EL MANUAL DE

Arquitectura QVD 4.1

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

QVD (Quality Virtual Desktop) es una solución *VDI* enfocada en Linux. El software está diseñado para virtualizar por completo el escritorio de Linux, por lo que los sistemas cliente son capaces de conectarse a un servidor central para 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 ventajas:

- Los usuarios pueden cambiar entre ordenadores en una red y seguir trabajando como si aún estuvieran en el mismo escritorio, con pleno acceso a todas sus aplicaciones y datos
- Los administradores tienen un mayor control sobre las aplicaciones que están instaladas en los sistemas de los usuarios, y son capaces de gestionar los datos de los usuarios con mayor facilidad para realizar copias de seguridad y análisis de virus, etc.
- Es más fácil para los administradores aprovisionar nuevos equipos de sobremesa y desplegar aplicaciones para los nuevos usuarios
- Hay menor tiempo de inactividad en el caso de fallos de hardware
- Los usuarios pueden hacer uso de una variedad de diferentes dispositivos para acceder a su escritorio y aplicaciones, incluyendo ordenadores portátiles, ordenadores personales y teléfonos inteligentes
- Los usuarios pueden trabajar con seguridad con el mismo escritorio y aplicaciones desde una ubicación remota sin necesidad de una *VPN*
- La mejora general del sistema y seguridad de datos
- Reducción de costes de hardware, mantenimiento y administración

El servidor **QVD** virtualiza cada escritorio Linux. Esto se puede lograr mediante el uso de una de las dos tecnologías de virtualización disponibles en el producto. Una posibilidad es *KVM (Kernel-based Virtual Machine)*, que se utiliza como un hipervisor tipo 1. La otra posibilidad, mucho más interesante, es *LXC (Linux Containers)*. Esta virtualización ayuda a mantener el entorno de cada usuario como su propia entidad discreta, para mejorar la seguridad y la estabilidad.

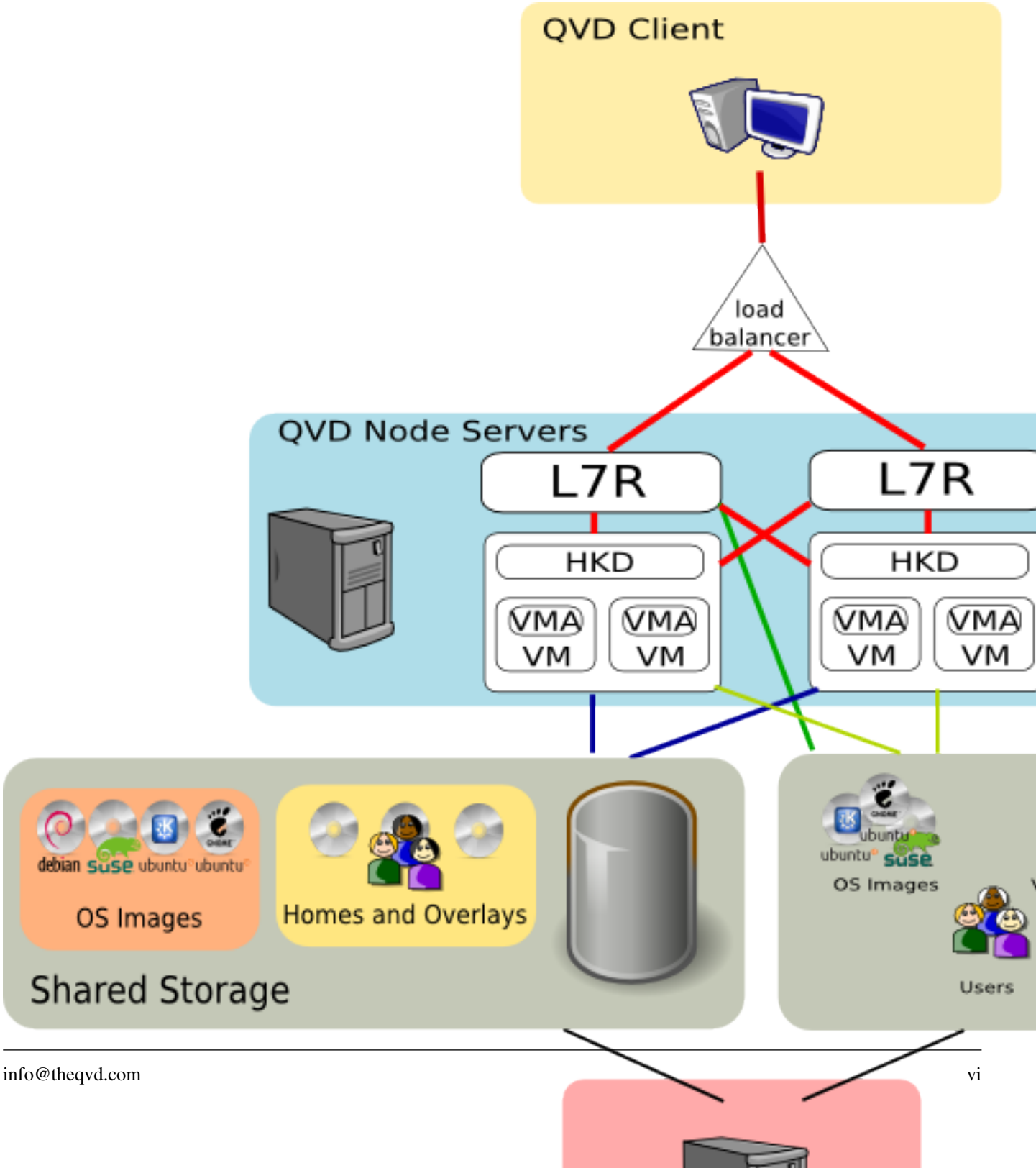
La virtualización permite servir múltiples sistemas operativos o entornos a los usuarios, en función de sus necesidades. Estos se cargan como imágenes independientes en el servidor de **QVD**. Estas imágenes proporcionan la base del sistema operativo y el entorno de trabajo, que se replican para cada máquina virtual. Cuando un usuario se conecta al servidor, haciendo uso de la aplicación cliente, una máquina virtual se inicia únicamente para ese usuario. Esto proporciona una "jaula" que impide que cualquier comportamiento inapropiado pueda afectar a otros usuarios. Cuando el usuario se desconecta, la máquina virtual se puede detener. Al detenerse, el entorno vuelve a su estado original, salvo la información propia del usuario. Esto significa que si el entorno del usuario de alguna manera se ha convertido en un problema, una simple desconexión puede revertir el sistema a su estado original. Esto proporciona un mejor nivel de seguridad que si un usuario estaba trabajando en una estación de trabajo independiente.

Con el fin de mantener los datos del usuario, tales como la configuración del escritorio, documentos y otra información específica del usuario, hay dos opciones. El primer, y más común método, es almacenar esta información en un recurso compartido *NFS*. De esta manera, los datos se pueden almacenar en un dispositivo *NAS* o dentro de un *SAN*, donde se pueda controlar 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 vuelve a cargar la imagen. Ambos enfoques son igualmente válidos. Al mantener los datos de usuario independientes de la imagen del sistema operativo, nos aseguramos de que, en el caso

de que una imagen de sistema está dañada o en caso de fallo de la misma, los administradores son capaces de reducir al mínimo el tiempo necesario para la recuperación de desastres.

El escritorio se accede desde cada estación de trabajo, haciendo uso de un cliente que utiliza el protocolo *NX* para comunicarse con el servidor. El protocolo *NX* se utiliza para manejar las conexiones *X-Windows* remotas y proporciona un nivel de compresión elevado que permite un alto rendimiento incluso cuando se accede al escritorio a través de una conexión con poco ancho de banda. Por otra parte, **QVD** es capaz de encapsular el protocolo *NX* con *SSL* para cifrar la conexión, de manera que los usuarios pueden trabajar de una manera segura incluso si se accede desde una localización remota. **QVD** proporciona software de cliente para funcionar en una variedad de sistemas operativos y dispositivos de base, desde Linux a `Windows`.

Resumen de la arquitectura de QVD



Infraestructura Una solución QVD se compone de tres componentes principales del lado servidor:

- **Nodo Servidor QVD**
- **Nodo Administración QVD**
- **Base de datos PostgreSQL**

Idealmente, cada uno de estos componentes debería estar alojado en un servidor dedicado por razones de estabilidad.

