carga las máquinas virtuales desde un fichero OVA. En ésta opción sólo necesitaremos especificar la ruta absouta al fichero OVA, y el host del clúster desde el cual se va a cargar el fichero.

> **¡Cuidado!**
>
> La función de importado de ficheros Open Virtualization Format sólo funciona con OVFs generados por oVirt y comprimidos con gzip.

La primera captura muestra la pantalla inicial de importación mediante dominio, en la cual vemos cómo se debe cargar primero el dominio, tras lo cual aparecerá la lista de máquinas virtuales del dominio, pudiendo importar una, o más máquinas.

La segunda muestra una sección con pestañas similar al *overview* de las máquinas virtuales, que nos permite cambiar ciertos aspectos de la configuración de la máquina, y definir el clúster y perfil de CPU al cual se va a importar la máquina.

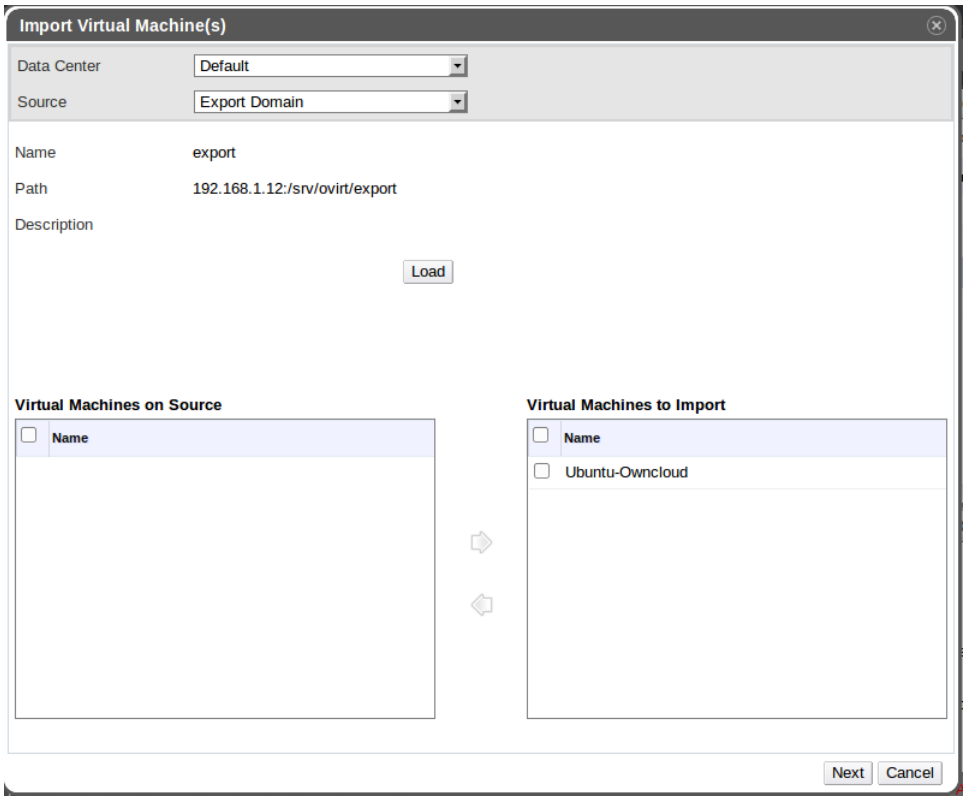


Figura 3.2: Iniciando la importación

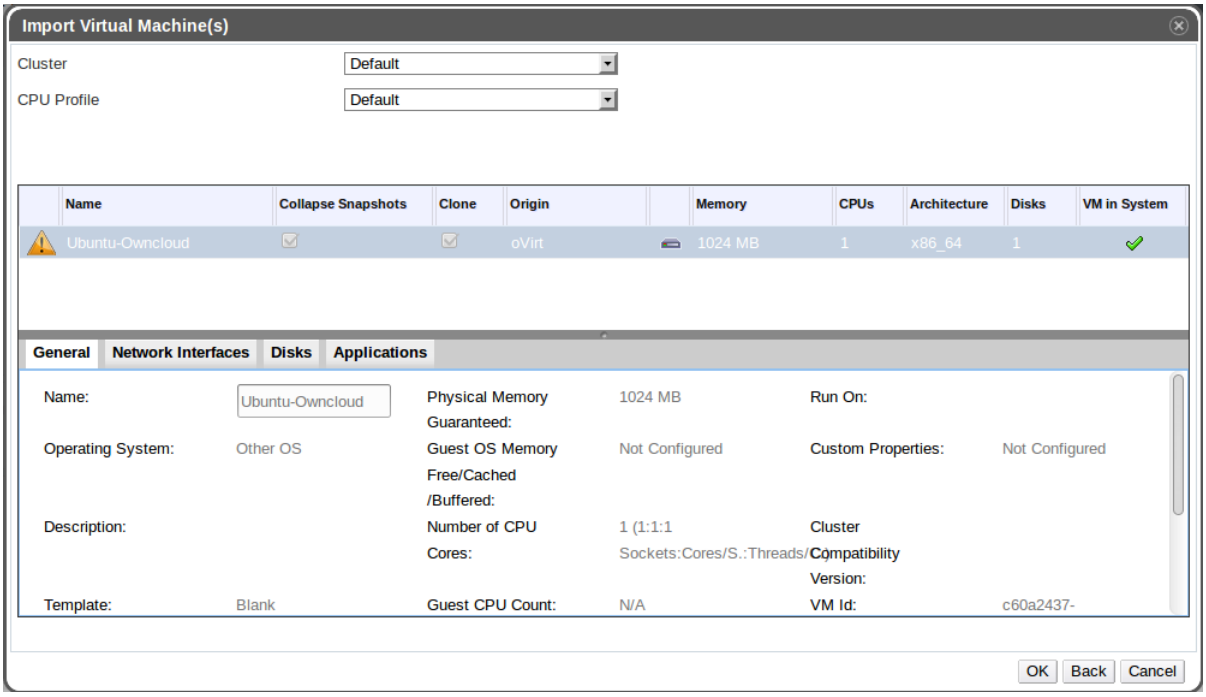


Figura 3.3: Siguiendo la importación

