TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

CONFIGURACIÓN DE UNA PLATAFORMA COMPUTACIONAL BASADA EN OPENSTACK PARA LA SIMULACIÓN DE PROCESOS INDUSTRIALES.

Curso 2015/2016

Alumno/a:

FRANCISCO JOSÉ GÁMIZ ARIAS

Director/es:

VICENTE GONZÁLEZ RUIZ

LUIS JOSÉ YEBRA MUÑOZ

Antes de empezar, me gustaría agradecer a las personas que han hecho posible el poder realizar este proyecto, empezando por mis tutores, **Vicente González** y **Luis J. Yebra**, que han estado en todo momento pendientes de mi progreso en el proyecto.

Dar agradecimientos especiales a **José Antonio Martinez García**, que aunque de manera extraoficial ha sido otro de los tutores en el trabajo, no se le ha podido incluir como tal por trámites administrativos. Sin él probablemente habría estado dando muchos tumbos en gran parte del proyecto y por eso gran parte del mérito de este trabajo es sin duda suyo.

Agradecer también a **mis amigos**, que siempre me han dado caña para que termine lo antes posible el proyecto para que pase más tiempo con ellos.

Y por último, pero no menos importante, agradecer a **la familia**, que siempre está ahí en las buenas y en las malas y siempre te dan fuerzas para superar las situaciones más difíciles que puedas imaginar, no solo incluyendo este proyecto.

ÍNDICE

1. INTRODUCCIÓN

1.1. Introducción a CIESOL. Historia y estructura

CIESOL [1] (Véase Ilustración 1) es un centro de investigación creado y gestionado en base a un convenio de colaboración firmado en abril de 2005 entre la Universidad de Almería y la Plataforma Solar de Almería del Centro de Investigaciones Energéticas Medioambientales y Tecnológicas (CIEMAT) del Ministerio de Economía y Competitividad. Está situado en el Campus Universitario de la Universidad de Almería y tanto el edificio que lo aloja como sus infraestructuras energéticas han sido financiados por fondos FEDER[1], la Junta de Andalucía y por el Proyecto Nacional de investigación de Carácter Singular y Estratégico sobre Arquitectura Bioclimática y Frío Solar (ARFRISOL).

Ilustración 1. Logo de CIESOL

En el centro (Véase Ilustración 2) se realizan actividades de investigación y de transferencia tecnológica relacionadas con las aplicaciones de la energía solar en las siguientes áreas: la química sostenible, la regeneración de aguas, el análisis ambiental, el modelado y el control automático de instalaciones solares, la domótica orientada a la eficiencia energética, el frío solar y la evaluación de recursos solares.

CIESOL recoge la experiencia de más de 20 años de colaboración entre grupos de investigación de la Universidad de Almería y la Plataforma Solar de Almería en los cuales se ha consolidado un vínculo formal que en abril de 2005 quedó reflejado en la firma de un convenio entre ambas instituciones para la creación de una estructura estable en forma de centro de investigación conjunto.

En su origen, la colaboración se centró en las áreas Química Solar y de Control de Instalaciones termosolares y a través de la misma investigadores de la Universidad de Almería contribuyeron, por ejemplo, a la mejora de las capacidades analíticas de los laboratorios de la PSA o al diseño e implementación de sistemas SCADA[2] específicos para diversos lazos y campos de ensayo de procesos solares térmicos de media y alta temperatura.

Ilustración 2. Edificio CIESOL

Con posterioridad se incorporaron actividades en el área de evaluación de recursos solares y, a partir de 2006, por la participación de la Universidad de Almería en el Proyecto Nacional de Carácter Singular y Estratégico sobre Arquitectura Bioclimática y Frío Solar ARFRISOL liderado por la Unidad de Eficiencia Energética en la Edificación del CIEMAT, se han añadido a las líneas de colaboración las de control orientado al confort térmico y al ahorro energético en edificios y el estudio de sistemas de refrigeración solar. Este proyecto ha permitido también contar con un conjunto de infraestructuras propias avanzadas dedicadas a garantizar la autosuficiencia energética del edificio.

En 2014 se incorpora la unidad funcional de Desalación y Fotosíntesis. Esta nueva unidad del CIESOL recoge la colaboración previa entre investigadores del Área de Ingeniería Química de la Universidad de Almería, que aporta siete investigadores, y de la Unidad de Desalación Solar de la Plataforma Solar de Almería, que aporta tres investigadores. Como aspectos particulares pueden citarse en primer lugar el hecho de que sus áreas de especialización son perfectamente integrables en las líneas de actuación del CIESOL y favorecerán la transversalidad y colaboración con el resto de unidades del centro y, en segundo lugar, que la nueva unidad acredita una contrastada y amplia experiencia en proyectos internacionales.

Aparte de esta colaboración directa entre investigadores, se han establecido a lo largo de este periodo diversos acuerdos para la financiación de becas y ayudas que han favorecido el acceso a las instalaciones de la PSA así como la formación especializada a decenas de estudiantes de la UAL, siendo uno de los resultados más relevantes del mismo la lectura de más de 20 tesis doctorales de alumnos de la UAL codirigidas por investigadores de la PSA y con uso de infraestructuras y recursos científicos de la misma.

La estructura funcional del centro (Véase Ilustración 3) está constituida por un comité de coordinación, órgano máximo de decisión y gestión, un equipo de dirección, al que pertenecen dos investigadores, uno por cada organismo matriz y un conjunto de 6 unidades que agrupan a los investigadores de las áreas temáticas específicas a efectos de promoción y gestión de proyectos y actividades en convocatorias públicas o en contratos con empresas. De forma suplementaria, el centro cuenta con comité evaluador externo que anualmente valora y supervisa la producción científica de sus diferentes unidades funcionales. En cuanto a su financiación, el centro cuenta con las asignaciones específicas vinculadas a cada uno de los proyectos y contratos en curso y con una aportación fija anual asumida de forma igualitaria por ambas instituciones para los gastos corrientes y de funcionamiento del centro.

Finalmente, el centro cuenta con una unidad de apoyo técnico a las infraestructuras científicas alojadas en el mismo, todas ellas de alto nivel y a disposición de los proyectos de investigación en las sus grandes áreas de trabajo, aplicaciones químicas de la energía solar y aprovechamiento térmico de la misma en los edificios e industrias.

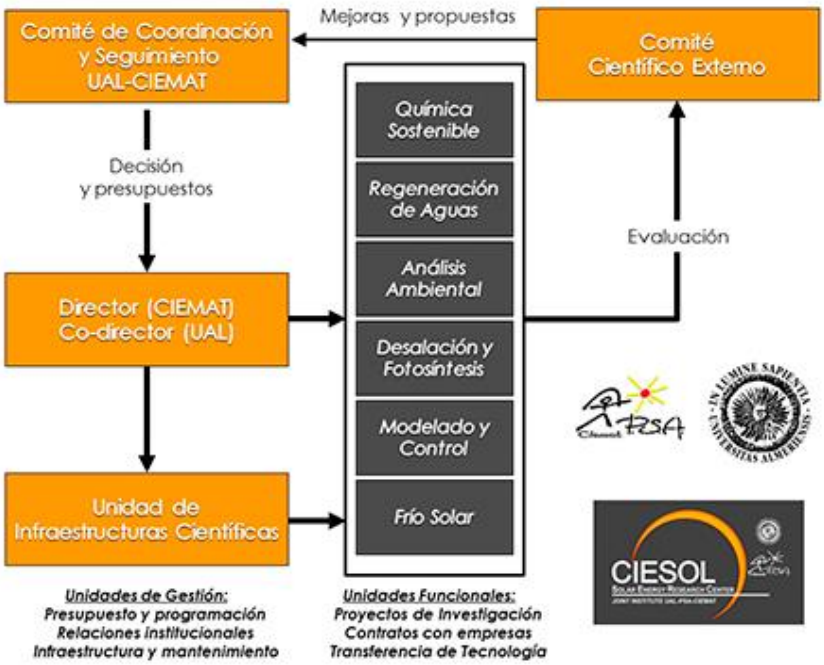


Ilustración 3. Organigrama funcional PSA-UAL

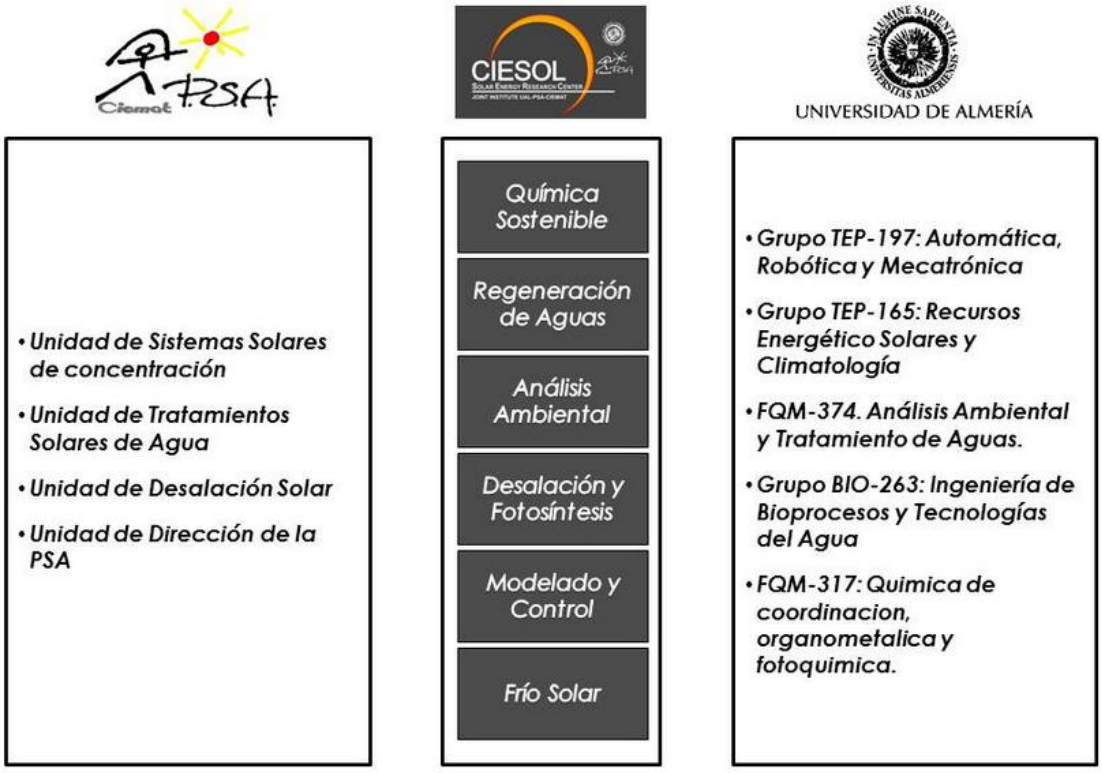


Ilustración 4. Grupos que forman CIESOL

1.2. Introducción a la PSA

La *Plataforma Solar de Almería (PSA)* [2], perteneciente al Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), es el mayor centro de investigación, desarrollo y ensayos de Europa dedicado a las tecnologías solares de concentración. La PSA desarrolla sus actividades integrada como una División de I+D dentro de la estructura del Departamento de Energía del CIEMAT.

Es un centro público de investigación del Gobierno de España, oficialmente considerado por la Comisión Europea como una Gran Instalación Científica europea y es también el mayor y más completo centro de I+D en el mundo dedicado a los sistemas de concentración solar térmica. La PSA está integrada en la organización de CIEMAT como una división de I+D del Departamento de Energía. La PSA es además una Infraestructura Científica y Técnica Singular (ICTS) del gobierno de España.

Las buenas condiciones solares, sus diversas instalaciones solares (Véase Ilustración 5) y el personal altamente cualificado de la PSA, proporcionan una infraestructura única para I+D, evaluación, demostración, formación y transferencia de la tecnología relacionada con aplicaciones de energía solar.

Ilustración 5. Vista aérea de la PSA

La PSA está situada en el sureste de España, en el desierto de Tabernas a 37º05'27.8" norte y 2º21'19" oeste. Recibe una radiación solar anual de más de 1900 kWh/(m^2•año) y la temperatura media anual es alrededor de 17ºC. La PSA cuenta con más de 30 años de experiencia en la operación, mantenimiento y evaluación de sistemas solares de concentración, sus componentes y distintos tipos de aplicaciones comerciales. Actualmente, la PSA tiene una gran variedad de instalaciones experimentales y laboratorios de I+D para actividades relacionadas con los sistemas de concentración solar térmica.

Los siguientes objetivos motivan sus actividades de investigación:

- Contribuir al establecimiento de un suministro energético mundial limpio y sostenible
- Contribuir a la conservación de las fuentes de energía europeas y a la protección de su clima y medioambiente.
- Promover la introducción en el mercado de las tecnologías termosolares y aquellas derivadas de procesos químicos solares.
- Contribuir al desarrollo de una industria termosolar española de exportación, competitiva.
- Reforzar la cooperación entre la industria y las instituciones científicas en el campo de la investigación, desarrollo, demostración y mercado de las tecnologías termosolares.
- Fortalecer la reducción de costes de las innovaciones tecnológicas contribuyendo al aumento de la aceptación en el mercado de las tecnologías termosolares.
- Promover una cooperación tecnológica norte-sur, especialmente en la zona mediterránea.
- Asistir a la industria en la identificación de oportunidades del mercado termosolar.

1.3. Introducción al cloud computing. OpenStack.

1.3.1. Definición e introducción

Si hablamos de OpenStack, debemos hablar de la computación en la nube, o lo que es lo mismo, *cloud computing* [3].

Este concepto, conocido también como servicio en la nube, informática en la nube o nube de cómputo, es un paradigma que permite ofrecer servicios de computación a través de una red, que usualmente es Internet.

En este tipo de computación todo lo que puede ofrecer un sistema informático se ofrece como servicio, de modo que los usuarios puedan acceder a los servicios disponibles "en la nube de Internet" sin conocimientos (o, al menos sin ser expertos) en la gestión de los recursos que usan. Según el IEEE Computer Society, es un paradigma en el que la información se almacena de manera permanente en servidores de Internet y se envía a cachés.

La computación en la nube son servidores desde Internet encargados de atender las peticiones en cualquier momento. Se puede tener acceso a su información o servicio, mediante una conexión a internet desde cualquier dispositivo móvil o fijo ubicado en cualquier lugar. Sirven a sus usuarios desde varios proveedores de alojamiento repartidos frecuentemente por todo el mundo.

Esta medida reduce los costos, garantiza un mejor tiempo de actividad y que los sitios web sean invulnerables a los delincuentes informáticos, a los gobiernos locales y a sus redadas policiales.

La computación en la nube es un nuevo modelo de prestación de servicios de negocio y tecnología, que permite incluso al usuario acceder a un catálogo de servicios estandarizados y responder con ellos a las necesidades de su negocio, de forma flexible y adaptativa, en caso de demandas no previsibles o de picos de trabajo, pagando únicamente por el consumo efectuado, o incluso gratuitamente en caso de proveedores que se financian mediante publicidad o de organizaciones sin ánimo de lucro.

El cambio que ofrece la computación en la nube es que permite aumentar el número de servicios basados en la red (Véase Ilustración 6). Esto genera beneficios tanto para los proveedores, que pueden ofrecer, de forma más rápida y eficiente, un mayor número de servicios, como para los usuarios que tienen la posibilidad de acceder a ellos, disfrutando de la 'transparencia' e inmediatez del sistema y de un modelo de pago por consumo. Así mismo, el consumidor ahorra los costes salariales o los costes en inversión económica (locales, material especializado, etc.).

Ilustración 6. Ejemplo de servicios en la nube.

La computación en la nube consigue aportar estas ventajas, apoyándose sobre una infraestructura tecnológica dinámica que se caracteriza, entre otros factores, por un alto grado de automatización, una rápida movilización de los recursos, una elevada capacidad de adaptación para atender a una demanda variable, así como virtualización avanzada y un precio flexible en función del consumo realizado, evitando además el uso fraudulento del software y la piratería.

El concepto de “nube informática” es muy amplio, y abarca casi todos los posibles tipo de servicio en línea, pero cuando las empresas predican ofrecer un utilitario alojado en la nube, por lo general se refieren a alguna de estas tres modalidades: el Software como Servicio (en inglés SaaS -Software as a Service-), Plataforma como Servicio (PaaS) e Infraestructura como Servicio (IaaS).

El software como servicio (SaaS) es un modelo de distribución de software en el que las aplicaciones están alojadas por una compañía o proveedor de servicio y puestas a disposición de los usuarios a través de una red, generalmente la Internet.

Plataforma como servicio (PaaS) es un conjunto de utilitarios para abastecer al usuario de sistemas operativos y servicios asociados a través de Internet sin necesidad de descargas o instalación alguna.

Infraestructura como Servicio (IaaS) se refiere a la subcontratación de los equipos utilizados para apoyar las operaciones, incluido el almacenamiento, hardware, servidores y componentes de red.

El concepto de la computación en la nube empezó en proveedores de servicio de Internet a gran escala, como Google, Amazon AWS, Microsoft y otros que construyeron su propia infraestructura.

De entre todos ellos emergió una arquitectura: un sistema de recursos distribuidos horizontalmente, introducidos como servicios virtuales de TI escalados masivamente y manejados como recursos configurados de manera continua.

1.3.2. Historia

Históricamente [3], el concepto fundamental de la entrega de los recursos informáticos a través de una red global tiene sus raíces en los años sesenta. La idea de una "red de computadoras intergaláctica" la introdujo en los años sesenta JCR Licklider, cuya visión era que todo el mundo pudiese estar interconectado y poder acceder a los programas y datos desde cualquier lugar, según Margaret Lewis, directora de mercadotecnia de producto de AMD. "Es una visión que se parece mucho a lo que llamamos cloud computing."

Otros expertos atribuyen el concepto científico de la computación en nube a John McCarthy (Véase Ilustración 7), quien propuso la idea de la computación como un servicio público, de forma similar a las empresas de servicios que se remontan a los años sesenta. En 1960 dijo: "Algún día la computación podrá ser organizada como un servicio público."

Ilustración 7. John McCarthy, pionero del cloud computing

Desde los años sesenta, la computación en nube se ha desarrollado a lo largo de una serie de líneas. La Web 2.0 es la evolución más reciente. Sin embargo, como Internet no empezó a ofrecer ancho de banda significativo hasta los años noventa, la computación en la nube ha sufrido algo así como un desarrollo tardío. Uno de los primeros hitos de la computación en nube es la llegada de Salesforce.com en 1999, que fue pionero en el concepto de la entrega de aplicaciones empresariales a través de una página web simple. La firma de servicios allanó el camino para que tanto especialistas como empresas tradicionales de software pudiesen publicar sus aplicaciones a través de Internet.

El siguiente desarrollo fue Amazon Web Services en 2002, que prevé un conjunto de servicios basados en la nube, incluyendo almacenamiento, computación e incluso la inteligencia humana a través del Amazon Mechanical Turk. Posteriormente en 2006, Amazon lanzó su Elastic Compute Cloud (EC2) como un servicio comercial que permite a las pequeñas empresas y los particulares alquilar equipos en los que se ejecuten sus propias aplicaciones informáticas.

"Amazon EC2/S3 fue el que ofreció primero servicios de infraestructura en la nube totalmente accesibles", según Jeremy Allaire, CEO de Brightcove, que proporciona su plataforma SaaS de vídeo en línea a las estaciones de televisión de Reino Unido y periódicos. George Gilder dijo en 2006: "El PC de escritorio está muerto. Bienvenido a la nube de Internet, donde un número enorme de instalaciones en todo el planeta almacenarán todos los datos que usted podrá usar alguna vez en su vida."

Otro hito importante se produjo en 2009, cuando Google y otros empezaron a ofrecer aplicaciones basadas en navegador. "La contribución más importante a la computación en nube ha sido la aparición de 'aplicaciones asesinas'[3] de los gigantes de tecnología como Microsoft y Google. Cuando dichas compañías llevan a cabo sus servicios de una manera que resulta segura y sencilla para el consumidor, el efecto 'pasar la pelota' en sí crea un sentimiento de mayor aceptación de los servicios online", según Dan Germain, jefe de la oficina de tecnología en IT proveedor de servicios Cobweb Solutions.

Otro de los factores clave que han permitido evolucionar a la computación en la nube ha sido, según el pionero en computación en la nube británico Jamie Turner, las tecnologías de virtualización, el desarrollo del universal de alta velocidad de ancho de banda y normas universales de interoperabilidad de software. Turner añadió: "A medida que la computación en la nube se extiende, su alcance va más allá de un puñado de usuarios de Google Docs. Sólo podemos empezar a imaginar su ámbito de aplicación y alcance. Casi cualquier cosa puede ser utilizada en la nube".

1.3.3. Características

En cuanto a las *características* [3] tenemos las siguientes:

- Agilidad: Capacidad de mejora para ofrecer recursos tecnológicos al usuario por parte del proveedor.
- Costo: los proveedores de computación en la nube afirman que los costos se reducen. Un modelo de prestación pública en la nube convierte los gastos de capital en gastos de funcionamiento. Ello reduce barreras de entrada, ya que la infraestructura se proporciona típicamente por una tercera parte y no tiene que ser adquirida por una sola vez o tareas informáticas intensivas infrecuentes.
- Escalabilidad y elasticidad: aprovisionamiento de recursos sobre una base de autoservicio en casi en tiempo real, sin que los usuarios necesiten cargas de alta duración.
- Independencia entre el dispositivo y la ubicación: permite a los usuarios acceder a los sistemas utilizando un navegador web, independientemente de su ubicación o del dispositivo que utilice (por ejemplo, PC, teléfono móvil).
- La tecnología de virtualización permite compartir servidores y dispositivos de almacenamiento y una mayor utilización. Las aplicaciones pueden ser fácilmente migradas de un servidor físico a otro.
- Rendimiento: Los sistemas en la nube controlan y optimizan el uso de los recursos de manera automática, dicha característica permite un seguimiento, control y notificación del mismo. Esta capacidad aporta transparencia tanto para el consumidor o el proveedor de servicio.

- Seguridad: Puede mejorar debido a la centralización de los datos. La seguridad es a menudo tan buena o mejor que otros sistemas tradicionales, en parte porque los proveedores son capaces de dedicar recursos a la solución de los problemas de seguridad que muchos clientes no pueden permitirse el lujo de abordar. El usuario de la nube es responsable de la seguridad a nivel de aplicación. El proveedor de la nube es responsable de la seguridad física.
- Mantenimiento: En el caso de las aplicaciones de computación en la nube es más sencillo, ya que no necesitan ser instalados en el ordenador de cada usuario y se puede acceder desde diferentes lugares.

1.3.4. Ventajas y desventajas / limitaciones.

La computación en la nube, como era de esperar, tiene una serie de *ventajas e inconvenientes* [3].

Las ventajas del cloud computing son las siguientes:

- ✓ Integración probada de servicios Red: Por su naturaleza, la tecnología de cloud computing se puede integrar con mucha mayor facilidad y rapidez con el resto de las aplicaciones empresariales (tanto software tradicional como cloud computing basado en infraestructuras), ya sean desarrolladas de manera interna o externa.
- ✓ Prestación de servicios a nivel mundial: Las infraestructuras de cloud computing proporcionan mayor capacidad de adaptación, recuperación completa de pérdida de datos (con copias de seguridad) y reducción al mínimo de los tiempos de inactividad.
- ✓ Una infraestructura 100% de cloud computing permite también al proveedor de contenidos o servicios en la nube prescindir de instalar cualquier tipo de software, ya que éste es provisto por el proveedor de la infraestructura o la plataforma en la nube. Un gran beneficio del cloud computing es la simplicidad y el hecho de que requiera mucha menor inversión para empezar a trabajar.
- ✓ Implementación más rápida y con menos riesgos: Se comienza a trabajar más rápido y no es necesaria una gran inversión. Las aplicaciones del cloud computing suelen estar disponibles en cuestión de días u horas en lugar de semanas o meses, incluso con un nivel considerable de personalización o integración.
- ✓ Actualizaciones automáticas que no afectan negativamente a los recursos de TI: Al actualizar a la última versión de las aplicaciones, el usuario se ve obligado a dedicar tiempo y recursos para volver a personalizar e integrar la aplicación. Con el cloud computing no hay que decidir entre actualizar y conservar el trabajo, dado que esas personalizaciones e integraciones se conservan automáticamente durante la actualización.

- ✓ Contribuye al uso eficiente de la energía: En este caso, a la energía requerida para el funcionamiento de la infraestructura. En los datacenters[4] tradicionales, los servidores consumen mucha más energía de la requerida realmente. En cambio, en las nubes, la energía consumida es sólo la necesaria, reduciendo notablemente el desperdicio.

Los inconvenientes, en cambio:

- La centralización de las aplicaciones y el almacenamiento de los datos origina una interdependencia de los proveedores de servicios.
- La disponibilidad de las aplicaciones está sujeta a la disponibilidad de acceso a Internet.
- Los 'datos sensibles'[5] del negocio no residen en las instalaciones de las empresas, lo que podría generar un contexto de alta vulnerabilidad para la sustracción o robo de información.
- La confiabilidad de los servicios depende de la salud tecnológica y financiera de los proveedores de servicios en nube. Empresas emergentes o alianzas entre empresas podrían crear un ambiente propicio para el monopolio y el crecimiento exagerado en los servicios.
- La disponibilidad de servicios altamente especializados podría tardar meses o incluso años para que sean factibles de ser desplegados en la red.
- La madurez funcional de las aplicaciones hace que continuamente estén modificando sus interfaces, por lo cual la curva de aprendizaje en empresas de orientación no tecnológica tiene unas pendientes significativas, así como su consumo automático por aplicaciones.
- La información de la empresa debe recorrer diferentes nodos para llegar a su destino, cada uno de ellos (y sus canales) son un foco de inseguridad. Si se utilizan protocolos seguros, HTTPS por ejemplo, la velocidad total disminuye debido a la sobrecarga que éstos requieren.
- Escalabilidad a largo plazo. A medida que más usuarios empiecen a compartir la infraestructura de la nube, la sobrecarga en los servidores de los proveedores aumentará, si la empresa no posee un esquema de crecimiento óptimo puede llevar a degradaciones en el servicio o altos niveles de jitter[6].

También existen ciertas limitaciones en el cloud computing:

- Pérdidas de datos/fuga: Los esfuerzos para controlar la seguridad de los datos de la computación en la nube no son muy buenos; de acuerdo con el acceso de control API y la generación de las claves, almacenamiento y configuración de deficiencias, permiten resultados en pérdidas de datos, y también permiten una escasa política de destrucción de datos. La fuga es la causa de la escasa política de destrucción de datos.
- Dificultad de valorar la fiabilidad de los proveedores: El proveedor de servicio de computación en la nube, controla la fuerza con la que se pueden realizar los esfuerzos, que actualmente se solían usar para controlar los accesos a los datos, los cuáles son diferentes en muchos proveedores y en estas circunstancias; pero no todo es suficiente, las compañías necesitan una evaluación de los proveedores y proponer qué y cómo filtran el programa personal.
- Fuerza de los mecanismos de autentificación: En la nube, hay muchísimos datos, aplicaciones y recursos almacenados. La computación en la nube es muy débil en los mecanismos de autentificación, por lo tanto el atacante puede fácilmente obtener la cuenta de usuario cliente y acceder a la máquina virtual.

1.3.5. Tipos de servicios.

La computación en la nube tiene *tres modalidades* [3] para llevar a cabo los diferentes propósitos que se requieran (Véase más abajo la Ilustración 8):

- Software como servicio: El software como servicio (en inglés Software as a Service, SaaS) se encuentra en la capa más alta y caracteriza una aplicación completa ofrecida como un servicio, (por demanda, vía multitenencia[7]) que significa una sola instancia del software que corre en la infraestructura del proveedor y sirve a múltiples organizaciones de clientes. Las aplicaciones que suministran este modelo de servicio son accesibles a través de un navegador web -o de cualquier aplicación diseñada para tal efecto- y el usuario no tiene control sobre ellas, aunque en algunos casos se le permite realizar algunas configuraciones. Esto le elimina la necesidad al cliente de instalar la aplicación en sus propios computadores, evitando asumir los costos de soporte y el mantenimiento de hardware y software.

- Plataforma como servicio: La capa media, que es la plataforma como servicio (en inglés Platform as a Service, PaaS), es la encapsulación de una abstracción de un ambiente de desarrollo y el empaquetamiento de una serie de módulos o complementos que proporcionan, normalmente, una funcionalidad horizontal (persistencia de datos, autenticación, mensajería, etc.). De esta forma, un arquetipo de plataforma como servicio podría consistir en un entorno conteniendo una pila básica de sistemas, componentes o APIs preconfiguradas y listas para integrarse sobre una tecnología concreta de desarrollo (por ejemplo, un sistema Linux, un servidor web, y un ambiente de programación como Perl o Ruby). Las ofertas de PaaS pueden dar servicio a todas las fases del ciclo de desarrollo y pruebas del software, o pueden estar especializadas en cualquier área en particular, tal como la administración del contenido. Ejemplos comerciales son Google App Engine, que sirve aplicaciones de la infraestructura Google; Microsoft Azure, una plataforma en la nube que permite el desarrollo y ejecución de aplicaciones codificadas en varios lenguajes y tecnologías como .NET, Java y PHP o la Plataforma G, desarrollada en Perl. Servicios PaaS como éstos permiten gran flexibilidad, pero puede ser restringida por las capacidades disponibles a través del proveedor. En este modelo de servicio al usuario se le ofrece la plataforma de desarrollo y las herramientas de programación por lo que puede desarrollar aplicaciones propias y controlar la aplicación, pero no controla la infraestructura.

- Infraestructura como servicio: La infraestructura como servicio (Infrastructure as a Service, IaaS) -también llamada en algunos casos Hardware as a Service, HaaS- se encuentra en la capa inferior y es un medio de entregar almacenamiento básico y capacidades de cómputo como servicios estandarizados en la red. Servidores, sistemas de almacenamiento, conexiones, enrutadores, y otros sistemas se concentran (por ejemplo a través de la tecnología de virtualización) para manejar tipos específicos de cargas de trabajo, desde procesamiento en lotes (“batch”) hasta aumento de servidor/almacenamiento durante los picos de carga. El ejemplo comercial mejor conocido es Amazon Web Services, cuyos servicios EC2 y S3 ofrecen cómputo y servicios de almacenamiento esenciales (respectivamente). Otro ejemplo es Joyent, cuyo producto principal es una línea de servidores virtualizados, que proveen una infraestructura en demanda altamente escalable[8] para manejar sitios web, incluidas aplicaciones web complejas escritas en Python, Ruby, PHP y Java.

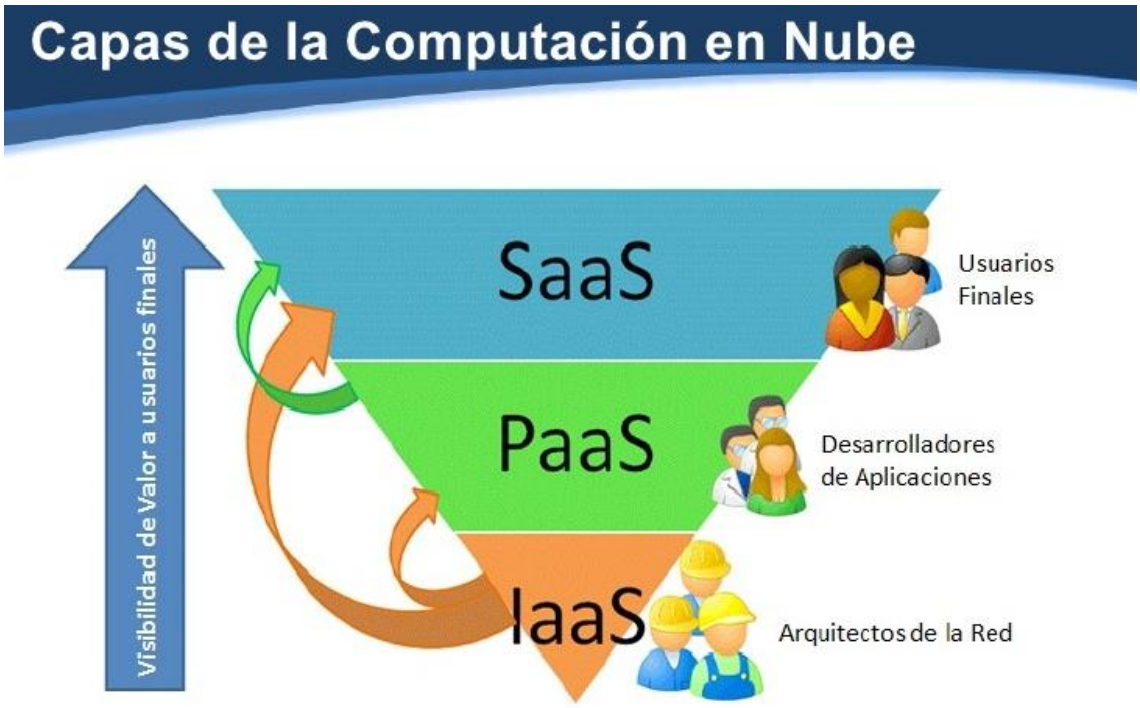


Ilustración 8. Organigrama de los tipos de servicio y tipos de usuarios en la nube.

1.3.6. Tipos de nubes

- Nube pública: es una nube computacional mantenida y gestionada por terceras personas no vinculadas con la organización. En este tipo de nubes tanto los datos como los procesos de varios clientes se mezclan en los servidores, sistemas de almacenamiento y otras infraestructuras de la nube. Los usuarios finales[9] de la nube no conocen qué trabajos de otros clientes pueden estar corriendo en el mismo servidor, red, sistemas de almacenamiento, etc. Aplicaciones, almacenamiento y otros recursos están disponibles al público a través del proveedor de servicios, que es propietario de toda la infraestructura en sus centros de datos; el acceso a los servicios sólo se ofrece de manera remota, normalmente a través de internet.

- Nube privada: es una buena opción para las compañías que necesitan alta protección de datos y ediciones a nivel de servicio. Las nubes privadas están en una infraestructura bajo demanda, gestionada para un solo cliente que controla qué aplicaciones debe ejecutarse y dónde. Son propietarios del servidor, red, y disco y pueden decidir qué usuarios están autorizados a utilizar la infraestructura. Al administrar internamente estos servicios, las empresas tienen la ventaja de mantener la privacidad de su información y permitir unificar el acceso a las aplicaciones corporativas de sus usuarios.

- Nube híbrida: combina los modelos de nubes públicas y privadas. Un usuario es propietario de unas partes y comparte otras, aunque de una manera controlada. Las nubes híbridas ofrecen la promesa del escalado, aprovisionada externamente a demanda, pero añaden la complejidad de determinar cómo distribuir las aplicaciones a través de estos ambientes diferentes. Las empresas pueden sentir cierta atracción por la promesa de una nube híbrida, pero esta opción, al menos inicialmente, estará probablemente reservada a aplicaciones simples sin condicionantes, que no requieran de ninguna sincronización o necesiten bases de datos complejas. Se unen mediante la tecnología, pues permiten enviar datos o aplicaciones entre ellas. Un ejemplo son los sistemas de correo electrónico empresarial.

- Nube comunitaria**:** De acuerdo con *Joyanes Aguilar en 2012* [4], el Instituto Nacional de Estándares y Tecnología (NITS, por sus siglas en inglés) define este modelo como aquel que se organiza con la finalidad de servir a una función o propósito común (seguridad, política…), las cuales son administradas por las organizaciones constituyentes o terceras partes.

1.3.7. Seguridad en la nube

De acuerdo al artículo de *computación en la nube* [3], la seguridad en la computación en la nube puede ser tan buena o mejor que la que existía en los sistemas tradicionales, porque los proveedores son capaces de proporcionar recursos que resuelvan problemas de seguridad que muchos clientes no pueden afrontar. Sin embargo, la seguridad todavía sigue siendo un asunto importante, cuando los datos tienen un matiz confidencial. Esto atrasa la adopción de la computación en la nube hasta cierto punto.

Seguridad como servicio

En el entorno de la nube, la seguridad es provista por los proveedores. Se pueden distinguir dos métodos: el primer método es que cualquiera puede cambiar sus métodos de entrega incluidos en los servicios de la nube. El segundo método es que los proveedores de servicio de la nube proveen seguridad solo como servicio en la nube, con información de seguridad de las compañías.

Seguridad del explorador

En el entorno de la nube, los servidores remotos son usados para la computación. Los nodos del cliente se usan solo para entrada/salida de operaciones, y para la autorización y autenticación de la información en la nube. Un navegador web estándar es una plataforma normalmente utilizada para todos los usuarios del mundo. Esto puede ser catalogado en dos tipos diferentes: Software como servicio (SaaS), Aplicaciones Web, o Web 2.0. Transport Layer Security (TLS), se suele emplear para la encriptación de datos y la autentificación del host.

Autenticación

En el entorno de la nube, la base para el control de acceso es la autenticación, el control de acceso es más importante que nunca desde que la nube y todos sus datos son accesibles para todo el mundo a través de internet. Trusted Platform Module (TPM) es extensamente utilizado y un sistema de autenticación más fuerte que el nombre de usuario y la contraseña. Trusted Computing Groups (TCG's) es un estándar sobre la autorización de usuarios y otras herramientas de seguridad de comunicación en tiempo real entre el proveedor y el cliente.

Pérdida de gobernanza

En las infraestructuras de la nube, el cliente necesariamente cede el control al proveedor (cloud provider) en varios asuntos, los cuales influyen negativamente sobre la seguridad. Al mismo tiempo, el acuerdo de nivel de servicio no suele tener el cometido de surtir este tipo de servicios en la parte del proveedor de la nube, lo que deja una brecha en las defensas de seguridad.

Lock-In

Esta es una pequeña oferta en este tipo de herramientas, los procedimientos o estándares de formatos de datos o interfaces de servicios que podrían garantizar los datos, las aplicaciones y el servicio de portabilidad. Esto puede hacer difícil para el cliente migrar de un proveedor a otro, o migrar los datos y servicios de nuevo a otro entorno informático. Esto introduce una particular dependencia en el proveedor de la nube para la provisión del servicio, especialmente a la portabilidad de los datos, el aspecto más fundamental.

Protección de los datos

La computación en la nube pone en riesgo la protección de datos para los usuarios de la nube y sus proveedores. En muchos casos, ocasiona dificultades para el proveedor (en el rol de controlador de la información) para asegurar la efectividad práctica del manejo de los datos del proveedor de la nube y para supervisar que los datos van por el camino correcto. Este problema se suele agravar en casos de múltiples transferencias de datos, por ejemplo entre sistemas federados[10]. Por otra parte, algunos proveedores de la nube, proporcionan información de sus prácticas de cercenamiento de datos. También hay algunas ofertas de certificaciones en el procesamiento de datos, las actividades de seguridad, y los controles de datos que tienen lugar, como por ejemplo la certificación SAS70. Las corrientes de datos de Internet están unidas al malware y de paquetes trampa para meter al usuario en una desconocida participación en actividades delictivas.

1.3.8. OpenStack

Cuando hablamos de computación en la nube, es inevitable pensar en OpenStack (Véase Ilustración 9) como una de las soluciones cloud más populares.

OpenStack [5] es un software de computación en la nube pensado para proporcionar una infraestructura como servicio (IaaS), como veíamos anteriormente en los tipos de servicios que ofrecía el cloud computing.

