EL MANUAL DE

Administración de QVD 4.0

QVD DOCUMENTATION

<documentation@theqvd.com>

Other contributors: *Nicolas Arenas, David Serrano, Juan Zea, Salvador Fandiño, Nito Martinez*

November 5, 2018

Contents

List of Figures

Prefacio

Este documento le proporcionará toda la información que necesita para instalar, administrar y gestionar cualquiera de los componentes QVD dentro de una solución QVD. Como producto de código abierto, QVD está creciendo y mejorándose constantemente. Nos esforzamos por mantener nuestra documentación lo más completa posible y animamos a los lectores a notificarnos de cualquier mejora para la documentación. Si tiene cualquier pregunta o sugerencia, por favor envíenos un correo electrónico a info@theqvd.com.

El documento se divide en tres partes principales:

- La primera parte discute el núcleo de componentes que forman una solución, cómo interactúan y cómo se instalan y configuran.
- La segunda parte trata de los problemas de integración y el ajuste de comportamientos dentro de QVD para lograr un mejor rendimiento o para ser más escalable.
- La tercera parte se dedica a proporcionarle toda la información que pueda necesitar para crear y administrar el disco del sistema operativo, imágenes y máquinas virtuales que se cargan en cada escritorio virtual.

Además, proveemos una bibliografía de material externo que le ayude a obtener una mejor comprensión de las diferentes tecnologías involucradas en una solución QVD. También proporcionamos un glosario de términos comúnmente usados.

Este manual se complementa con el de Arquitectura, y se recomienda su lectura previa para comprender ciertos conceptos.

¿Qué es QVD?

QVD (Quality Virtual Desktop) es una solución VDI (Virtual Desktop Infrastructure) enfocada en Linux. El software está diseñado para virtualizar completamente el escritorio Linux, para que los sistemas cliente puedan conectarse a un servidor central y cargar su entorno de escritorio y aplicaciones. Esto significa que cuando los usuarios trabajan desde su máquina local, todos los programas, aplicaciones, procesos y datos utilizados se mantienen en el servidor y se ejecutan de forma centralizada. La virtualización ofrece una serie de beneficios:

- Los usuarios pueden cambiar entre equipos en una red y seguir trabajando como si estuvieran ubicados en el mismo escritorio, con acceso completo a todas sus aplicaciones y datos
- Los administradores tienen un mayor control sobre las aplicaciones que se instalan en los sistemas del usuario, y son capaces de administrar los datos del usuario más fácilmente para realizar copias de seguridad y análisis de virus, etc
- Es más fácil para los administradores proporcionar nuevos escritorios y desplegar aplicaciones para nuevos usuarios
- Se reduce el tiempo de inactividad en caso de fallos de hardware
- Los usuarios pueden hacer uso de una variedad de dispositivos diferentes para acceder a su escritorio y aplicaciones, incluyendo computadoras portátiles, PCs y teléfonos inteligentes
- Los usuarios pueden trabajar de forma segura con el mismo escritorio y las aplicaciones de una ubicación remota sin el requisito de una VPN
- Mejora de la seguridad general del sistema y de los datos
- Reducción de costos de hardware, mantenimiento y administración

El servidor QVD virtualiza cada escritorio Linux. Esto se puede lograr utilizando una de las dos tecnologías de virtualización. Lo más común es que la máquina virtual del kernel de Linux (KVM) se utilice como un hipervisor completo tipo 1 (bare metal), sin embargo a partir de QVD 3.1, también es posible aprovechar Linux Containers (LXC) para lograr la virtualización del sistema operativo. Esta virtualización ayuda a mantener el entorno de cada usuario como su propia entidad discreta, para mejorar la seguridad y la estabilidad. La virtualización le permite servir varios sistemas operativos o entornos a sus usuarios, dependiendo de sus requerimientos. Estos se cargan como imágenes independientes en el servidor QVD.

En general, usted solo cargará una o dos imágenes para todos sus usuarios. Estas imágenes proporcionan el sistema operativo base y el entorno de trabajo, que se replica para cada máquina virtual. Cuando un usuario se conecta al servidor, haciendo uso de la aplicación cliente, se inicia una Máquina Virtual exclusiva para ese usuario. Esto proporciona también una "cárcel" que impide que un comportamiento inadecuado del sistema pueda afectar a otros usuarios. Cuando el usuario se desconecta, la máquina virtual se detiene. Esto significa que si el entorno del usuario se ha convertido en algo problemático, una desconexión puede revertir el entorno a su estado original. Esto proporciona un nivel de seguridad mucho mayor que si un usuario estuviera trabajando en una estación de trabajo independiente.

Con el fin de mantener los datos de usuario, tales como ajustes de escritorio, documentos y otra información específica del usuario, hay dos opciones. El primer enfoque, y el más común, es almacenar esta información en un recurso compartido NFS. De esta manera, los datos pueden almacenarse en un dispositivo NAS o en una SAN, donde se pueden administrar fácilmente. Una segunda opción es cargar una segunda imagen en la máquina virtual. Esta imagen es persistente, ya que puede ser actualizada por el usuario, y los cambios se almacenan para cada vez que se recarga la imagen. Ambos enfoques son igualmente válidos. Al mantener los datos de usuario separados de la imagen de núcleo, QVD ayuda a garantizar que en el caso de que una imagen

de núcleo esté dañada o en caso de fallo del sistema, usted será capaz de minimizar el tiempo necesario para la recuperación de desastres.

Al escritorio se accede desde cada estación de trabajo, haciendo uso de un cliente que utiliza el protocolo NX para comunicarse con el servidor y entregar el escritorio y las aplicaciones al cliente. El protocolo NX se utiliza para manejar conexiones remotas de X Windows y proporciona una compresión superior para permitir un alto rendimiento incluso cuando se accede al escritorio a través de una conexión con poco ancho de banda. Además, QVD es capaz de encapsular el protocolo NX con SSL para asegurar la conectividad de modo que los usuarios puedan trabajar de una manera segura y protegida, incluso si acceden a sus escritorios desde ubicaciones remotas. QVD proporciona software cliente para ejecutarse en una variedad de sistemas operativos y dispositivos básicos: Linux, Windows, OSX e incluso Android e Ios. Esto significa que donde quiera que usted se encuentre, independientemente del sistema al que tenga acceso, podrá ejecutar la aplicación cliente para acceder a su escritorio.

Algunas notas acerca de este manual

En general, se supone que la mayoría de los usuarios de QVD aprovecharán la Virtualización KVM ofrecida dentro del producto. Como resultado, la mayoría de esta guía asume que configurará el producto de esta manera. Si el usuario elige LXC para la virtualización, puede haber algunas diferencias de configuración. En los casos donde esto es significativamente importante, hemos incluido información para ambas plataformas de virtualización. Sin embargo, también hemos incluido un capítulo separado sobre la virtualización LXC que intenta proporcionar orientación adicional a los usuarios que decidan explorar esta opción. En la misma línea, aunque ofrecemos paquetes para otras distribuciones de Linux Como SUSE Linux Enterprise Server (SLES), suponemos que la mayoría de los usuarios utilizarán Ubuntu Linux. Como resultado, muchos de los comandos de esta guía, junto con las ubicaciones de los archivos de configuración, etc., se proporcionan generalmente con la suposición de que está utilizando Ubuntu Linux. Para los usuarios de SLES, también hemos intentado marcar las diferencias significativas de configuración.

Part I

Administración de QVD

La administración de QVD se puede realizar utilizando una de las tres herramientas siguientes:

- **QVD CLI**: una utilidad de línea de comandos que se puede instalar en cualquier nodo y permite manejar la solución en línea de comandos y automatizar tareas mediante scripts.
- **WAT**: una herramienta de administración basada en Web que permite al administrador acceder de forma remota a la solución y realizar una variedad de tareas administrativas mediante un navegador web estándar.
- **API**: La API (nueva en QVD 4) permite integrar la solución con terceros, ya que utiliza una interfaz REST. Además, es necesaria para que las dos anteriores funcionen, teniendo así la lógica centralizada.

Tanto la utilidad en línea de comandos (CLI) como el WAT requieren acceso a la API QVD y necesitarán ser configuradas para este propósito. La API deberá estar instalada en al menos un nodo de la solución, aunque puede instalarse en tantos como se desee. Casi todos los comandos que se pueden realizar a través de cualquiera de estas herramientas simplemente cambiarán valores para entidades dentro la QVD-DB (via API). Los diferentes comportamientos son ejecutados por los diversos elementos de los nodos de la solución QVD basándose en los cambios realizados en la QVD-DB. Las herramientas de administración de QVD también se utilizan para cargar nuevas imágenes en QVD y para configurar sus parámetros de tiempo de ejecución. Para facilitar esta funcionalidad, estas herramientas necesitan tener acceso a las carpetas donde son almacenadas y accedidas por los nodos de virtualización. Por lo general, este acceso es aprovisionado en un recurso compartido de archivos de red, como NFS.

Chapter 1

Configuración básica de QVD

En general, tanto los componentes arquitecturales de QVD como los administrativos requieren de conexión a la base de datos del producto. Esto implica que estos componentes necesitan un fichero de configuración con los datos de conexión a la misma. Esta es también la única configuración que es requerida en lo que a ficheros se refiere. El resto de configuraciones de la solución está en la base de datos y se realiza directamente a través de las herramientas administrativas pertinentes.

La configuración del HKD está dentro del archivo `/etc/qvd/node.conf`. Esta ruta de acceso se crea automáticamente si elige instalar QVD mediante los paquetes que proporcionamos. Si elige otro método, debe crearla manualmente, y puede usar la plantilla de configuración proporcionada:

```
root@myserver:~# cp -R /usr/share/qvd/config /etc/qvd
```

El archivo node.conf debe contener como mínimo lo siguiente:

```
#
# QVD Node Configuration
#
nodename = mycomputer

# Database connection information.
# database.host: where the QVD database is found
# database.name: the name of the QVD database
# database.user: the user account needed to connect
# database.password: the password needed to connect
database.host = mycomputer
database.name = qvddb
database.user = qvd
database.password = passw0rd

path.log = /var/log/qvd
log.filename = ${path.log}/qvd.log
log.level = INFO
```

Debe asegurarse de que el **nodename**, **database.host**, **database.name**, **database.user** y **database.password** contienen valores que coinciden con los que haya configurado. Una vez que estos ajustes estén en su lugar, cualquier utilidad que requiera acceso a la base de datos tendrá los detalles de configuración apropiados para hacerlo.

Las entradas relacionadas con el log se deben establecer aquí porque los componentes de QVD se inicializan antes de conectarse a la base de datos.

La configuración del componente API está en el archivo `/etc/qvd/api.conf`. La configuración es idéntica a node.conf y puede partir de este fichero para hacerla. Tan solo se requiere dos parámetros extra:

```
api.user = qvd
api.group = qvd
```

Estos parámetros definen los permisos con los que se ejecuta la API. Los aquí mostrados son solo un ejemplo. Por seguridad, y de acuerdo con la recomendación general, no utilice el usuario *root* para este propósito.

Desde la versión 4.0 de QVD, los componentes WAT y CLI ya no requieren conexión a la base de datos, sino a la API. Tienen ahora por tanto, sus propios ficheros de configuración. Para el CLI `/etc/qvd/qa.conf`:

```
qa.url = https://api.yourqvdserver.com:443/
qa.tenant = *
qa.login = superadmin
qa.password = superadmin
qa.format = TABLE
qa.insecure = 0
qa.ca = /etc/qvd/certs/ca.conf
```

Para el WAT `/usr/lib/qvd/lib/wat/config.json`:

```
{
   "apiUrl": "https://api.yourqvdserver.com:443"
}
```

En cualquier caso, deberá configurar todos estos ficheros según sus propias necesidades.

Otros parámetros de configuración de QVD

Fuera del archivo de configuración, QVD almacena la mayoría de su configuración en la QVD-DB. Hay una amplia gama de parámetros que se aplican a diferentes componentes dentro de la infraestructura QVD. Estos parámetros se pueden configurar utilizando la utilidad de administración de CLI de QVD. Discutimos los pasos apropiados para esto en el capítulo titulado utilidad de administración de QVD CLI. También es posible cambiarlos desde el **WAT** a partir de la versión 4.0. Esto está explicado en su propio manual.

Si bien es posible establecer cualquiera de los siguientes parámetros de configuración dentro del archivo node.conf, la configuración dentro de la QVD-DB siempre tendrá precedencia. Esto significa que si se realiza un cambio en los ajustes contenidos dentro de la base de datos, los ajustes almacenados en el archivo de configuración se vuelven obsoletos y esto es confuso para futuros trabajos administrativos. Por lo tanto, nosotros recomendamos encarecidamente que estas opciones solo se actualicen dentro de la base de datos utilizando la utilidad de administración de CLI de QVD.

Esta sección describe algunos de estos parámetros de configuración adicionales. Mientras que hay muchos otros ajustes que podrá ver utilizando la CLI de QVD, algunos de ellos (como los parámetros que comienzan por "intern") nunca deben ser modificados sin la ayuda de un ingeniero de soporte de QVD. En general, no recomendamos que cambie ninguno de estos parámetros de configuración sin supervisión del grupo de soporte de QVD. De hecho, en el WAT no aparecen estos parámetros.

Tenga en cuenta que para establecer estos parámetros, debería haber instalado y configurado QVD-DB.

Tip
Algunos parámetros de configuración pueden estar relacionados con los parámetros componente del sistema. En estos casos, es posible que deba actualizar el parámetro en más de un lugar. Un ejemplo típico sería el ajuste del `l7r.port` que afectaría a la configuración `client.host.port` .

Warning
Insistimos en que los parámetros intern son para uso interno del producto y no se espera que el administrador los modifique. Están sujetos a cambios en cualquier lanzamiento del software y están diseñados para ayudar a los desarrolladores a depurar el comportamiento dentro del producto.

Rutas del sistema QVD

Las siguientes opciones están disponibles para cambiar las rutas que QVD utiliza para buscar aplicaciones, certificados y otros datos específicos de QVD.

```
path.run = /var/run/qvd
path.log = /var/log
path.tmp = /var/tmp
path.storage.root = /var/lib/qvd/storage
path.storage.staging = ${path.storage.root}/staging
path.storage.images = ${path.storage.root}/images
path.storage.overlays = ${path.storage.root}/overlays
path.storage.homes = ${path.storage.root}/homes
path.ssl.certs = ${path.run}/ssl
path.ssl.ca.system = /etc/ssl/certs
path.ssl.ca.personal = .qvd/certs
path.cgroup = /sys/fs/cgroup
path.cgroup.cpu.lxc = /sys/fs/cgroup/cpu/lxc
path.serial.captures = ${path.tmp}/qvd

command.kvm = kvm
command.kvm-img = kvm-img
command.nxagent = /usr/bin/nxagent
command.nxdiag = /usr/bin/nxdiag.pl
command.x-session = /etc/X11/Xsession

command.useradd = /usr/sbin/useradd
command.userdel = /usr/sbin/userdel
```

Los valores anteriores son los valores predeterminados.

- * path.run *: la ruta de acceso (normalmente es referenciada por otras opciones de ruta de acceso)
- * path.log *: la ruta de acceso base para almacenar logs
- * path.tmp *: la ruta para almacenar archivos temporales
- * path.storage.root *: la ruta base para el área de almacenamiento principal utilizada por QVD
- * path.storage.staging *: este directorio se utiliza para almacenamiento temporal de DIs. En la versión 4.0, también puede utilizarse a modo de librería multitenant de imágenes.
- * path.storage.images *: el directorio de imágenes usado para almacenar DIs registradas
- * path.storage.overlays *: el directorio de *overlay* utilizado para mantener las imágenes qcow superpuestas
- * path.storage.homes *: el directorio donde se almacenan las imágenes de directorios de inicio de los usuarios cuando se utiliza qcow
- * path.ssl.certs *: la ruta para almacenar certificados SSL usados por QVD
- * path.ssl.ca.system *: la ruta donde se almacenan los certificados de CA del sistema
- * path.ssl.ca.personal *: la ruta donde se almacenan los certificados de CA locales o personales
- * path.serial.captures *: la ubicación utilizada para almacenar capturas en serie (si está activado)
- * command.kvm *: el comando para ejecutar KVM
- * command.kvm-img *: el comando utilizado para trabajar con discos virtuales QEMU dentro de KVM
- * command.nxagent *: la ruta de acceso al binario nxagent (habitualmente sólo es usado por el VMA en un OSF)
- * command.nxdiag *: la ruta de acceso al script nxdiag.pl utilizado por el VMA para asegurar que nxagent se ejecuta correctamente

- * Command.x-session *: la ruta de acceso al script de shell XSession ejecutado por el sistema cuando se inicia una sesión de X Windows
- * Command.useradd *: la ruta de acceso al script useradd utilizado por el sistema para agregar usuarios
- * Command.userdel *: la ruta de acceso al script userdel utilizado por el sistema para eliminar usuarios

Logging

Las siguientes opciones se pueden utilizar para cambiar la ruta al archivo de log y para controlar la salida de nivel de logging.

```
path.log = /var/log
log.filename = ${path.log}/qvd.log
log.level = INFO
```

Los valores anteriores son los valores predeterminados.

- * path.log *: ruta base de los archivos de log
- * log.filename *: la ruta de acceso al archivo de log
- * log.level *: la salida del nivel de logging, los valores pueden ser: ALL, DEBUG, INFO, WARN, ERROR, FATAL, OFF

Estas opciones deben configurarse en el archivo de configuración de QVD `/etc/qvd/node.conf` porque el sistema de logging se inicializa antes que la conexión a la base de datos. Si se establece estos valores en la base de datos se ignorarán.

QVD genera su log mediante `Log::Log4perl`, un módulo estándar de perl que tiene muchas posibilidades en cuanto a métodos de logging. Puede elegir enviar la salida de logs a syslog, a un archivo o incluso a una base de datos. Para mandar el log a syslog, las siguientes variables de configuración se pueden establecer en `node.conf`:

```
log4perl.appender.SYSLOG = Log::Dispatch::Syslog
log4perl.appender.SYSLOG.layout = Log::Log4perl::Layout::PatternLayout
log4perl.appender.SYSLOG.layout.ConversionPattern = %d %P %F %L %c - %m%n
log4perl.rootLogger = DEBUG, SYSLOG
log.level = DEBUG
```

Para obtener un desglose completo de las diferentes opciones de logging disponibles en Log4perl, consulte la documentación log4perl.

Si selecciona guardar sus logs en un archivo, asegúrese de utilizar alguna forma de control del crecimiento de sus archivos de log.

Opciones de configuración del L7R

Puede especificar las siguientes opciones adicionales para controlar el L7R:

```
l7r.as_user = root
l7r.use_ssl = 1
l7r.port = 8443
l7r.address = *
l7r.pid_file = ${path.run}/l7r.pid
l7r.auth.plugins = default
l7r.loadbalancer.plugin = default
l7r.loadbalancer.plugin.default.weight.ram = 1
l7r.loadbalancer.plugin.default.weight.cpu = 1
l7r.loadbalancer.plugin.default.weight.random = 1
```

Los valores establecidos anteriormente son los valores por defecto.

- * l7r.as_user *: el usuario que se debe utilizar para ejecutar el proceso QVD L7R

- * l7r.use_ssl *: utilizar o no SSL para cifrar conexiones de cliente
- * l7r.port *: el puerto que se debe escuchar para las conexiones del cliente con el L7R (la configuración client.host.port para cada cliente también debería estar configurada para este valor)
- * l7r.address *: la dirección IP a la que el L7R debe enlazar
- * l7r.pid_file *: la ruta al archivo PID que se crea cuando se está ejecutando el proceso L7R
- * l7r.auth.plugins *: se puede usar para proporcionar complementos de autenticación adicionales como OpenSSO
- * l7r.loadbalancer.plugin *: se puede utilizar para incluir un plugin de algoritmo de balanceo de carga alternativo
- * l7r.loadbalancer.plugin.default.weight.ram *: peso asignado a recursos de RAM para el algoritmo de balanceo de carga predeterminado
- * l7r.loadbalancer.plugin.default.weight.cpu *: peso asignado a los recursos de la CPU para el algoritmo de balanceo de carga predeterminado
- * l7r.loadbalancer.plugin.default.weight.random *: peso asignado al aleatorizador para el algoritmo de balanceo de carga predeterminado

Opciones de configuración del HKD

Puede especificar las siguientes opciones adicionales para controlar el HKD:

```
hkd.vm.starting.max = 6
```

El valor establecido anteriormente es el valor predeterminado. * * hkd.vm.starting.max *: el número máximo de máquinas virtuales que el HKD permitirá concurrentemente en estado "starting" (arrancando) antes de iniciar una nueva instancia en un nodo de servidor.

Opciones de VM

Hay algunas opciones que se pueden configurar para controlar el comportamiento de la máquina virtual dentro de QVD:

```
vm.overlay.persistent = 0
vm.kvm.virtio = 1
vm.network.ip.start =
vm.network.netmask =
vm.network.gateway =
vm.network.bridge =
vm.network.dns_server =
```

Tenga en cuenta que si utiliza la Utilidad de administración CLI Legacy puede necesitar también estos parámetros:

```
vm.vnc.redirect = 0
vm.vnc.opts =
vm.serial.redirect = 1
vm.serial.capture = 0
```

Los valores mostrados anteriormente son los valores por defecto.

- * vm.overlay.persistent *: si desea hacer uso de overlays persistentes para archivos temporales y de registro. Tenga en cuenta que esta persistencia no será extendida entre los nodos si el overlay se almacena localmente por ejemplo en una configuración btrfs
- * vm.kvm.virtio *: si utilizará el controlador virtio para la conexión en red (el OSF que se ejecuta en la imagen debe soportar el controlador virtio)

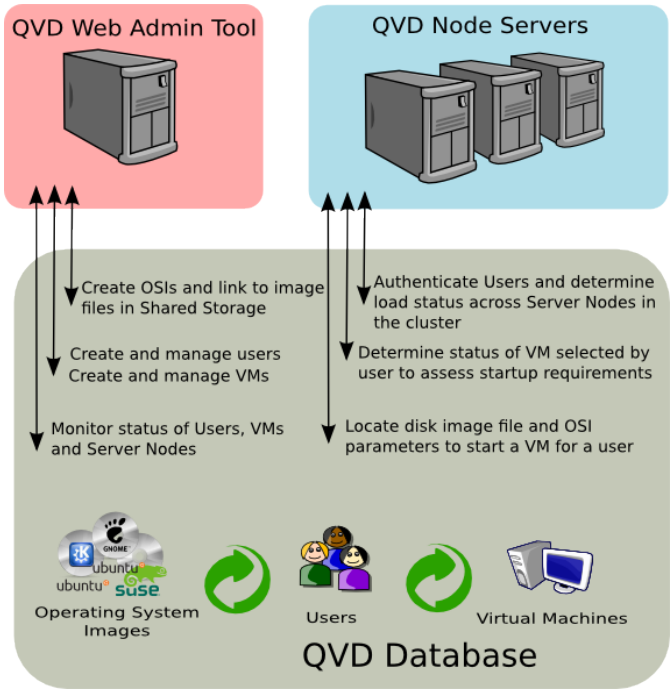
- * vm.vnc.redirect *: habilita el servidor VNC integrado cuando se utiliza la virtualización KVM. (Útil para solucionar problemas de una imagen)
- * vm.vnc.opts *: configuración adicional para el servidor VNC incorporado de KVM
- * vm.serial.redirect *: habilita la consola del puerto serie cuando se usa la virtualización KVM
- * vm.serial.capture *: captura la salida de la consola serie en el archivo
- * vm.network.ip.start *: la dirección IP de inicio para el rango asignado a las máquinas virtuales en la red reservada para QVD
- * vm.network.netmask *: máscara de red CIDR para el tamaño de la red reservada para QVD
- * vm.network.gateway *: IP del firewall en la red reservada para QVD que será transmitida por DHCP a las máquinas virtuales
- * vm.network.bridge *: Nombre de la interfaz en modo puente de red
- * vm.network.dns_server *: IP del servicio DNS que se servirá a las máquinas virtuales por DHCP en la red reservada para QVD

Tenga en cuenta que la configuración de *vm.network* generalmente se requiere para que los nodos QVD funcionen correctamente.

Chapter 2

QVD-DB

La QVD-DB es el pegamento que une todos los componentes QVD juntos. Hace uso de un SGBD subyacente de PostgreSQL versión 9.2 o superior (desde QVD4).



Toda la información de configuración y tiempo de ejecución de toda la infraestructura QVD se almacena en la base de datos y si falla, la plataforma al completo dejará de funcionar. Por eso, es altamente recomendable que la base de datos esté instalada en configuración de alta disponibilidad. Puede averiguar cómo configurar PostgreSQL en una Configuración HA en

Linux-HA + DRBD + PostgreSQL

y

High Avaibility PostgreSQL HOWTO.

Los requisitos reales de hardware para QVD-DB son muy modestos y cualquier servidor con sólo dos núcleos de CPU y 2 GB de RAM será capaz de cargar la base de datos.

Important
QVD sólo funciona con PostgreSQL 9.2 o superior, dado que se utilizan sus funciones de notificación

Instalación y configuración de QVD-DB

En el sistema en el que vaya a instalar QVD-DB, deberá agregar el repositorio de QVD a sus fuentes apt.

En primer lugar, agregue la clave pública de los paquetes QVD a sus claves de confianza (como root):

```
# wget -qO - https://www.theqvd.com/packages/key/public.key | sudo apt-key add -
```

Ahora, agregue el repositorio:

```
/etc/apt/sources.list.d/qvd.list
# apt-get update
```

Important

La forma recomendada de instalar la base de datos central es con el paquete perl-qvd-db. Necesitará además el software cliente de PostgreSQL para ejecutar estos pasos (consulte la documentación de PostgreSQL). Todos estos pasos requieren privilegios de root:

```
# apt-get install perl-qvd-db
```

Para SLES el proceso es similar.