3.4. Clonando máquinas virtuales

La clonación de máquinas virtuales en oVirt es probablemente la operación más sencilla a realizar. Simplemente debemos darle un nombre nuevo, y se generará una copia exacta de la máquina, configuración incluida.

3.5. Migrando máquinas virtuales

La migración de las máquinas virtuales permite mantener una máquina virtual en ejecución, mientras la carga de trabajo se traslada a otro host. Aunque en papel suena muy útil, en mis pruebas ha sido bastante *hit or miss*(acierto o error). Una vez apretado el botón para iniciar la migración, aparecerá la siguiente pantalla:

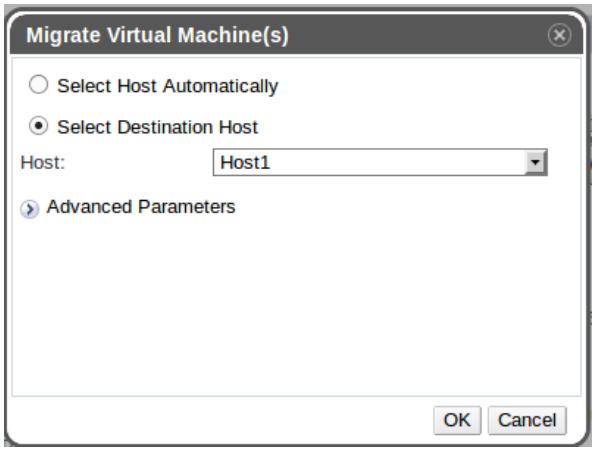


Figura 3.4: Iniciando la migración

Una vez seleccionado el host al cual queremos migrar, simplemente habrá que esperar. Si la migración ha funcionado, veremos como el campo Host del listado de máquinas virtuales habrá cambiado. Podemos ver cómo progresa la tarea en la parte de abajo de la web, en la sección Tasks:

