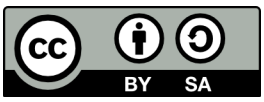
MEMORIA PROYECTO FINAL MASTER OFICIAL DE SOFTWARE LIBRE
(UNIVERSITAT OBERTA DE CATALUNYA)

Estado del Arte en Soluciones de Virtualización/Sistemas Gestores Cloud. Openstack

Especialidad: Administración de Redes y de Sistemas Operativos.

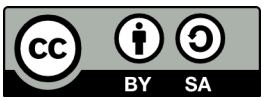
Autor: **José Luis Pérez Díaz**
Consultores: **Ing. Jordi Massaguer Pla**
Consultor Externo: **Antonio Rodil Garrido**
Entidad Colaboradora: **Artic S.L.**

31 de Diciembre de 2013



Licencia

Licencia Atribución-CompartirIgual 2.0 Genérica CC-BY-SA 2.0



Presentación del Proyecto Final de Master

El foco de este proyecto ha sido el análisis, y posterior implantación de una solución real Cloud, orientada a explotación de servicios de todo tipo, 100% operativa, basada en Openstack. Disponible, para su explotación por cualquier empresa, como ya se ha indicado, en su doble vertiente:

- (1) tanto como usuaria de la plataforma o
- (2) como explotadora de ésta para dar servicios a terceros.

Openstack puede ser uno los proyectos OpenSource más relevantes en el ámbito Cloud. Sin duda alguna, cuenta con el apoyo de (1) los principales fabricantes de Servidores como Dell y HP, (2) de fabricantes de Electrónica de Red como Cisco y Huawei, (3) principales fabricantes de S.O. Linux como RedHat y Canonical, y(4) proveedores de Servicios IaaS como RackSpace y es una garantía del potencial del proyecto y de su presumible extensión como Solución.

OpenStack es en realidad un conjunto de proyectos Opensource que operan como un "Lego" dentro de una arquitectura abierta y altamente escalable. Este aspecto es el que me resulta más fascinante.

El proyecto incluye la última versión liberada a finales de Octubre, OpenStack Havana, que contiene la totalidad de los proyectos de la Arquitectura, aun cuando algunos no estén maduros (Heat y Ceilometer) pero considerando que no son no nucleares en ésta y cubren funciones auxiliares. En cualquier caso, ambos proyectos están disponibles en la solución y completamente operativos.

La arquitectura diseñada constará de 6 servidores que contendrán todos los Componentes Openstack y las herramientas de Monitorización y Gestión de alarmas precisas, con conexión a internet e integrará fácilmente todo tipo de imágenes para instanciar servidores tipo Linux,Windows,etc.. orientados a dar servicios reales internos o externos.

Los aspectos operativos como pudieran ser: la automatización de la instalación, la escalabilidad, el rendimiento, la monitorización, la alta disponibilidad, la gestión y administración vía interface web, línea de comandos CLI, o API compatible con Amazon AWS, etc.. serán aspectos fundamentales de consideración del proyecto.

El proyecto se introducirá en los servicios básicos de Amazon AWS (de computación y almacenamiento) y se establecerán paralelismos entre Openstack y Amazon, que de facto es la solución líder en servicios IaaS, para los distintos tipos de servicios que ofrece Amazon y que puede ofrecer OpenStack.

Nota: Los apéndices contienen información de configuración y scripts de instalación que permitirán la reproducción de la arquitectura por parte de terceros.

Índice

1 Introducción

En los últimos años/décadas las necesidades de explotación en el uso de las tecnologías de la información y procesamiento de datos han crecido exponencialmente. Frente a este crecimiento las Arquitecturas informáticas han evolucionado para responder a este crecimiento.

La aparición de **Linux** fue fundamental en los inicios para crear nuevas arquitecturas favorecedoras del procesamiento paralelo. Otro elemento de cambio radical fue **Internet**, que además de la navegación web, facilitaba el acceso remoto (conectividad) a Sistemas distantes por parte de usuarios y Empresas.

Así estaban las cosas en la primera mitad de la década de 2000 con arquitecturas tipo Cluster, Grids de servidores,etc.. con amplias limitaciones, cuando empezaron a popularizarse las tecnologías de **Virtualización** que hacían posible implementar máquinas virtuales -que desacoplan en Hardware del software y permiten "replicar" el entorno de usuario sin tener que instalar todo el Sofware,etc.. para soportar aplicaciones.

Esta nueva arquitectura permitía distribuir la carga de trabajo/procesos,etc... de forma sencilla y abría una nueva puerta al Cálculo Distribuido que se llamó **Cloud Computing**. Este nuevo modelo emerge como un nuevo paradigma capaz de proporcionar recursos de cálculo y de almacenamiento que, además, resulta especialmente apto para la explotación comercial de las grandes capacidades de cómputo de proveedores de servicios en Internet.

Rápidamente el concepto Cloud Computing se establece y consolida como nuevo paradigma de cálculo escenario de las Plataformas TI. Sin embargo, es notorio, que este novedoso e interesante paradigma todavía en el año 2013 no está muy implantado en las Empresas de España.

Cloud Computing (según la definición del NIST – National Institute of Standars and Technology) *es un modelo tecnológico que permite el acceso ubicuo,adaptado y bajo demanda en red un conjunto compartido de recursos de computación configurables compartidos (redes, servidores, almacenamiento, aplicaciones, servicios,etc...) que pueden ser rápidamente aprovisionados y liberados con un esfuerzo mínimo de gestión o interacción mínima con el proveedor del servicio.*

Cloud Computing significa una revolución en la operativa de procesamiento dela información y gestionar las áreas IT. Tradicionalmente se realizaban inversiones enormes en hardware, licencias, CPDs, redes, Personal, Seguridad,etc... mientras que con el Cloud Computing no hay inversiones y los costes fijos se reducen drásticamente, y los suministros ofrecidos por los nuevos proveedores son de Servicios flexibles e instantáneos bajo demanda.

Las soluciones de Cloud Computing existentes se clasifican según diferentes criterios, que fundamentalmente son: Familias, y Formas de Implementación.

- ***Si atendemos como criterio a las Familias (Modelos de Servicio):***

1. **Infraestructure as a Service (IaaS).**

Se pone a disposición del usuario la infraestructura informática (cpu, discos, memoria,etc...) como servicio. El usuario busca evitar la inversión en Hardware, y gastos asociados a DataCenters propios o en alquiler. Con esta externalización típicamente las facturas de estos servicios se variabilizan y el pago es por uso en función de los recursos indicados previamente que hayan sido consumidos.

2. **Software as a Service (SaaS)**

Se entregan al cliente aplicaciones como servicio. Se ofrecen licencias de las aplicaciones para su uso bajo demanda. Los proveedores de los servicios SaaS pueden tener, por ejemplo, instalada la aplicación en sus servidores web y permitiendo mediante login acceder a los clientes a éstas mediante un navegador web).

Ejemplos de servicios ofrecidos para empresa son aplicaciones de CRM, ERP,etc...

3. **Platform as a Service (PaaS)**

Son servicios de plataformas orientadas al desarrollo, testing, despliegue,etc.. de aplicaciones del cliente.

- La información de los Sistemas se encuentra ubicada en la "nube pública"... Puede originar conflictos de por las exigencias de seguridad que pudieran desearse.

- ***Si atendemos como criterio a las Formas de Implementación (Formas de Integración y Explotación):***

1. **Cloud Público (Externo)**

Se ofrecen a los servicios de computación virtualizados (Sistemas operativos, plataformas, bases de datos,etc...) por parte de los proveedores a través de Internet (y/o VPN seguras).

- Los plazos de disponibilidad del servicio son mínimos. Los costes son variables completamente en función del principio de pago por uso.
- No se ser quiere inversión para la implementación de los servicios (Capex).
- Se externaliza el servicio completamente sobre el proveedor externo (o todos los servicios de la Empresa) si así fuera decidido.
- Se obtiene alta flexibilidad y escalabilidad (frente a redimensionamientos imprevistos).
- Favorece el uso de software estándar (appliance).

2. **Cloud Privado (Interno)**

Típicamente el usuario contrata el servicio por parte de un proveedor que puede actuar como mantenedor de la solución (también podría hacer el usuario), y además la solución puede existir en el Datacenter del usuario.

- Reducido plazo de puesta en servicio y alta flexibilidad en la asignación de recursos.
- Requiere inversión en Hardware y según, también en Software.
- Requiere Sistemas y BD locales.
- Se podría reutilizar el personal del área IT, y Hardware existente.
- Las soluciones pueden ser más específicas ya que el Hardware y Software es del usuario.
- El usuario controla completamente la infraestructura, y los sistemas de información. Se facilita el control y supervisión de la seguridad y protección de los datos almacenados.

3. **Cloud Híbrido**

Aúna las dos formas de Cloud indicadas previamente,tal que las entidades están conectadas mediante tecnología estándar que permita la portabilidad de datos y aplicaciones (por ejemplo balanceo de carga,etc..).

El mayor inconveniente puede estar en la mayor complejidad de la solución ya que requiere la integración de dos tipos de nube, con interfaces quizás no compatibles,etc...

2 Objetivos

Las tecnologías de Virtualización y de Cloud Computing (Clouds Privadas, Públicas o Híbridas) permiten a las compañías poner más el foco en las áreas de negocio específicas y "realmente importantes" de cada una de ellas, y reducir drásticamente inversiones en equipamiento IT, licencias, etc.. (CAPEX) y los costes de Administración y gestión de los sistemas de Información (OPEX).

Hoy por hoy no es imprescindible montar infraestructura de servidores, salvo que se quiera expresamente. Amazon AWS arrancó años atrás y de facto ha definido servicios *IaaS*, interfaces, APIs de conexión,etc.. y un modelo de negocio exitoso que todos desean replicar, tanto con entornos comerciales como con entornos Opensource.

Mirando hacia atrás, no se echará de menos los procesos habituales del personal de IT tales como: montar los servidores en los racks, cablear la conectividad, instalar el sistema operativo, rebotar in situ servidores cuando se "cuelgan",etc... cuando lo que se requiere por ejemplo para instalar un servidor en el presente (para cualquier entorno) es tirar de appliance o sandbox, el que nos convenga, y arrancar con éstos, instancias en Amazon AWS o sistemas privados con OpenStack en cuestión de minutos.

Dentro del mercado existen diferentes soluciones Cloud, y cada una de ellas tiene sus ventajas e inconvenientes. Todos los suministradores/actores juegan una particular partida de ajedrez para posicionarse competitivamente en situación ventajosa y tener acceso a cuotas de un mercado que crecerá enormemente en los próximos años.

Como avanzaré a continuación y se expondrá más en detalle en apartados posteriores:

- Amazon AWS es líder de soluciones Cloud Publicas. Ha definido de facto interfaces que mayormente han sido adoptados.

- OpenStack es una solución OpenSource de recorrido que pretende posicionarse independiente. Tiene mucho apoyo de Red Hat y Canonical,de fabricantes (como Dell y HP) y de Proveedores de servicios Cloud como RackSpace.

Sin duda constituye una revolución en la que si no participan las empresas (y particularmente las españolas) se pueden ver abocadas a perder posiciones competitivas en sus correspondientes mercados. Sin duda las empresas tienen condicionantes que deben contemplarse, pero en un escenario sencillo, cualquier empresa tiene menor coste utilizando Amazon AWS para alojar una página web que haciendo housing de un servidor propio o incluso hosting de un servidor dedicado en cualquier Datacenter.

Así pues, en líneas generales este Proyecto PFM pretende acercar al Lector de una forma eminentemente práctica y operativa a las dos soluciones Cloud IaaS mencionadas (OpenStack y Amazon AWS) y para ello:

- Demostrar la operativa, rendimiento, Gestion,etc.. de un entorno real de producción, que sea escalable basado en OpenStack. Para ello se instalará una solución completa distribuida con servidores y matrices de almacenamiento DELL. La solución debe estar visible para su evaluación por los Tutores.

- Demostrar la operativa de Amazon AWS y sus diferentes servicios. Validación real de los mismos.

- Comparar Ambos entornos (OpenStack y Amazon). Analizar el uso de OpenStack y Amazon AWS en Cloud Hibridos, y comparar con otras alternativas de OpenStack.

- La documentación del objetivo tiene que ser concreta y suficiente, que permita al lector ganar confianza de cara al uso y explotación real de cualquiera o ambos entornos.

3 Estudio de Viabilidad

3.1 Necesidades y requisitos del cliente

La empresa Artic S.L. es una empresa de tamaño medio que opera como ISP durante los últimos 3 años en el mercado español ofreciendo servicios de hosting de todo tipo de soluciones Opensource para Pymes. Se enfrenta a diferentes problemas como consecuencia de su rápido crecimiento y de los cambios radicales sufridos en el mercado español.

Principalmente:

- La Empresa Artic S.L. ha experimentado un crecimiento considerable en un corto período de tiempo que ha conducido a un crecimiento desordenado y poco planificado. La arquitectura es lineal en equipamiento y recursos humanos y no es sostenible. Se requiere urgentemente introducir una nueva arquitectura:
  - Que sean más gestionable y con menos recursos humanos.
  - De costes operativo reducido. (por ejemplo costes eléctrico y/o espacio Datacenter).
  - Que sea escalable y flexible para afrontar crecimientos futuros. Idealmente arquitecturas híbridas con Hw local propio y Hw remoto (de otros proveedores).
  - Que permitan la migración de servicios de la arquitectura de clientes inicial a la nueva, sin riesgos y con facilidad. Incluida la migración de clientes con servicios en competidores.
  - Que permita plazos de implantación de nuevos proyectos reducidos.
  - Que permitan recuperación rápida frente a fallos para ofrecer a los clientes un buen SLA operando en régimen de 24*7 con alta disponibilidad.
- El entorno competitivo es muy duro en oferta, y requieren estructura de coste flexible orientada a negocio generado, a la par que la capacidad de endeudamiento de la empresa impide nuevas inversiones en HW o SW.
- La Empresa Artic S.L. desea, apoyándose en la nueva infraestructura interna (y externa), desarrollar nueva gama de servicios para ofrecer a sus clientes.

El proyecto constará del análisis, diseño e implantación completa de la nueva arquitectura que será altamente escalable y flexible con bajos costes operativos. La arquitectura deberá proporcionar servidores virtuales de diferentes prestaciones en CPU, memoria, disco duro, interfaces,etc..., con provisión online y con bajos costes operativos.

El proyecto definirá los mecanismos operativos que deberán seguirse para aumentar o reducir la capacidad de computación del sistema según la evolución del negocio.

Se hará especial énfasis en la cuestión relativa a la seguridad ya que es uno de los aspectos claves en los Servicios Cloud. Toda la infraestructura debe garantizar que se pueda cumplir integralmente con la LOPD (Ley Orgánica de Protección de Datos).

En la medida de lo posible debe utilizarse Software libre.

3.2 Análisis de la situación actual

La infraestructura informática con al que cuenta la Empresa Artic S.L. para sus servicios internos y externos se basa en una arquitectura lineal de servidores Dell 810R y Dell 720R, cada uno de los cuales tiene funciones especificas.

El crecimiento de los servicios internos y/o externos se realiza añadiendo servidores adicionales linealmente.

La empresa Artic S.L. dispone de dos proveedores de internet independiente que proporcionan la conectividad mediante conexiones BGP. Las conexiones (ambas) son fast-ethernet con un caudal de cada conexión de 100 megabits/sg.

Nota: La empresa Artic S.L. dispone directamente de direccionamiento otorgado por RIPE y dispone igualmente de AS para routing BGP.

Detrás de router :

- existen una serie de Firewalls Cisco ASA 5520 que crean filtrado para agrupaciones de servidores según reglas especificas. Todo ello orientado a servicios gestionados de VPN y Firewall.
- Otros servidores se conectan directamente a la conexión que entrega el router a switches nivel 2. En este caso el filtrado debe hacerse en el propio servidor (por ejemplo con iptables).

A continuación se muestra el inventario del hardware que posee la Empresa Artic S.L.:

1) Hardware

- 2 Routers Cisco 7606. Interconexión a internet con BGP.
- 2 Firewalls Cisco Asa 5520
- 4 Switches Cisco 3550-24 ports
- 50 Servidores Dell 210R con 4GB de memoria RAM, 2,40GHz y 120 GB de espacio de almacenamiento (disco duro).
  - 45 Servidores son utilizados por servicios de cliente.
  - 5 Servidores son utilizados por servicios de la Empresa Artic S.L.
- 2 Servidores Dell 720R con 32 GB de memoria RAM, 3,0 GHZ y 600 GB de espacio de almacenamiento (disco duro).
  - 2 Servidores son utilizados por servicios de la Empresa Artic S.L.

2) Software

En cuanto al software que se utiliza en los distintos servidores:

- Hay 5 servidores con Windows 2003 Server.
- Hay 30 servidores con Centos 6.4
- Hay 17 servidores con Ubuntu 12.04.3 LTS

3) Backups

Todos los servicios de backup se realizan sobre unidades NFS montadas sobre equipos QNAP TAS421 de 4 discos de 2 Teras.

Se dispone de 4 unidades.

3.3 Definición de los requisitos del sistema

De la información obtenida del cliente podemos sintetizar los principales requisitos que la solución deberá ofrecer:

- El nuevo Sistema supondrá renovar la arquitectura tecnológica de la infraestructura existente. Pasar de una solución inicial de servidores lineal a una solución de virtualización/con software gestión Cloud dentro de las existentes en el mercado.

- El nuevo Sistema dispondrá de herramientas gráficas asociadas a la arquitectura que se incorpore que faciliten la gestión y administración. La carga de trabajo de operación hombres/máquina debe reducirse en al menos un 50%.

- El nuevo Sistema dispondrá de herramientas de monitorización automáticas para detección de cualquier problema de forma automática.

- El nuevo sistema supondrá una reducción del consumo eléctrico en un 80% al menos y del espacio ocupado de los servidores en un 75%.

- El nuevo Sistema será escalable en cuanto a computación y almacenamiento de la infraestructura inicial existente sobre proveedores externos tipo Amazon, Rackspace,etc... La capacidad inicial a instalar de computación y almacenamiento local es la existente en el momento de arrancar el proyecto + un 35% de sobredimensionamiento.
  - Debe soportar 65 servidores (65 VM) con un promedio de 2 GB de RAM y con un almacenamiento total de 8 Teras de Disco duro.

- El nuevo Sistema facilitará la instalación/provisión de servidores y proyectos específicos para clientes internos y externos en plazos de tiempo reducidos inferiores a 1 hora (sobre imágenes predefinidas).

- El nuevo Sistema dispondrá de API programable para la gestión de todo el entorno: para listar Servidores Virtuales, iniciarlos, pararlos, crearlos,etc... Incluso para la provisión automática desde una web de venta de servicios online.

- El nuevo Sistema ofrecerá mecanismos de recuperación rápida frente a eventualidades de error que pudieran darse, siempre para dar un SLA del 99.95%.

- El nuevo Sistema deberá poder instanciar tanto servidores Linux, como servidores Windows.

3.4 Estudio de alternativas de solución

El principal requisito es instalar una nueva arquitectura escalable, eficiente, y más gestionable que sustituya la arquitectura inicial, que reduzca los costes operativos. Para ello se debe considerarse tecnologías de Virtualización y de Sistemas de Gestión Cloud.

Además debe ser un sistema escalable (mayormente cuando se sobrepase la capacidad local) sobre infraestructura IaaS externa.

Podríamos indicar que las alternativas que se plantean tienen varios ámbitos de decisión:

- Hardware del Servidor.
- Almacenamiento local.
- Sistema de Virtualización.
- Gestor de la Nube
- Infraestructura externa para obtener escalabilidad sin inversión.
- Sistemas de monitorización y gestión

- En una primera instancia y en lo que se refiere al Hardware de servidor:
  - Una alternativa inicial sería la compra de equipamiento nuevo. Nuevos servidores para montar la arquitectura.
  - Una segunda alternativa conllevaría la reutilización del Hardware existente, ampliando unicamente en éstos memoria y discos duros de almacenamiento local.

- En lo que se refiere al almacenamiento local existen distintas opciones de matrices de almacenamiento, que se tendría que adquirir al no existir en la arquitectura antigua.
  - Arquitectura ISCSI
  - Arquitectura Fibre Channel

- Hay diferentes alternativas de entornos de Virtualización y de Software Libre, pero las más relevantes son: KVM y XEN.

Una comparativa de ambos sistemas de Virtualización va a continuación.

	full virt	paravirt	containers (OS virt)	license	architectures	performance	SMP guests	CPU / memory hotplug	standalone host	notes
Xen	OK	OK		GPL	i686, x86-64, IA64, PPC	paravirt very fast, full virt medium	OK	OK	OK	full virt needs VT / AMD-V
KVM	OK	OK		GPL	i686, x86-64, IA64, PPC, S390	paravirt very fast, full virt medium	OK	OK		full and para virt need VT / AMD-V, upstream

- En cuanto a los sistemas de Gestión Cloud de Software Libre

Existen diferentes soluciones OpenSource. Destacan OpenStack, OpenNebula, Eucalytus y CloudStack.

Table 1: Feature comparison of IaaS frameworks. ✓ indicates a positive evaluation. The more checkmarks the better.

	OpenStack	Eucalyptus 2.0	Nimbus	OpenNebula
Interfaces	EC2 and S3, Rest Interface. Working on OCCI ✓✓	EC2 and S3, Rest Interface. Working on OCCI ✓✓	EC2 and S3, Rest Interface ✓	Native XML/RPC, EC2 and S3, OCCI, Rest Interface ✓✓✓
Hypervisor	KVM, XEN, VMware Vsphere, LXC, UML and MS HyperV ✓✓✓	KVM and XEN. VMWare in the enterprise edition. ✓✓	KVM and XEN ✓	KVM, XEN and VMWare ✓✓
Networking	- Two modes: (a) Flat networking (b) VLAN networking -Creates Bridges automatically -Uses IP forwarding for public IP -VMs only have private IPs ✓✓✓	- Four modes: (a) managed; (b) managed-novLAN; (c) system; and (d) static - In (a) & (b) bridges are created automatically - IP forwarding for public IP -VMs only have private IPs ✓✓✓	- IP assigned using a DHCP server that can be configured in two ways. - Bridges must exists in the compute nodes ✓✓	- Networks can be defined to support Ebtable, Open vSwitch and 802.1Q tagging -Bridges must exists in the compute nodes -IP are setup inside VM ✓✓✓
Software deployment	- Software is composed by component that can be placed in different machines. - Compute nodes need to install OpenStack software ✓	- Software is composed by component that can be placed in different machines. - Compute nodes need to install OpenStack software ✓	Software is installed in frontend and compute nodes ✓✓	Software is installed in frontend ✓✓✓
DevOps deployment	Chef, Crowbar, Puppet ✓✓✓	Chef*, Puppet* ✓ (*according to vendor)	no	Chef, Puppet ✓✓
Storage (Image Transference)	- Swift (http/s) - Unix filesystem (ssh) ✓	Walrus (http/s) ✓	Cumulus (http/https) ✓	Unix Filesystem (ssh, shared filesystem or LVM with CoW) ✓
Authentication	X509 credentials, LDAP ✓✓✓	X509 credentials ✓	X509 credentials, Grids ✓✓	X509 credential, ssh rsa keypair, password, LDAP ✓✓✓
Avg. Release Frequency	<4month	>4 month ✓	<4 month	>6 month ✓
License	OpenSource - Apache ✓	OpenSource ≠ Commercial	OpenSource Apache ✓	OpenSource Apache ✓

- En cuanto a los Servicios Externos para escalado existentes en el mercado

Distinguimos entre Computación y Almacenamiento. Las dos compañías lideres del Mercado son:
a) Amazon AWS

- EC2 - Computación
- S3 – Almacenamiento
- Glacier – Backups

b) Rackspace

- Cloud Servers – Computación
- Cloud Files – Almacenamiento
- Swiftstack – Almacenamiento (Swift Openstack) **http://swiftstack.com/**

A continuación en la siguiente tabla figura una comparativa de Costes (Computación) de Amazon-Ec2 versus Rackspace. Cabe subrayarse que los modelos de costes son similares y del tipo Pago por uso, tal que si se comprometen períodos de tiempo más amplios los costes unitarios por hora se reducen. Son aspectos que deben tratarse cuidadosamente teniendo en cuenta los perfiles de consumo y uso de las plataformas (computación,almacenamiento,transferencia de datos,etc...) para obtener unos costes ajustados.

