Infraestructura como servicio –IaaS–

(*infraestructure as a service*)

PID_00241981

Remo Suppi Boldrito

Tiempo mínimo de dedicación recomendado: 10 horas

Universitat Oberta de Catalunya

Remo Suppi Boldrito

Ingeniero de Telecomunicaciones. Doctor en Informática por la UAB. Profesor del Departamento de Arquitectura de Computadores y Sistemas Operativos en la Universidad Autónoma de Barcelona.

Índice

Introducción

El término de IaaS –Infraestructura como servicio– es un concepto que se aplica también a la renovación tecnológica de un centro de IT, por lo cual en este módulo se denominará IaaS. Es una fase del *cloud* que comienza a desarrollarse a mediados de la anterior década como resultado de la evolución de las técnicas de virtualización y la reducción significativa de los costos del hardware de servidores de altas prestaciones [Cpt].

Infraestructura como servicio

Algunos autores lo denominan también *hardware as a service*.

Esta evolución de las técnicas de virtualización, sobre las cuales se habían desarrollado experiencias desde los años noventa mediante técnicas software, tienen un punto de inflexión cuando en 2006 Intel y AMD introducen (trabajando de forma independiente) la virtualización soportada por hardware, la cual permitía simplificar el software de virtualización, pero se obtenía muy pocas mejoras en prestaciones. En la siguiente generación de procesadores, se toman como punto de partida las experiencias anteriores, y se obtienen prestaciones significativas sobre las arquitecturas anteriores y las técnicas por software, actuando a nivel de la CPU y de las unidades de gestión de la memoria (*memory management unit*, MMU).

El VT-x (Intel Virtualization Technology), inicialmente llamada Vanderpool, es la tecnología de Intel que permite la virtualización en una plataforma x86 que a través de las extensiones de máquina virtual (se verá en el procesador como **vmx**) compuestas por 10 instrucciones permiten entrar y salir del modo de ejecución virtual y donde el SO *guest* percibe que se está ejecutado con todos los privilegios (*ring 0*) mientras que el SO *host* permanece protegido.

Hoy en día casi todas las versiones de los procesadores Intel soportan estas extensiones (en algunos casos está desactivada, mientras que en algunas placas base está desactivada por BIOS, pero –en una gran mayoría– se puede volver a activar). Intel ha evolucionado su arquitectura para mejorar las prestaciones a la virtualización incluyendo *extended page tables* (EPT) en Nehalem (2008) para la virtualización de las tablas de páginas de la MMU y en 2010, con la arquitectura Westmere, que permite que el procesador lógico se ejecute en modo real (denominado *unrestricted guest*) que utiliza el soporte de EPT.

La arquitectura Haswell (2013; forma parte de las series comerciales i3, i5, i7) incluye soporte para acelerar las virtualizaciones anidadas a través de una estructura de datos (*virtual machine control structure*, VMCS) que es manejada por cada MV y que se mantiene sobre la nueva arquitectura SkyLake de 4 y 8 procesos concurrentes (2015) [Iar][Ivt].

La extensión de virtualización AMD para plataformas x86 (se visualiza en procesador como **svm**) se llamó inicialmente AMD Secure Virtual Machine (SVM), pero posteriormente se cambió a AMD Virtualization (AMD-V) como se la conoce actualmente y referenciada como Pacifica.

El primer procesador fue Athlon 64 (Orleans, Windsor) en 2006 y posteriormente Turion 64, Opteron y Phenom (I & II), así como en la integración de CPU-GPU de AMD llamada Fusion (de la serie Sempron hay algunos que la incluyen y otros no). Las dos últimas microarquitecturas de AMD (Jaguar y Bulldozer) incluyen AMD-V, donde la primera es la utilizada tanto por PlayStation 4 como por Xbox One, líderes del segmento de mercado de consolas, mientras que la segunda, incluida en procesadores de la línea Opteron, para servidores y máquinas de escritorio.

Sobre la siguiente microarquitectura (Zen, 2017) todavía no se tienen noticias, pero los expertos aseguran que la línea orientada a servidores (se esperan 32 *cores* y 64 *threads* en la segunda mitad del 2017) no podrá faltar, aunque se desconocen las mejoras que aportará a la virtualización.

Es importante indicar que los AMD Opteron son los que incluyen la segunda generación de tecnología de virtualización llamada *rapid virtualization indexing* (o conocida también como *nested page tables*) y que posteriormente fue adoptada como *extended page tables* (EPT) por Intel [Aar].

Además de los avances en virtualización hay tecnologías y acontecimientos que se han favorecido del despegue del *cloud computing* y por consiguiente del IaaS como la primera capa de este paradigma.

Por un lado, la evolución del desarrollo de software y las operaciones IT, uniéndose como DevOps, para aunar las sinergias entre los desarrolladores y los profesionales en las tecnologías de la información (IT), con el objetivo de acelerar los procesos de creación de software permitiendo que el flujo de trabajo pueda ser más fluido y los pasos desde el desarrollo, prueba y despliegue de una aplicación pueda ser realizado en un tiempo óptimo y con una mayor fiabilidad, llegando a situaciones de despliegue continuo o entrega continua (*continuous deployment/delivery*). Como es previsible, esto requiere agilidad en la provisión y variedad de entornos, lo cual es muy complejo de gestionar sobre arquitecturas virtualizadas únicamente. Por otro lado, una reducción en la frecuencia de compra/instalación/puesta en marcha de nuevas infraestructuras para satisfa-

cer las necesidades de los programadores/negocio y también una evolución de las herramientas de medición y cálculo de uso que permiten favorecer formas de pago-por-uso o pago-por-recursos como modelo de negocio rentable tanto para los desarrolladores como para los responsables de IT (locales o externos). Es decir, todo un conjunto de circunstancias que han permitido una **evolución** como la que se conoce hoy en día, donde un desarrollador se conecta a una página web de un servicio de *cloud*, selecciona del catálogo los recursos que necesita, se aprovisionan las MV necesarias en pocos minutos y las puede utilizar de forma inmediata y gestionarlas a su voluntad pagando solo por el uso que ha hecho de ellas.

El IaaS permite proveer de infraestructura (puede ser física o virtual, aunque lo habitual es que sea virtual) donde el cliente escoge el tipo de máquina que desea (tipo CPU, *cores*, memoria, red, discos, etc.) y también en las mayorías de los casos, el SO y las cuales se aprovisionan en un breve tiempo (minutos) y el usuario puede comenzar a utilizarlas en forma inmediata y (normalmente) sin la intervención del proveedor.

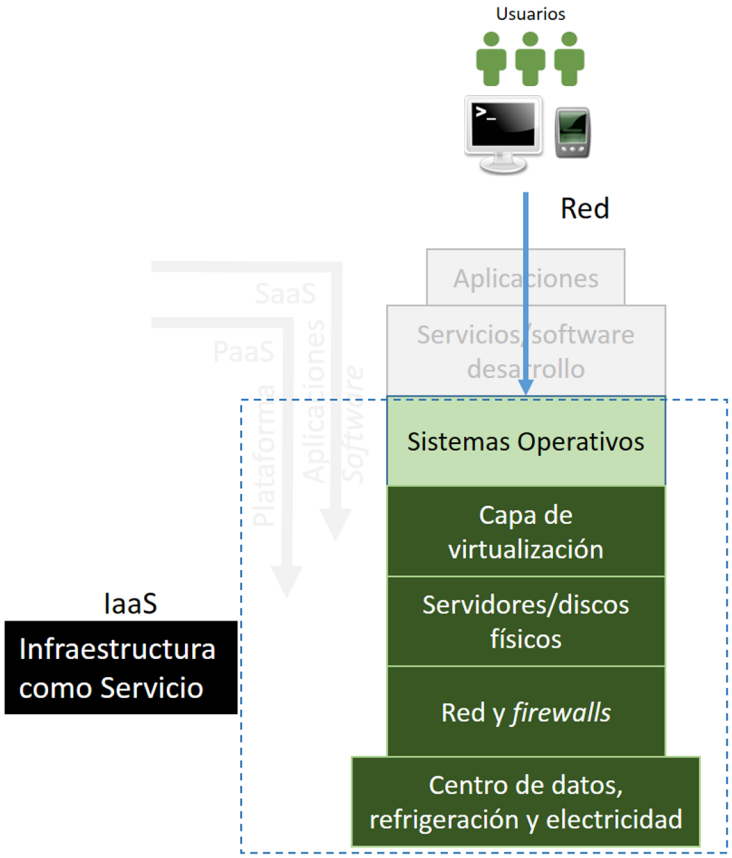
De acuerdo con el informe de Gartner el mercado del *cloud* público en 2016 crecerá un 17,2% (208.600 millones de dólares) donde la parte mayor será para IaaS, que se estima en un crecimiento del 42,8%. Considerando el *Magic Quadrant* para IaaS elaborado por la misma consultora, los principales actores del mercado IaaS en el año 2016 son AWS, Microsoft Azure y Google en grupo destacado (los dos primeros con diferencias), RackSpace, Virtustream, CenturyLink en el segundo grupo y en el tercer grupo VMware, NTT Communications, IBM SoftLayer y Futjitsu.

Es importante ver en el apartado de «proveedores no considerados», ya que hay un grupo de ellos (por ejemplo, Aliyun, Cisco, Oracle, Digital Ocean...) que no cumplen los requerimientos base para ser incluidos en el reporte, pero también tienen una parte del mercado. También es interesante observar los números presentados por el informe de Rightscale sobre la evolución del IaaS y del *cloud* en general sobre encuestas empresas y su comparación con el del 2015 para mostrar las evoluciones o el informe de actividad en cada plataforma *open source* dado por Qingye Jiang (aunque sobre datos de los años 2013-2014) donde compara las tendencias en las plataformas *opensource* con mayor impacto.

El esquema siguiente, ya mostrado anteriormente, muestra la parte que gestiona el IaaS y que puede ser tanto público como privado, ya que el usuario se encontrará delante de una interfaz de gestión que le permitirá realizar todas las operaciones necesarias.

Este tipo de servicio permite (aun siendo privado) delegar la gestión y necesidades de infraestructura IT, a las personas que tengan competencias para ello, dentro de los diferentes departamentos sin necesidad de intervención del departamento de IT. Si el IaaS es público, además, no es necesario contar con

personal especializado para comprar, instalar y mantener la infraestructura, ya que ello será realizado por el proveedor, y permite el escalado de recursos sin preocupaciones añadidas como podrían ser el espacio, la potencia eléctrica o la refrigeración. En el caso de los IaaS públicos toma vital importancia la SLA (*service level agreement*), que es el contrato-compromiso de la calidad de servicio que se debe prestar y las indemnizaciones a las que se tiene derecho en caso de que se incumplan.



Este tipo de infraestructura pueden utilizarse, por ejemplo, para un servicio de transacciones con carga estacional donde la demanda tendrá picos importantes. Esta demanda podrá ser absorbida por el aprovisionamiento automático de nuevos recursos cuando sea necesario, lo cual evitará a la empresa tener inmovilizada una infraestructura en previsión de que estos picos de demanda pudieran ocurrir. Otro caso de uso es por ejemplo un *virtual data center* (VDC), que ofrece una infraestructura conectada de servidores que prestan un servicio complejo pero alojado en un *cloud* integrando todas las operaciones de una empresa y que puede ser en modalidad pública o privada. En este caso, y cuando sea un IaaS público, se deberá tener en cuenta aspectos de alta disponibilidad, resguardos y qué política se sigue ante desastres o qué estrategias sobre la privacidad de la información, ya que toda la información de la empresa está en servidores físicos no controlados por la propia empresa.

Resumiendo, las **ventajas** que presenta el **IaaS público** para los usuarios son las siguientes:

- escalabilidad,
- eliminación de los costes en inversión y sobredimensionamiento por efectos de la demanda posible,
- modelo de pago por uso/recursos que permite reducir costos,
- reducción de infraestructura civil (potencia eléctrica, refrigeración, seguridad, edificio, etc.),
- reducción del personal técnico necesario en relación con un CPD (centro de procesamiento de datos) habitual, y
- reducciones de los puntos únicos de fallo.

En el caso de que el **IaaS** sea **privado**, la ventaja principal es la seguridad y privacidad de la información, ya que se encuentra en los servidores físicos de la empresa y, como se pueden delegar las responsabilidades de gestión y aprovisionamiento, se gana en agilidad y eficiencia. Más allá de las voluntades individuales, es muy importante, antes de la toma de decisiones de migración a un *cloud* público, analizar todos los factores que intervienen y considerar todas las ventajas o desventajas si es público, privado o híbrido para evitar inversiones en tiempo y dinero que luego no serán rentables o que se perderán dado que se deberá volver a situaciones anteriores.

Como **escenarios empresariales** habituales para IaaS, se pueden enumerar:

- DevOps (básicamente para desarrollo y pruebas) ya que los equipos se pueden configurar y desmontar rápidamente en entornos que satisfagan las necesidades de los desarrolladores, reduciendo el tiempo para que las aplicaciones estén probadas y listas para ser publicadas (es decir, permite reducir el *time to market*). Con ello se tendrá el entorno necesario y configurado de acuerdo a las necesidades de las pruebas, escalable y con una considerable reducción de costo (tiempo y dinero) si ello fuera aprovisionado en la propia empresa.
- Sitios web de alta densidad de peticiones: se pueden construir granjas de servidores balanceados para atender a una demanda temporal/estacional que la satisfaga, solo durante el tiempo que dure esta, y con el consiguiente ahorro de recursos y sin inmovilización de activos en la empresa.
- Servicios típicos de IT como son el almacenamiento, las copias de seguridad y la recuperación sin las complejidades que implica una infraestructura de este tipo y las necesidades de personal técnico experto, teniendo en

cuenta además los requerimientos que deben cumplir algunos servicios de estos en cuanto a la privacidad y LOPD (en Europa todos los países tienen requerimientos de LOPD).

- Pruebas sobre cómputo de altas prestaciones (HPC): es posible realizar pruebas de HPC, básicamente para analizar escalabilidad y prestaciones de un desarrollo, y probar con una gran cantidad de *cores* y aplicaciones desarrolladas. Por ejemplo, se realiza un desarrollo sobre 100 *cores*, pero se desea saber cómo será el comportamiento sobre 1.000 *cores*. El IaaS será una solución adecuada, de bajo costo y de reducción notable en el tiempo de provisión y del necesario para tener todo disponible para las pruebas. Asimismo, aquellas aplicaciones que necesiten HPC encontrarán en el IaaS una infraestructura que, si se consideran todos los costos, será rentable en relación con una local.

- *Big data*: análisis de cantidades muy grandes de datos (genes, información de transacciones financieras, análisis social...) que cuestan mucho tiempo y recursos almacenar/mover/procesar. El IaaS permitirá reducir los costes y la aplicación de herramientas a estos datos será muy eficiente y el ahorro de costes será considerable.

- Otros tipos de usos que se pueden considerar, entre otros: *e-learning* (demanda temporal y alto número de usuarios), servicios de *helpdesk*, *virtual data center*, proveedor de *hosting* virtual o de servicios a terceros, servicios de escritorio remoto, etc.

Más allá de analizar los principales actores del *cloud* público y ver las funcionalidades que ofrecen y cómo se realiza una sesión y acceso a un entorno de estas características, es interesante considerar las plataformas más representativas que ayudan a instalar, administrar y gestionar un *cloud* privado.

Algunas de ellas son: OpenStack, OpenNebula o Eucalyptus, que son utilizadas directamente por algunos proveedores para ofrecer servicios a terceros, por lo cual es interesante hacer pruebas de concepto ya que el *cloud* privado no diferirá en nada al *cloud* público y, además, presentará opciones interesantes a la hora de realizar integraciones o generar un *cloud* híbrido.

Para nuestros objetivos, es de especial interés las plataformas *open source* por todas las ventajas que presenta un modelo basado en estas premisas:

- acceso al código fuente y posibilidades de modificación si es necesario,

- reducción de la dependencia del proveedor de código propietario e independencia en las decisiones de continuar o cambiar de producto,

- ahorro en licencias (no confundir con ahorro en implantación, mantenimiento del producto),
- accesibilidad a un mayor número de opciones/herramientas con alta integración en muchos casos,
- decisión de despliegue o no después de pruebas sin compromisos,
- comunidad de usuarios activa,
- incremento de la seguridad ya que se puede acceder al código fuente y analizar funcionalidades no deseadas,
- solución de errores y nuevas implementaciones más rápidas y no sobre criterios comerciales, entre otras.

Si bien las tendencias todavía son que los proveedores de *cloud* público optan por software propietario (una encuesta TechTarget.com ha revelado que el 22% de 260 proveedores de *cloud* han adoptado OpenStack, mientras que un 29% de los encuestados que no utilizan esta plataforma han adoptado una solución propietaria), las tendencias de diferentes proveedores es utilizar plataformas *open source* de renombre y hacer las adaptaciones necesarias antes que comprar una plataforma a un tercero o desarrollar una propia.

1. IaaS público: casos de uso

1.1. AWS

Amazon Web Services (AWS) es una plataforma pública de servicios *cloud* IT y es referenciada por Gartner (2016) como el líder indiscutible en IaaS y dada su actividad (desde 2006) y trayectoria es considerado como un pionero en el campo del *cloud computing*.

Es una plataforma que ofrece una gran cantidad de servicios y una documentación detallada de cada uno de ellos con una serie de tutoriales cortos que permiten a un usuario realizar acciones y poner máquinas en funcionamiento en un tiempo muy breve. En relación con su política de precios adopta «pagar solo por lo que se usa» y «durante el tiempo que se necesite», ya que todos los servicios de AWS se encuentran disponibles bajo demanda y no requieren contratos a largo plazo ni sistemas de licencias. El pago por uso le permite adaptarse con facilidad a las necesidades (dinámicas) de una empresa sin comprometerse a dedicar presupuestos excesivos y mejorando la capacidad de respuesta ante los cambios, por lo cual no es necesario hacer grandes provisiones de presupuesto; esto reduce el riego de aprovisionar demasiada o insuficiente cantidad. Además, para iniciarse, AWS permite disponer de una capa gratuita para ayudar a los clientes a obtener experiencia práctica con los servicios *cloud* AWS durante doce meses. En esta modalidad se podrá utilizar una serie de productos y servicios de forma gratuita dentro de ciertos límites de uso, entre los que se pueden destacar (consultar la lista completa):

- Amazon EC2: Capacidad de cómputo de tamaño variable en la nube: 750 horas por mes de uso de instancia t2.micro en Linux, RHEL/SLES, Windows.

- Amazon S3: Infraestructura de almacenamiento de objetos segura, duradera y escalable. 5 GB de almacenamiento estándar. 20.000 solicitudes *Get*, 2.000 solicitudes *Put*.

- Amazon RDS: Servicio de bases de datos relacionales administrado para MySQL, PostgreSQL, MariaDB, Oracle BYOL o SQL Server. 750 horas de uso de instancias db.t2.micro Single-AZ de Amazon RDS. 20 GB de almacenamiento de base de datos: cualquier combinación de uso general (SSD) o magnético. 20 GB para resguardos (con almacenamiento magnético de RDS) 10 × 106 de E/S.

- AWS IoT: Conexión de dispositivos en el *cloud*. 250.000 mensajes (publicado o entregado) al mes.

- Amazon EC2 *Container Registry*: almacenamiento y recuperación de imágenes Docker. 500 MB al mes de almacenamiento.

AWS ofrece un amplio conjunto de productos globales basados en *cloud*, incluidas aplicaciones de informática, almacenamiento, bases de datos, análisis, redes, móviles, herramientas para desarrolladores, herramientas de administración, IoT, seguridad y *apps* empresariales. Estos servicios permiten reducir los costos de IT a un amplio conjunto de empresas con diferentes necesidades y cargas de trabajo como, por ejemplo, aplicaciones web y móviles, el desarrollo de juegos, el almacenamiento y procesamiento de datos, almacenamiento en general y archivado, entre otras. Entre los productos más importantes se pueden mencionar:

- **Cómputo**: EC2 (VPS), EC2 Container Registry/Service (gestión y ejecución imágenes Docker), LightsailLance (gestión de servidores privados virtuales), VPC (recursos aislados), Batch (ejecución de trabajos en lote a cualquier escala), Elastic Beanstalk (ejecución y administración de aplicaciones web), Lambda (ejecución de código en respuesta a eventos), Auto Scaling (elasticidad automática).

- **Almacenamiento**: S3 (almacenamiento escalable), EBS (almacenamiento en bloque para EC2), Elastic File System (almacenamiento de archivos administrado para EC2), Glacier (almacenamiento de archivos de bajo costo), Storage Gateway (integración de almacenamiento híbrido), Snowball (transferencia de datos a escala de petabytes), Snowball Edge (transferencia de datos a escala de petabyte con computación incluida), Snowmobile (transferencia de datos a escala de exabytes).

- **Bases de datos**: Aurora (BD relacional administrada de alto desempeño), RDS (servicio de bases de datos relacionales administrado para MySQL, PostgreSQL, Oracle, SQL Server y MariaDB), DynamoDB (BD NoSQL administrada), ElastiCache (sistema de almacenamiento de caché en memoria), Redshift (servicio de almacenamiento de datos rápido, sencillo y rentable), Database Migration Service (migración de BD con tiempo de inactividad mínimo).

- **Migración**: Además de los mencionados (DB Migration Service, Snowball, Snowball Edge, Snowmobile) Server Migration Service (migración *on-premise* de VPS).

- **Redes y entrega de contenido**: CloudFront (red de entrega de contenido global), Route 53 (sistema de nombres de dominio escalable), Direct Con-

nect (conexión de red dedicada a AWS), Elastic Load Balancing (equilibrio de carga de gran escala).

Además de herramientas para desarrolladores, de administración, seguridad e identidad, análisis, IA, servicios móviles, servicios de aplicaciones, mensajería, productividad, *streaming* de aplicaciones y escritorios, IoT, y desarrollo de juegos [Aws].

Esto permite que AWS pueda ser escogida para prácticamente cualquier caso de uso tanto del cómputo y almacenamiento de datos a las herramientas de implementación, de los directorios a la entrega de contenido y que pueden ser configurados y aprovisionados por alguno de los 50+ servicios que dispone. En este sentido es posible integrar los recursos de AWS con otros servicios (*cloud*, redes sociales, etc.) y en unos pocos pasos publicar contenidos, obtener información de BD externas gestionar recursos externos o utilizar alguno de los diferentes motores de base de datos, servidores de altas prestaciones o herramientas de *big data* (al 2014 AWS tenía contabilizadas 500 posibilidades para acciones sobre el análisis y tratamiento de la información).

El *cloud* de AWS está disponible en 190 países, 13 regiones geográficas, 35 zonas de disponibilidad y más de 50 puntos de presencia locales que le permiten extender sus servicios y soporte adecuado a cada entorno empresarial y adecuado a la realidad geográfica o legislación existente. Las regiones de la infraestructura se encuentran en Estados Unidos (sin incluir AWS GovCloud), Canadá, América del Sur, Europa, Asia Pacífico y entre sus principales clientes ($+1\times10^6$) se encuentran (lista reducida):

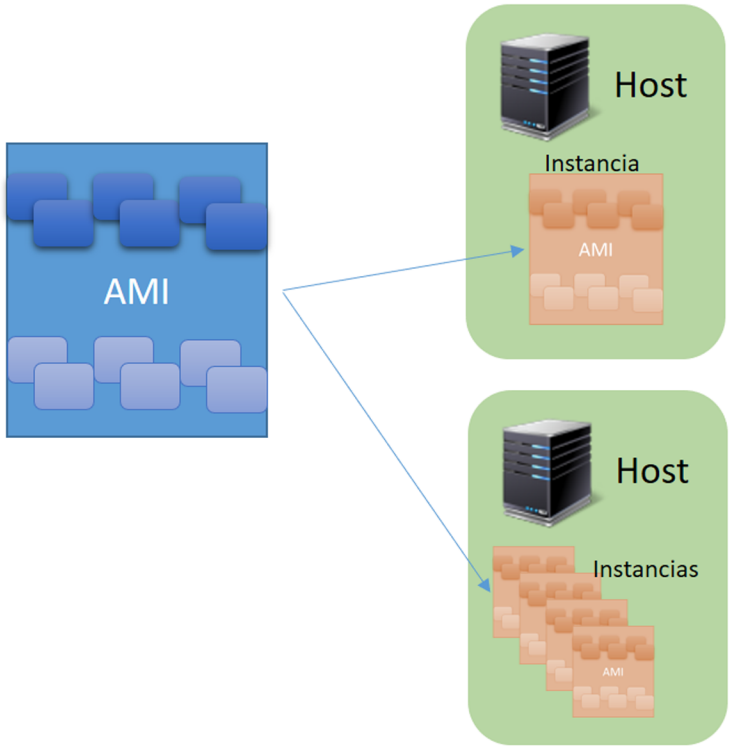
- **Compañías**: Hess, Kellogg's, SunCorp, Vodafone, Expedia, Dow Jones, Docomo, Novartis, Siemens, Adobe, ComCast, Condé Nast...
- ***Startups***: Airbnb, Unababel, Import.io, Instacart, Nextdoor, Lyft, Iflix, Relaeyes, Canary, Duolingo, Slack, Dropcam...
- **Sector público**: FDA, Finra, Code, CDC, PBS, ICRAR, USDA, KNewton, UCBerkeley, UCAS, Coursera, PenState...

Existe una gran cantidad de documentación que el usuario deberá revisar, pero a continuación se hará un resumen sobre los conceptos clave sobre AWS:

a) Interacción con AWS: Esto se puede realizar desde diferentes interfaces: AWS *Management Console* (interfaz web), *Command Line Interface* (AWS CLI), herramientas de línea de comandos (comandos para productos de AWS individuales), kits de desarrollo de software (SDK) de AWS (API específicas de lenguaje o plataforma de programación), API de consulta (API de bajo nivel a las que accede mediante solicitudes HTTP).

b) Instancias y AMI: Amazon Elastic Compute Cloud (Amazon EC2) proporciona capacidad de cómputo redimensionable que se puede utilizar para crear y alojar los entornos software. Una imagen de máquina de Amazon (AMI) es

una imagen que contiene una configuración de software (por ejemplo, un sistema operativo, un servidor de aplicaciones y aplicaciones) y, desde una AMI, se lanza una instancia, que es una copia de la AMI que se ejecuta como un servidor virtual en un equipo *host* del centro de datos de Amazon. Por ejemplo, la AMI de Amazon Linux incluye paquetes y configuraciones que proporcionan la integración adecuada con AWS ya que tiene preinstalada las herramientas API de AWS y CloudInit (Cloud-init es un paquete –estándar *de facto*– multi-distribución para la inicialización de una instancia del *cloud* como, por ejemplo, inicializar el *defaultlocale*, *hostname*, generar las llaves privadas de `ssh` y agregar las públicas del usuario, o inicializar los puntos temporales del *mount*). Las herramientas de la API de AWS permiten la ejecución de tareas dentro de la instancia de Amazon EC2 y CloudInit transmite las acciones de configuración de instancia en el momento del inicio a través de los campos de datos de usuario de EC2, lo cual permite la configuración remota de las instancias de EC2. Cuando lanza una instancia (ver figura siguiente), a partir de una AMI, sobre un *host*, primero se selecciona un tipo de instancia, que determina las capacidades de hardware (como memoria, CPU y almacenamiento) del equipo *host* de la instancia y luego se puede acceder a ella mediante el nombre DNS público o la dirección IP pública que se le ha asignado. Las instancias (que pueden ser múltiples) se ejecutarán sobre el *host* hasta que se las detiene o las termina o hasta que se produce un error (si una instancia falla, puede lanzar una nueva desde la AMI). Se recomienda comenzar con una AMI existente, la que se adapte mejor a las necesidades, iniciar una sesión en la instancia y personalizarla con software y configuraciones adicionales. Esta nueva configuración personalizada se puede guardar como una nueva AMI que se puede utilizar para lanzar nuevas instancias adaptadas.



c) VPC y subredes: Un *cloud* privado virtual (VPC) es una red virtual en la cuenta del usuario de AWS. Está aislada de forma lógica de otras redes virtuales del *cloud* de AWS y proporciona funcionalidad de red segura, siendo prácticamente idéntica a una red tradicional operada por el usuario en su propio centro de datos, con las ventajas que supone utilizar la infraestructura escalable de AWS.

Una **subred** es un segmento del rango de direcciones IP de una VPC en el que puede lanzar instancias. Las subredes le permiten agrupar instancias en función de sus necesidades locales y de seguridad. Para que las instancias de una subred puedan llegar a internet y al resto de los servicios de AWS, se debe añadir una puerta de enlace de internet a la VPC y una tabla de enrutamiento con una ruta hacia internet a la subred. AWS recomienda lanzar las instancias de EC2 en una VPC (el usuario tendrá una VPC predeterminada y deberá lanzar las instancias de EC2 en una VPC predeterminada o no predeterminada).

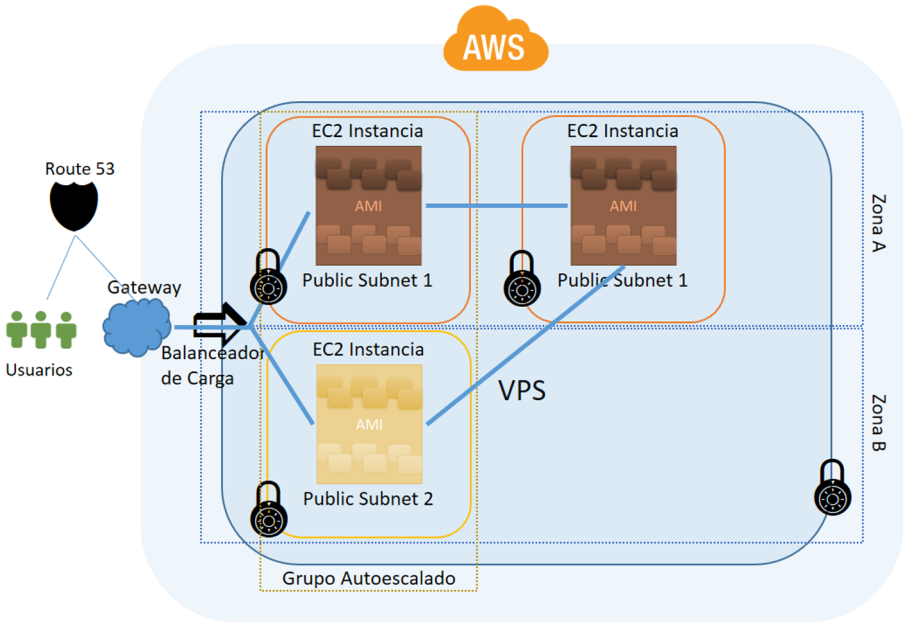
d) Grupos de seguridad: Estos funcionan como un *firewall* virtual de la instancia para controlar el tráfico entrante y saliente. Al lanzar una instancia se puede especificar uno o varios grupos de seguridad. Cuando crea un grupo de seguridad, se tienen que añadir reglas que controlan el tráfico entrante que se permite y un conjunto de reglas que controlan el tráfico saliente. Todo el tráfico restante se descarta. Se pueden modificar las reglas de un grupo de seguridad en cualquier momento y se aplican de forma instantánea.

e) Zonas alojadas de Amazon Route 53: Es un servicio DNS (sistema de nombres de dominio) en el *cloud* escalable de alta disponibilidad que permite traducir nombres de dominio (como www.example.com) en direcciones IP numéricas (como 192.0.2.1). AWS asigna direcciones URL tanto a sus recursos de AWS como a sus instancias de EC2, pero si se desea tener nombres de dominios propios se pueden registrar mediante Amazon Route 53, o si ya se dispone de ellos, transferirlos a Amazon Route 53.

f) Grupos de *auto scaling*: Admite grupos de servidores virtuales, que se podrán aumentar o reducir en su tamaño/prestaciones bajo demanda lanzando/terminando instancias de EC2.

g) Balanceador de carga: Permite distribuir el tráfico a varias instancias para distribuir el cómputo y para conseguir niveles aún mayores de tolerancia a fallos en las instancias iniciadas en diferentes zonas. A medida que se lanzan y terminan instancias, el balanceador de carga dirige automáticamente el tráfico a las instancias en ejecución y realiza comprobaciones de estado en cada instancia. Si una instancia no responde, el balanceador de carga puede redirigir automáticamente el tráfico a las instancias que se encuentran en buen estado.

h) Arquitectura: La figura siguiente muestra la unión de los conceptos mencionados:



En el diagrama se muestra una arquitectura donde existen instancias de EC2 en subredes diferentes donde los grupos de seguridad controlan el acceso a las instancias de las subredes públicas a través de protocolos como SSH o RDP y, además, controlan si las instancias pueden comunicarse entre ellas. El grupo de *auto scaling* mantiene un conjunto de instancias de EC2 que se pueden escalar para atender la carga actual; abarca diferentes zonas de disponibilidad como protección contra el posible fallo de una única zona. El balanceador de carga distribuye equitativamente el tráfico entre las instancias de EC2 del grupo de *auto scaling* y Route *53* ofrece un enrutamiento seguro y fiable de dominio a la infraestructura alojada en AWS.

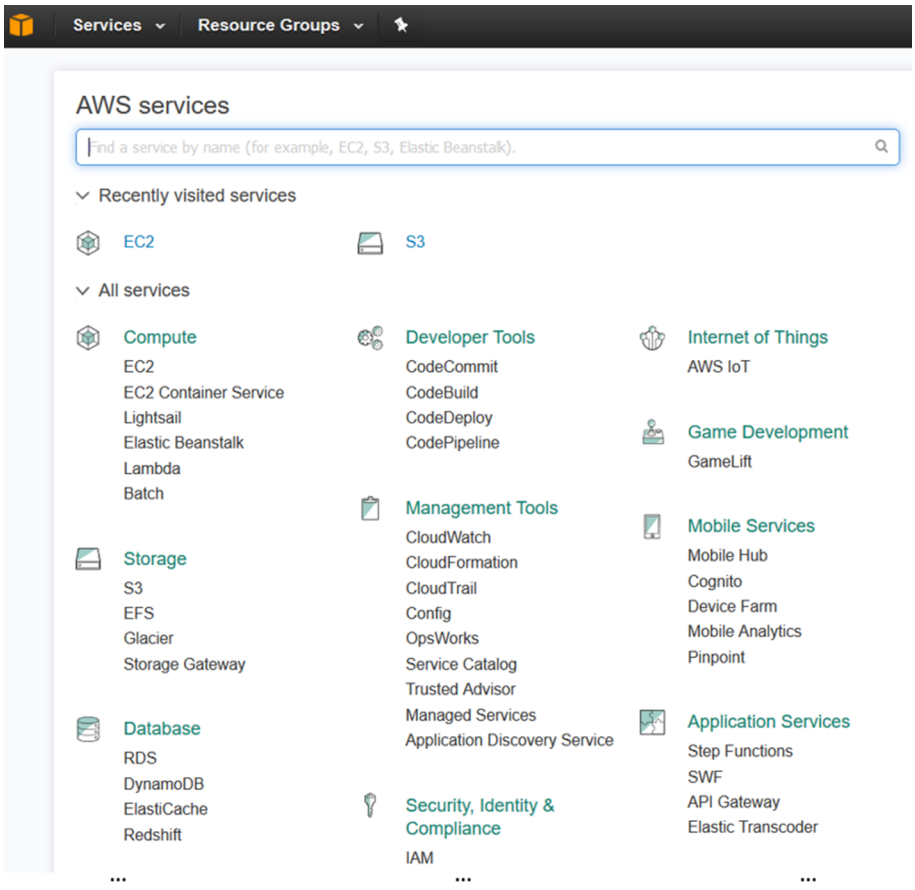
1.1.1. Registro, configuración y puesta en marcha

Como se ha mencionado anteriormente, AWS es un entorno con una gran cantidad de herramientas, opciones y funcionalidad para la casi totalidad de la infraestructura que se pueda necesitar.

En este apartado veremos aspectos introductorios y cómo comenzar, pero solo es una pequeña parte de las posibilidades con las cuales se cuenta (aún con la cuenta gratuita). Para ello, se accederá a través de la cuenta gratuita que ofrece AWS (Free Tier) por un año y con un saldo de 200 dólares, que podrá variar si el procedimiento es a través de la cuenta de un proveedor o intermediario (por ejemplo, la UOC).

En la misma URL (o en la prestación de servicios) se pueden encontrar los recursos que entran dentro de esta opción e incluso algunos que no tienen vencimiento de un año. Es importante tener en cuenta que hay algunos servicios no contemplados en la concesión gratuita y que el estudiante deberá tener en cuenta, ya que estos se facturarán aparte del crédito de la capa gratuita.

El registro habitual es la solicitud de la cuenta, rellenado de los formularios, introducción de una tarjeta de crédito (sobre la cual harán un cargo de 1 dólar para verificación de la identidad del usuario y facturación de servicios no contemplados en la capa gratuita) y verificación a través de un PIN que se ha de introducir en la llamada telefónica que realizará AWS (en pantalla aparecerá un código numérico que el usuario deberá introducir por teclado del TE en respuesta a una llamada automática que le hará AWS). Una vez validado, el usuario tendrá acceso a la consola de gestión (*dashboard*) como la mostrada en la figura (imagen parcial).



Existe una gran cantidad de documentación de cómo comenzar que es conveniente repasar [Aag] [Afs].

El primer apartado a tener presente es el de facturación (*billingdashboard*) dentro de la identificación del usuario (arriba a la derecha). En él, se tendrá acceso a toda la información relacionada con el crédito disponible, el gasto realizado e información relativa a la contabilidad de recursos (cabe tener en cuenta que los recursos se facturan por hora y mensualmente). Una herramienta gratuita y muy útil (accesible desde este mismo *dashboard*) es el Cost Explorer que permite ver gráficas de lo gastado durante los últimos trece meses para ver patrones y analizar costos temporales o fijos y tener información detallada de los recursos y su variabilidad estacional, y es muy útil cuando se deben hacer previsiones o presupuestos en medianas grandes empresas con diferentes usuarios/departamentos. Desde el mismo menú se podrán acceder a la gestión de la seguridad

(*security credentials*), donde podrán crear/modificar los certificados/passwd o crear usuarios con permisos restringidos (a través del IAM, *identity and access management*) para trabajar con parte del IaaS de forma segura.

1.1.2. Lanzar una instancia de MV

La primera prueba de concepto será lanzar una instancia de MV en EC2 y acceder remotamente a ella.

Amazon Elastic Compute Cloud (EC2) es un servicio vía una consola web que se puede utilizar para crear y ejecutar MV en el *cloud* (estas serán llamadas instancias) y en este caso se realizará sobre una MV Linux dentro de AWS Free Tier (es equivalente con otros SO, cabe consultar la documentación).

1) Dentro de la consola ir a EC2 → *Launch Instance*.

2) Configurar la instancia: seleccionar cual AMI (Amazon *Machine Image*) se utilizará (verificar que tenga el cartel de Free Tier), la cual es el esquema que indica que SO, servicios, software tendrá y que en esta prueba se seleccionará Amazon Linux AMI. AWS dispone de dos tipos de virtualización (basada en el hipervisor Xen): **paravirtual** (PV) o **Hardware Virtual Machine** (HVM). La principal diferencia (como ya se vio en el módulo de hipervisores) entre las AMI PV y HVM es la forma en la cual ellas accederán al hardware virtualizado y, para mejores prestaciones, se recomienda utilizar las AMI HVM.

3) Luego se podrá modificar las combinaciones de CPU, memoria, almacenamiento y red para que sea adecuada a las capacidades de las aplicaciones que se desean ejecutar (ver Amazon EC2 Instance Types). La opción por defecto es t2.micro (la cual está dentro del Free Tier) y es adecuada para nuestro fines (otras instancias no están dentro del Free Tier y, por lo tanto, sujetas a costo adicional que será cargado a la tarjeta de crédito introducida).

4) A continuación, se podrá revisar la configuración de los diferentes parámetros adicionales de la instancia o simplemente hacer un clic en *Review and Launch*. Es recomendable al inicio aceptar los parámetros por defecto hasta que se disponga de mayor conocimiento de las opciones/posibilidades de configuración.

5) El siguiente paso será la creación de un par de llaves (pública y privada) que serán utilizadas para acceder a la instancia. Es muy importante este paso ya que sin las llaves NO se podrá acceder a la instancia.

Para ello, primero hay que crear un nuevo par en la opción correspondiente, indicando un nombre (por ejemplo, *mykey*) y salvar la clave privada en un lugar seguro dentro del ordenador local (AWS recomienda salvarla en Windows en un directorio *.ssh* (por ejemplo, *c:\user\nombre_usuario\.ssh\mykey.pem*);

cabe recordar que para crear un archivo con un punto delante en Windows se debe crear como *.ssh.* (punto delante y detrás) y luego el SO quitará el segundo automáticamente.

En Linux/Mac, salvarla en el directorio del *usuario/.ssh* como es habitual (por ejemplo, *$HOME/.ssh/mykey.pem*).

Luego del menú seleccionarla y hacer *Launch Instance* para iniciar el despliegue de la MV. Cabe recordar cambiar los permisos (sobre todo si el *localhost* es compartido) para evitar que sea visible a otros usuarios con:

```
chmod 400 $HOME/.ssh/mykey.pem
```

6) Hacer clic en *View Instances* para ver el estado y los parámetros de la instancia entre los cuales se verá la IP pública que será necesaria para conectarse a ella (estos parámetros saldrán cuando la instancia esté creada y en marcha; si no se visualizan, cabe reactualizar la pantalla desde el icono arriba a la derecha).

La instancia pasará por diferentes estados (*Pending* → *Runnning*) y después el usuario podrá pasarla a otros siguiendo unas determinadas secuencias para pararla, reiniciarla, apagarla o borrarla (*Terminate*) de acuerdo a una secuencia de estados (ver diagrama).

7) Conexión a la instancia: para conectarnos a la instancia se utilizará un cliente `ssh` desde cualquier máquina (en el tutorial de AWS utilizan para Windows el cliente Git Bash). En Linux directamente abriendo un terminal (el cliente `ssh` está instalado en todas las distribuciones) y en Windows se puede utilizar el sugerido (Git Bash) o Putty o Mobaterm (que será el utilizado en esta prueba además del de Linux).

Para ello, desde la pantalla de la instancia obtener la IP pública y ejecutar (para esta AMI el usuario es ec2-user, cada instancia puede tener nombres diferentes, los habituales *ec2-user*, root, ubuntu, and fedora, y se deberá verificar antes de iniciar la sesión con el proveedor de la AMI):

```
ssh -i $HOME/.ssh/mykey.pem ec2-user@{IP_Address}
```

Desde Mobaterm, abrir una consola local y ejecutar el comando:

```
ssh -i 'c:\Users\{usuario}\.ssh\mykey.pem' ec2-user@[IP_Address}
```

Después se deberá aceptar la identidad de la máquina (*key fingerprint*) introduciendo yes y ya tendremos la conexión a la MV remota.

8) Una vez finalizado el trabajo con la instancia es recomendable borrarla (*Terminate*) para que no consuma recursos. Para ello, seleccionar la instancia desde el menú *Actions* → *Instance State* → *Terminate*. En unos instantes la instancia

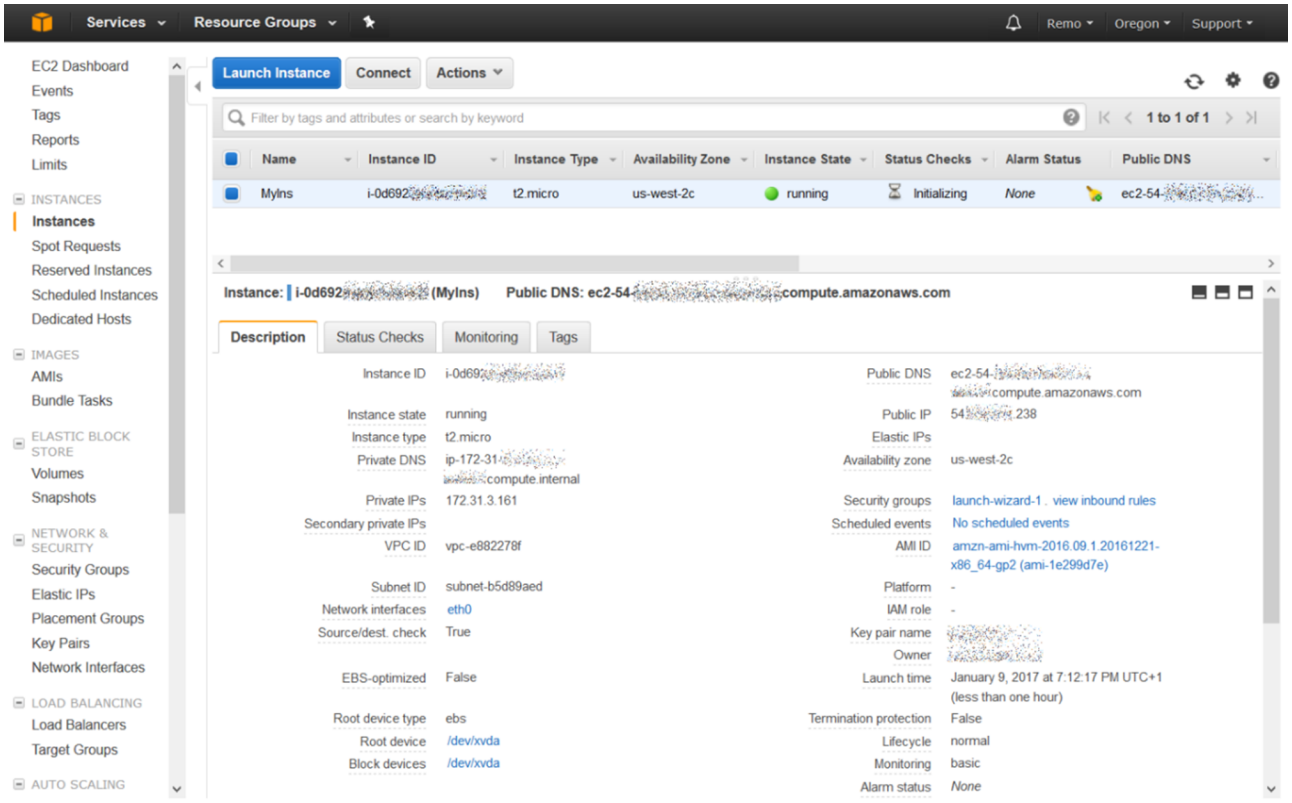
cambiará de estado a *Terminate* y luego de uno ±20' será borrada automáticamente desde el *dashboard*. Cabe recordar que esto también borrará todos los discos ECB asociados (y que no sean permanentes, por defecto no lo son) y se perderá todo el contenido.