Por otra parte, aunque es probable que solo se tenga un nodo de administración y un servidor `PostgreSQL`, es posible (y recomendable), tener más de un Nodo QVD. Por tanto, la mayoría de instalaciones requerirán de los siguientes componentes extra:

- Balanceador de carga
- Almacenamiento compartido (*NFS*, pej)

Usando un balanceador de carga delante de los nodos QVD, las conexiones de cliente pueden balancearse entre los nodos sanos para permitir el acceso al escritorio virtual del usuario. Esto reduce la configuración necesaria en el software cliente y también asegura la disponibilidad de los escritorios.

Puesto que cada nodo requerirá acceso a los recursos compartidos, como las imágenes de disco para las máquinas virtuales y el home de los usuarios, se configura por lo general un sistema de almacenamiento compartido para permitir dicho acceso.

Imágenes de sistema operativo para máquinas virtuales En cada una de las imágenes de disco que se cargan en las máquinas virtuales para servir los escritorios, se requiere el siguiente componente:

- **QVD VMA (Virtual Machine Agent)**

Este agente es responsable de aceptar conexiones del cliente a través de un nodo QVD. Facilita acceso al entorno de escritorio que se está ejecutando en la máquina virtual y permite también configurar el acceso a las impresoras del lado cliente y hacer streaming de audio al mismo. Si el cliente es la version Linux, también gestiona la redirección de dispositivos USB.

Software cliente Finalmente, tenemos el componente del lado cliente:

- **Cliente QVD**

El cliente se empaqueta para diversas distribuciones de linux, para `Microsoft Windows` e incluso para `OSX`. También existen versiones para `android` e `ios`, aunque su estado actual es beta.

En la mayoría de los entornos de producción, la arquitectura de un entorno **QVD** es tal que varios nodos QVD van a ejecutarse en paralelo. El entorno está diseñado para ejecutarse con un balanceador de carga, por lo que se puede tener una solución de alta disponibilidad.

El nodo Servidor QVD

Un nodo servidor de QVD se compone de un único binario, el **HKD** o **House Keeping Daemon**. Este componente recibe todas las conexiones con un enrutador de capa 7, el **L7R (Level 7 Router)**, que garantiza que todos los clientes se enrutan a la dirección IP virtual correcta que se ha configurado para la máquina virtual del usuario que se conecta. También es responsable de autenticar al usuario antes de la conexión, y del establecimiento de la sesión del cliente. El **HKD** está a la escucha en cada nodo QVD y por cada conexión entrante, ejecuta un proceso **L7R**.

El **HKD** realiza el seguimiento del estado de las máquinas virtuales. Es responsable de iniciar y parar las máquinas virtuales, así como la vigilancia de la salud de cada máquina virtual y la actualización de la información de estado en la base de datos de QVD, haciendo que otros nodos y las herramientas de administración son capaces de funcionar en consecuencia. En general, el **HKD** es responsable de la gestión de estado de la máquina virtual.

Comportamiento del HKD

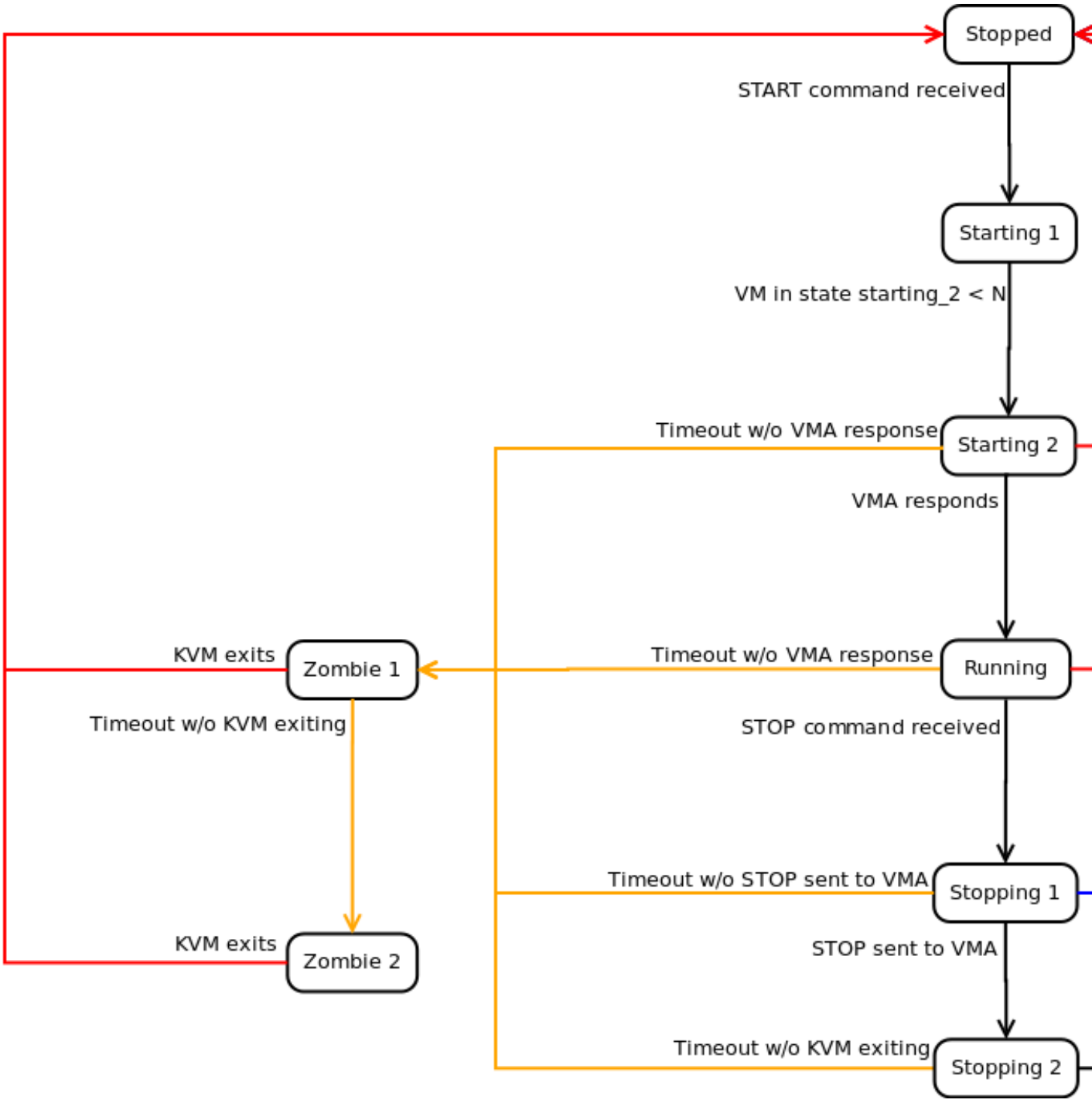
El **House Keeping Daemon** es responsable de la gestión de estados de máquinas virtuales basándose en la información que se detecta dentro de la base de datos de QVD. El **HKD** periódicamente realiza encuestas a la base de datos para determinar el estado de cada máquina virtual. Si el estado ha sido cambiado por otro elemento tal como la herramienta de administración web o un usuario conectándose, el **HKD** es responsable de ejecutar los comandos apropiados para efectuar el cambio de estado.

Cuando **QVD** está configurado para la virtualización *KVM*, el **HKD** ejecuta una instancia *KVM* para cada máquina virtual que necesita ser iniciada, y ofrece opciones de inicio en base a la información obtenida de la base de datos.

Cuando **QVD** está configurado para *LXC*, el **HKD** revisará en primer lugar si el archivo de imagen se ha descomprimido en la carpeta basefs en el área de almacenamiento compartido, y descomprime el archivo de imagen, si es que aún no se ha hecho. El **HKD** a continuación, utiliza el módulo fuse-unionfs para llevar a cabo un montaje tipo `bind` de la imagen en la carpeta basefs con un sistema de archivos `overlay` que regenera de forma automática el sistema de archivos inicial. Este montaje se lleva a cabo dentro de la carpeta rootfs en el almacenamiento compartido. Por último, el **HKD** carga la imagen recién montada en una instancia *LXC*.

Cuando se inicia la instancia de máquina virtual, el **HKD** comprobará que arranque la imagen correctamente, que tiene conectividad de red y que el **QVD-VMA** se está ejecutando dentro de la máquina virtual. Si cualquiera de estas comprobaciones falla, el **HKD** bloqueará la máquina virtual dentro de la base de datos de QVD. Después de un corto período de tiempo, el **HKD** matará a la máquina virtual en ejecución.

Durante cada ejecución del bucle que el **HKD** realiza, va a comprobar el estado de todas las máquinas virtuales que se están ejecutando. Si en la base de datos hay cambios, los lleva a cabo de inmediato. Si no, actualiza la información de su estado en la base de datos.