	Rackspace – Cloud Servers	**Amazon -EC2**
Uptime / Availability Guarantee	100%	99.95%
Time span	Current period	"service year" or the preceding 365 days
Time-to-resolution	1 hour	Not specified
Credits	· 5% of the fees for each 30 minutes of network or data center downtime, up to 100% of the fees· 5% of the fees for each additional hour of downtime past time-to-resolve, up to 100% of the fees	10% of bill per eligible credit period
Notification onus	Customer	Customer
Window	30 days after incident	30 days after incident
	Rackspace – Cloud Files	**Amazon – S3**
Availability	99.9%	99.9%
Definition	(i) The Rackspace Cloud network is down, or (ii) the Cloud Files service returns a server error response to a valid user request during two or more consecutive 90 second intervals, or (iii) the Content Delivery Network fails to deliver an average download time for a 1-byte reference document of 0.3 seconds or less, as measured by The Rackspace Cloud's third party measuring service.	"Error Rate" means: (i) the total number of internal server errors returned by Amazon S3 as error status "InternalError" or "ServiceUnavailable" divided by (ii) the total number of requests during that five minute period. We will calculate the Error Rate for each Amazon S3 account as a percentage for each five minute period in the monthly billing cycle.
Credits	99.89% – 99.5% 10% 99.49% – 99.0% 25% 98.99% – 98.0% 40% 97.99% – 97.5% 55% 97.49% – 97.0% 70% 96.99% – 96.5% 85% Less than 96.5% 100%	99 % – 99.9% 10% Less than 99% 25%

- Finalmente en cuanto a los sistemas de monitorización:

Existen multitud de opciones de monitorización, pero se consideran como alternativas para su evaluación:

- La operación integrada de Munin y Nagios.
- Ganglia
- Zenoss.

3.4.1.1 Modelos de costes servicios IaaS

Es particulamente importante entender con detalle, e incluso simular, los costes en los que se incurriría contratando servicios de proveedores externos. Amazon AWS lleva años de ventaja y de facto ha establecido modalidades de facturación, que en mayor o menor medida, todos replican, aun cuando los precios y/o servicios puedan diferir.

Amazon AWS facilita incluso vía web un simulador para poder estimar los costes en los que se incurre para los diferentes servicios: http://calculator.s3.amazonaws.com/calc5.html
Nota: Los precios cambian con relativa frecuencia, y es importante la validación continua de los modelos.

Típicamente los costes de los servicios de Computación (EC2) dependen de:

- Zona geográfica de contratación.
- Tipo Instancia contratada (diferentes capacidades y prestaciones)
  - Tiempo de uso recursos. (*)
- Tipo de Volumen contratado
  - Tiempo de uso recursos.(*)

- Ips asignadas
- Tráfico (medido en GB).
- EBS (Balanceadores – unidades y GB).

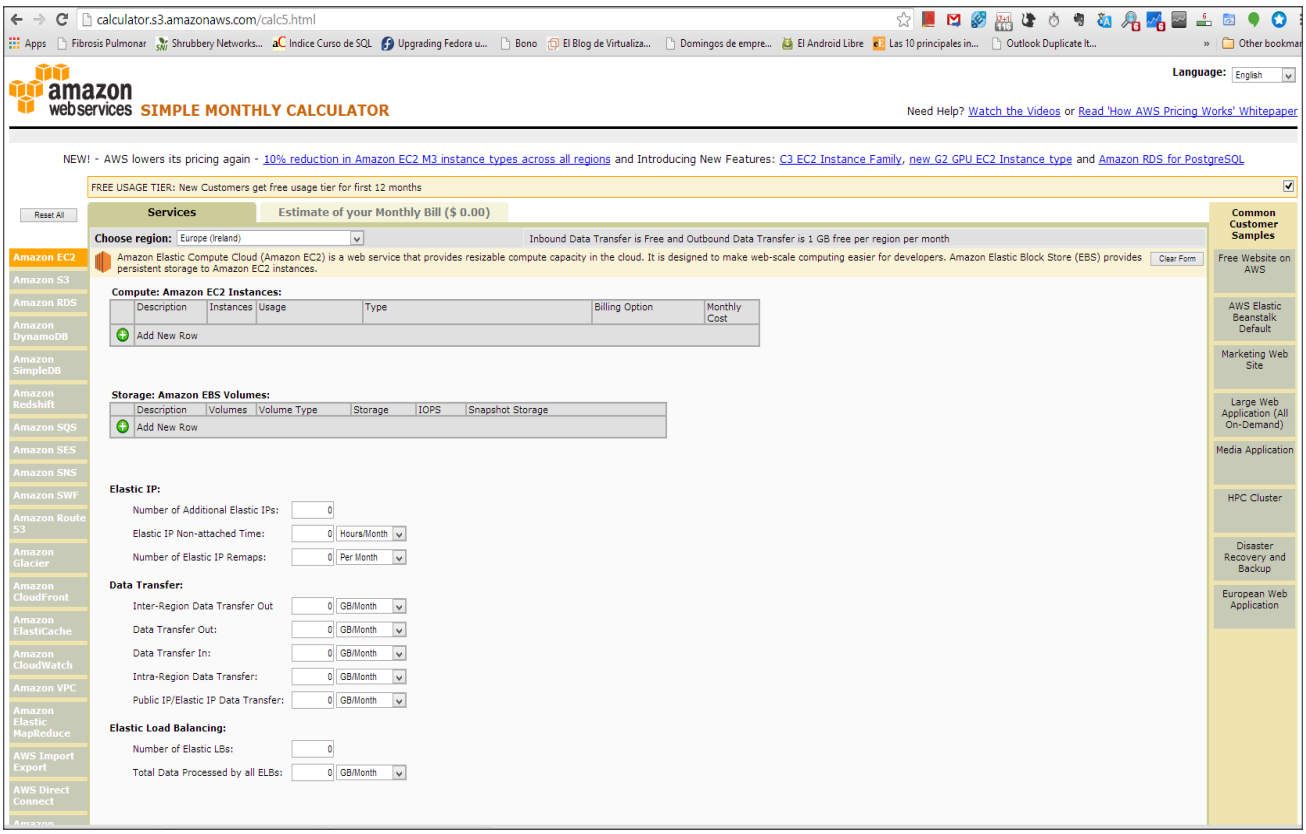
Para almacenamiento (S3) depende de:

- Tamaño
- Tipo de Redundancia del almacenamiento.
- Peticiones escritura/lectura
- Tráfico (medido en GB).

(*) La contratación tiene costes que fluctúan en la contratación según el pago sea comprometido por horas, meses, años, etc.. **Es crucial la correcta estimación de los recursos necesarios para minimizar los costes operativos.**

Hay diferentes consideraciones igualmente a considerar, que he obviado previamente, en cuanto al origen y destino del tráfico que tiene costes diferentes, pero que sin duda son de análisis obligatorio para obtener una previsión exacta de los costes.

La siguiente imagen corresponde al simulador que es extremadamente útil. Otros proveedores ofrecen interfaces similares para realizar la correcta planificación.



3.5 Valoración y elección de las posibles soluciones

A continuación y dentro de cada uno de los planos de decisión vistos previamente se han hecho consideraciones sobre cada opción que conducen a la elección de la opción más ajustada a los objetivos que persigue el proyecto.

- **Hardware del Servidor.**
  - Se opta por reutilizar todo el hardware posible de mayor capacidad de computación, principalmente los servidores DELL R720 a los que se hará un upgrade memoria a 128 GB, y almacenamiento local a 2 Teras.
  - Igualmente sobre 6 servidores DELL R210 tendrán upgrades a 16 a GB.
  - Se mantiene como proveedor de servidores DELL por mantener un "parque" de hardware homogéneo.

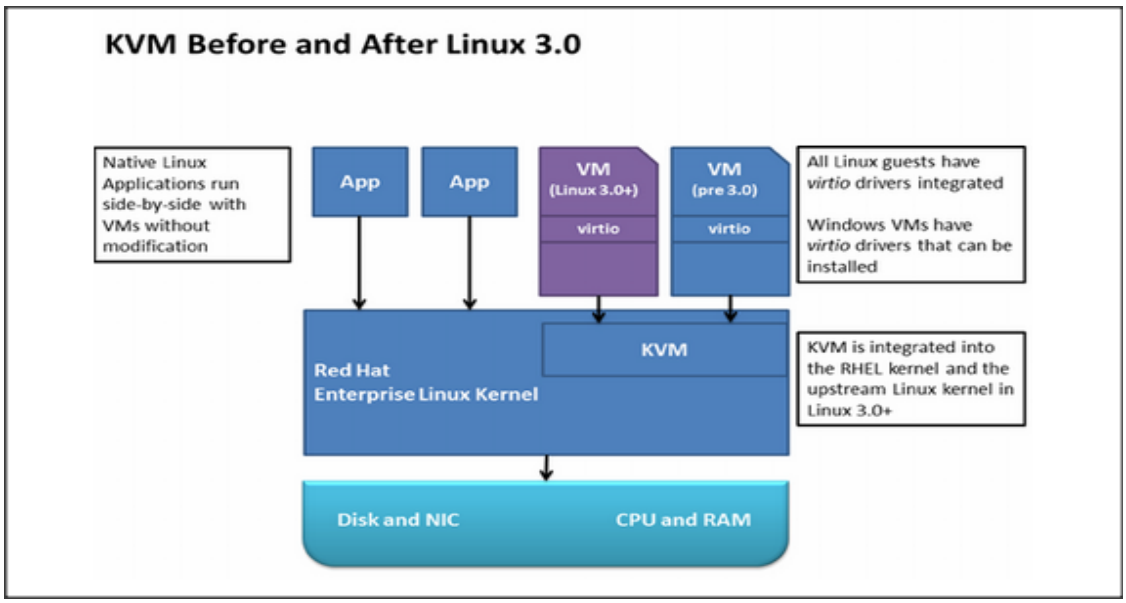
http://www.dell.com/es/grandes-corporaciones/p/poweredge-r720/pd

http://www.dell.com/es/grandes-corporaciones/p/poweredge-r210-2/pd

- **Almacenamiento local.**
  - Se opta por matrices de almacenamiento ISCI.
    ISCSI frente a Fibre Channel destaca por su sencillez de implantación y bajo coste. La tecnología ha evolucionado tal que las diferencias de rendimiento se han reducido considerablemente en los últimos años y es una buena opción.
    La maquina elegida es la PowerVault-md3200i.

http://www.dell.com/es/grandes-corporaciones/p/powervault-md32x0i-series/pd

- **Sistema de Virtualización.**
  - Se opta por KVM.
    Ambas opciones XEN, y KVM son interesantes, pero consideramos KVM de más fácil uso, gestión e integración. KVM es el hipervisor del kernel de Linux y puede tener sin problemas VM bajo Linux,Windows, Solaris y BSD. Su integración en Linux es una ventaja significativa.

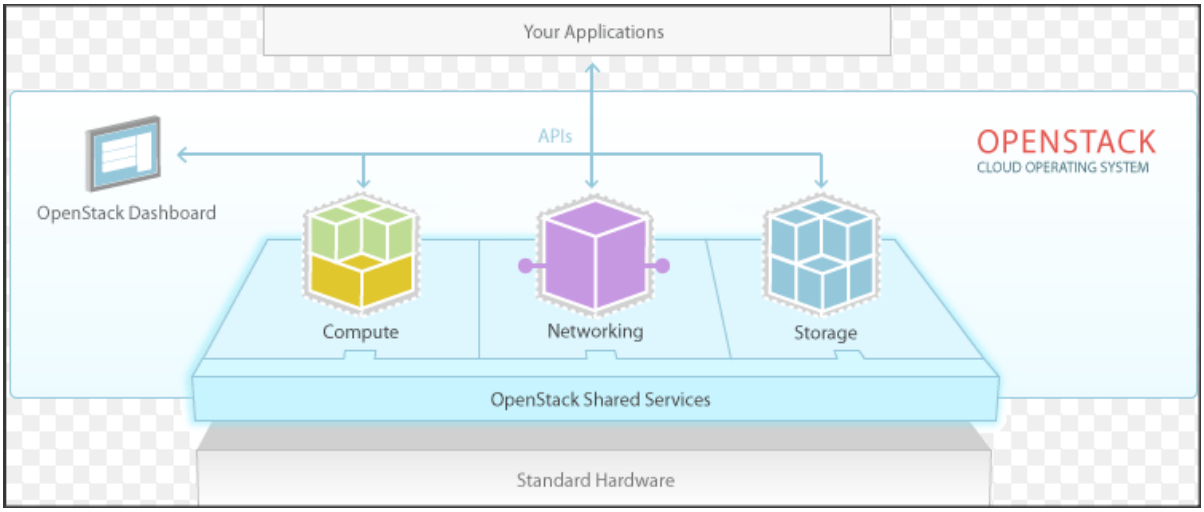


- **Gestor de la Nube**
  - Se opta por por OpenStack.

    Openstack es la solución Cloud Opensource más activa. Está promocionada entre otros por RedHat, Rackspace, HP, DELL,etc... y una gran comunidad global de empresas, desarrolladores,etc... El proyecto tiene como objetivo ofrecer soluciones para todos los tipos de nube, creando un estándar de la industria.

    Fundada por Rackspace Hosting y la NASA, OpenStack ha crecido hasta convertirse en una comunidad global de desarrolladores de software que colaboran en un sistema operativo estándar, abierto y escalable. Su misión es permitir a cualquier organización crear y ofrecer servicios Cloud que se puedan ejecutar en hardware estándar.

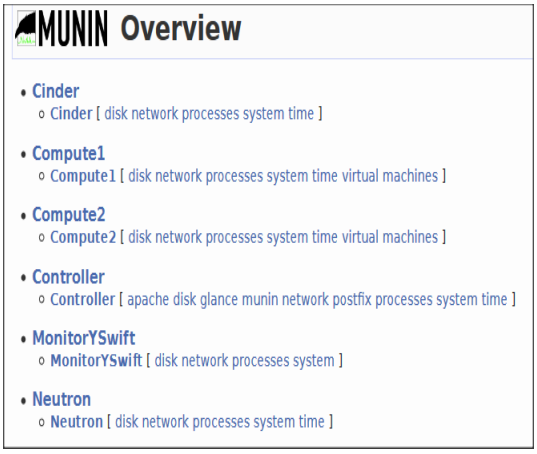
    Es probablemente la opción Opensource que tiene más opciones de ser una alternativa frente al apabullante dominio Cloud de Amazon AWS. Eso ha motivado su elección.

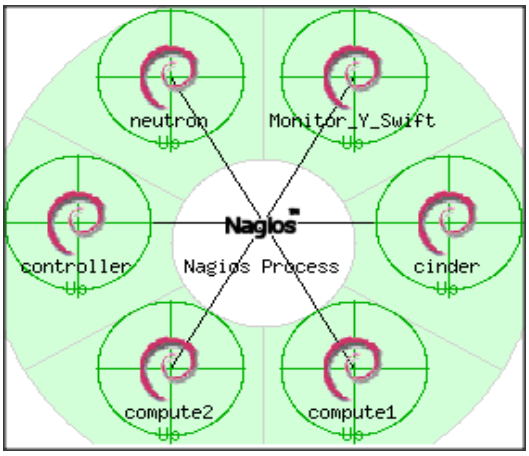


- **Infraestructura externa para obtener escalabilidad.**
  Siempre bajo demanda:
  - Se opta por usar para almacenamiento Amazon AWS (S3) y SwiftStack (Openstack Swift – Proveedor USA).
  - Se opta por usar para capacidad de Computación a Amazon AWS.

- **Sistemas de monitorización y gestión.**
  - Se opta por utilizar conjuntamente Munin y Nagios.

    Ambas herramientas son Opensource, son sencillas de gestión y permiten tanto la visualización gráfica de uso de recursos (todo tipo de recursos) de todas las entidades y procesos del sistema en ejecución como la monitorización de cada recursos, proceso, servicio,etc.. con la cadencia que se establezca y el disparo de alarmas para avisar, reiniciar servicios,etc...





4 Análisis del Sistema

4.1 Definición del Sistema

El Sistema objeto del proyecto será una solución IaaS basada en OpenStack de alta disponibilidad. Nota: Desde la perspectiva de la Arquitectura todos los entornos son parecidos y contienen las mismas "piezas" o "componentes".

OpenStack es un software OpenSource usado para la construcción de Clouds públicas y privadas. Es una colección de proyectos de software, a través de estos proyectos y/o componentes, OpenStack proporciona una completa plataforma operativa para la administración y gestión de Clouds.

La misión principal del Sistema objeto del Proyecto es proporcionar un entorno que cubra el ciclo completo de despliegues de servicio IasS y que proporcione el poder desplegar de forma sencilla, escalable, elástica y de cualquier tamaño, tanto Clouds públicos como Clouds privados.

Los componentes principales de la arquitectura Openstack son:

- ***OpenStack Compute (nova)*** es el controlador de la estructura básica del Cloud. Es el encargado de iniciar las instancias (máquinas virtuales) de los usuarios y grupos. También es el servicio encargado de la gestión de la red virtual para cada instancia o para las múltiples instancias que formen parte de un proyecto (tenant).

- ***OpenStack Object Storage (swift)*** es el servicio encargado del almacenamiento masivo de objetos a través de un sistema escalable, redundante y tolerante a fallos. Las posibles aplicaciones de Object Storage son numerosas, como por ejemplo: almacenamiento simple de ficheros, copias de seguridad, almacenamiento de streamings de audio/vídeo, almacenamiento secundario/ terciario, desarrollo de nuevas aplicaciones con almacenamiento integrado, etc.

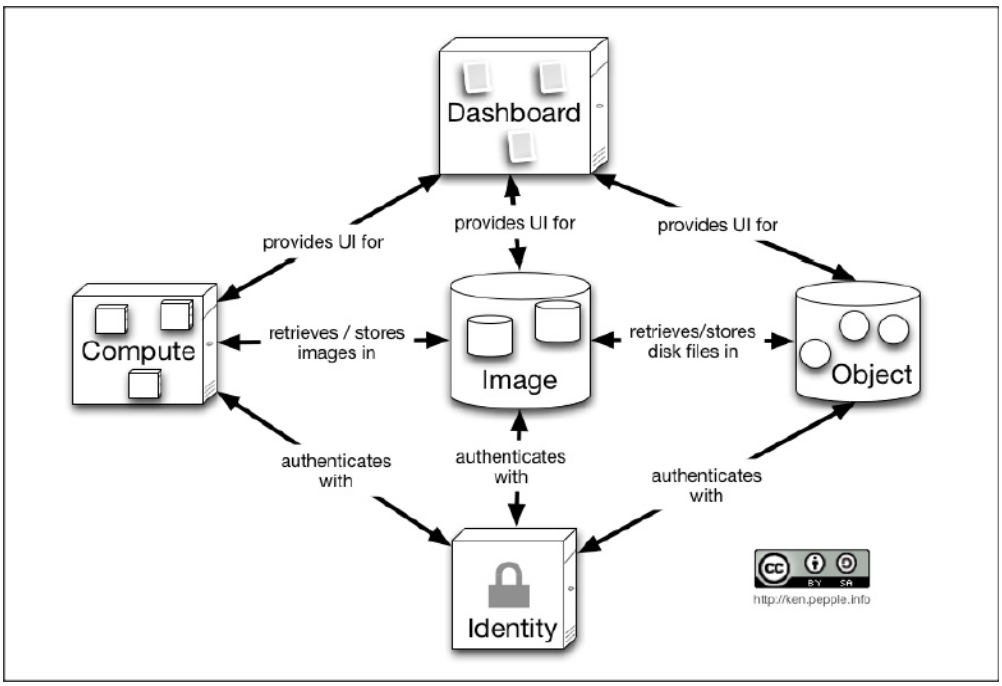
- ***OpenStack Identity Service (keystone)*** es un servicio usado para la autenticación entre el resto de componentes. Este servicio utiliza un sistema de autenticación basado en tokens y se incorporó en la versión 2012.1 de OpenStack.

- ***OpenStack Image Service (glance)*** es un servicio para la búsqueda y recuperación de imágenes de máquinas virtuales. Este servicio puede almacenar las imágenes directamente o utilizar mecanismos más avanzados como: usar Object Storage como servicio de almacenamiento, usar Amazon's Simple Storage Solution (S3) directamente, ó usar Object Storage como almacenamiento intermedio de S3.

.

- ***OpenStack Dashboard (Horizon)*** es un panel web para el manejo de instancias y volúmenes. Este servicio es realmente una aplicación web desarrollada en django que permite comunicarse con las diferentes APIs de OpenStack de una forma sencilla. OpenStack Dashboard es fundamental para usuarios noveles y en general para realizar acciones sencillas sobre las instancias.

El siguiente diagrama muestra las relaciones entre los componentes principales (Nova, Glance y Swift), cómo están relacionados y cómo pueden cumplir los objetivos propuestos por OpenStack para el despliegue de infraestructuras de cloud computing.



En la figura quedan claras las siguientes relaciones:
• *Horizon* proporciona un frontal gráfico basado en web para la gestión del resto de servicios de OpenStack

• Nova almacena y recupera imágenes de discos virtuales y sus datos asociados (metadatos) a través del servicio *Glance*.

• *Glance* almacena las imágenes en un directorio en disco, pero puede hacerlo a través del servicio *Swift*.

• El servicio *Keystone* es el encargado de la autenticación de todos los servicios.

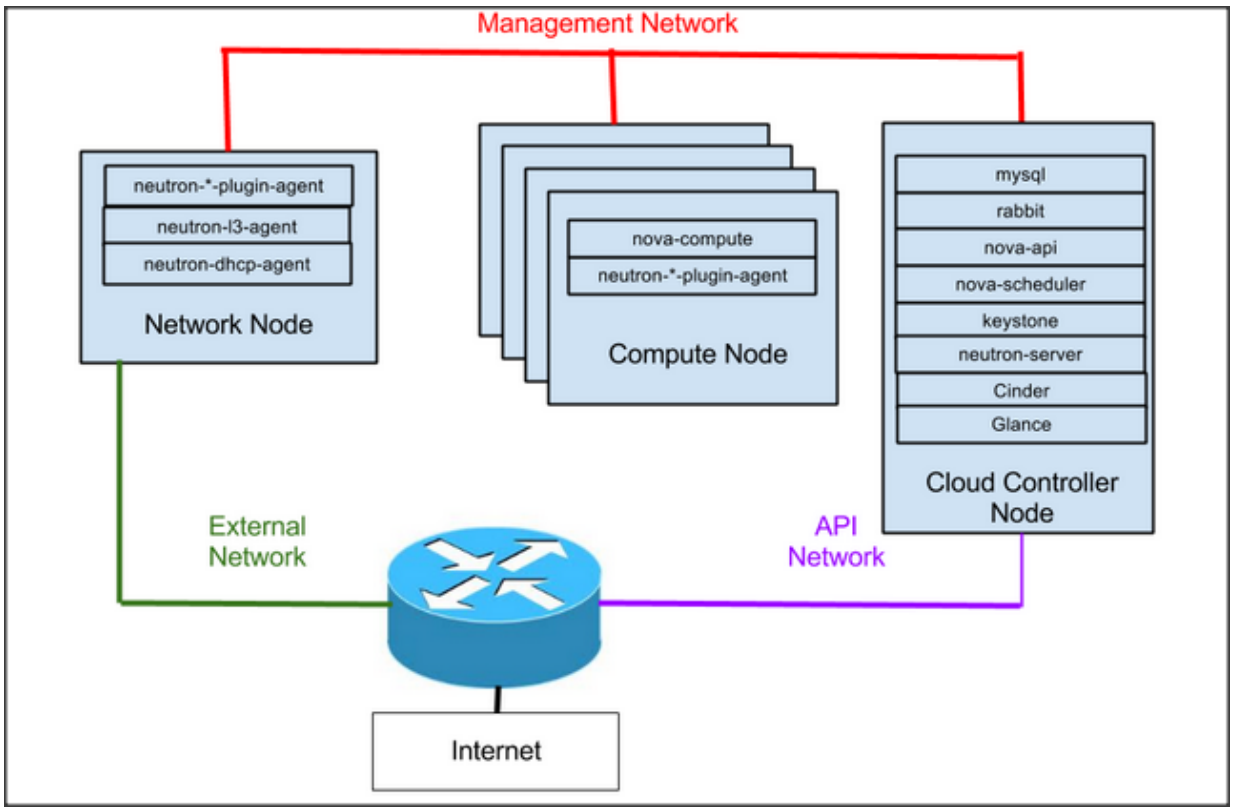
Esta es una visión muy simplificada de toda la arquitectura, asumiendo además que utilicemos todos los servicios. Por otro lado, muestra únicamente el lado "operador" del cloud, la imagen no representa cómo los consumidores del Cloud pueden realmente usarlo, ya que por ejemplo, se puede hacer uso del servicio Swift de forma directa.

