INTRODUCIÓN Á VIRTUALIZACIÓN

Seguridade e Alta Dispoñibilidade

INTRODUCIÓN Á VIRTUALIZACIÓN

Introdución

As CPU actuais son moi potentes, e a proliferación dos centros de procesamento de datos (***CPD***) causa que moitos equipos potentes estean prendidos as 24 horas do día desperdiciando recursos que o sistema en execución non necesita.

A ***virtualización*** permite aproveitar ó máximo eses recursos dividindo os recursos físicos en múltiples recursos virtuais dentro da mesma máquina. Deste xeito, en lugar de instalar unha máquina física por sistema operativo ou cliente, se poden compartir entre varios clientes e levar a cabo varios sistemas operativos de xeito simultáneo.

> A *virtualización* consiste na creación a través de software dunha versión virtual de algún recurso tecnolóxico, como pode ser unha plataforma hardware, un dispositivo de almacenamento ou outros recursos de rede.

En xeral é posible virtualizar calquera recurso, pero aquí nos referiremos á ***virtualización de plataformas,*** na que un equipo anfitrión (host) executa un programa (chamado Virtual Machine Monitor - VMM - ou hypervisor) que simula un entorno computacional completo para executar un software (***guest***). Este software funciona como si estivera instalado nunha plataforma real limitada ós recursos ofrecidos polo *hypervisor.*

O VMM (hypervisor) é un software que se encarga de proporcionar acceso ao hardware real aos sistemas virtualizados emulando por software hardware estándar. Podemos distinguir dous tipos de hypervisores:

- **Tipo I (bare metal) -** O hypervisor é o sistema operativo que está funcionando na máquina anfitrión. Iso implica que as chamadas do sistema virtualizado chegan directamente dende o hypervisor ao hardware real. Exemplos de hypervisores de este tipo son VMWARE ESXi, Xen, KVM ou Hyper-V.
- **Tipo II -** O hypervisor é unha aplicación máis funcionando encima do sistema operativo do equipo anfitrión. Iso implica que as chamadas do sistema virtualizado pasan do hypervisor ao sistema operativo da máquina, e de ahí ao hardware. Exemplos de este tipo son VMWARE Player ou VirtualBox.

Dentro da virtualización de plataformas podemos distinguir entre:

- Virtualización Hardware Completa
- Paravirtualización
- Virtualización a nivel de Sistema Operativo

Virtualización Hardware Completa

Cando se utiliza virtualización hardware completa, o hypervisor "simula" o hardware sobre o que se vai a executar o sistema, tanto os recursos de computación (CPU, xestión a memoria...) como dos distintos periféricos do sistema (disco, pantalla). Esta virtualización é moi costosa en termos de recursos e non era viable en produción debido a perda de velocidade que supoñía. Os hypervisores utilizaban técnicas de ***paravirtualización*** que permitían executar os sistemas invitados con maior velocidade.

As CPU modernas (a partir do ano 2005) posúen unha serie de extensións de virtualización (intel VT, amd-V) que proporcionan ***virtualización asistida por hardware.*** Estas extensións permiten certo grao de acceso por parte dos sistemas virtualizados aos recursos da CPU o que acelera enormemente o seu rendemento sen necesidade de recurrir a paravirtualización

Paravirtualización

Un dos sistemas que se buscou para acelerar a execución das máquinas virtuais foi a ***paravirtualización.*** Este sistema consiste en que en lugar de simular un hardware existente, o hypervisor facilita ao sistema virtualizado unha serie de chamadas que se executan a moita maior velocidade. Este sistema o empregou Xen con moito éxito, pero necesita a modificación do sistema operativo virtualizado para que realice as chamadas apropiadas ao hypervisor, e iso non era posible cos sistemas propietarios como Windows. Linux en cambio, foi executado con moito éxito en modo paravirtualizado.

Coas novas extensións de virtualización das CPU modernas a paravirtualización a este nivel quedou en desuso, qedándose limitada ao uso ***drivers paravirtualizados***. Estes drivers facilitan acceso a "dispositivos" como controladores de disco, de memoria ou de rede que en lugar de ser simulados por software, corresponden con accesos o hypervisor, que xestionará o acceso ao hardware real.

Hoxe en día, os sistemas virtualizados empregan maiormente as extensións de virtualización da CPU e un conxunto de drivers paravirtualizados, utilizando o que se chama ***virtualización híbrida***. Habitualmente os drivers paravirtualizados se instalan no sistema virtualizado mediante o que se coñece como "guest-additions".

Virtualización a nivel de Sistema Operativo

Outro tipo de virtualización é a virtualización a nivel de sistema operativo ou ***Containers***. Mentres que na virtualización hardware se emula por software unha CPU e un hardware sobre o que podemos instalar un sistema operativo, a virtualización a nivel de sistema operativo consiste en aillar recursos do sistema operativo de xeito que se crea un "contedor" no que se aillan os procesos dos demais. A todos os efectos, un proceso dentro dun contedor "funciona nunha máquina propia" xa que non "ve" o resto de procesos do sistema.

En Linux existen varias tecnoloxías, pero todas elas se apoian nunha característica do kernel denominada ***cgroups***. Mediante cgroups é posible agrupar e limitar recursos de xeito que dentro dun ***cgroup*** únicamente se ten ***acceso*** aos recursos do grupo restrinxidos polos límites impostos. Deste xeito, si configuramos un ***cgroup*** e facemos funcionar dentro de él servizos propios dun sistema teremos a todos os efectos unha "***máquina virtual***".

As vantaxes dos containers sobre as máquinas virtuais tradicionais son varias:

- Maior velocidade de arranque e parada: Non é necesario iniciar un sistema, simplemente configurar o cgroup correspondente e iniciar os servizos.
- Maior velocidade de execución: Mentres que na virtualización hardware sufrimos dunha perda de rendemento apreciable debido a necesidade da emulación do hardware, nos Containers a velcidade é "nativa", e decir, funciona a mesma velocidade que o sistema sen virtualizar sen ningunha perda de rendimento.
- Aproveitamento dos recursos: Como non é necesario virtualizar hardware non é preciso un hypervisor aforrando memoria e recursos de CPU.