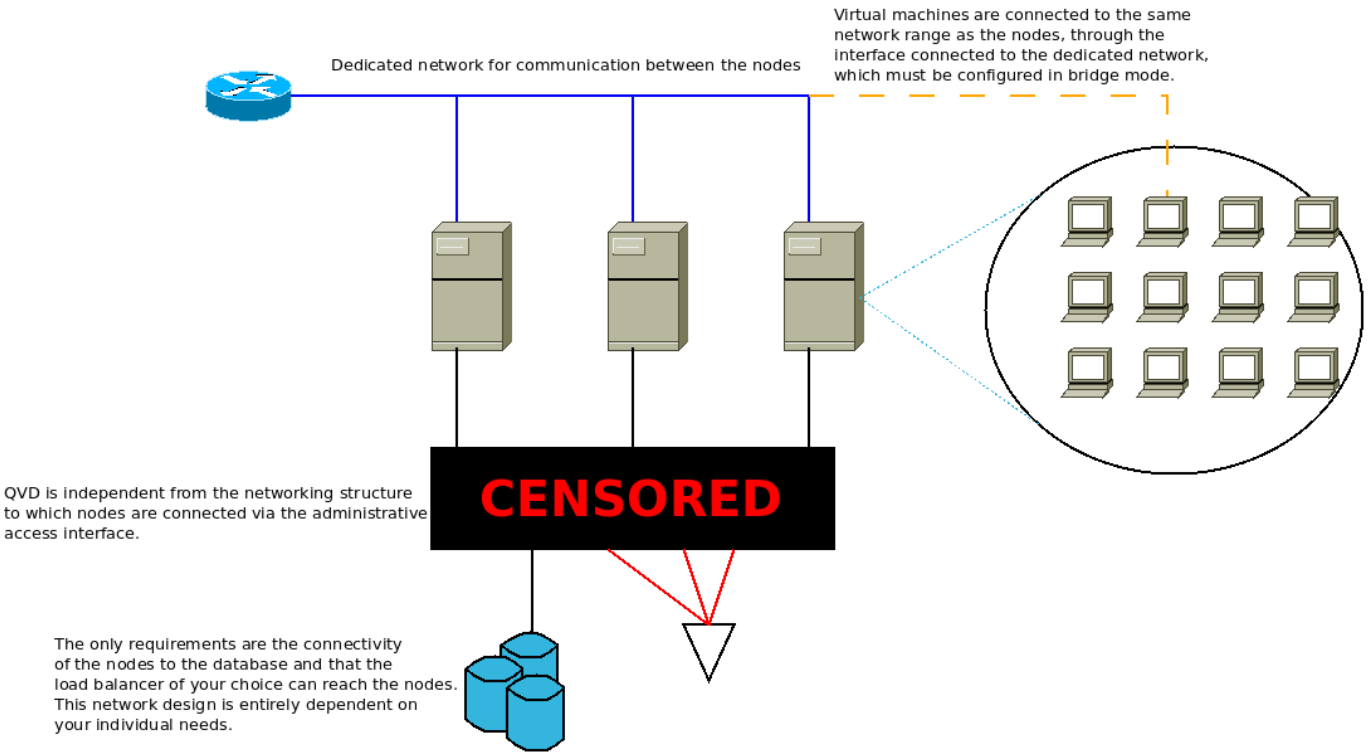


De acuerdo con el diagrama de arriba, estos son ejemplos típicos de los diferentes estados de la máquina que el **HKD** volverá para una máquina virtual puesta en marcha mediante *KVM*.

- Stopped: la máquina virtual no se ejecuta en ningún host.
- Starting 1: el **HKD** ha recibido la orden de marcha, pero está a la espera hasta que tenga los recursos disponibles para pasar al siguiente estado de la máquina.
- Starting 2: la máquina virtual ha iniciado el proceso de arranque, pero todavía no se ha completado.
- Running: la máquina virtual se está ejecutando en un nodo servidor.

- Stopping 1: el **HKD** recibió la orden de parada, pero está a la espera del **VMA** dentro de la máquina virtual para responder a la solicitud.
- Stopping 2: el **VMA** ha respondido a la solicitud de parada y la máquina virtual está en el proceso de apagado.
- Zombi 1: La máquina virtual se está ejecutando, pero no responde, una señal TERM ha sido enviada al proceso.
- Zombi 2: La máquina virtual se está ejecutando, pero no responde, una señal KILL ha sido enviado al proceso.

Arquitectura de red



En **QVD**, se reserva un rango de red (a elección del administrador) para las máquinas virtuales. Los nodos servidor deben estar conectados entre sí a través de este rango, es decir, deben tener una interfaz de red dedicada y conectada a él. Esta interfaz se utiliza para permitir al L7R encauzar conexiones de usuarios a las máquinas virtuales, y para permitir a las máquinas conectarse al mundo exterior (si se desea este comportamiento).

Note
Los nodos no realizan ninguna comunicación entre sí a través del rango de red reservado para las máquinas virtuales. Es más, los nodos nunca se comunican entre sí. Tan solo escuchan los cambios en la base de datos.

En general, es deseable que este rango se utilice en exclusiva para las máquinas virtuales. No obstante, el rango reservado para los nodos (que es desde el principio del rango hasta donde configure el administrador), puede dejar espacio para los servicios que el administrador considere necesarios en este rango.

Los nodos servidor, requieren de otra conexión a través de la cual se conectarán a la base de datos, y otros servicios, como el almacenamiento compartido o directorios ldap. **QVD** es totalmente independiente de la estructura de red en esta conexión, y queda supeditada a las necesidades del administrador.

Es posible también, utilizar técnicas más avanzadas como el vlan-tagging o virtual switching para utilizar un único puerto de red por nodo. Estas configuraciones son más complicadas, y quedan fuera de esta u otras guías que hemos publicado. Además, tampoco las recomendamos de cara a la estabilidad de la solución, ya que cruzar el tráfico de usuarios y base de datos podría dar lugar a problemas de conectividad y caídas de los nodos.

El cliente QVD y las relaciones entre L7R de distintos nodos

El **Cliente QVD** se conecta directamente al HKD. Este, hace un fork del proceso **L7R**, componente *broker* de QVD, que será el encargado de gestionar el resto de la comunicación con el cliente. El cliente inicia una conexión a través de HTTPS, en el que se solicita la presentación de credenciales HTTP de autenticación básica.

El **L7R** se conectará con la base de datos para determinar la forma de autenticación que debe tener lugar (es decir, a nivel local o usando un directorio LDAP externo) y tomar las medidas apropiadas para llevarlo a cabo. El **L7R** devolverá una respuesta HTTP OK si la autenticación se ha realizado correctamente, o devolverá un 401 no autorizado si falla la autenticación.

