Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Instalación y pruebas con una plataforma de gestión de nube.

Autor: Mario Gómez Estríngana

Tutor/es: José Manuel Arco Rodríguez

2021

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Grado en Ingeniería en Tecnologías de Telecomunicación

Trabajo Fin de Grado

"Instalación y pruebas con una plataforma de gestión de nube."

Autor: Mario Gómez Estríngana

Tutor/es: José Manuel Arco Rodríguez

TRIBUNAL:

Presidente: Antonio García Herráiz

Vocal 1º: Elisa Rojas Sánchez

Vocal 2º: José Manuel Arco Rodríguez

Fecha: Abril, 2021

Resumen

El objetivo del trabajo es la instalación y el estudio de un entorno de gestión de nube en producción con pocos servidores.

Para conseguirlo, se utilizan tres equipos, un nodo controlador y dos nodos de cómputo, en los que se instala, desde cero, un sistema operativo donde se lanza el software de gestión Cloud de virtualización OpenStack.

Una vez instalado dicho software se trata de incorporar al entorno, un servidor web público y realización de pruebas de carga, mantenimiento y monitorización.

Abstract

The goal of this project is the installation and study of a cloud management environment.

In order to achieve this goal, three computers are used, a controller and two compute nodes, in which, an operating system is installed from scratch, where the OpenStack virtualization cloud management software is launched.

Once said software is installed, it is a matter of incorporating a public web server into the environment and load, maintenance and monitoring tests are carried out.

Índice

Listado de Imágenes

Palabras clave

Devstack, Computación en la nube, Gestión de nube, Instancias, Nube híbrida.

Key words

Devstack, Cloud computing, Cloud management, Servers, Hybrid cloud.

Listado de Acrónimos

API	Application Programming Interfaces. Interfaz de programación de aplicaciones.
AWS	Amazon Web Services.
CLI	Command Line Interface. Interfaz de línea de comandos.
CPU	Central Processing Unit. Unidad de control de procesamiento.
IaaS	Infrastructure as a Service. Infraestructura como servicio.
ID	Identifier. Identificador.
IP	Internet Protocol. Protocolo de internet.
IT	Information Technology. Tecnología informática.
LDAP	Lightweight Directory Access Protocol. Protocolo ligero de acceso a directorios.
LVM	Logical Volume Manager. Gestor de volúmenes lógicos.
NFV	Network Function Virtualization. Virtualización de funciones de red.
OAuth	Open Authorization. Abierto para la autorización.
PaaS	Platform as a Service. Plataforma como servicio.
QoS	Quality of Service. Calidad de servicio.
RBAC	Role-Based Access Control. Control de acceso basado en roles.
RESTful	Programa basado en REST. Representational State Transfer. Transferencia de estado representacional.
SaaS	Software as a Service. Software como servicio.
SAML	Security Assertion Markup Language. Lenguaje de marcado para confirmaciones de seguridad.
SQL	Structured Query Language. Lenguaje de consulta estructurada.
VM	Virtual Machine. Máquina virtual.

Resumen extendido

La finalidad de este trabajo es introducirse en el Cloud Computing. Para ello se describe el desarrollo de este con el paso del tiempo y se recogen sus características, así como las ventajas e inconvenientes que el Cloud Computing ofrece.

Además, se recoge información de los distintos servicios que puede ofrecer una nube y los distintos tipos de nubes que existen.

Esta memoria describe de forma detallada, como realizar el despliegue de un entorno de gestión de nube.

Para realizar el despliegue se lanza el software de gestión cloud de virtualización, OpenStack, el cual es de código abierto. Existen diferentes métodos de instalación, a mano, es decir, componente a componente, con juju, con ansible, con MicroStack, DevStack…

El método de instalación de OpenStack elegido es DevStack.

Con DevStack se consigue levantar de forma automática y rápida un entorno OpenStack completo, basado en las últimas versiones de todos los servicios recogidos en el repositorio de GIT.

Tras realizar el despliegue, se describe el desarrollo de operaciones como lanzar instancias, acceder a ellas y eliminarlas.

También, se crean una serie de scripts para la creación y eliminación de usuarios, con la finalidad de automatizar las labores de administrador del entorno.

Además del despliegue básico, se introducen servicios adicionales para usos avanzados de OpenStack y se realizan pruebas para verificar el correcto de funcionamiento de estos.

Por último se describe la introducción de la nube privada OpenStack a AWS, dando paso a una nube híbrida para contar con más recursos, y la gestión de todo el entorno desde una única plataforma.

Extended abstract

The purpose of this work is to get into Cloud Computing. For this, its development over time is described and its characteristics are collected, as well as the advantages and disadvantages that Cloud Computing offers.

In addition, information is collected on the different services that a cloud can offer and the different types of cloud that exist.

This report describes in detail how to deploy a cloud management environment.

To carry out the deployment, the virtualization cloud management software, OpenStack, which is open source, is launched. There are different installation methods, by hand, that is, component to component, with juju, with ansible, with MicroStack, DevStack…

The chosen OpenStack launch method is DevStack.

With DevStack it is possible to build a complete OpenStack environment automatically and quickly, based on the latest versions of all the services collected in the Git repository.

After deployment, the development of operations such as launching, accessing, and deleting instances is described.

Also, some scripts are created for the creation and elimination of users in order to automate the tasks of the environment administrator.

In addition to the basic deployment, additional services are introduced for advanced uses of OpenStack and tests are carried out to verify their correct operation.

Finally, the introduction of the OpenStack private cloud to AWS is described, giving way to a hybrid cloud to have more resources, and the management of the entire environment from a single platform.

1. Introducción

El objetivo principal de este proyecto es el despliegue de una nube privada OpenStack, para el cual, se evalúan diferentes métodos de instalación teniendo en cuenta diferentes aspectos, como los equipos necesarios para el despliegue, el tiempo necesario para realizar este o las posibilidades que ofrece el entorno una vez desplegado entre otros, y se escoge el más apropiado. Una vez realizado el despliegue se desarrollaran diferentes objetivos secundarios, como el desarrollo de scripts para automatizar tareas de administración de la nube para crear usuarios a partir de un fichero de correos electrónicos y a su vez la eliminación de estos una vez acabe el periodo de uso de la nube; lanzamiento de instancias, incluyendo diferentes accesos a estas y su eliminación; usos avanzados de OpenStack, como movimiento de instancias entre los diferentes nodos con la finalidad de no perder información de los usuarios en caso de pérdida de recursos, monitorización para controlar en todo momento la utilización de los recursos o la carga de CPU que se está produciendo en el entorno y los motivos de esta, instalación de diferentes componentes adicionales que ofrezcan diferentes servicios, etc; y la integración de dos nubes, la privada (OpenStack) y una pública (AWS o Google Cloud) con el fin de tener un mayor número de recursos disponibles en el entorno.

Para comenzar con la memoria, en el capítulo “2. Introducción al cloud computing”, se realiza un breve resumen de su evolución con el paso del tiempo, desde sus inicios; se detallan sus características, las ventajas y desventajas que este proporciona; y los distintos tipos de servicios que son ofrecidos y las diferentes arquitecturas de nube que existen.

El siguiente capítulo, “3. OpenStack”, recoge todo lo referido al despliegue del entorno. Una introducción al software OpenStack, los diferentes componentes que ofrece con una breve descripción de los componentes básicos, una comparación de los diferentes métodos de instalación, así como, la instalación detallada de dos de ellos y diferentes operaciones con el método de instalación finalmente escogido: lanzar instancias, el desarrollo de los scripts para la creación automática de usuarios, migrar una instancia entre diferentes nodos y usos avanzados del entorno.

El capítulo “4. Nube Híbrida”, trata del proyecto de integración de OpenStack con AWS, OpenStack-Omni.

Y por último, los capítulos “5. Conclusiones”, “6. Presupuesto”, “7. Bibliografía” y “8. Anexos” recogen respectivamente las conclusiones y trabajo futuro del desarrollo del proyecto, el presupuesto tanto del software como del hardware para llevarlo a cabo, las referencias utilizadas y los archivos completos utilizados para la instalación del entorno.

2. Introducción al Cloud Computing

El Cloud Computing es un sistema de computación, redes y almacenamiento de datos a través de internet, el cual ofrece a las empresas la posibilidad de almacenar sus datos en servidores externos y acceder a ellos desde cualquier parte, haciendo uso de un dispositivo con conexión a internet.

2.1. Origen del Cloud Computing

Los comienzos del Cloud Computing se remontan a los años 50 [1], las grandes empresas tenían la necesidad de poder acceder a distinta información desde distintos puntos de acceso. El problema era, que debido al gran tamaño de la infraestructura de aquella época era imposible tener un ordenador en cada oficina, además de que era muy costoso, ocupaba demasiado espacio.

Por este motivo, se decidió apostar por la computación en sistema compartido, pensando en vender el uso del ordenador, el espacio y la memoria como si fuera otro servicio público.

Tras esta idea, años más tarde, se planteó la necesidad de unas redes intergalácticas de computación, un sistema que permitiera a varios usuarios compartir información. Sin embargo, este proyecto, como la computación en sistema compartido quedaron pausados y con el paso del tiempo llegó internet.

Tras la llegada progresiva de internet [2], en la década de los 90 cambió la forma en la que las personas se conectaban a internet. Se introdujo el correo electrónico, el cual podemos considerar servicio en la nube.

La integración del correo electrónico proporcionó un mayor desarrollo de servicios como el alojamiento de fotografías, ofimática o música bajo demanda.

Con estos avances comenzaron los servicios de IaaS. Estos servicios proporcionaban la posibilidad de aprovechar la escalabilidad de la computación en la nube sin hacer frente a gastos de capital inicial, ni a requisitos de mantenimiento continuo, lo que era favorable tanto para grandes como pequeñas empresas.

Poco a poco, la nube fue ganando popularidad a medida que las empresas iban comprendiendo mejor las ventajas que proporcionaba, su utilidad y los servicios que esta podía ofrecer.

Ya en el nuevo siglo, comenzaron los servicios de entrega de software a través de internet, lo que supuso que cualquier persona que tuviera conexión a internet fuera capaz de descargar un programa y/o aplicación de manera rentable y bajo demanda, a distancia.

Todos estos nuevos servicios en la nube proporcionaron a las empresas una manera de aprovisionar y escalar sus propias ofertas, eliminando la necesidad de tener que comprar y configurar una gran

cantidad de hardware y sistemas, ni contratar a personal altamente cualificado para administrar su infraestructura.

A partir de ahí, el Cloud Computing evoluciona rápidamente hasta llegar a nuestros días y a toda la gama de servicios que hoy utilizamos en cualquier ámbito profesional o privado.

2.2. Características del Cloud Computing

El Cloud Computing es una tecnología que ofrece distintos servicios [6], como almacenamiento de archivos, uso de aplicaciones o conexión de dispositivos, sin ocupar espacio en el disco duro de nuestro ordenador.

Independientemente del servicio que se utilice, las principales características del Cloud Computing son:

- **Almacenamiento externo**.

  Permite almacenar archivos en un espacio de internet sin ocupar espacio del disco duro del equipo propio.

  El tipo de archivos que se almacenen depende del espacio contratado en la nube. Se pueden almacenar desde informes y documentos de texto ligeros hasta bases de datos y contenidos audiovisuales de gran tamaño.

- **Uso de aplicaciones online.**

  Este servicio permite ahorrar espacio de instalación en los ordenadores y que por ejemplo, en una empresa todos utilicen la misma aplicación, en vez de distintas instalaciones. Lo que implica una mayor integración del trabajo y ahorro de tiempo.

  Puede tratarse de cualquier tipo de aplicación.

- **Conexiones desde cualquier lugar y momento.**

  Los usuarios tienen acceso a los servicios desde cualquier dispositivo: móvil, tableta, laptop o desktop.

  Esta característica potencia aún más las dos nombradas con anterioridad.

- **Comunicaciones globales.**

  El Cloud Computing ofrece la posibilidad de que personas que trabajan en el mismo negocio o proyecto establezcan comunicación audiovisual, en tiempo real o en diferido con multitud de opciones comunicativas como las proyecciones compartidas o las divisiones en grupo.

- **Seguridad avanzada.**

  Es necesario y muy importante que los datos e información estén protegidos en internet por lo que es imprescindible elegir un servicio en la nube que proporcione seguridad.

Las empresas dedicadas al Cloud Computing ofrecen medidas de seguridad muy fiables y es muy poco probable, a la par que muy difícil, que alguien pueda acceder a datos para los que no tiene los permisos correspondientes.

Además los servicios en la nube ofrecen copias de seguridad automáticas, lo que implica que la pérdida de tus datos y archivos se antoja complicada por mucho que se estropee tu equipo, todos tus archivos podrán ser recuperados de la nube.

Como las copias de seguridad se realizan de manera automática no perderás tiempo, ni recursos de tu equipo en esto.

En conclusión, tus archivos están más a salvo en la nube que en tu propio disco duro debido a la seguridad que ofrecen los servicios.

- **Autoservicio a petición.**

  Como los servicios de cómputo están en la nube [4], se pueden aprovisionar de forma automática bajo demanda. En el momento que algún usuario necesite algún recurso puede disponer de él.

- **Capacidad de medición.**

  El sistema de Cloud Computing [7] nos ayuda a medir los recursos que utilizamos en un modelo de consumo bajo demanda y elástico. El sistema tiene la capacidad de saber cuántos recursos, bajo qué circunstancias y durante cuánto tiempo se consumen.

- **Rápida elasticidad.**

  Los recursos pueden crecer o decrecer automáticamente [4].

  Sólo se consume lo que se necesita y por tanto sólo se facturan los servicios que se consumen. Esto permite optimizar el coste/consumo de los recursos.

- **Pago por uso.**

  En la nube el uso de recursos está medido [7], para que no tengan que pagar por servicios que no se necesitan.

  Solo se paga por lo que se usa.

- **Calidad de servicio.**

  El Cloud Computing asegura que la disponibilidad y eficiencia de los recursos se cumplen de forma correcta garantizando unos niveles de calidad de servicio (QoS).

  Los modelos de Cloud Computing ofrecen distintas topologías del recurso, las cuales llevan asociadas unas capacidades. Esto sirve para elegir el tipo que más se ajuste a nuestras necesidades.

2.3. Ventajas del Cloud Computing

- **Reducción de costes.**

  Gracias al Cloud Computing [8], las empresas se permiten no invertir en infraestructuras de IT propias y licencias de software.

- **Movilidad: acceso desde cualquier dispositivo y lugar.**

  Esta ventaja ofrece flexibilidad laboral tanto para trabajar como para atender a los clientes desde cualquier lugar..

- **Pago por uso y gasto bajo control.**

  La empresa paga únicamente los servicios necesarios en cada momento y puede ajustar los gastos a sus necesidades reales

- **Tecnología siempre actualizada.**

  Gracias a la nube, el cliente que contrata sus servicios se asegura siempre una tecnología actualizada y optimizada.

- **Capacidad de almacenamiento ilimitada.**

  La nube ofrece un almacenamiento prácticamente ilimitado. Esto quiere decir que el usuario no se ve obligado a ampliar sus propios equipos cada poco tiempo, por lo que también ahorra dinero.

- **Respeto al medio ambiente.**

  Al utilizar la nube se reduce la huella de carbono de una empresa, ya que los recursos en vez de estar almacenados en componentes físicos pasan a ser virtuales.

- **No necesitas ´lo último´ en ordenadores.**

  El Cloud Computing permite utilizar ordenadores con recursos menores, lo que también implica un ahorro de dinero.

- **Seguridad.**

  Si en el disco duro del equipo se produce cualquier fallo, este no afecta a los datos.

- **Dejas de estar atado a un PC.**

  Gracias al Cloud Computing te puedes permitir cambiar sin miedo los equipos, ya que las aplicaciones y los datos seguirán estando en la nube y tendrás acceso a ellos en el nuevo equipo.

- **Igualdad.**

  La nube permite que las pequeñas y medianas empresas estén en las mismas condiciones que las grandes.

- **Escalabilidad.**

  Todas las empresas no tienen los mismos requisitos de IT [10].

  El Cloud Computing permite a las empresas ampliar de una forma eficaz y veloz sus departamentos de IT.

- **Recuperación ante desastres.**

  El almacenamiento en la nube permite recuperar de forma rápida los datos ante cualquier problema que surja en los equipos.

- **Control.**

  La nube permite tener gran visibilidad y control sobre los datos, pudiendo elegir qué usuarios tienen acceso a qué datos y a qué nivel.

2.4. Desventajas del Cloud Computing

- **Se requiere una conexión permanente a internet.**

  Es imposible acceder a la nube si no se tiene acceso a internet. Esto implica que el periodo que este caída la conexión, no se podrá trabajar [11].

- **No funciona bien con conexiones de baja velocidad.**

  Una conexión a internet de baja velocidad hace que la computación en la nube en muchos casos sea imposible ya que bastantes descargas precisan de una gran cantidad de ancho de banda.

- **Algunas veces puede ser demasiado lento.**

  Aunque se tenga una conexión rápida, las aplicaciones en la web son más lentas que aplicaciones similares instaladas en el propio equipo.

- **Seguridad.**

  La información recorre diferentes nodos hasta llegar a su destino [9], siendo cada uno de estos un foco de inseguridad.
  La utilización de protocolos seguros disminuye la velocidad debido a la sobrecarga requerida.

- **Dependencia de un proveedor.**

  Los usuarios dependen de los proveedores de servicios debido a la centralización de las aplicaciones y la gran cantidad de información almacenada en la nube.

- **Control limitado.**

El cliente solo tiene un control limitado de las funciones del sistema. Para modificar o actualizar las funciones, el encargado será el proveedor del servicio.

- **Escalabilidad a largo plazo.**

  Cuantos más usuarios comparten la infraestructura de la nube [10], la sobrecarga en los servidores de los proveedores aumenta. Lo que implica que si no se posee un esquema de crecimiento optimo se producirán degradaciones en los servicios.

2.5. Tipos de Servicios

Los tipos de servicios que ofrece el Cloud Computing se pueden agrupar en tres grandes bloques.

- **IaaS (Infrastructure as a Service).**

Imagen 1: LaaS [15]

Con la contratación de este servicio, el cliente obtiene acceso a los recursos informáticos, como pueden ser servidores, espacio de almacenamiento o equipamiento para redes. Este servicio se realiza a través de una plataforma de virtualización.

Es el servicio que mayor control otorga a los usuarios y el más complejo de utilizar.

Es ideal para desarrolladores, permite escoger el sistema operativo y la cantidad de almacenamiento gracias a la total transparencia del servicio.

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

Imagen 2: PaaS [15]

Este servicio priva al usuario de control sobre la infraestructura de almacenamiento o redes debido a que es el proveedor quien proporciona el mantenimiento y gestión de la plataforma para el desarrollo de aplicaciones.

Este servicio mantiene la escalabilidad automática dependiendo de lo que requiera la situación.

Es un servicio orientado a los equipos de desarrollo de software.

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

Imagen 3: SaaS [15]

Con este servicio el cliente solo puede hacer uso del software que se encuentra alojado en la nube.

No puede hacer más operaciones.

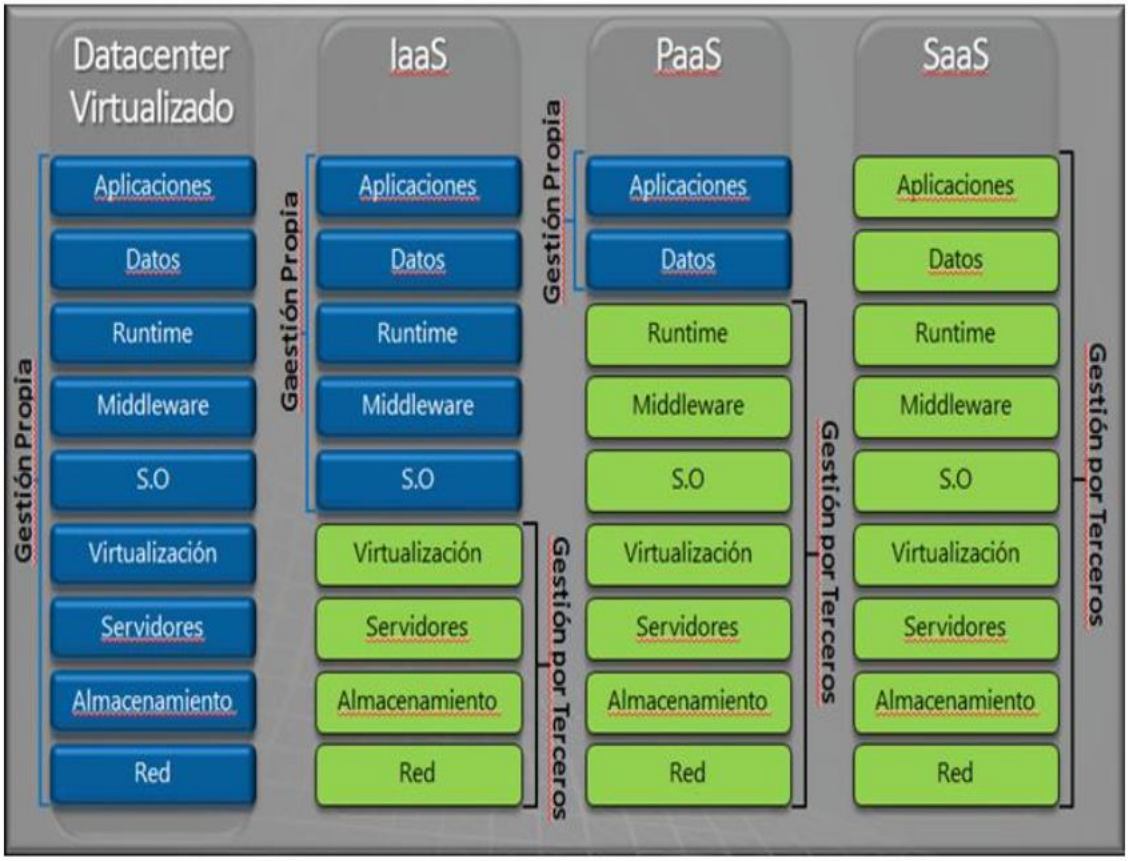


Imagen 4: Tipos de Servicios [14]

2.6. Tipos de Nubes

El Cloud Computing puede implementarse atendiendo a los siguientes modelos de nube.

- **Nube pública.**

  Sus servicios son ofrecidos [16] a cualquier usuario por empresas externas que se encargan de su mantenimiento y gestión.

  Algunas características de este modelo son:

  - La información es almacenada en servidores de terceros.
  - Los recursos de los servidores son compartidos por todos los clientes del proveedor.
  - No requiere una inversión inicial en infraestructura.
  - Mantenimiento y actualización corren a cargo de la empresa externa.