En primer lugar, agregue la clave pública de los paquetes QVD a sus claves de confianza (como root):

```
# rpm --import https://www.theqvd.com/packages/key/public.key
```

Ahora, agregue el repositorio:

```
# zypper ar http://theqvd.com/packages/sles/12SP1/stable QVD
# zypper ref
```

Utilice zypper para instalar la base de datos:

```
# zypper install perl-QVD-DB
```

Creación del usuario y la base de datos QVD

Necesitará crear un usuario dentro de PostgreSQL para acceder a la base de datos QVD, y necesitará crear la base de datos real donde QVD puede configurar sus tablas y almacenar sus datos. Para hacer esto, necesitará usar el comando sudo para cambiar a la cuenta de postgres:

```
$ sudo su - postgres
```

Como usuario *postgres*, puede crear cuentas de usuario de PostgreSQL con el comando `createuser`. Solicitará una contraseña para el nuevo usuario y algunos detalles sobre la cuenta de usuario. En general, puede responder *n* a todas las opciones que se presentan. Por ejemplo, para crear un usuario llamado qvd, utilizaría el siguiente comando.

```
postgres@myserver:~$ createuser -SDRP qvd
Enter password for new role: passw0rd
Enter it again: passw0rd
```

Tip
Para más información sobre este comando, utilice la documentación de PostgreSQL: http://www.postgresql.org/docs/-9.2/static/app-createuser.html

El nuevo usuario ahora puede ser asignado como propietario de una base de datos. Para crear una base de datos para QVD y asignar la propiedad, utilice el comando `createdb`. Utilice el parámetro -O para establecer el propietario de la base de datos a la cuenta que desea utilizar. En este caso, estableceremos el propietario para el nuevo usuario que creamos en el paso anterior.

```
postgres@myserver:~$ createdb -O qvd qvddb
```

Tip
Para más información sobre este comando, utilice la documentación de PostgreSQL: http://www.postgresql.org/docs/-9.2/static/app-createdb.html

Requisitos de configuración de PostgreSQL

Con el fin de permitir el acceso concurrente desde todos los nodos de la granja QVD y manejar las transacciones de forma coherente, el nivel de aislamiento de la transacción debe cambiarse de *read commited* a *serializable*. Este es un paso muy importante que no debe ser omitido o su base de datos podría volverse incoherente y QVD dejaría de funcionar.

Además, es necesario permitir el acceso de red a la base de datos. Por defecto, está configurada para sólo escuchar las consultas en localhost. Esto se debe cambiar para escuchar en todas las interfaces.

Para ello, debe editar los archivos de configuración de PostgreSQL postgresql.conf y pg_hba.conf. En Ubuntu se encuentran en /etc/postgresql/9.2/main. En SUSE , encontrará estos archivos en */var/lib/pgsql/data*.

El nivel de aislamiento de la transacción se controla mediante la configuración *default_transaction_isolation*. Para habilitar el acceso general de red a PostgreSQL, cambie la configuración listen_addresses de *localhost* a *.

```
root@myserver:~# cd /etc/postgresql/9.2/main #this would be /var/lib/pgsql/data on SLES
root@myserver:/etc/postgresql/9.2/main# vi postgresql.conf
listen_addresses = '*'
default_transaction_isolation = 'serializable'
```

Para habilitar el acceso de red para el usuario qvd, agregue la siguiente línea a pg_hba.conf (con el formato siguiente: host database user CIDR-address auth-method [auth-options]).

```
root@myserver:/etc/postgresql/9.2/main# vi pg_hba.conf
host  qvddb qvd 192.168.0.0/24  md5
```

Note
Asegúrese de reemplazar la red predeterminada 192.168.0.0/24 por la red que su plataforma QVD utiliza.

Reinicie PostgreSQL para que los cambios surtan efecto.

Para Ubuntu:

```
# service postgresql restart
```

y para SLES:

```
# /etc/init.d/postrgresql restart
```

Aprovisionamiento de QVD-DB

El paquete QVD-DB incluye un script que le ayudará a poblar la base de datos QVD con todas las tablas que son necesarias para que QVD funcione correctamente. Para que este script funcione, requiere que los ajustes de la base de datos QVD se hayan introducido correctamente en el archivo `/etc/qvd/node.conf`. Para desplegar la base de datos, ejecute qvd-deploy-db.pl (se encuentra en la carpeta `/usr/lib/qvd/bin/`. Agregue este directorio a sus rutas de sistema si va a estar administrando una solución QVD).

```
# qvd-deploy-db.pl
```

Una vez que haya ejecutado este comando, QVD-DB estará listo para ser utilizado por cualquier componente dentro del entorno QVD.

Reinicializando QVD

Si en algún momento sólo desea eliminar todos los nodos, imágenes, imágenes virtuales máquinas, etc. configuradas en QVD para comenzar de nuevo (por ejemplo, si lo están probando), puede usar el mismo comando con el parámetro `--force`:

```
# qvd-deploy-db.pl --force
```

Tenga en cuenta que no hay manera de deshacer esta operación una vez que se ha ejecutado. Se eliminarán todos los datos de la base de datos y se empezará desde cero. ¡Utilice este comando con cuidado!

Prueba de acceso a QVD-DB

Debe comprobar que puede acceder a la base de datos desde todos los nodos que la requieren (esto es, todos los nodos que tengan instalado el componente HKD o el componente API). La comprobación más sencilla es conectarse desde ellos a la base de datos y listar las tablas utilizadas por QVD. Para hacer esto, deberá asegurarse de que tiene el cliente de PostgreSQL instalado en el nodo desde el que se está conectando. Puede instalarlo mediante el paquete `postgresql-client` en Ubuntu y `postgresql` en SLES:

En Ubuntu:

```
# sudo apt-get install postgresql-client
```

En SLES:

```
# zypper install postgresql
```

Para listar las tablas en la base de datos QVD utilizando el cliente PostgreSQL, puede hacer lo siguiente:

```
anyuser@otherserver:~$ psql -U qvd -W -h myserver qvddb
Password for user qvd:
psql (9.2)

qvddb=> \d

                          List of relations
 Schema |                     Name                     |   Type   | Owner
--------+----------------------------------------------+----------+-------
 public | acl_role_relations                           | table    | qvd
 public | acl_role_relations_id_seq                    | sequence | qvd
 public | acls                                         | table    | qvd
 public | acls_id_seq                                  | sequence | qvd
 public | administrators                               | table    | qvd
 public | administrators_id_seq                        | sequence | qvd
```

```
public | all_acl_role_relations                          | view     | qvd
public | all_role_role_relations                          | view     | qvd
public | configs                                          | table    | qvd
public | di_properties                                    | table    | qvd
public | di_tags                                          | table    | qvd
public | di_tags_id_seq                                   | sequence | qvd
public | dis                                              | table    | qvd
public | dis_id_seq                                       | sequence | qvd
public | host_cmds                                        | table    | qvd
public | host_counters                                    | table    | qvd
public | host_properties                                  | table    | qvd
public | host_runtimes                                    | table    | qvd
public | host_states                                      | table    | qvd
public | hosts                                            | table    | qvd
public | hosts_id_seq                                     | sequence | qvd
public | log                                              | table    | qvd
public | log_id_seq                                       | sequence | qvd
public | operative_views_in_administrators                | view     | qvd
public | operative_views_in_tenants                       | view     | qvd
public | osf_properties                                   | table    | qvd
public | osfs                                             | table    | qvd
public | osfs_id_seq                                      | sequence | qvd
public | properties_list                                  | table    | qvd
public | properties_list_id_seq                           | sequence | qvd
public | qvd_object_properties_list                       | table    | qvd
public | qvd_object_properties_list_id_seq                | sequence | qvd
public | role_administrator_relations                     | table    | qvd
public | role_administrator_relations_id_seq              | sequence | qvd
public | role_role_relations                              | table    | qvd
public | role_role_relations_id_seq                       | sequence | qvd
public | roles                                            | table    | qvd
public | roles_id_seq                                     | sequence | qvd
public | session                                          | table    | qvd
public | ssl_configs                                      | table    | qvd
public | tenants                                          | table    | qvd
public | tenants_id_seq                                   | sequence | qvd
public | user_cmds                                        | table    | qvd
public | user_properties                                  | table    | qvd
public | user_states                                      | table    | qvd
public | users                                            | table    | qvd
public | users_id_seq                                     | sequence | qvd
public | versions                                         | table    | qvd
public | views_setups_attributes_administrator            | table    | qvd
public | views_setups_attributes_administrator_id_seq     | sequence | qvd
public | views_setups_attributes_tenant                   | table    | qvd
public | views_setups_attributes_tenant_id_seq            | sequence | qvd
public | views_setups_properties_administrator            | table    | qvd
public | views_setups_properties_administrator_id_seq     | sequence | qvd
public | views_setups_properties_tenant                   | table    | qvd
public | views_setups_properties_tenant_id_seq            | sequence | qvd
public | vm_cmds                                          | table    | qvd
public | vm_counters                                      | table    | qvd
public | vm_properties                                    | table    | qvd
public | vm_runtimes                                      | table    | qvd
public | vm_states                                        | table    | qvd
public | vms                                              | table    | qvd
public | vms_id_seq                                       | sequence | qvd
public | wat_log                                          | table    | qvd
public | wat_log_id_seq                                   | sequence | qvd
public | wat_setups_by_administrators                     | table    | qvd
public | wat_setups_by_administrators_id_seq              | sequence | qvd
public | wat_setups_by_tenants                            | table    | qvd
```

```
 public | wat_setups_by_tenants_id_seq                | sequence | qvd
(69 rows)


qvddb=> \q
```

Copia de seguridad y restauración de QVD-DB

Una técnica de copia de seguridad muy simple implicaría el volcado de toda la base de datos PostgreSQL a un archivo:

```
# pg_dump -U postgres postgres > yourfile.backup
```

Para revertir la base de datos para que coincida con un archivo de copia de seguridad, puede ejecutar el siguiente comando:

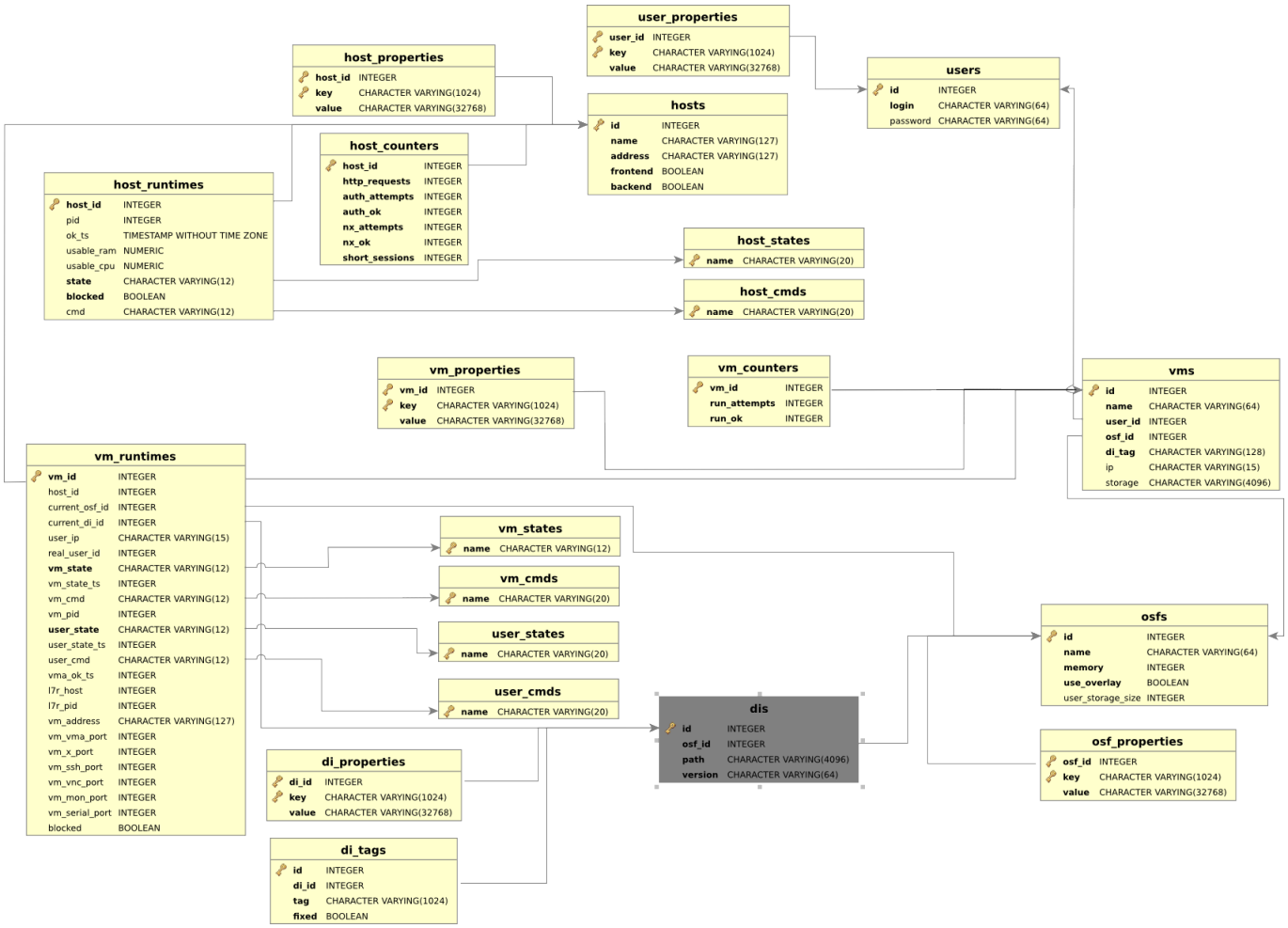
```
# psql -U postgres postgres < yourfile.backup
```

Tip
Para operaciones avanzadas, revise http://www.postgresql.org/docs/9.2/static/backup.html

Modelo relacional de datos QVD-DB

El diagrama siguiente muestra el modelo general de datos de QVD-DB.

Important
Se recomienda que los administradores no intenten modificar directamente en la base de datos, ya que es muy probable que la instalación de QVD deje de funcionar.


user_properties
user_id INTEGER
key CHARACTER VARYING(1024)
value CHARACTER VARYING(32768)
host_properties
host_id INTEGER
key CHARACTER VARYING(1024)
value CHARACTER VARYING(32768)
users
id INTEGER
login CHARACTER VARYING(64)
password CHARACTER VARYING(64)
hosts
id INTEGER
name CHARACTER VARYING(127)
address CHARACTER VARYING(127)
frontend BOOLEAN
backend BOOLEAN
host_counters
host_id INTEGER
http_requests INTEGER
auth_attempts INTEGER
auth_ok INTEGER
nx_attempts INTEGER
nx_ok INTEGER
short_sessions INTEGER
host_runtimes
host_id INTEGER
pid INTEGER
ok_ts TIMESTAMP WITHOUT TIME ZONE
usable_ram NUMERIC
usable_cpu NUMERIC
state CHARACTER VARYING(12)
blocked BOOLEAN
cmd CHARACTER VARYING(12)
host_states
name CHARACTER VARYING(20)
host_cmds
name CHARACTER VARYING(20)
vm_properties
vm_id INTEGER
key CHARACTER VARYING(1024)
value CHARACTER VARYING(32768)
vm_counters
vm_id INTEGER
run_attempts INTEGER
run_ok INTEGER
vms
id INTEGER
name CHARACTER VARYING(64)
user_id INTEGER
osf_id INTEGER
di_tag CHARACTER VARYING(128)
ip CHARACTER VARYING(15)
storage CHARACTER VARYING(4096)
vm_runtimes
vm_id INTEGER
host_id INTEGER
current_osf_id INTEGER
current_di_id INTEGER
user_ip CHARACTER VARYING(15)
real_user_id INTEGER
vm_state CHARACTER VARYING(12)
vm_state_ts INTEGER
vm_cmd CHARACTER VARYING(12)
vm_pid INTEGER
user_state CHARACTER VARYING(12)
user_state_ts INTEGER
user_cmd CHARACTER VARYING(12)
vma_ok_ts INTEGER
l7r_host INTEGER
l7r_pid INTEGER
vm_address CHARACTER VARYING(127)
vm_vma_port INTEGER
vm_x_port INTEGER
vm_ssh_port INTEGER
vm_vnc_port INTEGER
vm_mon_port INTEGER
vm_serial_port INTEGER
blocked BOOLEAN
vm_states
name CHARACTER VARYING(12)
vm_cmds
name CHARACTER VARYING(20)
user_states
name CHARACTER VARYING(20)
user_cmds
name CHARACTER VARYING(20)
dis
id INTEGER
osf_id INTEGER
path CHARACTER VARYING(4096)
version CHARACTER VARYING(64)
osfs
id INTEGER
name CHARACTER VARYING(64)
memory INTEGER
use_overlay BOOLEAN
user_storage_size INTEGER
osf_properties
osf_id INTEGER
key CHARACTER VARYING(1024)
value CHARACTER VARYING(32768)
di_properties
di_id INTEGER
key CHARACTER VARYING(1024)
value CHARACTER VARYING(32768)
di_tags
id INTEGER
di_id INTEGER
tag CHARACTER VARYING(1024)
fixed BOOLEAN

Chapter 3

Nodos servidor QVD

Los nodos servidor QVD son el motor de la infraestructura QVD. Los nodos ejecutan un solo componente binario, el *HKD* o *House Keeping Daemon* que realiza un seguimiento del estado de las máquinas virtuales. El HKD es responsable de iniciar y detener las máquinas virtuales. Supervisa también el estado de cada máquina virtual y actualiza la información de estado dentro de la base de datos QVD, de modo que otros nodos y las herramientas de administración puedan funcionar en consecuencia. En general, el HKD es responsable de administrar el estado de las máquinas virtuales.

El HKD también invoca el *L7R* (Level 7 Router) intermediario entre cliente y servidor, responsable de autenticar usuarios, establecer sesiones y enrutar al usuario hasta su máquina virtual cuando se conecta. En general, el L7R es responsable de administrar el estado del usuario.

La instalación habitual de los nodos es en cúmulo (cluster) o granja. Esto significa que dentro de un despliegue típico es probable que tenga cualquier número de nodos de servidor.

Para familiarizarse con la arquitectura general de un nodo servidor consulte el Manual de Arquitectura de QVD.

Instalación de un nodo servidor QVD

En cualquiera de los sistemas en los que va a instalar los componentes de nodo servidor QVD , tendrá que agregar el repositorio QVD a sus fuentes de apt.

En primer lugar, agregue la clave pública de los paquetes QVD a sus claves de confianza (como root):

```
# wget -qO - https://www.theqvd.com/packages/key/public.key | sudo apt-key add -
```

Ahora, agregue el repositorio:

```
/etc/apt/sources.list.d/qvd.list
# apt-get update
```

Important

Para instalar todos los componentes del nodo de servidor QVD y sus dependencias, ejecute el siguiente comando:

```
# apt-get install perl-qvd-node
```

Instalación del nodo Servidor QVD en SLES

En primer lugar, agregue la clave pública de los paquetes QVD a sus claves de confianza (como root):

```
# rpm --import https://www.theqvd.com/packages/key/public.key
```

Ahora, agregue el repositorio:

```
# zypper ar http://theqvd.com/packages/sles/12SP1/stable QVD
# zypper ref
```

Para instalar todos los componentes del nodo servidor QVD en SLES, ejecute el siguiente comando:

```
# zypper install perl-QVD-HKD perl-QVD-L7R
```

Esto instalará todos los componentes del nodo qvd junto con cualquier dependencia. En general, recomendamos que la utilidad de administración CLI de QVD se instale en todos los nodos Servidor de QVD, ya que es común trabajar directamente desde estos sistemas y permite configurar QVD rápidamente. Más información sobre esta utilidad en Utilidad de Administración CLI de QVD.

Configuración básica

Como con la mayoría de los otros componentes de la infraestructura de QVD, cada nodo servidor QVD requiere acceso a la base de datos QVD. Deberá asegurarse de que el archivo de configuración del nodo servidor QVD está escrito correctamente para que el nodo servidor QVD funcione correctamente. Puede averiguar cómo hacerlo en el capítulo titulado Configuración básica de QVD.

A diferencia de la mayoría de los otros componentes, los nodos servidor de QVD requieren una entrada adicional dentro del archivo de configuración base de QVD para poder buscar rápidamente dentro del QVD-DB. Esta es una entrada de línea única que se debe anexar a su configuración, y que contiene el **nodename** que debe coincidir con el nombre que usted asignó a su nodo cuando lo registró en la QVD-DB, ya sea utilizando el WAT o mediante la utilidad de administración CLI de QVD. En general, le recomendamos que nombre sus nodos utilizando el nombre de host del sistema en el que se están ejecutando.

Esto se puede hacer rápidamente así:

```
echo "nodename=`hostname`" >> /etc/qvd/node.conf
```

Requisitos de red

Los nodos servidor QVD hacen uso de un puente de red y de interfaces de red virtuales para facilitar interfaces de red a cada una de las máquinas virtuales que se ejecutan en el nodo. Con el fin de proporcionar direcciones IP a máquinas virtuales, QVD también ejecuta un servidor DHCP que asignará las direcciones IP dentro del rango de la red virtual a los hosts virtuales a medida que se inician. Por lo tanto es muy importante que elija un rango de red que sea poco probable que entre en conflicto con cualquiera de sus otras infraestructuras existentes para este fin.

Note
Los servicios que se ejecutan en sistemas de la misma red IP pueden verse afectados por QVD o cualquiera de las máquinas virtuales que se ejecutan en QVD.

Hay una serie de pasos de configuración que puede ser necesario realizar manualmente para configurar correctamente la red para un nodo servidor QVD. A menudo hay otras maneras de lograr una configuración de red apropiada, por lo que los proporcionamos sólo como directrices.

Establecer dnsmasq para ser controlado por QVD

QVD utiliza dnsmasq como servidor DHCP y DNS para las máquinas virtuales que se ejecutan en un nodo. Para funcionar correctamente, dnsmasq necesita ser ejecutado por el proceso HKD.

En primer lugar, compruebe que dnsmasq está instalado. En Ubuntu, ejecute los siguientes comandos y compruebe el estado:

```
# dpkg -s dnsmasq
```

En SUSE:

```
# rpm -q dnsmasq
```

Si no está instalado, hágalo ahora utilizando su gestor de paquetes, ya sea `apt-get install dnsmasq`, o `zypper install dnsmasq`.

De forma predeterminada, el paquete Ubuntu inicia el proceso que se ejecuta como un demonio en segundo plano, así que debe evitar que comience automáticamente. Esto se hace con los siguientes comandos en Ubuntu:

```
# service dnsmasq stop
# sed -i s/ENABLED=1/ENABLED=0/ /etc/default/dnsmasq
```

En SLES dnsmasq se gestiona bajo el comando chkconfig y se deshabilita de forma predeterminada, por lo que no debería necesito hacer nada. Sin embargo, en caso de que dnsmasq se haya habilitado o por asegurarse, puede comprobar que está desactivado ejecutando el siguiente comando como root:

```
# chkconfig dnsmasq off
```

Note
Este paso es esencial para que QVD funcione utilizando la virtualización KVM. Para LXC es posible especificar si se debe o no hacer uso de DHCP para configurar la red dentro de sus máquinas virtuales.

Configurar el reenvío IP

La redirección IP (IP Forwarding) es necesaria para encaminar a los clientes a la ubicación correcta. Puede habilitarla rápidamente ejecutando el siguiente comando.

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Desafortunadamente, al reiniciar el sistema host, este cambio se perderá. Para que sea permanente, puede editar `/etc/sysctl.conf` y descomentar la línea:

```
net.ipv4.ip_forward=1
```

Puede obligar a sysctl a recargar su configuración después de haber editado este archivo ejecutando:

```
# sysctl -p
```

Configurar un puente de red

Hay varias formas de configurar el puente de red y el enrutamiento apropiado para asegurarse de que un cliente QVD se enruta a la máquina virtual correcta.

El método más fácil es configurar una interfaz de red estática y un conjunto de reglas de enrutamiento **iptables** para realizar el NAT necesario para traducir las direcciones IP entre su red real y virtual.

Para configurar su red en Ubuntu, edite el archivo `/etc/network/interfaces` y agregue las líneas siguientes:

```
auto qvdnet0
iface qvdnet0 inet static
  pre-up brctl addbr qvdnet0
  pre-up iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 192.168.0.2
  pre-up iptables -t nat -A PREROUTING -d 192.168.0.2 -p tcp --dport 8443 -j DNAT --to- ↩
      destination 10.3.15.1
  post-down brctl delbr qvdnet0
  address 10.3.15.1
  netmask 255.255.255.0
```

Es importante señalar que en el ejemplo anterior necesitará cambiar la dirección IP **192.168.0.2** a la dirección IP de la interfaz de red a la que desea que sus clientes se conecten. En el ejemplo de arriba usamos el rango **10.3.15.0/24** para la red virtual utilizada por QVD. Este rango debe ser único dentro de su infraestructura y debe dedicarse al uso de QVD, de modo que los servicios que arranquen en QVD no interfieran en otros sistemas dentro de su red.

Si bien hay otros enfoques más limpios para configurar su red, estos a veces tienen problemas con interfaces de red particulares tales como WIFI. El enfoque mencionado anteriormente debería funcionar para la mayoría de los sistemas.

Una vez que haya escrito la configuración de red en un archivo, debe levantar la interfaz de puente de red.

```
# ifup qvdnet0
```

Configurar un puente de red en SLES

Si utiliza SLES, le recomendamos que utilice Yast2 para configurar su puente de red.

Abra Yast y vaya a Dispositivos de red → Configuración de red → Agregar.

Defina las opciones siguientes:

- Tipo de dispositivo:"bridge"
- Nombre de configuración:"0" (La cadena será un sufijo de br, así que aquí el nombre del puente será br0).
- Deje todos los campos restantes como están.
- Seleccione Siguiente.
- Seleccione el dispositivo físico que desea que forme parte del puente. (Marque eth0 por ejemplo).
- Seleccione Siguiente.
- Seleccione Aceptar.
- El dispositivo de red se configurará automáticamente en unos segundos.