Figura 3.5: Migración exitosa

3.6. Snapshots

Una instantánea o *snapshot* es el estado de una máquina virtual y sus discos en cualquier punto en el tiempo. Esto permite volver a un estado anterior si ocurre cualquier error que deje la máquina virtual inservible.

Para realizar un Snapshot, basta con darle a **Create Snapshot** en la barra de herramientas:

> **Nota**
>
> Si se usa almacenamiento NFS, tanto los discos virtuales, como los snapshots, son ficheros. Ésto quiere decir que las copias de seguridad pueden realizarse tranquilamente mediante scripts que copien los ficheros relevantes, y automatizarlos mediante *crontab*

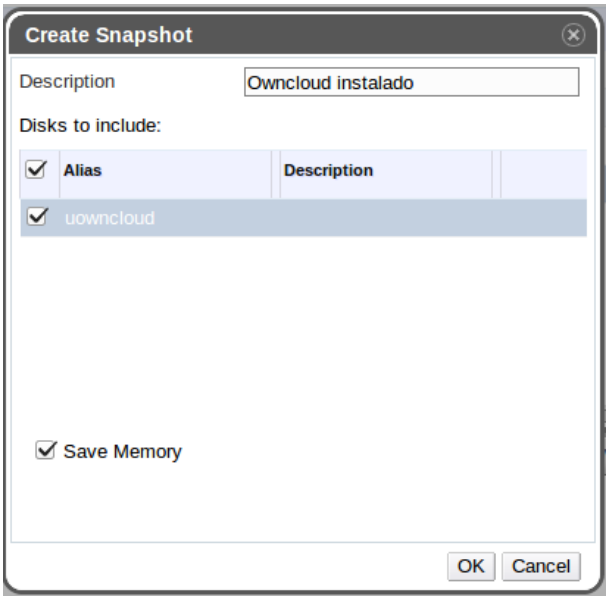


Figura 3.6: Creación de snapshots

Capítulo 4

Conclusiones

4.1. Objetivos cumplidos

Durante el transcurso del trabajo, puedo considerar los siguientes objetivos como completados:

- Se ha estudiado el uso de la virtualización como herramienta para asegurar servicios.
- Se ha estudiado el uso de oVirt como plataforma de gestión del *datacenter*
- Assegurar els servidors de l'IES La Bastida a través de la virtualització
- Se ha probado la migración de máquinas virtuales, aunque con resultados variados.
- Estudiados los mecanismos de copias de seguridad en oVirt.

Considerando como el único objetivo que no ha sido completado satisfactoriamente de forma regular es la migración de máquinas virtuales, considero el proyecto un éxito, si menos tomándolo como una prueba de concepto de lo que se puede hacer con oVirt.

Resulta una lástima que hayan cosas que no se hayan podido completar satisfactoriamente, o averiguar por qué fallaban, como en el caso de la migración de máquinas virtuales, pero desde bastante pronto en el proyecto decidí que éste iba a ser más una prueba de concepto que un proyecto a ser implementado tal y como saliera de éste crédito.

4.2. Análisis de la planificación y metodología

Acercándonos al final del proyecto, considero que la metodología que se decidió seguir este año, realmente no difiere en mucho a como se había hecho en años anteriores. El seguimiento del proyecto fue casi inexistente, cosa que a mí personalmente me llevó a una dinámica que no dista demasiado de cualquier otro proyecto. Por supuesto que adherise rigurosamente al plan de trabajo me correspondía a mí, pero se hubiera agradecido una mayor supervisión, sobretodo en materia de contenido del documento.