- **Nube privada.**

  Al contrario que la nube privada, sus servicios son ofrecidos a un número limitado de usuarios utilizando una red de una empresa, la cual puede estar alojada en las propias instalaciones de la empresa o en las de un proveedor externo.

  Las principales características de este modelo son:

  - Los servidores y la infraestructura son de nuestra propiedad.
  - Ofrece mayor grado de seguridad al no estar gestionada por terceros.
  - Hardware dedicado y adaptado a nuestras necesidades.
  - Se requiere una inversión inicial superior.
  - Costes de mantenimiento y actualización superiores a los de la nube pública.

- **Nube híbrida.**

  Este modelo es un combinación entre la nube pública y la nube privada. Se utilizan recursos de una u otra nube en función de las necesidades.

  Podemos disponer de servidores propios y utilizar en casos puntuales la nube pública para funcionalidades especificas o conexiones con herramientas externas.

- **Multicloud.**

  Consiste en la combinación de varias nubes [17] entre las que se desplazan los distintos servicios y que pueden trabajar simultáneamente. Con este servicio los clientes se protegen ante posibles interrupciones del servicio.

3. Despliegue de OpenStack

OpenStack es un software libre [22] y de código abierto distribuido bajo los términos de la licencia Apache.

Es un sistema operativo cloud que controla grandes grupos de recursos informáticos [18], de almacenamiento y de redes en un centro de datos, todos administrados y aprovisionados a través de API con mecanismos de autenticación comunes.

Este software ofrece un panel de control, que brinda a los administradores el control al mismo tiempo que permite a sus usuarios aprovisionar recursos a través de una interfaz web.

Además de la funcionalidad estándar de infraestructura como servicio, los componentes adicionales brindan orquestación, gestión de fallos y administración de servicios, entre otros servicios, con el fin de garantizar una alta disponibilidad de las aplicaciones de usuario.

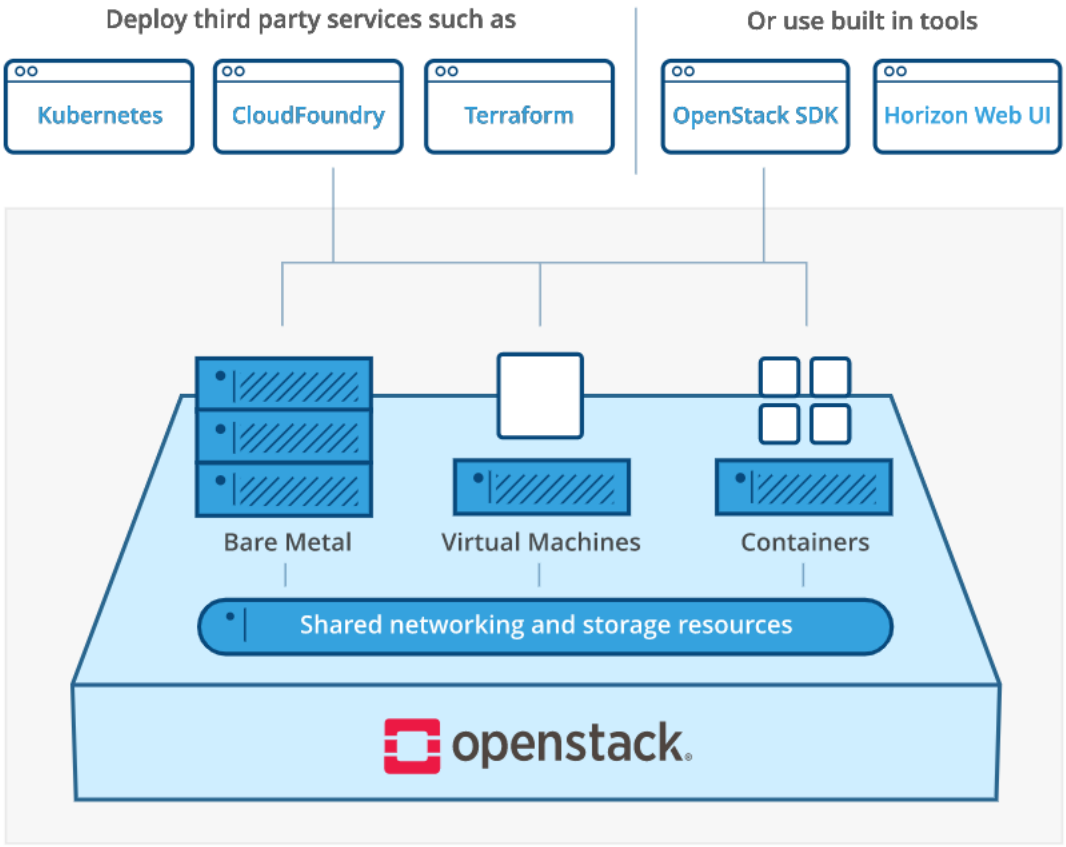


Imagen 5: OpenStack [18]

3.1. Componentes de OpenStack

OpenStack se divide en servicios para permitir al usuario conectar y reproducir componentes en función de sus necesidades.

En el mapa de OpenStack se puede ver donde encajan esos servicios y cómo pueden trabajar juntos.

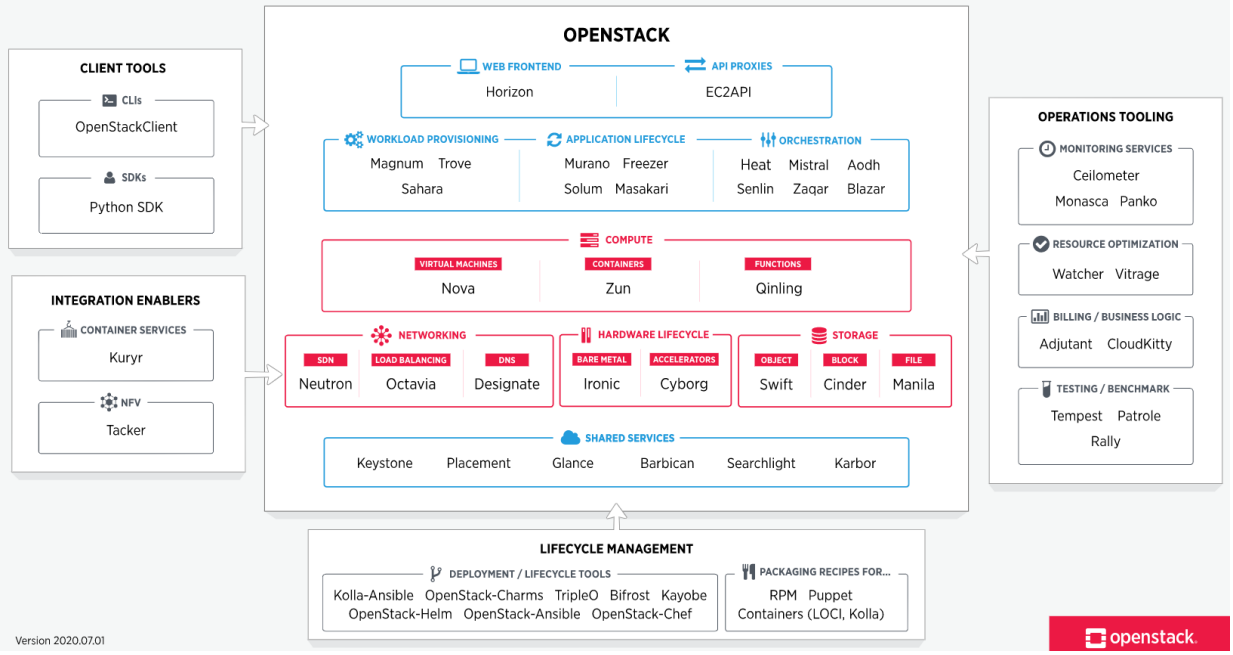


Imagen 6: Componentes OpenStack [18]

Los componentes de OpenStack se clasifican en diferentes grupos [27]:

SERVICIOS OPENSTACK

- Computación.
  - Nova: Servicio de computación.
  - Zun: Servicio de contenedores.
- Ciclo de vida del hardware.
  - Ironic: Servicio de aprovisionamiento completo.
  - Cyborg: Gestión del ciclo de vida de los aceleradores.
- Almacenamiento.
  - Swift: Almacenamiento de objetos.
  - Cinder: Almacenamiento en bloque.
  - Manila: Sistemas de archivos compartidos.

- Redes.
    - Neutron: Redes.
    - Octavia: Equilibrador de carga.
    - Designate: Servicio DNS.
- Servicios compartidos.
    - Keystone: Servicio de identidad.
    - Placement: Servicio de colocación.
    - Glance: Servicio de imagen.
    - Barbican: Gestión de claves.
- Orquestación.
    - Heat: Orquestación.
    - Senlin: Servicio de agrupación en clústeres.
    - Mistral: Servicio de flujo de trabajo.
    - Zaqar: Servicio de mensajería.
    - Blazar: Servicio de reserva de recursos.
    - Aodh: Servicio de alarmas.
- Aprovisionamiento de cargas de trabajo.
    - Magnum: Aprovisionamiento del motor de orquestación de contenedores.
    - Sahara: Aprovisionamiento del marco de procesamiento de Big Data.
    - Trove: Base de datos como servicio.
- Ciclo de vida de la aplicación.
    - Masakari: Servicio de alta disponibilidad de instancias.
    - Murano: Catálogo de aplicaciones.
    - Solum: Automatización del ciclo de vida del desarrollo de software.
    - Freezer: copia de seguridad, restauración y recuperación ante desastres.
- API Proxies.
    - EC2API: EC2 API proxy.
- Interfaz web.
    - Horizon: dashboard.

HERRAMIENTAS DE OPERACIONES

- Servicios de monitorización.
    - Ceilometer: Servicio de medición y recopilación de datos.
    - Panko: Evento, servicio de indexación de metadatos.
    - Monasca: Monitorización.
- Optimización de recursos.
    - Watcher: Servicio de optimización.
    - Vitrage: Servicio de análisis de causa raíz.
- Facturación/Lógica empresarial.
    - Adjutant: Automatización de procesos operativos.
    - Cloudkitty: Facturación y devoluciones de cargo.

- Pruebas/Benchmark.
    - Rally: Herramienta de evaluación comparativa.
    - Tempest: Conjunto de pruebas de integración de OpenStack.
    - Patrole: Conjunto de pruebas de integración de OpenStack RBAC.

COMPLEMENTOS DE LOS SERVICIOS

- Complementos del servicio Swift.
    - Storlets: Almacenamiento de objetos computables.

FACILITADORES DE INTEGRACIÓN

- Contenedores.
    - Kuryr: Integración de redes OpenStack para contenedores.
- NFV.
    - Tacker: Orquestación de NFV.

3.2. Componentes básicos

Hay seis servicios estables que son básicos y gestionan la informática, las conexiones de red, el almacenamiento, la identidad, las imágenes y una gran variedad de servicios opcionales según la consolidación del desarrollo.

Estos seis servicios principales son los que constituyen la infraestructura que permite al resto de los proyectos gestionar los paneles, la coordinación, el aprovisionamiento de equipos sin sistema operativo, la mensajería, los contenedores y la gobernabilidad.

3.2.1. Nova

Es un servicio de computación en la nube [21], que proporciona al proyecto OpenStack una forma de aprovisionar instancias o servidores virtuales.

Nova permite la creación de máquinas virtuales, servidores Baremetal/Ironic y proporciona un soporte limitado para los contenedores del sistema.

Nova se ejecuta como un conjunto de demonios sobre los servidores Linux para proporcionar el servicio.

Este servicio requiere la instalación de componentes adicionales para poder desempeñar correctamente sus funcionalidades básicas. Estos componentes son: Keystone, Placement, Neutron y Glance.

Además para facilitar el uso del servicio de computación Nova, es aconsejable instalar también los servicios Horizon y Openstack client.

3.2.2. Neutron

Neutron es el servicio de OpenStack que proporciona conexión de red entre dispositivos de interfaz administrados por otros servicios OpenStack.

Es un sistema para la gestión de redes y direcciones IP que ofrece a los usuarios un autoservicio real a través de sus configuraciones de red.

Las direcciones IP flotantes [22] que proporciona este servicio permiten que el tráfico sea dirigido dinámicamente a cualquiera de sus recursos informáticos.

El servicio Neutron permite a los usuarios crear sus propias redes, controlar el tráfico y conectar los servidores y dispositivos a una o más redes.

Requiere la instalación previa del servicio de computación en la nube Nova y es el responsable de aprovisionar las redes virtuales o físicas a las que se conectan las instancias de cómputo durante el arranque.

3.2.3. Swift

El almacenamiento de objetos OpenStack (Swift) se utiliza para el almacenamiento de datos escalable y redundante. Es un sistema de almacenamiento a largo plazo para grandes cantidades de datos estáticos que se pueden recuperar y actualizar [23].

Los objetos se almacenan en varios dispositivos de hardware, y el software OpenStack es el responsable de garantizar la integridad de estos.

Los clústeres de almacenamiento se escalan horizontalmente agregando nuevos nodos. Si un nodo falla, OpenStack trabaja para replicar su contenido de otros nodos activos.

Debido a que OpenStack utiliza lógica de software para garantizar la replicación y distribución de datos en diferentes dispositivos, Swift es ideal para un almacenamiento rentable y escalable.

Es un componente independiente a los demás servicios.

3.2.4. Cinder

Cinder es un componente OpenStack que proporciona almacenamiento en bloque como servicio.

Este servicio virtualiza la gestión de dispositivos [24] de almacenamiento en bloque y proporciona a los usuarios finales una API de autoservicio para solicitar y consumir esos recursos sin necesidad de saber dónde se encuentra realmente implementado su almacenamiento o en qué tipo de dispositivo. Esto se hace mediante el uso de una implementación de referencia (LVM) o controladores de complementos para otro almacenamiento.

El almacenamiento de bloques es recomendado para las situaciones donde el rendimiento es sensible [22], como 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.

El servicio Cinder requiere la instalación del componente Keystone para un buen desarrollo de sus funcionalidades.

3.2.5. Keystone

Es el servicio de identidad de OpenStack.

Keystone proporciona autenticación de cliente API [25], descubrimiento de servicios y autorización distribuida de múltiples inquilinos mediante la API de identidad de OpenStack.

Es compatible con múltiples formas de autenticación: LDAP, OAuth, OpenID Connect, SAML y SQL.

Es un componente independiente a los demás servicios.

3.2.6. Glance

Los servicios de imágenes [26] de Openstack (Glance) permiten descubrir, registrar y recuperar imágenes de máquinas virtuales. Las imágenes almacenadas se pueden utilizar como plantillas.

Además este servicio permite almacenar y catalogar un número ilimitado de copias de seguridad [22]. Ofrece la oportunidad de almacenar imágenes de disco y de servidores en una gran variedad de ubicaciones, desde sistemas de archivos simples hasta incluso sistemas de almacenamiento de objetos OpenStack (Swift).

La API de servicios de imagen (GLANCE) es una API RESTful [26] que permite consultar información sobre las imágenes de VM, así como la recuperación de la imagen real y permite a los usuarios transmitirlas a nuevos servidores.

Este servicio muestra dependencia del componente Keystone para un buen desarrollo de sus funcionalidades.

3.3. Diferentes métodos de instalación

Existe una gran variedad de métodos de instalación de OpenStack. Hay métodos que ofrecen una implementación con limitación de recursos y otros que permiten un despliegue más avanzado.

En este apartado se va a realizar una comparación entre tres de los diversos métodos de instalación, a mano (componente a componente), con JuJu, MicroStack y DevStack.

Se seleccionará la mejor opción según la dificultad de la instalación y los recursos que ofrecen para completar los objetivos propuestos en el proyecto.

COMPONENTE A COMPONENTE

Este método es el que más recursos y posibilidades ofrece, pero también es el más costoso. Su implementación conllevaría demasiado tiempo ya que hay que ir instalando sus componentes uno a uno y seguramente serían necesarios más equipos. Por lo tanto, queda totalmente descartado.

CON JUJU

Este método es bastante completo ya que canonical ofrece una colección de charms de juju para OpenStack, con los que se consigue una implementación a escala de forma muy rápida y confiable de una nube.

El problema de esta instalación es que son necesarios mas recursos, ya que se necesitan varios equipos más porque además de juju, es necesario instalar MAAS. Los equipos necesarios serian:

- El servidor MAAS.
- El cliente juju.
- El controlador juju.
- Y todos los nodos de la nube

Debido al límite de recursos y la necesidad de incrementar el número de equipos para llevar a cabo esta instalación, este método aunque sea muy rápido y completo, queda descartado.

MICROSTACK

De primeras es el método que más conviene. Es una solución que ofrece una instalación muy rápida y simple instalando un paquete snap. Proporciona los servicios Nova, Keystone, Glance, Horizon y Neutron.

Para realizar la instalación con MicroStack se requieren:

- 2 CPU.
- 8 Gb de memoria RAM.
- 100 Gb de espacio en disco
- Una versión de Ubuntu 18.04 LTS o superior.

Como se muestra en apartados posteriores se llevó a cabo la instalación con MicroStack, pero, después se desestimó debido a que es un entorno que todavía está en desarrollo y sólo se puede ejecutar localmente, de modo que no se puede acceder por ssh a las instancias desde cualquier equipo situado en la misma red.

DEVSTACK

DevStack es una alternativa de instalación de OpenStack que quizá es más lenta que con MicroStack y algo menos sencilla, pero sigue siendo un método muy eficaz, rápido y simple para el despliegue. DevStack está compuesto por unos scripts que facilitan la instalación. Proporciona los servicios Nova, Cinder, Glance, Keystone, Neutron y Horizon.

Para realizar la instalación con DevStack se requieren:

- 1 CPU.
- 1 Gb de memoria RAM.
- 60 Gb de espacio en disco
- Es preferible una versión de Ubuntu 18.04 LTS o superior.

Debido a que se necesitan menos recursos para su instalación y que si ofrece salida a la red externa para poder acceder a las instancias por ssh desde cualquier equipo conectado a la misma red, es el método de instalación elegido para el despliegue aunque ofrezca un proceso algo más costoso en cuanto tiempo y a dificultad que MicroStack. Es el más adecuado para completar los objetivos propuestos en este proyecto.

3.4. Despliegue de OpenStack por medio de MicroStack

MicroStack ofrece una implementación de OpenStack de uno o varios nodos que puede ejecutarse directamente en su estación de trabajo. Es un entorno de prueba pero también está diseñado para dispositivos periféricos, loT y dispositivos.

MicroStack proporciona una instalación de OpenStack muy rápida y sencilla ya que todos los servicios de OpenStack y las bibliotecas de apoyo están empaquetados en un solo paquete que se puede instalar, actualizar o eliminar fácilmente. MicroStack incluye todos los componentes clave de OpenStack: Keystone, Nova, Neutron, Glance y Cinder.

Requisitos.

Para instalar MicroStack y que tenga un funcionamiento correcto se necesita:

- Una máquina corriendo Linux:
  - Con un procesador multinúcleo.
  - Con al menos 8 gb de RAM.

Configuración de los equipos.

Los sistemas operativos recomendados en los que MicroStack se ejecutará sin errores son Ubuntu 18.04 LTS y Ubuntu 20.04 LTS.

Para la instalación, el sistema operativo escogido ha sido Ubuntu server 20.04 LTS, el cual hemos instalado desde cero por medio de un USB booteable utilizando la aplicación RUFUS.

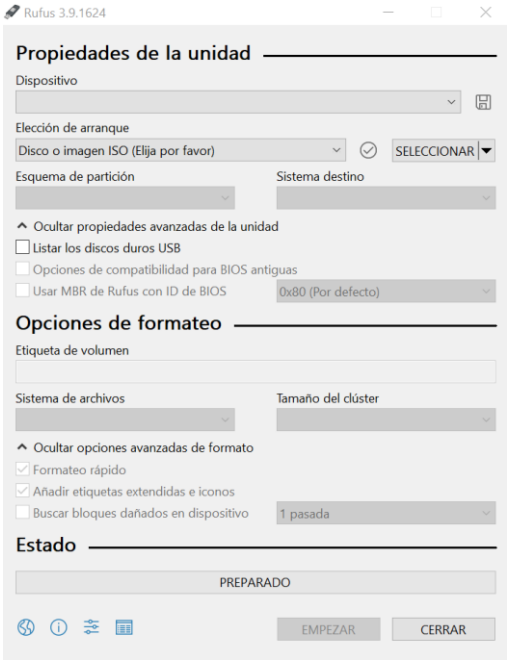


Imagen 7: Rufus MicroStack

Basta con seleccionar el dispositivo, elegir la imagen con la que queremos preparar el USB booteable y darle a empezar.

Después de tener preparado el USB booteable lo insertamos en el equipo y accedemos a la bios, desde esta, seleccionamos el USB y elegimos la opción “instalar Ubuntu”, a partir de aquí, será suficiente con seguir los pasos que nos van apareciendo para la configuración del sistema.

PASOS PREVIOS A LA INSTALACIÓN DE MICROSTACK:

Una vez instalado y configurado el sistema operativo, se configura la red para un correcto funcionamiento de la nube.

Lo primero de todo es actualizar el sistema. Para ello se ejecutan los siguientes comandos:

```
sudo apt update
```

```
sudo apt upgrade
```

Después de haber terminado de actualizar el sistema, se accede al archivo de configuración de red:

```
sudo nano /etc/netplan/00-installer-config.yaml
```

En este, se va a configurar la ip estática tanto del nodo de control, como de los nodos de cómputo, en ambos equipos la configuración es análoga, únicamente cambiando en cada uno la dirección ip.

A continuación, se muestra como ejemplo el archivo de configuración del nodo de control:

```
# This is the network config written by 'subiquity'
network:
  ethernets:
    eno1:
      dhcp4: no
      dhcp6: no
      addresses: [192.168.1.68/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [192.168.1.1, 8.8.8.8, 8.8.4.4]
  version: 2
  renderer: networkd
```

Imagen 8: netplan controller MicroStack

Para establecer esta configuración se deben ejecutar los siguientes comandos:

- Primero vamos a comprobar que es correcta y aceptada la configuración establecida mediante el comando:

```
sudo netplan try
```

- Una vez comprobado que en nuestra configuración no hay errores. Se procede a aplicar la configuración. Para ello introducimos los comandos:

```
sudo netplan apply
```

```
sudo netplan generate
```

Tras haber aplicado la configuración de la dirección ip de forma estática. Mediante el comando:

```
ip -c a
```

Podemos comprobar que en la interfaz escogida se ha configurado la ip deseada.

Y para comprobar que se tiene acceso a internet con la nueva configuración, simplemente se hace un ping a Google:

```
ping www.google.com
```

Tras finalizar la configuración de los equipos previa a la instalación de MicroStack, se procede con la instalación de la nube.

Instalación de MicroStack.

PASO 1: DESCARGA E INSTALACIÓN

Se descarga e instala MicroStack. Se puede instalar desde varios canales.

