Virtualización de Servidores
KVM: Kernel-based Virtual Machine

Proyecto de Innovación

Implantación y puesta a punto de la infraestructura de un cloud computing privado para el despliegue de servicios en la nube

IES Gonzalo Nazareno
Dos Hermanas (Sevilla)

IES Los Albares
Cieza (Murcia)

IES La Campiña
Arahal (Sevilla)

IES Ingeniero de la Cierva
Murcia

Cofinanciado por:

Unión Europea
Fondo Social Europeo
"El FSE invierte en tu fut

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE

CC BY SA

¿Qué vamos a ver? (1)

Conceptos Básicos

Usando KVM

- Instalación
- Configuración Inicial
- Gestión de Máquinas Virtuales
- Almacenamiento: pools y volúmenes
- Formatos de imágenes
- Virtual Networking
- KSM
- Herramientas

Virtualización de Servidores: KVM

¿Por dónde vamos?

Conceptos Básicos

Contextualización (1)

Virtualización de Servidores

- La virtualización de servidores se ha convertido en muy poco tiempo en una tecnología confiable y una solución muy extendida en servidores x86.
- KVM, Kernel-based Virtual Machine, es uno de los últimos hipervisores en hacer su entrada en el mercado, ya abarrotado, de la virtualización.
- Como otros hipervidores tipo 1, se integra en un sistema operativo existente, principalmente Linux, aunque también se ha portado a FreeBSD y Illumos (antes OpenSolaris).
- Originariamente solo en arquitecturas x86, portado actualmente a S/390, PowerPC, IA-64 y en progreso a ARM.
- KVM presenta ciertas peculiaridades únicas en su arquitectura.

KVM (1)

¿Qué es KVM?

KVM

KVM (Kernel-based Virtual Machine) es un hipervisor tipo 1 que proporciona una solución completa de virtualización para el SO Linux en arquitecturas x86 que cuenten con las extensiones de virtualización Intel VT/AMD-V.

Cuando se carga el módulo KVM, Linux se convierte en un hipervisor "bare-metal" (tipo 1) capaz de ejecutar varias máquinas virtuales (VM) aisladas. KVM hospeda las VMs como procesos, por lo que cada VM puede beneficiarse de todas las características del kernel de Linux, incluyendo todas aquellas referentes al hardware, seguridad, almacenamiento, etc.

Origen de KVM (1)

Un poco de historia

- KVM se inició como un proyecto Open Source por la empresa israelí Qumranet.
  - Curiosamente se inició como un proyecto centrado en una solución VDI para clientes Windows.
- Qumranet fue adquirida en 2008 por Red Hat.
- Red Hat ha centrado su estrategia en KVM para definirlo como la mejor solución de virtualización en el mundo Open Source.
  - En RHEL 5 (Red Hat Enterprise Linux), la solución de virtualización escogida era Xen, en RHEL 6 se abandonó por KVM.
- KVM se implementa como un módulo en el kernel de Linux. Se integró en la rama principal del kernel de Linux en la versión 2.6.20, en el año 2007. Actualmente es el hipervisor de virtualización oficial del kernel de Linux.

Origen de KVM (2)

Un poco de historia

- IBM también es una de las empresas que apuesta fuerte por KVM.
  - Ha contribuido en áreas como la gestión de memoria, mejoras en el rendimiento y el subsistema de E/S virtual. IBM tiene a un gran número de desarrolladores trabajando sobre KVM y lo ofrece como solución en su portfolio de software para servidores.
- Otras distribuciones de Linux también comenzaron a invertir en KVM e incluirlo en sus distribuciones como el caso de SUSE ó Ubuntu.

Linux y KVM (1)

Soporte oficial

KVM se incluyó oficialmente en la rama principal del kernel en la versión 2.6.20 (lanzada en febrero de 2007). Implicaciones:

- KVM forma ahora parte integral del kernel de Linux, por lo que aparece en todas las distribuciones ya que todas llevan un kernel más actual que el 2.6.20. Que forme parte y tenga capacidades, no implica que haya soporte comercial ni que se incluyan las herramientas necesarias.
- KVM es capaz de aprovecharse de la comunidad en torno a Linux, cualquier mejora sobre el kernel, es una mejora que beneficia a KVM. Cualquier desarrollador de Linux puede también beneficiarse del uso de KVM.

Linux y KVM (2)

Soporte oficial

- El número de versión de KVM depende de la versión del kernel de Linux. El kernel 2.6.35 será la revisión número 15 de KVM. Vendedores comerciales sí pueden incluir sus propios números de versión en KVM.
- KVM hereda todos los drivers y el amplio soporte hardware de Linux, permitiendo poder ejecutarse en cualquier plataforma x86 donde Linux lo haga. Actualmente se están realizando ports de KVM a otras arquitecturas.

KVM (1)

Arquitectura

- KVM es un módulo cargable del kernel que permite al SO Linux actuar un hipervisor "bare metal" tipo 1.
- Un hipervisor puede verse como un SO especializado en la ejecución de máquinas virtuales.
  - Pero un hipervisor tipo 1 también tiene que tratar con otras tareas más estándares como gestión de memoria, planificación de procesos, drivers de dispositivos, E/S.
- El enfoque de KVM es:
  - Implementar dentro del módulo todo lo referente a la gestión de VM.
  - No reinventar la rueda y permitir que *todo lo demás* lo realice un SO probado y eficaz como Linux.

KVM (1)

Venntajas del enfoque módulo+SO

- Los desarrolladores de KVM centran su atención optimizar la ejecución de procesos que *representan* máquinas virtuales.
- No se replican esfuerzos entre los desarrolladores de KVM y los del kernel.
- Todos los avances dentro de kernel de Linux como SO, se aplican simultáneamente en KVM:
  - **Planificación, control de recursos y gestión de memoria.**
    - Bajo KVM en Linux, las VM son simplemente procesos, cualquier mejora de Linux en la gestión de procesos se aplica directamente en la gestión de VM.
  - **Almacenamiento.**
    - Las imágenes de disco de las VM se tratan como cualquier otro fichero o dispositivo de Linux.

KVM (2)

Venntajas del enfoque módulo+SO

- Se puede utilizar para las imágenes cualquier tipo de almacenamiento soportado actualmente por Linux: discos locales, una gran variedad de sistemas de ficheros, sistemas NAS, iSCSI, SAN, etc.
- Cualquier mejora en la pila de almacenamiento será aprovechada por KVM.

- **Soporte de hardware.**
  - KVM hereda todo el ecosistema de dispositivos Linux. KVM podrá acceder a cualquier dispositivo soportado por Linux.
  - QEMU se utiliza para proporcionar los dispostivos de E/S que las máquinas virtuales "ven".
  - Las mejoras de Linux en cuanto a número de CPUs/cores y grandes cantidades de RAM, permiten a KVM escalar tal como Linux escala.
- **Seguridad.**

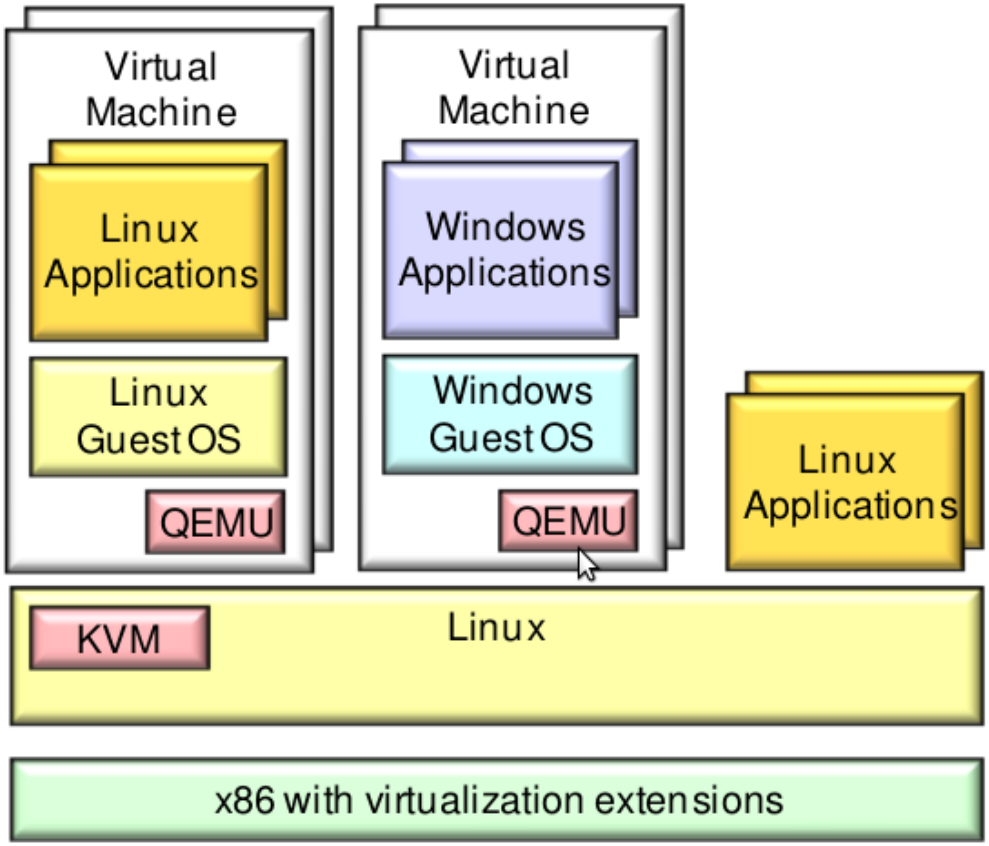
KVM (3)

Venntajas del enfoque módulo+SO

- KVM también puede aprovecharse del modelo de seguridad de Linux proporcionado tanto por SELinux como por AppArmor. Ambos proporcionan, básicamente, "sandboxes" donde encerrar los procesos. Si un proceso se ve comprometido, esto no afectará al resto del sistema. Este concepto se aplica de la misma forma a las VM, ya que una VM es también un proceso.
- La seguridad proporcionada por SELinux y AppArmor aisla a las VM entre sí y a las VM del hipervisor.

KVM (1)

Arquitectura



Source: IBM, 2011

KVM (1)

Extensiones nativas para x86

- KVM necesita de forma obligatoria las extensiones de virtualización incluidas por Intel y AMD:
  - Intel Virtualization Technology (Intel VT-x), codename *Vanderpool* (2005).
  - AMD Virtualization (AMD-V), codename *Pacifica* (2006).
- Permiten a los hipervisores un rendimiento mayor en modo virtualización completa.
- De esta forma la virtualización completa es mucho más fácil de implementar y ofrece un mayor rendimiento.
- Básicamente estas extensiones añaden un nuevo modo de ejecución que permiten la ejecución de SSOO invitados sin modificar de forma eficiente, pero sin dar un control total a ciertos recursos como la memoria y el procesador.

KVM (2)

Extensiones nativas para x86

- Aunque el procesador la incluya, hay que activarla en BIOS.

KVM y Windows (1)

Aunque KVM esté muy fuertemente ligado a Linux, el soporte de Windows como invitado es algo fundamental.

- El origen del soporte de Windows está en las misma raíces del proyecto KVM.
- Qumranet inició el desarrollo de KVM pensando en la virtualización de escritorios Windows.
- Más tarde, en 2009, Red Hat firmó con Microsoft un acuerdo de interoperabilidad, por el que acordaron probar y soportar sus SSOO sobre los hipervisores de la otra compañía.
- De esta forma se asegura el rendimiento de los SSOO invitados así como buenas "maneras" entre ambas empresas.
- SUSE y Microsoft firmaron un acuerdo parecido.

Como invitados funcionan las siguientes versiones de Windows:

KVM y Windows (2)

- Windows Server (2008/2008R2/2003/2000), Windows 7, Windows Vista, Windows XP, Windows NT.

Gestión de la virtualización (1)

Xen: una historia tumultuosa

Un aspecto clave para el éxito de un despliegue en virtualización es la gestión.

- Uno de los aspectos que ha impedido el éxito de Xen ha sido el no presentar una interfaz de gestión única, lo que llevó a varias versiones de Xen en el mercado incompatibles entre sí.

Xen era el hipervisor por defecto en RHEL 5, Red Hat lo reemplazó completamente por KVM en RHEL 6.

Gestión de la virtualización (1)

Los estándares son buenos

Linux y KVM se han estandarizado a través de **libvirt** y **libguestfs**, usadas como APIs base para la gestión de las máquinas virtuales y de las imágenes.

- Además, libvirt es capaz de gestionar no solo a KVM, sino a otros hipervisores como Xen, VMware ESX, VMware Workstation/Player, OpenVZ, MS Hyper-V, VirtualBox, etc.
- Muchas de las herramientas utilizadas de alto nivel como **virsh** ó **virt-manager** han sido construidas sobre la librería base **libvirt**.