Por outra banda, as desventaxes tamén son evidentes:

- Non estamos virtualizando. Todos os procesos están funcionando baixo o mesmo kernel (o do sistema), de modo que ***calquera fallo pode afectar o resto de Containers e ao propio sistema***. Na virtualización hardware é moito máis difíicl que esto pase, xa que consiste en distintos sistemas funcionando en entornos virtuais.
- O aillamento entre os sistemas virtuais é moito menor, polo que **potencialmente é máis fácil o acceso dende un Container a información de outro,** o que na virtualización hardware é moito máis complexo.
- **Non é posible executar sistemas operativos que requiran un kernel distinto o que está funcionando no sistema.**

Hoxe en día os Containers se empregan con moita frecuencia, sobre todo as seguintes tecnoloxías:

LXC

LX**C (Linux Containers)** e a tecnoloxía de Containers hoxe en día estándar en linux, aínda que existen outros sistemas similares con máis tempo de uso nos contornos profesionais como Virtuozzo ou OpenVZ. **LXC permite a execución dun contorno operativo de xeito aillado** proporcionando a todos os efectos unha "máquina virtual" baseada en Containers.

Dockers

Dockers é unha tecnoloxía moi en boga hoxe en día. A diferencia de LXC, Docker non está orientado a execución de contornos operativos, se non a execución de aplicacións e servizos. Deste xeito, podemos despregar un Docker que nos ofreza un servizo Web, un servizo de e-Mail, un contorno WordPress ... etc. Metres que LXC nos proporciona un sistema de uso xeral similar a unha máquina virtual na que instalaremos os servicios desexados e executaremos aplicacións, Docker proporciona un xeito rápido de desrpregue de servizos concretos, facilitando a creación de sistemas orientados a microservizos.

Dockers dispón de repositorios de servizos en internet dende o que se pode despregar un servizo con un simple comando e un mínimo de configuración para xestionar a conectividade de rede dende e hacia o servizo despregado e o seu acceso ao almacenamento externo, xa que un “Docker” por defecto funciona completamente aillado do resto do sistema.

Docker é multiplataforma, aínda que o seu uso principal é baixo Linux, é posible despregar contedores baseados en Windows e incluso de Dockers Linux baixo Windows facendo uso de Hyper-V e WSL.

Ademáis do despregue rápido de microservizos, Docker permite a execución de aplicacións de distintas versións no sistema, sen depender de ningunha libraría nin software externo e sin interferir entre elas.

Kubernetes

Kubernetes é un sistema de xestión para Dockers, que permite automatizar a administración dos servizos que se ofrecen mediante dockers, organizando e automatizando a implementación e o escalado. Mediante Kubernetes se poden configurar sistemas de despregue de servizos automatizados baixo demanda, co obxecto de balancear a carga e facilitar a dispoñibilidade. E un “orquestador” de contedores.

https://en.wikipedia.org/wiki/X86_virtualizationá
https://en.wikipedia.org/wiki/Hardware-assisted_virtualization
https://en.wikipedia.org/wiki/Paravirtualization

RECURSOS DE VIRTUALIZACIÓN

Un sistema virtual utiliza os seguintes recursos:

- **Virtualización de Procesamento**: A proporciona o Hypervisor simulando o hardware
- **Virtualización de Almacenamento**: A máquina virtual necesita acceder a algún sistema de almacenamento físico, que debe ser proporcionado de xeito externo.
- **Virtualización da conectividade**: Para que a máquina virtual se comunique co exterior tamén se precisa que se lle proporcionen ao hypervisor recursos de conectividade

Os sistemas virtualizados a miúdo inclúen sistemas complexos para proporcionar estes recursos como "pools de almacenamento" ou redes virtuais.

Virtualización do Almacenamento

Unha máquina virtual almacena a súa información nun recurso de almacenamento que se lle proporciona ao hypervisor de xeito que a VM o ve como un disco do sistema virtualizado. Ese recurso pode ser **un dispositivo de bloques** real (como unha partición ou un disco real) ou **un ficheiro** que a máquina virtual ve como un dispositivo de bloques.

A localización do almacenamento tamén é importante. Cando creamos unha máquina virtual en realidade non nos preocupa o Hypervisor da rede que está simulando a máquina. Ese hypervisor pode ser calquera dos dispoñibles na rede, e estar situado en calquera ordenador físico. Dende o punto de vista da máquina virtual é irrelevante: E unha máquina máis conectada a rede.
O único requisito é que a conectividade sexa a mesma dende todos os hypervisores e que **o almacenamento da máquina virtual sexa tamén o mesmo**.

Mentres que é relativamente simple proporcionar a mesma conectividade en todos os hypervisores mediante o uso de *bridges* ou *switches* virtuais, o asegurar o acceso aos mesmos datos dende calquera hypervisor da rede implica necesariamente un almacenamento compartido. Si pretendemos facilitar dispositivos de bloques para o almacenamento ás máquinas virtuais, necesitaremos un sistema de compartición de dispositivos (como FiberChannel ou **iSCSI**), mentres que si o que queremos é facilitar ficheiros a elección pasa polo uso de **NFS.**

> E posible crear "pools" de almacenamento do que as máquinas collan o seu almacenamento de xeito que ese pool se atope distribuído en diversas máquinas da rede mediante sistemas distribuídos como Ceph, GFS2 (RedHat) ou OCFS2 (Oracle), proporcionando tolerancia a fallos e balanceos de carga

Almacenamento en Ficheiros

E posible proporcionar ás máquinas virtuais almacenamento utilizando ficheiros en disco. O método máis simple é o uso dun ficheiro directamente para almacenar información como si fora un dispositivo de bloques (formato **raw**), pero os distintos hypervisores soportan formatos de ficheiros máis elaborados que facilitan características especiais como:

- **Fácil redimensionamento -** E posible variar o tamaño do ficheiro e adaptar o sistema de forma apropiada
- **Snapshots -** E posible facer copias "on write" dos ficheiros. O ficheiro copiado inicialmente non ocupará nada e se realizará de xeito instantáneo. Os contidos reais se leen do ficheiro orixinal, e se irán escribindo únicamente os cambios. Os snapshots son moi útiles para o despregue rápido de máquinas virtuais e para facer probas que poden desfacerse fácilmente.
- **CoW (Copy on Write) -** O ficheiro inicialmente non ocupa, se non que o seu tamaño é virtual. O espacio se irá ocupando a medida que se escribe nel.