El canal elegido para la instalación se muestra en el comando que se ejecuta:

```
sudo snap install microstack --beta --devmode
```

Cuando el proceso de instalación ha terminado se ve el siguiente mensaje en el terminal:

"microstack (beta) ussuri from Canonical✓ installed"

PASO 2: INICIALIZACIÓN NODO DE CONTROL

Este paso consiste en inicializar MicroStack para que se configuren las redes y las bases de datos. Para esto se ejecuta el siguiente comando en el nodo control:

```
sudo microstack init --auto --control
```

Con este comando se especifica que la inicialización es automática y además, se indica en el equipo que se ejecuta que es el nodo de control.

PASO 3: AÑADIR NODOS DE CÓMPUTO

A continuación se le indica al nodo de control que se procede a añadir los nodos de cómputo.

Esto se realiza mediante el siguiente comando:

```
microstack add-compute
```

Este comando proporciona un mensaje en el terminal de una cadena de caracteres que debe ser introducida al inicializar el nodo de cómputo. Esta cadena está activa durante 20 minutos para su utilización, en caso contrario deberá ser repetido el comando.

Para facilitar la realización de este paso, lo mejor es conectarse al nodo de control para introducir el comando y al nodo de cómputo que queremos inicializar mediante ssh, ya que así podremos copiar y pegar la cadena debido a su gran longitud.

PASO 4: INICIALIZACIÓN NODOS DE CÓMPUTO

Este procedimiento, también consiste en la inicialización, pero en este caso se van a inicializar los dos nodos de cómputo utilizando la cadena de caracteres obtenida en el paso previo.

El proceso es similar en ambos nodos de cómputo:

Primero se introduce el siguiente comando para comenzar la inicialización:

```
sudo microstack init
```

Una vez introducido el comando, el en terminal se muestra un mensaje para introducir la cadena de caracteres correspondiente obtenida en el procedimiento previo.

Se introduce la cadena y en pocos minutos el nodo de cómputo es inicializado.

PASO 5: INTERACTUAR CON OPENSTACK

Se puede interactuar con openstack a través de la web GUI accediendo al Dashboard o por medio del CLI.

Bastará con introducir en el navegador nuestra ip del nodo de control: http://192.168.1.68/

Aparecerá esta página para introducir los credenciales:

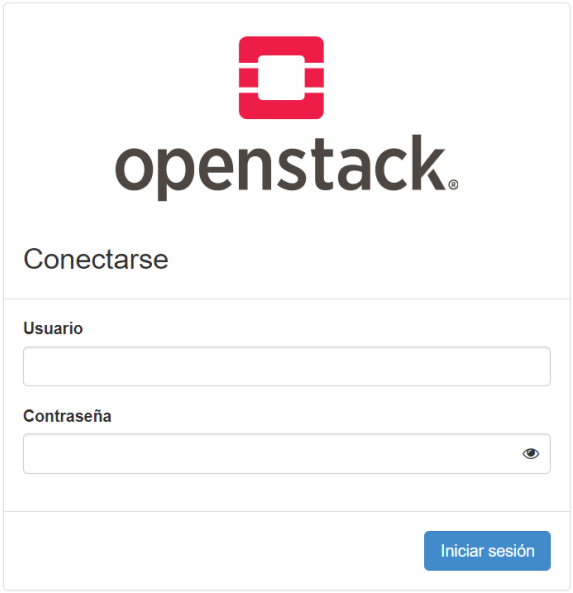


Imagen 9: Dashboard MicroStack

El usuario será “admin” y la contraseña hay que generarla desde el nodo de control ejecutando el siguiente comando:

```
sudo snap get microstack config.credentials.keystone-password
```

Se genera una contraseña por defecto que será mostrada en el terminal, por ejemplo:

“OAEHxLgCBz7Wz4usvolAAt61TrDUz6zz” y ya se puede acceder a la nube.

También se puede interactuar con OpenStack por medio de la línea de comandos.

```
microstack.openstack
```

Este comando permite acceder al terminal de OpenStack donde se pueden ejecutar diferentes comandos para realizar diferentes operaciones o comprobaciones.

Se puede crear un alias para acceder al CLI y evitar escribir microstack.openstack cada vez que escribamos un comando. Para ello introducimos en el terminal:

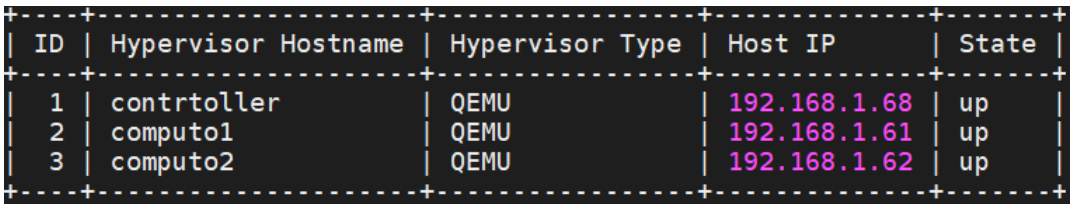
```
sudo snap alias microstack.openstack openstack
```

El comando sustituye microstack.openstack por openstack por lo que ahora ejecutando openstack accederemos al terminal.

Hay una gran variedad de comandos para ejecutar, podemos crear redes, usuarios, roles, routers, así como sólo listarlos.

Algunos ejemplos de los comandos que se pueden ejecutar son:

```
openstack hypervisor list
```



ID	Hypervisor Hostname	Hypervisor Type	Host IP	State
1	contrtoller	QEMU	192.168.1.68	up
2	computo1	QEMU	192.168.1.61	up
3	computo2	QEMU	192.168.1.62	up

Imagen 10: Hypervisor list MicroStack

Este comando nos muestra la lista de hipervisores.

También se puede utilizar para comprobar que todos los nodos, tanto el de control, como los de cómputo, se han añadido correctamente.

Además muestra el estado de cada uno de los nodos.

```
openstack catalog list
```

```
+-----------+-----------+-----------------------------------------------------------------------------+
| Name      | Type      | Endpoints                                                                   |
+-----------+-----------+-----------------------------------------------------------------------------+
| nova      | compute   | microstack                                                                  |
|           |           |   internal: http://192.168.1.68:8774/v2.1                                   |
|           |           | microstack                                                                  |
|           |           |   public: http://192.168.1.68:8774/v2.1                                     |
|           |           | microstack                                                                  |
|           |           |   admin: http://192.168.1.68:8774/v2.1                                      |
|           |           |                                                                             |
| neutron   | network   | microstack                                                                  |
|           |           |   public: http://192.168.1.68:9696                                          |
|           |           | microstack                                                                  |
|           |           |   internal: http://192.168.1.68:9696                                        |
|           |           | microstack                                                                  |
|           |           |   admin: http://192.168.1.68:9696                                           |
|           |           |                                                                             |
| cinderv2  | volumev2  | microstack                                                                  |
|           |           |   internal: http://192.168.1.68:8776/v2/c6387add556e4e10b400ee14219df97d    |
|           |           | microstack                                                                  |
|           |           |   public: http://192.168.1.68:8776/v2/c6387add556e4e10b400ee14219df97d      |
|           |           | microstack                                                                  |
|           |           |   admin: http://192.168.1.68:8776/v2/c6387add556e4e10b400ee14219df97d       |
|           |           |                                                                             |
| placement | placement | microstack                                                                  |
|           |           |   admin: http://192.168.1.68:8778                                           |
|           |           | microstack                                                                  |
|           |           |   internal: http://192.168.1.68:8778                                        |
|           |           | microstack                                                                  |
|           |           |   public: http://192.168.1.68:8778                                          |
|           |           |                                                                             |
| keystone  | identity  | microstack                                                                  |
|           |           |   public: http://192.168.1.68:5000/v3/                                      |
|           |           | microstack                                                                  |
|           |           |   admin: http://192.168.1.68:5000/v3/                                       |
|           |           | microstack                                                                  |
|           |           |   internal: http://192.168.1.68:5000/v3/                                    |
|           |           |                                                                             |
| glance    | image     | microstack                                                                  |
|           |           |   admin: http://192.168.1.68:9292                                           |
|           |           | microstack                                                                  |
|           |           |   internal: http://192.168.1.68:9292                                        |
|           |           | microstack                                                                  |
|           |           |   public: http://192.168.1.68:9292                                          |
|           |           |                                                                             |
| cinderv3  | volumev3  | microstack                                                                  |
|           |           |   public: http://192.168.1.68:8776/v3/c6387add556e4e10b400ee14219df97d      |
|           |           | microstack                                                                  |
|           |           |   internal: http://192.168.1.68:8776/v3/c6387add556e4e10b400ee14219df97d    |
|           |           | microstack                                                                  |
|           |           |   admin: http://192.168.1.68:8776/v3/c6387add556e4e10b400ee14219df97d       |
|           |           |                                                                             |
+-----------+-----------+-----------------------------------------------------------------------------+
```

Imagen 11: Catalog list MicroStack

Este comando sirve para listar los servicios del catálogo de servicios.

Como podemos observar al instalar microstack, se instalan automáticamente los servicios:

Nova, Neutron, Cinderv2, Placement, Keystone, Glance y Cinderv3.

Además, este comando nos muestra el tipo de servicio que es cada uno y la url para acceder a cada servicio.

Parar y arrancar de nuevo los servicios de la nube.

Cada vez que queramos apagar los equipos deberemos ejecutar primero en los nodos de cómputo y una vez completado en el nodo de control el siguiente comando:

```
sudo snap disable microstack
```

Cuando volvamos a encender los equipos deberemos introducir el siguiente comando pero en el orden contrario en los equipos:

```
sudo snap enable microstack
```

Una vez completado todo eso deberemos inicializar de nuevo los equipos, pero el tiempo que se tardará en completarse este paso será notablemente inferior que el empleado al instalar microstack por primera vez.

Lanzar una instancia.

Una vez se ha terminado la instalación y se ha comprobado que se puede interactuar con la nube correctamente, ya se puede operar con ella.

Una de las operaciones básicas que se debe poder realizar en una nube es lanzar una instancia.

Para ello se utiliza el comando:

```
microstack launch cirros --name test
```

Este comando lanza una instancia de nombre “test” y que utiliza la imagen “cirros”.

Si la instancia se ha lanzado correctamente, en el terminal se observa el siguiente mensaje:

```
Creating local "microstack" ssh key at /home/mario/snap/microstack/common/.ssh/id_microstack
Launching server ...
Allocating floating ip ...
Server test launched! (status is BUILD)

Access it with `ssh -i /home/mario/snap/microstack/common/.ssh/id_microstack cirros@10.20.20.129`
```

Imagen 12: Lanzar instancia MicroStack

Después de haber lanzado la instancia, para que esta tenga acceso a internet, se ejecutan los siguientes comandos

```
sudo iptables -t nat -A POSTROUTING -s 10.20.20.1/24 ! -d 10.20.20.1/24 -j MASQUERADE
```

```
sudo sysctl net.ipv4.ip_forward=1
```

Microstack configura por defecto los grupos de seguridad que se aplican a las instancias entre otras cosas:

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	-	Delete Rule
Egress	IPv6	Any	Any	::/0	-	-	Delete Rule
Ingress	IPv4	Any	Any	-	default	-	Delete Rule
Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv6	Any	Any	-	default	-	Delete Rule

Displaying 6 items

Imagen 13: Grupos de Seguridad MicroStack

Como se observa, está activado el acceso por ssh.

Para acceder por ssh a la instancia se ejecuta el siguiente comando:

```
ssh cirros@10.20.20.129
```

Una vez ejecutado el comando, pide que se introduzca la contraseña.

La contraseña por defecto es: “gocubsgo”.

Otra manera de acceder a la instancia es a través del Dashboard.

Los pasos a seguir son los siguientes: Project -> Compute -> Instances.

Cuando muestra el listado de instancias lanzadas, se selecciona la instancia a la que se quiere acceder, en este caso la de nombre “test”.

En la parte derecha de la instancia, se accede al menú desplegable y se selecciona “consola”.

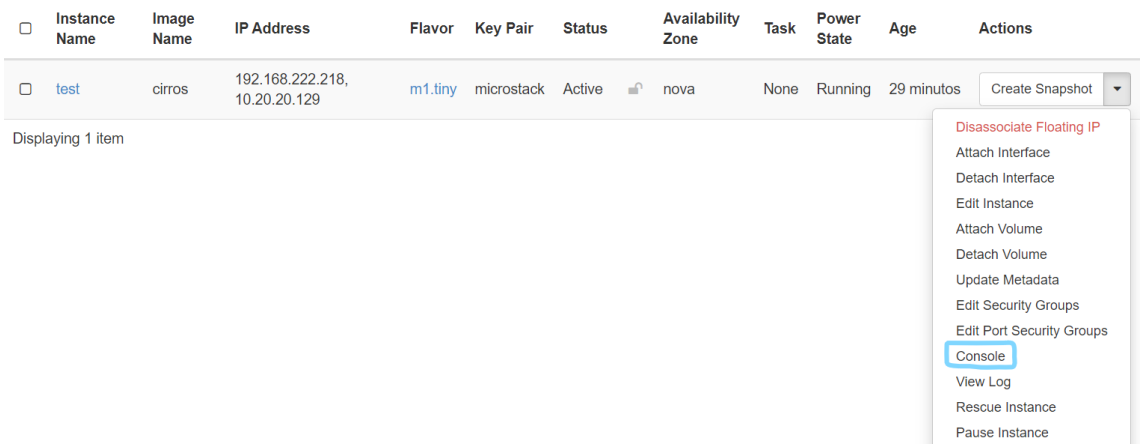


Imagen 14: Acceder a instancia MicroStack

Una vez se accede a la consola se pide el nombre de usuario y contrasella. Los credenciales por defecto son:

Usuario: “cirros”.

Contraseña: “gocubsgo”.

Una vez se ha conseguido acceder a la consola, se debe comprobar que esta tiene acceso a internet y funciona correctamente.

Para comprobar que tiene conexión a internet se introduce el comando:

```
Ping 8.8.8.8
```

Para desconectarse de la instancia bastará con ejecutar:

```
exit
```

De forma análoga estos pasos y comprobaciones se realizan cuando accedemos a la instancia por ssh.

A continuación se muestra una imagen en la que se recoge la topología de red:

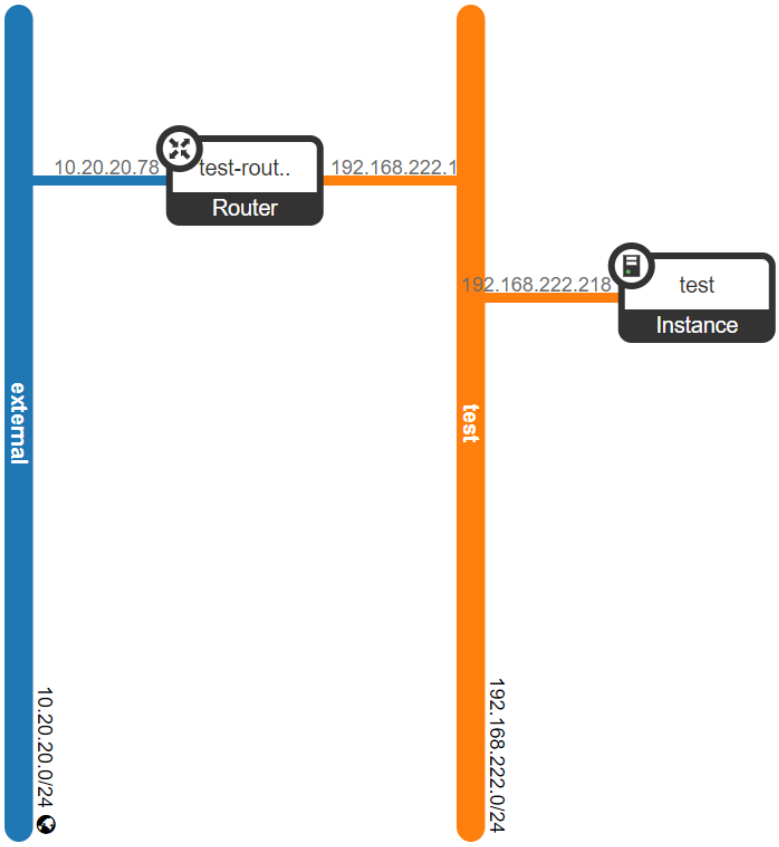


Imagen 15: Topología de red MicroStack

En la imagen se muestra que la instancia está lanzada en la red “test”, la cual está conectada a la red pública “external” por medio de un router.

Como MicroStack es un entorno de prueba configura la red externa 10.20.20.0/24, por lo que no se puede acceder a las instancias desde cualquier equipo situado en la misma red en la que están el nodo de control y los nodos de cómputo, pues para esto la red “external” debería ser la misma, 192.168.1.0/24.

Debido a esto, la implementación por medio de MicroStack no cumple los objetivos propuestos en el desarrollo de la nube. Por lo que se decide implementar instalando DevStack.

3.5. Despliegue de OpenStack por medio de DevStack

Devstack está formado por una serie de script extensibles que son utilizados para levantar rápidamente un entorno OpenSatck completo basado en las últimas versiones de todo, desde el repositorio de git master. Se utiliza de forma interactiva como entorno de desarrollo y como base para gran parte de las pruebas funcionales del proyecto OpenStack.

Para instalar devstack es necesario comenzar con una instalación mínima de un sistema Linux.

La versión más probada y recomendable para el sistema operativo es Ubuntu 18.04.

Se decide realizar el despliegue en un entorno formado por tres equipos, en el cual uno será el nodo controlador y los otros dos serán los nodos de cómputo.

El sistema operativo elegido para el despliegue es la versión 20.04 de Ubuntu-Desktop debido a que en cualquier versión de Ubuntu server al realizar el despliegue aparecía un error del sistema, en el terminal, que se repetía periódicamente.

3.5.1. Configuración de los equipos

El primer paso es descargar la imagen de internet y preparar un usb arrancable que será utilizado para instalar el sistema operativo en los tres equipos. Para ello configuramos el usb utilizando la aplicación Rufus.

Esta aplicación tiene un uso muy sencillo, pues sólo hay que seleccionar el dispositivo usb que queremos configurar con la imagen deseada y empezar.

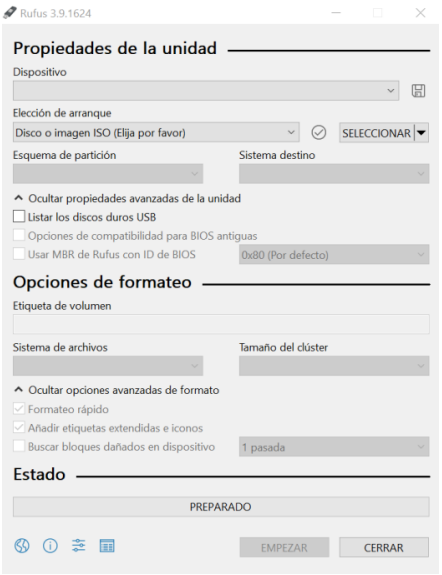


Imagen 16: Rufus DevStack

Una vez se ha terminado la configuración del usb, se introduce en el equipo, se accede a la bios, se selecciona el dispositivo e instalamos Ubuntu de cero siguiendo los pasos que nos van apareciendo.

La instalación del sistema operativo se debe realizar en los tres equipos que forman parte del despliegue.

Tras haberse instalado correctamente el sistema operativo, lo primero es actualizar los equipos utilizando los siguientes comandos:

```
sudo apt update
```

```
sudo apt upgrade
```

Después de haber actualizado los tres equipos, se procede con la configuración de red. En cada equipo se configuran las direcciones IP de forma estática.

Para acceder al archivo de configuración de red, se ejecuta en cada equipo:

```
sudo nano /etc/netplan/ 01-network-manager-all.yaml
```

En el nodo controlador se configuran dos interfaces de red, cada una con su IP correspondiente ya que serán necesarias para la posterior implementación de devstack.