Gestión de la virtualización (1)

KVM y libvirt: arquitectura

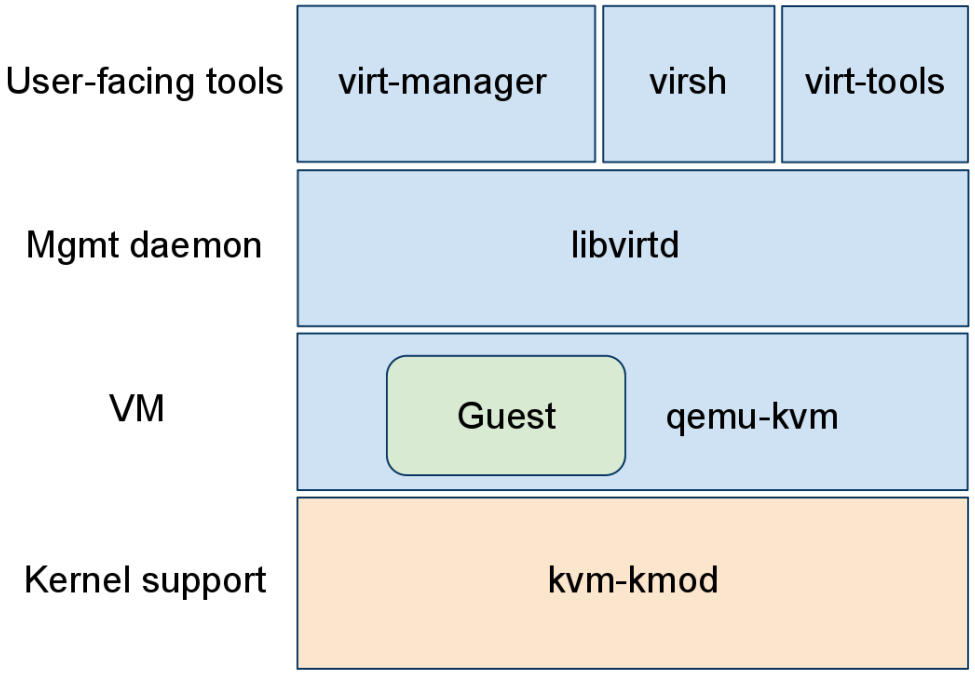


Figura: ©IBM DeveloperWorks https://www.ibm.com/developerworks/

Características principales (1)

Lo que KVM ofrece

Las características principales de KVM son:

- **Seguridad**
  - Bajo KVM, cada máquina virtual se implementa como un proceso.
  - KVM se aprovecha del modelo de seguridad estándar de Linux: SELinux/AppArmor. Estos modelos proporcionan el aislamiento y el control de recursos necesarios.
  - El proyecto **SVirt**, un esfuerzo de la comunidad para integrar el control de acceso obligatorio (MAC) con KVM, está construido sobre SELinux y proporciona una infraestructura que permite al administrador definir políticas de aislamiento para las VMs.
  - Básicamente SVirt se asegura que los recursos de una VM no puedan ser accedidos por otro proceso (ó VM).
- **Gestión de memoria**

Características principales (2)

Lo que KVM ofrece

- KVM hereda las características de gestión de memoria de Linux. La memoria utilizada por una VM se gestionará de la misma forma que la de otro proceso, podrá ser guardada en disco (swapped), utilizada en páginas grandes (large pages). El soporte NUMA de Linux, permite también el uso a las VMs de grandes cantidades de memoria.
- KVM soporta las últimas características de virtualización de memoria proporcionadas por los fabricantes como EPT (Extended Page Table de Intel) ó RVI (Rapid Virtualization Indexing de AMD). Estas tecnologías persiguen reducir el uso de la CPU y aumentar el rendimiento de los hipervisores.
- La compartición de páginas de memoria se consigue a través de la características añadida a Linux llamada KSM (Kernel Same-page Merging).

Características principales (3)

Lo que KVM ofrece

    - KSM escanea las páginas de memoria de cada máquina virtual, si dos páginas coinciden, KSM las une en una sola página que se comparte entre las dos máquinas, almacenando únicamente una copia.
    - Si en cualquier momento, una de las VM modifica la página, se le da una copia privada.
  - KSM permite aumentar el número de máquinas virtuales que un solo sistema puede alojar.
- **Almacenamiento**
  - KVM puede utilizar cualquier tipo de almacenamiento soportado por Linux para el almacenamiento de las imágenes (discos) de las VMs.
  - Esto incluye: discos locales (IDE/SCSI/SATA), NAS (Network Attached Storage: NFS, SAMBA/CIFS, etc) ó SAN (iSCSI y Fibre Channel). La E/S Multipath se puede utilizar para mejorar el rendimiento y proporcionar redundancia.

Características principales (4)

Lo que KVM ofrece

- KVM también soporta el almacenamiento de imágenes en sistemas de ficheros distribuidos como GFS2, OCFS ó GlusterFS. De esta forma las imágenes de las máquinas virtuales pueden ser compartidas por varios hipervisores.
- Las imágenes de disco soportan aprovisionamiento bajo demanda evitando tener que reservar todo el espacio inicialmente. El formato nativo de KVM es QCOW2, el cual permite la realización de snapshots, compresión y cifrado.

- **Live migrations**
  - KVM permite migraciones en caliente (live migrations), esta característica permite mover una VM en ejecución entre servidores físicos (hipervisores) sin interrupción del servicio.

Características principales (5)

Lo que KVM ofrece

- Estas migraciones son transparentes al usuario, la VM permanece encendida, las conexiones de red activas y las aplicaciones en ejecución mientras que la máquina se realoja en un nuevo servidor físico.
- KVM también permite almacenar el estado de una VM a disco para permitir su almacenamiento y posterior reanudación.

- **Drivers**
  - KVM soporta virtualización híbrida. En los SSOO invitados hay que instalar drivers paravirtualizados que permiten utilizar una interfaz de E/S optimizada en vez de emular estos dispositivos. Estos drivers permiten altos ratios de rendimiento en la E/S para dispositivos de bloques y dispositivos de red.
  - El hipervisor KVM utiliza el estándar VirtIO. VirtIO es un estándar de drivers paravirtualizados desarrollado por IBM y Red Hat con la ayuda de la comunidad Linux.

Características principales (6)

Lo que KVM ofrece

- VirtIO es una interfaz independiente del hipervisor diseñada para el desarrollo de drivers que puedan ser utilizados sobre varios hipervisores. El objetivo principal es conseguir una mayor interoperabilidad con los invitados.
- Los drivers VirtIO están incluidos en todos los kernel modernos de Linux (a partir de la versión 2.6.25).
- Red Hat ha desarrollado drivers VirtIO optimizados para E/S de red y disco para los SSOO Windows. Dichos drivers están certificados por Microsoft a través de su programa de certificación WHQL (Microsoft's Windows Hardware Quality Labs).

- **Rendimiento y escalabilidad**
  - KVM posee los mismos rasgos de rendimiento y escalabilidad que caracteriza a Linux.
  - KVM soporta VM de hasta 16 CPUs virtuales y 256 GB de RAM.

Características principales (7)

Lo que KVM ofrece

- El rendimiento de aplicaciones como SGBD Oracle, SAP, LAMP, MS Exchange sobre KVM puede oscilar entre el 95 % y el 135 % comparado con su ejecución en servidores físicos.
- Se han conseguido ratios de hasta 600 máquinas virtuales en un solo servidor físico.

Implementaciones comerciales (1)

Como contribuidor de KVM, Red Hat es una de las principales empresas en soportar de forma comercial a KVM.

- Red Hat introdujo a KVM por primera vez en RHEL 5.4, adoptando únicamente KVM y abandonando Xen en RHEL 6.

Red Hat empaqueta KVM de dos formas distintas:

- Como parte de su distribución Red Hat Enterprise Linux (RHEL).
- De forma independiente, en su producto RHEV-H.

Pero no únicamente Red Hat:

- La empresa SUSE a través de su SLES, soporta ambos, KVM y Xen.
- Canonical a través de su distribución Ubuntu.
- Todas las demás, ya que KVM forma parte de la rama principal del kernel de Linux.

KVM y la nube (1)

Los hipervisores y las tecnologías de virtualización han ido creciendo y expandiendo su influencia en varios campos. Uno de las evoluciones lógicas ha sido su uso como elemento básico para la creación de nubes de muy diversos tipos y usos. Varios proyectos sobre Cloud Computing utilizan KVM como hipervisor, por citar alguna:

- OpenStack.
- CloudStack.
- OpenNebula.
- Etc.

KVM (1)

Ventajas y desventajas

Ventajas:

- Incluido en la rama principal del kernel de Linux. Instalación prácticamente nula.
- Cuenta con todas las ventajas que le proporciona ser parte integral del kernel de Linux.
- KVM es un hipervisor ligero, de alto rendimiento y bajo coste.
- Gran soporte.
- Listo para su uso en entornos en producción.
- Modelos de seguridad avanzado proporcionado por SELinux.
- Soporte de invitados Windows, Linux, Android, Familia BSD (OpenBSD, FreeBSD, NetBSD), Solaris, etc.

KVM (2)

Ventajas y desventajas

- Lista completa en: http://www.linux-kvm.org/page/Guest_Support_Status

Desventajas:

- Proyecto muy joven.
- No hay herramientas sofisticadas para la gestión de servidores y para la gestión/creación de máquinas virtuales.
- KVM aún puede mejorar mucho más en áreas como: soporte de redes virtuales, soporte de almacenamiento virtual, seguridad, alta disponibilidad, tolerancia a fallos, gestión de energía, soporte HPC/tiempo real, etc.

KVM (3)

Ventajas y desventajas

- Ejemplo: actualmente se está trabajando en una forma de almacenamiento más eficiente denominada VirtFS. Consiste en un SF disponible en el anfitrión y en el invitado, el invitado es capaz de utilizar la caché que el anfitrión mantiene sobre el SF, acelerando el acceso a ficheros en el invitado.

Virtualización de Servidores: KVM

¿Por dónde vamos?

Usando KVM
- Instalación
- Configuración Inicial
- Gestión de Máquinas Virtuales
- Almacenamiento: pools y volúmenes
- Formatos de imágenes
- Virtual Networking
- KSM
- Herramientas

Instalación (1)

Prerrequisitos

- Para la ejecución de KVM se necesita de un procesador con las extensiones de virtualización x86.
- Se puede comprobar esto a través de los comandos:

```
root@ayla:~# egrep -c '(vmx|svm)' /proc/cpuinfo
2
```

- 0 : no hay soporte para KVM.
- 1 ó más: hay soporte de virtualización para KVM. Pero hay que comprobar también que esté activo en BIOS.

Instalación (1)

Prerrequisitos

En Ubuntu está disponible el comando *kvm-ok*:

```
root@ayla:~# kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

Instalación (1)

¿Host de 32 ó 64 bits?

Para virtualización con KVM se recomienda que tanto el procesador como el sistema Linux utilizado sea de 64 bits:

- Se supera la barrera de los 2GB de memoria.
  - Con 32 bits no se puede asignar más de 2GB a una VM.
- Se pueden ejecutar máquinas virtuales de 32 y 64 bits.
  - Con hosts de 32 bits no se pueden ejecutar MV de 64 bits.

Se puede verificar la arquitectura con el comando:

```
root@ayla:~# uname -m
x86_64
```

Instalación (1)

Software necesario

- Instalamos los siguientes paquetes básicos:

```
root@ayla:~# apt-get install qemu-kvm libvirt-bin bridge-
    utils
```

  - **qemu-kvm**: sistema KVM. Incluyendo el módulo del kernel.
  - **libvirt-bin**: toolkit C para la interacción con el hipervisor.
  - **bridge-utils**: herramientas para la configuración de bridges Ethernet.
- Los usuarios que vayan a gestionar las máquinas virtuales deben pertenecer a los grupos:
  - *libvirtd* y *kvm*.

Instalación (1)

Software adicional

Como software adicional se puede instalar:

```
root@ayla:~# apt-get install virtinst virt-manager ubuntu-vm-
    builder virt-viewer
```

- **virtinst**: proporciona herramientas en línea de comandos para la creación y clonado de VMs.
- **virt-manager**: interfaz gráfica para la gestión de VMs.
- **ubuntu-vm-builder**: scripts para la automatización de la creación de VMs basadas en Ubuntu.
- **virt-viewer**: permite la conexión a la consola de la máquina virtual a través del protocolo VNC.

Instalación (1)

Verificando la instalación

Podemos asegurarnos de que todo funciona correctamente a través de:

```
alex@ayla:~# virsh -c qemu:///system list
 Id Nombre                    Estado
----------------------------------
```

Instalación (1)

Si hay algún problema...

Si surge algún problema de conexión y configuración, hay que comprobar los permisos de:

- El socket /var/run/libvirt/libvirt-sock
- El fichero de dispositivo /dev/kvm

```
root@ayla:~# ls -l /var/run/libvirt/libvirt-sock
srwxrwx--- 1 root libvirtd 0 2012-09-16 11:26 /var/run/libvirt/
    libvirt-sock
root@ayla:~# ls -l /dev/kvm
crw-rw----+ 1 root kvm 10, 232 2012-09-16 11:26 /dev/kvm
```

Si se cambian los permisos, hay que reiniciar el subsistema KVM:

Instalación (2)

Si hay algún problema...