Los componentes mencionados se distribuirán en nodos/servidores a efectos de garantizar que el sistema sea robusto, redundante y escalable. El diseño repartirá los procesos y funciones entre el grupo de servidores siempre buscando el reparto de carga que permita la operación más estable,redundante y escalable posibles. Este reparto y asignación es crucial para garantizar la no existencia de cuellos de botella, por ejemplo a la hora de lanzar instancias, o crear volúmenes, etc...

La siguiente figura refleja un esquema funcional que nos muestra un reparto de los Componentes de Openstack y sus procesos que podría ser una primera aproximación que se considerará como base para el diseño que se aplique finalmente.

Se consideran varios tipos de Servidores en la definición del sistema que son función de su uso:
a) Servidores de Control (en sentido genérico)
b) Servidores de Computación.
c) Servidores de Almacenamiento convencional
d) Servidores de Almacenamiento de objetos
e) Servidores de Monitorización.

En el esquema siguiente sólo aparecen 3 tipos de servidores: Controller, Computación y Red.



La arquitectura necesaria debe cumplir con los requisitos del proyecto. Entre estos:

- La arquitectura planteada permitirá gestionar Online nubes públicas o privadas con múltiples servidores virtualizados de prestaciones diferentes.
- El sistema tendrá interfaces de provisión automáticas que reducirán drásticamente la atención de personal especializado. Los tiempos de provisión serían cuasi-inmediatos.
- La monitorización de cada uno de los sistemas será automática, teniendo igual que en el punto anterior, un impacto directo sobre la reducción de atención humana a la infraestructura.
- La reducción drástica del número de servidores tendrá un impacto directo en la reducción del consumo eléctrico y de la ocupación de espacio en el Datacenter.
- El nuevo sistema sería escalable tanto en almacenamiento como computación.
- Los Apis de OpenStack estarían disponibles a efectos de que se pudieran programar todo tipo de tareas, también de backups o de snapshots para cubrir contingencias,etc... En este último caso con el fin de garantizar SLAs del 99,95% en casos de problemas severos.

4.2 Requisitos del Proyecto

A continuación se detallan los requisitos del proyecto. La mayoría de ellos ya son satisfechos por el sistema actual, y deben ser contemplados para que se sigan cumpliendo con el nuevo sistema:

• Requisitos legales.

La informacioń mantenida por los sistemas informáticos de la Empresa Artic S.L. se encuadrán en el nivel básico y medio definidos por la LOPD (Ley Orgánica de Protección de Datos). A tal efecto el nuevo sistema debe cumplir con todas las normas definidas en esta Ley.

Los Clientes que hagan uso de la infraestructura asumirán en el contrato con la Empresa Artic S.L. la responsabilidad que les corresponda por el uso de infraestructura informática con conexión directa a Internet.

Nota: Todos los servicios de la Empresa Artic S.L. tendrán mecanismos de Firewall para impedir el acceso. Los servicios de los Clientes de la Empresa Artic S.L. asumirán la responsabilidad que les corresponda técnicamente en la implementación de estos filtros en sus Máquinas Virtuales.

• Requisitos de propiedad intelectual y licencias.

Todo el software utilizado por el Sistema se apoya en Software Libre.

• Requisitos de acceso único.

Los datos de la Empresa Artic S.L. y sus clientes se almacenarán sobre servicios distribuidos de archivos mediante NFS. Está opción contemplará los requisitos de seguridad para la aceptación unicamente de servicios correspondientes de la máquinas clientes y con los criterios adecuados en cuanto a los permisos de lectura-escritura para cada máquina cliente especifica.

• Requisitos de acceso web OpenStack DashBoard Horizon.

Los servicios de acceso a la Web de Openstack de configuración deberán contemplar la posibilidad de acceso cifrado (https) mediante SSL, ya que el acceso a algunas partes de la información deberá ser confidencial para todos los usuarios. Mediante login/password los usuarios se autentificarán y tendrán acceso a la web y a su funcinalidad.

• Requisitos sobre la base de datos del Sistema.

La base de datos del Sistema será accesible desde la red Interna de la Empresa Artic S.L. Igualmente será accesible vía servicios web desde servidores en la parte pública, y siempre con login y password.

• Requisitos del sistema de seguridad.

El Sistema tendrá restringido el acceso a las ips/puertos de las máquinas que configuran la arquitectura OpenStack, con el objeto de evitar intrusiones en estos equipos principales.
Dado que la Empresa dispone también de servidores virtualizados, éstas máquinas también tendrán el acceso restringido.

Como Firewall se utilizará en la infraestructura equipamiento CISCO ASA 5520. Las políticas de seguridad resultantes de la nueva solución han de ser idénticas a las existentes previamente.

• Requisitos de gestión de las copias de respaldo e información de los sistemas.

Esta establecido un mecanismo de backup diario (rota cada semana):

- tanto de la base de datos
- como de los ficheros relevantes.
- Como de los snapshots que se generen de las máquinas virtuales.

• Requisitos tecnológicos, mantenimiento y administración.

El personal técnico de la empresa debe recibir la formación necesaria para administrar la nueva infraestructura.

Esta formación debe estar mas enfocada en la arquitectura OpenStack y los distintos componentes de la misma. Particularmente tendría foco en el interface de Gestión Horizon para administrar la arquitectura.

• **Requisitos de organización.**

La empresa Artic S.L. tendrá que:

a) Realizar una definición del organigrama de las personas implicadas con una definición concreta de sus áreas de responsabilidad.

b) Deberá impartirse y comunicarse internamente el alcance de la solución implantada.

c) Deberán procedimentarse los mecanismos operativos que sean requeridos a todos los niveles por la nueva plataforma. Se definirán los procesos de negocio necesarios.

• **Requisitos de seguridad.**

Se implementarán herramientas de monitorización de la red y los sistemas, que permitan detectar intrusiones. Debe haber un log de eventos del sistema que permitan reconstruir lo ocurrido recientemente.

4.3 Definición de interfaces de Usuario

*** Categorías de Usuarios**

Se entiende que el usuario para gestionar el sistema se dividirá en categorías o tipos de usuario (con jerarquía de permisos y funciones):

- El tipo **Usuario** es el más básico de todos. Tiene acceso a las funcionalidades de manejo de perfiles de conexión, como crear perfiles, borrarlos y clonarlos.
  También tiene acceso a autenticarse en Openstack mediante el uso éstos perfiles.

- El tipo **Usuario Gestor** es la clase de usuario más habitual.Tiene acceso a todas las funcionalidades propias de la clase Usuario por herencia, y además puede gestionar servidores y volúmenes.
  Las operaciones típicas sobre estos recursos de Openstack serán tanto listarlos como obtener detalle de cualquier recurso.

- El tipo **Usuario Administrador** es la clase de usuario que tiene acceso a todas las funcionalidades del Sistema.
  Aparte de las propias heredadas de Usuario y Usuario Gestor, también es capaz de gestionar imágenes , sabores , usuarios y proyectos.
  El tipo de operaciones y relaciones de estos tres tipos de usuarios se verán con más detalle en el diagrama de casos de uso más adelante.

*** Tipos de Accesos**

Acceso vía SSH (Administrador servidores Anfitrión OpenStack).

Los servidores de la arquitectura de OpenStack tendrán acceso vía ssh a través de cuentas especificas.

Sólo se podrá acceder por ssh desde rangos previamente definidos.

Acceso WEB – Interface de Gestion de Openstack (HORIZON)

Horizon es el interface de usuario que se utiliza para gestionar/administrar los servicios de OpenStack. Pueden crearse instancias, imagenes, claves, gestionar volumenes con instancias, manipular contenedores de objetos Swift,etc... Además, da incluso acceso tipo VNC de consola y se puede conectar a los diferentes instancias. La funcionalidad que ofrece Horizon es la siguiente:

- Gestión de Instancias: Crear, arrancar o parar instancias; gestión de logs de consola y conexión vía VNC; “atachar” volumenes a una instancia,etc...

- Gestión de Acceso y de seguridad: Crear grupos de seguridad, crear claves, etc...
- Gestión de tipos de Instancias: Gestión de plantillas de imagenes,etc... Creación, edición, borrado,etc...
- Gestión del servicio de Catalogo
- Gestión de usuarios: Creación de usuarios, borrado, asi cuotas y uso para cada proyecto,etc...
- Gestión de volumenes: Creación de volumenes y snapshots.
- Gestión de Almacén de objetos: creación, borrado, de objetos y contenedores,etc...
- Gestión de variables de entorno para los proyectos.
- Etc...

La propia web gestiona que recursos pueden ser accedidos por un determinado usuario en función del role que tenga asignado.

Nota: Cuando se habla de interface WEB hacemos referencia a conexiones http/https desde cualquier tipo de navegador.

Acceso VNC – Interface de Gestion de Openstack (HORIZON)

Como se ha indicado previamente Horizon proporciona de facto un acceso especifico de consola tipo VNC para las máquinas virtuales. Tanto VNC, como HTML5 Spice y se pueden utilizar tanto del Dashboard del Openstack, como vía línea de comandos (uno u otro).

API Openstack (REST)

El proyecto Openstack consta de subproyectos para cada uno de los componentes de la arquitectura. Todos ellos se comunican vía API (Rest) entre sí.

La existencia de estos API propicia la programación de aplicaciones externas con todo tipo de propósitos, ya que la comunicación con cualquier componente de OpenStack es fácil y rápida.

Hay ejemplos desarrollados como: Ceilometer, moniker, heat,etc...

4.4 Especificación del plan de pruebas

La plataforma que se pondrán en operación será de naturaleza crítica ya que alojará muchos servidores virtuales, y por tanto muchos procesos/servicios que pudieran verse afectados. Es de especial criticidad por tanto la ejecución de el plan de pruebas con el máximo rigor.

El plan de pruebas pretende en los planos (pruebas unitarias y de integración) dar respuesta para validar el alcance del proyecto y al cumplimiento de los objetivos definidos y aún más a garantizar que se comporta como una plataforma estable lista para la producción.

4.4.1 Pruebas Unitarias

- **Servicios Openstack**

  La arquitectura OpenStack consta de los siguientes servicios:

- Control de Identidad (Keystone)
- Almacenamiento de objetos (Swift)
- Almacenamiento de imagenes (Glance)
- Almacenamiento de Bloques (Cinder)
- Computación (Nova)
- Red (Quantum)
- Gestión/Dashboard (Horizon)
- Orquestación (Heat)
- Medición (Ceilometer)

Se harán pruebas de validez para cada uno de los componentes, ya sea con línea de comandos o con scripts desarrollados a tal efecto (que harán uso en cada caso del api correspondiente del servicio).

- **Conectividad y operación entre los diferentes componentes servicios.**

  Debe verificarse que los distintos servicios interoperan entre si correctamente.

- **Seguridad**

Se validará individualmente el acceso de los diferentes usuarios para cada una de las jerarquías de tal modo que acepte o restrinja éstos según corresponda.

- **Monitorización**

Se trata de monitorizar individualmente cada uno de los procesos (servicios de la arquitectura indicados previamente) de forma automática y la obtención de indicaros de operación de ésto servicios que puedan mostrarse gráficamente.

- **Validación servicios Amazon y Rackspace (computación y almacenamiento)**

Individualmente y por separado se harán pruebas de los servicios de computación y de almacenamiento de Amazon y Rackspace. Se validará su correcta operativa de cara a su potencial uso en escenarios de escalabilidad híbrida con la plataforma Openstack.

4.4.2 Pruebas de Integración

- **Configuración de nubes con servidores Linux y Windows.**
  El objeto de estas pruebas conducirá a crear Proyectos de Gestión de Cloud, asignando imagenes preconfiguradas al sistema, que son utilizadas posteriormente para configurar instancias con éstas en Windows y Linux. Los sistemas configurados con plantillas predefinidas deben ser operativos tal cual los serían en hosts dedicados.
- **Configuraciones operativas (Pruebas Multinodo).**
  Se harán pruebas tanto en modo nodo único (single node) como multimodo (multinode).

- **Escenarios de escalabilidad (Aumentar Almacenamiento)**
  Se harían pruebas de escalabilidad con los escenarios en los que repentinamente necesitasemos almacenamiento que no tuviéramos en la plataforma local.

- **Escenarios de escalabilidad (Aumentar Computación).**
  Se harían pruebas de escalabilidad con los escenarios en los que repentinamente necesitásemos computación adicional que no tuviéramos en la plataforma local.

- **Escenarios de Stress/Análisis de rendimiento**
  Se harían pruebas con scripts para la creación automatica de instancias, su arranque, su detección de una forma cíclica y se vería el impacto en el uso de los recursos globales del sistema y en la operativa práctica de instancias que estuvieran en operación.

- **Análisis comparativo con OpenNebula**
  Se haría una comparativa particularmente en lo relativo a Stress/Rendimiento con un sistema gemelo en OpenNebula.

En última instancia, se intentaría si fuera posible, utilizar scripts que pudieran ejecutarse para comprobar la correcta operación de todos los módulos conjuntamente.

https://wiki.openstack.org/wiki/Openstack-integration-test-suites

Particularmente "Tempest".
http://docs.openstack.org/developer/tempest/overview.html

5 Diseño del Sistema

5.1 Arquitectura

5.1.1 Arquitectura Openstack

El entorno que se va a crear constará inicialmente de 6 servidores. En los sucesivos apartados se profundizará en la funcionalidad de cada Módulo/Subproyecto Openstack.

- *Nodo Controller* --> Controlador Cloud.
  - Servidor Controller01 Constará de:
    - Sistema Web Dashboard Openstack Horizon
    - Sistema Seguridad Keystone
    - Sistema Control de imagenes Glance
    - Sistema Orquestación Heat
    - Sistema Api Nova y Glance
- *Nodo Controller-bis* (Incluirá una maqueta de Swift – Object Storage).
    - Base de Datos y sistema de gestión de Colas (Comunicación entre sistemas).
    - Sistemas de Monitorización (Nagios, Munin)
    - Sistema Swift.
- *Nodo Network* --> Controlador de Redes. Asigna y gestiona direccionamiento IP.
  - Servidor Neutron01 Constará de:
    - Sistema Red Openstack Neutron
    - Sistema switch virtual Openvswitch.
    - Sistema Api Neutron
- *Nodo Compute* --> Controlador Computación. Las instancias de virtualización con KVM se ubicarían en este componente.
  - Servidor Compute01 y Compute02 Constara de:
    - Sistema OpenStack Nova-Compute apoyado en KVM & Libvirt.
    - Agentes de red.
    - Iniciador Open-iscsi
- *Nodo Storage*--> Controlador almacenamiento. Responsable de almacenar imagenes,volumenes,etc... que corresponden a las instancias creadas.
  - Servidor Cinder1 Constará de:
    - Sistema Openstack Cinder
    - Servidor iSCSI
    - Sistema Api Cinder
  - Opcionalmente Matriz_iSCSI01 y Matriz_iSCSI02

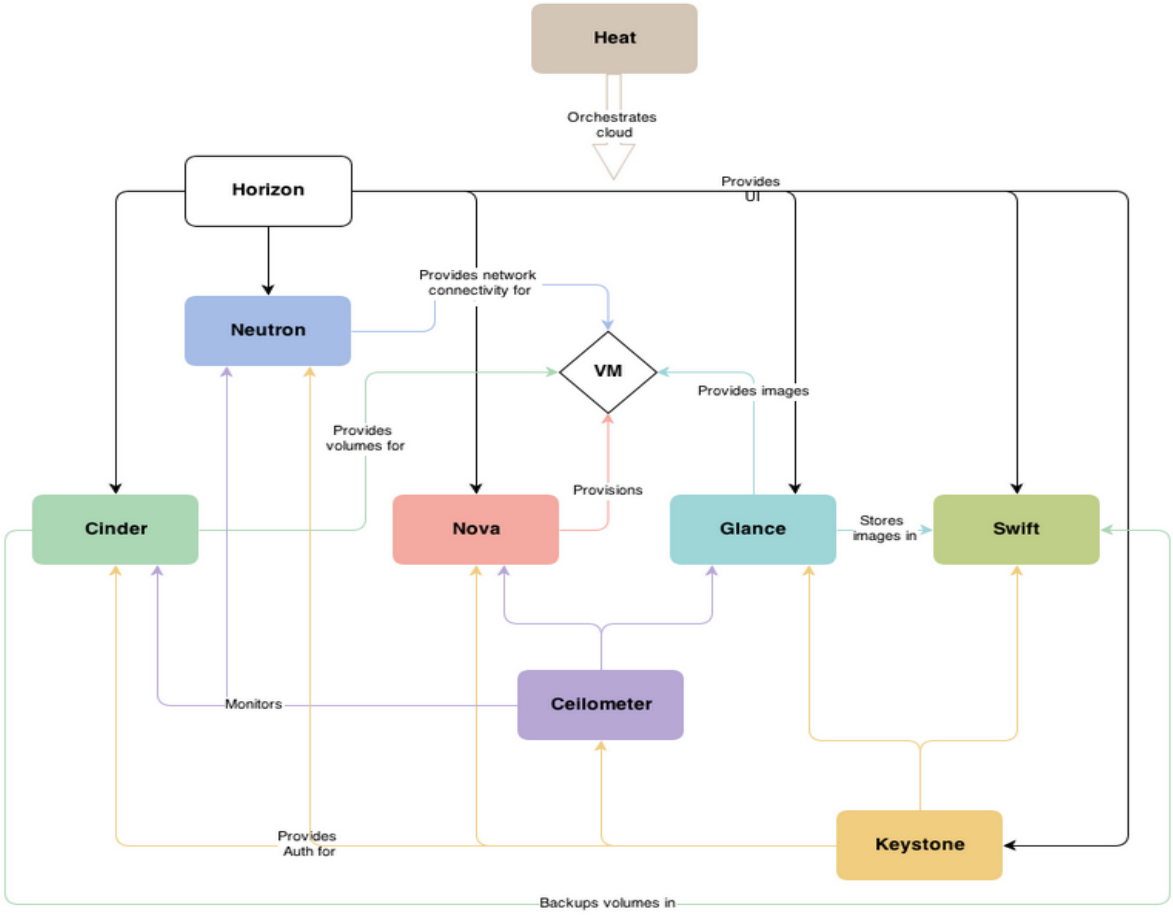
Cada uno de los items indicados serán procesos en operación que deberán monitorizarse para verificar permanentemente su disponibilidad para que el sistema opere con normalidad.

- Electrónica de Red:
  - Switches layer 2
  - Routers para conexión a internet
  - Firewalls salida a internet.

La arquitectura esta preparada para:

- Escalar en computación local.
- Escalar en almacenamiento.
- Introducir redundancia en los elementos de gestión del cloud openstack.
- Escalar externamente.

A continuación se describirá la arquitectura Openstack (que se dividirá en los servidores previamente indicados), dividida en subsistemas o módulos.



Nota: Como ya se ha dicho, la división e instalación de los componentes Openstack en los servidores, se puede hacer siguiendo múltiples criterios

5.1.1.1 Subsistema Keystone

http://docs.openstack.org/developer/keystone/

Keystone, es el componente de OpenStack encargado de la autentificación y la autorización de los distintos componentes desde la versión Essex y tiene dos funciones principales:

- Gestión de usuarios: Keystone es el encargado de mantener un registro de usuarios y los permisos que tienen cada uno de ellos.

- Registro los servicios ofrecidos: Keystone ofrece un catálogo de los servicios ofrecidos, así como la forma de acceder a sus APIs.

Los conceptos fundamentales de la gestión de usuarios son:

- **Usuario:** Podemos guardar su nombre, correo electrónico y contraseña.

- **Proyecto** (tenant en la jerga de OpenStack): En un proyecto podemos ejecutar un conjunto de instancias con características en común, por ejemplo pueden estar todas las instancias en el misma red, pueden utilizar una serie de imágenes de sistemas o tener limitado el uso de recursos del cloud.

- **Rol:** Nos indica qué operaciones puede realizar cada usuario. A un usuario se le pueden asignar diferentes roles en cada proyecto.

Los conceptos fundamentales del registro de servicio son:

- **Servicio:** Corresponde a un componente de OpenStack que puede utilizar el módulo de autentificación.
- **Endpoints:** Representa las URL que nos permiten acceder a las API de cada uno de los servicios o componentes de OpenStack

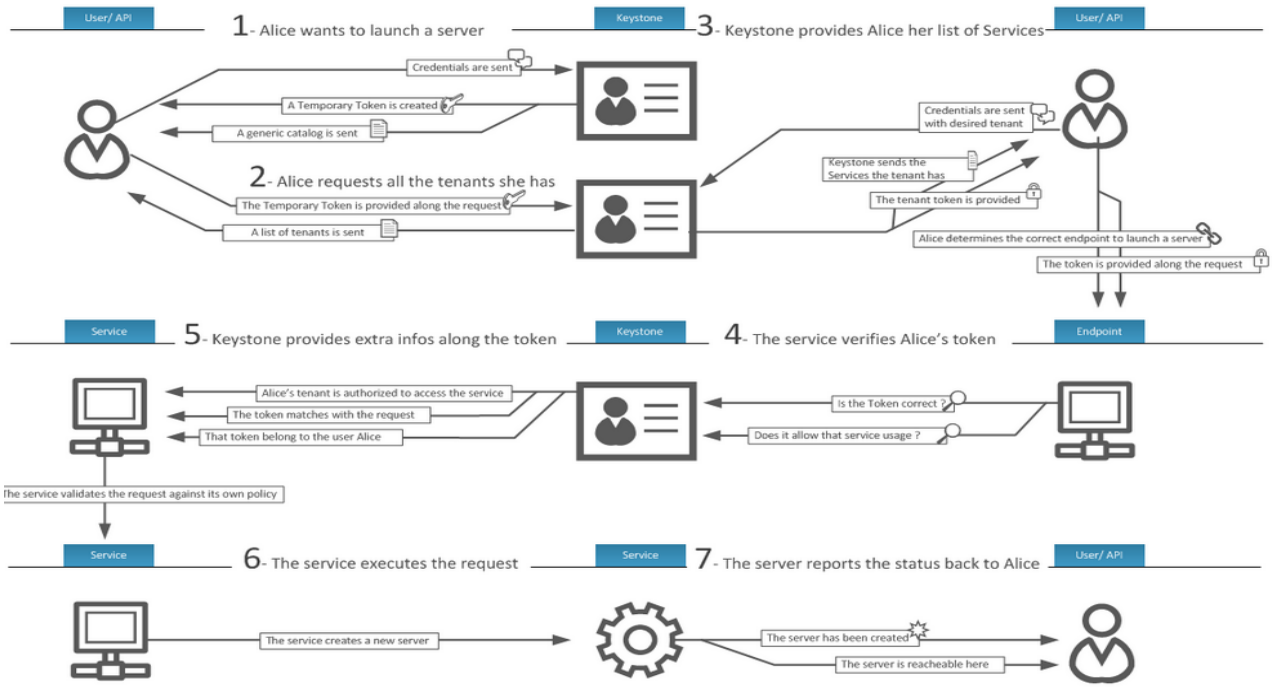
Nota: Estos conceptos serán fundamnetales al aplicar el diseño, configuración y operación del sistema.