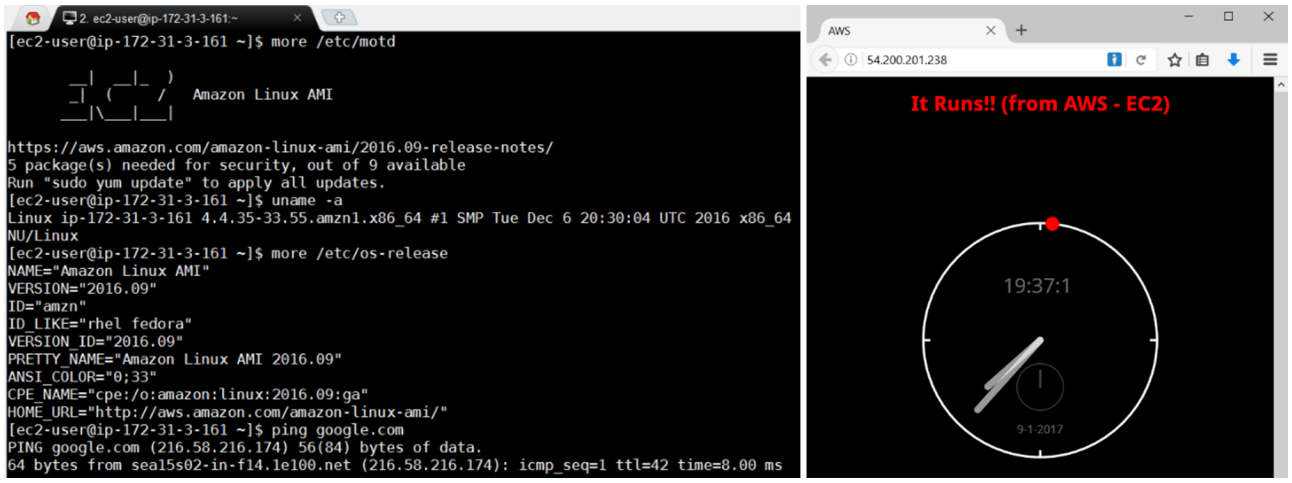
9) Una prueba adicional realizada sobre esta instancia fue la instalación de un servicio web (la instancia es Fedora):

```
sudo -i
yum install httpd
service httpd start
```

Desde Mobaterm (y `scp`) se copiaron los archivos para cargar una página web de pruebas en */var/www/html* (también se puede hacer con el comando `scp` directamente). Uno de los problemas que se encontró es que cuando se creó esta máquina virtual solo se habilitó el puerto 22 para la conexión `ssh`, pero no el 80 para http, pero esto se puede modificar (en caliente, es decir no hay que apagar la máquina) en Network&*Security* → *Security Groups* → seleccionar el grupo al cual pertenece la instancia → pestaña *Inbound* → *Edit* y agregar el nuevo puerto (puerto 80, protocolo http, *source any 0.0.0.0*).

Las imágenes a continuación muestran el *Dashboard* → *Instances* con la instancia en ejecución (*MyIns*) y la conexión por `ssh` (con conectividad externa y la conexión web a la misma IP). Algunas partes de la imagen se han ocultado por privacidad de los datos.





1.1.3. Creación de Website WordPress

En esta prueba se creará un servidor virtual en EC2, pero con una AMI que ya tiene instalado WordPress y solo se deberá configurar el propio servicio.

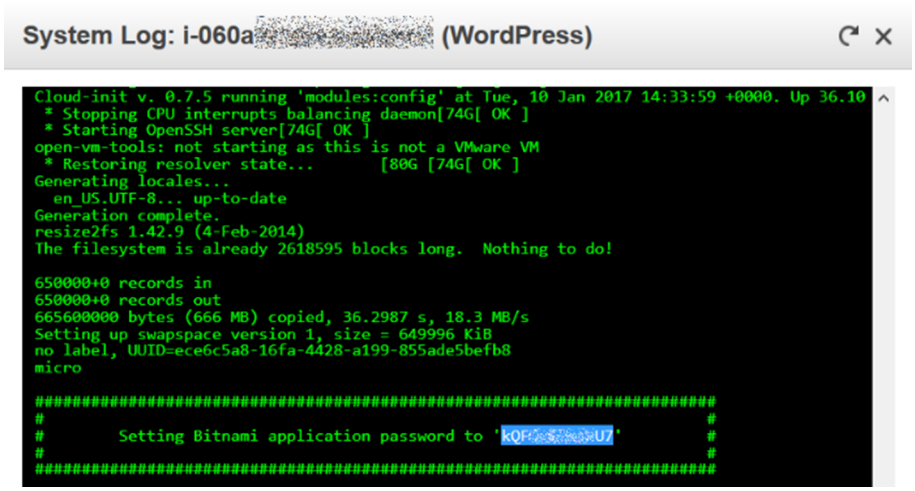
1) Desde *EC2 dashboard* → *Launch Instance* → *AWS Marketplace* → filtrar por WordPress y seleccionar *WordPress powered by BitNami* → *Select*.

2) Se mostrará una página de precios que el software no tiene costo $0.00 y los precios de las diferentes instancias (en t2.micro = $0,012/hora). Al final de la página hacer *Continue* y seleccionar la instancia indicada en la página siguiente.

3) Se podrán revisar las siguientes pantallas *Add Storage* → *Tag Instance* donde se podrá introducir un nombre (por ejemplo, WordPress) → *Review and Launch* → revisar la información → *Launch*.

4) La siguiente pantalla será la de las llaves de acceso, pero en este caso no necesitaremos acceder por `ssh` (aunque se podría configurar) ya que se hará por la propia página de administración de WordPress. Para ello, cabe seleccionar *Proceed without a key pair* → marcar en caja de advertencia → *Launch* → *View Instances* y después de unos instantes ya se podrá conectar a la IP pública de la instancia.

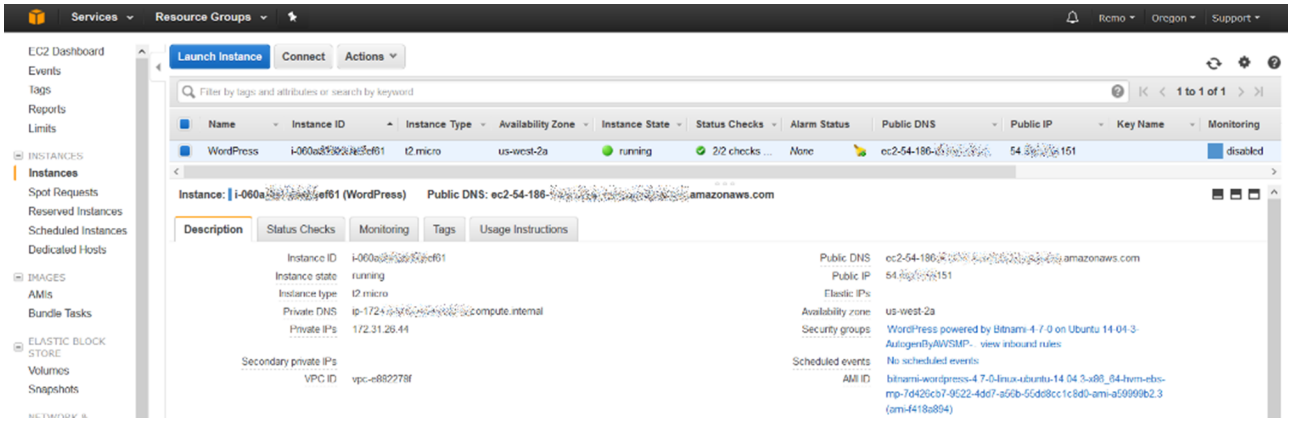
5) Para conectarse y hacer cambios en el sito es necesario encontrar el *passwd*; para ello, en el *Dashboard* → *Instances* → seleccionar la instancia → *Actions* → *InstanceSetting* → *GetSystemLog* y se abrirá una ventana (como la mostrada a continuación) en la cual se buscará una caja formada por '*' donde se encontrará el *passwd* del usuario user de la administración de WordPress. Luego abrir un navegador con la URL *http://{IP_Publica}/admin* y conectarse con el usuario/*passwd* indicados y ya se podrán hacer las modificaciones que se estimen oportunas.

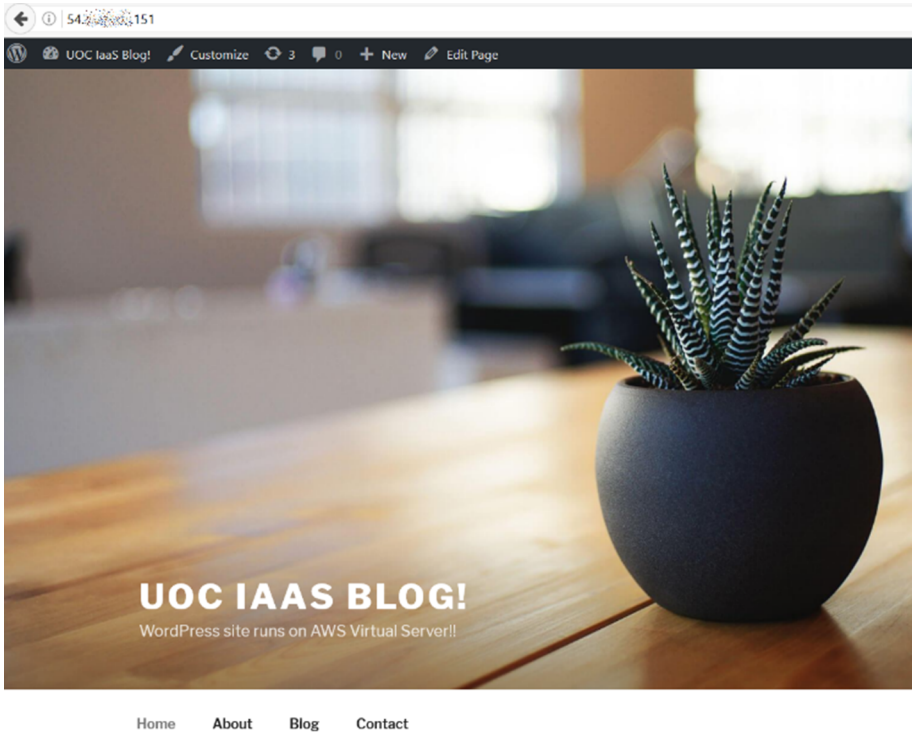


6) El siguiente paso sería registrar un *domain name*, así el sitio podría ser encontrado fácilmente y conectar la instancia con el nombre asignado utilizando Amazon Route 53.

7) En producción no será necesario, pero en la cuenta Free Tierhay hacer un *Terminate* sobre la instancia para no gastar recursos cuando no se utilice o finalicen las pruebas.

En las imágenes siguientes se puede observar el *dashboard* con la instancia y su acceso después de modificada (algunos datos han sido borrados por privacidad).





1.1.4. Gestión de archivos en S3 con consola y CLI

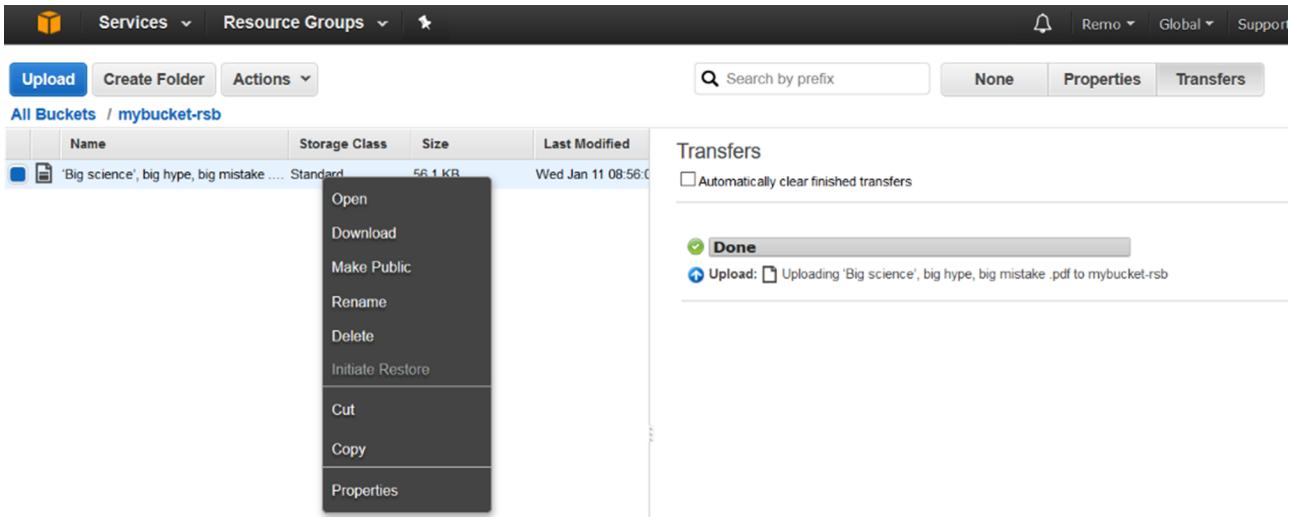
Amazon Simple Storage Service (S3) es un almacenamiento de objetos que permite acceder mediante una interfaz web o a través de la CLI y que permite almacenar y recuperar información desde cualquier sitio, con alta disponibilidad y escalable.

Se puede utilizar como repositorio masivo (*Data Lake*) para tareas de análisis, resguardo (*backup*), como repositorio de recuperación de desastres o cómputo sin servidor. Muchas aplicaciones nativas del *cloud* utilizan S3 como alma-

cenamiento principal y se pueden ordenar por niveles automáticamente en clases de almacenamiento a largo plazo (de menor costo) como S3 Estándar – Acceso poco frecuente y Amazon Glacier para el archivado.

- **Para acceder a S3**: desde *AWS Console* → S3 → *Create Bucket* (nombre único y en minúsculas) y en la región EE.UU. Estándar → *Create*.

- **Para gestionar los archivos**: ir al *bucket* deseado y a través de los botones se puede cargar, crear directorios o realizar acciones con los archivos (abrir, borrar, renombrar, etc.). La figura siguiente muestra un resumen de esta gestión.



Para trabajar desde la CLI se deberá primero crear un usuario y luego instalar el comando `aws` para gestionar AWS por CLI.

1) Crear un grupo/usuario: Desde *Security*, *Identity*... → IAM (*Identity & Access Management*) → *Groups* → *Create New Group* → Seleccionar en *Policy Type* → *Adminstrator Acces*. Desde *Users* → *Add user* → (seleccionar *Programmatic access*) → Permissions → Seleccionar el grupo antes creado *Review* → *Create User*. Accediendo se verá que tiene permisos de administrador (pero no para acceder a la consola).

Desde este usuario, en la pestaña *Security Credentials* → *Create access key* → *Download .csv file* y guardar este archivo que será necesario luego.

2) CLI: se deberá instalar `aws` sobre la máquina Linux (ver la documentación para otros SO). Desde un terminal como *root* ejecutar (en Ubuntu/Debian están el repositorio, pero si en la distribución que se tenga no estuviera se pueden bajar e instalar desde AWS):

```
apt-get install awscli
```

3) Configurar: como el usuario que desee gestionar los repositorios en S3 ejecutar:

```
aws configure
```

E introducir los datos del *archivo.csv* previamente almacenado: *AWS Access Key ID* (es el primer campo y es similar a AKIAPWINCOKAO3U4FWTN), *AWS Secret Access Key* (es el segundo campo y similar a 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), *Default region name = us-east-1, Default output format = json.*

4) Gestión de archivos: se puede crear un *bucket* (el nombre debe ser único a nivel AWS) con:

```
aws s3 mb s3://mybucket-rsb
```

Subir un archivo:

```
aws s3 cp vbox-scripts-9.0.zip s3://mybucket-rsb/
```

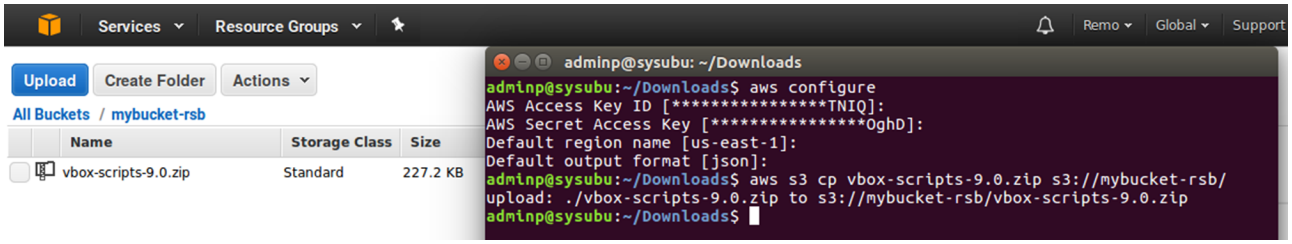
Para descargar el archivo:

```
aws s3 cp s3://mybucket-rsb/vbox-scripts-9.0.zip ./
```

Borrar un archivo:

```
aws s3 rm s3://mybucket-rsb/vbox-scripts-9.0.zip
```

La figura siguiente muestra la ejecución en CLI y la interfaz web.



El siguiente paso sería configurar el AWS Storage Gateway para vincular el programa local de resguardos (*backups*) disponible para que los resguardos se realicen sobre S3.

Como se ha podido mostrar, a pesar de ser una plataforma con gran cantidad de herramientas y opciones, es muy simple iniciarse con ella y el lector dispondrá de gran cantidad de información para utilizar todos los recursos (que parecen inacabables) disponibles en AWS (como siguientes pasos consultar los diferentes tutoriales y la documentación disponible).

1.2. Azure

Microsoft Azure es una plataforma de *cloud computing* propiedad de Microsoft para crear, implementar y administrar aplicaciones y servicios a través de internet, en centros de datos gestionados por la misma empresa, y que proporciona servicios SaaS, PaaS e IaaS y que soporta diversos lenguajes de programación, herramientas y entornos diferentes, incluyendo software y sistemas de Microsoft específicos o de terceros.

Azure fue anunciado en octubre de 2008 (en la conferencia *Professional Developers Conference*) y lanzado como producto comercial en febrero de 2010 como Windows Azure (ahora Microsoft Azure desde el 2014) y tiene una alta reputación en el entorno empresarial, no solamente para sistemas Windows, y es referenciada por Gartner (2016) como otro de los líderes indiscutibles en IaaS.

Microsoft promociona una **lista** más de 600 **servicios** de Azure que, entre los más significativos, se puede mencionar [Mcd]:

- **Procesos**: máquinas virtuales en la modalidad IaaS que permiten lanzar MV de Windows o Linux de uso general, así como imágenes de máquinas preconfiguradas para diferentes paquetes de software (*stacks*). *App Service* como plataforma como servicio (PaaS) para crear aplicaciones web y móviles para cualquier plataforma o dispositivo integrando soluciones SaaS y conectando con aplicaciones locales. *Container Service* es un servicio para para gestionar contenedores dentro de nodos Azure. *Batch* permite ejecutar aplicaciones en paralelo a gran escala y de informática de alto rendimiento (HPC) de manera eficaz en la nube.

- **Redes**: *Virtual Network* representación lógica de la propia red en la nube (VNet) que permite controlar completamente los elementos de la red, segmentar la red virtual en subredes y usarla para iniciar Azure MV o Cloud Services. *Load Balancer* proporciona alta disponibilidad y un elevado rendimiento de red para las aplicaciones. *Application Gateway* permite que los clientes puedan optimizar, a través de un controlador de entrega de aplicaciones (ADC), la productividad de los clústeres de servidores web. *VPN Gateway* implementa una puerta de enlace de una red privada virtual (VPN) para enviar tráfico de red entre redes virtuales de Azure y ubicacio-

nes locales, así como entre redes virtuales dentro de Azure (de red virtual a red virtual).

- **Almacenamiento**: *Storage Services* proporciona las API REST para almacenar y acceder a datos en el *cloud*. *Table service* para almacenar texto estructurado en colecciones a las que se accede mediante la clave de partición y la clave primaria actuando como una base de datos no relacional NoSQL. *Blob Service* para almacenar texto no estructurado y datos binarios como *blobs* a los que se puede acceder mediante una ruta http/s (que además incorpora mecanismos de seguridad para controlar el acceso a los datos). *Queue Service* proporciona comunicación asíncrona mediante mensajes que utilizan colas y *File Service* que permite gestionar datos en el *cloud* utilizando las API REST o el protocolo SMB.

- **Base de datos**: *DocumentDB* servicio de base de datos NoSQL que implementa un subconjunto de la sentencia SQL SELECT en documentos JSON. *Redis Cache* implementación gestionada de Redis. *StorSimple* para gestionar las tareas de almacenamiento entre los dispositivos locales y el almacenamiento en el *cloud*. *SQL Database* para implementar aplicaciones *cloud* con tecnología SQL Server y que se integra Active Directory, Microsoft System Center y Hadoop. *SQL Data Warehouse* servicio de *data warehousing* diseñado para manejar consultas en conjuntos de datos que excedan 1TB.

- **Móvil**: *Mobile Apps Service* ofrece una plataforma de desarrollo de aplicaciones móviles escalable con un amplio conjunto de funcionalidades para estos dispositivos. *Mobile Engagement* es una plataforma de participación de usuarios de software como servicio (SaaS) que proporciona información orientada a datos en el uso de las aplicaciones, segmentación de usuarios y habilita las notificaciones *push* y la mensajería en la aplicación.

- ***Service Bus*** permite la comunicación de las aplicaciones que se ejecutan dentro de Azure o dispositivos externos, lo cual permite construir aplicaciones escalables y confiables en una arquitectura orientada a servicios (SOA). Este soporta *Event Hubs* (para el acceso a eventos y telemetría), *Queues* (permiten la comunicación unidireccional), *Topics* (proporcionan comunicación unidireccional utilizando un patrón de abonado), *Relays* (proporcionan comunicación bidireccional).

- ***Azure Search*** para la búsqueda de texto y un subconjunto de filtros estructurados que utilizan API REST.

- ***Monitoring and Diagnostics***: analizar el rendimiento, descubrir cuál es la fuente de errores si los hay, analizar la seguridad e identificar las tendencias de consumo.

- ***Media Services***: infraestructura PaaS que se puede utilizar para la codificación, la protección de contenido, la transmisión o análisis.

- **CDN**: red de distribución de contenido global (CDN) para audio, vídeo, aplicaciones e imágenes que ofrece a los desarrolladores una solución global para entregar contenido de alto ancho de banda que se hospeda en Azure o en cualquier otra ubicación.

- ***Azure Automation*** permite automatizar las tareas manuales, de ejecución prolongada, propensas a errores y frecuentemente repetidas que se realizan normalmente en un entorno *cloud* empresarial. También se integra con Microsoft SMA.

- ***IoT Hub***: servicio administrado que permite la comunicación bidireccional fiable y segura entre dispositivos IoT y un *back-end*.

- ***Azure Active Directory***: es el directorio basado en la nube multiempresa y el servicio de administración de identidades de Microsoft.

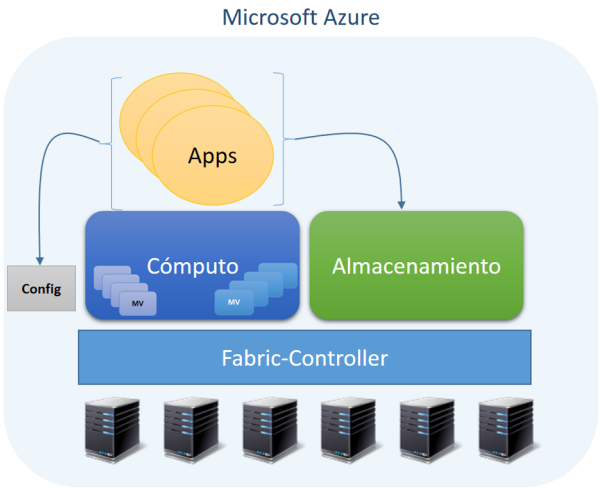
- ***Machine Learning***: mediante Azure ML como parte de Cortana *Intelligence Suite* permite la analítica predictiva y la interacción con datos usando lenguaje natural y voz a través de Cortana. HDInsight es una distribución de Apache Hadoop que crea los componentes de Hadoop a partir de la distribución de Hortonworks Data Platform (HDP) e implementa y aprovisiona clústeres administrados con confiabilidad y disponibilidad.

- **Herramientas de desarrollo**: Visual Studio Team, Application Insights.

Una lista completa se puede consultar en la documentación para Windows Azure.

Azure está disponible en 30 regiones (8 regiones adicionales en el futuro próximo) en América, Europa, Asia Pacífico, que permiten dar una cobertura global y adaptada a las necesidades y legislación de cada mercado. Entre los grandes clientes de Azure se pueden nombrar Heineiken, Sulekha, Hershey, Hendrick Motorsports, Crystal Group, Tyco, First Tech, Hollands Kroon, Tacoma Public Schools, Jet.com, Ecolab, Real Madrid CF, GE Healthcare, NBC Universal, 3M, entre otros.

La estrategia de Microsoft Azure permite la ejecución de *apps* tanto en la plataforma *cloud* como en sistema locales (WServer, W7/8/10, WMobile...) sobre la base de *.Net Services*, *SQL Services* y *Live Services*. La figura siguiente muestra la estructura básica de Azure [Iaa].



Azure está formado por un conjunto de máquinas ubicadas en los CPD de Microsoft en agrupaciones denominadas clústeres donde cada uno de ellos es gestionado por el *Fabric-Controller* (FC), que es el responsable de aprovisionamiento, gestión, provisión de servicios y gestión del ciclo de vida de los procesos/servicios formando el *kernel* del *Cloud Operating System* de Azure (formado por un conjunto de imágenes de Windows Server virtuales modificados que se ejecutan sobre los nodos de la infraestructura Azure) [Haw].

Azure considera (a partir de las definiciones de Hyper-V) diferentes conceptos:

a) ***Role***: es un nombre que se le da a una configuración específica de una MV Azure.

b) ***Service***: Azure permite a los usuarios ejecutar servicios que se ejecutan en una instancia de una MV en un rol preconfigurado, como web o *worker roles*. Un *service* es una cola de instancias que son gobernadas por los mismos parámetros y política.

c) ***Web role***: una instancia preconfigurada para ejecutar el entorno de Web Server (*Internet Information Services*, IIS).

d) ***Worker role***: una instancia configurada para ejecutar aplicaciones por el usuario en la MV o cargadas en estas.

e) VM *role*: MV creada por el usuario y cargada a través del portal que son diferentes de las gestionadas dentro del web y *worker roles* ya que estas no deben ser cargadas en el portal y son mantenidas por Azure.

Cuando un usuario se conecta en Azure y solicita una nueva instancia, este pasa la petición a RDFE (*RedDog Front End*) y el cual lo traslada a FC que a su vez y con los parámetros incluidos por el usuario, la localización del usuario, proximidad, busca nodos disponibles (2 diferentes para la tolerancia a fallos) para crear las instancias solicitadas. Cuando encuentra los que satisfagan este

requisito, el SO *host* crea la MV (a través de Hyper-V) y 3 discos virtuales, vhd (*D:*\ para la imagen del SO, *C:*\ para los archivos temporales de usuario un tercero para el *role* VHD –que será la siguiente letra disponible– para los archivos específicos del rol). Cuando todo esto está listo el agente inicia las MV, actualiza las referencias y ya está todo disponible para el usuario. Es interesante ampliar estos conceptos con la referencia [Haw] para ver cómo Azure gestiona los roles y las instancias para garantizar el 99,99% de fiabilidad, la seguridad y otros aspectos innovadores de la plataforma.

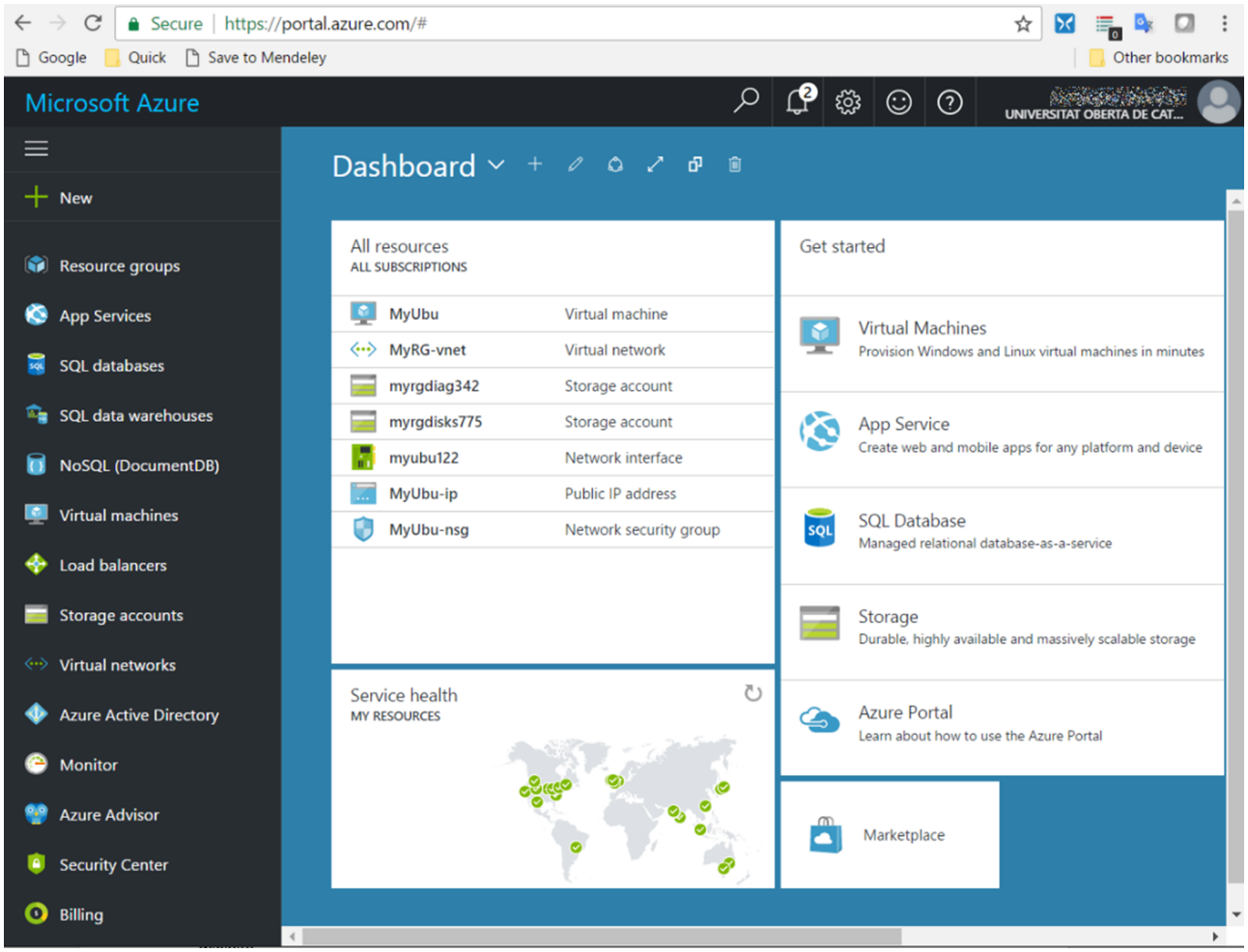
La documentación es extensa y se recomienda seguirla en el orden de la capas más importantes de la infraestructura: Azure Compute, Azure Storage, Azure SQL, Azure Virtual Network.

1.2.1. Registro, configuración y puesta en marcha

Azure también proporciona una cuenta gratuita a la que el usuario se puede suscribir con un crédito de 170€ (30 días); con ella se podrán aprovisionar MV (hasta 14), 40 bases de datos SQL, 50 redes virtuales u 8 TB de almacenamiento para un mes y se podrán utilizar todas las herramientas/entornos para la creación de aplicaciones web, móviles y también la API que usen los servicios *Redis Cache*, *Search* o *Content Delivery Network*. La cuenta también permite utilizar los servicios de datos masivos con *MachineLearning*, *StreamAnalytics* y *Hadoop* o crear aplicaciones de Internet de las cosas (IoT) en tiempo real con supervisión y control de errores e incluso continuar disfrutando de algunos de ellos después del vencimiento (lista completa).

El registro no es diferente a lo habitual; no obstante, al final se debe hacer una suscripción (aunque se puede obviar, pero finalmente se deberá hacer) para acceder a crear recursos y el mecanismo de verificación de identidad de Microsoft es a través de una tarjeta de crédito [Afa].

Una vez creada la cuenta se puede acceder al portal de Azure, el cual presentará un *dashboard* como el que se muestra a continuación [Azp].



La documentación de Azure es muy extensa y el lector dispondrá de tutorial y casos de usos que le ayudarán a realizar las primeras acciones sobre la plataforma. El *dashboard* está organizado de forma similar a otras plataformas y dispone de un menú vertical desde el cual se puede acceder a los principales recursos de la misma. En la parte superior derecha se mantienen los accesos a las notificaciones, configuración (aspecto), alertas y cuenta del usuario. La dinámica de la interfaz es un panel que se va extendiendo a la derecha con la ruta relativa en la parte superior para el acceso rápido y vuelta atrás a las diferentes pantallas; además, todo se puede personalizar (colores, tamaños, pantallas ancladas, etc.). Si bien en las notificaciones irá saliendo información respecto al crédito disponible, es importante consultar el apartado de facturación (*billing*), donde existirá un desglose detallado de los costes y en qué se han realizado.

1.2.2. Creación de una MV desde el portal

Para la creación de una MV debemos realizar los pasos siguientes:

1) Hay que tener una suscripción activa (y sino, se accederá a la sección donde darla de alta) y hacer *New* → *Compute* → *UbuntuServer16.04* → Verificar el *deployment model* sea *ResourceManager* → *Create*.

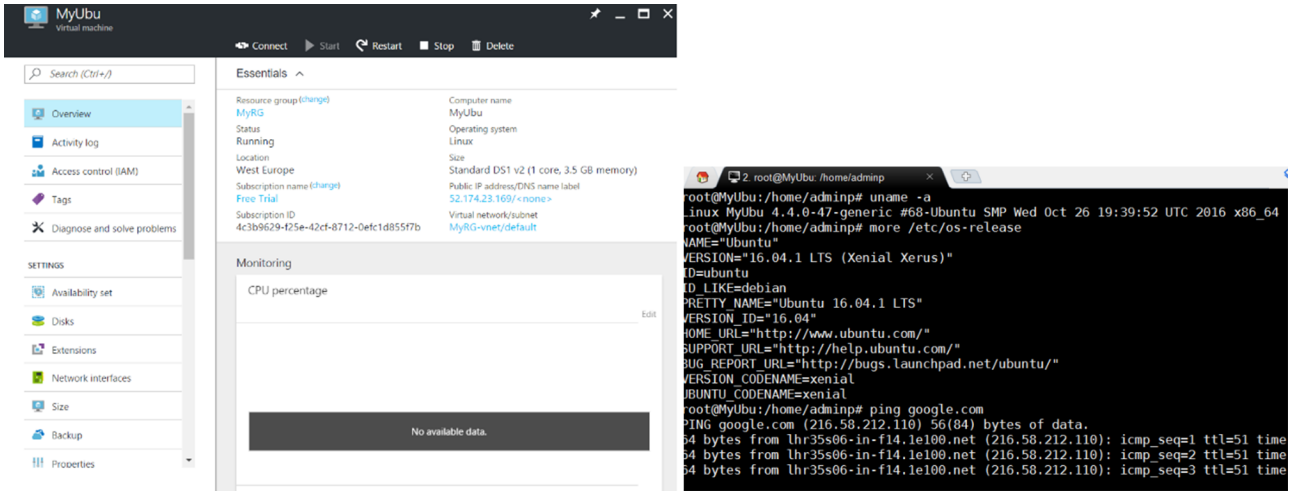
2) Luego se deberán indicar los parámetros básicos de la MV: nombre, tipo de disco, nombre del usuario y *passwd* (o también la llave pública si se accede por llaves privada/pública), la suscripción y el *data centre* (por ejemplo, West Europe; cabe ir con cuidado, pues los precios pueden variar en función del servicio/DC) e indicar un *resourcegroup* (agrupación que luego permitirá aplicar la misma política a todos los recursos → OK.

3) A continuación, se deberá definir el tamaño de la MV;en este caso se ha escogido la más pequeña (DS1_V2 con 1 *core*, 3,5GB RAM, 2 discos y 7 GB de SSD con un precio estimado de 42,66 €/mes) y se deberán revisar otras configuraciones (que establece por defecto) como la cuenta de disco, la red, la subred, la IP y los puertos y cuestiones adicionales como monitorización, alta disponibilidad o extensiones.

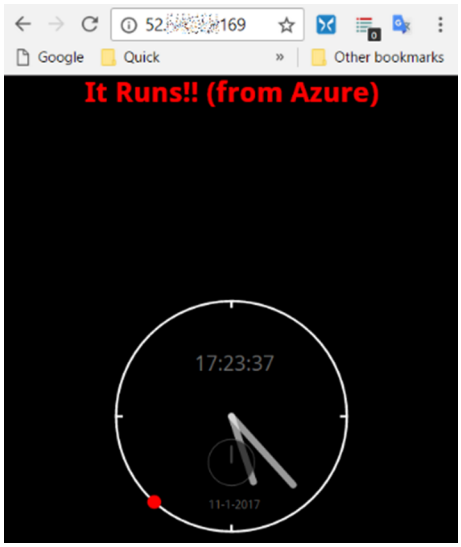
4) Finalmente, se realizará una validación y de unos instantes se iniciará el despliegue y ya se podrá ver en el *dashboard* (en estado de *deploying* primero y luego de *running*).

A partir de este momento se podrá acceder a la MV, ver la IP pública asignada y conectarse remotamente a través de `ssh`. Desde el *dashboard* → MV se podrá acceder a toda la información relacionada con ella y, si es de interés, anclar al portal algunas de las gráficas que sean de utilidad.

En las figuras siguientes se muestra el *dashboard* con diversas gráficas de monitorización ancladas, los detalles de la MV y el acceso por `ssh` y su conectividad externa.



Si desde a máquina se desea agregar un servicio, como Apache, se puede instalar y arrancar de la forma convencional, pero la MV no dispone de puerto 80 abierto (esto se realiza en el último paso de la configuración agregando una *inboud security rule*. Para ello se debe ir a la red de la MV (o desde el menú vertical) y acceder a *Security Group*, creando una nueva regla para esta MV, y agregar el puerto 80/tcp. Después de unos instantes ya se podrá acceder al servidor web en la MV como muestra la figura.



1.2.3. Gestión del almacenamiento

El almacenamiento de Azure permite que las aplicaciones/recursos dispongan de espacio persistente, escalable/elástico, de alta disponibilidad y con diferentes tipos de datos (*blob*–objetos–, tablas NoSQL, mensajes en cola y archivos). La arquitectura de almacenamiento utiliza un sistema de creación automática de particiones que equilibra la carga de los datos automáticamente en función del tráfico, de forma que cuando crece la demanda, se le asignan los recursos adecuados para ajustarse a ella. Como clientes permite el montado desde los SO más importantes (Windows, Linux/MacOS) y dispone de una API REST para integrarse y compartir datos con otras aplicaciones mediante HTTP/HTTPS.

En cuanto a los **tipos de almacenamiento** se pueden clasificar en:

- **Almacenamiento de *blobs*** (datos de objetos no estructurados): puede ser cualquier tipo de datos como binarios, texto, un documento, un archivo multimedia o un instalador de aplicación.

- **Almacenamiento de tablas** (conjuntos de datos estructurados): permite almacenar datos del tipo clave-atributo NoSQL para el despliegue rápido de grandes cantidades de datos y el acceso inmediato a los mismos.

- **Almacenamiento en cola**: solución de cola de mensajes para el procesamiento de flujos de trabajo y para la comunicación entre los componentes de servicios en el *cloud*.

- **Almacenamiento de archivos**: almacenamiento compartido para aplicaciones que utilizan el protocolo SMB, por lo cual las MV, los servicios en el *cloud* de Azure y las aplicaciones externas/clientes pueden compartir datos a través de recursos compartidos montados o a través de la API REST.

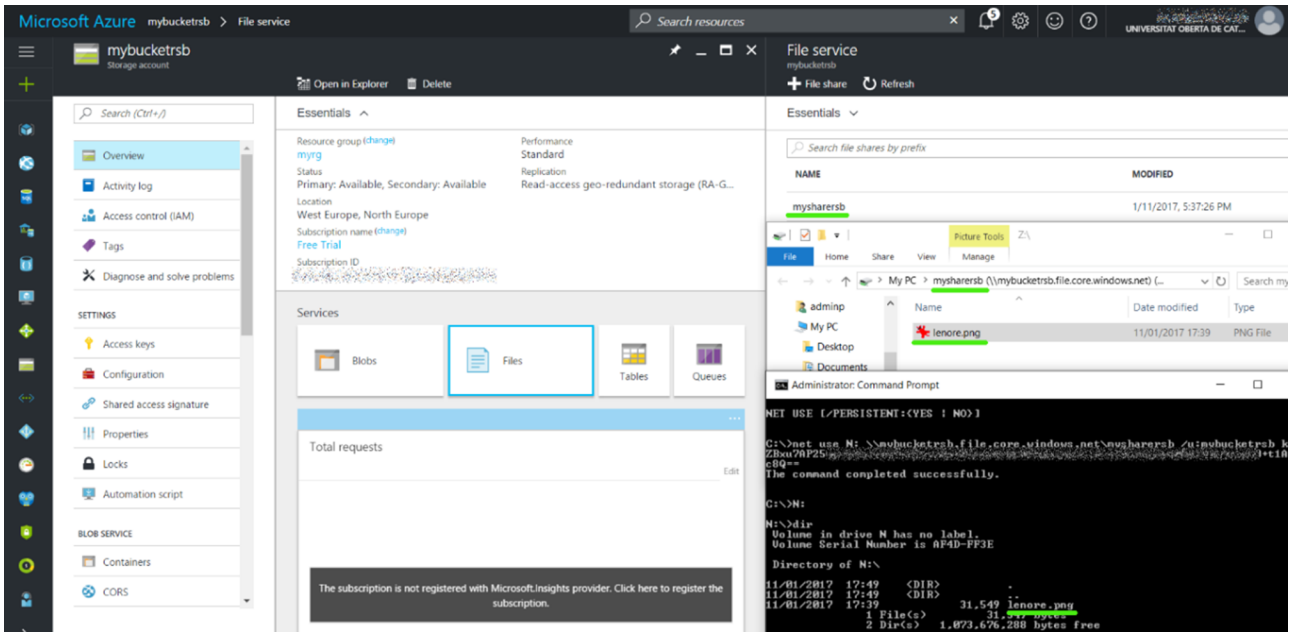
Para crear una cuenta de almacenamiento *New* → *Storage* → hay que seleccionar el nombre de acuerdo a las indicaciones (*mybucketrsb* en nuestro ejemplo) y el resto de parámetros (se aconseja dejar los que pone por defecto en los primeros pasos) e introducir el *Resource Group* (se puede utilizar el que ya se tiene disponible o crear uno nuevo) y marcar (o no) si se desea anclar el recurso al *dashboard*. Luego se puede acceder desde el *dashboard* o desde el menú lateral y crear un *file shared* seleccionando *FileService* → proporcionando un nombre (*mysharersb* en nuestro ejemplo) y dentro de este hacer el *upload* de los ficheros y realizar la gestión que con ellos se desee.

Para montar estos recursos desde un Windows se puede hacer desde PowerShell, desde una terminal `cmd` o desde la interfaz gráfica. Desde una terminal de `cmd` se deberá ejecutar:

```
net use N: \\mybucketrsb.file.core.windows.net\mysharersb /u:mybucketrsb [acces-key]
```

Donde el recurso forma parte de la URL, el usuario será el nombre del recurso (*mybucketrsb*) y la llave se obtendrá desde el menú vertical del *Bucket* creado (accediendo a *Access Key*).

Desde la interfaz gráfica se abrirá un explorador de archivos en *Computer* → *Map Network Drive* → indicar la letra, la URL y conectarse con credenciales diferentes → introducir el usuario/clave. La figura siguiente muestra el *File Shared* (*mysharersb*) desde el portal y el acceso (abajo a la derecha) a un archivo (*lenore.png*) desde un Windows 10 a través de la interfaz gráfica y desde la terminal.



Para eliminar el acceso compartido desde la interfaz gráfica desde el mismo menú anterior (*Disconnect network drive*) y desde la terminal: `net use /del N`:

1.2.4. Crear una *web app* sobre Linux

Para crear una aplicación web sobre Azure desde un entorno desarrollo local en Linux es muy simple (consultar a, b), para ello: *New* → *Web+Mobile* → *Web App* → introducir el nombre y el *Resource Group* (en este caso se ha escogido el mismo de ejemplos anteriores) → *Create*.

Desde el panel vertical (o del *dashboard*) escoger *App Services* → la aplicación creada en la cual ya veremos toda la información; alguna de las cuales deberemos modificar para adecuarla al método de desarrollo a través de Git. Dentro del menú de nuestra aplicación → *AppDevelopment* → *Quick Start* → PHP → *Local Source Control* → *Configure deployment credentials* (introducir un usuario y *passwd*) y en *Developer Options* → *Choose Source* → *Git Local Repository* con lo cual ya veremos en *Overview* la URL para hacer un *clone* (será algo como: *https://myuserrsb@myapprsb.scm.azurewebsites.net:443/myapprsb.git*).

Sobre nuestra máquina Linux tendremos nuestro proyecto; en este caso utilizaremos uno de prueba en GitHub):

```
git clone https://github.com/Azure-Samples/app-service-web-html-get-started.git
cd app-service-web-html-get-started
git remote add azure \
 https://myuserrsb@myapprsb.scm.azurewebsites.net:443/myapprsb.git
git push azure master
```

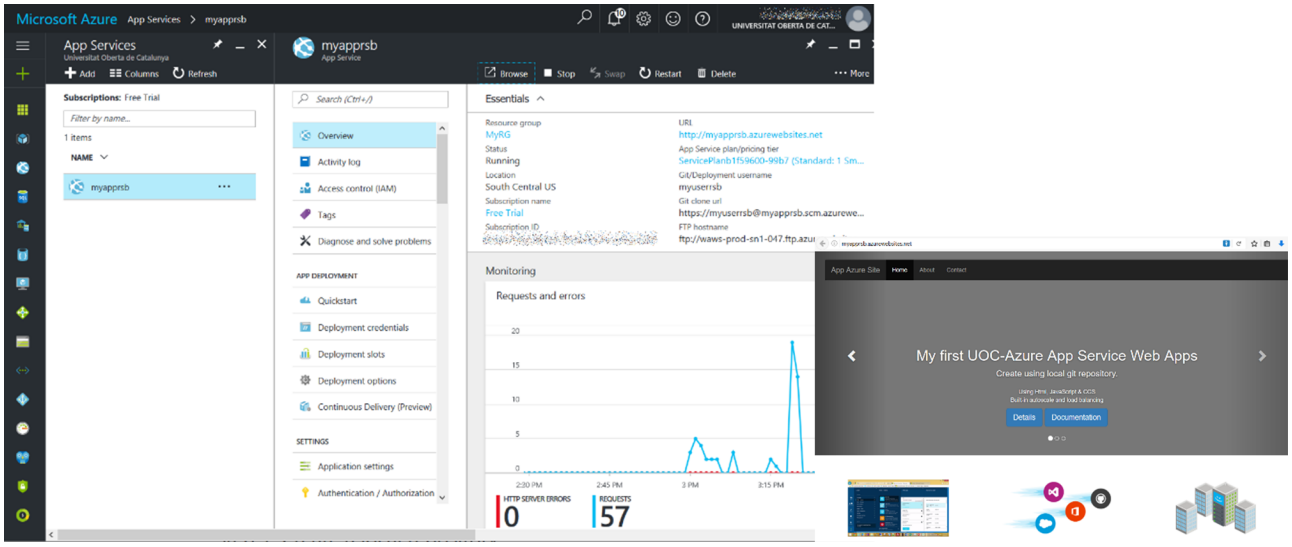
Para visualizarla desde la misma *App Service Overview* → *Browse* (en el *tab* superior donde está toda la información de la *app*) o desde Linux ejecutando:

```
az appservice web browse --name myapprsb --resource-group {RGroup}
```

El *Resource Group* es donde se ha creado la aplicación y está también la ventana de *Overview*. Si se desea modificar y hacer una actualización del repositorio local es muy fácil:

```
git add .
git commit -m "Modificación del index.html"
git push azure master
```

Y ya se podrá ver la actualización. En las figuras siguientes se muestra la *App Service* y el sitio web desplegado con modificaciones del original.