Formatos típicos con soporte destas características son **vdi** (VirtualBox), **vmdk** (VMWare), **vhd** (Xen, HyperV) e **qcow2** (KVM).

qemu-img

qemu-img é a utilidade que se utiliza co hypervisor KVM para xestionar os ficheiros de disco para as máquinas virtuais. É capaz de converter e xestionar ficheiros de *varios formatos*, aínda que o hypervisor KMVM so traballa cos formatos ***qcow2*** ou ***raw***.

Entre outras cousas pode:

a) **Crear novos ficheiros:**

// qemu-img create -f *formato NomeFicheiro size*G
qemu-img create -f qcow2 systemdisk.qcow2 100G

b) **Converter entre formatos**

// qemu-img convert -f *formato_orixe* -O formato_ficheiro_orixe ficheiro_destino
qemu-img convert -f qcow2 -O raw systemdisk.qcow2 systemrawdisk.raw

c) **Cambio do tamaño do ficheiro**

- ***A maior tamaño***

// qemu-img resize ficheiro_de_disco +sizeG
qemu-img resize systemdisk.qcow2 +20G

O cambio a maior tamaño require a **POSTERIOR** creación ou cambio do taamño das particións ou do sistema de arquivos dentro da máquina virtual que faga o uso do disco

- **A menor tamaño**

// qemu-img resize ficheiro_de_disco *size*G
qemu-img resize systemdisk.qcow2 80G

O cambio a menor tamaño require o cambio do tamaño **PREVIO** dos sistemas de ficheiros e das particións dentro da máquina virtual que faga uso do disco

d) **Crear un snapshot**

// qemu-img create -f qcou2 -b ficheiro_de_disco_base ficheiro_a_crear
qemu-img create -f qcow2 -b systemdisk.qcow2 snapdisk.qcow2

O formato do disco base pode ser raw ou qcow2, o snapshot ten que ser qcow2
A conversión dun snapshot nun disco independente se pode realizar mediante unha conversión:
qemu-img -f qcow2 -O qcow2 snapdisk.qcow2 disk.qcow2

e) **Mesturar os cambios dun snapshot na imaxe base**

// qemu-img commit ficheiro_de_disco_snapshot
qemu-img commit snapdisk.qcow2

f) **Obter información dun ficheiro de disco**

// qemu-img info ficheiro_de_disco
qemu-img info systemdisk.qcow2

g) **Cambiar a base dun snapshot**

// qemu-img rebase -b new_base_file ficheiro_snapshot
qemu-img rebase -b neworixe.qcow2 snapdisk.qcow2

Snapshots

Os snapshots de discos virtuais teñen dous obxectivos posibles: A rápida creación de máquinas virtuais e ter a posibilidade de reverter o estado da máquina a unha situación anterior.

- Para crear rápidamente unha máquina virtual basta con ter varios discos "base" cos sistemas instalados. Estas imaxes poden ser so lectura para evitar a súa modificación. Creando un snapshot, dispoñemos de xeito automático dun novo disco cun sistema idéntico o do sistema base. Alguns entornos de xestión de VM permiten facelo de xeito "automático"

- Si creamos un snapshot dun disco que está utilizando unha máquina virtual, e substituímos o disco orixinal polo snapshot, poderemos elixir posteriormente si descartar os cambios feitos dende a creación do snapshot (eliminando o snapshot) ou conservalos (facendo "commit" do snapshot a imaxe base). Esto se pode realizar dende o entorno de xestión das VM.

Almacenamento en dispositivos
Unha posibilidade que ofrece maior rendemento que o uso de ficheiros como discos é o uso directo de discos ou particións como almacenamento para as máquinas virtuais. A cambio de un rendemento algo maior, sacrificamos as características ofrecidas polo almacenamento en disco (redimensionamento, snapshots, copy-on-write....).

Unha solución excelente e amplamente utilizada é o uso dun xestor de volumes de disco. Os xestores de volumes permiten aglutinar o espazo de almacenamento de varios dispositivos e repartilo en volumes do tamaño desexado, otorgando unha gran flexibilidade en canto a xestión de espazo, xa que sempre podemos engadir ao xestor máis discos físicos. Ademais comunmente ofrecen a posibilidade de realizar stripping (RAID 0), raid, snapshotting, redimensionamento, thin-provisioning... etc. O sistema xestor de volumes de disco en Linux é **LVM2.** LVM2 nos ofrece todas estas características avanzadas que nos permiten unha flexibilidade enorme na xestión do almacenamento para máquinas virtuais.

Almacenamento Remoto
Si o que queremos é almacenar os discos de xeito centralizado e accesible por todos os hypervisores de xeito que sexa posible a migración "en quente" (sen apagar nin parar) das máquinas virtuais entre os hosts da rede dispoñemos de dous métodos principais:

- *Si o almacenamento emprega ficheiros, **NFS.***
  NFS é un sistema para compartir arquivos e directorios na rede de xeito moi rápido e eficiente. Está soportado por todos os UNIX e por Windows Server e Windows 10 Enterprise.

- *Si o almacenamento emprega dispositivos (particións ou volumes LVM), FiberChannel ou **iSCSI**.*
  FiberChannel permite compartir dispositivos a través de redes de fibra óptica. Se utilizan en SAN dentro de centros de procesamento de datos, e o seu elevado prezo fai inviable o seu uso en pequenas instalacións. ***iSCSI*** é unha versión do protocolo iSCSI para redes ethernet, polo que permite compartir dispositivos a través de redes ethernet estándar.

LVM (Logical Volume Mánager)

LVM é o sistema de xestión de volumes de almacenamento propio de Linux. En LVM o espacio mínimo de transferencia de información é o **Extent.** Por defecto **o tamaño do extent é de 4M.** En debian, as utilidades de xestión de LVM veñen no paquete **lvm2 (apt install lvm2).** En **LVM** falamos dos seguintes conceptos:

- **Volume físico** (**PV**): Os volumes físicos son os dispositivos que ofrecen espazo de almacenamento ao xestor de volumes. Poden ser discos completos, particións ou calquera dispositivo de bloques almacenamento). Se xestionan mediante os comandos *pvcreate, pvresize, pvmove*, ...
- **Grupo de volumes** (**GV**): Un grupo de volumes xestiona varios volumes físicos que lle dan a súa capacidade para ofrecer almacenamento. E posible aumentar e disminuír o tamaño do grupo de volumes simplemente retirando ou agregando volumes lóxicos.
- **Volume Lóxico** (**LV**): Un volume lóxico é un espazo de almacenamento que podemos utilizar como un dispositivo de bloques (disco). Os volumes lóxicos se crean a partir do espazo que ofrece un grupo de volumes.
- **Snapshot:** Un snapshot é un volume lóxico que non ocupa máis espazo que as modificacións que se fagan a partir da súa creación. A diferencia dos snapshots de ficheiros, os snapshots LVM permiten a modificación tanto do propio snapshot como do volume lóxico orixina, o que o fai moi apropiado para o seu uso no despregue rápido de discos para máquinas virtuais, e sobre todo para a realización de backups en quente.