Ilustración 9. Logo de OpenStack

Su misión es proveer una solución flexible tanto para nubes públicas como privadas, sean estas de cualquier tamaño, y para esto se consideran dos requerimientos básicos: las nubes deben ser simples de implementar y masivamente escalables.

Para cumplir con estos principios, OpenStack está dividido en diferentes componentes que trabajan en conjunto. Esta integración es lograda a través de interfaces de programación de aplicaciones (APIs) que cada servicio ofrece y consume.

Gracias a estas APIs, los servicios pueden comunicarse entre ellos y además se posibilita que un servicio sea reemplazado por otro de similares características siempre que se respete la forma de comunicación. OpenStack es extensible y se ajusta a las necesidades de quien desee implementarlo.

1.4. Introducción a la virtualización. Diferentes arquitecturas.

De acuerdo con este artículo sobre la *virtualización* [6], se define como la creación a través de software de una versión virtual de algún recurso tecnológico, como puede ser una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento u otros recursos de red. En los ámbitos de habla inglesa, este término se suele conocer por el numerónimo[11] "v12n". Dicho de otra manera, se refiere a la abstracción de los recursos de una computadora, llamada Hypervisor o VMM (Virtual Machine Monitor) que crea una capa de abstracción entre el hardware de la máquina física (host) y el sistema operativo de la máquina virtual (virtual machine, guest), dividiéndose el recurso en uno o más entornos de ejecución.

Esta capa de software (VMM) maneja, gestiona y dirige los cuatro recursos principales de una computadora (CPU, Memoria, Dispositivos Periféricos y Conexiones de Red) y así podrá repartir dinámicamente dichos recursos entre todas las máquinas virtuales definidas en el computador central. Esto hace que se puedan tener varios ordenadores virtuales ejecutándose en el mismo ordenador físico.

Históricamente, tal término se viene usando desde 1960, y ha sido aplicado a diferentes aspectos y ámbitos de la informática, desde sistemas computacionales completos, hasta capacidades o componentes individuales.

La virtualización se encarga de crear una interfaz externa que encapsula una implementación subyacente mediante la combinación de recursos en localizaciones físicas diferentes, o por medio de la simplificación del sistema de control. Un avanzado desarrollo de nuevas plataformas y tecnologías de virtualización ha hecho que en los últimos años se haya vuelto a prestar atención a este concepto. La máquina virtual en general simula una plataforma de hardware autónoma incluyendo un sistema operativo completo que se ejecuta como si estuviera instalado. Normalmente varias máquinas virtuales operan en un computador central. Para que el sistema operativo "guest" funcione, la simulación debe ser lo suficientemente grande (siempre dependiendo del tipo de virtualización).

Existen diferentes formas de virtualización: es posible virtualizar el hardware de servidor, el software de servidor, sesiones de usuario, aplicaciones y también se pueden crear máquinas virtuales en una computadora de escritorio. Entre los principales proveedores de software que han desarrollado tecnologías de virtualización integrales (que abarcan todas las instancias: servidor, aplicaciones, escritorio) se encuentran, por ejemplo Citrix, VMware y Microsoft. Estas compañías han diseñado soluciones específicas para virtualización, como XenServer, VMware ESX y Windows Server 2008 Hyper-V para la virtualización de servidores. Si bien la virtualización no es un invento reciente, con la consolidación del modelo de la computación en la nube, la virtualización ha pasado a ser uno de los componentes fundamentales, especialmente en lo que se denomina infraestructura de nube privada.

Virtualización de plataforma

La virtualización de plataforma se lleva a cabo en una plataforma de hardware mediante un software host, que es un programa de control que simula un entorno computacional (máquina virtual) para su software "guest". Este software "guest", que normalmente es un sistema operativo completo, se ejecuta como si estuviera instalado en una plataforma de hardware autónoma. Usualmente muchas máquinas virtuales son simuladas en una máquina física dada. Para que el sistema operativo "guest" funcione, la simulación debe ser lo suficientemente grande como para soportar todas las interfaces externas de los sistemas huéspedes, las cuales pueden incluir (dependiendo del tipo de virtualización) los drivers de hardware.

Tipos de virtualización de plataforma:

- Virtualización completa: Esta es en donde la máquina virtual simula un hardware suficiente para permitir un sistema operativo "huésped" sin modificar (uno diseñado para la misma CPU) para ejecutar de forma aislada. Típicamente, muchas instancias pueden ejecutarse al mismo tiempo. Este enfoque fue el pionero en 1966 con CP-40 y CP[-67]/CMS, predecesores de la familia de máquinas virtuales de IBM. Ejemplos: VMWare Workstation, Oracle VM Virtualbox, KVM, etc.

- Virtualización parcial (Address Space Virtualization): La máquina virtual simula múltiples instancias de gran parte (pero no de todo) del entorno subyacente del hardware, particularmente los espacios de direcciones. Tal entorno acepta compartir recursos y alojar procesos, pero no permite instancias separadas de sistemas operativos "huésped". Aunque no es vista como dentro de la categoría de máquina virtual, históricamente éste fue un importante acercamiento, y lo usaron en sistemas como CTSS, el experimental IBM M44/44X, y podría mencionarse que en sistemas como OS/VS1, OS/VS2 y MVS.

- Virtualización por S.O. o semi-parcial**:** Virtualizar significa instalar un sistema operativo dentro de otro al que se le llama anfitrión (host), mediante el uso de una máquina virtual. Frecuentemente denominada virtualización compartida del Sistema Operativo o virtualización del SO, la virtualización del Sistema Operativo virtualiza servidores en la capa del sistema operativo (kernel). Este método de virtualización crea particiones aisladas o entornos virtuales (VEs) en un único servidor físico e instancia de SO para así maximizar los esfuerzos de administración del hardware, software y centro de datos. La virtualización de Hypervisor tiene una capa base (generalmente un kernel Linux que se muestra aquí como un hypervisor o SO estándar, lo mismo que Windows Server 2008 R2 Hyper-V) que se carga directamente en el servidor base. Para asignar hardware y recursos a las máquinas virtuales (VMs), es recomendable que todo el hardware del servidor esté virtualizado. La siguiente capa superior muestra cada chip, placa, etc. que debe virtualizarse para que así pueda ser asignado a las VMs. Una vez en la VM, hay una copia completa de un sistema operativo y finalmente la aplicación o carga de trabajo.

La Virtualización de un S.O. mejora el rendimiento, gestión y eficiencia. En la base reside un sistema operativo anfitrión estándar, como en el caso de Parallels Virtuozzo que incluye Windows y un sistema con núcleo Linux. A continuación encontramos la capa de virtualización, con un sistema de archivos propietario y una capa de abstracción de servicio de kernel que garantiza el aislamiento y seguridad de los recursos entre distintos contenedores. La capa de virtualización hace que cada uno de los contenedores aparezca como servidor autónomo. Finalmente, el contenedor aloja la aplicación o carga de trabajo.

A continuación enumeramos las ventajas de la virtualización:

- ✓ Reutilización de hardware existente (para utilizar software más moderno) y optimizar el aprovechamiento de todos los recursos de hardware.
- ✓ Rápida incorporación de nuevos recursos para los servidores virtualizados.
- ✓ Reducción de los costes de espacio y consumo necesario de forma proporcional al índice de consolidación[12] logrado (Estimación media 10:1).
- ✓ Administración global centralizada y simplificada.
- ✓ Nos permite gestionar nuestro CPD[13] como un pool de recursos o agrupación de toda la capacidad de procesamiento, memoria, red y almacenamiento disponible en nuestra infraestructura.
- ✓ Mejora en los procesos de clonación y copia de sistemas: Mayor facilidad para la creación de entornos de test que permiten poner en marcha nuevas aplicaciones sin impactar a la producción, agilizando el proceso de las pruebas.
- ✓ Aislamiento: un fallo general de sistema de una máquina virtual no afecta al resto de máquinas virtuales.
- ✓ Mejora de TCO[14] y ROI[15].
- ✓ No sólo aporta el beneficio directo en la reducción del hardware necesario, sino también los costes asociados.
- ✓ Reduce los tiempos de parada.
- ✓ Migración en caliente de máquinas virtuales (sin pérdida de servicio) de un servidor físico a otro, eliminando la necesidad de paradas planificadas por mantenimiento de los servidores físicos.
- ✓ Balanceo dinámico de máquinas virtuales entre los servidores físicos que componen el pool de recursos, garantizando que cada máquina virtual se ejecute en el servidor físico más adecuado y proporcionando un consumo de recursos homogéneo y óptimo en toda la infraestructura.
- ✓ Contribución al medio ambiente -Green IT[16]- por menor consumo de energía en servidores físicos.
- ✓ Alta disponibilidad.

La virtualización se puede hacer desde un sistema operativo Windows (ya sea XP, Vista u otra versión que sea compatible con el programa que utilicemos), en el que virtualizamos otro sistema operativo como Linux o viceversa, o que tengamos instalado Linux y queramos virtualizar una versión de Windows.

Existen varios tipos de virtualización, aparte de la virtualización por plataforma:

- Virtualización asistida por hardware**:** Virtualización asistida por hardware son extensiones introducidas en la arquitectura de procesador x86 para facilitar las tareas de virtualización al software ejecutándose sobre el sistema. Si cuatro son los niveles de privilegio o anillos de ejecución en esta arquitectura, desde el cero o de mayor privilegio, que se destina a las operaciones del kernel de SO, al tres, con privilegios menores que es el utilizado por los procesos de usuario, en esta nueva arquitectura se introduce un anillo interior o ring que será el que un hypervisor o Virtual Machine Monitor usará para aislar todas las capas superiores de software de las operaciones de virtualización.

- La virtualización de almacenamiento**:** Se refiere al proceso de abstraer el almacenamiento lógico del almacenamiento físico, y es comúnmente usado en SANs ("Storage Area Network" Red de área de almacenamiento). Los recursos de almacenamiento físicos son agregados al "storage pool" (almacén de almacenamiento), del cual es creado el almacenamiento lógico.

- Particionamiento**:** Es la división de un solo recurso (casi siempre grande), como el espacio de disco o el ancho de banda de la red, en un número más pequeño y con recursos del mismo tipo que son más fáciles de utilizar. Esto es muchas veces llamado “zoning”, especialmente en almacenamiento de red.

- Máquina virtual**:** La entenderemos básicamente como un sistema de virtualización, denominado "virtualización de servidores", que dependiendo de la función que esta deba de desempeñar en la organización, todas ellas dependen del hardware y dispositivos físicos, pero casi siempre trabajan como modelos totalmente independientes de este. Cada una de ellas con sus propias CPUs virtuales, tarjetas de red, discos etc. Lo cual podría especificarse como una compartición de recursos locales físicos entre varios dispositivos virtuales.

- Hypervisor de almacenamiento: Es un pack portátil de gestión centralizada, utilizado para mejorar el valor combinado de los sistemas de disco de almacenamiento múltiples, incluyendo los modelos diferentes e incompatibles, complementando sus capacidades individuales con el aprovisionamiento extendido, la réplica y la aceleración del rendimiento del servicio. Su completo conjunto de funciones de control y monitorización del almacenamiento, operan como una capa virtual trasparente entre los pools de disco consolidados para mejorar su disponibilidad, velocidad y utilización.

1.4.1. KVM

Kernel-based Virtual Machine o KVM [7], (en español, Máquina virtual basada en el núcleo) es una solución para implementar virtualización completa con Linux. Está formada por un módulo del núcleo (con el nombre kvm.ko) y herramientas en el espacio de usuario, siendo en su totalidad software libre. El componente KVM para el núcleo está incluido en Linux desde la versión 2.6.20.

KVM permite ejecutar máquinas virtuales utilizando imágenes de disco que contienen sistemas operativos sin modificar. Cada máquina virtual tiene su propio hardware virtualizado: una tarjeta de red, discos duros, tarjeta gráfica, etc.

Los requisitos son: procesador x86, o x86_64, con soporte para virtualización. La tecnología de virtualización recibe la denominación VT en Intel y SVM en AMD. KVM puede ejecutar instancias Linux/Unix/Windows de 32 o 64 bits

1.4.2. QEMU

QEMU [8] es un emulador de procesadores basado en la traducción dinámica de binarios (conversión del código binario de la arquitectura fuente en código entendible por la arquitectura huésped). QEMU también tiene capacidades de virtualización dentro de un sistema operativo, ya sea GNU/Linux, Windows, o cualquiera de los sistemas operativos admitidos; de hecho es la forma más común de uso. Esta máquina virtual puede ejecutarse en cualquier tipo de Microprocesador o arquitectura (x86, x86-64, PowerPC, MIPS, SPARC, etc.). Está licenciado en parte con la LGPL y la GPL de GNU.

El objetivo principal es emular un sistema operativo dentro de otro sin tener que reparticionar el disco duro, empleando para su ubicación cualquier directorio dentro de éste. El programa no dispone de GUI, pero existe otro programa llamado QEMU manager que puede hacer de interfaz gráfica si se utiliza QEMU desde Windows. También existe una versión para GNU/Linux llamada qemu-launcher. En Mac OS X puede utilizarse el programa Q que dispone de una interfaz gráfica para crear y administrar las máquinas virtuales. Existe también una variante que permite emular la computadora japonesa NEC PC-9801 hecha por Takeda Yoshida. QEMU posee dos modos de operación:

- Emulación del modo usuario**:** Puede ejecutar programas compilados para un tipo de CPU en otro tipo de CPU. Las llamadas al sistema son pensadas para endianness y desarreglos en 32/64 bits. Wine y Dosemu son los principales objetivos de QEMU.

- Modo de emulación completo de sistema de ordenador**:** QEMU emula un sistema informático completo, incluyendo procesador y varios periféricos. Este puede ser usado para proveer hosting virtual a varios ordenadores virtuales en un único ordenador. QEMU puede arrancar varios sistemas operativos, incluyendo entre otros Linux, Microsoft Windows, DOS, y BSD. Admite además la emulación de varias plataformas de hardware, incluyendo x86, AMD64, Alpha, MIPS, y Sparc. La mayoría del programa está bajo licencia LGPL, y el modo de emulación de usuario tiene licencia GPL. La versión para Windows utiliza la capa de sonido FMOD, que es un programa comercial.

2. INTERESES Y OBJETIVOS PARA LA REALIZACIÓN DE ESTE PROYECTO.

Este proyecto surgió con la idea de aprovechar equipamiento informático alojado en el centro mixto CIESOL, de cara a utilizarlo como simulador de aplicaciones y sistemas utilizados en la Plataforma Solar de Almería. Ésta dispone de numerosos sistemas como el campo ACUREX, hornos solares, desaladora solar, etc.

Como principal objetivo, este proyecto quiere explotar el máximo rendimiento de la arquitectura del clúster (equipamiento informático) y servirse de la óptima configuración realizada para albergar el software necesario, como el diseño SCADA, los autómatas, sensores y actuadores de un determinado sistema, entre otros, y así poder simular correctamente cualquier sistema de la PSA.

Así mismo, y una vez diseñado el sistema informático para proceder con la simulación, se desea mantener activo dicho equipamiento para proyectos futuros, como la simulación de nuevos sistemas de la PSA.

3. METODOLOGÍA DEL TRABAJO Y MATERIAL.

Se ha diseñado una metodología y una serie de pasos para estructurar la realización del trabajo y alcanzar los objetivos del mismo.

En primer lugar, trabajaremos con un clúster de ordenadores, compuesto de 11 nodos. Cada nodo, en términos de hardware, posee una memoria RAM de 2 GB, una capacidad HDD de 80 GB, y un procesador AMD Dual Core. Cada nodo dispone de 2 tarjetas de red, a excepción del nodo de control, que tendrá 3 tarjetas, y el nodo de red, que tendrá 4.

Sobre este clúster, alojado en una habitación contigua a uno de los laboratorios del CIESOL, (dicha habitación provista con ventilación y refrigeración) es donde realizaremos el trabajo. Para ello, se han seguido los pasos que se han diseñado anteriormente.

3.1. Documentación sobre OpenStack y servicios

En este paso se debe conocer el software requerido para implantar la arquitectura con la que realizaremos el trabajo. Para ello, utilizaremos el software cloud OpenStack, que cubrirá las necesidades del trabajo.

Ya se ha hablado en la introducción de OpenStack. Ahora se profundizará más sobre los servicios y APIs que ofrece, y que debemos implantar para llevar a cabo nuestros objetivos, siguiendo como referencia la *documentación sobre OpenStack* [9].

Hay varios tipos de servicios:

- **Core Services (servicios básicos)**

  - Nova (compute): Es el servicio que gestiona y automatiza los recursos del equipo, y que trabaja con las tecnologías disponibles para la virtualización, que serían KVM, Xen, QEMU, y además la tecnología Hyper-V, la tecnología Sphere de VMWare, y la tecnología de contenedores Linux como LXC. Como función principal, se encarga del ciclo de vida de las instancias de cómputo en el entorno OpenStack, y es responsable de generar dichas instancias, además de organizarlas y ajustarlas según los nodos de cómputo que existan.
    Está escrito en Python y usa muchas bibliotecas externas, como Eventlet (programación concurrente), Kombu (para la comunicación AMQP) y SQLAlchemy (para acceder a la base de datos). La arquitectura de Nova está diseñado para escalar horizontalmente en hardware estándar, sin requisitos hardware y software propietarios, y proporcionar la capacidad de integración con sistemas legados y tecnologías de terceros.

- Keystone (identity): Es el servicio que ofrece autenticación a los usuarios, como indica su nombre. Ofrece un directorio central de usuarios asignados a los servicios de OpenStack que pueden acceder. Actúa como un sistema de autenticación común en todo el sistema operativo para la nube y se puede integrar con los servicios de directorio backend como LDAP. Es compatible con múltiples formas de autenticación, incluyendo nombre de usuario y contraseña de credenciales estándar, sistemas basados en tokens e inicios de sesión de estilo AWS (Amazon Web Services).
- Glance (Image): Es el servicio que proporciona servicios de descubrimiento, de inscripción y de entrega de los discos y del servidor de las imágenes. Las imágenes almacenadas se pueden utilizar como plantilla. También se puede utilizar para almacenar y catalogar un número ilimitado de copias de seguridad. En este caso, será un simple servicio que guardará imágenes de sistemas operativos en forma de plantilla. La API de servicios de imagen proporciona una interfaz REST estándar para consultar información sobre las imágenes de disco y permite a los clientes transmitir las imágenes a nuevos servidores.
- Neutron (networking): También llamado "OpenStack Networking", es el servicio que gestiona las redes y las direcciones IP en OpenStack. Como principal característica, asegura que la red no tenga el problema del cuello de botella o el factor limitante en un despliegue en la nube y ofrece a los usuarios un autoservicio real a través de sus configuraciones de red. Proporciona conectividad con otros servicios de OpenStack. Además, proporciona modelos de redes para diferentes aplicaciones o grupos de usuarios. Los modelos estándar incluyen redes planas o VLAN para la separación de los servidores y el tráfico. Gestiona las direcciones IP, lo que permite direcciones IP estáticas o DHCP reservados. Puede proporcionar también direcciones IP flotantes, que permiten que el tráfico se redirija dinámicamente a cualquiera de sus recursos informáticos, que permite redirigir el tráfico durante el mantenimiento o en caso de fracaso. Los usuarios pueden crear sus propias redes, controlar el tráfico y conectar los servidores y los dispositivos a una o más redes. Los administradores pueden aprovechar las redes definidas por software de tecnología (SDN) como OpenFlow para permitir altos niveles de multiempresa y escala masiva. Neutron tiene un marco que permite la extensión de servicios de red adicionales, como los sistemas de detección de intrusos (IDS), balanceo de carga, cortafuegos y redes privadas virtuales (VPN) para ser implementada y administrada.

- **Optional Services (servicios opcionales)**

  - Horizon (Dashboard): Es el servicio que proporciona interfaz gráfica en un navegador web. Proporciona a los administradores y usuarios una interfaz gráfica para el acceso, provisión y automatización de los recursos basados en la nube. El diseño permite la ejecución de productos y servicios de terceros, tales como la facturación, monitoreo y herramientas de gestión adicionales. Entre sus principales funciones (que se verán más adelante con detalle) destacan el lanzamiento de instancias, asignación de direcciones IP y configuración de accesos de control.
  - Swift (Object Storage): Es el servicio que ofrece un sistema de almacenamiento redundante y escalable. Los objetos y archivos se guardan en varias unidades de disco repartidas por los servidores del centro de datos, con el software OpenStack responsable de asegurar la replicación e integridad de los datos en el clúster. En caso de que un servidor o disco duro falle, OpenStack replica su contenido desde otros nodos activos a nuevas ubicaciones en el clúster. Debido a que OpenStack utiliza la lógica del software para asegurar la replicación de datos y la distribución a través de diferentes dispositivos, se pueden utilizar discos duros y servidores de bajo coste.
  - Cinder (Block Storage): Es el servicio que proporciona dispositivos de almacenamiento a nivel de bloque persistentes para usar con instancias de cómputo. El sistema de almacenamiento de bloques gestiona la creación, aplicación y el desprendimiento de los dispositivos de bloque a los servidores. Volúmenes de almacenamiento de bloque se integran plenamente en el servicio Compute y la interfaz gráfica de Dashboard permite a los usuarios en la nube gestionar sus propias necesidades de almacenamiento. Además del almacenamiento del servidor local de Linux, se pueden utilizar distintas plataformas de almacenamiento como Ceph, CloudByte, Coraid, EMC, GlusterFS, Hitachi Data Systems, IBM Storage, etc. El almacenamiento de bloques es apropiado para escenarios donde el rendimiento es sensible, como por ejemplo el almacenamiento de bases de datos, sistemas de archivos expandibles, o la prestación de un servidor con acceso al almacenamiento a nivel de bloque en bruto.
  - Ceilometer (Telemetry): Es el servicio que se encarga de monitorizar y medir variables del entorno de OpenStack, en términos de facturación, benchmarking, escalabilidad y propuestas estadísticas. Proporciona un único punto de contacto para los sistemas de facturación, proporcionando todas las mediciones que se necesitan para establecer la facturación del cliente a través de todos los componentes actuales y futuras de OpenStack.
    La entrega de los datos es trazable y auditable, dichos datos deben de ser fácilmente extensibles para apoyar nuevos proyectos, y los agentes que realizan las recolecciones de datos deben ser independientes de todo el sistema.

- Heat (Orchestration): Es el servicio que orquesta distintas configuraciones que deseemos lanzar en OpenStack. Se pueden configurar múltiples aplicaciones compuestas en la nube utilizando plantillas, a través de una API REST OpenStack nativa y una API de consultas compatibles con CloudFormation. Es útil para realizar configuraciones de manera masiva, como por ejemplo, lanzar una configuración que consista en lanzar 20 instancias virtuales con unas determinadas especificaciones hardware, determinada configuración de red, etc.
- Trove (Database): Es el servicio que ofrece una base de datos escalable que funciona como un servicio de aprovisionamiento de motores de bases de datos relacionales y no relacionales. Inicialmente, el servicio se concentra en proveer recursos de aislamiento a alto rendimiento mientras se automatizan tareas administrativas complejas como: lanzamiento, configuración, parcheo, copias de seguridad, restauración y monitorización.

3.2. Instalación del sistema operativo en los nodos de la infraestructura física.

Para una correcta instalación de la plataforma, todos los nodos se encuentran formateados y limpios, por lo que hay que empezar instalando un sistema operativo que nos permita posteriormente instalar el software OpenStack necesario.

Por razones de estabilidad y compatibilidad, y tratando de elegir la versión más novedosa posible, se opta por instalar Ubuntu Server versión 14.04 LTS en todos los nodos.

Durante el proceso de instalación, diferenciaremos los nodos mediante sus nombres de máquina, que será “cieclusterXX”. Así, el primer nodo sería ciecluster01.

Una vez instalado el sistema operativo en todos los nodos, debemos proceder a configurar la topología de red que implementaremos para proseguir con la instalación.

3.3. Configuración de la infraestructura de red. IPtables.

Una vez que ya está instalado el sistema operativo, se debe configurar una topología de red para poder instalar el software OpenStack y tener actualizados todos los paquetes y el sistema operativo.

La topología consta de 10 ordenadores conectados en red local a través de un switch, de los cuales el nodo de control y el nodo de red estarán conectados a otro switch, y éste a su vez estará conectado a otro ordenador auxiliar que hará las funciones de router y proporcionará acceso a Internet. Gráficamente sería algo parecido a esto:

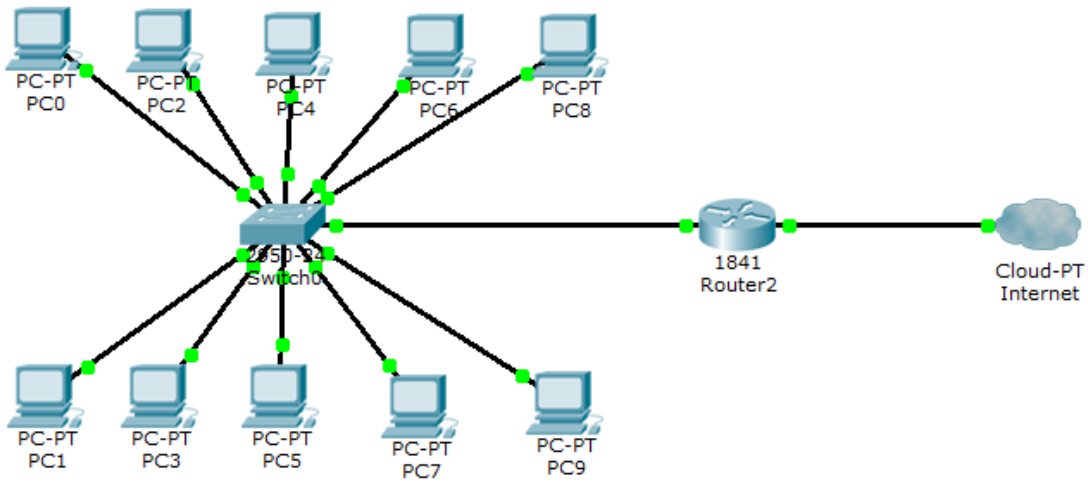


Ilustración 10. Topología de red

Para la red local, se utilizará la red 10.0.0.0/24 para conectar los ordenadores entre sí. Para la red externa, se utilizará la red 192.168.137.0/24 para dar conexión a Internet.

Debemos configurar las tarjetas de red de cada nodo, empezando por el nodo de control con la dirección 10.0.0.2. Hay que abrir el archivo de configuración utilizando un editor de textos como el nano, con el comando:

```
$ sudo nano /etc/network/interfaces
```

Una vez abierto el archivo, realizamos la siguiente configuración, en este caso en el nodo de control:

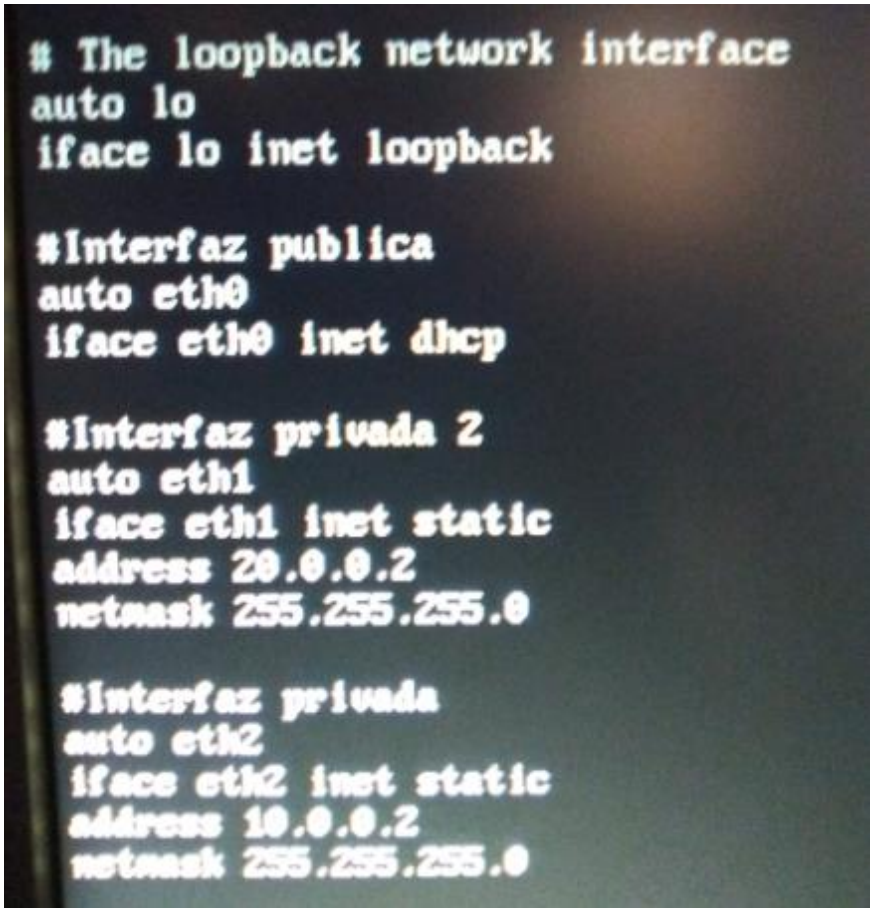


Ilustración 11. Contenido de /etc/network/interfaces

La dirección IP para la red externa debemos proporcionarla mediante DHCP. Guardamos los cambios y reiniciamos el servicio de red con el comando:

```
$ sudo /etc/init.d/networking restart
```

Ahora las interfaces de red en cuestión deben tener las nuevas IPs:

Ilustración 12. Resultado de ifconfig

Comprobamos que tenemos acceso a Internet con un update:

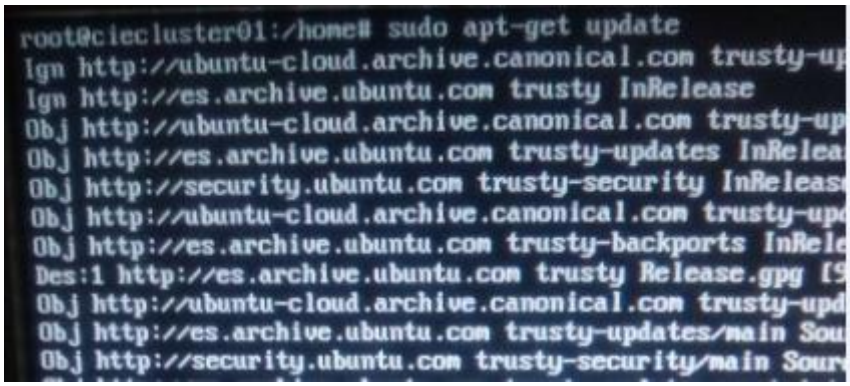


Ilustración 13. Conexión exitosa a Internet

Ahora, el nodo de control dispone de conexión a Internet y se encuentra en la red local 10.0.0.0/24. Para el nodo de red realizamos exactamente el mismo proceso, pero con una IP de la red local distinta, en este caso la 10.0.0.8, y una IP de la red externa distinta. Para el resto de nodos, repetimos el proceso asignándoles IPs de la red local.

Una vez que tenemos la red montada probamos la conectividad entre los nodos:

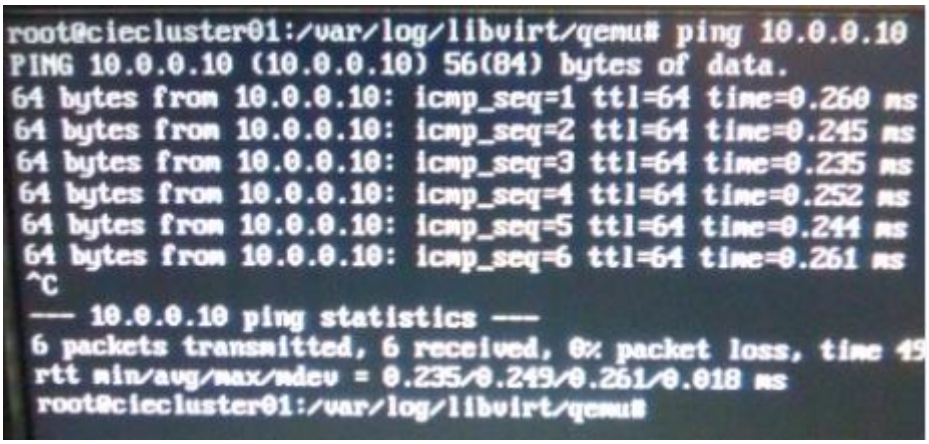


Ilustración 14. Conectividad entre nodos

Ahora tenemos el siguiente problema. Tenemos conexión a Internet en el nodo de control y en el nodo de red, pero no en el resto de nodos, que llamaremos a partir de ahora nodos de cómputo. Sin conexión a Internet en esos nodos, no podemos instalar el software OpenStack necesario para dichos nodos, por lo que debemos proporcionarles conexión a Internet a partir del nodo de control o de red. En este caso, elegimos el nodo de control, ya que es indiferente desde cuál brindar conexión a Internet. Para hacer esto, es necesario utilizar las reglas de Iptables.

Las reglas que utilizaremos de Iptables serán sencillas, pero permitirán que el resto de nodos dispongan de conexión al exterior.

Ilustración 15. Reglas de IPtables

Lo que hemos hecho en Iptables es habilitar el bit de forwarding para que nuestro nodo no ignore los paquetes que no vayan destinados a sí mismo, y activar NAT para que los equipos conectados a este nodo puedan salir a Internet mediante la IP de dicho nodo.

Aplicamos el script y probamos que tenemos conexión a Internet en todos los nodos de cómputo:

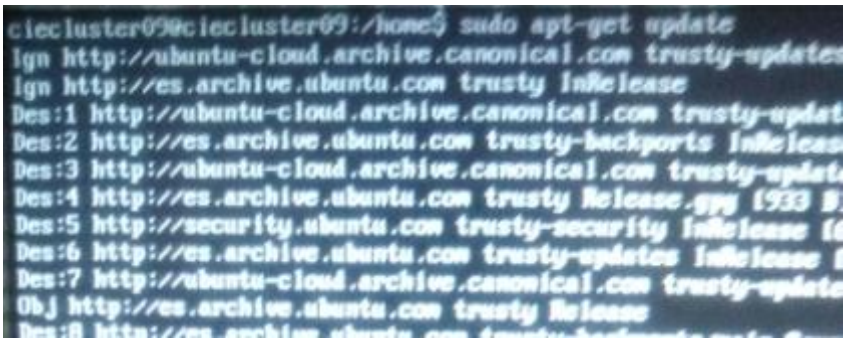


Ilustración 16. Conexión a Internet en nodo de cómputo

Una vez configurada la topología de red, con los nodos teniendo conectividad entre sí, y con conexión a Internet, ya podemos proceder a instalar el software OpenStack.

3.4. Instalación de servicios OpenStack y verificación de su funcionamiento.

Como se dijo anteriormente, la versión de OpenStack que se utilizará será la versión Kilo, liberada en Abril de 2015. Repasando un poco, sabemos que OpenStack se compone de varios servicios que lo hacen funcionar, como son: Nova, Compute, Neutron, Horizon, Keystone, Glance…etc. Todos ellos se organizan de la siguiente manera (Ver ANEXO I, pág. 150).

Una de las características destacadas que tiene OpenStack es que tiene un montón de posibilidades de configuración para hacerlo funcionar, siempre y cuando los servicios básicos (Keystone, Neutron, Glance y Nova) estén funcionando. Todo depende de los servicios que queramos añadir y utilizar en función a nuestras necesidades.