Es evidente que, al igual que en las otras plataformas, esta es una plataforma con una gran cantidad de aplicaciones/funcionalidades a las que el usuario deberá dedicarle tiempo y esfuerzo a la hora de gestionarlas y adaptarlas a sus necesidades. La documentación es muy completa y organizada e incorpora ejemplos y casos de uso para mejorar la curva de aprendizaje de los *sysadmin*/ usuarios [Ado].

1.3. Google

Google Cloud Platform (GCP) es una plataforma de recursos (servidores y discos y recursos virtuales como por ejemplo máquinas virtuales –MV–, entre otros) que se encuentran en los CPD de Google distribuidos por regiones el mundo y que pueden ser utilizados en diferentes modalidades de *cloud*.

Las regiones incluyen Estados Unidos, Europa Occidental y Asia Oriental, donde cada región implica un conjunto de zonas que se encuentran aisladas unas de otras para permitir la redundancia y la tolerancia a fallos, así como para reducir la latencia utilizando los recursos más cercanos a los clientes. Esta delimitación por zonas permite que a algunos recursos (considerados globales) se pueda acceder desde cualquier zona (por ejemplo, imágenes preconfiguradas de disco, instantáneas de disco y redes), pero a otros solo se puede acceder desde la misma región (por ejemplo, instancias de máquina virtual, sus tipos y discos).

Cualquier recurso de GCP que se asigne o use deberá pertenecer a un **proyecto** que es algo así como la entidad gestora en la cual se está trabajando y estará formada por las configuraciones, permisos y metadatos que describen las aplicaciones del usuario y que permitirán que los recursos en un mismo proyecto

pueden cooperar fácilmente, por ejemplo, mediante la comunicación a través de una red interna, y estarán separados de los recursos de otros proyectos (solo se podrán comunicar a través de una conexión de red externa).

Cada proyecto dispondrá de un **ID único** dentro de la plataforma (y vinculado a la cuenta de facturación) que será necesario para determinados comandos o llamadas a la API y que será perpetuo sobre GCP (aunque se haya eliminado el proyecto).

GCP ofrece tres **formas básicas** para interactuar con los servicios y recursos:

1) GPC consola (interfaz de usuario web que permite GUI para gestionar proyectos y recursos de la plataforma de la nube).

2) Interfaz de línea de comando (CLI, el SDK de Google Cloud proporciona una herramienta `gcloud` que da acceso a comandos para gestionar el flujo de trabajo y los recursos de la plataforma).

3) Librerías (en el SDK) que permiten crear y administrar recursos fácilmente a través de las API con la finalidad doble de proporcionar acceso a los servicios y ayudar en la administración para la gestión de recursos.

La política de costos de GPC permite ofrecer la mejor relación prestaciones/costo, facturación por minutos y descuentos automáticos con el incremento de la utilización, así como como una política de pago-por-uso y durante-tiempo-de-uso. Es interesante consultar la calculadora de costos y el TCO (consultar la referencia [GCP] donde se compara el costo anual de Server Web App Package en diferentes proveedores, siendo GPC –datos de noviembre de 2015– 6.415$, Azure 2,2 veces más caro, Rackspace-Manage Cloud 1,6 veces y sobre IBM/Softlayer 1,4 veces.

Google ofrece diferentes servicios y recursos agrupados por cómputo, contenedores, almacenamiento, redes, *big data*, *machine learning*, *operations*, *auth* y *security*, gestión de recursos, y herramientas de desarrollo. A continuación, se analizarán los más representativos (lista completa):

- Cómputo: Compute Engine (control sobre las infraestructuras), App Engine (una plataforma para crear *web apps* y *mobile backends* escalables), Container Engine/Registry (para ejecutar/almacenar contenedores Docker bajo el control de Kubernetes).

- Almacenamiento y BD: Cloud Storage (almacenamiento de objetos), Cloud SQL (MySQL database), Cloud Bigtable (NoSQL database), Cloud Datastore (NoSQL, schemaless database), Persistent Disk (block storage para instancias de MV).

- Red: Virtual Network (red gestionada para los recursos en GCP), Load Balancing, CDN (Content Delivery Network), Cloud DNS.

- *Big data*: BigQuery (*data warehouse for large-scale data analytics*), CloudDataflow (procesamiento en tiempo real de datos *batch* y *stream*), Cloud Dataproc (servicio Spark y Hadoop).

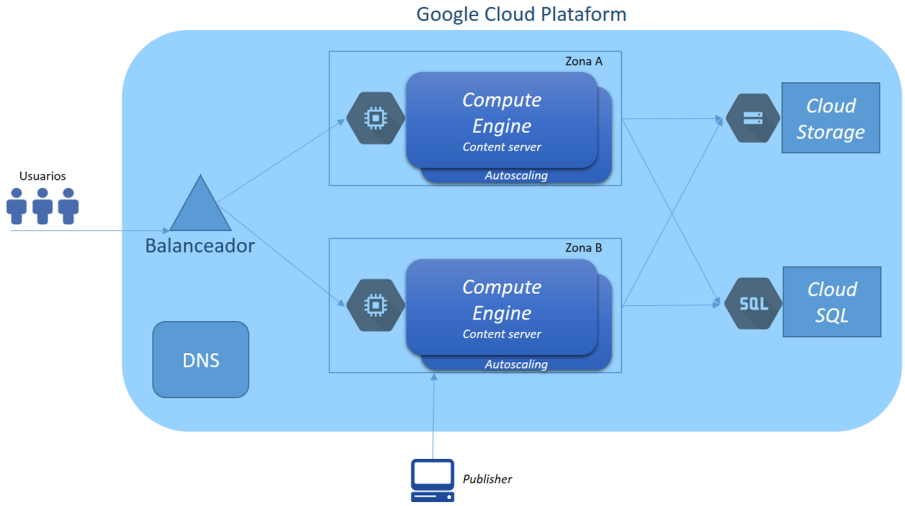
- *Machine Learning*: *job search, natural language, speech, translation, visión* API.

- Herramienta de gestión: Stackdriver, Deployment Manager, Cloud Endpoints, Cloud Mobile App.

- Herramientas de desarrollo

- *Identity & Security*: *Cloud IAM* (*Identity & Access Management*), Cloud Resource Manager, Cloud Security Scanner, Cloud Platform Security Overview.

La figura siguiente muestra la arquitectura de GCP en uno de sus roles habituales (gestión y procesamiento de contenido).



Además de la infraestructura IaaS (o PaaS/SaaS) y la gran cantidad de herramientas y opciones de que dispone, existe una serie de **ventajas** (promocionadas por el propio Google) de GCP en relación con sus competidores, entre las que se pueden mencionar:

- Gestión de MV y *uptime*: gracias a las posibilidades de migración en caliente (totalmente soportada entre *hosts* cercanos) incluso en situaciones de carga extrema e incluido su almacenamiento SSD (de hasta 1,5 TB) y que, además, las MV no necesitan ser reiniciadas para las actualizaciones de software del *host* u otras tareas operativas, el tiempo disponible es en la mayoría de los casos del 100% garantizando así un rendimiento previsible en las aplicaciones.

- Políticas de costos: facturación por minuto, descuentos de uso sostenido con precios óptimos, tipos de máquinas personalizadas y precios especiales para casos de uso particulares; ofrecen un modelo que permite obtener mejor precio-rendimiento en relación con otros proveedores.

- Tipos de máquinas y contenedores: permite configurar la combinación correcta de memoria y CPU virtual para una carga de trabajo determinada, evitando así la necesidad de sobreaprovisionamiento y reduciendo los costos.

- Almacenamiento secundario: Google *Cloud Storage Nearline* ofrece disponibilidad de almacenamiento de datos de resguardo a bajo costo e interfaces de alto rendimiento para la recuperación rápida de los mismos (es utilizado por muchos usuarios como único nivel de almacenamiento).

- Balanceadores de carga globales (posibilidad de escalar a 1 millón de usuarios en segundos) y tiempos de arranque más rápidos (en el rango de 40-50 segundos), lo cual permite aumentar la capacidad de servicio en respuesta al tráfico entrante.

- Herramientas para los usuarios de *big data* basadas en los mismos servicios que se utilizan en Google. BigQuery, Cloud Datalab y CloudDataproc permiten cambiar la forma en que se analizan y se utilizan los datos aportando rendimiento y reduciendo el tiempo de obtención de resultados de horas a segundos.

- Incorporación del Google Cloud Machine Learning como servicio (ahora en versión beta) que ofrece a los usuarios acceso a los sistemas de aprendizaje profundo ofrecido por servicios como Google Translate, Google Fotos, búsqueda por voz en Google y respuestas inteligentes en Gmail.

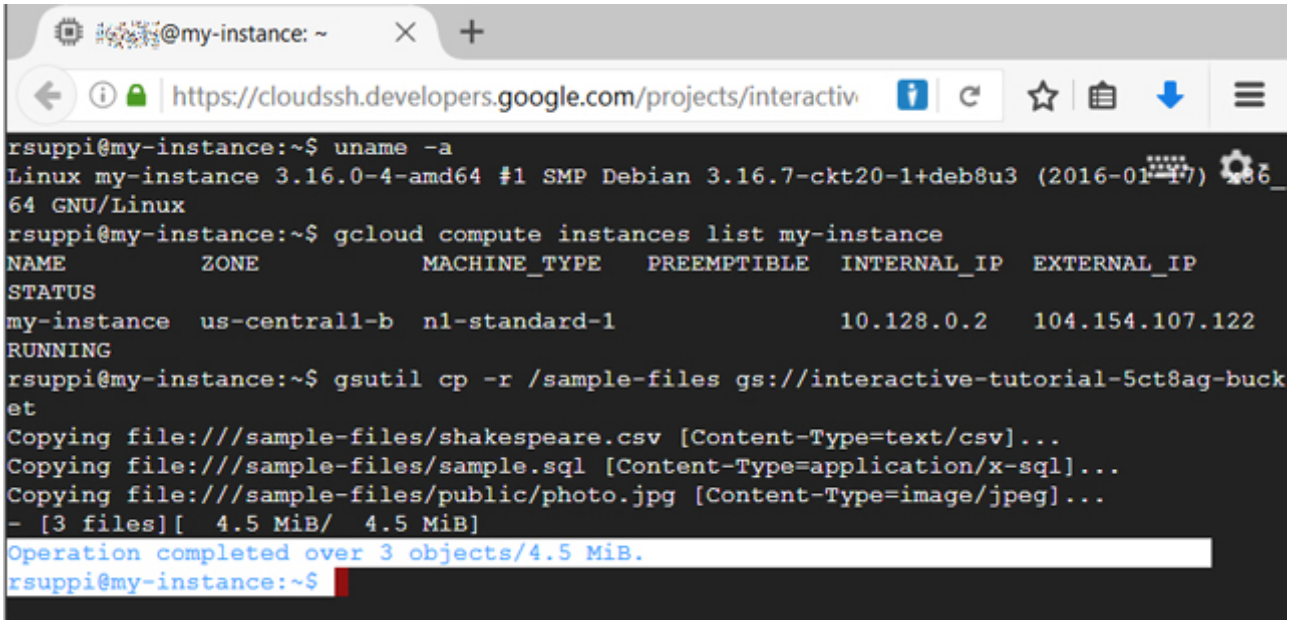
Google dispone de grandes clientes tanto para su plataforma IaaS como para PaaS, además de él mismo para sus SaaS (Gmail, YouTube, Gmaps...), entre los que se pueden enumerar: Spotify, Airbus, Coca Cola, Philips, Evernote, Niantic, Bestbuy, Motorola, Ocado, Fis, HTC, Heathrow, Kan Academy, PocketGems, Zulily, BestBuy, entre otros.

1.3.1. Registro, configuración y puesta en marcha

Google ofrece una cuenta gratuita con 330$de crédito durante sesenta días para probar todos los recursos de la plataforma GCP y el período finaliza cuando se alcanza uno de los dos límites. Durante la suscripción gratuita, se puede utilizar cualquiera de los servicios de Cloud Platform para aprovisionar servidores, implementar aplicaciones administradas, almacenar archivos, consultar petabytes de datos, entre otras, aunque existen unas ciertas limitaciones en algunas opciones (por ejemplo, MV = ocho núcleos concurrentes). Para el acceso se necesitará una cuenta de Google y una tarjeta de crédito (aunque

hay un mecanismo alternativo si no se dispone de una tarjeta y es a través de una cuenta bancaria). Esta tarjeta solo es con fines de identificación y Google se compromete a no cargar ningún costo hasta que el usuario pase a la versión de pago.

La documentación de la plataforma es muy detallada y dispone de un tutorial interactivo de los principales servicios que a través de una MV y una línea de comandos guía al usuario para lanzar un servidor Apache en una MV ejecutándose en Google Compute Engine, almacenar un conjunto de archivos haciendo unos públicos y otros privados en Google Cloud Storage, crear una BD en Google Cloud SQL (equivalente a MySQL) y cargar datos en Google BigQuery y ejecutar un búsqueda (*query*) en segundos. En la figura se puede ver la MV (a través de una ventana del navegador), la IP utilizando el comando `gcloud compute instances list my-instance` (para luego poder acceder al servidor Apache) y la copia de archivos.

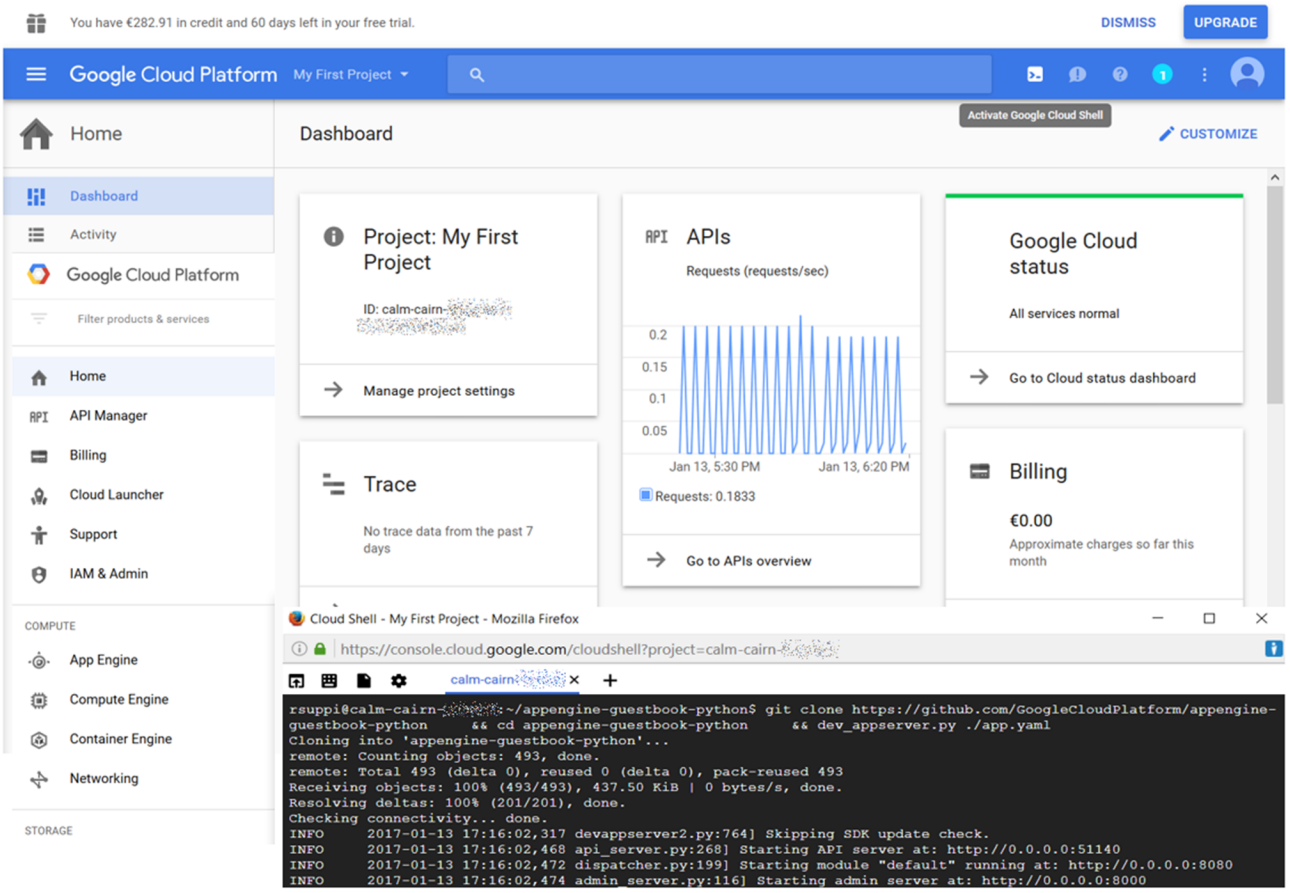


En este *Getting Started*, se sugiere continuar con diferentes secciones/orientaciones como pueden ser *Mobile* (5 servicios para conocer cómo desarrollar *apps* para móviles), *Web Apps & Websites* (4 servicios para construir website), *Big Data* (4 servicios orientados a procesar y analizar datos masivos) o *Cloud Launcher* (desplegar uno de los 150+ *stacks* que dispone GCP). También es interesante conocer los diferentes tipos de almacenamiento de que dispone GCP o seguir los tutoriales sobre diferentes aspectos y escenarios en los cuales GCP puede aportar valor y conocer las herramientas para estimar el costo de utilización de la GCP como pueden ser *Pricing calculator* y *Cost-comparison calculator* (en relación con AWS) que complementan el apartado facturación (*billing*), donde se dispondrá de toda la información detallada de gastos y detalles de los mismos.

El *dashboard* de GPC muestra un entorno de información (centro), una barra de acceso a los servicios (a la izquierda) y la configuración de la cuenta, usuario, el Google *Cloud Shell* (un entorno *shell* para gestionar los recursos desplegados en GCP) en la parte superior derecha (y también el crédito disponible) y el menú de proyectos (con uno creado inicialmente para la suscripción pero que el usuario puede crear más desde dicho menú). El Google *Cloud Shell* es una MV que se aprovisiona en el momento y hace de interface CLI para el comando *gcloud* y donde se pueden ejecutar acciones complejas como, por ejemplo, aprovisionar una aplicación (*guestbook*) desde *Github* y ejecutarla en un único comando (y de la cual se puede visualizar el resultado desde el primer botón de la ventana: *WebPreview*):

```
git clone https://github.com/GoogleCloudPlatform/appengine-guestbook-python \
  && cd appengine-guestbook-python \
  && dev_appserver.py ./app.yaml
```

La imagen siguiente muestra el *dashboard*, la barra de menús (que se visualiza en el icono  y la ventana de GC Shell con el despliegue de la aplicación.



1.3.2. Creación de una MV Linux

Google Compute Engine permite desplegar MV en los CPD de Google soportando conexión de altas prestaciones (fibra), de alta disponibilidad y con escalado desde una MV a un entorno de *cloudcomputing* global y con balanceo de carga. Las MV se inician rápidamente desde un esquema seleccionado por el usuario que indica el tipo de recursos y prestaciones de que dispondrá la máquina, pero también es posible que el usuario pueda definir una máquina propia adaptada a sus necesidades.

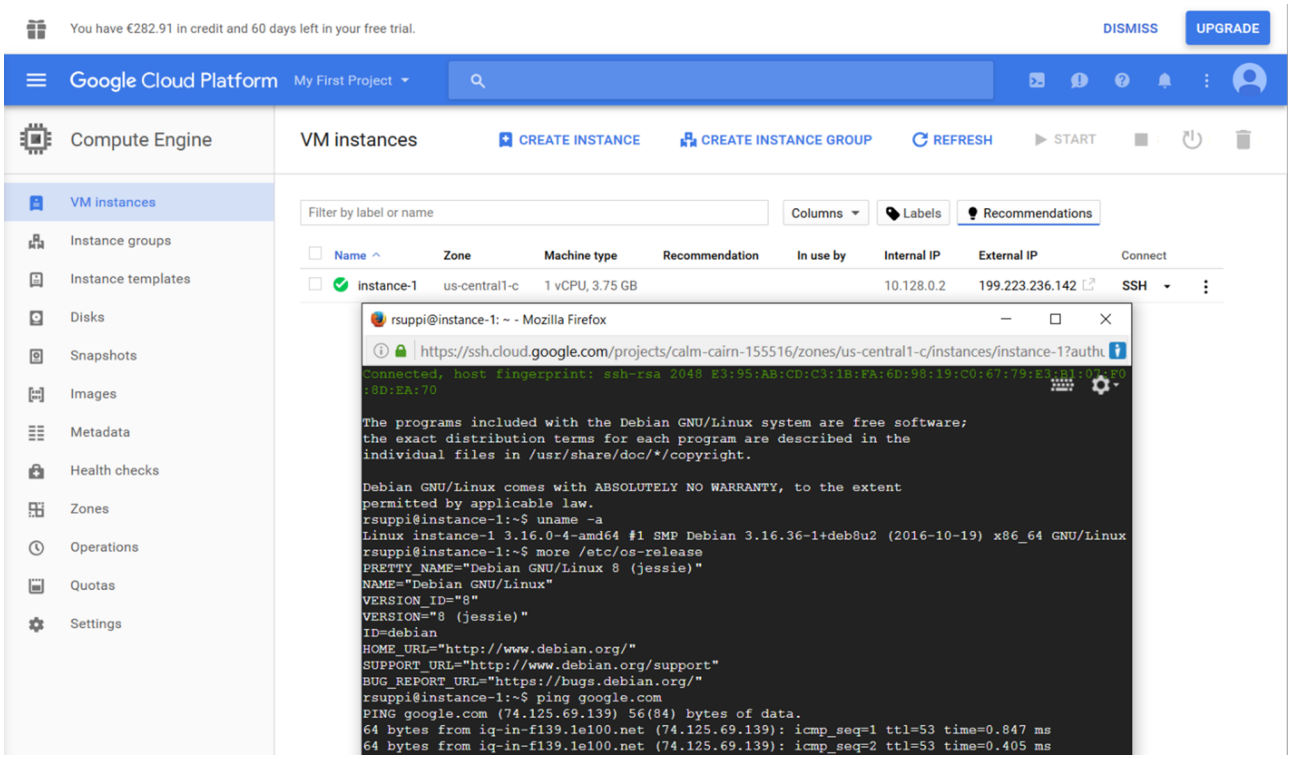
Es posible utilizar dispositivos locales de altas prestaciones (SSD) para mejorar el IOPS de la MV, redes privadas para interconectarlas e incluso mapear estas sobre dispositivos de fibra locales para mejorar el ancho de banda.

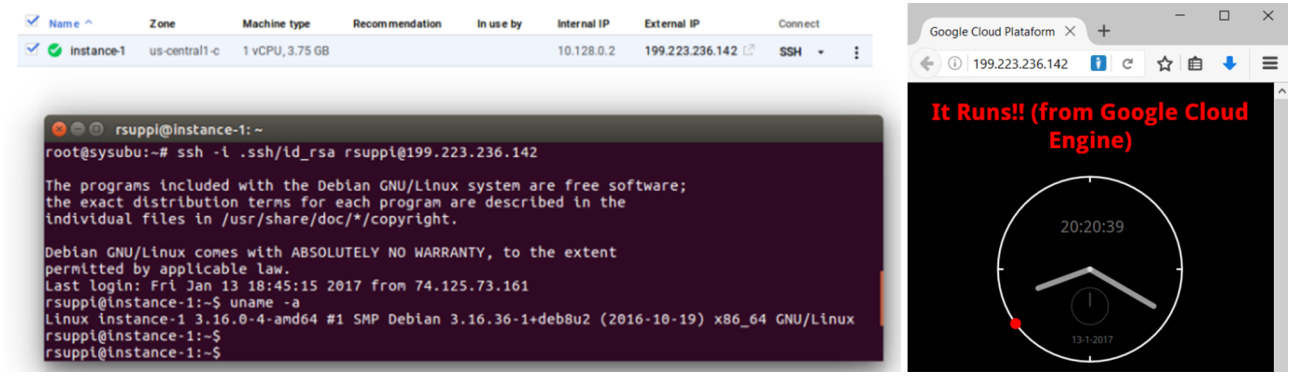
La facturación se realiza por minutos (a partir de un mínimo de 10'), pero se pueden aplicar descuentos por uso sostenido o ejecuciones de altas cargas de trabajo a largo plazo.

Para crear una MV Linux, desde el *Dashboard* → *Compute Engine* → *VM Instances* → *Create Instance* (se puede ajustar diversos valores como el SO, la red, los puertos, los discos, etc.). Después de creada, hay que acceder a la máquina a través del icono de SSH que permitirá abrir una sesión sobre un navegador, o también el comando `gcloud` para conectarse a ella, por ejemplo desde GC Shell):

```
gcloud compute --project "abcd-abcde-abcdef" ssh --zone "us-central1-c" "instance-1"
```

También es posible conectarse con otro cliente SSH pero para ello hay que generar el par de llaves pública y privada e introducir la clave pública en la sección *Meta datata* de *Compute Engine*. La imagen a continuación muestra la MV y su conexión a través de SSH en el navegador y a través de llave pública desde un Ubuntu. Finalmente, en la máquina se ha instalado un servidor Apache y a través de la interfaz SSH Web se han subido unos archivos de pruebas (reloj SVG + javascript) y se ha visualizado la página a través del servicio que se está ejecutando en la MV (cuando se crea la máquina hay que activar el puerto http/https y, en caso contrario, ir a *Dashboard* → *Network* → *Firewall* y activar el puerto para la interfaz que está utilizando la MV).





Para generar las llaves pública-privada desde Linux (máquina desde la cual se quiere acceder a la instancia):

```
ssh-keygen -t rsa -f ~/.ssh/gokey -C {USERNAME}
chmod 400 ~/.ssh/gokey
cat ~/.ssh/gokey.pub
```

Donde {USERNAME} es el nombre de usuario que se desea tener para acceder a la instancia de la MV. Después, hay que cambiarle las protecciones a la llave privada y visualizarla (utilizar el comando `cat` ya que otros pueden introducir caracteres adicionales), copiarla e ir al menú *Compute Engine* del proyecto donde está desplegada (o donde se desplegará la MV) → *Metadata* → *SSH Keys* → *Edit* y pegar la llave pública copiada anteriormente (verificar que tenga un formato como `ssh-rsa {KEY_VALUE} {USERNAME}`) → *Save*. Si la máquina

está en ejecución se deberá reiniciar y luego ya se podrá conectar a la MV por SSH a través de la IP pública indicada en la consola. Para otros SO el procedimiento es similar.

La CLI de GCP es muy potente y para crear la máquina desde, por ejemplo, GC Shell una MV con 2 *cores* y 3GB RAM:

```
gcloud compute instances create my-vm --custom-cpu 2 --custom-memory 3
```

También, GPC dispone de un tipo de MV llamado *preemptible* que Google puede finalizar su ejecución en caso de que los necesite para otras tareas, pero por contrapartida son mucho más baratas (son muy útiles si las aplicaciones que se están ejecutando son tolerantes a fallos, por ejemplo, trabajo de procesamiento *batch* ya que el costo se reduce significativamente y bajo esta premisa no están cubierta por la SLA). Para crear una máquina preemptible en la CLI:

```
gcloud compute instances create my-vm --zone us-central1-b --preemptible
```

1.3.3. Gestión del almacenamiento

Existen en GCP diferentes opciones de almacenamiento como, por ejemplo, Persistent Disk (*block storage* para las MV o contenedores), Cloud Storage (*blob store*, para archivos o datos no estructurados), *Bigtable* (BD NoSQL), Cloud Datastore (BD de documentos NoSQL), Cloud SQL (BD MySQL), BigQuery (*Enterprise Data Warehouse*, EDW, con SQL), Drive (espacio colaborativo para almacenamiento, compartición y edición de archivos), Firebase Storage (accesos a Cloud Storage desde móviles y sin servidor), Firebase Realtime Database (BD en tiempo real NoSQL JSON para aplicaciones móviles), Firebase Hosting (contenido web para móviles).

En este ejemplo nos centraremos sobre uno de los más utilizados, que es Cloud Storage, y que puede ser configurado en diferentes modalidades:

1) Multi-regional: almacenamiento redundante de alta disponibilidad distribuido entre regiones geográficas.

2) Regional: lo mismo que el anterior, pero ubicado en una región para incrementar las prestaciones.

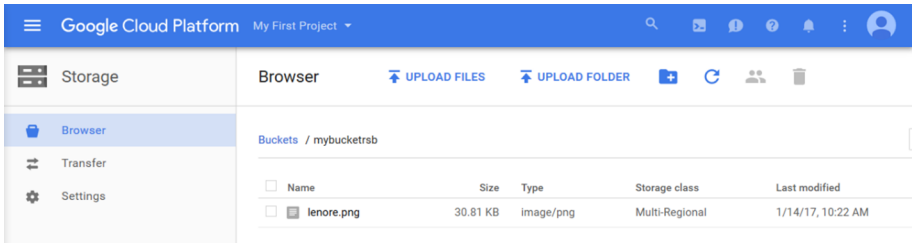
3) *Nearline*: almacenamiento de bajo costo para almacenar datos pocos utilizados.

4) *Coldline*: almacenamiento de muy bajo costo para archivado o *backup*.

Entre las principales características a destacar, está la redundancia que permite garantizar un 99,999999999% de fiabilidad, disponibilidad (Multi-regional permite un 99,95 % y Regional ofrece 99,9%, *Nearline* y *Coldline* un 99%), escalabilidad, consistencia (GET global sobre la última escritura).

Para crear un nuevo repositorio desde el menú principal → *Storage* → *Create Bucket* → seleccionar el tipo deseado –comparar los precios–, e indicar el nombre.

Luego, desde la interfaz ya se puede gestionar sobre este *bucket* los archivos, directorios, etc. La imagen a continuación muestra el entorno de GCS y con un archivo (*lenore.png*) dentro del *bucket*.



Es posible montar un *bucket* dentro de una máquina virtual a través de FUSE (*Filesystem in Userspace*). Cloud Storage FUSE es un adaptador *opensource* que permite montar *buckets* desde Cloud Storage como sistema de archivos de Linux u OSX. Hay diferentes formas de hacerlo, pero la más simple es creando un *bucket* y asociando un servicio a él.

Para ello ir a *IAM & Admin* → *Service Account* → *Create Service Account* → e indicar un nombre y un *Rol* → *Storage* → *Admin*.

Luego sobre este servicio generar la llave privada (a la derecha del servicio en el menú de tres puntos verticales) y guardarla (*key.json*).

Crear una instancia de una máquina virtual y en *Identity & API Access* seleccionar el servicio creado.

Conectarse a la MV por SSH e instalar *gsfuse*:

```
export GCSFUSE_REPO=gcsfuse-'lsb_release -c -s'
echo "deb http://packages.cloud.google.com/apt $GCSFUSE_REPO main" | sudo tee \
 /etc/apt/sources.list.d/gcsfuse.list
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo apt-get update
sudo apt-get install gcsfuse
```

Inicializar las credenciales subiendo a la máquina virtual el archivo guardado previamente (key.json) y haciendo:

```
export GOOGLE_APPLICATION_CREDENTIALS=/directorio_donde se encuentre/key.json
gcloud beta auth application-default login
```

Lo cual no indicará una URL y quedará a la espera de un código de verificación. Con la URL se debe copiarla en un navegador, seleccionar la cuenta, los permisos y nos dará el código que habrá que copiarlo e insertarlo en el comando anterior de la MV. Luego hacer:

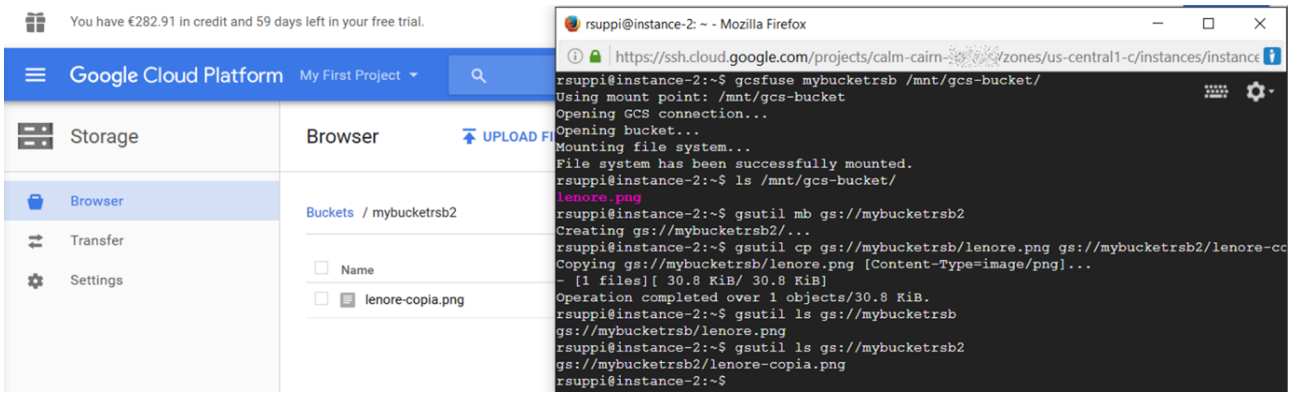
```
unset GOOGLE_APPLICATION_CREDENTIALS
mkdir -p /mnt/gcs-bucket
gcsfuse mybucketrsb /mnt/gcs-bucket/
```

También se podría utilizar el comando `gsutil` para crear *bucket*, copiar, etc.:

```
gsutil mb gs://mybucketrsb2
gsutil cp gs://mybucketrsb/lenore.png gs://mybucketrsb2/lenore-copia.pngCopying \
 gs://mybucketrsb/lenore.png
```

La imagen siguiente muestra la interfaz de GCS con el nuevo *bucket* creado y la interfaz SSH de la MV desde donde se ha montado el directorio *mybucketrsb* y el acceso al contenido o las instrucciones para crear otro *bucket* y copiar un archivo desde el anterior.

Información adicional sobre configuración de las credenciales, creación y habilitación de *Service Accounts* para las instancias, utilización de `gsutil`, o conexión de *Storage Buckets* a las instancias.



Con los permisos adecuados podríamos acceder al archivo como *https://storage.cloud.google.com/<bucket>/<object>* en nuestro caso *https://storage.cloud.google.com/mybucketrsb/lenore.png*

Para modificar los permisos:

```
gsutil acl ch -r -u AllUsers:READ gs://mybucketrsb/lenore.png
```

1.3.4. Crear un servicio de Drupal utilizando Cloud Launcher (LAMP)

En este ejemplo se instalará Drupal ejecutándose desde GCE, lo cual se puede hacer de diferentes formas. La más simple es con la opción de Cloud Launcher → *Drupal* y desplegarla. Cloud Launcher ofrece 100+ *stacks* configurados y preparados para acelerar la implantación y el despliegue de entornos para reducir el tiempo y sobre todo sin necesidad de tener grandes conocimientos sobre el servicio/configuración/requerimientos del mismo. En el otro extremo sería desplegar una máquina virtual e instalar todos los requerimientos previos (Mysql, Apache, PHP) que necesita Drupal y luego seguir los pasos de Instalación indicados en *Installing Drupal 8*.

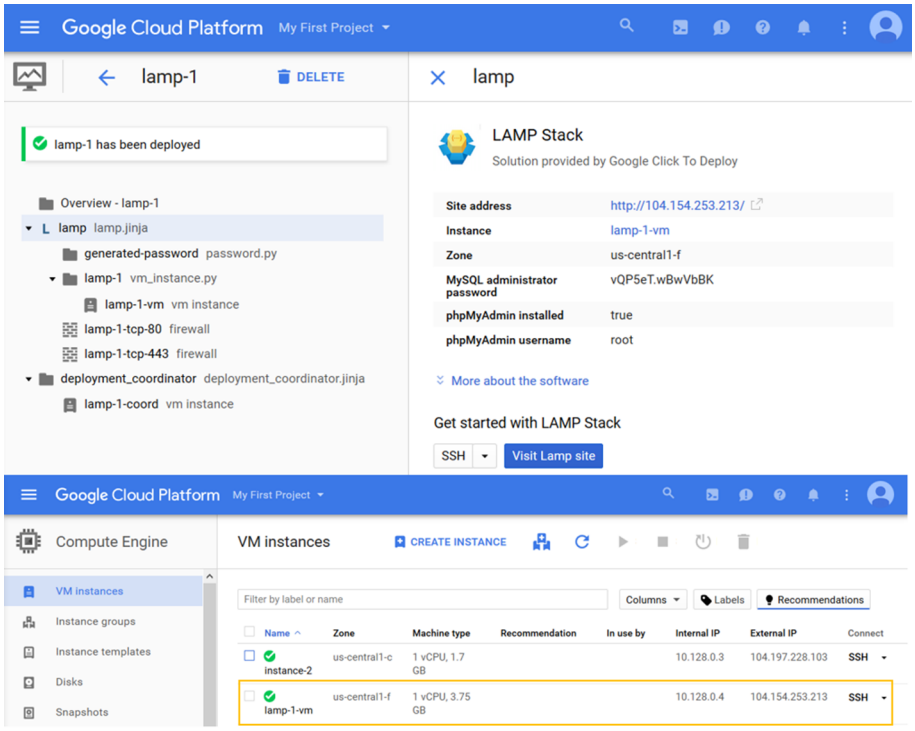
En este caso utilizaremos un paso intermedio para conocer las diferentes opciones de GCP que es instalando un *stack* LAMP (Linux + Apache + MySQL + PHP) desde *CloudLauncher* y luego sobre esta máquina virtual instalar y configurar Drupal. Para ello desde *Cloud Launcher → LAMP Stack → Launch on CE →* revisar las opciones → *Deploy*. En este caso se ha utilizado *Lamp Stack* ya que la versión del LAMP de Bitnami utiliza una estructura de archivos diferentes de las habituales con base en el */opt* que puede generar ciertas dificultades a los usuarios que no tienen experiencia en estos *stacks* (no obstante, cabe tener en cuenta los precios ya que *Lamp Stack* cuesta 25,95$/mes –incluido descuento de duración continuada y utiliza una instancia n1-standard-1–, mientras que LAMP Bitnami 4,49 $/mes utilizando una f1-micro).

Una vez desplegada la instancia veremos el panel del *Deployment Manager* y el de *Compute Engine → VM Instances* en ejecución.

El panel del DM es importante ya que tendremos la IP de la instancia (también desde GCE) y el *passwd* de *Mysql* y otras instrucciones (ver imagen siguiente). Luego de ello, hay que acceder a la instancia por SSH desde el panel de despliegue o desde GCE para instalar y configurar Drupal.

La instalación se puede hacer mediante Drush que es un CLI e interface para Drupal, pero en nuestro lo haremos de la forma manual ya que para lo que se desea es más simple (ver documentación del despliegue con Drush). Para ello es necesario bajar (en este caso la versión 8.2.5) y mover la versión de Drupal *a /var/www/html*

```
wget https://ftp.drupal.org/files/projects/drupal-8.2.5.tar.gz
tar xzvf drupal-8.2.5.tar.gz
sudo mv drupal-7.8.2.5 /var/www/html/drupal
sudo chown -R www-data /var/html/drupal/sites
```



Para crear la base de datos la forma más fácil es acceder desde el DM al `phpMyAdmin` (o desde la CLI) que tiene instalado y con el usuario *root* y el *passwd* indicado crear una base de datos. Luego conectarse a la URL *http://IP_MV/drupal* y configurar el Drupal con la base de datos creada y toda la información que solicita la instalación. Luego de unos instantes ya dispondremos del Drupal en desplegado y se podrán hacer las adaptaciones necesarias al nuevo servicio y crear la primera página.

Es importante luego ir al apartado de *Configuration* del Drupal y acabar de ajustar las opciones de seguridad como *Trusted Host Settings*, *Update notifications* y otros parámetros para terminar la configuración del sitio.

Google Compute Engine no permite conexiones de salida sobre los puertos 25, 465, and 587 por lo cual no es posible enviar mails desde la MV y es necesario utilizar un tercer servicio que haga de *Relay Host*. En esta prueba se utilizará SendGrid donde se deberá crear una cuenta para utilizarlo como *Relay*

Host (nos permitirá enviar 400k correos/30 días). Luego desde Drupal obtener la dirección del módulo (extensión tgz) de autentificación de SMTP que en nuestro caso será el de la versión 8.x. Antes de realizar la instalación verificar que los directorios *drupal/modules* y *drupal/core* pertenecen a *www-data* (sino, no se podrá instalar el módulo), sino pertenecen a ejecutar:

```
sudo chown www-data /var/www/html/drupal/core
sudo chown www-data /var/www/html/drupal/modules
```

Luego, dentro de Drupal 8 como administrador, ir a *Extend* y habilitar el módulo de *Update Manager* que cuando esté habilitado permitirá ver en *Extend* → *Install new Module* → introducir la URL del módulo → *Install*. En *Extend* ir al módulo instalado (*SMTP Authentication Support*) → *Configure* e introducir los siguientes datos: Activación del módulo (*On*), *SMTP server: smtp.sendgrid.net*, *SMTP port: 2525*, usuario/*passwd* de SendGrid creados anteriormente, la dirección desde donde se desea que vengan los mails. Con ello ya se puede probar de enviar un mail a través de *Contact* y verificar el funcionamiento.

Otros aspectos interesantes a analizar es la monitorización que nos muestra GCP sobre lo que está ocurriendo en la plataforma: reportes de utilización que dejará en GCS un reporte diario/mensual según se haya configurado, los reportes de actividad desde el menú vertical → *Logging* que mostrará los eventos configurados en el sistema y dentro de cada máquina (haciendo un clic en cada instancia) mostrará diferentes estadísticas sobre la utilización de los recursos e información relacionada sobre la instancia.

Como se ha comentado en las anteriores plataformas, GCP tiene una gran cantidad de herramientas y funcionalidades que el *SysAdmin* deberá analizar y experimentar con ellas, así como una gran cantidad de tutoriales sobre las cuales obtener información y casos de uso [Gdo].

1.4. CloudSigma

CloudSigma es una empresa (~50 empleados) fundada en 2009 en Suiza que ofrece *cloud servers* y *cloud hosting* donde el usuario define las propiedades y el entorno a través de una interfaz web.

Dispone de 9 localizaciones (Estados Unidos, APAC y Europa) con una clasificación de Tier III o equivalente para cada centro de datos (además de estar certificada con la ISO 27001). Entre su catálogo ofrece servidores *cloud* flexibles y alojamiento de VPS (*virtual private server*) en cualquiera de sus localizaciones que permiten ejecutar SO (sin modificar) tales como Linux, Windows,

FreeBSD y CoreOS, pudiendo el usuario definir el tamaño de servidor, almacenamiento y las redes desde una GUI simple y clara en muy poco tiempo y a unos precios aceptables.

En cuanto a calidad de los recursos, el proveedor ofrece dispositivos SSD (de hasta 5TB), HDD (de hasta 100TB), redes de 10GigE con aprovisionamiento instantáneo y con IP públicas/VLAN, hasta 40 núcleos de CPU y 128 GB de RAM por servidor y centros de datos independientes y se aplica la legislación de sitio de localización.

En cuanto a la capa de virtualización, utilizan KVM ofreciendo una asignación de recursos dedicada y segura para la MV del cliente.

Esta plataforma permite integrarse con otras plataformas, entre las cuales se pueden mencionar:

OpenStack: permite gestionar cargas en CloudSigma mediante una infraestructura OpenStack propia, es decir, creación de un *cloud* híbrido de una forma simple y fácil.

Ubuntu: permite la configuración durante el arranque de los servidores Ubuntu inyectando claves SSH, actualizando o instalando paquetes entre otras opciones.

Permite dar soporte a **Docker** y configurar *CoreOS* a través de la propia API o sobre una consola web directamente.

Apache Jclouds: permite gestionar CloudSigma desde otros *cloud* (privados y públicos) con este controlador java (utilizado por gran cantidad de proveedores).

Apache Libcloud: controlador que permite una funcionalidad similar al anterior, pero en Python.

Fog.io: Librería en Ruby que permite gestionar CloudSigma.

Ansible: permite usar Dynamic Inventory Sources con los servidores CloudSigma.

Abiquo Hybrid Cloud: gestión de la infraestructura pública de CloudSigma con la infraestructura privada.

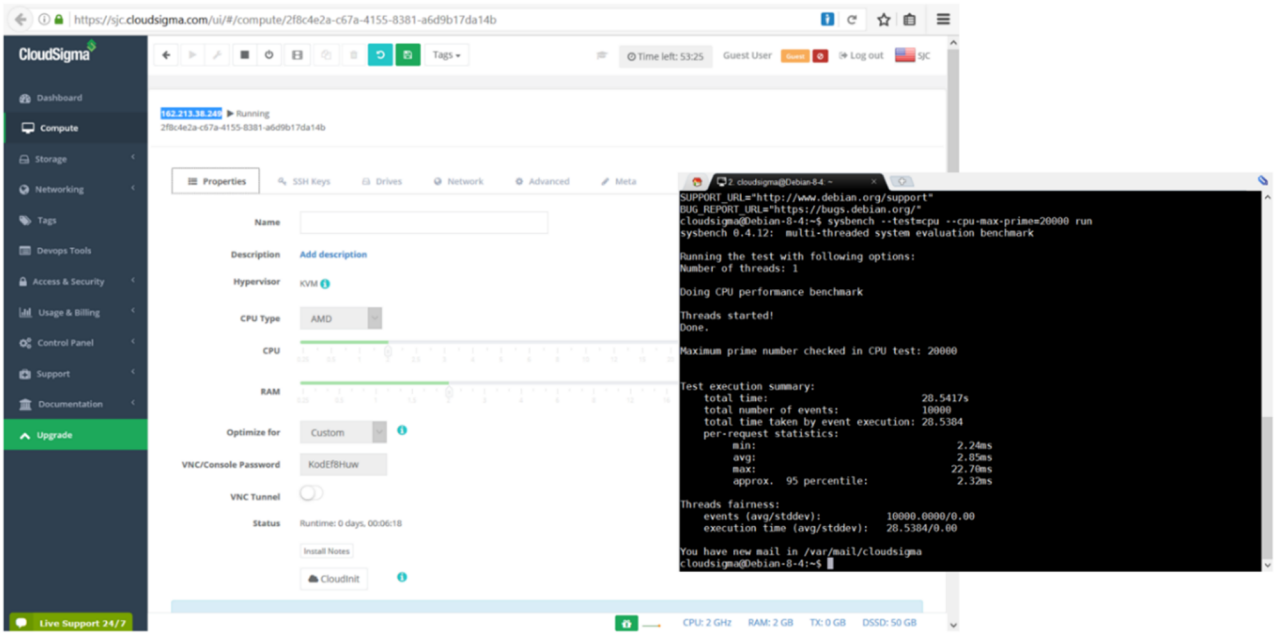
Pycloudsigma Library: CloudSigma API en Python para gestionar el *cloud* definido en CloudSigma.