> O espazo que se indica cando creamos un snapshot fai referencia unicamente ao volume de cambios que soportan, *si se agota este espazo o snapshot se corrompe.*

- **Thin Pool:** Un thin-pool é un volume lóxico especial, a partir do que podemos crear novos volumes lóxicos. Os volumes lóxicos creados nun thin-pool (Thin Volumes) so ocupan o espazo utilizado, non o espazo máximo. Cando un thin volume borra datos do disco, pasan a estar a disposición do pool a través dunha operación que se chama ***trim*** ou ***fstrim***, que o pode asignar a un thin volume distinto.
- **Thin Volume:** Un thin-volume é un volume lóxico creado dentro dun thin-pool. O seu tamaño é "virtual", porque non ocupa máis cando a información está presente. Deste xeito é posible crear thin volumes moito maiores do espazo realmente existente.

- **RAID:** O grupo de volumes é capaz de organizar o reparto da información entre os seus volumes físicos de xeito que se poden crear volumes lóxicos organizados en RAID 0, RAID 1, RAID 4, RAID 5, RAID 6 ou RAID 10...

No ficheiro de configuración de LVM (en Debian /etc/lvm/lvm.conf) é posible especificar que o xestor asigne automáticamente espazo do grupo de volumes aos snapshots ou thin-pools que o precisen:

snapshot_autoextend_threshold= grao de ocupación máxima (100, desactiva a asignación automática)
snapshot_autoextend_percent= porccentaxe que aumentamos (si está dispoñible o espacio)
thin_pool_autoextend_threshold= grao de ocupación máxima (100, desactiva a asignación automática)
thin_pool_autoextend_percent= porcentaxe que aumentamos (si está dispoñible o espacio)

Os thin pools non se poden reducir de tamaño.

Operacións típicas con LVM

Dados 4 volumes físicos (/dev/sda1, /dev/sdb1, /dev/sdc1 e /dev/sd1) vexamos unha serie de operacións comúns:

- Crear un grupo de volumes "**Datos**" con 2 dispositivos
  - **vgcreate Datos /dev/sda1 /dev/sdb1**
- Aumentar o espazo do grupo de volumes con 1 dispositivo adicional
  - **vgextend Datos /dev/sdc1**
- Substituír un dispositivo por outro
  - Necesitamos liberar o espazo ocupado do volume físico que queremos quitar (insertar antes o novo):
    - **vgextend Datos /dev/sdd1**
    - **pvmove /dev/sda1 /dev/sdd1**
    - **vgreduce Datos /dev/sda1**
- Crear un volume lóxico "**webserver**" de 20G
  - **lvcreate -n webserver -L 20G Datos**
- Aumentar o tamaño do volume lóxico 'webserver' en 20G máis
  - **lvresize -r -L+20G Datos/webserver**
- Disminuir o tamaño do volume lóxico 'webserver' en 20G
  - **lvresize -r -L-20G Datos/webserver**
- Crear un volume lóxico "**database**" en RAID 1
  - **lvcreate --type raid1 -m 1 -n database -L20G Datos**
- Crear un volume lóxico "**templates**" con 3 stripes (varios discos en RAID 0) de 20G
  - **lvcreate -i 3 -n templates -L 20G Datos**
- Crear un snapshot do volume database chamado database-sn de 512M
  - **lvcreate -s -n database-sn Datos/database -L 512M**
- Eliminar o snapshot database-sn
  - **lvremove Datos/database-sn**
- Crear un thin-pool de 10G "pool"
  - **lvcreate -T -L10G Datos/pool**
- Crear un thin-volume "ftpserver" no pool de 100G
  - **lvcreate -T -V100G -n ftpserver Datos/pool**
- Ver información do grupo de volumes
  - **vgdisplay -v**
- Ver información dos volumes e dispositivos
  - **lvs -a -o +devices**

NFS (Network File System)

Si queremos proporcionar almacenamento en rede para máquinas virtuais a alternativa máis simple é o uso de NFS. NFS é un sistema para compartir arquivos en rede extremadamente rápido e utilizado dende hai moitos anos en moitos ámbitos. Hoxe en día as versións de NFS en uso son NFSv3 e NFSv4, que presentan algunhas diferenzas, sendo a máis importante no soporte da seguridade ofrecida. Os servidores NFS soportan simultáneamente ambas versións.

Para compartir un conxunto de carpetas mediante NFS, debemos instalar o servidor (en Debian **apt install nfs-kernel-server**) e a continuación configurar o arquivo **/etc/exports** onde citaremos o conxunto de carpetas que queremos exportar.

> *Cando modifiquemos o ficheiro **/etc/exports** podemos executar exportfs **-a** para que o servidor aplique os cambios.*

Consideracións de Seguridade

Os recursos NFSv3 fían a seguridade ao sistema de permisos estándar do sistema de arquivos (Propietario / Grupo / Outros). O servidor permitirá ou denegará o acceso a un arquivo/carpeta compartida baseándose unicamente en eses permisos. Cando montamos unha carpeta remota mediante NFS, todas as lecturas e escrituras se realizarán co UID/GID que teña o proceso que realiza o acceso. O protocolo enviará esa UID/GID coa petición e será a UID/GID que se utilizará no servidor. ***Iso implica que un usuario nun cliente, potencialmente pode facerse pasar polo usuario que desexe no servidor simplemente alterando o valor do seu UID/GID.***

> *En NFSv3 os permisos utilizan únicamente os GID/UID nunca os nomes de usuario e grupo, que son completamente ignorados*

Para paliar ese problema, os sistemas son exportados coa opción **root_squash** que transforma o UID/GID de root no UID/GID dun usuario sen privilexios (como **nobody:nogroup** ou o UID/GUID concreto que se desexe). Si non desexamos este comportamento podemos indicar **no_root_squash** na exportación da carpeta en */etc/exports* (altamente desaconsellado).