¿Qué es el ADMIN_TOKEN y para qué se utiliza?

Keystone introduce en OpenStack un sistema de autenticación basado en tokens, de manera que todos los elementos del cloud (usuarios y servicios principalmente), no se autentican directamente unos a otros, sino que lo hace con un actor intermedio mediante tokens, este actor intermedio encargado de verificar la autenticidad de cada uno de los elementos es Keystone.

Un proceso típico de autenticación en OpenStack puede verse en la siguiente imagen, en la que se muestran los pasos que se dan desde que el usuario se acredita frente a Keystone hasta que lanza una instancia.

El siguiente diagrama muestra la operativa típica de keystone.



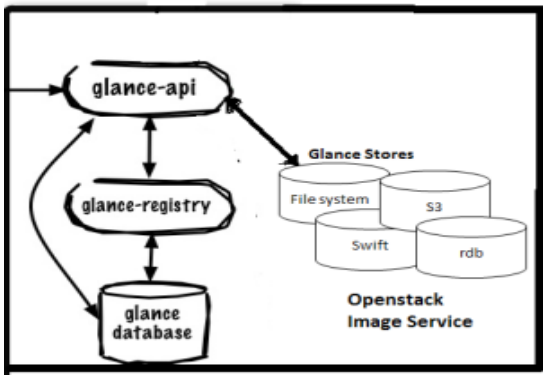
5.1.1.2 Subsistema Glance

http://docs.openstack.org/developer/glance/

El proyecto Glance proporciona los servicios necesarios para la búsqueda, localización y obtención de imágenes para las máquinas virtuales del cloud. Al igual que el resto de componentes de OpenStack, Glance posee una API RESTful que permite solicitudes tanto de los metadatos de las imágenes para las máquinas virtuales, como la solicitud en sí de una imagen.

Las imágenes que están disponibles a través de Glance, se pueden almacenar en diferentes ubicaciones, desde un simple sistema de ficheros que es la opción por defecto a un sistema de almacenamiento de objetos como OpenStack Swift.

Posiblemente glance sea el componente más sencillo de todo el conjunto de proyectos de OpenStack y el ritmo de desarrollo y cambios no tiene nada que ver con el de otros componentes.

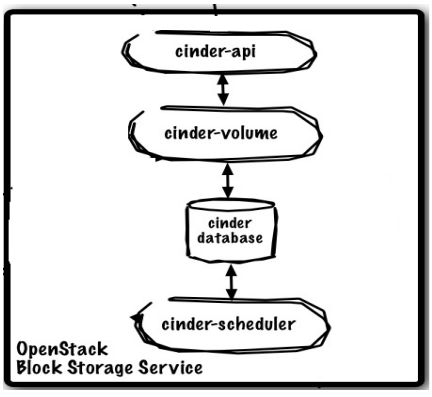


5.1.1.3 Subsistema Cinder

http://docs.openstack.org/developer/cinder/

Es el servicio de almacenamiento por volumen o bloque (block storage). Permite que las máquinas virtuales puedan acceder e interactuar con dispositivos de almacenamiento virtuales por medio de una interfaz iSCSI.

Brinda una funcionalidad comparable a la que tendrías utilizando un sistema SAN, con sus ventajas (alto desempeño para lectura/escritura) y limitaciones (un dispositivo solamente puede ser usado por una sola máquina virtual al mismo tiempo). Sin embargo, a diferencia de un SAN convencional, el API de Cinder te permite que de manera programática puedas asignar un dispositivo a una u otra máquina virtual conforme lo necesites.



5.1.1.4 Subsistema Swift

http://docs.openstack.org/developer/swift/

Es un almacén de objetos distribuido que viene a cumplir una función análoga al Simple Storage Service (S3) de Amazon Web Services. Dada su naturaleza, Swift es altamente escalable tanto en términos de tamaño (varios petabytes) como capacidad (billones de objetos) y cuenta con capacidad nativa de redundancia y tolerancia a fallos.

¿Qué es un objeto?

Esta noción es fundamental para entender este servicio puesto que la administración de objetos es su principal objetivo.

Un objeto es una entidad que contiene información, pero a diferencia de los archivos con los que usualmente trabajamos, los objetos no son organizados en una jerarquía.

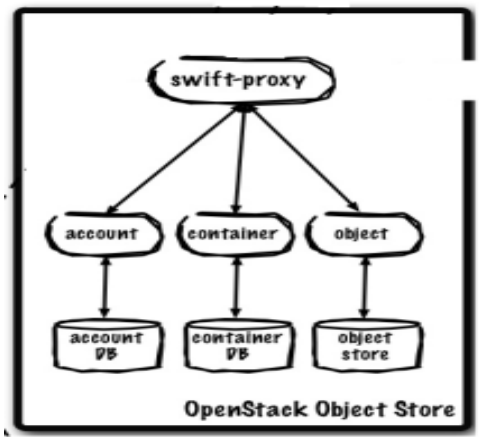
Cada objeto existe en el mismo nivel en un espacio de direcciones llamado *pool de almacenamiento*. Un objeto no puede ser almacenado dentro de otro objeto.

Tanto los archivos como los objetos tienen metadatos asociados a los datos que contienen, pero los objetos son caracterizados por tener metadatos *extendidos*. Cada objeto tiene asignado un identificador único que permite a un servidor o usuario final recuperarlo sin necesidad de conocer la ubicación física de la información. Esto es muy útil para automatizar y racionalizar almacenamiento de datos en entornos de *cloud computing*.

Usualmente se compara el almacenamiento de objetos con el estacionamiento en un *restaurant*. Cuando un cliente solicita este servicio, intercambia las llaves de su auto por un recibo. El cliente no conoce dónde será estacionado su auto o cuántas veces el encargado moverá su auto mientras el cena. En esta analogía, el identificador único de los objetos está representado por el recibo.

Volviendo a Swift, podemos mencionar

- El servidor proxy se encarga de aceptar los pedidos entrantes, como archivos para subir, modificaciones a los metadatos o creación de contenedores; también provee archivos y un listado de los contenedores.
- El servidor de cuentas maneja las cuentas asociadas con el servicio.
- El servidor de contenedores realiza un mapeo de los contenedores, carpetas, dentro del servicio.
- El servidor de objectos administra los objectos, archivos.
- También se corren servicios de replicación, para proveer consistencia y disponibilidad a través del cluster, auditoría y actualización.

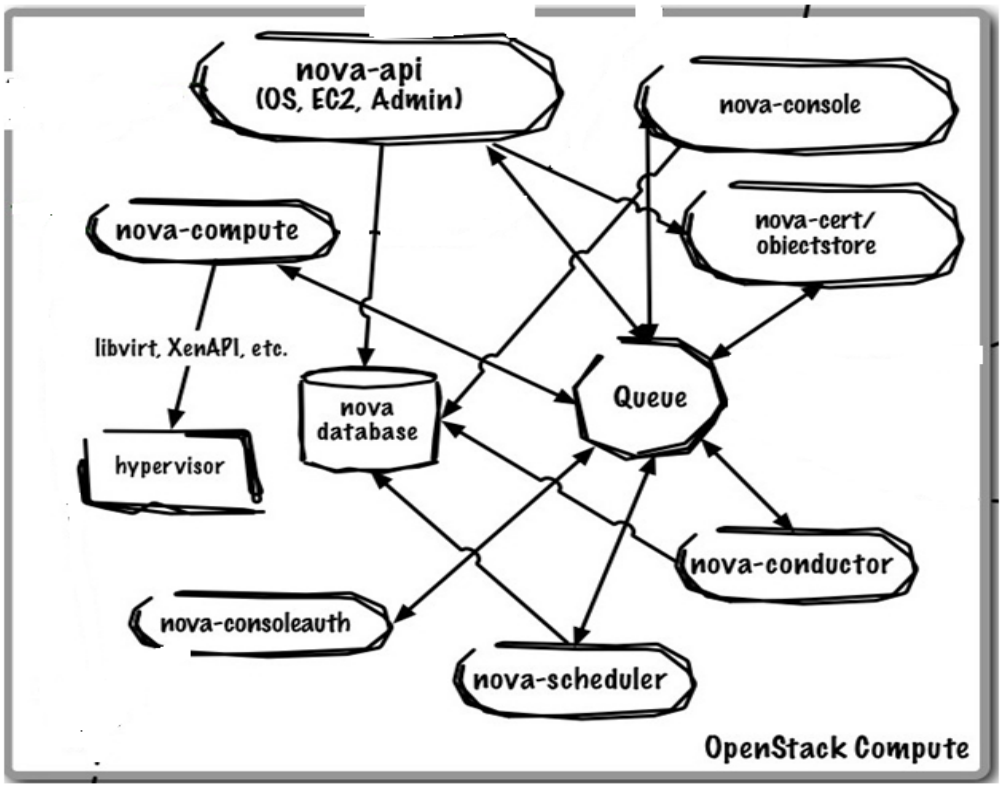


5.1.1.5 Subsistema Nova

http://docs.openstack.org/developer/nova/

Nova es el subsistema más complejo e importante de Openstack. Compuesto de muchos procesos, tal que:

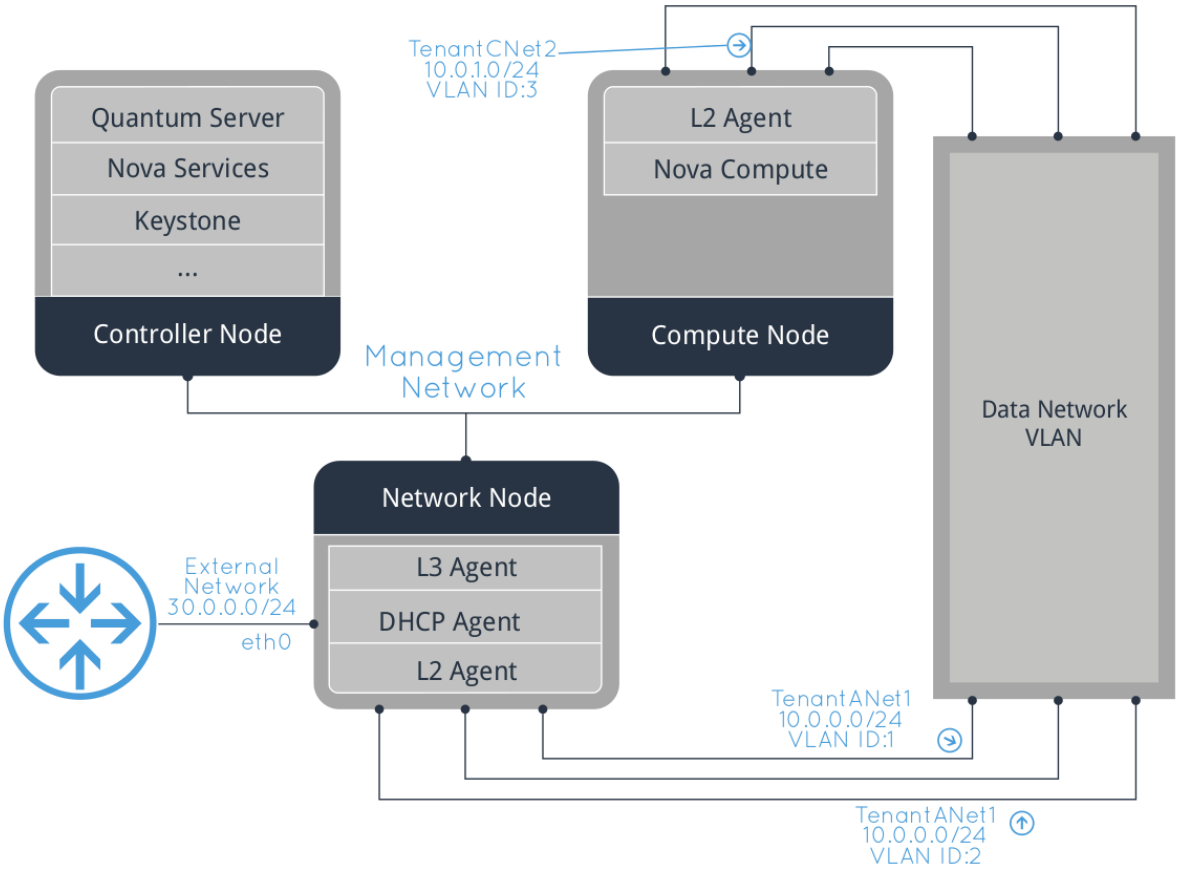
- El cliente nova-api acepta y responde a las llamadas del usuario final.
- La virtualización es administrada por nova-compute. Crea/finaliza las instancias de VMs a través de la API del hipervisor utilizado.
- El almacenamiento es controlado por nova-volume (ahora reemplazado por Cinder). Administra la creación, vinculación y desvinculación de volúmenes de almacenamiento persistente a instancias.
- Las redes son gestionadas por nova-network (ahora reemplazado por Neutron). Acepta tareas relacionadas a redes y las ejecuta.
- La planificación es realizada por nova-schedule. Toma los pedidos de VMs de la cola y determina dónde debería ejecutarse.
- La cola, por defecto RabbitMQ, es el nodo central para el pasaje de mensajes entre daemons.
- También dispone de una base de datos que almacena la mayoría de los datos de estado de compilación y ejecución.
- nova-consoleauth, nova-novncproxy, nova-console permiten a los usuarios acceder a las instancias virtuales a través de un *proxy*.
- Al crear una instancia deberán seleccionar entre las opciones de configuraciones de recursos virtuales disponibles, llamadas *flavors*. Luego, pueden agregarse recursos como volúmenes de almacenamiento persistente y direcciones IP públicas.



5.1.1.6 Subsistema Neutron (Quantum)

El subsistema de Red es muy importante ya que facilita la conectividad entre las instancias entre sí o con la Red Pública. Tipicamente el susbsistema facilita 3 opciones: Flat Networks (con ips fijas), Flat Network con DCHP, Vlan Network. Los procesos:

- quantum-server acepta pedidos de la API y los enruta hacia los plugins de Quantum que correspondan.
- Los plugins y agentes de Quantum son los encargados de realizar las tareas, como enchufar/desenchufar puertos, crear redes y subredes y direccionamiento de IPs.
- También dispone de una cola para enrutar información entre el quantum-server y los distintos agentes.
- Posee una base de datos para almacenar el estado de determinados plugins.



5.1.1.7 Subsistema Horizon

Horizon es una aplicación web modular desarrollada con el framework de Python Django, cuyo objetivo principal es proporcionar una interfaz a los servicios de OpenStack al administrador del cloud y a los usuarios.

Horizon no proporciona toda la funcionalidad que podemos conseguir a través del intérprete de comandos, pero lo "poco" que hace lo hace correctamente.

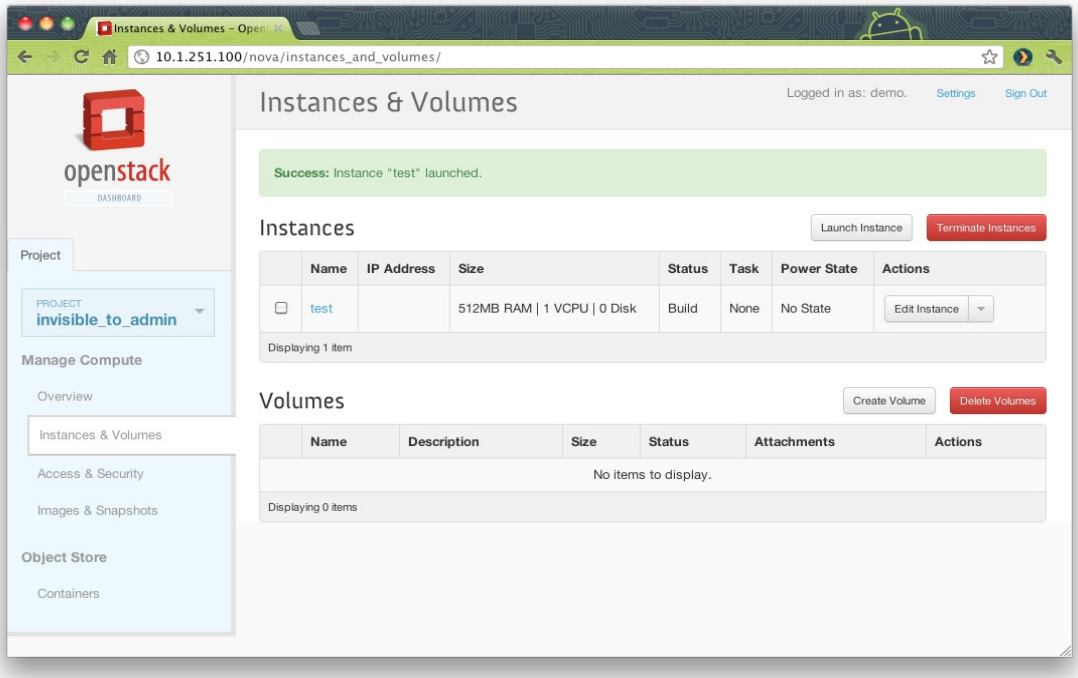
Como cualquier aplicación web, la arquitectura de Horizon es bastante simple:

- Horizon normalmente se despliega a través del módulo de Apache mod_wsgi, el cual implementa la interfaz WSGI que permite al servidor Apache ejecutar aplicaciones Python.

El código de Horizon está separado en dos módulos Python reutilizables, uno de ellos mantiene toda la lógica de la aplicación y es el encargado de interactuar con varias de las APIs de OpenStack, mientras que el otro es el encargado de la presentación, permitiendo fácilmente la adaptabilidad e integración con la apariencia del sitio web.

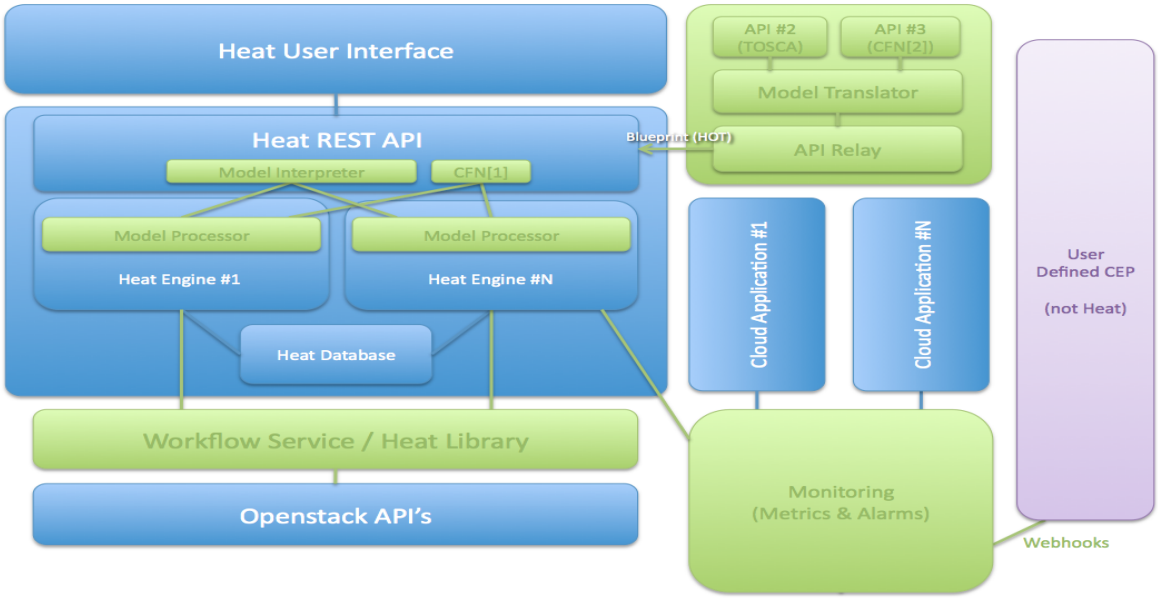
- Una base de datos. Horizon almacena muy pocos datos, ya que utiliza los datos del resto de servicios.

Desde el punto de vista de la red, este servicio debe ser accesible por los usuarios a través de la web (tráfico HTTP), de la misma forma que necesita poder acceder a las APIs públicas del resto de servicios. Si además se usa la funcionalidad de administración, necesita además conectividad a las APIs de administración de los endpoints (las cuales no son accesibles por los usuarios finales).



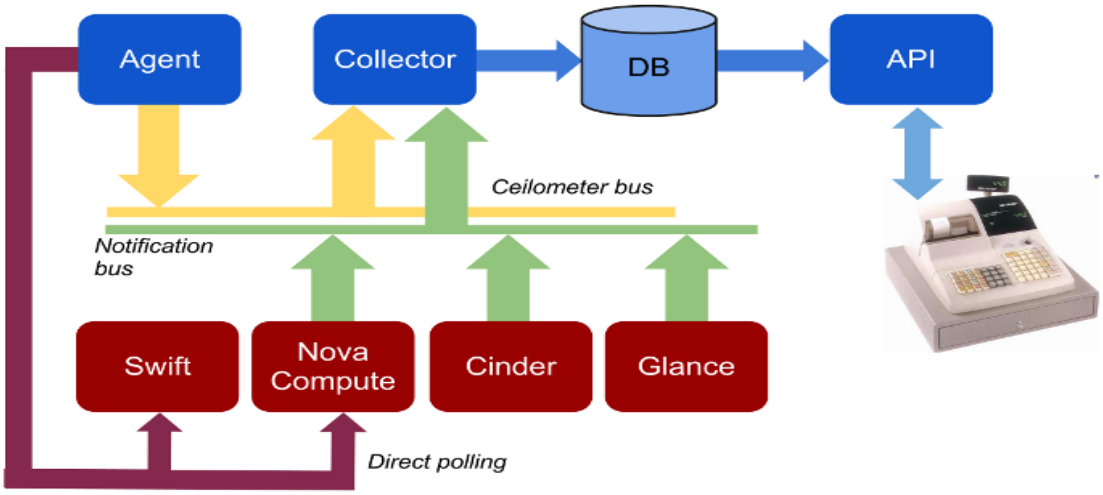
5.1.1.8 Subsistema Heat

Heat es responsable del nuevo sistema de Orquestación que facilita la creación de entornos Cloud compuestos de multiples instancias basados en plantillas. Es de aparición reciente en la última versión de Openstack Havana.



5.1.1.9 Subsistema Ceilometer

Openstack es una plataforma Opensource que crece continuamente, sin embargo un problema identificado desde el principio, era que no tenía mecanismos para obtener datos del uso de la plataforma de una forma centralizada. Tengamos presente que los servicios IaaS en muchos casos son servicios tipo Pago por Uso. Ceilometer viene a resolver esta problemática.

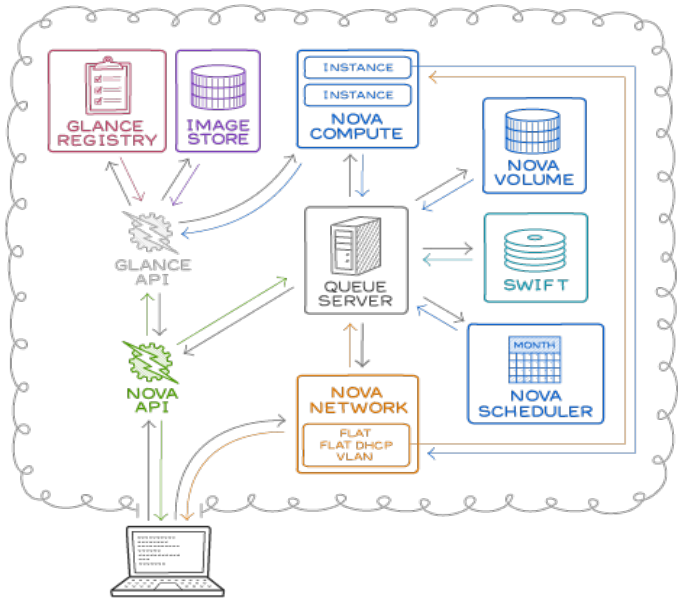


Por las especificas implicaciones cubriremos específicamente la parte de Computación, Gestión de Redes y la parte de Gestión de Almacenamiento.