```
root@ayla:~# rmmod kvm_intel
root@ayla:~# rmmod kvm
root@ayla:~# modprobe -a kvm_intel
root@ayla:~# modprobe -a kvm
```

Configuración inicial (1)

Almacenamiento de las máquinas virtuales

Para la configuración inicial de KVM hay que crear un pool de almacenamiento para guardar las imágenes de las VMs. Seguimos los siguientes pasos:

1. Editamos el fichero /tmp/pool-default.xml con el siguiente contenido:

```
<pool type='dir'>
  <name>default</name>
  <target>
    <path>/var/lib/libvirt/images</path>
  </target>
</pool>
```

2. Definimos el pool en libvirt:

```
root@ayla:~# virsh pool-define /tmp/pool-default.xml
```

Configuración inicial (2)

Almacenamiento de las máquinas virtuales

3. Lo iniciamos:

```
root@ayla:~# virsh pool-start default
```

4. Lo configuramos para que si inicie siempre de forma automática:

```
root@ayla:~# virsh pool-autostart default
```

5. Comprobamos:

```
root@ayla:~# virsh pool-list --all
Nombre                 Estado     Inicio automático
----------------------------------------
default                activo     si
```

6. El fichero de creación del pool se encuentra en: */etc/libvirt/storage/default.xml*

Configuración inicial (1)

Almacenamiento por defecto

- Por defecto se utiliza un pool de almacenamiento basado en directorio.
- El directorio por defecto es */var/lib/libvirt/images*
- Este directorio puede ser una partición aparte, un volumen lógico LVM o cualquier otro tipo de almacenamiento.
- Se pueden crear mucho más pools y de diferentes tipos.

Configuración inicial (1)

Red

Para la configuración de la red seguimos los siguientes pasos:

1. Iniciamos la red por defecto:

```
root@ayla:~# virsh net-start default
```

2. Configuramos para que se inicie siempre de forma automática:

```
root@ayla:~# virsh net-autostart default
```

3. Listamos la configuración de red disponible.

```
root@ayla:~# virsh net-list
Nombre                  Estado     Inicio automático
----------------------------------------
default                 activo     si
```

Configuración inicial (2)

Red

Sobre la configuración de red por defecto (default):

- La VMs reciben una IP por DHCP.
- El anfitrión hace NAT para la conexión de los invitados.
- El fichero de configuración se encuentra en: */etc/libvirt/qemu/networks/default.xml*

Configuración inicial (1)

Red

El contenido del fichero es:

```
<network>
  <name>default</name>
  <bridge name="virbr0" />
  <forward/>
  <ip address="192.168.122.1" netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.122.2" end="192.168.122.254" />
    </dhcp>
  </ip>
</network>
```

Configuración Inicial (1)

Servicio libvirt

Siempre que hagamos cualquier cambio de configuración podemos reiniciar el sistema libvirt a través de:

```
root@ayla:~# service libvirt-bin restart
```

También:

```
root@ayla:~# service libvirt-bin start
```

```
root@ayla:~# service libvirt-bin stop
```

```
root@ayla:~# service libvirt-bin status
```

Creación de máquinas virtuales (1)

Varias formas

- Hay varias utilidades que permiten la creación de máquinas virtuales sobre KVM.
- Destacamos las siguientes:
  - Comando **virt-install**.
  - Comando **vmbuilder** (antes ubuntu-vm-builder).
  - Aplicación gráfica **virt-manager**.
  - Directamente usando QEMU/KVM (comando **qemu**/**kvm**).

virt-install (1)

Descripción

- **virt-install** es un comando que permite el aprovisionamiento de nuevas máquinas virtuales.
- Es una herramienta en línea de comandos que permite la creación de máquinas virtuales Xen y KVM utilizando libvirt.

virt-install (1)

Características

Como características destacamos:

- Está basada en libvirt, por lo que puede trabajar sobre varios hipervisores.
- Permite la instalación de SSOO tanto en modo texto (consola serie) como en modo gráfico (VNC ó SDL).
- Se pueden crear VM con varios discos, varias interfaces de red, dispositivos de audio, dispositivos USB ó PCI, entre otros.
- Soporta varios métodos de instalación:
  - Locales: discos duros, ficheros, unidad CD/DVD, ...
  - Remotos: NFS, HTTP, FTP, ...
  - Por red: PXE.
  - A partir de imágenes existentes.

virt-install (1)

Uso

Básicamente hay que indicar:

- Nombre de la VM.
- Cantidad de RAM.
- Almacenamiento.
- Método de instalación.

Por lo que las opciones obligatorias son son: *--name*, *--ram*, *--disk*; más las relativas a la instalación como *--cdrom*, *--location*, ...

virt-install (1)

Ejemplo: instalar Ubuntu 12.04 desde una imagen ISO

```
virt-install --connect qemu:///system
             --virt-type=kvm
             --name VirtualMachine01
             --ram 1024
             --vcpus=2
             --disk path=/var/lib/libvirt/images/
                 VirtualMachine01.img,size=8
             --cdrom /var/lib/libvirt/images/ubuntu-12.04-
                 server-amd64.iso
             --os-type linux
             --os-variant=ubuntuprecise
             --graphics vnc,keymap=es
             --noautoconsole
             --network network=default
             --description "Ubuntu 12.04 Server"
```

virt-install (1)

Varias cosas sobre la máquina recién creada

Nos podemos conectar a la consola para ver el proceso de instalación a través del protocolo VNC, de varias formas:

```
alex@ayla:~# virt-viewer -c qemu:///system VirtualMachine01
```

```
alex@ayla:~# vinagre localhost:0
```

```
alex@ayla:~# vncviewer localhost:0
```

Se puede configurar el acceso a VNC a través de un password. Para ello creamos la máquina virtual usando el parámetro:

```
virt-install [...]
             --graphics vnc,keymap=es,password=myPassword
             [...]
```

virt-install (2)

Varias cosas sobre la máquina recién creada

Una vez acabado el proceso de instalación, podemos volver a ejecutar la VM a través del comando:

```
alex@ayla:~# virsh --connect qemu:///system start
    VirtualMachine01
```

virt-install (1)

Ejemplo: instalar una Ubuntu 12.04 iniciando desde PXE

```
virt-install --connect qemu:///system
             --virt-type=kvm
             --name VirtualMachine01
             --ram 1024
             --vcpus=2
             --disk path=/var/lib/libvirt/images/
                 VirtualMachine01.img,size=8
             --os-type linux
             --os-variant=ubuntuprecise
             --graphics vnc,keymap=es
             --noautoconsole
             --network network=default
             --description "Ubuntu 12.04 Server"
```

virt-install (1)

Ejemplo: instalar Ubuntu 12.04 con un instalador HTTP

```
virt-install --connect qemu:///system
             --virt-type=kvm
             --name VirtualMachine01
             --ram 1024
             --vcpus=2
             --disk path=/var/lib/libvirt/images/
                 VirtualMachine01.img,size=8
             --location http://archive.ubuntu.com/ubuntu/dists/
                 precise/main/installer-amd64/
             --os-type linux
             --os-variant=ubuntuprecise
             --graphics vnc,keymap=es
             --noautoconsole
             --network network=default
             --description "Ubuntu 12.04 Server"
```

virt-install (1)

Ejemplo: instalar Ubuntu 12.04 desde un CD físico

```
virt-install --connect qemu:///system
             --virt-type=kvm
             --name VirtualMachine01
             --ram 1024
             --vcpus=2
             --disk path=/var/lib/libvirt/images/
                 VirtualMachine01.img,size=8
             --cdrom /dev/cdrom
             --os-type linux
             --os-variant=ubuntuprecise
             --graphics vnc,keymap=es
             --noautoconsole
             --network network=default
             --description "Ubuntu 12.04 Server"
```

virt-install (1)

Ejemplo: instalar Windows Server 2008R2 desde una imagen ISO

```
virt-install --connect qemu:///system
             --virt-type=kvm
             --name WindowsServer2008
             --ram 1024
             --vcpus=2
             --disk path=/var/lib/libvirt/images/WindowsServer.
                 img,size=10
             --cdrom /media/WIN2K8/WindowsServer2008R2.iso
             --os-type windows
             --os-variant=win2k8
             --graphics vnc,keymap=es
             --noautoconsole
             --network network=default
             --description "Windows Server 2008 R2"
```

virt-install (1)

Instalación de Windows XP/2003/7/2008 con drivers VirtIO

Para la instalación de Windows hay que tener en cuenta los siguientes pasos:

1. Lanzamos el comando virt-install prestando especial atención a:
   - *--disk ...,bus=virtio*
   - *--network ...,model=virtio*
2. Iniciamos la instalación de Windows añadiendo un CD con los drivers VirtIO para Windows.
3. Cuando el instalador de Windows solicite una unidad para la instalación, cargamos los drivers.
4. Acabamos la instalación de Windows.

virt-install (1)

Ejemplo: instalar Windows Server 2008R2 con drivers VirtIO

```
virt-install --connect qemu:///system
             --virt-type=kvm
             --name WindowsServer2008
             --ram 1024
             --vcpus=2
             --disk path=/var/lib/libvirt/images/WindowsServer.
                 img,size=10,bus=virtio
             --cdrom /media/WIN2K8/WindowsServer2008R2.iso
             --disk path=/media/WIN2K8/virtio-win-0.1-30.iso,
                 device=cdrom
             --os-type windows
             --os-variant=win2k8
             --graphics vnc,keymap=es
             --noautoconsole
             --network network=default,model=virtio
             --description "Windows Server 2008 R2"
```

vmbuilder (antes ubuntu-vm-builder) (1)

Descripción

- **ubuntu-vm-builder** es un comando que permite la creación de máquinas virtuales de línea de comandos.
- El comando es capaz de instalar una máquina virtual de forma desatendida.
- Renombrado a **vmbuilder** en las últimas versiones.
- Soporta varios hipervisores como KVM, Xen, VMware Workstation 6 y VMware Server. Como distribuciones solo soporta a Ubuntu.
- El programa hace lo siguiente:
  - Crea un directorio en el directorio actual.
  - Crea un disco virtual.
  - Realiza una instalación de Ubuntu en dicho disco.
  - Instalación muy rápida si se opta por un mirror local.

vmbuilder (1)

Ejemplo: instalar Ubuntu 12.04

```
vmbuilder kvm ubuntu
          --suite precise
          --flavour virtual
          --arch amd64
          --hostname ubuntu-server
          --libvirt qemu:///system
          --rootsize=2046
          --swapsize=256
          --user alex
          --pass changeMe
          -d prueba
          --addpkg openssh-server
          --mirror http://de.archive.ubuntu.com/ubuntu
```

Gestión de máquinas virtuales (1)

XML

Se utilice la herramienta que se utilice, la configuración de la máquina virtual reside en un fichero XML como éste:

```
<domain type='kvm'>
  <name>Ubuntu01</name>
  <uuid>18dbd172-97f7-242f-ebc8-c8b377b8ec71</uuid>
  <description>Ubuntu 12.04 Server</description>
  <memory>524288</memory>
  <currentMemory>524288</currentMemory>
  <vcpu>2</vcpu>
  <os>
    <type arch='x86_64' machine='pc-1.0'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
```

Gestión de máquinas virtuales (2)

XML

```
<emulator>/usr/bin/kvm</emulator>
<disk type='file' device='disk'>
  <driver name='qemu' type='raw'/>
  <source file='/var/VirtualMachines/libvirt/Ubuntu01.img'/>
  <target dev='vda' bus='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
</disk>
<disk type='block' device='cdrom'>
  <driver name='qemu' type='raw'/>
  <target dev='hdc' bus='ide'/>
  <readonly/>
  <address type='drive' controller='0' bus='1' unit='0'/>
</disk>
<controller type='ide' index='0'>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1'/>
</controller>
<interface type='network'>
  <mac address='52:54:00:75:cc:68'/>
  <source network='default'/>
  <model type='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
<serial type='pty'>
  <target port='0'/>
</serial>
```

Gestión de máquinas virtuales (3)

XML

```
    <console type='pty'>
      <target type='serial' port='0'/>
    </console>
    <input type='mouse' bus='ps2'/>
    <graphics type='vnc' port='-1' autoport='yes' keymap='es'/>
    <video>
      <model type='cirrus' vram='9216' heads='1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
    </video>
    <memballoon model='virtio'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
    </memballoon>
  </devices>
</domain>
```

Gestión de máquinas virtuales (1)

Herramientas

Para la gestión de las máquinas virtuales se cuenta con dos herramientas:

- **virsh**: herramienta en modo comando.
- **virt-manager**: herramienta gráfica.

Gestión de máquinas virtuales con virsh (1)

Descripción

- **virsh** es la principal herramienta para la gestión completa de KVM.
- Está basada en libvirt.
- Permite iniciar,pausar y apagar máquinas virtuales. Pero también permite otras funciones como gestión de la red, de los pool de almacenamiento, de los volúmenes, de las snapshots, etc.
- Presenta dos modos de funcionamiento, modo comando y modo interactivo.
- Como comando su sintaxis es la siguiente:

```
virsh [OPTION]... [COMMAND\_STRING]

virsh [OPTION]... COMMAND [ARG]...
```

virsh (1)

Comandos

Para conectarnos al hipervisor podemos optar por:

- Conectarnos en local:

```
virsh
virsh -c qemu:///system
```

- Conectarnos a un hipervisor remoto:

```
virsh -c qemu+ssh://usuario@host:port/system
```

Entre los comandos genéricos más importantes de virsh destacamos:

help Muestra la ayuda.

quit

exit Sale del intérprete.

virsh (2)

Comandos

cd	Cambia el directorio actual.
pwd	Muestra el directorio actual.
connect	Se (re)conecta al hipervisor.
uri	Muestra la URI del hipervisor al que estamos conectados.
hostname	Muestra el hostname del hipervisor.
sysinfo	
nodeinfo	
nodecpustats	
nodememstats	
capabilities	Muestra diversa información sobre el hipervisor/nodo.

virsh (3)

Comandos

- **list** Muestra un listado de las VMs que están en ejecución.
- **list --all** Muestra un listado de todas las VMs.

Entre los comandos sobre dominios más importantes destacamos:

- **console** Conecta a la consola virtual de una VM.
- **create** Crea un VM desde su fichero de configuración XML.
- **dumpxml** Muestra la configuración de la VM en XML.
- **edit** Permite editar la configuración XML de una VM.
- **dominfo** Muestra la información básica de una VM
- **migrate** Permite migrar una VM de un anfitrión a otro.
- **start** Inicia una VM.

virsh (4)

Comandos

autostart Permite configurar una VM para que se inicie de forma automática durante el inicio del anfitrión.

shutdown Apaga la VM de forma ordenada.

destroy Termina inmediatamente una VM, apagado hardware.

reboot Reinicia una VM.

reset Reinicia inmediatamente una VM.

save

restore Detiene/restaura una VM. Al detenerla se realiza un volcado de la RAM a un fichero en disco.

suspend

resume Suspende/reinicia una VM. Una VM suspendida sigue en memoria pero no entra en la planificación del hipervisor.

virsh (5)

Comandos

undefine Elimina la VM. No borra los discos virtuales. Antes de ejecutar este comando conviene parar la VM con el comando *destroy*.

vncdisplay Muestra la dirección IP y el número de puerto de la consola VNC asociada a la máquina virtual.

Para más ayuda sobre comandos de dispositivos, red, interfaces de red, pools de almacenamiento, volúmenes, contraseñas, snapshots, filtros de red, etc., se recomienda leer la página man del comando virsh.

Gestión de máquinas virtuales con virt-manager (1)

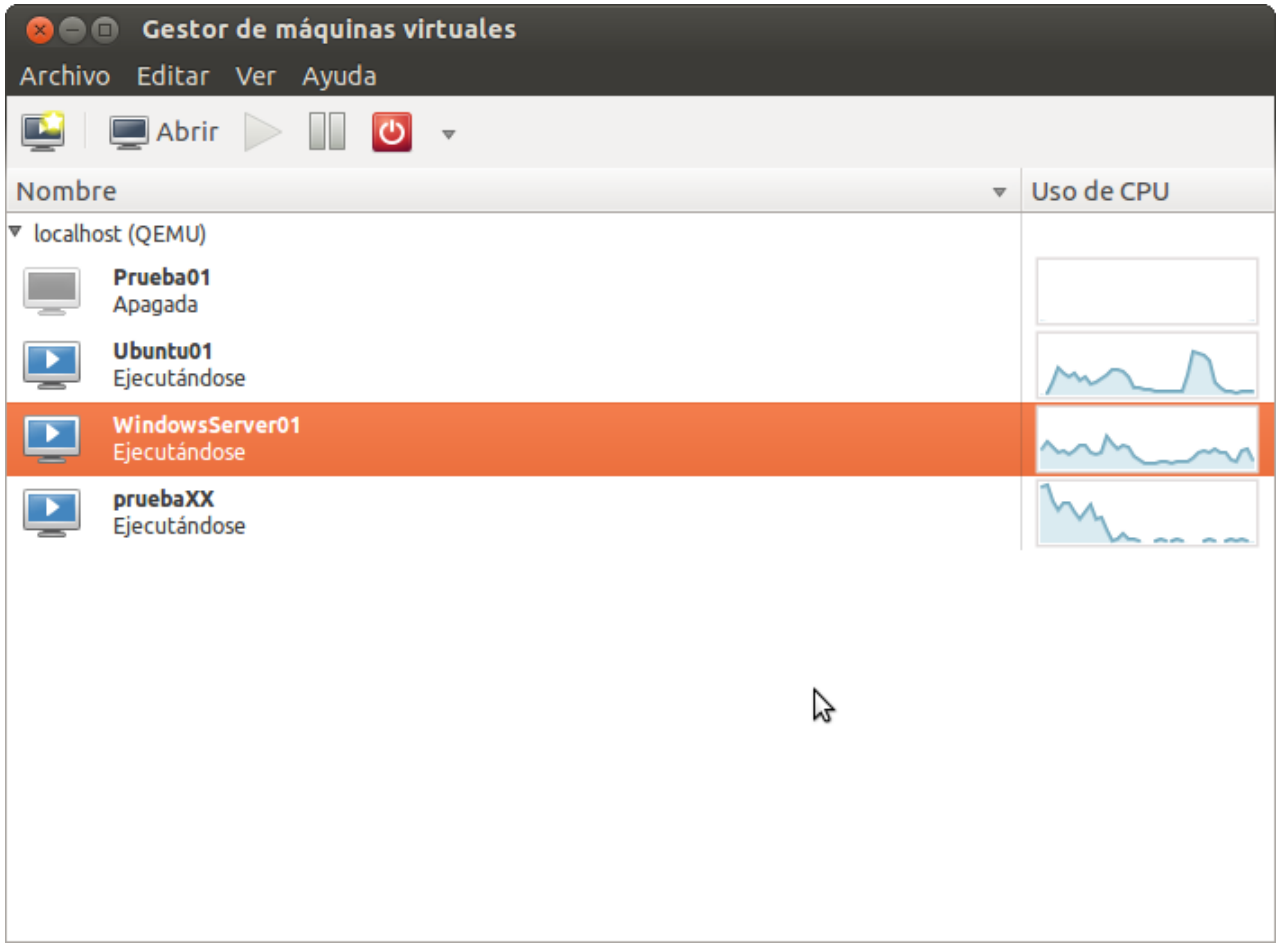
Descripción

virt-manager es una herramienta gráfica para la gestión de máquinas virtuales. Proporciona las siguientes características:

- Capaz de gestionar todo el ciclo de vida de una VM (start/shutdown, save/restore, suspend/resume).
- Permite crear máquinas virtuales.
- Permite la gestión de pools de almacenamiento.
- Gestión de redes virtuales.
- Permite el acceso a la consola gráfica de la VM.
- Muestra estadísticas de rendimiento.
- Permite conectarse a hipervisores remotos.
- Muy completa.

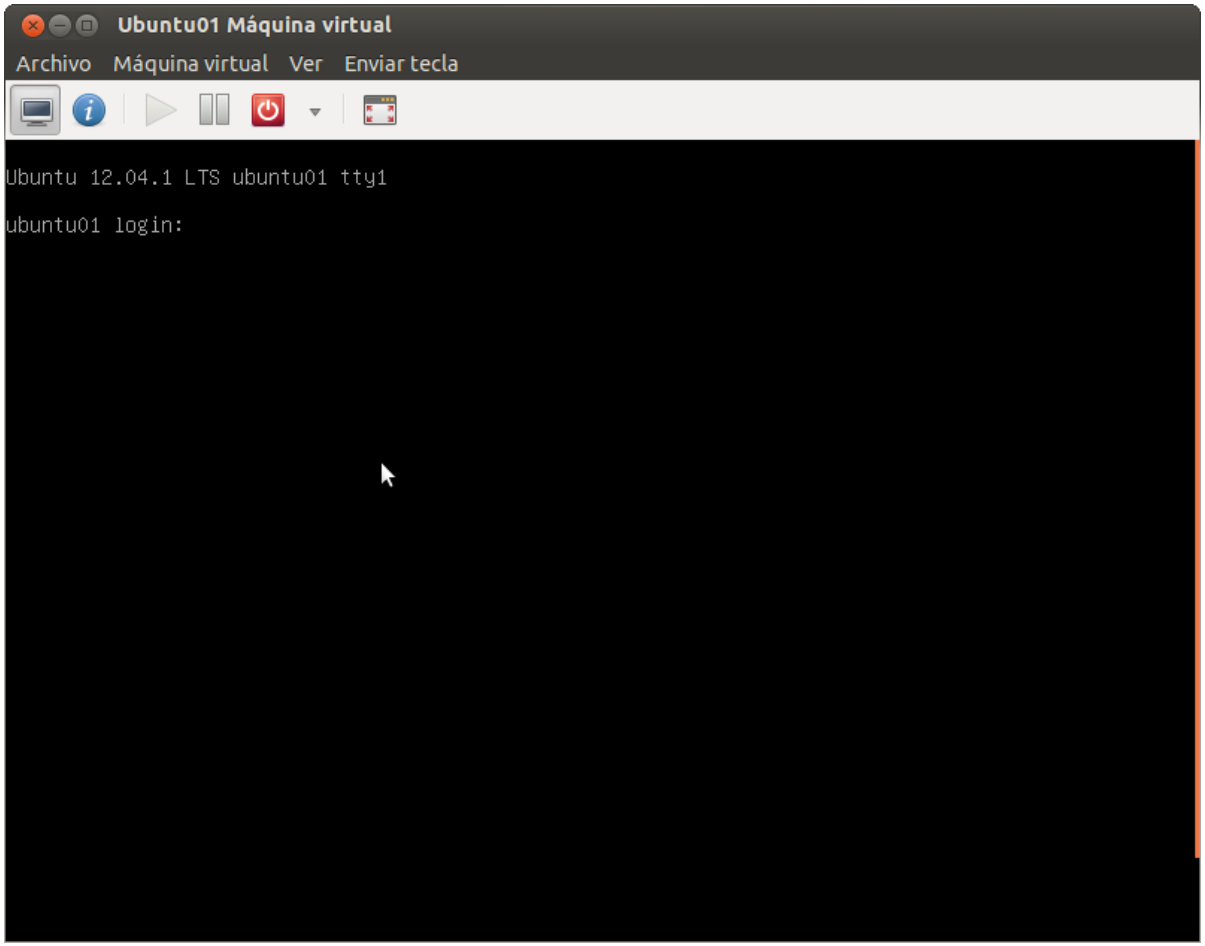
virt-manager

Vista principal



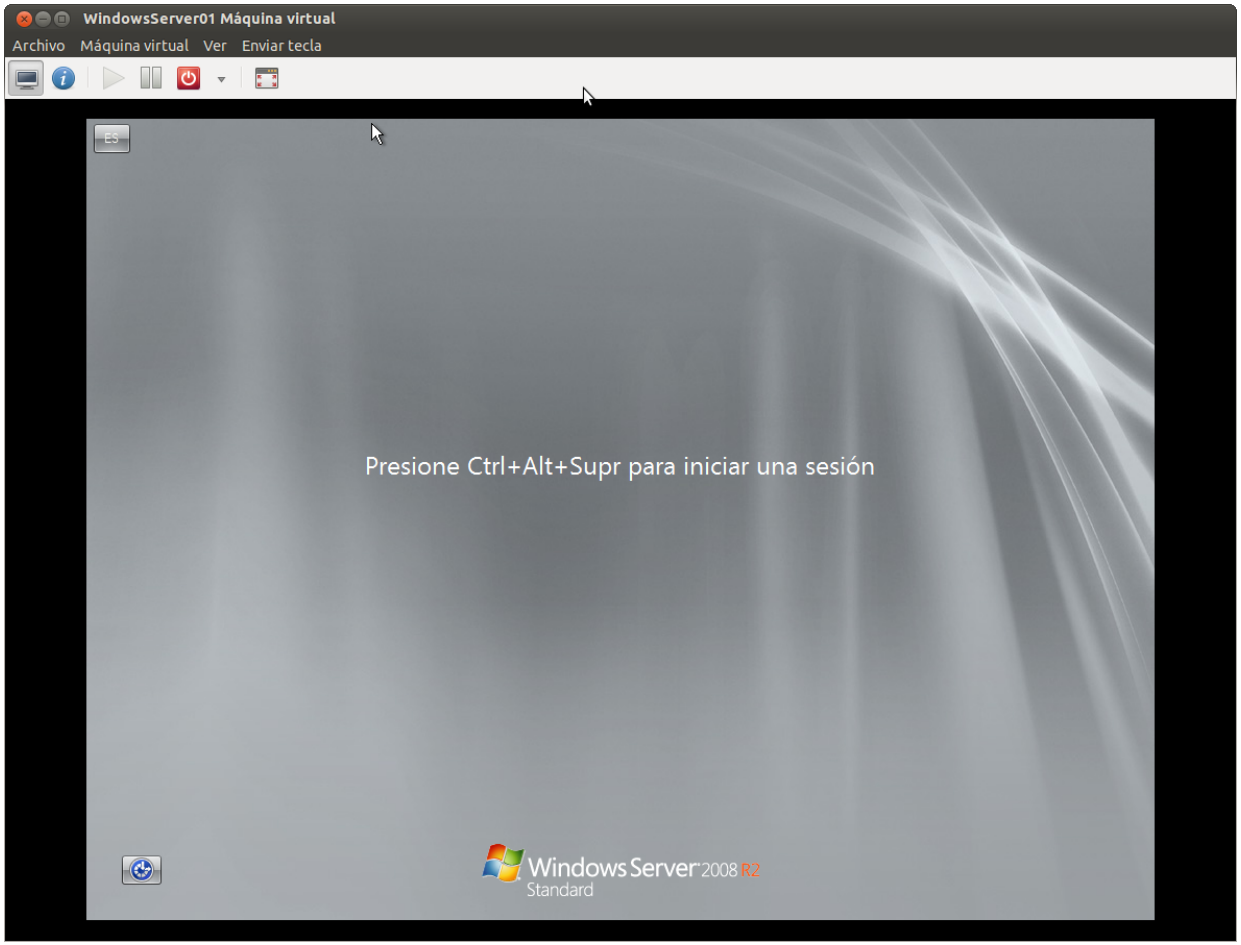
virt-manager

Ejecutando una máquina virtual Ubuntu



virt-manager

Ejecutando una máquina virtual Windows Server 2008 R2



KVM y el almacenamiento (1)

Tipos de almacenamiento

Desde el punto de vista de la virtualización, KVM puede utilizar dos tipos de almacenamiento:

- **Almacenamiento local**.
  - Es el almacenamiento que se produce al conectar directamente a un host un dispositivo de almacenamiento.
  - Muy útil para entornos de desarrollo, pruebas y pequeños despliegues.
  - No se recomienda para entornos con muchas máquinas virtuales o que requieran migraciones en caliente (live migrations).
  - Ejemplos: directorios locales, discos (PATA/SATA/SAS) y grupos de volúmenes LVM.
- **Almacenamiento en red**
  - Cubre dispositivos de almacenamiento que se comparten en una red a través de protocolos estándares.

KVM y el almacenamiento (2)

Tipos de almacenamiento

- Ejemplos: almacenamiento compartido a través de los protocolos Fibre Channel, iSCSI, NFS, GFS2 y SCSI RDMA (Infiniband).

Pools de Almacenamiento (1)

Definición

Pool de almacenamiento

Un *pool de almacenamiento* es un fichero, un directorio o cualquier dispositivo de almacenamiento gestionado por *libvirt* con el propósito de proporciona almacenamiento para las máquinas virtuales.

Como pool de almacenamiento por defecto, libvirt utiliza el directorio **/var/lib/libvirt/images**. Esto es configurable.
También se puede utilizar almacenamiento no gestionado por libvirt.

Volúmenes (1)

Definición

Un pool de almacenamiento se divide en volúmenes de almacenamiento.

Volumen

Un volumen de almacenamiento es la abstracción de una partición física, un volumen LVM, un imagen de disco basada en fichero o cualquier otro tipo de almacenamiento gestionado por libvirt.

Los volúmenes de almacenamiento se presentan a las máquinas virtuales como almacenamiento local.

Volúmenes (1)

Cómo referenciar un volumen

Se puede referenciar un volumen específico de tres formas:

- Por el nombre del pool de almacenamiento y del volumen.
- Por la ruta completa del almacenamiento en el anfitrión.
- Por su identificador único.

Ejemplos:

```
alex@ayla:~# virsh vol-info --pool VMs Prueba01.img
Nombre:          Prueba01.img
Tipo:            archivo
Capacidad:       4,00 GB
Ubicación:      1,27 GB

alex@ayla:~# virsh vol-info /media/MUSICA/libvirt/Prueba01.img
Nombre:          Prueba01.img
Tipo:            archivo
Capacidad:       4,00 GB
```

Volúmenes (2)

Cómo referenciar un volumen

```
Ubicación:     1,27 GB

alex@ayla:~# virsh vol-info d3ec47d1-b028-46c1-b8b6-72748355
    f7ae
Nombre:         ImagenPrueba01
Tipo:           block
Capacidad:      20,00 GB
Ubicación:     20,00 GB
```

Creación de pools de almacenamiento (1)

- No es aconsejable asignar dispositivos completos a máquinas virtuales.
  - Ejemplo: mejor asignar una partición como /dev/sdb1 que un dispositivo como /dev/sdb.
  - Al pasar un dispositivo completo, el invitado puede crear particiones o volúmenes LVM, esto puede ocasionar problemas en el anfitrión.

Para crear el pool seguimos los siguientes pasos:

1. En el caso de pools basados en dispositivos, creamos una tabla de particiones GPT.
2. Creamos el fichero XML definiendo el pool.
3. Definimos el pool
4. Iniciamos el pool y lo marcamos para que se inicie automáticamente.

Creación de pools de almacenamiento (1)

Ejemplo: pool basado en dispositivos

1. Creamos en el dispositivo una tabla de particiones GPT:

```
root@ayla:~# parted /dev/sdc
GNU Parted 2.3
Usando /dev/sdc
¡Bienvenido/a a GNU Parted! Teclee 'help' para ver la lista
    de órdenes.
(parted) mklabel
¿Nuevo tipo de etiqueta de disco? gpt
Aviso: La etiqueta de disco existente en /dev/sdc se
    destruirá y todos los datos
en este disco se perderán. ¿Desea continuar?
Sí/Yes/No? Sí
(parted) quit
Información: Puede que sea necesario actualizar /etc/fstab.
root@ayla:~#
```

Creación de pools de almacenamiento (2)

Ejemplo: pool basado en dispositivos

2. Creamos el fichero XML definiendo el pool en /root/pool-disk.xml con el siguiente contenido:

```
<pool type='disk'>
  <name>pool-disk</name>
  <source>
    <device path='/dev/sdc'/>
    <format type='gpt'/>
  </source>
  <target>
    <path>/dev</path>
  </target>
</pool>
```

---

3. Definimos el pool:

Creación de pools de almacenamiento (3)

Ejemplo: pool basado en dispositivos

```
root@ayla:~# virsh pool-define pool-disk.xml
El grupo pool-disk ha sido definido desde pool-disk.xml

root@ayla:~# virsh pool-list --all
Nombre                 Estado     Inicio automático
-----------------------------------------
default                activo     si
pool-disk              inactivo   no
VMs                    inactivo   si
```

4. Iniciamos el pool y lo marcamos para que se inicie automáticamente:

Creación de pools de almacenamiento (4)

Ejemplo: pool basado en dispositivos

```
root@ayla:~# virsh pool-start pool-disk
Se ha iniciado el grupo pool-disk

root@ayla:~# virsh pool-autostart pool-disk
El grupo pool-disk ha sido marcado como iniciable automá
    ticamente

root@ayla:~# virsh pool-list --all
Nombre               Estado     Inicio automático
-----------------------------------------
default              activo     si
pool-disk            activo     si
VMs                  inactivo   si
```

5. Verificamos el tamaño asignado:

Creación de pools de almacenamiento (5)

Ejemplo: pool basado en dispositivos

```
root@ayla:~# virsh pool-info pool-disk
Nombre:          pool-disk
UUID:            8d9fbcd7-97ca-e037-5c0b-a0a861cf4eac
Estado:          ejecutando
Persistente:     si
Autoinicio:      si
Capacidad:       3,73 GB
Ubicación:      0,00
Disponible:      3,73 GB
```

Creación de pools de almacenamiento (1)

Ejemplo: pool basado en directorios

La creación es muy parecida a la anterior, salvo en:

- El fichero XML de definición del pool tiene este aspecto:

```
<pool type='dir'>
    <name>VMs</name>
    <target>
        <path>/media/VMs/libvirt</path>
    </target>
</pool>
```

- Al directorio debe tener acceso el usuario *libvirt-qemu* y el grupo *kvm*.

Uso de pools (1)

virt-install

- Podemos utilizar un pool indicandolo en el comando de creación de la máquina virtual.
- Utilizamos el parámetro *--disk*:

```
virt-install ...
    --disk pool=pool-disk,size=2
...
```

- ... esto creará un volumen de 2GB dentro del pool, el nombre del volumen coincidirá con el de la máquina virtual.

Gestión de volúmenes (1)

Creación de volúmenes

Se puede crear un volumen dentro de un pool de almacenamiento a través del siguiente comando:

```
virsh # vol-create-as pool-myDisk volume01 2G
Se ha creado el volumen volume01
virsh #
```

Podemos listar los volúmenes a través del comando:

```
virsh # vol-list pool-myDisk
Nombre                  Camino
-----------------------------------------
volume01                /dev/sdc1
```

Gestión de volúmenes (1)

Clonación

Podemos clonar un volumen dentro del mismo pool a través del comando:

```
virsh # vol-clone --pool pool-myDisk volume01 clone01
Se ha clonado el volumen clone01 a partir de volume01

virsh # vol-list pool-myDisk
Nombre                  Camino
-----------------------------------------
clone01                 /dev/sdc2
volume01                /dev/sdc1
```

Gestión de volúmenes (1)

Borrado

Podemos borrar un volumen a través del comando:

```
virsh # vol-delete clone01 --pool pool-myDisk
Se ha eliminando el volumen clone01

virsh # vol-delete volume01 --pool pool-myDisk
Se ha eliminando el volumen volume01

virsh # vol-list pool-myDisk
Nombre                Camino
-----------------------------------------
```

Gestión de volúmenes (1)

Uso de volúmenes

Durante la instalación de una máquina virtual podemos indicar que use como disco virtual un volumen disponible:

- Tanto en la interfaz gráfica *virt-manager*...
- ...como con el comando *virt-install* con el formato *pool/volumen*. Ejemplo:

```
virt-install ...
    --disk vol=pool-myDisk/volume01
...
```

Añadiendo almacenamiento a una VM (1)

Añadiendo almacenamiento basado en fichero

Se puede añadir tanto una imagen nueva como una ya existente de cualquier formato: raw, qcow2, ISO, etc. Para ello seguimos los siguientes pasos:

1. Para una disco virtual basado en un fichero se recomienda reservar previamente el espacio. Se puede crear un fichero a través del comando *dd* así (imagen de 4GB):

```
root@ayla:~# dd if=/dev/zero of=/var/lib/libvirt/images/
    imagen.img bs=1M count=4096
```

Añadiendo almacenamiento a una VM (2)

Añadiendo almacenamiento basado en fichero

- Pero también se puede crear (en ext3, ext4, XFS, ...) un "sparse file". Un "sparse file" es un fichero de un tamaño específico pero del que no se reserva realmente el espacio. Perfecto para pruebas ya que se crean muy rápidos, pero pueden ocasionar problemas de rendimiento y de integridad de datos. Se crean también con el comando dd:

```
root@ayla:~# dd if=/dev/zero of=/var/lib/libvirt/images/
    imagen.img bs=1M seek=4096 count=0
```

2. Editamos un fichero XML donde se especifique el nuevo almacenamiento:

```
<disk type='file' device='disk'>
    <driver name='qemu' type='raw' cache='none' io='threads'/>
    <source file='/var/lib/libvirt/images/NewImage.img'/>
    <target dev='vdb' bus='virtio'/>
</disk>
```

Añadiendo almacenamiento a una VM (3)

Añadiendo almacenamiento basado en fichero

3. Iniciamos la VM a la que conectaremos el nuevo almacenamiento:

```
virsh # start Ubuntu01
```

4. Asociamos el nuevo volumen a la máquina virtual:

```
virsh # attach-device Ubuntu01 /tmp/NewStorage.xml
```

5. El nuevo disco no aparece hasta que no se reinicie la máquina virtual, para que la detección sea automática, hay que tener cargado en la VM el módulo *acpiphp*. Podemos hacer este cambio permanente en la VM con el comando:

```
root@ubuntu01:~# echo acpiphp >> /etc/modules
```

Añadiendo almacenamiento a una VM (1)

Añadiendo discos y otros dispositivos de bloques

Podemos añadir un disco físico o cualquier otro dispositivo de bloques, como una memoria USB, siguiendo los siguientes pasos:

1. Conectar físicamente el dispositivo al anfitrión y esperar que esté accesible.
2. Utilizar el comando *virsh-attach* como en el siguiente ejemplo:

```
virsh # attach-disk Ubuntu01 /dev/sdc vdb --driver qemu
```

3. Podemos desconectar el dispositivo una vez utilizado a través de:

```
virsh # detach-disk Ubuntu01 vdb
```

Formatos de Imágenes (1)

Al crear una máquina virtual debemos crear uno o varios discos virtuales, existen muchos tipos de formatos de imagen de disco, destacan:

- **raw** Formato de imagen de disco en crudo (no estructurado). Es el formato básico que puede crearse con herramientas como *dd*. No está optimizado para su utilización en virtualización, pero puede manejarse con herramientas básicas del sistema (dd, parted, fdisk, mount, kpartx, etc.). Muy portable entre hipervisores.
- **vhd** Usado por herramientas de virtualización como VMware, Xen, VirtualBox y otros.
- **vmdk** Formato utilizado por VMware en productos como VMware Workstation.

Formatos de Imágenes (2)

- **vdi** Formato soportado por VirtualBox.
- **iso** Volcado del sistema de ficheros contenido en un CD/DVD.
- **qcow2** Formato de imagen soportado por QEMU con características avanzadas como expansión dinámica, copy-on-write, snapshots, cifrado, compresión, etc.
- **vpc** Formato de imagen de Virtual PC.
- **ami** Formato Amazon Machine Image.

Los formatos soportados por QEMU/KVM son: raw, cow, qcow, qcow2, vmdk y vdi.
El más potente de todos: **qcow2**.

Gestión de imágenes de disco (1)

qemu-img

Para la gestión de imágenes virtuales de disco se puede utilizar el comando *qemu-img* (paquete qemu-utils). Este comando permite:

- Chequear una imagen de disco.
- Realizar conversiones entre formatos de imagen.
- Proporcionar información de la imagen.
- Gestionar las snapshots contenidas en una imagen.
- Redimensionar una imagen.

Los formatos soportados son: raw, qcow2, qcow, cow, vdi, vmdk, vpc y cloop.
Al crear una imagen con virt-install podemos especificar el formato a través de la opción *format* del parámetro disk: Ejemplo:

- --disk pool=default,size=4,format=qcow2

Virtual Networking (1)

Redes entre las máquinas virtuales

- Para la implementación de redes virtuales *libvirt* utiliza el concepto de *switches virtuales*.
- Un switch virtual es una implementación software de un switch al que se pueden conectar máquinas virtuales. El tráfico de red de una máquina virtual se redirige a través de este switch.
- Un anfitrión Linux representa un switch virtual como una interfaz de red.
- Al instalar y configurar libvirt, la red virtual por defecto se denomina *default*.
- *default* es un switch virtual representado por la interfaz *virbr0*:

Virtual Networking (2)

Redes entre las máquinas virtuales