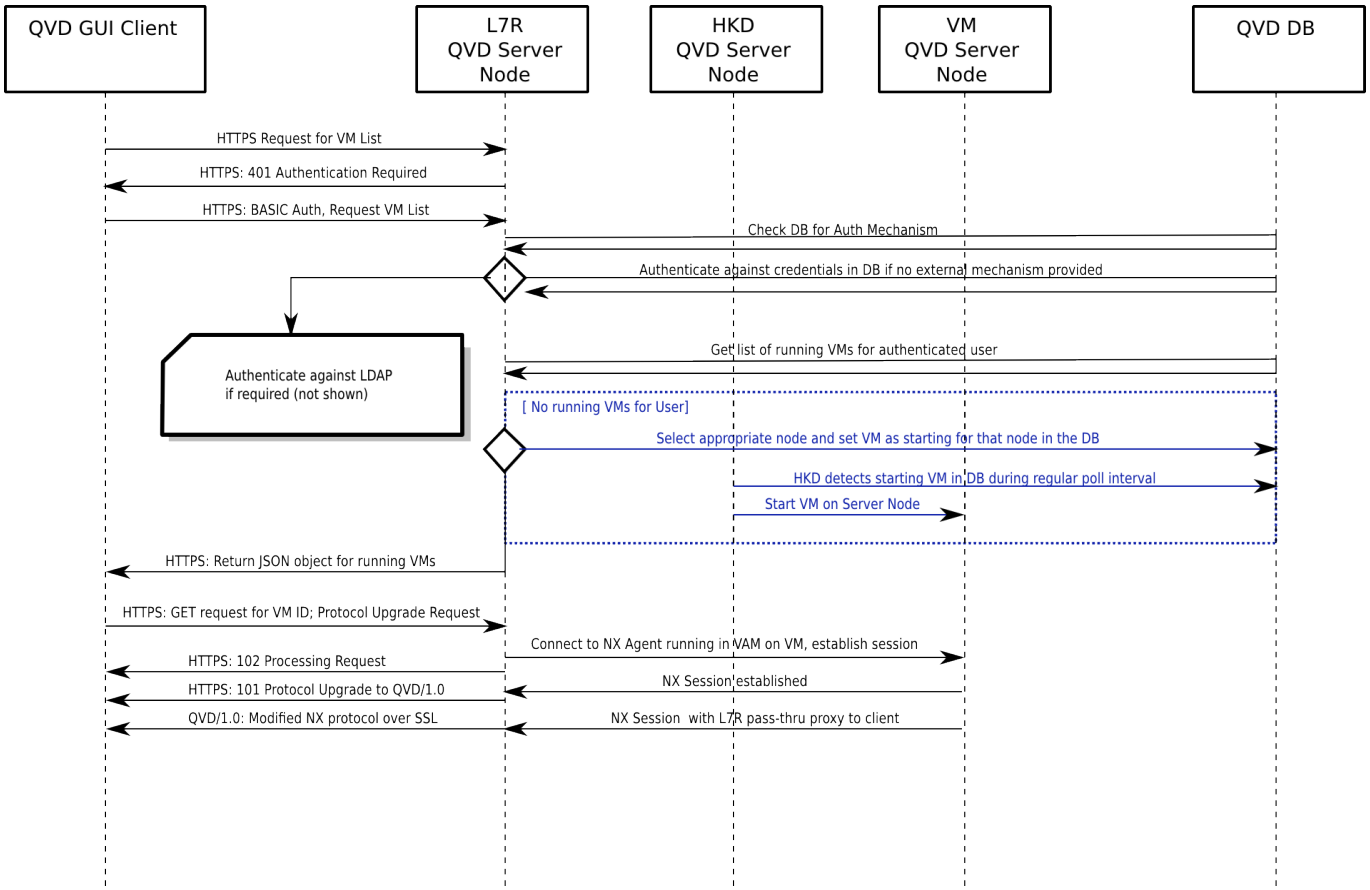
Una vez autenticado, el cliente solicita una lista de máquinas virtuales que están disponibles para el usuario. El servidor responde con una lista con formato JSON de identificadores de máquinas virtuales y su nombre correspondiente. El cliente selecciona una máquina virtual para conectarse y envía una solicitud GET con el identificador de la máquina virtual en una variable GET estándar. También pide una actualización del protocolo de QVD / 1.0 dentro de las cabeceras de petición HTTP.

Note
El L7R soporta una estructura de autenticación mediante plugins. Esto quiere decir que se soportan otros métodos de autenticación, como Single Sign On, autenticación por directorios LDAP o Active Directory, y también autenticaciones en varios pasos.

El **L7R** lleva a cabo las medidas necesarias para asegurarse de que la máquina virtual está en marcha y en espera de las conexiones que utilizan el protocolo *NX*. Si la máquina virtual no se está ejecutando en un nodo servidor, se determinará en qué nodo que debe iniciarse en forma automática y arrancará una máquina virtual para ese usuario. En cualquier otro caso, el **L7R** determinará en qué nodo se está ejecutando la máquina virtual y reenviará todas las solicitudes a esta máquina para toda comunicación posterior, incluyendo la comprobación para ver que una sesión de *NX* se puede configurar. Durante este proceso, el **L7R** devolverá una serie de respuestas HTTP 102 que van indicando el progreso del arranque y conexión con la máquina virtual. Si la máquina virtual está disponible, el **L7R** establece una conexión con el *nxagent* que se ejecuta en la máquina virtual y se convierte en un canal de comunicaciones tipo proxy transparente para la sesión de *NX*. Una vez que la sesión está configurada, el **L7R** emitirá una última HTTP 101 (Protocolos de conmutación) de respuesta para el cliente, y el protocolo para todas las futuras interacciones con el cliente se actualizarán a *NX*, asegurado mediante *SSL*. El **L7R** actualiza la base de datos de QVD para establecer el estado de la máquina virtual para indicar que un cliente está conectado.

Desde este punto en adelante, todas las comunicaciones entre el cliente y la máquina virtual se llevan a cabo a través del protocolo *NX* a través de la **L7R**. Cuando el cliente se desconecta, el **L7R** actualiza la base de datos de QVD para representar el cambio en el estado de desconexión del usuario respecto a la máquina virtual.

El flujo del proceso se indica en el siguiente diagrama:



L7R con balanceo de carga

Como ya se ha mencionado, los nodos servidor están diseñados para un entorno con balanceo de carga. Con el fin de atender a esto, el elemento **L7R** de cada **HKD** es capaz de redirigir el tráfico de una determinada máquina virtual a cualquier otro nodo servidor en el entorno.

La configuración habitual es tal que una máquina virtual puede iniciar en cualquiera de los nodos servidor. Cuando un usuario se autentica mediante cualquier **L7R** dentro de la granja, el **L7R** determina en qué nodo servidor se está ejecutando actualmente la máquina virtual a la que el usuario desea conectarse. Esto se logra mediante una consulta a la base de datos. Si se detecta que dicha máquina virtual se está ejecutando en el entorno, la **L7R** redireccionará todo el tráfico hacia el nodo servidor adecuado.

Si ninguna máquina virtual se está ejecutando actualmente para el usuario, el **L7R** hace uso de un algoritmo interno para determinar el nodo más adecuado para iniciar una nueva máquina virtual para el usuario. Este algoritmo se basa en la evaluación de qué nodo tiene la mayor cantidad de recursos libres, calculado como la suma ponderada de memoria RAM libre, CPU sin uso, y un número aleatorio para conseguir un poco de entropía en el resultado.

Cuando un nodo apropiado ha sido seleccionado, la base de datos se actualiza de modo que una máquina virtual será iniciada por el **HKD** en el host correcto. El **L7R** entonces redirigirá todo el tráfico para esa conexión al nodo servidor que se ha seleccionado para ejecutar la nueva máquina virtual.