Configurar QVD para su red

Para que QVD administre correctamente la configuración de la máquina virtual y el enrutamiento subsiguiente, necesitará cambiar algunos ajustes de configuración dentro de QVD-DB. Se recomienda que utilice la Utilidad de Administración CLI de QVD para hacer esto. También puede utilizar el WAT si ya lo ha configurado.

Estos ajustes se utilizan para proporcionar un entorno de red dedicado para las Máquinas virtuales. Debe utilizar direcciones IP y rangos de red que no entren en conflicto con su infraestructura de red existente. En el ejemplo a continuación se utiliza el rango **10.3.15.0/24** para la red virtual utilizada por QVD.

```
# qa4 config set tenant_id=-1,key=vm.network.ip.start,value=10.3.15.50
# qa4 config set tenant_id=-1,key=vm.network.netmask,value=24
# qa4 config set tenant_id=-1,key=vm.network.gateway,value=10.3.15.1
# qa4 config set tenant_id=-1,key=vm.network.dns_server,value=10.3.15.254
# qa4 config set tenant_id=-1,key=vm.network.bridge,value=qvdnet0
```

Important
Si está ejecutando **AppArmor** en su máquina host, podrá comprobar que evita que las máquinas host accedan a Internet. Tenemos un perfil de AppArmor para QVD que está disponible en los paquetes. En cualquier caso, también es posible deshabilitar AppArmor con `/etc/init.d/apparmor teardown`. Esto detendrá AppArmor y permitirá ejecutar normalmente QVD. Si esto es inaceptable en el entorno en producción, utilice el perfil referido y pida ayuda al equipo de soporte QVD si es necesario.

Estos ajustes se describen con más detalle en la sección del **Manual de administración de QVD** titulado **Virtual Machine Options** en el capítulo **Configuración básica de QVD**.

Configuración de SSL

El servidor QVD necesita un certificado x509 y una clave privada para proteger las conexiones de red. Para una instalación en producción debe utilizar un certificado expedido por una autoridad de certificación reconocida, como Verisign o Thawte. Para las pruebas puede utilizar un certificado autofirmado. Proporcionamos instrucciones sobre creación de certificados autofirmados en la *Guía de instalación de QVD*. Si tiene un certificado firmado por un tercero, puede registrarlo con QVD utilizando la utilidad de administración de CLI de QVD:

```
# qa4 config ssl key=/path/to/private/key.pem,cert=/path/to/server/certificate.pem
```

Chapter 4

La API de QVD

En la versión 4 de **QVD**, se ha creado una interfaz *REST* que permite controlar la solución. La interfaz se ha creado con el propósito de facilitar la integración del productor con terceros, aunque a nivel interno se han rediseñado tanto el WAT como la línea de comandos para hacer uso de la misma.

La API es por tanto ahora el estándar de control de la solución, y tiene por tanto documentación propia. Esta documentación no es necesaria para administrar la plataforma, y a priori solo tiene interés para integradores y programadores.

Los comandos de arranque y parada son:

```
# /etc/init.d/qvd-api start
# /etc/init.d/qvd-api stop
```

El log está en:

```
/var/log/qvd/qvd-api.log
```

Chapter 5

Herramienta de administración web de QVD

En la versión 4 de **QVD**, la herramienta de administración web (WAT) ha crecido exponencialmente en capacidades y funcionalidad. Tanto así, que desde esta versión en adelante tiene su propio manual de administración. Por favor, vea dicho manual para obtener información sobre la misma.

Tenga en cuenta que dicho manual es complementario a este, y probablemente deba leer el presente documento antes de poder comprender por completo el del WAT.

Note
Como decíamos en el apartado anterior, el WAT depende de la API, y no puede funcionar sin esta.

Chapter 6

Utilidad de administración CLI de QVD

Important
Desde la versión 4.0, QVD tiene una nueva herramienta de administración: `qa4`. Esta herramienta tiene una sintaxis distinta a la anterior, y se documenta aquí. La herramienta anterior se ha mantenido por razones de retro-compatibilidad y está documentada en el anexo Utilidad de administración Legacy CLI de QVD.

La utilidad QVD de administración en línea de comandos es un script perl que puede interactuar con la QVD-DB a través de la API (desde la versión 4 de QVD) para realizar una amplia gama de operaciones dentro de la infraestructura QVD. Se puede utilizar como una alternativa a la herramienta de administración web QVD (QVD-WAT) y puede ser instalada en cualquier sistema con acceso a la API.

Instalación y configuración de la utilidad de administración de CLI de QVD

En cualquiera de los sistemas en los que va a instalar la Administración de CLI de QVD , deberá agregar el repositorio QVD a las fuentes de apt.

En primer lugar, agregue la clave pública de los paquetes QVD a sus claves de confianza (como root):

```
# wget -qO - https://www.theqvd.com/packages/key/public.key | sudo apt-key add -
```

Ahora, agregue el repositorio:

```
/etc/apt/sources.list.d/qvd.list
# apt-get update
```

Important

Para instalar la Utilidad de administración CLI de QVD, ejecute el siguiente comando:

```
# apt-get install perl-qvd-admin4
```

El proceso es similar para SLES.

En primer lugar, agregue la clave pública de los paquetes QVD a sus claves de confianza (como root):

```
# rpm --import https://www.theqvd.com/packages/key/public.key
```

Ahora, agregue el repositorio:

```
# zypper ar http://theqvd.com/packages/sles/12SP1/stable QVD
# zypper ref
```

Para instalar la utilidad de administración CLI de QVD, ejecute el siguiente comando:

```
# zypper install perl-QVD-Admin4
```

La utilidad de administración de QVD requiere acceso a la API de QVD. Debe asegurarse de que el fichero `qa.conf` esté correctamente configurado, tal y como se detalla en: Configuración básica de QVD.

Listado de comandos QVD CLI

La utilidad de administración CLI de QVD proporciona un gran conjunto de funciones administrativas que pueden ser utilizadas para controlar componentes y elementos que están involucrados en el entorno QVD.

Puede obtener una lista completa de las funciones mediante el parámetro *usage*:

```
# qa4 usage

================================================================================
                            AVAILABLE COMMANDS
================================================================================

   For a specific explanation of the following commands run:
   usage <COMMAND>
   i.e. usage login

== CLI GENERAL MANAGEMENT COMMANDS

   usage (retrieves instructions about the usage of the app)

   login (Intended to log in as a QVD administrator)

   logout (Intended to log out)

   password (Intended to change current QVD administrator password)

   block (Intended to change current QVD administrator pagination block)

   version (Retrieves information about the QVD version the app is connected to)

   log (retrieves log entries about the QVD server activity)

== QVD OBJECTS COMMANDS

    vm (Intended to QVD virtual machines management)

    user (Intended to QVD users management)

    host (Intended to QVD hosts management)

    osf (Intended to QVD OSFs management)

    di (Intended to QVD disk images management)
```

```
    tenant (Intended to QVD tenants management)

    role (Intended to QVD roles management)

    acl (Intended to QVD acls management)

    admin (Intended to QVD administrators management)

    config (Intended to QVD configuration management)
```

Cualquiera de los comandos presentados anteriormente puede ser invocado con `usage` para obtener una descripción más detallada de la sintaxis.

Por ejemplo:

```
# qa4 usage vm

================================================================================================================

                                                VM COMMAND USAGE
================================================================================================================


== CREATING A NEW VM

  vm new <ARGUMENTS>

  For example:
  vm new name=myvm, user=myuser, osf=myosf (Creates a VM with user 'myuser', osf 'myosf'  ↩
      and name 'myvm')

== GETTING VMs

  vm get
  vm <FILTERS> get
  vm <FILTERS> get <FIELDS TO RETRIEVE>

  For example:
  vm get (retrieves default fields of all VMs)
  vm name=myvm get (retrieves default fields of all VMs with name 'myvm')
  vm name=myvm get name, id, ip (retrieves 'name', 'id' and 'ip' of VMs with name 'myvm')

  Ordering:

  vm ... order <ORDER CRITERIA>
  vm ... order <ORDER DIRECTION> <ORDER CRITERIA>

  For example:
  vm get order name (Ordering by 'name' in default ascendent order)
  vm get order asc name, id (Ordering by 'name' and 'id' in ascendent order)
  vm get order desc name, id (Ordering by 'name' and 'id' in descendent order)

== UPDATING VMs

  vm set <ARGUMENTS>
  vm <FILTERS> set <ARGUMENTS>

  For example:
  vm set di_tag=default (Sets new value for di_tag in all VMs)
  vm name=myvm set name=yourvm, di_tag=default (Sets new values for name and di_tag in VM  ↩
      with name myvm)
```

```
  Adding custom properties:

  vm <FILTERS> set property key=value
  vm <FILTERS> set property key=value, key=value, ...

  For example:
  vm set property mykey=myvalue (Sets property mykey in all VMs)
  vm name=myvm set property mykey=myvalue, yourkey=yourvalue (Sets properties mykey and  ↩
      yourkey in VM with name myvm)

  Deleting custom properties:

  vm <FILTERS> del property key
  vm <FILTERS> del property key, key, ...

  For example:
  vm del property mykey (Deletes property mykey in all VMs)
  vm name=myvm del property mykey, yourkey (Deletes properties mykey and yourkey in VM with ↩
       name myvm)

  Blocking/Unblocking VMs

  vm <FILTERS> block
  vm <FILTERS> unblock

  For example:
  vm block (Blocks all VMs)
  vm name=myvm block (Blocks VM with name myvm)

== REMOVING VMs

  vm del
  vm <FILTERS> del

  For example:
  vm del (Removes all VMs)
  vm name=myvm del (Removes VM with name myvm)

== EXECUTING VMs

  vm <FILTERS> start
  vm <FILTERS> stop
  vm <FILTERS> disconnect

  For example:
  vm start (Starts all VMs)
  vm name=myvm stop (Stop VM with name myvm)


================================================================================================

                                          DEFAULT FILTERS
================================================================================================


  The filter key 'name', and the operator '=' are consirered as defaults. So the following  ↩
      is a right
  syntax:

  <QVD OBJECT COMMAND> <QVD OBJECT NAME> <ACTION COMMAND>

  For example:
```

```
  vm myVM get (Equal to vm name=myVM get)
  vm myVM set name=yourVM (Equal to vm name=myVM set name=yourVM)

================================================================================================

                                          COMPLEX FILTERS
================================================================================================


  A filter is a key/value pair. Key and value can be related by means of different kinds of
  IDENTITY OPERATORS (=, >, <, etc.). Different operators allow different kinds of values
  (numeric, alphanumeric, arrays, ranges...). Morover, two or more filters can be joined by ↩
       means
  of LOGICAL OPERATORS (coordination, disjunction and negation operators).

  DIFFERENT IDENTITY OPERATORS:
  Supported operators:

  =  (equal)
  != (not equal)
  <  (less than)
  >  (greater that)
  <= (less or equal than)
  >= (greater or equal than)
  ~  (matches with a commodins expression: the SQL LIKE operator)

  For example:
  key1 = 1,
  key1 < 3,
  key1 > 3,
  key1 <= 3,
  key1 >= 3
  key1 = [1,2,3] (key1 must be in (1, 2, 3))
  key1 = [1:3] (key1 must be between 1 and 3)
  key1 = This_is_a_chain
  key1 = 'This is a chain' (A value with blanks must be quoted)
  key1 = "This is a chain" (A value with blanks must be quoted)
  key1 ~ %s_is_a_ch% (key1 must match the SQL commodins expression %s_is_a_ch%)
  key1 ~ '%s is a ch%' (key1 must match the SQL commodins expression %s_is_a_ch%)
  key1 ~ "%s is a ch%" (key1 must match the SQL commodins expression %s_is_a_ch%)

  LOGICAL OPERATORS
  Supported operators

  , (the AND operator)
  ; (the OR operator)
  ! (the NOT operator)

  (These operators have left precedence. In order to override this behaviour you must
   grup filters with '(' and ')')

  For example:
  key1=value, key2=value, key3=value (key1 AND key2 AND key3)
  (key1=value; key2=value), key3=value ((key1 OR key2) AND key3))
  !key1=value (This expresion means: NOT key1)
  !key1=value, key2=value, key3=value (NOT ( key1 AND key2 AND key3))
  (! key1=value), key2=value, key3=value ((NOT key1) AND key2 AND key3))

================================================================================================
```

Uso de filtros para controlar operaciones

Muchas de las operaciones disponibles a través de la utilidad de administración CLI de QVD también admiten filtros para limitar una acción a un elemento o entidad particular. Los filtros son esencialmente coincidencias con los elementos dentro de las columnas de la tabla, tal y como proporcionaría una consulta SQL estándar en la declaración *WHERE*. Los filtros aceptan comparadores, comodines y operadores lógicos.

Si por ejemplo, quisiéramos obtener máquinas con id mayor a 12000 pertenecientes al tenant qvd_test:

```
# qa4 vm "id>12000,tenant=qvd_test" get
.-------+----------+------+---------+-----------+------+--------------------------------------+----
| id    | tenant   | name | blocked | user      | host | di  ↩
                                     | ip          | ip_in_use | di_in_use | state   |  ↩
   user_state   |
+-------+----------+------+---------+-----------+------+--------------------------------------+----
| 12604 | qvd_test | Java |       0 | test_user |      | 14000-Desktop_Development-Java.tar ↩
   .gz | 10.10.53.79 |           |           | stopped | disconnected |
'-------+----------+------+---------+-----------+------+--------------------------------------+----

Total: 1
```

Cuando se utiliza la utilidad de administración CLI para ver el estado de la VM, es común utilizar filtros para ver máquinas virtuales en un estado concreto o pertenecientes a un usuario particular o donde una máquina virtual tiene un *user_state* definido.

El siguiente ejemplo le mostrará cómo encadenar juntos una serie de filtros para ver todas las máquinas virtuales que pertenecen a usuarios con nombres de usuario que comienzan por *qvd*, donde además la máquina virtual está parada y el usuario está desconectado:

```
# qa4 vm "user~qvd%,state=stopped,user_state=disconnected" get
.-------+---------+------------------+---------+----------+------+------------------------------+--
| id    | tenant  | name             | blocked | user     | host | di  ↩
                              | ip           | ip_in_use | di_in_use | state   | user_state ↩
      |
+-------+---------+------------------+---------+----------+------+------------------------------+--
| 10800 | qvd_43  | Desktop_Standard |       0 | qvd-test |      | 10000-Desktop_Standard. ↩
   tar.gz | 10.13.25.100 |          |           | stopped | disconnected |
| 10900 | qvd_75  | Desktop_Standard |       0 | qvd-test |      | 10000-Desktop_Standard. ↩
   tar.gz | 10.13.25.180 |          |           | stopped | disconnected |
| 10500 | qvd_264 | Desktop_Standard |       0 | qvd-test |      | 10000-Desktop_Standard. ↩
   tar.gz | 10.13.23.120 |          |           | stopped | disconnected |
| 10100 | qvd_115 | Desktop_Standard |       0 | qvd-test |      | 10000-Desktop_Standard. ↩
   tar.gz | 10.13.24.220 |          |           | stopped | disconnected |
| 10200 | qvd_155 | Desktop_Standard |       0 | qvd-test |      | 10000-Desktop_Standard. ↩
   tar.gz | 10.13.24.160 |          |           | stopped | disconnected |
| 10102 | qvd_77  | Desktop_Standard |       0 | qvd-test |      | 10000-Desktop_Standard. ↩
   tar.gz | 10.13.25.30  |          |           | stopped | disconnected |
'-------+---------+------------------+---------+----------+------+------------------------------+--

Total: 6
```

Operaciones administrativas básicas

En esta sección examinaremos algunas de las tareas administrativas más comunes para las que se utiliza la utilidad de administración CLI de QVD.

Cambiar los ajustes de configuración de QVD

QVD tiene una amplia gama de ajustes de configuración muy específicos que controlan varios componentes dentro de la infraestructura. Discutimos algunos de ellos aquí.

Para cambiar un parámetro de configuración de QVD mediante la utilidad de administración CLI de QVD, puede hacer lo siguiente:

```
# qa4 config set tenant_id=-1,key=myproperty,value="this is a value"
```

También es posible obtener todos los ajustes de configuración actuales de la base de datos y enumerarlos:

```
# qa4 config get
```

Por último, es posible devolver una configuración a su valor original:

```
# qa4 config set tenant_id=-1,key=qvd.prop default
```

Añadir un nodo servidor QVD

Es común utilizar la utilidad de administración CLI de QVD para agregar nuevos nodos servidor QVD a la base de datos de QVD. Esto se puede hacer muy rápidamente desde la línea de comandos con la siguiente directriz:

```
# qa4 host new name=myhost, address=10.3.15.1
```

La eliminación de un nodo de servidor QVD es igual de sencilla:

```
# qa4 host name=myhost del
```

Añadir un OSF

Puede agregar fácilmente un OSF a QVD usando la utilidad de Administración CLI de QVD:

```
# qa4 osf new name=myOSF
```

Sólo hay un valor obligatorio para agregar un OSF, que es **name**. Si los otros parámetros se dejan sin especificar, sus valores por defecto se utilizan en su lugar. Estos son:

- * Memoria * = 256
- * Use_overlay * = y
- * User_storage_size * = undef (sin límite para el almacenamiento de usuarios)

Puede obtener una lista de OSF disponibles actualmente haciendo lo siguiente:

```
# qa4 osf get
```

Añadir un DI

Utilizando la utilidad de administración CLI de QVD, puede adjuntar una imagen de disco (DI) a cualquier OSF existente dentro del sistema. Este proceso puede llevar tiempo, ya que además de actualizar la base de datos el archivo de imagen de disco real es copiado en el directorio `storage/images` dentro del almacenamiento compartido.

Al adjuntar un DI a un OSF particular, mantenemos separado el disco real de la imagen que se servirá a los usuarios finales. Esto significa que puede realizar cambios en la imagen de disco y simplemente actualizar el OSF, de modo que cuando un usuario vuelve a conectar la imagen es automáticamente actualizada sin que el usuario experimente ninguna discontinuidad en el servicio.

```
# qa4 di new disk_image=test-image.tar.gz, osf=retest
```

Ambos **disk_image** y **osf** son obligatorios para agregar un DI. Cuando se agrega la DI, la imagen especificada en la ruta se copia en el área de almacenamiento de sólo lectura configurada para almacenar DIs activas (normalmente `/var/lib/qvd/storage/images`).

Puede obtener una lista de imágenes disponibles actualmente haciendo lo siguiente:

```
# qa4 di get
```

Etiquetar DI

Los DI pueden ser etiquetados con cadenas arbitrarias a voluntad. Para etiquetar un DI como predeterminado, use el comando **di tag**:

```
# qa4 di <filtro> tag <tag1>,<tag2>,...
# qa4 di disk_image=mydi tag default, head, mytag
```

Puede etiquetar DIs con cualquier cadena, no sólo **default** o **head**. Esto le permite usar nombres significativos para las etiquetas, por ejemplo "software_bug_fixed", para su uso dentro del campo **DI Tag** de las máquinas virtuales.

Las etiquetas son útiles, ya que permiten adjuntar una nueva versión de una imagen de disco a un OSF sin afectar a nadie que esté usando la imagen actual o predeterminada para un OSF. Esto le permite implementar un cambio y migrar determinadas máquinas virtuales utilizando un OSF distinto para la nueva imagen mientras lo prueba. Si la imagen falla por alguna razón o no cumple con sus requisitos, es sencillo volver a la imagen predeterminada y permitir que sus usuarios continúen trabajando mientras hace correcciones.

Seleccionando la etiqueta DI que las máquinas virtuales utilizarán

Con el fin de decirle a una máquina virtual que debe utilizar una DI que tiene una etiqueta específica, editamos la VM Para cambiar su campo `di_tag`. Así que si por ejemplo acabamos de corregir un error software en un DI y establecemos la etiqueta "software_bug_fixed", podemos usar ese DI en una VM usando el siguiente comando:

```
# qa4 vm set di_tag=default
# qa4 vm name=myvm set di_tag=default
```

En el siguiente arranque de la VM myvm, utilizará el DI con esta etiqueta.

Agregar y eliminar usuarios

Es común utilizar la utilidad de administración CLI de QVD para agregar y quitar usuarios rápidamente.

```
# qa4 user new login=peter,password=s3cr3t
# qa4 user login=guest3 del
```

También puede enumerar todos los usuarios de QVD mediante la opción de lista:

```
# qa4 user get
```

Tenga en cuenta que, como se explica en la Guía de instalación, no es aconsejable crear un usuario cuyo nombre de usuario ya exista en cualquier imagen de disco.

Restablecimiento de una contraseña de usuario

Puede cambiar la contraseña de un usuario mediante la Utilidad de administración CLI de QVD:

```
# qa4 user name=guest set passwd=newpassword
```

En el ejemplo anterior, estamos cambiando la contraseña para el usuario de inicio de sesión *guest*. Se le pedirá que proporcione una nueva contraseña.

Añadir y eliminar máquinas virtuales

Añadir y eliminar máquinas virtuales mediante la utilidad de administración CLI de QVD es fácil. Puede especificar el usuario y el OSF por ID o por nombre:

```
# qa4 vm new name=GuestVM,user_id=1 osf_id=1
# qa4 vm new name=GuestVM,user=peter,osf=myOFS
```

Puede eliminar fácilmente una máquina virtual utilizando el siguiente comando:

```
# qa4 vm "name=GuestVM" del
```

Inicio y detención de máquinas virtuales

La utilidad de administración CLI de QVD se puede utilizar para iniciar y detener máquinas virtuales. Si no se especifica con un filtro en particular, la acción será iniciar o detener todas las maquinas virtuales. Normalmente se ejecuta este comando especificando un filtro para identificar la máquina virtual real que desea iniciar o detener. Ejemplos:

```
# qa4 vm "user~guest%" stop
# qa4 vm id=1 start
```

La política de balanceo determina el nodo donde arrancará la VM. is started.

Bloqueo y desbloqueo de máquinas virtuales

Las Máquinas Virtuales se pueden marcar como "bloqueadas". Cuando estén en este estado, la aplicación QVD Client no podrá conectarse a la Máquina Virtual. Esta tarea puede ser ejecutada por un administrador para realizar una tarea administrativa o puede tener lugar cuando el HKD no sea capaz de gestionar correctamente una máquina virtual. Los comandos siguientes se pueden utilizar para marcar una máquina virtual como "bloqueada" o se puede utilizar para desbloquear una máquina virtual que se ha establecido en este estado.

```
# qa4 vm "id=2" block
# qa4 vm "name=GuestVM "unblock
```

Consulte *bloqueo y desbloqueo de la máquina virtual* en el manual del WAT para obtener más información sobre cómo configurar este estado.

Solución de problemas de máquinas virtuales

Note
En esta versión de la herramienta de administración, se han deshabilitado los viejos métodos de acceso por consola o vnc a las máquinas virtuales. Estos métodos están en desuso y no tienen interés frente a las nuevas herramientas que provee el WAT.

Configuración de propiedades personalizadas para una máquina virtual

QVD incluye la opción de configurar propiedades personalizadas para una máquina virtual que puede establecer y recuperar mediante la utilidad de administración CLI de QVD. Esto es útil si necesita escribir sus propios comportamientos o desea aprovecharse de los `hooks` VMA (Vea el apartado al respecto).

Las propiedades personalizadas se utilizan a menudo cuando usted escribe sus propios plugins para el L7R, como los módulos de autenticación o balanceo de carga.

Las propiedades personalizadas son compatibles con los parámetros de configuración: **host**, **user** y **vm**

Para agregar una propiedad personalizada, puede usar el comando $_{\text{propset}}$:

```
# qa4 user name=myuser set property mykey=myvalue, yourkey=yourvalue
```

Se puede obtener el contenido de todas las propiedades personalizadas que se han establecido para un parámetro de configuración utilizando el comando:

```
# qa4 user get property
```

Finalmente, puede eliminar una propiedad personalizada mediante el comando $_{\text{propdel}}$:

```
# qa4 user name=juan property del beverage
```

Chapter 7

Cliente QVD GUI

El cliente QVD está disponible para plataformas Microsoft Windows, Linux y Mac OS X. El cliente está disponible en inglés y español y por defecto se ajustará a la configuración regional del sistema.

Instalación del cliente de Windows

Puede descargar el instalador del software cliente de QVD desde:

http://theqvd.com/download#_windows

Una vez que haya terminado de descargar el instalador, ejecútelo como un archivo ejecutable normal y siga el asistente durante el proceso de instalación.

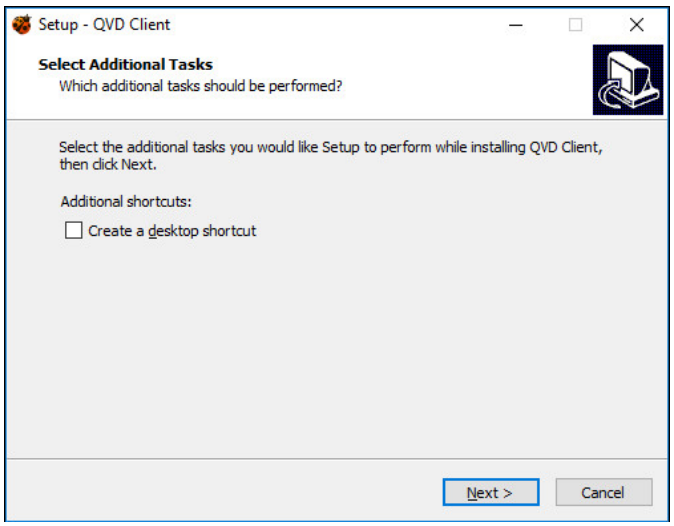


Figure 7.1: El Asistente de instalación de Windows QVD Client

Una vez que haya terminado la instalación, puede ejecutar el cliente desde el acceso directo en el escritorio de Windows (si ha seleccionado agregar el acceso directo) o desde el menú QVD en el menú Aplicaciones. Esto abrirá el cliente para que esté listo para conectarse.