5.1.2 Computación

5.1.2.1 Nova-compute

La arquitectura como hemos hablado es distribuida, y dentro de Nova se incluyen componentes centrales de la misma que permiten la conexión entre todos los componentes. Dentro de este apartado consideramos lo que corresponde a la computación (Nova-Compute). Ver imagen superior para ver su relación con el resto de los componentes de la arquitectura.



Openstack Compute – por Rackspace

- La virtualización es administrada por nova-compute. Crea/finaliza las instancias de VMs a través de la API del hipervisor utilizado.
- nova-consoleauth, nova-novncproxy, nova-console permiten a los usuarios acceder a las instancias virtuales a través de un proxy.
- Al crear una instancia deberán seleccionar entre las opciones de configuraciones de recursos virtuales disponibles, llamadas flavors. Luego, pueden agregarse recursos como volúmenes de almacenamiento persistente y direcciones IP públicas.

Por definición es uno de los elementos más suceptibles de escalar internamente. Atendiendo a que se requieran sucesivamente más servidores virtuales. En el proyecto que nos ocupa arrancamos con 2 servidores Nova-Compute.

5.1.2.2 Virtualización (Hipervisor KVM)

Openstack puede ser utilizado con hipervisores XEN y KVM. Se ha optado por KVM y por ello a continuación se hace una introducción sobre éste y el porqué de su elección. También mencionaremos por encima un componente crucial para la gestión de las instancias virtualizadas que es la librería libvirt.

- **KVM**

Para empezar KVM es un módulo más del kernel. Esto significa que no es un kernel "diferente" como ocurre con XEN. Al ser un modulo del kernel:

- Lo podemos cargar y descargar en tiempo de ejecución.
- Soporta los demás módulos
- Viene por defecto del Kernel (Xen no viene por defecto).

Por otra parte, debido a que es parte del kernel, "hereda" (o dicho de otra forma, utiliza) muchas de las ventajas implícitas del kernel como son:

- escalabilidad
- soporte para diferentes procesadores (x86_64,x86,Itanium)
- gestión de memoria
- NUMA

Frente a XEN, KVM puede compartir páginas de memoria, y de esta forma nos permite ahorrar memoria y por tanto, dinero.

Finalmente cada mencionar, que siendo parte del kernel, tiene menos líneas de código, ya que mucha funcionalidad la aprovecha del propio kernel. El desarrollo es más fácil y está centralizado.

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

- **Libvirt.**

La biblioteca libvirt es una APIde Linux sobre las capacidades de virtualización de Linux que soporta una variedad de hipervisores, incluyendo Xen y KVM, como así también QEMU y algunos productos de virtualización para otros sistemas operativos. Este software es ideal y crucial para la gestión de las máquinas virtuales y su administración remota a través de conexiones seguras con SSH. Además incluye una librería en C para el desarrollo de aplicaciones e incluye además interfaces para diferentes lenguajes como Python, Perl, Ruby, Java y PHP.

http://libvirt.org/

5.1.3 Aspectos relativos a la Gestión de Redes

Uno de los aspectos más importantes y más complejos a la hora de configurar una infraestructura de Cloud Computing es la red. Esto hace completamente indispensable conocer todas las opciones de configuración de red que OpenStack proporciona, para poder diseñar una infraestructura de comunicaciones acorde a nuestras necesidades.

OpenStack soporta 3 tipos de escenarios de networking:

- Flat Network Manager
- Flat DHCP Network Manager
- VLAN Network Manager

En principio, las tres clases de redes pueden coexistir en un único sistema cloud. Sin embargo, no cabe la posibilidad de elegir dentro de un proyecto entre los tres tipos de redes.

- *Flat Network Manager*

En modo Flat, el administrador de la red especifica una subred. Las direcciones IP para las máquinas virtuales se toman de esta subred y se inyectan en la imagen durante la creación de la VM. Cada instancia recibe una única dirección IP del pool de direcciones disponibles.

El administrador de la red debe configurar el bridge de Linux (normalmente denominando br100, aunque esto es configurable) tanto en el nodo que hospeda al gestor de la red, como en los nodos que ejecutan las instancias. Todas las instancias del sistema se unen al mismo bridge, configurado manualmente por el administrador de la red.

- *Flat DHCP Network Manager*

En el modo `FlatDHCP`, OpenStack inicia un servidor DHCP(a través del proceso dnsmasq) para la asignación de direcciones IP a las instancias desde la subred especificada, además de configurar el bridge de red.

De la misma forma que en el modo Flat, en el modo FlatDHCP todas las instancias se conectan a un único bridge en el nodo de computación, en este modo, además, se ejecuta un servidor DHCP encargado de configurar las instancias (dependiendo del modo single/multi-host) junto a cada uno de los procesos nova-network -neutron). En este modo, Compute hace un poco más de trabajo aparte de unir al bridge una interfaz de red (la indicada a través de `flat_interface`, `eth0` por defecto), configura y ejecuta un proceso dnsmasq, que actúa como servidor DHCP sobre el bridge configurado,

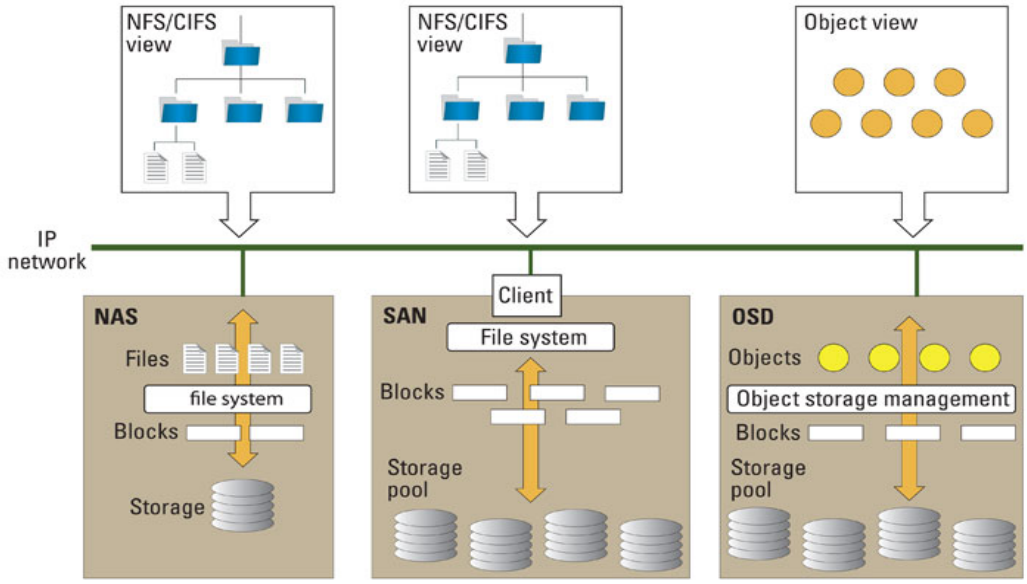
- *VLAN Network Manager*

El modo de red `VLAN` es el modo por defecto para una instalación de OpenStack. En este modo, Compute crea una VLAN y un bridge para cada proyecto. Para una instalación de múltiples máquinas, el modo de red VLAN necesita de un switch que soporte etiquetado VLAN (802.1q). Cada proyecto consigue un rango de direcciones privadas que son accesibles únicamente desde dentro de la VLAN. Para que un usuario pueda acceder a las instancias de su proyecto, se tiene que crear una instancia especial VPN (con nombre en clave cloudpipe). Compute genera un certificado y una clave para que el usuario pueda acceder a la VPN y la inicia de forma automática. Esto proporciona un segmento de red privado para las instancias de cada proyecto y permite que puedan ser accedidas desde Internet a través de una conexión VPN. En este modo, cada proyecto obtiene su propia VLAN, su propio bridge Linux y su subred.

Las subredes son especificadas por el administrador de la red, y se asignan de forma dinámica a un proyecto cuando se requiere. Dentro de cada una de las VLANs se inicia un servidor DHCP para configurar las direcciones IP a las máquinas virtuales desde la subred asignada al proyecto. Todas las instancias que pertenecen a un proyecto se unen a un mismo bridge dentro de la misma VLAN del proyecto. OpenStack Compute crea todos los bridges de Linux necesarios así como las VLANs cuando se requiere.

5.1.4 Aspectos relacionados al Almacenamiento

Uno de los aspectos que orginan más confusión es lo relativo a la diferenciación entre el almacenamiento de objetos y el almacenamiento por bloques como iSCSI o FibreChannel (SAN). Ambos son completamente diferentes y debe tenerse en cuenta. Conceptualmente se puede intuir en la siguiente imagen y se describirá a continuación.



Los dispositivos tipo SAN por ejemplo sólo exponen dispositivos de blqoues al sistema (como por ejemplo en Linux son los dispositivos /dev/sdb,etc...), mientras que al almacenamiento de objetos puede ser accedidos vía APIs y aplicaciones (como por ejemplo la aplicación del servicio box.com).

Es importante la comprensión de la siguiente tabla, particularmente por la relación entre el almacenamiento y las instancias (VM).

Almacenamiento Openstack

	Ephemeral storage	Block storage	Object storage
Used to...	Run operating system and scratch space	Add additional persistent storage to a virtual machine (VM)	Store data, including VM images
Accessed through...	A file system	A block device that can be partitioned, formatted and mounted (such as, /dev/vdc)	REST API
Accessible from...	Within a VM	Within a VM	Anywhere
Managed by...	OpenStack Compute (Nova)	OpenStack Block Storage (Cinder)	OpenStack Object Storage (Swift)
Persists until...	VM is terminated	Deleted by user	Deleted by user
Sizing determined by...	Administrator configures size settings, known asflavors	Specified by user in initial request	Amount of available physical storage
Example of typical usage...	10 GB first disk, 30GB second disk	1 TB disk	10s of TBs of dataset storage

El almacenamiento Efímero es el utilizado por los SO de las imágenes y su vida está completamente vinculado a la vida de las instancias.

El almacenamiento de Bloques es que correspondería a Volúmenes que puedes asignar a las instancias.

El almacenamiento de Objetos se gestiona mediante las aplicaciones.

5.1.4.1 Almacenamiento de Bloques

Si hablamos de Almacenamiento, un bloque es una secuencia de bytes o bits, con una longitud fija (block size). Por tanto los datos se dice que se estructuran en bloques. Este tipo de estructura de los datos facilita la escritura y lectura de los mismos, ya que ambas operaciones son siempre a nivel de bloques a la vez.

La mayoría de los sistemas de ficheros se basan en dispositivos que operan con niveles de abstración para almacenar o recuperar específicos bloques de datos. En los sistemas de ficheros, un bloque único puede contener sólo una parte de un fichero, y para ficheros de diferentes tamaño se originan ineficiencias de espacio debido a la que las longitudes de los ficheros no siempre puede ser múltiplos del tamaño del bloque, y por tanto siempre los últimos bloques se usan parcialmente.

El almacenamiento de bloques se hace transparente por los sistemas de ficheros o bases de datos frente a los usuarios y aplicaciones. Los volúmenes físicos o lógicos accedidos vía operaciones I/O pueden ser internos dentro del servidor, conectados directamente mediante SCSI o Fibre Channel, o pueden accederse vía SAN (Storage Area Networks) usando protocolos como por ejemplo iSCSI.

Cuando hablamos de servicios Cloud, el almacenamiento de Bloques es importante y se usa en la mayoría de los casos para almacenar imágenes de las Máquinas Virtuales o almacenar ficheros de los usuarios (backups de todo tipo, documentos,etc...)

5.1.4.2 Almacenamiento de Objetos

Es una arquitectura de almacenamiento que gestiona los datos mediante objetos, como enfoque diferentes a las arquitecturas tradicionales de Sistemas de Ficheros que gestiona los datos co una jerarquía de ficheros definida y los datos los ubica por bloques.

Cada objeto tipicamente incluye el dato en si mismo, una cantidad variable de "metadata", y un identificador global único.

El almacenamiento de objetos se puede implementar en múltiples niveles, incluso a nivel de dispositivos, de sistema o de interface. En cada caso, el almacenamiento de objetos busca habilitar funcionalidades no resueltas en otro tipo de arquitecturas de alamacenamiento, como que los interfaces no sean directamente programables por la aplicación, un espacio de nombres que puedan incluir múltiples dispositivos de hardware, funciones de replicación de datos, y una distribución de datos con granularidad a nivel de objetos.

La principal ventaja de este tipo de almacenamiento es su bajo coste de implementación comparado con el coste de los sistemas NAS,SAN,etc.. además de la seguridad (por la redundancia de datos) y la escalabilidad que ofrece.

A este tipo de Almacenamiento corresponden los servicios Amazon S3 y OpenStack swift (CloudFiles RackSpace y SwiftStack).

Nota: El archipopular Dropbox utiliza S3 para almacenar los archivos de millones de usuarios.
https://www.dropbox.com/help/7/es_ES

5.1.5 Interfaces API

- Openstack es gestionable vía:
  - *Interface Web (Horizon)*
  - *Interface Línea de comandos.*
    - Por ejemplo: Para obtener los proyectos configurados en el servicio.

```
root@controller:~# keystone tenant-list
+----------------------------------+---------+---------+
|                id                |   name  | enabled |
+----------------------------------+---------+---------+
| 42901f7c9572470c90841fd7021f51d4 |  admin  |   True  |
| 1caa6e6efb374c6c85390eea82ac12d6 | service |   True  |
+----------------------------------+---------+---------+
root@controller:~#
```

Para obtener las credenciales ec2 que usaremos por el api Amazon con Euca2ools.

```
root@controller:~# keystone ec2-credentials-list
+--------+----------------------------------+----------------------------------+
| tenant |              access              |              secret              |
+--------+----------------------------------+----------------------------------+
| admin  | xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx | xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx |
+--------+----------------------------------+----------------------------------+
root@controller:~#
```

  - *Api especifico de Openstack*

```
root@controller:~# curl -i -X GET http://controller:35357/v2.0/tenants -H "User-Agent: python-keystoneclient" -H "X-Auth-Token: token"

HTTP/1.1 200 OK

Vary: X-Auth-Token

Content-Type: application/json

Content-Length: 232

Date: Sun, 29 Dec 2013 12:18:29 GMT

{"tenants_links": [], "tenants": [{"description": null, "enabled": true, "id": "1caa6e6efb374c6c85390eea82ac12d6", "name": "service"}, {"description": "", "enabled": true, "id": "42901f7c9572470c90841fd7021f51d4", "name": "admin"}]}

root@controller:~#
```

  - *Api Amazon AWS*
    - Por ejemplo listar las imagenes disponibles

```
root@controller:~# euca-describe-images
IMAGE   ami-00000003    None (CirrOS 0.3.1)     1caa6e6efb374c6c85390eea82ac12d6        available       public                  machine instance-store
IMAGE   ami-00000004    None (Ubuntu-12.04-server-cloudimg-amd64)       42901f7c9572470c90841fd7021f51d4        available public                machine         instance-store
IMAGE   ami-00000001    None (Ubuntu-12.04-server-cloudimg-i386)        42901f7c9572470c90841fd7021f51d4        available public                machine         instance-store
IMAGE   ami-00000006    None (Fedora_Core_19-i386)      42901f7c9572470c90841fd7021f51d4        available       public  machine                 instance-store
IMAGE   ami-00000005    None (Fedora_Core_19-x86_64)    42901f7c9572470c90841fd7021f51d4        available       public  machine                 instance-store
IMAGE   ami-00000007    None (Windows_server_2012_r2_standar_eval)      42901f7c9572470c90841fd7021f51d4        available public                machine         instance-store
IMAGE   ami-00000008    None (F17-x86_64-cfntools)      42901f7c9572470c90841fd7021f51d4        available       private  machine                 instance-store
IMAGE   ami-00000009    snapshot        42901f7c9572470c90841fd7021f51d4        available       private                 machine instance-store
root@controller:~#
```

Openstack dispone de diferentes entornos de desarrollo o SDK, que son una parte crucial para escribir aplicaciones de todo tipo relacionadas con la Operativa de Openstak.

https://wiki.openstack.org/wiki/SDKs

Algunos aspectos importantes y que deben considerarse:

- Resulta cómodo y necesario cuando se use el interface de comandos e invocaciones en curl exportar las variables de entorno que se utilizan:
  - En el caso del interface de comandos o invocaciones con Curl http.

    root@controller:~# export OS_TENANT_NAME=admin
    root@controller:~# export OS_USERNAME=username
    root@controller:~# export OS_PASSWORD=password
    root@controller:~# export OS_AUTH_URL="controller:5000/v2.0/"
  - En el caso de Euca2ools API amazon

    root@controller:~# export EC2_ACCESS_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
    root@controller:~# export EC2_SECRET_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
    root@controller:~# export EC2_URL='http://controller:8773/services/Cloud'

Nota: las mismas variables deberán utilizarse con Amazon, con los valores obtenidos al dar de alta el servicio.

- Tokens

  Aunque pueden usarse otros escenarios de autenticación para un entorno Openstack, tipicamente se hace vía Keystone y a través de Tokens.

  Tipicamente la operativa consiste en obtener un token a través del usuario y password determinado de la cuenta y del proyecto (tenant). Via API sería, algo así cómo:

root@controller:~# curl -i 'http://controller:5000/v2.0/tokens' -X POST -H "Content-Type: application/json" -H "Accept: application/json" -d '{"auth": {"tenantName": "admin", "passwordCredentials": {"username": "username", "password": "password"}}}'

HTTP/1.1 200 OK

Vary: X-Auth-Token

Content-Type: application/json

Content-Length: 8380

Date: Sun, 29 Dec 2013 11:40:51 GMT

{"access": {"token": {"issued_at": "2013-12-29T11:40:51.817114", "expires": "2013-12-30T11:40:51Z", "id":.....RESTO INFORMACION TOKEN.....}

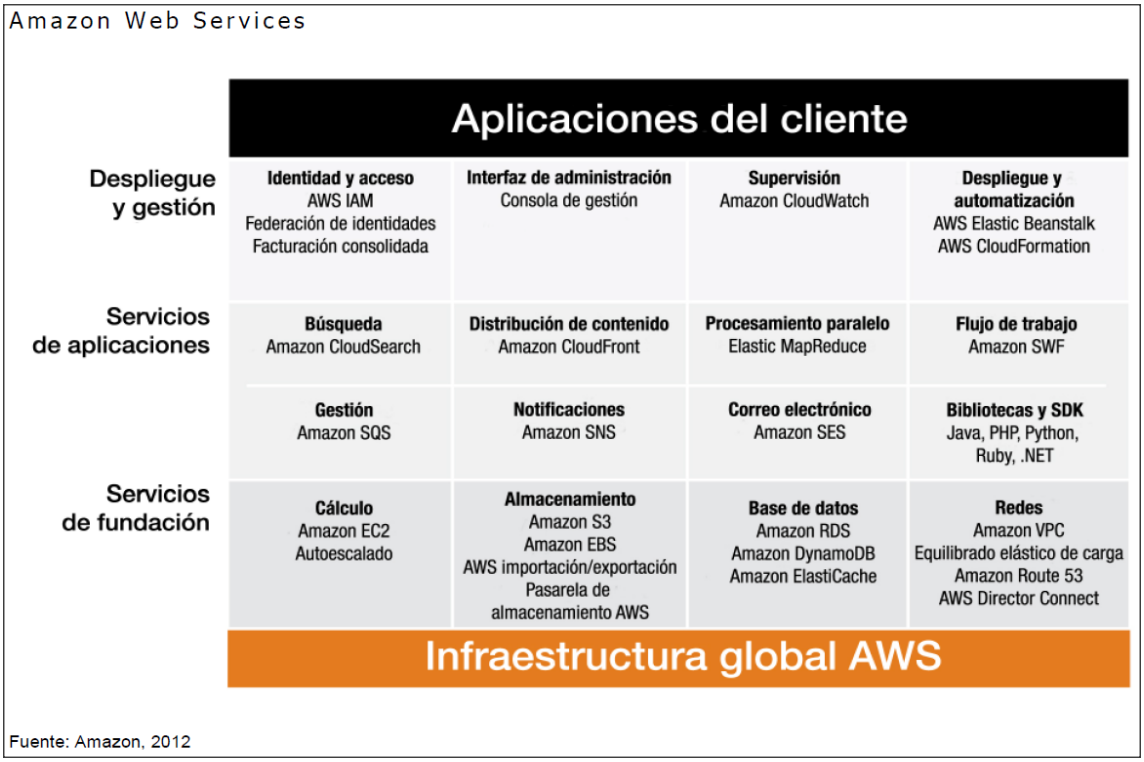
Es relevante además de conocer el valor del Token, saber que típicamente tiene unos plazos de expiración (ver mensaje anterior) de 24 horas, tras los cuales, se rechazaría cualquier petición vía API con el Token expirado, y debería volver a solicitarse otro.

Nota: En el fichero keystone.conf se puede definir un Token de Administrador para la configuración inicial, que luego deber borrarse, aunque si no se borra es más cómodo para realizar pruebas,etc..

5.2 Amazon AWS

Sin lugar a dudas Amazon es el Líder en todo tipo de Servicios Cloud (principalmente IaaS) y es un referente para todo el Mercado que todos desean copiar. Por todo ello merece la pena al menos enumerar, como haremos a continuación, todos los servicios que ofrecen, para entender la innovación que ofrece continuamente al mercado, y por donde presumiblemente orientarán sus esfuerzos el resto de alternativas (ya sean proveedores o soluciones).

Artic S.L. Aun cuando optará por montar su propia infraestructura tendrá en consideración, igualmente, los siguientes servicios para su propio uso o para sus clientes.



Fuente: Amazon, 2012

5.2.1 Amazon AWS: Informática y redes

- **Amazon EC2** Servidores virtuales en la nube
  - **Auto Scaling**
  - **Elastic Load Balancing**
- **Amazon WorkSpaces** Escritorios virtuales en la nube
- **Amazon VPC** Recursos de nube aislados
- **Amazon Route 53** Sistemas de nombres de dominio (DNS) escalables
- **AWS Direct Connect** Conexión de red dedicada a AWS

5.2.2 Amazon AWS: Almacenamiento y CDN

- **Amazon S3** Almacenamiento escalable en la nube
- **Amazon Glacier** Almacenamiento de archivos en la nube a bajo coste
- **Amazon EBS** Volúmenes de almacenamiento de bloque EC2
- **AWS Import/Export** Transferencia de datos de gran volumen
- **AWS Storage Gateway** Integra entornos de TI en las instalaciones con el almacenamiento en la nube
- **Amazon CloudFront** Red de entrega de contenido global (CDN)

5.2.3 Amazon AWS: Base de datos

- **Amazon RDS** Servicio de base de datos relacional gestionada para MySQL, Oracle y SQL Server
- **Amazon DynamoDB** Almacenamiento de datos NoSQL, rápido, predecible y de alta escalabilidad
- **Amazon ElastiCache** Servicio de almacenamiento en memoria caché
- **Amazon Redshift** Servicio de almacén de datos rápido, potente, totalmente gestionado y con un escalado de petabytes

5.2.4 Amazon AWS: Análisis

- **Amazon EMR** Infraestructura Hadoop alojada
- **Amazon Kinesis** Procesamiento de transmisión de datos en tiempo real
- **AWS Data Pipeline** Servicio de orquestación para flujos de trabajo periódicos controlados por datos
- **Amazon Redshift** Servicio de almacén de datos rápido, potente, totalmente gestionado y con un escalado de petabytes

5.2.5 Amazon AWS: Servicios de aplicaciones

- **Amazon AppStream** Transmisión de aplicaciones de latencia baja
- **Amazon CloudSearch** Servicio de búsqueda gestionada
- **Amazon SWF** Servicio de flujo de trabajo para coordinar componentes de aplicaciones
- **Amazon SQS** Servicio de cola de mensajes
- **Amazon SES** Servicio de envío de correos electrónicos
- **Amazon SNS** Servicio de notificaciones automatizado
- **Amazon FPS** Servicio de pago basado en API
- **Amazon Elastic Transcoder** Transcodificación multimedia escalable y fácil de usar

5.2.6 Amazon AWS: Implementación y gestión

- **AWS Management Console** Interfaz de usuario basada en web
- **AWS Identity and Access Management (IAM)** Controles de acceso configurables de AWS
- **AWS CloudTrail** Registro de actividad del usuario
- **Amazon CloudWatch** Supervisión de aplicaciones y recursos
- **AWS Elastic Beanstalk** Contenedor de aplicaciones de AWS
- **AWS CloudFormation** Plantillas para crear recursos en AWS
- **AWS OpsWorks** Servicios de gestión de aplicaciones de DevOps
- **AWS CloudHSM** Almacenamiento de claves en hardware para el cumplimiento de las regulaciones

5.2.7 Amazon AWS: AwsMarketplace

awsmarketplace

Software de socios preconfigurado para ejecutarse en AWS. Listo para instanciarse en pocos minutos.