> *Especificando en /etc/exports **all_squash** se mapearán todos os usuarios a nobody:nogroup*

Para que este sistema de xestión de permisos funcione de modo apropiado precisamos que:

- ***Todas as máquinas da rede teñan os mesmos usuarios cos mesmos UID/GID***
- ***Ningún usuario ten permisos de administración sobre un equipo conectado á rede***
- ***Non se debe permitir a conexión de equipos alleos, de xeito que se poda garantizar o punto anterior.***

Mentres que estas condicións se poden satisfacer de modo relativamente simple en redes pequenas, en redes grandes é moi complicado facelo, polo que se fai necesario o uso de NFSv4. **NFSv4** é a última especificación de NFS, e as súas aportacións máis importantes son:

- **Presenta dous modos de seguridade**: Seguridade **sys** e seguridade ***Kerberos***. A seguridade **sys** é idéntica a que facilita NFSv3. A seguridade ***Kerberos*** emprega un servidor de autenticación centralizado que permite utilizar unha mesma base de datos de usuarios para toda a rede. Existen 3 modos de seguridade kerberos: ***krb5*** (so autenticación), ***krb4i*** (autenticación e integridade) e ***krb4p*** (autenticación, integridade e cifrado). Esta opción se controla mediante o parámetro **sec=** nas opcións de exportación especificadas en */etc/exports.*

- **Permite a autenticación de equipos:** Mentres que en NFSv3 non existe ningún modo de verificación dos clientes NFS, NFSv4 permite o uso de *Kerberos* para asegurar a identidade das equipas conectadas. Para elo basta especificar gss/krb5[ip] en lugar da ip do equipo/rede en *etc*exports

- **Permite o uso de nomes de usuario e grupo para xestionar a seguridade**, o que se pode utilizar para facer "idmapping" que consiste en trasladar os nomes de usuario/grupo dos clientes en outro distinto no servidor mediante o servizo ***rpc.idmapd*** configurado mediante o ficheiro ***/etc/idmapd.conf***. Este sistema de mapeo está pensado para o seu uso mediante seguridade *Kerberos*, pero é posible facelo con seguridade ordinaria (sys) facendo ***echo 0 > /sys/module/nfsd/parameters/nfs4_disable_idmapping***

- **Soporta acls (*nfsv4_acl*).** Estas ACL proporcionan un control dos permisos moito máis finos que as ACL POSIX e son cercanas as soportadas polo sistema NTFS.

No uso de NFS para virtualización interesa sobre todo a velocidade, polo que non resulta de interese kerberos nin ningunha outra característica que poda ralentizar a velocidade de transmisión. NFSv3 pode ser unha boa opción si mantemos os hosts de virtualización nunha rede segura separada do resto da rede.

O ficheiro /etc/exports

O modo en que se comparten as carpetas difiere lixeiramente entre NFSv3 e NFSv4, aínda que é compatible. En NFSv3 se comparten carpetas individuais mentres que en NFSv4 se comparten "árbores de arquivos".

En NFSv4 todos as carpetas compartidas parten dunha carpeta raíz que debe ser exportada coa opción **fsid=0.** Cando queremos exportar carpetas situadas fora desta árbore, as podemos montar cun **mount -o bind.** Podemos utilizar esta mesma estratexia para NFSv3, e así poder utilizar un mesmo */etc/exports.*

Para montar a carpeta ***/srv/nfs/share*** *tendo como rootfs NFSv4* ***/srv/nfs*** *en NFSv4 faríamos:*

mount server:/share /mountpoint

A mesma montaxe en NFSv3 sería:

mount server:/srv/nfs/share /mountpoint

O formato do ficheiro /etc/exports é :

<u>Formato NFSv3</u>

carpeta_a_compartir ***equipo/rede (opcións de compartición)***

As opcións de compartición máis importantes son:

ro - So lectura	**rw** - lectura/escritura	**no_root_squash** - Non descarga root
sync - Escritura síncrona	**async** - escritura asíncrona	**all_squash** - Descarga todos os usuarios

<u>Formato NFSv4</u>

carpeta_a_compartir ***equipo/rede(opcións de compartición)***
carpeta_a_compartir ***gss/krb5[ip](opcións de compartición)***

Ademais das opcións soportadas por NFSv3, en NFSv4 dispoñemos de:

sec= (pode ser sys, krb5, krb5i ou krb5p) **fsid=** (id do sistema de arquivos)

iSCSI

iSCSI é unha implantación do protocolo empregado por ***SCSI*** sobre ethernet, o que permite "enganchar" discos ofrecidos a través da rede como si foran discos do propio sistema. É unha alternativa barata a outros sistemas moito máis custosos como o *Fiber Channel Protocol*. Mediante iSCSI podemos expoñer un dispositivo local a través da rede de xeito que poda ser utilizado por un equipo remoto. Os dispositivos ofrecidos se denominan ***targets*** mentras que os clientes que solicitan o uso do dispositivo se denominan ***initiators.*** Tanto os targets, como os initiators se identifican mediante nomes denominados **iqn (iSCSI Qualified Name).**

Formato do iqn

Os iqn dos initiator e dos targets deben ser únicos na rede. E o seu nome habitualmente respeta o seguinte formato: ***iqn.yyyy-mm.autoridade-de-nomeamento:nome***.