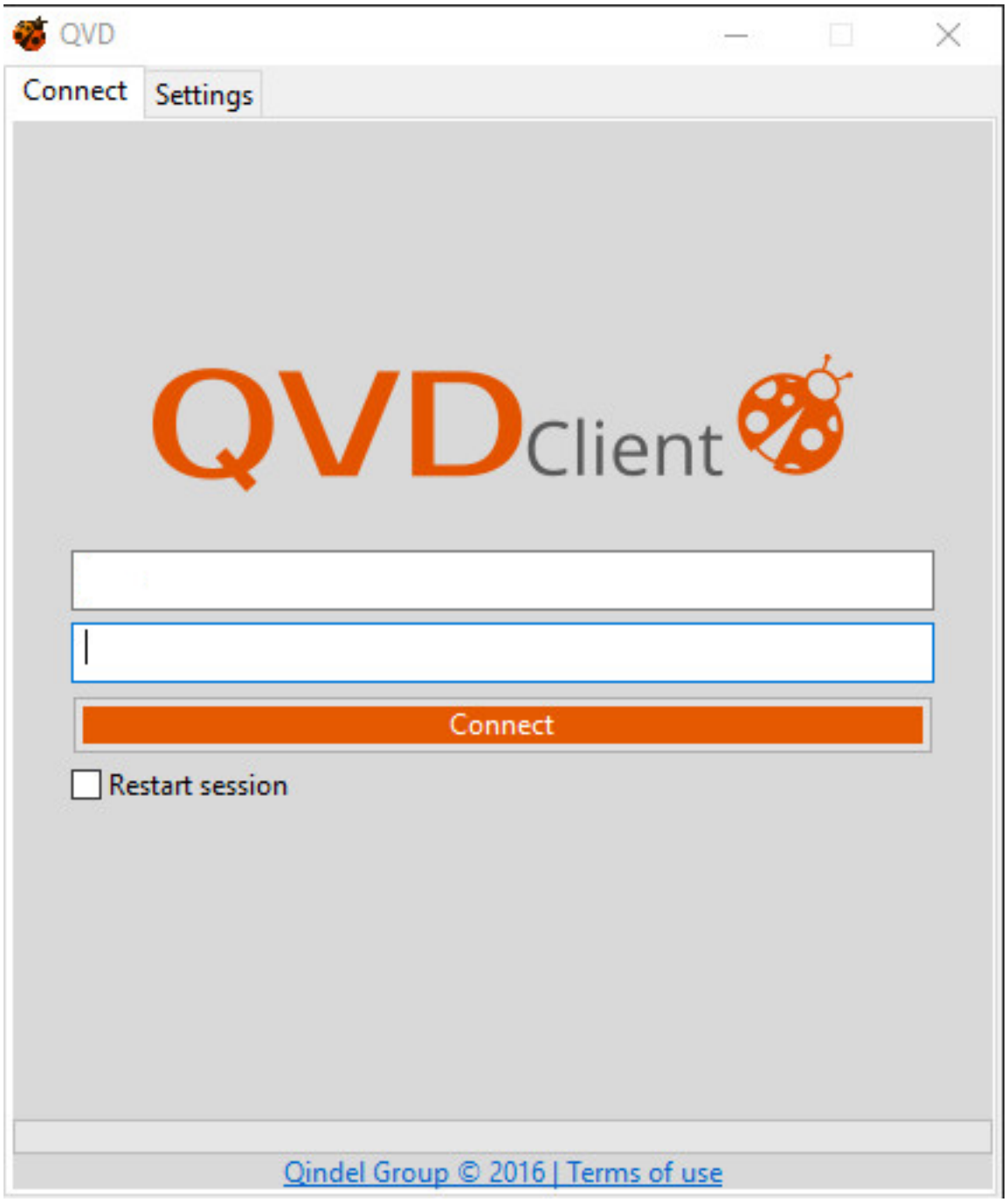


Figure 7.2: El cliente de Windows QVD

Instalación del cliente Mac OS X

Puede descargar el paquete de cliente QVD de:

http://theqvd.com/download#_os_x

El paquete instalará el cliente QVD en el directorio `Applications`. Para ejecutar el cliente, haga doble clic en el icono de la mariquita o a través del sistema de búsqueda nativo `spotlight` pulsando Comando + Barra espaciadora y tecleando `qvd`.

Figure 7.3: El cliente de Mac OS X QVD

Instalación del cliente Linux

Instalar el cliente QVD en una plataforma Ubuntu Linux es un procedimiento sencillo.

En primer lugar, agregue la clave pública de los paquetes QVD a sus claves de confianza (como root):

```
# wget -qO - https://www.theqvd.com/packages/key/public.key | sudo apt-key add -
```

Ahora, agregue el repositorio:

```
/etc/apt/sources.list.d/qvd.list
# apt-get update
```

Important

Ahora podrá instalar el cliente con el siguiente comando.

```
# apt-get install perl-qvd-client
```

En SLES el proceso es similar.

En primer lugar, agregue la clave pública de los paquetes QVD a sus claves de confianza (como root):

```
# rpm --import https://www.theqvd.com/packages/key/public.key
```

Ahora, agregue el repositorio:

```
# zypper ar http://theqvd.com/packages/sles/12SP1/stable QVD
# zypper ref
```

Ahora podrá instalar el cliente con el comando zypper.

```
zypper install perl-QVD-Client
```

Dependiendo del entorno de escritorio, debería poder acceder al cliente dentro del menú "Aplicaciones", normalmente en el submenú "Internet". Como alternativa, puede ejecutar la GUI del cliente desde la consola utilizando el comando `qvd-client`.

Conexión a su escritorio virtual

Una vez que tenga el cliente GUI en ejecución, puede ingresar el **Nombre de usuario** para el usuario que creó en QVD, la **Contraseña** que configuró para el usuario, el **Servidor** nombre de host o dirección IP para el nodo del servidor QVD creado, y puede elegir el nivel de compresión de la conexión seleccionando **Tipo de conexión**.

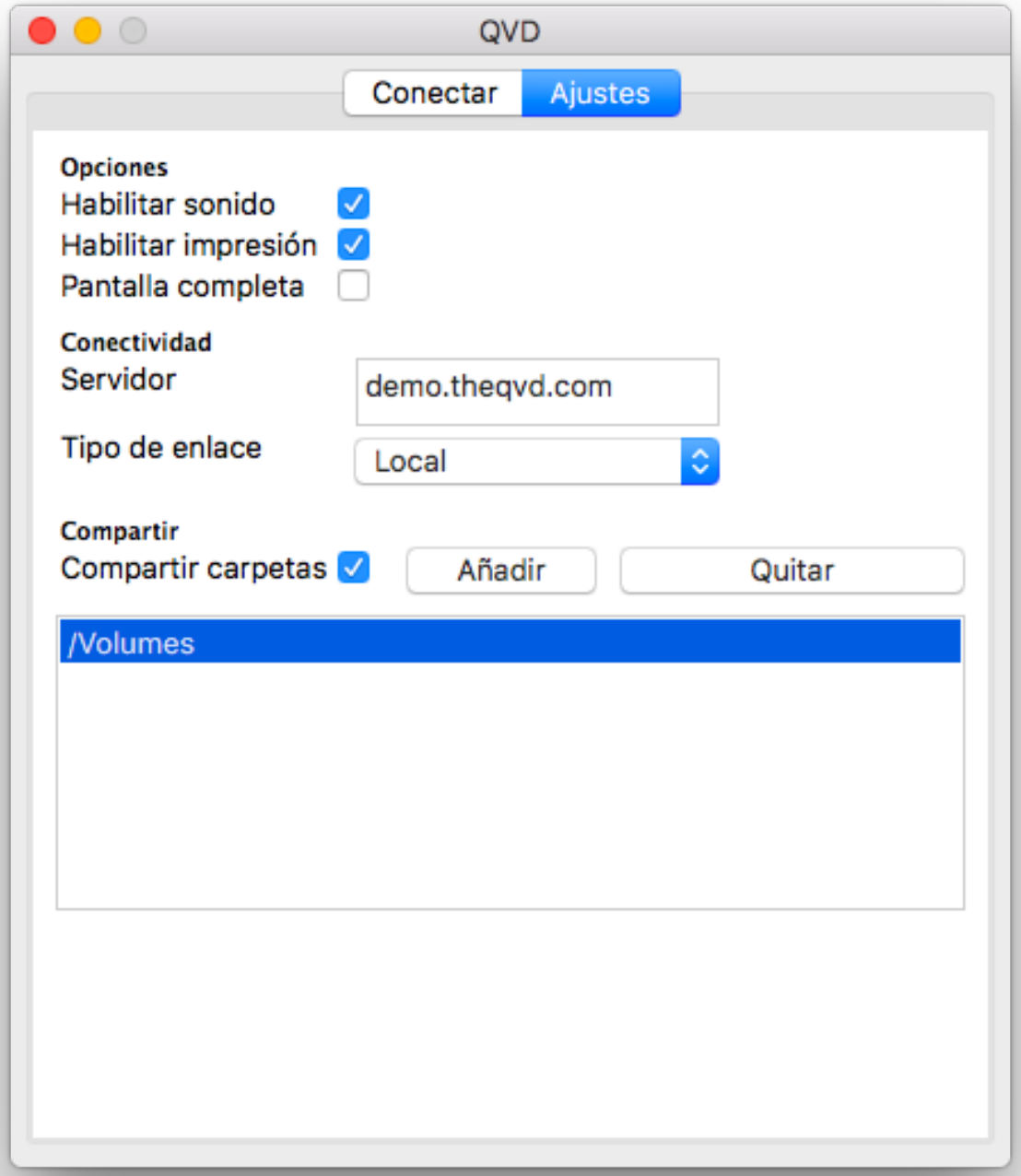


Figure 7.4: Ingrese los detalles de su conexión QVD en la pantalla de configuración.

Figure 7.5: Ingrese los detalles de su cuenta de usuario en la pantalla principal.

De forma predeterminada, el **Tipo de conexión** se establece en *Local*. Esta configuración es apropiada para las conexiones a través de una red de área local. También hay opciones para *ADSL*, que sería apropiado para cualquier conexión de banda ancha, y para *módem* que se puede utilizar en casos donde el ancho de banda está severamente limitado o deteriorado.

Cambiar el **tipo de conexión** aumentará la compresión utilizada para entregar el escritorio virtual a través de la conexión de red.

También aumenta la cantidad de caché que realiza el cliente para limitar la cantidad de refresco de pantalla que debe realizarse.

En general, el uso de la compresión pesada y el almacenamiento en caché todavía ofrecerá a sus usuarios la capacidad de trabajar cómodamente dentro de sus escritorios virtuales. Sin embargo, la calidad de la representación gráfica será un poco inferior.

Una vez que haya completado la introducción de los detalles de su conexión, simplemente haga clic en el botón etiquetado **Connect** y su escritorio virtual debe cargar.

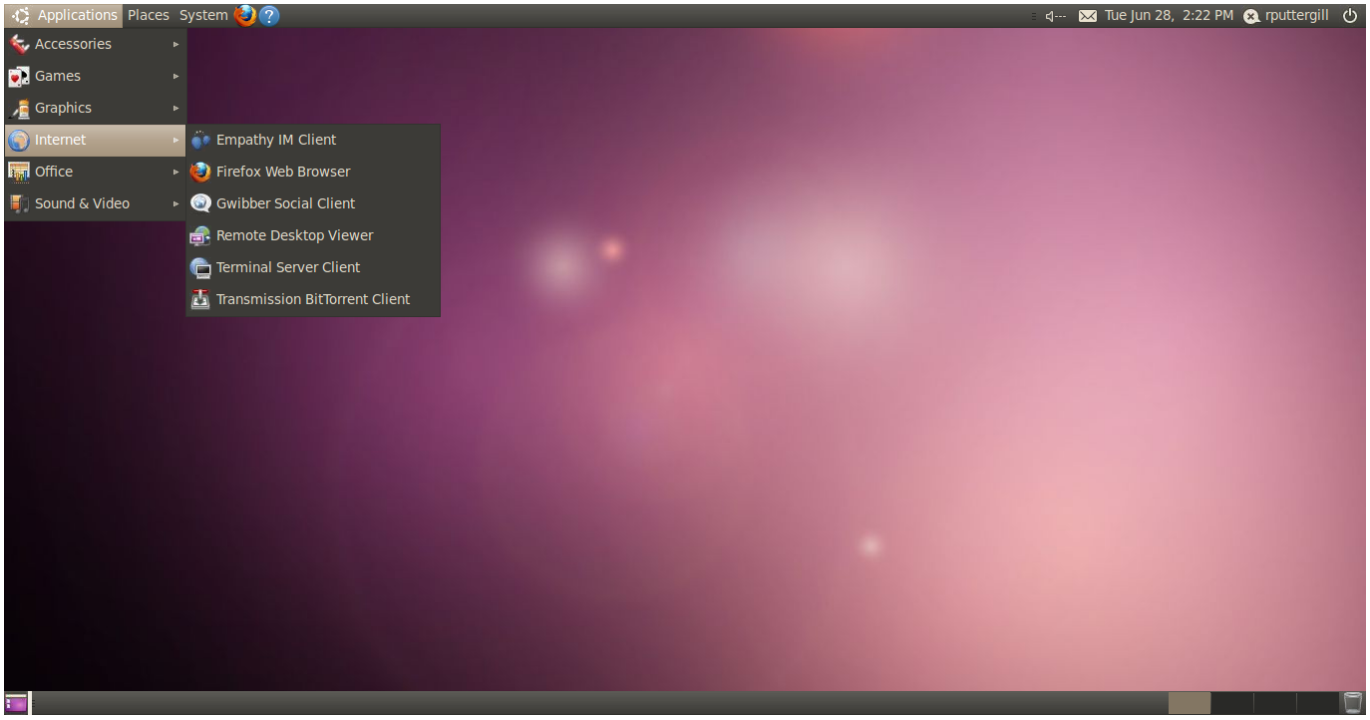


Figure 7.6: Un escritorio Gnome cargado bajo QVD

Carpetas compartidas del cliente QVD

QVD puede proporcionar acceso a carpetas locales en la máquina cliente del usuario desde el escritorio virtual.

Uso de carpetas compartidas

Para activar esta opción, márquela en la pestaña de configuración.

Cuando se activan las carpetas compartidas por primera vez, se redirecciona el directorio principal del usuario (**%USERPROFILE%** para clientes Windows y /home/%user% para Linux y Mac OS X).

Al activar la opción se muestra un listado de carpetas compartidas. Desde esta pantalla es posible eliminar las carpetas del listado y añadir otras.

Las carpetas aparecerán en el directorio personal del usuario en una subcarpeta denominada *redirect*. GNOME y otros escritorios Linux recientes también mostrarán un icono de acceso directo en el escritorio y en el administrador de archivos, dependiendo de la configuración.

Configuración adicional para el cliente QVD

El cliente QVD ofrece varias opciones de configuración que se pueden utilizar para ajustar el rendimiento y personalizar la experiencia del usuario. De forma predeterminada, estas configuraciones se encuentran en el archivo `client.conf`. Bajo Mac

OS X y Linux, este archivo se encuentra en el directorio personal del usuario en `~/.qvd/client.conf`, así como en `/etc/qvd`, aunque el archivo de usuario lo reemplaza si existe. En Windows, `client.conf` se encuentra dentro de `%APPDATA%\.qvd\client.conf`.

El cliente QVD GUI también ofrece un acceso conveniente a algunas de las configuraciones básicas del usuario en la pestaña "Configuración" en el inicio. La pestaña se puede activar o desactivar en `client.conf` - ver más abajo para más detalles.

Archivo de configuración de QVD

Puede especificar las siguientes opciones adicionales para controlar el software cliente en una estación de trabajo:

```
client.link = local
client.geometry = 1024x768
client.fullscreen = 1
client.slave.enable = 1
client.slave.command "/path/to/custom/slave-command"
client.audio.enable = 1
client.printing.enable = 1
client.host.port = 8443
client.host.name = loadbalancer.mydomain.com
client.user.name = guest
client.use_ssl = 1
client.force.host.name = loadbalancer.mydomain.com
client.force.link = local
client.remember_password = 0
client.show.remember_password = 0
client.show.settings = 1
client.usb.enable=
client.usb.share_list=139a:0608@3210bf331301,0f45:2421
client.file_sharing.enable=1
client.share.0=/home/usuario/Documentos
client.share.1=
```

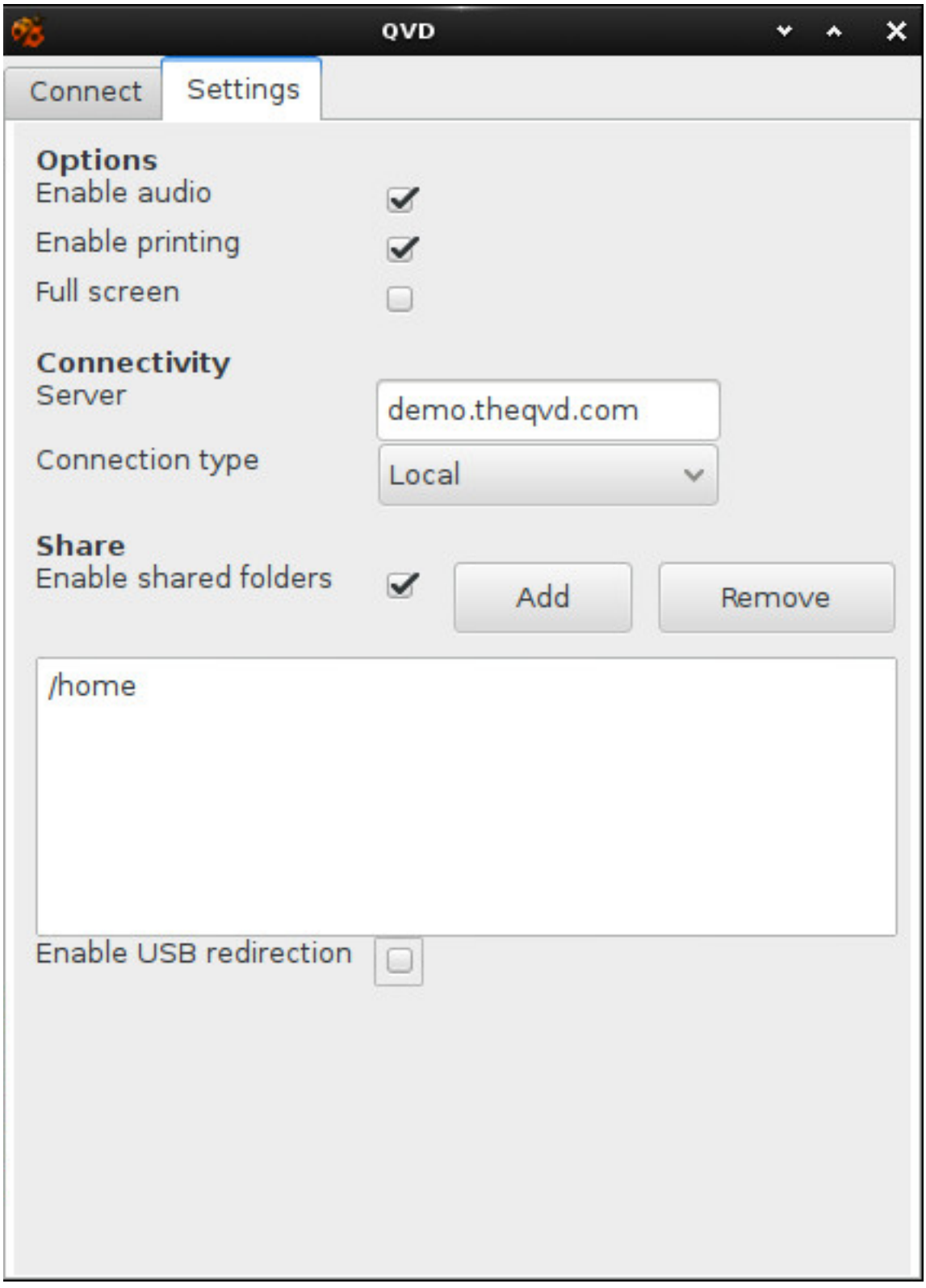
Los valores establecidos anteriormente son los valores por defecto.

- **client.link**: puede ser: modem, isdn, adsl, wan, lan, local o una especificación de ancho de banda (56k, 1m, 100m...)
- **client.geometry**: usado para informar del tamaño de la pantalla del cliente. Si no se usa, pasará a modo pantalla completa
- **client.fullscreen**: modo pantalla completa
- **client.slave.enable**: habilitar o no el slave channel, que gobierna las funciones de carpeta compartida e impresión
- **client.slave.command**: comando a ejecutar para el esclavo
- **client.audio.enable**: habilitar PulseAudio server en el cliente. La imagen de la máquina virtual del usuario debe soportar esta opción o no funcionará
- **client.printing.enable**: Habilitar las impresoras compartidas en el cliente. La imagen de la máquina virtual del usuario debe soportar esta opción o no funcionará
- **client.host.port**: Puerto del HKD al que debería conectarse el cliente
- **client.host.name**: Host del HKD o balanceador de carga al que debe conectarse el cliente
- **client.user.name**: Nombre de usuario para la autenticación (en modo multitenant el formato sería user@tenant)
- **client.use_ssl**: Activar o desactivar el uso de SSL en la comunicación con el servidor QVD. Activado de forma predeterminada
- **client.force.host.name**: Fuerza el host name en el cliente para que solo pueda conectarse a este host (imposibilita que el usuario via interfaz gráfica lo cambie)
- **client.force.link**: Lo mismo que el anterior pero para el tipo de enlace

- **client.show.remember_password**: Controla si el usuario tiene la opción de guardar su contraseña o no (*Recordar contraseña* se muestra dentro de la GUI)
- **client.remember_password**: Controla si se guarda o no la contraseña de usuario
- **client.show.settings**: mostrar o esconder la pestaña de configuración
- **client.usb.enable**: Habilita la compartición de dispositivos USB (solo en linux)
- **client.usb.share_list**: Listado de dispositivos compartidos
- **client.file_sharing.enable**: Habilita la compartición de carpetas
- **client.share.[número]**: Listado de carpetas compartidas (una línea por carpeta, el número comienza en cero y deben ser sucesivos)

Configuración de la GUI de QVD

La siguiente imagen muestra los ajustes actualmente disponibles en el cliente QVD.



- **Reiniciar VM actual** (en la pantalla principal): esto apaga la VM en ejecución a la que el usuario está intentando conectar. Útil para las situaciones en el que el usuario necesita actualizar a la última imagen que ha dispuesto el administrador y también para potencialmente permitir al usuario resolver por sí mismo problemas de inestabilidad en su VM (no puede conectar, no se muestra correctamente, etc. . .)
- **Activar audio**: activa el servidor PulseAudio en el cliente

- **Habilitar impresión**: permite compartir impresoras con la máquina virtual
- **Pantalla completa**: ejecuta el cliente en modo de pantalla completa
- **Compartir carpetas**: permite activar la compartición de carpetas, añadirlas y borrarlas
- **USB/IP**: Esta opción es solo para el cliente Linux. Permite utilizar la característica del kernel linux USB/IP. Con ella se puede compartir dispositivos USB locales con la máquina virtual remota, de manera que a todos los efectos sería como si físicamente el dispositivo estuviera conectado a ella. Esta característica del kernel es experimental, y por tanto nuestra funcionalidad también lo es. Algunos dispositivos, como unidades flash y similares tienen un comportamiento bastante correcto; mientras que otros más sensibles a retardos, como las webcams funcionan bastante mal.

Registros del cliente de QVD

Por defecto, el cliente QVD deja sus logs en `~/.qvd/qvd-client.log` en Mac OS X y Linux y `%APPDATA%\.qvd\qvd-client.log` en Windows. Puede cambiar los niveles de logging a uno de ALL, DEBUG, INFO, WARN, ERROR, FATAL, OFF en el archivo de configuración de la siguiente manera:

```
log.level = ALL
```

Binarios QVD

Además de los clientes QVD estándar, el equipo QVD ofrece varios binarios compilados para las plataformas anteriores, así como algunos otros como iOS, Android, Raspberry Pi y FreeBSD. Tenga en cuenta que estos binarios son experimentales y pueden ser incompletos. Con la excepción de los clientes Windows y OS X, los binarios se compilan de forma estática y deben poder ejecutarse sin bibliotecas adicionales.