```
root@ayla:~# ifconfig virbr0
virbr0    Link encap:Ethernet  direcciónHW fe:54:00:15:11:e7
          Direc. inet:192.168.122.1  Difus.:192.168.122.255
               Másc:255.255.255.0
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  M
              étrica:1
          Paquetes RX:11063 errores:0 perdidos:0 overruns:0
              frame:0
          Paquetes TX:21110 errores:0 perdidos:0 overruns:0
              carrier:0
          colisiones:0 long.colaTX:0
          Bytes RX:626015 (626.0 KB)  TX bytes:30959924
              (30.9 MB)
```

Virtual Networking (3)

Redes entre las máquinas virtuales

```
root@ayla:~# ip addr show virbr0
4: virbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
    noqueue state UP
    link/ether fe:54:00:15:11:e7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global
        virbr0
```

Virtual Networking (1)

Switches virtuales

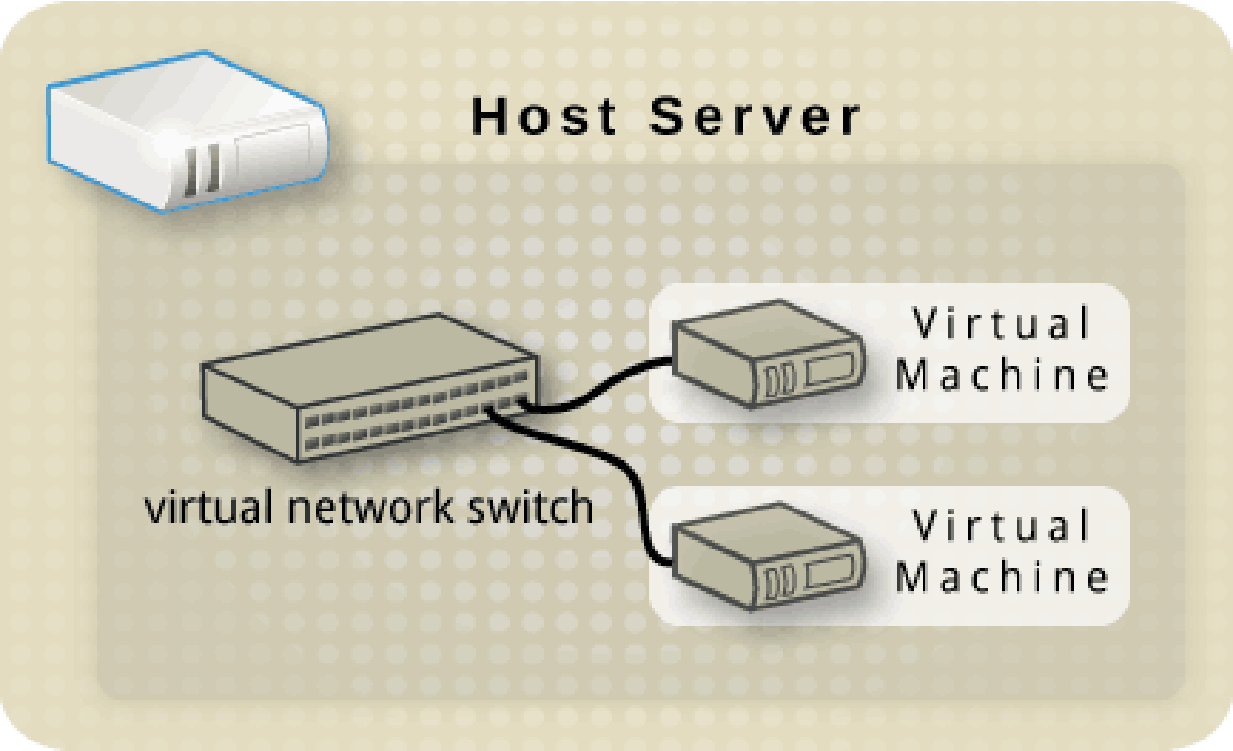


Figura: Copyright ©2012 Red Hat, Inc.

Redes virtuales (1)

NAT

- Por defecto, los switches virtuales operan en modo NAT.
  - Concretamente MASQUERADING.
- IP masquerading permite a las máquinas virtuales conectarse hacia el exterior utilizando la dirección IP del anfitrión.
- Por defecto, las máquinas del exterior no pueden conectarse a las máquinas virtuales de un switch que opera en modo NAT.

Redes Virtuales (1)

NAT

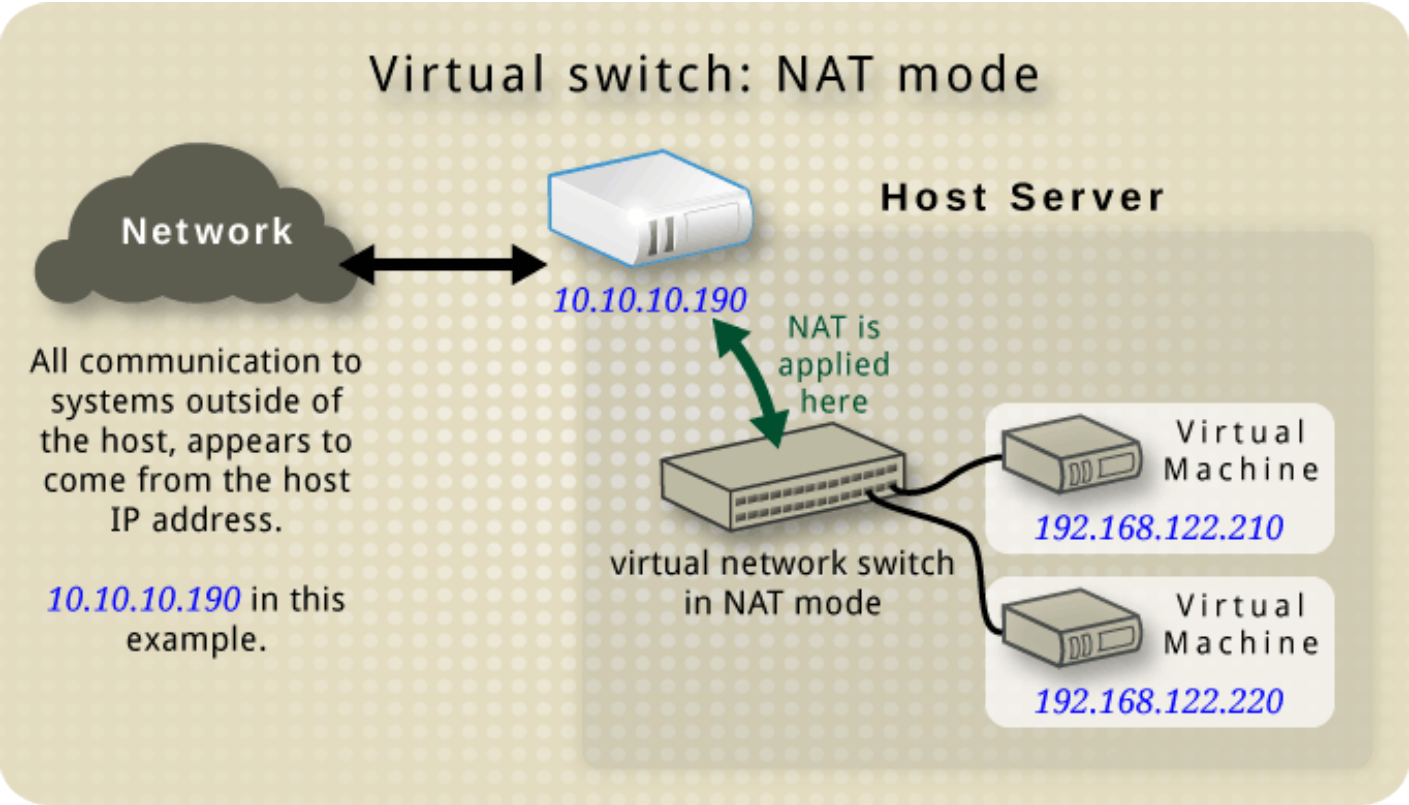


Figura: Copyright ©2012 Red Hat, Inc.

Redes Virtuales (1)

DNS y DHCP

- La configuuración TCP/IP se puede asignar a las VMs a través de DHCP.
- Basta con asignar un pool de direcciones al switch virtual.
- *libvirt* utiliza el programa *dnsmasq* para este propósito.
  - libvirt inicia y configura de forma automática una instancia de dnsmasq por cada switch virtual configurado.

Redes Virtuales (1)

DNS y DHCP

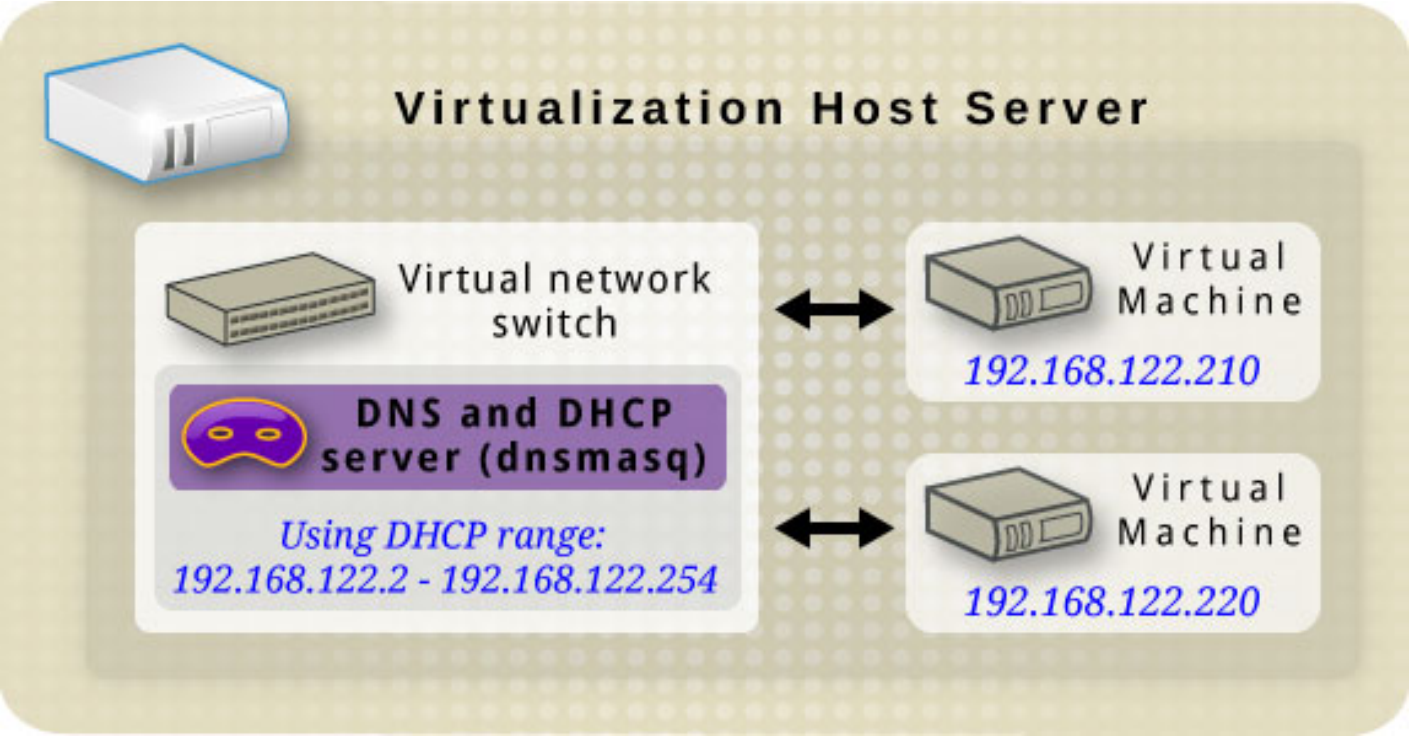


Figura: Copyright ©2012 Red Hat, Inc.

Redes Virtuales (1)

Otros modos

Además de en modo NAT, los switches virtuales pueden trabajar en los siguientes:

- **Routed mode**.