- ***yyyy-mm*** - son o ano e mes no que se comezou a utilizar a "autoridade de nomeamento",
- ***autoridade-de-nomeeamento*** - é normalmente o nome de dominio do servidor iSCSI en formato inverso (com.example.iscsiserver)
- **nome** é o nome do recurso iSCSI ofrecido

Configuración do Servidor

En Debian temos dous servidores distintos: IET e LIO. LIO é o sistema adoptado oficialmente polo Kernel e se xestiona mediante a utilidade **targetcli** (*apt install targetcli-fb*). **targetcli** nos permite xestionar os dispositivos compartidos mediante unha "árbore" similar a un sistema de arquivos. As "ramas" relevantes son:

- **backstores**
  Aquí configuramos o dispositivo a compartir. En "fileio" si queremos utilizar un ficheiro para ofrecelo como disco, ou en "block" si queremos compartir un disco, unha partición ou un volume LVM.
- **iscsi**
  Aquí configuramos a compartición do dispositivo que desexemos entre os configurados en backstores:
  - En primeiro lugar debemos crear unha **iqn** para o target (**create).** Se aconsella utilizar como ano e mes os da creación do target, e detrás dos dous puntos o nome que queramos para o disco compartido.
  - Unha vez creado o target, aparecerán dentro de este a "carpeta" **tpg1.** Dentro dela poderemos ver **acls**, **luns** e **portals**. A 'carpeta' **acls** é para configurar os permisos de conexión, a 'carpeta' **luns** para indicar qué dispositivos imos compartir e a 'carpeta' **portals** indicará en que IPs e portos se esperará a conexión dos initiator.
    - Dentro de **luns** (Un LUN era a identificación dun dispositivo nun bus iSCSI) agregamos os "backstores" que queremos compartir. Podemos poñer un dispositivo por target (1 LUN por BUS iSCSI) ou varios. Para as máquinas virtuais é máis flexible varios, a non ser que teñamos varios discos ligados a unha soa máquina. O lun se crea con ***create backstore nomedolun.*** Por exemplo, create mylvmvolume lun1 (mylvmvolume é o nome que lle dimos en backstores)
    - Dentro de **acls** debemos engadir con ***create iqn*** os clientes que desexamos que se podan conectar ao recurso. Iso creará unha carpeta que corresponde coa iqn. Dentro dela podemos configurar un usuario ou unha password con ***set auth***, ou ver a configuración actual con ***get auth.***

      - ***O iqn do cliente ten que ser único. Debian xera un ficheiro con iqn único na carpeta /etc/iscsi/initiatorname.iscsi no momento da instalación de open-iscsi***

Cando saímos da aplicación a configuración se garda automáticamente. O normal sería que esta configuración se cargara automaticamente no inicio do servidor, pero en Debian actualmente non ocurre así.
Debemos crear un servizo systemd que se encargue. O ficheiro de servizo sería así:

```
[Unit]
Description=Restore LIO kernel target configuration
Requires=sys-kernel-config.mount
After=sys-kernel-config.mount network.target local-fs.target network-config.service

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/bin/targetctl restore
ExecStop=/usr/bin/targetctl clear
SyslogIdentifier=target

[Install]
WantedBy=multi-user.target
```

Conexión ao disco compartido (cliente)

Nos clientes necesitaremos o paquete (en Debian) ***open-iscsi***, que nos proporcionará as utilidades de xestión do cliente. A máis importante é **iscsiadm.** As operacións típicas son:

- **Descubrimento dos discos iSCSI**

```
iscsiadm -m discovery -o update -t sendtargets  -p  servidoriSCSI
```

Este comando lee os nodos iSCSI ofrecidos polo servidor e crea en /*etc/*iscsi/nodes uns ficheiros coa configuración de conexión de cada disco (LUN). Entre outras cousas, configura a autenticación da conexión cos parámetros:

node.session.auth.authmethod =
node.session.auth.username =
node.session.auth.password =

Os valores que coloca ahí por defecto se collen do ficheiro ***/etc/iscsi/iscsid.conf***

- **Conexión a un disco iSCSI**

```
iscsiadm -m node  --targetname targetName  -p  servidoriSCSI --login
```

Unha vez realizada a conexión ao disco, o sistema amosará un novo dispositivo (/dev/sdX) co que podemos traballar exactamente igual que con calquera disco local.

A conexión dun mesmo disco por varias máquinas simultáneamente é posible, pero o seu uso simultáneo destruirá a información a non ser que se usen sistemas de arquivos especialmente deseñados para o uso concurrente (OCFS, GFS2 .. etc)

- **Desconexión dun disco iSCSI**

```
iscsiadm -m node  --targetname targetName  -p  servidoriSCSI --logout
```

- **Listaxe de conexións iSCSI**

```
iscsiadm -m session
```

Creando os dispositivos

Cando utilizamos iSCSI para as nosas máquinas virtuais non podemos prever qué nome vai ter o dispositivo cando se realiza a conexión. Como non é factible cambiar a configuración da máquina virtual cada vez que conectamos un dispositivo iSCI debemos conseguir que o disco se conecte sempre co mesmo nome.

Linux utiliza ***udev*** para a xestión dos dispositivos *plug&play*. Este sistema se encarga de asignar nome ós dispositivos que aparecen no sistema unha vez arrancado, e de eliminar os dispositivos cando o hardware correspondente é desconectado. Mediante unhas regras específicas podemos conseguir que os discos reciban sempre o mesmo nome. Unha boa elección é ***/dev/nomeVM***

1. ***Determinar UUID do dispositivo iSCSI***
   - Conectamos o dispositivo iSCSI e buscamos o dispositivo creado examinando */var/log/syslog*
   - Obtemos o **ID** do dispositivo mediante o comando ***/lib/udev/scsi_id –whitelisted dispositivo***

2. ***Escritura da regla iSCSI***
   - En */etc/udev/rules.d/* creamos/editamos o ficheiro ***20-persistent-iscsi.rules*** e coa ***ID*** obtida no apartado anterior engadimos a regla:

```
KERNEL=="sd[a-z]", SUBSYSTEM=="block", PROGRAM="/lib/udev/scsi_id --whitelisted --replace-whitespace /dev/$name", RESULT=="ID", SYMLINK+="nomeVM"
```

   Isto creará en */dev* un enlace simbólico */dev/nomeVM* ó dispositivo iSCSI */dev/sdX* correspondente no momento de conectar o dispositivo e o eliminará cando o desconectemos.

   Reiniciamos o servizo udev: ***systemctl restart udev***

3. ***Proba do funcionamento***
   Para probar o correcto funcionamento basta con desconectar o disco iSCSI e volvelo a conectar. Si miramos en /dev veremos o novo dispositivo creado. Debemos configurar a máquina virtual para que use ese dispositivo como disco.

Virtualización da Rede

As máquinas virtuais necesitan conectarse á rede exactamente igual que os equipos físicos. Para proporcionar conectividade, o hypervisor crea unha parella de tarxetas ethernet virtuais conectadas de xeito que a información que se envía por unha se recibe pola outra (veth pair). Unha das ethernet faise visible á máquina virtual, mentres que a outra queda dispoñible no hypervisor. A conectividade se xestiona mediante ***pontes (bridges)*** e ***switches virtuais***, e se ailla (si é necesario) da rede física mediante o uso de ***VLAN***.