- Linux Estos deben ejecutarse en cualquier distribución reciente sin bibliotecas adicionales, simplemente dele permisos de ejecución y ejecute.
  - qvdclient_x86_64 (64 bit client)
  - qvdclient_i386 (32 bit client)
- FreeBSD Al igual que con el cliente Linux, dele permisos de ejecución y ejecute.
  - qvdclient_freebsd-amd64 (64 bit client)
- Mac OS X and IOS clients.
  El cliente OS X necesita libstdc++ y libSystem, que debería estar disponible en versiones recientes de OS X.
  - qvdclient_x86_64-apple-darwin11 (64 bit client)
    Los binarios de IOS necesitarán Cydia y un servidor X.
  - qvdclient (multi arch IOS client)
  - qvdclient_armv7-apple-darwin11 (para dispositivos IOS más viejos, aunque debería funcionar en los nuevos)
  - qvdclient_armv7s-apple-darwin11 (para dispositivos IOS modernos)
  - qvdclient_i386-apple-darwin11 (mejor elección para emuladores)

Los binarios tienen algunas opciones obligatorias que se pueden obtener mediante el uso del parámetro -? , por ejemplo

```
$ ./qvdclient -?
./qvdclient [-?] [-d] -h host [-p port] -u username -w pass [-g wxh] [-f]

  -? : shows this help
  -v : shows version and exits
  -d : Enables debugging
```

```
  -h : indicates the host to connect to. You can also set it up in the env var QVDHOST.
       The command line argument takes precedence, if specified
  -p : indicates the port to connect to, if not specified 8443 is used
  -u : indicates the username for the connection. You can also set it up in the env var  ↩
      QVDLOGIN
       The command line argument takes precedence, if specified
  -w : indicates the password for the user. You can also set it up in the env var  ↩
      QVDPASSWORD
       The command line argument takes precedence, if specified
  -g : indicates the geometry wxh. Example -g 1024x768
  -f : Use fullscreen
  -l : Use only list_of_vm (don't try to connect, useful for debugging)
  -o : Assume One VM, that is connect always to the first VM (useful for debugging)
  -n : No strict certificate checking, always accept certificate
  -x : NX client options. Example: nx/nx,data=0,delta=0,cache=16384,pack=0:0
  -c : Specify client certificate (PEM), it requires also -k. Example -c $HOME/.qvd/client. ↩
      crt -k $HOME/.qvd/client.key
  -k : Specify client certificate key (PEM), requires -c. Example $HOME/.qvd/client.crt -k  ↩
      $HOME/.qvd/client.key
```

Es posible que desee establecer variables de entorno para fines de depuración y evitar que sus credenciales sean visibles. El cliente QVD reconoce las siguientes variables:

```
QVDHOST : Servidor al que desea conectar
QVDLOGIN : Usuario para autenticar con el servidor
QVDPASSWORD : Contraseña del usuario
QVD_DEBUG : Habitilita el log de depuración
QVD_DEBUG_FILE : Especifica en qué fichero guardar el log de depuración
DISPLAY : Requerido para ejecutar correctamente.
          Adicionalmente, en algunos sistemas podría necesitar ejecutar lo siguiente:
            export DISPLAY=localhost:0; xhost + localhost
            xhost +si:localuser:$LOGNAME
```

XAUTHLOCALHOSTNAME Solución alternativa

Algunas distribuciones recientes de Linux introdujeron la variable de entorno XAUTHLOCALHOSTNAME para almacenar el nombre de host local al inicio de la sesión X. Las bibliotecas NX no reconocen esta variable, sino que se refieren al archivo `X authority file`. Esto puede resultar en una autenticación correcta de QVD, pero que no se puede conectar totalmente al escritorio QVD con el error `X connection failed with error 'No protocol specified'`. Hay tres soluciones para esto.

Habilitar la autenticación basada en host:

```
$ export DISPLAY=localhost:0; xhost + localhost
```

Habilitar la autenticación de usuario local interpretada por el servidor:

```
$ xhost +si:localuser:$(whoami)
```

Agregue el nombre de host al archivo `X authority file`:

```
$ xauth add "$(/bin/hostname)/unix:0" MIT-MAGIC-COOKIE-1 \
 $( xauth list "localhost/unix:0" | awk '{print $3}' )
```

Part II

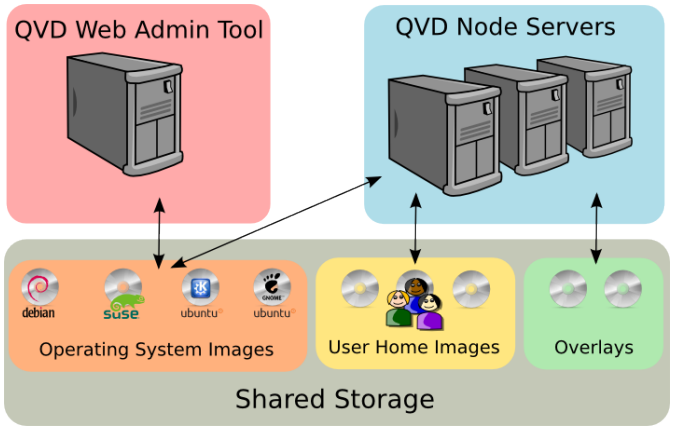
Consideraciones de diseño e integración

En esta parte del manual, exploramos cosas que afectarán el diseño de su solución, como las tecnologías de virtualización, los requisitos de almacenamiento y mecanismos de autenticación.

Chapter 8

Almacenamiento compartido

Dado que hay varios componentes del lado del servidor dentro de la infraestructura QVD, y cada uno de ellos suele estar instalado en diferentes sistemas, es importante disponer de una instalación de almacenamiento compartido accesible a todos los hosts de su granja de servidores.



Los servicios de intercambio de archivos de red actualmente soportados son GFS o OCFS2 en la parte superior de algunos servidores SAN (es decir, iSCSI, AoE, etc.) y NFS. QVD generalmente mantiene todos los archivos de uso común en la ubicación del directorio: /var/lib/qvd/storage

Note
Todas las rutas utilizadas por QVD son elementos configurables, por lo que debe tener en cuenta que aunque se trata de la ubicación predeterminada, los nombres de la infraestructura pueden ser diferentes dependiendo de su configuración. Puede comprobar estos ajustes mediante la utilidad de administración en línea de comandos de QVD o el WAT.

Mientras que algunos de los componentes no necesitan acceso a todas las carpetas del almacenamiento de QVD y, en algunos casos, puede optar por tener algunas de estas carpetas en ejecución localmente en un sistema, recomendamos que todas ellas sean accesibles dentro de alguna forma de almacenamiento compartido basado en red.

Tip
Mantener una copia de seguridad de las carpetas QVD que se guardan en su almacenamiento compartido es muy recomendable. Como mínimo, debe asegurarse de que las carpetas donde se almacenan los datos personales del usuario y las carpetas donde se encuentran las imágenes de disco se respalden de acuerdo con su estrategia de recuperación de desastres.

Chapter 9

Carpetas de almacenamiento

Hay una variedad de carpetas que pertenecen a la ruta de almacenamiento. Muchas de ellas son específicas para el tipo de virtualización que elija utilizar dentro de su entorno.

Almacenamiento General

- **staging**: ubicación temporal para todos los DIs que desee tener disponibles en el WAT con el fin de cargar como nueva imagen. Los archivos localizados aquí estarán disponibles en el WAT cuando selecciona agregar una imagen. El archivo de imagen se copia desde este directorio a **Images** cuando se crea la imagen mediante una de las herramientas de administración. La carpeta *staging* puede estar alojada localmente o en un recurso compartido de red, pero debe ser accesible desde la API.
- **images**: ubicación de los DIs (Disk Images) que son cargados por los nodos para cada máquina virtual que se crea. Esta carpeta debe ser accesible por todos los nodos servidor de QVD y la API. Se puede almacenar en un recurso compartido de red, pero en una configuración simple, en la que el WAT no se utilice o esté en el mismo sistema que el nodo servidor de QVD, puede alojarse localmente lo que ayudará a mejorar el rendimiento. Tenga en cuenta que cuando se utiliza KVM como virtualización, la imagen se carga en la máquina virtual directamente desde este directorio. Cuando se utiliza la virtualización LXC, la imagen es extraída desde este directorio en el directorio **basefs**, antes de que sea cargada.

Directorios de almacenamiento KVM

- **homes**: ubicación del directorio personal del usuario. En KVM, los datos del home del usuario se almacenan en un único archivo individual como imagen qcow2. El directorio **homes** debe ser accesible a todos los nodos servidor de QVD, normalmente en algún tipo de almacenamiento compartido por red como NFS, OCFS2 o GFS2.
- **overlays**: ubicación utilizada para almacenar *overlays* para datos que están en constante escritura en el sistema operativo para que funcione correctamente, como por ejemplo archivos temporales, datos variables, etc. Normalmente esta carpeta puede ser alojada localmente, pero para el comportamiento persistente en sus máquinas virtuales, puede elegir almacenarlos en un recurso compartido de red y configurar QVD para máquinas virtuales persistentes.

Directorios de almacenamiento LXC

- **basefs**: ubicación de los DIs (Disk Images) que los nodos cargan para cada máquina virtual que se crea. Estas deben ser accesibles por todos los nodos servidor de QVD y la API. Esta carpeta se puede almacenar en un recurso compartido de red, pero en una configuración simple, en la que todo esté en el mismo sistema que el nodo servidor de QVD, puede alojarse localmente lo que ayudará a mejorar el rendimiento. La carpeta basefs contendrá un subdirectorio para cada DI, que a su vez contendrá el árbol de archivos para un sistema operativo.

- **homefs**: ubicación del directorio personal del usuario. En LXC, los datos del home se almacenan en subdirectorios dentro del directorio **homefs**, denominado de acuerdo con el user-id y el osf-id almacenados en la QVD-DB. El directorio **homefs** debe ser accesible por todos los nodos servidor de QVD, normalmente en algún tipo de recurso compartido de archivos de red, como NFS, OCFS2 o GFS2.
- **overlayfs**: ubicación utilizada para almacenar *overlays* de datos que están en constante escritura por el sistema operativo, tales como archivos temporales y logs de aplicaciones. Por lo general, esta carpeta se puede alojar localmente, pero para el comportamiento persistente en sus máquinas virtuales, puede elegir almacenarlos en un recurso compartido de red y configurar QVD para Máquinas virtuales persistentes.
- **rootfs**: ubicación del LXC en ejecución una vez que todos los puntos de montaje requeridos estén montados y configurados. Normalmente esta carpeta es local para cada Nodo Servidor QVD, por razones de rendimiento, pero también podría ser almacenada en el almacenamiento compartido.

Directorios de almacenamiento LXC (BTRFS)

Con la versión 3.2, QVD introdujo soporte para el sistema de archivos btrfs. Esta difiere sutilmente de la configuración LXC estándar extrayendo cada imagen de disco en un subvolumen btrfs de solo lectura y realizando un snapshot del overlay (datos no persistentes) para cada nueva VM. Esto es considerablemente más eficiente que extraer la imagen en una carpeta debido a las capacidades de copia en escritura de btrfs, ofreciendo así una reducción significativa en la carga al almacenamiento compartido mediante la descarga persistente en el nodo local. Esta ganancia de rendimiento tiene un precio, sin embargo, en el sentido de que estos datos se perderán en el caso de que un cliente sea asignado a un nuevo nodo.

- **basefs**: ubicación de los DIs extraídos (Disk Images) que cargan los nodos para cada máquina virtual que se crea. Con un sistema btrfs, cada imagen se desempaqueta en su propio subvolumen en *basefs*. Este volumen será utilizado por cada VM que utilice esta DI. Para un sistema btrfs esto debe ser almacenado localmente en cada nodo.
- **homefs**: ubicación del directorio personal del usuario. Al igual que con la configuración LXC normal, los datos de origen se almacenan dentro de los subdirectorios del directorio **homefs** que se nombra de acuerdo con el id de usuario y el osf-id almacenados en la QVD-DB. El directorio **homefs** debe ser accesible por todos los nodos servidor de QVD normalmente en algún tipo de recurso compartido en red, como NFS, OCFS2 o GFS2.
- **overlayfs**: ubicación de datos no persistentes, como archivos temporales y de log. Cuando se inicia un contenedor, se crean los datos de overlay dentro de su propio subvolumen btrfs, dentro de *overlayfs*. Por cada subsiguiente VM utilizando esta imagen de disco, QVD crea una instantánea de este subvolumen y la instantánea se utiliza como el sistema de archivos raíz del contenedor. Desde btrfs se pueden crear instantáneas y subvolúmenes muy baratos con muy poca sobrecarga. Esto sucede casi en tiempo real. Dado que en un sistema btrfs esto debe hacerse localmente, reduce enormemente la carga en el almacenamiento compartido. Sin embargo, hay que señalar que, por lo tanto, estos datos no son persistentes y se perderán si el balanceador de carga dirige un usuario a otro nodo. QVD todavía mantendrá la configuración `vm.overlay.persistent` pero esta persistencia será sólo para sesiones consecutivas en el mismo nodo.
- **rootfs**: ubicación del contenedor LXC en ejecución una vez que todos los puntos de montaje requeridos estén montados y configurados. Con una configuración de btrfs, estos datos tienen que ser almacenados localmente en el nodo.

NFS

En esta sección del documento le daremos instrucciones para configurar NFS para QVD, ya que este es uno de los protocolos más utilizados para el almacenamiento compartido. Proporcionaremos instrucciones para Ubuntu 16.04 y para SUSE Linux Enterprise Server (SLES) 12.1, sin embargo, debería ser capaz de extrapolar estas instrucciones para proporcionar acceso NFS para cualquier distribución.

Tip
Le recomendamos que ejecute el siguiente proceso antes de instalar cualquier componente del servidor QVD para garantizar que cuando se instalen los componentes QVD, automáticamente utilizan el recurso compartido NFS desde el principio. De este modo, es menos probable que tenga problemas para migrar archivos y crear directorios a largo plazo.

Instalación del servidor NFS

Primero instale el servidor NFS. Para Ubuntu, esto se puede hacer como root usando el comando:

```
# apt-get install nfs-kernel-server
```

Y para SLES:

```
# zypper install nfs-kernel-server
```

Configuración del servidor NFS

Añada una entrada a `/etc /exports` de la siguiente manera:

```
/var/lib/exports * (rw, sync, no_subtree_check, no_root_squash)
```

Tenga en cuenta que esto significaría que en su servidor NFS, configuraría cada uno de los directorios QVD dentro de la ruta `/var/lib/exports`. Puede elegir otra ubicación más adecuada si prefiere alojar estos archivos en otra ruta. Una vez que haya agregado su entrada de ruta dentro de las exportaciones de NFS Server, debe volver a cargar el servidor NFS. Para Ubuntu (como root):

```
# /etc/init.d/nfs-kernel-server reload
```

Del mismo modo, Para SLES:

```
# /etc/init.d/nfsserver reload
```

Desde ahora, el servidor NFS debería hacer que la ruta configurada esté disponible red.

Montaje del directorio NFS en los hosts de QVD

Cada sistema host que esté ejecutando cualquier componente de QVD necesitará ser configurado para acceder al recurso compartido NFS que hemos configurado en el servidor NFS. Primero, cree el punto de montaje en sus sistemas host:

```
# mkdir -p /var/lib/qvd/storage
```

Asegúrese de tener las herramientas necesarias para acceder a un recurso compartido NFS instalado en sus sistemas anfitriones. En Ubuntu, debe instalar `nfs-common`:

```
# apt-get install nfs-common
```

En SLES, instale `nfs-client`:

```
# zypper install nfs-client
```

Para asegurarse de que el sistema de archivos NFS siempre está montado en el arranque, debe añadir la siguiente línea en `/etc/fstab`:

```
nfsserver:/var/lib/exports /var/lib/qvd/storage nfs rw,soft,intr,rsize=8192,wsize=8192 0 0
```

Tenga en cuenta que en la línea anterior, *nfsserver* es el nombre del servidor que aloja el recurso compartido NFS. Debería sustituir esto por la dirección IP o el nombre de host resuelto por DNS de su servidor NFS. Una vez que haya terminado de editar su fstab, debería ser capaz de montar el recurso NFS en sus sistemas host:

```
# mount /var/lib/qvd/storage
```

Por último, debe comprobar que el recurso NFS se ha montado correctamente. Puede hacer esto ejecutando el comando `mount` y luego verificando la salida para ver que el recurso NFS aparece:

```
mount
...
nfsserver: /var/lib/exports on /var/lib/qvd/storage type nfs (rw,soft,intr,rsize=8192,wsize ↩
    =8192,addr=172.20.64.22)
```

Chapter 10

Virtualización LXC dentro de QVD

Conceptos básicos de la tecnología LXC

La virtualización LXC se ejecuta directamente dentro del núcleo del sistema operativo host, de modo que los procesos que se ejecutan dentro de los contenedores invitados son realmente visibles dentro del host. Como resultado, los contenedores comparten el espacio del kernel, lo que es más eficiente, ya que sólo se carga un único kernel en el host que ejecuta sus escritorios virtuales. Por otro lado, esto significa que todas las máquinas virtuales necesariamente ejecutan el mismo kernel, por lo que las personalizaciones del kernel por máquina no son posibles. Dado que muchas distribuciones modifican las opciones de configuración del kernel para adaptarse al entorno, normalmente es aconsejable que se use la misma distribución para cada contenedor que se esté ejecutando en la plataforma host.

Cada contenedor tiene su propio espacio de archivos que se ejecuta en el sistema de archivos del host, llamado **rootfs**. La jerarquía del sistema de archivos de un contenedor es idéntica a la instalación de Linux que se está ejecutando y se puede acceder directamente desde el entorno del host. Esto hace que sea posible acceder al sistema de archivos subyacente de un contenedor en ejecución desde su host principal. Esto puede ser muy útil desde una perspectiva de solución de problemas y depuración. También facilita el acceso y la actualización de los contenedores LXC a los administradores del sistema. Por otro lado, la naturaleza de la virtualización LXC y sus requisitos específicos hacen que sea más difícil de configurar y más fácil de romper. En particular, es esencial que los procesos y scripts que requieran acceso directo al hardware (como udev) no se ejecuten dentro del espacio de contenedor. Con frecuencia, los requisitos y las dependencias del paquete pueden dificultar el mantenimiento para administradores inexpertos.

Important
Como se puede imaginar, no recomendamos que intente realizar operaciones de escritura en un contenedor en ejecución desde el host. Aunque técnicamente es posible, puede producir resultados inesperados.

A diferencia de un chroot tradicional, LXC proporciona un alto nivel de aislamiento de procesos y recursos. Esto significa que a cada contenedor se le puede asignar su propia dirección de red y puede ejecutar procesos sin afectar directamente a otros contenedores o al sistema host principal.

LXC se puede utilizar fuera de QVD con bastante facilidad, y cualquier imagen LXC que se pueda ejecutar dentro de QVD se puede cargar en cualquier sistema linux con soporte LXC. Para ejecutar un contenedor usando LXC fuera de QVD, necesitará crear un archivo de configuración para su contenedor, proporcionando detalles de puntos de montaje, grupos de control, requisitos de red y acceso a la consola. Consulte *man lxc.conf* para ver las opciones. Al ejecutar un contenedor LXC dentro de QVD, la solución generará automáticamente un archivo de configuración para el contenedor antes de que se inicie, para asegurarse de que está configurado correctamente para ejecutarse en el entorno QVD.

Los contenedores se pueden crear directamente desde el sistema de archivos de cualquier instalación linux funcional, pero casi con toda seguridad requerirá algunas modificaciones para poder trabajar. Esta modificación suele implicar la eliminación de cualquier proceso o secuencias de comandos que tengan acceso directo al hardware, y la recreación manual de nodos de

dispositivos. La mayoría de las distribuciones con soporte LXC también incluyen *plantillas*, que son básicamente scripts bash que configurarán una instalación básica del sistema operativo dentro de un contenedor para usted automáticamente.

Las plantillas pueden ahorrar mucho tiempo y deben utilizarse como una línea de base hacia la creación de sus imágenes LXC. Sin embargo, varían entre distribuciones y por lo general no pueden generar una configuración totalmente funcional, y sin duda requieren instalar muchos más paquetes manualmente para crear un contenedor hasta un nivel donde sea utilizable dentro de QVD.

Cuándo utilizar LXC

Es importante que sea capaz de determinar los mejores casos de uso para LXC, en lugar de utilizar KVM. Ambas tecnologías tienen sus ventajas y deben utilizarse en diferentes situaciones. En general, LXC debería ofrecer un rendimiento superior a KVM y escalará mejor, ya que la virtualización que ofrece tiene menos sobrecarga. Por otro lado, KVM ofrecerá una mayor flexibilidad y le permitirá ejecutar una mayor variedad de sistemas operativos invitados.

Estas son algunas de las directrices básicas que debe seguir al determinar si desea o no utilizar LXC:

- La imagen de disco que va a crear se compartirá entre muchos usuarios
- El sistema operativo invitado que desea instalar utiliza exactamente el mismo núcleo que el anfitrión (es decir, el núcleo será idéntico). Se recomienda encarecidamente que la distribución de invitados sea idéntica a la del anfitrión
- Desea abstraer aún más la virtualización para poder ejecutar otros sistemas operativos invitados en QVD, ejecutando KVM dentro de un entorno LXC. Esta es una configuración muy compleja y no se presentará en esta documentación.

En general, es más fácil configurar y configurar QVD para utilizar la virtualización KVM, y ha resultado ser más fácil para los administradores trabajar con imágenes KVM. Si tiene alguna duda o si es nuevo en QVD, le recomendamos que utilice KVM.

Si ya tiene una buena comprensión de QVD y ya está familiarizado con LXC, encontrará que el soporte para LXC dentro de QVD le permitirá implementar configuraciones LXC complejas muy fácilmente. También encontrará que la naturaleza de este tipo de virtualización le proporciona una capacidad de administración mucho mejor y que es capaz de lograr un uso más eficiente de su hardware.

Detalles de implementación de QVD LXC

En esta sección, analizaremos de cerca cómo LXC se implementa dentro de QVD.

Almacenamiento y estructura de imágenes de disco

QVD hace uso de `unionfs-fusible` para manejar los puntos de montaje del estilo de union. Esto permite a QVD montar directorios como el directorio personal del usuario y datos temporales típicamente manejados como overlays dentro de KVM en el LXC en ejecución. Dado que la imagen de disco LXC es un sistema de sólo lectura, `unionfs-fuse` facilita el montaje de áreas de almacenamiento con escritura. Por lo tanto, es esencial que el módulo `fuse` del kernel se cargue al inicio del sistema.

Dado que la implementación de LXC difiere dramáticamente de KVM, QVD almacena todos los datos asociados con cada implementación de máquina virtual en directorios lógicamente distintos dentro de la ubicación de almacenamiento de QVD.

Mientras que los directorios de "staging" e "images" se usan en común con KVM, la mayoría de la actividad funcional tiene lugar fuera de estos directorios. Cuando se inicia una máquina virtual para un usuario, la imagen de disco que se utilizará dentro de la máquina virtual se extrae literalmente del tarball almacenado en el directorio *images* en una subcarpeta dentro de la carpeta *basefs*.

Cuando se inicia la máquina virtual, el sistema de archivos que se extrae bajo la carpeta *basefs* se monta junto con el directorio personal del usuario, almacenados en la carpeta *homefs* y los overlays relevantes (en *overlayfs*) en un directorio en tiempo de ejecución dentro de *rootfs*. Este directorio de tiempo de ejecución se utiliza para cargar el LXC y servir el Escritorio Virtual al usuario final.

El contenido, la estructura y el propósito de la carpeta se describen con más detalle a continuación.

basefs

Para cada máquina virtual que se ha iniciado dentro del entorno QVD, se crea una subcarpeta dentro de la carpeta *basefs*. Esta subcarpeta se nombra usando como prefijo el ID asignado a la máquina virtual dentro de QVD-DB y tiene como sufijo el nombre del archivo de imagen de disco que se cargó para esa máquina. Por lo tanto, dentro de la carpeta basefs, es probable que vea carpetas con nombres similares a los siguientes:

```
# ls -l /var/lib/qvd/storage/basefs/
total 4
drw-r--r-- 18 root root 4096 2012-02-20 11:59 2-image.0.22.tgz
```

En este ejemplo, la carpeta se denomina *2-image.0.22.tgz*. Esto se debe a que el sistema de archivos contenido en esta carpeta pertenece a la máquina virtual con un ID == 2, y la imagen de disco que se carga aquí es del archivo de imagen de disco denominado *image.0.22.tgz*. Dentro de esta carpeta se encuentra un sistema de archivos linux típico:

```
# ls /var/lib/qvd/storage/basefs/2-image.0.22.tgz/
bin  dev  etc  lib  lib64  media  mnt  opt  root  sbin  selinux  srv  sys  tmp  usr  var
```

Utilizando *unionfs-fuse*, el sistema de ficheros representado en esta carpeta se montará conjuntamente con los sistemas de ficheros representados en las carpetas *homefs* y *overlayfs* dentro de la carpeta *rootfs* en tiempo de ejecución.

homefs

Los datos personales del usuario se almacenan dentro de la carpeta *homefs*. Según la convención, los directorios de inicio del usuario se almacenan dentro de una carpeta denominada de la siguiente manera:

```
<id de la VM>-<id del usuario>-homefs
```

Esto hace posible que un solo usuario tenga múltiples directorios personales para diferentes escritorios virtuales, basados en la máquina virtual en la que está montado el directorio personal.

overlayfs

Este directorio contiene overlays utilizados en la máquina virtual. Dado que el contenido de la imagen de basefs se trata como un sistema de archivos de sólo lectura, se crea un overlay para manejar los datos que la máquina virtual en ejecución necesite escribir en el sistema operativo. Normalmente, estos datos están en forma de logs, PIDs de tiempo de ejecución, archivos de bloqueo y archivos temporales.

Cada máquina virtual tiene su propio directorio de overlays, y se denomina siguiendo la convención:

```
<id of DI>-<id of VM>-overlayfs
```

Tenga en cuenta que si una máquina virtual no se inicia correctamente por alguna razón, se crea una carpeta de overlays temporal. Esta carpeta se nombra con el prefijo "deleteme-". La carpeta se conserva para permitirle ver los archivos de log específicos de una máquina virtual que pueden no haber iniciado, para ayudarle con el proceso de depuración.

rootfs

Este directorio contiene los puntos de montaje para ejecutar instancias de la máquina virtual LXC. Cada punto de montaje se denomina siguiendo una convención donde está prefijado con el ID de la máquina virtual dentro del QVD-DB. El directorio de punto de montaje se crea cuando se inicia el contenedor. Dado que sólo se utiliza para montar el sistema de archivos de un contenedor en ejecución, sólo contendrá algo cuando un contenedor se esté ejecutando. Si se detiene el contenedor, el directorio se desmonta y permanecerá vacío hasta que se reinicie el contenedor.

Redes

De forma similar a la implementación de KVM de QVD, es necesario crear una interfaz de puente en el host de Nodo QVD. Cuando se inicia una máquina virtual, se crea una interfaz de red virtual y se enlaza con la interfaz de puente. Para que funcione correctamente, deberá configurar QVD con el intervalo de IP utilizado para las máquinas virtuales. La interfaz del puente debe configurarse con una dirección IP por debajo del rango utilizado para las máquinas virtuales pero aún así en la misma red.