A continuación se muestra el archivo de configuración del nodo controller:

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    eno1:
      dhcp4: no
      addresses: [192.168.1.83/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [192.168.1.1, 8.8.8.8, 8.8.4.4]
    enp2s0:
      dhcp4: no
      addresses: [192.168.1.84/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [192.168.1.1, 8.8.8.8, 8.8.4.4]
```

Imagen 17: netplan controller DevStack

Tras guardar el archivo, se debe aplicar la configuración. Para ello se ejecutan los siguientes comandos:

Para comprobar que la configuración no tiene errores:

```
sudo netplan try
```

Si la configuración es aceptada, hay que aplicarla y generarla con los comandos:

```
sudo netplan apply
```

```
sudo netplan generate
```

De forma análoga se configuran los archivos de red de los nodos de cómputo, en los cuales, por el contrario, solo será necesario el uso de un único interfaz en cada uno.

Archivo del computo1:

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    eno1:
      addresses: [192.168.1.86/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [192.168.1.1, 8.8.8.8, 8.8.4.4]
```

Imagen 18: netplan computo1 DevStack

Archivo del computo2:

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    eno1:
      addresses: [192.168.1.81/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [192.168.1.1, 8.8.8.8, 8.8.4.4]
```

Imagen 19: netplan computo2 DevStack

Y se aplican los tres mismos comandos en cada equipo, después de guardar los archivos para generar la nueva configuración:

```
sudo netplan try
```

```
sudo netplan apply
```

```
sudo netplan generate
```

Para comprobar que se tiene acceso a internet con la nueva configuración, simplemente se hace un ping a Google:

```
ping www.google.com
```

Tras finalizar la configuración de los equipos previa a la instalación de Devstack, se procede con el despliegue de la nube.

3.5.2. Instalación Devstack

NODO DE CONTROL

El paso del despliegue de la nube con Devstack se hace primero en el nodo controlador.

Para comenzar se actualiza el equipo con los comandos:

```
sudo apt update
```

```
sudo apt -y upgrade && sudo apt -y dist-upgrade
```

Después de haber actualizado correctamente el equipo, se procede a instalar git:

```
sudo apt -y install git
```

El siguiente paso a la instalación de git es la creación del usuario “Stack” y asignación a este de sudo privilegios.

Se crea un nuevo usuario no root llamado “Stack” para la instalación de Devstack:

```
sudo useradd -s /bin/bash -d /opt/stack -m stack
```

Al usuario “Stack” se le proporcionan privilegios sudo:

```
echo “stack ALL=(ALL) NOPASSWD: ALL” | sudo tee /etc/sudoers.d/stack
```

Una vez hemos creado el usuario y asignado a este ciertos privilegios se procede a la descarga de Devstack.

Para comenzar con la descarga lo primero es cambiar al usuario “Stack” con el comando:

```
sudo su - stack
```

Ya dentro del usuario “Stack”, se comprueba que efectivamente la ruta en la que se a realizar la descarga es la deseada (/opt/stack):

```
pwd
```

Si la ruta es la correcta, el siguiente paso es clonar el repositorio de Devstack, utilizando git (se debe especificar la versión estable que se quiere descargar, en este caso “victoria”, ya que es la versión estable más actual):

```
git clone https://opendev.org/openstack/devstack -b stable/victoria
```

Tras clonar el repositorio Devstack, se continua con la creación del archivo de configuración de devstack y la instalación.

Para crear el archivo de configuración, se accede al directorio Devstack:

```
cd devstack
```

Y dentro de este, se copia el archivo de configuración "local.conf" del directorio samples:

```
cp samples/local.conf .
```

Con el archivo "local.conf" ya copiado en el directorio devstack, se accede a dicho archivo para editarlo:

```
nano local.conf
```

En el archivo de configuración se deben añadir las siguientes líneas:

```
MULTI_HOST=1
HOST_IP=192.168.1.83
FLOATING_RANGE=192.168.1.0/24
Q_FLOATING_ALLOCATION_POOL=start=192.168.1.230,end=192.168.1.254
PUBLIC_NETWORK_GATEWAY=192.168.1.83
```

Imagen 20: local.conf controller parte 1

Con esas líneas se está indicando lo siguiente:

HOST_IP: ip configurada para el nodo controlador.
FLOATING_RANGE: indica el rango (la red) donde estarán alojadas las IPs flotantes, es decir, la red pública de la nube.
Q_FLOATING_ALLOCATION_POOL: indica el rango de direcciones que serán utilizadas como IPs flotantes.
PBLIC_NETWORK_GATEWAY: indica el gateway de la red pública, la red de acceso a internet.
Es la dirección ip del nodo controlador aplicada en uno de los interfaces configurados.
Será la dirección que se añade al interfaz br-ex cuando se ejecute la configuración.
MULTI_HOST: indica que es un laboratorio de múltiples nodos si su valor es uno.

Además de añadir estas líneas al archivo, también se editan las siguientes configuraciones por defecto:

```
ADMIN_PASSWORD=mariotfg
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
```

Imagen 21: local.conf controller parte 2

Al editar esas líneas lo que se está imponiendo es que la contraseña de administrador de la nube (ADMIN_PASSWORD), que se utiliza entre otras cosas para acceder al dashboard o utilizar la interfaz de línea de comandos CLI, sea "mariotfg".

Al terminar con el archivo de configuración, se inicia la instalación:

```
./stack.sh
```

Se sabrá que la instalación ha finalizado y se ha realizado correctamente cuando en el terminal aparezca lo siguiente:

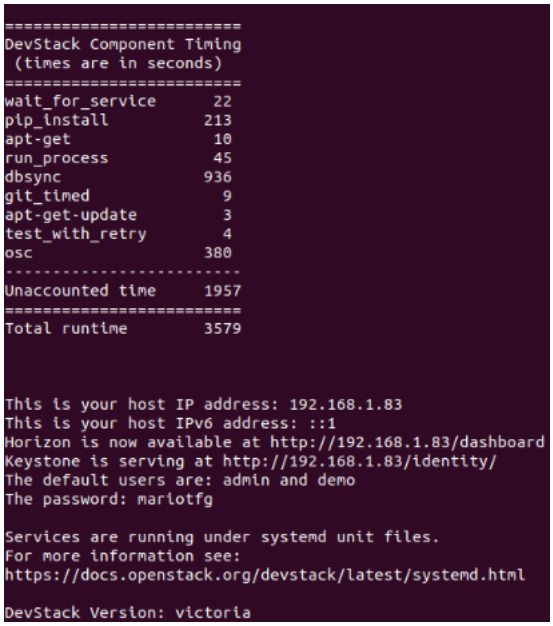


Imagen 22: Fin instalación DevStack controller

Una vez terminada la instalación, para confirmar la configuración se deben ejecutar los siguientes comandos:

Primero se elimina la dirección IP del interfaz eno1:

```
sudo ip addr del 192.168.1.83/24 dev eno1
```

Y seguido, se crea un puente de red entre el interfaz br-ex y el eno1:

```
sudo ovs-vsctl add-port br-ex eno1
```

Con esta configuración, al crear el puente de red, el nodo controlador quedaría aislado a internet y debido a esto se configuran únicamente en este nodo dos interfaces de red, lo que hace que el nodo controlador no pierda el acceso a internet en ningún momento.

Tras haber confirmado la instalación, se accede a la nube a través del buscador introduciendo la siguiente url:

https://server-ip/dashboard donde “server-ip” es la dirección ip del nodo controlador.

Para iniciar sesión en el dashboard, se utilizará el usuario “admin” y la contraseña configurada en el archivo de configuración “mariotfg”.

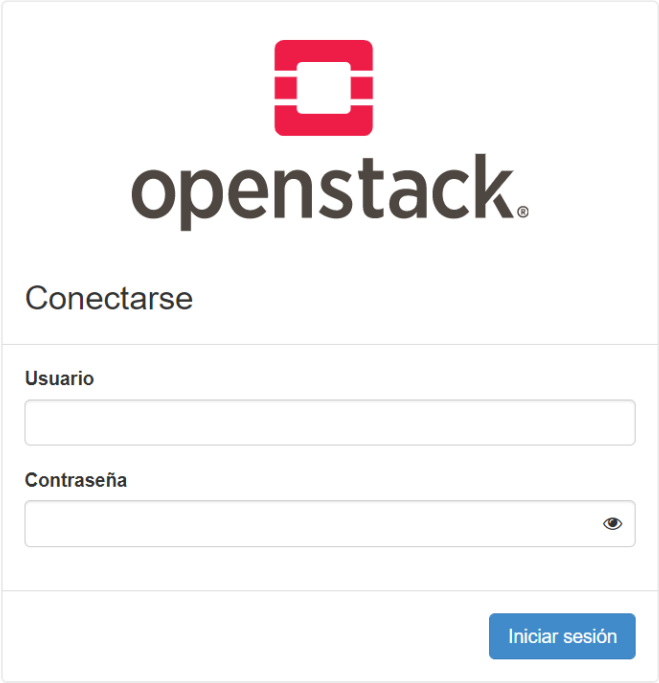


Imagen 23: Dashboard DevStack

NODOS DE CÓMPUTO

Tras haber terminado la instalación en el nodo controlador y haber comprobado que se tiene acceso a la nube a través del dashboard, se procede a añadir los nodos de cómputo.

En ambos nodos de cómputo casi todo el proceso de instalación es similar al realizado en el nodo controlador.

Primero se actualizan los equipos:

```
sudo apt update
```

```
sudo apt -y upgrade && sudo apt -y dist-upgrade
```

Seguido se instala git:

```
sudo apt -y install git
```

Tras la instalación de git, se crea el usuario “Stack” y se le asignan sudo privilegios:

```
sudo useradd -s /bin/bash -d /opt/stack -m stack
```

```
echo “stack ALL=(ALL) NOPASSWD: ALL” | sudo tee /etc/sudoers.d/stack
```

Después, se descarga Devstack:

```
sudo su - stack
```

```
pwd
```

```
git clone https://opendev.org/openstack/devstack -b stable/victoria
```

El siguiente paso antes de iniciar la instalación es crear el archivo de configuración:

```
cd devstack
```

```
cp samples/local.conf .
```

```
nano local.conf
```

En los archivos de configuración de cada nodo de cómputo añadimos y editamos las siguientes líneas:

Archivo cómputo1:

```
ADMIN_PASSWORD=mariotfg
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
```

Imagen 24: local.conf computo1 parte 1

```
MULTI_HOST=1
HOST_IP=192.168.1.86
FLOATING_RANGE=192.168.1.0/24
Q_FLOATING_ALLOCATION_POOL=start=192.168.1.249,end=192.168.1.254
PUBLIC_NETWORK_GATEWAY=192.168.1.83
DATABASE_TYPE=mysql
SERVICE_HOST=192.168.1.83
MYSQL_HOST=$SERVICE_HOST
RABBIT_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
ENABLED_SERVICES=n-cpu,q-agt,c-vol,placement-client
NOVA_VNC_ENABLED=True
NOVNCPROXY_URL="http://$SERVICE_HOST:6080/vnc_lite.html"
VNCSERVER_LISTEN=$HOST_IP
VNCSERVER_PROXYCLIENT_ADDRESS=$VNCSERVER_LISTEN
```

Imagen 25: local.conf computo1 parte 2

Archivo cómputo2:

```
ADMIN_PASSWORD=mariotfg
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
```

Imagen 26: local.conf computo2 parte 1

```
MULTI_HOST=1
HOST_IP=192.168.1.81
FLOATING_RANGE=192.168.1.0/24
Q_FLOATING_ALLOCATION_POOL=start=192.168.1.249,end=192.168.1.254
PUBLIC_NETWORK_GATEWAY=192.168.1.83
DATABASE_TYPE=mysql
SERVICE_HOST=192.168.1.83
MYSQL_HOST=$SERVICE_HOST
RABBIT_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
ENABLED_SERVICES=n-cpu,q-agt,c-vol,placement-client
NOVA_VNC_ENABLED=True
NOVNCPROXY_URL="http://$SERVICE_HOST:6080/vnc_lite.html"
VNCSERVER_LISTEN=$HOST_IP
VNCSERVER_PROXYCLIENT_ADDRESS=$VNCSERVER_LISTEN
```

Imagen 27: local.conf computo2 parte 2

Como se puede observar, la única diferencia en el archivo de configuración es la dirección IP de cada equipo.

Una vez configurado el archivo local.conf se inicia la instalación en cada cómputo:

```
./stack.sh
```

La instalación habrá terminado cuando en el terminal de cada nodo de cómputo se muestre lo siguiente:

Nodo de cómputo1:

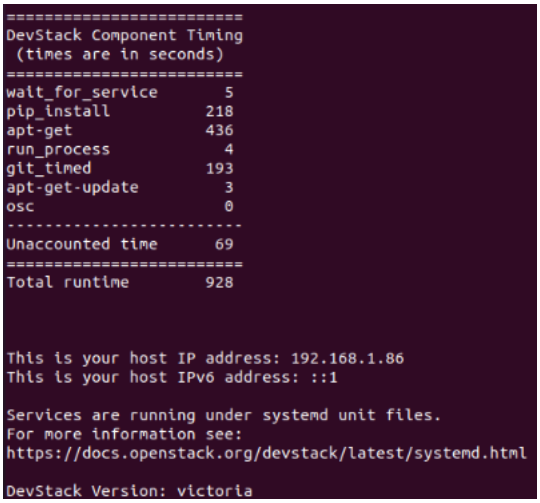


Imagen 28: Fin instalación DevStack computo1

Nodo de cómputo2:

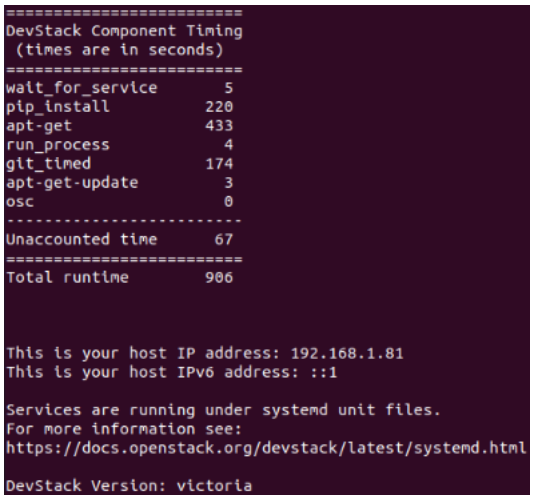


Imagen 29: Fin instalación DevStack computo2

Cuando termine la instalación en cada nodo de cómputo, se requiere una implementación de Cellsv2 para confirmar que los nodos de cómputos son añadidos a la nube.

Los servicios de nodo de cómputo deben ser asignados a una celda antes de que puedan utilizarse.

Esta implementación se realiza desde el nodo de control o nodo controlador desde el usuario stack, en el directorio devstack.

Para realizar esta configuración se ejecutan varios comandos. Como estos comandos que se ejecutan utilizan la base de datos de la API, antes se debe ejecutar:

```
nova-manage api_db sync
```

Una vez sincronizado, lo primero, se crean los registros necesarios para la base de datos cell0:

```
nova-manage cell_v2 map_cell0
```

Nunca hay hosts en el cell0 por lo que no se precisa nada más para su configuración.

Se debe crear otra celda que será la primera que albergue hosts de cómputo reales y en la cual las instancias se pueden programar.

Se crea el registro de celda:

```
nova-manage cell_v2 create_cell --verbose --name cell1
```

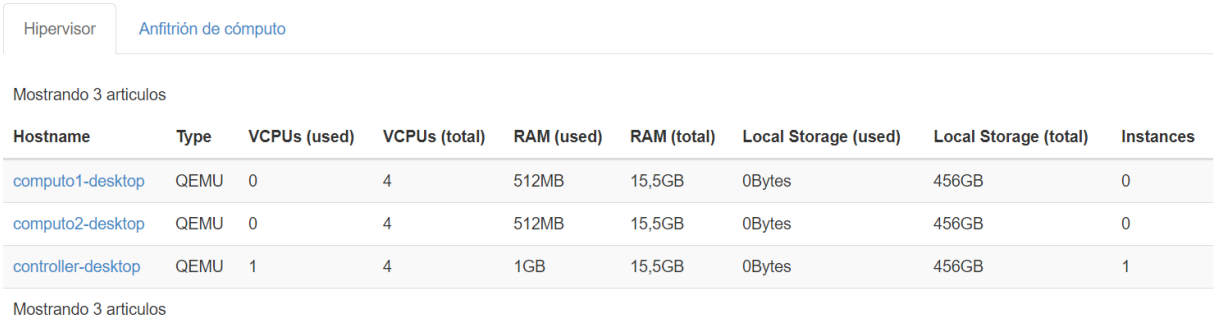
Como se está agregando una nueva celda, para inicializar el esquema de celdas se debe ejecutar:

```
nova-manage db sync
```

Por último se escanean los nodos de cómputo disponibles y se agregan a la celda. Para esto, es imprescindible haber iniciado anteriormente los nodos de cómputo de modo que se registren como un servicio en ejecución.