> Mentres que nos hypervisor tipo II se xestiona a conectividade da rede con ferramentas propias (Vboxmanage en Virtualbox) independentes da xestión de rede do sistema, nos tipo I as ferramentas de configuración da rede son as estándar do sistema.

TAP/TUN devices

Mediante un dispositivo TAP/TUN o sistema operativo pode enviar información a un programa no espazo do usuario (como unha máquina virtual). Por outra banda, a información escrita polo programa de usuario no dispositivo pasa directamente á pila de rede do sistema operativo (como si fora unha tarxeta de rede normal).

O seu uso principal é a creación de VPN (Virtual Private Networks) e a conectividade de máquinas virtuais. VirtualBox e KVM empregan dispositivos TUN/TAP para dotar de conectividade ás máquinas virtuais. Os dispositivos TUN operan en capa-3 (rede), mentres que os dispositivos TAP operan en capa-2 (enlace)

Para crear e eliminar estes dispositivos se pode utilizar o comando **ip:**

- **Crear o dispositivo TAP tapX**

```
ip tuntap add tapX mode tap
```

- **Crear o dispositivo TUN tunX**

```
ip tuntap add tunX mode tun
```

- **Eliminar o dispositivo TUN tunX**

```
ip tuntap del tunX mode tun
```

- **Eliminar o dispositivo TAP tapX**

```
ip tuntap del tapX mode tap
```

MacVTap

Os dispositivos MacVTap asocian un dispositivo ***tap*** cunha tarxeta física dun xeito similar a como faría unha ponte (*bridge*).

- **Creación de un dispositivo macvtap0 asociado á eth0**

```
ip link add link eth0 name macvtap0 type macvtap
```

Isto creará un dispositivo ***/dev/tapX*** asociado co dispositivo ***macvtap0***

MacVTap pode funcionar de tres xeitos distintos::

- **VEPA (Virtual Ethernet Port Aggregator) –** A información que se escribe en /*dev/*tapX sae directamente polo dispositivo físico ata chegar ao switch, que dirixirá o paquete ao seu destiño (*que pode ser outra máquina no mesmo dispositivo macvtap*). O switch debe soportar 'hairpin' (Reflective Relay), o switch debe ser capaz de reenviar un frame polo mesmo porto que o recibe. Os switches comunmente non soportan este modo de operación.
- **Bridge** – As máquinas virtuais no mesmo dispositivo MacVTap se comunican directamente sen facer uso da tarxeta física
- **Private** – As máquinas no mesmo dispositivo MacVTap non se poden comunicar

vEth (Parellas de ethernets)

Como xa comentamos unha parella de v-ethernets (veth) é unha parella de dispositivos virtuais nos que a información que se escribe nun membro se obten dende o outro (podemos facer un símil a un cabo de rede con conectando dúas tarxetas ethernet).

- **Creación da parella de ethernets virtuais** *devicename* **e** *devicename-sw*

```
ip link add dev devicename  type veth peer name devicename-sw
```

Unha vez creada a parella de ethernets un extremo o podemos comunicar con outras redes mediante un switch virtual (openvswich) ou unha ponte (bridge), mentres que o outro extremo é o que recibiría IP e se utilizaría para o envío e recepción da información. Este sistema é o máis utilizado para virtualización a nivel de sistema operativo (Containers como LXC) mentres que a virtualización hardware comunmente emprega TUN/TAP.

Pontes (Bridges)

Unha ponte agrupa varias tarxetas ethernet de xeito que a información pode circular entre elas dun xeito similar a como opera un "hub". Todas as máquinas (virtuais ou físicas) cunha interfaz na mesma ponte poderán comunicarse entre elas de xeito directo (sen necesidade dun gateway). Podemos xestionar unha ponte mediante o paquete **bridge-utils** (*apt install bridge-utils*):

- **Crear a ponte** *bridgename*

```
brctl addbr bridgename
```

- **Engadir a ethernet interface a** *bridgename*

```
brctl addif bridgename interface
```

- **Quitar a ethernet interface de** *bridgename*

```
brctl delif bridgename interface
```

- **Eliminar a ponte** *bridgename*

```
brctl delbr bridgename
```

- **Amosar as macs das equipas vistas en** *bridgename*

```
brctl showmacs bridgename
```

- **Amosar a información das pontes (si especificamos un nome de ponte, so desa ponte)**

```
brctl show
```

Aínda que o xeito moderno é o uso do comando ***ip*** :

- **Crear a ponte** *bridgename*

```
ip link add bridgename type bridge
```

- **Engadir a ethernet** *dispositvo* **á ponte** *bridgename*

```
ip link set dispositivo master bridgename
```

- **Eliminar a ethernet** *dispositivo* **da ponte**

```
ip link set dispositivo nomaster
```

- **Amosar a información das pontes**

```
bridge link
```

- **Eliminar a ponte** *dispositivo*

```
ip link del dispositivo type bridge
```

Cando creamos un dispositivo novo estará en estado DOWN, teremos que habilitalo mediante o comando ***ip link set dispositivo up***

OpenVSwitch

Open Vswitch é unha versión de bridge pensada para o uso en redes de servidores de virtualización, entre outras vantaxes sobre os bridges tradicionais podemos destacar:

- ***Mobilidade do Estado:*** Cando migramos unha máquina virtual o estado da rede (as táboas de MAC, o estado de reenvío de IP, as políticas de rutas, as regras de QoS ... etc) deberían migrar con elas. Open vSwitch proporciona soporte para a implementación desta migración.
- ***Resposta ós cambios na rede***: As redes de virtualización sufren continuos cambios na rede. Open vSwitch proporciona características que permiten xestionar estes cambios (OVSDB, OpenFlow, LLDP,CDP, OSPF)
- ***Soporte de túneis GRE e de tagging nas interfaces***: O soporte do tagging proporciona unha fácil integración das máquinas virtuais nas VLAN 802.1Q

Para utilizar openvswitch dun modo sinxelo basta con instalar (en Debian) o paquete openvswitch-switch (*apt install openvswitch-switch*).

Open vSwitch almacena a configuración de rede nunha base de datos, de xeito que os cambios que realicemos son permanentes entre reinicios da máquina.