Como puntos destacados de su SLA se pueden mencionar: punto de referencia de *uptime* 100%, ×50 créditos de tiempo contra cualquier tiempo de inactividad superior a 15 minutos y 1 ms de latencia máxima interna. Es interesante consultar las condiciones completas que varían para Suiza y EE.UU.

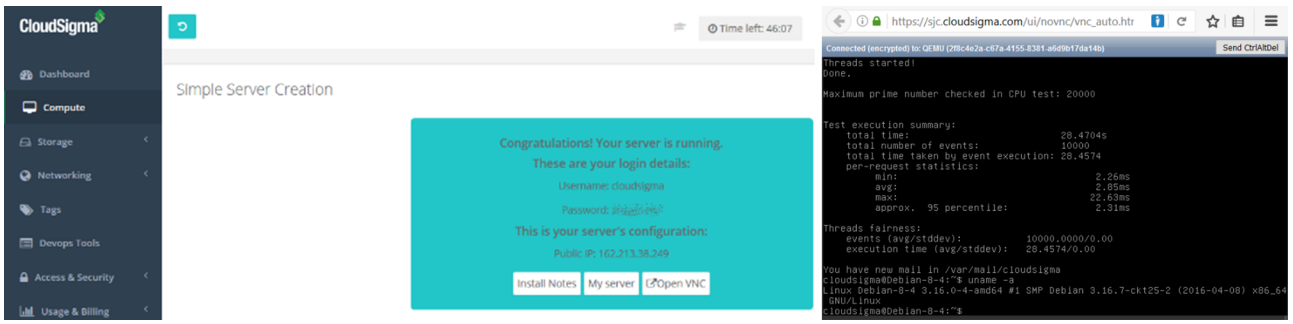
Como prueba funcional se puede acceder por una hora (de forma gratuita y sin tarjeta de crédito) a los servicios ofrecidos por el proveedor o entrar con GitHub a través de la URL *https://fra.cloudsigma.com/ui/#/* donde en este caso es el CPD en Frankfurt, pero se puede cambiar desde la interfaz. Cuando se está a punto de vencer el crédito de una hora, se puede registrar y enviarán a través de correo la activación de la cuenta por siete días y también es posible introducir un TE donde se obtendrá un código y así obtener más recursos de prueba (de acuerdo a la información mostrada, el registro manual está deshabilitado temporalmente dado el alto número de peticiones que tienen).

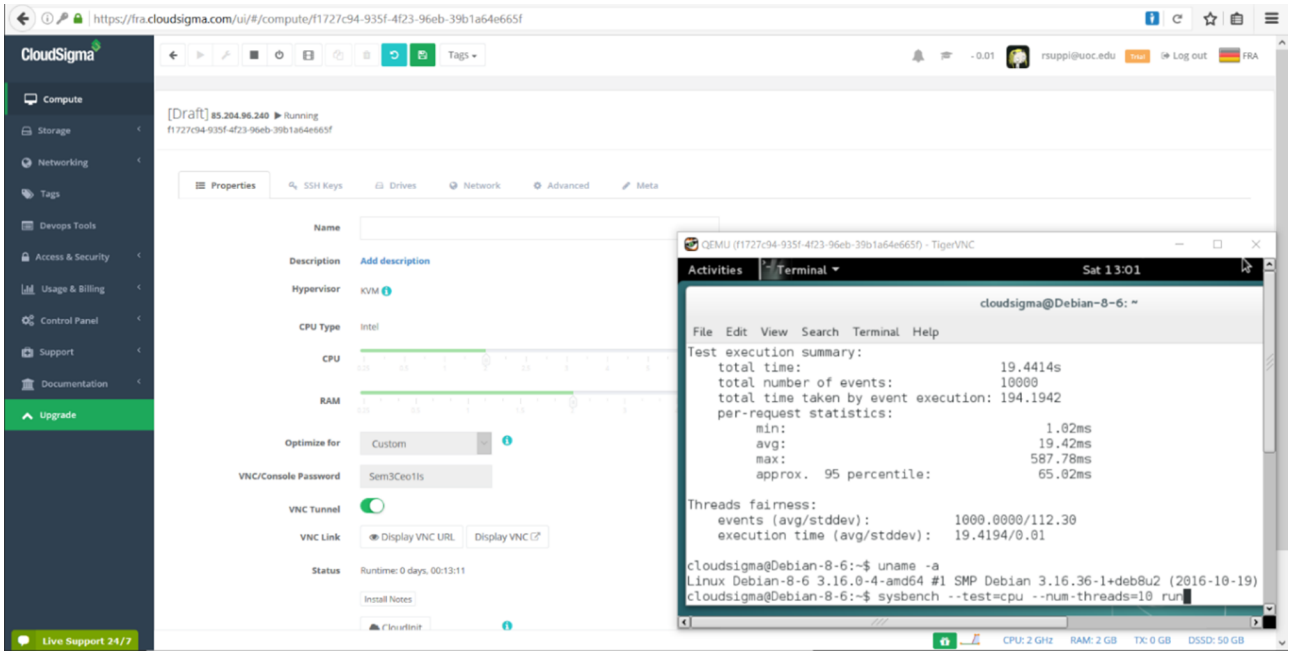
Una vez dentro, el usuario dispone de una interfaz como la mostrada en la figura siguiente donde puede acceder a las principales opciones para gestionar los servicios, entre los más importantes: *Compute*, *Storage*, *Networking* para gestionar y definir servidores, almacenamiento y redes respectivamente, *Devops Tools* para integrar los controladores antes mencionados, *Access & Security* donde se definen las llaves SSH y políticas de seguridad, *Usage & Billing* donde se tiene control del gastos de recursos y el costo/facturación que se ha realizado.

En este primer caso se ha generado un servidor Debian e instalado (utilizando `sudo`) el programa `sysbench` (antes de hacer un `apt-get update`) para ejecutar un *benchmark* de CPU cuyo resultado se muestra en una conexión remota a través de `ssh` (en la parte inferior se pueden ver los detalles de la máquina):



Cuando se despliega la máquina se provee toda la información necesaria para acceder a este servidor y es posible conectarse a través de noVNC (se abre en un navegador) o a través de VNC (con la URL y *passwd* provistos, como se muestra en las imágenes siguientes (en la última figura la provisión ha sido un Debian 8.6 Desktop, como se puede observar en la interfaz gráfica):





Como conclusión, este proveedor presenta un entorno amigable con recursos a disposición del usuario y mediante un método rápido y simple; se pueden desplegar los servidores y probar su potencialidad sin mayores complicaciones y dispone de diversos controladores para crear/integrar fácilmente en un *cloud* híbrido (ver tutoriales e información relacionada).

Es necesario tener en cuenta que, en las suscripciones gratuitas, se pueden borrar los servidores para probar otros, pero también es necesario borrar el espacio de almacenamiento ya que si no da un error de «crédito superado», pero no indica qué es porque el disco del servidor previo todavía está asignado.

1.5. Máquinas virtuales/contenedores en línea (entornos basados en IDE)

Con la potencialidad del *cloud* y las necesidades de reducir el tiempo de desarrollo a la vez que ofrecer servicios a desarrolladores sobre diferentes espacios de trabajo (*workspaces*) y entornos de desarrollo (*stacks*), han surgido una serie de servicios, denominados *cloud IDE* (*integrated development environment*), que en muchos casos permiten crear entornos pseudo-IaaS para que el desarrollador incluya el software que desee y realice las acciones pertinentes sobre una instancia de una máquina virtual con SO o contenedor sobre un SO determinado.

Son entornos orientados al IDE y por ello están limitados, ya que no necesariamente se pueden escoger qué tipo de recursos (virtualizado) dispondrá esa máquina ni crear infraestructuras interconectadas o con redes internas (aun-

que en algunas plataformas es posible), pero sí que muestran la potencialidad de la virtualización y el despliegue inmediato de MV y contenedores sobre el *cloud* a nivel de infraestructura.

Para mostrar algunos casos de uso, y como la mayoría está orientada a IDE, es conveniente crearse una cuenta en Github ya que generalmente se puede acceder por vinculación con este entorno de desarrollo.

Asimismo, se han descartado entornos que si bien pueden ser muy potentes, requieren métodos adicionales de control (por ejemplo, una tarjeta de crédito, por ejemplo, Cloud9) o aquellos en los que es necesario una suscripción de pago (por ejemplo, Koding).

1.5.1. Codeanywhere

Es una multiplataforma IDE (2013) en *cloud* que permite a los usuarios escribir, editar, colaborar y ejecutar proyectos software en 120 lenguajes sobre un navegador o dispositivo móvil. La plataforma está escrita íntegramente en JavaScript y utiliza OpenVZ para sus entornos de desarrollo (llamados *DevBoxes*) donde el usuario podrá ejecutar su código o conectase a sus MV por SSH (o FTP) y también vincularlo a Dropbox o Google Drive, o simplemente subir los archivos desde el ordenador local con *Drag&Drop* (archivos y directorios).

El usuario dispone además de un terminal con los comandos habituales en Linux (y `npm`, que es el gestor de paquetes para *JavaScript runtime environment Node.js*), editor avanzado, depurador, gestor de archivos, y soporta integración con repositorios Git, Github o Bitbucket e integración con Heroku. Cuenta con una comunidad muy activa de más de 700k desarrolladores (algunos de compañías como Accenture, CNN, SalesForce, Reuters) y diferentes planes de suscripción, pero en este caso se utilizará la suscripción gratuita (que según la información mostrada es de una cuenta perpetua).

Este tipo de plan (*free*) es más que aceptable ya que tiene todas las herramientas al igual que las otras suscripciones, pero no incluye soporte/doble método de autentificación, solo permite una revisión (versión del archivo), las conexiones remotas están limitadas a una y a un contendor con 256 MB de RAM, 2 GB de disco (y este se apagará cuando el usuario se desconecte) [Caw].

El procedimiento para acceder es a través de la URL de *https://codeanywhere.com/* y en esta prueba se accederá a través Github (previamente se habrá tenido que crear una cuenta y, en el momento de acceder a Codeanywhere, se deberá autorizar desde Github que pueda vincularse).

Desde el *Dashboard* se podrán crear los proyectos y gestionar todos los aspectos vinculados al entorno, cuentas y opciones y en el apartado de proyectos se podrán crear nuevos (limitado a uno en la suscripción gratuita).

En el presente caso de uso, se ha creado un entorno CentOS 6.5 (con las especificaciones antes indicadas), que tiene acceso `sudo` y `yum`, acceso por SSH (en el presente caso *host12.codeanyhost.com:53887*), acceso a HTTP y *Websocket ports*, y también acceso a las aplicaciones ejecutándose en el contenedor (por ejemplo, un servidor web) ya que los puertos 1024-9999 están disponibles, incluido HTTPS a través del puerto 3000. Para crear una máquina simplemente en el símbolo del usuario → *Account Dashboard* o en la URL *https://codeanywhere.com/dashboard*, seleccionar *Projects* → *Create New Project* y en el *Connection Wizard* seleccionar el tipo de entorno deseado (en este caso se ha seleccionado un Centos6.5) y se creará el nuevo entorno (*Workspace*) con el nombre indicado (que se verá a la izquierda y con el botón derecho todas las opciones para gestionarlo).

Sobre el entorno se ha abierto una Terminal SSH (*Workspace* → botón derecho → *SSH Terminal*). Se ha instalado Apache y se ha creado una página web:

```
sudo yum install httpd
sudo vi /var/www/html/index.html
sudo service httpd restart
```

Luego se ha accedido por la URL (o directamente haciendo clic en *Run* o con el botón derecho sobre el *Workspace* → *Info* y copiando la URL). Las figuras siguientes muestran el entorno y el acceso a la URL indicada.

Para conectarse remotamente se deberán crear un par de claves (pública y privada) y copiar la pública en el espacio de trabajo; para ello, en un terminal hacer:

```
ssh-keygen
cat $HOME/.ssh/id_rsa.pub
```

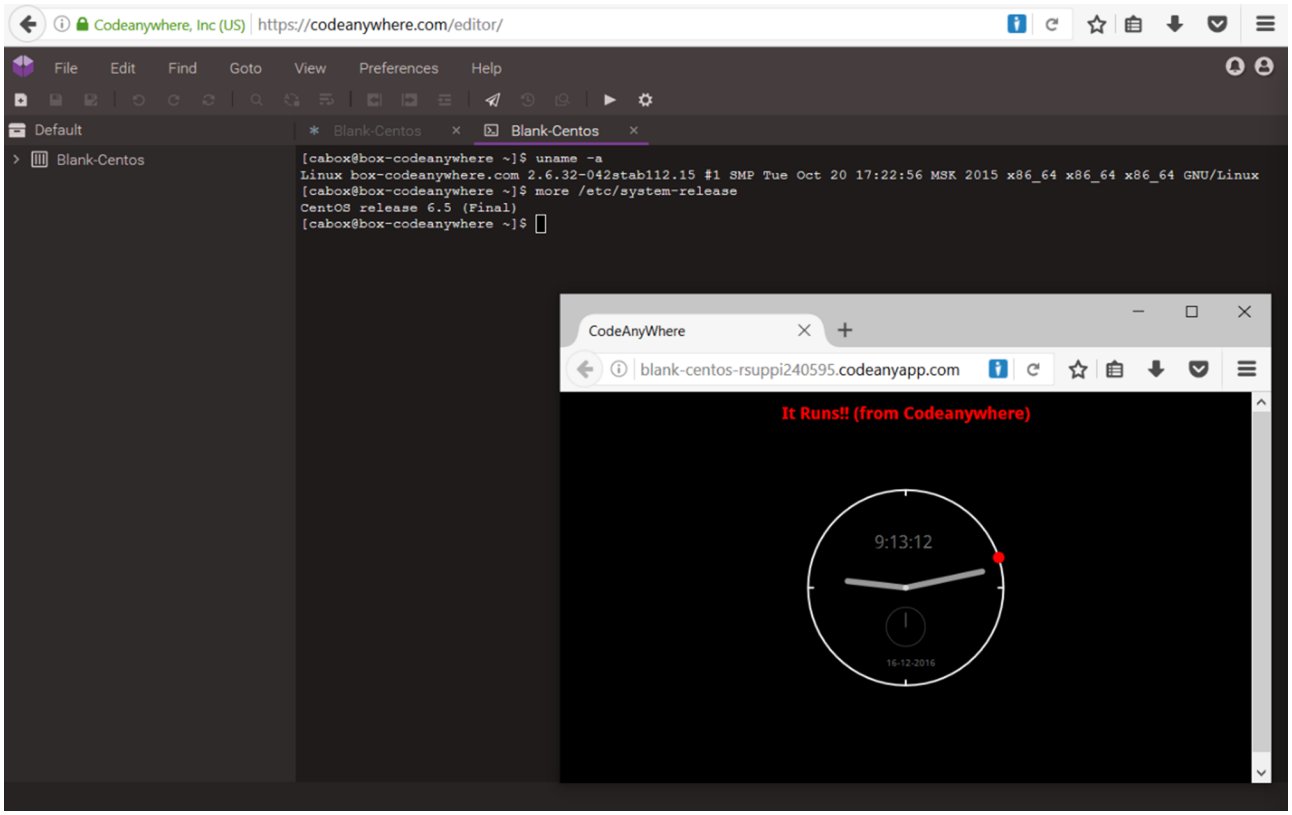
Y copiar la llave.

En Codeanywhere abrir un terminal SSH y copiar la llave en *.ssh/authorized_keys* (verificar que la llave queda en una única línea; si no, borrar los saltos de línea que se podrían generar en la copia).

```
vi .ssh/authorized_keys
```

Luego mirar la dirección y puerto en *Workspace* → *info* (que será algo como *host12.codeanyhost.com:51985*) y conectarse con:

```
ssh cabox@host12.codeanyhost.com -p 5198
```



Uno de los aspectos interesantes de este entorno es la posibilidad de diferentes conexiones (GitHub, Git, BitBucket, SSH/SFTP), así como el almacenamiento en *cloud* (Drive, OneDrive, Dropbox, S3, DO) o la posibilidad de compartir un archivo/directorio/entorno con otro usuario para trabajar conjuntamente. Solo se debe tener en cuenta que se ha utilizado la suscripción gratuita (*free*) que solo permite un entorno (*workspace*), pero la de 5 entornos tiene un costo de 2 $ por usuario/mes (facturado anualmente).

Como se puede observar, es un entono IDE-SaaS pero que también permite un pseudo-IaaS donde las máquinas se podrán interconectar, pero el usuario no podrá definir la infraestructura de red como en un IaaS real.

1.5.2. Codenvy

Codenvy es un entorno basado también en espacios de trabajo que provee una infraestructura que puede ser tanto IaaS como PaaS; no obstante, no tiene grandes posibilidades de interconexión, aunque permite el desarrollo muy ágil de las tareas sobre el espacio virtual. Si bien puede considerarse como un SaaS de autoservicio, también es posible configurar una máquina desnuda con el SO operativo solamente o el entorno también está disponible para infraestructura local [Con].

Codenvy es una plataforma (comercial) pero que permite suscripciones gratuitas (hasta 3GB RAM) basada en el proyecto en *open source* Eclipse Che, siendo Codenvy un contenedor donde se ejecutarán los espacios de trabajo y *plugins* de Eclipse Che.

Además de las características propias de Eclipse Che, Codenvy aporta la distribución del espacio de trabajo en un clúster elástico con Docker Swarm, monitorización, actualización, escalado, gestión de permisos y herramientas de gestión de políticas de recursos, autenticación de usuario y *factories* que añaden acciones a los espacios de trabajo portables de Eclipse Che.

Eclipse Che fue diseñado como un sistema de un solo usuario y puede ser utilizado por los usuarios de Codenvy en sus máquinas locales para proporcionar acceso sin conexión a sus espacios de trabajo sobre Codenvy y con la potencia y eficacia de Docker para sustentar cada *Workspace* y con una arquitectura que permite el escalado lineal en función de la carga y el número de usuarios. Entre sus usuarios de renombre están RedHat, Intuit, Sap, YouthDigital; trabajan conjuntamente con Docker, Bitnami, Circleci, Gradle, Puppets, CLoudBee, CodeShip y Wercker para aunar sinergias y unificar criterios en la estrategia de *Continuous Delivery*.

El entorno de trabajo está dividido en una sección vertical izquierda donde se verá el *Dashboard*, *Workspaces*, *Stacks* y *Factories* y, a continuación, los *Workspaces* definidos (que se podrán tener tantos activos como memoria RAM se disponga). Luego se podrá crear un *Workspace* (desde el acceso directo o desde el *Workspace*) y se podrá seleccionar el *runtime* que se desee (en nuestro caso se ha creado uno de tipo *Blank* basado en Ubuntu y uno basado en PHP que integra el servidor Apache para hacer la prueba de funcionalidad).

Luego, dentro del espacio de trabajo, existen tres zonas (en la configuración por defecto), la superior izquierda para el gestor de archivos y la superior derecha para el editor, y la inferior para la consola y terminales que se puede abrir mostrándose la interfaz al SO totalmente funcional (con `sudo -i` para pasar como *root* y se pueden utilizar los comandos de administración para instalar/actualizar paquetes).

También se puede cargar la llave pública (en el menú superior *Profile* → *Preferences* → *SSH*) y conectarse remotamente con la información dada en la parte inferior en el *tab* SSH que es algo como (verificar la URL y el puerto y recordar de reiniciar el espacio después de cargar la llave pública):

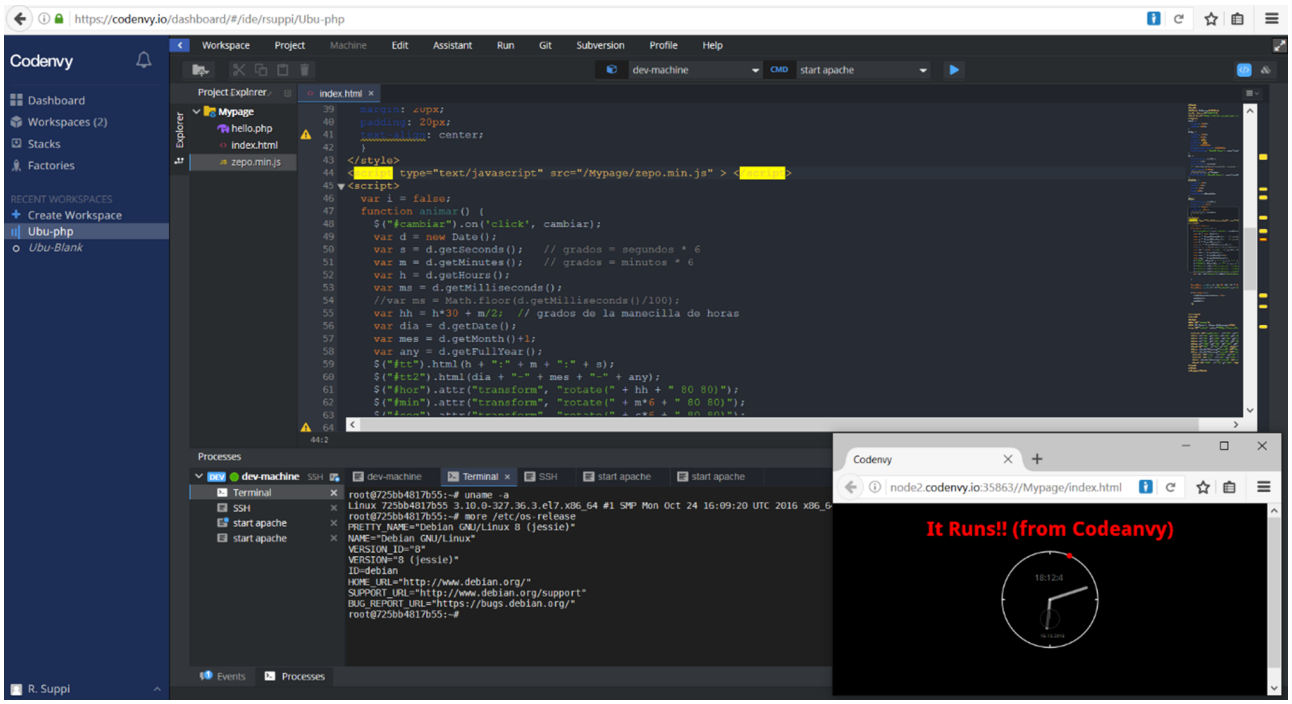
```
ssh user@node2.codenvy.io -p 35979
```

En función del entorno en la parte superior, hay dos tabuladores que permiten ver las opciones de la *dev-machine* (la que se está ejecutando) y a su derecha los comandos (CMD) que se pueden ejecutar (por ejemplo, reiniciar, parar, ejecutar Apache en nuestro entorno).

Es importante tener en cuenta que este entorno permite tener diferentes *Workspaces*, pero en la suscripción gratuita está limitada por la cantidad de memoria (3 GB) y donde cada uno ocupa 2 GB, por lo cual (de este tipo) solo se puede tener uno en ejecución, pero permite tener diferentes definidos. También es interesante, desde el punto de vista del desarrollador, el hecho de que permite integrarse con Git y Subversion y una serie de herramientas para integrar, construir, depurar y compartir, orientadas a facilitar el trabajo de creación de software; también, los usuarios pueden trabajar en su entorno local (por ejemplo, Eclipse Che) y luego integrarse rápidamente con el entorno para compartir su trabajo con otras personas del grupo.

Este entorno se puede utilizar como SaaS, pero también hay posibilidades de instalarlo localmente o en un IaaS (no es gratuito) y existe una guía detallada de la arquitectura y los pasos para su instalación y configuración.

La figura siguiente muestra la edición de un archivo html5 + javascript y su ejecución a través de un navegador.



Como conclusión, si bien está más orientado a un SaaS, el entorno también puede interpretarse como pseudo-IaaS, pero teniendo en cuenta que la interconectividad es *As-Is* y que el usuario podrá interconectar aplicaciones internas y externas de una forma muy simple y sin las posibilidades que tiene un IaaS real.

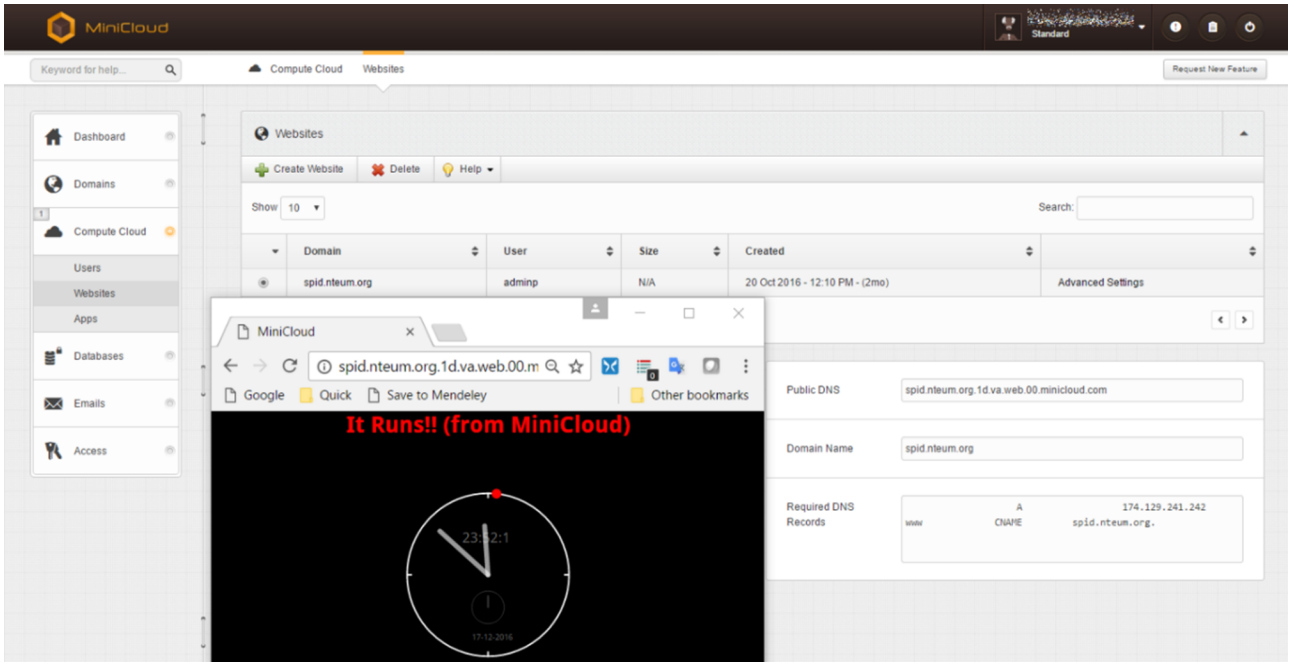
1.5.3. Minicloud

Minicloud es una empresa (radicada en San Diego, Estados Unidos) que tiene por misión prestar servicios de *cloud* empresariales de forma flexible, escalable y fácil de usar mediante una interfaz web simplificada. No es un *cloud* IDE, pero sus VPS están orientadas a servicios web, DB y mail que, si bien disponen de una API, solo es posible cambiar los parámetros de la máquina, pero no ajustar el SO o el software instalado. Según la información aportada se diferencian en diferentes aspectos con Rackspace, Godaddy VPS, DreamHost VPS entre las cuales se puede mencionar: la máquina inicial es gratuita (CPU 240MHz, 32MB RAM, 1GB de disco, 1Mbps de red y 1 Websites) frente a 15 $ o 30 $en los otros proveedores (la más pequeña tiene un costo de 3,12$), tiempo de despliegue frente a horas de algunos proveedores, facturación por minutos (no por horas/mes que utilizan otros proveedores).

Más allá de un política de precios muy diferentes a otros proveedores, permite escalar los servicios (básicamente Web) de forma automática o cambiar los recursos de los ya definidos para mejorar sus prestaciones en forma instantánea y sin interrupciones. También permite modificar sobre la marcha el almacenamiento para adecuarlo a las necesidades del cliente y gestionar la redundancia, directivas de seguridad o monitorización. El proveedor garantiza que cada entorno es totalmente aislado, ejecutándose en servidores redundantes para proveer de alta disponibilidad; dispone de herramientas y servicios para garantizar la seguridad de los sitios alojados (por ejemplo, ante DDoS o cualquier intento de *hacking*) con autenticación de dos pasos para proteger el robo de credenciales. Además, entre las opciones de despliegue, permite ejecutar los sitios web no solo de múltiples servidores, sino también de múltiples centros de datos (con certificación Tier-III) que son completamente independientes y redundantes para mejorar la disponibilidad y dar respuesta a un alto nivel de transacciones.

Como prueba funcional, se puede realizar un registro para el que será necesario un TE para completarlo y una dirección de correo. Luego se podrá acceder al entorno que presenta una barra lateral de herramientas: *Dashboard*, *Domains*, *Compute Cloud Databases*, *Emails* y *Access*. En nuestro caso, nos interesa la de *ComputeCloud* donde se dará de alta primero un usuario donde se definirá el VPS (características, localización, escalabilidad) y luego se podrá crear un sitio web sobre este recurso (en la siguiente opción).

En la información proporcionada por el proveedor se indicará la IP y el dominio asociado (tanto el real como el registro a poner en el DNS para el propio). Mediante al IP y el usuario/*passwd* creado anteriormente se podrá acceder por `ssh` y copiar mediante `scp` los archivos sobre el servidor (se deberá ubicar en */home/adminp/dominio_escogido/www*) y ya se podrá acceder por la URL indicada. La figura a continuación muestra el entorno y el sitio web desplegado sobre la infraestructura (de suscripción gratuita).

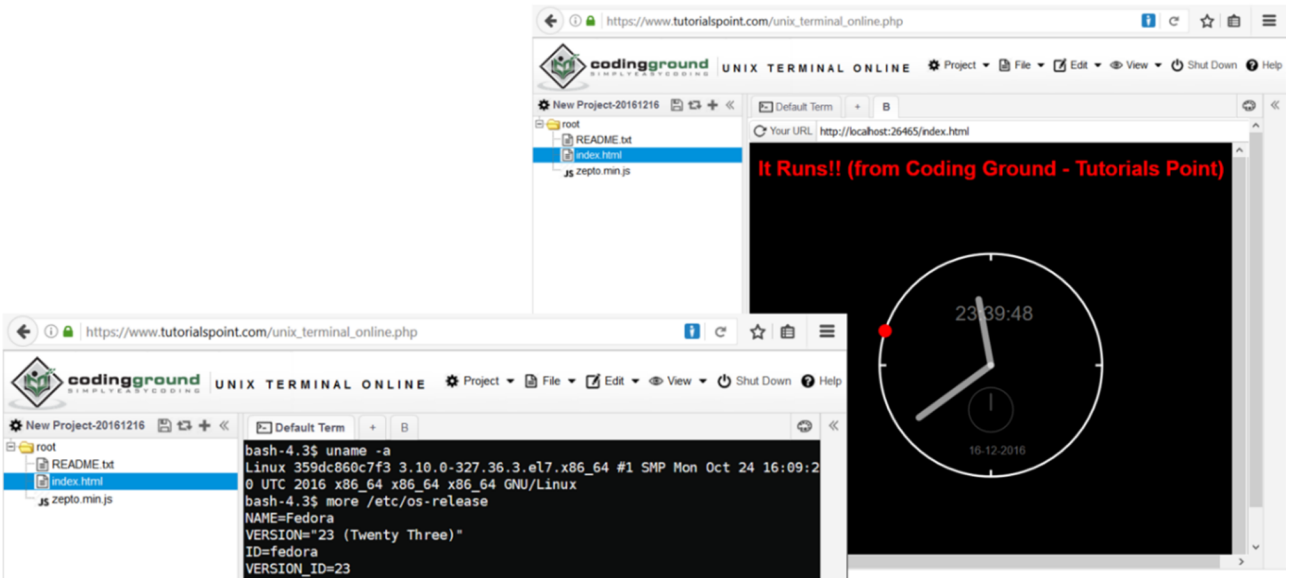


Como resumen, MiniCloud es un sitio proveedor de infraestructura para empresas, pero orientado a sitios web, bases de datos y correo (aunque hay *apps* en desarrollo de futura aparición), pero no se tiene contacto con el SO (*Linux 1d.va.web.00.minicloud.com 2.6.32-379.14.1.lve1.1.9.9.el6.x86_64 #1 SMP Thu Dec 6 07:12:24 EST 2012 x86_64 x86_64 x86_64 GNU/Linux*) que es un Linux de 64b pero en un *kernel* 2.6 y que se encuentra limitado a nivel de administración (por ejemplo, no es posible instalar nuevas aplicaciones o no dispone de gestión privilegiada a través del `sudo`).

1.5.4. Tutorial Point: Coding Ground

Como complemento a una gran cantidad de material y herramientas sobre diferentes aspectos de sistemas operativos, lenguajes de programación, entornos, herramientas en ámbito de la IT, Tutorial Point ofrece un macroentorno en línea gratuito basado en la premisa *Simply Easy Learning*, llamado Coding Groud de despliegue inmediato para conectarse a una máquina virtual. Sobre este entorno es posible trabajar sobre diferentes herramientas con 17 terminales en línea (para Centos, Ipython, Lua, Memcached, Mongo DB, MySQL, Node.JS, Numpy, Oracle, Octave, PowerShell, PHP, R Programming, Redis, Ruby, Scipy, Sympy) y 96 IDE en línea.

Si bien todo está orientado a enseñar y aprender sobre estos sistemas, puede ser un punto interesante iniciar el trabajo en un entorno virtualizado sobre múltiples terminales y un navegador incrustado todo desde un navegador. La imagen siguiente muestra la ejecución de una máquina Fedora 23 con los archivos que se han subido y ejecutado en la máquina local. Si bien es inmediato y no es necesario ningún registro/autorización, es un entorno básico que puede servir para probar diferentes máquinas (o *stacks*), pero se debe tener en cuenta que, aunque permite tener MV enlínea, no es posible configurar ninguna característica de ellas ni su interconectividad dado que su objetivo es otro.



2. IaaS privado: casos de uso

A continuación se describirán e instalarán, en sus últimas versiones y como prueba de concepto, las plataformas IaaS más utilizadas o de renombre dentro de este ámbito (orden alfabético).

2.1. Apache CloudStack

Apache CloudStack es una plataforma *open source* (*GitHub*) para desplegar y administrar grandes redes de máquinas virtuales, como una plataforma de infraestructura como servicio (IaaS) altamente escalable y de alta disponibilidad.

CloudStack es utilizado por diversos proveedores de servicios para ofrecer servicios de *cloud* público, y por compañías para ofrecer *cloud* privados o como parte de una solución de *cloud* híbrido. CloudStack incluye todas las características que se necesita en un IaaS como orquestación de los servicios, *network-as-a-service*, administración de usuarios y cuentas, API nativa completa y abierta, contabilidad de recursos e interfaz de usuario (GUI).

Actualmente, CloudStack soporta los hipervisores más populares (VMware, KVM, Citrix XenServer, Xen Cloud Platform (XCP), Oracle VM Server y Microsoft Hyper-V) y los usuarios pueden administrar su *cloud* mediante una GUI, con una CLI o con una API RESTful con todas las funciones (además, CloudStack proporciona una API compatible con AWS EC2 y S3 para organizaciones que deseen implementar nubes híbridas).

Una lista de los usuarios institucionales/corporativos más representativos se puede obtener desde la página del proyecto.

Como **arquitectura general**, CloudStack se despliega como un servidor de administración (*management server*) y los recursos que se van a administrar (hipervisores). Una instalación mínima consistirá en una máquina que ejecuta *CloudStack Management Server* y otra para actuar como infraestructura *cloud* (en este caso, una infraestructura muy simple que consiste en un *host* que ejecuta el software del hipervisor).

En casos de prueba, se podría incluso montar el servidor de recursos en el mismo servidor de administración, pero para una instalación de producción, el *management server* deberá ser multi-nodo con alta disponibilidad, además de los *hosts* necesarios para desplegar el *cloud*.

CloudStack involucra una serie de conceptos básicos que es conveniente revisar: regiones, zonas, *pods*, *hosts*, además de almacenamiento primario/secundario, redes físicas, entre otros.

Para la contribución y desarrollo de CloudStack existe una wiki específica con información técnica sobre el desarrollo y las API. En el apartado 101 (para desarrolladores) existen indicaciones como montar DevCloud que es una *virtual appliance* (OVA), basada en Xen, para crear un entorno de pruebas sobre Virtualbox. No obstante, la versión de Xen está desactualizada (2014) y de hecho ni tan solo están los enlaces a las imágenes, aunque estas *-devcloud2.ova-* se pueden obtener de algún repositorio en internet. No obstante, la versión KVM está disponible y se puede instalar sin problemas [Acs][Aag].

2.1.1. Instalación sobre Centos 6.8 con *host* KVM

Para esta instalación se han seguido los pasos detallados en la última versión de la documentación *Quick Installation Guide for CentOS 6* con la única diferencia de que se ha realizado sobre una MV KVM con virtualización anidada y una NIC que durante la instalación del SO se ha mantenido en NAT y posteriormente se ha configurado como *bridge* (con IP 192.168.1.200).

Los pasos en resumen son:

1) Configurar la red: */etc/sysconfig/network-scripts/ifcfg-eth0*

```
DEVICE=eth0
HWADDR=aa:bb:cc:dd:ee:ff
NM_CONTROLLED=no
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.1.200
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
DNS1=8.8.8.8
DNS2=8.8.4.4
```

Modificar */etc/hosts* incluyendo
192.168.1.200 cloudstack.nteum.org cloudstack

```
chkconfig network on
service network start
```

Modificar */etc/ssh/sshd_config* para permitir la conexión del usuario *root* por `ssh` (*PermitRootLogin=yes*) y reiniciar el servicio de SSH y luego el *passwd* de *root*.

2) SELinux: configurarlo a *permissive*

```
setenforce 0
```

Asegurarse de que en */etc/selinux/config to* esté *SELINUX=permissive*

3) NTP. Instalación del servicio:

```
yum -y install ntp
chkconfig ntpd on
service ntpd start
```

4) Repositorio de CloudStack: crear */etc/yum.repos.d/cloudstack.repo* con la siguiente información:

```
[cloudstack]
name=cloudstack
baseurl=http://cloudstack.apt-get.eu/centos/6/4.9/
enabled=1
gpgcheck=0
```

5) NFS. Instalación del servicio:

```
yum -y install nfs-utils
mkdir -p /export/primary
mkdir /export/secondary
```

Incluir en */etc/exports*:

```
/export/secondary *(rw,async,no_root_squash,no_subtree_check)
/export/primary *(rw,async,no_root_squash,no_subtree_check)
```

Dado que CentOS 6.x utiliza NFSv4, es necesario modificar */etc/idmapd.conf* quitando el comentario de la clave *Domain* para poner nuestro dominio (*nteum.org*) y modificar */etc/sysconfig/nfs*

```
LOCKD_TCPPORT=32803
LOCKD_UDPPORT=32769
MOUNTD_PORT=892
RQUOTAD_PORT=875
STATD_PORT=662
STATD_OUTGOING_PORT=2020
```

También modificar */etc/sysconfig/iptables* para permitir las conexiones:

```
-A INPUT -s 172.16.10.0/24 -m state --state NEW -p udp --dport 111 -j ACCEPT
-A INPUT -s 172.16.10.0/24 -m state --state NEW -p tcp --dport 111 -j ACCEPT
```

```
-A INPUT -s 172.16.10.0/24 -m state --state NEW -p tcp --dport 2049 -j ACCEPT
-A INPUT -s 172.16.10.0/24 -m state --state NEW -p tcp --dport 32803 -j ACCEPT
-A INPUT -s 172.16.10.0/24 -m state --state NEW -p udp --dport 32769 -j ACCEPT
-A INPUT -s 172.16.10.0/24 -m state --state NEW -p tcp --dport 892 -j ACCEPT
-A INPUT -s 172.16.10.0/24 -m state --state NEW -p udp --dport 892 -j ACCEPT
-A INPUT -s 172.16.10.0/24 -m state --state NEW -p tcp --dport 875 -j ACCEPT
-A INPUT -s 172.16.10.0/24 -m state --state NEW -p udp --dport 875 -j ACCEPT
-A INPUT -s 172.16.10.0/24 -m state --state NEW -p tcp --dport 662 -j ACCEPT
-A INPUT -s 172.16.10.0/24 -m state --state NEW -p udp --dport 662 -j ACCEPT
```

Luego reiniciar los servicios:

```
service iptables restart
service rpcbind start
service nfs start
chkconfig rpcbind on
chkconfig nfs on
```

6) Instalación de MySQL y del *management server*:

```
yum -y install mysql-server
```

Modificar */etc/my.cnf* en la sección [mysqld]:

```
innodb_rollback_on_timeout=1
innodb_lock_wait_timeout=600
max_connections=350
log-bin=mysql-bin
binlog-format = 'ROW'
```

Reiniciar el servicio:

```
service mysqld start
chkconfig mysqld on
```

7) Instalar el conector MySQL-Python: crear */etc/yum.repos.d/mysql.repo* con el siguiente contenido:

```
[mysql-connectors-community]
name=MySQL Community connectors
baseurl=http://repo.mysql.com/yum/mysql-connectors-community/el/$releasever/$basearch/
enabled=1
gpgcheck=1
```

Importar la llave GPG desde MySQL e instalar el conector:

```
rpm --import http://repo.mysql.com/RPM-GPG-KEY-mysql
```

```
yum install mysql-connector-python
```

8) Instalación ACS-MS e inicialización de las BD:

```
yum -y install cloudstack-management
cloudstack-setup-databases cloud:password@localhost --deploy-as=root
cloudstack-setup-management
```

9) Carga del *System Template*:

```
/usr/share/cloudstack-common/scripts/storage/secondary/cloud-install-sys-tmplt \
-m /export/secondary \
-u http://cloudstack.apt-get.eu/systemvm/4.6/systemvm64template-4.6.0-kvm.qcow2.bz2 \
-h kvm -F
```

10) Instalación del nodo KVM: se instalará en la misma MV, pero podría ser otra en la cual se deberían repetir los primeros pasos que se han realizado en esta (configuración de la red, *hostname*, SELinux, NTP, y Repositorio). La instalación del agente será:

```
yum -y install cloudstack-agent
```

Modificación de */etc/libvirt/qemu.conf* para permitir la conexión por VNC:

```
vnc_listen=0.0.0.0
```

Configuración de */etc/libvirt/libvirtd.conf* para modificar los parámetros de conexión:

```
listen_tls = 0
listen_tcp = 1
tcp_port = "16059"
auth_tcp = "none"
mdns_adv = 0
```

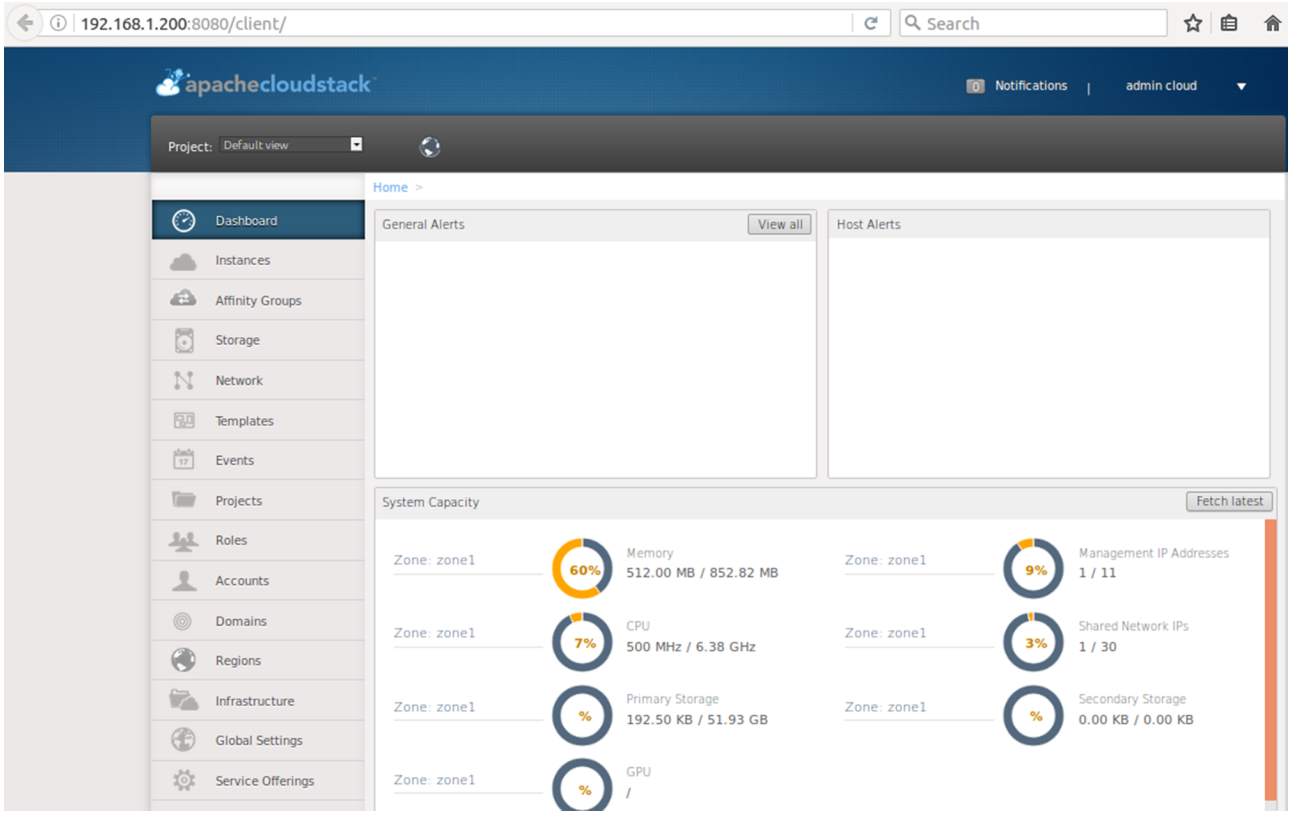
Y modificar */etc/sysconfig/libvirtd* quitando el '#' a la siguiente línea *#LIBVIRTD_ARGS="--listen*" (deberá quedar sin #) y reiniciar *libvirt*:

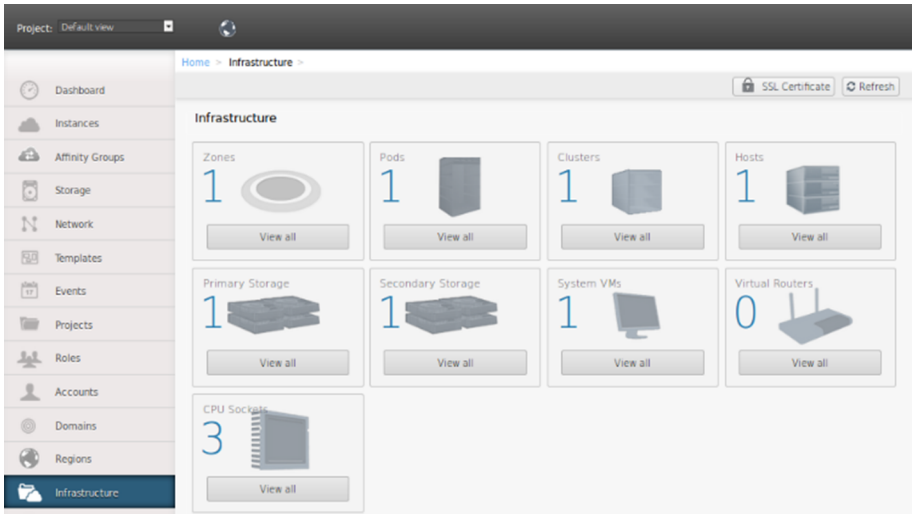
```
service libvirtd restart
```

11) Configuración del *cloud*: acceder desde un navegador a *http://192.168.1.200:8080/client* con usuario *admin* y *passwd* **password** para inicializar la zona/pod/cluster/host.

```
Name: Zone1
Public DNS 1: 8.8.8.8
Public DNS 2: 8.8.4.4
Internal DNS1: 8.8.8.8
Internal DNS2: 8.8.4.4
Pod Name: Pod1
Gateway: 192.168.1.1
Netmask: 255.255.255.0
Start/end reserved system IPs: 192.1681.210-192.168.1.220
Guest Gateway: 192.168.1.1
Guest netmask: 255.255.255.0
Guest start/end IP: 192.1681.221-192.168.1.250
Cluster Name: Cluster1
Hypervisor: KVM
Host: 192.168.1.200.
Username: root
Password: passwd del root en la MV Centos.
Primary Storage Name: Primary1
Server: 192.168.1.200
Path: /export/primary
Secondary Storage NFS server: 192.168.1.200
Path: /export/secondary
```

Haciendo clic en *Launch* se inicializará todas las partes del *cloud* y en otra ventana con la misma URL ya se podrá ver el *cloud*. Las figuras a continuación muestran el *dashboard* del *cloud* desplegado, la infraestructura y la conexión vía `ssh` a una máquina virtual.