El nodo de Administración QVD

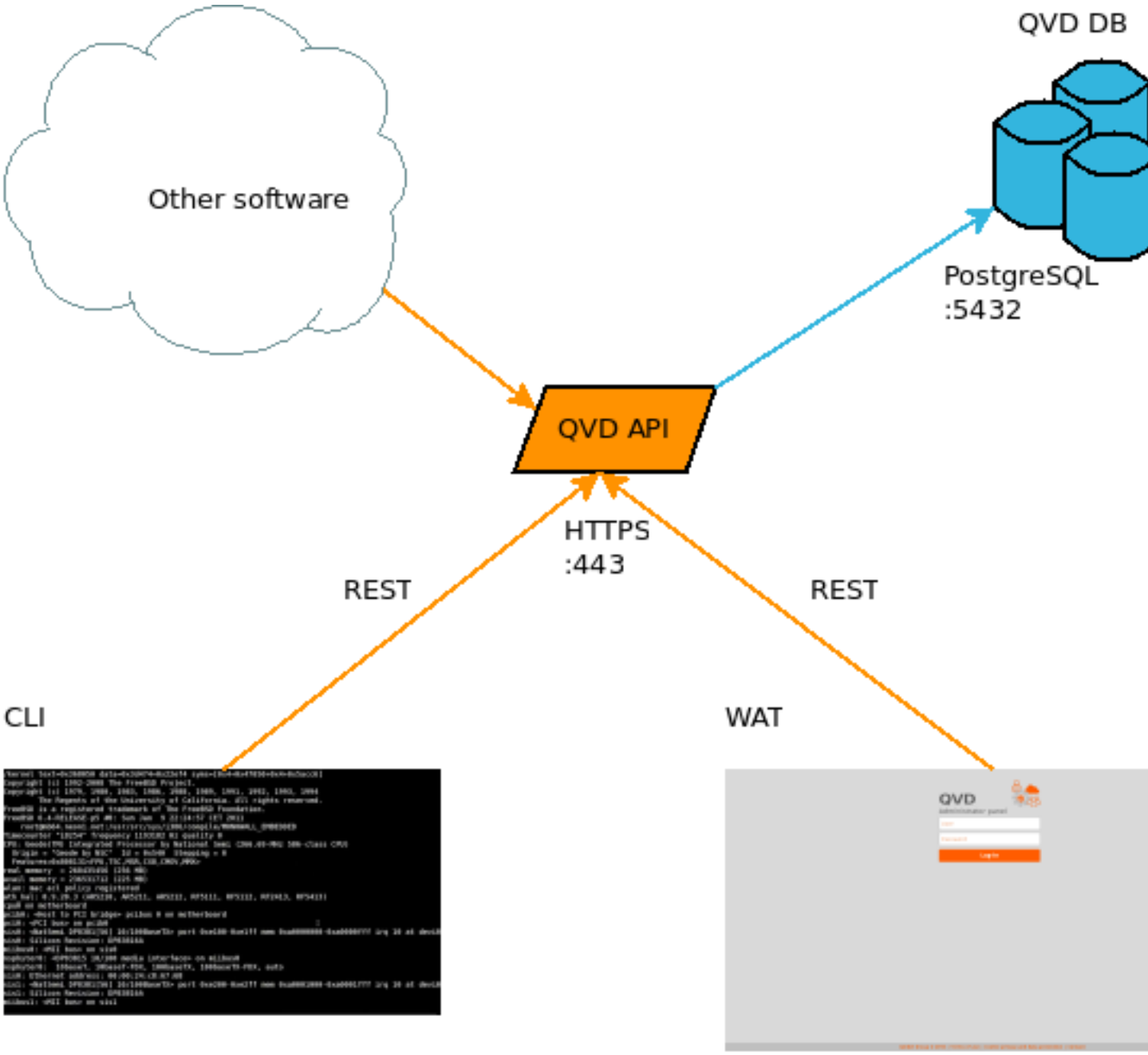


Figure 2: Componentes de un nodo de Administración

Un nodo de administración típico consta de los siguientes componentes:

- **QVD-API**
- **Herramienta de administración en línea de comandos**
- **WAT (Web Admin Tool)**

Cada uno de estos componentes es independiente de los demás, en el sentido de que pueden estar instalados en máquinas distintas. Sin embargo, tanto el **WAT** como la herramienta en línea de comandos son dependientes de la API. Esta debe estar instalada en al menos un nodo de la instalación para permitir el funcionamiento de las otras dos herramientas. Por ejemplo, es muy típico instalar la herramienta en línea de comandos en todos los nodos QVD, para facilitar al administrador del sistema la gestión de la solución.

API

La API es un componente estructural de **QVD**. Consiste únicamente de una interfaz *REST* conectada a la base de datos. Es necesaria para poder gestionar la solución ya que es la única vía de comunicación de las herramientas de administración. Se puede utilizar también para integrar la solución con otros sistemas que sean capaces de utilizar la interfaz *REST*.

CLI

La herramienta en línea de comandos permite acceder a las mismas funciones que el **WAT**, pero en línea de comandos linux. Al igual que el WAT, es un cliente de la API, y no puede funcionar sin ella. Su uso es, por supuesto, más complejo que del **WAT**, y por ello queda probablemente limitada a su uso por parte de un administrador de sistemas. Por otra parte, al igual que la **API**, permite la integración con otros sistemas, ya que se puede utilizar para ejecutar tareas de forma programática. Su uso en crontab está especialmente indicado para programar distintos comportamientos que pueden ser deseables, como por ejemplo apagar automáticamente máquinas virtuales que lleven si uso cierto tiempo, borrado de información obsoleta, chequeos de configuración u obtención de informes.

WAT

El **WAT** es el **panel de administración Web de QVD**. Una herramienta web con la que se pueden gestionar usuarios, máquinas virtuales, nodos, imágenes y parámetros de configuración de **QVD**.

Para ello, mostrará en pantalla listados con los elementos del sistema con información suficiente para poder configurarlos así como detectar problemas. Dispone de controles de filtrado y multitud de acciones posibles sobre los elementos de **QVD** así como crear, actualizar o eliminarlas; y otras más específicas como arrancar o parar una máquina virtual, bloquear un usuario por tareas de mantenimiento, etc.

Cliente-Servidor En la administración de **QVD**, el WAT corresponde a la parte de **cliente**, nutriéndose del servidor vía HTTP. De este modo extrae y gestiona la información de **QVD** a través de **llamadas autenticadas a la API** del servidor. Esta API también sirve a la aplicación de administración en línea de comandos (**QVD CLI**).

Tenants

También es nuevo en esta versión el sistema **multitenant**. QVD tiene ahora dos modos de funcionamiento: **monotenant** y **multitenant**.

- **Monotenant**: Todos los administradores del sistema conviven en un mismo ámbito o *tenant*. Este modo de funcionamiento sería el equivalente a cómo funcionaba el **WAT** en versiones anteriores a **QVD 4**.
- **Multitenant**: Podrán existir diferentes ámbitos o *tenants*. En ellos, se podrán crear elementos de **QVD** independientes entre sí y administradores que los gestionen. En este caso cada *tenant* se comportará como una instalación de **WAT monotenant**, pudiendo otorgar a los administradores permisos para poder gestionar más o menos elementos con mayor o menor control.

Un sistema por defecto es **monotenant**. Viene creado un usuario administrador con el que tenemos acceso total y con él podremos crear elementos de **QVD** y a otros administradores con los permisos más o menos limitados para gestionar diferentes partes del **WAT**.

Estos permisos harán referencia a qué elementos ver o gestionar (Usuarios, Máquinas virtuales, etc.) pero no se podrá dar acceso sobre un subconjunto de los mismos.

Por ejemplo, si a un administrador le damos permisos de lectura sobre las imágenes de disco, podrá ver todas las imágenes del sistema, no podremos limitarlo a un subconjunto de ellas.

Este tipo de disgregación se realizará en el modo **multitenant**.

Por ejemplo, a un administrador se le podrán asignar permisos de lectura sobre imágenes de disco, con el que solo podrá ver las que haya en su *tenant*, y un nivel más avanzado de gestión en máquinas virtuales, con el que podrá, además de visualizar, crear y actualizar las máquinas virtuales a las que tenga acceso (las de su *tenant*).

Los administradores de un *tenant* estarán aislados en su *tenant*, sin que sepan que existen otros ámbitos. Solo verán los elementos de QVD que hay en ese *tenant*. El administrador ni siquiera será consciente de si está trabajando en un **WAT monotenant** o en un *tenant* dentro de un **WAT multitenant**.

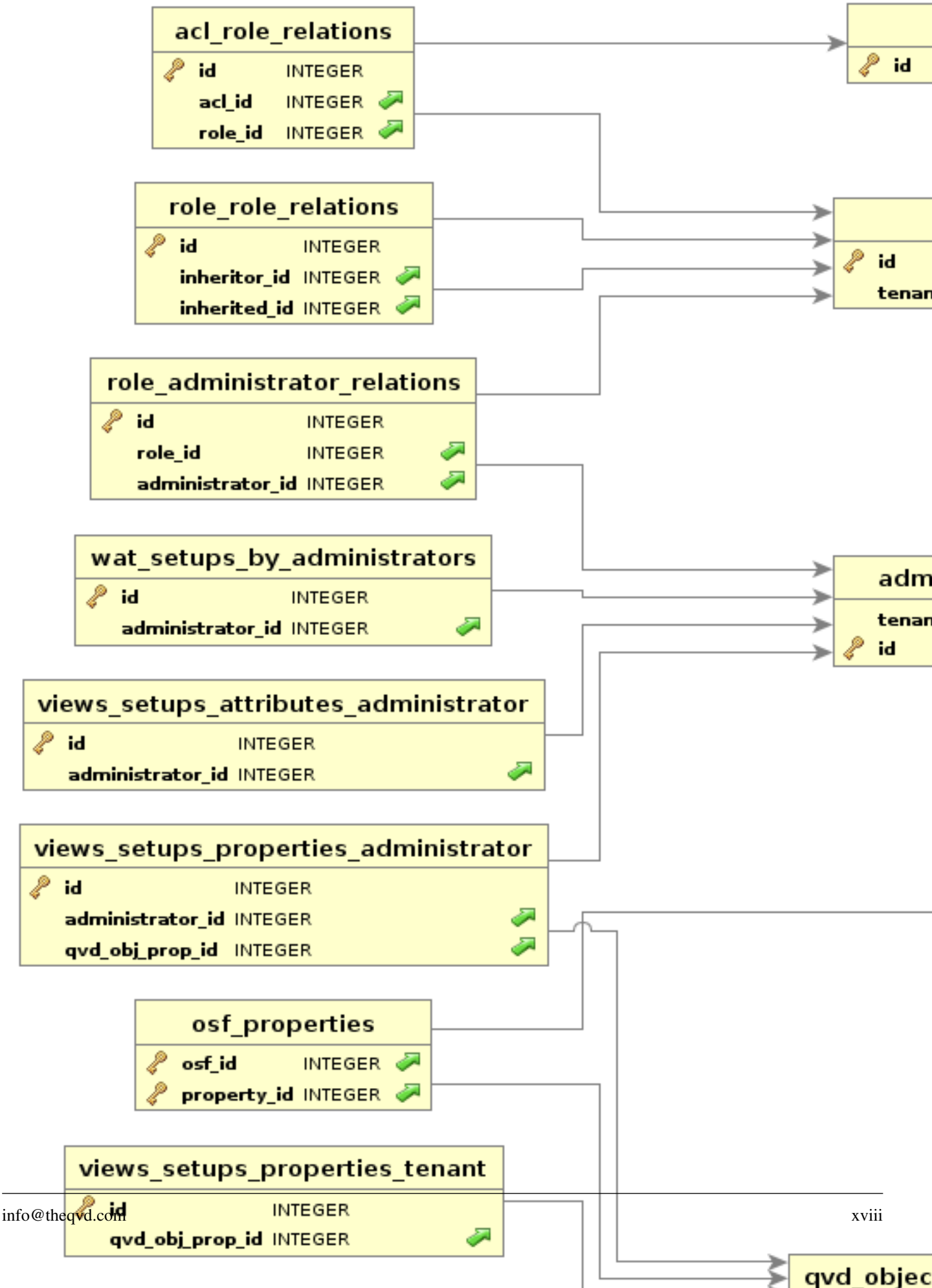
En un **WAT multitenant**, existirá un ámbito superior al que denominaremos **Supertenant** ó *Tenant* que englobará a todos los demás. Los administradores de este **Supertenant** están pensados para tareas de configuración y supervisión ya que podrán gestionar elementos de **QVD** de cualquier tenant siendo conscientes de la distribución, pudiendo filtrar elementos por *tenant*, o elegir en qué *tenant* crear un determinado elemento.