- Software de infraestructura (403)
  - **Pilas de aplicaciones**
  - **Sistemas operativos**
  - **Bases de datos y caché**

- **Infraestructura de red**

- Software para empresas (399)
  - **Inteligencia empresarial**
  - **Colaboración**
  - **Gestión de contenido**
  - **CRM**

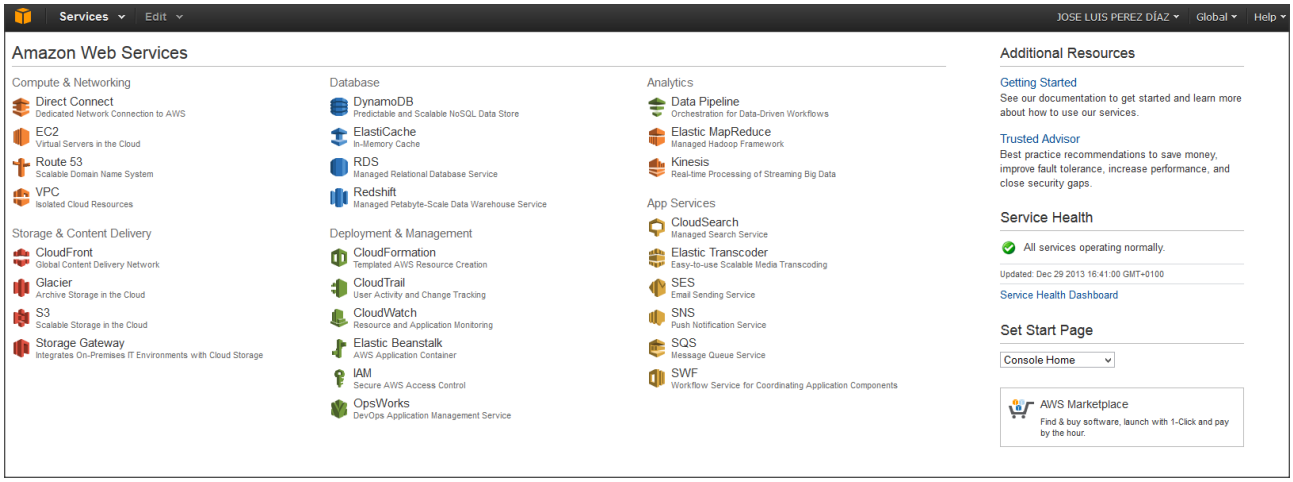
- Herramientas para desarrolladores (117)
  - **Registro de problemas y errores**
  - **Supervisión**
  - **Control de fuentes**
  - **Pruebas**

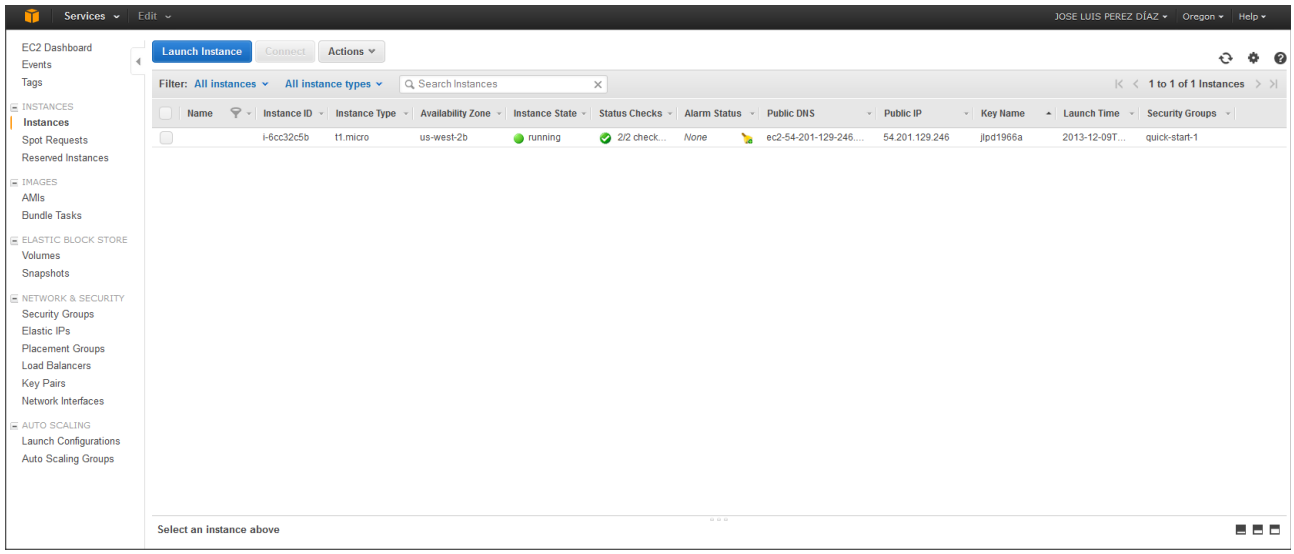
5.2.8 Interfaces Gestión Amazon AWS

El acceso se puede obtener via los siguientes interfaces:

- *Consola de Admnistración Amazon AWS.*

Una vez creada la cuenta, se dispone de un acceso https seguro con login y password. A continuación la consola que he creado desde la que se tiene acceso completo a todos los servicios.





- *Via API, por ejemplo utilizando Euca2ools* https://wiki.debian.org/euca2ools

  Se podrán comprobar el gran "parecido" a su uso con Openstack (ya que el API es el mismo).

  - Primero definiríamos las variables de entorno

```
[root@fc18-atica ~]# export EC2_ACCESS_KEY=Access Key obtenido de amazon
[root@fc18-atica ~]# export EC2_SECRET_KEY="Ec2 Secret key obtenido de Amazon"
[root@fc18-atica ~]# export EC2_URL=http://ec2.amazonaws.com

[root@fc18-atica ~]# euca-describe-regions
REGION  eu-west-1       ec2.eu-west-1.amazonaws.com
REGION  sa-east-1       ec2.sa-east-1.amazonaws.com
REGION  us-east-1       ec2.us-east-1.amazonaws.com
REGION  ap-northeast-1  ec2.ap-northeast-1.amazonaws.com
REGION  us-west-2       ec2.us-west-2.amazonaws.com
REGION  us-west-1       ec2.us-west-1.amazonaws.com
REGION  ap-southeast-1  ec2.ap-southeast-1.amazonaws.com
REGION  ap-southeast-2  ec2.ap-southeast-2.amazonaws.com
[root@fc18-atica ~]#

[root@fc18-atica ~]# euca-describe-availability-zones
AVAILABILITYZONE        us-east-1a      available
AVAILABILITYZONE        us-east-1b      available
AVAILABILITYZONE        us-east-1c      available
[root@fc18-atica ~]#
```

  Debemos obtener vía euca_describe_regions la disponibilidad del interface para configurar la url especifica que se haya de usar. Pero se pueden elegir otras zonas diferentes de las que son por defecto vía explicitando una nueva URL de EC2 como se indica a continuación:

```
[root@fc18-atica ~]# EC2_URL=https://eu-west-1.ec2.amazonaws.com euca-describe-availability-zones
AVAILABILITYZONE        eu-west-1a      available
AVAILABILITYZONE        eu-west-1b      available
AVAILABILITYZONE        eu-west-1c      available
[root@fc18-atica ~]#
```

  A partir de este momento se puede hacer uso de todas los métodos del API para gestionar absolutamente todos los servicios disponibles. Por ejemplo: Podríamos consultar las imagenes debian squeeze:

```
[root@fc18-atica ~]# euca-describe-images -a | egrep 'available.+public.+i386' | awk '{print $2,$3}' | grep -i debian | sort -k
2 | grep squeeze
ami-9b76bff2 2ndquadrant-ami-us-east-1/debian-squeeze-i386-beta2.img.manifest.xml
ami-718b4318 324077455750/debian-squeeze-i386-ebs-beta2.img
ami-2ae17343 379101102735/debian-squeeze-i386-20130224
ami-0ce41865 624081292306/debian-6.0-squeeze-base-i386-20110417
ami-ce3dc6a7 762644096235/marcfiasse.com/debian_base_i386_squeeze_602_10Go_US_frBE_30062011
ami-386ed551 764905731938/debian-6.06-squeeze-base-intacto
ami-64fe190d alestic/debian-6.0-squeeze-base-20090215.manifest.xml
ami-e048af89 alestic/debian-6.0-squeeze-base-20090418.manifest.xml
ami-fb46a792 alestic/debian-6.0-squeeze-base-20090804.manifest.xml
ami-daf615b3 alestic/debian-6.0-squeeze-base-20091011.manifest.xml
ami-adfe19c4 alestic/debian-6.0-squeeze-desktop-20090215.manifest.xml
ami-0256b16b alestic/debian-6.0-squeeze-desktop-20090419.manifest.xml
ami-f946a790 alestic/debian-6.0-squeeze-desktop-20090804.manifest.xml
ami-d8f615b1 alestic/debian-6.0-squeeze-desktop-20091011.manifest.xml
ami-2c28ba45 aws-marketplace/debian-squeeze-i386-20130224-e4554303-3a9d-412e-9604-eae67dde7b76-ami-2ae17343.1
ami-aa46b4c3 camptocamp-us-east-1/debian-6.0-squeeze-i386-server-20110323.manifest.xml
ami-e2ed108b fewbytes-images/debian-squeeze-i386-1.manifest.xml
ami-269f614f fewbytes-images/debian-squeeze-i386-2.manifest.xml
ami-463dc62f marcfiasse.com/debian_base_i386_squeeze_602_10Go_US_frBE_30062011.img.manifest.xml
ami-c041bda9 punch-ami-us-east-1/debian-6.0-squeeze-base-i386-20110426.manifest.xml
[root@fc18-atica ~]#
```

  Como ya se ha dicho las opciones son muy similares a las disponibles con OpenStack.

- *Via SDK.*

Existen multitud de bibliotecas, recursos y recursos disponibles en Java, PHP, Ruby, Python, etc...

Sin lugar a dudas Amazon AWS es una opción que forzosamente se ha de tener encuenta, al menos para complementar en caso de necesidad la infraestructura propia existente.

A continuación introduciremos más en detalle por si importancia los 2 servicios más importantes de Amazón EC2 (computación) y S3 (Almacenamiento).

5.2.9 EC2 Computación Elástica (EBS Almacenamiento Bloques)

EC2 es el bloque fundamental de servicio alrededor del que tenemos todos los demás. Es un servicio que proporciona operaciones remotas de máquinas virtuales (VM) sobre la infraestrucura de Amazon. A continuación, apuntamos los conceptos fundamentales del servicio.

- *Máquina virtual o instancia.*

Una máquina virtual se conoce como "instancia". Amazon define diferentes tipos de instancias que difieren entre sí en función de los resursos de CPUs,memoria, almacenamiento,etc.. que contienen.

Se podria decir que alquilar una instancia EC2 proporciona la misma funcionalidad que tendríamos si alquilásemos un servidor "hosted" gestionado por un tercero.

Alrededor de EC2 hay una serie de servicios (ELB, EBS, S3, Ips elásticas, auto-scaling,etc..) que configuran los bloques de configuración de la solución AWS. Pensemos que una instancia EC2 no se autoescala por si misma, sino que lo hace a través de otros recursos contratables de Amazon.

- *Conectividad/Transferencia*

Cuanto hablamos de servicios de Amazon, es fundamental considerar todo lo relativo a la conectividad a Internet y el ancho de banda/datos transferidos, e igualmente el origen y el destino de los datos. Ya ha mencionado previamente, y es particularmente importante para instancias con elevado tráfico. Típicamente AWS factura separadamente el tráfico recibido del enviado y tienen diferentes reglas de facturación que deben de considerarse.

- *Regiones o Zonas*

Otro factor importante es donde se desean contratar los recursos, también conocido como regiones o zonas (que son función de donde se ubican los recursos hosteados).

- *Volumenes / Servicio EBS*

Finalmente hablaremos sobre la opción de crear volumenes o Discos duros para su conexión a instancias previamente creadas. Es el servicio complementario EBS, en el que los volumenes se conectan y son persistentes, e independientemente del ciclo de vida de una instancia). Una vez que se montan, la instancia puede interactuar con el volumen como si fuera un disco local, formatearlo como su file system o instalar aplicaciones directamente.

El almacenamiento EBS permite tamaños de volumenes desde 1 GB a 1 TB. Por otra parte es importante considerar que un volumen puede asignarse a una única instancia cada vez, pero muchos volúmenes pueden asignarse a una única instancia.

- *Backups (sobre S3) y Snapshots*

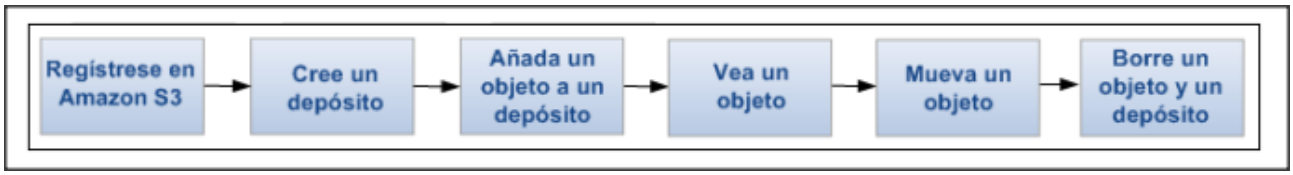
En general sueles ser útil hacer backup de los sanpshots (imagen de los datos del sistema) sobre servicios S3. Incluso se pueden configuran snapshots para backups incrementales: sólo los bloques del dispósitivo que hayan cambiado desde el útlimo snapshot se salvan.

Nota: Salvando alguna especificidad, la operativa de Openstack con instancias y volumenes es muy similar.

5.2.10 S3 Almacenamiento de Objetos

Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento de Objetos para Internet. Se puede usar S3 para almacenar y recuperar cuaquier cantidad de datos en cualquier momento desde Internet.

Como el resto de los servicios puede funcionar vía API o vía la consola de Administrador de Amazon AWS. El siguiente dibujo muestra los pasos requeridos para almacenar ficheros. Es relevante la fase de crear el depósito (Bucket) para posteriormente mover o recuperar los objetos de éste.



S3 sólo almacena objetos, no los cambia en ningún caso y eso lo hace especialmente apropiado por ejemplo y entre otros usos para: páginas web estáticas (sin procesamiento lado servidor), y por almacenar ficheros grandes (por ejemplo imagenes AMIs Amazon o imagenes qcow OpenStack).

Amazon AWS factura por un lado el tamaño almacenado y por otro lado los Datos transferidos.

Hay una considerable variedad de herramientas ya construidas (comerciales y opensource) orientadas por ejemplo a gestionar backups. Me permito subrayar el proyecto Duplicity (opensource), que permite facilmente como veremos en un anexo posteriormente realizar backups sobre S3, fácil, barato y encriptando la información.

Finalmente Amazon dispone de un servicio de Backup (específico) muy barato y seguro, pensado para tener un coste muy bajo siempre que la información no se recuperé con frecuencia. El coste es sensiblemente inferior al de S3, pero como ya he dicho su utilizad es muy especifica y debe evitar el acceso a la información con frecuencia.

Nota: Salvando alguna especificidad, la operativa de S3 es muy similar a la de OpenStack Swift

5.3 Arquitectura del Sistema

5.3.1 Introducción a la Arquitectura

Los distintos componentes software de la arquitectura (explicados previamente) se distribuirán entre servidores de la forma que garanticen la mayor escalabilidad y tolerancia a fallos. Como se ha indicado previamente la arquitectura consta de 4 tipos de nodos que se concretan inicialmente en 6 servidores.

A continuación haremos un desarrollo evolutivo de los principios que darán lugar a la arquitectura final. Es a efectos didácticos. Se explican dos escenarios de diseño evolutivos, tal que el escenario final resultante de aplicación en el proyecto será el Escenario número 2 descrito a continuación.

Nota: La arquitectura resultante será la consecuencia de la distribución de los servicios de OpenStack. Tipicamente se escala en Computación y almacenamiento (y en menor medida en Red-Networking). El resto de los equipos son de gestión del Cloud Openstack.

- **ESCENARIO 1 - SIMPLE**

Si se hubiera de aplicar un diseño simple con dos servidores, desde una perspectiva lógica lo que tendría sentido sería la siguiente arquitectura(aun cuando cabrían otros escenarios):

- Consta de dos servidores (visión simplificada)
  - Uno de **Computación**.
  - Otro que consta de **TODOS** los elementos restantes de la arquitectura:
    - Consola de gestión.
    - Interfaces (APIs) de gestión de openstack.
    - Gestores de colas
    - Almacenamiento,etc...
- Constaría de 2 redes:
  - Una interna para comunicación entre las dos máquinas.
  - Otra externa, para conexión con el mundo exterior.

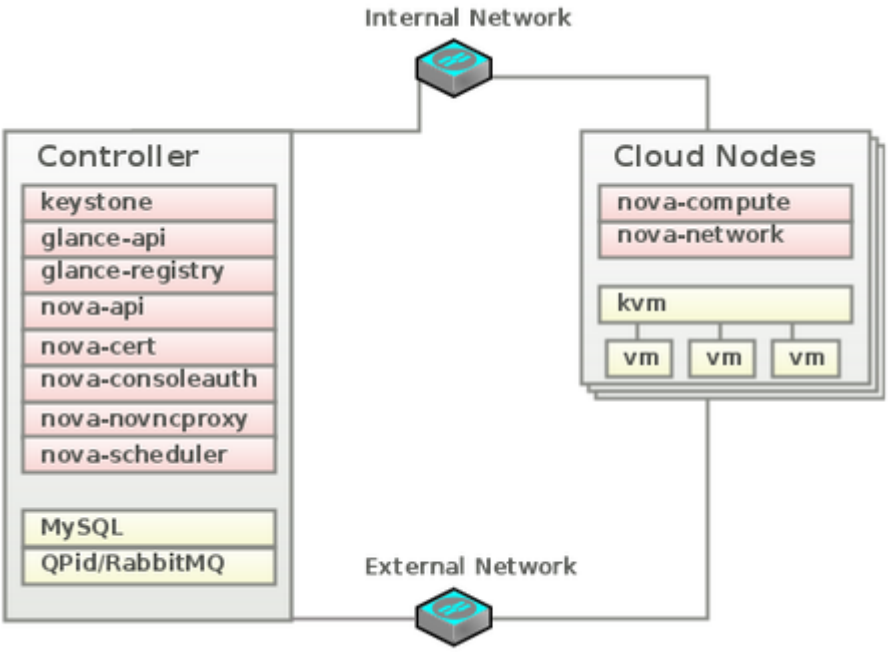
A partir de este escenario cabría aplicar diferentes criterios para escalar, a nivel interno, en caso de necesidad por aumento de demanda:

- Para aumento de computación → Instalar nuevos servidores (Cloud Nodes) de computación. El controlador (Controller) automáticamente balancearía sobre sucesivos servidores las instancias que se fueran creando.
- Para aumento de almacenamiento → Instalar nuevos servidores con conexión mayormente iscsi sobre los que se crearían grupos de volúmenes que estarían a disposición de la arquitectura tal que dinámicamente crearían volúmenes lógicos.

A partir de ahí cabría aplicar escalado externo teniendo en cuenta la compatibilidad que pudiera haber con los proveedores. Entre ellos y para el caso del almacenamiento:

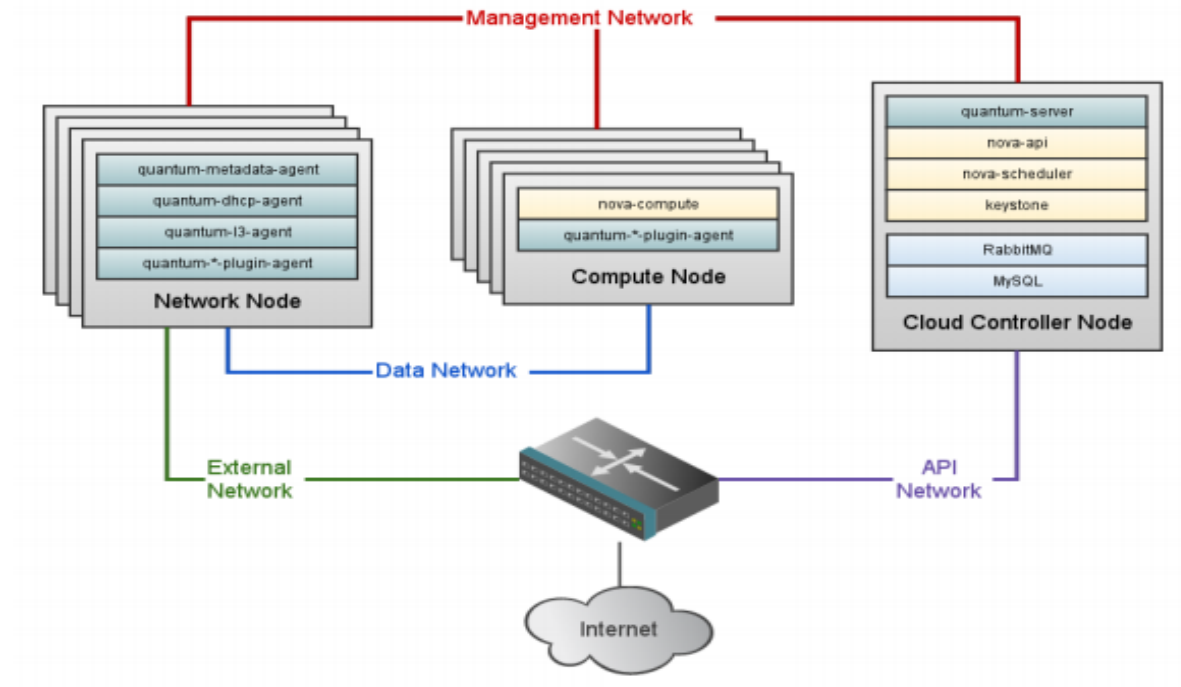
- Amazon S3 → http://aws.amazon.com/es/s3/
- Swifstack → http://swiftstack.com/product/ (como ejemplo de servicio completamente compatible con Openstack Swift).

El esquema que aplicaría a este escenario es el que sigue a continuación:



- **ESCENARIO 2 – Con mayor escalabilidad – Escenario similar al de este proyecto**

En este escenario creamos una arquitectura con 3 tipos de servidores (es la evolución lógica del escenario indicado anteriormente):



- Un Nodo Controller (consola de gestión, interface APIs, gestor de colas, almacenamiento,etc..)
- N nodos computación.
- M nodos de Red.

Nota importante: Cabría decirse que debería poderse desdoblar el almacenamiento en un nuevo grupo de Nodos independientes del Nodo Controller (como se indica previamente).

Se observa la introducción de nuevas Subredes (que describiremos posteriormente) para segmentar el tráfico de red y aumentar el rendimiento.

Para escalar internamente tendríamos los mismos condicionantes anteriormente vistos, si bien se observa que el aspecto que deberíamos añadir y controlar es el relativo al almacenamiento. Se visualiza la necesidad de crear un nuevo tipo de servidor o dispositivo que permita simplemente añadiendo dispositivos escalar la solución desde el plan del almacenamiento.