2.1.2. Despliegue de CloudStack sobre Ubuntu 14.04 con *host* KVM

El despliegue sobre Ubuntu es similar que en Centos, pero existen diferencias. Teniendo en cuenta los pasos indicados en la documentación de ShapeBlue (uno de los grandes integradores de tecnologías Cloudstack), se desplegará un *cloud* Apache CloudStack 4.9 (ACS 4.9 es la última versión) sobre una única máquina con KVM y una red básica. Esta prueba puede ser hecha sobre *bare-metal* o, como en este caso, en una máquina virtual KVM con Ubuntu 14.04 LTS instalado. Es importante que la MV tenga la virtualización anidada habilitada y 3GB RAM y 30 GB de disco, con ip propia (modo *bridge*) sobre el *host* (en nuestro caso será 192.168.1.200).

Lo pasos a realizar serán:

1) inicialización de la red (*bridge*),

2) instalación de *cloudstack-management/ cloudstack-common*,

3) instalación de MySQL *server*,

4) inicialización del NFS para el almacenamiento primario y secundario,

5) instalación de *template systemvm*,

6) configuración del *host* KVM *host* e instalación de *cloudstack-agent*,

7) y finalmente configuración de *firewall*.

Prerrequisitos:

```
apt-get install ntp openssh-server build-essential
```

Modificar */etc/ssh/sshd_config* para permitir la conexión del usuario *root* por ssh (*PermitRootLogin=yes*) y reiniciar el servicio SSH y el *passwd* de *root*.

```
sudo -i
passwd root
```

Red: CloudStack requiere que los *hosts* KVM dispongan de 2 *bridges* (cloudbr0 y cloudbr1) –deben ser estos nombres ya que ACS utiliza estos nombres por defecto– que permitirán que la MV puedan comunicarse con el *host*, entre ellas mismas o con el exterior.

```
apt-get install bridge-utils
```

Modificar */etc/network/interfaces* con una configuración similar a la siguiente (reemplazar la IP por la de la MV).

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet manual
# Public network
auto cloudbr0
iface cloudbr0 inet static
  address 192.168.1.200
  netmask 255.255.255.0
  gateway 192.168.1.1
  dns-nameservers 8.8.8.8 8.8.4.4
  bridge_ports eth0
  bridge_fd 5
  bridge_stp off
  bridge_maxwait 1
# Private network
auto cloudbr1
iface cloudbr1 inet manual
  bridge_ports none
  bridge_fd 5
  bridge_stp off
  bridge_maxwait 1
```

En este caso son se utilizará cloudbr1 ya que se desplegará una zona básica con una configuración básica de red que solo utilizará cloudbr0 para ello. Hacer un *reboot* de la MV después de esta configuración y verificar que se continúa teniendo conexión externa.

***Management server* y MySQL**: para la instalación se utilizarán los paquetes ya configurados para Ubuntu en *http://cloudstack.apt-get.eu/ubuntu* (también se podría construir desde el código fuente).

```
echo "deb http://cloudstack.apt-get.eu/ubuntu $(lsb_release -s -c) 4.8" \
> /etc/apt/sources.list.d/cloudstack.list
wget -O - http://cloudstack.apt-get.eu/release.asc|apt-key add -
apt-get update
```

Verificar que */etc/apt/sources.list.d/cloudstack.list* tenga como contenido *deb http://cludstack.apt-get-eu/ubuntu trusty 4.8*. Instalar ACS-MS y Mysql:

```
apt-get install cloudstack-management cloudstack-common mysql-server
```

Introducir un *passwd* para Mysql (y recordarlo, pues será necesario más adelante). Modificar */etc/mysql/my.cnf* (no es necesario en esta pequeña instalación, pero sí para instalaciones en producción), en la sección [mysqld] agregar:

```
innodb_rollback_on_timeout=1
innodb_lock_wait_timeout=600
max_connections=350
log-bin=mysql-bin
binlog-format = 'ROW'
```

Reiniciar la base de datos e inicializarla:

```
service mysql restart
cloudstack-setup-databases cloud:cloudpassword@localhost \
 --deploy-as=root:introducir_passwd_root -i 192.168.1.200
```

Donde el *passwd* será del *root* de la base de datos (introducido durante la instalación) y la IP la asignada a cloudbr0.

Almacenamiento: inicialización del NFS para el almacenamiento primario y secundario y carga de *systemvm*.

```
mkdir -p /export/primary /export/secondary
apt-get install nfs-kernel-server quota
echo "/export *(rw,async,no_root_squash,no_subtree_check)" > \
  /etc/exports
exportfs -a
sed -i -e 's/^RPCMOUNTDOPTS=--manage-gids$/RPCMOUNTDOPTS="-p 892 \
  --manage-gids"/g' /etc/default/nfs-kernel-server
sed -i -e 's/^NEED_STATD=$/NEED_STATD=yes/g' /etc/default/nfs-common
sed -i -e 's/^STATDOPTS=$/STATDOPTS="--port 662 --outgoing-port 2020"/g'\
   /etc/default/nfs-common
sed -i -e 's/^RPCRQUOTADOPTS=$/RPCRQUOTADOPTS="-p 875"/g' /etc/default/quota
```

```
service nfs-kernel-server restart
```

Descargar *systemvm* y desplegarlo:

```
wget http://cloudstack.apt-get.eu/systemvm/4.6/systemvm64template-4.6.0-kvm.qcow2.bz2
 la URL debe ser toda en una línea
cd /usr/share/cloudstack-common/scripts/storage/secondary/
./cloud-install-sys-tmplt -m /export/secondary \
 -f systemvm64template-4.6.0-kvm.qcow2.bz2 -h kvm \
 -o localhost -r cloud -d cloudpassword
```

Tener en cuenta que en el parámetro -f hay que incluir toda la ruta a donde se ha descargado el archivo anterior.

KVM y *agent setup*: inicialización de nodo KVM (el que ejecutará las máquinas virtuales).

```
apt-get install qemu-kvm cloudstack-agent
sed -i -e 's/listen_tls = 1/listen_tls = 0/g' /etc/libvirt/libvirtd.conf
echo 'listen_tcp=1' >> /etc/libvirt/libvirtd.conf
echo 'tcp_port = "16509"' >> /etc/libvirt/libvirtd.conf
echo 'mdns_adv = 0' >> /etc/libvirt/libvirtd.conf
echo 'auth_tcp = "none"' >> /etc/libvirt/libvirtd.conf
sed -i -e 's/\# vnc_listen.*$/vnc_listen = "0.0.0.0"/g' /etc/libvirt/qemu.conf
sed -i -e 's/libvirtd_opts="-d"/libvirtd_opts="-d -l"/' \
 /etc/init/libvirt-bin.conf
service libvirt-bin restart
```

Inicialización del *firewall*: reemplazar la red por la que corresponda.

```
NETWORK=192.168.1.0/24
iptables -A INPUT -s $NETWORK -m state --state NEW -p udp --dport 111 -j ACCEPT
iptables -A INPUT -s $NETWORK -m state --state NEW -p tcp --dport 111 -j ACCEPT
iptables -A INPUT -s $NETWORK -m state --state NEW -p tcp --dport 2049 -j ACCEPT
iptables -A INPUT -s $NETWORK -m state --state NEW -p tcp --dport 32803 -j ACCEPT
iptables -A INPUT -s $NETWORK -m state --state NEW -p udp --dport 32769 -j ACCEPT
iptables -A INPUT -s $NETWORK -m state --state NEW -p tcp --dport 892 -j ACCEPT
iptables -A INPUT -s $NETWORK -m state --state NEW -p udp --dport 892 -j ACCEPT
iptables -A INPUT -s $NETWORK -m state --state NEW -p tcp --dport 875 -j ACCEPT
iptables -A INPUT -s $NETWORK -m state --state NEW -p udp --dport 875 -j ACCEPT
iptables -A INPUT -s $NETWORK -m state --state NEW -p tcp --dport 662 -j ACCEPT
iptables -A INPUT -s $NETWORK -m state --state NEW -p udp --dport 662 -j ACCEPT
apt-get install iptables-persistent
```

Deshabilitar *apparmour* sobre libvirtd:

```
ln -s /etc/apparmor.d/usr.sbin.libvirtd /etc/apparmor.d/disable/
```

```
ln -s /etc/apparmor.d/usr.lib.libvirt.virt-aa-helper /etc/apparmor.d/disable/
apparmor_parser -R /etc/apparmor.d/usr.sbin.libvirtd
apparmor_parser -R /etc/apparmor.d/usr.lib.libvirt.virt-aa-helper
```

Configurar ufw:

```
ufw allow mysql
ufw allow proto tcp from any to any port 22
ufw allow proto tcp from any to any port 1798
ufw allow proto tcp from any to any port 16509
ufw allow proto tcp from any to any port 5900:6100
ufw allow proto tcp from any to any port 49152:49216
```

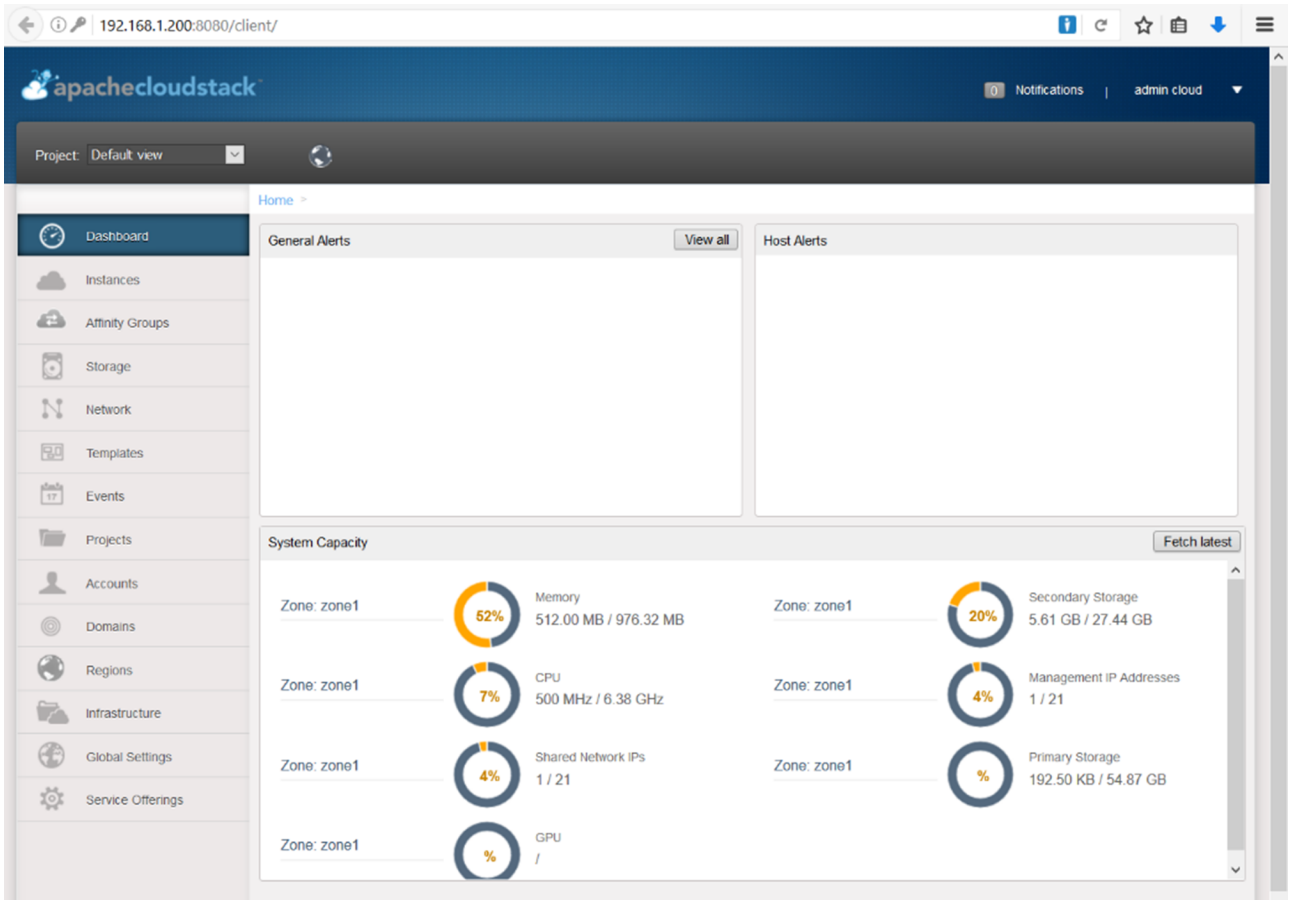
Lanzar el *cloud*: parar el tomcat6, arrancar los servicios y conectarse:

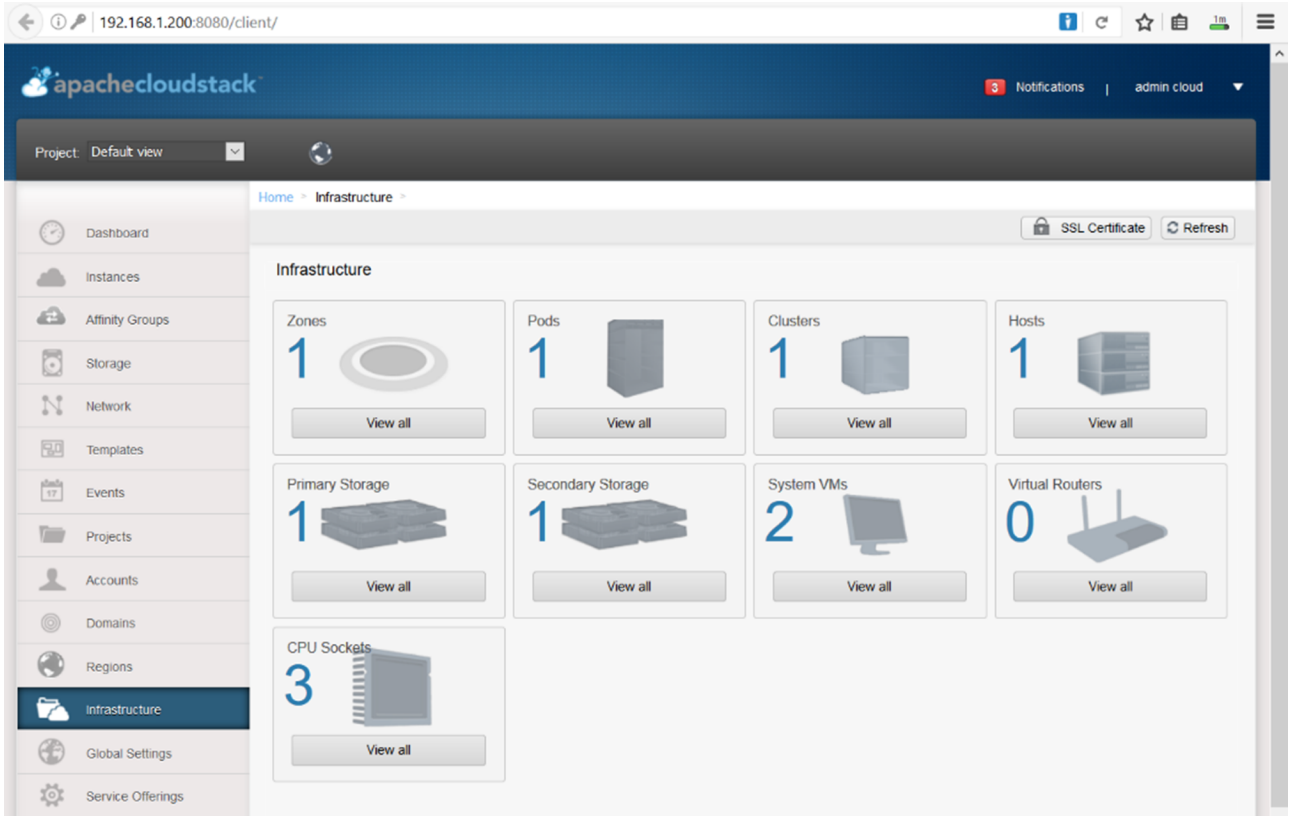
```
/etc/init.d/tomcat6 stop
/etc/init.d/cloudstack-agent start
/etc/init.d/cloudstack-management start
```

En un navegador poner la URL *http://cloudbr0-IP:8080/client* donde se verá la página de conexión con usuario *admin* and *passwd* **password**. A continuación, se deberá proceder a la configuración de una zona básica (adecuar las IP):

```
Introducir un nombre, DNS External DNS 8.8.8.8, Internal 8.8.8.8
Network: gateway 192.168.1.1, netmask 255.255.255.0, IP range 192.168.1.220-250
Guest network: gateway 192.168.1.1, netmask 255.255.255.0, IP range 192.168.1.100-130
Introducir un cluster name, hypervisor KVM
Añadir KVM host, IP 192.168.1.200, user root, password el que tenga la MV
Añadir primary NFS storage, IP 192.168.1.200, path /export/primary
Añadir secondary NFS storage, IP 192.168.1.200, path /export/secondary
```

Hacer clic en *Launch* y se desplegará el clúster (recargar la URL para ver el clúster desplegado). Se pueden ver los errores en */var/log/cloudstack/management/management-server.log y /var/log/cloudstack/agent/agent.log* por los posibles errores. Consultar la documentación de administración en [Aag]. Las dos figuras siguientes, similares a las de Centos, muestran el *dashboard* del clúster y la infraestructura desplegada.


192.168.1.200:8080/client/
apachecloudstack
Notifications
admin cloud
Project: Default view
Home
Dashboard
Instances
Affinity Groups
Storage
Network
Templates
Events
Projects
Accounts
Domains
Regions
Infrastructure
Global Settings
Service Offerings
General Alerts
View all
Host Alerts
System Capacity
Fetch latest
Zone: zone1
52%
Memory
512.00 MB / 976.32 MB
Zone: zone1
20%
Secondary Storage
5.61 GB / 27.44 GB
Zone: zone1
7%
CPU
500 MHz / 6.38 GHz
Zone: zone1
4%
Management IP Addresses
1 / 21
Zone: zone1
4%
Shared Network IPs
1 / 21
Zone: zone1
%
Primary Storage
192.50 KB / 54.87 GB
Zone: zone1
%
GPU
/


192.168.1.200:8080/client/
apachecloudstack
Notifications
admin cloud
Project: Default view
Home
Infrastructure
SSL Certificate
Refresh
Dashboard
Instances
Affinity Groups
Storage
Network
Templates
Events
Projects
Accounts
Domains
Regions
Infrastructure
Global Settings
Service Offerings
Infrastructure
Zones
1
View all
Pods
1
View all
Clusters
1
View all
Hosts
1
View all
Primary Storage
1
View all
Secondary Storage
1
View all
System VMs
2
View all
Virtual Routers
0
View all
CPU Sockets
3
View all

2.1.3. Instalación de *CloudStack Management Server* desde el repositorio fuente (*git*)

Para los desarrolladores es útil disponer de un entorno con el código fuente para modificar o agregar funcionalidad, lo cual es posible cargando el código y compilarlo/instalarlo desde el repositorio. La documentación se refiere a un SO Ubuntu 12.04/CentOS 6.4 y Apache CloudStack 4.2 (ACS 4.2). No obstante, en esta prueba se ha utilizado una máquina virtual KVM con Ubuntu 16.04 64b (Xenial) y ACS 4.8 sin problemas. Los pasos han sido: instalación de los prerrequisitos, compilación e instalación desde el código fuente, instalación del simulador (opcional), vinculación a un hipervisor (por ejemplo, DevCloud si se tiene disponible).

Prerrequisitos: después de instalado Ubuntu 16.04 se ha realizado:

```
apt-get update
apt-get upgrade
apt-get install ntpd
apt-get install openjdk-8-jdk
apt-get install tomcat7
apt-get install mysql-server (recordar el passwd del usuario root)
apt-get install git
apt-get install maven
apt-get install python-pip python-setuptools libssl-dev
apt-get install genisoimage python-mysql.connector
```

Descargar y compilar/instalar desde el código fuente:

```
cd cloudstack
git clone https://git-wip-us.apache.org/repos/asf/cloudstack.git
git checkout 4.8
mvn -Pdeveloper,systemvm clean install
```

Se pueden saltar los test con -DskipTests ya que el tiempo de compilación es (muy) largo. Luego modificar *utils/conf/db.properties el campo de db.root.password*= con el *passwd* del *root* de la BD (el que se puso durante la instalación de Mysql) y desplegar la BD:

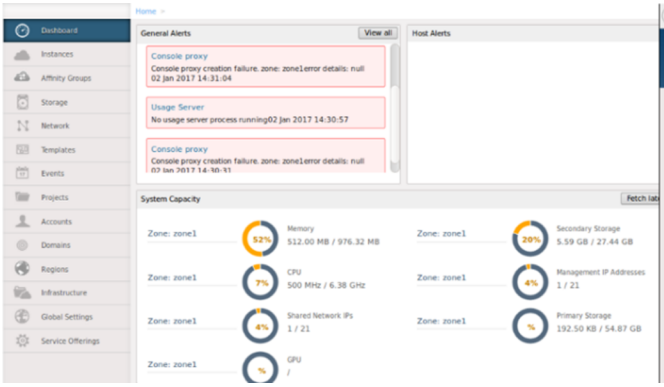
```
mvn -P developer -pl developer -Ddeploydb
```

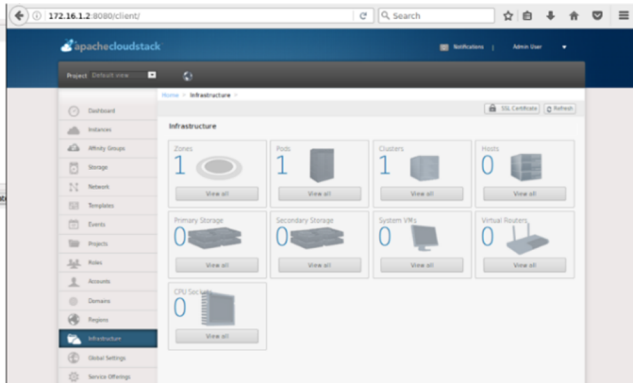
Ejecutar ACS con *jetty* como prueba, tener en cuenta que Tomcat puede estar en ejecución en el puerto 8080 y se debe detener antes de ejecutar *jetty*:

```
service tomcat7 stop
mvn -pl :cloud-client-ui -Dorg.eclipse.jetty.annoattions.maxWait=120 jetty:run
```

Conectarse a ACS usando *http://localhost:8080/client/* con usuario/*passwd* como *admin* y *password* respectivamente. Si no se dispone de entorno gráfico en la máquina virtual, se puede poner esta en modo *bridge* y hacerlo desde el *host* reemplazando *localhost* por la IP de la MV. Verificar que en el caso de tener *iptables* habilitado los puertos 8080, 8250 y 9090 deben estar abiertos.

El siguiente paso será configurar una Zona/Pod/Cluster/Host a través de la interfaz gráfica y desplegar el clúster, como se puede ver en las figuras siguientes (no se aprecia ningún *host* ya que no se ha configurado).





Finalmente, al *management server* desplegado se puede anexar DevCloud como hipervisor (no recomendado por EOL) o configurar uno propio (como se realizó en el apartado anterior). Consultar la documentación para mayor detalle sobre la instalación de un host KVM [Aag].

Si no se desea configurar infraestructura/almacenamiento/hipervisores/... se pueden ejecutar pruebas utilizando un simulador, que incorpora ACS 4.8, llamado Marvin. Este se encuentra disponible en el código fuente de ACS y permite simular la infraestructura del centro de datos, proporcionando a ACS un archivo de configuración que define el número de zonas/*pods*/clústeres/*hosts*, tipos de almacenamiento... y con ello probar el servidor de administración como si estuviera administrando infraestructura de producción [Acs].

Como resumen final, se puede observar que Apache Cloudstack es una plataforma compleja, que involucra un conjunto amplio de tecnologías, con unas funcionalidades/prestaciones excelentes, adaptables a cualquier tipo de entorno corporativo y con una comunidad muy activa. No obstante, dada su complejidad, la curva de aprendizaje es lenta y los procesos de instalación/configuración no están tan automatizados/simplificados como en otras plataformas. La documentación es detallada, pero existen diversas versiones que pueden confundir (y algunas, por ejemplo, en la de desarrolladores, *cwiki.apache.org*, algunas secciones están desactualizadas). Es por ello que el *sysadmin* deberá dedicar tiempo y esfuerzo a sintonizar ACS de acuerdo a sus necesidades e integrar toda la infraestructura hardware del entorno consultando las guías disponibles [Acs].

2.2. Eucaliptus

Eucalyptus (formalmente HPE Helion Eucalyptus) es una plataforma distribuida *open source* que permite el despliegue de *clouds* que se pueden integrar fácilmente con Amazon Web Services (AWS).

Su diseño permite ejecutarse sobre hardware no específico/habitual (*commodity hardware*) con servicios compatibles como AWS-EC2 y *auto scaling* presentando unas prestaciones y funcionalidades estables, robustas y avanzadas. Uno de los aspectos interesantes de su diseño es que puede escalar dinámicamente (agregar o reducir) recursos dependiendo de la carga de trabajo de las aplicaciones que lo hacen especialmente útiles en sistemas empresariales donde la carga sea variable y sin posibilidades de previsión, siendo compatible con los estándares de AWS (EC2, S3, EBS, IAM, *Auto Scaling*, *Elastic Load Balancing*, *Cloud Watch*, y *CloudFormation*).

También para uso en *cloud* privados aporta escalabilidad/eficiencia/agilidad en el aprovisionamiento de recursos mejorando las tareas habituales de un CPD [Hpe].

Existe una serie de dudas sobre la política y los siguientes pasos del proyecto HP Helion Eucalyptus que preocupan a la comunidad pero que HP se ha encargado de matizar, asegurando que Eucalyptus continuará sobre el paraguas de Helion (ver Register, InforWorld, Fortune Tech entre otros).

La **arquitectura conceptual** está formada por cuatro capas donde en el nivel inferior, capa de Nodo, se encuentran los *Node Controler* (NC) y las MV, en la siguiente, *Cluster*, donde se ejecutan el *Cluster Controler* (CC) y el *StorageControler* (SC), en la siguiente, *Cloud*, el *CloudControler* (CLC) y el *Scalable Object Storage* (SOS), y en la superior, *GUI & API*, la consola de administración y la API compatible con AWS.

Como **servicios representativos** de la arquitectura se puede mencionar:

- ***Facing Services***: la mayor parte de la AWS API son gestionadas por un componente denominado *User-Facing Services* (UFS) que implementa los servicios web compatibles con AWS, sirve de punto de integración/vinculación y se ejecuta en el mismo *host* que la consola de administración.

- **Consola de administración**: es una GUI basada en web que permite a los usuarios gestionar los recursos disponibles; para los administradores es una herramienta para gestionar usuarios, grupos, recursos, políticas etc.

- ***Object Storage Gateway*** (OSG): los UFS también incluyen el OSG que es el equivalente a *AWS Simple Storage Service* (S3). El OSG permite que los

administradores puedan disponer de un servicio flexible para implementar diferentes políticas de almacenamiento sobre dispositivos de bajo costo con una interface S3. Eucalyptus provee un almacenamiento básico denominado *Walrus*, apto para entornos de evaluación o pequeñas instalaciones, pero para medias/grandes instalaciones se recomiendan OSG sobre soluciones de almacenamiento dedicadas.

- ***Cloud Controller*** (CLC): este módulo, escrito en Java, mantiene una BD que registra los recursos/eventos en el *cloud*, así como un servicio de DNS que puede integrarse con implementaciones compatibles de *CloudFormation*.

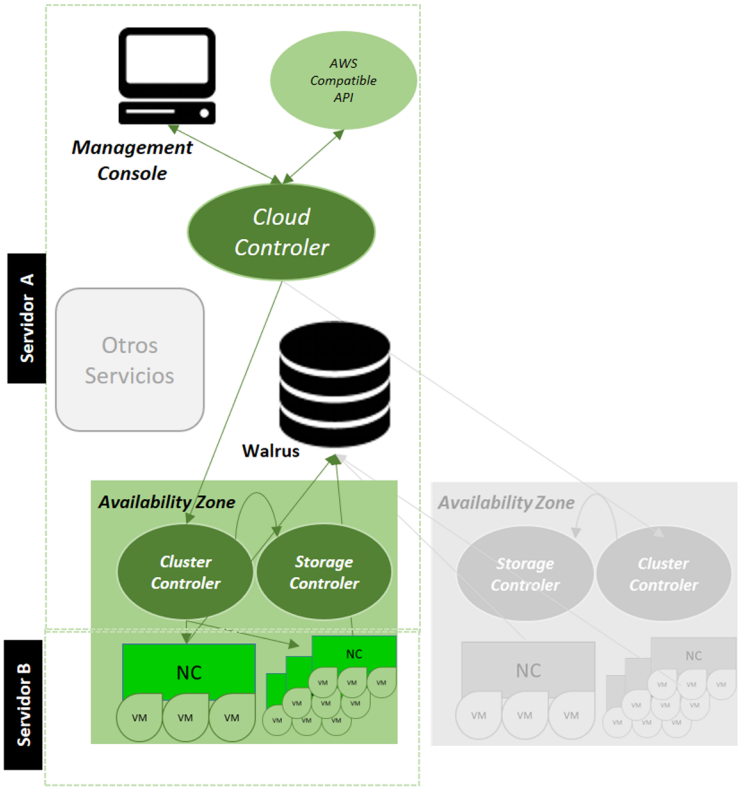
- ***Object Storage Provider*** (OSP): el OSP es la plataforma que trabaja conjuntamente con el OSG para proveer el servicio de almacenamiento compatible con S3. Walrus es uno de los proveedores de almacenamiento básico (simple) para OSG que utiliza un sistema de archivo compatible POSIX.

- ***Cluster Controller*** (CC): es equivalente a una *AWS availability zone*, y en un *cloud* Eucalyptus se pueden tener múltiple clústeres. Este módulo, escrito en C, actúa de intermediario entre el CLC y el *Node Controller* (NC) y trabaja juntamente con el *Storage Controler*.

- ***Storage Controller*** (SC): escrito en Java, provee el equivalente a la funcionalidad de *AWS Elastic Block Store* (EBS). Este se comunica con *Cluster Controller* (CC) y el *Node Controller* (NC) dentro de la arquitectura distribuida del *cloud* y maneja los volúmenes de bloques y *snapshots*. Si una instancia necesita escribir un disco de datos persistente utilizará un volumen EBS servido por el *Storage Controller* y el cual podrá disponer de interfaces para el sistema local, NFS, iSCSI y SAN. Normalmente, reside en el mismo *host* que el *Cluster Controller*.

- **Node Controller** (NC): escrito en C, es quien gestiona las instancias de las MV y los puntos de red virtuales, descargando las imágenes de la MV desde el OSG y creando cachés en el disco local.

Eucalyptus puede ser instalado desde los paquetes binarios en formato RPM de forma simple y fácil (utilizando el comando *yum*) o desde el código fuente (Github), los cuales se deberán configurar y compilar (tarea ardua y solo recomendada para administradores que necesiten adaptaciones especiales o integraciones con hardware específico). La figura siguiente muestra una arquitectura mínima (de demostración/experimentación) de un despliegue de Eucalyptus.



Eucalyptus incorpora los templates con las definiciones de MV de acuerdo a la terminología EC2 aunque algunas de ellas pueden requerir servidores específicos (no commodity hardware), por ejemplo (formato vCPU, MEM, DISCO) de uso general (t) t2.micro (1, 1, 160), t2.small (1, 2, 160), t2.medium (2, 4, 160), equilibrio entre recursos sin SSD (m) m1.small (1, 1.7, 160), m1.medium (1, 3.75, 410), m2.xlarge (2, 17.1, 420), m2.2xlarge (4, 34.2, 850), equilibrio entre recursos y SSD (m3) m3.medium (1, 3.75, 4), m3.large (2, 7.5, 32), optimizadas para cómputo (c) c1.medium (2, 1.7, 350), etc. Se debe tener en cuenta que en AWS instancias como t1, m1, c1/c3, entre otras, son consideradas de la generación anterior y han evolucionado a t2, m2, c3 respectivamente, y que la relación prestaciones/precio es mucho mejor en las instancias de las nuevas generaciones (lista de instancias de AWS actuales).

En las versiones actuales (4.3.x), Eucalyptus, puede ser instalado sobre CentOS 7 o RHEL 7 y sobre arquitecturas x86_64 únicamente y con KVM en los nodos (ver matriz de compatibilidad versión 4.3.1) y en las indicaciones/requerimientos generales en los cuales indica que no puede ser instalado sobre MV sino solo sobre máquinas físicas.

2.2.1. Instalación de Eucalyptus FastStart

Esta instalación permite ejecutar todos los servicios en una única máquina con el fin de experimentar con la infraestructura IaaS.

En una primera prueba se intentó ejecutar sobre una MV KVM (sin hacer caso a las indicaciones de los desarrolladores en cuanto a que solo se puede ejecutar en *bare-metal*), y se logró su instalación, pero el resultado no fue totalmente satisfactorio ya que (después de un tiempo considerable) se logró la puesta en marcha de todos los servicios, pero finalmente surgían problemas con la red que se debía configurar manualmente para así poder acceder a los servicios. Como resultado de esto se obtuvo una gran experiencia en los servicios de Eucalyptus, *cookbooks* de Chef, e instalaciones desasistidas que utiliza FastStart. Para desarrolladores, existe la posibilidad de utilizar una máquina virtual y Vagrant utilizando `eucadev` (*tools for Eucalyptus developers and testers*) que permite desplegar un *cloud* de Eucalyptus en una MV provista por Vagrant (o en una instancia de AWS u otro Eucalyptus) con un esfuerzo mínimo.

Finalmente, se decidió instalar una CentOS 7.3 (1611) con un procesador X86_64 y con extensiones de virtualización, 5GB RAM (luego se mostrará que con una instancia *m1.small* ya utiliza 4,4GB), 30GB de disco (pero puede ser la mitad para casos de pruebas) y con una tarjeta de red Ethernet (el *script* no instalará nada si es wifi) configurada en modo estático (IP, netmask, gateway, DNS). La instalación de CentOS 7 se realizó con al Minimal-ISO, se modificó el */etc/hostname* y se configuró el FQDN en */etc/hosts* para incluir la máquina. El *script* de FastSart detecta si se está ejecutándose el *Network Manager* y no continuará, por lo cual es necesario quitarlo:

```
systemctl disable NetworkManager.service
systemctl stop NetworkManager.service
yum remove NetworkManager
```

Se debe actualizar e instalar `ntp`, `bridge-utils`, configurar la zona horaria y crear una interfaz de *Bridge* (br0) que será utilizada luego (no es necesario configurarla):

```
yum update
yum install ntp bridge-utils
service ntpd start
ntpq -p (para verificar que todo funciona)
chkconfig ntpd on
date mmddyyhhmm (inicializar la fecha)
timedatectl set-timezone Europe/Madrid
brctl addbr br0
```

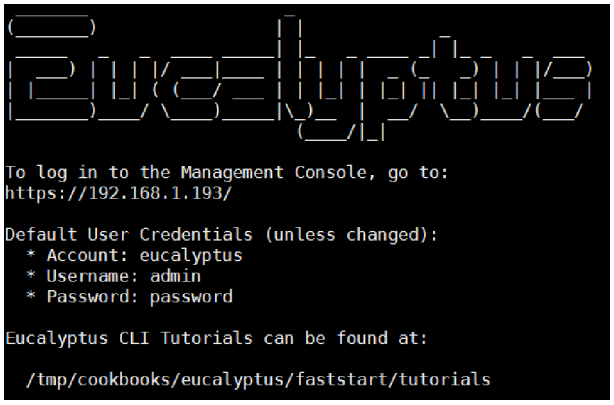
Después de reiniciar la máquina para actualizar los cambios y verificar que todo está correcto, ejecutar como *root*:

```
bash <(curl -Ls hphelion.com/eucalyptus-install)
```

Después de unas advertencias (por el ACPI/*sleep* y el tamaño del disco –sugiere al menos 100G–, pues sino indica que solo se podrán poner muy pocas instancias de MV en funcionamiento que en nuestro caso aceptaremos, ya que solo es una versión de pruebas), se iniciará la descarga e instalación de paquetes. Durante la instalación se han realizado una serie de preguntas para seleccionar la interfaz sobre la cual funcionará el *cloud* (se deberá escoger la interfaz antes configurada), el servidor de ntp, y seleccionar si se desea o no que se instalen servicios adicionales como *load balancer* e *image management* (10-15 minutos adicionales); en esta prueba se ha seleccionado que no. Desde otra máquina, por medio de una conexión `ssh`, se ha podido observar la evolución de la instalación en el archivo de log (indicado durante la ejecución) en un comando similar a:

```
tail -f /var/log/euca-install-mm-dd-yy-hh-mm-ss.log
```

La instalación pasará por **3 fases** y finalmente podrá alguna cosa como la imagen que se encuentra a continuación, (que es similar a la que se verá cuando se acceda como *root* por `ssh` una vez finalizada la instalación) que nos da la IP de la consola de administración, la cuenta, el usuario y *passwd* por defecto, así como la ruta a *scripts*/tutoriales para ejecutar determinadas acciones sobre el *cloud*.



El *sysadmin* también dispondrá de todos los comandos CLI (*euca2ootls*) que le permitirá gestionar todo el *cloud* desde la línea de comandos. Por ejemplo, para crear una nueva cuenta (*myaccount*) y asociarle un usuario (*admin*) y *passwd* (*pAssWoRd*):

```
euare-accountcreate -a myaccount
euare-useraddloginprofile --as-account myaccount -u admin -p pAssWoRd
```

Una vez finalizada la instalación, se observará que no se puede conectar a la URL indicada, primero porque en la configuración de la consola de administración (*eucaconsole*) indica en */etc/eucaconsole/console.ini* → *host=127.0.0.1* por lo cual se debería cambiar por 0.0.0.0 (o IP del cloud) y reiniciar el servicio con:

```
service eucaconsole restart
```

No obstante, veremos que si lo dejamos así y lo intentamos en la URL *localhost* también la conexión no es satisfactoria, y desde la documentación se puede observar que Nginx está haciendo de *proxy* pero que si se introduce la URL *http://localhost:8888* se podrá acceder a la consola donde una advertencia muestra que la conexión no es por el puerto 443 y no permite el acceso. Después de analizar y ver los errores de Nginx, se detectó que los problemas estaban en la carga de */etc/eucaconsole/console.crt* y son para la versión 4.3+ derivados de la protección de SELinux cuando está en modo *enforcing*.Si se miran las protecciones de los archivos:

```
ls -lZ /etc/eucaconsole/console.crt
   /etc/eucaconsole/console.key
-rw-------. eucaconsole eucaconsole system_u:object_r:eucaconsole_conf_t:s0
   /etc/eucaconsole/console.crt
-rw-------. eucaconsole eucaconsole system_u:object_r:eucaconsole_conf_t:s0
   /etc/eucaconsole/console.key
```

Se observa que no son correctas y debería ser:

```
matchpathcon /etc/eucaconsole/console.crt /etc/eucaconsole/console.key
/etc/eucaconsole/console.crt  system_u:object_r:cert_t:s0
/etc/eucaconsole/console.key  system_u:object_r:cert_t:s0
```

Para solucionarlo se debe ejecutar:

```
restorecon /etc/eucaconsole/console.crt /etc/eucaconsole/console.key
ls -lZ /etc/eucaconsole/console.crt /etc/eucaconsole/console.key
-rw-------. eucaconsole eucaconsole system_u:object_r:cert_t:s0   /etc/eucaconsole/console.crt
-rw-------. eucaconsole eucaconsole system_u:object_r:cert_t:s0   /etc/eucaconsole/console.key
```

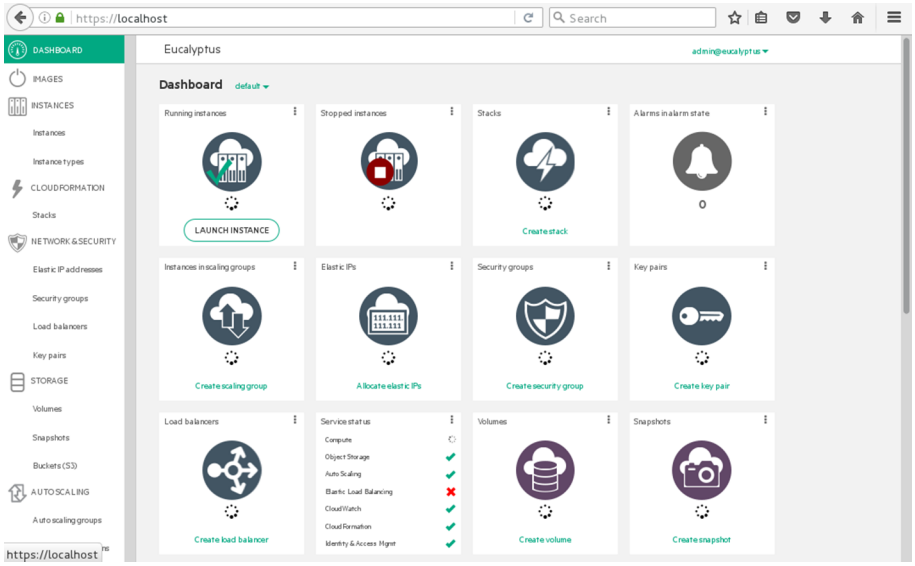
Y reiniciar el servicio.

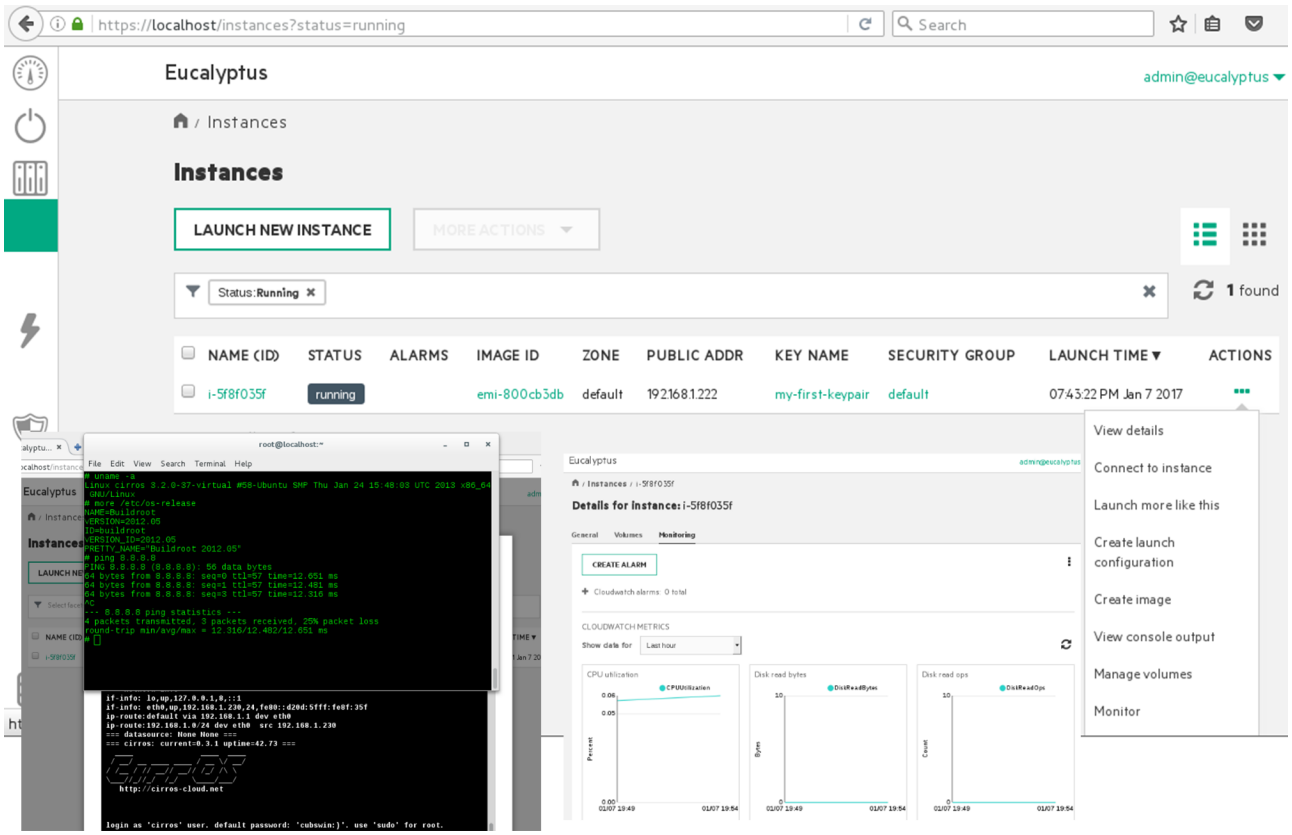
```
service eucaconsole restart
```

Para mayor información consultar la guía sobre cómo comenzar [Egs], cómo iniciarse en `euca2ools` [Ee2] y la guía de configuración de la consola de administración [Eec].

Una vez conectados a https://localhost/ e introduciendo la cuenta/usuario/*passwd* (ya sea por defecto o los creados nuevos), se accederá al *dashboard* y se podrá comenzar a configurar todos los aspectos del IaaS (imágenes, instancias, almacenamiento, red, usuarios/grupos, etc.).

Las figuras siguientes muestran el *dashboard* con el menú vertical, el despliegue de una instancia (con la imagen precargada que dispone la instalación, CirrOS), la visualización de la consola (desde el menú de *Actions*), la conexión `ssh` a la IP de la consola para verificar su funcionalidad y conectividad y la monitorización de la instancia a través de mismo menú *Actions*.





En el caso que se desee poner en marcha para producción, se deben observar los requisitos que deben cumplir las máquinas y planificar la infraestructura, la ubicación de los servicios, el modo de red, y luego configurar las dependencias,

instalar e iniciar su ejecución y finalmente registrar los servicios y ajustar el entorno (DNS, usuarios, almacenamiento, servicios de imágenes, balanceo de carga, etc.) [Edo].

En esta breve introducción/instalación de Eucalyptus solo se han mostrado los aspectos iniciales de toda la funcionalidad y prestaciones que dispone tanto para *cloud* privados como para híbridos (a remarcar la potencialidad de integrarse con AWS de forma muy fácil; queda para el lector experimentar con ella). Independientemente de las circunstancias del proyecto dentro de HP, esta plataforma es muy estable pero también muy compleja, dispone de una comunidad de desarrolladores eficientes con el soporte de los profesionales del proyecto HPHelion que aportan conocimientos y valor al software *open source* y cuenta con una extensa documentación. Probablemente, el comentario más frecuente sobre esta infraestructura es su complejidad y que el *sysadmin* deberá dedicarle tiempo y esfuerzo para disponer de una infraestructura funcional estable y configurada (cuesta ponerla en marcha, aún en versiones de prueba, para iniciados).

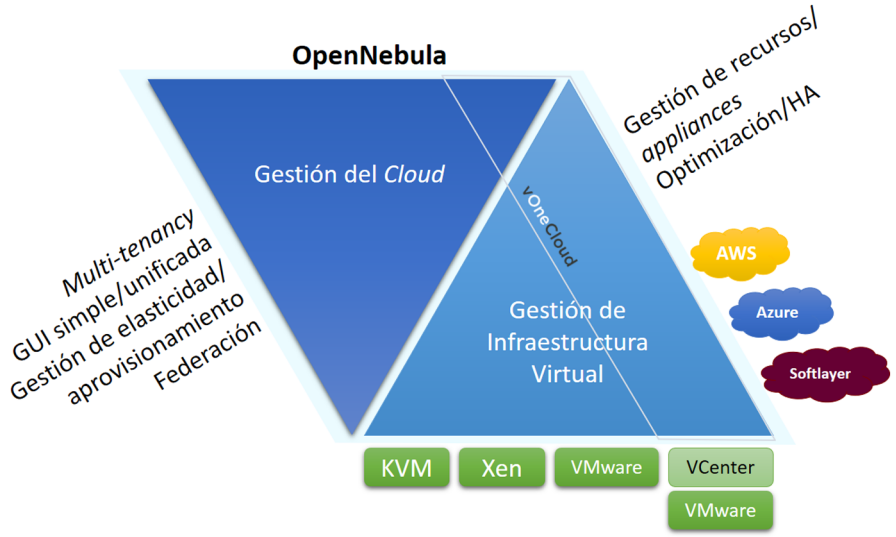
2.3. OpenNebula

Después de nueve años desde la fundación de proyecto *opensource* OpenNebula se ha consolidado como uno de los entornos *cloud* de mayor éxito y progresión. Entre sus usuarios se encuentran los entornos académicos/institucionales/de investigación (tanto para la formación/desarrollo en estas tecnologías como en producción), y las empresas con más de 100K de descargas de los repositorios del proyecto en el último año, más de 1.500 *clouds* conectados al *marketplace* y con decenas de instalaciones singulares con cientos de miles de *cores* gestionados por el entorno y con una comunidad muy activa en el desarrollo de la plataforma.

OpenNebula ofrece un entorno simplificado, pero con una gran funcionalidad y flexibilidad para la gestión integral de los centros de datos virtualizados que permite la configuración y administración de IaaS privados, públicos e híbridos. El desarrollo constante de la plataforma ha permitido dotarla de la interoperabilidad necesaria para que pueda integrarse fácilmente con los activos existentes de un CPD, evitando así la pérdida de inversiones y el bloqueo de proveedores.

Su funcionalidad se distribuye en tanto en la capa virtualización de centro de datos como en la de infraestructura *cloud*. En la primera, aporta herramientas para la administración de virtualización del centro de datos que permiten consolidar servidores e integrar los activos de IT existentes, así como la configuración del almacenamiento y la creación de redes.

En este modelo de despliegue, OpenNebula se integra directamente con los hipervisores utilizados (KVM, Xen o VMware ESXi) y tiene un control completo sobre los recursos físicos y virtuales, proporcionando funciones avanzadas de gestión de capacidad, optimización de recursos, alta disponibilidad y continuidad de negocio permitiendo además federar centros de datos, implementar *cloudbursting* u ofrecer portales de autoservicio para los usuarios. En la segunda capa, gestión del *cloud*, permite disponer de una plataforma para el aprovisionamiento *multi-tenant*, que funciona por encima de la administración de la infraestructura IT (como por ejemplo, VMware vCenter), y así satisfacer las necesidades de provisión, elasticidad y *multi-tenancy* para el aprovisionamiento de centros de datos virtuales, federación o *cloud* híbrido. Esto permite gestionar recursos internos integrados con *clouds* públicos, mientras que la infraestructura base es administrada por herramientas habituales en la gestión/operación de infraestructuras IT.



Entre las principales características que ofrece OpenNebula se pueden considerar: arquitectura, interfaces y código abierto a la comunidad, flexibilidad, interoperabilidad y portabilidad, estabilidad (apto para su uso en entornos de producción de clase empresarial), escalabilidad, *SysAdmin*-centrismo (control completo del *cloud*), sencillez y facilidad en implementar/operar/utilizar, y eficiencia. Todo ello permite al administrador de infraestructuras contar con una herramienta con la cual puede dar una respuesta rápida a las necesidades de infraestructura con redimensionamiento dinámico de los recursos físicos mediante la adición de nuevos *hosts* y el particionamiento dinámico de clústeres para satisfacer los requisitos de capacidad de los servicios. Por otro lado, y dado que permite la utilización de los recursos heterogéneos existentes, se puede disponer de un entorno de gestión centralizado de toda la infraestructura virtual y física distribuida, eliminando así los silos de aplicaciones. Esto representa un ahorro operativo con la consolidación de servidores a un número reducido de sistemas físicos, lo cual reduce el espacio utilizado, el esfuerzo de administración y los requisitos de potencia eléctrica (tanto para funcionamiento como para refrigeración). Y, además, gracias a su capacidad de interoperabili-

dad, permite ajustar las inversiones ya que puede ampliar la infraestructura a un *cloud* público en determinados momentos de carga pico sin tener que sobredimensionar la infraestructura propia para estos casos puntuales.

OpenNebula dispone de una distribución optimizada, llamada vOneCloud, para trabajar con instalaciones gestionadas por VMware vCenter. Esta distribución despliega una infraestructura *cloud* de forma simple sobre recursos gestionados por vSphere y vCenter Operations Manager agregando aprovisionamiento elástico y *multi-tenancy* de OpenNebula. vOneCloud es *opensource*, no requiere licencias y es totalmente configurable; puede ser adaptado fácilmente a las necesidades del centro de datos permitiendo crear un autoservicio de *cloud* encima de la infraestructura VMware (sin interferir) y aprovechando la inversión que se ha realizado en ella.

Entre los principales usos que se le dan a OpenNebula están la creación de *cloud* privados/públicos, *hosting* y HPC y *science clouds*. Como usuarios importantes se pueden mencionar (orden alfabético no exhaustivo): Akamai, Avalon, BBC, Blackberry, BonFire, Cesga, China Mobile, CloudTM, CSC, Csic, Csiro, Csuc, Dell, DGrid, EGEE, EGI, Esa, Fermilab, Fuze, Harvard Seas, IBM, LexisNexis, Nasa, Rentalia, RedIris, Reservoir, Sigma, TeleData, Unisys, Unity, Virtion, entre otros (ver lista completa).

El despliegue de un *cloud* puede ser una tarea difícil ya que existen muchos conceptos y componentes interrelacionados y el grado de familiaridad con ellos definirá la dificultad en la instalación/configuración/operación del *cloud*. Una pequeña guía que ayude a esta tarea podrá ser:

- **Elección del hipervisor**: OpenNebula da soporte a dos de los más utilizados, que son KVM y VMware (a través de vCenter). Después de tener el *cloud* bajo un hipervisor se pueden agregar otros, como LXC o Xen, que son soportados por la comunidad y en el catálogo de OpenNebula Addons.

- **Diseñar el *cloud***: es recomendable definir qué características, prestaciones, escalabilidad y disponibilidad tendrá el *cloud*. Para ello se pueden utilizar las guías *Cloud Architecture Design guides* que ayudarán en la planificación y arquitectura del mismo.

- **Instalar el *front-end***: este proceso es independiente del hipervisor y puede ser complementado con la configuración de un clúster de alta disponibilidad para OpenNebula (y así reducir el tiempo de fallos) o la instalación de MyQSL como alternativa a Sqlite para dar soporte a grandes infraestructuras.

- **Instalar los *hosts* de virtualización**: serán los nodos de virtualización (en los cuales se ejecutarán las MV) que pueden ser nodos KVM o vCenter.

En el caso vCenter, un *host* representa un clúster vCenter con todos sus *hosts* ESXi.

- **Integración con la infraestructura del centro de datos**: será necesario realizar la configuración de red (802.1Q VLANs, ebtables, *Open vSwitch* o VXLAN), el almacenamiento (LVM *datastore*, Ceph, Dev, o iSCSI *datastore*), así como la integración con el sistema de monitorización, *Virtual Machine* HA o PCI *Passthrough*.

- **Configurar los servicios**: básicamente autentificación (interna o externa a través de `ssh`, x509, `ldap` o *Active Directory*) y configuración/adaptación de las vistas para los diferentes roles de OpenNebula.

- **Definir como operará el *cloud***: definir el modelo de aprovisionamiento (sobre todo en grandes infraestructuras) para garantizar el nivel de aislamiento y de *multi-tenancy* deseado a través de la gestión de grupos y mecanismos de *Access Control List*. Estos, juntos con los roles, permitirán un control detallado sobre el acceso a los recursos, cuotas y contabilización de uso para su posterior cargo a través de la plataforma de facturación. También será posible definir VDC (*virtual data center*) a un grupo/s de usuarios como una unidad lógica de asignación que puede estar mapeada sobre diferentes recursos físicos distribuidos.

- **Administrar los recursos virtuales**: ello incluirá
  a) la gestión de las imágenes de MV almacenadas en catálogos (*datastores*) utilizadas para definir MV o compartirlas con otros usuarios,
  b) gestión de las redes virtuales (organizadas en catálogos) que proveerán la comunicación entre la MV a través de *virtual routers*,
  c) gestión de los esquemas de MV (*templates*) que son definiciones de MV que luego serán instanciadas como tales en el sistema.

- **Creación de la MV**: preparación y creación de la MV para los usuarios. OpenNebula utiliza la contextualización para enviar información a la MV durante su arranque (información de red, credenciales de acceso, etc.).

- **Instalación de componentes avanzados**: tales como EC2 Query y EBS para la interfaz con *cloud* públicos, OneFlow para el autoescalado, federación de diferentes OpenNebulas, gestión de la información desde/hacia la MV por medio de OneGate, integración con el *marketplace* para compartir, aprovisionar o utilizar imágenes, configuración del *cloud bursting* (Amazon EC2 y Microsoft Azure).

- **Integración con otros componentes** (para integradores y desarrolladores): a través de múltiples interfaces, OpenNebula puede ser integrado fácilmente con otras herramientas/recursos. Esta integración puede ser utilizando las API de la plataforma o desarrollando *plugins* que permitan interrelacionarse con recursos/plataformas tales como infraestructura (almace-

namiento, monitorización, red, autentificación, virtualización, *cloudbursting*...) o plataformas externas (facturación, portales autoservicio, monitorización).

2.3.1. SandBox

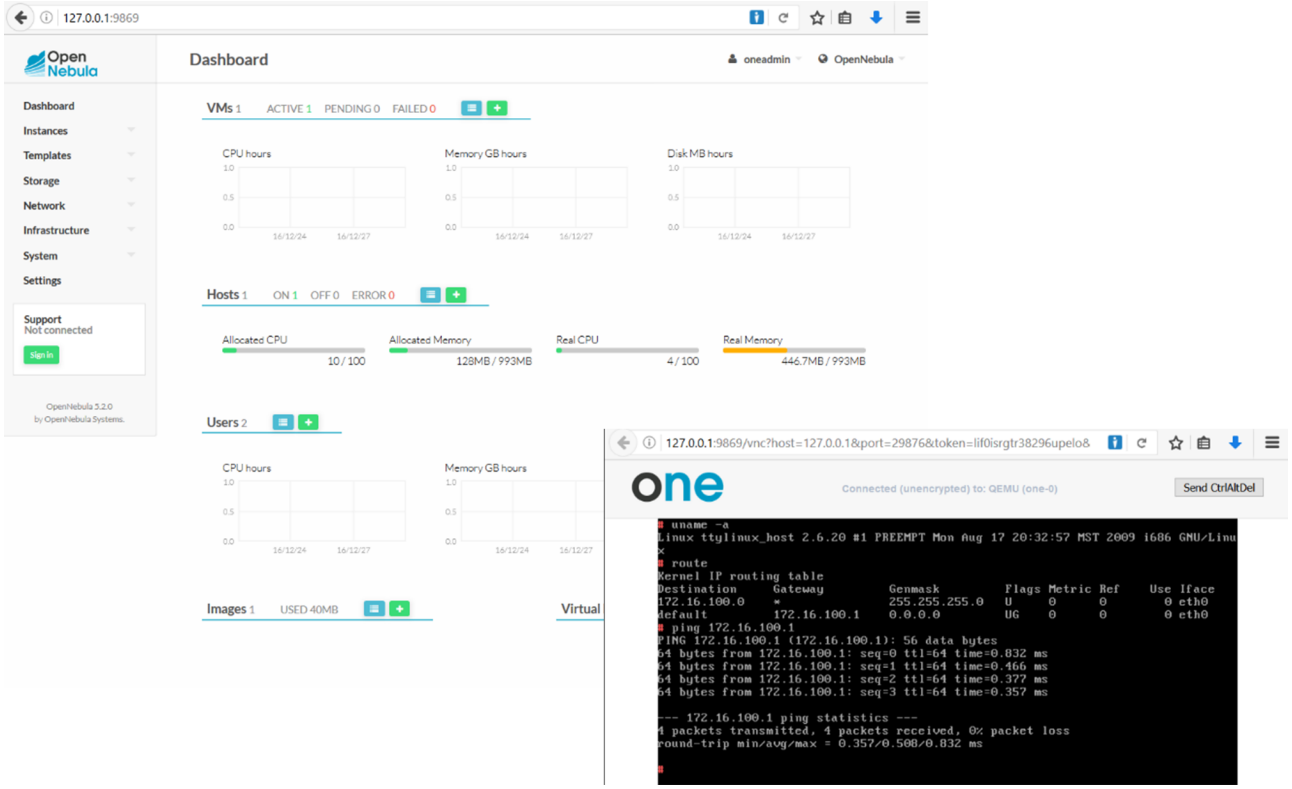
Una forma de tomar contacto con OpenNebula es utilizar la *appliance* que proveen los desarrolladores que permite familiarizarse con los conceptos involucrados y sin mayores esfuerzos, ya que la misma incorpora todos los elementos básicos del *cloud* OpenNebula en una única MV (que puede ser para Virtualbox o Amazon AWS). En esta prueba se utilizará la de VirtualBox, que tiene como SO CentOS 7 y preconfigurado el *front-end* OpenNebula 5.2.0 más un *host* de virtualización utilizando QEMU para ejecutar MV. Obviamente, esta instalación es con fines de prueba, pero ofrece un entorno completo donde conocer y experimentar con la potencialidad de la plataforma y opcionalmente agregar otros nodos físicos para construir un *cloud* a pequeña escala. Los usuarios podrán conectarse a OpenNebula *cloud*, manejar los recursos, lanzar instancias de MV y realizar toda la administración sin tener que configurar la infraestructura física. Los requerimientos serán de 1 GB RAM libre por MV, 10 GB de espacio de disco, un SO de 64 bits y con extensiones hardware de virtualización [One].

1) Instalación: en primer lugar descargar la *appliance* y luego, desde VirtualBox, importarla a través del menú *File* → *Import Appliance*. Luego se podrá poner en marcha y se verá el *prompt* ***one-sandbox login***: al cual se podrá acceder con usuario y *passwd* **opennebula**. Los siguientes pasos serán acceder a la interfaz gráfica y poner en marcha una MV.

2) Acceder a Sunstone (GUI): para conectarse a la interfaz gráfica, abrir un navegador sobre el *host* (donde se está ejecutando Virtualbox) y poner como URL *http://127.0.0.1:9869* (Virtualbox tiene mapeado el puerto en la interfaz NAT) introduciendo como usuario *oneadmin* y *passwd* **opennebula**. La pantalla que se mostrará es el *dashboard* de OpenNebula y con todas las opciones con una vista de *Admin*. Si no se desea ver todas las opciones se puede cambiar la vista –*View*– en la identificación del usuario oneadmin –arriba a la derecha– y seleccionar una, por ejemplo *Cloud*, que será la que verán los usuarios finales y que es mucho más simple (el usuario oneadmin podrá configurar las vistas y seleccionar qué verán los usuarios en cada una de ellas).

3) Crear un MV: en esta vista *cloud* (o en el *dashboard* de *admin* → *Instances* → *VM*) hacer un clic en el botón ✚, seleccionar el *template* (por ejemplo, ttylinux), modificar los parámetros si se considera necesario y hacer clic en *Create*. Esto desencadenará una serie de acciones hasta que la MV pase a estado de *Runing* y se gestione la MV: podrá acceder por VNC (usuario *root* y *passwd*

password), salvar el estado, reiniciar, parar, apagarla, etc. La imagen siguiente muestra el *dashboard* con la ejecución de la MV ttylinux y el estado del *cloud* y la interfaz VNC de la MV.



4) CLI Interface: desde la consola de la MV se podrán ejecutar diversos comandos como usuario oneadmin. Para ello, ejecutar `su oneadmin`.

```
onehost list         los recursos activos.
onevnet list         las redes activas.
onevnet show 0       parámetros específicos de la red id=0
oneimage list        las imágenes creadas
onetemplate list     los template de MV disponibles
onetemplate show 0   más detalles sobre el template id=0
```

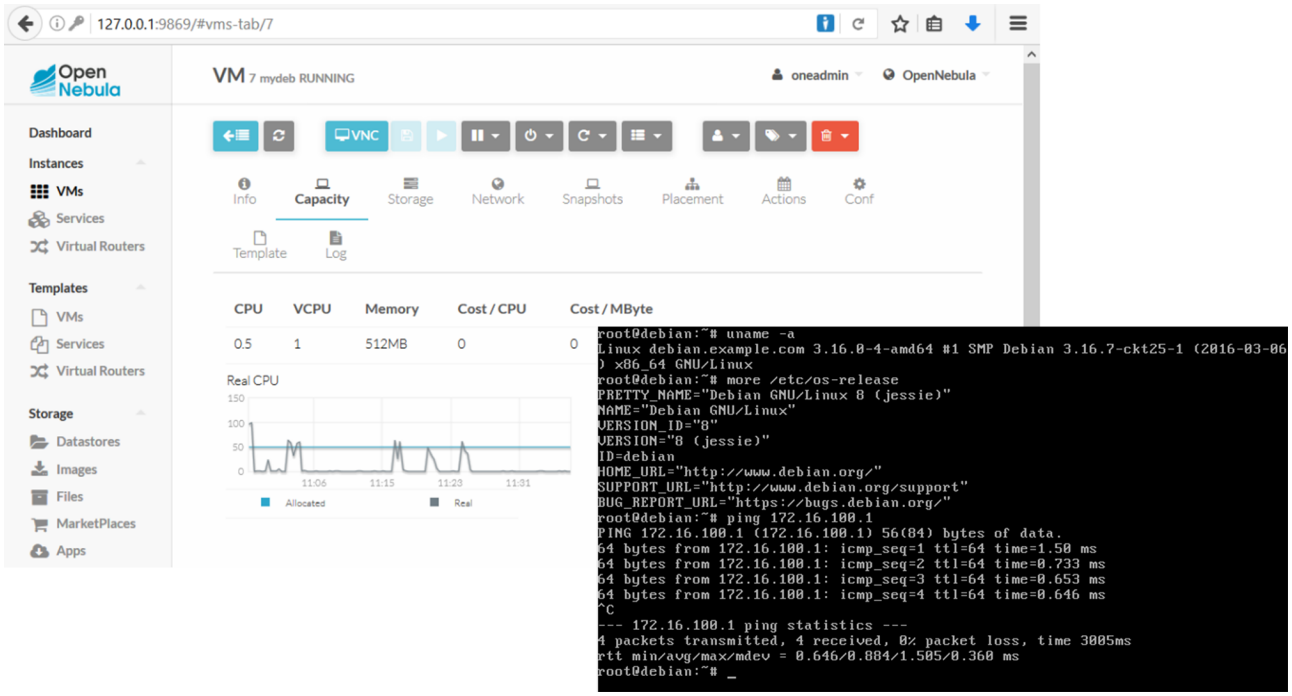
5) Agregar contenido al *cloud*: para agregar otra imagen de máquina virtual se puede ir al *Marketplace* (*Storage* → *Apps*) seleccionar una imagen y hacer un clic en el botón *OpenNebula*. En *Storage* → *Images* se podrá ver el estado como LOCKED (hasta que se termine de descargar que luego pasará al estado READY) y con lo cual ya se podrá crear una MV. En esta prueba se ha escogido una Debian 8.4 KVM y que tiene OpenNebula *contextualization package* instalado y al *root* se accede por llave pública. Para ello, se deberá antes introducir la llave pública del usuario *oneadmin* para que por contexto se introduzca en la máquina virtual y se pueda acceder al *root*. Para ello ejecutar sobre la consola (del *sandbox*):

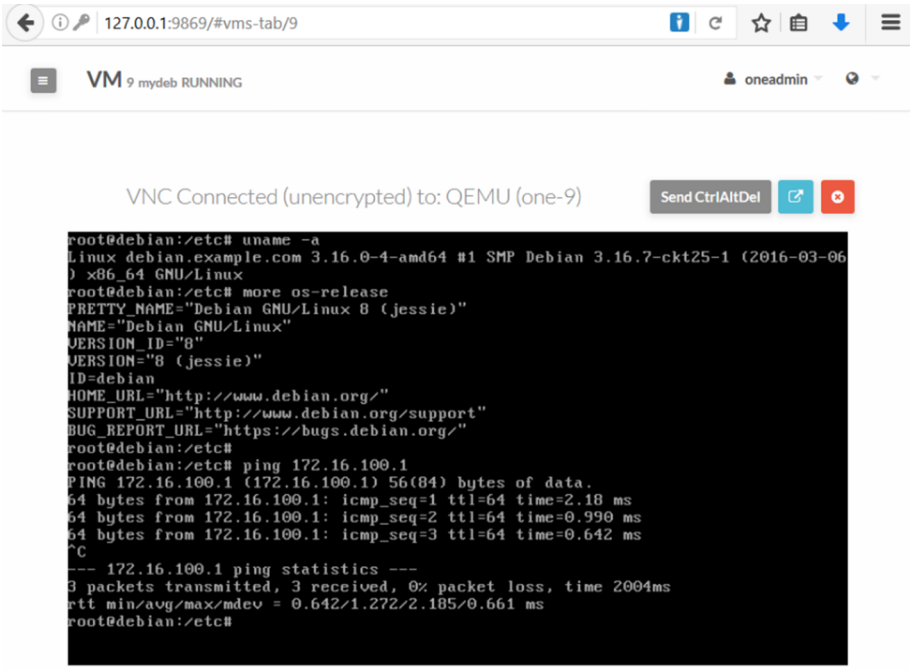
```
cd /var/lib/one/.ssh
```

```
echo SSH_PUBLIC_KEY='"' 'cat ./id_dsa.pub' '"' > tmpfile
oneuser update oneadmin -a tmpfile
rm -f tmpfile
```

Con ello se verá que en *Setting* → *info* aparece la *ssh-public-key* del usuario *oneadmin*. Es importante que las “ deben ir entre ‘ ’, es decir ‘ ” ’, ya que la llave debe quedar entre “ ”. En el template correspondiente a esta imagen se podrá observar que el valor de la llave que se incluirá en la instancia será SSH_PUBLIC_KEY = "$USER[SSH_PUBLIC_KEY]"]. Con esto ya se podrá crear la MV (importante seleccionar una red y que la CPU sea por ejemplo 0,5 y 1 VCPU), que se verá primero en un estado PENDING y luego pasará a estado de RUNING. Si se observa la *Conf* y el *Template* de la máquina virtual se podrá observar que se ha incluido la llave pública del usuario *oneadmin*.

Desde la consola (del *sandbox*) se podrá ejecutar `ssh root@172.16.100.201` (o la ip que se le haya asignado en la red virtual; se podrá consultar en la información de la MV) y tener acceso a la MV instanciada. Con ello ya se podría cambiar el *passwd* del *root* y poder acceder a través de VNC. Las figuras siguientes muestran la ejecución de la máquina virtual y la carga que representa, la consola con la conexión `ssh` y, luego de haber cambiado el *passwd* del *root*, la conexión por VNC.





Consultar información detallada en [One].

2.3.2. Instalación del *front-end*

Existen diferentes formas de instalar OpenNebula: añadiendo el repositorio OpenNebula de paquetes al sistema, bajando e instalando los paquetes desde el repositorio, o bajando el código fuente, compilarlo e instalarlo. En este caso, se seguirá la primera opción y se realizará sobre una MV KVM Debian 8.6 con virtualización anidada activada [Ofe].

1) **Modificar** el */etc/hostname* y */etc/hosts* para indicar un FDQN.

2) **Instalar** las llaves y los repositorios:

```
wget -q -O- http://downloads.opennebula.org/repo/Debian/repo.key | apt-key add -
echo "deb http://downloads.opennebula.org/repo/5.2/Debian/8 stable opennebula" > \
   /etc/apt/sources.list.d/opennebula.list
```

3) **Instalar** el software:

```
apt-get update
apt-get install opennebula opennebula-sunstone opennebula-gate opennebula-flow
/usr/share/one/install_gems
```

Durante la instalación se genera un *passwd* aleatorio para el usuario *oneadmin* en */var/lib/one/.one/one_auth* con el formato *oneadmin:<password>*. Editarlo y poner uno que se desee.

4) Reiniciar y verificar que todo funciona:

```
systemctl start opennebula
systemctl start opennebula-sunstone
su - oneadmin
oneuser show
```

Si todo es correcto mostrará información de usuarios/sistema

5) Conectarse a Sunstone por medio de la URL *http://<fontend_address>:9869* y como usuario *oneadmin* y el *passwd* que se haya escogido y se tendrá el *dashboard* de OpenNebula y se podrá operar con él de forma análoga que con el del *SandBox*.

Consultar la documentación sobre la estructura de los directorios y configuraciones adicionales [Ofe].

2.3.3. Instalación de un nodo

Como nodo para la ejecución de MV se agregará un nodo *host* externo que para esta prueba será una MV KVM Debian 8.6 con virtualización anidada activada (similar a la del *front-end*)[Oni].

Siguiendo los pasos indicados en la documentación:

1) Modificar el */etc/hostname y /etc/hosts* para indicar un FDQN.

2) Instalar llaves y repositorios:

```
wget -q -O- http://downloads.opennebula.org/repo/Debian/repo.key | \
apt-key add -
echo "deb http://downloads.opennebula.org/repo/5.2/Debian/8 \
 stable opennebula" > /etc/apt/sources.list.d/opennebula.list
```

3) Instalar el software y reiniciar:

```
sudo apt-get install opennebula-node
sudo service libvirtd restart
```

4) Configurar acceso por SSH sin *passwd*: OpenNebula Front-end se conecta con el nodo *host* utilizado SSH, por lo cual se debe distribuir la llave pública del usuario *oneadmin* y que se deberá almacenar en el archivo */var/lib/one/.ssh/authorized_keys* de todos los nodos. Durante la instalación del *front-end* se generaron las llaves *id_rsa*, *id_rsa.pub* y se actualizó el archivo *authorized_keys*, que ahora habrá que copiar a los nodos. También se deberá crear el *archivo known_hosts* para copiarlo a todos los nodos *host*.

```
ssh-keyscan <frontend> <node1>... >> /var/lib/one/.ssh/known_hosts
```

Es importante poner tanto las IP como los alias indicados en */etc/hosts* para evitar tener problemas con la identificación de las máquinas.

5) Copiar los archivos: sobre el nodo inicializar el *passwd* del usuario *oneadmin* y ejecutar desde el *front-end*.

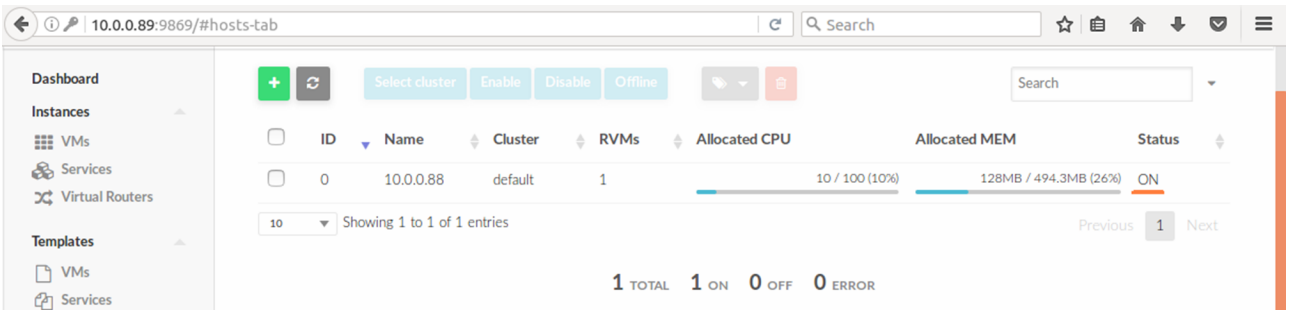
```
scp -rp /var/lib/one/.ssh <node1>:/var/lib/one/
```

6) Verificar que se puede conectar como usuario *oneadmin* utilizado `ssh` a los nodos y a al propio *front-end*.

7) Creación de una red: para la conexión a red con el *front-end* (para transferir y gestionar las MV) es conveniente utilizar una red privada. Hay varios modelos de redes, pero la más simple es utilizar un *bridge* que incluya el dispositivo físico y que será el utilizado por OpenNebula para gestionar las MV. Por ejemplo, en un *host* típico se tendrán dos dispositivos de red, uno para las direcciones públicas (por ejemplo, eth0) y otro (por ejemplo, eth1) para la LAN privada, y por lo tanto se tendrán dos *bridges* (br0 y br1) a los cuales se les puede dar el nombre que se desee, pero se debe mantener en todos los *hosts*.

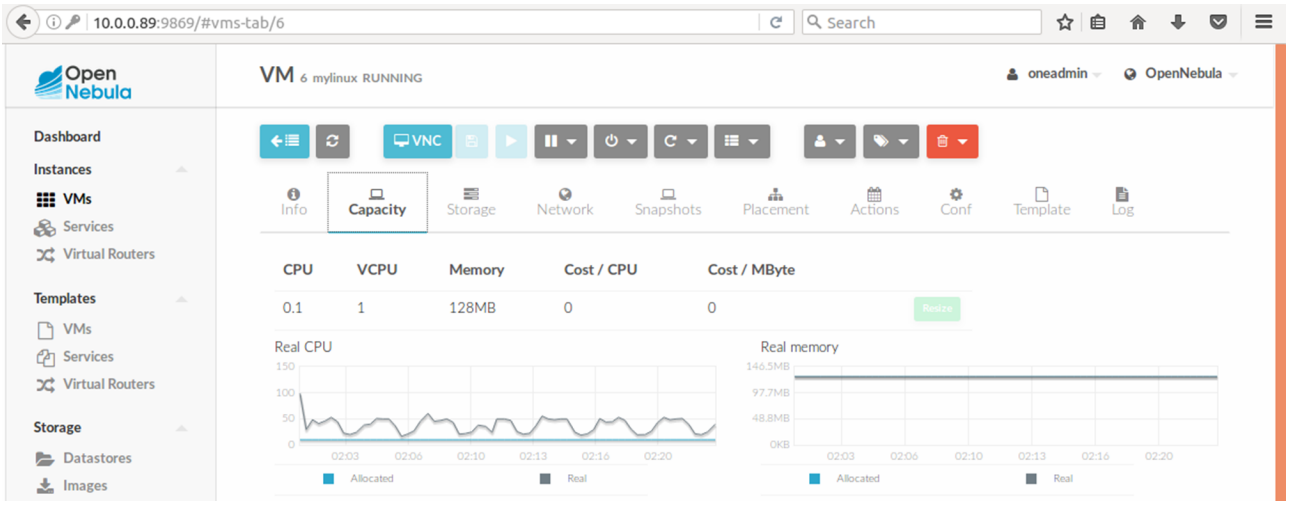
8) Almacenamiento: se utilizará el propio del *front-end* en el sistema de archivos, pero se podría configurar otro tipo de almacenamiento como Ceph, NFS, LVM, etc.

9) Agregar el nodo *host* a OpenNebula: en *Infrastructure* → *host* hace clic en ✚ y agregar los datos del nuevo *host*. Después de reactualizar la interfaz ↻ si todo es correcto se deberá ver el estado del nodo en ON, como se muestra en la figura.



10) A continuación, se podría verificar **la instalación** y cargar una imagen desde el *marketplace* y hacer una instancia sobre el *host*. Para ello, hay que crear una red primero y modificar el *template* de la imagen cargada para adecuarla a la instalación (básicamente escoger el disco). La figura siguiente muestra el

resultado de la ejecución sobre el *host* de la máquina *ttylinux* descargada desde el *marketplace* e instanciada sobre el *host* (no sobre el *front-end* como ocurría en el *Sandbox*).



Como se ha podido observar, OpenNebula es una plataforma con una gran cantidad de opciones y funcionalidad en las cuales el *sysadmin* deberá analizar y experimentar. Los siguientes pasos pueden ser la configuración del almacenamiento, y la configuración de la red ya que son los puntos principales para adecuar el *cloud* a las necesidades de funcionamiento y a la infraestructura disponible. A continuación podría ser la guía de operaciones, y temas complementarios como la configuración de la autentificación para integrarlo con LDAP/AD, y la configuración de Sunstone para aspectos específicos de la GUI.

2.4. OpenQRM (Community Edition)

OpenQRM es una empresa que desarrolla *openQRM Data Center Management and Cloud Computing Platform* en dos versiones: Community (*open source*) y Enterprise Editions (bajo licencia) y ofrece servicios y soporte en su ámbito de experiencia (*cloud computing* y *datacenter management*). Las diferencias entre las plataformas se pueden consultar en *OpenQRM Comparison* (incluidos precios y soporte) donde la versión Community es la adecuada gestionar pequeñas y medianas instalaciones, ya sea con fines comerciales o no, mientras que las versiones *Empresariales* disponen de todos los componentes necesarios para gestionar grandes centros de datos con eficiencia y alta disponibilidad.

La filosofía de openQRM está basada en la división de la plataforma en una parte *core*, que implementa un entorno base con la funcionalidad mínima y de interconexión, y una serie de módulos (*plugins*) que son administrados por el *core* e implementan una funcionalidad específica (por ejemplo, virtualización, imágenes, almacenamiento, monitorización...).

Esto representa unos beneficios importantes como desarrollo rápido y robusto, fácil integración a través de API, alta fiabilidad y control de errores bien localizados, escalabilidad, ampliación fácil con la incorporación de nuevos *plugins* y actualización eficiente, entre otras. Esta división de funcionalidad (*core* + *plugins*) permite una gran adaptación a las necesidades del cliente, ya que para un determinado CPD, solo se incluye la combinación de módulos necesaria, además de OpenQRM *core* para gestionarlos.

Entre las **principales características** de openQRM se tiene: separación completa del hardware (servidores físicos/máquinas virtuales) del software (imágenes de servidor), el hardware solo es usado como un recurso de cómputo y puede ser reemplazado fácilmente sin necesidad de adaptar o reconfigurar totalmente el servidor/imagen de servidor, soporte para diferentes tecnologías de virtualización (permite MV VMWare, Xen, KVM, OpenVZ, LXC, Docker...), migraciones P2V (físico a virtual), V2P (virtual a físico) y V2V (virtual a virtual), configuración automática de monitorización a través de Nagios/Collectd, soporte para alta disponibilidad, imágenes de servidor preparadas para desplegarse (almacén de imágenes), administración de almacenamiento integrado (soporta NAS-NFS, iSCSI, LVM...), entre las más importantes.

Su instalación es un proceso simple pero su configuración es compleja, aunque está bien documentada.

1) Documentación: Para esta prueba se ha tomado como base la documentación y básicamente la *Install openQRM on Debian* (1) y *Virtualization with KVM and openQRM on Debian* (2).

2) Entorno utilizado: Las pruebas se han realizado sobre una máquina virtual KVM con una IP propia (*bridged*) y 32 GB de disco en la cual se ha instalado Debian 8.6 solo con openSSHServer (se han realizado pruebas con Ubuntu 15.x/16.x, pero no han sido satisfactorias dada la relación con php5 de OpenQRM y que este paquete ha sido reemplazado por php7 en estas versiones; es posible hacerlo, pero se han de modificar *scripts* de instalación). Es importante, sobre Debian, tener al menos tres particiones; en este ejemplo se ha creado una para la partición *root* (/) de 20GB, otra (lógica) de 2 GB para el *swap*

y una tercera (primaria) de 10 GB para las imágenes de las máquinas virtuales a la cual se le ha dado formato e inicializado con ext4 pero sin montar. Luego de instalado se ha verificado la conexión a internet y se ha ejecutado:

```
apt-get update && apt-get upgrade
```

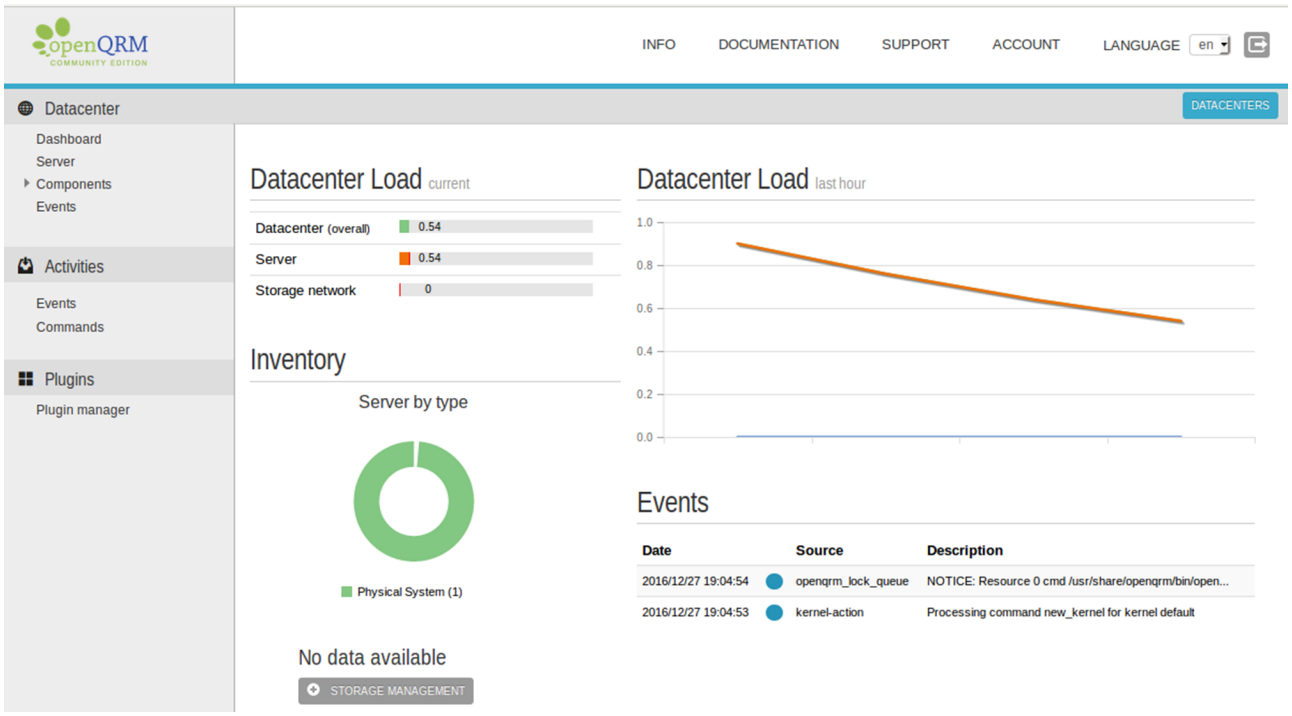
3) OpenQRM: Se ha descargado la versión 5.3.1 Community Edition desde la página web y después de rellenar todos los datos (también se pueden hacer las pruebas con la versión 5.2.1.1 de Sourceforge).

4) Instalación: Siguiendo los pasos indicados en (1), como *root*, se ha ejecutado