Para más detalles, refiérase al manual de administración de QVD.

Base de datos QVD

La base de datos es una instalación estándar de **PostgreSQL**. Comentar únicamente que se requiere la versión 9.3.

Adjuntamos este esquema simplificado de la base de datos como referencia, pero no se va a describir en este documento.


acl_role_relations
id INTEGER
acl_id INTEGER
role_id INTEGER
role_role_relations
id INTEGER
inheritor_id INTEGER
inherited_id INTEGER
role_administrator_relations
id INTEGER
role_id INTEGER
administrator_id INTEGER
wat_setups_by_administrators
id INTEGER
administrator_id INTEGER
views_setups_attributes_administrator
id INTEGER
administrator_id INTEGER
views_setups_properties_administrator
id INTEGER
administrator_id INTEGER
qvd_obj_prop_id INTEGER
osf_properties
osf_id INTEGER
property_id INTEGER
views_setups_properties_tenant
id INTEGER
qvd_obj_prop_id INTEGER

No obstante, sí se van a describir algunos elementos necesarios para engranar la arquitectura física con la funcional.

Objetos de la base de datos

Para poder administrar correctamente una solución **QVD**, es importante conocer estos objetos de la base de datos y la relación entre ellos:

DI (Disk Image) La **DI**, como su nombre indica, es una imagen de disco de un sistema operativo. El único requisito que tienen, es tener instalado y configurado el **VMA** y un escritorio linux estándar y compatible con la solución. Aparte de esto, los requisitos propios del tipo de imagen que sea (*KVM* o *LXC*).

OSF (Operating System Flavour) El **OSF** es un conjunto de recursos que el administrador define para utilizar con un conjunto de imágenes (límites de memoria, disco y otras configuraciones). Cuando se da de alta una nueva **DI**, se le debe asignar un **OSF** al que pertenece. También se le pueden añadir tags personalizados, y si no la solución la etiquetará automáticamente. En breve veremos para qué sirve esto.

Al crear una máquina virtual para un usuario, también se le asigna un **OSF**. De esta manera, cuando se pone en marcha la máquina virtual, se carga dicho **OSF** y todos los parámetros de configuración asociados. Además del **OSF**, las máquinas virtuales también tienen tags asociados. Es mediante estos tags, que se decide cual de las **DI** asociadas al **OSF** se va a utilizar como imagen base de la máquina virtual.

Tags (Etiquetas) La capacidad de etiquetar es importante, ya que permite cambiar fácilmente las versiones de imagen de disco para múltiples máquinas virtuales. Si se va a realizar un cambio en una imagen de disco que utiliza un gran número de usuarios, se puede gestionar el problema mediante etiquetas. El caso clásico es asignar una etiqueta temporal a la nueva imagen, que también tiene asignada en exclusiva una máquina virtual del administrador. Dicho administrador, puede de esta manera arrancar su máquina y comprobar que todo va en la nueva imagen como él espera. Si es así, el paso a producción es tan simple como cambiar la etiqueta de dicha **DI** a la que están utilizando las máquinas de los usuarios en producción. Si finalmente algo va mal, la reversión es tan simple como volver a cambiar la etiqueta a la anterior **DI**.

La etiqueta de la **DI** solo se comprueba en el momento del arranque de la máquina virtual. Esto quiere decir que las máquinas virtuales en ejecución no se van a ver afectadas por los cambios hasta que el usuario se desconecte. Para estos asuntos, las imágenes cuentan con tiempos de expiración soft y hard. La combinación de ambos se puede utilizar para notificar al usuario que su imagen ha sido actualizada y que debe desconectarse cuando pueda, y/o directamente obligarle a desconectarse apagando su máquina virtual.

ACLs (Access Control List) Las **ACLs** son una nueva característica de **QVD**. En esta versión se ha programado un sistema de estricto control de acceso a los recursos de la solución. Para cada instalación se puede definir un listado de roles con una serie de permisos asignados, y relacionar los administradores con estos roles, de manera que puedan o no ejecutar cierto tipo de acciones. Este sistema de permisos tiene una gran granularidad y permite escalar la solución a un gran número de usuarios con permisos de administración.

Roles Los roles son también una nueva característica de **QVD**. Permiten definir conjuntos de ACLs predefinidos, que se pueden asignar a los usuarios para facilitar la gestión de los permisos descritos en el apartado anterior.

Usuarios Cada persona que vaya a acceder a la solución, necesita una cuenta de usuario. Estas cuentas se registran en la base de datos, con id, nombre de usuario y contraseña. Como el resto de objetos de la base de datos, se pueden etiquetar para tareas administrativas y scripts que automaticen tareas. Como nota, destacar que la contraseña guardada en la base de datos no tienen por qué estar en uso, si se está utilizando un plugin de autenticación ajeno a la solución (como sería la autenticación LDAP, por ejemplo).

VM A cada usuario, le corresponde una colección propia de máquinas virtuales. Para conectar, se le debe definir al menos una. Las máquinas virtuales, pertenecen a su vez a un único tipo de OSF, y también tienen como comentábamos antes una etiqueta. Cuando se pide a la solución que encienda la máquina virtual, se busca la imagen que corresponde a la etiqueta y OSF que están registradas para esa máquina virtual en la base de datos, y se utiliza para arrancarla.

Hosts Los nodos servidor QVD son a su vez un objeto en la base de datos. Se guarda de ellos su dirección ip y nombre (que debe corresponder al hostname definido en el fichero de configuración de cada uno de ellos). Al estar disponibles como objeto de configuración permiten gestionar su disponibilidad.

Tecnologías de virtualización

QVD soporta dos diferentes tecnologías de virtualización: *KVM (Kernel Virtual Machine)* y *LXC (Linux Containers)*. Cada tecnología de virtualización tiene su propio conjunto de ventajas y será más útil para casos de uso específicos. Por lo tanto, una buena comprensión de sus propias necesidades, y una comprensión de estas dos tecnologías ayudan a determinar cómo configurar la implementación de **QVD**.

La virtualización KVM *KVM* es un hipervisor de tipo 1 que se ejecuta dentro del núcleo del sistema operativo Linux. El hipervisor garantiza la separación absoluta del sistema operativo subyacente, lo cual le permite cargar sistemas operativos completamente distintos en cada máquina virtual y que funcionen como si estuvieran en ejecución en equipos completamente separados.

Si bien existe cierto debate sobre si *KVM* es en realidad un hipervisor tipo 1, ya que requiere el núcleo de Linux con el fin de funcionar, la mayoría de los expertos de virtualización están de acuerdo en que combinado con el núcleo de Linux, las funciones de *KVM* son exactamente las mismas que las de cualquier otro hipervisor tipo 1, como Xen o ESXi de VMware. De hecho, en los informes de referencia SPECvirt 2011, *KVM* ocupó el segundo lugar en rendimiento por detrás de VMware ESX, lo que indica un alto grado de viabilidad como una plataforma de virtualización de calidad comercial. Ya que *KVM* utiliza la separación absoluta, es mucho más fácil de configurar y administrar que *LXC*. Sin embargo, a pesar de que ofrece un rendimiento competitivo con otros hipervisores de hardware, cada máquina virtual ejecuta necesariamente su propio núcleo. Los recursos deben ser dedicados a cada máquina virtual, se esté utilizando o no. De esta manera, *KVM* no es tan eficiente como *LXC*, pero ofrece una flexibilidad mucho mayor y facilidad de gestión.