Configuración de la base QVD

De forma predeterminada, QVD está configurado para utilizar la virtualización KVM. Antes de intentar cargar cualquier imagen de disco LXC en la infraestructura QVD, debe asegurarse de que QVD ha sido reconfigurado para usar LXC. Para ello, debe actualizar los siguientes parámetros de configuración mediante la utilidad de línea de comandos de QVD Admin (podría usar también el WAT):

```
# qa4 config set tenant_id=-1,key=vm.hypervisor,value=lxc
# qa4 config set tenant_id=-1,key=vm.lxc.unionfs.bind.ro,value=0
# qa4 config set tenant_id=-1,key=vm.lxc.unionfs.type,value=unionfs-fuse
# qa4 config set tenant_id=-1,key=command.unionfs-fuse,value=/usr/bin/unionfs
```

En SLES, el binario unionfs es proporcionado por QVD y se encuentra en */usr/lib/qvd/bin/unionfs*, por lo tanto el último comando en la lista anterior debería ser modificado para reflejar esta ruta.

Tenga en cuenta que una vez que haya restablecido estos parámetros del sistema QVD, tendrá que reiniciar el HKD en cada uno de sus nodos Servidor QVD:

```
# /etc/init.d/qvd-hkd restart
```

Suponiendo que ya ha configurado su red correctamente, QVD debe ser capaz de cargar y ejecutar imágenes LXC.

Grupos de control LXC (cgroups)

Los contenedores QVD utilizan *cgroups* (grupos de control), una característica del kernel de Linux diseñada para limitar el uso de recursos de los grupos de procesos en el sistema. Un sistema de archivos virtual se monta bajo el directorio */sys/fs/cgroup* en el que se pueden configurar varios controladores, conocidos como *subsistemas*. Este directorio se puede cambiar modificando la configuración *path.cgroup* en la base de datos QVD, aunque esto no debería ser necesario.

De forma predeterminada, los subsistemas *cpu* del contenedor se colocan bajo el directorio por defecto */sys/fs/cgroup/cpu* Este comportamiento está controlado por la configuración QVD *path.cgroup.cpu.lxc* que se define de forma predeterminada como siguiente:

```
path.cgroup.cpu.lxc=/sys/fs/cgroup/cpu/lxc
```

Una vez más, no debería ser necesario cambiar esto en una instalación predeterminada.

Cargando imágenes LXC en QVD

Las herramientas de administración estándar de QVD, como el WAT y la utilidad de administración de línea de comandos QVD, se pueden utilizar para cargar una imagen LXC en QVD. La diferencia importante aquí es que el archivo tomará la forma de un archivo comprimido con tar-gzip, a diferencia de una imagen qcow2. Es absolutamente imperativo que el entorno haya sido preconfigurado para la virtualización LXC, o las imágenes se copiarán en la carpeta incorrecta y no se cargarán cuando se inicie una máquina virtual.

Para cargar una imagen LXC en QVD desde la línea de comandos, deberá seguir los siguientes pasos:

Agregue un OSF para alojar su archivo de imagen:

```
# qa4 osf new name=MyOSF
```

Ahora agregue su imagen de disco al OSF

```
# qa4 di new path=/var/lib/qvd/storage/staging/my_image.tar.gz,osf=MyOSF
```

Esto llevará algún tiempo, ya que la imagen se copiará desde el directorio de preparación en el directorio de imágenes.

Ahora agrega un usuario para poder probar la imagen

```
# qa4 user new name=test,password=test
```

Finalmente, cree una Máquina Virtual para el Usuario y adjunte el OSF que esta Máquina Virtual usará en tiempo de ejecución

```
# qa4 vm new name=TestVM,user=test,osf=MyOSF
```

Si su instalación de QVD está configurada correctamente, ahora podrá iniciar la máquina virtual y probarla.

Inicio de una máquina virtual LXC

Iniciar una máquina virtual LXC dentro de QVD no es diferente al inicio de una máquina virtual KVM. Puede iniciarse manualmente mediante la utilidad de línea de comandos Admin de QVD o bien puede iniciarse automáticamente mediante HKD cuando un cliente intenta conectarse al escritorio virtual manejado por la máquina virtual. Para fines de prueba, por lo general es aconsejable iniciar la máquina virtual desde la línea de comandos:

```
# qa4 vm name=TestVM start
```

Puede supervisar el proceso de inicio desde el WAT listando el estado de la máquina virtual.

```
# qa4 vm name=vmTest get
.-------+--------+--------+---------+--------+-----------------+-------------------------------+----

| id    | tenant | name   | blocked | user   | host            | di  ↩
                              | ip          | ip_in_use   | di_in_use                     | ↩
    state   | user_state   |
+-------+--------+--------+---------+--------+-----------------+-------------------------------+----

| 19165 | testin | vmTest |       0 | testin | qvd-host-node-9 | 90191-Desktop_Standard.tar ↩
    .gz | 10.2.153.13 | 10.2.153.13 | 90191-Desktop_Standard.tar.gz | running | disconnected ↩
     |
'-------+--------+--------+---------+--------+-----------------+-------------------------------+----
```

A medida que la máquina virtual arranca, cambiará de estado. Si la máquina virtual se inicia sin ningún problema, debe comprobar que es capaz de conectarse a ella mediante el cliente QVD y que el escritorio virtual se procesa. Si la máquina virtual no se inicia correctamente o no puede acceder al escritorio virtual, es posible que tenga que realizar alguna depuración.

Acceso a una máquina virtual LXC para depuración

Puesto que en realidad es posible acceder al sistema de archivos de una imagen LXC en ejecución directamente desde el sistema operativo del host, es posible revisar los archivos de log y realizar una buena cantidad de depuración directamente desde el host donde se está ejecutando la máquina virtual. Es importante notar, sin embargo, que es altamente desaconsejable intentar cualquier operación de escritura en el sistema de archivos de un contenedor en ejecución.

Cuando se inicia una máquina virtual LXC, sus diversos componentes se combinan utilizando unionfs y se montan en la carpeta *rootfs* en el área de almacenamiento de QVD (generalmente /var/lib/qvd/storage/rootfs). Dentro de este espacio, es posible ver el estado de una instancia en ejecución en tiempo real. Esto significa que puede acceder rápidamente a los archivos de log y a la información de configuración desde el sistema operativo del host. A menudo, esta es la mejor manera de depurar rápidamente los problemas dentro de una instancia en ejecución.

Si esto no basta, también puede conectarse a la instancia mediante ssh a través de la ip de la misma. Obtenga la ip de la VM mediante un comando como:

```
# qa4 vm name=test get
.-------+--------+--------+---------+--------+----------------+-------------------------------+----
| id    | tenant | name   | blocked | user   | host           | di  ↩
                              | ip          | ip_in_use   | di_in_use                     | ↩
    state   | user_state   |
+-------+--------+--------+---------+--------+----------------+-------------------------------+----
| 19165 | testin | vmTest |       0 | testin | qvd-host-node-9 | 90191-Desktop_Standard.tar ↩
   .gz | 10.2.153.13 | 10.2.153.13 | 90191-Desktop_Standard.tar.gz | running | disconnected ↩
    |
'-------+--------+--------+---------+--------+----------------+-------------------------------+----
```

Una vez que tenga la IP, podrá ejecutar el comando *ssh* para acceder a la máquina virtual.

Creación de imágenes de disco LXC

Para obtener más información, consulte las guías de creación de imágenes.

Chapter 11

Autenticación

Aunque QVD proporciona su propio *framework* de autenticación, que almacena usuarios en la base de datos QVD, es bastante común usar otro *framework* de autenticación para que los cambios en las contraseñas de usuario, etc., no necesiten ser replicados dentro la QVD-DB. QVD proporciona cierto nivel de integración con recursos externos. En este capítulo, exploraremos dos de los requisitos de integración más comunes, y el nivel de soporte ofrecido por QVD para estos *framework* de autenticación.

Integración LDAP y Active Directory

El *framework* de autenticación más utilizado es LDAP, y QVD es compatible con LDAP de por sí. Esto incluye la compatibilidad con Active Directory que hace uso de LDAP versiones 2 y 3.

Pruebe su servidor LDAP

Antes de intentar configurar QVD para autenticar contra LDAP o Active Directory, es recomendable probar su servidor primero para asegurarse de que tiene las credenciales correctas y el nombre distintivo (DN) para el servidor. Para hacer esto, tendrá que tener `ldap-utils` (Ubuntu) o `openldap2-client` (SLES) instalado en su nodo QVD.

Una vez hecho esto, consulte su servidor con el siguiente comando para LDAP:

```
# ldapsearch -xLLL -H ldap://example.com -b"dc=ejemplo,dc=com" cn=admin cn
```

Si utiliza Active Directory, utilice este comando en su lugar:

```
# ldapsearch -LLL -H ldap://example.com: 389 -b 'dc=ejemplo,dc=com' \
-D 'EJEMPLO\jdoe' -w 'contraseña' '(sAMAccountName=jdoe)'
```

En cualquier caso, modifique su consulta para que coincida con la de su propio servidor. Compruebe los resultados de la consulta LDAP para asegurarse de que la conexión a su Servidor LDAP es posible desde sus nodos QVD.

Configuración de QVD para la autenticación LDAP

La autenticación LDAP se configura dentro de QVD mediante la configuración de en la base de datos QVD. Esto se puede lograr utilizando la utilidad de administración de CLI de QVD o el WAT.

```
# qa4 config set tenant_id=-1,key=l7r.auth.mode,value='ldap'
# qa4 config set tenant_id=-1,key=auth.ldap.host,value='ldap://example.com:389'
# qa4 config set tenant_id=-1,key=auth.ldap.base,value='dc=example,dc=com'
```

En el ejemplo anterior, hemos cambiado el modo de autenticación QVD a LDAP, estableciendo el host LDAP a `example.com` y el puerto` 389`. También se ha establecido el basedn donde se buscarán todos los usuarios a `dc=example,dc=com`. Con estos elementos básicos de configuración, QVD buscará en el directorio LDAP un nombre de usuario coincidente y, a continuación, realizará un BIND contra ese usuario con las credenciales suministradas por el cliente. De forma predeterminada, la búsqueda se realiza con scope `base` y filtro `(uid =% u)`. Utilizando nuestro ejemplo de host anterior, un cliente con el nombre de usuario definido como *invitado* necesitaría una entrada correspondiente `uid=invitado,dc=example,dc=com` dentro del servidor LDAP que se ejecuta en el host *example.com* disponible en el puerto *389*. Puede cambiar el alcance y la configuración del filtro para la búsqueda, para permitir que QVD escanee otras ramas y atributos para encontrar un usuario coincidente:

```
# qa4 config set tenant_id=-1,key=auth.ldap.scope,value=sub
# qa4 config set tenant_id=-1,key=auth.ldap.filter,value='(|(uid =%u)(cn=%u))'
```

Los ejemplos anteriores cambian el ámbito de búsqueda predeterminado para la autenticación LDAP a *sub* y el filtro hará que la búsqueda coincida con los usuarios con el *uid* o el *cn* igual al nombre de usuario proporcionado.

Configuración de QVD para la configuración de Active Directory

La configuración de Active Directory es similar a la configuración de LDAP, pero con un pequeño ajuste en el parámetro `auth.ldap.filter` como sigue:

```
# qa4 config set tenant_id=-1,key=l7r.auth.mode,value='ldap'
# qa4 config set tenant_id=-1,key=auth.ldap.host,value='ldap://example.com:389'
# qa4 config set tenant_id=-1,key=auth.ldap.base,value='OU=People,DC=example,DC=com'
# qa4 config set tenant_id=-1,key=auth.ldap.scope,value='sub'
# qa4 config set tenant_id=-1,key=auth.ldap.binddn,value='CN=Administrator,CN=Users,DC= ↩
    example,DC=com'
# qa4 config set tenant_id=-1,key=auth.ldap.bindpass,value='contraseña'
# qa4 config set tenant_id=-1,key=auth.ldap.filter,value='(sAMAccountName=%u)'
```

Aquí QVD coincide con el nombre del usuario en `sAMAccountName` que es el nombre de inicio de sesión para el usuario de Windows.

Limitaciones

Si bien es trivial que QVD autentique a los usuarios contra LDAP, aún necesita crear usuarios coincidentes dentro de su QVD-DB para asignar máquinas virtuales para ellos. El `Auto Plugin` discutido en la siguiente sección le permite aprovisionar usuarios y asignarles una máquina virtual predeterminada automáticamente al conectarse. Las herramientas QVD que le permiten cambiar las contraseñas de usuario dentro de QVD no actualizarán contraseñas LDAP, ya que esto puede afectar al funcionamiento de otras instalaciones dentro de su infraestructura. Aunque estas herramientas reportarán que han tenido éxito en un cambio de contraseña, es importante entender que la contraseña que ha sido cambiada es la que se almacena en la QVD-DB para el usuario, y no la contraseña dentro del directorio LDAP. Si utiliza la autenticación LDAP, el proceso de inicio de sesión ignora completamente las contraseñas almacenadas en QVD-DB. Los cambios de contraseña se hacen mediante las herramientas que usted utiliza generalmente para administrar sus usuarios.

Referencia LDAP

- **l7r.auth.plugins** (Requerido). Establecer a "ldap" para habilitar esta funcionalidad.
- **auth.ldap.host** (Obligatorio). Puede ser un host o un uri LDAP como se especifica en Net::LDAP
- **auth.ldap.base** (Obligatorio). La base de búsqueda donde encontrar a los usuarios con el auth.ldap.filter (ver abajo)
- **auth.ldap.filter** (Opcional. Por defecto *(uid =%u)*). La cadena %u se sustituirá por el nombre de inicio de sesión
- **auth.ldap.binddn** (Opcional. Por defecto vacía). El enlace inicial para encontrar los usuarios. De forma predeterminada, El enlace inicial se hace como anónimo a menos que se especifique este parámetro. Si contiene la cadena %u, que se sustituye por el inicio de sesión

- **auth.ldap.bindpass** (Opcional. Por defecto vacío). La contraseña para el binddn
- **auth.ldap.scope** (Opcional. Por defecto *base*). Consulte el atributo Net::LDAP scope en la operación de búsqueda. Si está vacía, se utilizará la contraseña proporcionada durante la autenticación
- **auth.ldap.userbindpattern** (Opcional. Por defecto vacía). Si se especifica, se intenta un bind inicial con esta cadena. El atributo de inicio de sesión se sustituye por %u.
- **auth.ldap.deref** (Opcional. Por defecto never). Cómo los alias son dereferenced, los valores aceptados son never, search, find y always. Vea Net::LDAP para más información.
- **auth.ldap.racf_allowregex** (Opcional. Por defecto no establecido). Esto es un regex para permitir autenticar algunos Códigos de error RACF. Un ejemplo de configuración sería "^R004109". Uno de los casos comunes es R004109 que devuelve un código ldap 49 (credenciales inválidas) y un mensaje de texto como "R004109 La contraseña ha expirado (srv_authenticate_native_password))". Si no tiene RACF esto probablemente no es para usted.

Ejemplos de errores RACF (con su texto original): * **R004107** The password function failed; not loaded from a program controlled library. * **R004108** TDBM backend password API resulted in an internal error. * **R004109** The password has expired. * **R004110** The userid has been revoked. * **R004128** Native authentication password change failed. The new password is not valid or does not meet requirements. * **R004111** The password is not correct. * **R004112** A bind argument is not valid. * **R004118** Entry native user ID (ibm-nativeId,uid) is not defined to the Security Server.

Algoritmo de autenticación

Si se define **auth.ldap.userbindpattern**, se intenta un bind con este DN sustituyendo %u por el login. Si tiene éxito se autentica al usuario y si falla se intentan siguientes los pasos:

- Si está definido **auth.ldap.binddn**, se intenta un bind como dicho usuario.
- Si no está definido, se intenta el bind como usuario anónimo.
- Se busca el *userdn*, con la ruta de búsqueda especificada en **auth.ldap.base** y el filtro de usuario especificado como **Auth.ldap.filter**.
- En **auth.ldap.filter** se sustituye **%u** por el nombre de inicio de sesión.
- Si no se encuentra ningún usuario, la autenticación falla.
- Si se encuentra un *userdn* se intenta un bind con ese usuario.

Plugin automático de aprovisionamiento de usuarios

Cuando se utiliza un mecanismo de autenticación alternativo, como el Plugin de autenticación LDAP, existe un requisito común para usuarios que se han autenticado contra el plugin proporcionado. El `Auto Plugin` se ha diseñado para satisfacer este requisito. Cuando QVD está configurado para autenticarse en una fuente externa, como LDAP, normalmente no se tiene ningún registro para los usuarios que inician sesión. Si el Plugin Auto está habilitado, el registro del usuario se crea automáticamente. Los registros de usuario creados de esta manera se proporcionan con una máquina virtual predeterminada, como se especifica en la configuración del plugin. Para habilitar el plugin, agregue "auto" a la lista de plugins habilitados en la configuración del L7R :

```
# qa4 config set tenant_id=-1,key=l7r.auth.plugins,value=auto,ldap
```

Tenga en cuenta que en el ejemplo anterior, estamos utilizando el **plugin auto** en conjunción con el plugin de autenticación ldap. Ahora necesitará indicar al complemento automático qué OSF utilizará al crear un Máquina virtual para el usuario:

```
# qa4 config set tenant_id=-1,key=auth.auto.osf_id,value=1
```

En este caso, los nuevos usuarios que se autentiquen mediante LDAP obtendrán una VM con el ID de OSF que hemos especificado. También es posible forzar el uso de una imagen de disco DI Tag. Esto puede hacerse ajustando la siguiente opción de configuración:

```
# qa4 config set tenant_id=-1,key=auth.auto.di_tag,value="prueba"
```

Un ejemplo típico para usar el plugin LDAP y el plugin auto juntos requeriría que ejecutase los siguientes comandos:

```
# qa4 config set tenant_id=-1,key=l7r.auth.plugins,value=auto,ldap
# qa4 config set tenant_id=-1,key=auth.ldap.host,value='ldaps://myldap.mydomain.com:1636'
# qa4 config set tenant_id=-1,key=auth.ldap.base,value='ou=People,dc=example,dc=com'
# qa4 config set tenant_id=-1,key=auth.ldap.scope,value=sub
# qa4 config set tenant_id=-1,key=auth.ldap.filter,value='(&(objectClass=inetOrgPerson)(cn ↩
    =%u))'
# qa4 config set tenant_id=-1,key=auth.auto.osf_id,value=1
```

Tip
Utilice comillas alrededor de cualquier carácter especial que pueda estar sujeto a expansión shell o que pueda ser intepretado de manera inesperada como pipes, corchetes, ampersand, etc...

Esto significaría que como usuarios autenticados por primera vez, usando QVD, ellos se autenticarían usando su repositorio LDAP y obtendrían automáticamente un escritorio predeterminado para comenzar a trabajar inmediatamente.

Part III

Balanceo de carga

Chapter 12

Introducción

QVD está diseñado para ser utilizado en un entorno de balanceo de carga. Dado que una instalación habitual hace uso de varios nodos servidor de QVD para ejecutar todas las máquinas virtuales, es común que los clientes se conecten a éstos a través de una balanceador hardware. Puesto que las máquinas virtuales pueden ejecutarse en cualquier nodo, y los clientes pueden conectarse a cualquier otro nodo, el L7R de QVD realiza conexiones puente con el nodo correcto. De hecho, dado que cada nodo tiene recursos limitados, las máquinas virtuales en ejecución deben ser equitativamente distribuidas para maximizar los recursos del sistema a través de la granja de nodos servidores.

Si una máquina virtual no se está ejecutando para el usuario que se conecta, el L7R determina qué nodo servidor es el más apropiado para iniciar una nueva máquina virtual para ese usuario. QVD utiliza su propio algoritmo de balanceo de carga para determinar qué nodo debe utilizarse para la nueva máquina virtual. Esta algoritmo evalúa qué nodo tiene la mayor cantidad de recursos libres, calculado como la suma ponderada de RAM libre, CPU no utilizada y un valor aleatorio para traer algo de entropía al resultado. Una vez que el mejor nodo servidor ha sido seleccionado, la QVD-DB se actualiza para indicar que la máquina virtual debe iniciarse en este nodo servidor y la máquina virtual es iniciada automáticamente por el HKD del nodo. Todo este proceso se conoce como *Balanceo de Carga QVD* y se utiliza para que las máquinas virtuales se distribuyan equitativamente en todos los nodos servidor. Esto maximiza los recursos disponibles para cualquier máquina virtual, preservando así la máxima funcionalidad.

Chapter 13

Comprobación de salud QVD

Cuando se utiliza un balanceador de carga externo para encaminar el tráfico a los diferentes nodos servidor de QVD, generalmente necesitará algún método para realizar comprobaciones de estado contra cada una de las instancias del HKD. El componente L7R incluye un servicio de comprobación de salud que responde a través de HTTP. Normalmente, necesitará configurar su balanceador de carga para realizar un *HTTP GET* En la URL: *https://hostname/qvd/ping* donde *hostname* es el nombre de host o IP para la instancia del nodo servidor. La consulta devolverá una cadena de texto con el contenido "I'm alive!" si el servidor está sano y disponible. Algunos de nuestros usuarios aprovechan el balanceo de carga de software proporcionado por Linux Virtual Server (LVS). Un ejemplo de la configuración requerida en el archivo Ldirectord.cf :

```
autoreload = no
checkinterval = 1
checktimeout = 3
negotiatetimeout = 3
logfile="/var/log/ldirectord.log"
quiescent = yes
virtual = 150.210.0.72:8443
    checkport = 8443
    checktype = negotiate
    httpmethod = GET
    protocol = tcp
    real = 150.210.4.1:8443 gate 10
    real = 150.210.4.2:8443 gate 10
    receive = "I am alive!"
    request = "/qvd/ping"
    scheduler = wlc
    service = https
```

Chapter 14

Cambio de la ponderación en el balanceador de carga de QVD

El algoritmo de balanceo de carga QVD predeterminado calcula la carga actual del sistema para cada uno de los nodos servidor disponibles multiplicando la RAM disponible, CPU disponible y un número aleatorio con el fin de puntuar cada sistema en el clúster. Como ya hemos mencionado, estas cifras están ponderadas, de modo que usted puede alterar las funciones del balanceador de carga.

Aumentar el peso de la variable RAM en el algoritmo hará que se balancee la carga al agregar prioridad a sistemas con más RAM disponible.

Aumentar el peso de la variable CPU en el algoritmo hará que se balancee la carga para añadir precedencia a los sistemas con más CPU disponible.

Aumentar el peso de la variable aleatoria en el algoritmo hará que se balancee la carga para aumentar la probabilidad de que se elija un servidor aleatorio. Estos pesos se controlan como ajustes de configuración dentro de QVD-DB y pueden ser alterados mediante la utilidad de administración en línea de comandos de QVD y el WAT:

```
# qa4 config set tenant_id=-1,key=l7r.loadbalancer.plugin.default.weight.cpu,value=3
# qa4 config set tenant_id=-1,key=l7r.loadbalancer.plugin.default.weight.ram,value=2
# qa4 config set tenant_id=-1,key=l7r.loadbalancer.plugin.default.weight.random,value=1
```

En el ejemplo anterior, hemos asignado más peso a los recursos de la CPU, ligeramente menos a la RAM, y aún menos al aleatorizador. Esto dará como resultado máquinas que se están iniciando en los nodos servidor que tienden a tener más recursos de CPU disponible.

Creación de un balanceador de carga QVD personalizado

Dado que QVD es un producto de código abierto construido en gran parte en Perl, es relativamente simple construir su propio balanceador de carga QVD personalizado que utilice el algoritmo que usted desee. Un caso de uso típico sería donde usted tiene un conjunto dedicado de Nodos servidor que prefiere utilizar sobre otro conjunto. QVD tiene un sistema de plugins para balanceo de carga. Un plugin de balanceo de carga es un plugin de la Subclase de QVD::L7R::LoadBalancer::Plugin que tiene que estar dentro del paquete QVD::L7R::LoadBalancer::Plugin.

API para plugins

get_free_host($vm) = $host_id

Devuelve el id del nodo en el que se debe iniciar la máquina virtual $vm. Un balanceador de carga tiene que implementar al menos este método. El parámetro $vm es un objeto QVD::DB::Result::VM. Le da acceso a los atributos y propiedades de la máquina virtual. Los atributos y propiedades del usuario de la VM y OSF se pueden acceder a través de $vm→user y $vm→osf respectivamente. Se puede acceder a otros datos a través de QVD::DB.

init()

Inicializa el balanceador de carga. Use esto si su balanceador de carga tiene que ser configurado, por ejemplo cargando una caché persistente.

Ejemplo mínimo: asignación aleatoria

Este balanceador de carga asigna máquinas virtuales a nodos backend aleatorios.

```
package QVD::L7R::LoadBalancer::Plugin::Random;

use QVD::DB::Simple;
use parent 'QVD::L7R::LoadBalancer::Plugin';

sub get_free_host {
    my ($self, $vm) = @_;
    my $conditions = { backend => 'true',
                       blocked => 'false',
                       state   => 'running' };

    my $attr = { columns  => 'host_id' };

    my @hosts = rs(Host)->search_related('runtime', $conditions, $attr)->all;
    return $hosts[rand @hosts]->host_id;
}

1;
```

Part IV

Operating System Flavours y máquinas virtuales

En esta parte del manual, encontrará toda la información que necesita para crear, editar y administrar un OSF (Operating System Flavour) que obtendrá al cargar sus máquinas virtuales. También exploramos el VMA (Agente de Máquina Virtual) con un poco más de detalle para ver cómo se puede utilizar para activar sus propias funcionalidades basadas en acciones realizadas por usuarios que acceden a la máquina virtual.

Chapter 15

Introducción

Un OSF (Operating System Flavour) se compone realmente de dos elementos: un DI (Disk Image), y algunos parámetros de tiempo de ejecución que se almacenan en la QVD-DB cuando la imagen de disco se carga en QVD usando el WAT o la línea de comandos.