```
nova-manage cell_v2 discover_hosts
```

Se puede comprobar que se han añadido los nodos de cómputo correctamente a la nube desde el dashboard: Administrador->Compute->Hipervisores


Hipervisor | Anfitrión de cómputo

Mostrando 3 articulos

Hostname	Type	VCPUs (used)	VCPUs (total)	RAM (used)	RAM (total)	Local Storage (used)	Local Storage (total)	Instances
computo1-desktop	QEMU	0	4	512MB	15,5GB	0Bytes	456GB	0
computo2-desktop	QEMU	0	4	512MB	15,5GB	0Bytes	456GB	0
controller-desktop	QEMU	1	4	1GB	15,5GB	0Bytes	456GB	1

Mostrando 3 articulos

Imagen 30: Hypervicor list DevStack

3.5.3. Lanzar una Instancia

Para explicar cómo se lanza una instancia se usará el proyecto predeterminado “demo”.

Antes de lanzar una instancia se accede a la topología de red de este proyecto, haciendo uso del dashboard. Una vez iniciada la sesión en él, la ruta que se debe seguir es:

Proyecto->Red->Topología de red.

La topología de red del proyecto “demo” es la siguiente:

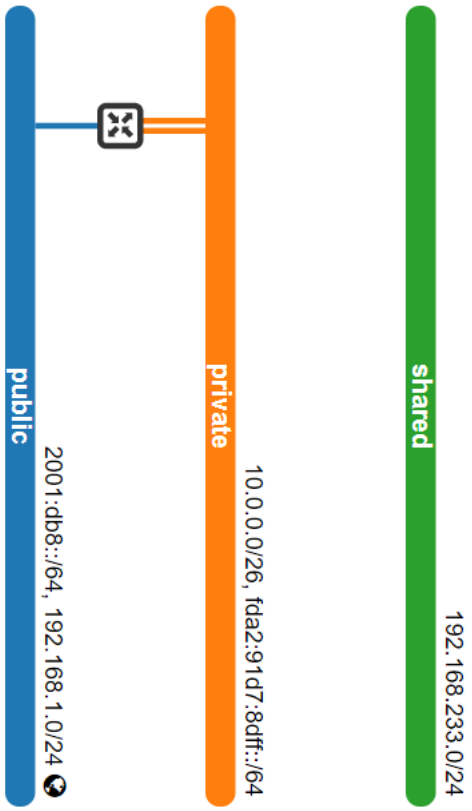


Imagen 31: Topología de red DevStack

El objetivo es lanzar una instancia en la red “private”, que es la que está conectada mediante un router a la red externa “public”.

Para lanzar una instancia se accede también desde el dashboard al apartado:

Proyecto->Compute->Instancias.

Una vez en esta ubicación, en la parte derecha de la pantalla, se selecciona “Lanzar Instancia”.

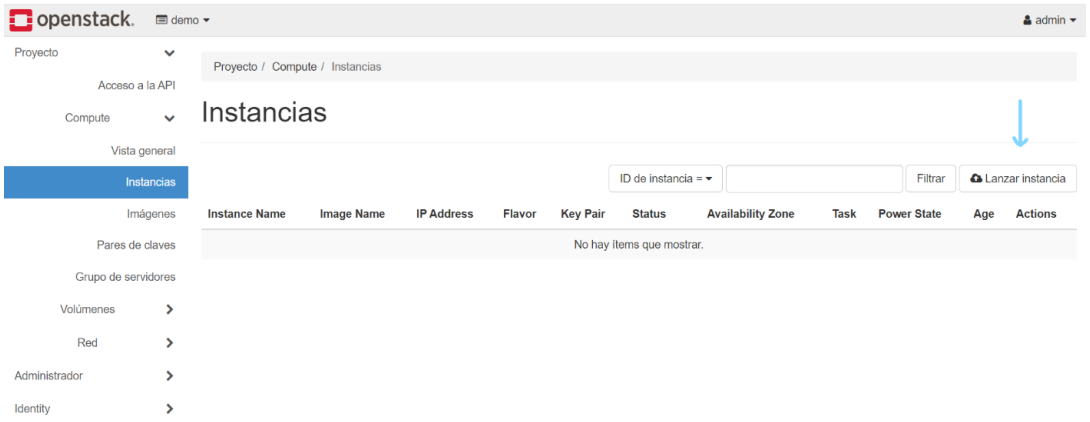


Imagen 32: lanzar instancia DevStack

Aparecerá un menú en el que se debe configurar la instancia antes de ser lanzada.

Lo primero que se debe editar, antes de ejecutar la instancia, es su nombre en el apartado detalles:

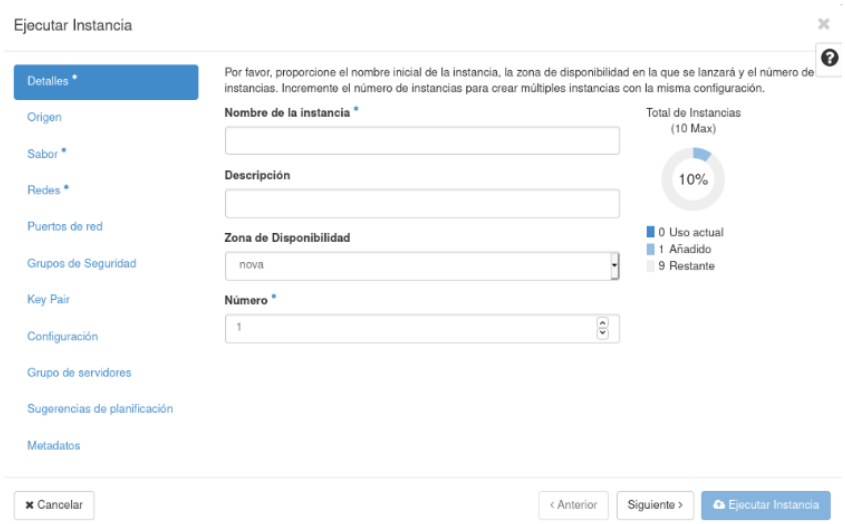


Imagen 33: Nombre instancia

Seguido, en el apartado origen, se elige la imagen que servirá como plantilla para la nueva instancia y también si se utilizará almacenamiento permanente con la creación de un nuevo volumen:

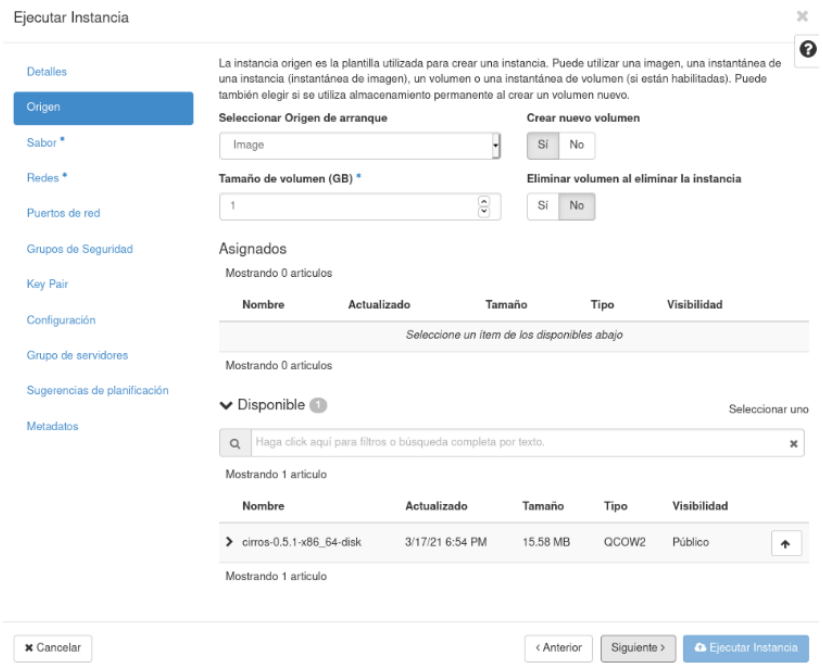


Imagen 34: imagen instancia

En este caso se selecciona la imagen “cirros”:

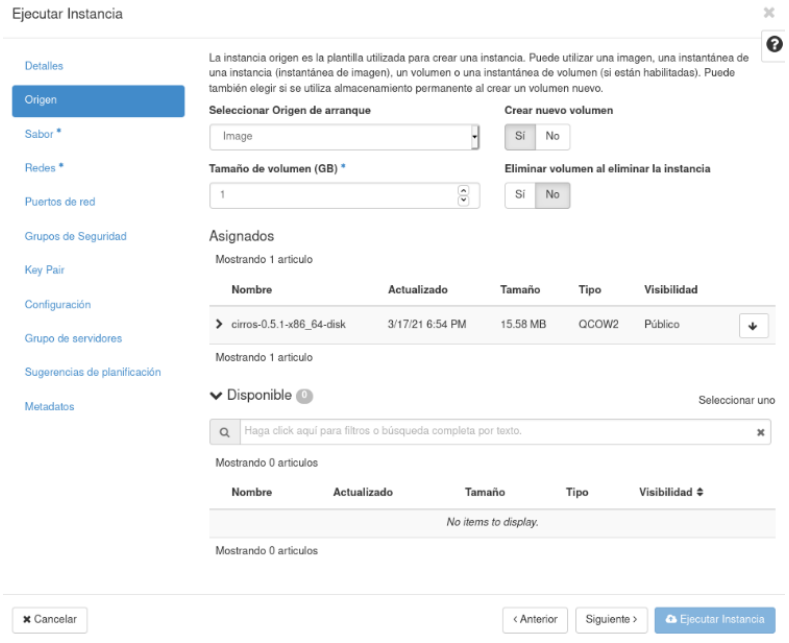


Imagen 35: imagen cirros

Una vez configurado el origen, se configura el sabor.

El sabor define el tamaño que tendrá la instancia respecto a CPU, memoria y almacenamiento.

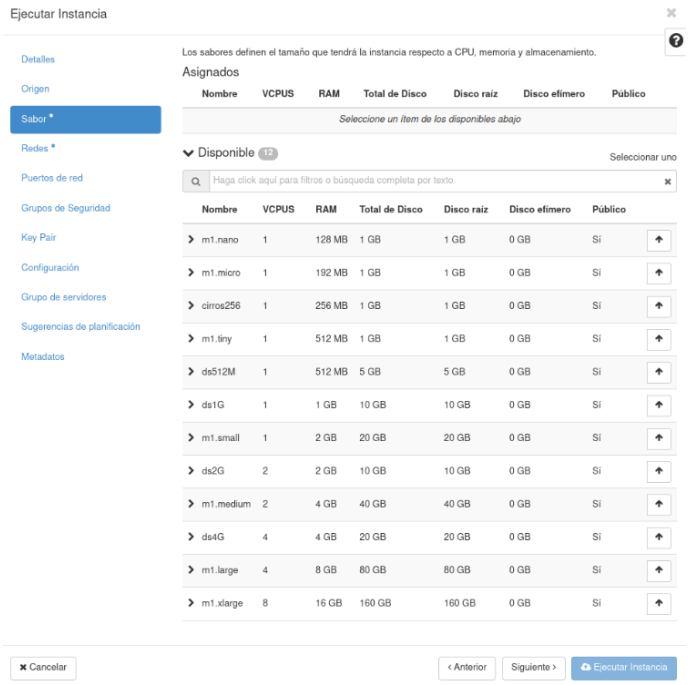


Imagen 36: Sabor instancia

En este caso, el sabor que se escoge es el “m1.tiny”.

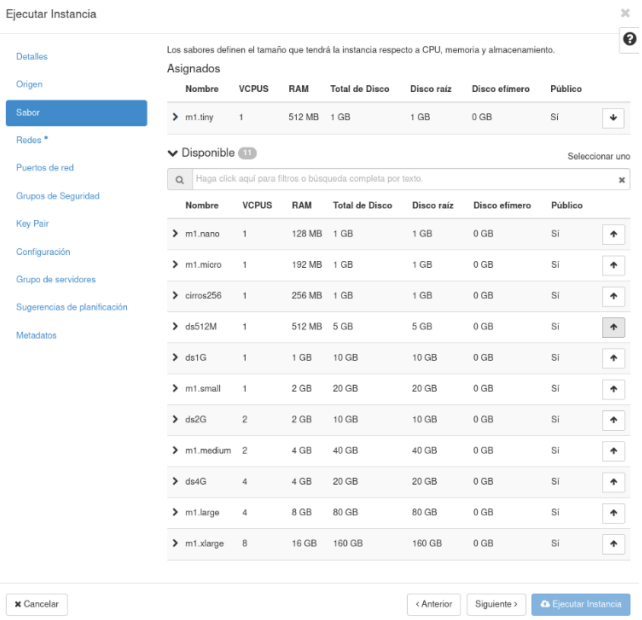


Imagen 37: Sabor m1.tiny

El último paso antes de ejecutar la instancia es elegir la Red.

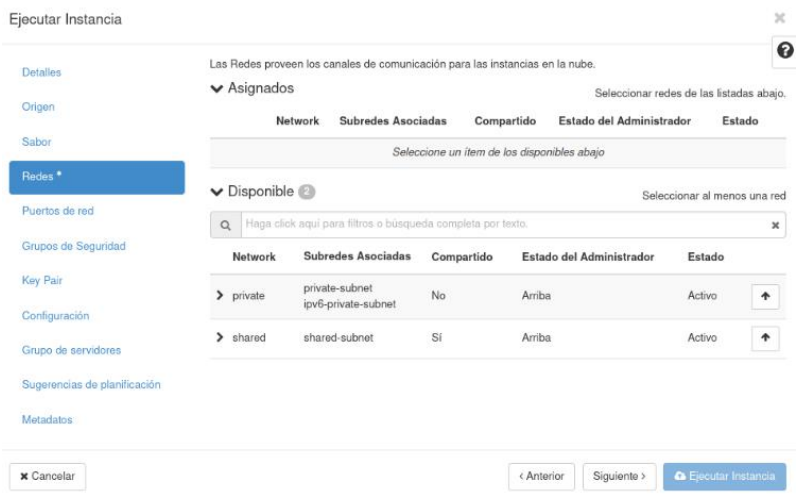


Imagen 38: Red instancia

Como bien se ha explicado antes, se debe elegir la red private para que nuestra instancia tenga acceso a la red.

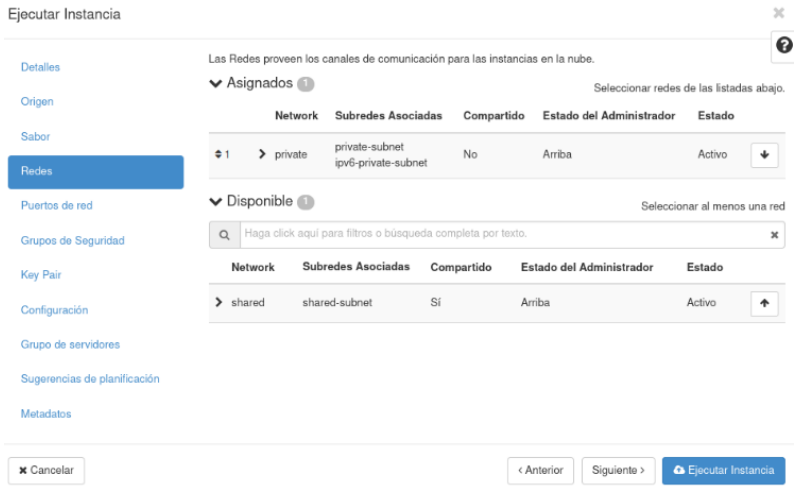


Imagen 39: Red privada

Una vez finalizado el paso de seleccionar la red en la que se va a lanzar la instancia se puede ejecutar la instancia.

Estos son los pasos básicos para ejecutar una instancia. Pero solo con esta configuración no es suficiente.

Para que la instancia salga correctamente a la red externa, desde el menú desplegable de la propia instancia, se debe “Asociar IP flotante”.

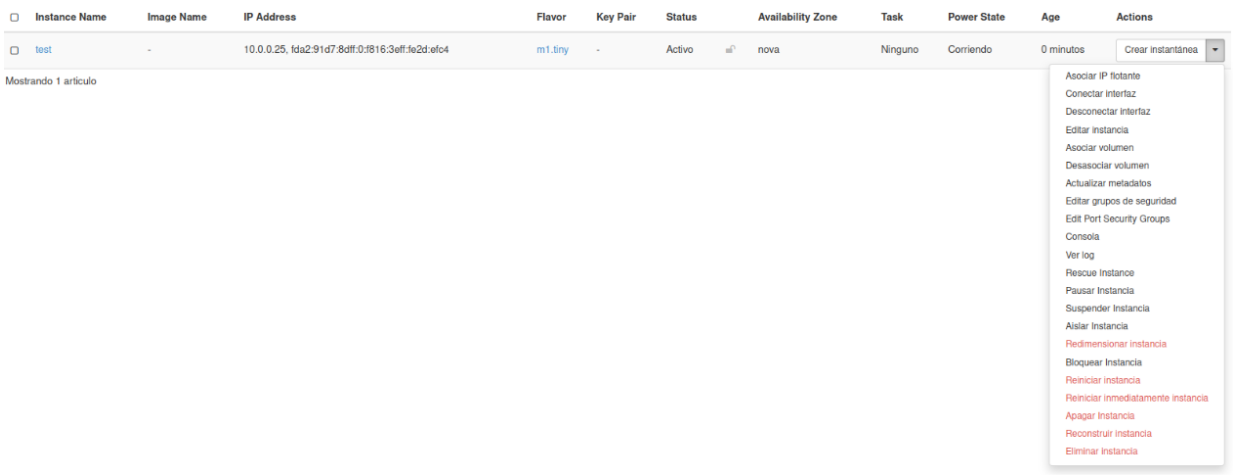


Imagen 40: IP flotante

Al seleccionar esta IP, aparece un nuevo menú para configurar esta IP.

Imagen 41: Asociar IP

En este menú se debe hacer click en el icono “+” del apartado dirección IP. Tras esto aparece otra pantalla nueva:

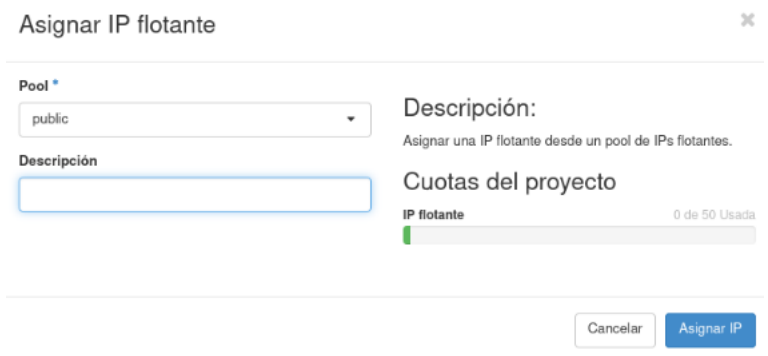


Imagen 42: Asignar IP

En esta pantalla se elige la red en la que estará ubicada la IP flotante, que para que la instancia tenga acceso a la red externa y por tanto a internet, debe ser la red “public”. Una vez elegida dicha red se selecciona “Asignar IP”.

Después de asignar la IP, surge un nuevo menú en el que nos muestra cual es la IP flotante en concreto.

Imagen 43: IP instancia

Basta con hacer click en “Asociar” y la IP estará configurada y asignada a la instancia correctamente.

En el apartado de topología de red se puede comprobar que la instancia se ha lanzado con éxito en la red que se pretendía:

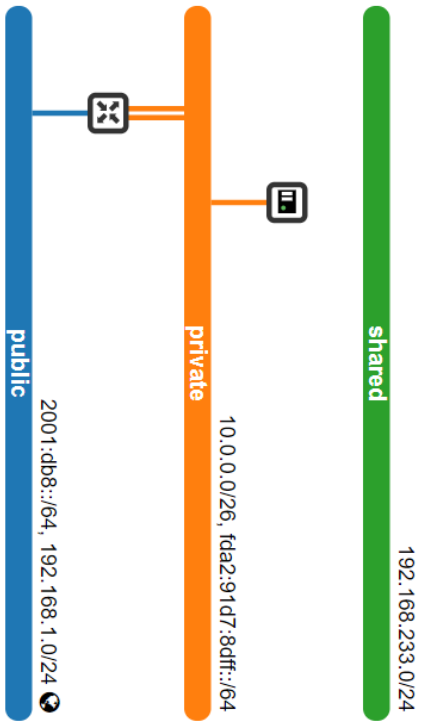


Imagen 44: Topología de red Instancia

ACCEDER A LA INSTANCIA

Una vez la instancia está lanzada, configurada correctamente y corriendo, hay dos maneras de acceder a ella, desde el dashboard, mediante su propia consola, o por ssh.

Acceder utilizando su propia consola es muy sencillo. En el mismo desplegable donde se escogía la acción "Asignar Ip flotante", ahora se selecciona "consola".

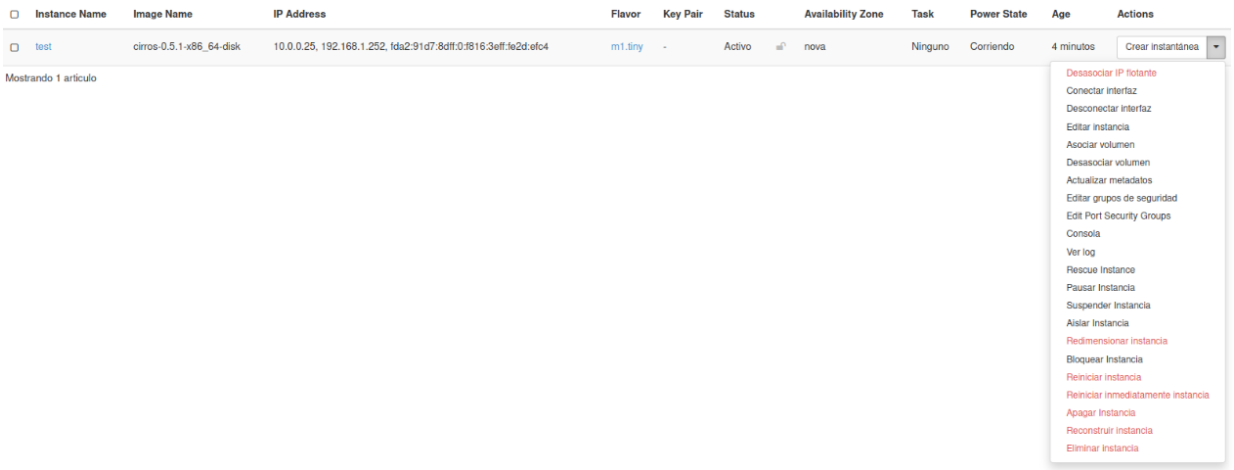


Imagen 45: Acceder a la consola de la instancia

Enseguida aparece la consola de la instancia.

Para acceder a la instancia hay que usar el usuario "cirros" y la contraseña "gocubsgo" como bien indica.

Una vez se ha accedido se puede comprobar que hay conexión a internet realizando un ping.

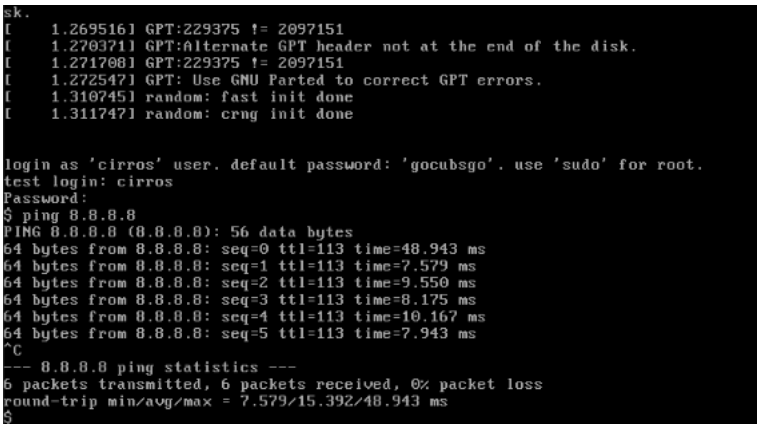


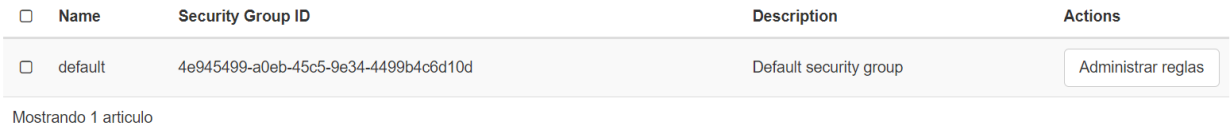
Imagen 46: Consola instancia

Del mismo modo, para conectarse a la estancia por ssh, lo primero que se debe hacer es configurar los grupos de seguridad a través del dashboard.

Esta configuración se hace en el apartado:

Proyecto->Red->Grupos de seguridad.

En el grupo de seguridad “default”, se selecciona administrar reglas.

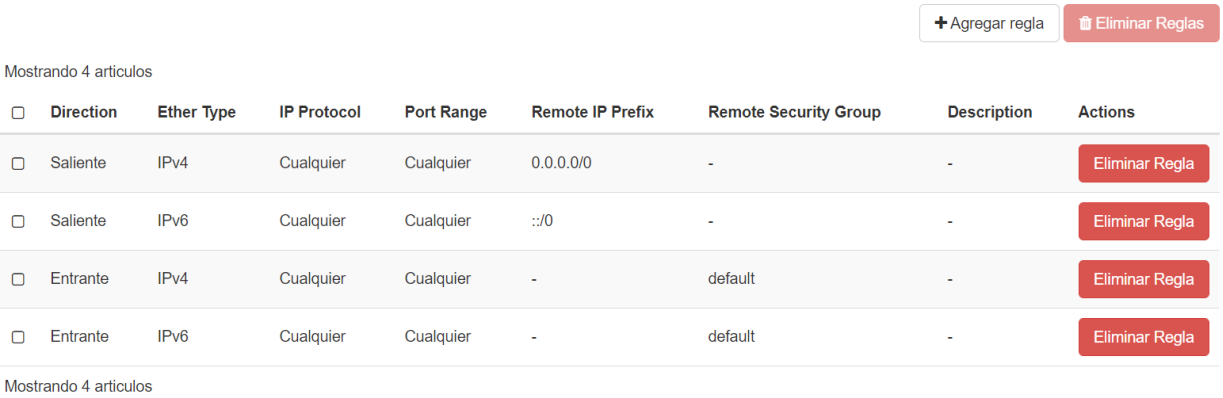


Name	Security Group ID	Description	Actions
default	4e945499-a0eb-45c5-9e34-4499b4c6d10d	Default security group	Administrar reglas

Mostrando 1 articulo

Imagen 47: Grupo de Seguridad default

Tras dicha selección, aparece una nueva pantalla con todas las reglas que vienen ya configuradas por defecto:



+Agregar regla | Eliminar Reglas

Mostrando 4 articulos

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
Saliente	IPv4	Cualquier	Cualquier	0.0.0.0/0	-	-	Eliminar Regla
Saliente	IPv6	Cualquier	Cualquier	::/0	-	-	Eliminar Regla
Entrante	IPv4	Cualquier	Cualquier	-	default	-	Eliminar Regla
Entrante	IPv6	Cualquier	Cualquier	-	default	-	Eliminar Regla

Mostrando 4 articulos

Imagen 48: Reglas de Seguridad Grupo default

Seleccionando “Agregar regla” se aparece un nuevo menú en el que se elige habilitar la regla ssh para para permitir el acceso remoto a la instancia.

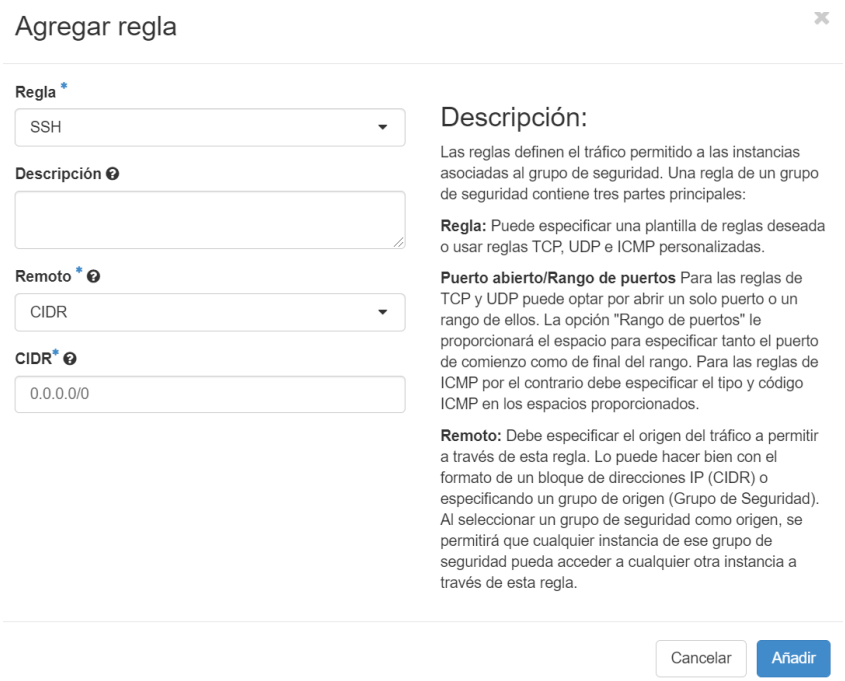


Imagen 49: Regla SSH

Tras haber configurado los grupos de seguridad habiendo habilitado el acceso por ssh a las instancias, se verifica que se ha configurado correctamente. Para ello, se abre un terminal desde el equipo propio (debe estar conectado a la misma red), y se escribe “ssh cirros@ipflotante”, en este ejemplo, “ssh cirros@192.168.1.252”.

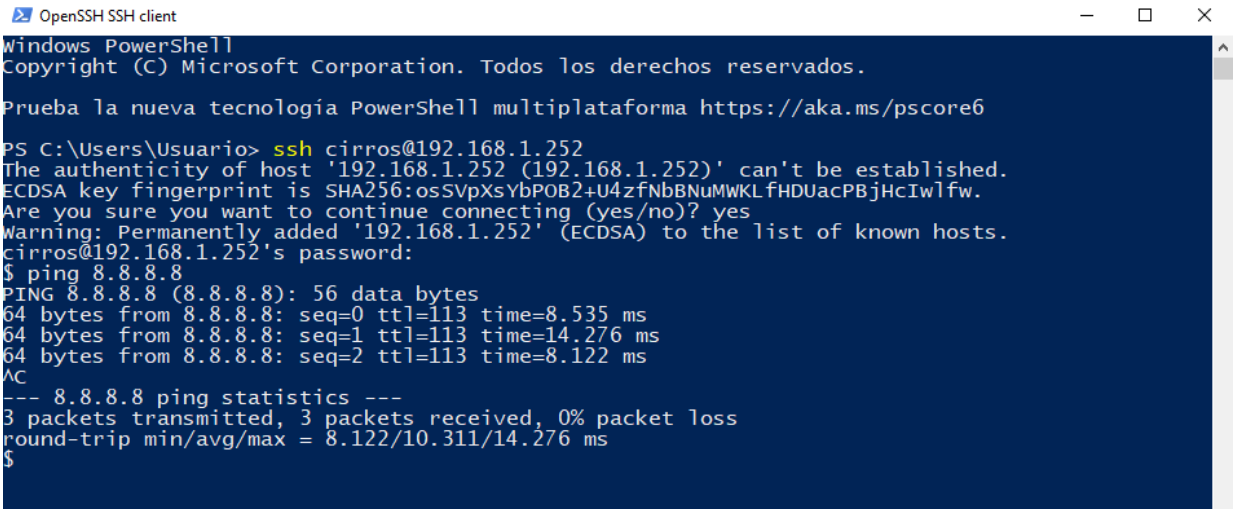


Imagen 50: Acceder a la instancia por SSH

ELIMINAR INSTANCIA

Otro punto importante a la hora de trabajar con instancias es eliminarlas.

Para eliminar una instancia una vez se termina de usar, lo primero que se debe hacer es apagarla. Esto se realiza desde el menú desplegable propio de cada instancia.

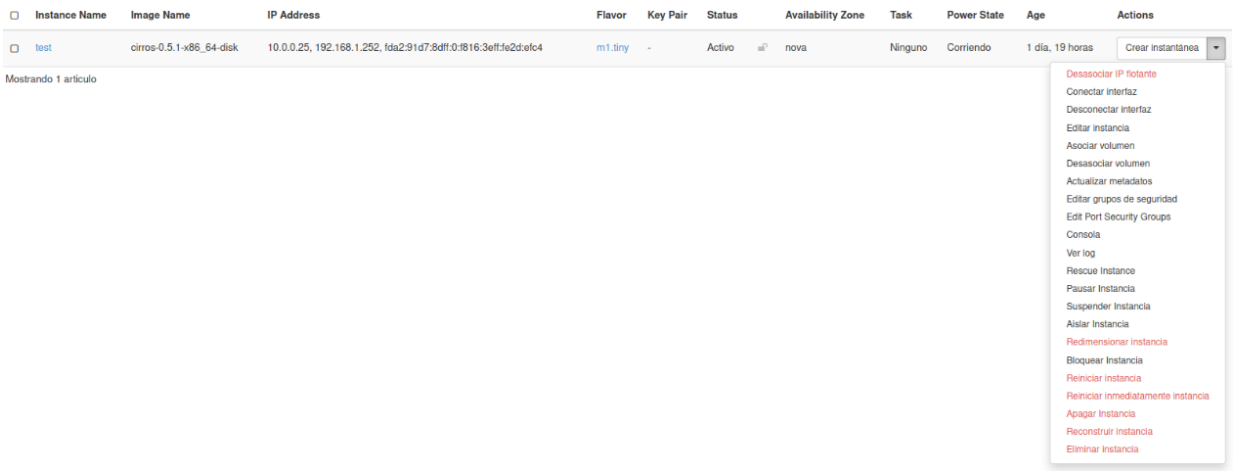


Imagen 51: Desasociar IP flotante

Una vez apagada la instancia, desde el mismo menú se debe seleccionar “desasociar ip flotante” para que la ip quede libre para futuras instancias.

Al seleccionar “desasociar ip flotante” aparece una nueva pantalla:

Es muy importante antes de desasociar marcar la casilla “Release Floating IP” para liberar la IP y no borrarla de la instancia únicamente.

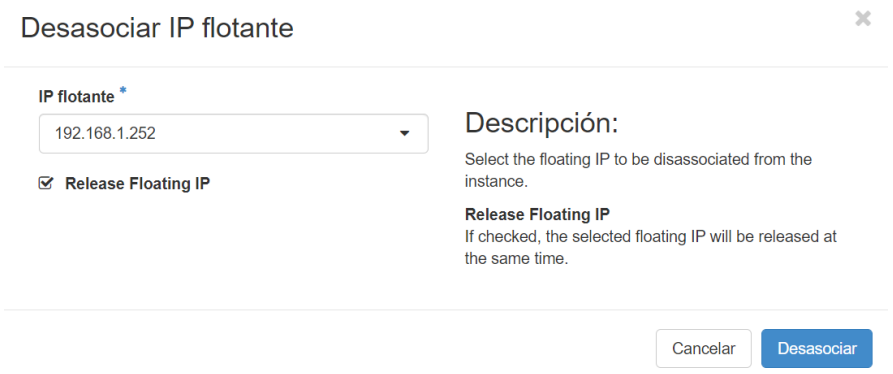


Imagen 52: Liberar IP flotante

Una vez la IP flotante ha sido desasociada de la instancia y liberada correctamente, bastará con acceder de nuevo al menú desplegable propio de cada instancia y elegir la opción “Eliminar instancia”.

3.5.4. Migrar una Instancia

Se puede cambiar el host en el que se encuentra implementada una instancia.

Esta operación es una labor de administrador, para poder llevarla a cabo es muy importante que la instancia esté corriendo.

Se puede realizar tanto desde el dashboard como desde el CLI.

Dashboard

En el menú desplegable administrador->Compute->Redes->instancias.

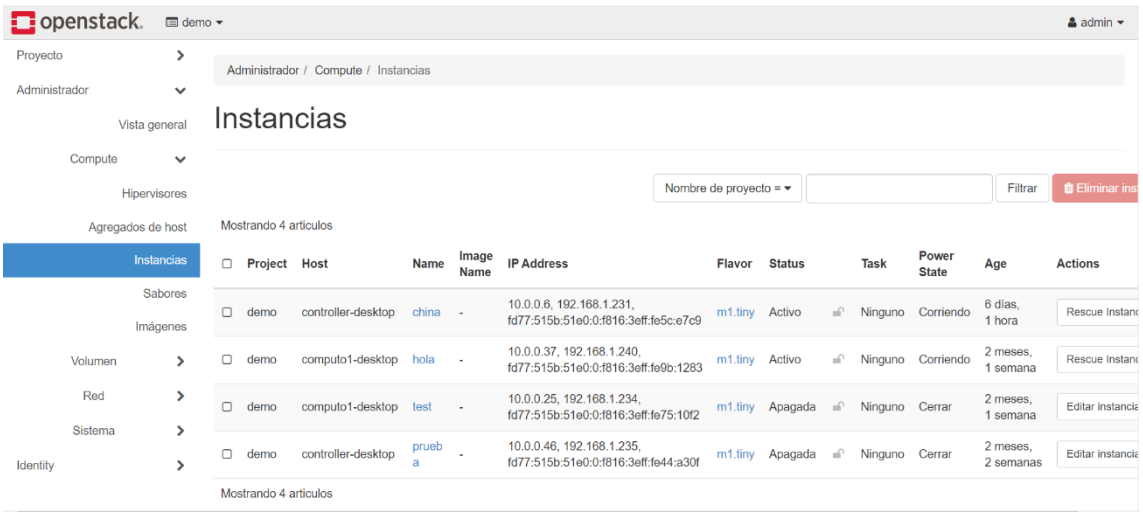


Imagen 53: Instancias

En la parte de la derecha de la instancia deseada, se selecciona la flecha para visualizar el menú desplegable.

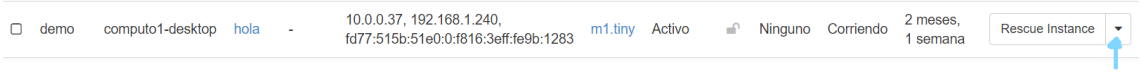


Imagen 54: Instancia de ejemplo

Y se elige la opción “Migrar instancia en vivo”.

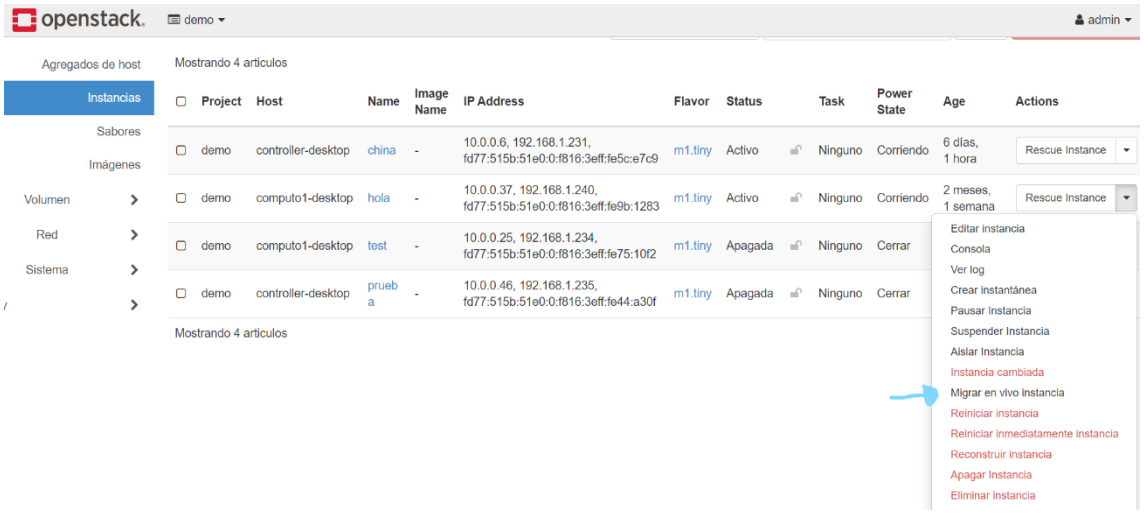


Imagen 55: Menú de la instancia

Tras seleccionar esta opción aparecerá la siguiente ventana en la que se elige el host en el que estará la nueva ubicación de la instancia.

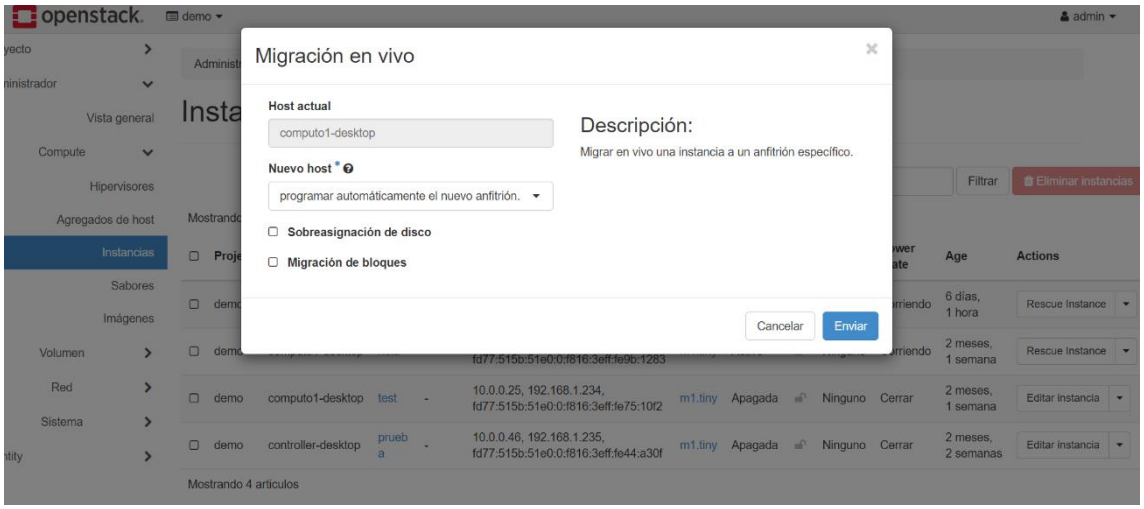


Imagen 56: Migración en vivo

CLI

Para migrar la instancia desde el CLI se usa el siguiente comando:

```
openstack server migrate
    [--live <hostname>]
    [--shared-migration | --block-migration]
    [--disk-overcommit | --no-disk-overcommit]
    [--wait]
    <server>
```

Imagen 57: Comando migrar instancia

Bastará con indicar el nombre del host destino (--live <hotname>) y el nombre de la instancia que se quiere migrar (<server>).

3.5.5. Scripts para crear y eliminar proyectos

Para facilitar las labores de administrador, se ha programado tanto un script para la creación automática de proyectos como un script para la eliminación automática de estos.

CREAR PROYECTOS

Lo primero que se debe hacer para poder ejecutar ambos scripts es cargar el fichero que proporciona autorización para utilizar el CLI de OpenStack. Cuando se ejecute cualquier script hace referencia a este archivo y pide que se introduzca la contraseña de administrador de la nube, que es la misma que se usa para iniciar sesión en el dashboard y que previamente hemos configurado.

En relación con el script de creación de proyectos, consiste en generar, a partir de un fichero con correos electrónicos, un usuario a cada correo.

Se va a explicar la estructura del script y el funcionamiento de los comandos que se utilizan.

Lo primero es crear un proyecto, cada usuario debe tener su propio proyecto, ya que de no ser así podrían acceder a los recursos del resto de usuarios. Para esto se hace uso del comando OpenStack:

```
openstack project create
    [--domain <domain>]
    [--parent <project>]
    [--description <description>]
    [--enable | --disable]
    [--property <key=value>]
    [--or-show]
    <name>
```

Imagen 58: Comando crear proyecto

En este entorno se crean todos los usuarios en el dominio default, por lo que es lo único que hay que indicar en el comando junto con el nombre del proyecto.

El motivo de que todos los proyectos se creen en el mismo dominio es porque en la nube los recursos se dividen en dominios (puede equivaler a una compañía) y dentro de los mismos en proyectos (lo que equivale, por ejemplo, a los distintos departamentos dentro de la compañía).

El siguiente paso es configurar el proyecto, se añaden a todos los proyectos los usuarios demo y admin, a ambos usuarios se les asigna el rol de administrador para que tengan acceso a todos los proyectos, ya que ambos tienen ese papel sobre todo el entorno y además el usuario demo ofrece la topología de red deseada.

El comando que se utiliza en este caso es el siguiente:

```
openstack role add
    --domain <domain> | --project <project> [--project-domain <project-domain>]
    --user <user> [--user-domain <user-domain>] | --group <group> [--group-domain <group-domain>]
    --role-domain <role-domain>
    --inherited
    <role>
```

Imagen 59: Comando añadir rol

Tras haber añadido los usuarios con rol de administrador al proyecto, se crea el usuario correspondiente, propio de cada proyecto con el comando:

```
openstack user create
    [--domain <domain>]
    [--project <project> [--project-domain <project-domain>]]
    [--password <password>]
    [--password-prompt]
    [--email <email-address>]
    [--description <description>]
    [--enable | --disable]
    [--or-show]
    <user-name>
```

Imagen 60: Comando crear usuario

Con este comando se crea el usuario, indicando el proyecto al que pertenece, su correo electrónico y configurando la contraseña con la que tendrá acceso al dashboard.

La contraseña para cada usuario será nombre_usuario + año + mes, entendiendo que el año y el mes corresponden con la creación del usuario. Un ejemplo de esto sería, si el correo electrónico proporcionado en el fichero es mario.gomeze@uah.es, el nombre de usuario será “mario.gomeze” y la contraseña “mario.gomeze202103” (si este usuario hubiera sido creado en marzo de 2021).

La contraseña es muy sencilla de cambiar posteriormente a través del dashboard.

Usuario->Ajustes->Cambiar contraseña.

Primero se accede al menú de ajustes de cada usuario. Este se encuentra en la parte superior derecha, en el dashboard.

En este ejemplo el usuario es admin.

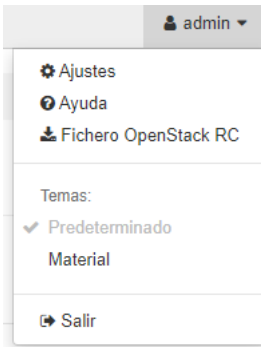


Imagen 61: Ajustes usuario

Tras acceder a los ajustes, se selecciona cambiar contraseña.

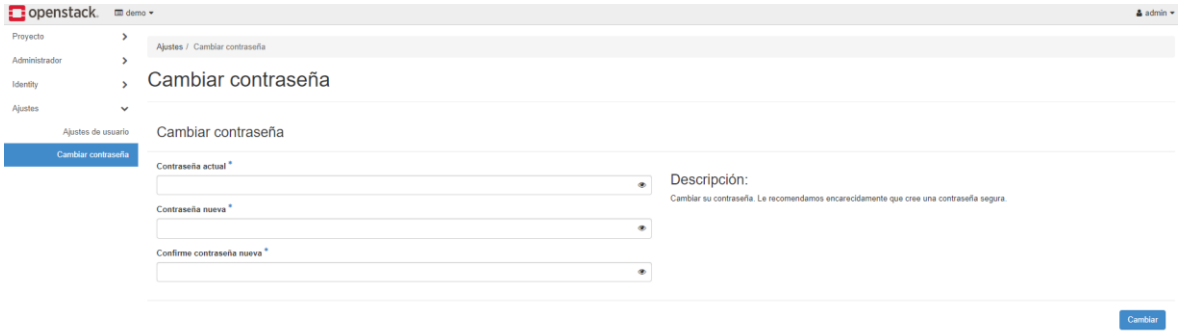


Imagen 62: Cambiar contraseña

Después de crear el usuario se le debe asignar el rol member con el comando explicado anteriormente.

Tras tener el proyecto ya creado y el usuario creado y configurado, lo siguiente es la configuración del proyecto.

En cada proyecto se genera una red distinta:

```
openstack network create
    [--project <project> [--project-domain <project-domain>]]
    [--enable | --disable]
    [--share | --no-share]
    [--description <description>]
    [--availability-zone-hint <availability-zone>]
    [--enable-port-security | --disable-port-security]
    [--external [--default | --no-default] | --internal]
    [--provider-network-type <provider-network-type>]
    [--provider-physical-network <provider-physical-network>]
    [--provider-segment <provider-segment>]
    [--qos-policy <qos-policy>]
    [--transparent-vlan | --no-transparent-vlan]
    [--tag <tag> | --no-tag]
    <name>
```

Imagen 63: Comando crear red

En el comando se debe indicar el proyecto en el que se crea, que la red no es compartida, que es una red interna y el nombre de la red.

Además, se crea una subred, propia de cada proyecto y con un rango diferente en cada proyecto:

```
openstack subnet create
    [--project <project> [--project-domain <project-domain>]]
    [--subnet-pool <subnet-pool> | --use-default-subnet-pool [--prefix-length <prefix-length>]]
    [--subnet-range <subnet-range>]
    [--allocation-pool start=<ip-address>,end=<ip-address>]
    [--dhcp | --no-dhcp]
    [--dns-nameserver <dns-nameserver>]
    [--gateway <gateway>]
    [--host-route destination=<subnet>,gateway=<ip-address>]
    [--ip-version {4,6}]
    [--description <description>]
    [--ipv6-ra-mode {dhcpv6-stateful,dhcpv6-stateless,slaac}]
    [--ipv6-address-mode {dhcpv6-stateful,dhcpv6-stateless,slaac}]
    [--network-segment <network-segment>]
    [--service-type <service-type>]
    [--tag <tag> | --no-tag]
    --network <network>
    <name>
```

Imagen 64: Comando crear subred

Con este comando se impone la red dentro de la cual se crea esta subred, se le añade el dns 8.8.8.8, el proyecto en el que se crea y el rango y el nombre que tiene.

Por último, se crea un router para cada proyecto, indicando este y el propio nombre del router.

```
openstack router create
    [--project <project> [--project-domain <project-domain>]]
    [--enable | --disable]
    [--distributed | --centralized]
    [--ha | --no-ha]
    [--description <description>]
    [--availability-zone-hint <availability-zone>]
    [--tag <tag> | --no-tag]
    <name>
```

Imagen 65: Comando crear router

Este router debe conectar la red externa con la red privada de cada proyecto de tal modo que al lanzar una instancia en la red privada, estas tengan acceso a internet y se pueda acceder a ellas por ssh.

Para realizar esta conexión se configura en el router el Gateway a la red externa:

```
openstack router set
    [--name <name>]
    [--enable | --disable]
    [--distributed | --centralized]
    [--description <description>]
    [--route destination=<subnet>,gateway=<ip-address> | --no-route]
    [--ha | --no-ha]
    [--external-gateway <network> [--enable-snat|--disable-snat] [--fixed-ip subnet=<subnet>,ip-address=<ip-address>]]
    [--tag <tag>] [--no-tag]
    <router>
```

Imagen 66: Comando configurar router

Y se añade también al router la subred creada antes, propia de cada proyecto con el comando:

```
openstack router add subnet
    <router>
    <subnet>
```

Imagen 67: comando añadir subred al router

Todo este proceso será repetido con cada correo que aparezca en el fichero hasta que el script detecte el fin del fichero.

Para ejecutar este script:

```
./crearusuarios.sh
```

ELIMINAR PROYECTOS

Antes de poder eliminar los respectivos usuarios y sus correspondientes proyectos, se deben eliminar las instancias y liberar las direcciones IP flotantes para poder eliminar las redes y que no quede bloqueada ninguna IP sin que esté en uso.

Para eliminar las instancias, primero se muestran para que la salida se imprima en un fichero, mediante el comando:

```
openstack server list
    [--quote {all,minimal,none,nonnumeric}]
    [--reservation-id <reservation-id>]
    [--ip <ip-address-regex>]
    [--ip6 <ip-address-regex>]
    [--name <name-regex>]
    [--instance-name <server-name>]
    [--status <status>]
    [--flavor <flavor>]
    [--image <image>]
    [--host <hostname>]
    [--all-projects]
    [--project <project>]
    [--project-domain <project-domain>]
    [--user <user>]
    [--user-domain <user-domain>]
    [--long]
    [-n]
    [--marker <server>]
    [--limit <num-servers>]
    [--deleted]
    [--changes-since <changes-since>]
```

Imagen 68: Comando listar instancias

Se utiliza la opción "—all-projects" para mostrar las instancias de todos los proyectos.

Además con el uso de "grep" se consigue filtrar para que en el fichero sólo se imprima el ID de cada instancia.

Una vez se tienen recogidos en un fichero, únicamente los IDs de las instancias, eliminarlas es muy simple.

```
openstack server delete [--wait] <server> [<server> ...]
```

Imagen 69: Comando eliminar instancia

Para este comando solo es necesario el ID.

Del mismo modo, se actúa con las direcciones IP flotantes, usando “grep” y el siguiente comando, se consigue recoger en un fichero todas las direcciones.

```
openstack ip floating list
```

Imagen 70: Comando listar IP flotantes

Hay que usar “floating ip” en vez de “ip floating”, ya que este último está en desuso.

Después de haber recogido todas las IP en el fichero, a partir de este, se eliminan.

```
openstack ip floating delete
    <floating-ip> [<floating-ip> ...]
```

Imagen 71: Comando eliminar IP flotante

En este comando igual, hay que usar “floating ip” en vez de “ip floating”, ya que este último está en desuso.

Después de haber eliminado todas las instancias y direcciones IP flotantes, se procede a eliminar los usuarios y proyectos, a partir de un fichero de correos electrónicos, del mismo modo que en el script para crear usuarios.

Lo primero es eliminar la red privada que fue creada para cada usuario. Para ello se elimina la puerta de enlace de esta red al router con el comando:

```
openstack router remove subnet
    <router>
    <subnet>
```

Imagen 72: Comando eliminar subred del router

Lo siguiente que hay que eliminar es la subred propia de la red.

```
openstack subnet delete
    <subnet> [<subnet> ...]
```

Imagen 73: Comando eliminar subred

Y por último cuando ya esté eliminada la puerta de enlace y la subred, se permite eliminar la red.