La virtualización LXC *LXC* proporciona virtualización a nivel de sistema. De este modo, actúa como una alternativa a la virtualización completa a nivel de hardware proporcionada por el hipervisor *KVM*. *LXC* se comporta de una manera similar a un entorno enjaulado dentro de Linux, pero ofrece un mayor nivel de aislamiento y la gestión de los recursos entre los contenedores a través del uso de espacios de nombres y cgroups. Por ejemplo, los identificadores de proceso (PID), recursos de red y soportes para cada contenedor se pueden aislar de otros contenedores y se pueden agrupar de forma lógica para aplicar normas de gestión de recursos y otras políticas específicas para cada contenedor. Esto le permite obtener muchos beneficios de la virtualización manteniendo a la vez bajos los requisitos generales de recursos, así como volver a usar el mismo núcleo en todas las máquinas virtuales. *LXC* está totalmente soportado por el núcleo Linux, y está incluido en **QVD** desde la versión 3.1.

Máquinas virtuales y VMA

Como comentábamos anteriormente, el **HKD** es el encargado de arrancar las máquinas virtuales de los usuarios. En un entorno de producción, es habitual que exista un número de diferentes servidores QVD ejecutándose en paralelo. Cada máquina virtual, es una instancia separada que se ejecuta en alguno de los nodos servidor de QVD. Si se conecta un usuario, autentica contra un **L7R**, y de entre sus máquinas virtuales, solicita conectarse a una que está parada, el **L7R** usará su algoritmo de balanceo de carga para determinar qué nodo debe ejecutar la máquina virtual del usuario y la base de datos será actualizada de forma que la máquina virtual se inicie en el nodo apropiado.

Las máquinas virtuales hacen uso de puntos de montaje superpuestos con el fin de utilizar mejor los diferentes elementos del sistema operativo invitado, y de hacer que los datos de usuario sean persistentes. Por ejemplo, aunque la actividad de escritura no es persistente en la imagen real que se carga, es importante que los datos escritos en la carpeta de inicio del usuario o de escritorio se almacenen para futuras conexiones con el escritorio virtual.

Dentro de las instancias de QVD que hacen uso de la virtualización *KVM*, esto se logra mediante el almacenamiento de directorio personal del usuario dentro de una imagen tipo *qcow2*. Esto se monta sobre el directorio personal del usuario en la imagen de sistema operativo que se ha cargado en la máquina virtual.

En los casos que hacen uso de *LXC*, esto se logra mediante el aprovechamiento de los puntos de montaje tipo *union_mount* (más información en Wikipedia). Los datos del directorio personal del usuario y los datos `overlay` se almacenan en un directorio independiente fuera de la imagen base que se utiliza para la máquina virtual. Estas carpetas pueden ser montadas sobre la imagen base en tiempo de ejecución, con el fin de crear un contenedor específico para cada usuario y máquina virtual.

Note
También se puede utilizar btrfs (Wikipedia) para gestionar las imágenes y puntos de montaje, pero no se va a discutir en este documento. Vea la **Guía de Administración**.

La imagen *qcow2* con el directorio personal del usuario normalmente se almacena en un recurso compartido de red, de modo que sea accesible a cualquier nodo servidor en la granja. Si la máquina virtual del usuario se arranca más adelante en un nodo servidor diferente, el directorio personal del usuario se puede cargar en tiempo de ejecución y los datos siempre estarán disponibles para el usuario. Los puntos de montaje superpuestos también se pueden utilizar para hacer que otros datos, tales como logs y archivos temporales sean persistentes desde la perspectiva del usuario. Este comportamiento es configurable, y por tanto se puede elegir si las máquinas son totalmente efímeras y ni siquiera guardamos los datos del usuario, si guardamos al menos los datos del usuario, o si guardamos la máquina al completo con todos los cambios que sufra estando en ejecución.

Dependiendo de la tecnología de virtualización configurada en **QVD**, las máquinas virtuales se pondrán en marcha, ya sea usando *KVM* o *LXC*. Sin embargo, es importante entender que las imágenes de estas dos tecnologías son muy diferentes y no es viable intercambiarlas.

Una vez en funcionamiento, cada máquina virtual debe cargar el **QVD-VMA (Virtual Machine Agent)** con el fin de funcionar correctamente. El **VMA** se asegurará de que el componente *nxagent* está disponible para que el cliente sea capaz de conectarse al escritorio virtual. También responde a las consultas del **L7R**, con lo que éste puede determinar el estado del usuario y la vm, realimentando de esta manera la base de datos. Cuando se crea una imagen, es de fundamental importancia que el **VMA** esté instalado y configurado o la imagen no se podrá utilizar en **QVD** en absoluto.

Diagramas de arquitectura de alto nivel

En esta sección se muestran algunos esquemas simples que explican la arquitectura de una implementación típica **QVD** para mostrar cómo interactúan los diferentes componentes.

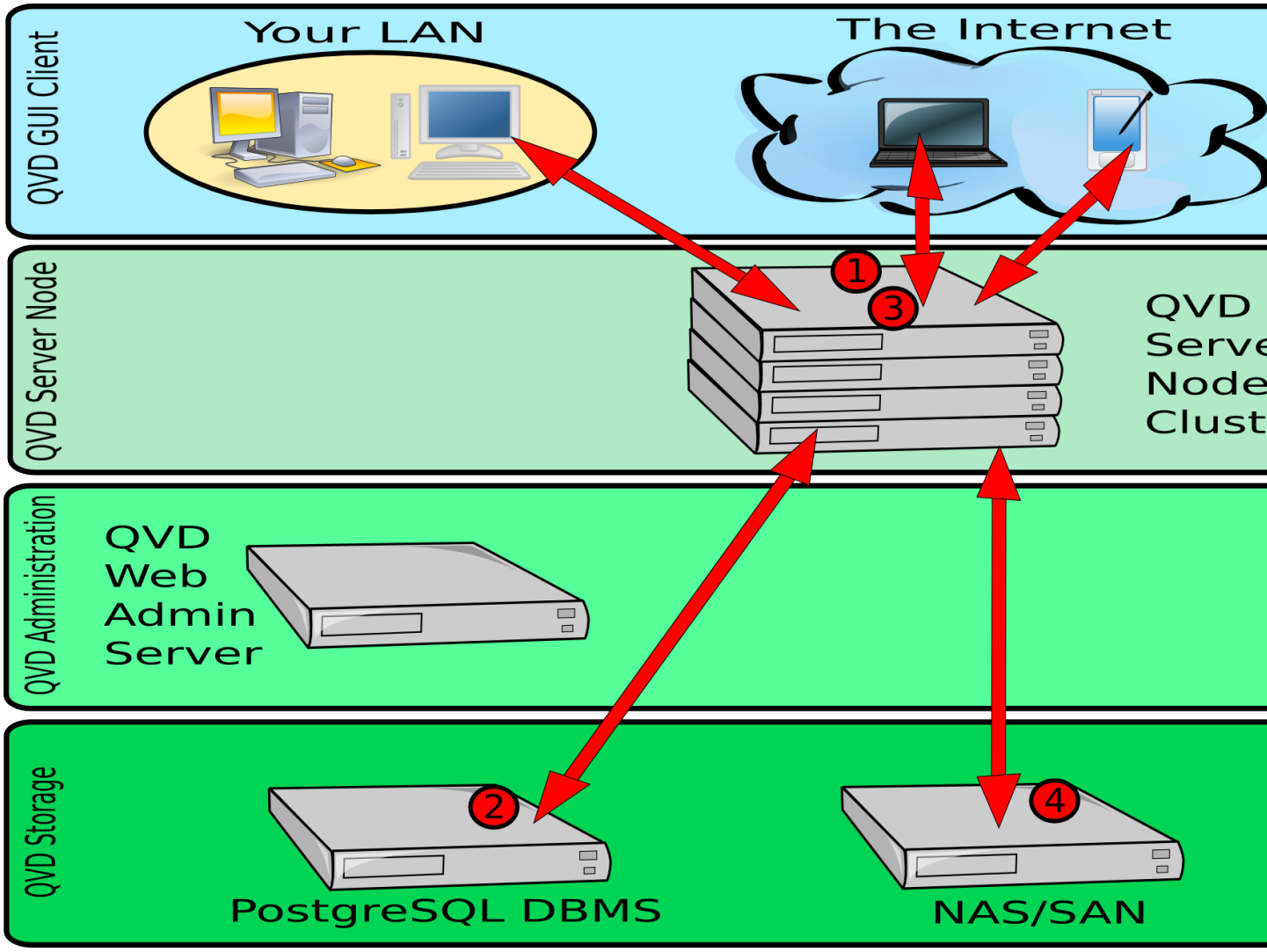


Figure 4: Interacción Cliente y servidor en la arquitectura QVD

En el diagrama anterior, se muestran las interacciones entre la aplicación cliente, los nodos servidor, el almacenamiento compartido y la base de datos `PostgreSQL`.