QVD utiliza DIs para servir a grupos de usuarios que hacen uso de un conjunto común de aplicaciones. Mediante el uso de una única imagen para cada grupo de usuarios, se hace más fácil administrar entornos de escritorio para todos los usuarios. También mejora la seguridad general, ya que puede aplicarse distintas políticas para cada grupo de usuarios. De esta manera, si un grupo de usuarios requiere una aplicación particular, puede instalarla una vez y el cambio se aplicará a todos los usuarios que comparten la misma DI.

Igualmente, puede eliminar una aplicación del escritorio de un grupo completo. Las DI pueden duplicarse fácilmente, de modo que pueda crear rápidamente entornos para diferentes subconjuntos de usuarios. Al copiar una imagen de base, puede editar la copia y proporcionar aplicaciones adicionales u otras personalizaciones a un segundo grupo de usuarios sin tener que repetir la instalación de un sistema operativo completo. De esta forma, QVD puede reducir enormemente la administración y el mantenimiento, mejorar la conformidad con el escritorio y facilitar la implementación de la políticas de seguridad.

Para más detalles, lea por favor el manual sobre creación de imágenes de disco.

Chapter 16

Ciclo de vida de imágenes y VMs

QVD facilita la actualización y modernización de las imágenes de disco. En pocas palabras se puede actualizar una imagen y marcar las máquinas virtuales que la están ejecutando con una fecha de expiración. Así el administrador puede estar seguro de que los usuarios comenzarán a ejecutar la última imagen de disco a partir de una fecha. La necesidad de esta característica se hace evidente cuando consideramos un escenario donde el usuario está conectado a una máquina virtual, pero una nueva DI está disponible. Desconectar al usuario al azar no es deseable y aquí es donde esta característica muestra su valor.

Los límites de expiración se pueden configurar como `hard` y `soft`. Los límites `soft` se pueden establecer para activar un `hook` o mecanismo a la elección del administrador para alertar al usuario de la necesidad de cerrar la sesión tan pronto como sea factible. Lo bueno de los límites `soft` es que son extremadamente flexibles: el `hook` puede ser cualquier ejecutable de Linux, lo que significa que el administrador elige cómo quieren reaccionar al límite `soft` que se está alcanzando.

Por el contrario, los límites `hard` son una proposición mucho más simple. Para los usuarios particularmente incisivos que tal vez están indispuestos u optan por ignorar las solicitudes de cierre de sesión, los límites rígidos proporcionan una opción para reiniciar de forma forzada una máquina virtual. Esto significa que el administrador puede establecer un límite para garantizar que todos los usuarios estén utilizando la última imagen de disco con las nuevas características y actualizaciones de seguridad que se consideren necesarias.

Los límites de caducidad pueden establecerse en el WAT (consulte la documentación de esta herramienta).

También puede hacerse a través del qa4:

```
# qa4 vm di_id=1000 set expiration_soft=now
```

Los parámetros de fecha y hora son bastante flexibles, aceptando los mismos formatos que el comando `at`. La página de manual `at` describe esto con mayor detalle:

```
At  allows  fairly  complex  time specifications, extending the POSIX.2
standard.  It accepts times of the form HH:MM to run a job at a specific time
of day.  (If that time is already past, the next day is assumed.)  You may also
specify midnight, noon, or teatime (4pm) and you can have a time-of-day
suffixed with AM or PM for running in the morning or the evening.  You can also
say what day the job will be run, by giving  a  date  in  the  form month-name
day  with  an  optional year, or giving a date of the form MMDD[CC]YY,
MM/DD/[CC]YY, DD.MM.[CC]YY or [CC]YY-MM-DD.  The specification of a date must
follow the specification of the time of day.  You can also give times like now
+ count time-units, where the time-units can be minutes, hours, days, or weeks
and you can tell at to run the job today by suffixing the time with today and
to run the job tomorrow by suffixing the  time  with tomorrow.
```

Note
La expiración de máquinas virtuales está diseñada para actualizarlas a la última versión de una imagen de disco. De hecho, en el WAT, cuando se actualizan las etiquetas de imagen, y dicho cambio afecta a máquinas virtuales en ejecución, se invita automáticamente al administrador a fijar una fecha de expiración para dichas máquinas. No obstante, también se puede utilizar para fijar límites arbitrarios en el tiempo de conexión a las máquinas virtuales según determine el administrador.

Establecimiento de límites de caducidad

Los límites de caducidad para una máquina virtual son opcionales y pueden adoptar la forma de límites `soft` y `hard`, establecidos utilizando los argumentos `expire-soft` y `expire-hard`. Se puede configurar uno o ambos. Normalmente, los límites se establecerán al reetiquetar imágenes de disco que afecten a máquinas virtuales a través del WAT.

Pero también se pueden fijar en línea de comandos:

```
# qa4 vm di_name=1000-ubuntu-13-04.tar.gz set expiration_soft=now expiration_hard=tomorrow
Total: 3
```

En este caso, todas las máquinas virtuales en ejecución que utilicen dicha imagen de disco, tendrán sus fechas de caducidad establecidas en `now` para soft y `tomorrow` para lo hard.

Tip
En el WAT se puede elegir las fechas en un calendario, para comidad del administrador.

Los límites también se pueden establecer directamente sobre máquinas virtuales, sin tener en cuenta la di que utilizan:

```
# qa4 vm id=15 set expiration_soft=now
Total: 1
```

Tip
Los tiempos de expiración `hard` y `soft` se eliminan cuando se reinicia una máquina virtual, independientemente de si se ha alcanzado el tiempo designado.

Límite de Expiración Soft

El House Keeping Daemon o HKD realiza un seguimiento del estado de las máquinas virtuales y supervisa los tiempos de caducidad. Cuando se alcanza un límite `soft`, el HKD alertará a la máquina virtual a través del agente de máquina virtual (VMA) que se ejecuta como un demonio dentro de cada máquina virtual. Si está configurado, el VMA llamará al `hook` (un ejecutable de la elección del administrador) que se ejecutará solicitando al usuario que cierre la sesión (por ejemplo) .

El HKD continuará monitoreando los límites de vencimiento de una máquina virtual a intervalos de una hora. Si el límite sigue existiendo, es decir, la máquina virtual aún no se ha reiniciado, alertará de nuevo al VMA, que a su vez llamará al ejecutable apropiado y continuará haciéndolo hasta que el límite haya sido eliminado.

Configuración de la DI

Los ajustes de límite `soft` no están habilitados por defecto en QVD 4.0. Queda como tarea del creador de la imagen configurar el VMA y decidir sobre la acción tomada cuando se alcanza un límite `soft`.

Configuración de VMA

Para configurar el VMA para que actúe sobre un límite de caducidad `soft`, se debe declarar la siguiente línea en `/etc/qvd/vma.conf` en la imagen de disco:

```
vma.on_action.expire = <path to executable>
```

Hooks del VMA para expiración

Debido a que el ciclo de vida de las imágenes en QVD está diseñado para ser lo más flexible posible, la forma en que la máquina virtual interactúa con el usuario es enteramente responsabilidad del administrador. A continuación ofrecemos un ejemplo simplificado de un `hook` que el QVD VMA podría llamar:

- El primer script, hook.sh, identifica al usuario que ejecuta el escritorio local en la máquina que luego llama a un segundo script notify-user.sh.
- notify-user.sh se ejecuta como ese usuario, y a su vez invoca una ventana emergente xmessage en el escritorio para solicitar al usuario que reinicie.
- La opción que el usuario selecciona determina el código de retorno: si es 0, se ejecuta un reinicio, si no, se termina el script.

hook.sh

```
#!/bin/bash
user=$(ps ho user -p $(pgrep xinit))
script=/usr/local/bin/notify-user.sh
su - $user -c $script
if [ $? -eq 0 ] ; then
  /sbin/telinit 6
fi
```

notify-user.sh

```
#!/bin/bash
message="Please reboot ..."
export DISPLAY=:100
xmessage -buttons ok:0,wait:1 $message
```

Caution
Esto es sólo un ejemplo simple de un `hook` de expiración QVD para dar al lector una idea del concepto. No está diseñado para su uso en un entorno de producción. En el mundo real probablemente desearía determinar si el script ya ha sido llamado y aún no ha recibido una respuesta del usuario, para evitar así una proliferación de xmessages.

Límite de Expiración Hard

Los límites de expiración `hard` funcionan de manera similar a los límites `soft`, con una diferencia significativa. En lugar de llamar a un ejecutable para que tal vez solicite o advierta al usuario de un inminente reinicio, el HKD simplemente reinicia la máquina.

Chapter 17

Actualización manual de imágenes

KVM

El procedimiento aquí descrito no debe ejecutarse sobre una imagen que esté siendo utilizada por máquinas virtuales. Debe detener antes todas estas imágenes (manualmente o mediante límites de expiración) y bloquearlas para que no puedan arrancar, antes de llevarlo a cabo. También puede por supuesto, realizar una copia de la imagen y trabajar sobre ella, no interrumpiendo así el trabajo de sus usuarios.

Una vez que todas las máquinas virtuales han sido detenidas, o cuando haya terminado la copia, ejecute la imagen dentro de KVM.

En Ubuntu, puede ejecutarla de la siguiente manera:

```
# kvm -hda example.img -m 512
```

Y en SLES:

```
# qemu-kvm -hda example.img -m 512
```

KVM cargará una máquina virtual y le permitirá iniciar sesión como el usuario que creó cuando instaló el sistema operativo. Ahora puede realizar cualquier tarea de administración como este usuario.

Cuando haya completado cualquier trabajo en la imagen, apáguela. Puede marcar las máquinas virtuales que requieren acceso a la imagen como "desbloqueadas" y permitir que se inicien o añadir la imagen como si fuera nueva, y modificar las etiquetas en consecuencia. Si lo hace a través del WAT, le ofrecerá automáticamente expirar las máquinas virtuales cuya etiqueta se ha visto afectada, y esto obligará a sus usuarios a actualizarse según los límites que usted fije.

LXC

La condición del apartado anterior se repite aquí. No debe realizar cambios sobre una imagen que esté siendo utilizada, ya que provocaría inestabilidades en la solución. Deténga las máquinas pertinentes o realice una copia de la imagen que desea modificar.

Una vez hecho esto, busque el contenedor LXC que desea editar en el almacenamiento compartido dentro del directorio *basefs* (si ha parado las máquinas, y si no donde haya usted copiado la imagen). Dependiendo de sus requerimientos, puede usar chroot en el directorio y trabajar directamente dentro de él como sea necesario o puede cargarlo como una instancia de LXC. Dado que cargar una imagen en una instancia LXC por separado generalmente requiere que configure correctamente la red y proporcione un archivo de configuración, generalmente se recomienda que intente realizar modificaciones en una imagen utilizando un chroot.

El siguiente ejemplo muestra cómo puede utilizar bind mounts y chroot para acceder a una imagen de disco LXC para realizar actualizaciones:

```
# mount -o bind /proc /var/lib/qvd/storage/basefs/1-image1.tgz/proc/
# mount -o bind /dev /var/lib/qvd/storage/basefs/1-image1.tgz/dev/
# mount -o bind /sys /var/lib/qvd/storage/basefs/1-image1.tgz/sys/
# chroot /var/lib/qvd/storage/basefs/1-image1.tgz
#
```

Cuando haya terminado de realizar cambios, recuerde cerrar y desmontar los bind mount:

```
# exit
# umount /var/lib/qvd/storage/basefs/1-image1.tgz/proc
# umount /var/lib/qvd/storage/basefs/1-image1.tgz/dev
# umount /var/lib/qvd/storage/basefs/1-image1.tgz/sys
#
```

Cuando termine, pruebe a arrancar una de las vm que utilizan la imagen que acaba de modificar, o cárguela como una imagen nueva si había realizado una copia para no desconectar usuarios.

Si todo ha ido bien, marque las máquinas virtuales que requieren acceso a la imagen como "desbloqueadas" o en caso de haber hecho una imagen nueva, actualice los tags.

Es importante que pruebe la imagen LXC después de hacer cambios para asegurarse de que nada ha cambiado una configuración que pueda tener acceso directo al hardware. Los paquetes que tienen *udev* como una dependencia a menudo pueden dar problemas si no ha tomado medidas para evitar que udev se ejecute.

Chapter 18

VMA HOOKS

Introducción

Los `hook` del VMA se pueden configurar dentro de una DI para activar la funcionalidad dentro de una máquina virtual cuando se producen eventos relacionados con QVD. Esto le permite modificar automáticamente el comportamiento de la plataforma para adaptar el sistema operativo a los requisitos particulares relacionados con el cliente y resolver problemas concretos.

Los `hooks` se agregan como entradas de configuración dentro del archivo de configuración VMA en la DI subyacente. Por lo tanto, editando el archivo `/etc/qvd/vma.conf` y añadiendo una entrada similar a la siguiente:

```
vma.on_action.connect = /etc/qvd/hooks/connect.sh
```

Es posible garantizar que la secuencia de comandos `/etc/qvd/hooks/connect.sh` se ejecutará cada vez que un usuario se conecte a la máquina virtual.

También es posible que QVD proporcione scripts con parámetros de línea de comandos que son específicos de QVD, como por ejemplo:

- Cambios de estado, acciones o el proceso de aprovisionamiento que ha activado la llamada al `hook`.
- Propiedades de la máquina virtual definidas en la base de datos de administración.
- Parámetros generados por los complementos de autenticación.
- Parámetros de conexión del usuario.
- Parámetros suministrados por el programa cliente.

Los `hooks` tienen su propio archivo de log, almacenado en `/var/log/qvd/qvd-hooks.log` en la máquina virtual. Esto hace posible ver qué `hooks` han activado scripts para ejecutar y depurar cualquier comportamiento inusual.

Hooks de acción

Los `hooks` de acción se ejecutan cada vez que comienza una acción particular.

Si el `hook` falla con un código de error distinto de cero, la acción se abortará.

Todos los `hooks` de acción reciben estos parámetros:

- *qvd.vm.session.state*: estado actual del servidor X-Windows
- *qvd.hook.on_action*: Acción que activa el `hook`.

conectar

Clave: **vma.on_action.connect**

Este `hook` se ejecuta cuando un usuario inicia (o reanuda) una sesión de X-Windows utilizando el Cliente QVD. El script se ejecutará después de que se hayan activado todos los `hooks` de aprovisionamiento.

También recibe los parámetros siguientes de forma predeterminada:

- *qvd.vm.user.name*: el inicio de sesión del usuario.
- *qvd.vm.user.groups*: grupos a los que pertenece el usuario.
- *qvd.vm.user.home*: el directorio del usuario en /home.

Este `hook` es capaz de recibir otros parámetros de conexión y cualquier parámetro adicional asignado a la VM dentro del QVD-DB.

preconectar

Clave: **vma.on_action.pre-connect**

Este `hook` se ejecuta cuando un usuario inicia (o reanuda) una sesión de X-Windows desde el Cliente QVD, con la diferencia de que activará un script para ejecutarse antes de que se implementen cualquiera de los `hooks` de aprovisionamiento.

Los parámetros para la pre-conexión son los mismos que para la conexión.

detener

Clave: **vma.on_action.stop**

Este `hook` se ejecuta cuando una sesión de X-Windows recibe una solicitud para ser cerrada. Este comportamiento suele producirse cuando el VMA recibe una solicitud de este tipo del QVD-WAT o de la utilidad de administración de la CLI de QVD.

No hay parámetros adicionales para este `hook`.

suspender

Clave: **vma.on_action.suspend**

Este `hook` se ejecuta cuando se suspende una sesión de X-Windows. Esto suele ocurrir si un usuario cierra la aplicación de cliente QVD.

No hay parámetros adicionales para este `hook`.

apagado

Clave: **vma.on_action.poweroff**

Este `hook` se ejecuta cuando se cierra la máquina virtual.

No hay parámetros adicionales para este `hook`.

expirar

Clave: **vma.on_action.expire**

Este `hook` se ejecuta cuando se alcanza un límite de caducidad `soft` en la máquina virtual. Normalmente, esto se utilizará para pedir al usuario que reinicie lo más pronto posible para actualizar una imagen de disco.

No hay parámetros adicionales para este `hook`.

Note
No hay `hook` para el límite de caducidad dura, cuando se alcanza el límite el HKD reiniciará por la fuerza la máquina virtual.

Hooks de estado

Los `hooks` de estado se ejecutan cuando se producen cambios dentro de la sesión de X-Windows. Estos `hooks` siempre recibirán el parámetro *qvd.hook.on_state* con el estado X-Windows actual.

conectado

Clave: **vma.on_state.connected**

Este `hook` se ejecuta una vez que se ha establecido correctamente una conexión entre el Cliente QVD y el servidor X-Windows que se ejecuta en la máquina virtual.

suspendido

Clave: **vma.on_state.suspended**

Este `hook` se ejecuta una vez que el usuario cierra el Cliente QVD y la sesión X-Windows está en estado suspendido.

detenido

Clave: **vma.on_state.disconnected**

Este `hook` se ejecuta cuando termina la sesión de X-Windows.

Hooks de aprovisionamiento

Los `hooks` de aprovisionamiento reciben los mismos parámetros que están disponibles para el `hook` de acción *connect*.

agregar usuario

Clave: **vma.on_provisioning.add_user**

Cuando un usuario está conectado por primera vez, si el usuario todavía no existe, una nueva cuenta se crea para él en la máquina virtual.

De forma predeterminada, la cuenta se crea con el comando `useradd`.

El `hook` *add_user* permite a un administrador modificar este proceso y crear la cuenta de usuario utilizando un método alternativo o una secuencia de comandos.

after_add_user

Clave: **vma.on_provisioning.after_add_user**

Una vez creada la cuenta de usuario, este `hook` se puede utilizar para realizar acciones adicionales relacionadas con la configuración de la cuenta de usuario dentro de la máquina virtual, como la configuración automática de un cliente de correo electrónico u otras tareas similares.

mount_home

Clave: **vma.on_provisioning.mount_home**

De forma predeterminada, QVD monta la primera partición del dispositivo configurada con la entrada vma.user.home.drive en el directorio "/home" donde se crea el directorio principal del usuario (por el `hook` *add_user*). Si esta partición no existe, se crea sobre la marcha.

Con este `hook` es posible cambiar este proceso para que se produzca algún otro comportamiento, como montar un directorio /home desde un servidor NFS.

Part V

Procedimientos operacionales

En esta parte del manual, cubriremos temas relacionados con los procedimientos operativos cotidianos, como copias de seguridad y logs, junto con algunos de los comandos más utilizados en el QVD para controlar el acceso a un nodo servidor o el realizamiento de tareas administrativas básicas.

Chapter 19

Copias de seguridad

Copia de seguridad de QVD-DB

Puesto que QVD hace uso de la base de datos PostgreSQL, la copia de seguridad de sus datos QVD puede hacerse mediante comandos estándar de copia de seguridad y restauración de PostgreSQL.

Para hacer una copia de seguridad de su base de datos, simplemente puede ejecutar el siguiente comando para guardar el contenido de la base de datos en el archivo *qvddb.sql*:

```
# sudo su - postgres
# pgdump qvd > qvddb.sql
```

Restaurar la base de datos es tan simple como canalizar el contenido SQL de nuevo contra el cliente pgsql:

```
# sudo su - postgres
# pgsql qvd < qvddb.sql
```

PostgreSQL también le da la capacidad de canalizar el contenido de una base de datos a otra, por lo que es relativamente simple replicar la base de datos:

```
# pgdump -h host1 qvd | pgsql -h host2 qvd
```

Para requisitos mas complejos de copia de seguridad, consulte directamente la documentación de PostgreSQL en http://www.postgresql.org/docs/8.3/static/backup-dump.html para obtener más información.

Recuerde que una vez que haya volcado su base de datos al archivo, se debe hacer una copia de seguridad del archivo siguiendo su estrategia habitual de copia de seguridad.

Tenga en cuenta que también puede encontrar que es útil hacer copias de seguridad de los archivos de configuración de su base de datos, de modo que si necesita reinstalar y configurar su base de datos, podrá hacerlo rápidamente y con los datos de configuración disponibles. En los sistemas Ubuntu, estos archivos suelen estar en */etc/postgresql/9.2/main*. En SUSE Linux , los encontrará en */var/lib/pgsql/data*.

Copia de seguridad del almacenamiento compartido

Todas las imágenes de disco de QVD, directorio personal del usuario y datos de overlay se almacenan normalmente en alguna forma de almacenamiento compartido accesible mediante un protocolo de intercambio de archivos de red como NFS, GFS o OCFS2. Aunque los datos específicos, como los datos de overlay y las imágenes almacenadas en el directorio de *staging*, no son críticos durante la recuperación de desastres, recomendamos que estos datos se respalden junto con imágenes de disco activas y directorio personal de usuario si es posible.

Entender cómo se almacenan los archivos en el almacenamiento compartido al que accede QVD le ayudará a planificar una estrategia de copia de seguridad razonable. Para obtener más información, consulte la sección titulada Almacenamiento compartido.

Copia de seguridad de los archivos de configuración

Dado que la mayoría de los datos de configuración de QVD se almacenan en la base de datos y los archivos de configuración de QVD son relativamente sencillos de crear, no suelen considerarse prioritarios dentro de una estrategia de copia de seguridad. No obstante, para casi todos los componentes dentro de la infraestructura QVD, los archivos de configuración se almacenan en /etc/qvd. Tenga en cuenta que todos los nodos del servidor QVD deben tener archivos de configuración idénticos (exceptuando el parámetro hostname), por lo que no tiene por qué almacenar más que una copia.

Chapter 20

Logging

Registros de la base de datos

Puesto que QVD hace uso de la base de datos Open Source PostgreSQL, las opciones de `logging` se controlan editando los archivos de configuración de PostgreSQL. Para cambiar los parámetros de `logging`, consulte la documentación de PostgreSQL en: http://www.postgresql.org/docs/8.3/static/runtime-config-logging.html

En Ubuntu, PostgreSQL mantiene los logs de la base de datos en: /var/log/postgresql/. En SUSE, PostgreSQL mantiene los logs de la base de datos en: /var/lib/pgsql/data/pg_log.

Tip
Si es nuevo en PostgreSQL, puede encontrar útil la herramienta pgAdmin, especialmente diseñada para supervisar el estado del servidor, los logs y las transacciones. Puede obtener más información sobre la herramienta en http://www.pgadmin.org/

Registros de nodo servidor QVD

Los nodos del servidor QVD también guardan sus propios archivos de log. Estos se encuentran normalmente en /var/log/qvd.log. La salida y *facilities* se controlan utilizando el módulo de `logging` para perl Log4perl. La configuración del log en QVD puede controlarse estableciendo varios parámetros de configuración. Éstos se tratan con mayor profundidad en la sección de Configuración de Log.

Registros de máquina virtual de QVD

El VMA se instala dentro de la imagen de disco que es utilizada por cada máquina virtual cuando se inicia. De forma predeterminada, el VMA escribirá sus logs localmente dentro de la máquina virtual, pero puede configurarse opcionalmente para iniciar sesión con un demonio compatible con syslog en el nodo host o en un servidor remoto.

Registro local

Si el log no está configurado en el vma.conf de una máquina virtual, por defecto quedará registrado en su propio archivo en /var/log/qvd.log dentro de la máquina virtual. Esto se puede establecer explícitamente, o cambiar, dentro del archivo vma.conf de la siguiente manera:

```
log.level = DEBUG
log.filename = /var/log/qvd/qvd.log
```

El nivel de log en sí puede ser uno de ALL, DEBUG, INFO, WARN, ERROR, FATAL, OFF.

Note
El contenido que no sean datos de usuario que se escribe en el disco dentro de una máquina virtual hará uso de las capacidades de overlay proporcionadas por QVD.

Para revisar los datos de log escritos en un archivo dentro de una máquina virtual, necesitará tener acceso a la máquina virtual en ejecución a través de la consola:

```
# qa vm console -f id=1
```

Note
Observe que esta es la utilidad qa legacy. La nueva utilidad qa4 no soporta esta función y no está previsto que lo haga.

Si está utilizando la virtualización LXC, a menudo es más fácil acceder a los archivos de log directamente desde el nodo servidor donde se está ejecutando. Recuerde que para un entorno en ejecución, su sistema de archivos se construye a medida que se inicia y se monta en /var/lib/qvd/storage/rootfs. Esto significa que para cualquier máquina virtual es posible ver directamente los archivos de log desde el host principal donde se está ejecutando:

```
# tail /var/lib/qvd/storage/rootfs/1-fs/var/log/qvd.log
```

QVD también almacena una copia de seguridad del overlay para cualquier máquina virtual LXC que no se inicie correctamente. Estas copias de seguridad se pueden ver en /var/lib/qvd/storage/overlayfs y normalmente se guardan con el prefijo *deleteme-*, siguiendo una convención similar a la seguida para el nombramiento de overlays con éxito. Para obtener más información, consulte overlayfs.

Registro remoto

Registrar la actividad remotamente en un demonio que soporte el protocolo syslog puede ser deseable por un par de razones. En primer lugar, mantiene los logs para todas las máquinas virtuales que se hayan configurado así en un lugar donde el acceso a los logs resulta más fácil y lógico. En segundo lugar, en una situación en la que el administrador puede no ser capaz de acceder al log de una máquina virtual por alguna razón, por ejemplo, si no se está iniciando, el logging remoto puede ayudar a identificar el problema.

Para configurar el logging remoto, necesitará un servidor de logging remoto configurado y realizar algunos cambios dentro de la imagen de disco, tanto en el archivo vma.conf de QVD como en la configuración de syslog para enviar mensajes syslog al servidor de logging remoto.

Para demostrar esto usaremos el enlace Rsyslog, que se ha convertido en la utilidad de logging por defecto para muchas de las principales distribuciones de Linux en los últimos años, incluyendo Ubuntu, SUSE y Red Hat, y es confiable y fácil de configurar. Debido a que QVD utiliza Log4perl, debería ser independiente del demonio de logging destino, por lo que podría utilizar estas instrucciones con syslog-ng entre otras alternativas si es necesario.