```
tar xvzf openQRM-3.5.1-Community-Edition.tgz
cd openQRM-3.5.1-Community-Edition
./install-openqrm.sh
```

Después de instalar/configurar una considerable cantidad de paquetes, el instalador solicitará el *passwd* para el usuario *root* de Mysql que se seleccione el tipo de servidor Postfix para el correo (seleccionar *Local Only* y proveer un dominio de mail), y el *passwd* de Nagios. Finalmente, la instalación proveerá la URL de conexión *http://ip_máquina/openqrm* y donde el *usuario/passwd* será *openqrm/openqrm*.

Accediendo a *http://ip_máquina/openqrm con el usuario/passwd* indicados, se realizará la última configuración (dispositivo de red, base de datos –Mysql– y usuario/*passwd* para ésta: *root* y el *passwd* introducido durante la instalación respectivamente). Con ello ya se tendrá OpenQRM instalado y preparado para configurar los recursos. La imagen a continuación muestra el *dashboard* del mismo.

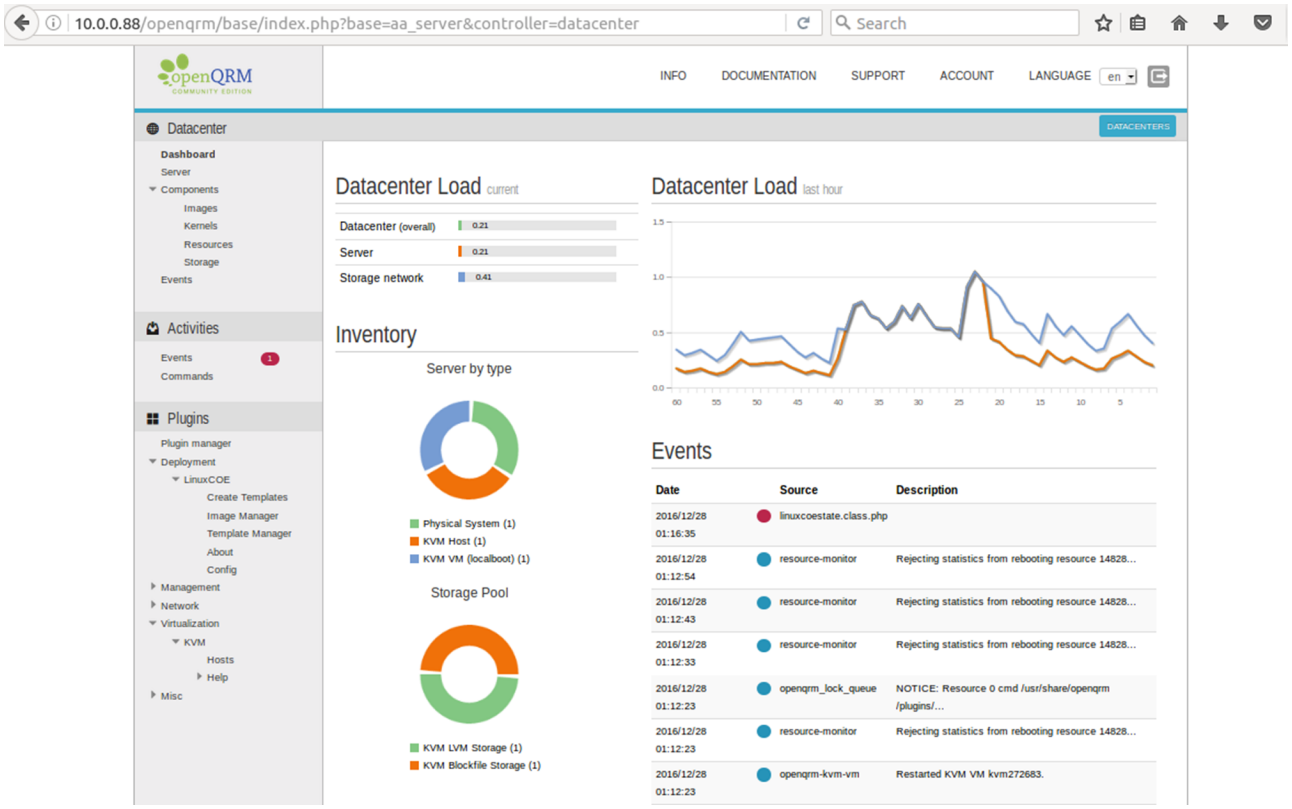


5) Instalación de máquinas KVM sobre OpenQRM: En la documentación (2) se tiene una imagen de la arquitectura QRM y cómo están relacionadas las diferentes partes. Siguiendo esta documentación, se han añadido los *plugins* en el *Plugin Manager* y en este orden ya que hay precedencias: *dns, dhcpd, tftpd, network-manager, local-server, device-manager, noVNC, sshterm, linuxcoe, kvm*. Posteriormente se han creado los siguientes recursos: *Host Object, LVM Group, Network Bridge, LinuxCOE* y *KVM Virtual Machine*.

Si bien está correctamente explicado y con imágenes paso-a-paso, la secuencia de pasos es compleja ya que existen muchas opciones y, en un primer contacto con la plataforma, se puede hacer pesado para un administrador que se inicia en esta plataforma. Cuando se finaliza la configuración, se puede conectar con noVNC a la MV creada y se puede realizar el *install* del sistema operativo; cuando este termina, se puede acceder a él y verificar que todo funciona correctamente.

En la documentación mencionada existen pasos adicionales para instalar el *openqrm-local-vm-client* sobre la máquina desplegada que permitirá recoger estadísticas y realizar gestiones internas sobre ella. También existe un apartado interesante que es cómo crear nuevas MV a través del clonado.

La imagen siguiente muestra el *dashboard* de OpenQRM con una MV funcionando y gestionada por la plataforma.



El siguiente paso en un entorno IaaS sería instalar el *plugin* de *cloud* en OpenQRM y configurar una infraestructura *cloud*; en esta edición (Community) puede ser realizada como *cloud* privado sobre KVM. La documentación *Cloud Computing with openQRM* es una guía detallada de cómo realizar este despliegue, pero dado los recursos que consume se recomienda hacer las pruebas con un servidor físico y por lo menos 100 GB de disco.

Como se ha podido observar, la plataforma OpenQRM es un entorno de tipo empresarial que, incluyendo la versión Community, dispone de una gran cantidad de funcionalidad y prestaciones para satisfacer las necesidades actuales de los CPD y IaaS. Es importante tener en cuenta que la formación del personal de administración en esta plataforma es muy importante, ya que sus posibilidades y complejidad son elevadas y es por ello que se necesita *sysadmins* con dedicación y experimentados para administrarla, siendo la versión Community un punto de inicio muy atractivo para administradores noveles o no familiarizados con entornos de este tipo.

2.5. OpenStack

OpenStack es un sistema operativo *cloud opensource* que controla un conjunto (generalmente grande o muy grande) de nodos de cómputo, almacenamiento y redes a través de un centro de datos gestionado por un panel de control (*dashboard*) que permite a los administradores actuar sobre los recursos y a los usuarios aprovisionarlos; todo ello a través de una interfaz web.

OpenStack cuenta con un extenso repertorio de integradores/proveedores/consultores/comunidad que, sobre la base de OpenStack, proveen soluciones/formación/integraciones adaptadas para las empresas de todo tipo que deseen disponer de un *cloud* privado/híbrido o que quieran dar servicios a terceros basados en esta tecnología (*OpenStack Marketplace*).

En OpenStack hay dos partes esenciales que son:

a) *Compute*

Los recursos de cómputo son accesibles a través de las API para desarrolladores que crean aplicaciones en el *cloud* y por medio de interfaces web para administradores y usuarios. La arquitectura de cálculo está diseñada para escalar horizontalmente en hardware estándar, permitiendo reducción de costos y reutilización de la infraestructura disponible o la compra de hardware no específico (*commodity cluster computing*).

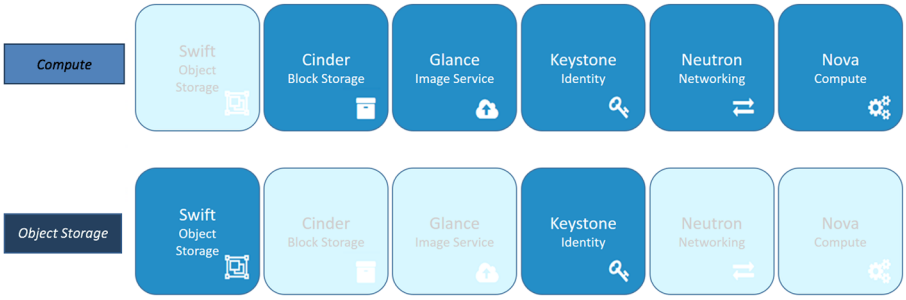
b) Object storage

Este módulo se puede implementar independientemente del *cloud* o junto con otras plataformas. Es el responsable para implementar almacenamiento escalable y eficiente ya que proporciona una plataforma de almacenamiento distribuida y accesible a través de una API, que se puede integrar directamente en las aplicaciones o utilizarse para la copia de seguridad, como archivo de datos.

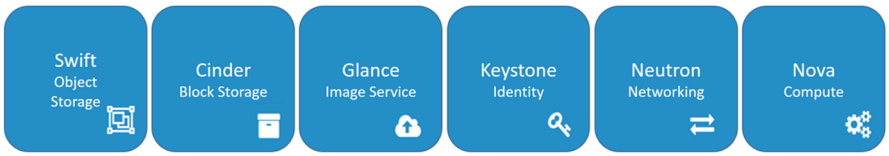
Compute + Object storage

Estos módulos juntos (*Compute and Object Storage*) proporcionan el conjunto completo de servicios básicos de OpenStack y API abiertas, ofreciendo un entorno robusto con capacidades de *cloud computing*, almacenamiento horizontal para la gestión de imágenes, instantáneas, copias de seguridad, máquinas virtuales y dispositivos de bloque [Odo].

Esto se traduce en unos servicios base (*core services*) que son:



Y que conjuntamente tenemos los módulos básicos (*core*) de OpenStack.



Para analizar las **posibilidades** de OpenStack se puede comenzar con:

- ***Cloud* público**: desde el MakerPlace se puede consultar una que satisfaga las necesidades de prueba y costo para acceder rápidamente a uno de los diversos *clouds* en producción que existen en el mundo (en su gran mayoría será necesario una tarjeta de crédito, aunque sean en modalidad de prueba) como por ejemplo TryStack.org.

- **Entorno de desarrollo local**: DevStack es una excelente opción para instalar y ejecutar un *cloud* OpenStack (incluido un portátil o una MV) que permite a los usuarios potenciales de este entorno cuáles son las potencialidades del mismo y las posibilidades que tiene.

- **Otro proyecto interesante** a tener en cuenta es Kolla que, mediante Docker y *playbooks* Ansible, proporciona contenedores y herramientas de despliegue para la puesta en marcha de OpenStack (ver *Quick Start*). Estas herramientas simplifican la creación de contenedores Docker (uno por cada uno de los servicios independientes de OpenStack) y también permiten la orquestación para desplegar dichos contenedores y así obtener una infraestructura OpenStack funcional y manejable.

- **Entorno de desarrollo/producción local**: Utilizando algunas distribuciones, en particular como Mirantis o Ubuntu Autopilot, se podrá instalar un entorno escalable que permitirá añadir servidores y almacenamiento para luego poder ponerlo en producción. En algunos casos, se necesitará una infraestructura mínima que impondrá determinadas condiciones hardware (por ejemplo, Ubuntu Autopilot necesita 5 máquinas: 1 para *MAASserver*, 1 para Autopilot, 3 o más para el *cloud* y *switch* dedicado para la LAN privada del *cloud*).

- También se pueden seguir las instalaciones para las distribuciones habituales a través de las guías propuestas por OpenStack u otros instaladores como RDO (sobre VBox) o OpenStack on Ansible.

Para el aprendizaje y experimentación con OpenStack se pueden también utilizar los Training Labs que permiten tener un entorno de pruebas en poco tiempo a través de MV en Virtualbox o KVM.

2.5.1. DevStack

DevStack es un *script* (extensible) para aprovisionar de forma simple un entorno de desarrollo OpenStack, aunque también se puede utilizar como herramienta para iniciar/ejecutar los servicios de OpenStack o probar sus parámetros. Se debe tener en cuenta que DevStack no ha sido pensado como instalador general de OpenStack; sin embargo, ha evolucionado para recoger un gran número de opciones de configuración, plataformas y servicios (más allá de lo que se pensó al inicio), aunque existe un conjunto de ellas que nunca se han probado.

Los elementos soportados son:

- SO: Ubuntu (LTS más el desarrollo actual), Fedora (versión actual más anterior), RHEL: versión actual (*major release*). Otras plataformas: no se asume, pero pueden ser incluidas.

- Base de datos: MySQL, PostgreSQL (incluidas en los paquetes del SO).

- Queues: Rabbit, Opid (incluidas en los paquetes del SO).

- Servidor web: Apache (incluidas en los paquetes del SO).

- Red OpenStack: (los valores por defecto Nova Network, opcionalmente se puede usar Neutron). Nova Network: FlatDHCP (en el caso de seleccionar Neutron se utiliza una configuración cercana a la original (FlatDHCP) utilizando linuxbridge o OpenVSwitch).

- Servicios: los servicios configurados por defecto son: Identity (*keystone*), Object Storage (*swift*), Image Service (*glance*), Block Storage (*cinder*), Compute (*nova*), Networking (*nova*), Dashboard (*horizon*), Orchestration (*heat*). El resto no es incluido directamente, pero se puede anexar en *stack.sh* utilizando el mecanismo de *plugin* para llamar a los *scripts* de inicialización y puesta en marcha de los nuevos servicios.

- Configuración del nodo: *single node* (multi-node generalmente no se tiene en cuenta por los servicios *core*, aunque se pueden probar en configuraciones mínimas).

- *DevStack exercise scripts*: si bien estos ya no se utilizan como pruebas de integración (estos han pasado al proyecto de test de Openstack Tempest), todavía se mantienen como muestras de cómo usar OpenStack desde la línea de comandos y para pruebas operacionales rápidas.

Los scripts de DevStack son *open source* también y debe tenerse en cuenta que su instalación realizará cambios importantes en el sistema durante la instalación, por lo cual se aconseja instalarlo sobre un sistema o MV dedicado a este propósito. En el presente caso de uso se realizará sobre una máquina virtual sobre KVM (no es necesario que tengan virtualización anidada, pero es aconsejable por cuestiones de rendimiento).

1) Crear una MV e instalar Linux. Devstack soporta Ubuntu 14.04/16.04, Fedora 23/24, CentOS/RHEL 7, así como Debian y OpenSUSE. En este caso se ha escogido Ubuntu 16.04 ya que es la más probada y hay mucha documentación relacionada de las experiencias de diferentes usuarios.

2) Agregar un usuario: Devstack deberá ejecutarse como usuario *no-root* con `sudo` habilitado. Como *root* (o con `sudo`) hacer:

```
adduser stack
echo "stack ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
su - stack
```

3) Descargar DevStack (como usuario *slack*):

```
git clone https://git.openstack.org/openstack-dev/devstack
cd devstack
```

4) Crear un archivo con la información de configuración (los 4 *passwd*: *admin*, DB, Rabbit, Service) en */home/stack/devstack/local.conf* (reemplazar pAssWoRd con el que se desee):

```
vi local.conf
       [[local|localrc]]
       ADMIN_PASSWORD=pAssWoRd
       DATABASE_PASSWORD=$ADMIN_PASSWORD
       RABBIT_PASSWORD=$ADMIN_PASSWORD
       SERVICE_PASSWORD=$ADMIN_PASSWORD
```

5) Iniciar la instalación (cabe tener paciencia, pues tarda unos 60 minutos, dependiendo de la conexión a internet y de la máquina).

```
./stack.sh
```

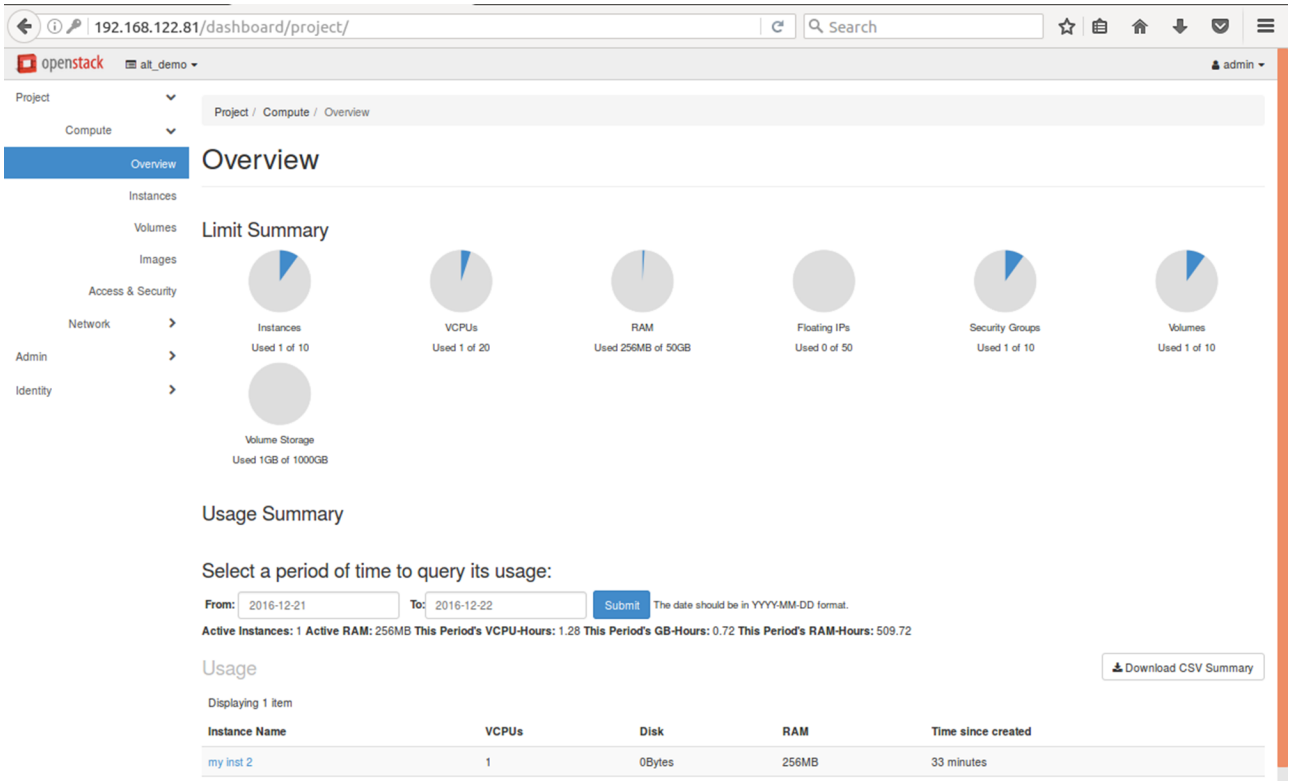
6) Una vez finalizada la instalación (se habrán instalado *keystone*, *glance*, *nova*, *cinder*, y *horizon*), se indicará la IP de conexión al *dashboard* (algo como *http://ip_server/dashaboard*) de entrada (usuario *admin*, *passwd* el que se ha definido anteriormente). A partir de ellos se puede acceder a la GUI de OpenStack, crear una red, y luego instancias de MV y luego administrar estas, los volúmenes y comprobar la funcionalidad de las mismas.

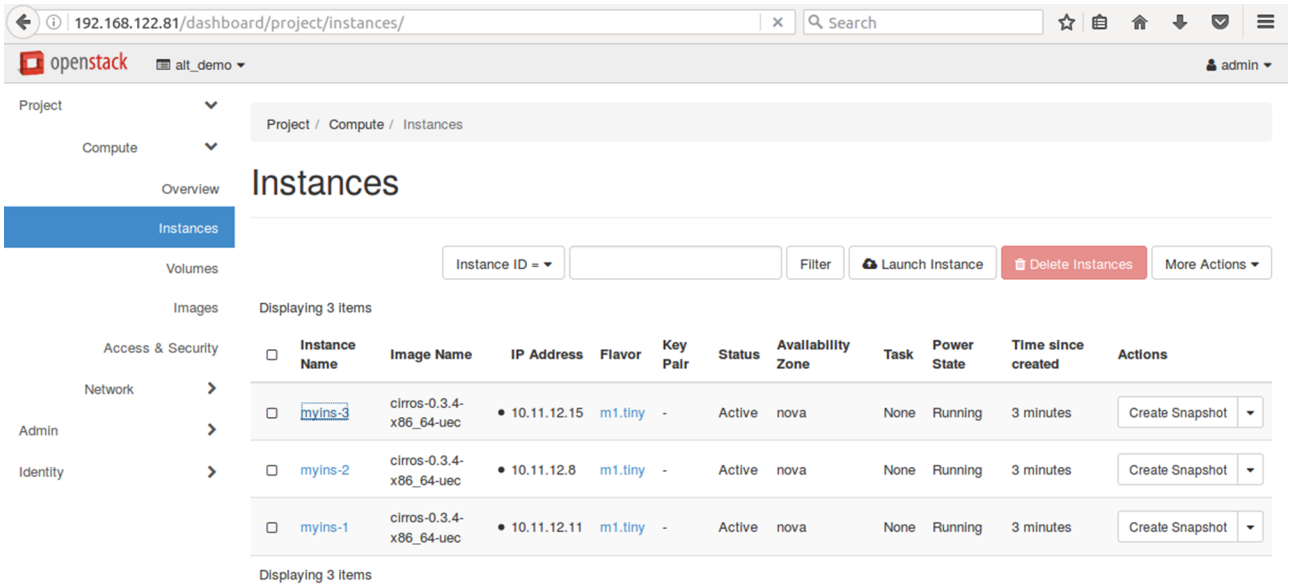
7) Pasos adicionales: se puede ejecutar *source openrc* en el Shell y utilizar la línea de comandos para manejar Devstack. También ir a `cd /opt/stack/tempest` y ejecutar los test de tempest que ha sido configurados para Devstack o hacer analizar la información de servicios mediante `screen` y hacer cambios.

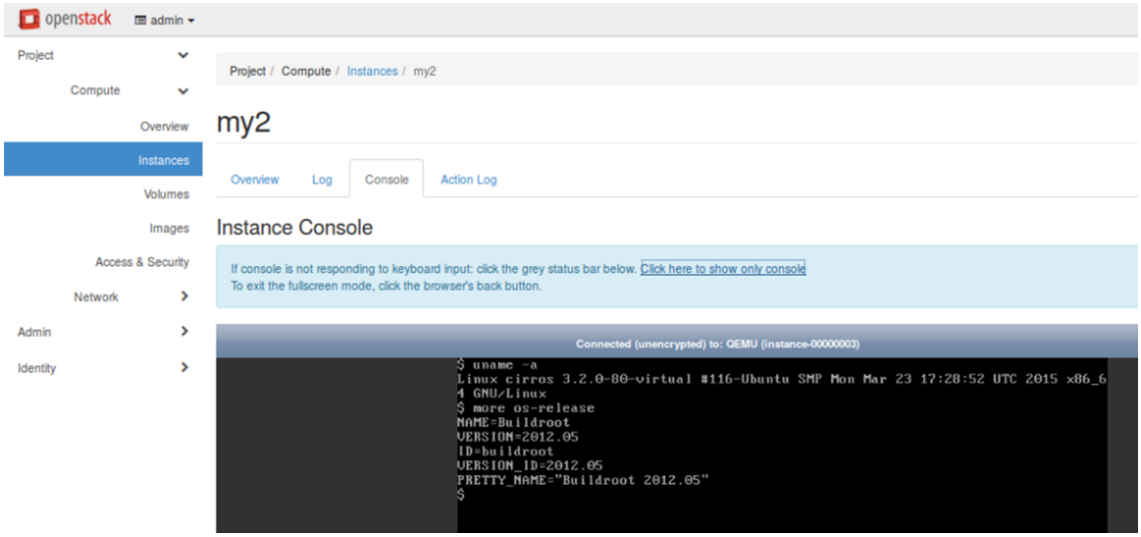
8) También es interesante aprender sobre la configuración del sistema para adaptar *devstack* a nuestras necesidades, incluidos cambios en el sistema de red por defecto para adecuarlo a nuestro entorno.

9) Guías adicionales sobre DevStack: All-In-One Single VM, Multi-Node Lab, Using DevStack with Neutron Networking, Configure DevStack with KVM-based Nested Virtualization, Nova and DevStack, Plugins.

Las dos figuras siguientes muestran el entorno GUI de Openstack desde el panel de control y con diferentes instancias de MV en ejecución de esta prueba y la consola de una instancia.







Como se puede apreciar, OpenStack despliega una infraestructura IaaS muy potente y con altas prestaciones, pero a su vez compleja y que necesita una inversión en tiempo y recursos para conocer y aprovechar las posibilidades y opciones que ofrece.

2.5.2. Kolla

La comunidad de Cisco-DevNet en los *Leaning Labs* ha generado una máquina virtual, basada en Kolla, con todos los contenedores que se ejecuta sobre Virtualbox y que junto con la documentación sobre la configuración e instalación se encuentran en CiscoLive Berlin 2016 Kolla Liberty VM - Box. Esta MV está basada en la versión Liberty de Openstack, pero, en esta prueba, se utilizará una desarrollada en un *Learnig Lab* actualizado con la versión posterior Mitaka (en este caso hay que registrarse en Cisco, pero es gratuito). El único inconve-

niente de estas MV es que necesitan una gran cantidad de memoria RAM (8GB sugerida por los desarrolladores, pero en esta prueba se ha realizado con 6 GB –sobre un *host* de 8 GB– y ha funcionado correctamente) [Cll]. Los pasos para instalarlo (se han realizado las pruebas tanto en Linux como en Windows):

1) Descargar el *appliance* ciscolive_kolla_mitaka.ova, instalar Virtualbox (en este caso se ha utilizado 5.1.10) e instalar el Extension Pack de Virtualbox (verificar que ambos paquetes sean de la misma versión). Deshabilitar VPN y asegurarse de tener conexión a internet en el *host*.

2) En esta instalación, OpenStack utilizará dos adaptadores de *red Host-Only* y uno *Nat-Network*. Desde *Preferences* → *Network* crear los tres adaptadores con los siguientes parámetros:

- *NAT Networks*: 10.100.100.0/24, verificar que *Enable Network* y *Supports DHCP* están marcados.

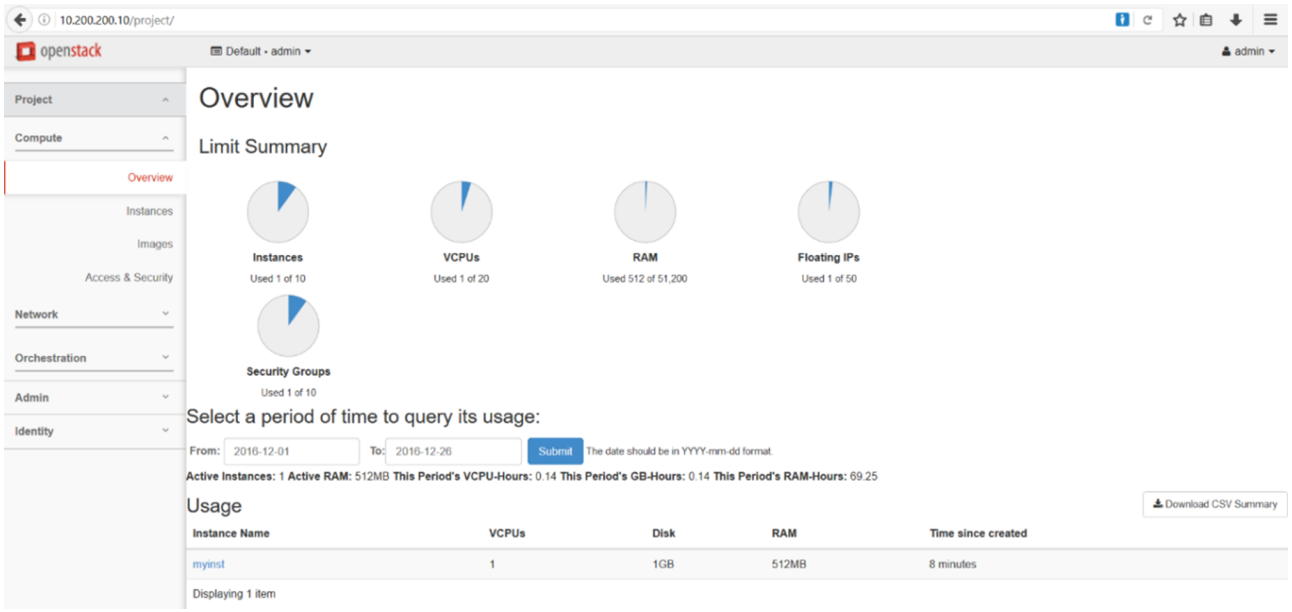
- *Host-only Networks*: crear dos adaptadores (en Linux se llamarán *vboxnet0/1* pero Windows les asignará un nombre diferente, por ejemplo,*VirtualBox Host Only Ethernet Adapter 2/3*). La configuración para el primer adaptador (*vboxnet0/VirtualBox Host Only Ethernet Adapter 2*) deberá ser: IP 10.200.200.1, *netmask* 255.255.255.0. Habilitar el *DHCPServer* e introducir los siguientes parámetros: IP 10.200.200.5, *netmask* 255.255.255.0 y IP inferior 10.200.200.50 e IP superior 10.200.200.200. Para el segundo adaptador (*vboxnet1/VirtualBox Host Only Ethernet Adapter 3*): IP 192.168.57.1, *netmask* 255.255.255.0 y deshabilitar el *DHCP Server*.

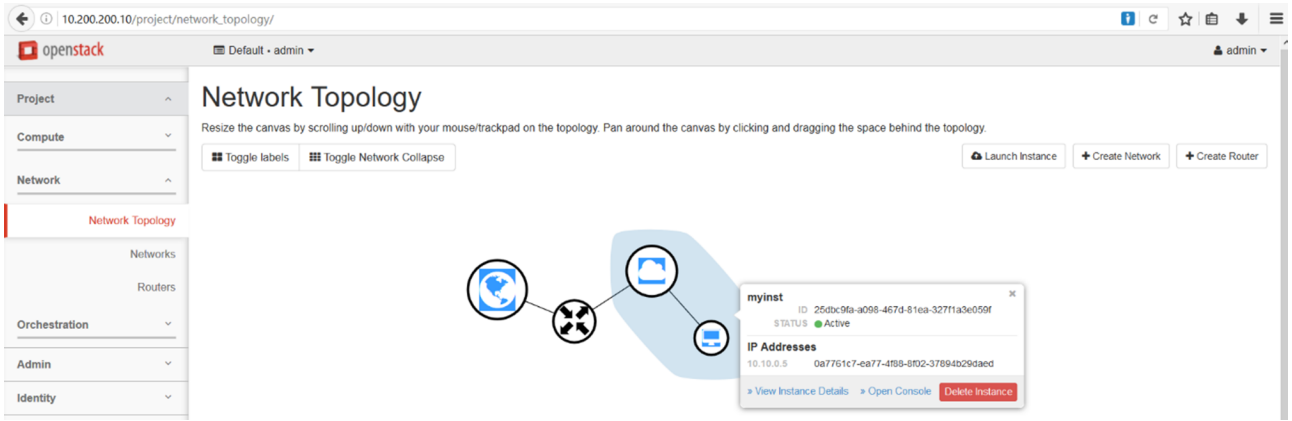
3) A continuación, importar la *appliance ciscolive_kolla_mitaka.ova* (en formato OVA) pero NO ponerla en marcha. Ir a apartado de red de la MV y verificar que el primer adaptador es *hostonly* y está sobre vboxnet0 (o el primer adaptador en Windows) y que la opción *Promiscuous Mode* (en *Advanced*) está seleccionada como *Allow All* y *Cable Connected* está marcado. El segundo adaptador debe estar como *host only* y sobre vboxnet1 (o el segundo adaptador en Windows) y que la opción *PromiscuousMode* (en *Advanced*) está seleccionada como *Allow All* y *Cable Connected* está marcado. Finalmente, que el tercer adaptador es *NatNetwork* y está apuntando a la NatNetwork creada con *Promiscuous Mode* (en *Advanced*) en *Deny* y *CableConnected* marcado.

4) Poner la MV en funcionamiento. Sobre Windows puede dar error de que no existe vboxnet0/1 pero se debe ir a los adaptadores de red, desmarcarlos, reiniciar Virtualbox, volverlos a marcar como el punto anterior y este reasignará los nombres. Luego del arranque se podrá ver la consola y acceder con usuario *osdev* y *passwd* **Cisco!Devnet** y sobre el *host* abrir un navegador para acceder al OpenStack *dashboard* (Horizon UI) en la *http://10.200.200.10/* con los siguientes parámetros *Domain*: *default*, *Username*: *admin*, *Password*: *password*.

5) Con ello ya se dispondrá de la interfaz de OpenStack, se podrá crear una instancia (tiene *cirrosOS* precargado o se podrá cargar otro), ver la red, acceder a la instancia, o utilizar la OpenStack CLI para interactuar con OpenStack o ver los contenedores.

La figura siguiente muestra una imagen del *dashboard* con una instancia en ejecución y la configuración de la topología de la red.





Como consideración final, los desarrolladores de DevNet justifican el uso de VirtualBox ya que es *opensource* y funciona muy bien en los diferentes SO, pero tiene el inconveniente de que no soporta virtualización anidada. Por ello, Kolla ha sido configurado para utilizar QEMU (sin KVM) cuando se lanza una instancia de una MV en el *cloud* OpenStack. Esto impone un impacto en las prestaciones en comparación con KVM (opción habitual en OpenStack), pero se debe tener en cuenta que este entorno está orientado a la experimentación y aprendizaje y no a la producción.

Como se puede observar, Kolla es una opción interesante para desplegar un clúster OpenStack sin las necesidades de recursos (humanos, hardware) que necesita una instalación de producción y permite reemplazar el proceso de despliegue inflexible, costoso y de uso intensivo de recursos de OpenStack con un proceso de despliegue flexible, accesible y económico y así facilitar el conocimiento sobre la plataforma y con ello su adopción.

2.5.3. Mirantis

Mirantis es una empresa (1999) orientada a los servicios de *cloudcomputing* en un modelo al B2B y centrada en el desarrollo y despliegue de OpenStack (básicamente como IaaS privados). Son miembros del Consejo de Administración de la OpenStack Foundation (institución sin ánimo de lucro establecida en 2012 para promover OpenStack y su comunidad) y uno de los contribuidores importantes (Top 3) en el desarrollo de soluciones de *cloud* (privados/públicos) y proveedor de servicios basados en OpenStack, y utilizando herramientas *open source*. Su modelo de negocio se basa en el desarrollo de servicios para las empresas, despliegue y formación sobre el entorno OpenStack, además de recibir inversiones de terceros (capital riesgo). La distribución de Mirantis (primera en 2013) aporta una visión diferente de OpenStack que permite, para hardware certificado y sobre SO habituales, disponer de una infraestructura en dos ámbitos diferenciados: desarrolladores y IT/Ops.

Entre los proyectos más importantes desarrollados/con participación de Mirantis, se pueden enumerar:

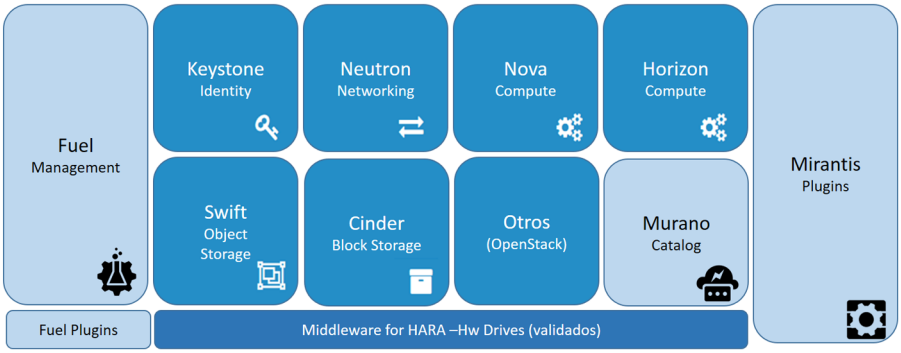
- **Fuel**: es un *toolkit* (incluido en la distribución) para desplegar y administrar *clouds* OpenStack ya que permite el despliegue, configuración y validación de todas las características de la infraestructura a través de una GUI, pudiendo el administrador escoger qué componentes de OpenStack se ejecutan y dónde. Con ellos es posible desplegar y gestionar múltiples *clouds* (*environments*) sobre una infraestructura.

- **Murano**: es un catálogo de aplicaciones (incluido en la distribución) desarrollado por Mirantis y miembros de la comunidad OpenStack para gestionar de forma dinámica servicios de terceros sobre la infraestructura.

- **Sahara**: proyecto que simplifica la creación de clústeres *Hadoop* con la colaboración de miembros de Apache SF y la OpenStack Foundation, también con la participación de Red Hat y Hortonworks.

- **Rally**: proyecto para medir las prestaciones de *clouds* OpenStack.

Hoy la empresa se encuentra inmersa en constantes cambios y alianzas para promover su negocio, estrategias habituales en el ámbito de IT, y se ha asociado con Ubuntu (dejando RH) y el mayor proveedor de *cloud* público en China (*Ucloud*). La imagen a continuación muestra la arquitectura de Mirantis sobre OpenStack [Mar].



Para la instalación de Mirantis se utilizará el procedimiento (automático) que se recomienda a través de VirtualBox en una instalación local. Es este caso se utilizará Virtualbox sobre Ubuntu 16.04 LTS:

1) Descargar la ISO de Mirantis y los *scripts* de VirtualBox.

2) Descomprimir los *scripts* y en el *directorio_creado/iso* copiar la ISO previamente descargada (algo como *fuel-virtualbox-stable-mitaka/iso*).

3) Modificar el *script config.sh* para adecuarlo al hardware disponible (no reducir el espacio de disco por debajo de los 50GB, y adecuar la memoria).

4) Instalar el paquete *expect* que en Ubuntu 16.04 no viene instalado:

```
apt-get install expect
```

5) Ejecutar *launch.sh*.

6) Después de un tiempo (largo) tendremos la interfaz de la máquina virtual con los usuarios/*passwd* (*root*, **r00tme**) y la interfaz en *https://ip_interna:8443* con usuario/*passwd* (*admin*,***admin***).

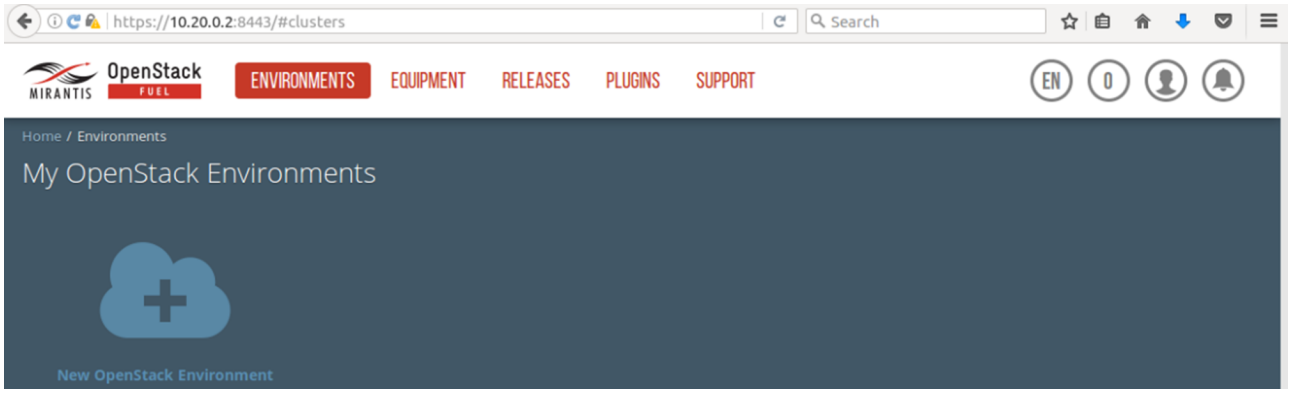
7) Es importante verificar la conexión NAT externa, ya que la interfaz eth2 no está levantada por defecto, y adecuar el *resolv.conf* para el servicio de DNS externo, pues cuando se instala dará un error *failed to build bootstrap*. Solucionada la conectividad, es necesario ejecutar `fuel-bootstrap build --activate`.

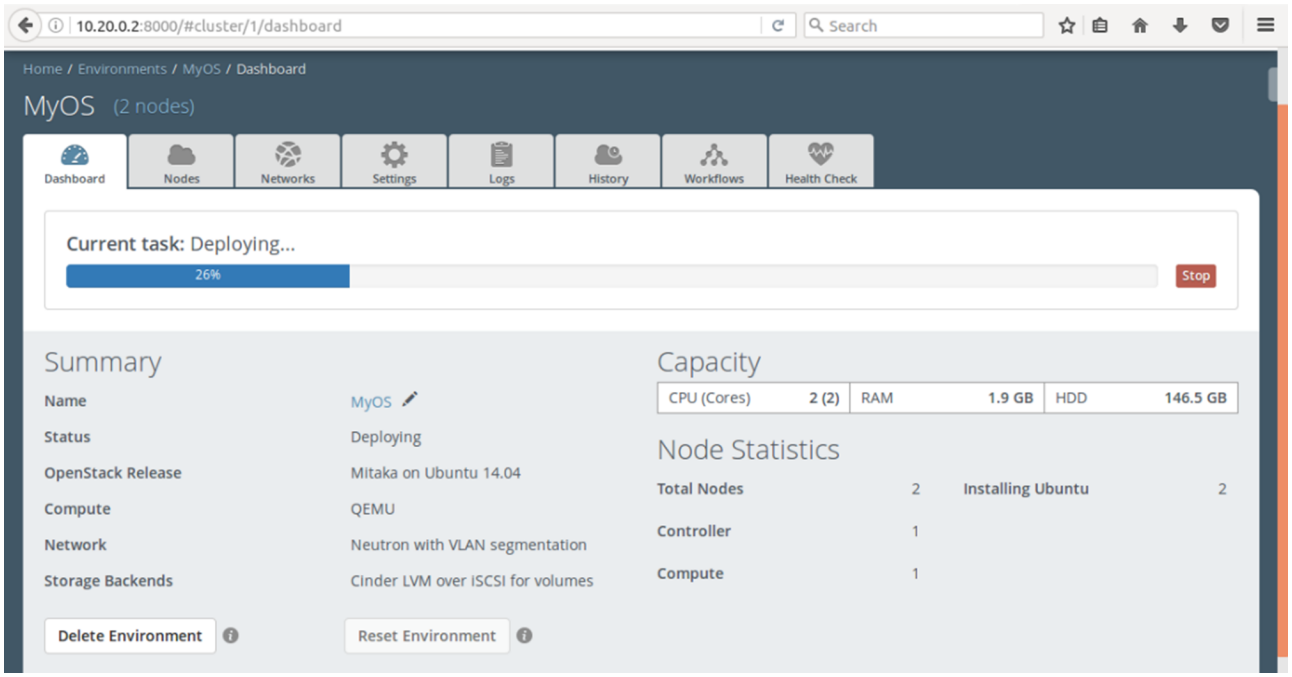
8) Tras el punto anterior se podrá observar que no se han creado los nodos, por lo cual se deberá ejecutar `./actions/slave-nodes-create-and-boot.sh` que se pondrán en marcha y harán el *boot* por PXE.

9) Una vez que se crean las imágenes y no hay errores, se puede actualizar de la versión 9 a la 9.1 siguiendo las instrucciones.

10) Con todo ello ya se visualizarán los nodos y se podrá agregar un rol (*Controller*, *Compute*, *Cinder*, *SO...*, aunque mínimo uno como *Controller* + uno como *Compute*) y en el panel principal (*Dashboard*) se podrá hacer el *Deploy* que construirá el *cloud* OpendStack.

Las figuras siguientes muestran el entorno inicial antes de crear un *Environment* (*cloud* Openstack) y la siguiente con un entorno creado (MyOS) y un despliegue mínimo de dos nodos (un nodo *Controller* y un nodo *Compute*). Es importante tener en cuenta que, aunque sea una prueba funcional solamente, es necesaria una gran cantidad de espacio de disco para los nodos, como se puede ver en *config.sh* (si bien se puede reducir un poco, es necesario contar con gran cantidad de espacio y memoria RAM). Consultar el documento *From zero to OpenStack-hosted website in 4 easy steps* donde se muestran los pasos a seguir para un despliegue de un *cloud* y la creación de un proyecto y servicio web.





Es importante tener en cuenta que Mirantis es un paquete muy potente que tiene mucha funcionalidad y opciones (no instaladas en este paquete) y despliega una infraestructura muy potente, pero también muy compleja, y que se le debe dedicar tiempo y recursos para experimentar con ella. La propia empresa realiza (una de sus partes de negocio) cursos de formación y certificación a diferentes niveles para clientes empresariales.

Esta plataforma ha tenido un gran crecimiento en los dos últimos años y es utilizada por clientes como Adobe, AT&T, Bailan, Comcast, EMC, G-Core labs, Intel IT, Naturalis, Paypal, Shenzhen, Tata, Telstra, Wells Fargo y Wargaming, entre otros.

2.6. oVirt

Tal y como se mencionó en el módulo 1, oVirt es una plataforma *open source* avanzada de gestión y administración de virtualización desarrollada como un proyecto de la comunidad, bajo el auspicio de Red Hat, y donde se generan los desarrollos que luego formarán parte del producto orientado a empresas *Red Hat Enterprise Virtualization* (RHEV).

Es importante destacar que la misma filosofía que se aplica entre los desarrollos de Fedora y su relación con RH Enterprise Linux (RHEL) se mantiene entre oVirt y RHEV; oVirt integra una gran cantidad de desarrollos (propios de RH y de la comunidad) con un gran dinamismo en opciones y funcionalidades, pero sin la estabilidad requerida para un producto empresarial, sin soporte y con un ciclo de desarrollo/vital corto. Por su parte, RHEV es un producto estable que

contiene los desarrollos y calidad/estabilidad necesarios en un producto empresarial y que puede dar garantías para un servicio crítico (obviamente, luego estas mejoras revierten en oVirt, pero en un ciclo más largo de desarrollo).

Esta plataforma permite una administración centralizada de MV, cómputo, almacenamiento y recursos de red utilizando una GUI (*front-end*) basada en web e independiente de la plataforma desde la cual se accede. oVirt solo soporta como hipervisor KVM sobre una arquitectura x86-64 y conceptualmente es similar a VMware vSphere (algunos autores lo consideran como la versión *open source* de este producto). Si bien no se puede considerar un IaaS y podría entrar en el módulo de hipervisores, se considera que, por sus características avanzadas y roles, más cercana a una plataforma de servicio de virtualización (o VaaS, *virtualization as a service*).

Entre las **principales características** de oVirt se pueden enumerar: administración de múltiples MV, GUI de usuario para gestionar todos los aspectos de VDC (*virtual data center*), diferentes métodos de asignación de MV a *hosts*, soporte de diferentes tecnologías de almacenamiento (NFS, iSCSI, FC, Posix FS, Gluster FS), migración/*snapshots*«en caliente» y clonado simplificado desde *snapshots*, esquemas (*templates*) y soporte de *cloud-init* para la creación automática de MV, acceso a las MV mediante Spice/VNC/RDP, fácil agregación/eliminación de nuevos nodos, supervisión total del entorno, control de cuotas de uso de recursos (almacenamiento, cálculo, red), múltiples VLAN con soporte SR-IOV, consola de autoservicio y roles, uso eficiente de KVM y *opensource* con el respaldo de RH y comunidad muy activa.

Además, permite la integración con diferentes plataformas para complementar su funcionalidad como por ejemplo: OpenStack Glance/Cinder y Neutron para el aprovisionamiento de disco y red respectivamente, Foreman/Katello para el aprovisionamiento de MV, Cockpit para la administración, Ansible para el despliegue automatizado, ManageIQ para el control completo del ciclo de vida de la infraestructura virtual, entre otras [Ovd].

Los **requerimientos** para oVirt son 4 GB de memoria y 20 GB de disco (almacenamiento de red opcional), rama de SO derivados de RH (Fedora 23, CentOS Linux 7.2, Red Hat Enterprise Linux 7.2, Scientific Linux 7.2), navegadores habituales (Firefox, Chrome, Safari, Explorer 11/Edge) y cliente móvil –moVirt– (opcional) para Android 4.1 o superior.

La **arquitectura** de oVirt está formada por:

- *Virtual machine* ***hosts*** que utilizan el KVM (*Kernel-based Virtual Machine*).

- **Agentes y herramientas** que se ejecutan sobre los *hosts* (VDSM, Virtual Desktop Server Manager, QEMU, y libvirt) que permiten gestionar las MV, el almacenamiento o la red.

- **oVirt** es una plataforma centralizada del entorno oVirt que provee una interfaz gráfica donde el administrador/usuario puede ver/gestionar/aprovisionar los recursos.

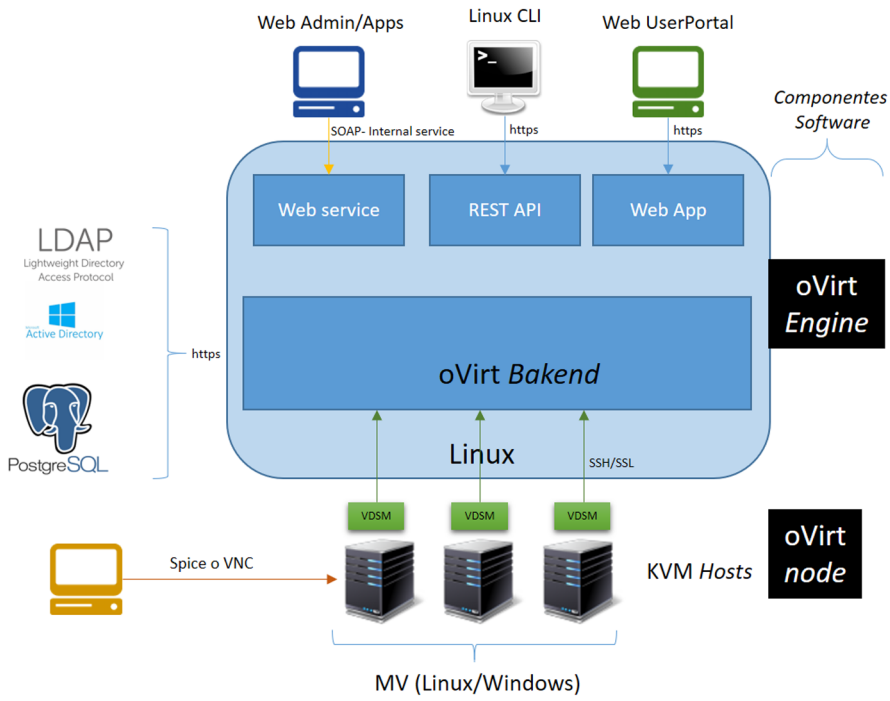
- Dominios de **almacenamiento** que tendrán los recursos virtuales como MV, esquemas (*templates*) o ISO.

- Una **base de datos** para registrar los cambios y el estado del entorno.

- Acceso a un **directorio externo** como fuente de usuarios/autentificación.

- **Red** para vincular todo el entorno incluyendo las redes físicas y redes lógicas.

La figura siguiente muestra un esquema de la arquitectura.



Como se puede observar en la figura anterior, oVirt está formado por la unión de dos componentes software: **oVirt Engine** y **oVirt Node**.

En **oVirt Engine**, el *back-end* está escrito en Java, mientras que el *front-end* en GWT web toolkit; oVirt Engine se ejecuta en WildFly *application server*, puede integrarse con servicios LDAP/AD, almacena los datos con una base de datos PostgreSQL y soporta una API RESTful para integrarse con otras herramientas o añadir nuevas funcionalidades.

oVirtNode es un servidor que ejecuta RHEL, CentOS, SL, Fedora (o de forma experimental con Debian), con KVM habilitados y con el daemon VDSM (*Virtual Desktop and Server Manager*) para la interacción con el *back-end* y el con-

trol de los recursos sobre el nodo (cómputo, almacenamiento, red). Los nodos pueden ser agrupados formando clústeres que pueden ser gestionados de forma uniforme desde el portal para mejorar el RAS (*reliability, availability & serviceability*).

Desde el 2015, oVirt Engine puede ser instalada sobre un servidor (forma habitual) o puede ser desplegada sobre un clúster de nodos, que son administrados por ella, y dentro de una máquina virtual (*self-hosted engine*: SHE). Esta SHE puede ser instalada manualmente o vía una *appliance* virtual y, en una configuración HA, puede ser ejecutada en cualquiera de los nodos.

2.6.1. oVirt Live

oVirt Live es una Fedora Live CD o CentOS Live CD con oVirt preinstalado basado en el *plugin All In One* (a ser suplantado por SHE) con fines de evaluación/demostraciones/previsualización del entorno para usuarios nuevos que deseen experimentar antes de su instalación. Puede ser cargado de diferentes medios tales como DVD (donde los cambios se perderán al siguiente *boot*), USB (puede ser RO o con almacenamiento), o MV (donde se puede ejecutar la ISO sobre una MV utilizando virtualización anidada para poder ejecutar MV dentro de esta MV).

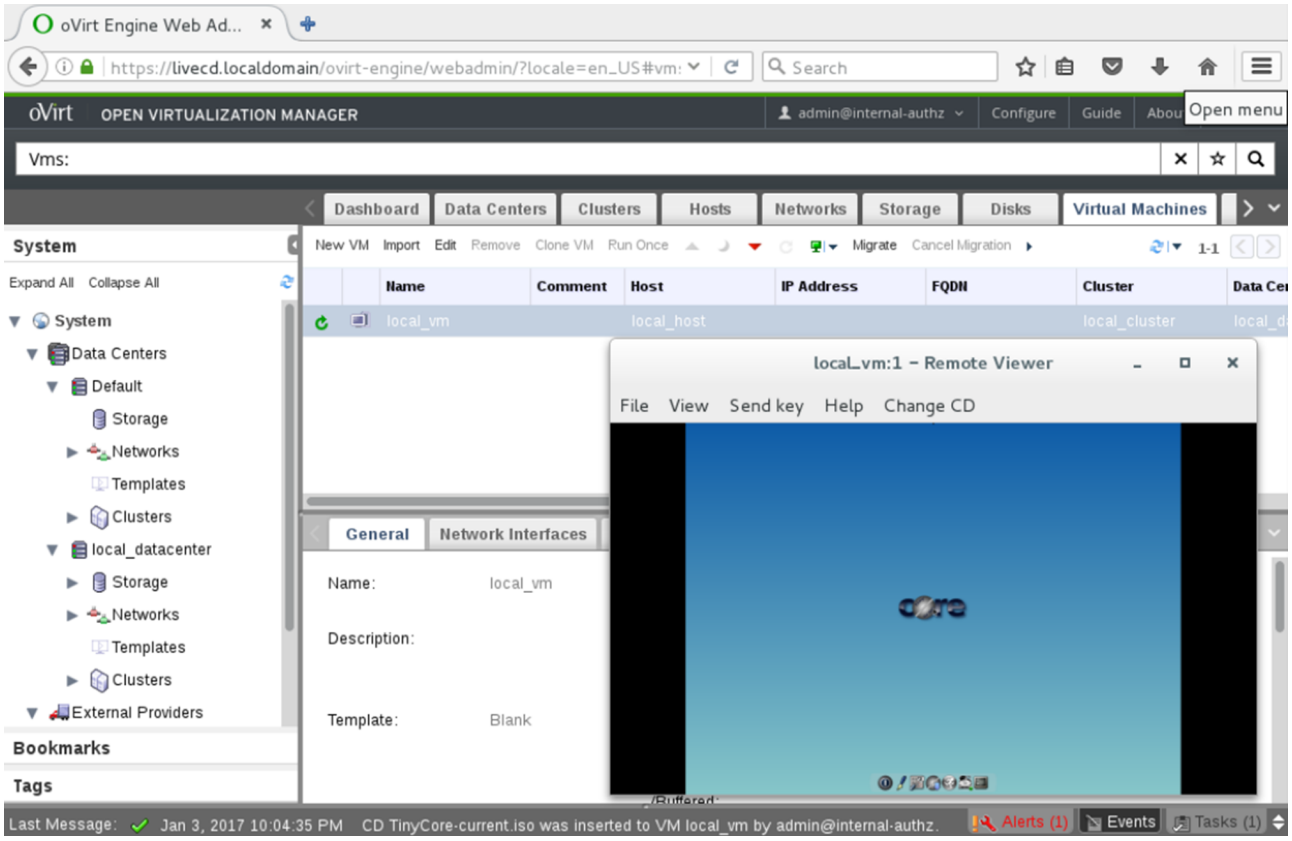
En esta prueba se ha utilizado una MV KVM con 4GB de RAM (si es menor, genera errores) con virtualización anidada habilitada y se ha arrancado desde la ISO.

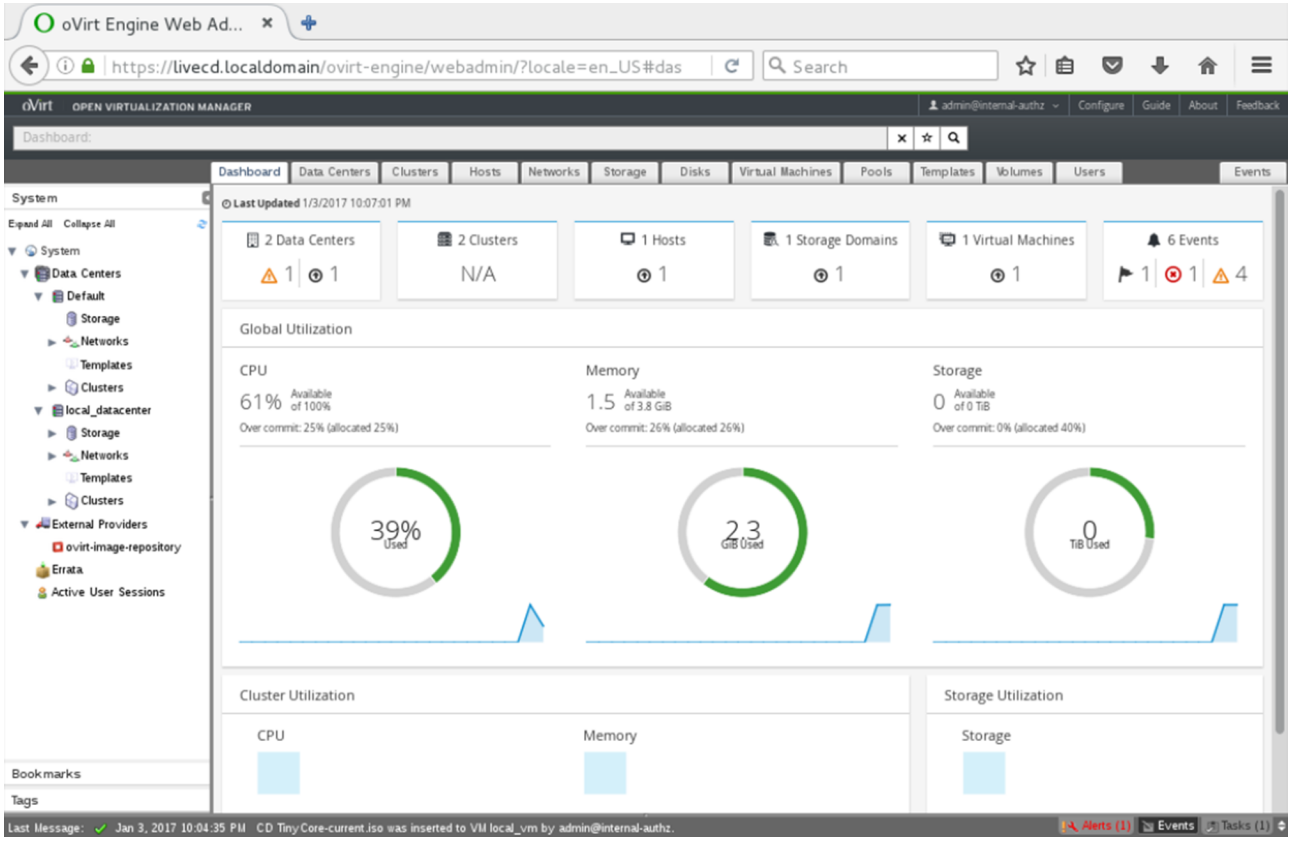
Después de indicar que la instalación sea automática, se genera la secuencia de instalación de oVirt *engine/BDs/Host* y se arranca el servidor de aplicaciones que cuando finaliza inicia un navegador que después de unos segundos se podrá conectar al portal.

Con usuario *admin* y *passwd* ***ovirt*** con todas las funcionalidades activas con las cuales se podrán experimentar.

Como se podrá observar está inicializado un *host*, y sobre el cual es posible poner en marcha una máquina virtual que existe disponible, clonarla o crear nuevas.

Es importante tener en cuenta que, como todo está en memoria, dependerá de la cantidad que se disponga para la cantidad de MV que se puedan tener simultáneamente. Las figuras a continuación muestran la ejecución de la máquina virtual con la consola y el *dashboard* de oVirt donde se muestra la carga del sistema y los recursos disponibles/ocupados.


oVirt Engine Web Ad...
https://livecd.localdomain/ovirt-engine/webadmin/?locale=en_US#vm:
Search
oVirt OPEN VIRTUALIZATION MANAGER
admin@internal-authz
Configure
Guide
About
Open menu
Vms:
Dashboard
Data Centers
Clusters
Hosts
Networks
Storage
Disks
Virtual Machines
System
Expand All
Collapse All
System
Data Centers
Default
Storage
Networks
Templates
Clusters
local_datacenter
Storage
Networks
Templates
Clusters
External Providers
Bookmarks
Tags
New VM
Import
Edit
Remove
Clone VM
Run Once
Migrate
Cancel Migration
1-1
Name
Comment
Host
IP Address
FQDN
Cluster
local_vm
local_host
local_cluster
local_vm:1 – Remote Viewer
File
View
Send key
Help
Change CD
General
Network Interfaces
Name:
local_vm
Description:
Template:
Blank
Last Message:
Jan 3, 2017 10:04:35 PM
CD TinyCore-current.iso was inserted to VM local_vm by admin@internal-authz.
Alerts (1)
Events
Tasks (1)


oVirt Engine Web Ad...
https://livecd.localdomain/ovirt-engine/webadmin/?locale=en_US#das
Search
oVirt OPEN VIRTUALIZATION MANAGER
admin@internal-authz
Configure
Guide
About
Feedback
Dashboard:
Dashboard
Data Centers
Clusters
Hosts
Networks
Storage
Disks
Virtual Machines
Pools
Templates
Volumes
Users
Events
System
Expand All
Collapse All
System
Data Centers
Default
Storage
Networks
Templates
Clusters
local_datacenter
Storage
Networks
Templates
Clusters
External Providers
ovirt-image-repository
Errata
Active User Sessions
Bookmarks
Tags
Last Updated 1/3/2017 10:07:01 PM
2 Data Centers
1
1
2 Clusters
N/A
1 Hosts
1
1 Storage Domains
1
1 Virtual Machines
1
6 Events
1
1
4
Global Utilization
CPU
61%
Available of 100%
Over commit: 25% (allocated 25%)
Memory
1.5
Available of 3.8 GiB
Over commit: 26% (allocated 26%)
Storage
0
Available of 0 TiB
Over commit: 0% (allocated 40%)
39%
Used
2.3
GiB Used
0
TiB Used
Cluster Utilization
CPU
Memory
Storage Utilization
Storage
Last Message:
Jan 3, 2017 10:04:35 PM
CD TinyCore-current.iso was inserted to VM local_vm by admin@internal-authz.
Alerts (1)
Events
Tasks (1)

2.6.2. Instalación de oVirt

En la guía de instalación se pueden encontrar los requisitos para desplegar oVirt donde los mínimos para oVirt Engine son *dual-core server* con 4 GB RAM y 25 GB de disco y una red de 1Gbps, y para cada *host*, *dual-core server* con AMD-V/Intel VT habilitada, mínimo 1 GB RAM, 3 GB de disco y 1 Gbps de red. El sistema deberá soportar por lo menos un tipo de almacenamiento (NFS, iSCSI, FCP, Local, POSIX FS, GlusterFS) y para la MV, las imágenes de los sistemas operativos que se deseen instalar (Windows XP-Vista7-8-2003-2008-2012), RHEL 5+/6+, CentOS 6+/7+, Fedora 16+, Ubuntu 12.04+, openSUSE 12).

Para la instalación de oVirt Engine se ha utilizado un MV KVM con 3,2GB de RAM y 20GB de disco, con virtualización anidada (aunque en esta no es necesaria estrictamente) y un NIC con IP propia (aunque para la instalación del SO se ha configurado primero en NAT) y sobre ella se ha instalado CentOS 7.0. oVirt Engine provee la GUI para administrar los recursos físicos y lógicos de la infraestructura subyacente, la cual se podrá gestionar a través de un navegador, por ejemplo, Firefox (en este caso desde el *host* KVM ya que en CentOS se ha instalado sin entorno gráfico).

Sobre esta MV (oVirt Engine) también se podrán ejecutar MV; solamente deben ser un CentOS y con la virtualización anidada habilitada (como es nuestro caso).

Después de instalado el servidor (con IP propia), como *root*, modificar */etc/hosts* para dar de alta la IP de la MV y un dominio FQDN y */etc/hostname* para reflejar el nombre (será necesario luego para la interfaz web), ejecutar `yum -y update` y reiniciar para que los cambios sean actualizados.

Luego, suscribirse los repositorios de oVirt en la última versión estable e instalar los paquetes y sus dependencias (250+):

```
yum install http://resources.ovirt.org/pub/yum-repo/ovirt-release40.rpm
yum -y install ovirt-engine
```

Finalmente, ejecutar el instalador:

```
engine-setup
```

Este realizará una serie de preguntas sugiriendo el valor por defecto, que será el adecuado en una gran cantidad de ocasiones, pero se deberá ir con cuidado en otras, por ejemplo, *passwd* para el usuario *admin* de gestión de la plataforma, el NFS Share de la plataforma para tener un dominio ISO. Después de verificar que todo es correcto en una lista (*Configuration Preview*), al aceptarla se iniciará la configuración que terminará indicando a la URL dónde se deberá conectar para acceder a la plataforma.

En nuestro caso será desde el *host* KVM, pero como se debe acceder por FQDN se deberá insertar en el */etc/host* de esta máquina, la IP y el FQDN de la MV oVirt Engine. Luego se podrá acceder a la URL *https://ovirteng.nteum.org:443/ovirt-engine*, se deberá aceptar el certificado de la conexión *https* y acceder con el usuario *admin* y el *passwd* introducido durante la configuración.

Luego se deberán realizar una serie de ajustes antes de poder crear la primera MV, entre los más importantes:

1) Configurar los *data centers*: es la entidad lógica más importante que contiene una serie de recursos (físicos y lógicos) manejados por oVirt (clústeres de *hosts*, MV, almacenamiento y redes). Por defecto, oVirt creará uno durante la instalación que se podrá ver en el árbol lateral o en la pestaña de *data centers*, pero se podrá borrar y crear otro nuevo o modificar este (ver la documentación de Administración).

2) Configurar clúster: es un conjunto de *hosts* físicos que son tratados como un único elemento para la ejecución de MV. Comparten la misma infraestructura de red/almacenamiento y permiten la migración de MV «en caliente» entre los diferentes *hosts* del clúster. Por defecto, oVirt crea un clúster durante la instalación y es posible navegar por el árbol de la izquierda de la interfaz o por la pestaña de *Clusters*.

3) Red: durante la instalación oVirt define una red de administración (red por defecto para el *data center*). Esta es utilizada entre el *manager* y el *host*, pero pueden definirse nuevas redes lógicas para datos o almacenamiento con el fin de mejorar las prestaciones de las MV. Para acceder a estas opciones a través del árbol de la izquierda o a través de la pestaña horizontal (la red por defecto es *ovirtmgmt*).

4) Configurar los *hosts*: estos pueden ser o bien a través de una *appliance* (oVirt Node) diseñada específicamente para oVirt o instalando un *Fedora/CentOS* y agregándola al *Manager*. En el primer caso se debe consultar los pasos de descarga e instalación y, luego de instalado, la plataforma lo reconocerá y solo se deberá aprobar su uso.

En el caso de *Fedora/CentOS* se deberá instalar el SO (una instalación mínima es suficiente) y hacer unos ajustes para, posteriormente, agregarlo a la plataforma, y esta instalará todo lo necesario para su utilización. En nuestro caso se ha utilizado un MV KVM con 2GB RAM, 20GB de disco, virtualización anidada habilitada, IP propia (*bridge*) y se ha configurado */etc/hostname* y el */etc/hosts* con FQDN e IP (tanto del nodo como del oVirt Engine). A continuación, como *root*, se deberá configurar el repositorio que recibirá los paquetes durante la actualización:

```
yum localinstall http://plain.resources.ovirt.org/pub/yum-repo/ovirt-release40.rpm
```

Luego, adecuar los puertos para los diferentes servicios y comunicación con el servidor editando */etc/sysconfig/iptables* para agregar:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [10765:598664]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp --dport 22 -j ACCEPT
-A INPUT -p tcp --dport 16514 -j ACCEPT
-A INPUT -p tcp --dport 54321 -j ACCEPT
-A INPUT -p tcp -m multiport --dports 5634:6166 -j ACCEPT
-A INPUT -p tcp -m multiport --dports 49152:49216 -j ACCEPT
-A INPUT -p tcp -m state --state NEW
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -m physdev ! --physdev-is-bridged -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Sobre CentOS7 ejecutar:

```
yum install iptables-service
systemctl restart iptables
chkconfig iptables on
```

También se debe editar el archivo de configuración de red /etc/sysconfig/network-scripts/ifcfg-ens3 para deshabilitar el *Network Manager* ya que genera conflicto con los servicios de oVirt (agregar en este fichero *NM=no*) y, además, permitir el *login* por *ssh* mediante *root* (*/etc/ssh/sshd_config* → *PermitRootLogin yes*).

Luego, desde el portal de administración ir a la pestaña de *Hosts* → *New* y rellenar los siguientes campos: *Data Center* (seleccionar el *Defaultdatacenter* o el que se desee si se tiene más de uno), el *Cluster* (en nuestro caso el *Default Cluster*), un nombre (*Name*), la IP del *host*, el *passwd* del *root*, y la casilla para agregar las reglas de *iptables* adicionales que ya se han configurado (si el *host* fuera físico se podría seleccionar el *power management* (PM), pero en nuestro caso lo dejaremos deshabilitado, lo cual generará una advertencia, pero podrá ser utilizado sin problemas). Después de aceptar la configuración (dos veces por la advertencia del PM) se verá el *host* con un icono de *Installing* y en la barra inferior de estado los paquetes que se están descargando. Cuando finalice el icono pasará a verde y estado *Up*.

5) Configurar el almacenamiento: después de configurar las redes y *hosts* será necesario agregar almacenamiento tanto para las ISO (ya está definido en la instalación, solo se debe anexar) y el de datos donde se almacenarán las MV. En nuestro caso, se utilizará *Network File System* (NFS), y se deberá crear un dominio. Para ello primero deberemos, sobre la máquina oVirt Engine, crear los directorios (adicionales al ISO que se creó durante la instalación):

```
mkdir /export/data
chown vdsm:kvm /export/data
mkdir /import_export
chown vdsm:kvm /export/import_export
vi /etc/exports
 # agregar teniendo en cuenta que esta es la menos restrictiva y se puede
 # restringir solamente a los hosts
hosts
    /export/data  *(rw,sync,no_subtree_check,all_squash,anonuid=36,anongid=36)
    /export/import_export  *(rw,sync,no_subtree_check,all_squash,anonuid=36,anongid=36)
```

Luego ir a la *GUI → Default Data Center → Storage → New Domain* e introducir un nombre, el *Data Center* (que saldrá preseleccionado), *Domain Function/Storage Type = Data → NFS* y el *Export Path* (en nuestro caso o *virteng.nteum.org:/export/data*) y *Use Host* el que podrá utilizar este (saldrá preseleccionado).

Luego de aceptar el nuevo *NFS data domain* se mostrará sobre la pestaña de *Storage* (en estado «bloqueado» primero durante su inicialización y luego en «listo» cuando esté disponible).

Para anexar el ISO domain ir al Data Centers y en la ventana de *detalles → Storages → Attach ISO* seleccionar el que aparece por defecto y aceptar y, después de unos instantes, este aparecerá activo en la ventana *Storage*.

Para agregar una imagen al repositorio (necesaria para luego crear una MV), se descargará la ISO en un directorio temporal (*/tmp*). En esta prueba se utilizará TinyCore 7.2 (http://tinycorelinux.net/7.x/x86/release/TinyCore-7.2.iso), se obtendrá el nombre del repositorio y se cargará la imagen:

```
engine-iso-uploader list   nos da el Domain, ISO_DOMAIN en este caso
engine-iso-uploader upload -i ISO_DOMAIN /tmp/TinyCore-7.2.iso
```

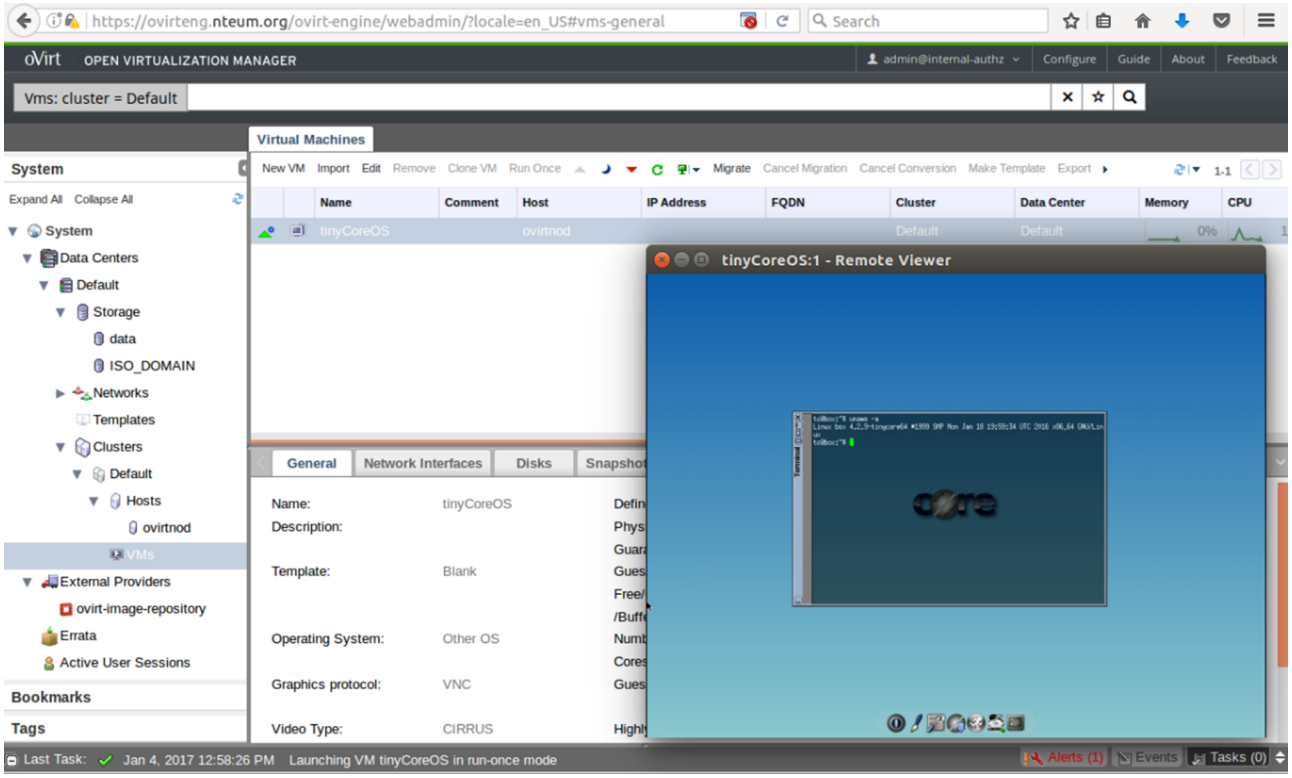
Luego de cargadas, se podrán ver en el portal en la pestaña de *Storage → ISO_DOMAIN → Images*

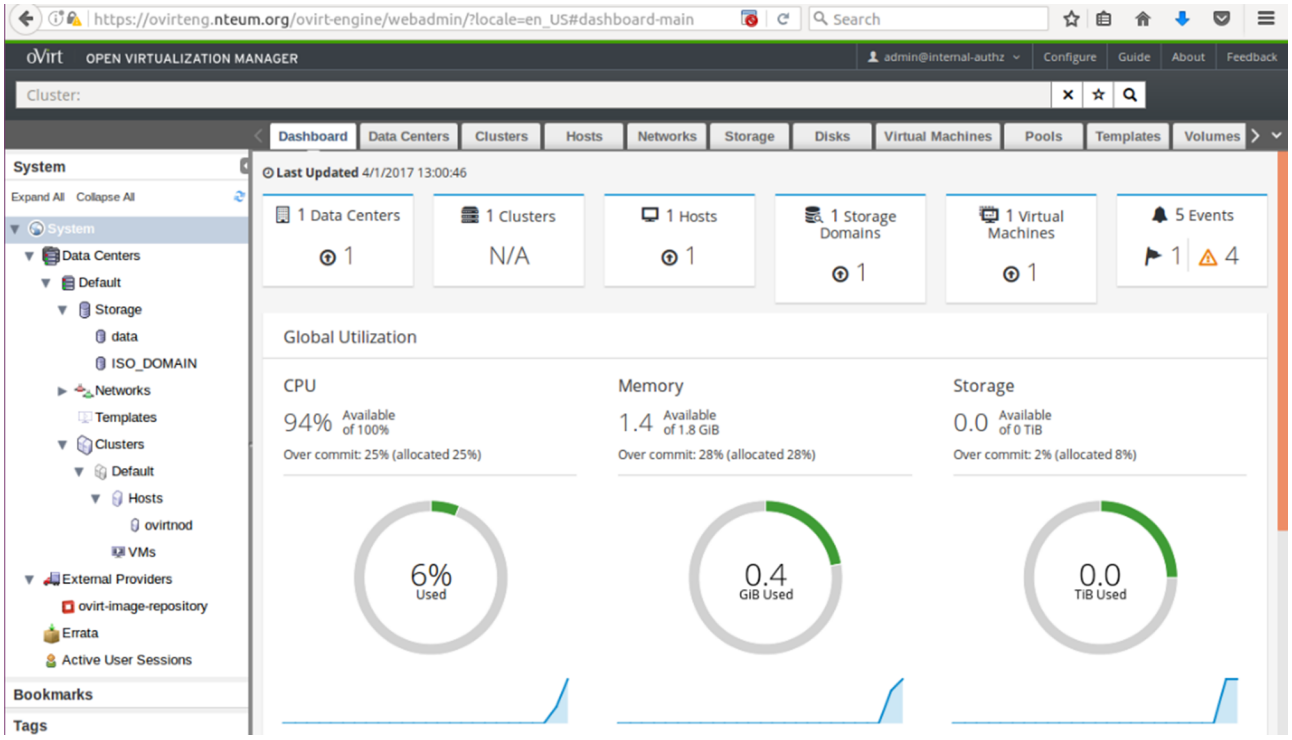
6) Crear una *Virtual Machine*: desde la pestaña de *Virtual Machines → New VM. Seleccionar Default Cluster, Template, OS, Instance, Optimized, Name → OK*. Luego se abrirá otra ventana que permitirá seleccionar el disco y finalmente seleccionar *Run Once* y seleccionar *Attach CD* seleccionando la *ISO* de *TinyCore*

y cambiar el orden de inicio y OK para iniciar la MV. Desde la *Consola* (icono en la parte superior) se abrirá una ventana que ejecutará VNC (se puede cambiar a noVNC) y tener la interfaz con la MV.

Consultar la documentación para información más detallada y/o ajustes adicionales [Ova][Ovu].

Las figuras siguientes muestran la ejecución de un MV TinyCore sobre oVirt Node junto con su consola gráfica VNC y el *dashboard* de oVirt Engine (sobre otra MV) con la monitorización de Esta.





Al igual que en las recomendaciones anteriores sobre otras plataformas, las opciones y posibilidades de esta son muy extensas y se deberá dedicar tiempo y recursos a experimentar con todas sus posibilidades/prestaciones. Es una plataforma muy estable, activa, con una comunidad muy involucrada y la participación de personal cualificado de RH [Ova][Ovu].

2.7. Nimbus

> Si bien Nimbus ha sido un proyecto pionero (desde 2005) en cuanto a la aportación de ideas y la generación de infraestructura IaaS (y como entorno experimental de *Science Clouds*), es un proyecto que desde 2014 no recibe ninguna actualización (como se puede observar en Github).

El grupo desarrollador de Nimbus, en la actualidad, está enfocado en un nuevo proyecto (desde 2014 y financiado por la NSF-USA) llamado *Chamaleon* como un entorno experimental configurable para la investigación del *cloud* a gran-escala.

No obstante, el proyecto Nimbus continúa disponible como *open source* para la comunidad (con actualizaciones hasta 2014, y si bien solicitan programadores en 2016, se entiende que es para el proyecto Chamaleon) y es posible realizar pruebas con la infraestructura software disponible siguiendo la guía para *sysadmin* que provee documentación para la instalación, configuración y prueba de la versión 2.10.1 (último *patch* 2013):

- Verificación de las dependencias
- Instalación de servidor Nimbus y Cumulus
- Instalación del servicio DHCPd y configuración de la red
- Instalación de los *hosts* (VMM)
- Configuración del acceso (configuración de SSH)
- Test finales

En resumen, se considera que el proyecto Nimbus es una infraestructura importante pero que en la actualidad ha quedado obsoleta y no se recomienda como infraestructura en producción, aunque puede ser útil para casos de estudio y experimentación con infraestructuras IaaS [Nim].

Actividades

1. Implementar un par de máquinas virtuales sobre una infraestructura de *cloud* público que se puedan interconectar entre ellas.

2. Considerando las diferentes plataformas IaaS, implementar una MV en cada una de ellas y demostrar su conectividad.

3. Teniendo en cuenta los entornos IDE públicos, desplegar una MV e implementar un servicio de web verificando su conectividad.

Glosario

access control list Mecanismo de seguridad para el acceso a recursos/objetos mediante listas de atributos.

AD (*active directory*) Implementación de servicio de directorio de Microsoft que utiliza otros servicios (LDAP, DNS, DHCP y Kerberos).

AMD *virtualization* (AMD-V) Tecnología de AMD que permite la virtualización en la plataforma x86 de AMD.

AMI Imagen de máquina virtual de Amazon.

Ansible Herramienta para el despliegue automático de aplicaciones y de infraestructura IT (equivalente a Puppet/Chef).

Apache CloudStack, Eucaliptus, OpenNebula, OpenQRM (*community edition*), OpenStack Plataformas *open source* de IaaS.

Apache jclouds Proyecto *open source* que provee un conjunto de herramientas para entorno Java que permite crear aplicaciones portables a través de los *clouds* sin preocuparse por las especificidades de estos.

API Conjunto de subrutinas, funciones y procedimientos/métodos que ofrece cierta librería para que pueda ser utilizado por otro programa como una capa de abstracción e interacción.

API RESTful API que permite proveer interoperabilidad entre ordenadores en internet.

As-Is Término legal que no se responsabiliza por lo que pueda pasar o la calidad del servicio que se pueda prestar (generalmente utilizado en servicios gratuitos).

autoscaling Tecnología que permite aumentar o reducir el tamaño/prestaciones de los recursos bajo demanda de AWS.

AWS (Amazon Web Services) Infraestructura *cloud* públicos.

Azure Infraestructura de *cloud* público de Microsoft.

bigdata Cantidades de datos elevadas (decenas de cientos de petabytes) que necesitan tecnologías y análisis diferentes a los habituales.

BitBucket Entorno de colaboración en línea basado en Git.

CDN Red de distribución de contenido global que permite entregar contenido de alto ancho de banda que se hospeda en un *cloud*.

Chef Herramienta para el despliegue automático de aplicaciones y de infraestructura IT (equivalente a Ansible/Puppet).

Cinder, Glance, Keystone, Neutron, Nova, Swift, Horizon Módulos básicos de OpenStack.

CLI (*command line interface*) Comandos (en modo texto) en un terminal o consola.

Cores Núcleo de procesamiento e integrado junto a otros iguales en un único chip de procesador.

Cloud-init Paquete –estándar *de facto*– multi-distribución para la inicialización de una instancia del *cloud*.

CloudSigma Infraestructura de *cloud* público de Cloudsigma.

Codeanywhere, Codenvy, Minicloud, Tutorial Point-Coding Ground Máquinas virtuales/contenedores en línea (entornos basados en IDEs) o pseudo IaaS.

CPD Centro de procesamiento de datos.

DDoS (*distributed denial-of-service*) Ataque sobre un servidor/servicio con la intención de saturarlo y que no pueda prestar dicho servicio.

DevOps Unión de *developement* y *operations*; movimiento que tiene su origen en la mejora de la comunicación, colaboración e integración entre desarrolladores de software y los profesionales en las tecnologías de la información (IT).

DevStack, Kolla, Mirantis Conjunto de *scripts*/procesos para desplegar OpenStack.

DHCP (Dynamic Host Configuration Protocol) Protocolo que permite a un ordenador (cliente) obtener, en forma dinámica, desde un servidor todos los parámetros de red para su configuración.

DNS (*domain name system*) Servicio de nombres que transforma un nombre.dominio en IP y viceversa.

Docker Entorno de virtualización eficiente y estable a nivel de SO.

EC2 (*elastic compute cloud*) Entorno de cómputo de AWS.

EOL (*end of life*) Final del período de tiempo de vida útil de un producto (hardware o software) en el cual ya no se recibirán actualizaciones o soporte.

FC (*fabric-controller*) Módulo responsable de aprovisionamiento, gestión, provisión de servicios y gestión del ciclo de vida de los procesos/servicios en Azure. Constituye el *kernel* del Cloud Operating System de Azure.

Firewall Programa informático que controla el acceso de un ordenador a la red y viceversa, por motivos de seguridad.

GCE (Google Compute Engine) Componente principal de GCP que permite el despliegue, administración y control de MV. Es utilizada por los propios servicios de Google como Google Search Engine, Gmail, YouTube, entre otros.

GCP (Google Cloud Plataform) Infraestructura de *cloud* público de Google.

Github Entorno en línea para el almacenamiento y gestión de proyectos utilizando Git. Incluye almacenamiento y visualización del código fuente, edición en línea, *wikis* y *ticketing* (soporte y control del software). Gratis para código *open source* público.

GUI (Graphical User Interface) Entorno que permite en forma gráfica interactuar con el usuario (opuesto a la CLI).

HDD (Hard Disk Drive) Dispositivo de almacenamiento de datos de tecnología magnética/mecánica.

host Denominación que se le da generalmente a una máquina que dispone de servicios de virtualización y permite la ejecución de máquinas virtuales.

HPC (*high perfomance computing*) Infraestructura para el cómputo de altas prestaciones.

Hyper-V Hipervisor y tecnología de virtualización de Microsoft.

IaaS (*infrastructure as a service*) Infraestructura como servicio es uno de los tres modelos fundamentales en el *cloud computing*, junto con el de plataforma como servicio (PaaS, *platform as a service*) y el de software como servicio (SaaS, *software as a service*).

IDE (*integrated development environment*) Entorno integrado de desarrollo.

IIS (*internet information services*) Servidor web y servicios vinculados de Microsoft.

IoT (*internet of things*) Internet de las cosas; es un concepto que se refiere a la interconexión digital de objetos cotidianos con internet.

IT (*information technology*) Equivalente a tecnologías de la información y la comunicación (TIC).

Kerberos Protocolo de autenticación desarrollado por el MIT que permite a dos ordenadores en una red insegura demostrar su identidad mutuamente de manera segura.

kernel Núcleo del sistema operativo (SO).

Kubernetes Plataforma *open source* para automatizar el despliegue, escalado y operaciones de contenedores de aplicaciones entre clústeres.

KVM (*kernel-based virtual machine*) Hipervisor asistido por hardware para Linux.

LDAP (Lightweight Directory Access Protocol) Protocolo (a nivel de aplicación) que proporciona un servicio distribuido de directorio.

LOPD Ley orgánica de protección de datos de carácter personal (en Europa todos los países tienen requerimientos de LOPD).

machine learning Campo de investigación en las ciencias de la computación/inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a los ordenadores aprender por sí mismos.

MMU (*memory management unit*) Unidad del procesador que gestiona la memoria y la memoria virtual del mismo.

MV (*máquinas virtuales*, VM) Instancias en ejecución de una máquina virtual.

Nimbus Plataforma IaaS (una de las pioneras en el campo del *cloud computing*, EOL, última actualización de 2013).

NFS (*network file system*) Sistema de archivos de red.

NTP (*network time protocol*) Protocolo de internet para sincronizar los relojes de los sistemas informáticos.

OpenVZ Tecnología de virtualización en el nivel de sistema operativo (contenedores) para Linux.

oVirt Administrador de altas prestaciones para gestionar todos los elementos de un entorno virtualizado. Proyecto con el soporte de RH.

Redis Motor de base de datos en memoria que utiliza almacenamiento en tablas de *hashes* (clave/valor) y que puede ser usada también como una base de datos persistente.

RHEV (Red Hat Enterprise Virtualization) Plataforma de virtualización de Red Hat basada en KVM y RHEL (Red Hat Enterprise Linux).

RDP (*remote desktop protocol*) Protocolo propietario de Microsoft que permite la interconexión entre un cliente y un servidor mostrando el escritorio/consola del servidor sobre el cliente de forma remota.

RDS (*relational database service*) Servicio de bases de datos relacionales administrado de AWS para MySQL, PostgreSQL, MariaDB, Oracle BYOL o SQL Server.

S3 (*simple storage service*) Infraestructura de almacenamiento de objetos segura/escalable de AWS.

SandBox Entorno que permite en forma aislada ejecutar un conjunto de procesos sin interferir con el resto (externo).

SELinux (*security-enhanced Linux*) Módulo de seguridad para el núcleo de Linux que proporciona un mecanismo para soportar políticas de seguridad.

SLA (*service level agreement*) Contrato-compromiso de la calidad de servicio que se debe prestar por parte de un proveedor y las indemnizaciones a las que se tiene derecho el cliente en caso que se incumplan.

SO *guest* SO que se ejecuta sobre un hardware virtualizado.

SSH (*secure shell*) Intérprete de órdenes seguro para la conexión remota (comunicación encriptada).

stacks Entornos de desarrollo basados en una tecnología/servicios/lenguajes.

startup Empresa emergente e incipiente del ámbito tecnológico (aunque puede ser de otros ámbitos).

SSD (*solid state disk*) Disco de estado sólido.

TCO (*total cost of ownership*) Método de cálculo para determinar los costes directos e indirectos y beneficios, relacionados con la compra de hardware y software.

Tier-III Niveles de servicio de calidad de un centro de datos.

time to market Tiempo a reducir entre que se desarrolla una aplicación y está disponible para los usuarios.

threads Conjunto de instrucciones que tienen identidad propia y que se pueden ejecutar eficiente y concurrentemente si el procesador soporta tecnología *multithreading* o diferentes *cores*.

URL (*uniform resource locator*) Identificador que permite denominar recursos dentro del entorno de internet para que puedan ser localizados.

Vagrant Herramienta para la creación y configuración de entornos de desarrollo virtualizados. Originalmente para VirtualBox y sistemas de configuración tales como Chef, Salt y Puppet pero actualmente permite múltiples proveedores, como VMware, EC2, LXC, entre otros.

vCPU CPU virtual.

VDC (Virtual Data Center) Infraestructura conectada de servidores (virtualizados) y que prestan un servicio complejo y está alojado en un *cloud*.

VMware ESXi Hipervisor de VMware.

VNC (Virtual Network Computing) Entorno *open source*, basado en una estructura cliente-servidor, que permite tomar el control del ordenador servidor remotamente a través de un ordenador cliente (noVNC implementación del cliente VNC a través de HTTP).

VPC (*virtual private cloud*) Red virtual dedicada a la cuenta del usuario de AWS.

VPN (*virtual private network*) Red privada virtual que permite la extensión segura de la red de área local (LAN) sobre una red pública o no controlada como internet.

VT-x (Intel Virtualization Technology) Tecnología de Intel que permite la virtualización en una plataforma x86.

Bibliografía

Todos los enlaces han sido visitados en diciembre de 2016/17.

[Aar] AMD Virtualization solutions. AMD. <http://www.amd.com/en-us/solutions/servers/virtualization>

[Acs] Apache CloudStack. Documentation. 2016. <http://docs.cloudstack.apache.org/en/latest/>

[Aag] Apache CloudStack. Administration Guide. 2016 <http://docs.cloudstack.apache.org/projects/cloudstack-administration/en/4.9/>

[Ado] Azure Documentation. 2017. <https://docs.microsoft.com/en-us/azure/>

[Afa] Azure Free Account. 2017.<https://azure.microsoft.com/es-es/free/>

[Afs] AWS. Free Software. 2017.<https://aws.amazon.com/s/dm/optimization/server-side-test/free-tier/free_nc/#software>

[Ags] AWS. Getting Started Tutorials. 2017. <https://aws.amazon.com/getting-started/>

[Avt] AMD Virtualization Technology. 2017. <http://www.amd.com/en-us/solutions/servers/virtualization>

[Aws] AWS Services Overview. September 2016. <http://www.slideshare.net/AmazonWebServices/aws-services-overview-september-2016-webinar-series>

[Azp] Azure Portal. 2017.<https://portal.azure.com>

[Caw] CodeAnyWhere. Documentation. 2016. <http://docs.codeanywhere.com/>

[Cll] OpenStack on your Laptop. Cisco 2016. <https://learninglabs.cisco.com/lab/openstack-install/step/1> (necesita registro gratuito en la comunidad DevNet antes de acceder). Alternativa: <https://communities.cisco.com/community/developer/openstack/blog/2016/02/25/trying-openstack-using-kolla>

[Con] Codenvy. Documentation V5.0. 2016. <https://codenvy.readme.io/docs/introduction>

[Cpt] Cloud Computing: Paradigms and Technologies. 2013. A.Shawish and M. Salama. Inter-cooperative Collective Intelligence: Techniques and Applications. Vol. 495. Series Studies in Computational Intelligence. pp 39-67. 2013 <http://link.springer.com/chapter/10.1007%2F978-3-642-35016-0_2>

[Edo] Eucalyptus Cloud Official Documentation. Version 4.3.1. 2017. <https://docs.eucalyptus.com/eucalyptus/4.3.1/>

[Egs] Eucalyptus Getting Started. 2017. <https://docs.hpcloud.com/eucalyptus/4.3.0/index.html#user-guide/getting_started.html>

[Ee2] Eucalyptus Getting Started with Euca2ools. 2017. <https://docs.hpcloud.com/eucalyptus/4.3.0/index.html#user-guide/getting_started_euca2ools.html>

[Eec] Eucalyptus. Console Guide. 2017. <https://docs.hpcloud.com/eucalyptus/4.2.2/index.html#shared/console_section.html>

[GCP] Google Cloud Platform. PC Mag. Reviews. 2015. http://www.pcmag.com/article2/0,2817,2496296,00.asp>

[Gdo] Google Cloud Platform. Documentation. 2017. <https://cloud.google.com/docs>

[Haw] How Azure Actually Works. Mark Russinovich. 2010. <http://itknowledgeexchange.techtarget.com/cloud-computing/how-azure-actually-works-courtesy-of-mark-russinovich/>

[Hpe] HP Helion Eucalyptus. Official Documentation. 2017. <https://docs.eucalyptus.com/eucalyptus/latest/#shared/index.html>

[Iar] Skylake Architecture. Virtualization Technology. Intel. <http://ark.intel.com/es-es/products/codename/37572/Skylake#@All> <http://www.intel.es/content/www/es/es/virtualization/virtualization-technology/intel-virtualization-technology.html>

[Iaa] Introducing Microsoft Azure. 2008. David Chappell, Chappell & Associates. <http://download.microsoft.com/download/e/4/3/e43bb484-3b52-4fa8-a9f9-ec60a32954bc/Azure_Services_Platform.pdf>

[Ivt] Intel virtualization Technology. 2017. <http://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html>

[Mar] Mirantis Architecture. 2016. <https://www.mirantis.com/software/openstack/architecture/>

[Mcd] Cloud Design Patterns. 2016. Microsoft. <https://azure.microsoft.com/en-us/resources/infographics/cloud-design-patterns/ https://technet.microsoft.com/en-us/library/dn919927.aspx>

[Nim] Nimbus. Installation Guide Version 2.10.1. 2013. <http://www.nimbusproject.org/docs/2.10.1/admin/z2c/index.html>

[Odo] OpenStack Documentation. Newton 2016. <http://docs.openstack.org/>

[Ofe] OpenNebula. Front-end Installation. 2016. <http://docs.opennebula.org/5.2/deployment/opennebula_installation/frontend_installation.html>

[One] OpenNebula. Deployment Guide. 2016. <http://docs.opennebula.org/5.2/deployment/index.html>

[Oni] OpenNebula. KVM Node Installation. 2016 <http://docs.opennebula.org/5.2/deployment/node_installation/kvm_node_installation.html>

[Ova] oVirt Administration Guide. 2017. <https://www.ovirt.org/documentation/admin-guide/administration-guide/>

[Ovu] oVirt User Guide. 2017. <https://www.ovirt.org/documentation/user-guide/user-guide/>

[Ovd] oVirt Documentation. 2016. <https://www.ovirt.org/documentation/>

[Qrm] OpenQRM. Documentation & HowTo's. 2016. <http://www.openqrm.com/rm-HowTos.html>