Una vez seleccionada la configuración deseada, que en nuestro caso va a ser utilizar únicamente los servicios básicos de OpenStack, debemos analizar los requisitos hardware mínimos en los nodos para poder seguir adelante. Para ello, tenemos el siguiente diagrama:

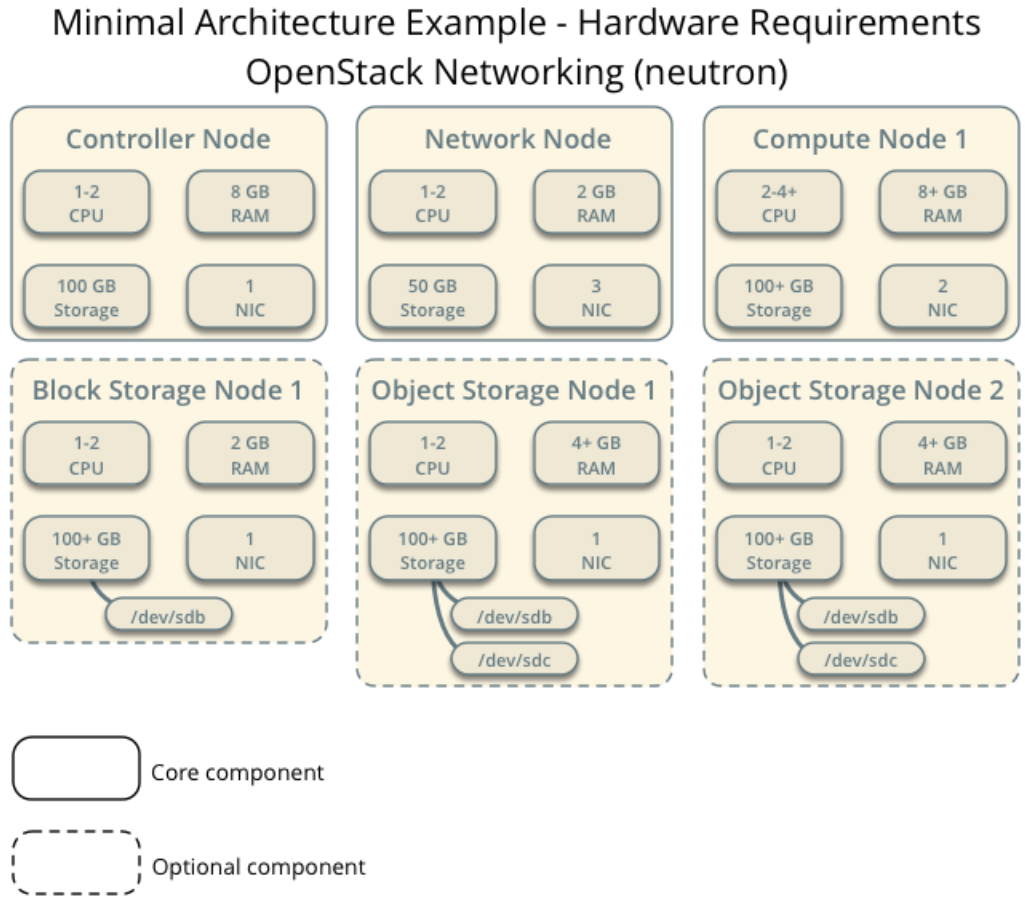


Ilustración 17. Especificaciones hardware de OpenStack

Una vez vistos los requerimientos hardware, debemos decidir que estructura de nodos vamos a utilizar, ya que hay 2 disponibles: utilizar 1 nodo de red, 1 nodo de control, y el resto de nodos de cómputo (llamémosla estructura 1), o bien, 1 nodo de control, y el resto de nodos de cómputo (llamémosla estructura 2). Cada una de estas 2 estructuras debe llevar consigo una configuración de red distinta cada uno. Veámoslo en diagramas de ejemplo:

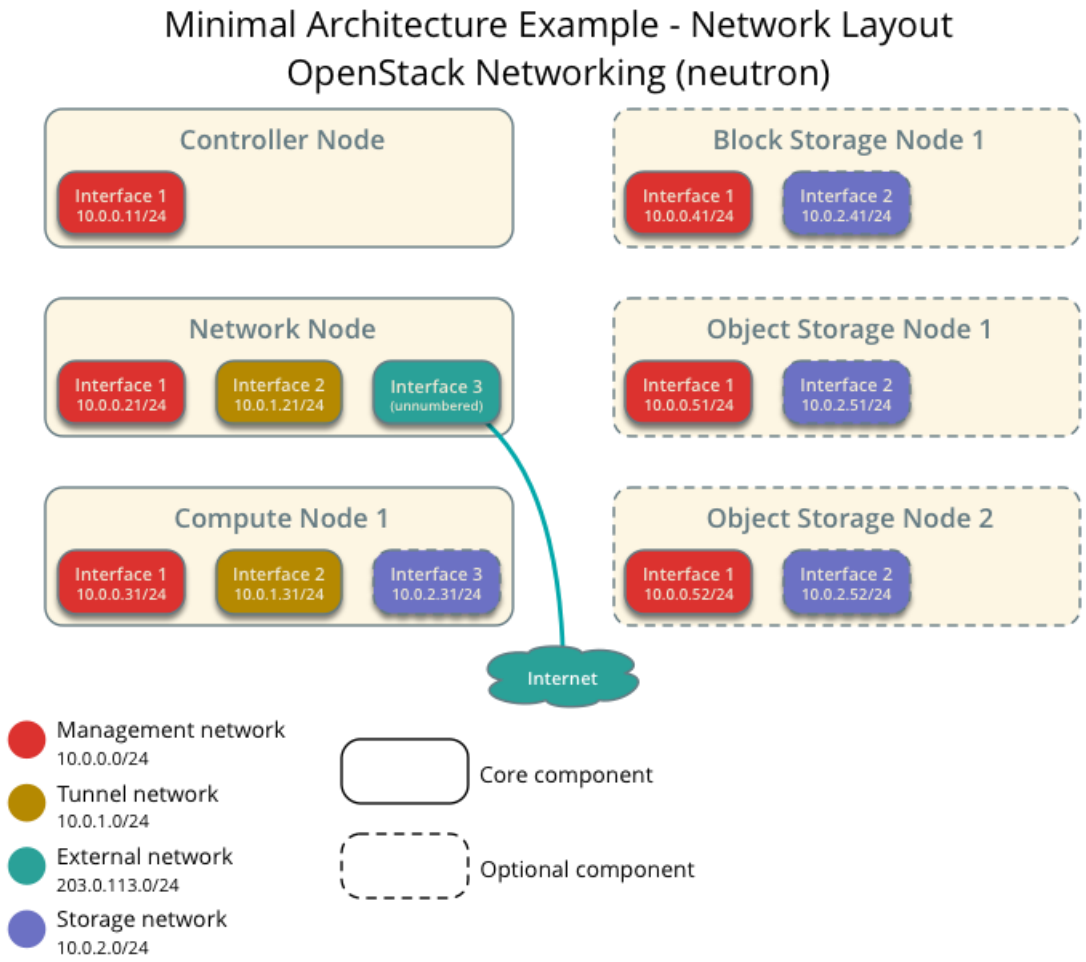


Ilustración 18. Estructura 1

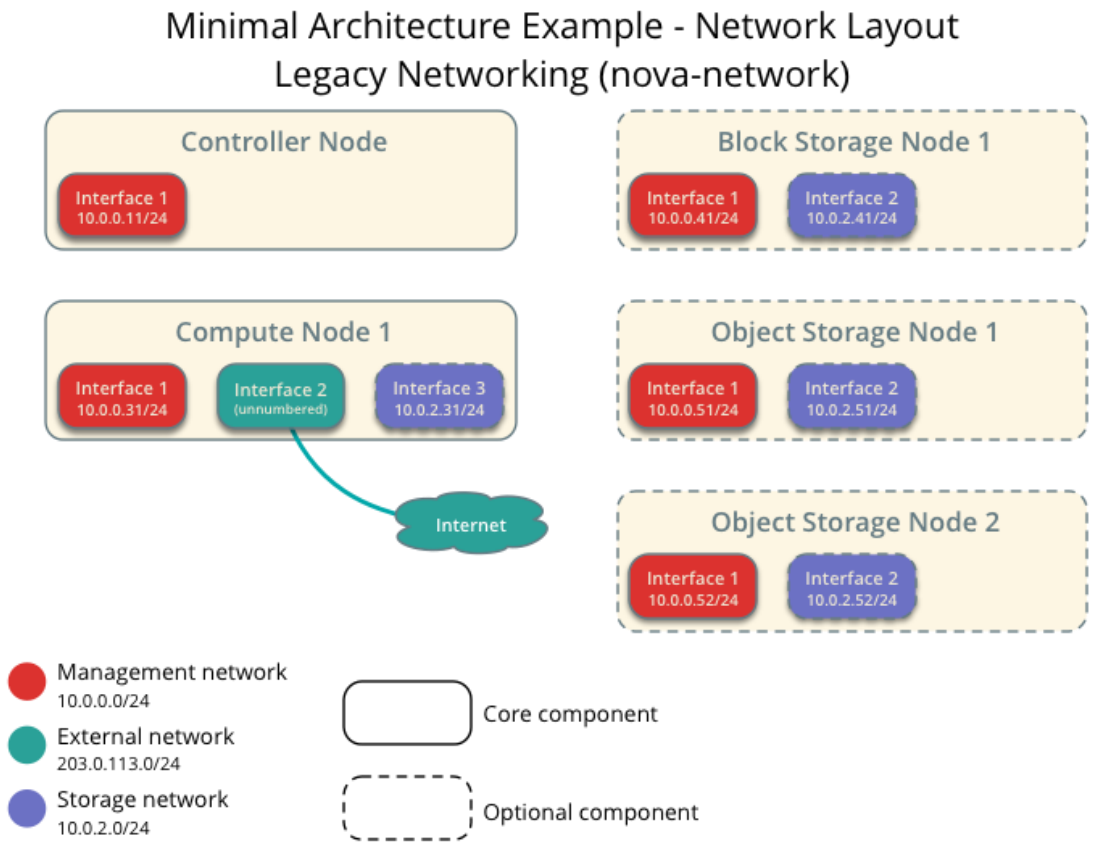


Ilustración 19. Estructura 2

Para cada configuración de red, se utilizará un servicio de red. Para la estructura 1 se utiliza el servicio de red ya citado anteriormente, Neutron, y para la estructura 2 se utiliza el servicio de red Nova-network. Éste último ni ha sido citado anteriormente ni lo vamos a utilizar en este trabajo, ya que actualmente se encuentra obsoleto y no se ajustaría a las necesidades previstas, por lo que automáticamente se descarta la opción de utilizar la estructura 2.

Una vez elegida la estructura y la configuración de servicios que queremos, aquí vemos como se organizan los servicios entre los nodos:

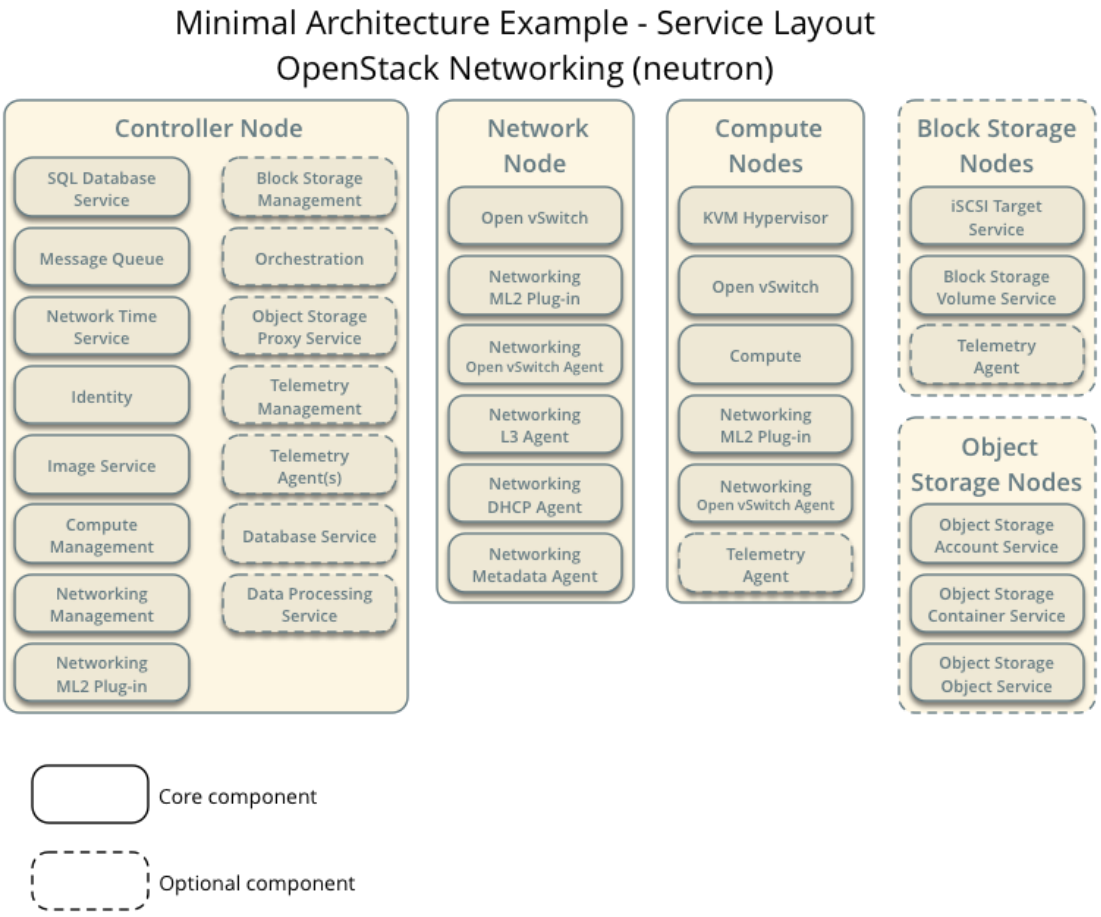


Ilustración 20. Organización de servicios en nodos

En OpenStack los servicios soportan múltiples métodos de seguridad como contraseñas, políticas y encriptación. De manera adicional, se incluyen un servidor de base de datos, y un notificador de mensajes a nivel de seguridad de contraseña. Para facilitar la instalación, utilizaremos solo seguridad a nivel de contraseñas donde sea necesario.

3.4.1. Prefase

Como prefase de la instalación, debemos revisar la configuración de red, y añadir una interfaz de red en el nodo de red para utilizarla como puente entre la red externa y las instancias virtuales.

El nodo de control tendrá la IP: 10.0.0.2, y lo llamaremos controller. Tendrá este nombre para facilitar las posteriores configuraciones, en lugar de referirse a él utilizando la dirección

IP. El nodo de red tendrá la IP: 10.0.0.8 El resto de nodos de cómputo tendrán las direcciones 10.0.0.X, siendo X: 4,5,6,9,10,11,12,13.

Siguiendo con la prefase de la instalación, debemos instalar algunos servicios ajenos a OpenStack comunes a todos los nodos.

NTP (Nodo de control)

En primer lugar, debemos instalar el servicio NTP para sincronizar los servicios entre los nodos. Para ello, escribimos el comando:

```
# apt-get install ntp
```

Por defecto, el nodo de control sincroniza la hora mediante un rango de servidores públicos. Sin embargo, se puede editar de manera opcional el archivo /etc/ntp.conf para configurar servidores alternativos.

- Editamos el archivo /etc/ntp.conf y hay que añadir, modificar o eliminar las siguientes líneas para que se ajusten a nuestras necesidades:

Ilustración 21. Archivo de configuración NTP

Donde cada término significa:

Restrict: Significa que permite la conexión a su servicio al cliente indicado.
Iburst: Significa que se envían un montón de paquetes cuando no se obtiene conexión con el primer intento.
Nomodify: Impide la reconfiguración de ntpd (con los comandos ntpq o ntpdc).
Noquery: Impide conocer el estado dumping de los datos ntp (con los comandos ntpq o ntpdc).
Kod: Es un paquete llamado Kiss-of-death y sirve para notificar la denegación de la conexión al servicio, en lugar de descartar más paquetes, por cuestiones de seguridad.
Notrap: Significa que rechaza utilizar el servicio de paquetes trampa para encontrar hosts.
NTP_SERVER: Es el servidor que utilizaremos para sincronizar la hora. Debemos elegir un servidor de bajo stratum.

- Después, reiniciar el servicio NTP:

```
# service ntp restart
```

NTP (Resto de nodos)

Seguimos con el servicio NTP. Para instalarlo en el resto de nodos, se hace de manera similar al nodo de control:

```
# apt-get install ntp
```

- Editar el archivo /etc/ntp.conf. Comentar o quitar todo excepto una línea server y referenciarla al nodo controller:

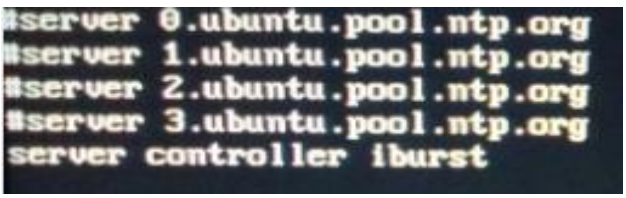


Ilustración 22. Configuración NTP en otros nodos

- Reiniciar el servicio NTP:

```
# service ntp restart
```

Una vez instalado el servicio NTP en todos los nodos, debemos verificar su funcionamiento.

- Ejecutar este comando en el nodo de control:

```
# ntpq -c peers
```

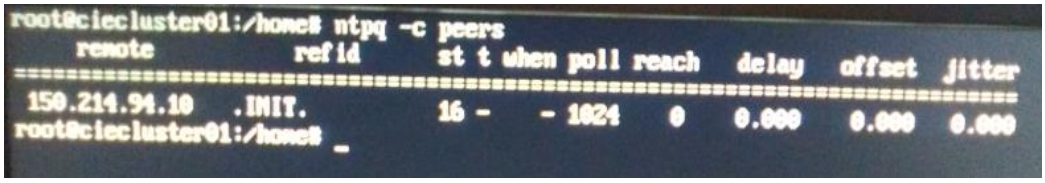


Ilustración 23. Resultado del comando

- Ejecutamos este comando en el nodo de control:

```
# ntpq -c assoc
```

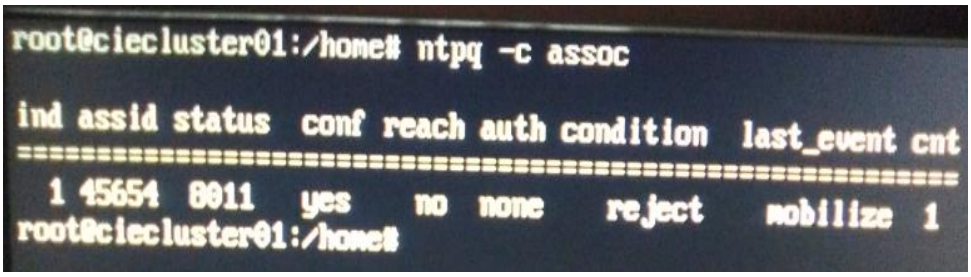


Ilustración 24. Resultado del comando

- Ejecutamos este comando en el resto de nodos, en uno cualquiera

```
# ntpq -c peers
```

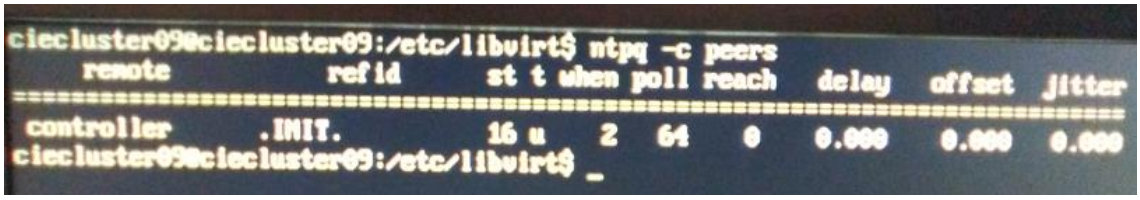


Ilustración 25. Resultado del comando

- Ejecutamos este comando también en el resto de nodos

```
# ntpq -c assoc
```

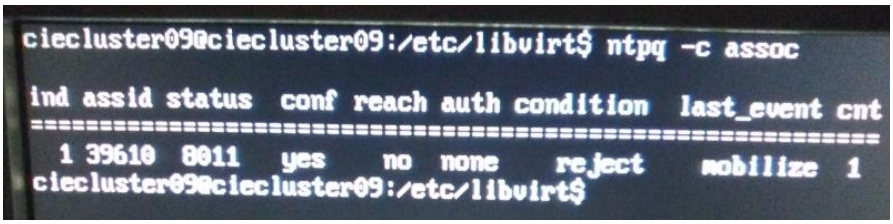


Ilustración 26. Resultado del comando

Repositorio OpenStack y paquetes básicos (Todos los nodos)

Instalado y verificado el servicio NTP en todos los nodos, es el turno de instalar los paquetes OpenStack comunes a todos los nodos.

Es importante antes de nada desactivar cualquier actualización automática, ya que puede tener consecuencias negativas en nuestro entorno OpenStack, en términos de incompatibilidad o fallos no programados.

El primer paso es instalar el repositorio de OpenStack. Para ello, ejecutamos el siguiente comando:

```
# apt-get install ubuntu-cloud-keyring
# echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu"
"trusty-updates/kilo main" > /etc/apt/sources.list.d/cloudarchive-
kilo.list
```

Para finalizar la instalación de este repositorio, debemos actualizar los paquetes instalados y la distribución, con este comando:

```
# apt-get update && apt-get dist-upgrade
```

Al incluir también una posible nueva versión del kernel del sistema operativo, es posible que el proceso tarde hasta 15 minutos en completarse, y es recomendable reiniciar el ordenador después de actualizar.

Base de datos SQL (Nodo de control)

La mayoría de los servicios de OpenStack usan una base de datos tipo SQL para almacenar información. La base de datos suele funcionar desde el nodo de control. OpenStack soporta MariaDB o MySQL dependiendo de la distribución, aunque también soporta PostgreSQL. En nuestro caso, utilizamos MariaDB, ya que es la versión más compatible con los servicios de OpenStack.

- Instalamos los siguientes paquetes:

  ```
  # apt-get install mariadb-server python-mysqldb
  ```

Elegimos una contraseña para el usuario root (administrador) de la base de datos.

- Creamos y editamos el archivo /etc/mysql/conf.d/mysqld_openstack.cnf, y hacemos lo siguiente:

    a. En la sección [mysqld], escribir en bind-address la dirección IP del nodo de control, que es donde está alojada la base de datos.

```
[mysqld]
...
bind-address = 10.0.0.11
```

    b. Siguiendo en la sección [mysqld], establecer las siguientes configuraciones para habilitar opciones interesantes y el sistema de caracteres UTF-8:

```
[mysqld]
...
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8
```

- Para finalizar la instalación:

    a. Reiniciar el servicio de bases de datos:

```
# service mysql restart
```

    b. Brindar seguridad al servicio de bases de datos:

```
# mysql_secure_installation
```

```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL
MariaDB
SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.
```

```
Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the
MariaDB
root user without the proper authorisation.

Set root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing
anyone
to log into MariaDB without having to have a user account created
for
them.  This is intended only for testing, and to make the
installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] Y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.
This
ensures that someone cannot guess at the root password from the
network.

Disallow root login remotely? [Y/n] Y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone
can
access.  This is also intended only for testing, and should be
removed
before moving into a production environment.

Remove test database and access to it? [Y/n] Y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so
far
will take effect immediately.

Reload privilege tables now? [Y/n] Y
 ... Success!

Cleaning up...
```

```
All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Cola de mensajes (Nodo de control)

OpenStack utiliza un servicio de cola de mensajes para coordinar operaciones y recopilar información sobre el estado de los servicios. Normalmente se ejecuta en el nodo de control. OpenStack soporta un gran número de servicios de cola de mensajes, incluyendo RabbitMQ, Qpid, y ZeroMQ. Sin embargo, la mayoría de distribuciones soportan un servicio de cola de mensajes en particular, en concreto, el RabbitMQ.

- Para instalar el servicio de cola de mensajes:

```
# apt-get install rabbitmq-server
```

- Una vez instalado el servicio, hay que configurarlo:

  a. Añadir el usuario “openstack”:

  ```
  # rabbitmqctl add_user openstack RABBIT_PASS
  ```

  b. Permitir configuración, escritura y lectura para el usuario “openstack”:

  ```
  # rabbitmqctl set_permissions openstack ".*" ".*" ".*"
  ```

3.4.2. Servicio Keystone (Identity)

Conceptos del servicio

El servicio de identidad Keystone realiza las siguientes funciones:

- Seguimiento de usuarios y sus respectivos permisos.
- Proporciona un catálogo de servicios a partir de los ‘endpoint’ de su API.

Al instalar el servicio Keystone, se debe registrar cada servicio en la instalación de OpenStack. Keystone puede monitorizar qué servicios de OpenStack están instalados, y su localización en la red.

Para entender este servicio, debemos tener claros varios conceptos:

Usuario: Representación digital de una persona, sistema o servicio que utiliza los servicios cloud de OpenStack. El servicio identidad verifica que las peticiones entrantes son hechas por el usuario que reivindica dichas peticiones. Los usuarios tienen un login y pueden tener asignados 'tokens' para acceder a los recursos. También pueden ser directamente asignados a un proyecto en particular y comportarse como si estuvieran dentro del mismo.

Credenciales: Datos que confirman la identidad de usuario. Por ejemplo: nombre de usuario y contraseña; nombre de usuario y clave API, o un token de autenticación proporcionado por el servicio de identidad.

Autenticación: El proceso de confirmación de identidad de un usuario. El servicio de identidad de OpenStack autoriza una solicitud entrante verificando una serie de credenciales proporcionadas por el usuario. Inicialmente estas credenciales son un nombre de usuario y una contraseña, o un nombre de usuario y una clave API. Cuando las credenciales son validadas, el servicio de identidad genera un token de autenticación que el usuario utilizará en posteriores peticiones.

Token: Una cadena de caracteres alfanuméricos utilizada para acceder a las APIs de OpenStack y sus recursos. Un token puede ser eliminado en cualquier momento y es válido para una duración determinada. Aunque el servicio de identidad de OpenStack soporta autenticación basada en tokens en esta versión, el propósito es soportar protocolos adicionales en el futuro. Su principal propósito es convertirse en un servicio de integración, y no ser única y exclusivamente una solución de mantenimiento y almacenamiento de identidades.

Proyecto: Un contenedor usado para agrupar o aislar recursos. También agrupan y aíslan objetos de identidad. Dependiendo del servicio que se esté ejecutando, un proyecto puede referenciar a un cliente, cuenta o a una organización.

Servicio: Un servicio OpenStack, como Compute (Nova), Object Storage (Swift) o el servicio de Imagen (Glance). Proporciona uno o más 'endpoints' donde los usuarios pueden acceder a los recursos y llevar a cabo determinadas operaciones.

Endpoint: Una dirección accesible por red donde se accede a un servicio, normalmente una dirección URL. Si se utiliza una extensión para modelos, se puede crear un 'modelo endpoint' que representa los modelos de todos los servicios disponibles en todas las regiones.

Rol: Una personalidad con una serie de derechos y privilegios definidos para llevar a cabo una serie de operaciones. En el servicio de identidad, un token que es asignado a un usuario incluye la lista de roles. Los servicios que están siendo utilizado por ese usuario determinan cómo interpretan los roles que tiene un usuario, y las operaciones o recursos a los que cada rol proporciona el acceso.

Cliente Keystone: Una interfaz de comandos para la API del servicio de identidad de OpenStack. Por ejemplo, los usuarios pueden ejecutar los comandos 'keystone service-create' y 'keystone endpoint-create' para registrar servicios en sus instalaciones de OpenStack.

El siguiente diagrama muestra el flujo de proceso del servicio identidad de OpenStack (Ver ANEXO II, pág. 151).

Instalación y configuración

Esta sección describe como instalar y configurar el servicio identidad de OpenStack, llamado Keystone, en el nodo de control. Esta configuración pone en marcha el servidor HTTP Apache para manejar peticiones y el servicio Memcached para almacenar tokens en lugar de una base de datos SQL.

A. Prerequisitos

Antes de nada, se debe crear una base de datos y un token de administración.

Para crear la base de datos:

- Usar el cliente de acceso a bases de datos para conectarse al servidor de bases de datos utilizando el usuario root:

```
$ mysql -u root -p
```

- Crear la base de datos 'keystone':

```
CREATE DATABASE keystone;
```

- Conceder acceso a la base de datos 'keystone'

```
 GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost'
IDENTIFIED BY 'KEYSTONE_DBPASS';
 GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%'
 IDENTIFIED BY 'KEYSTONE_DBPASS';
```

  Reemplazar *KEYSTONE_DBPASS* por una contraseña deseada.

- Salir del cliente de la base de datos.

Para crear el token de administración:

- Generar un valor aleatorio para usarlo como token de administración durante la configuración inicial:

```
# openssl rand -hex 10
```

Ilustración 27. Token generado

B. Componentes del servicio

Los archivos de configuración por defecto pueden variar según la versión, por lo que podría ser necesario añadir las secciones que se indicarán a continuación en caso de que no existan previamente.

En esta versión, la librería obsoleta Eventlet del servicio Keystone se desecha en favor de un servidor WSGI. Se utilizará el servidor HTTP Apache con mod_wsgi para atender peticiones en los puertos 5000 y 35357, de manera que se puede simular perfectamente el comportamiento del servicio Keystone. Dicho servicio utiliza también los puertos 5000 y 35357, por lo que se desactivará para evitar conflicto.

- Desactivar el inicio automático del servicio Keystone después de su instalación:

```
# echo "manual" > /etc/init/keystone.override
```

- Ejecutar el siguiente comando para instalar los paquetes:

```
# apt-get install keystone python-openstackclient
apache2 libapache2-mod-wsgi memcached python-memcache
```

- Editar el archivo /etc/keystone/keystone.conf y realizar las siguientes configuraciones:

    a. En la sección [DEFAULT], definir el valor inicial del token de administración:

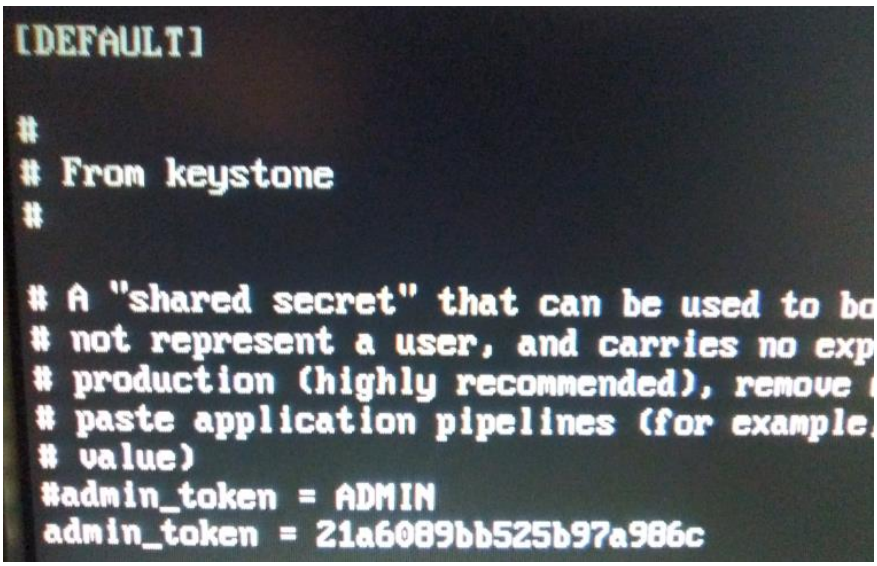


Ilustración 28. Token admin de archivo keystone.conf

    b. En la sección [database], configurar el acceso a la base de datos:

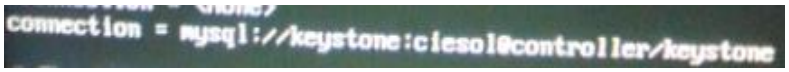


Ilustración 29. Conexión a base de datos Keystone

c. En la sección [memcache], configurar el servicio Memcache:

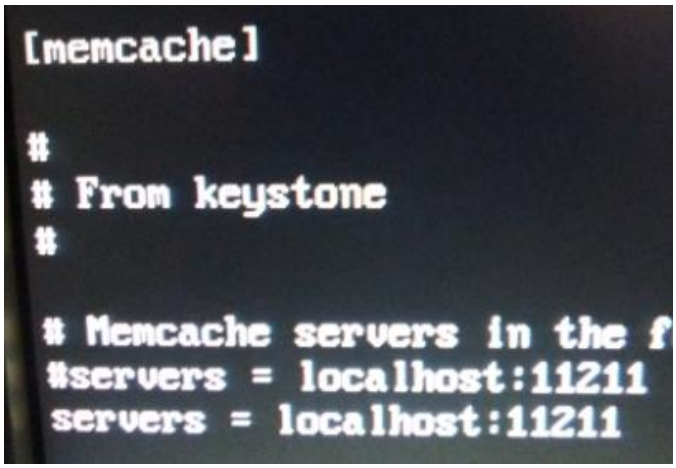


Ilustración 30. Sección [memcache]

d. En la sección [token], configurar el proveedor del token UUID y el driver de Memcached:

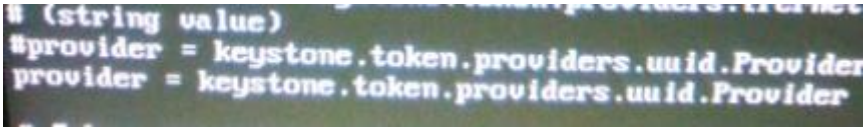


Ilustración 31. Sección [token]

e. En la sección [revoke], configurar el driver de revocación SQL:

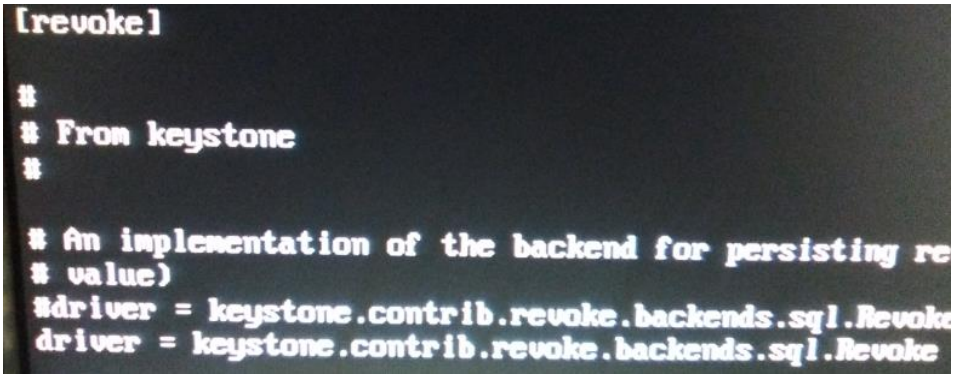


Ilustración 32. Sección [revoke]

f. (Opcional) Para facilitar la solución de posibles problemas, habilitar los logs del archivo en la sección [DEFAULT]:

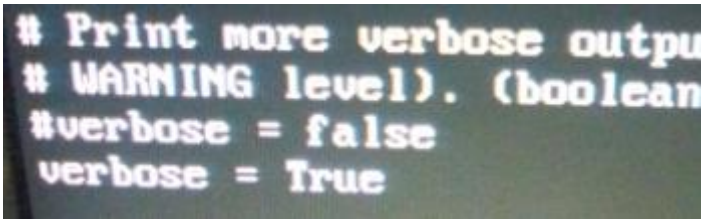


Ilustración 33. Habilitación de logs para errores

- Reflejar los cambios en la base de datos del servicio identidad:

```
# su -s /bin/sh -c "keystone-manage db_sync" keystone
```

C. Configuración del servidor HTTP Apache

- Editar el archivo /etc/apache2/apache2.conf y configurar el parámetro ServerName para que referencie a controller (nodo control):

```
ServerName controller
```

- Crear el archivo /etc/apache2/sites-available/wsgi-keystone.conf con el siguiente contenido:

```
Listen 5000
Listen 35357

<VirtualHost *:5000>
    WSGIDaemonProcess keystone-public processes=5 threads=1
user=keystone display-name=%{GROUP}
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /var/www/cgi-bin/keystone/main
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    <IfVersion >= 2.4>
      ErrorLogFormat "%{cu}t %M"
    </IfVersion>
    LogLevel info
    ErrorLog /var/log/apache2/keystone-error.log
    CustomLog /var/log/apache2/keystone-access.log combined
</VirtualHost>
```

```
<VirtualHost *:35357>
    WSGIDaemonProcess keystone-admin processes=5 threads=1
user=keystone display-name=%{GROUP}
    WSGIProcessGroup keystone-admin
    WSGIScriptAlias / /var/www/cgi-bin/keystone/admin
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    <IfVersion >= 2.4>
      ErrorLogFormat "%{cu}t %M"
    </IfVersion>
    LogLevel info
    ErrorLog /var/log/apache2/keystone-error.log
    CustomLog /var/log/apache2/keystone-access.log combined
</VirtualHost>
```

- Activar los hosts virtuales del servicio de identidad:

```
# ln -s /etc/apache2/sites-available/wsgi-keystone.conf
/etc/apache2/sites-enabled
```

- Crear la estructura de directorio para los components del servidor WSGI:

```
# mkdir -p /var/www/cgi-bin/keystone
```

- Copiar los componentes de WSGI desde el repositorio OpenStack al siguiente directorio:

```
# curl http://git.openstack.org/
cgit/openstack/keystone/plain/
httpd/keystone.py?h=stable/kilo | tee /var/www/cgi-
bin/keystone/main /var/www/cgi-bin/keystone/admin
```

- Ajustar propiedad y permisos en el directorio y en los archivos contenidos:

```
# chown -R keystone:keystone /var/www/cgi-bin/keystone
# chmod 755 /var/www/cgi-bin/keystone/*
```

D. Finalizando…

- Reiniciar el servidor HTTP Apache:

```
#  service apache2 restart
```

- (Opcional) Por defecto, los paquetes de Ubuntu Server crean una base de datos SQLite. Como esta configuración usa una base de datos SQL, se puede eliminar el archivo de base de datos SQLite:

```
# rm -f /var/lib/keystone/keystone.db
```

Creando la entidad de servicio y el endpoint API.

El servicio de identidad proporciona un catálogo de servicios y sus localizaciones. Cada servicio que se añada a nuestro entorno OpenStack requiere una entidad de servicio y un número de endpoints API.

A. Prerequisitos

Por defecto, la base de datos del servicio de identidad no contiene información de apoyo para la autenticación convencional ni para el catálogo de servicios. Debemos utilizar un token de autenticación temporal que se creó anteriormente para inicializar la entidad de servicio y el endpoint API del servicio identidad.

Se debe asignar el valor del token de autenticación al entorno OpenStack con el parámetro --os-token o asignar dicho valor a la variable de entorno OS_TOKEN.

De manera similar, se debe asignar el valor de la URL del servicio de identidad al entorno OpenStack con el parámetro --os-url o asignar dicho valor a la variable de entorno OS_URL. En nuestro caso, utilizaremos las variables de entorno para facilitar el proceso.

- Configurar el token de autenticación:

  ```
  $ export OS_TOKEN=ADMIN_TOKEN
  ```

  Reemplazar el valor ADMIN_TOKEN por el token de autenticación que generamos anteriormente.

- Configurar el endpoint URL:

  ```
  $ export OS_URL=http://controller:35357/v2.0
  ```

B. Crear la entidad de servicio y el endpoint API

El servicio de identidad gestiona un catálogo de servicios en nuestro entorno OpenStack. Los servicios utilizan este catálogo para ver los servicios disponibles en nuestro entorno.

- Crear la entidad de servicio para el servicio de identidad:

```
$ openstack service create --name keystone --description
"OpenStack Identity" identity
```

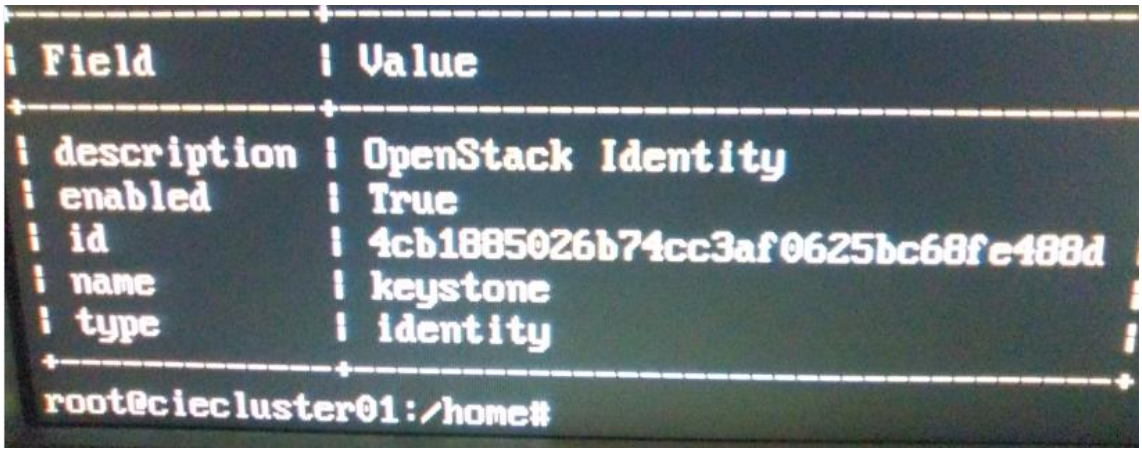


Ilustración 34. Creación de entidad de servicio Identity

- El servicio de identidad gestiona un catálogo de endpoints API asociados con los servicios de nuestro entorno OpenStack. Dichos servicios utilizan este catálogo para ver cómo comunicarse con otros servicios de nuestro entorno.
  OpenStack utiliza tres endpoints API para cada servicio: admin, internal y public. El endpoint admin permite la modificación de usuarios y proyectos por defecto, mientras que los endpoints public e internal no lo permiten. Esto se debe a que cada variante de endpoint está situada en distintas redes, por motivos de seguridad.
  Para simplificar, utilizaremos todos los endpoints en la misma red local, y la región por defecto `RegionOne`.

Para crear el endpoint API del servicio de identidad:

```
$ openstack endpoint create
--publicurl http://controller:5000/v2.0
--internalurl http://controller:5000/v2.0
--adminurl http://controller:35357/v2.0
--region RegionOne identity
```

Ilustración 35. Creación de endpoint Identity

Crear proyectos, usuarios y roles

El servicio de identidad proporciona servicios de autenticación para cada servicio de OpenStack. El servicio de autenticación utiliza una combinación de dominios, proyectos, usuarios y roles.

A. Crear proyectos, usuarios y roles

- Crear un proyecto de administración, usuario y rol para operaciones administrativas en nuestro entorno:

a. Crear el proyecto admin:

```
$ openstack project create --description "Admin Project"
admin
```

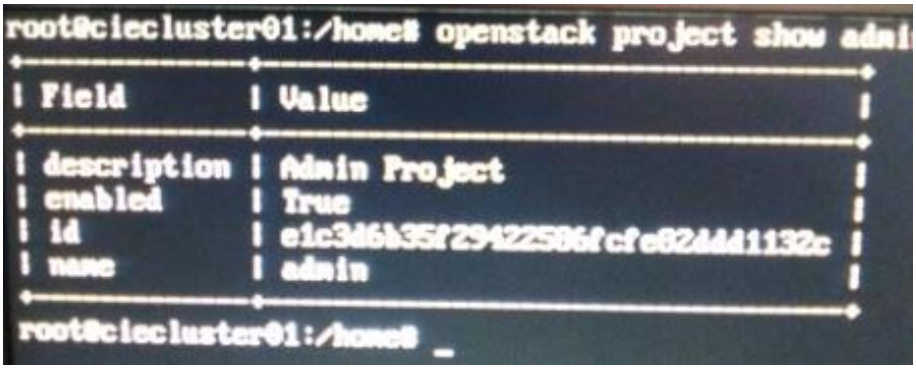


Ilustración 36. Creación de proyecto admin

b. Crear el usuario admin:

```
$ openstack user create --password-prompt admin
User Password:
Repeat User Password:
```

Ilustración 37. Creación de usuario admin

c. Crear el rol admin:

```
$ openstack role create admin
```

Ilustración 38. Creación de rol admin

d. Añadir el rol admin al proyecto y usuario admin

```
$ openstack role add --project admin --user admin admin
```

Cualquier rol que creamos debe referenciar a los roles especificados en el archivo policy.json en el archivo o directorio de configuración de cada servicio de OpenStack. La política por defecto para la mayoría de servicios concede permisos de administración al rol admin.

- Utilizaremos un proyecto “servicio” que contendrá un usuario único para cada servicio que añadamos a nuestro entorno.

a. Creamos el proyecto “servicio”:

```
$ openstack project create --description "Service Project"
service
```

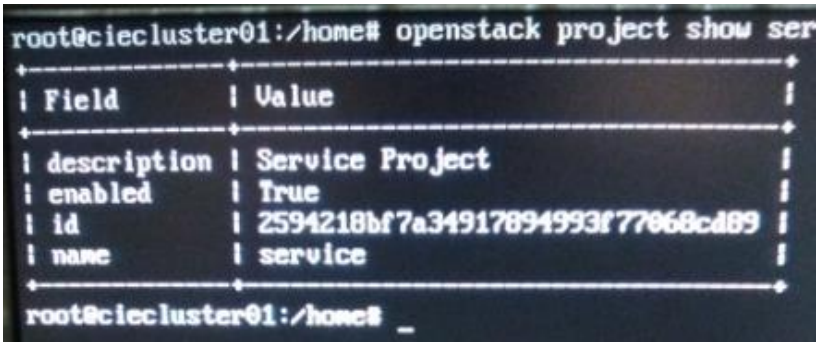


Ilustración 39. Creación de proyecto servicio

- Las tareas regulares (no administrativas) deberían utilizar un usuario y proyecto que no tengan privilegios. Como ejemplo, crearemos el usuario y proyecto “demo”.

    a. Creamos el proyecto “demo”:

```
$ openstack project create --description "Demo Project" demo
```

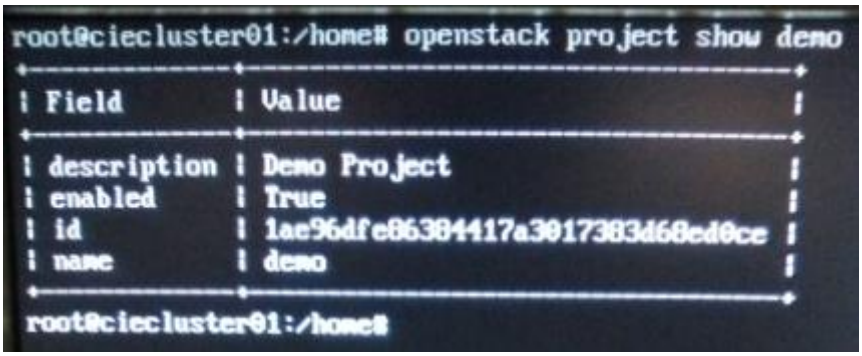


Ilustración 40. Creación de proyecto demo

    b. Creamos el usuario “demo”:

```
$ openstack user create --password-prompt demo
User Password:
Repeat User Password:
```

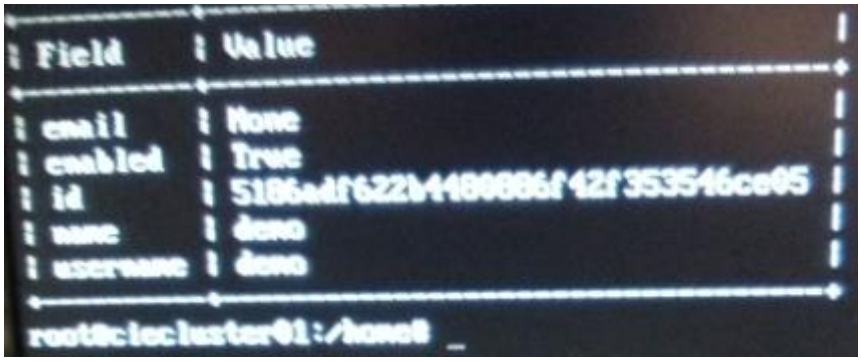


Ilustración 41. Creación de usuario demo

c. Creamos el rol “user”:

```
$ openstack role create user
```

Ilustración 42. Creación rol user

d. Añadimos el rol “user” al proyecto y usuario “demo”:

```
$ openstack role add --project demo --user demo user
```

Podemos repetir este proceso para crear proyectos y usuarios adicionales.

Verificar la configuración

Verificaremos que todo esté correctamente instalado antes de proseguir con los otros servicios.

- Por razones de seguridad, desactivamos el mecanismo de autenticación por token temporal. Esto lo hacemos editando el archivo /etc/keystone/keystone-paste.ini y eliminando el parámetro “admin_token_auth” de las secciones [pipeline:public_api], [pipeline:admin_api] y [pipeline:api_v3].

- Eliminamos el valor de las variables OS_TOKEN y OS_URL

```
$ unset OS_TOKEN OS_URL
```

- Para el usuario admin, solicitamos un token de autenticación de la API del servicio de identidad v2.0:

```
  $ openstack --os-auth-url http://controller:35357 --os-
project-name admin --os-username admin --os-auth-type password
token issue
  Password:
```

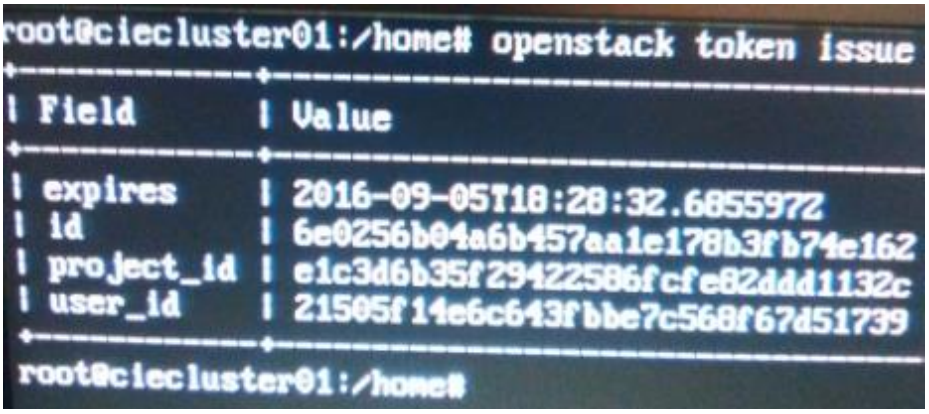


Ilustración 43. Solicitud de token de autenticación v2.0

- La versión 3 de la API del servicio de identidad soporta dominios que contienen proyectos y usuarios. Ambos pueden utilizar los mismos nombres en diferentes dominios. Sin embargo, para usar la versión 3, las peticiones deben contener al menos el dominio "default" o usar IDs. Para facilitar las cosas, utilizaremos el dominio "default".

```
  $ openstack --os-auth-url http://controller:35357 --os-
project-domain-id default --os-user-domain-id default --os-
project-name admin --os-username admin --os-auth-type password
token issue
  Password:
```

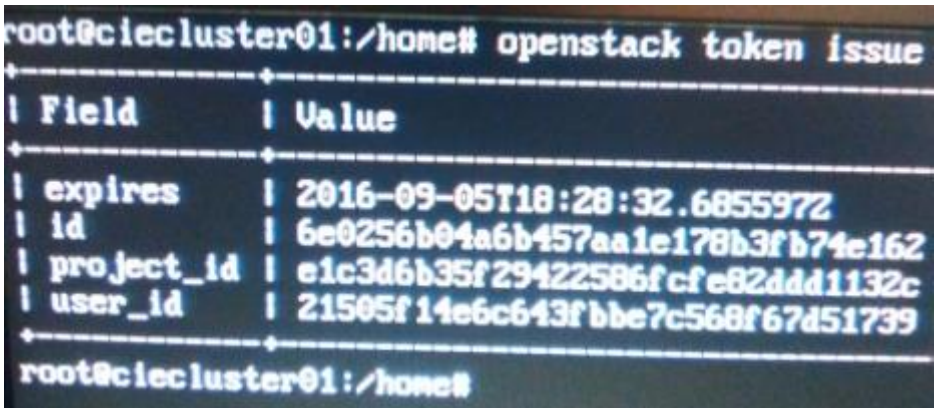


Ilustración 44. Solicitud de token de autenticación v3.0

- Como usuario admin, mostramos los proyectos para verificar que el usuario admin puede ejecutar comandos con privilegios solo administrador, y verificar además que el servicio de identidad contiene los proyectos creados anteriormente.

```
  $ openstack --os-auth-url http://controller:35357 --os-
project-name admin --os-username admin --os-auth-type password
project list
  Password:
```

Ilustración 45. Listado de proyectos

- Como usuario admin, mostramos los usuarios para verificar que el servicio de identidad contiene los usuarios creados anteriormente.

```
  $ openstack --os-auth-url http://controller:35357 --os-
project-name admin --os-username admin --os-auth-type password
user list
  Password:
```

Ilustración 46. Listado de usuarios

- Como usuario admin, mostramos los roles para verificar que el servicio de identidad contiene los roles creados anteriormente.

```
  $ openstack --os-auth-url http://controller:35357 --os-
project-name admin --os-username admin --os-auth-type password
role list
  Password:
```

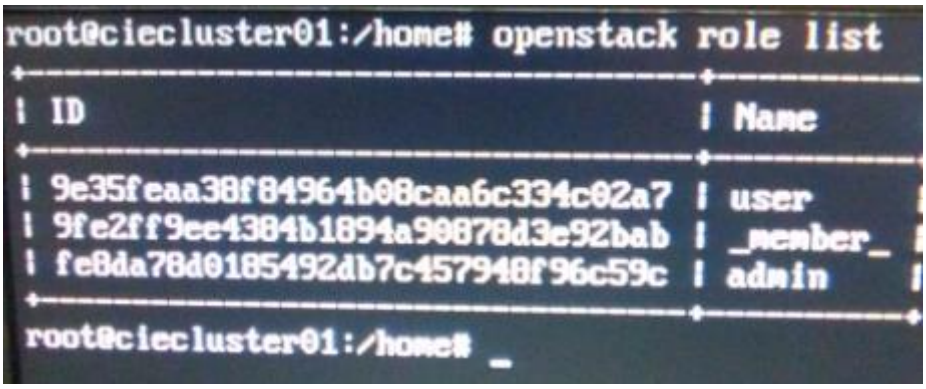


Ilustración 47. Listado de roles

- Como usuario demo, solicitamos un token de autenticación de la API versión 3:

```
  $ openstack --os-auth-url http://controller:5000 --os-
project-domain-id default --os-user-domain-id default --os-
project-name demo --os-username demo --os-auth-type password
token issue
  Password:
```

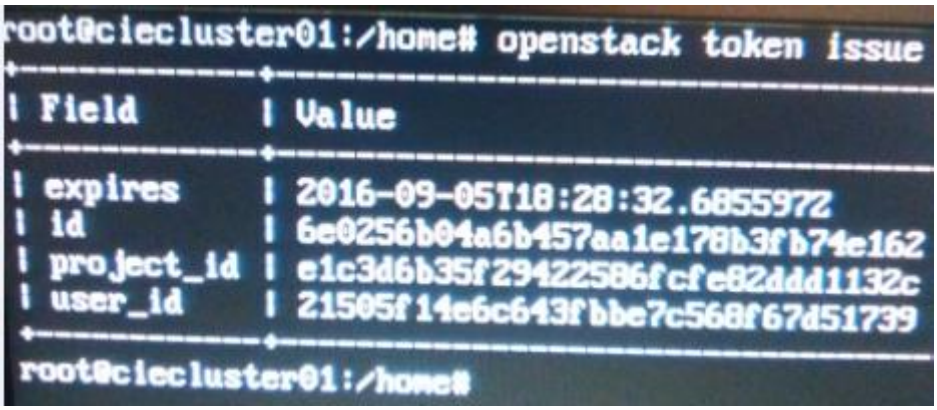


Ilustración 48. Solicitud de token de autenticación como usuario demo

- Como usuario demo, solicitamos un listado de usuarios para verificar que no podemos ejecutar comandos con privilegios de solo administrador.

```
$ openstack --os-auth-url http://controller:5000 --os-
project-domain-id default --os-user-domain-id default --os-
project-name demo --os-username demo --os-auth-type password
user list
```

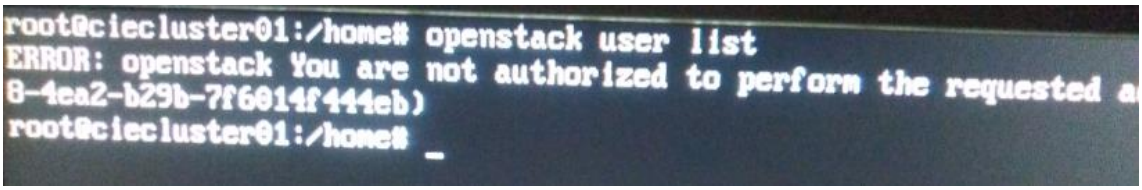


Ilustración 49. Acceso denegado como usuario demo

(Opcional) Creación de scripts para variables de entorno para OpenStack

Anteriormente se han utilizado algunas variables de entorno para interactuar con el servicio de identidad a través de la interfaz de comandos propia de OpenStack. Para incrementar la eficiencia de estas operaciones, se pueden utilizar scripts para variables de entorno. Dichos scripts son conocidos como archivos OpenRC. Estos scripts normalmente contienen variables y opciones comunes para todos los clientes, pero también soportan variables únicas.

A. Creación de los scripts

Vamos a crear los scripts para variables de entorno para los usuarios admin y demo y los proyectos admin y demo.

- Creamos el archivo con nombre admin-openrc.sh en cualquier directorio y añadimos el siguiente contenido:

```
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://controller:35357/v3
```

  Recordando que ADMIN_PASS es la contraseña que seleccionamos anteriormente para el usuario admin cuando configurábamos el servicio de identidad.

- Creamos el archivo con nombre demo-openrc.sh en cualquier directorio y añadimos el siguiente contenido:

```
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=demo
export OS_TENANT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=DEMO_PASS
export OS_AUTH_URL=http://controller:5000/v3
```

  Recordando que DEMO_PASS es la contraseña que seleccionamos anteriormente para el usuario demo cuando configurábamos el servicio de identidad.

B. Cargar los scripts creados.

Para ejecutar servicios como usuario y proyecto específicos, simplemente cargamos uno de los scripts que hemos creado. Por ejemplo:

- Para cargar el script de las variables del usuario admin:

```
$ source admin-openrc.sh
```

- Solicitar un token de autenticación:

```
$ openstack token issue
```

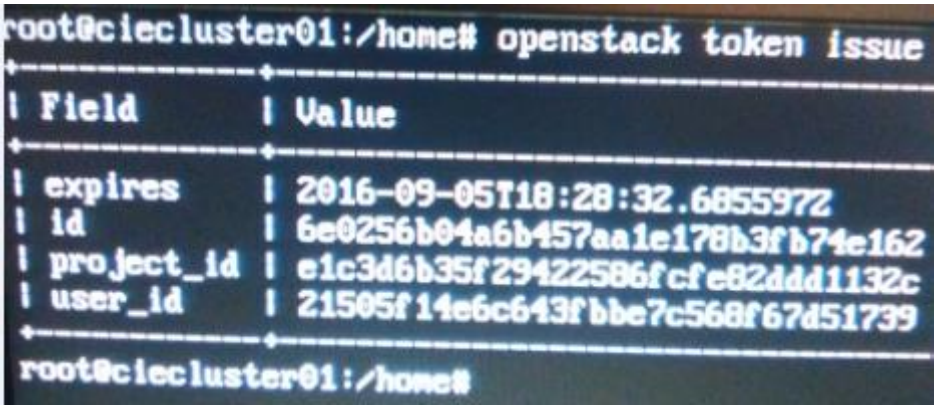


Ilustración 50. Solicitud de token con script de variables

De esta forma, tendremos ya cargadas y asignadas todas las variables que necesitemos para realizar operaciones en nuestro entorno OpenStack, y no habrá que cargarlas cada vez que queramos utilizar algún comando en la interfaz de OpenStack. Este script debe cargarse cada vez que se reinicie el sistema.

3.4.3. Servicio Glance (Image)

Conceptos

El servicio de imagen de OpenStack (Glance) habilita a los usuarios a descubrir, registrar y recuperar imágenes de máquinas virtuales. Ofrece una API REST que posibilita el hacer uso de los datos de una imagen de máquina virtual y utilizar una réplica de la misma. Se pueden almacenar dichas imágenes mediante el servicio de imagen en un determinado número de localizaciones, desde sistemas de archivos simples a sistemas de almacenamiento de objetos como el Almacén de Objetos de OpenStack (servicio Swift).

Principalmente acepta solicitudes API para imágenes de discos o servidores. También soporta el almacenamiento de imágenes de discos o servidores en varios tipos de repositorio, como el antes mencionado Almacén de Objetos de OpenStack.

Para soportar el proceso de caché, están funcionando determinados procesos simultáneamente en el servicio de imagen de OpenStack, como por ejemplo, procesos de replicación, que aseguran consistencia y disponibilidad. Otros procesos que ayudan a lo mismo son por ejemplo, procesos auditores y actualizadores.

El servicio de imagen de OpenStack tiene los siguientes componentes:

- ✓ glance-api: Acepta llamadas a la API del servicio de imagen para el proceso de creación, recuperación y almacenamiento de imágenes.
- ✓ glance-registry: Almacena, procesa y recupera metadatos de imágenes. Los metadatos incluyen parámetros como el tamaño y el tipo.
- ✓ Base de datos: Almacena metadatos de imágenes y nos da opción a elegir nuestra base de datos de acuerdo a nuestras preferencias. Las más utilizadas: MySQL o SQLite.
- ✓ Repositorio de almacenamiento para archivos de imágenes: Se admiten varios tipos de repositorio, incluyendo sistemas de archivos simples, Almacén de Objetos, dispositivos de bloques RADOS, HTTP, y Amazon S3. Algunos de estos repositorios solo se pueden usar en modo lectura.

Instalación y configuración

Aquí describiremos como instalar y configurar el servicio de imagen, llamado Glance, en el nodo de control. Esta configuración almacena imágenes en el sistema de archivos local.

A. Prerequisitos

Antes de instalar el servicio de imagen, se debe crear una base de datos, credenciales para este servicio, y un endpoint API.

- Para crear la base de datos:

    a. Usar el cliente de acceso a bases de datos para conectarse al servidor de bases de datos utilizando el usuario root:

```
$ mysql –u root –p
```

b. Crear la base de datos glance:

```
CREATE DATABASE glance;
```

c. Conceder acceso a la base de datos glance:

```
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost'
IDENTIFIED BY 'GLANCE_DBPASS';
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED
BY 'GLANCE_DBPASS';
```

Reemplazar *GLANCE_DBPASS* por una contraseña deseada.

d. Salir del cliente de la base de datos.

- Cargar las credenciales de admin a través del script que creamos anteriormente:

```
$ source admin-openrc.sh
```

- Para crear las credenciales del servicio hacemos lo siguiente:

a. Creamos el usuario glance:

```
$ openstack user create --password-prompt glance
User Password:
Repeat User Password:
```

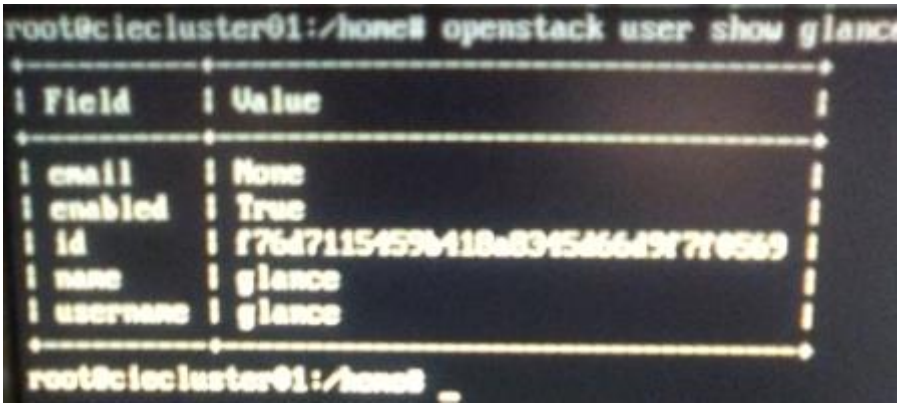


Ilustración 51. Creación del usuario glance

b. Añadir el rol admin al usuario glance y al Proyecto servicio:

```
$ openstack role add --project service --user glance admin
```

c. Crear la entidad de servicio glance:

```
$ openstack service create --name glance --description
"OpenStack Image service" image
```

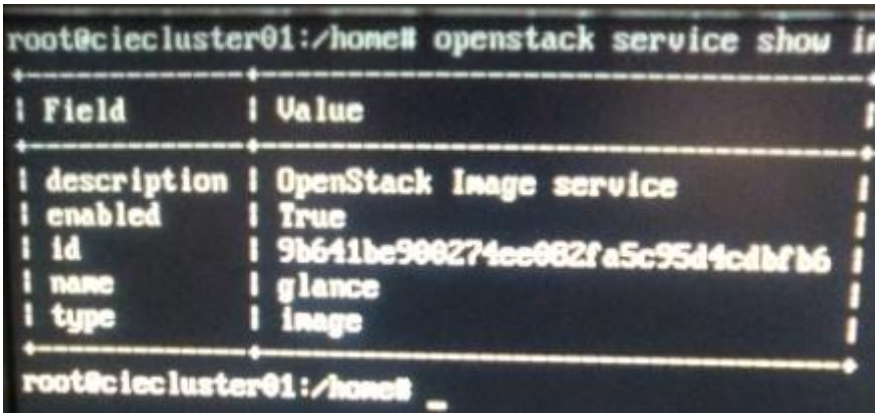


Ilustración 52. Creación de servicio image

d. Crear el endpoint API del servicio de imagen:

```
$ openstack endpoint create
--publicurl http://controller:9292
--internalurl http://controller:9292
--adminurl http://controller:9292
--region RegionOne image
```

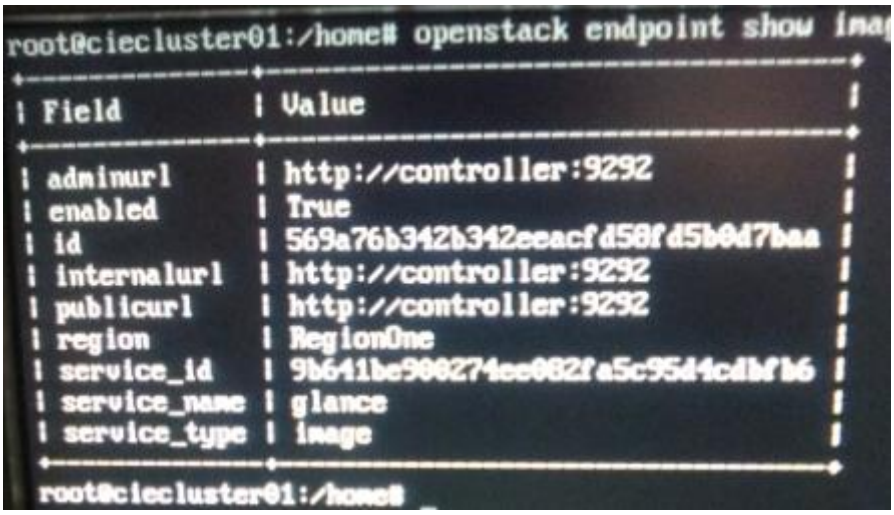


Ilustración 53. Creación de endpoint image

B. Componentes del servicio.

Los archivos de configuración por defecto pueden variar según la versión, por lo que podría ser necesario añadir las secciones que se indicarán a continuación en caso de que no existan previamente.

- Instalar los siguientes paquetes:

```
# apt-get install glance python-glanceclient
```

- Editar el archivo /etc/glance/glance-api.conf y realizar las siguientes configuraciones:

    a. En la sección [database], configurar el acceso a la base de datos glance:

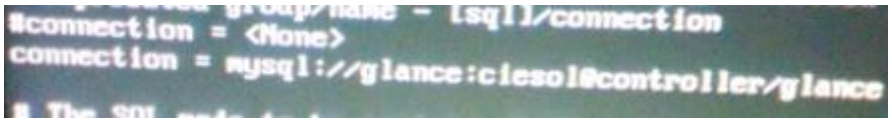


Ilustración 54. Conexión a base de datos glance

    b. En las secciones [keystone_authtoken] y [paste_deploy], configurar el acceso al servicio de identidad:

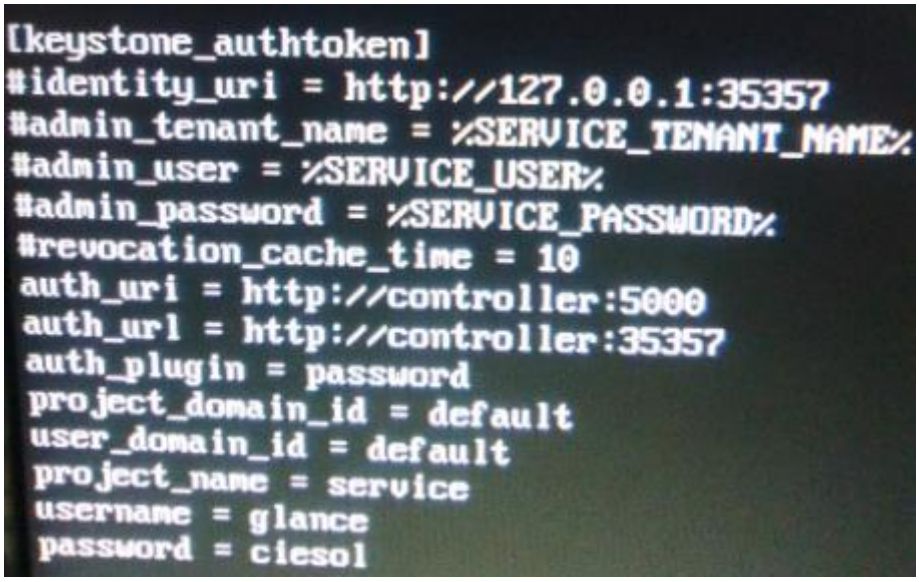


Ilustración 55. Sección [keystone_authtoken]

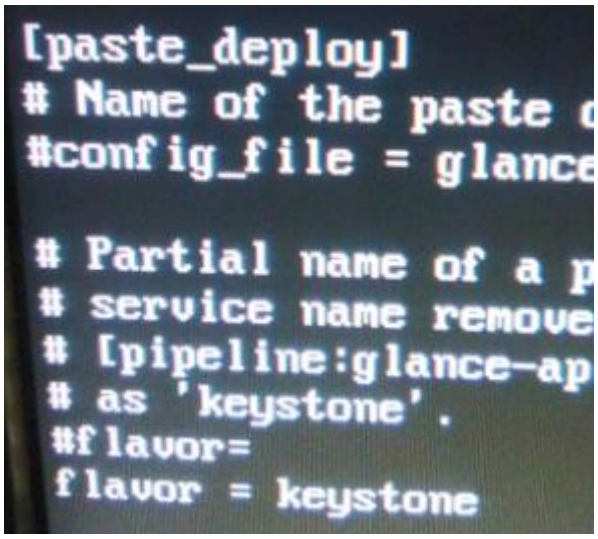


Ilustración 56. Sección [paste_deploy]

c. En la sección [glance_store], configurar el almacenamiento de archivos local y localización de los archivos de imagen:

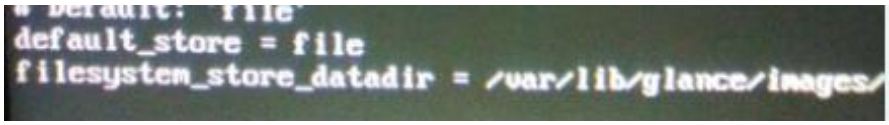


Ilustración 57. Sección [glance_store]

d. En la sección [DEFAULT], configurar el driver de notificación "noop" para desactivar las notificaciones ya que solo pertenecen al servicio de telemetría:

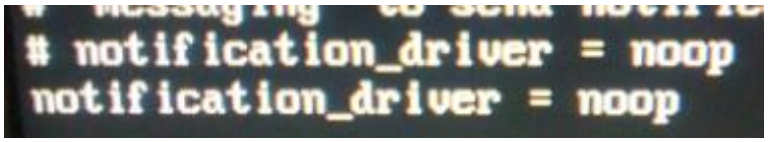


Ilustración 58. Configuración de driver noop

e. (Opcional) Para ayudar a la solución de posibles problemas, habilitamos los logs en la sección [DEFAULT]:

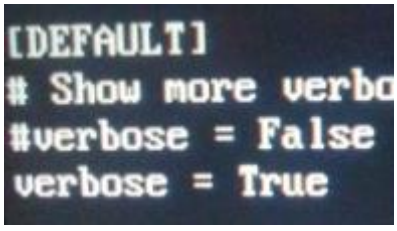


Ilustración 59. Habilitación de logs para errores

- Editamos el archivo /etc/glance/glance-registry.conf y realizamos las siguientes configuraciones:

  a. En la sección [database], configurar el acceso a la base de datos glance:

Ilustración 60. Conexión a base de datos glance

  b. En las secciones [keystone_authtoken] y [paste_deploy], configurar el acceso al servicio de identidad:

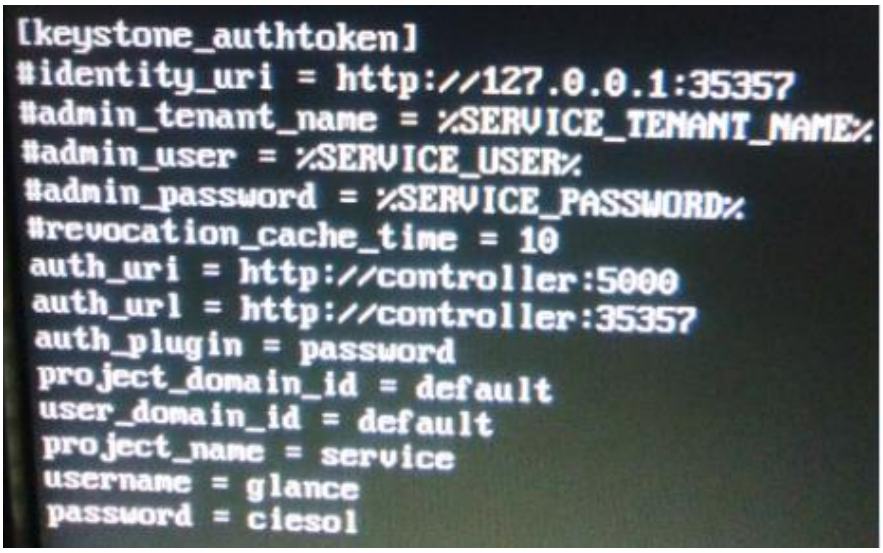


Ilustración 61. Sección [keystone_authtoken]

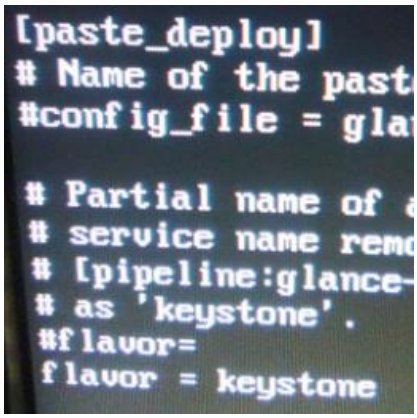


Ilustración 62. Sección [paste_deploy]

c. En la sección [DEFAULT], configurar el driver de notificación “noop” para desactivar las notificaciones ya que solo pertenecen al servicio de telemetría:

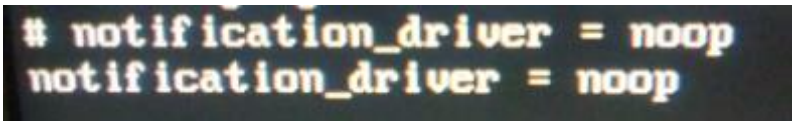


Ilustración 63. Configuración de driver noop

d. (Opcional) Para ayudar a la solución de posibles problemas, habilitamos los logs en la sección [DEFAULT]:

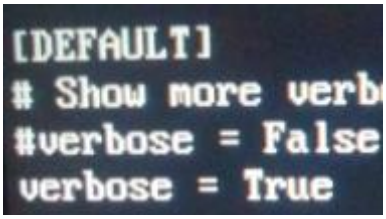


Ilustración 64. Habilitación de logs para errores

e. Reflejar los cambios en la base de datos del servicio de imagen:

```
# su -s /bin/sh -c "glance-manage db_sync" glance
```

C. Finalizando…

- Reiniciar el servicio de imagen:

```
# service glance-registry restart
# service glance-api restart
```

- (Opcional) Por defecto, los paquetes de Ubuntu crean una base de datos SQLite. Como esta configuración utiliza una base de datos SQL, podemos eliminar el archivo de base de datos SQLite:

```
# rm -f /var/lib/glance/glance.sqlite
```

Verificar instalación y configuración

Verificaremos el funcionamiento de la instalación utilizando CirrOS, una pequeña imagen de Linux que nos ayudará a testear nuestro entorno.

- Para cada script que diseñamos anteriormente, configuraremos unos parámetros que habilitarán al cliente del servicio de imagen para usar la versión 2.0 de su API:

```
  # echo "export OS_IMAGE_API_VERSION=2" | tee -a admin-
openrc.sh demo-openrc.sh
```

- Una vez hecho esto, cargamos el script de admin:

```
$ source admin-openrc.sh
```

- Creamos un directorio temporal local:

```
$ mkdir /tmp/images
```

- Introducimos el archivo de imagen en el directorio local que acabamos de crear. Tenemos varias opciones, como por ejemplo, introducir el archivo de manera local mediante un dispositivo de almacenamiento, o bien descargándola desde una URL. En nuestro caso, para mayor comodidad, lo descargaremos de un repositorio a través de la herramienta wget:

```
$ wget -P /tmp/images http://download.cirros-
cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
```

- Ahora que ya tenemos el archivo de imagen, creamos la imagen dentro del servicio de imagen utilizando el formato de disco QCOW2, formato de contenedor bare, y visibilidad pública para que todos los proyectos puedan acceder a ella:

```
$ glance image-create --name "cirros-0.3.4-x86_64" --file
/tmp/images/cirros-0.3.4-x86_64-disk.img --disk-format qcow2 --
container-format bare --visibility public --progress
```

- Verificamos que se ha subido correctamente la nueva imagen al servicio de imagen:

```
$ glance image-list
```

Ilustración 65. Listado de imágenes disponibles

- (Opcional) Finalmente, podemos eliminar el directorio temporal y el archivo de imagen utilizado anteriormente:

  ```
  $ rm -r /tmp/images
  ```

3.4.4. Servicio Nova (Compute)

Conceptos

El servicio de cómputo se usa para alojar y gestionar sistemas de computación en la nube. Es una de las partes que forman una Infraestructura como Servicio (IaaS). Principalmente está implementado en el lenguaje Python.

Nova interactúa con el servicio Keystone (Servicio de identidad) para la autenticación, con el servicio Glance (Servicio de imagen) para usar imágenes de disco o servidor, y el servicio Horizon (Interfaz web) para la interfaz de usuario y administración. El acceso al servicio de imagen está limitado por proyectos y por usuarios; determinadas "quotas" están limitadas por proyecto (por ejemplo, número de instancias que se pueden crear).

Una de las principales características de este servicio es que puede escalar horizontalmente con hardware estándar, esto quiere decir que mejorará su rendimiento al añadir más nodos al sistema.

El servicio de cómputo Nova se compone de los siguientes parámetros:

- ✓ Nova-api: Acepta y responde llamadas del usuario final a la API del servicio de cómputo. Las APIs que soporta el servicio son la API de cómputo de OpenStack, la API de Amazon EC2, y una API especial para usuarios con privilegios para llevar a cabo operaciones administrativas. Sigue algunas políticas e inicia la mayoría de actividades de orquestación, como arrancar una instancia.
- ✓ Nova-api-metadata: Acepta solicitudes de metadatos provenientes de las instancias. Se usa generalmente cuando se trabaja en modo multi-host en instalaciones con nova-network.
- ✓ Nova-compute: Un daemon que crea y termina instancias de máquinas virtuales a través de APIs de Hypervisors. Por ejemplo: XenAPI para XenServer, libvirt para KVM o QEMU, VMwareAPI para VMware, entre otros. Su modo de trabajo es ligeramente complicado. El daemon acepta solicitudes de una cola de operaciones y lleva a cabo dichas solicitudes, que básicamente son una serie de órdenes de sistema, como por ejemplo lanzar una instancia KVM y actualizar su estado en la base de datos.
- ✓ Nova-scheduler: Acepta peticiones para generar una instancia de máquina virtual de una cola y organiza en que nodo se va a ejecutar dicha instancia.
- ✓ Nova-conductor: Hace de intermediario entre nova-compute y la base de datos. Elimina los accesos directos a la base de datos de la nube creada por nova-compute. Escala de manera horizontal, pero no se está ejecutando en los nodos donde está funcionando nova-compute.
- ✓ Nova-cert: Un daemon que sirve para utilizar certificados X509. Solo se necesitan para la API EC2.
- ✓ Nova-network: Similar a nova-compute, acepta tareas de red de una cola y manipula la red. Lleva a cabo tareas como habilitar interfaces puente o cambiar reglas de Iptables.
- ✓ Nova-consoleauth: Autoriza tokens de usuarios provistos por proxies de consola, dichos proxies necesarios para hacer funcionar este proceso. Se pueden ejecutar proxies de cualquier tipo para que este proceso funcione en una configuración de cluster.
- ✓ Nova-novncproxy: Proporciona un proxy para acceder a las instancias a través de una conexión VNC. Soporta clientes de navegación web novnc.
- ✓ Nova-spicehtml5proxy: Proporciona un proxy para acceder a las instancias a través de una conexión SPICE. Soporta navegadores web con HTML5.
- ✓ Nova-xvpvncproxy: Proporciona un proxy para acceder a las instancias a través de una conexión VNC. Soporta clientes de OpenStack específicos en Java.
- ✓ Nova-objectstore: Una interfaz S3 para registrar imágenes mediante el servicio de imagen de OpenStack. Usado principalmente para instalaciones que necesiten utilizar "euca2ools". Esta aplicación se comunica con nova-objectstore en lenguaje S3, y dicho proceso traduce las peticiones S3 en peticiones al servicio de imagen.

- ✓ Cliente nova: Habilita una interfaz de comando para los usuarios admin o usuarios finales.
- ✓ Cola: Un "hub" central para transmitir mensajes entre daemons. Normalmente implementada con RabbitMQ, pero puede implementarse con cualquier cola de mensajes AMPQ, como Apache Qpid o Zero MQ.
- ✓ Base de datos SQL: Almacena la mayoría de estados de tipo fecha de creación o tiempo de ejecución en una infraestructura en la nube, como por ejemplo: Tipos de instancia disponibles, instancias en uso, redes disponibles y proyectos. El servicio de cómputo puede soportar cualquier base de datos que soporte SQL-Alchemy. Las más comunes son SQLite3, MySQL y PostgreSQL.

Instalación y configuración

En esta sección explicaremos como instalar el servicio de cómputo, llamado Nova, en el nodo de control y nodos de cómputo.

A. Prerequisitos (NODO CONTROL)

Como en los anteriores servicios, debemos crear una base de datos, credenciales y un endpoint API:

- Para crear la base de datos:

    a. Usar el cliente de acceso a bases de datos para conectarse al servidor de bases de datos utilizando el usuario root:

    ```
    $ mysql –u root –p
    ```

    b. Crear la base de datos nova:

    ```
    CREATE DATABASE nova;
    ```

    c. Conceder acceso a la base de datos nova:

    ```
    GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost'
IDENTIFIED BY 'NOVA_DBPASS';
    GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%'
IDENTIFIED BY 'NOVA_DBPASS';
    ```

    Reemplazar *NOVA_DBPASS* por una contraseña deseada.

d. Salir del cliente de la base de datos.

- Cargar las credenciales de admin a través del script que creamos anteriormente:

```
$ source admin-openrc.sh
```

- Para crear las credenciales del servicio hacemos lo siguiente:

a. Creamos el usuario nova:

```
$ openstack user create --password-prompt nova
User Password:
Repeat User Password:
```

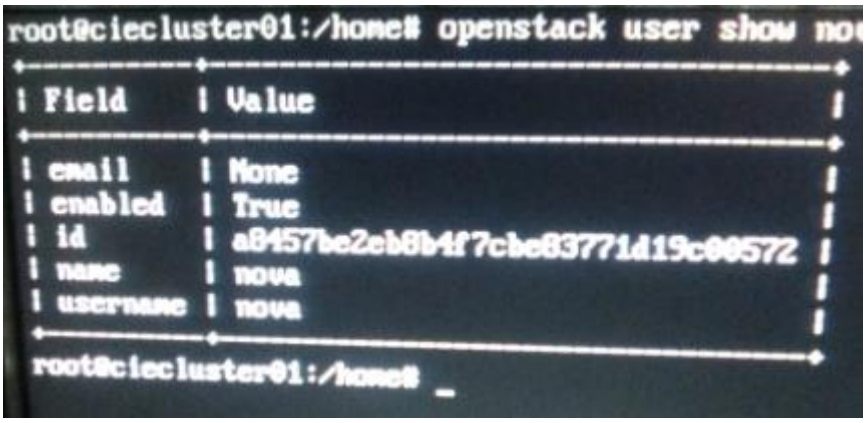


Ilustración 66. Creación de usuario nova

b. Añadir el rol admin al usuario nova y al Proyecto servicio:

```
$ openstack role add --project service --user nova admin
```

c. Crear la entidad de servicio nova:

```
$ openstack service create --name nova --description
"OpenStack Compute" compute
```

Ilustración 67. Creación de servicio compute

- Crear el endpoint API del servicio de cómputo:

```
$ openstack endpoint create
--publicurl http://controller:8774/v2/%\(tenant_id\)s
--internalurl http://controller:8774/v2/%\(tenant_id\)s
--adminurl http://controller:8774/v2/%\(tenant_id\)s
--region RegionOne compute
```

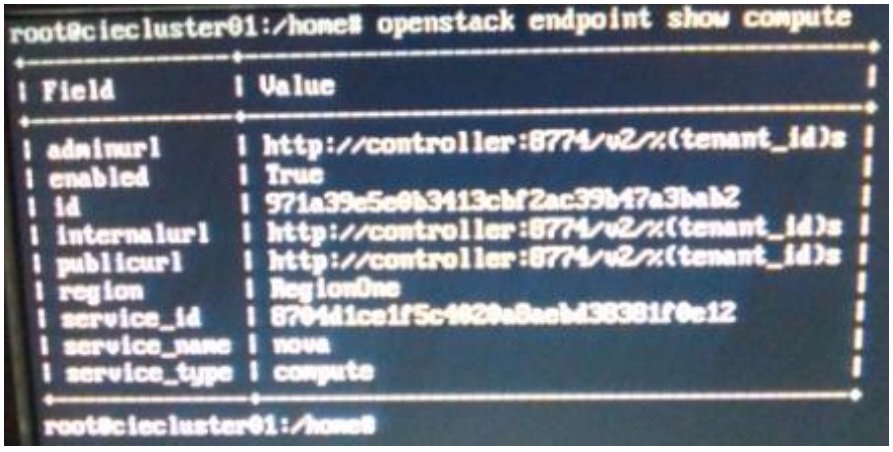


Ilustración 68. Creación del endpoint compute

B. Componentes del servicio (NODO CONTROL)

Como en los otros servicios, los archivos de configuración por defecto pueden variar según la versión, por lo que podría ser necesario añadir las secciones que se indicarán a continuación en caso de que no existan previamente.

- Instalamos los siguientes paquetes:

```
# apt-get install nova-api nova-cert nova-conductor nova-
consoleauth nova-novncproxy nova-scheduler python-novaclient
```

- Editamos el archivo /etc/nova/nova.conf y realizamos las siguientes configuraciones:

  a. Añadir la sección [database], y en ella incluir el acceso a la base de datos:

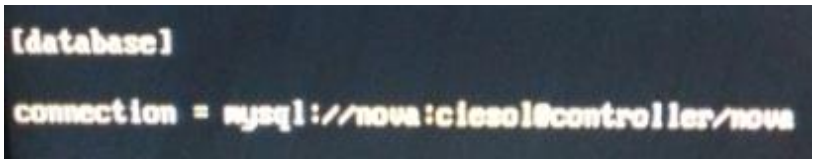


Ilustración 69. Conexión a base de datos nova

  b. En las secciones [DEFAULT] y [oslo_messaging_rabbit], configurar el acceso a la cola de mensajes RabbitMQ:

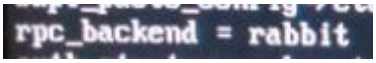


Ilustración 70. Habilitando acceso a RabbitMQ

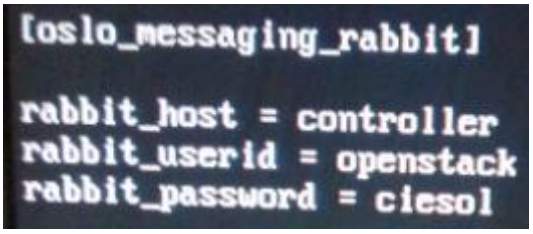


Ilustración 71. Parámetros de RabbitMQ

c. En las secciones [DEFAULT] y [keystone_authtoken], configurar el acceso al servicio de identidad:

```
auth_strategy = keystone
```

Ilustración 72. Configurando acceso a keystone

```
[keystone_authtoken]

auth_uri = http://controller:5000
auth_url = http://controller:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
password = ciesol
```

Ilustración 73. Sección [keystone_authtoken]

d. En la sección [DEFAULT], configurar la opción “my_ip” para utilizar la ip local del nodo, en este caso el nodo de control:

```
my_ip = 10.0.0.2
```

Ilustración 74. IP local del nodo de control

e. En la sección [DEFAULT], configurar el proxy VNC para utilizar la ip local del nodo de control:

```
vncserver_listen = 10.0.0.2
vncserver_proxyclient_address = 10.0.0.2
```

Ilustración 75. Configuración proxy VNC

f. En la sección [glance], establecer la localización del servicio de imagen, en este caso, la ip o el nombre de host del nodo de control:

```
[glance]
host = controller
```

Ilustración 76. Sección [glance]

g. En la sección [oslo_concurrency], configurar el directorio “lock”:

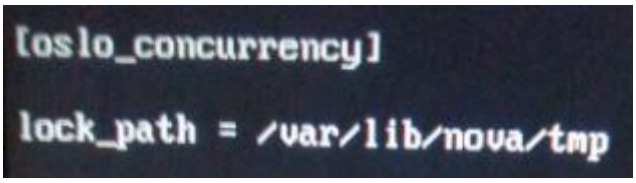


Ilustración 77. Configurando directorio lock

h. (Opcional) Para ayudar a solucionar posibles problemas, habilitar los logs en la sección [DEFAULT]:

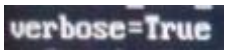


Ilustración 78. Habilitación de logs para errores

- Reflejar los cambios en la base de datos del servicio de cómputo:

```
# su -s /bin/sh -c "nova-manage db sync" nova
```

C. Finalizando… (NODO CONTROL)

- Reiniciar los procesos del servicio Nova:

```
# service nova-api restart
# service nova-cert restart
# service nova-consoleauth restart
# service nova-scheduler restart
# service nova-conductor restart
# service nova-novncproxy restart
```

- (Opcional) Por defecto, los paquetes de Ubuntu crean una base de datos SQLite. Como esta configuración utiliza una base de datos SQL, podemos eliminar el archivo de base de datos SQLite:

```
# rm -f /var/lib/nova/nova.sqlite
```

D. Componentes del servicio (NODOS CÓMPUTO)

Ahora procederemos a instalar los componentes del servicio de cómputo en los nodos de cómputo. Debemos instalar los hypervisors para poder lanzar instancias de máquinas virtuales. En nuestro caso, utilizaremos QEMU con la extensión KVM sólo en los nodos que soporten aceleración hardware para las máquinas virtuales. Esta configuración está pensada para que nuestro entorno sea escalable de manera horizontal añadiendo nodos adicionales.

Como en los otros servicios, los archivos de configuración por defecto pueden variar según la versión, por lo que podría ser necesario añadir las secciones que se indicarán a continuación en caso de que no existan previamente.

- Instalamos los siguientes paquetes:

```
# apt-get install nova-compute sysfsutils
```

- Editamos el archive /etc/nova/nova.conf y realizamos las siguientes configuraciones:

  a. En las secciones [DEFAULT] y [oslo_messaging_rabbit], configurar el acceso a la cola de mensajes RabbitMQ:

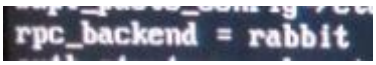


Ilustración 79. Habilitando RabbitMQ

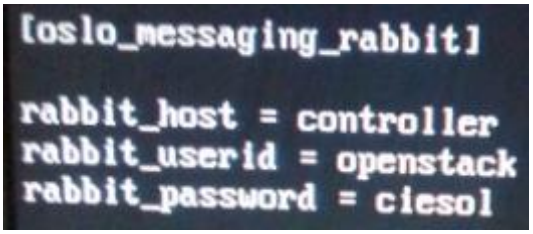


Ilustración 80. Parámetros de RabbitMQ

  b. En las secciones [DEFAULT] y [keystone_authtoken], configurar el acceso al servicio de identidad:

Ilustración 81. Configurando el acceso a keystone

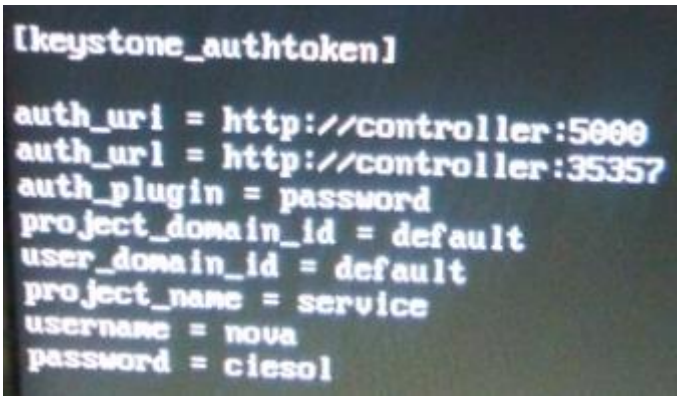


Ilustración 82. Sección [keystone_authtoken]

c. En la sección [DEFAULT], configurar la opción “my_ip” para utilizar la ip local del nodo de cómputo en cuestión:

Ilustración 83. IP local de un nodo de cómputo

d. En la sección [DEFAULT], habilitar y configurar el acceso por consola remota:

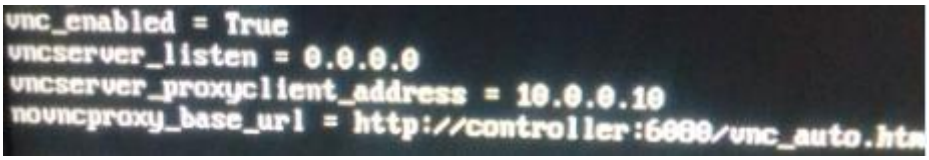


Ilustración 84. Configuración de proxy VNC

El parámetro “vncserver_listen” indica que se escuchan todas las direcciones IP.
El parámetro “vncserver_proxyclient_address” solo escucha la dirección IP del nodo de cómputo.
El parámetro “novncproxy_base_url” es la dirección que introducimos en el navegador y que nos permite acceder a la instancia a través de la conexión VNC.

e. En la sección [glance], establecer la localización del servicio de imagen, en este caso, la ip o el nombre de host del nodo de control:

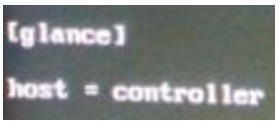


Ilustración 85. Sección [glance]

f. En la sección [oslo_concurrency], configurar el directorio "lock":

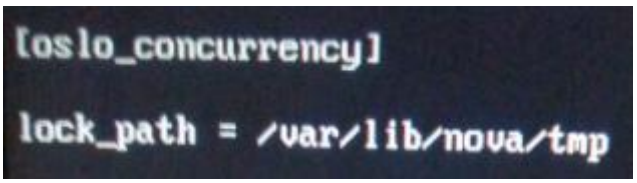


Ilustración 86. Configuración de directorio lock

g. (Opcional) Para ayudar a solucionar posibles problemas, habilitar los logs en la sección [DEFAULT]:

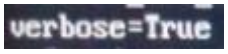


Ilustración 87. Habilitando logs para errores

E. Finalizando… (NODOS CÓMPUTO)

- Determinar si el nodo de cómputo que estamos configurando soporta aceleración hardware. Esto lo averiguamos con el siguiente comando:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

Si el comando devuelve un valor de 1 o más, el nodo soportaría aceleración hardware y no requeriría ninguna configuración adicional, es decir, se podría utilizar KVM. Si devuelve un valor de 0, el nodo no soporta aceleración hardware y por lo tanto debemos configurar la librería libvirt para utilizar QEMU en lugar de KVM.

Ilustración 88. Determinando si se puede usar KVM o QEMU

Como nos devuelve un 0, debemos configurar el nodo para utilizar QEMU.

a. Editamos el archivo /etc/nova/nova-compute.conf, la sección [libvirt]:

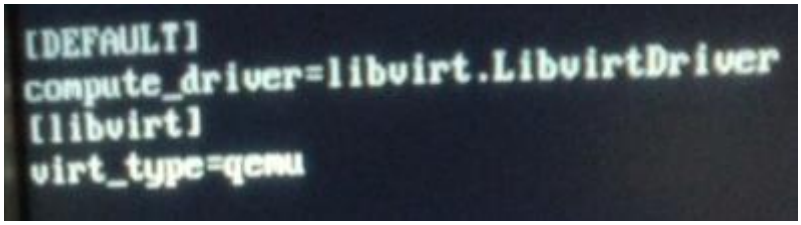


Ilustración 89. Habilitando QEMU en nodo de cómputo

- Reiniciar el servicio de cómputo:

```
# service nova-compute restart
```

- (Opcional) Por defecto, los paquetes de Ubuntu crean una base de datos SQLite. Como esta configuración utiliza una base de datos SQL, podemos eliminar el archivo de base de datos SQLite:

```
# rm -f /var/lib/nova/nova.sqlite
```

Verificar instalación y configuración

Ahora procedemos a revisar la configuración del servicio para verificar que todo se ha realizado de manera correcta. Para ello, ejecutaremos unos comandos de comprobación en el nodo de control.

- Cargamos las credenciales de admin mediante nuestro script habitual:

```
$ source admin-openrc.sh
```

- Listamos los componentes del servicio para verificar el lanzamiento y el registro de cada componente, y que están corriendo y funcionando correctamente:

```
$ nova service-list
```

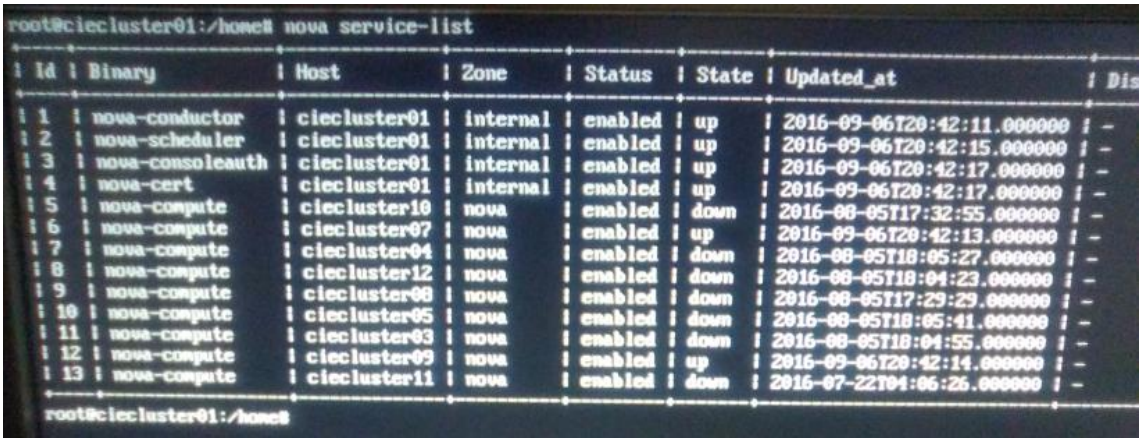


Ilustración 90. Listado de los servicios de Nova en todos los nodos

- Listamos los endpoints API del servicio de identidad para verificar la conectividad con dicho servicio:

```
$ nova endpoints
```

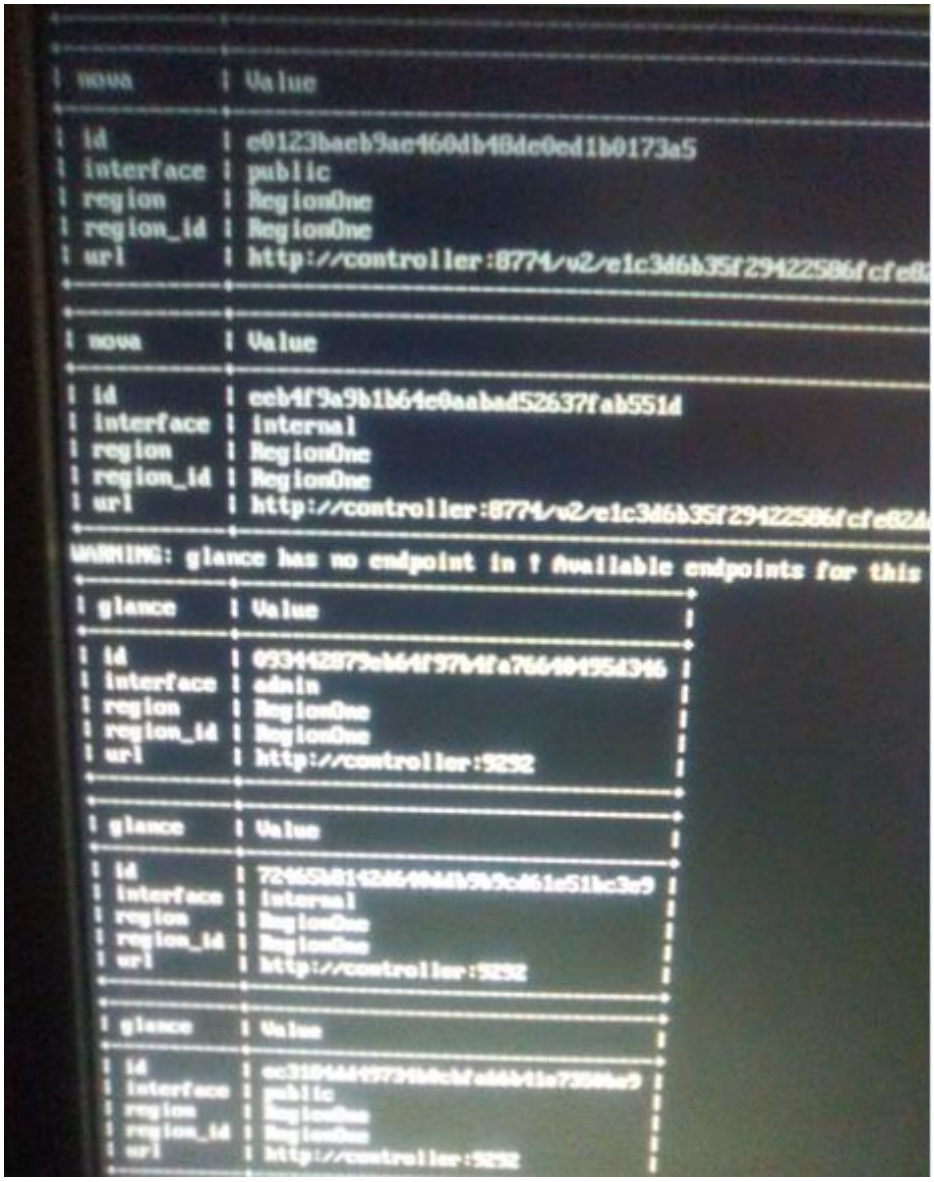


Ilustración 91. Listado de endpoints creados

- Por último, solicitamos el catálogo de imágenes del servicio de imagen para verificar la conectividad con el servicio de imagen (Recordamos que creamos anteriormente una imagen del sistema CirrOS en el servicio de imagen):

```
$ nova image-list
```

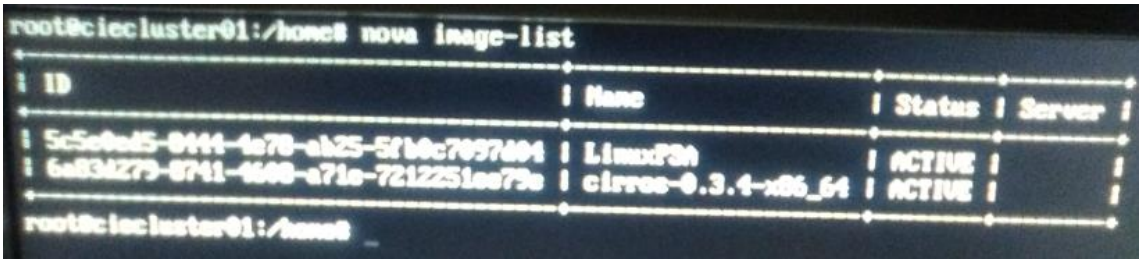


Ilustración 92. Listado de imágenes disponibles a través de Nova

3.4.5. Servicio Neutron (Networking)

En esta sección explicaremos como instalar y configurar el servicio de red Neutron.

Existen 2 posibilidades para instalar un servicio de red en nuestro entorno OpenStack: el servicio OpenStack Networking (Neutron) y el servicio Legacy Network (nova-network). Nova-network se encuentra obsoleto, y Neutron ofrece más opciones y posibilidades, además de ajustarse también a nuestras necesidades, así que nos quedamos con Neutron.

Conceptos

Neutron nos permite crear y asignar interfaces gestionadas por otros servicios OpenStack. Se pueden implementar plugins adicionales para hacer posibles diferentes configuraciones y equipamientos de red y software, proporcionando flexibilidad a la arquitectura de OpenStack. Este servicio interactúa principalmente con el servicio de cómputo Nova para proveer redes y conectividad a las instancias virtuales.

Incluye los siguientes componentes:

- ✓ Neutrón-server: Recibe y redirige solicitudes API al plugin de red OpenStack correspondiente para su ejecución.
- ✓ Plugins y agentes de red OpenStack: Activan y desactivan puertos, crean redes y subredes, y proveen direccionamiento IP. Estos plugins y agentes se diferencian dependiendo del vendedor y de las tecnologías usadas en una determinada nube. El servicio de red de OpenStack distribuye plugins y agentes para switches físicos y virtuales de Cisco, productos de NEC OpenFlow, Open vSwitch, puenteo Linux, y productos NSX VMware. Los agentes y plugins más comunes son L3 (Layer 3), DHCP (Dynamic Host IP addressing), entre otros.
- ✓ Cola de mensajes: Componente usado por la mayoría de instalaciones que incluye el servicio Neutron para redirigir la información entre el componente neutron-server y varios agentes, y también usado como una base de datos para guardar estados de red para ciertos plugins.

El servicio Neutron controla todas las facetas de red para la Infraestructura de Red Virtual (VNI en inglés) y la Infraestructura de Red Física (PNI) en nuestro entorno OpenStack. Habilita a los proyectos la posibilidad de crear topologías virtuales de red avanzadas, para las que se incluyen servicios como cortafuegos, balanceadores de carga, y redes privadas virtuales (VPN).

El servicio de red proporciona las redes, subredes y abstracciones de objetos en routers. Cada abstracción tiene ciertas funcionalidades: las redes contienen subredes, y los routers encaminan el tráfico entre diferentes subredes y redes.

Cada router tiene una puerta de enlace conectada a una red, y varias interfaces conectadas a unas subredes. Estas subredes pueden acceder a unas máquinas en otras subredes conectadas al mismo router.

En cualquier instalación del servicio de red Neutron existe al menos una red externa. Al contrario que otras redes, la red externa no es sólo una red definida virtualmente. Además, representa una vía de acceso de la red física y externa accesible desde fuera de OpenStack. Las IPs de la red externa son accesibles desde cualquier persona física en la red física ajena. Debido a este motivo, DHCP está desactivado en esta red física.

Además de las redes externas, también existe una o más redes internas. Estas redes internas están conectadas directamente a las máquinas virtuales. Sólo las máquinas virtuales, cualquier red interna, o aquellas subredes conectadas a través de interfaces al mismo router pueden acceder a las máquinas virtuales conectadas a esa red interna directamente.

Para acceder desde la red física a las máquinas virtuales y viceversa, es necesario disponer de routers entre las redes. Cada router tiene una puerta de enlace que está conectada a una red y varias interfaces conectadas a subredes. Como un router físico, las subredes pueden acceder a las máquinas de otras subredes conectadas al mismo router, y dichas máquinas pueden acceder a la red física de fuera a través de la puerta de enlace del router.

Adicionalmente, se pueden reservar IPs en la red externa para asignarlas a los puertos de la red interna. En el momento que algún dispositivo realice una conexión a una subred, se crea un puerto. Se pueden asociar direcciones IP de la red externa con los puertos a ciertas máquinas virtuales. De esta manera, los dispositivos en la red física pueden acceder a las máquinas virtuales.

El servicio de red soporta también reglas de seguridad, también llamadas grupos de seguridad. Estos grupos habilitan a los administradores la posibilidad de definir reglas de cortafuegos. Una máquina virtual puede pertenecer a uno o más grupos de seguridad, y el servicio de red Neutron aplica las reglas en esos grupos para bloquear o desbloquear puertos, rangos de puertos, o regular el tráfico para esa máquina virtual.

Instalación y configuración (NODO CONTROL)

A. Prerequisitos

Antes de instalar el servicio de imagen, se debe crear una base de datos, credenciales para este servicio, y un endpoint API.

- Para crear la base de datos:

    a. Usar el cliente de acceso a bases de datos para conectarse al servidor de bases de datos utilizando el usuario root:

    ```
    $ mysql -u root –p
    ```

    b. Crear la base de datos neutron:

    ```
    CREATE DATABASE neutron;
    ```

    c. Conceder acceso a la base de datos neutron:

    ```
    GRANT ALL PRIVILEGES ON neutron.* TO
    'neutron'@'localhost' IDENTIFIED BY 'NEUTRON_DBPASS';
    GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%'
    IDENTIFIED BY 'NEUTRON_DBPASS';
    ```

Reemplazar *NEUTRON_DBPASS* por una contraseña deseada.

d. Salir del cliente de la base de datos.

- Cargar las credenciales de admin a través del script que creamos anteriormente:

```
$ source admin-openrc.sh
```

- Para crear las credenciales del servicio hacemos lo siguiente:

a. Creamos el usuario neutron:

```
$ openstack user create --password-prompt neutron
User Password:
Repeat User Password:
```

Ilustración 93. Creación de usuario neutron

b. Añadir el rol admin al usuario neutron y al Proyecto servicio:

```
$ openstack role add --project service --user neutron admin
```

c. Crear la entidad de servicio neutron:

```
$ openstack service create --name neutron --description
"OpenStack Networking" network
```

Ilustración 94. Creación de entidad de servicio neutron

d. Crear el endpoint API del servicio de imagen:

```
$ openstack endpoint create
--publicurl http://controller:9696
--internalurl http://controller:9696
--adminurl http://controller:9696
--region RegionOne network
```

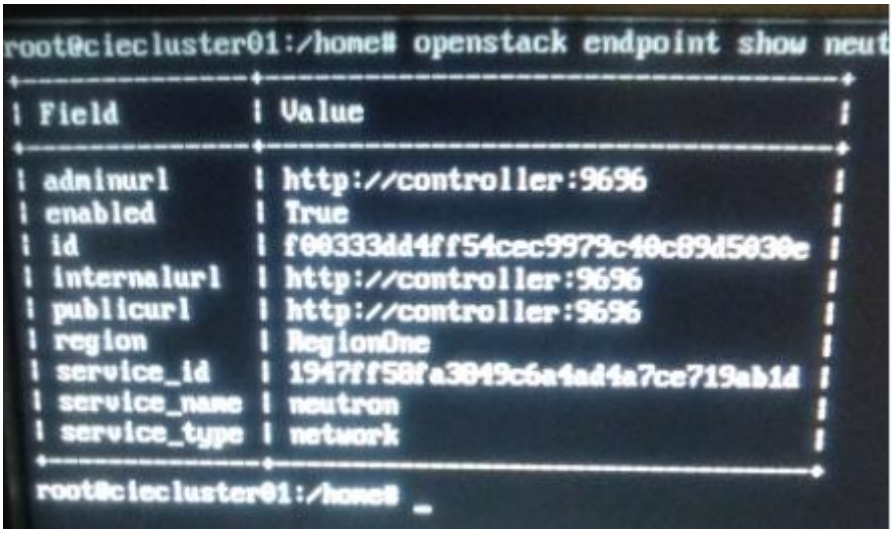


Ilustración 95. Creación de endpoint neutron

B. Componentes del servicio

La configuración del servicio de red incluye la base de datos, mecanismo de autenticación, cola de mensajes, notificaciones de cambios en la topología de red, y plugins.
Como en los otros servicios, los archivos de configuración por defecto pueden variar según la versión, por lo que podría ser necesario añadir las secciones que se indicarán a continuación en caso de que no existan previamente.

- Instalamos los siguientes paquetes:

```
# apt-get install neutron-server neutron-plugin-ml2 python-neutronclient
```

- Editamos el archivo /etc/neutron/neutron.conf y realizamos las siguientes configuraciones:

a. En la sección [database], configuramos el acceso a la base de datos:

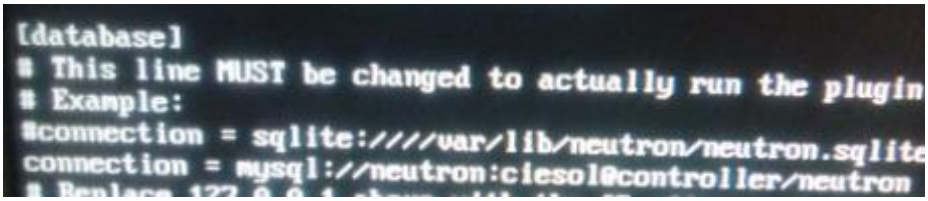


Ilustración 96. Conexión a la base de datos neutron

b. En las secciones [DEFAULT] y [oslo_messaging_rabbit], configurar el acceso a la cola de mensajes RabbitMQ:

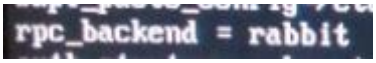


Ilustración 97. Habilitando RabbitMQ

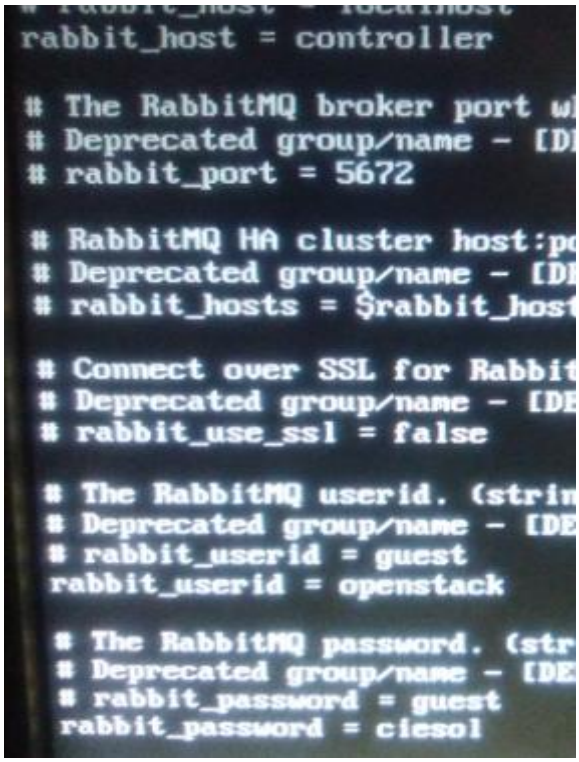


Ilustración 98. Parámetros de RabbitMQ

c. En las secciones [DEFAULT] y [keystone_authtoken], configurar el acceso al servicio de identidad:

```
auth_strategy = keystone
```

Ilustración 99. Configurando el acceso a keystone

```
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = ciesol
#identity_uri = http://127.0.0.1:5000
#admin_tenant_name = %SERVICE_TENANT_NAME%
#admin_user = %SERVICE_USER%
#admin_password = %SERVICE_PASSWORD%
```

Ilustración 100. Sección [keystone_authtoken]

d. En la sección [DEFAULT], habilitar el plugin Modular Layer (ML2), el servicio router y el solapamiento de direcciones IP:

Ilustración 101. Habilitación de plugin ML2

Ilustración 102. Habilitando solapamiento de IPs

Ilustración 103. Habilitando servicio router

e. En las secciones [DEFAULT] y [nova], configurar el servicio de red para que notifique al servicio de cómputo de cambios en la topología de red:

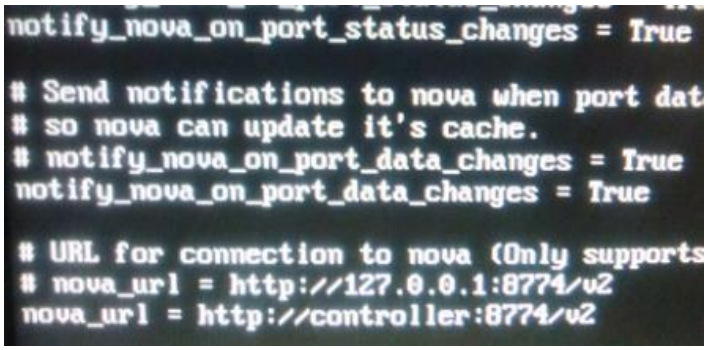


Ilustración 104. Sección [DEFAULT]

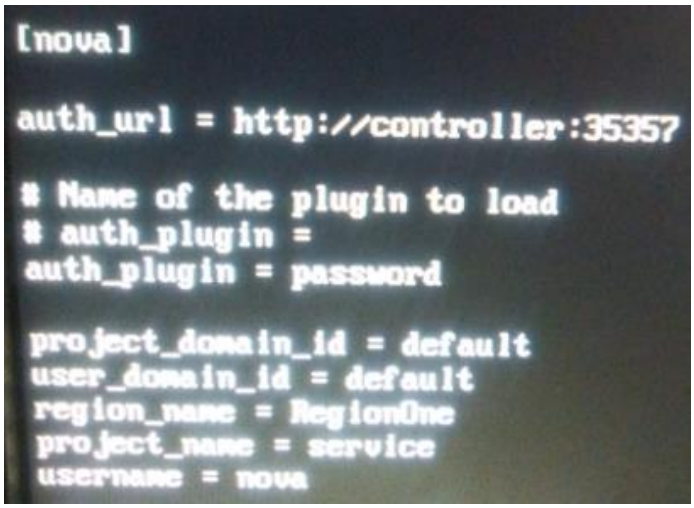


Ilustración 105. Sección [nova]

f. (Opcional) Para ayudar a la solución de posibles problemas, habilitar los logs en la sección [DEFAULT]:

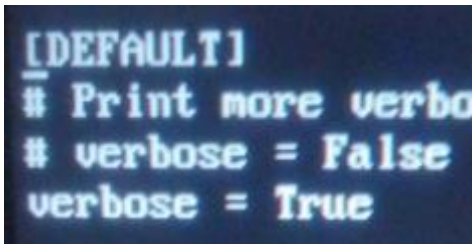


Ilustración 106. Habilitando logs para errores

C. Configuración del plugin Modular Layer 2 (ML2)

El plugin ML2 utiliza el agente Open vSwitch (OVS) para crear el framework de red virtual para las instancias. Sin embargo, el nodo de control no necesita los componentes de OVS debido a que no controla el tráfico de red de las máquinas virtuales.

- Editamos el archivo /etc/neutron/plugins/ml2/ml2_conf.ini y realizamos las siguientes configuraciones:

  a. En la sección [ml2], habilitar en el parámetro “type_drivers”: flat, VLAN, encapsulación genérica de routing (GRE), y LAN virtual extensible (VXLAN); en el parámetro tenant_network_types: gre, y habilitar el mecanismo OVS:

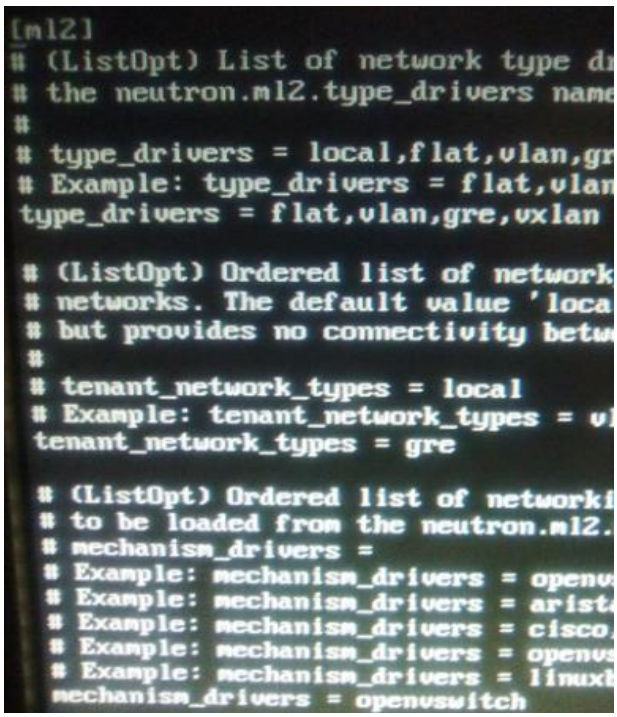


Ilustración 107. Sección [ml2]

b. En la sección [ml2_type_gre], configurar el rango de ids para túnel:

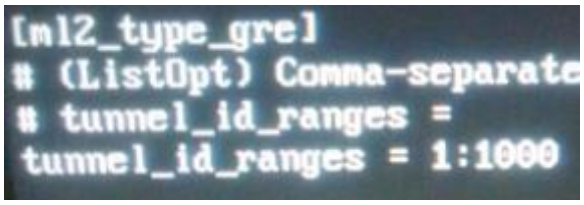


Ilustración 108. Sección [ml2_type_gre]

c. En la sección [security_group], habilitar los grupos de seguridad, habilitar ipset, y configurar el driver de OVS para Iptables:

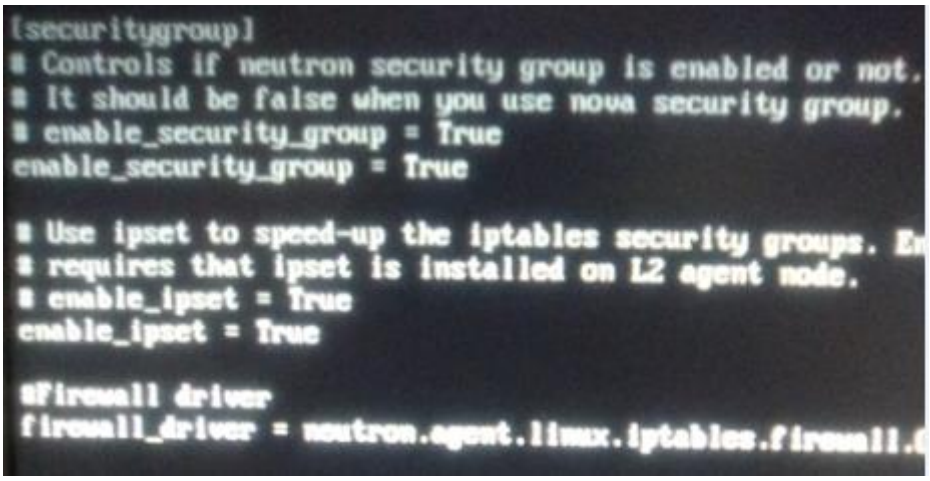


Ilustración 109. Sección [security_group]

D. Configurar el servicio de cómputo para utilizar el servicio de red

Por defecto, los paquetes de distribución implementan el servicio de cómputo para utilizarlo con nova-network. Como utilizaremos Neutron en su lugar, se debe reconfigurar el servicio de cómputo Nova para controlar las redes mediante el servicio de red.

- Editamos el archivo /etc/nova/nova.conf en el nodo de control y realizamos las siguientes configuraciones:

    a. En la sección [DEFAULT], configurar las APIs y drivers:

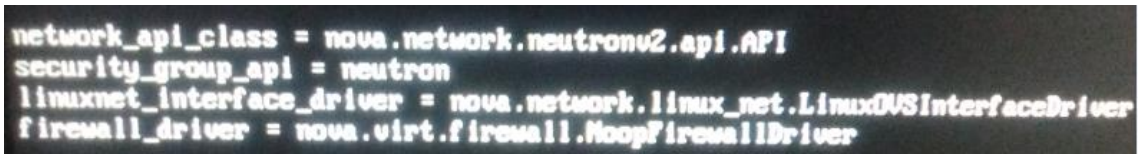


Ilustración 110. Configuración de APIs y drivers.

    b. En la sección [neutron], configuramos los parámetros de acceso:

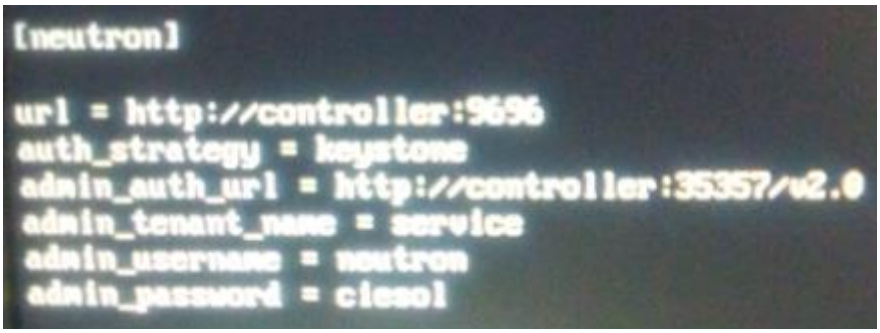


Ilustración 111. Sección [neutron]

E. Finalizando…

- Reflejamos los cambios en la base de datos Neutron:

```
  # su -s /bin/sh -c "neutron-db-manage --config-file
/etc/neutron/neutron.conf --config-file
/etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

- Reiniciamos los procesos del servicio de cómputo:

```
# service nova-api restart
```

- Reiniciamos el servicio de red:

```
# service nova-api restart
```

- (Opcional) Por defecto, los paquetes de Ubuntu crean un archivo de base de datos SQLite. Como esta configuración utiliza una base de datos SQL, podemos eliminar el archivo de SQLite:

```
# rm -f /var/lib/neutron/neutron.sqlite
```

F. Verificando la instalación

Ahora verificaremos que la configuración se ha realizado correctamente mediante unos comandos de comprobación que ejecutaremos en el nodo de control:

- Cargamos el script de variables de entorno para admin:

```
$ source admin-openrc.sh
```

- Solicitamos un listado de extensiones cargadas para asegurar el lanzamiento exitoso de los procesos del servicio de red:

```
$ neutron ext-list
```

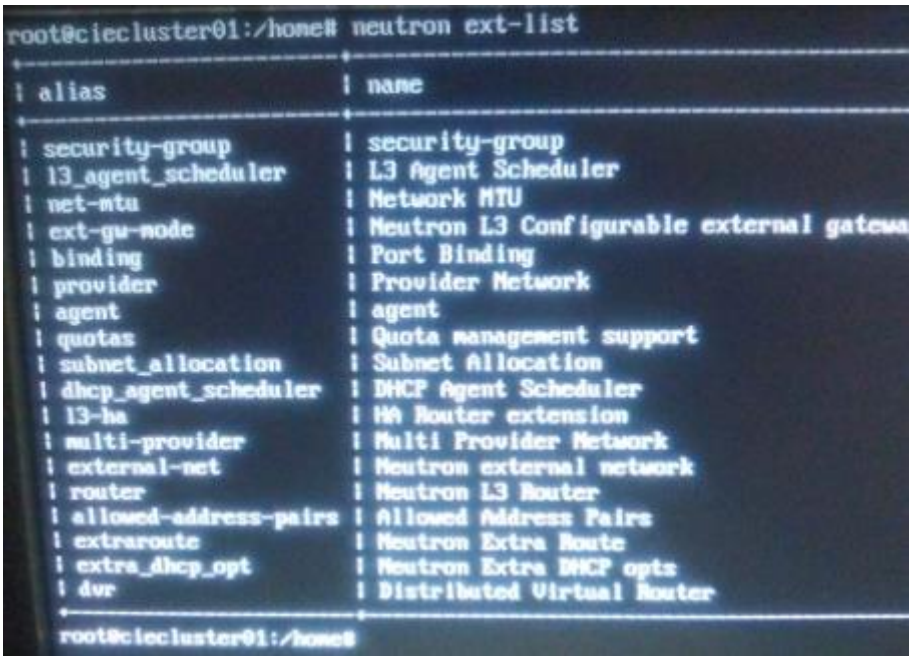


Ilustración 112. Listado de extensiones cargadas

Instalación y configuración (NODO DE RED)

Ahora es el turno de configurar el nodo de red, que se encargará de controlar el tráfico externo e interno, y de los servicios DHCP para la red virtual.

A. Prerequisitos

Antes de configurar el servicio de red en este nodo, es necesario realizar unas configuraciones en los parámetros de red del kernel:

- Editamos el archivo /etc/sysctl.conf y realizamos los siguientes ajustes:

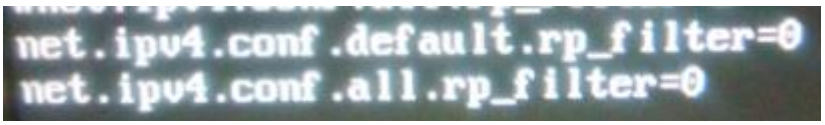


Ilustración 113. Configuración parámetros de red del kernel

- Aplicamos los cambios:

```
# sysctl -p
```

B. Componentes del servicio

A continuación instalaremos los componentes del servicio de red.

Como en los otros servicios, los archivos de configuración por defecto pueden variar según la versión, por lo que podría ser necesario añadir las secciones que se indicarán a continuación en caso de que no existan previamente.

- Instalamos los siguientes paquetes:

```
# apt-get install neutron-plugin-ml2 neutron-plugin-
openvswitch-agent neutron-l3-agent neutron-dhcp-agent neutron-
metadata-agent
```

- Editamos el archivo /etc/neutron/neutron.conf y realizamos las siguientes configuraciones:

    a. En la sección [database] comentamos cualquier parámetro de conexión ya que el nodo no accede a la base de datos.

    b. En las secciones [DEFAULT] y [oslo_messaging_rabbit], configuramos el acceso a la cola de mensajes RabbitMQ:

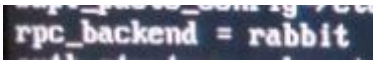


Ilustración 114. Habilitando RabbitMQ

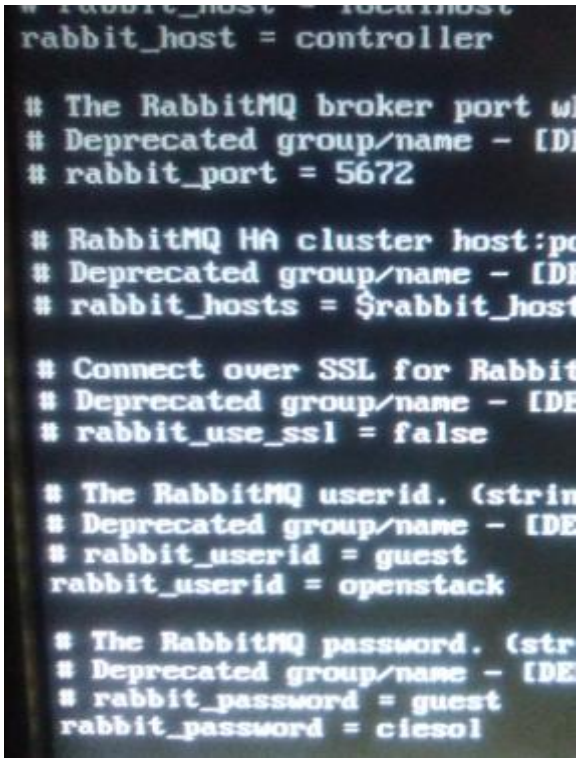


Ilustración 115. Parámetros de RabbitMQ

c. En las secciones [DEFAULT] y [keystone_authtoken], configurar el acceso al servicio de identidad:

Ilustración 116. Configurando acceso a keystone

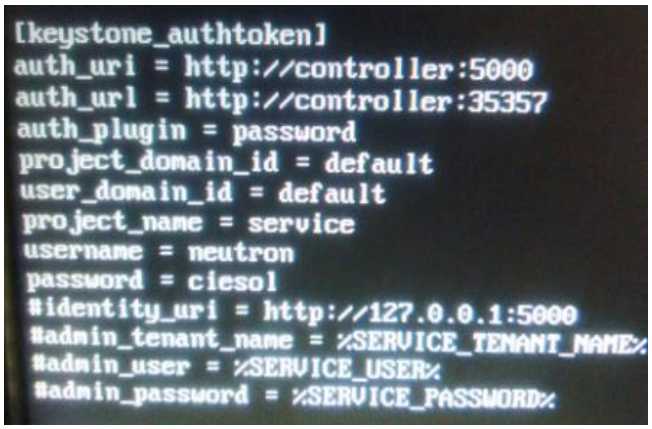


Ilustración 117. Sección [keystone_authtoken]

d. En la sección [DEFAULT], habilitar el plugin ML2, servicio router y el solapamiento de IPs:

Ilustración 118. Habilitando plugin ML2

Ilustración 119. Permitiendo solapamiento de IPs

Ilustración 120. Habilitando servicio router

e. (Opcional) Para asistir a la solución de posibles problemas, habilitar los logs en la sección [DEFAULT]:

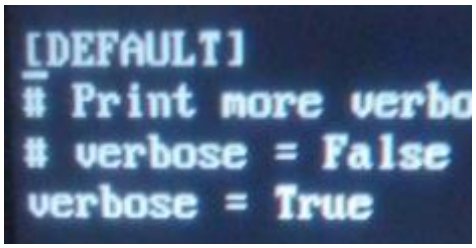


Ilustración 121. Habilitar logs para errores

C. Configuración del plugin ML2

El plugin ML2 utiliza el agente Open vSwitch para crear el framework de red virtual para las máquinas virtuales.

- Editar el archivo /etc/neutron/plugins/ml2/ml2_conf.ini y realizamos las siguientes configuraciones:

    a. En la sección [ml2], habiliar en el parámetro “type_drivers”: flat, VLAN, encapsulación genérica de routing (GRE), y LAN virtual extensible (VXLAN); en el parámetro tenant_network_types: gre, y habilitar el mecanismo OVS:

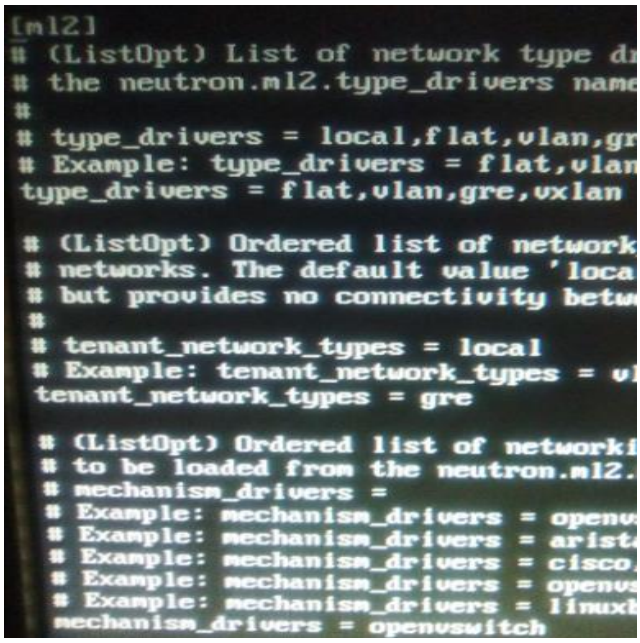


Ilustración 122. Sección [ml2]

    b. En la sección [ml2_type_flat], configurar el proveedor flat como red externa:

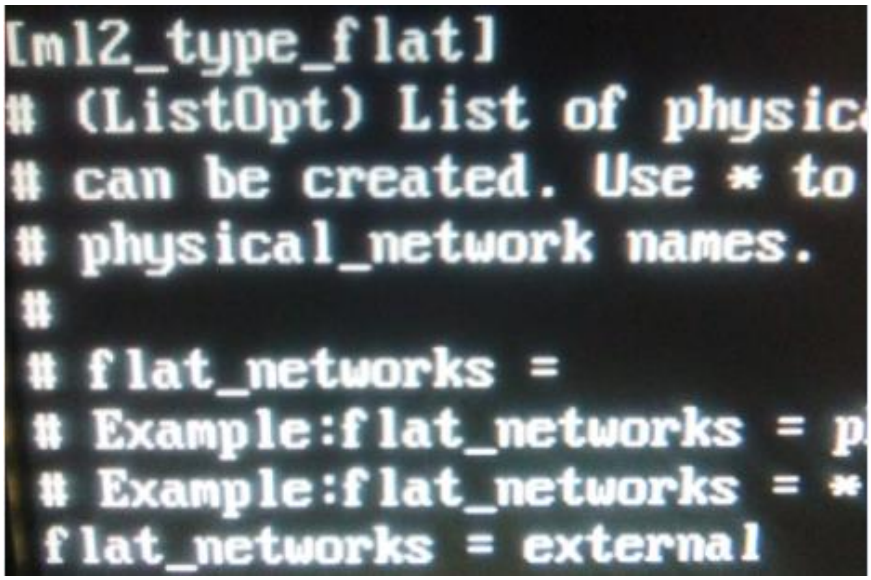


Ilustración 123. Sección [ml2_type_flat]

c. En la sección [ml2_type_gre], configurar el rango de ids túnel:

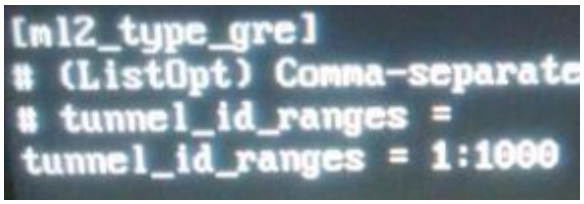


Ilustración 124. Configuración de rango de ids

d. En la sección [security_group], habilitar los grupos de seguridad, habilitar ipset, y configurar el driver de OVS para Iptables:

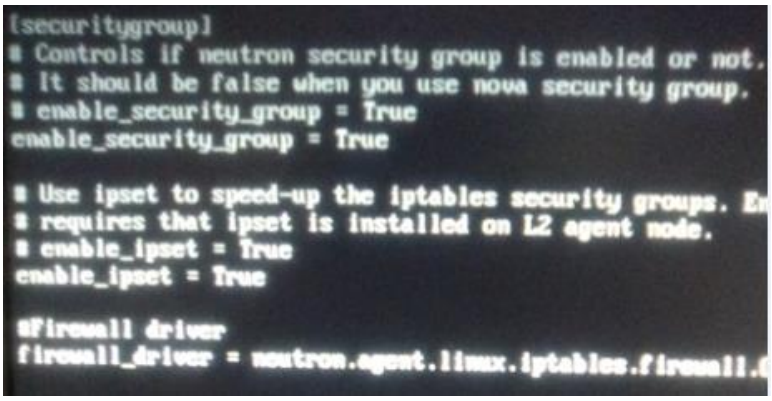


Ilustración 125. Configuración de grupo de seguridad

e. En la sección [ovs], habilitar tunelamiento, configurar el endpoint local del túnel, y referenciar flat a la conexión puente de la red externa:

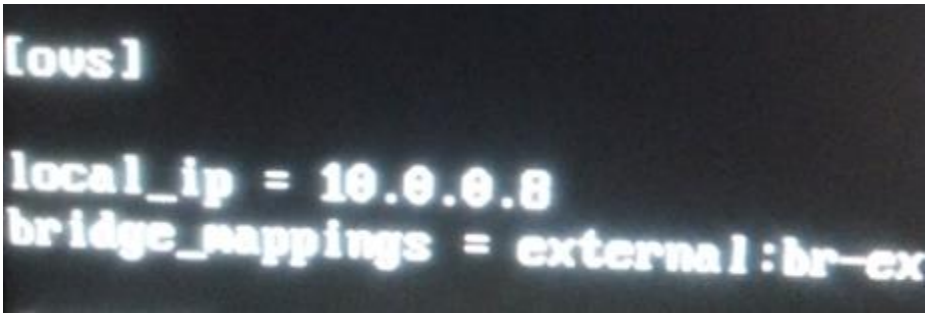


Ilustración 126. Sección [ovs]

f. En la sección [agent], habilitar tunelamiento GRE:

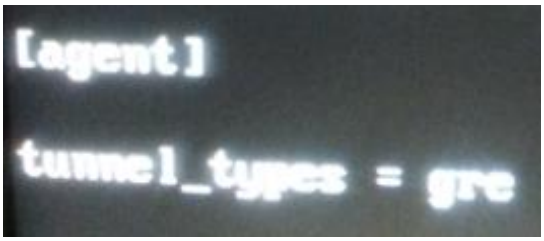


Ilustración 127. Sección [agent]

D. Configurar el agente Layer-3 (L3)

Es necesario configurar este plugin ya que nos proporciona servicios de routing para las redes virtuales.

- Editamos el archivo /etc/neutron/l3_agent.ini y realizamos las siguientes configuraciones:

  a. En la sección [DEFAULT], configurar el driver de interfaz, la conexión puente externa, y habilitamos la detección de espacios de nombres de router extintos:

Ilustración 128. Configurando driver de interfaz

Ilustración 129. Conexión puente externa

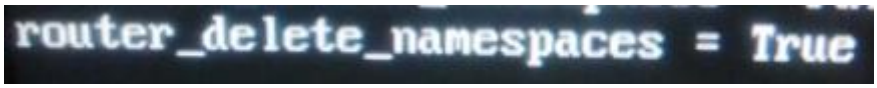


Ilustración 130. Configuración para eliminar espacios de nombres extintos

b. (Opcional) Habilitamos los logs en la sección [DEFAULT] para asistir a la solución de posibles problemas:

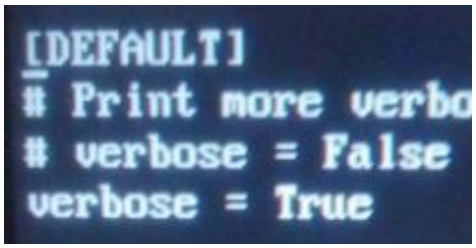


Ilustración 131. Habilitación de logs para errores

E. Configurar el agente DHCP

Debemos configurar el agente DHCP ya que proporciona prestaciones DHCP para las redes virtuales.

- Editar el archivo /etc/neutron/dhcp_agent.ini y realizar las siguientes configuraciones:

    a. En la sección [DEFAULT], configurar la interfaz y los drivers para DHCP, y habilitar la eliminación de espacios de nombres DHCP extintos:

Ilustración 132. Configuración de interfaz driver

Ilustración 133. Configuración para eliminar espacios de nombres DHCP extintos.

    b. (Opcional) Para asistir a la solución de posibles problemas, habilitamos los logs en la sección [DEFAULT]:

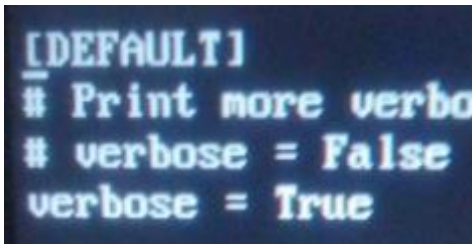


Ilustración 134. Habilitación de logs para errores

- (Opcional) Los protocolos de tunelamiento como GRE incluyen encabezados en los paquetes que incrementan el malgasto de ancho de banda y disminuyen el espacio disponible para datos de usuario. Sin conocimiento de la infraestructura de red virtual, las instancias tienden a enviar paquetes utilizando la máxima unidad de transmisión Ethernet por defecto (MTU), que son 1500 bytes. Las redes de protocolo de Internet (IP) contienen el agente PMTUD para detectar MTU fin-a-fin y ajustar el tamaño del paquete correspondientemente. Sin embargo, algunos sistemas operativos y redes no soportan PMTUD, causando degradación en el rendimiento o fallos de conectividad. Se pueden prevenir estos problemas habilitando “frames jumbo” en la red física que contengan redes virtuales. Los frames jumbo soportan MTUs hasta aproximadamente 9000 bytes que negarían el impacto del malgasto de ancho de banda del GRE en las redes virtuales. El problema es que muchos dispositivos de red no soportan estos frames jumbo, y los administradores de entornos OpenStack no tienen control sobre esta situación.
  Dadas estas complicaciones, se pueden también prevenir este tipo de problemas reduciendo el valor de MTU. Determinar un valor adecuado lleva tiempo y muchos experimentos, pero normalmente un valor de 1454 bytes funciona en la mayoría de entornos. Podemos configurar el servidor DHCP y que asigne direcciones IP a las instancias virtuales para que se ajuste el valor MTU.

    a. Editamos el archivo /etc/neutron/dhcp_agent.ini y realizamos la siguiente configuración:

        i. En la sección [DEFAULT], habilitamos el archivo de configuración “dnsmasq”:

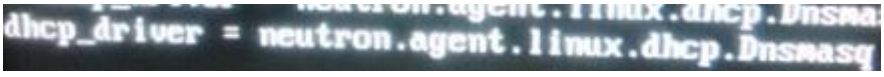


Ilustración 135. Configuración dnsmasq

b. Creamos el archivo /etc/neutron/dnsmasq-neutron.conf y lo editamos de esta manera:

   i. Habilitamos la opción DHCP MTU (26) y la configuramos con el valor 1454 bytes:

Ilustración 136. Configuración DHCP MTU

c. "Matamos" cualquier proceso existente "dnsmasq":

```
# pkill dnsmasq
```

F. Configurar el agente metadata

El agente de metadatos proporciona información acerca de la configuración, como las credenciales de las instancias virtuales:

- Editamos el archivo /etc/neutron/metadata_agent.ini y realizamos las siguientes configuraciones:

  a. En la sección [DEFAULT], configuramos los parámetros de acceso:

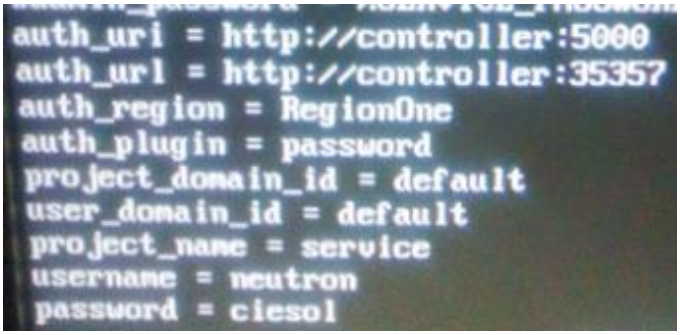


Ilustración 137. Sección [DEFAULT]

b. En la sección [DEFAULT], configurar el host de metadatos:

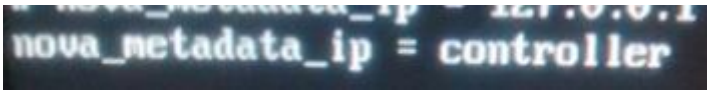


Ilustración 138. Configurando host de metadatos

c. En la sección [DEFAULT], configuramos una palabra secreta para el proxy de metadatos:

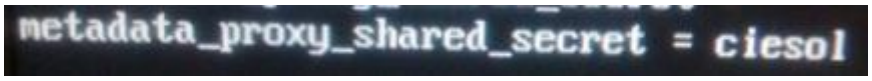


Ilustración 139. Configuración de palabra secreta

d. (Opcional) Como es habitual, podemos habilitar los logs para solucionar posibles problemas:

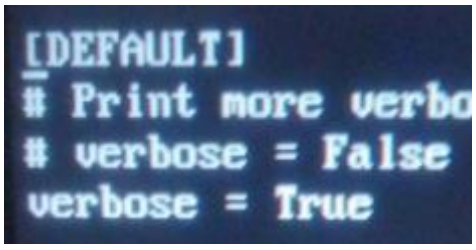


Ilustración 140. Habilitación de logs para errores

- En el NODO DE CONTROL, debemos editar el archivo /etc/nova/nova.conf y realizar los siguientes cambios:

    a. En la sección [neutron], habilitar el proxy de metadatos y configurar la palabra secreta:

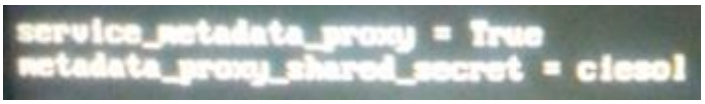


Ilustración 141. Palabra secreta en sección [neutron]

d. Siguiendo en el NODO DE CONTROL, reiniciamos el servicio de cómputo:

```
# service nova-api restart
```

G. Configuración del servicio Open vSwitch (OVS)

Como se ha mencionado anteriormente, el servicio OVS proporciona el framework para las redes virtuales. La interfaz puente "br-int" controla el tráfico interno de las instancias virtuales en OVS. La interfaz puente "br-ex" controla el tráfico externo de las instancias virtuales también en OVS. El puente externo requiere un puerto en la interfaz de la red física para proporcionar a las instancias conexión a la red externa. En resumen, este puerto conecta las redes externas virtuales y físicas a nuestro entorno.

- Reiniciamos el servicio OVS:

```
# service openvswitch-switch restart
```

- Añadimos la interfaz puente externa:

```
# ovs-vsctl add-br br-ex
```

- Añadimos un puerto a la interfaz puente que se conecta a la interfaz de la red externa física, en nuestro caso la (INTERFAZ):

```
# ovs-vsctl add-port br-ex INTERFACE_NAME
```

Dependiendo de nuestra interfaz de red, podría ser necesario desactivar la característica GRO para lograr una conexión viable entre las instancias y la red externa:

```
# ethtool -K INTERFACE_NAME gro off
```

H. Finalizando…

1. Reiniciamos los servicios de red:

```
# service neutron-plugin-openvswitch-agent restart
# service neutron-l3-agent restart
# service neutron-dhcp-agent restart
# service neutron-metadata-agent restart
```

I. Verificar instalación

Debemos verificar que la configuración que hemos hecho es correcta. Para ello, debemos ejecutar unos comandos de comprobación en el nodo de control:

- Cargamos el script de variables de entorno para admin:

```
$ source admin-openrc.sh
```

- Solicitamos el listado de agentes del servidor de red para verificar que están en ejecución y funcionando correctamente:

```
$ neutron agent-list
```

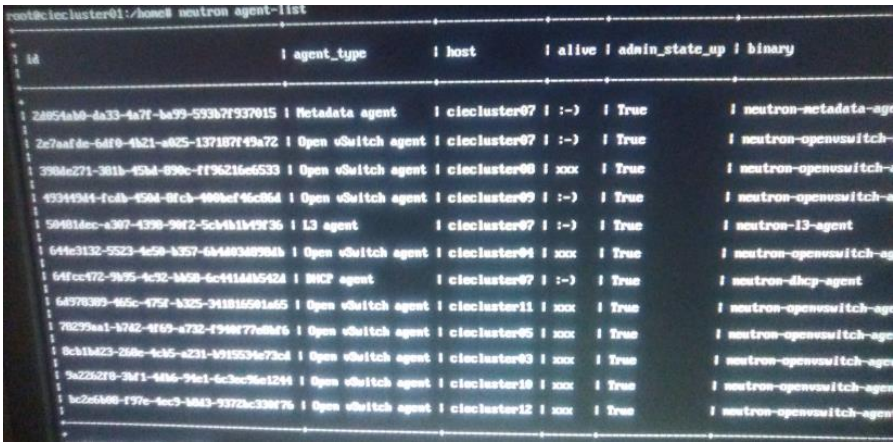


Ilustración 142. Listado de plugins activos

Instalación y configuración (NODOS CÓMPUTO)

A. Prerequisitos

Antes de configurar el servicio de red en este nodo, es necesario realizar unas configuraciones en los parámetros de red del kernel:

- Editamos el archivo /etc/sysctl.conf y realizamos los siguientes ajustes:

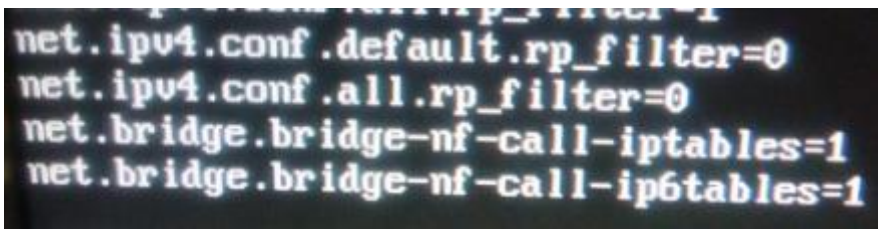


Ilustración 143. Configuración de sysctl

- Aplicamos los cambios:

```
# sysctl -p
```

B. Componentes del servicio

A continuación instalaremos los componentes del servicio de red.

Como en los otros servicios, los archivos de configuración por defecto pueden variar según la versión, por lo que podría ser necesario añadir las secciones que se indicarán a continuación en caso de que no existan previamente.

- Instalamos los siguientes paquetes:

```
# apt-get install neutron-plugin-ml2 neutron-plugin-
openvswitch-agent
```

- Editamos el archivo /etc/neutron/neutron.conf y realizamos las siguientes configuraciones:

    a. En la sección [database] comentamos cualquier parámetro de conexión ya que el nodo no accede a la base de datos.

    b. En las secciones [DEFAULT] y [oslo_messaging_rabbit], configuramos el acceso a la cola de mensajes RabbitMQ:

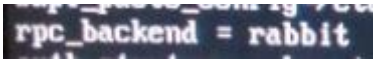


Ilustración 144. Habilitación de RabbitMQ

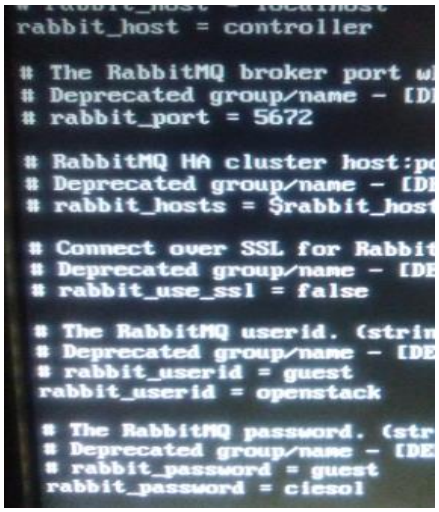


Ilustración 145. Parámetros de RabbitMQ

c. En las secciones [DEFAULT] y [keystone_authtoken], configurar el acceso al servicio de identidad:

Ilustración 146. Configurando acceso a keystone

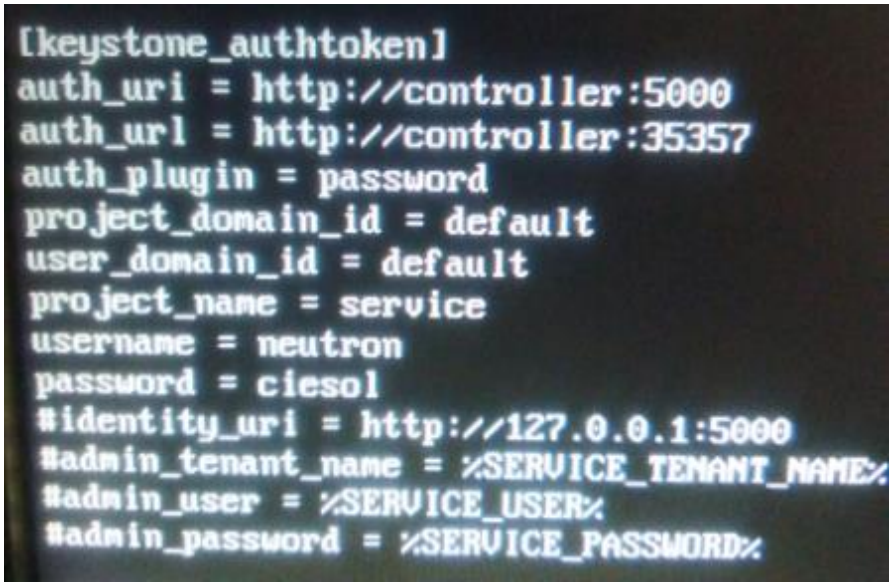


Ilustración 147. Sección [keystone_authtoken]

d. En la sección [DEFAULT], habilitar el plugin ML2, servicio router y el solapamiento de IPs:

Ilustración 148. Habilitando ML2

Ilustración 149. Habilitando solapamiento de IPs

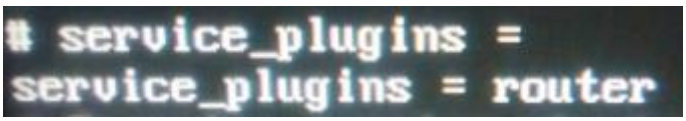


Ilustración 150. Habilitando servicio router

e. (Opcional) Para asistir a la solución de posibles problemas, habilitar los logs en la sección [DEFAULT]:

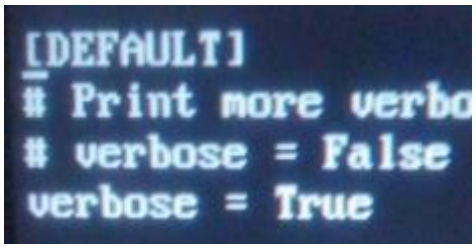


Ilustración 151. Habilitando logs para errores

C. Configuración del plugin ML2

El plugin ML2 utiliza el agente Open vSwitch para crear el framework de red virtual para las máquinas virtuales.

- Editar el archivo /etc/neutron/plugins/ml2/ml2_conf.ini y realizamos las siguientes configuraciones.

a. En la sección [ml2], habiliar en el parámetro “type_drivers”: flat, VLAN, encapsulación genérica de routing (GRE), y LAN virtual extensible (VXLAN); en el parámetro tenant_network_types: gre, y habilitar el mecanismo OVS:

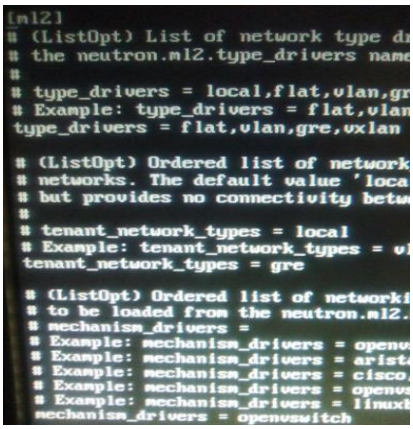


Ilustración 152. Sección [ml2]

b. En la sección [ml2_type_gre], configurar el rango de ids túnel:

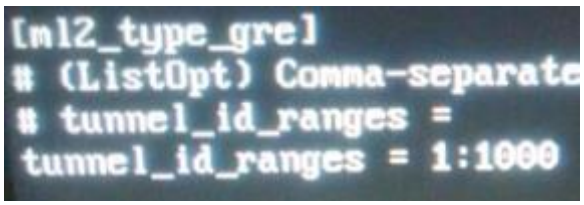


Ilustración 153. Configuración de rango de ids

c. En la sección [security_group], habilitar los grupos de seguridad, habilitar ipset, y configurar el driver de OVS para Iptables:

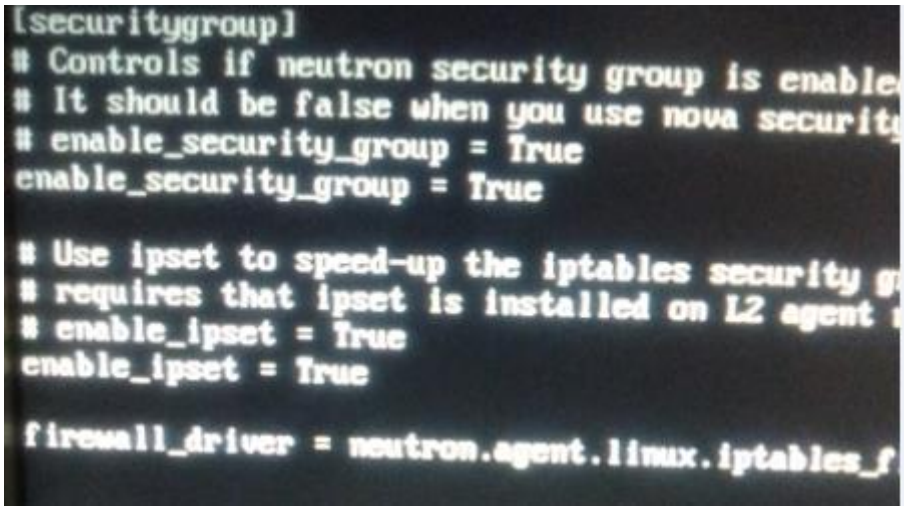


Ilustración 154. Sección [security_group]

d. En la sección [ovs], habilitar tunelamiento y configurar el endpoint local del túnel:

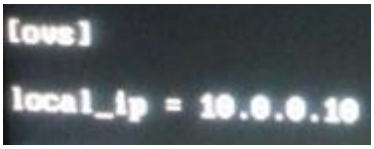


Ilustración 155. Sección [ovs]

e. En la sección [agent], habilitar tunelamiento GRE:

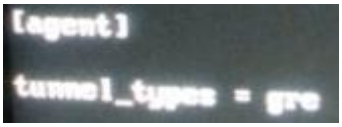


Ilustración 156. Sección [agent]

D. Configuración del servicio Open vSwitch (OVS)

Como se ha mencionado anteriormente, el servicio OVS proporciona el framework para las redes virtuales.