  - En este modo el switch virtual se conecta a la LAN física del anfitrión, pasando todo el tráfico entre anfitrión y VMs sin el uso de NAT.
  - El switch virtual puede examinar el tráfico para tomar decisiones de enrutamiento.
  - Todas las máquinas virtuales se encuentran en su propia subred, enrutadas a través del switch virtual.
  - El switch virtual opera en nivel 3.
  - Desde fuera solo se puede acceder a las máquinas virtuales si se definen reglas de enrutamiento.
- **Isolated mode**.

Redes Virtuales (2)

Otros modos

- En este modo todas las máquinas virtuales conectadas al switch virtual pueden comunicarse solamente entre ellas y con el anfitrión.
- Las VMs no reciben tráfico desde fuera ni pueden conectarse.
- El uso de *dnsmasq* permite funcionalidad básica como DHCP y DNS.
- La resolución de nombres DNS sigue funcionando.
  - Situación curiosa: se resuelven nombres de máquinas de la red externa, pero no se pueden realizar solicitudes de eco ICMP (pings).

Redes Virtuales (1)

Routed mode

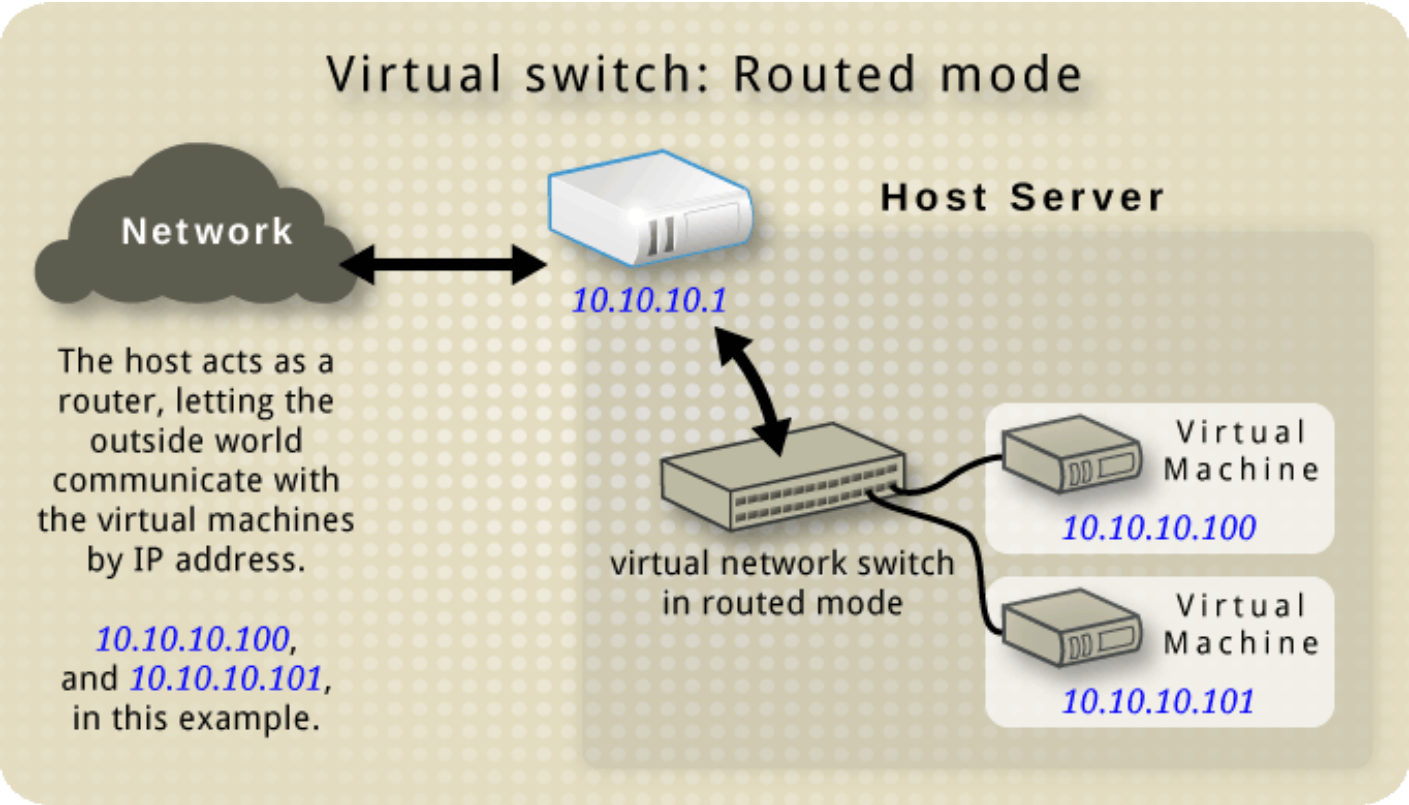


Figura: Copyright ©2012 Red Hat, Inc.

Redes Virtuales (1)

Isolated mode

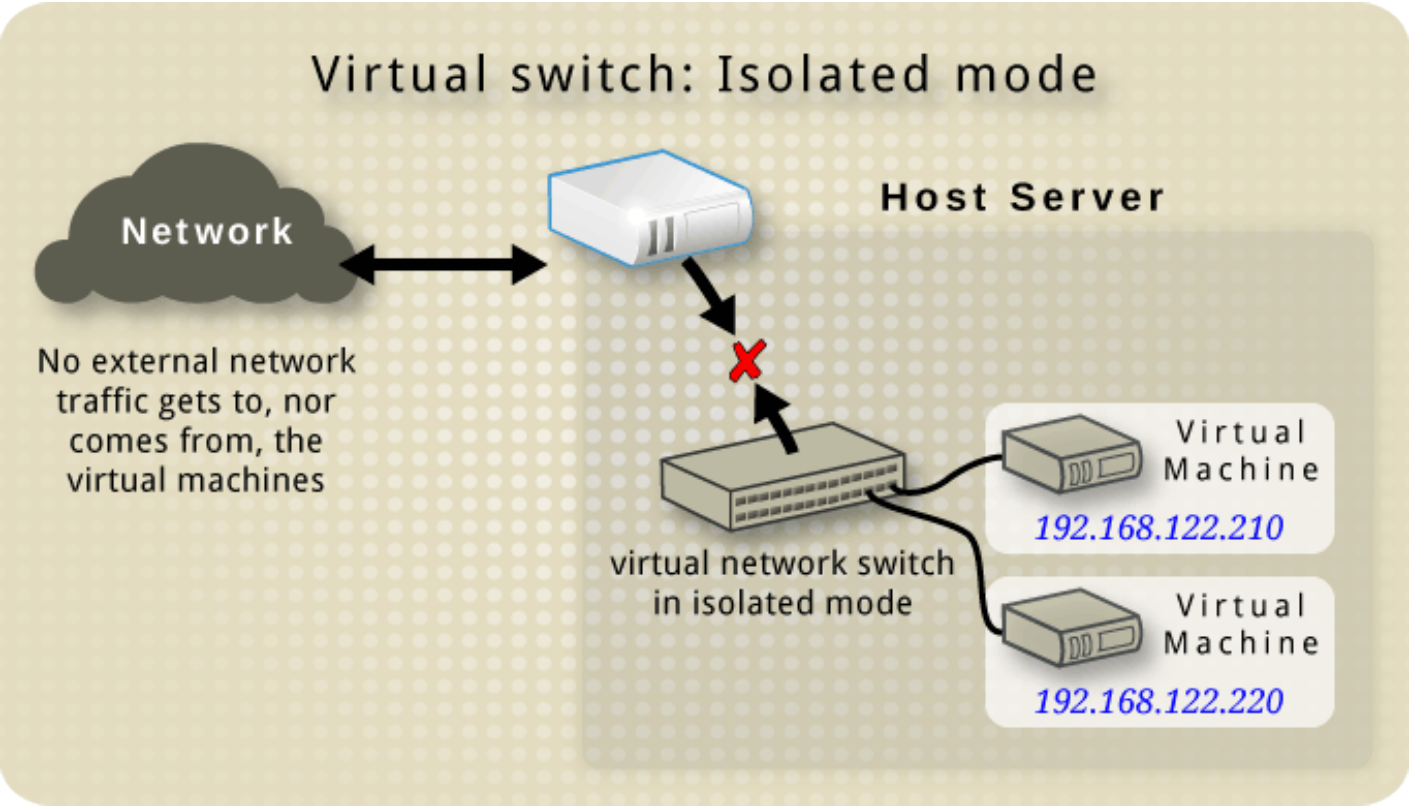


Figura: Copyright ©2012 Red Hat, Inc.

Redes Virtuales (1)

Configuración por defecto

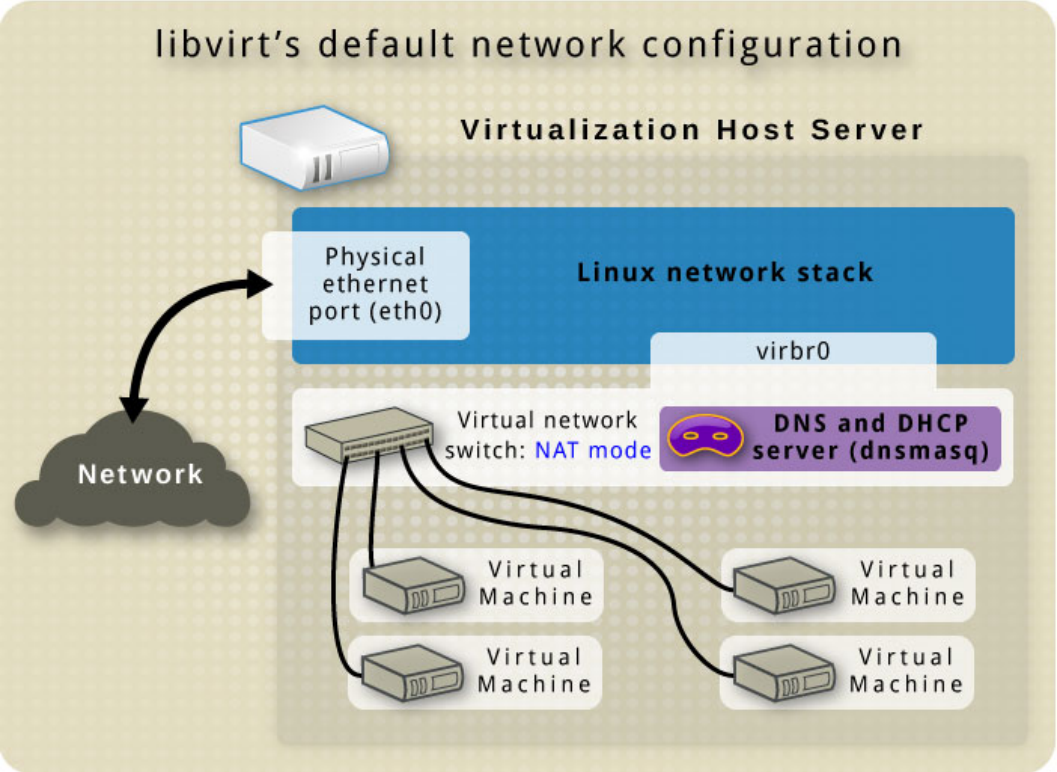


Figura: Copyright ©2012 Red Hat, Inc.

Redes Virtuales (1)

Modo bridge

Es posible que nos interesa conectar todas las máquinas virtuales directamente a la red física del anfitrión.

- La conexión es directa. Habrá que asignar IPs a nuestras VMs o que tomen la dirección por DHCP si en la red donde está conectado el anfitrión hay un servidor DHCP.
- Es necesario modificar la configuración de red del anfitrión.

Para ello hay que seguir los siguientes pasos:

1. Creamos un bridge, *br0*, para la conexión de las máquinas virtuales.
2. Enlazamos la interfaz de red del anfitrión, ¿eth0? al bridge br0.
3. Al crear/configurar las máquinas virtuales usamos el nuevo bridge.

Basta editar el fichero /etc/network/interfaces con este contenido:

Redes Virtuales (2)

Modo bridge