Si rsyslog no estuviera disponible en su servidor, instálelo de la siguiente manera para Ubuntu:

```
# apt-get install rsyslog rsyslog-relp
```

O, si utiliza SUSE:

```
# zypper in rsyslog rsyslog-module-relp
```

Warning
Esto probablemente desinstalará cualquier otro programa syslog conflictivo que tenga, así que asegúrese de que esto sea aceptable para la máquina (nodo QVD o no) que está utilizando.

Una vez hecho esto, tendremos que configurar rsyslog para aceptar conexiones remotas. En este ejemplo, usaremos el enlace: Reliable Event Logging Protocol (RELP), ya que consideramos que es justo para eso, pero por supuesto puede usar TCP o UDP como usted prefiera. Para configurar rsyslog para usar RELP, cree el archivo `30-remote.conf` en la carpeta `/etc/rsyslog.d/` e ingrese la siguiente configuración:

```
$ModLoad imrelp
$InputRELPServerRun 2514

$template remotefile,"/var/log/%HOSTNAME%-%syslogfacility-text%.log"
*.* ?remotefile
```

Esto carga el módulo de entrada RELP y establece que el servidor escuche en el puerto 2514. A continuación, indica a rsyslog que genere el nombre de archivo del log dinámicamente, dependiendo del nombre de host del cliente. La siguiente línea le dice a rsyslog que registre todos los mensajes en este archivo formado dinámicamente. Ahora, reinicie rsyslog:

```
# service rsyslog restart
```

A continuación, tendrá que configurar la imagen QVD para registrar remotamente a este servidor. Dentro de la imagen QVD que va a utilizar, cree en la carpeta `/etc/rsyslog.d/` un archivo llamado `00-remote.conf` e ingrese la siguiente configuración:

```
$ModLoad omrelp
*.* :omrelp:<hostname or IP address>:2514
```

Asegúrese de ingresar la dirección IP o el nombre de host del servidor de logging. Esta configuración indicará a rsyslog en la máquina virtual que tiene que cargar el módulo de salida RELP y usar el puerto 2514 en su servidor rsyslog. Además, mandará toda la salida (*.*) a este host remoto.

Note
Asegúrese de que el módulo RELP está disponible, y si no instálelo (el paquete es `rsyslog-relp` en Ubuntu y `rsyslog-module-relp` en SUSE).

Finalmente, edite el archivo `/etc/qvd/vma.conf` en la máquina virtual e ingrese lo siguiente para indicar a QVD que deje sus logs en syslog:

```
log4perl.appender.SYSLOG = Log::Dispatch::Syslog
log4perl.appender.SYSLOG.layout = Log::Log4perl::Layout::PatternLayout
log4perl.appender.SYSLOG.layout.ConversionPattern = %d %P %F %L %c - %m%n
log4perl.rootLogger = DEBUG, SYSLOG
log.level = DEBUG
```

Con esta configuración, su imagen debe mandar el log de QVD al rsyslog remote que haya configurado. Para comprobarlo, use el comando *logger* dentro de su imagen y la salida debe estar en los logs del servidor de logging.

Chapter 21

Comandos usados comúnmente

Al realizar tareas de administración habituales, puede resultar complicado trabajar con el WAT, incluso a pesar de las mejoras de este en la última versión. La línea de comandos es particularmente útil, si necesita realizar comportamientos programados o automáticos. Para los administradores de sistemas, incluimos esta sección para proporcionar ejemplos de algunos de los comandos más utilizados que se pueden emitir con la Utilidad de administración CLI de QVD. Si necesita más orientación sobre los comandos disponibles, consulte de nuevo el capítulo Utilidad de administración CLI de QVD> en el que hablamos de la utilidad en detalle.

Administración de nodos servidor QVD

Al realizar el mantenimiento en un nodo servidor, es común que se requiera que los usuarios no puedan acceder al nodo servidor mientras está trabajando. Esto se puede lograr fácilmente utilizando la utilidad de administración de CLI de QVD:

```
# qa4 host name~'qvd%' block           # bloquear el acceso a cualquier nodo cuyo nombre  ↩
    comience con 'qvd'
# qa4 host address='192.168.0.2' get   # listar los detalles del servidor con la IP  ↩
    '192.168.0.2'
```

Una vez que haya terminado de realizar el mantenimiento, recuerde desbloquear el host:

```
# qa4 host name~'qvd%' unblock   # desbloquear el acceso a cualquier servidor cuyo nombre  ↩
    comience con 'qvd'
```

Important
Es crucial que todos los nodos de una instalación QVD estén sincronizados en el tiempo, es decir, que el uso de NTP es esencial para evitar que el sistema se comporte de manera impredecible. A menos que esté utilizando Xen o similar y todos los nodos estén sincronizadas por la máquina anfitriona, necesitará instalar el paquete NTP apropiado para su sistema (llamado `ntp` para Ubuntu y SLES) en cada sistema que vaya a ser un nodo, y configurar cada uno para sincronizar con un servidor NTP central. Dado que sus nodos no tienen por qué tener acceso a Internet, puede ser una buena idea tener un dispositivo en su red que actúe como el servidor de hora local para los demás, y esto también facilita la correlación de eventos del sistema. Esto está más allá del alcance de esta guía, por favor vea http://ntp.org para más información.

Administración de VM

Al realizar el mantenimiento en máquinas virtuales o al actualizar imágenes, a menudo es necesario impedir el acceso a un grupo de usuarios u obligarlos a desconectarse. Los ejemplos siguientes proporcionan algunos usos comunes de la manutención de máquinas virtuales con la utilidad de administración CLI de QVD:

```
# qa4 vm user_id=5 block            #Bloquear el acceso a máquinas cuyo usuario tenga el id ↩
    =5
# qa4 vm name=test unblock          #Desbloquear el acceso a cualquier máquina con name= ↩
    test
# qa4 vm id=23 disconnect_user      #Forzar la desconexión del usuario en la máquina con id ↩
    =23
# qa4 vm id=1 start                 #Arranque manual de la máquina con id=1
# qa4 vm osf_id=2 stop              #Parada manuald de todas las máquinas pertenecientes al ↩
     osf con id=2
# qa4 vm user_id=5 del              #Borrar todas las máquinas pertenecientes al usuario  ↩
    con id=5
```

Part VI

ANEXOS

Chapter 22

Utilidad de administración Legacy CLI de QVD (3.X)

La utilidad QVD de administración en línea de comandos es un script perl que puede interactuar con la QVD-DB para realizar una amplia gama de operaciones dentro de la infraestructura QVD. Se puede utilizar como una alternativa a la herramienta de administración web QVD (QVD-WAT) y puede ser instalada en cualquier sistema con acceso a la QVD-DB.

Important
A partir de QVD 3.1.1, el archivo `/usr/lib/qvd/bin/qvd-admin.pl` se ha vinculado simbólicamente a `/usr/bin/qa`. Las versiones futuras de QVD utilizarán `qa` como la herramienta de línea de comandos QVD. Por supuesto, puede utilizar la ruta de acceso al script perl.

Instalación y configuración de la utilidad de administración de CLI de QVD

En cualquiera de los sistemas en los que va a instalar la Administración de CLI de QVD , deberá agregar el repositorio QVD a las fuentes de apt.

En primer lugar, agregue la clave pública de los paquetes QVD a sus claves de confianza (como root):

```
# wget -qO - https://www.theqvd.com/packages/key/public.key | sudo apt-key add -
```

Ahora, agregue el repositorio:

```
/etc/apt/sources.list.d/qvd.list
# apt-get update
```

Important

Para instalar la Utilidad de administración CLI de QVD, ejecute el siguiente comando:

```
# apt-get install perl-qvd-admin
```

El proceso es similar para SLES.

En primer lugar, agregue la clave pública de los paquetes QVD a sus claves de confianza (como root):

```
# rpm --import https://www.theqvd.com/packages/key/public.key
```

Ahora, agregue el repositorio:

```
# zypper ar http://theqvd.com/packages/sles/12SP1/stable QVD
# zypper ref
```

Para instalar la utilidad de administración CLI de QVD, ejecute el siguiente comando:

```
# zypper install perl-QVD-Admin
```

La utilidad de administración de QVD requiere acceso a la base de datos QVD. Deberá asegurarse de que el archivo de configuración del nodo QVD esté configurado correctamente para que esta herramienta funcione como debe. Puede averiguar cómo hacerlo en el capítulo de QVD-DB en la sección titulada Configuración de la base de datos QVD.

Listado de comandos QVD CLI

La utilidad de administración CLI de QVD proporciona un gran conjunto de funciones administrativas que pueden ser utilizadas para controlar componentes y elementos que están involucrados en el entorno QVD.

Puede obtener una lista completa de las funciones mediante el parámetro *--help*:

```
root@altar:~# qa --help
Valid command expected, available subcommands:
   config del
   config get
   config set
   config ssl
   di add
   di del
   di list
   di tag
   di untag
   host add
   host block
   host del
   host list
   host propdel
   host propget
   host propset
   host unblock
   osf add (*)
   osf del (*)
   osf list
   user add
   user del
   user list
   user passwd
   user propdel
   user propget
   user propset
   vm add
   vm block
   vm del
   vm disconnect_user
   vm edit
   vm list
   vm propdel
   vm propget
```

```
  vm propset
  vm ssh
  vm start
  vm stop
  vm unblock
  vm vnc
```

Tip
Cualquiera de los comandos presentados anteriormente puede ser invocado con el parámetro `--help` para obtener una descripción más detallada de la sintaxis.

Uso de filtros para controlar operaciones

Muchas de las operaciones disponibles a través de la utilidad de administración CLI de QVD también admiten filtros (utilizando el modificador -f) para limitar una acción a un elemento o entidad particular. Los filtros son esencialmente coincidencias con los elementos dentro de las columnas de la tabla, tal y como proporcionaría una consulta SQL estándar en la declaración *WHERE*. Los filtros aceptan el asterisco (*) como carácter comodín y (,) como separador para construir cadenas de declaraciones AND. Los filtros más utilizados son el ID de un elemento o el nombre. Usando el `host list` como ejemplo, puede limitar las entradas devueltas por filtrado en `id` o `nombre`:

```
# qa host list -f name=sha*

Id Name    Address      HKD       Usable RAM Usable CPU VMs assigned Blocked State
-------------------------------------------------------------------------------
1  shamash 192.168.0.12 103:14:31 296.676    16172.48   0            0       starting
```

Cuando se utiliza la utilidad de administración CLI para ver el estado de la VM, es común utilizar filtros para ver máquinas virtuales en un estado concreto o pertenecientes a un usuario determinado o donde una máquina virtual tiene un *user_state* particular. El siguiente ejemplo le mostrará cómo encadenar juntos una serie de filtros para ver todas las máquinas virtuales que pertenecen a usuarios con nombres de usuario que comienzan por *al*, donde además la máquina virtual se está ejecutando y el usuario está conectado:

```
# qa vm list -f user=al*,state=running,user_state=connected
Id Name    User    Ip             OSF  DI_Tag  DI             Host      State   UserState  ↩
       Blocked
-------------------------------------------------------------------------------

1  alison  alison  172.20.127.254 test default 2012-03-05-000 qvd_test  running connected  ↩
       0
7  antony  antony  172.20.127.232 live default 2012-02-15-000 qvd_test2 running connected  ↩
       0
```

Operaciones administrativas básicas

En esta sección examinaremos algunas de las tareas administrativas más comunes para las que se utiliza la utilidad de administración CLI de QVD.

Cambiar los ajustes de configuración de QVD

QVD tiene una amplia gama de ajustes de configuración muy específicos que controlan varios componentes dentro de la infraestructura. Discutimos algunos de ellos aquí.

Para cambiar un parámetro de configuración de QVD mediante la utilidad de administración CLI de QVD, puede hacer lo siguiente:

```
# qa config set myproperty="this is a value"
```

También es posible obtener todos los ajustes de configuración actuales de la base de datos y enumerarlos:

```
# qa config get
```

Por último, es posible eliminar un parámetro de configuración QVD de la base de datos:

```
# qa config del myproperty
```

Añadir un nodo servidor QVD

Es común utilizar la utilidad de administración CLI de QVD para agregar nuevos nodos servidor QVD a la base de datos de QVD. Esto se puede hacer muy rápidamente desde la línea de comandos con la siguiente directriz:

```
# qa host add name=NewNode address=192.168.0.12
```

La eliminación de un nodo de servidor QVD es igual de sencilla:

```
# qa host del -f "name=NewNode"
```

Configurando SSL para QVD

Los nodos de servidor QVD deben configurarse para utilizar SSL. Actualmente, la única manera de hacerlo es mediante el uso de la utilidad de administración CLI de QVD:

```
# qa config ssl --help

config ssl: Sets the SSL certificate and private key
usage: config ssl key=mykey.pem cert=mycert.pem

    Sets the SSL certificate to the one read from the file mycert.pem, and the
    private key to the one read from mykey.pem.

    Example: config ssl key=certs/server-key.pem cert=certs/server-cert.pem
```

Se recomienda que siempre que sea posible utilice un certificado firmado por CA de confianza.

Añadir un OSF

Puede agregar fácilmente un OSF a QVD usando la utilidad de Administración CLI de QVD si está en un host que tiene acceso al almacenamiento compartido donde se almacenan las "imágenes":

```
# qa osf add name=myOSF use_overlay=no memory=1024 user_storage_size=2048
```

Sólo hay un valor obligatorio para agregar un OSF, que es **name**. Si los otros parámetros se dejan sin especificar, sus valores por defecto se utilizan en su lugar. Estos son:

- * Memoria * = 256
- * Use_overlay * = y
- * User_storage_size * = undef (sin límite para el almacenamiento de usuarios)

Puede obtener una lista de OSF disponibles actualmente haciendo lo siguiente:

```
# qa osf list
```

Añadir un DI

Utilizando la utilidad de administración CLI de QVD, puede adjuntar una imagen de disco (DI) a cualquier OSF existente dentro del sistema. Este proceso puede llevar tiempo, ya que además de actualizar la base de datos el archivo de imagen de disco real es copiado en el directorio `storage/images` dentro del almacenamiento compartido.

Al adjuntar un DI a un OSF particular, mantenemos separado el disco real de la imagen que se servirá a los usuarios finales. Esto significa que puede realizar cambios en la imagen de disco y simplemente actualizar el OSF, de modo que cuando un usuario vuelve a conectar la imagen es automáticamente actualizada sin que el usuario experimente ninguna discontinuidad en el servicio.

```
# qa di add path=/var/lib/qvd/storage/staging/qvd-guest.img osf_id=1
```

Ambos **path** y **osf_id** son obligatorios para agregar un DI. Cuando se agrega la DI, la imagen especificada en la ruta se copia en el área de almacenamiento de sólo lectura configurada para almacenar DIs activas (normalmente `/var/lib/qvd/storage/images`).

Puede obtener una lista de imágenes disponibles actualmente haciendo lo siguiente:

```
# qa di list
```

Etiquetar DI

Los DI pueden ser etiquetados con cadenas arbitrarias a voluntad. Para etiquetar un DI como predeterminado, use el comando **di tag**:

```
# qa di tag di_id=42 tag=default
```

Puede etiquetar DIs con cualquier cadena, no sólo **default** o **head**. Esto le permite usar nombres significativos para las etiquetas, por ejemplo "software_bug_fixed", para su uso dentro del campo **DI Tag** de las máquinas virtuales.

Las etiquetas son útiles, ya que permiten adjuntar una nueva versión de una imagen de disco a un OSF sin afectar a nadie que esté usando la imagen actual o predeterminada para un OSF. Esto le permite implementar un cambio y migrar determinadas máquinas virtuales utilizando un OSF distinto para la nueva imagen mientras lo prueba. Si la imagen falla por alguna razón o no cumple con sus requisitos, es sencillo volver a la imagen predeterminada y permitir que sus usuarios continúen trabajando mientras hace correcciones.

Seleccionando la etiqueta DI que las máquinas virtuales utilizarán

Con el fin de decirle a una máquina virtual que debe utilizar una DI que tiene una etiqueta específica, editamos la VM Para cambiar su campo `di_tag`. Así que si por ejemplo acabamos de corregir un error software en un DI y establecemos la etiqueta "software_bug_fixed", podemos usar ese DI en una VM usando el siguiente comando:

```
# qa vm edit di_tag=software_bug_fixed -f vm_id=42
```

En el siguiente arranque de la VM con ID 42, utilizará el DI con esta etiqueta.

Agregar y eliminar usuarios

Es común utilizar la utilidad de administración CLI de QVD para agregar y quitar usuarios rápidamente.

```
# qa user add login=peter password=s3cr3t
# qa user del -f login=guest3
```

También puede enumerar todos los usuarios de QVD mediante la opción de lista:

```
# qa user list
```

Tenga en cuenta que, como se explica en la Guía de instalación, no es aconsejable crear un usuario cuyo nombre de usuario ya exista en cualquier imagen de disco.

Restablecimiento de una contraseña de usuario

Puede cambiar la contraseña de un usuario mediante la Utilidad de administración CLI de QVD:

```
# qa user passwd user=guest
```

En el ejemplo anterior, estamos cambiando la contraseña para el usuario de inicio de sesión *guest*. Se le pedirá que proporcione una nueva contraseña.

Añadir y eliminar máquinas virtuales

Añadir y eliminar máquinas virtuales mediante la utilidad de administración CLI de QVD es fácil. Puede especificar el usuario y el OSF por ID o por nombre:

```
# qa vm add name=GuestVM user_id=1 osf_id=1
# qa vm add name=GuestVM user=peter osf=myOFS
```

Puede eliminar fácilmente una máquina virtual utilizando el siguiente comando:

```
# qa vm del -f "name=GuestVM"
```

Inicio y detención de máquinas virtuales

La utilidad de administración CLI de QVD se puede utilizar para iniciar y detener máquinas virtuales. Si no se especifica con un filtro en particular, la acción será iniciar o detener todas las maquinas virtuales. Normalmente se ejecuta este comando especificando un filtro para identificar la máquina virtual real que desea iniciar o detener. Ejemplos:

```
# qa vm stop -f "user=guest*"
# qa vm start -f "id=1"
```

La política de balanceo determina el nodo donde arrancará la VM.

Bloqueo y desbloqueo de máquinas virtuales

Las Máquinas Virtuales se pueden marcar como "bloqueadas". Cuando estén en este estado, la aplicación QVD Client no podrá conectarse a la Máquina Virtual. Esta tarea puede ser ejecutada por un administrador para realizar una tarea administrativa o puede tener lugar cuando el HKD no sea capaz de gestionar correctamente una máquina virtual. Los comandos siguientes se pueden utilizar para marcar una máquina virtual como "bloqueada" o se puede utilizar para desbloquear una máquina virtual que se ha establecido en este estado.

```
# qa vm block -f "id=2"
# qa vm unblock -f "name=GuestVM"
```

Consulte *bloqueo y desbloqueo de la máquina virtual* en el manual del WAT para obtener más información sobre cómo configurar este estado.

Solución de problemas de máquinas virtuales

Note
Dado que esta herramienta es Legacy, es posible que estas opciones no funcionen.

La Utilidad de administración de CLI de QVD también proporciona opciones para conectarse a una máquina virtual sin utilizar la aplicación cliente. Esto es útil para depurar una imagen que no funciona en QVD. Actualmente, las opciones soportadas incluyen la consola de puerto serie, acceso SSH y VNC. Para acceder a la consola serie de una máquina virtual, utilice el siguiente comando:

```
# qa vm console -f id=1
```

Esto abrirá una sesión al puerto serie de la máquina virtual con el *Id* de *1*. Si está utilizando la virtualización KVM, su máquina virtual necesita tener la consola en serie activada: vea por ejemplo el Serial Console HOWTO for Ubuntu.

Para entrar por ssh en una máquina virtual, ejecute el siguiente comando:

```
# qa vm ssh -f name=myVM -- -l qvd
```

Esto abrirá una conexión SSH a la máquina virtual llamada *myVM* utilizando el nombre de usuario *qvd*. Su máquina virtual necesita tener OpenSSH instalado y configurado.

Si está utilizando la virtualización KVM con el acceso VNC habilitado y tiene `vncviewer` instalado, puede ejecutar el siguiente comando para abrir un VNC.

```
# qa vm vnc -f name=myVM
```

La consola VNC no está disponible cuando se utiliza LXC.

Configuración de propiedades personalizadas para una máquina virtual

QVD incluye la opción de configurar propiedades personalizadas para una máquina virtual que puede establecer y recuperar mediante la utilidad de administración CLI de QVD. Esto es útil si necesita escribir sus propios comportamientos o desea aprovecharse de los `hooks` VMA (Vea el apartado al respecto).

Las propiedades personalizadas se utilizan a menudo cuando usted escribe sus propios plugins para el L7R, como los módulos de autenticación o balanceo de carga.

Las propiedades personalizadas son compatibles con los parámetros de configuración: **host**, **user** y **vm**

Para agregar una propiedad personalizada, puede usar el comando $_{\text{propset}}$:

```
# qa user propset beverage=beer -F login=rowan
propset in 1 users.
```

Se puede obtener el contenido de todas las propiedades personalizadas que se han establecido para un parámetro de configuración utilizando el comando $_{\text{propget}}$:

```
# qa user propget
rowan   beverage=beer
```

Finalmente, puede eliminar una propiedad personalizada mediante el comando $_{\text{propdel}}$:

```
# qa user propdel beverage
Are you sure you want to delete the prop in all users? [y/N] y
```

Part VII

Glosario

QVD

Virtual Quality Desktop, un conjunto de componentes de servidor y una aplicación cliente que proporciona acceso remoto de escritorio virtual a los usuarios.

QVD Client

Un cliente NX modificado capaz de conectarse a una máquina virtual que se ejecuta en un nodo servidor QVD. El cliente está disponible para los sistemas operativos Linux, Windows y MAC OSX. También existen versiones beta para IOS y Android.

QVD-DB

La base de datos QVD. Esto se instala sobre un servidor PostgreSQL RDBM. Todos los componentes del lado servidor dentro de la infraestructura QVD dependen de la base de datos para comunicarse entre sí y para implementar la funcionalidad.

QVD Server Node

Un host que ejecuta los componentes QVD Server Node, incluyendo el HKD e instancias del L7R (dependiendo de si hay clientes conectados o no). Normalmente hay varios nodos servidor QVD dentro de una implementación típica. Las máquinas virtuales a las que accede el cliente QVD se ejecutan en diferentes nodos servidor QVD.

QVD-WAT

La Herramienta de Administración Web de QVD. Se trata de una interfaz gráfica de usuario basada en web que permite a un administrador configurar y supervisar el funcionamiento del entorno QVD.

QVD CLI Administration Utility

Una secuencia de comandos Perl que proporciona una interfaz de línea de comandos con la que se puede supervisar y administrar el entorno QVD.

HKD

El House Keeping Daemon es un demonio del nodo servidor QVD. Es responsable de iniciar y detener máquinas virtuales y de realizar comprobaciones de estado de la máquina virtual. El HKD también crea un proceso L7R para cada conexión entrante.

L7R

El Layer-7 Router actúa como intermediario para todas las conexiones del Cliente QVD. Es invocado por el HKD. Es responsable de autenticar a los usuarios y de encaminar las solicitudes de cliente al nodo servidor apropiado que ejecuta la máquina virtual para el usuario autenticado. También supervisa el estado de la sesión.

VMA

El Virtual Machine Agent es un componente QVD que se ejecuta dentro de una máquina virtual para facilitar la conectividad del cliente con el escritorio virtual y que es responsable de escuchar las solicitudes de administración enviadas por el HKD. También proporciona una serie de `hooks` que permiten a un administrador personalizar comportamientos dentro de la máquina virtual.

VMA Hook

Una instalación dentro del VMA que permite activar otra funcionalidad (generalmente a través del uso de scripts) dentro de la máquina virtual, basada en cambios de estado particular dentro de la máquina virtual.

Máquina virtual

Una máquina virtual es un sistema virtualizado que funciona encima de otro sistema operativo. Por lo general, el sistema virtualizado carga un OSF con el fin de ejecutar un sistema operativo virtual.

OSF

El Operating System Flavour es una entidad que permite al administrador crear "sabores" o tipologías de máquina virtual. Así el administrador puede asignar a los usuarios máquinas de estos "sabores", según sus necesidades. El OSF suele configurarse en QVD junto con parámetros de tiempo de ejecución específicos, como la cantidad de memoria del sistema que debe estar disponible para él o la persistencia de su disco; también incluye el set de imágenes disponibles para el osf diferenciadas por etiquetas.

DI

Una Disk Image es una imagen qcow2 que se ha creado como un disco virtual que contiene un sistema operativo instalado. En el caso LXC esta imagen es en realidad un disco de imagen de sistema operativo completo comprimido con tar-gz. Estas imágenes se asocian a un OSF.

Usuario

Persona que utiliza el servicio QVD, normalmente conectado mediante el Cliente QVD.

Administrador

Un usuario que tiene permiso para acceder a la plataforma de gestión, generalmente a través de QVD-WAT o a través de la utilidad de administración de QVD CLI

Session

Período en el que un usuario está realmente conectado a una máquina virtual.

KVM

Máquina virtual del kernel. Se trata de un hipervisor que se instala en el kernel de Linux para lograr la virtualización de tipo 1. QVD hace uso de KVM para ejecutar las máquinas virtuales necesarias para servir a los usuarios finales.

LXC

Contenedores de Linux. Esta es una tecnología de virtualización incluida en el kernel de Linux. Utiliza un enfoque similar al comando *chroot* de Linux estándar, pero proporciona un mayor grado de separación de recursos. QVD puede hacer uso de LXC en lugar de KVM para ejecutar las máquinas virtuales necesarias para servir los escritorios virtuales a los usuarios finales. La virtualización LXC requiere mucho menos recursos, lo que le permite aprovechar mejor el hardware existente para atender a más usuarios.