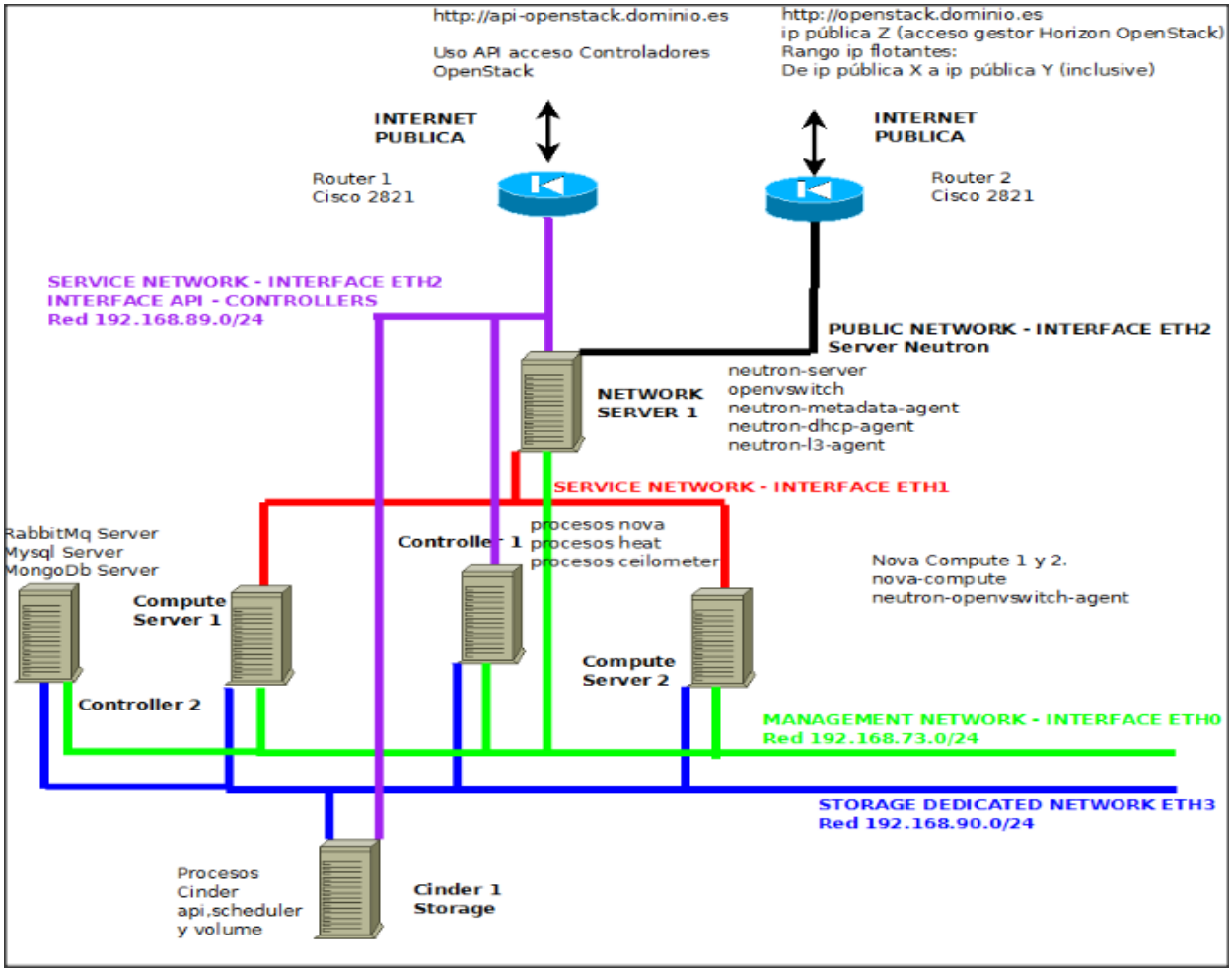
Para escalar externamente la situación es exactamente la misma a la descrita en el punto anterior.

En base a los dos escenarios indicaros previamente llegaríamos a la definición final que sería la arquitectura elegida en el proyecto.

5.3.2 Arquitectura Final del Proyecto

El siguiente diagrama refleja sinteticamente la arquitectura del sistema. Se hace notar que:
a) Hay un interface de acceso a los Apis
b) Hay un interface de acceso a la web de administración.
c) Hay un rango de ips flotantes publicas para los servidores virtuales.

Todos los nodos estarán equipados con 4 interfaces independientes de red gigaethernet.



Los tipos de Nodos/servidores son 4 (se pueden observar en el esquema previo). Se escalaría añadiendo el mismo tipo de Nodo a la estructura para aumentar computación, almacenamiento, y red.

- *Nodo Controller* 1--> Controlador Cloud.
  Proporcionará toda la funcionalidad del Sistema excepto los servicios de Red, computación y almacenamiento.
  - Web gestión dashboard (HORIZON)
  - APIs (NOVA-API)
  - Seguridad (KEYSTONE)
  - Servicio Imagener GLANCE
  - Planificador (NOVA-SCHEDULER)
  - Servicio vnc (NOVA-CONSOLE)

- *Nodo Controller* 2--> Controlador Cloud.
  - Gestión Colas y Base de datos (RABBITMQ-SERVER y MYSQL).
  - Gestion MONGODB (estadísticas Cloud Ceilometer).

Nota: Inicialmente se consideró integrar ambos servidores en uno sólo. La experiencia ha demostrado que el rendimiento aumenta drásticamente al "desdoblar ambos servidores" para reducir la carga.

- *Nodo Network* --> Controlador de Redes. Asigna y gestiona direccionamiento IP.
  - Servicio DHCP
  - Layer 2 switching
  - Layer 3 routing
  - Floating Ips
  - Metadata Connectivity
- *Nodo Compute* --> Controlador Computación. Las instancias de virtualización con KVM se ubicarían en este componente. Este host almacenará las instancias virtuales.

- *Nodo Storage*--> Controlador almacenamiento. Este servidor almacenará todas las imagenes y volúmenes del sistemas. Existen diversas opciones (Cinder, Swift,etc...) que se analizarán posteriormente con más detalle.

*** Redes requeridas para el sistema.**

Openstack típicamente puede tener hasta 4 distintas redes. Nota: Las redes se pueden combinar y reutilizar. En algunos escenarios sencillos la Red de Gestión, de datos y de API son comúnmente la misma red.

Así pues, las redes son:

- Red de Gestión (Management Network): Utilizada para la comunicación interna entre los componentes de Openstack. Las direcciones ips sólo deben ser accesible dentro del Data Center.
- Red de Datos (Data Network): Utilizada por las máquinas virtuales dentro de cada Cloud provisionado. Los requerimientos de direccionamiento de la red dependen del "plugin" de Networking utilizado.
- Red Externa (External Network): Proporciona a las máquinas Virtuales acceso a Internet en algunos despliegues de redes. La/s direcciones Ips en esta network deben ser accesibles desde cualquiera en internet.
- Red Conexión API (API Network). Da acceso al API de Openstack de los distintos componentes de la arquitectura, y debe ser accesible para cualquiera desde internet. Esta red podría ser la misma red que la externa.

Finalmente mencionar que por temas de almacenamiento rápido iSCSI también es habitual la existencia de una red adicional de conexión para almanamiento (redes iSCSI).

- Red Iscsi/NFS almacenamiento (Storage Network).

DESCRIPCION INTERFACES Y DIRECCIONAMIENTO.

- *Red de Almacenamiento.*
  - Todos los servidores, equipados con 4 interfaces de 1 Gigabitethernet, utilizarán el interface número 4 – eth3.
  - La subred utilizada será la 192.168.90.0/24
  - El servidor inicial de almacenamiento Cinder1 tendrá la ip 192.168.90.126

- *Red Interna o de Gestión.*
  - El interface gigabitethernet utilizado será el número 1 – eth0.
  - La subred utilizada será la 192.168.73.0/24

- *Red de Datos.*
  - Es la red que conecta entre sí los servidores de computación y con el servidor de networking Neutron.
  - El direccionamiento es dinámico dentro del rango 10.190.240.0/24
  - El interface gigabitethernet utilizado será el segundo, eth1.

- *Red de Conexión (uso externo del api para gestión del Cloud).*
  - Es la red que permite acceder al api de los subsistemas de Openstack. Accederán vía un router con una ip publica nateada al puerto que correspondiera.
  - El interface utilizado será el el tercero, eth2.
  - La subred utilizada será la 192.168.89.0/24

5.3.3 Hardware utilizado

A continuación definiremos el hardware utilizado y las especifidades de cada servidor en función del papel que tendrán dentro de la solución.

Nota: Se ha optado por servidores del fabricante DELL.

Node Controller

- Servidores R720R
  Resumen caracteristicas. http://www.dell.com/us/business/p/poweredge-r720xd/fs

Nodo Red

- Servidores R720R
  Resumen caracteristicas. http://www.dell.com/us/business/p/poweredge-r720xd/fs

Nodo Computación.

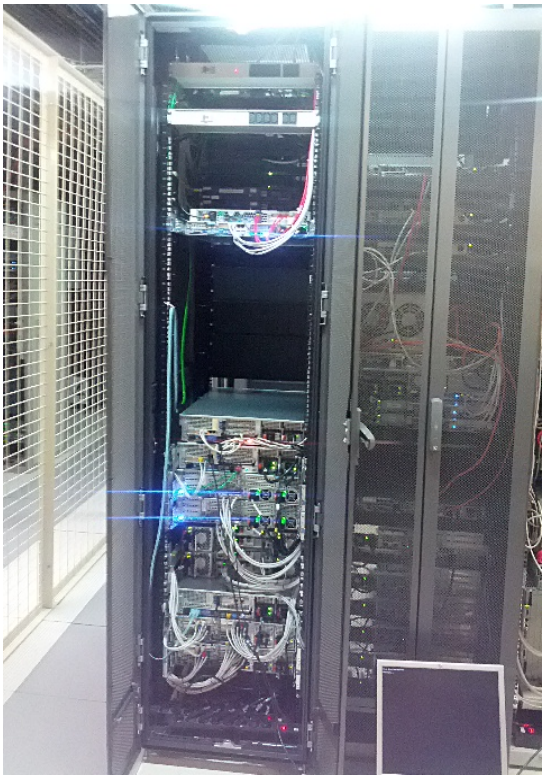
- Servidores R720R
  Resumen caracteristicas. http://www.dell.com/us/business/p/poweredge-r720xd/fs

Nodo Almacenamiento

- Resumen caracteristicas. http://www.dell.com/us/business/p/powervault-md32x0i-series/pd?refid=powervault-md32x0i-series&baynote_bnrank=0&baynote_irrank=0&~ck=baynoteSearch&isredir=true

5.3.4 Fotos Instalación

Foto anterior y posterior. Electrónica de Red en parte superior y servidores en la inferior.
Los servidores de la arquitectura están etiquetados como se indica a continuación:
A) Nodo Controller → Controller [Controller 1]
B) Nodo MonitorYSwift → Swift [Controller 2]
C y D) Nodo Compute1→ Compute1 Nodo Compute2 → Compute12
E) Nodo Red1 → Neutron
F) Nodo Alamacenamiento → Cinder



5.3.5 Instalación y Configuración

El software usado de base para el proyecto es Ubuntu 12.04.3 LTS (64 bits).
http://www.ubuntu.com/download/server

Se dispone de una serie de Scripts en general que se utilizarían para una instalación en un sólo máquina a efectos de poder probar la solución.

En el caso que nos ocupa al tener 5 tipos de servidores se adaptarán para que cada servidor contenga su funcionalidad.

5.3.5.1 Instalación y configuración Openstack Havana

En el Anexo 1 se adjuntan scripts de instalación y configuración para automatizar el proceso de instalación. Todo el proceso de instalación está ampliamente documentado en la documentación de Openstack, si bien hay detalles no descritos lo suficientemente requerido. Los scripts permiten la instalación en pocas horas de un entorno distribuido con n servidores.

Nota: El automatizar la instalación/provisión es una buena idea ya que evita errores de configuración que pueden hacer perder horas posteriormente, analizando los logs y depurando el problema.

Hay tres aspectos que no está automatizados, y que dependen de el tipo de instalación.

El primero es el que corresponde a la red. Dependerá todo del tipo de configuración que se desee, que interfaces,etc.. y no resulta fácil automatizarlo. A tal efecto se incluye la configuración de referencia de los ficheros de configuración de los interfaces.

El segundo corresponde a los hosts. Conviene nombrar los hosts y replicar la configuración de todos ellos con sus ips, en cada host de la arquitectura.

Finalmente, hay algo importante que corresponde los servidores de computación. Si se desea, que es lo normal, que permitan Migraciones en caliente, es preciso que el usuario nova entre todos ellos sea "passwordless" a efectos de conexión ssh.

5.3.6 Sistema de Monitorización y control de Alarmas

La plataforma será monitorizado mediante el entorno Opensource munin y nagios.

El entorno nagios estará orientado a monitorizar todos los Procesos críticos dentro de la arquitectura (que estén activos), a que la carga de las cpus no exceda umbrales del 80%, a que la ocupación de los discos duros no excedan el 80% y que haya pings permanentes sin grandes retardos con todas las máquinas.

Caso de que se activase cualquiera de las alarmas indicadas, nagios enviará un email automáticamente a las cuenta jlp@dominio.es (o cualquier otra que se indicase).

Requerirá la instalación en el servidor Controller 2 del software del Nagios y en todos los restantes del servidor Nagios-nrpe (para gestión remota,etc..). http://www.nagios.org/
Nota: Todo ello además de la instalación del un rango amplio de plugins,etc.. necesarios para el mongodb,mysql,etc...

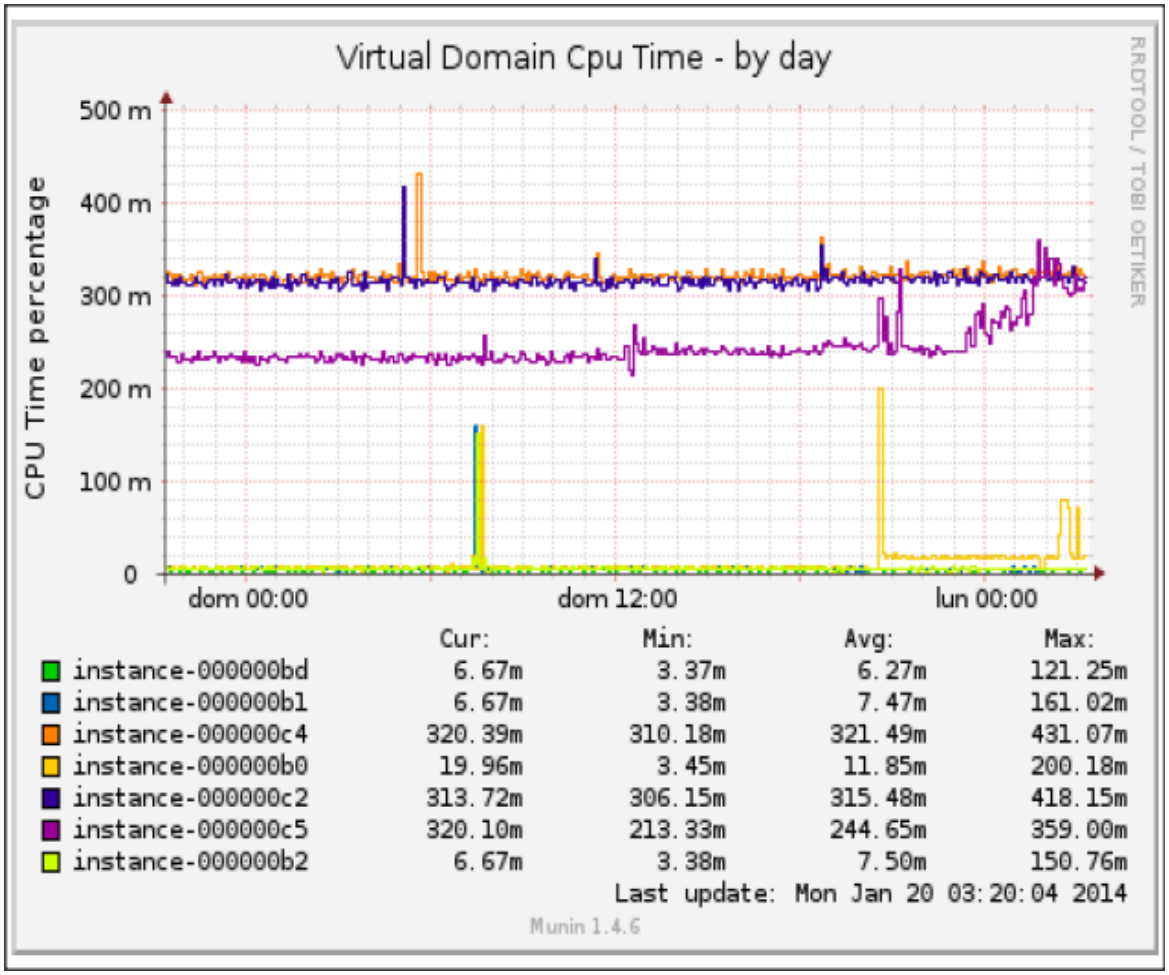
Host	Service	Status	Last Check	Duration	Attempt	Status Information
Monitor_Y_Swift	Current Load	OK	2013-12-31 19:07:41	4d 16h 52m 49s	1/4	OK - load average: 0.23, 0.53, 0.54
	Current Users	OK	2013-12-31 19:08:10	4d 16h 51m 59s	1/4	USERS OK - 0 users currently logged in
	Disk Space	OK	2013-12-31 19:08:38	4d 16h 51m 9s	1/4	DISK OK
	HTTP	OK	2013-12-31 19:09:07	4d 16h 50m 19s	1/4	HTTP OK: HTTP/1.1 200 OK - 454 bytes in 0,001 second response time
	MYSQL	OK	2013-12-31 19:07:35	4d 1h 39m 47s	1/4	Uptime: 333056 Threads: 37 Questions: 5374098 Slow queries: 0 Opens: 275 Flush tables: 1
	Mongo Connect Check	OK	2013-12-31 19:07:46	4d 1h 24m 36s	1/4	OK - Connection took 0 seconds
	Mongo Free Connections	OK	2013-12-31 19:08:14	4d 1h 24m 8s	1/4	OK - 0 percent (8 of 16000 connections) used
	SSH	OK	2013-12-31 19:10:31	4d 16h 49m 29s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1.1 (protocol 2.0)
	Total Processes	OK	2013-12-31 19:10:34	4d 16h 48m 39s	1/4	PROCS OK: 138 processes
cinder	Current Load	OK	2013-12-31 19:11:01	4d 16h 21m 21s	1/4	OK - load average: 1.18, 0.68, 0.58
	Current Users	OK	2013-12-31 19:09:07	4d 16h 20m 52s	1/4	USERS OK - 0 users currently logged in
	Disk Space	OK	2013-12-31 19:09:31	4d 16h 20m 24s	1/4	DISK OK
	SSH	OK	2013-12-31 19:07:27	4d 16h 19m 55s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1.1 (protocol 2.0)
	Total Processes	OK	2013-12-31 19:07:46	4d 16h 24m 36s	1/4	PROCS OK: 138 processes
	check_cinder-api-proc	OK	2013-12-31 19:09:55	4d 16h 16m 8s	1/4	PROCS OK: 2 processes with args 'cinder-api'
	check_cinder-scheduler-proc	OK	2013-12-31 19:10:19	4d 16h 15m 39s	1/4	PROCS OK: 2 processes with args 'cinder-scheduler'
	check_cinder-volume-proc	OK	2013-12-31 19:12:12	4d 16h 15m 10s	1/4	PROCS OK: 3 processes with args 'cinder-volume'
	check_port_cinder-api_8776	OK	2013-12-31 19:10:40	4d 3h 51m 42s	1/4	TCP OK - 0,000 second response time on port 8776
compute1	Current Load	OK	2013-12-31 19:10:09	4d 16h 22m 13s	1/4	OK - load average: 0.05, 0.36, 0.48
	Current Users	OK	2013-12-31 19:10:37	4d 16h 21m 45s	1/4	USERS OK - 1 users currently logged in
	Disk Space	OK	2013-12-31 19:11:06	4d 16h 21m 16s	1/4	DISK OK
	SSH	OK	2013-12-31 19:10:43	4d 16h 20m 48s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1.1 (protocol 2.0)
	Total Processes	OK	2013-12-31 19:11:07	4d 16h 20m 19s	1/4	PROCS OK: 138 processes
	check_libvirtd	OK	2013-12-31 19:08:57	4d 16h 18m 50s	1/4	PROCS OK: 2 processes with args 'libvirtd'
	check_nova_compute	OK	2013-12-31 19:08:51	4d 3h 53m 31s	1/4	PROCS OK: 2 processes with args 'nova-compute'
	check_openvswitch-agent	OK	2013-12-31 19:11:58	4d 16h 16m 3s	1/4	PROCS OK: 2 processes with args 'neutron-openvswitch-agent'
compute2	Current Load	OK	2013-12-31 19:09:15	4d 16h 23m 34s	1/4	OK - load average: 0.11, 0.42, 0.50
	Current Users	OK	2013-12-31 19:10:42	4d 16h 23m 6s	1/4	USERS OK - 1 users currently logged in
	Disk Space	OK	2013-12-31 19:09:45	4d 16h 22m 37s	1/4	DISK OK
	SSH	OK	2013-12-31 19:10:13	4d 16h 22m 9s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1.1 (protocol 2.0)
	Total Processes	OK	2013-12-31 19:10:42	4d 16h 21m 40s	1/4	PROCS OK: 137 processes
	check_libvirtd	OK	2013-12-31 19:09:11	4d 16h 18m 11s	1/4	PROCS ACEPTAR: 2 procesos con args 'libvirtd'
	check_nova_compute	OK	2013-12-31 19:11:55	4d 3h 50m 43s	1/4	PROCS ACEPTAR: 2 procesos con args 'nova-compute'
	check_openvswitch-agent	OK	2013-12-31 19:09:08	4d 16h 18m 14s	1/4	PROCS ACEPTAR: 2 procesos con args 'neutron-openvswitch-agent'
controller	Current Load	OK	2013-12-31 19:07:36	4d 16h 19m 46s	1/4	OK - load average: 0.25, 0.54, 0.54
	Current Users	OK	2013-12-31 19:07:55	4d 16h 24m 27s	1/4	USERS OK - 0 users currently logged in
	Disk Space	OK	2013-12-31 19:09:21	4d 16h 23m 58s	1/4	DISK OK
	HTTP	OK	2013-12-31 19:09:53	0d 1h 7m 29s	1/4	HTTP OK: HTTP/1.1 200 OK - 454 bytes in 0,001 second response time
	SSH	OK	2013-12-31 19:09:50	4d 16h 22m 32s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1.1 (protocol 2.0)
	Total Processes	OK	2013-12-31 19:10:18	4d 16h 22m 4s	1/4	PROCS OK: 147 processes
	check_ceilometer-agent-central	OK	2013-12-31 19:07:42	0d 7h 54m 40s	1/4	PROCS OK: 2 processes with args 'ceilometer-agent-central'
	check_ceilometer-api	OK	2013-12-31 19:10:09	3d 22h 57m 13s	1/4	PROCS OK: 2 processes with args 'ceilometer-api'

El entorno munin permitirá monitorizar gráficamente todos los servicios fundamentales de los servidores y particularmente de los servidores virtualizados.

Requerirá la instalación en el servidor Controller del software Munin y en el resto (también en éste) instalar agentes Munin-node.
Nota: Todo ello además de la instalación del un rango amplio de plugins,etc.. necesarios para el mongodb,mysql,etc...

Por polling cada cierto tiempo, obtendremos en el caso del nagios alarmas por estados incorrectos de procesos,puertos,carga,etc... y en este caso tendremos información gráfica sobre todos los aspectos de interés de los Servidores Host y Huéspedes.

En el caso que nos ocupa mostramos la carga de 16 instancias (VM) en ejecución sobre un servidor Anfitrión (Host Compute1).