- Reiniciamos el servicio OVS:

```
# service openvswitch-switch restart
```

E. Configurar el servicio de cómputo para utilizar el servicio de red

Por defecto, los paquetes de distribución implementan el servicio de cómputo para utilizarlo con nova-network. Como utilizaremos Neutron en su lugar, se debe reconfigurar el servicio de cómputo Nova para controlar las redes mediante el servicio de red.

- Editamos el archivo /etc/nova/nova.conf en los nodos de cómputo y realizamos las siguientes configuraciones:

    a. En la sección [DEFAULT], configurar las APIs y drivers:

```
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

Ilustración 157. Configuración de APIs y drivers

    b. En la sección [neutron], configuramos los parámetros de acceso:

```
[neutron]

url = http://controller:9696
auth_strategy = keystone
admin_auth_url = http://controller:35357/v2.0
admin_tenant_name = service
admin_username = neutron
admin_password = ciesol
```

Ilustración 158. Sección [neutron]

F. Finalizando…

- Reiniciamos el servicio de cómputo:

```
# service nova-compute restart
```

- Reiniciamos el agente Open vSwitch:

```
# service neutron-plugin-openvswitch-agent restart
```

G. Verificar instalación

Debemos verificar la configuración del servicio de red en los nodos de cómputo para comprobar el buen funcionamiento de dicho servicio. Para ello, ejecutamos unos comandos de comprobación en el nodo de control.

- Cargamos el script de variables de entorno para admin:

```
$ source admin-openrc.sh
```

- Solicitamos el listado de agentes del servicio de red para comprobar que funcionan correctamente en los nodos:

```
$ neutron agent-list
```

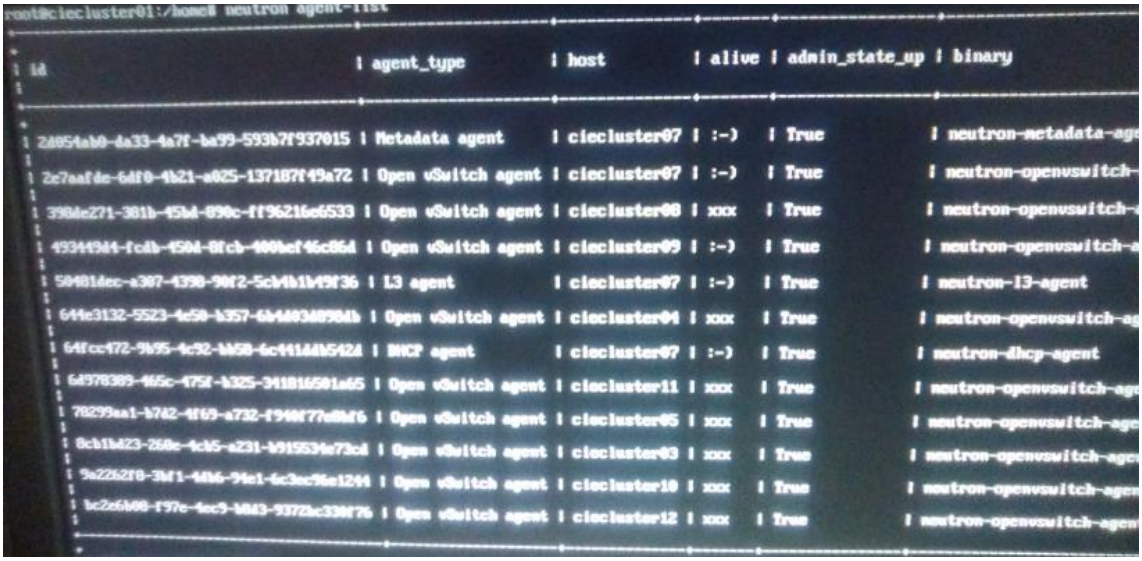


Ilustración 159. Listado de agentes en ejecución

Ahora, deben funcionar también los agentes en los nodos de cómputo, que sería únicamente el OVS en cada uno.

Creación de la infraestructura de red

Una vez instalados los servicios de identidad, imagen, cómputo y red, ya estamos capacitados para lanzar instancias, pero antes de esto, debemos crear nuestra infraestructura de red para brindar conexión a las instancias.

Después de crear las redes, que serán 2, una externa y otra interna para las instancias, debemos verificar su funcionamiento, ya que de no hacerlo ahora, se podrían presentar problemas a la hora de lanzar instancias.

Este es el diseño de la infraestructura de red que utilizaremos en nuestro entorno:

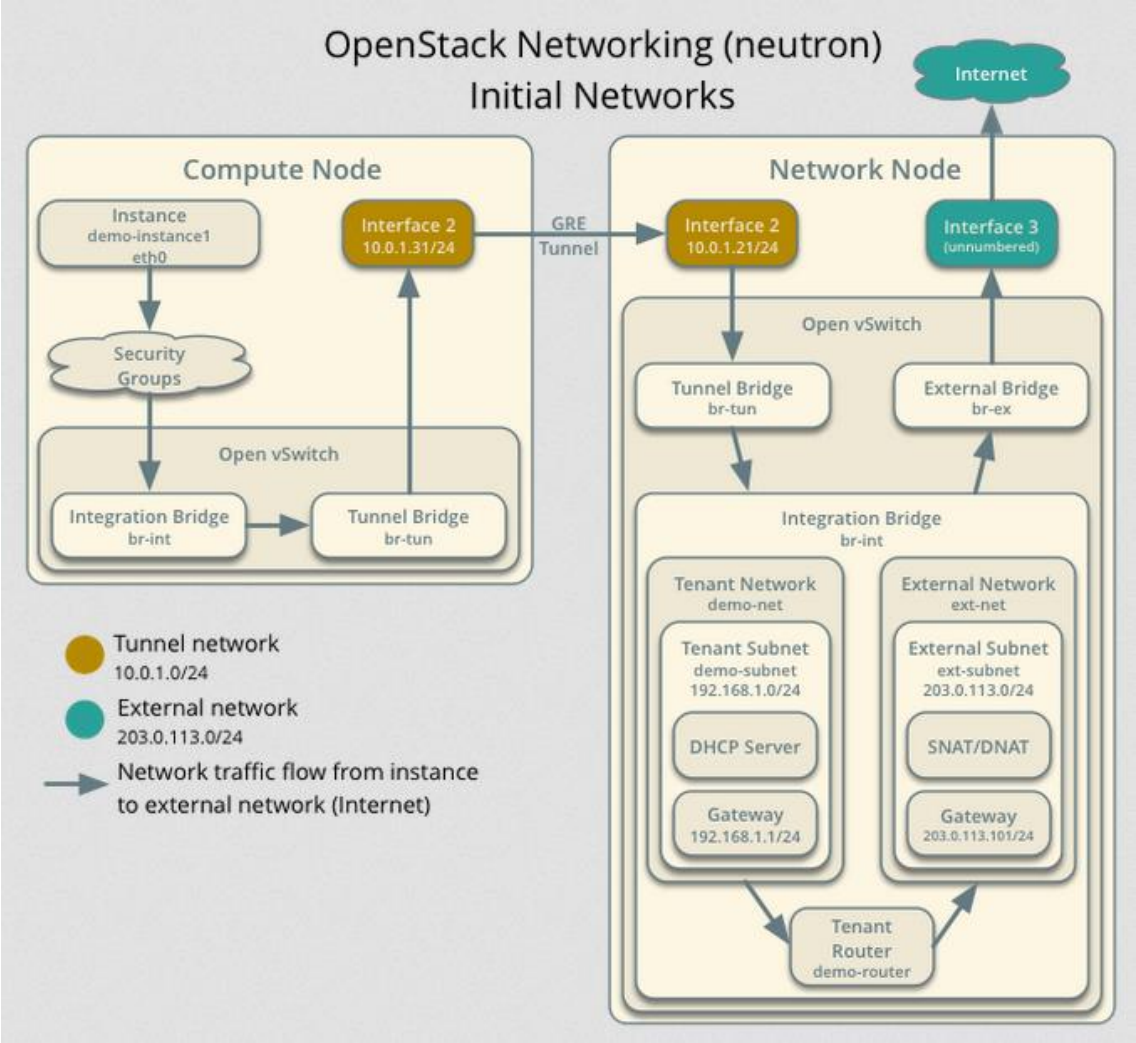


Ilustración 160. Infraestructura de red en OpenStack

Las redes IP mostradas en el diagrama no coinciden con las que se están utilizando en la instalación:

- Nuestra red túnel utilizaría la red 10.0.0.0/24 en lugar de 10.0.1.0/24
- Nuestra red externa utilizaría la red 192.168.137.0/24 en lugar de 203.0.113.0/24

A. Red externa

Empezaremos con la red externa:

- Cargamos el script con las variables de entorno para admin:

```
$ source admin-openrc.sh
```

- Creamos la red externa:

```
$ neutron net-create ext-net --router:external --
provider:physical_network external --provider:network_type flat
```

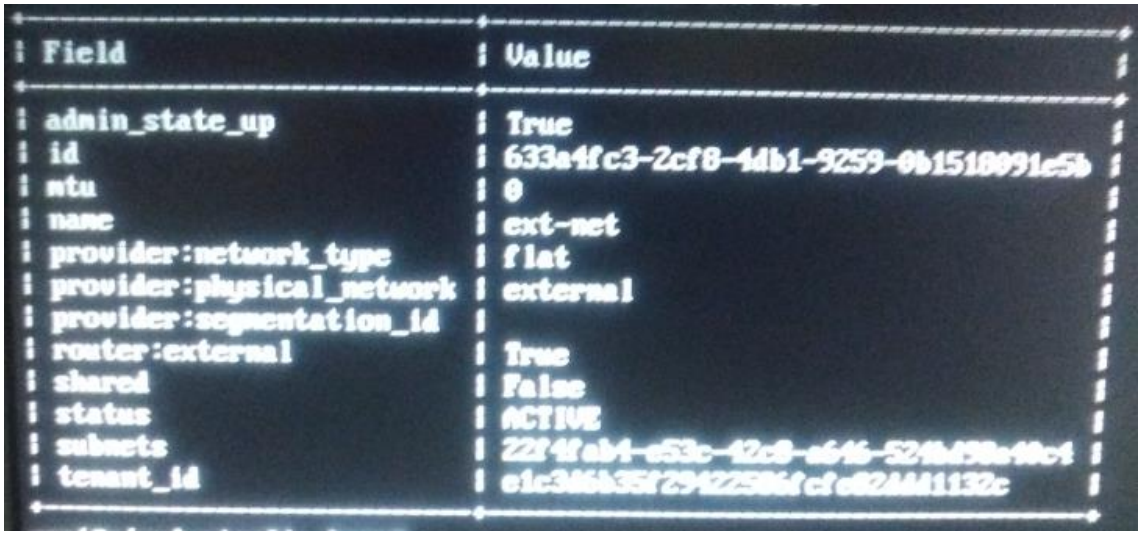


Ilustración 161. Creación de red externa

Como una red física, una red virtual necesita una subred asignada a ella. La red externa comparte la misma subred y puerta de enlace asociada con la red física conectada a la red externa en el nodo de red. Para ello:

- Creamos la subred:

```
$ neutron subnet-create ext-net 192.168.137.0/24 --name ext-subnet --allocation-pool start=192.168.137.100,end=192.168.137.150 --disable-dhcp --gateway 192.168.137.1
```

El rango de direcciones de esta subred pertenece a la red externa desde la que nos conectamos a Internet, es decir, 192.168.137.0/24:

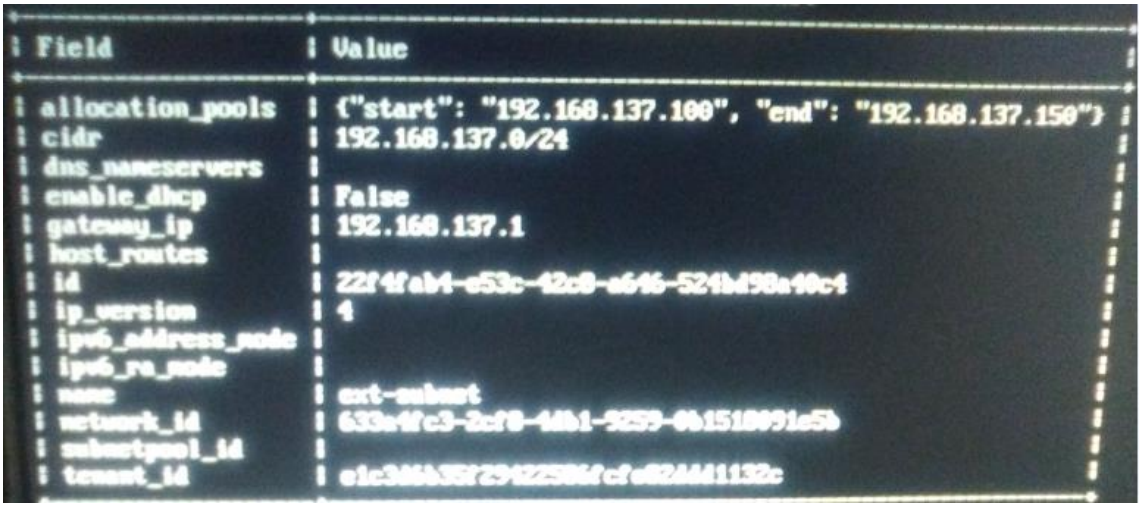


Ilustración 162. Creación de subred externa

B. Red interna para máquinas virtuales (tenant)

Ahora procedemos a configurar la red tenant:

- Cargamos el script con las variables de entorno para admin, si no lo tenemos ya cargado previamente.
- Creamos la red tenant:

```
$ neutron net-create demo-net
```

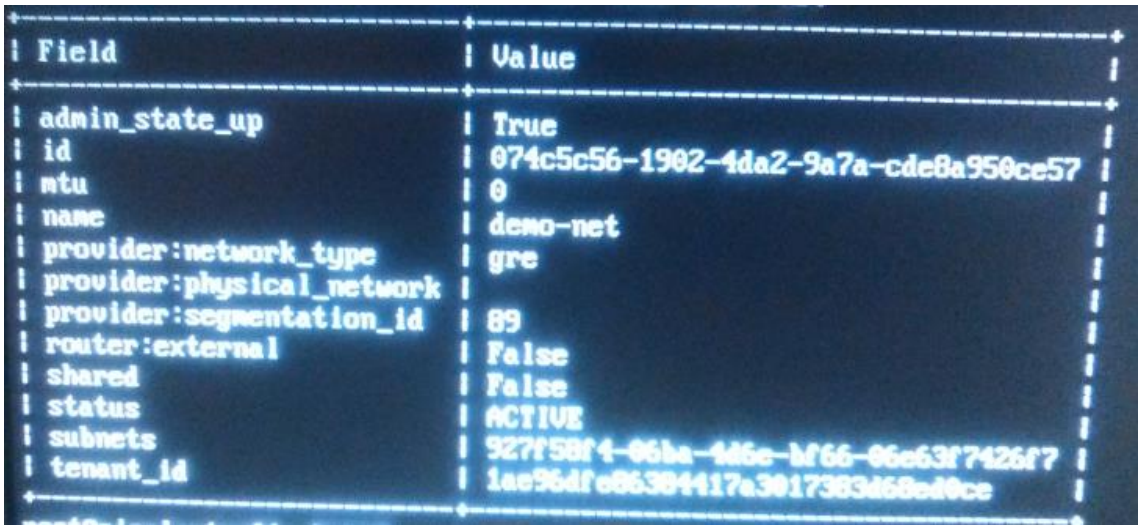


Ilustración 163. Creación de red interna

De manera similar a la red externa, la red tenant necesita una subred asignada a ella. Podemos elegir cualquier red, ya que la arquitectura que tenemos aisla este tipo de redes. Por defecto, esta subred utiliza DHCP para que las instancias puedan adquirir IPs. Para ello:

- Creamos la subred:

```
$ neutron subnet-create demo-net 192.168.1.0/24 --name demo-subnet --dns-nameserver 8.8.4.4 --gateway 192.168.1.1
```

El rango de direcciones de esta subred pertenece a una red cualquiera definida específicamente para eso. Utilizaremos la red 192.168.1.0/24:

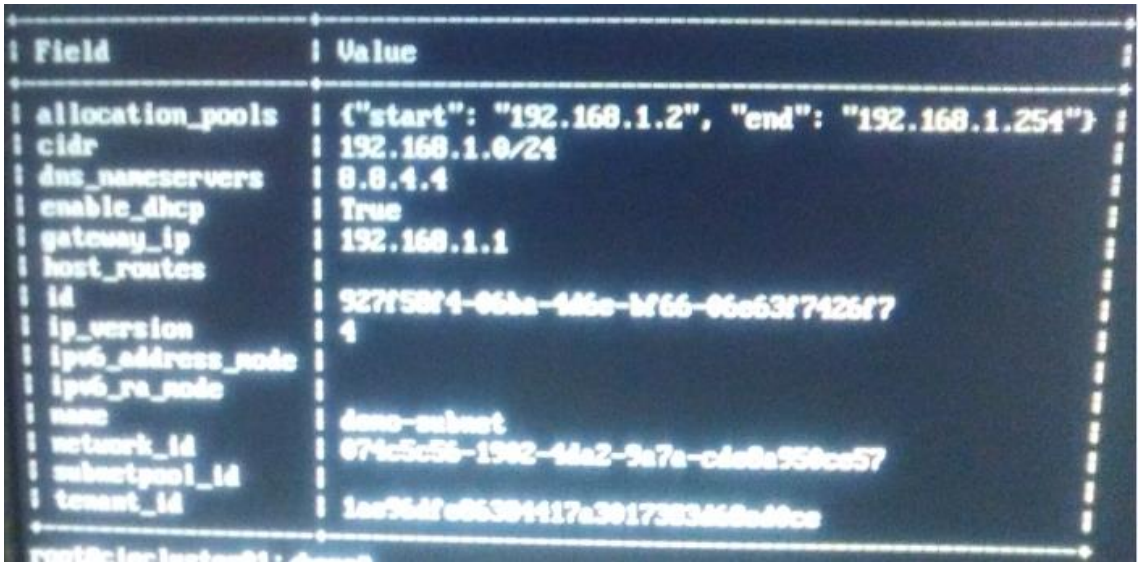


Ilustración 164. Creación de subred interna

Ahora, para poder enlazar la red de las máquinas virtuales y la red externa, debemos utilizar un router virtual. Para ello:

- Creamos el router virtual, cuyo nombre será “demo-router”:

```
$ neutron router-create demo-router
```

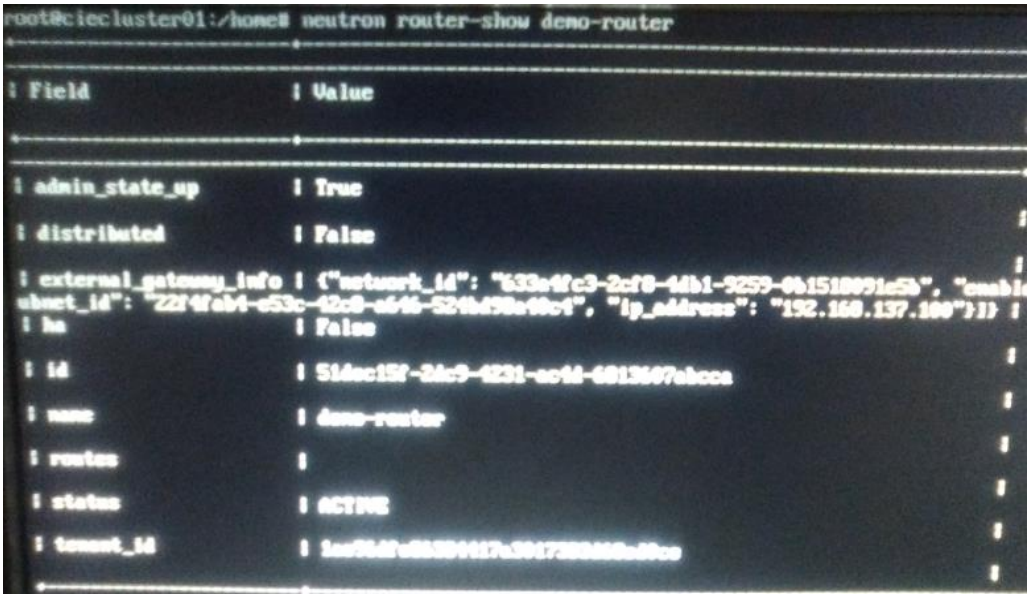


Ilustración 165. Creación de router virtual

- Asociamos el router a la red tenant:

```
$ neutron router-interface-add demo-router demo-subnet
```

- Asociamos el router a la red externa utilizando una puerta de enlace:

```
$ neutron router-gateway-set demo-router ext-net
```

C. Verificar conectividad de las redes

Debemos verificar ahora que las redes están correctamente configuradas, y que existe conectividad entre las redes externa y tenant.

Hacemos ping desde un dispositivo en la red externa hacia la puerta de enlace del router virtual:

```
$ ping -c 4 192.168.137.100
```

Ilustración 166. Ping exitoso al router virtual

3.4.6. Lanzamiento de instancia

Ya estamos capacitados para poder lanzar una instancia de máquina virtual.

Una instancia es una máquina virtual albergada en un nodo de cómputo. Para probar el buen funcionamiento, lanzaremos una instancia de prueba con la imagen CirrOS que tenemos guardada en el servicio de imagen Glance.

A. (Opcional) Generar par de claves

La mayoría de imágenes en la nube soportan autenticación por clave pública con más frecuencia que el mecanismo convencional de usuario y contraseña. Antes de lanzar una instancia, generaremos la posibilidad de autenticarnos con este mecanismo, para brindar algo más de seguridad.

- Cargamos el script de variables de entorno para demo:

```
$ source demo-openrc.sh
```

- Generamos y añadimos el par de claves:

```
$ nova keypair-add demo-key
```

- Verificamos que se ha añadido el par de claves:

```
$ nova keypair-list
```

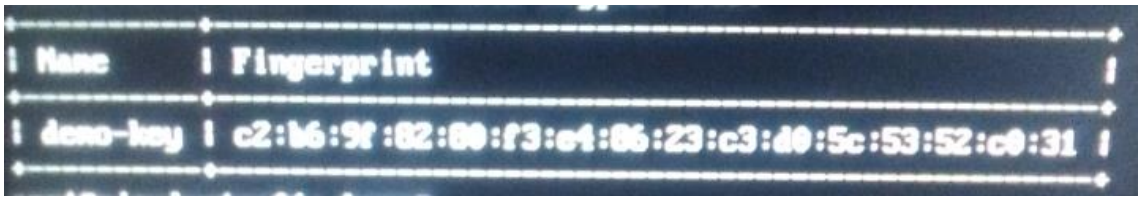


Ilustración 167. Listado de par de claves

B. Lanzando la instancia

Para lanzar una instancia, debemos especificar el flavor, el nombre de la imagen, la red, el grupo de seguridad, las claves, y el nombre de la instancia. Seguiremos unos sencillos pasos para lanzar la instancia de manera sencilla:

- Un flavor es la cantidad de recursos virtuales que destinamos a la instancia (incluye procesador, memoria RAM y almacenamiento). Para ver una lista de perfiles flavor disponibles:

```
$ nova flavor-list
```

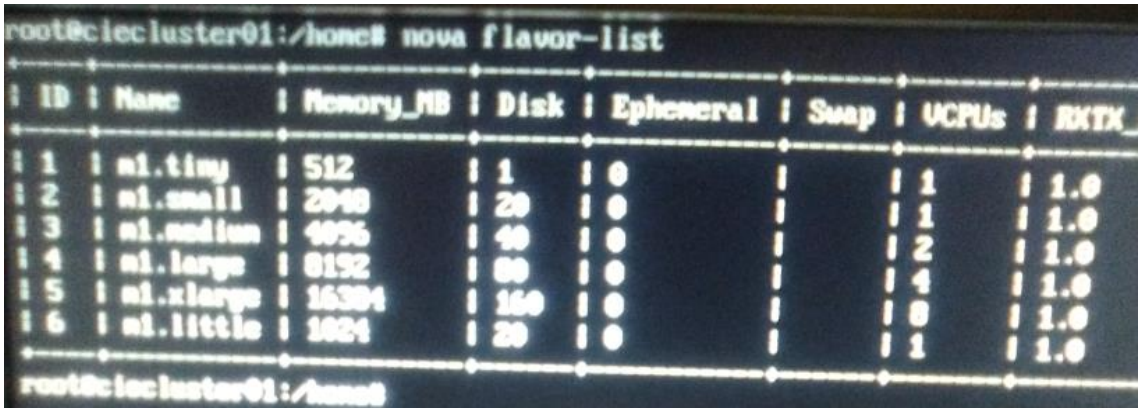


Ilustración 168. Listado de flavor

- Revisamos las imágenes disponibles (debe ser solo 1):

```
$ nova image-list
```

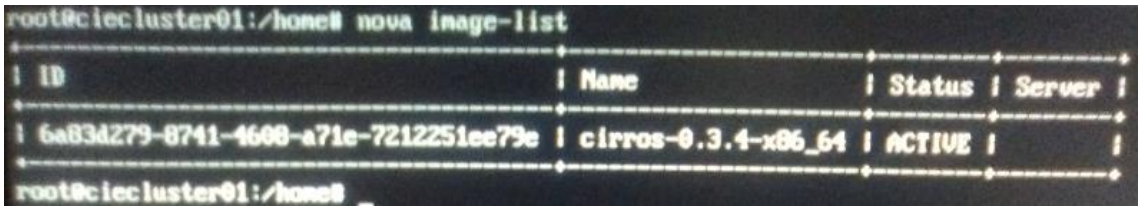


Ilustración 169. Listado de imágenes

- Revisamos las redes disponibles (deben ser 2):

```
$ neutron net-list
```

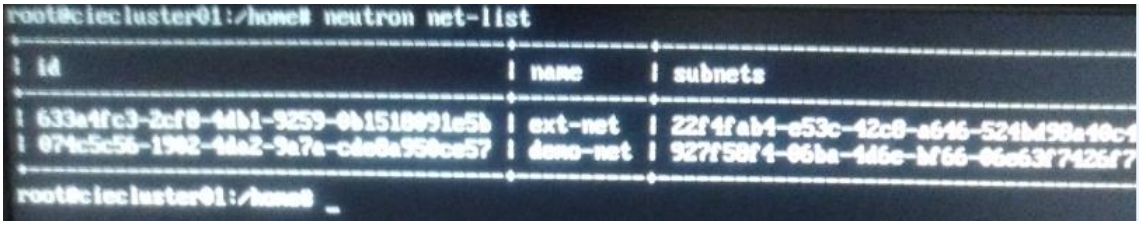


Ilustración 170. Listado de redes

- Revisamos los grupos de seguridad disponibles (debe ser 1):

```
$ nova secgroup-list
```

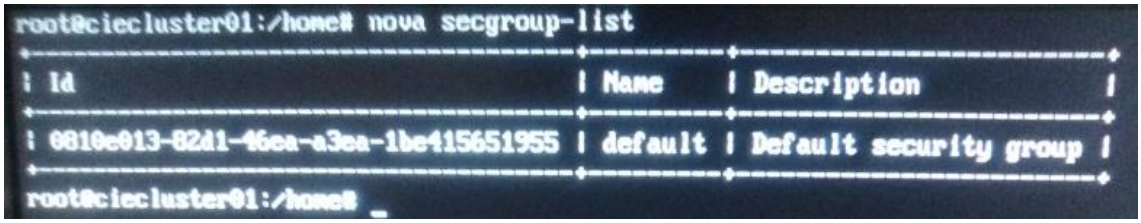


Ilustración 171. Listado de grupos de seguridad.

- Lanzamos la instancia (Utilizaremos la red tenant, el grupo de seguridad por defecto, el par de claves que hemos creado anteriormente, y el flavor tiny):

```
$ nova boot --flavor m1.tiny --image cirros-0.3.4-x86_64 --
nic net-id=DEMO_NET_ID --security-group default --key-name
demo-key demo-instance1
```

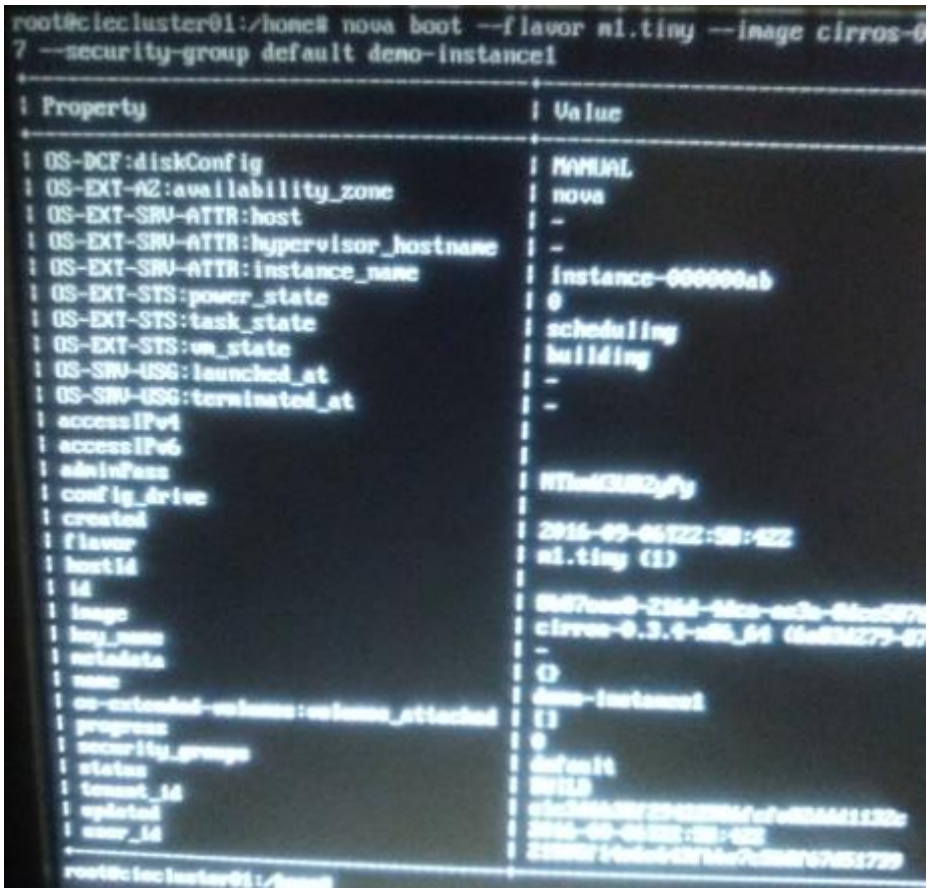


Ilustración 172. Instancia lanzada

- Comprobamos el estado de nuestra instancia:

```
$ nova list
```

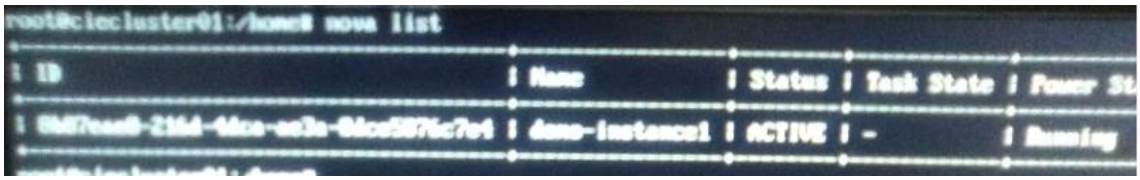


Ilustración 173. Instancia funcionando correctamente

Al principio el estado será BUILD y Spawning. Esto quiere decir que se está creando y generando. Una vez haya terminado de manera correcta, presentará el estado ACTIVE y Running.

C. Acceder a la instancia usando una consola virtual VNC

- Obtenemos una URL para conectarnos a la instancia via web por VNC:

```
$ nova get-vnc-console demo-instance1 novnc
```

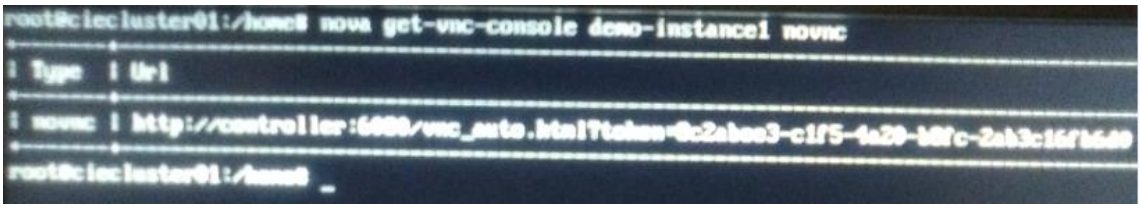


Ilustración 174. URL para acceso a la instancia via web

Si por casualidad el navegador web no puede resolver nombres de host, sustituimos el nombre por su correspondiente dirección IP.

La imagen CirrOS incluye autenticación mediante usuario y contraseña y proporciona dichas credenciales en la pantalla de login. Comprobamos que se puede acceder con éxito a la instancia:

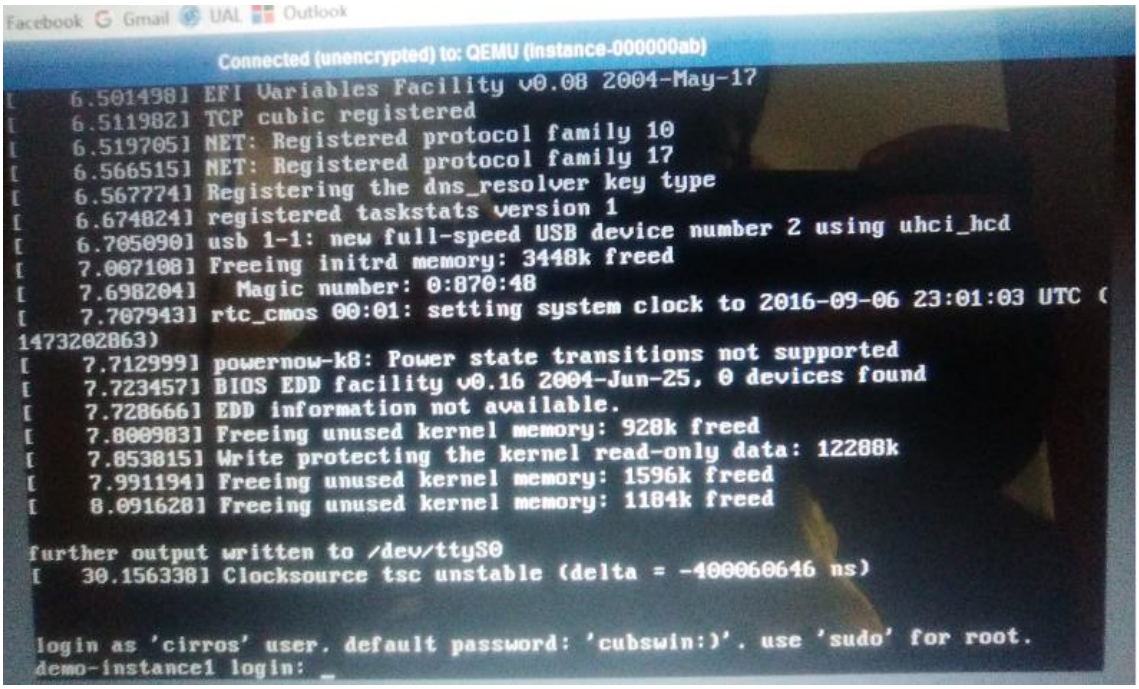


Ilustración 175. Acceso a instancia por VNC

Una vez entremos en la instancia e iniciemos sesión, debemos comprobar la conectividad en la instancia.

Hacemos ping al router virtual:

```
$ ping -c 4 192.168.1.1
```

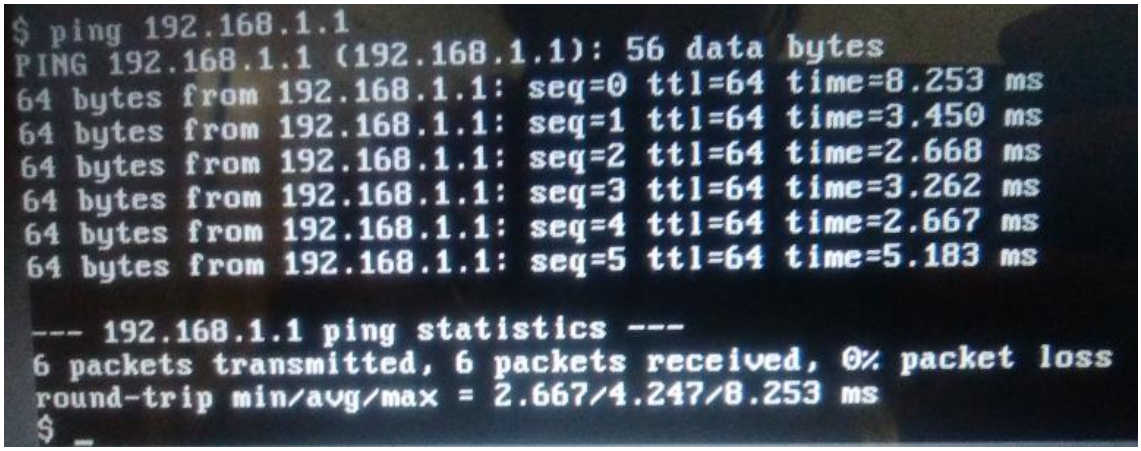


Ilustración 176. Ping desde instancia al router virtual

D. Acceso remoto a la instancia virtual

- Añadimos reglas al grupo de seguridad:

    a. Permitimos el servicio ICMP para poder hacer ping:

```
$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
```

    b. Permitimos el acceso SSH:

```
$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

- Creamos una IP flotante en la red externa:

```
$ neutron floatingip-create ext-net
```

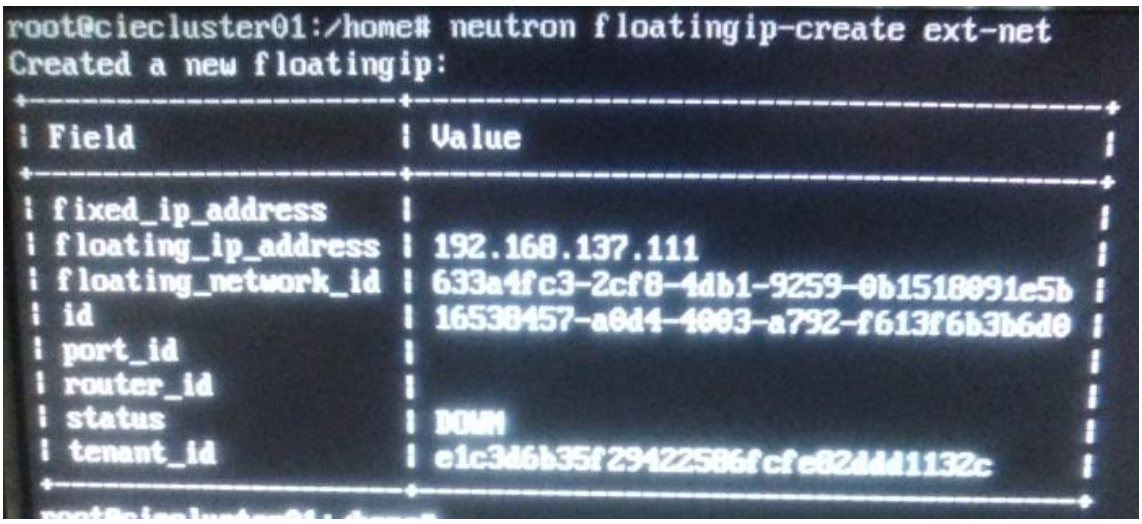


Ilustración 177. Creación de IP flotante

- Asociamos la IP flotante reciente a la instancia:

```
$ nova floating-ip-associate demo-instance1 192.168.137.111
```

- Comprobamos el estado de la IP flotante en la instancia:

```
$ nova list
```

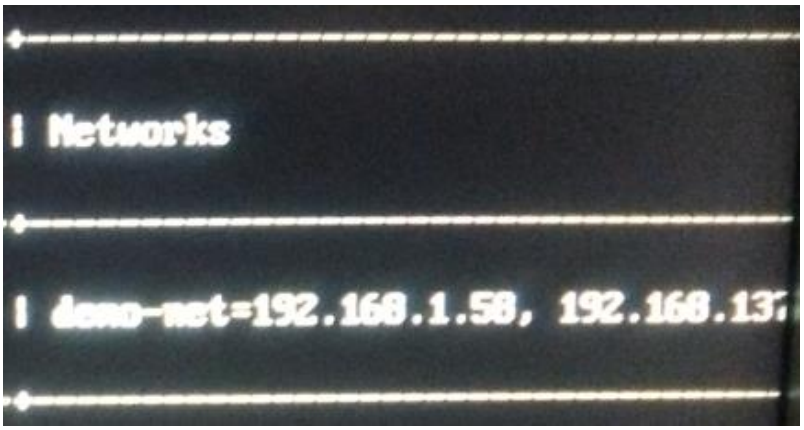


Ilustración 178. Instancia con IP flotante

- Comprobamos la conectividad haciendo ping desde el nodo de control o cualquier host de la red externa a la IP flotante:

```
$ ping -c 4 192.168.137.111
```

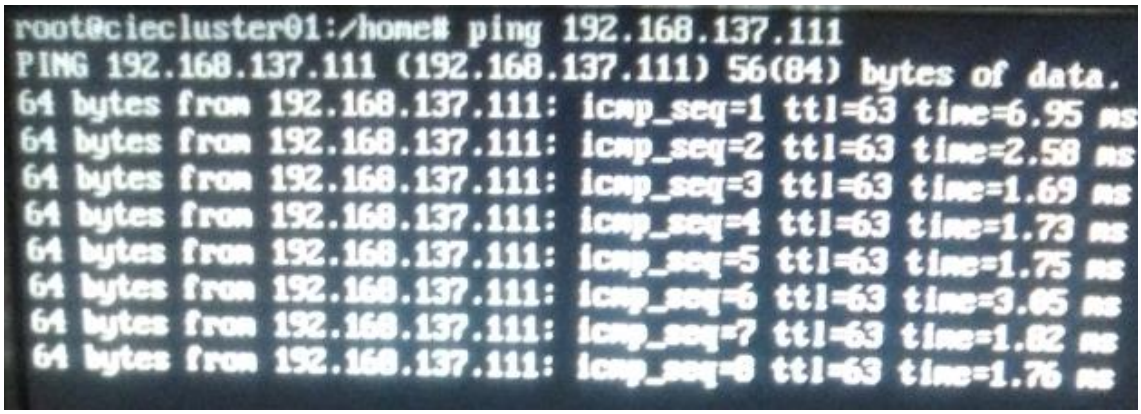


Ilustración 179. Ping exitoso a la instancia

- Por último, podemos acceder a la instancia a través de SSH desde el nodo de control o cualquier host de la red externa:

```
$ ssh cirros@192.168.137.111
```

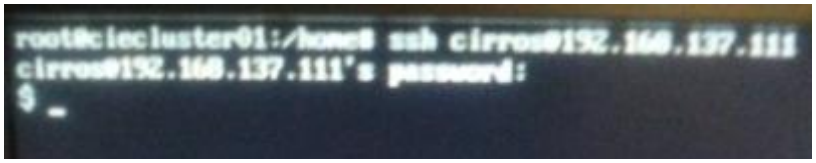


Ilustración 180. Conexión SSH a la instancia

3.5. Creación de instancias de máquinas virtuales que simulan arquitectura x86 con QEMU

Ya tenemos una instancia creada, con conectividad, y accesible a ella de manera remota. De esta manera, podemos crear las instancias que necesitemos para alcanzar los objetivos de nuestro trabajo, siempre limitados por los recursos hardware de nuestro clúster de ordenadores.