- **Crear o switch** *switchname*

```
ovs-vsctl add-br switchname
```

- **Engadir a ethernet** *device* **a** *switchname*

```
ovs-vsctl add-port switchname device
```

- **Eliminar a ethernet** *device* **de** *switchname*

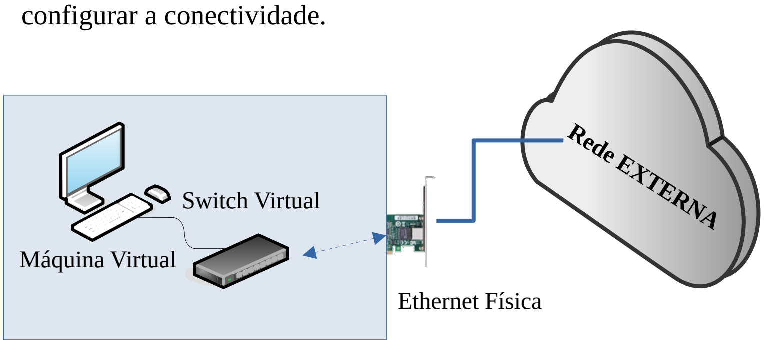
```
ovs-vsctl del-port switchname device
```

- **Eliminar o switch** *switchname*

```
ovs-vsctl del-br switchname
```

Rutado e NAT

Unha vez conectadas as máquinas virtuais á ponte (bridge) ou switch (openvswitch) elixido será necesario configurar a conectividade.

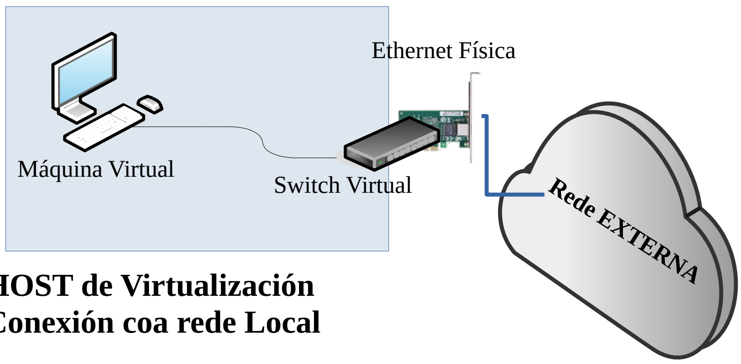


HOST de Virtualización
Configuración de rede NAT

Nesta configuración o host de virtualización dispón dunha ponte ou switch virtual que agrupará na mesma rede física a todas as máquinas que estén conectadas.

O Host de virtualización formará parte da rede virtual si se lle pon unha IP ao switch virtual (actuará como outra tarxeta de rede do host). Unha alternativa é crear unha parella veth, poñer un extremo no swich virtual e unha IP ao outro extremo.

Para a conexión a rede externa é necesario ***activar o reenvío*** de paquetes no host (ip_forward) e facer ***SNAT*** nos paquetes de saída, ou ***configurar a ruta*** a rede virtual en todas as equipas da rede externa



HOST de Virtualización
Conexión coa rede Local

Nesta configuración o host de virtualización dispón dunha ponte ou switch virtual que agrupará na mesma rede física a todas as máquinas que estén conectadas, pero a diferencia do anterior a ethernet física do host de virtualización tamén forma parte deste switch.

O Host de virtualización formará parte da rede virtual si se lle pon unha IP ao switch virtual (actuará como outra tarxeta de rede do host). Unha alternativa é crear unha parella veth, poñer un extremo no swich virtual e unha IP ao outro extremo. Si non se fai ninguha das dúas cousas, o host de virtualización non terá conectividade de rede

A conectividade da máquina virtual co exterior é directa, non precisando ningunha configuración especial.

VLAN 802.1Q

As vlan 802.1Q pretenden permitir que polo mesmo medio físico podan transmitirse tráfico pertencente a múltiples redes sen interferir. Este protocolo engade 4 bytes ao encabezado das tramas ethernet ***cunha etiqueta numérica***, polo que switches que non soporten este protocolo poden descartar os paquetes. Cando un paquete chega a un porto pertencente a unha VLAN pode tratarse de dúas formas distintas:

- **O porto é untagged** (ou *de acceso*): Normalmente conecta hosts incapaces de utilizar VLAN, polo que os paquetes que chegan deben estar sen etiquetar, e os paquetes que se orixinan no host van sen etiqueta. Cando o o paquete chega ao porto do switch, se inxecta o número de VLAN de xeito que únicamente se poda dirixir a outros portos da mesma VLAN (tagged ou untagged). O paquete abandoará o switch sin etiqueta (si o porto de saída é untagged) ou con etiqueta (si o porto de saida é tagged) .
- **O porto é tagged (ou *trunk*)**: Normalmente conecta switches con soporte VLAN ou hosts que utilicen VLAN. Neste caso o paquete de orixe ven xa cunha etiqueta. O paquete se acepta si o porto está configurado para aceptar paquetes desa VLAN e se dirixirá ao porto apropiado. O paquete sairá con etiqueta (si o porto é tagged) ou sen etiqueta (si o porto é untagged)

Un porto dun switch so pode pertencer a unha VLAN untagged, pero pode estar en varias VLAN como tagged

Configuración

Si queremos conectar máquinas virtuais a través dunha VLAN necesitamos que o switch virtual que utilicemos utilice un ***trunk*** coa etiqueta correspondente, e ter configurado o switch físico de xeito apropiado. Para configurar un ***bridge*** nunha VLAN (so ten sentido en bridges que teñan conectividade coa tarxeta física) necesitamos que os paquetes que salgan da tarxeta física leven a etiqueta da VLAN, e tamén que se acepten únicamente paquetes bem etiquetados.

Para isto, a ponte debe levar en lugar da tarxeta física, unha "versión" da tarxeta física que leve a etiqueta, que podemos crear co seguinte comando:

ip link add link ***tarxetafisica*** name ***nometarxetavlan*** type vlan id ***etiqueta***

Bastará con insertar nometarxetavlan na ponte en lugar de tarxetafísica, e as máquinas virtuais conecatadas á ponte se comunicarán con esa VLAN.

Si utilizamos openvswitch, a configuración é mais simple. Basta con indicar a VLAN desexada no momento de insertar a interface da máquina virtual no vswitch:

ovs-vsctl add-port *virtualswitch* ***interface-vm*** tag=***etiqueta***