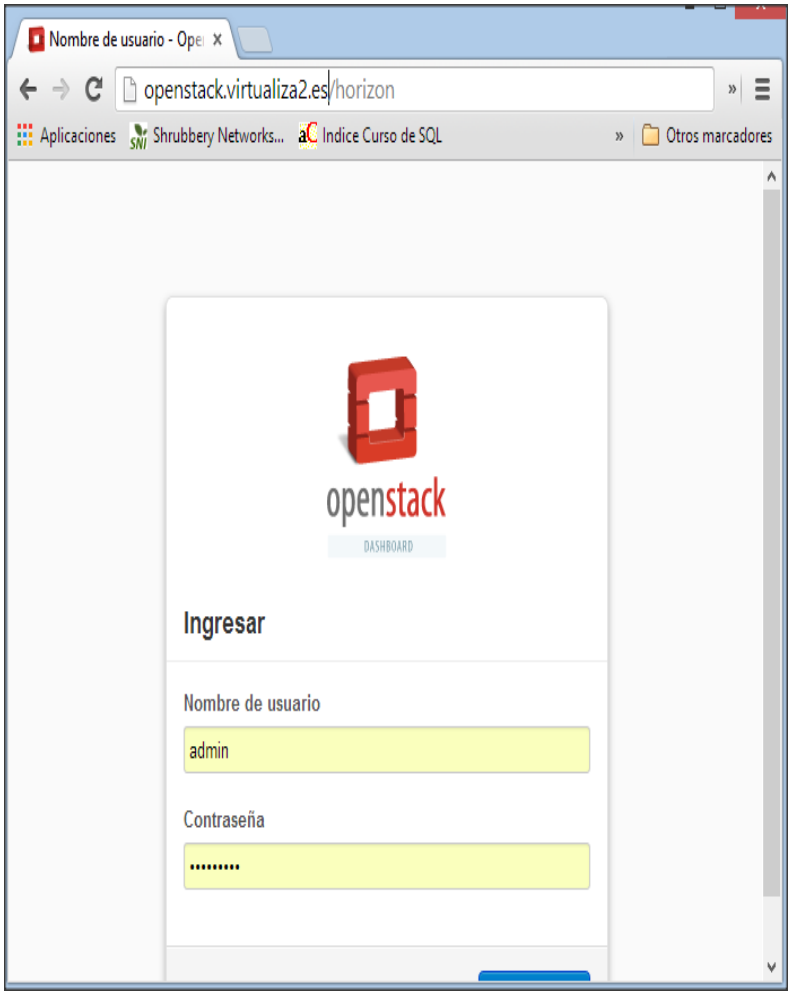
5.4 Interfaces con la Plataformas

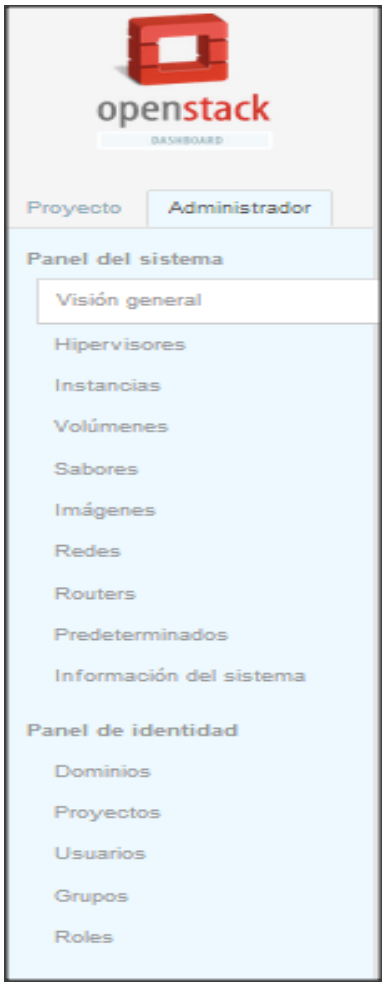
La Plataforma será accesible vía 2 interfaces:

- Vía API remoto (se puede utilizar para pruebas peticiones http mediante el comando curl).
- Via Comando de Linea.
- Via el interface web de Openstack (Horizon).

5.4.1 Interface Web. Horizon Dashboard.

A través de un explorador (cualquiera) podríamos tener acceso a la url http://openstack.dominio.es y tendríamos acceso al login que permitiría acceso a todas los opciones de operación y mantenimiento de los diferentes usuarios, y proyectos, ateniendonos a los privilegios (roles) asignados a cada usuario.





Una vez que el usuario entra al sistema, tendría acceso a menús que permitirían las configuraciones más típicas:

- Lanzar instancias, pararlas, reiniciarlas, terminarlas,etc..
- Crear volúmenes, y asignarlos a instancias
- Cargar imágenes
- Gestionar redes, routers,etc..
- Migrar servidores entre nodos de computación
- etc...

5.4.2 Interfaces con API

Otra vía de gestión de la plataforma que incluso puede ser mucho más ágil es mediante la utilización del api. Ya sea mediante programación, o mediante por ejemplo el uso del comando curl:
curl http://api-openstack.dominio.es:5000/v2.0/ + parámetros.

[root@fc18-atica ~]# curl http://192.168.73.128:5000/v2.0/
{"version": {"status": "stable", "updated": "2013-03-06T00:00:00Z", "media-types": [{"base": "application/json", "type": "application/vnd.openstack.identity-v2.0+json"}, {"base": "application/xml", "type": "application/vnd.openstack.identity-v2.0+xml"}], "id": "v2.0", "links": [{"href": "http://localhost:5000/v2.0/", "rel": "self"}, {"href": "http://docs.openstack.org/api/openstack-identity-service/2.0/content/", "type": "text/html", "rel": "describedby"}, {"href": "http://docs.openstack.org/api/openstack-identity-service/2.0/identity-dev-guide-2.0.pdf", "type": "application/pdf", "rel": "describedby"}]}}

Ello permitiría automatizar cualquier tipo de actividad, desde lanzar o terminar instancias, asignar o desasignar volúmenes,etc...

En los anexos se listarán algunos de los comandos más típicos de gestión con las respuestas típicas que dan cada uno de ellos. El resultado sería análogo al de la utilización de la web.

5.5 Especificación de estándares, normas de diseño y construcción

Es recomendable que los documentos creados a partir de este punto, y que serán motivo de revisión por parte de diferentes personas con diferentes grados de conocimientos técnicos, compartan características y uniformidad, así como un formato adecuado. En nuestro caso particular de ejemplo se optará por:

Documentación de Diseño:

Estos documentos se deben poder consultar tanto por el personal técnico, como por otros miembros de la empresa sin formación de estas características.
Se acuerda trabajar en formato ODT, que es el estándar de Open Office para los documentos susceptibles de sufrir modificaciones con control de correcciones y basado en una plantilla predefinida que contenga: *título* del documento, *responsable* del documento, *lista de autores* que han intervenido y la *fecha* en la cual lo han hecho, *resumen ejecutivo* de cambios introducidos (es decir, el control de versiones) indicando cambio, fecha y autores. Asimismo, se utilizará cada documento y se generará simultáneamente en el formato PDF para su comunicación y publicación a personas vinculadas al proyecto, pero sin capacidad de introducir cambios.

Diagramas de Diseño:

Cuando aplique se utilizará notación UML.

Documentación Técnica:

Se ha creado un blog donde se publicará toda la documentación generada describiendo todo el proceso de instalación, toda la información de construcción del sistema, así como todos la informacioń que fuere necesaria para el mantenimiento. La url del blog es http://www.dominio.es

Este blog contendrá en la Categoría OpenStack toda la información relativa a la arquitectura Openstack, toda la información de instalación y toda la información descriptiva del uso de la plataforma.

Repositorio Software plataforma:

Para facilitar la replicabilidad del entorno objeto de este proyecto:

- Se utilizará algún repositorio especifico accesible mediante herramienta Git.
- Los scripts utilizados de instalación que automatizan el proceso se colgarán igualmente en el blog, y será accesibles desde éste.

5.6 Especificaciones de desarrollo y pruebas

Una vez que se instale el sistema las pruebas que se realizarán son las siguientes:

Funcionalidad del Core:

- Lanzar una instancia. Terminarla.
- Asignar un volumen.
- Asignar ips flotantes.
- Separar nodos de computación. Migración.

Nova Arranque desde un volumen:

- Crear una instancia de arranque
- Crear y arrancar desde un volumen (de arranque)
- Crear una imagen desde un instancia (volumen con copia).

Funcionalidad Red Neutron

- Configurar y usar topologias de red diferentes en Neutron.

Funcionalidad Swift

- Carga y descarga de ficheros.
- Subida de ficheros segmentados.
- Test/Auditoría replicación.
- Añadir dispositivo de almacenamiento a un anillo.

Funcionalidad del Dashboard Horizon

- Login y funcionalidad básica de inicio.
- Arrancar y terminar instancias con Dashboard.
- Gestión proyectos y usarios con Dashboard.
- Acceso mediante VNC a las instancias.

Funcionalidad básica Heat

- Configuración básica Heat.
- Lanzar un stack básico con wordpress.
- Detener y arrancar servicios Heat.

Funcionalidad avanzada Heat

- Ejecutar nose unit tests
- Ejecutar Load Balancing testcase
- Ejecutar Relational Database service testcase
- Ejecutar EIP testcase
- Ejecutar EBS testcase
- Ejecutar EBS and EIP testcase
- Ejecutar Single Instance Wordpress testcase
- Ejecutar Compsed Instances testcase
- Ejecutar 2 instances EBS testcase
- Ejecutar 2 instances with EBS and EIP
- Ejecutar 2 instances Wordpress testcase
- Ejecutar HAProxy Single Instance testcase

- Ejecutar CFN API actions testcase
- Ejecutar AutoScaling testcase

5.6.1 Funcionalidad Ceilometer

- Instalar Ceilometer
- Habilitar medición sobre glance
- Habilitar medición sobre nova
- Análisis inicial de "metering Store".

Todas las pruebas realizadas se documentarán individualmente y serán parte del Anexo I de esta memoria.

5.7 Requisitos de Implantación

La plataforma nueva, llamada a sustituir equipamiento (servidores en producción) se implantará de manera paulatina e inicialmente sólo con servidores/servicios nuevos y no implicará una migración inmediata de los servidores en producción.

Tras un período de maduración (y una vez que se hubiera formado el personal de mantenimiento/soporte) en el uso de las mismas se establecerá un plan para migrar y apagar los servidores actuales que funcionan individualmente. Este plan de migración no es objeto de este proyecto.

6 Implantación

El objetivo que se persigue es la paulatina incorporación del nuevo entorno para la instalación de nuevos servidores, y la paulatina migración de los servidores ya existentes al nuevo entorno. Todo se hará pausadamente y posteriormente a la formacioń que se impartiría. Siempre a efectos de confirmar la estabilidad de la plataforma y el nivel de capacitación suficiente del personal para dar soporte a la misma.

6.1 Formación

Una vez que se haga entrega de la plataforma, se le transferirá al Responsable de Sistemas de la Compañía toda la documentación del proyecto a efectos de que éste puede impartir la formación necesaria para la operación y mantenimiento del entorno. El Plan de formación no forma parte del alcance del proyecto.

6.2 Implantación del sistema y pruebas y nivel de servicios

Previa a la puesta en producción del sistema, se harán pruebas integrales del Sistema simulando la puesta en producción de diferentes máquinas virtuales con distintas capacidades: de computación, almacenamiento, cpus, memoria y direccionamiento privado y publico.

- Concretamente se crearía un proyecto Cloud con un usuario administrador que tendría que lanzar las siguientes instancias:

a) 3 máquinas de 1 CPU – 10 Gigas de Disco – 1 Giga de RAM – Imagen Ubuntu 12.04.03 – con 1 ip pública.

b) 3 máquinas de 1 CPU – 50 Gigas de Disco – 2 Gigas de RAM – Imagen Ubuntu 12.04.03 – con 1 ip pública.

c) 3 máquinas de 2 CPUs – 100 Gigas de Disco – 4 Gigas de RAM – Imagen Ubuntu 12.04.03 – con 1 ip pública.

- Se harían pruebas accediendo vía las ips públicas a las 9 máquinas.
- Se añadirían volúmenes de 20 Gigas a 1 server de cada grupo.
- Se crearían snapshots de 1 server de cada grupo.
- Se recuperaría la máquina vía el snapshot.
- Se analizaría el rendimiento/gráficas de las variables operativas más importantes de las máquinas vía las herramientas de monitorización. Así como el envió de alertas frente a fallos de las máquinas.

  Se documentarían las pruebas y se examinarían conjuntamente con el Responsable de Sistemas.

7 Futuro del Proyecto OpenStack

El fúturo del Proyecto Openstack es muy prometedor. La adopción de este Proyecto OpenSource en el Proyecto en cuestión no implica asumir riesgo alguno a corto/medio plazo.

Actualmente intentan poner el foco en mejorar la cultura,organización y gobernanza de la Comunidad, y mejorar la calidad del código, del que se dice, no sin falta de razón que tiene demasiados Bugs (que sin embargo se corrigen muy rápido).

Hay nuevos proyectos en marcha para la próxima release de OpenStack que se liberará en Abril de 2014 "IceHouse" (se incluirán algunos de los siguientes en la próxima Release):

- Trove (previo nombre RedDwarf) → Módulo/Proyecto para ofrecer servicios tipo Database as a Service.
- Marconi → Módulo/Proyecto para ofrecer servicios de Mensajes
- Savanah → Módulo/Proyecto para ofrecer Hadoop as a Service.
- Manila → Módulo/Proyecto relativo a nuevos Sistemas de Almacenamiento.
- Solem → Módulo/Proyecto relativo a servicios PaaS.
- Designate DNSaaS → Módulo/Proyecto relativo a servicios DNSaaS
- Triple O → Módulo/Proyecto que permitirá a los Administradores Cloud desplegar Clouds Openstack dentro de una nube OpenStack.

Lo indicado previamente no hace sino confirmar la ambición de generar nuevos proyectos que se puedan encajar voluntariamente como piezas de Lego dentro de la completa arquitectura.

El Proyecto en su conjunto está muy apoyado por pesos pesados de la Industria que es consciente de que los dos "enemigos" que se encuentran a cierta distancia por delante son Amazon AWS y Vmware.

Simplemente para corroborar lo anteriormente dicho, recientemente se han publicitado acuerdos de colaboración de gran alcance entre DELL y Red HAT para expandir la solución OpenStack.

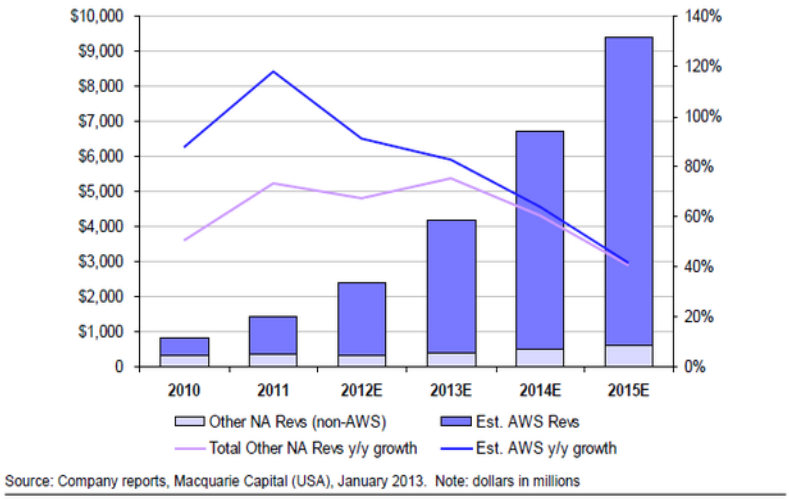
http://diarioti.com/dell-y-red-hat-desarrollaran-conjuntamente-soluciones-cloud-de-openstack-empresariales/72435

8 Futuro de servicios Amazon Aws

Por otra parte, en lo que refiere a Amazon AWS, su liderazgo en servicios IaaS (precio y calidad de servicio) será incuestionable en los próximos años, y no se vislumbra a medio plazo nadie que pueda hacerles sombra como Proveedores de Servicios.

No deja de sorprender que incluso bajando en sucesivas ocasiones los precios de los servicios que ofrece, su capacidad de crecer parece ilimitada. Hay análisis que valoran la compañía en los próximos 18 meses en unos 50.000 millones de US$.

Fig 8 Estimated AWS Revenue Growth



Source: Company reports, Macquarie Capital (USA), January 2013. Note: dollars in millions

http://readwrite.com/2013/11/20/amazon-web-services-business-worth-50-billion-by-2015-analyst-projects#awesm=~orshU1raHDcYSi

9 Conclusiones

Estoy muy contento con el Proyecto abordado y con el resultado del mismo.

* Por razones profesionales siempre había deseado implantar un sistema Cloud y OpenStack ha sido todo un descubrimiento para mí. El interés del proyecto era que fuera real , y que fuera de aplicación el conocimiento adquirido y la solución implantada, para arrancar 50,100, 500,... servidores virtuales con imágenes de Linux, de Windows,etc.. en producción real y se ha logrado plenamente. La arquitectura me ofrece garantías para soportar grandes capacidades de Computación y Almacenamiento en plena producción. Tanto con Servidores Windows como Linux.

Me he visto obligado a instalar unas cuantas veces el entorno y sus módulos, y a utilizar con frecuencia los logs para depurar problemas, y me ha ayudado a entender una gran parte del sentido de cada Módulo, su operativa,etc.. Ha sido un proceso duro, pero muy enriquecedor.

Desde esta perspectiva los objetivos se han cumplido plenamente y estoy satisfecho.

* También quería abordar dentro del proyecto como alternativa la opción de usar Amazon Aws, y me ha resultado interesante el uso de los servicios de Amazon Aws y su comparación, incluso a nivel de opciones, con lo que te ofrece OpenStack. Ha sido realmente interesante y como no podría ser de otra forma, en computación y almacenamiento los servicios/opciones son semejantes.

Desde esta otra perspectiva (de Amazon AWS) los objetivos se han cumplido plenamente y estoy satisfecho.

* Volviendo a OpenStack, he de decir que al principio me parecía una arquitectura pesada y demasiado complicada, pero una vez que la he ido entendiendo y he ido viendo su utilidad, he de decir que es lo que más me ha gustado, como se encajan los Módulos para dar servicios vía APIs específicos. Ha sido un descubrimiento.

* Las nuevas estructuras de Almacenamiento de Objetos (S3, Swift, etc..) han sido un descubrimiento que desconocía, y que considero muy interesante. Nota: Ahora entiendo como funciona Dropbox y otros servicios similares.

* Las opciones que se abren a nivel de Redes Definidas por Software con el OpenStack/Neutron (openvswitch) me han parecido extraordinarias. Me ha costado entenderlo, pero es de lo que entiendo más complejo y de lo sin duda más importante dentro de la Arquitectura. Estoy muy contento de haber tenido que afrontarlo, y es algo en lo quiero profundizar en el futuro, redes SDN,etc...

* Veo como una necesidad imperiosa que las redes utilizadas por estos entornos sean de 10 Gigaethernet para mejorar el rendimiento, por indisponibilidad de equipamiento he usado 1 gigaethernet y se nota. También lo he leido en diferentes documentos técnicos de fabricantes.

* Para obtener mejores prestaciones tuve que hacer cambios y redistribuir los procesos en los controladores (que contenían demasiados en un servidor no demasiado potente). Al hacerlo se notaba que mejora mucho el rendimiento general de la plataforma. Desde luego mejorando el tema del acceso a disco mejora el rendimiento de la plataforma, y es un aspecto crítico.

* Inicialmente temía que no hubiera demasiada información, y la verdad es que la hay, muy extensa y bien estructurada dentro del Proyecto OpenStack. Es un proyecto bien soportado y considero que es una Alternativa de Garantías. Todo ello, aun cuando, el proyecto OpenStack supongo que por la rapidez , y la juventud que tiene, no tiene muy afinados totalmente los procesos de Calidad del Software, pero se vislumbra con claridad que terminarán solucionando estas deficiencias.

Algunos aspectos que me hubiera gustado abordar, aun cuando estaban fuera del Alcance del Proyecto, pero que por carencia de tiempo no han podido afrontarse, mencionaría los siguientes (*todos suceptibles de ser abordados en posteriores Proyectos como ampliaciones del mismo*):

* Como ya he indicado, lo relativo a la conexión entre sí de los diferentes proyectos vía API, es extremadamente flexible. Me hubiera gustado disponer de más tiempo para afrontar algún proyecto de programación de Automatizadores de Servicios mediante el uso de los correspondientes APIs, a través de los SDKs,etc...

* He podido meterme algo con los "schedulers" de computación, y almacenamiento, pero sin duda, tienen mucha flexibilidad para poder distribuir la asignación siguiendo diferentes lógicas. Puede tener grandes implicaciones en el rendimiento de la plataforma globalmente, y considero interesante profundizar por ese camino.

* He podido hacer pruebas por encima de las VM, pero no hasta el extremo de poder simular carga y ver rendimiento con máquinas similares no virtualizadas.

* Me hubiera gustado poder dedicarle más tiempo a Swift con suficientes servidores para establecer una estructura real completa de alta capacidad. Por otra parte he visto que OpenStack puede trabajar con entornos novedosos de almacenamiento como Ceph que desconocía, también GlusterFs,etc.., y considero interesante la inmersión en ello.

* Ceilometer como Proyecto OpenStack está recién salido del horno, y funciona correctamente, pero todavía no dispone de interfaces que aprovechen toda la información que aporta, por ejemplo para Tarificar los servicios según modelos de "Pay as you go". Es interesante trabajar en esta dirección según entiendo.

* Alrededor de los sistemas de orquestación HEAT y las plantillas hay todo un mundo, en el que caben todos los escenarios de servicios,etc... He probado escenarios de servicios que funcionan perfectamente e igulamente he llegado a ver los escenarios de configuración de servicios LbaaS, pero no llegué a probarlos. Juju de Ubuntu me parece igualmente muy interesante como un híbrido en ensamblado de paquetes y sistema de orquestación. Hubiera querido también profundizar en esto.

* Otros aspectos que considero interesantes para profundizar en ellos es lo relativo a los sistemas de Gestión y Automatización, como Puppets, Chef, DevOps, Ansible, etc.. que pudiera ser parte de otro Proyecto.

Tengo la intención de seguir en lo sucesivo el proyecto y como he dicho que el mismo sea de aplicación directa en los Proyectos que tengo bajo mi responsabilidad profesional en la Empresa en la actualmente trabajo.

Finalmente, quiero agradecer a mis tutores toda la ayuda aportada durante la relación de este Proyecto fin de Master.

10 Bibliografía

- Wikipedia → http://es.wikipedia.org http://en.wikipedia.org
- Documentación Amazon AWS → http://aws.amazon.com/es/documentation/
- Documentación OpenStack → http://docs.openstack.org/ http://www.openstack.org
- Documentacion KVM → http://www.linux-kvm.org/page/Documents
- Documentación Euca2ools → https://www.eucalyptus.com/docs
- Documentacón Xen → http://www.xenproject.org/help/documentation.html
- Documentación Ubuntu → https://help.ubuntu.com/
- Documentación RedHat → http://openstack.redhat.com/Quickstart
- Documentación Rackspace → http://www.rackspace.com/es/?CMP=GEO_US
- Documentación SwiftStack → http://swiftstack.com/
- RightScale → http://www.rightscale.com/
- Documentación OpenNebula → http://opennebula.org/
- Documentación Nagios → http://www.nagios.org/
- Documentación Munin → http://munin-monitoring.org/
- Documentación Ganglia → http://ganglia.sourceforge.net/
- Documentación MongoDb → http://www.mongodb.org/
- Blog Julien → http://julien.danjou.info/blog/
- CloudBase → http://www.cloudbase.it/ws2012r2/
- StackOps → http://www.stackops.com/
- Deploying Openstack → http://it-ebooks.info/book/563/
- Openstack cloud Computing Openbook → http://books.google.es/booksid=ej29HPm0r30C&hl=es&source=gbs_similarbooks

11 Anexos

11.1 Anexo 1 : Ficheros de instalación Scripts

11.2 Anexo 2 : Ficheros de Configuración.

11.3 Anexo 3 : Pruebas Amazon AWS

11.4 Anexo 4 : Pruebas OpenStack