1. La aplicación cliente puede conectarse a través de una LAN o a través de Internet. La conexión inicialmente hace uso del

protocolo HTTPS para manejar la autenticación inicial y para establecer una sesión.

2. El componente **L7R** dentro del **HKD** se conecta a la base de datos `PostgreSQL` para comprobar los ajustes de configuración y para autenticar al usuario. Si la autenticación se ha delegado en algún otro servicio integrado como LDAP, el **L7R** obtendrá la información adecuada de la base de datos y realizará los pasos necesarios para autenticar al usuario. El **L7R** también utiliza la base de datos para obtener información acerca de qué máquina virtual (o máquinas) debe servir al usuario junto con otra información relacionada. Por último, el nodo servidor actualizará periódicamente la información de estado acerca de las sesiones, las máquinas virtuales y los usuarios dentro de la base de datos con fines de gestión.

3. Una vez autenticado, el servidor y el cliente renegocian una conexión de protocolo *NX* asegurada mediante *SSL*. El cliente es capaz de conectarse a un escritorio cargado dentro de la máquina virtual asignada que se ejecuta en el nodo servidor.

4. Antes de cualquier conexión desde el cliente, el nodo servidor carga una imagen desde el almacenamiento compartido en una máquina virtual. El almacenamiento compartido se suele acceder desde un sistema de archivos de red montado por *NFS*. Cuando se inicia la máquina virtual para un usuario concreto, se crea el directorio personal del usuario (más adelante se detalla qué significa esto, según usemos *KVM* o *LXC*). Esto también se almacena en un recurso compartido de red. Al mantener la imagen de inicio del usuario, el **OSF** y overlays dentro del almacenamiento compartido, **QVD** es capaz de garantizar automáticamente que el usuario todavía es capaz de acceder al mismo escritorio sea cual sea el nodo servidor al que se conecte. Esto proporciona tolerancia a fallos y redundancia.

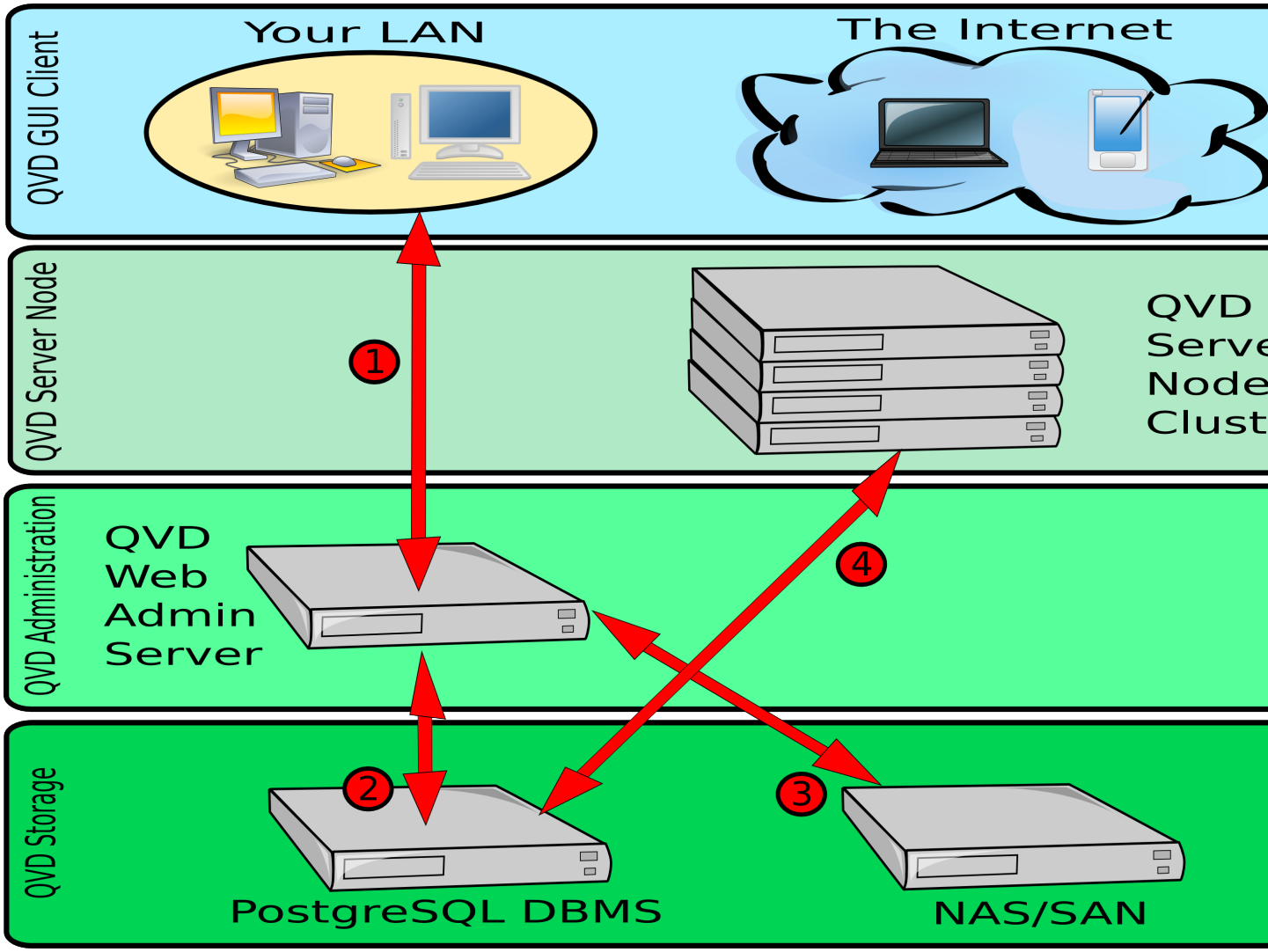


Figure 5: QVD-WAT e Interacciones en el nodo de servidor Arquitectura QVD

En el diagrama anterior, podemos ver las diferentes interacciones que están involucradas en el funcionamiento de la herramienta de administración Web de QVD, el **WAT**. El **WAT** interactúa exclusivamente con la base de datos QVD-DB, que es quien interactúa con el resto de componentes de la solución. Las interacciones mencionadas aquí están simplificadas, ya que hay un gran número de operaciones que se pueden realizar utilizando **QVD-WAT**.

1. Un administrador puede conectarse al **WAT** desde un navegador web común. La conexión tiene lugar a través de HTTPS. Estas credenciales se almacenan en la base de datos `PostgreSQL`.

2. El **WAT** hace uso de la base de datos `PostgreSQL` para almacenar información de configuración introducida por el administrador a través de la interfaz web. El **WAT** también extrae información como el estado de las máquinas virtuales, usuarios y sesiones de la base de datos, para presentar al administrador a través de la interfaz web.

3. Las **DI** disponibles en el almacenamiento compartido, pueden ser administrados desde el **WAT**. Se pueden habilitar y deshabilitar, así como cambiar sus etiquetas, haciéndolas de esta manera disponibles para los usuarios. También es posible cargar nuevas **DI** desde el **WAT**, siempre que este tenga acceso al almacenamiento compartido (No se muestra en la imagen).

4. El **HKD** en cada nodo servidor, realiza peticiones periódicamente a la base de datos `PostgreSQL` para recoger los cambios de configuración y estado realizadas por el **WAT**. Por ejemplo, cuando desde el **WAT** se inicia o se detiene una máquina virtual, este cambio se lleva a cabo dentro de la base de datos, y cuando en las próximas peticiones de los **HKD** a la base de datos se determina que el estado de una máquina virtual se ha cambiado, el cambio se ejecutará en uno de los nodos. Como podemos ver, el **WAT** no interactúa directamente con cualquier nodo de servidor en particular, sino que utiliza la base de datos `PostgreSQL` como intermediaria.