Para nuestras máquinas virtuales, necesitamos una imagen de sistema operativo que se adapte a nuestras necesidades, como por ejemplo, una imagen de Ubuntu Server de arquitectura 32 bits con compilador de lenguaje C instalado. La imagen tendrá una RAM de 1 GB y una capacidad de 10 GB de almacenamiento, y estará en formato QCOW2.

Para crear nuestro conjunto de instancias realizamos lo siguiente:

- Como hicimos anteriormente en el servicio de imagen, debemos crear y cargar una imagen acorde, como la que hemos descrito antes. Introducimos el archivo en el nodo de control, que es donde vamos a crearla, mediante una url, un dispositivo de almacenamiento, etc. En este caso, la traemos ya creada desde un dispositivo USB, por lo que mediante los comandos “mount” y “umount” la cargamos en el nodo en un directorio.

```
$ glance image-create --name "LinuxPSA" --file
/tmp/images/ubuntu-server.img --disk-format qcow2 --container-
format bare --visibility public --progress
```

- Comprobamos que se ha cargado correctamente en el servicio Glance o Nova:

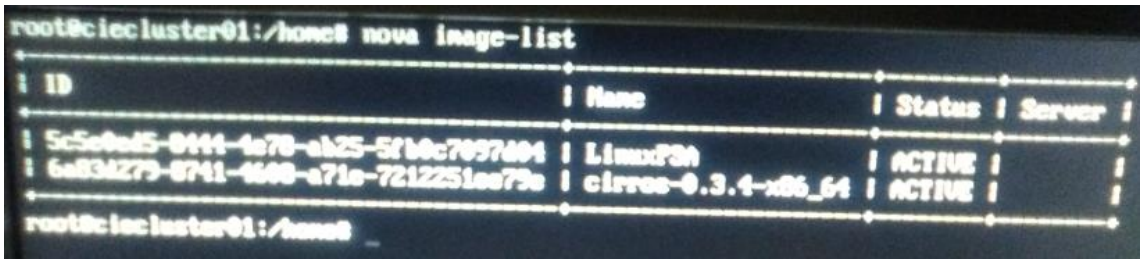


Ilustración 181. Listado de imágenes

- Ahora, procedemos a crear y levantar las instancias necesarias para el planteamiento del sistema que podría simularse en nuestro entorno. Para ello, sería necesario levantar unas 10 instancias aproximadamente. Creamos una instancia de esta imagen, de manera similar a como lo hicimos anteriormente con la imagen de prueba:

  ```
  $ nova boot --flavor m1.small --image LinuxPSA --nic net-
  id=DEMO_NET_ID --security-group default instancia1
  ```

  Repetimos el proceso 9 veces más:

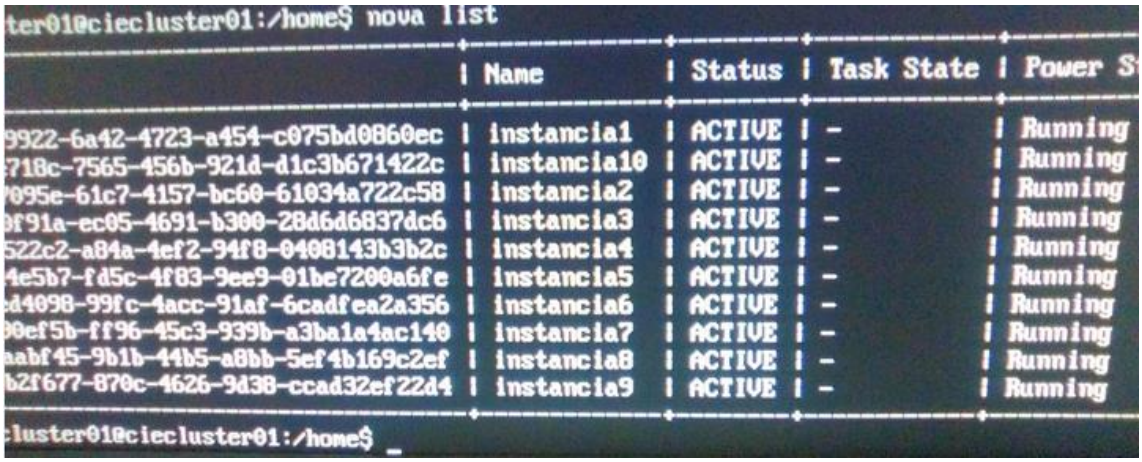


Ilustración 182. Creación de diez instancias

3.6. Descripción de ejemplo de sistema para posible simulación con OpenStack

La creación de una plataforma computacional de máquinas virtuales mediante OpenStack tiene como objetivo en este trabajo justificar su utilización para la simulación de sistemas alojados en la PSA, ya que a pesar de los componentes aparatosos que pueda tener un sistema de la PSA, como placas solares, heliostatos, autómatas, captadores, tanques, etc., todos ellos no dejan de estar controlados y monitorizados a través de ordenadores y software que los gestione, por lo que una configuración adecuada y unas conexiones necesarias con Internet y con ciertos componentes del sistema, hacen posible su funcionamiento desde fuera de la PSA.

Podemos simular varios sistemas, en función de las conexiones y los recursos de los que podamos disponer de ellos, pero en concreto, y para posibles futuros proyectos, con la configuración actual sería posible simular el sistema del *campo de captadores cilindro-parabólicos TCP-100* [10]:

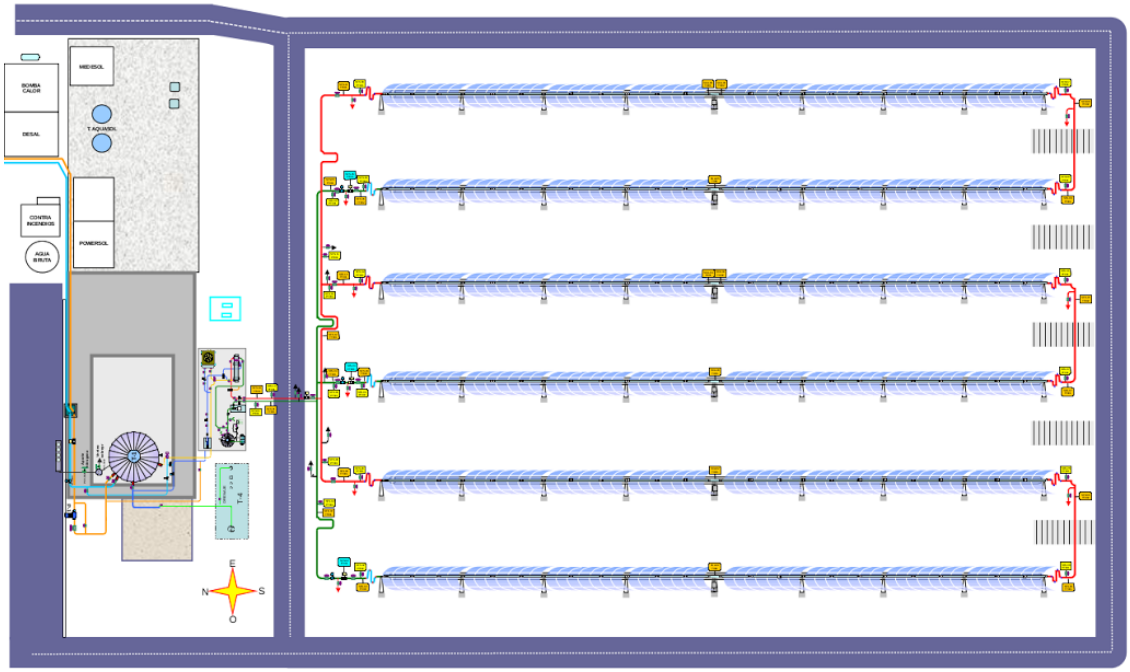


Ilustración 183. Esquema funcional del sistema

La composición del esquema es la siguiente, de acuerdo al artículo *Control de campos de colectores solares* [11].

Se dispone de un campo de captadores cilindro-parabólicos con seguimiento solar en elevación del tipo ACUREX. El campo Acurex está formado por colectores solares distribuidos del tipo Acurex, modelo 3001. Estos colectores son parabólicos y con seguimiento del sol en un solo eje (elevación). El campo está dispuesto en 20 filas de colectores, las cuales forman 10 lazos paralelos. En total en el campo hay 480 módulos orientados de este a oeste y la superficie total de espejos es de 2674 m^2.

Los tubos receptores, situados en la línea focal, emplean el flujo solar concentrado para calentar un aceite térmico, proveniente de la parte inferior de un tanque de almacenamiento donde el aceite se encuentra térmicamente estratificado, y a cuya parte superior es devuelto una vez ha aumentado su temperatura. El tanque está conectado a varios sistemas para utilizar la energía almacenada en él (turbina y planta desaladora solar).

El campo está provisto con un sistema de seguimiento del sol que mueve los espejos alrededor de un eje paralelo a aquel en el que se sitúa la tubería. El mecanismo de seguimiento puede alcanzar 3 posibles estados: Seguimiento (‘track’), Desenfocado (‘desteer’), y Bocabajo (‘stow’).

La planta está sometida a perturbaciones en la energía de entrada, que pueden ser lentas, debidas a variaciones de insolación a lo largo de un día claro, o bien bruscas, causadas fundamentalmente por la aparición de nubes y por variaciones de la temperatura de entrada al campo en la puesta en marcha del sistema de conversión de potencia. Todas estas perturbaciones obligan a variar continuamente el flujo de control, lo que a su vez provoca que el tiempo de residencia del fluido en el campo sea variable.

Ilustración 184. Serie de captadores cilindro-parabólicos

Ilustración 185. Vista aérea del campo

En términos más específicos, el sistema se compone de 3 autómatas PXA270 de la gama LinPac, 6 controles locales corriendo procesos que utilizan librerías de los autómatas y leen y escriben variables de memoria compartida y variables procedentes de sensores (presión, temperatura, caudal, etc.) y actuadores (bombas, válvulas, etc.), y un PC que alberga el diseño SCADA del sistema, todo conectado entre sí en una red Ethernet industrial, por lo que a efectos lógicos permite replicar la arquitectura en nuestro entorno.

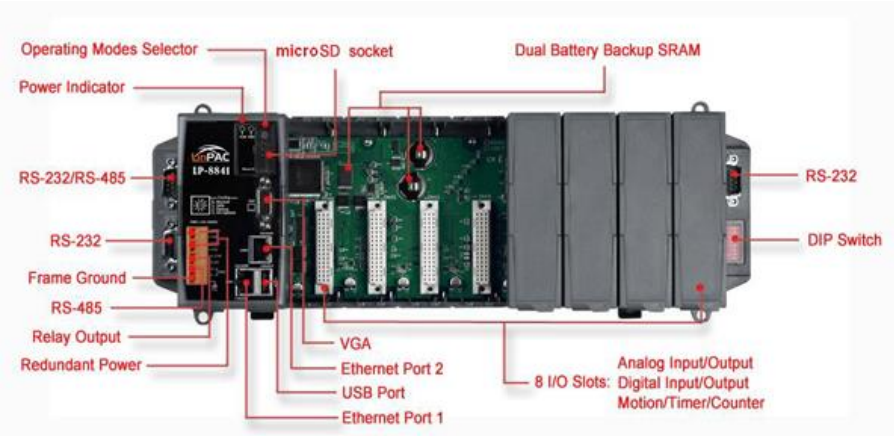


Ilustración 186. Autómata LinPac

A nivel de software, el sistema funciona de esta manera. En cada autómata, que funciona con Linux para su arquitectura, corren varios procesos pesados (estilo UNIX, sin hilos). Los autómatas corren un código fuente en C que al compilarlo, se genera un proceso que se comunica por 2 vías con otros.

Utilizando primitivas del sistema para memoria compartida entre procesos "shm_open,...", donde en un segmento de memoria compartida sitúa las variables que lee desde (o escribe hacia) la planta TCP-100, que son variables provenientes de sensores o actuadores. El segmento de memoria accedido en modo lectura/escritura por este proceso también es accedido por otro que es un servidor MODBUS, y que exporta las variables en modo lectura y escritura fuera del autómata (normalmente hacia el SCADA), a través de TCP/IP (de hecho, es un servidor MODBUS-TCP).

La segunda vía de "comunicación" es el acceso a las tarjetas, a través de la librería que contiene funciones como, por ejemplo: Open_Slot() ó AnalogInAll_87K(). A estas funciones no tendríamos acceso en las máquinas virtuales, por lo que tendríamos que rediseñar el código fuente para omitir las llamadas a esas librerías y hacer que funcione.

Resumiendo, como elementos a destacar en el sistema son:

- Diseño SCADA que controla el sistema citado. En él están representadas todas las variables y secuencias que deben correr en el sistema, incluidos sensores, actuadores, procesos de automatización, etc.
- 3 autómatas LinPac, de los que ya hemos hablado anteriormente.
- 6 controles que manejan y procesan los valores de todas las variables que entran y salen.
- Código fuente en C que corre en los autómatas y en los controles, que podemos consultar en el ANEXO III (pág. 152). Éste código podemos introducirlo en las máquinas virtuales, compilarlo y ejecutarlo de forma que simularían el funcionamiento de los autómatas LinPac.

4. CONCLUSIONES SOBRE EL TRABAJO REALIZADO Y DETALLES DE LO APRENDIDO

El principal objetivo de este Trabajo de Fin de Grado ha sido diseñar un sistema de máquinas virtuales con el objetivo de poder utilizar sus capacidades y su potencial en términos de rendimiento y paralelismo para realizar simulaciones de cualquier sistema (siempre dependiendo de las limitaciones y del acceso a los recursos que sean posibles) que se utilice en la Plataforma Solar de Almería o cualquier otro sitio con características e infraestructuras similares.

Sería interesante poder llevar a cabo la simulación de alguno de los sistemas alojados en la PSA, en este caso del sistema que hemos tratado en este proyecto, que es el del campo de captadores cilindro-parabólicos TCP-100, pero dada la carga lectiva del TFG y el tiempo adicional que ello supondría, se propone para un futuro trabajo, ya sea en otro Trabajo de Fin de Grado, o como continuación del mismo en un Trabajo de Fin de Máster.

Dados los recursos físicos algo limitados de los que se disponían y del tiempo disponible, considero que se han alcanzado los objetivos que se proponían en este Trabajo de Fin de Grado.

Personalmente, me llevo una gran experiencia y un gran aprendizaje con este TFG, ya que nunca había trabajado de primera mano con software de computación en la nube tan potente como OpenStack, ni con infraestructuras de ordenadores masivas como lo ha sido el clúster de ordenadores que aloja todo el software con el que se ha trabajado.

He aprendido a utilizar gran parte del software OpenStack, que es bastante complejo de montar e instalar, pero afortunadamente dispone de una documentación exquisita y de una gran comunidad de usuarios, que han contribuido sin duda a llevar a cabo la instalación de todo el software necesario. Además de todo lo visto en el trabajo, OpenStack tiene infinitas posibilidades de configuración y bastantes características adicionales que, de haber tenido algo más de tiempo o haber dispuesto de más recursos en términos de hardware, podría haber experimentado.

5. GLOSARIO DE TÉRMINOS.

[1]FEDER: Fondo Europeo de Desarrollo Regional.

[2]SCADA: Supervisión, Control y Adquisición de Datos.

[3]Aplicaciones asesinas: Es una aplicación informática determinante cuya implantación supone la asimilación definitiva por los usuarios.

[4]Datacenters: Son las ubicaciones donde se concentran los recursos necesarios para el procesamiento de la información de una organización.

[5]Datos sensibles: Datos de carácter personal, tipo político, convicciones religiosas…en definitiva datos privados para cada usuario.

[6]Jitter: Señal de ruido no deseada, causante de un retraso en una señal.

[7]Multitenencia: Es un principio de arquitectura en el que una instancia de una determinada aplicación se ejecuta en un servidor, pero sirviendo a múltiples clientes u organizaciones.

[8]Escalabilidad: Es la propiedad de un sistema que indica su habilidad para adaptarse a determinados cambios sin perder calidad, incluso mejorándola.

[9]Usuarios finales: Personas o usuarios que utilizan o manipulan de manera directa un producto software.

[10]Sistemas federados: Es un tipo especial de sistema de gestión de bases de datos.

[11]Numerónimo: Palabra que contiene números, calcada del inglés.

[12]Índice de consolidación: Capacidad para lograr la consolidación de recursos que pretende la virtualización de un sistema.

[13]CPD: Centro de Procesamiento de Datos, o también datacenter.

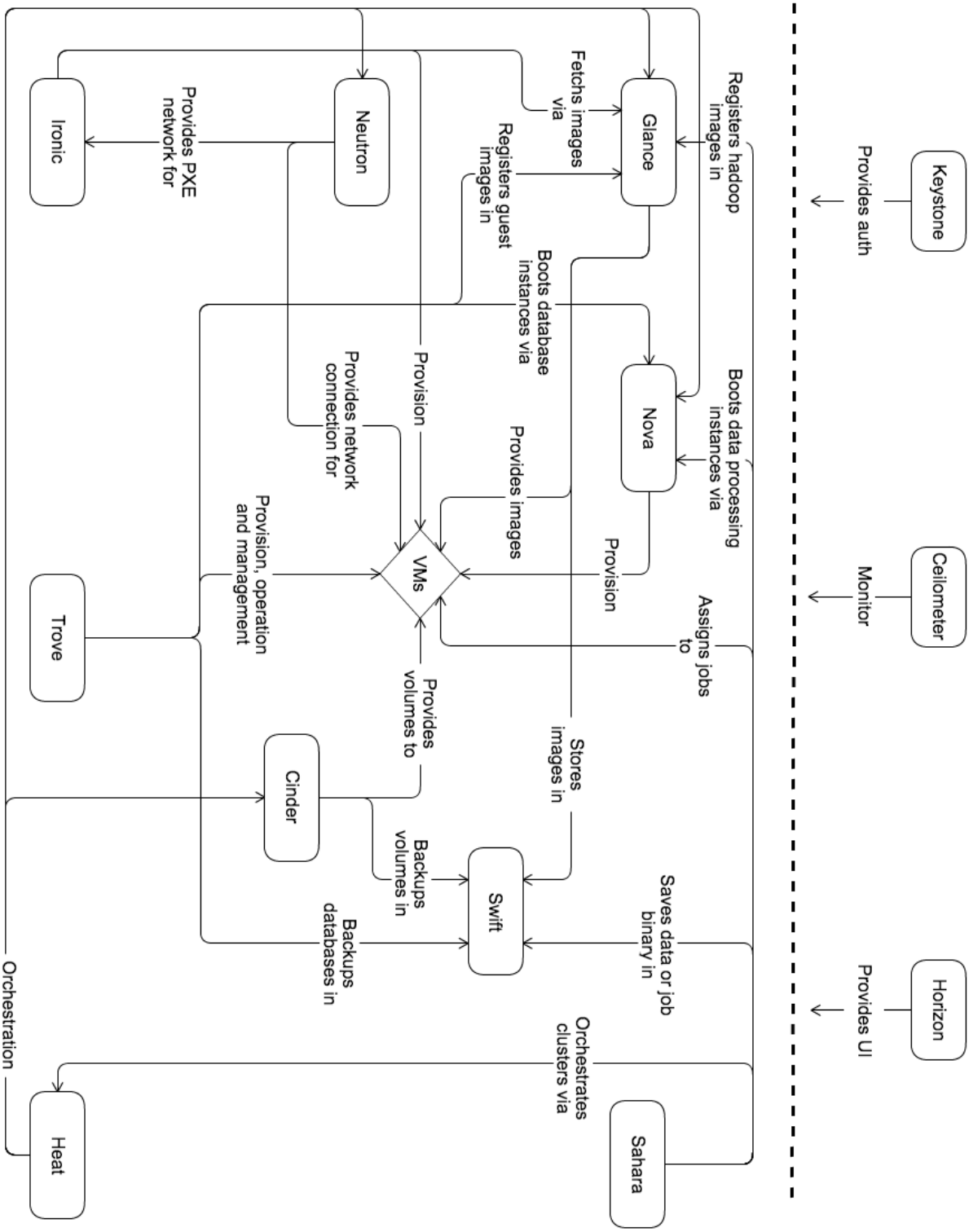
[14]TCO: Coste Total de Propiedad. Método que ayuda a determinar los costes directos e indirectos relacionados con la compra de equipos o programas informáticos.

[15]ROI: Retorno Sobre Inversión. Beneficio que se obtiene por cada unidad monetaria invertida durante un período de tiempo.

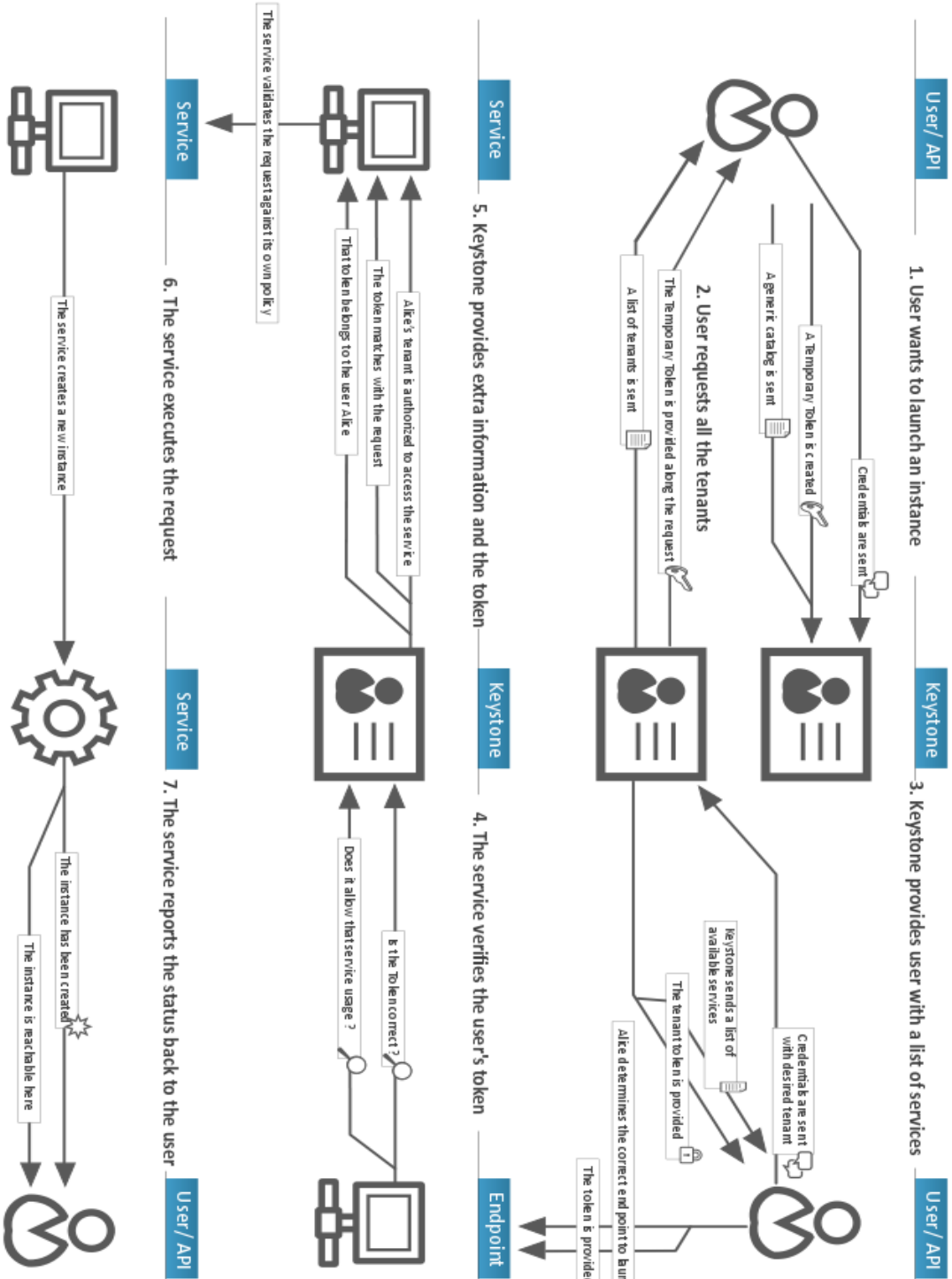
[16]Green IT: Es el uso eficiente de los recursos computacionales minimizando el impacto medioambiental.

6. ANEXOS

- **ANEXO I: Diagrama de servicios de OpenStack.**



- **ANEXO II: Diagrama del servicio Keystone.**



- **ANEXO III: Código fuente en C de autómatas LinPac.**

```
/* Módulo de Entradas Analógicas: I-87017 en Slot 01 de CPU1
Almacenar en Shared memory las capturas de tarjeta

   v6.0 2014.07.11 - J.A.R

*/

// Archivos a incluir
#include<stdio.h>
#include<stdlib.h>
#include "icpdas_mbs.h"
#include "msw.h"

//Declaración de Variables
char szSend[80], szReceive[80];       // Variables asociadas
DWORD dwBuf[12];                      // a lectura de canal
float fBuf[12];
float EA;                          // Intensidad de entrada a canal
int senal_min=4, senal_max=20, rango_min, rango_max;  // Rango 4-20mA
float SCC_ACC_EA01, SCC_ACC_EA02, TSS_SV_LT_001_EA01,
TSS_SV_PT_001_EA01,TSS_SV_PT_002_EA01, TSS_SV_TT_001_EA01,
TSS_SV_TT_002_EA01, TSS_SV_TT_003_EA01;    //Tag Canales
long l;



//Declaración Función conversión mA a valor temperatura, presión,...
float conversion(int rango_min,int rango_max, float EA);
```

```
//Inicio del programa

int main()

{

      //Declaración de Variables

      int wRetVal;


      // Abrir Shared Memory

      if ((shm_id[AI]= shm_open(MBS_AI, (O_CREAT | O_EXCL |
      O_RDWR),S_IRWXU )) > 0 ) {

                ftruncate(shm_id[AI], 2000*sizeof(int));

      } else if ((shm_id[AI]= shm_open(MBS_AI, (O_CREAT |
      O_RDWR),S_IRWXU)) < 0){

                perror("shm_open");

      }


      iMemory_AI=mmap(NULL,2000*sizeof(int),PROT_READ|PROT_WRITE,MAP_
      SHARED,shm_id[AI],0L);

      if (iMemory_AI == MAP_FAILED) {

                perror("mmap");

                exit(0);

      }

      if(close(shm_id[AI])<0){

                perror("shm_id close failed\n");

      }

      //Chequeo de Slot

      wRetVal = Open_Slot(0);

      if (wRetVal > 0) {

            printf("Apertura de Slot Fallida!\n");

            return (-1);

      }

      //Chequeo de puerto Com1 (Asignado al bastidor)

      wRetVal = Open_Com(COM1, 115200, Data8Bit, NonParity,
OneStopBit);
```

```
if (wRetVal > 0) {
      printf("Apertura de Puerto Fallida!\n");
      return (-1);
}

//Elección del Slot 01 de la CPU
ChangeToSlot(1);

//--- Función de Entradas Analógicas ----
**(AnalogInAll_87K()**)
// Declaración de parámetros asociados a la función
dwBuf[0] = 1;                    // COM Port
dwBuf[1] = 00;                   // Address
dwBuf[2] = 0x87017;              // ID
dwBuf[3] = 0;                    // CheckSum disable
dwBuf[4] = 100;                  // TimeOut , 100 msecond
dwBuf[6] = 0;                    // string debug

      //Funcion AI
      wRetVal = AnalogInAll_87K(dwBuf, fBuf, szSend,
      szReceive);
      if (wRetVal) {
            printf("Error AI-87017, Código de
            error=%d\n",wRetVal);
      }
      else {
            //Conversión a intensidad
            SCC_ACC_EA01=conversion(0, 25,fBuf[0]);
            //Convierte la entrada en "palabra"
            l = *((unsigned long *)&SCC_ACC_EA01);
            iMemory_AI[0]=*((short *)&l);   /* Float (Little-
            Endian) - 1er Reg. Menos Signif. */
```

```
// Almacena la palabra en dos registros en memoria
iMemory_AI[1]=*((short *)&l+1);
// Conversión a potencia
SCC_ACC_EA02=conversion(0, 5400, fBuf[1])
//Convierte la entrada en "palabra"
l = *((unsigned long *)&SCC_ACC_EA02);
iMemory_AI[2]=*((short *)&l);   /* Float (Little-
Endian) - 1er Reg. Menos Signif. */
// Almacena la palabra en dos registros en memoria
iMemory_AI[3]=*((short *)&l+1);

// Conversión a nivel
TSS_SV_LT_001_EA01=conversion(0, 100,fBuf[2]);
// Convierte la entrada en "palabra"
l = *((unsigned long *)&TSS_SV_LT_001_EA01);
iMemory_AI[4]=*((short *)&l);   /* Float (Little-
Endian) - 1er Reg. Menos Signif. */
// Almacena la palabra en dos registros en memoria
iMemory_AI[5]=*((short *)&l+1);
// Conversión a presión
TSS_SV_PT_001_EA01=conversion(0, 8, fBuf[3]);
// Convierte la entrada en "palabra"
l = *((unsigned long *)&TSS_SV_PT_001_EA01);
iMemory_AI[6]=*((short *)&l);   /* Float (Little-
Endian) - 1er Reg. Menos Signif. */
//Almacena la palabra en dos registros en memoria
iMemory_AI[7]=*((short *)&l+1);


// Conversión a presión
TSS_SV_PT_002_EA01=conversion(0, 8, fBuf[4]);
//Convierte la entrada en "palabra"
l = *((unsigned long *)&TSS_SV_PT_002_EA01);
```

```
        iMemory_AI[8]=*((short *)&l);    /* Float (Little-
        Endian) - 1er Reg. Menos Signif. */

        // Almacena la palabra en dos registros en memoria

        iMemory_AI[9]=*((short *)&l+1);

        // Conversión a temperatura

        TSS_SV_TT_001_EA01=conversion(-10, 315, fBuf[5]);


        //Convierte la entrada en "palabra"

        l = *((unsigned long *)&TSS_SV_TT_001_EA01);

        iMemory_AI[10]=*((short *)&l);  /* Float (Little-
        Endian) - 1er Reg. Menos Signif. */

        // Almacena la palabra en dos registros en memoria

        iMemory_AI[11]=*((short *)&l+1);

        // Conversión a temperatura

        TSS_SV_TT_002_EA01=conversion(-10, 315, fBuf[6]);

        // Convierte la entrada en "palabra"

        l = *((unsigned long *)&TSS_SV_TT_002_EA01);

        iMemory_AI[12]=*((short *)&l);  /* Float (Little-
        Endian) - 1er Reg. Menos Signif. */

        // Almacena la palabra en dos registros en memoria

        iMemory_AI[13]=*((short *)&l+1);

        // Conversión a temperatura

        TSS_SV_TT_003_EA01=conversion(-10, 315, fBuf[7]);

        // Convierte la entrada en "palabra"

        l = *((unsigned long *)&TSS_SV_TT_003_EA01);

        iMemory_AI[14]=*((short *)&l);  /* Float (Little-
        Endian) - 1er Reg. Menos Signif. */



        // Almacena la palabra en dos registros en memoria

        iMemory_AI[15]=*((short *)&l+1);

}
```

```
	Close_Com(COM1);			// Cerrar puerto
	Close_SlotAll();			// Cerrar Slot
	return 0;
}

// Función conversión mA a valor temperatura, presión,...
float conversion(int rango_min,int rango_max,float EA)
{
	float valor;
	valor = ((EA-senal_min)*(rango_max-rango_min))/(senal_max-
	senal_min)+rango_min;
	return valor;
}
```

7. REFERENCIAS BIBLIOGRÁFICAS.

[1] Centro de Investigación en Energía Solar. (2015). *CIESOL*. Recuperado el 15 de Agosto de 2016, de http://www.ciesol.es/index.php?Idioma=ES&Opcion=6&Pagina=80

[2] Plataforma Solar de Almería. (2013). *PSA*. Recuperado el 2 de Septiembre de 2016, de http://www.psa.es/es/index.php

[3] Wikipedia, La Enciclopedia Libre. (2016). *Computación en la nube*. Recuperado el 2 de Septiembre de 2016, de https://es.wikipedia.org/wiki/Computación_en_la_nube

[4] Joyanes Aguilar, L. (2012). *Computación en la nube: notas para una estrategia española en cloud computing*. Recuperado el 1 de Septiembre de 2016, de http://revista.ieee.es/index.php/ieee/article/view/10

[5] Wikipedia, La Enciclopedia Libre. (2016). *OpenStack*. Recuperado el 18 de Agosto de 2016, de https://es.wikipedia.org/wiki/OpenStack

[6] Wikipedia, La Enciclopedia Libre. (2016). *Virtualización*. Recuperado el 27 de Agosto de 2016, de https://es.wikipedia.org/wiki/Virtualización

[7] Wikipedia, La Enciclopedia Libre. (2015). *Kernel-based Virtual Machine*. Recuperado el 22 de Agosto de 2016, de https://es.wikipedia.org/wiki/Kernel-based_Virtual_Machine

[8] Wikipedia, La Enciclopedia Libre. (2016). QEMU. Recuperado el 22 de Agosto de 2016, de https://es.wikipedia.org/wiki/QEMU

[9] OpenStack Project. (2015). *Documentation for Kilo (April 2015)*. Recuperado el 10 de Mayo de 2016, de http://docs.openstack.org/kilo/

[10] Plataforma Solar de Almería. (2015). *Informe Técnico Anual 2015*. Recuperado el 2 de Septiembre de 2016, de http://www.psa.es/es/techrep/2015/ANNUAL_REPORT_2015.pdf

[11] R. Rubio, Francisco, F. Camacho, Eduardo, Berenguel Soria, M. (2006). Control de campos de colectores solares. Revista Iberoamericana de Automática e Informática Industrial, 4(26-45), 1697-7912.