4.3. Planes de futuro para el proyecto

Una vez concluido el curso, continuaré trabajando en el proyecto, y experimentando con oVirt y otras herramientas, pues el objetivo era encontrar la manera de implementar un sistema alternativo en el IES La Bastida.

Uno de los puntos que más ganas tengo de investigar es el uso de ZFS, recientemente incorporado a Ubuntu 16.04, como sistema de ficheros detrás del almacenamiento en red. En la comparativa de ZFS, btrfs, XFS, ext4 y LVM/KVM realizada por Gionatan Danti [3], ZFS sale bastante bien parada de la comparativa con los otros sistemas de ficheros, y ofrece también una serie de características adicionales, como la caché ARC, soporte nativo de *snapshots*, *striping* dinámico, deduplicación de datos, y más.

Existe también otro punto importante que no he podido tratar en el proyecto por falta de tiempo, el uso de soluciones basadas en **contenedores**, como Proxmox VE.

Capítulo 5

Anexo

5.1. Instalando CentOS 7

La instalación de CentOS se basa en el instalador Anaconda, que ya viene siendo usado en Red Hat Enterprise Linux y Fedora. Difiere de otros instaladores, en que en lugar de ser lineal, presenta al usuario una *landing screen*, en la cual se encuentran las diferentes preguntas que típicamente se realizan durante una instalación: El particionado de los discos, idioma, distribución de teclado, paquetes a instalar... La siguiente captura muestra dicha pantalla:

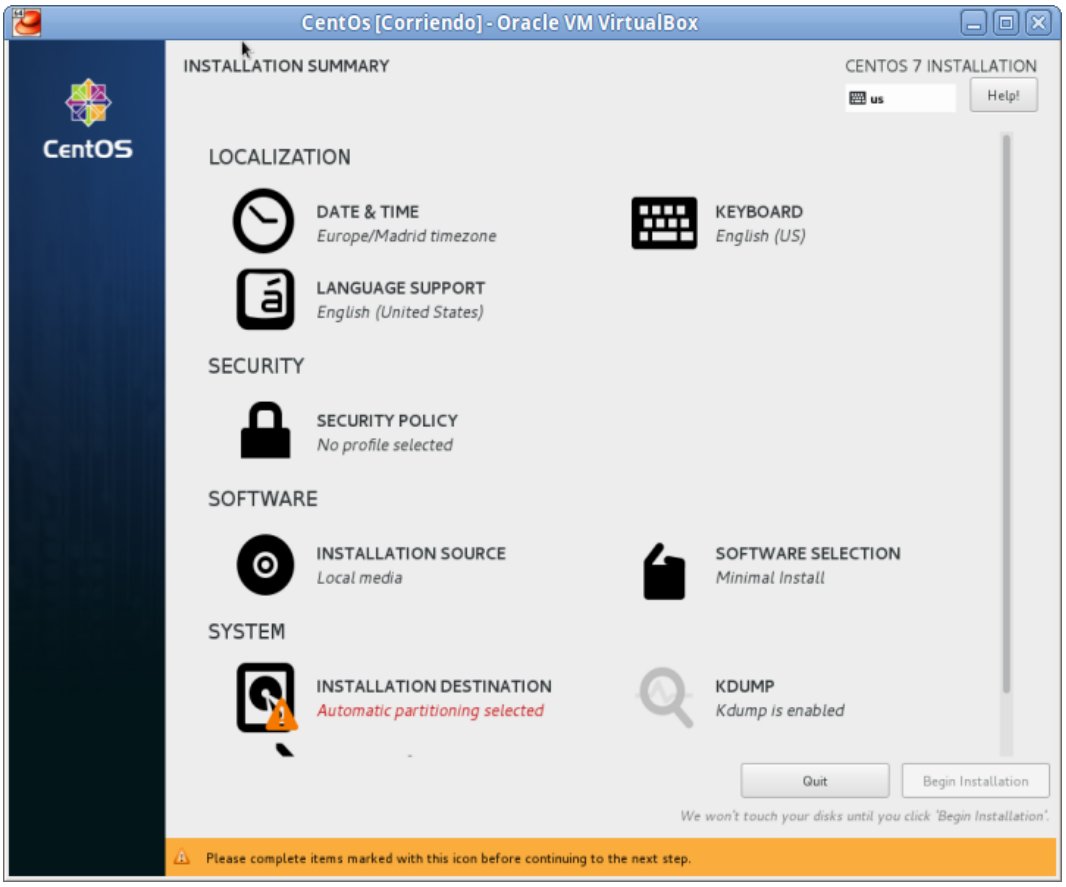


Figura 5.1: Pantalla principal del instalador Anaconda en CentOS

Algunos de los apartados son bastante sencillos y se explican ellos mimsos, así que no incluiré capturas de los mismos.

5.1.1. Localización

5.1.1.1. Fecha y hora

Éste apartado es muy similar al de otros instaladores. Un mapa del planeta aparecerá, dónde podremos seleccionar nuestra ciudad. Acorde a la selección, se ajustará la hora del sistema. Es posible también desactivar NTS, y introducir tanto la fecha como la hora de forma manual.

5.1.1.2. Teclado

Bastante simple también. Aparecerán dos cuadros, uno a la izquierda y otro a la derecha. En la parte inferior del panel izquierdo habrá una lista de botones, que corresponden a Añadir, Quitar, Mover arriba y Mover abajo. Cuando seleccionemos Añadir, aparecerá una lista de idiomas, y las distribuciones de cada idioma una vez se haya seleccionado uno.

En el derecho, simplemente, se podrá probar la distribución de teclado seleccionada.

5.1.1.3. Soporte de idiomas

La pantalla de soporte de idiomas es casi exactamente la misma que la primera que encontramos nada más arrancar el instalador, con la única diferencia que en lugar de escoger un único idioma, podemos ir marcando varios, para que luego se instale soporte para todos aquellos idiomas marcados.

5.1.2. Discos

Éste es el único apartado *obligatorio* para completar la instalación; todos los demás apartados pueden dejarse con los valores por defecto. Una vez accedemos al apartado, se nos presenta la siguiente pantalla:

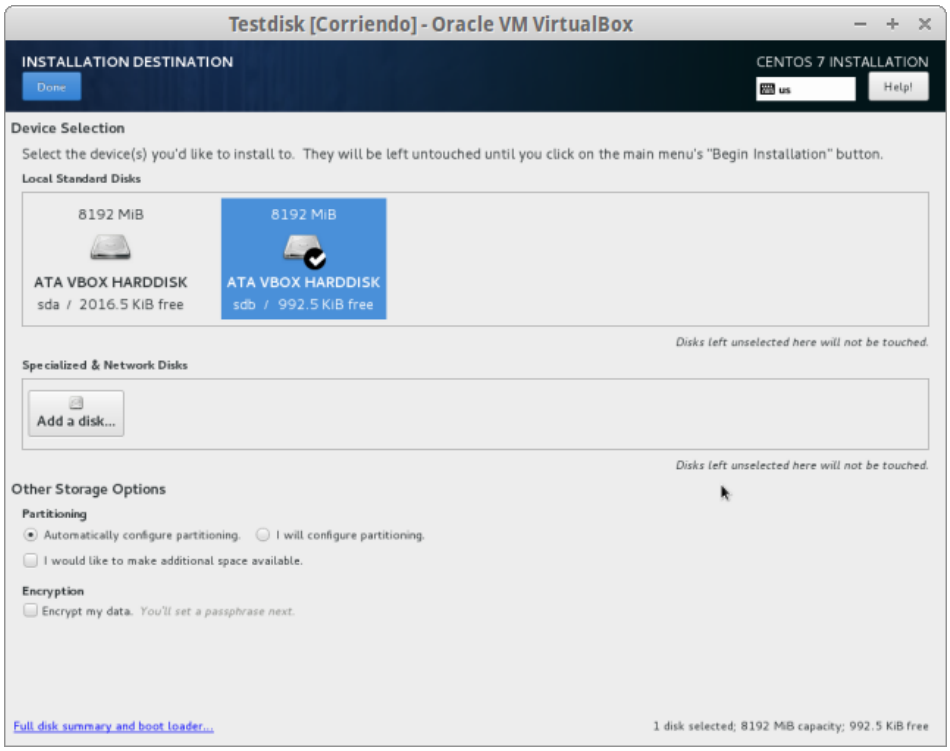


Figura 5.2: Selección de discos

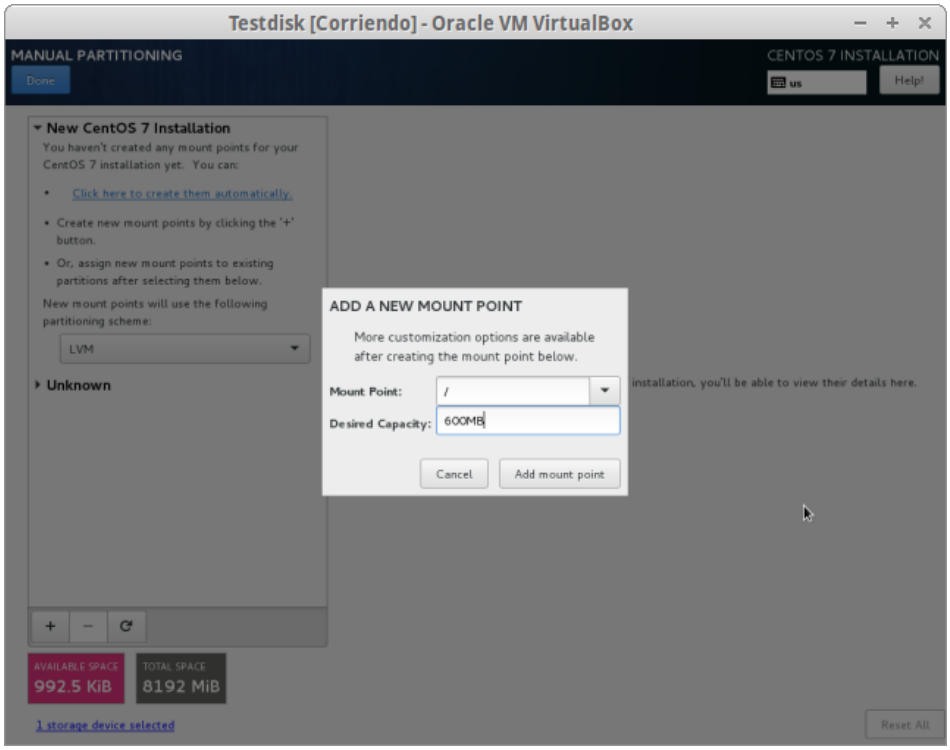


Figura 5.3: Añadiendo punto de montaje

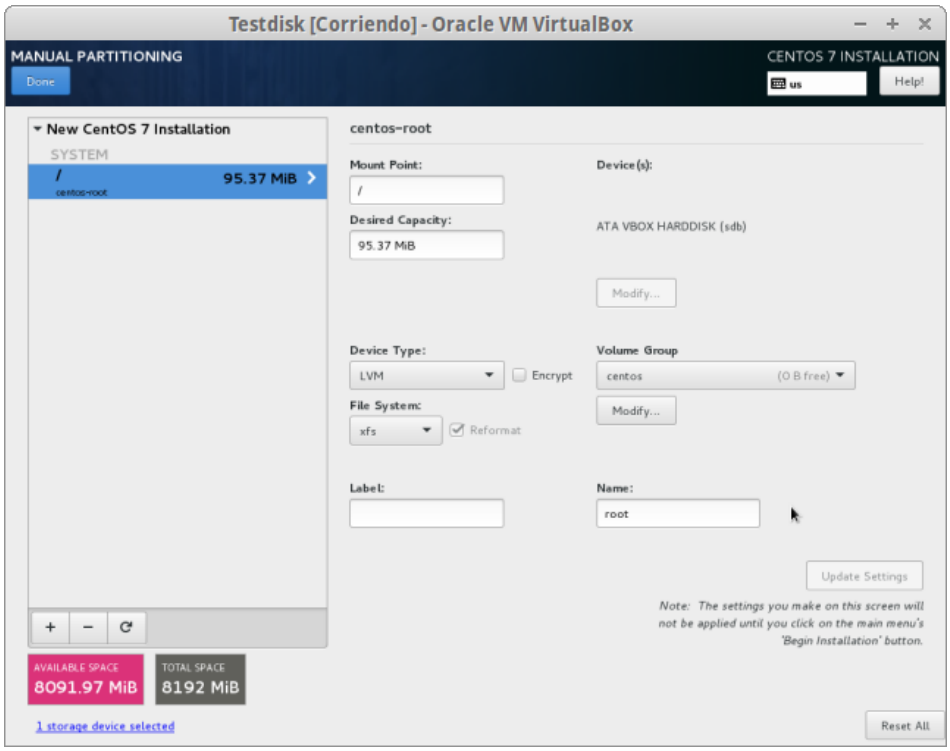


Figura 5.4: Editando punto de montaje

5.1.3. Continuando la instalación

Una vez realizado el particionamiento de los discos, la instalación se iniciará, y deberemos editar la contraseña del usuario root, y crear un usuario adicional.

5.2. Trabajando remotamente

Dadas las circunstancias del proyecto (No dispongo del hardware físico en el Puig), y que ahora sólo voy aproximadamente ocho horas a la semana a la Bastida, he tenido que buscarme la vida para poder trabajar con las máquinas virtuales en el Puig.

La solución que detallaré ahora a continuación no es muy complicada, pero todo gira en torno a que La Bastida tenga conexión a Internet. Si no puedo llegar hasta su proxy, no puedo trabajar. Una vez dicho esto, lo primero que tengo que hacer para poder acceder a la máquina de la red local de la Bastida donde está instalada la interfaz web del motor, es convertir a mi portátil en un proxy SOCKS.Ésto lo puedo hacer mediante el siguiente túnel SSH:

Creando el túnel

```
ssh -D 10080 enrique@ieslabastida.xtec.cat -p 2209
```

Mientras la sesión SSH siga activa, mi portátil estará actuando como un proxy SOCKS.Si configuro Firefox con dicho proxy como en la siguiente captura:

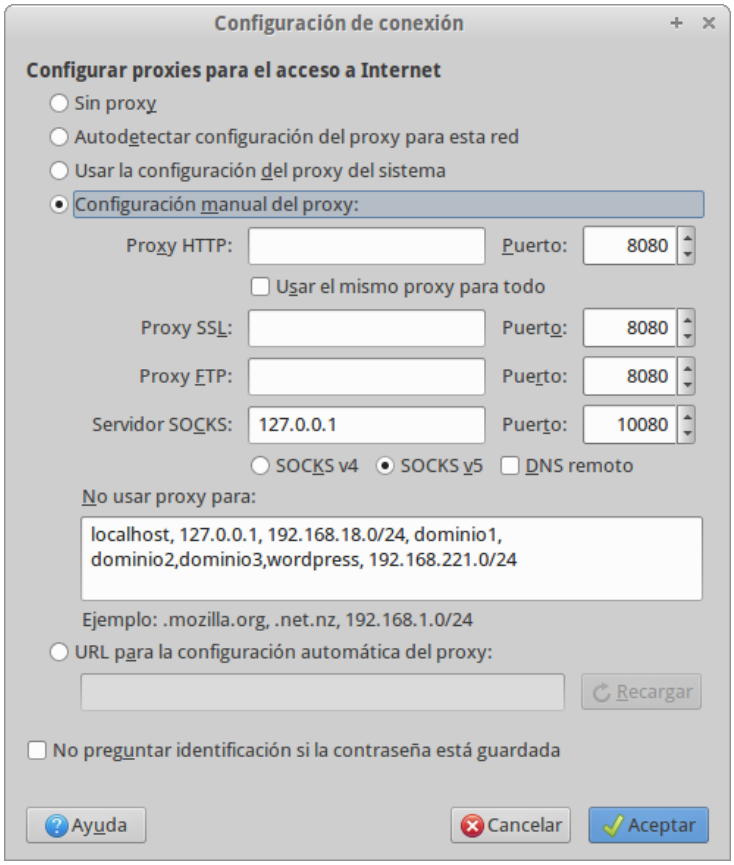


Figura 5.5: Configuración del proxy en Firefox

Todo el tráfico pasará por el puerto 10080 de mi portátil, que no es ni más ni menos un túnel SSH al Proxy de la Bastida, que atenderá la petición, y me permitirá conectarme al portal de administración.

5.2.1. Consola remota mediante el proxy SOCKS

oVirt permite utilizar un cliente de consola remota para conectarse a las máquinas virtuales, y trabajar con el escritorio remoto. Ésto se hace mediante ficheros con extensión .vv que permiten a un programa como remote-viewer conectarse. El único problema, es que éste programa no incluye configuración de red, ni la distribución que utilizo (Xubuntu 15.10) incluye configuración global del proxy. La solución a éste problema radica entonces en la librería **tsocks**, que intercepta todo el tráfico saliente, y lo re-envía a un servidor SOCKS.Está pensado para aplicaciones que por defecto no tratan bien con servidores proxy SOCKS.

La siguiente figura muestra la breve configuración (sólo dos líneas) necesarias para hacer que tsocks mande el tráfico por el mismo proxy SOCKS que utilizamos antes:

Configurando tsocks

```
server      = 127.0.0.1
server_port = 10080
```

Una vez editado el fichero de configuración /etc/tsocks, ya podemos ejecutar cualquier programa tal que `tsocks` *programa*, y todo el tráfico de red que éste genere irá a pasar al servidor SOCKS.

Glosario

API *Application Programming Interface*, siglas en inglés de interfaz de programación de aplicaciones, se refiere a un conjunto de rutinas, funciones, y demás elementos para programar aplicaciones..

deduplicación de datos es una técnica de compresión de datos que consiste en eliminar copias duplicadas de datos repetidos en el soporte de almacenamiento, reemplazándolas por una referencia a la copia original..

hipervisor es el programa encargado de crear, gestionar y ejecutar máquinas virtuales..

overhead es cualquier combinación de tiempo indirecto de computación, memoria, ancho de banda, o cualquier recurso necesario para completar un objetivo determinado..

SPOF *Single Point of Failure*, siglas en inglés de punto único de fallo, se refiere a cualquier elemento de un sistema informático, que en caso de fallo, deja inoperativo a la totalidad del sistema..

Bibliografía

[1] VMware. *Understanding Full Virtualization, Paravirtualization and Hardware Assist*. http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf. [Online; visitado el 29 de mayo de 2016]. 2007.

[2] Red Hat. *oVirt Quick Start Guide*. http://www.ovirt.org/Quick_Start_Guide. [Online; visitado el 15 de enero de 2016]. 2015.

[3] Gionatan Danti. *ZFS, BTRFS, XFS, EXT4 and LVM with KVM - a storage performance comparison*. http://www.ilsistemista.net/index.php/virtualization/47-zfs-btrfs-xfs-ext4-and-lvm-with-kvm-a-storage-performance-comparison.html?limitstart=0. [Online; visitado el 20 de mayo de 2016]. 2015.