```
auto br0
iface br0 inet dhcp
    bridge_ports    eth0
    bridge_stp      off
    bridge_maxwait  0
    bridge_fd       0
```

Si el anfitrión tiene una IP estática el contenido sería éste:

```
auto br0
iface br0 inet static
    address 192.168.1.100
    netmask 255.255.255.0
    broadcast 192.168.1.255
    network 192.168.1.0
    gateway 192.168.1.254
    dns-nameservers 192.168.1.200 192.168.1.201
    dns-search iescierva.net
```

Redes Virtuales (3)

Modo bridge

```
        bridge_ports    eth0
        bridge_stp      off
        bridge_maxwait  0
        bridge_fd       0
```

Al crear una máquina virtual basta con indicar el bridge a utilizar:

```
virt-install ...
    --network bridge=br0
...
```

También podemos cambiar la configuración de una máquina editando su definición XML y cambiando esto:

Redes Virtuales (4)

Modo bridge

```
<interface type='network'>
  <mac address='52:54:00:75:cc:68'/>
  <source network='default'/>
  <model type='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
```

Por esto:

```
<interface type='bridge'>
  <mac address='52:54:00:75:cc:68'/>
  <source bridge='br0'/>
  <model type='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
```

Redes Virtuales (1)

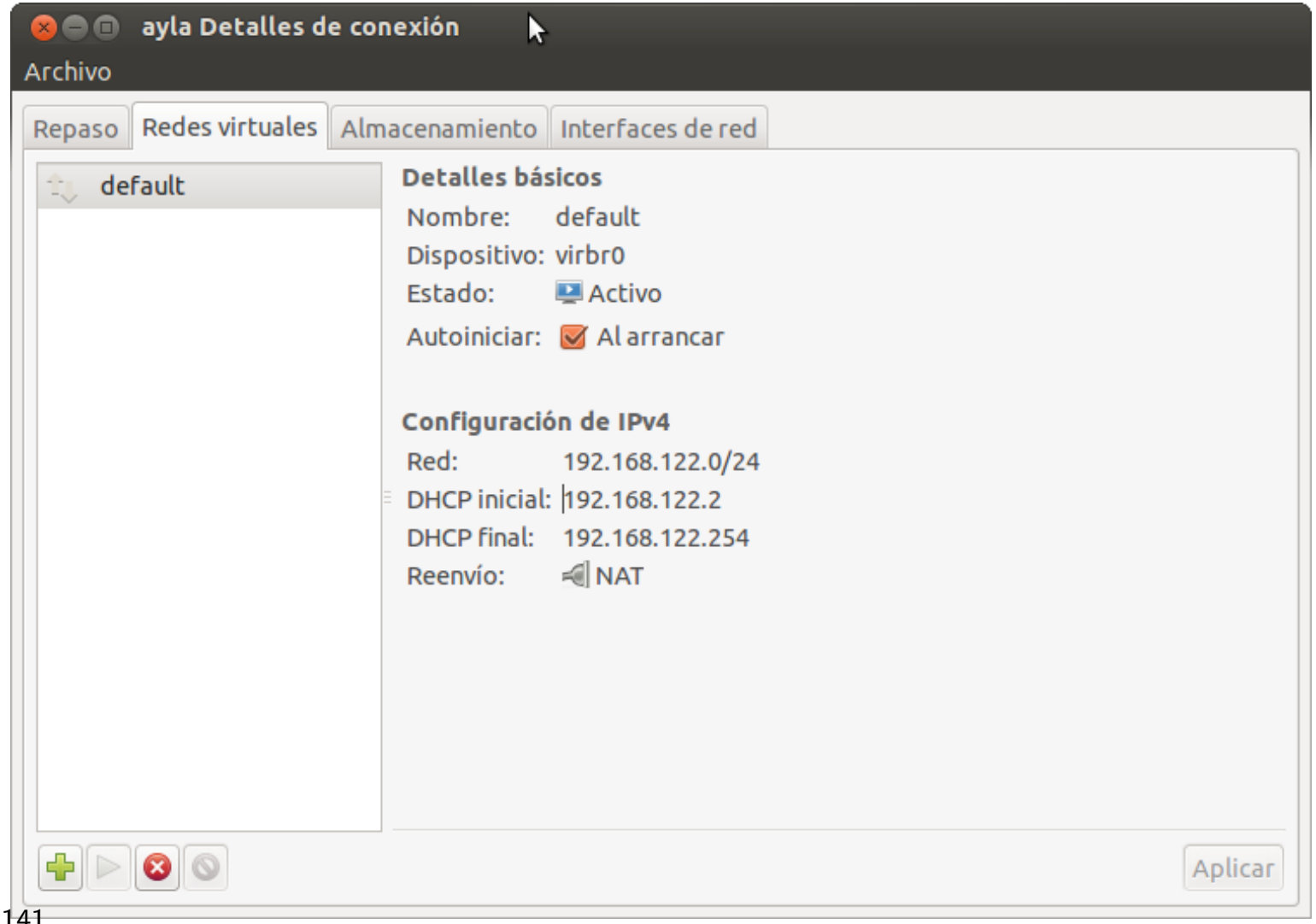
Gestión de Redes Virtuales

Para la gestión de redes virtuales utilizaremos virt-manager:

- Menú Editar -- Detalles de la conexión -- Redes virtuales

Redes Virtuales (2)

Gestión de Redes Virtuales



KSM (1)

Concepto

- El concepto de memoria compartida es muy común en los SSOO modernos:
  - Creación de procesos (copy on write).
  - Compartición de páginas que contienen código ejecutable.
  - Librerías.
  - Memoria compartida por varios procesos para comunicación.
  - Etc.
- KSM es una nueva característica de Linux para compartir memoria.
  - KSM permite que el kernel examine varios procesos y compare sus zonas de memoria.
  - Si el kernel se encuentra páginas idénticas, se fusionan todas en una sola página.

KSM (2)

Concepto

- Esta página se marca como copy on write, si cualquier proceso la modifica, se crea una copia nueva para ese proceso.

KSM (1)

KSM y virtualización

- KSM resulta muy útil cuando se utiliza junto con KVM.
- Cuando se ejecuta una máquina virtual, el contenido de la imagen del SO puede ser compartido por todas las máquinas virtuales que ejecuten el mismo sistema operativo y/o las mismas aplicaciones.
- KSM solo identifica y unifica las páginas que sean idénticas y que no supongan un impacto en la seguridad del anfitrión o de los invitados.
- KSM proporciona mejoras tanto en velocidad como en utilización de memoria:
  - Se produce un aumento del rendimiento al reducirse los fallos de caché, ya que los datos comunes de varios procesos solo se almacenan una vez.

KSM (2)

KSM y virtualización

- Se utiliza muchas menos memoria cuando se lanzan varias máquinas virtuales idénticas.

KSM (1)

KSM y Ubuntu

KSM viene activado y configurado por defecto en Ubuntu. Podemos obtener datos del funcionamiento de KVM a través de los ficheros contenidos en el directorio **/sys/kernel/mm/ksm/**:

- full_scans: Escaneos realizados.
- pages_shared: Total de páginas compartidas.
- pages_sharing: Total de páginas presentes compartidas.
- pages_to_scan: Páginas no escaneadas.
- pages_unshared: Páginas que no se van a seguir compartiendo.
- pages_volatile: número de páginas volátiles.
- run: Booleano que indica si el proceso *KSM* se está ejecutando.
- sleep_millisecs: Milisegundos que dura el tiempo de bloqueo de ksmd.

Herramientas (1)

Herramientas para el acceso offline a discos virtuales

Instalando los siguientes paquetes:

- guestfish
- libguestfs-tools
- guestmount

... se tiene acceso a un conjunto de herramientas que hace posible el acceso tanto en lectura como escritura de imágenes y de discos virtuales.

Se pueden utilizar estas herramientas con diferentes fines:

- Ver y obtener ficheros de los discos virtuales.
- Editar y crear ficheros en los discos virtuales.
- Revisar y modificar configuraciones de las máquinas virtuales.

Herramientas (2)

Herramientas para el acceso offline a discos virtuales

- Leer y escribir el registro en las máquinas virtuales Windows.
- Preparar imágenes de nuevos discos.
- Rescatar y reparar máquinas virtuales que no arranquen correctamente o necesiten cambios en sus configuraciones.
- Monitorizar el uso en disco de las máquinas virtuales.
- Auditorías de seguridad.
- Desplegar invitados clonando y modificando plantillas.
- Leer imágenes ISO de CDs/DVDs.

Herramientas (3)

Herramientas para el acceso offline a discos virtuales

Importante

Nunca utilizar estas herramientas para modificar un disco virtual que está siendo utilizado por una máquina virtual en ejecución. Aunque las herramientas siempre avisan de esta situación, es posible que no lo hagan en determinados casos.

Herramientas (1)

Terminología

Varios términos importantes relacionados con estas herramientas:

- **libguestfs** (GUEST FileSystem LIBrary): es la librería que proporciona la funcionalidad básica para abrir y leer/escribir ficheros en los discos virtuales.Está escrita en C y proporciona una API de bajo nivel para la escritura de nuevos programas.
- **guestfish** (GUEST Filesystem Interactive SHell): es la shell interactiva que expone toda la funcionalidad de *libguestfs*. También puede utilizarse desde scripts.
- Varias herramientas virt, como **virt-df**, **virt-rescue**, **virt-resize**, y **virt-edit**, están construidas sobre *libguestfs* y proporcionan una forma de realizar tareas específicas desde la línea de comandos.

Herramientas (2)

Terminología

- Las librerías **hivex** y **Augeas** permiten la edición del registro de Windows y la edición de ficheros de configuración Linux respectivamente. Aunque son independientes de *libguestfs*, ambas le proporcionan un valor añadido.
- **guestmount**: es una interfaz entre *libguestfs* y *FUSE*. Su principal uso es montar sistemas de ficheros de imágenes de disco. Comando innecesario si tenemos el resto de herramientas pero de una gran utilidad.

Herramientas (1)

guestfish

guestfish es una shell interactiva que se puede utilizar tanto desde la línea de comandos como desde scripts para acceder a los sistemas de ficheros de las máquinas virtuales.
Ejemplos de uso:

- Comenzar a ver/editar un sistema de ficheros:

```
root@ayla:~# guestfish --ro -a /var/lib/libvirt/images/
    WindowsServer2008R2.img
```

```
root@ayla:~# guestfish --ro -d WindowsServer01
```

- Listar todos los sistemas de ficheros que contiene una imagen:

Herramientas (2)

guestfish

```
><fs> run
><fs> list-filesystems
/dev/vda1: ext4
/dev/vda2: unknown
/dev/vda5: swap
><fs>
```

- Otros comandos interesantes son: **list-devices**, **list-partitions**, **lvs**, **pvs**, **vfs-type**, **file**.
- Inspeccionar los ficheros:

Herramientas (3)

guestfish

```
><fs> run
><fs> list-filesystems
/dev/vda1: ext4
/dev/vda2: unknown
/dev/vda5: swap
><fs> mount /dev/vda1 /
><fs> ll /
total 104
drwxr-xr-x 23 0 0  4096 Oct 15 19:34 .
drwxr-xr-x 21 0 0  4096 Oct 28 17:27 ..
drwxr-xr-x  2 0 0  4096 Oct 18 16:47 bin
drwxr-xr-x  3 0 0  4096 Oct 15 19:35 boot
drwxr-xr-x  3 0 0  4096 Oct 15 18:11 dev
drwxr-xr-x 86 0 0  4096 Oct 28 16:47 etc
drwxr-xr-x  3 0 0  4096 Oct 15 19:05 home
...
drwxrwxrwt  2 0 0  4096 Oct 27 09:22 tmp
drwxr-xr-x 10 0 0  4096 Oct 15 18:08 usr
```

Herramientas (4)

guestfish

```
drwxr-xr-x 12 0 0  4096 Oct 28 16:47 var

><fs>
```

Se pueden utilizar también los comandos **ls**, **ll**, **cat**, **more**, **download**, **tar-out**, ...

- También se pueden modificar ficheros a través de los comandos **edit**, **vi**, **emacs**, **write**, **mkdir**, **upload**, **tar-in**, ...
- También se permite formatear, crear particiones, getionar volúmenes lógicos, ect. A través de comandos como: **mkfs**, **part-add**, **lvresize**, **lvcreate**, **vgcreate**, **pvcreate**, ...

Herramientas (1)

Otros comandos

Existen comandos que simplifican mucho ciertas tareas:

- **virt-cat**: muestra el contenido de un fichero.

```
root@ayla:~# virt-cat Ubuntu01 /etc/issue
```

- **virt-edit**: permite editar un fichero del disco virtual.

```
root@ayla:~# virt-edit Ubuntu01 /etc/issue
```

- **virt-edit**: a través de la opción -e permite realizar cambios sobre ficheros de forma no interactiva.
- **virt-ls**: permite mostrar el contenido de un directorio (incluso de forma recursiva).

Herramientas (2)

Otros comandos

```
root@ayla:~# virt-ls Ubuntu01 /home/alex
```

- **virt-df**: similar al comando *df*, muestra el uso de los sistemas de ficheros en una imagen de disco virtual.

```
root@ayla:~# virt-df Ubuntu01
```

Herramientas (1)

Más herramientas

Entre otras herramientas tenemos también a nuestra disposición:

- **virt-rescue**: herramienta interactiva que inicia un disco virtual en modo de recuperación de tal forma que se puedan realizar tareas de mantenimiento y recuperación/rescate.
- **virt-resize**: permite expandir y reducir imágenes de disco.
- **virt-inspector**: permite inspeccionar una imagen generando un informe en XML con los sistemas operativos que contiene la imagen.
- **virt-win-reg**: herramienta que permite manipular el registro de Windows de las imágenes de máquinas virtuales Windows.

Bibliografía (1)

Para saber más...

- Tholeti B. P.
  Hypervisors, virtualization, and the cloud: Dive into the KVM hypervisor.
  IBM DeveloperWorks. September 2011.
  http://www.ibm.com/developerworks/cloud/library/cl-hypervisorcompare-kvm/index.html
- Kusnetzky, D.
  Virtualization: A Manager's Guide.
  Ed. O'Reilly, 1ª ed. 2011.

Bibliografía (2)

Para saber más...

Chen, G., Gillen, A.
White Paper. KVM for Server Virtualization: An Open Source Solution Comes of Age.
IBM. October 2011.
http://www-03.ibm.com/systems/resources/systems_virtualization_idc__kvmforservervirtualization.pdf

KVM - Community Ubuntu Documentation.
https://help.ubuntu.com/community/KVM

Linux KVM.
http://www.linux-kvm.org/page/Main_Page

Bibliografía (3)

Para saber más...

Red Hat Enterprise Linux 6. Virtualization Administration Guide.

https://access.redhat.com/knowledge/docs/Red_Hat_Enterprise_Linux/?locale=en-US

Wikipedia: Hardware Virtualization.

http://en.wikipedia.org/wiki/Hardware_virtualization

Wikipedia: Virtualization.

http://en.wikipedia.org/wiki/Virtualization

Xen.

http://www.xen.org/