```
openstack network delete
    <network> [<network> ...]
```

Imagen 74: Comando eliminar red

Cuando se ha eliminado la red al completo es posible eliminar el router mediante el uso del siguiente comando:

```
openstack router delete
    <router> [<router> ...]
```

Imagen 75: Comando eliminar router

En este comando lo único que se indica es el nombre de cada router.

Tras haber eliminado todos los elementos del proyecto se puede proceder a eliminar el usuario.

```
openstack user delete
    [--domain <domain>]
    <user> [<user> ...]
```

Imagen 76: Comando eliminar usuario

Y por último, para concluir con la eliminación de usuarios, se elimina el proyecto con el comando:

```
openstack project delete
    [--domain <domain>]
    <project> [<project> ...]
```

Imagen 77: Comando eliminar proyecto

En el script, todo este proceso se repite con cada uno de los correos del fichero hasta que se detecta fin de fichero.

Para ejecutar este script:

```
./eliminarusuarios.sh
```

3.5.6. Usos avanzados OpenStack

Además de los componentes básicos que se instalan de forma predeterminada con DevStack, se han instalado componentes adicionales desde el archivo local.conf para profundizar en el uso del entorno OpenStack.

Como se vió anteriormente OpenStack ofrece una gran variedad de componentes. Los elegidos para este proyecto han sido:

- Monitorización: Ceilometer y Monasca.
- Facturación y Lógica empresarial: CloudKitty.
- Orquestación: Aodh.

CEILOMETER

Este componente se instala añadiendo en el archivo local.conf del nodo controlador lo siguiente:

```
# Enable the Ceilometer devstack plugin
enable_plugin ceilometer https://git.openstack.org/openstack/ceilometer.git
CEILOMETER_BACKEND=gnocchi
```

Imagen 78: Ceilometer local.conf

Ceilometer [34] es un buen componente adicional que permite crear distintos medidores y recoger una gran variedad de estadísticas, lo cual proporciona una cantidad de información que resulta muy eficiente a la hora de monitorizar el entorno.

Ceilometer recopila, normaliza y transforma los datos producidos por los servicios OpenStack de manera eficiente. Estos datos se utilizan para crear diferentes vistas y ayudar a resolver varios casos de telemetría.

Este servicio se amplía con Aodh, que es un servicio para definir alarmas, adicional para la monitorización y Gnocchi, que es una base de datos de recursos y métricas.

El servicio ceilometer combinado con Gnocchi es muy utilizado para obtener información adicional, de imágenes, instancias… Pero este servicio tiene una mayor utilidad si se usa para monitorizar.

La telemetría mide los recursos de la nube OpenStack. Para modelar datos usa las siguientes abstracciones:

- Medidor: mide un aspecto específico del uso de recursos, como la existencia de una instancia en ejecución o el rendimiento continuo, el uso de CPU para una instancia. Existen medidores para cada tipo de recurso.

- Muestra: un punto de datos individual que está asociado con un medidor específico.
- Estadística: consiste en la recopilación de datos durante un periodo de tiempo.
- Alarma: notificación para cuando no se cumplan un conjunto de reglas.

Un buen ejemplo del funcionamiento de este servicio sería crear en el dashboard una imagen de Ubuntu server 16.04 y lanzar una instancia con esta imagen, una vez lanzada la instancia se le instala apache y se trata de estresar el servidor. Esto produce que se aumenten el número de peticiones por segundo, provocando así un aumento en el uso de la CPU por parte del servidor. El objetivo de esta prueba es supervisar el uso de la CPU por parte de la instancia.

Para ello hay que ejecutar el siguiente comando antes de estresar el servidor y después de estresar el servidor, para ver como al estresar el servidor aumenta el uso de la CPU.

```
$ ceilometer sample-list -m cpu_util
+--------------------------------------+----------+-------+---------------+------+---------------------+
| Resource ID                          | Name     | Type  | Volume        | Unit | Timestamp           |
+--------------------------------------+----------+-------+---------------+------+---------------------+
| 3965b41b-81b0-4386-bea5-6ec37c8841c1 | cpu_util | gauge | 3.98333333333 | %    | 2013-10-02T10:50:12 |
+--------------------------------------+----------+-------+---------------+------+---------------------+
```

Imagen 79: Comando ceilometer monitorizar uso CPU

Esta prueba no se ha podido verificar, ya que no se ha podido acceder al servidor Ubuntu-16 después de crearlo para estresarlo.

CLOUDKITTY

Este componente se instala añadiendo en el archivo local.conf del nodo controlador lo siguiente:

```
# cloudkitty
enable_plugin cloudkitty https://opendev.org/openstack/cloudkitty master
enable_service ck-api, ck-proc
CLOUDKITTY_COLLECTOR=gnocchi
```

Imagen 80: Cloudkitty local.conf

CloudKitty es un proyecto de servicio de tarificación, diseñado para traducir métricas en precios. Admite múltiples recopiladores, múltiples políticas de tarificación y múltiples salidas.

Un ejemplo de su uso es el siguiente:

Se inicia sesión en el dashboard, y en el apartado administrador->Rating->Rating modules se habilita el módulo Hashmap.

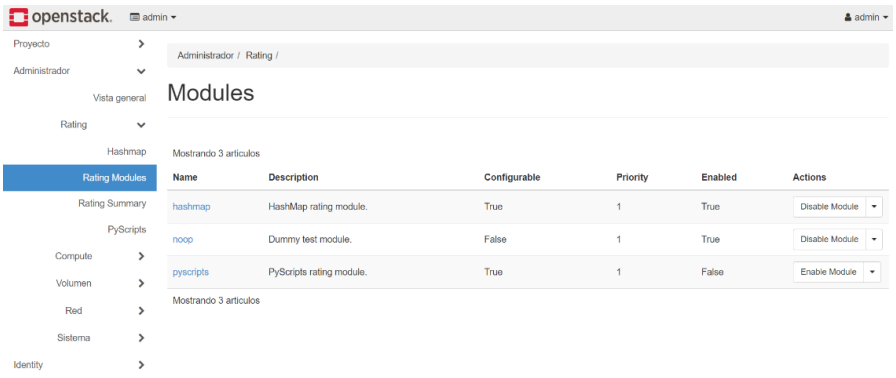


Imagen 81: Cloudkitty rating modules

Una vez habilitado este módulo, se accede a su panel de configuración: Administrador>Rating>Hashmap, y se crea un servicio llamado instancias.

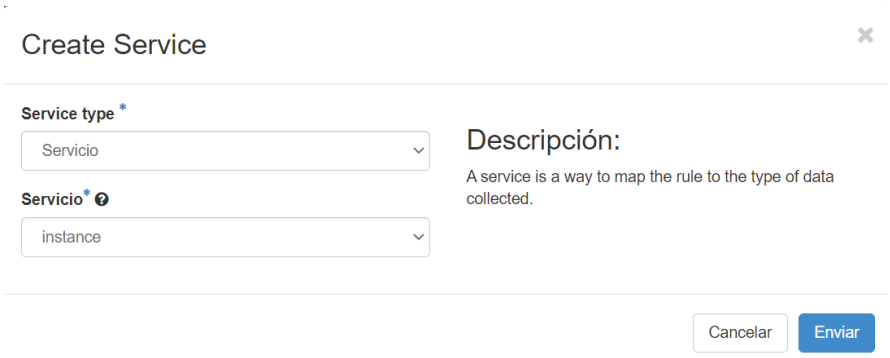


Imagen 82: Cloudkitty crear servicio

Después de crearse, se accede a él.

Dentro del servicio instancias se crea un campo que se llame flavor_id. Este campo se va a utilizar para hacer coincidir los sabores (flavor) de las instancias en ejecución.

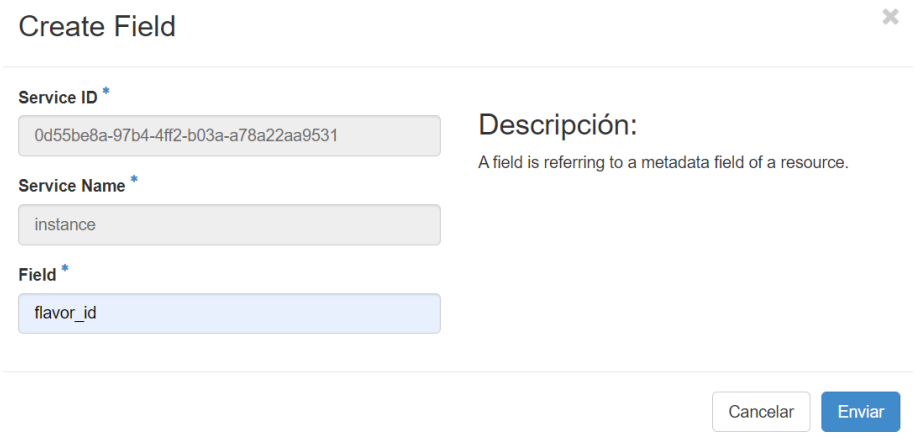


Imagen 83: Cloudkitty crear campo

Cuando se crea el campo, se accede a él haciendo click en su nombre.

Para cargar las instancias en ejecución en función de su sabor, necesitamos obtener los ID de los sabores que queremos usar.

```
stack@controller-desktop:~/devstack$ openstack flavor list
+----+-----------+-------+------+-----------+-------+-----------+
| ID | Name      |   RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+-----------+-------+------+-----------+-------+-----------+
| 1  | m1.tiny   |   512 |    1 |         0 |     1 | True      |
| 2  | m1.small  |  2048 |   20 |         0 |     1 | True      |
| 3  | m1.medium |  4096 |   40 |         0 |     2 | True      |
| 4  | m1.large  |  8192 |   80 |         0 |     4 | True      |
| 42 | m1.nano   |   128 |    1 |         0 |     1 | True      |
| 5  | m1.xlarge | 16384 |  160 |         0 |     8 | True      |
| 84 | m1.micro  |   192 |    1 |         0 |     1 | True      |
| c1 | cirros256 |   256 |    1 |         0 |     1 | True      |
| d1 | ds512M    |   512 |    5 |         0 |     1 | True      |
| d2 | ds1G      |  1024 |   10 |         0 |     1 | True      |
| d3 | ds2G      |  2048 |   10 |         0 |     2 | True      |
| d4 | ds4G      |  4096 |   20 |         0 |     4 | True      |
+----+-----------+-------+------+-----------+-------+-----------+
```

Imagen 84: Comando listar sabores

Se crea una relación que coincida con el sabor elegido, por ejemplo, m1.tiny.

Se especifica el coste en dólares por instancia y por periodo de recolección (3600 segundos).

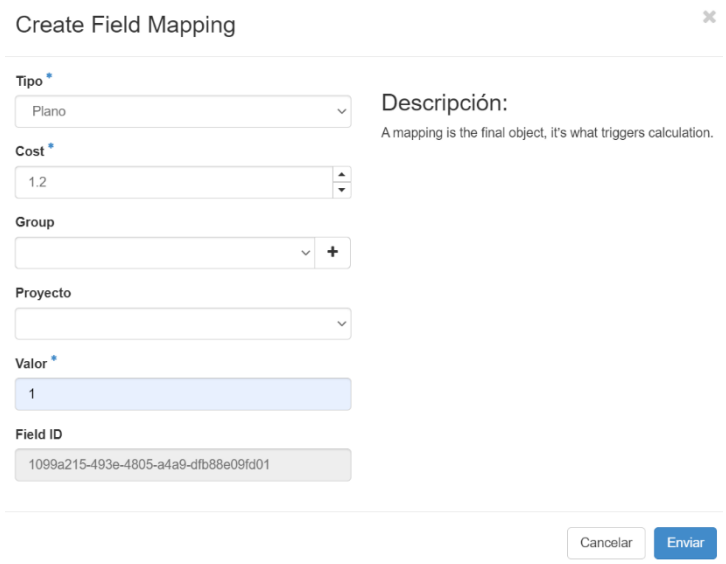


Imagen 85: Cloudkitty configurar coste instancia

Una vez se ha ajustado el coste relacionado con el sabor, se puede ajustar también en función de la imagen utilizada.

Para ello, lo primero es listar el ID de las imágenes existentes.

```
stack@controller-desktop:~/devstack$ openstack image list
+--------------------------------------+--------------------------+--------+
| ID                                   | Name                     | Status |
+--------------------------------------+--------------------------+--------+
| f64390a2-ef9e-4fc2-a5d5-c736a1f982e6 | cirros-0.5.1-x86_64-disk | active |
| 3bfc3f87-0928-4011-98f8-e32b62781212 | ubuntu-18                | active |
+--------------------------------------+--------------------------+--------+
```

Imagen 86: Comando listar imágenes

Una vez obtenido el ID de la imagen elegida, se ha de volver a la sección de campos del servicio de instancias y se crea un nuevo campo llamado image_id.

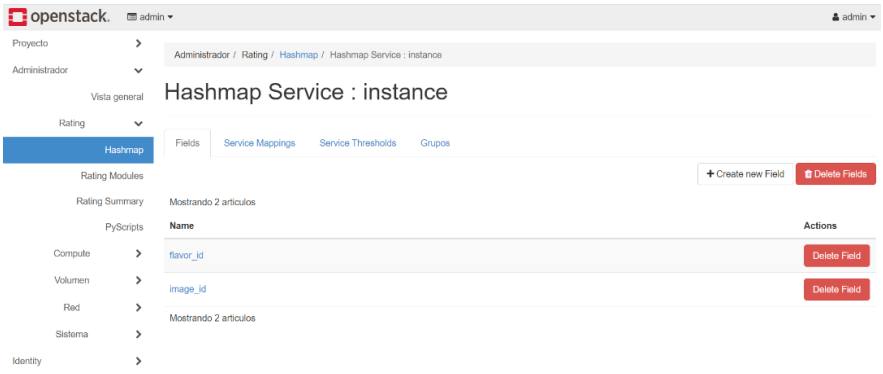


Imagen 87: Cloudkitty hashmap service

Tras crearlo se accede a él y se crea una asignación plana basada en image_id de la misma manera que se creó para flavor_id, especificando el nuevo coste.

Primero se crea el mapeo para la imagen cirros, especificando su ID.

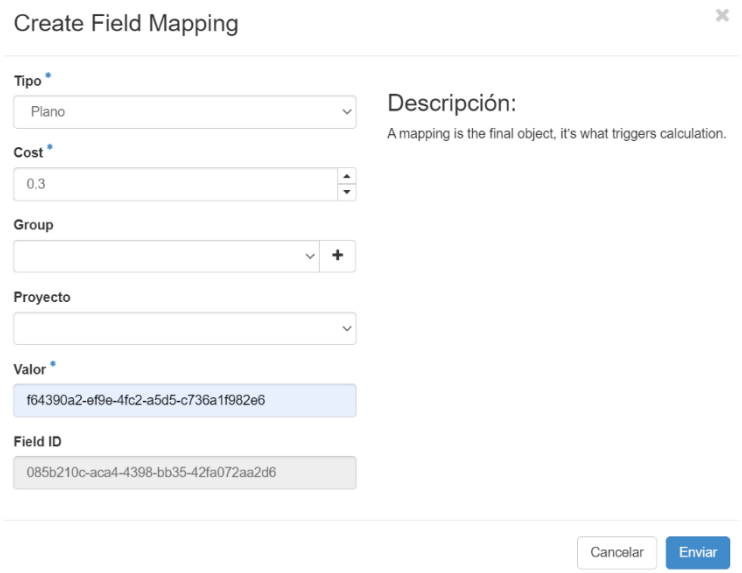


Imagen 88: Cloudkitty configurar coste imagen

Ahora se crea un servicio de imágenes como se hizo con el servicio de instancias.

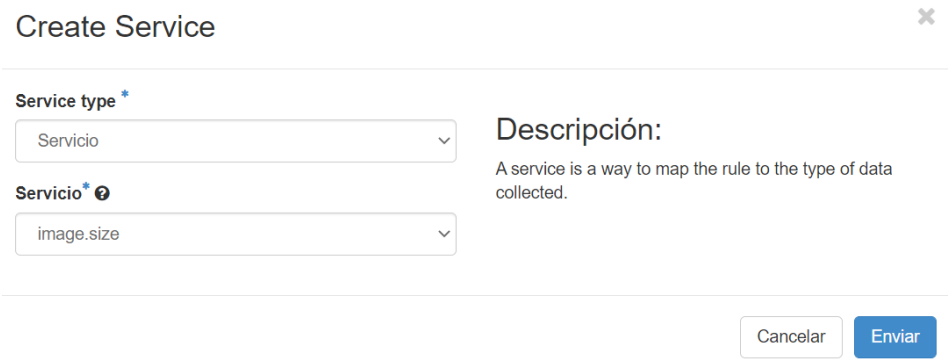


Imagen 89: Cloudkitty image.size service

Y se crea una relación de servicios. El coste será por MiB por periodo de recolección (3600s).

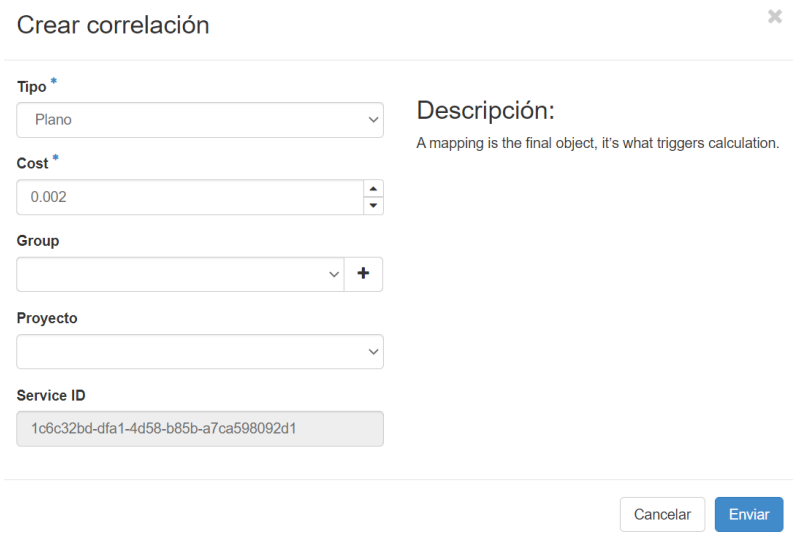


Imagen 90: Cloudkitty crear correlación

Si ahora se accede al apartado Project->Compute-> instances y se lanza una instancia, después de configurar el lanzamiento de esta, en el apartado Price, aparecerá el coste del lanzamiento de la instancia.

Imagen 91: Cloudkitty lanzar instancia

AODH

Este componente se instala añadiendo en el archivo local.conf del nodo controlador lo siguiente:

```
# aodh
enable_plugin aodh https://git.openstack.org/openstack/aodh
```

Imagen 92: Aodh local.conf

Aodh sirve para configurar alarmas [32] basadas en umbrales, eventos o compuestas.

Es un servicio, que como bien se ha comentado anteriormente extiende el servicio de ceilometer por lo que es un componente adicional para la monitorización.

Las alarmas brindan un monitoreo como servicio orientado al usuario para los recursos que se ejecutan en OpenStack. Este tipo de supervisión garantiza que se pueda escalar automáticamente dentro o fuera de un grupo de instancias a través del servicio de Orquestación, pero también se pueden usar alarmas para el conocimiento general del estado de sus recursos en la nube.

Para verificar su uso, se toma el mismo ejemplo que se utilizó para ceilometer. Se crea una alarma basada en un umbral, de tal modo que la alarma se activará cuando el uso de la CPU en una instancia particular exceda un porcentaje, en este caso, el 70.0 %.

Para superar este umbral y que se active la alarma, habría que estresar el servidor.

La alarma se crea con el siguiente comando:

```
$ aodh alarm create \
  --name cpu_hi \
  --type gnocchi_resources_threshold \
  --description 'instance running hot' \
  --metric cpu_util \
  --threshold 70.0 \
  --comparison-operator gt \
  --aggregation-method mean \
  --granularity 600 \
  --evaluation-periods 3 \
  --alarm-action 'log://' \
  --resource-id INSTANCE_ID \
  --resource-type instance
```

Imagen 93: Aodh comando crear alarma

Esta prueba no se ha podido verificar, ya que no se ha podido acceder al servidor Ubuntu-16 después de crearlo para estresarlo.

4. Nube Híbrida

La nube Híbrida consiste en la combinación de varios clouds, públicos y privados, consiguiendo así obtener las ventajas que proporcionan ambos tipos de infraestructuras.

Los clouds que forman una nube híbrida, públicos y privados, son entidades únicas e individuales.

Para transferir recursos y cargas de trabajo se utiliza un interfaz de programación de aplicaciones. Su objetivo es mantener en el cloud privado las cargas de trabajo más críticas y aprovechar el potencial de la nube pública para aumentar los recursos y optimizarlos.

Imagen 94: Nube híbrida

Una arquitectura de nube híbrida presenta las siguientes características:

- Integración: la arquitectura de nube híbrida presenta la necesidad de que exista una perfecta combinación entre la infraestructura de nube privada ya existente y los recursos que proporciona una nube pública.

- Escalabilidad: cuando sean necesarios más recursos, la nube híbrida debe ser capaz de extender estos servicios respondiendo a las necesidades, de forma inmediata y automática.

- Personalización: la nube híbrida permite adoptar una estrategia cloud, es decir, los recursos y servicios que se ofrecen, en función de las necesidades particulares que necesita el usuario en cada momento.

- Seguridad: la nube híbrida debe ofrecer diferentes niveles de seguridad, entre ellos la gestión de datos críticos.

- Orquestación y automatización: todos los elementos de la nube híbrida deben funcionar en armonía unos con otros y estar perfectamente sincronizados.

Para conseguir estas características en la nube híbrida el proyecto OpenStack está consolidado como la mejor plataforma de desarrollo cloud de código abierto.

En este proyecto se ha levantado un entorno de nube privada OpenStack con DevStack. El objetivo sería integrar el cloud privado con un cloud público y poder controlar las dos nubes desde una única plataforma.

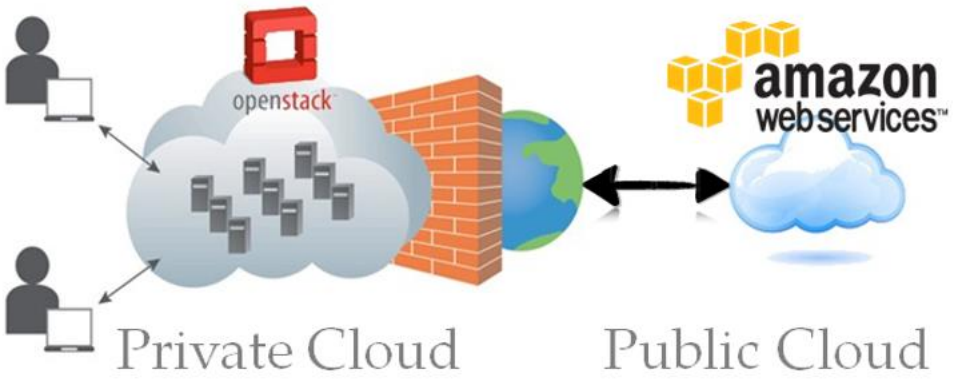


Imagen 95: OpenStack y AWS

4.5. OpenStack-Omni

Actualmente, la alternativa de código abierto a VMware y AWS para la gestión de la nube híbrida es OpenStack-Omni [31].

OpenStack-Omni es un proyecto OpenStack Big Tent fundado por Platform9 que permite una amplia aplicación de las API de OpenStack para gestionar nubes públicas de otros proyectos externos a OpenStack.

El conjunto de controladores de Omni se conecta a varios componentes de OpenStack, como Neutron, Nova y Cinder, y permite a los usuarios administrar la nube pública además de la infraestructura de nube local, es decir, la nube privada, desde la misma plataforma.

Omni permite el lanzamiento de instancias de AWS y la visualización de estas mediante OpenStack Horizon; y creación e integración de VM nativas basadas en AWS, Volúmenes redes, subredes, routers, etc., mediante el uso de comandos estándar OpenStack CLI o REST.

Omni es compatible con los tres principales proveedores de nube pública: AWS, Azure y Google Cloud.

Este apartado se ha a descrito de forma teórica y se tendrá en cuenta como trabajo futuro debido a que la instalación de Omni sería posible de forma manual o con juju y no con DevStack.

5. Conclusiones

Este proyecto ha sido una buena manera de introducirse en el mundo del cloud computing, investigando su historia y evolución, comprendiendo tanto las características como las ventajas y desventajas que este ofrece y estudiando los tipos de cloud que existen y se pueden desarrollar, así como, los servicios que ofrece cada uno de ellos a sus usuarios.

Se ha desplegado con éxito un cloud privado OpenStack, utilizando DevStack como método para levantar el entorno. Una vez desplegado se ha recogido el procedimiento para el lanzamiento de instancias, así como, el acceso a estas para trabajar con ellas.

Además, se ha conseguido elaborar una serie de scripts para automatizar el entorno en tareas como la creación y eliminación de usuarios, y se han documentado usos avanzados de OpenStack, como la instalación y utilización de componentes adicionales que ofrecen diferentes servicios como monitorización, tarificación…; y la migración de instancias entre distintos nodos, en el caso de que haya que prescindir de los servicios de un nodo en concreto por cualquier motivo.

Por último, no se ha conseguido integrar AWS en el entorno OpenStack para dar lugar a la nube híbrida, pero, se ha investigado como desarrollarlo para futuras implementaciones con OpenStack-Omni.

5.5. Trabajo futuro

Los conocimientos adquiridos y el entorno desplegado en este proyecto pueden servir como base o raíz para desarrollar un entorno más completo y de uso más avanzado o profesional.

Con la inclusión de diferentes nodos adicionales se puede aumentar la cantidad de recursos disponibles en el cloud. Lo que proporcionaría un mayor abastecimiento y, por lo tanto, la posibilidad de utilización por un mayor número de usuarios.

En cuanto a lo que OpenStack se refiere, es posible aumentar los servicios ofrecidos por el entorno. OpenStack como se ha visto, pese a ser de código abierto, cuenta con una gran variedad de componentes adicionales que con su integración en el cloud proporcionarían un uso más avanzado del entorno.

Por último, OpenStack-Omni es un proyecto desarrollado por Platform9 que con su integración daría la posibilidad de desarrollar un entorno de cloud híbrido.

6. Presupuesto

En este apartado se va a determinar el presupuesto necesario para la realización del proyecto.

Para ello el proyecto se divide en dos bloques, Hardware y Software. En este último se llevará a cabo una cuantificación del esfuerzo.

Los valores utilizados son estimaciones lo más aproximadas posibles.

5.1. Hardware

Para la creación del cluster se han utilizado tres equipos prestados por la universidad y un equipo propio con las siguientes características:

EQUIPOS	**PRECIO**
EQUIPO PERSONAL: i7 8ª Gen 4 cores, 16Gb RAM, 512Gb SSD.	800€
3 EQUIPOS PRESTADOS POR LA ESCUELA: i5, 16Gb RAM, 512Gb HDD.	3x500€

TOTAL: 2300€

5.2. Software

En este apartado, la cuantificación del esfuerzo se ha dividido en varias fases:

- **Instalación Ubuntu Desktop 20.04.**
- **Estudio, implementación y utilización OpenStack:** en esta segunda parte el tiempo fue invertido en el estudio de los métodos de instalación microstack y devstack, la implementación de este último y la familiarización con el entorno, tanto para la utilización del dashboard como la del OpenStack CLI.
- **Automatización de labores de administrador:** esta fase consistió en el estudio y desarrollo de scripts en bash para la creación y eliminación automática de usuarios a partir de un fichero con correos electrónicos.

- **Estudio, implementación y utilización de servicios adicionales OpenStack:** el desarrollo de esta fase concluyo en la instalación y pruebas de distintos servicios de uso avanzado del entorno desplegado.
- **Estudio OpenStack-Omni:** esta última fase estuvo dedicada al estudio de convertir el entorno de nube privada en un entorno de nube híbrida, utilizando como nube pública AWS.
- **Elaboración de la memoria.**

5.2.1. Cuantificación del esfuerzo

Se realiza una estimación aproximada en horas para cuantificar el tiempo empleado en las fases descritas en el apartado anterior.

FASE	**HORAS**
INSTALACIÓN UBUNTU DESKTOP 20.04	6
ESTUDIO, IMPLEMENTACIÓN Y UTILIZACIÓN OPENSTACK	200
AUTOMATIZACIÓN LABORES DE ADMINISTRADOR	20
ESTUDIO, IMPLEMENTACIÓN Y UTILIZACIÓN SERVICIOS ADICIONALES OPENSTACK	150
ESTUDIO OPENSTACK OMNI	10
ELABORACIÓN MEMORIA	50

TOTAL: 436 HORAS.

7. Bibliografía

[1] Historia del cloud computing. Enatec. Available: https://einatec.com/historia-cloud-computing/

[2] Historia y evolución del cloud computing. Arsys blog. (2020). Available: https://www.arsys.es/blog/historia-cloud/

[3] Cloud computing: características y funcionamiento. Ealde business school. (2019). Available: https://www.ealde.es/cloud-computing-caracteristicas-funcionamiento/

[4] Dan Mannion. Características de la nube. Microsoft. (2014). Available: https://news.microsoft.com/es-xl/5-caracteristicas-que-definen-la-nube/

[5] Características básicas cloud computing. Ealde business school. (2019). Available: https://www.ealde.es/caracteristicas-basicas-cloud-computing/

[6] Características más relevantes del Cloud Computing. Kyocera. Available: https://www.kyoceradocumentsolutions.es/es/smarter-workspaces/business-challenges/the-cloud/las-5-caracteristicas-del-cloud-computing-mas-relevantes-para-los-negocios.html

[7] Víctor Cuervo. Características del cloud computing. AiT. (2018). Available: http://www.arquitectoit.com/cloud/caracteristicas-cloud-computing/

[8] Eva María Oviedo. Ventajas del cloud computing. Telefónica. (2015). Available: https://empresas.blogthinkbig.com/10-ventajas-del-cloud-computing/

[9] Cloud computing. Ventajas y desventajas. Zierzo Telecom. Available: https://zierzo.es/cloud-computing-ventajas-desventajas/

[10] Lucía Torres Álvarez. Ventajas de los servicios del cloud computing. Cice. (2020). Available: https://www.cice.es/noticia/ventajas-cloud-computing/

[11] Ventajas del cloud computing. IBM. Available: https://www.ibm.com/es-es/cloud/learn/benefits-of-cloud-computing

[12] Ventajas y desventajas del cloud computing. Enae. (2010). Available: https://www.enae.es/blog/ventajas-y-desventajas-del-cloud-computing

[13] Servicios del cloud computing. Akamai. Available: https://www.akamai.com/es/es/resources/cloud-computing-services.jsp

[14] Volker Bachmann. Tipos de servicios dentro del cloud computing. Escuela de organización industrial. (2012). Available: https://www.eoi.es/blogs/volkerbachmann/2012/01/14/tipos-de-servicios-dentro-del-cloud-computing/

[15] Tipos de informática en la nube. AWS. Available: https://aws.amazon.com/es/types-of-cloud-computing/

[16] Tipos de cloud computing. BeServices. (2018). Available: https://www.beservices.es/cloud-computing-tipos-n-5324-es

[17] Tipos de nubes de cloud computing. Openwebinars. Available: https://openwebinars.net/blog/tipos-de-cloud-computing/

[18] OpenStack. OpenStack. Available: https://www.openstack.org/software/

[19] OpenStack Services. OpenStack. Available: https://www.openstack.org/software/project-navigator/openstack-components/#openstack-services

[20] OpenStack. RedHat. Available: https://www.redhat.com/es/topics/openstack

[21] OpenStack compute (nova). OpenStack. Available: https://docs.openstack.org/nova/latest/

[22] OpenStack. Wikipedia. Available: https://es.wikipedia.org/wiki/OpenStack

[23] OpenStack Introduction to Object Storage (swift). OpenStack. Available: https://docs.openstack.org/swift/pike/admin/objectstorage-intro.html

[24] OpenStack components: Cinder. OpenStack. Available: https://www.openstack.org/software/releases/mitaka/components/cinder

[25] OpenStack Victoria components: keystone. OpenStack. Available: https://www.openstack.org/software/releases/victoria/components/keystone

[26] OpenStack Victoria components: Glance. OpenStack. Available: https://www.openstack.org/software/releases/victoria/components/glance

[27] OpenStack Services. OpenStack. Available: https://www.openstack.org/software/project-navigator/openstack-components#openstack-services

[28] OpenStack MicroStack. MicroStack. Available: https://microstack.run/docs

[29] Ubuntu OpenStack. Canonical. Available: https://ubuntu.com/openstack/install

[30] OpenStack command list. OpenStack. Available: https://docs.openstack.org/python-openstackclient/pike/cli/command-list.html

[31] OpenStack Omni. Platform9. Available: https://platform9.com/blog/openstack-omni-the-open-source-alternative-to-vmware-aws-for-hybrid-cloud/

[32] OpenStack Aodh. OpenStack. Available: https://docs.openstack.org/aodh/rocky/admin/index.html

[33] OpenStack devstack Victoria plugin registry. OpenStack. Available: https://docs.openstack.org/devstack/victoria/plugin-registry.html

[34] OpenStack ceilometer. OpenStack. Available: https://docs.openstack.org/newton/user-guide/cli-ceilometer.html

8. Anexos

7.1. Archivo configuración de red del nodo controller

```
# Let NetworkManager manage all devices on this system
network:
 version: 2
 renderer: NetworkManager
 ethernets:
  eno1:
   dhcp4: no
   addresses: [192.168.1.83/24]
   gateway4: 192.168.1.1
   nameservers:
    addresses: [192.168.1.1, 8.8.8.8, 8.8.4.4]
  enp2s0:
   dhcp4: no
   addresses: [192.168.1.84/24]
   gateway4: 192.168.1.1
   nameservers:
    addresses: [192.168.1.1, 8.8.8.8, 8.8.4.4]
```

7.2. Archivo configuración de red del nodo computo1

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    eno1:
      dhcp4: no
      addresses: [192.168.1.86/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [192.168.1.1, 8.8.8.8, 8.8.4.4]
```

7.3. Archivo configuración de red del nodo computo2

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    eno1:
      dhcp4: no
      addresses: [192.168.1.81/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [192.168.1.1, 8.8.8.8, 8.8.4.4]
```

7.4. Archivo local.conf del nodo controller

```
# Sample ``local.conf`` for user-configurable variables in ``stack.sh``


# NOTE: Copy this file to the root DevStack directory for it to work properly.


# ``local.conf`` is a user-maintained settings file that is sourced from ``stackrc``.
# This gives it the ability to override any variables set in ``stackrc``.
# Also, most of the settings in ``stack.sh`` are written to only be set if no
# value has already been set; this lets ``local.conf`` effectively override the
# default values.


# This is a collection of some of the settings we have found to be useful
# in our DevStack development environments. Additional settings are described
# in https://docs.openstack.org/devstack/latest/configuration.html#local-conf
# These should be considered as samples and are unsupported DevStack code.


# The ``localrc`` section replaces the old ``localrc`` configuration file.
# Note that if ``localrc`` is present it will be used in favor of this section.
[[local|localrc]]


# Minimal Contents
# ----------------
```

```
# While ``stack.sh`` is happy to run without ``localrc``, devlife is better when
# there are a few minimal variables set:

# If the ``*_PASSWORD`` variables are not set here you will be prompted to enter
# values for them by ``stack.sh``and they will be added to ``local.conf``.
ADMIN_PASSWORD=mariotfg
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD

# ``HOST_IP`` and ``HOST_IPV6`` should be set manually for best results if
# the NIC configuration of the host is unusual, i.e. ``eth1`` has the default
# route but ``eth0`` is the public interface.  They are auto-detected in
# ``stack.sh`` but often is indeterminate on later runs due to the IP moving
# from an Ethernet interface to a bridge on the host. Setting it here also
# makes it available for ``openrc`` to include when setting ``OS_AUTH_URL``.
# Neither is set by default.
MULTI_HOST=1
HOST_IP=192.168.1.83
FLOATING_RANGE=192.168.1.0/24
Q_FLOATING_ALLOCATION_POOL=start=192.168.1.230,end=192.168.1.254
PUBLIC_NETWORK_GATEWAY=192.168.1.83
```

```
# Logging
# -------

# By default ``stack.sh`` output only goes to the terminal where it runs.  It can
# be configured to additionally log to a file by setting ``LOGFILE`` to the full
# path of the destination log file.  A timestamp will be appended to the given name.
LOGFILE=$DEST/logs/stack.sh.log

# Old log files are automatically removed after 7 days to keep things neat.  Change
# the number of days by setting ``LOGDAYS``.
LOGDAYS=2

# Nova logs will be colorized if ``SYSLOG`` is not set; turn this off by setting
# ``LOG_COLOR`` false.
#LOG_COLOR=False



# Using milestone-proposed branches
# ---------------------------------

# Uncomment these to grab the milestone-proposed branches from the
# repos:
#CINDER_BRANCH=milestone-proposed
#GLANCE_BRANCH=milestone-proposed
```

```
#GLANCE_BRANCH=milestone-proposed

#HORIZON_BRANCH=milestone-proposed

#KEYSTONE_BRANCH=milestone-proposed

#KEYSTONECLIENT_BRANCH=milestone-proposed

#NOVA_BRANCH=milestone-proposed

#NOVACLIENT_BRANCH=milestone-proposed

#NEUTRON_BRANCH=milestone-proposed

#SWIFT_BRANCH=milestone-proposed


# Using git versions of clients

# ------------------------------

# By default clients are installed from pip.  See LIBS_FROM_GIT in

# stackrc for details on getting clients from specific branches or

# revisions.  e.g.

# LIBS_FROM_GIT="python-ironicclient"

# IRONICCLIENT_BRANCH=refs/changes/44/2.../1


# Swift

# -----


# Swift is now used as the back-end for the S3-like object store. Setting the

# hash value is required and you will be prompted for it if Swift is enabled

# so just set it to something already:

SWIFT_HASH=66a3d6b56c1f479c8b4e70ab5c2000f5
```

```
# For development purposes the default of 3 replicas is usually not required.
# Set this to 1 to save some resources:
SWIFT_REPLICAS=1


# The data for Swift is stored by default in (``$DEST/data/swift``),
# or (``$DATA_DIR/swift``) if ``DATA_DIR`` has been set, and can be
# moved by setting ``SWIFT_DATA_DIR``. The directory will be created
# if it does not exist.
SWIFT_DATA_DIR=$DEST/data
```

7.5. Archivo local.conf del nodo computo1

```
# Sample ``local.conf`` for user-configurable variables in ``stack.sh``


# NOTE: Copy this file to the root DevStack directory for it to work properly.


# ``local.conf`` is a user-maintained settings file that is sourced from ``stackrc``.

# This gives it the ability to override any variables set in ``stackrc``.

# Also, most of the settings in ``stack.sh`` are written to only be set if no

# value has already been set; this lets ``local.conf`` effectively override the

# default values.


# This is a collection of some of the settings we have found to be useful

# in our DevStack development environments. Additional settings are described

# in https://docs.openstack.org/devstack/latest/configuration.html#local-conf

# These should be considered as samples and are unsupported DevStack code.


# The ``localrc`` section replaces the old ``localrc`` configuration file.

# Note that if ``localrc`` is present it will be used in favor of this section.

[[local|localrc]]


# Minimal Contents

# ----------------
```

```
# While ``stack.sh`` is happy to run without ``localrc``, devlife is better when
# there are a few minimal variables set:

# If the ``*_PASSWORD`` variables are not set here you will be prompted to enter
# values for them by ``stack.sh``and they will be added to ``local.conf``.
ADMIN_PASSWORD=mariotfg
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD

# ``HOST_IP`` and ``HOST_IPV6`` should be set manually for best results if
# the NIC configuration of the host is unusual, i.e. ``eth1`` has the default
# route but ``eth0`` is the public interface.  They are auto-detected in
# ``stack.sh`` but often is indeterminate on later runs due to the IP moving
# from an Ethernet interface to a bridge on the host. Setting it here also
# makes it available for ``openrc`` to include when setting ``OS_AUTH_URL``.
# Neither is set by default.
MULTI_HOST=1
HOST_IP=192.168.1.86
FLOATING_RANGE=192.168.1.0/24
Q_FLOATING_ALLOCATION_POOL=start=192.168.1.249,end=192.168.1.254
PUBLIC_NETWORK_GATEWAY=192.168.1.83
DATABASE_TYPE=mysql
SERVICE_HOST=192.168.1.83
```

```
MYSQL_HOST=$SERVICE_HOST

RABBIT_HOST=$SERVICE_HOST

GLANCE_HOSTPORT=$SERVICE_HOST:9292

ENABLED_SERVICES=n-cpu,q-agt,c-vol,placement-client

NOVA_VNC_ENABLED=True

NOVNCPROXY_URL="http://$SERVICE_HOST:6080/vnc_lite.html"

VNCSERVER_LISTEN=$HOST_IP

VNCSERVER_PROXYCLIENT_ADDRESS=$VNCSERVER_LISTEN

#HOST_IPV6=2001:db8::7



# Logging

# -------


# By default ``stack.sh`` output only goes to the terminal where it runs.  It can

# be configured to additionally log to a file by setting ``LOGFILE`` to the full

# path of the destination log file.  A timestamp will be appended to the given name.

LOGFILE=$DEST/logs/stack.sh.log


# Old log files are automatically removed after 7 days to keep things neat.  Change

# the number of days by setting ``LOGDAYS``.

LOGDAYS=2
```

```
# Nova logs will be colorized if ``SYSLOG`` is not set; turn this off by setting
# ``LOG_COLOR`` false.
#LOG_COLOR=False



# Using milestone-proposed branches
# ---------------------------------


# Uncomment these to grab the milestone-proposed branches from the
# repos:
#CINDER_BRANCH=milestone-proposed
#GLANCE_BRANCH=milestone-proposed
#HORIZON_BRANCH=milestone-proposed
#KEYSTONE_BRANCH=milestone-proposed
#KEYSTONECLIENT_BRANCH=milestone-proposed
#NOVA_BRANCH=milestone-proposed
#NOVACLIENT_BRANCH=milestone-proposed
#NEUTRON_BRANCH=milestone-proposed
#SWIFT_BRANCH=milestone-proposed


# Using git versions of clients
# -----------------------------
# By default clients are installed from pip.  See LIBS_FROM_GIT in
```

```
# stackrc for details on getting clients from specific branches or
# revisions.  e.g.
# LIBS_FROM_GIT="python-ironicclient"
# IRONICCLIENT_BRANCH=refs/changes/44/2.../1


# Swift
# -----


# Swift is now used as the back-end for the S3-like object store. Setting the
# hash value is required and you will be prompted for it if Swift is enabled
# so just set it to something already:
SWIFT_HASH=66a3d6b56c1f479c8b4e70ab5c2000f5


# For development purposes the default of 3 replicas is usually not required.
# Set this to 1 to save some resources:
SWIFT_REPLICAS=1


# The data for Swift is stored by default in (``$DEST/data/swift``),
# or (``$DATA_DIR/swift``) if ``DATA_DIR`` has been set, and can be
# moved by setting ``SWIFT_DATA_DIR``. The directory will be created
# if it does not exist.
SWIFT_DATA_DIR=$DEST/data
```

7.6. Archivo local.conf del nodo computo2

```
# Sample ``local.conf`` for user-configurable variables in ``stack.sh``


# NOTE: Copy this file to the root DevStack directory for it to work properly.


# ``local.conf`` is a user-maintained settings file that is sourced from ``stackrc``.

# This gives it the ability to override any variables set in ``stackrc``.

# Also, most of the settings in ``stack.sh`` are written to only be set if no

# value has already been set; this lets ``local.conf`` effectively override the

# default values.


# This is a collection of some of the settings we have found to be useful

# in our DevStack development environments. Additional settings are described

# in https://docs.openstack.org/devstack/latest/configuration.html#local-conf

# These should be considered as samples and are unsupported DevStack code.


# The ``localrc`` section replaces the old ``localrc`` configuration file.

# Note that if ``localrc`` is present it will be used in favor of this section.

[[local|localrc]]


# Minimal Contents

# ----------------
```

```
# While ``stack.sh`` is happy to run without ``localrc``, devlife is better when
# there are a few minimal variables set:

# If the ``*_PASSWORD`` variables are not set here you will be prompted to enter
# values for them by ``stack.sh``and they will be added to ``local.conf``.
ADMIN_PASSWORD=mariotfg
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD

# ``HOST_IP`` and ``HOST_IPV6`` should be set manually for best results if
# the NIC configuration of the host is unusual, i.e. ``eth1`` has the default
# route but ``eth0`` is the public interface.  They are auto-detected in
# ``stack.sh`` but often is indeterminate on later runs due to the IP moving
# from an Ethernet interface to a bridge on the host. Setting it here also
# makes it available for ``openrc`` to include when setting ``OS_AUTH_URL``.
# Neither is set by default.
MULTI_HOST=1
HOST_IP=192.168.1.81
FLOATING_RANGE=192.168.1.0/24
Q_FLOATING_ALLOCATION_POOL=start=192.168.1.249,end=192.168.1.254
PUBLIC_NETWORK_GATEWAY=192.168.1.83
DATABASE_TYPE=mysql
SERVICE_HOST=192.168.1.83
```

```
MYSQL_HOST=$SERVICE_HOST

RABBIT_HOST=$SERVICE_HOST

GLANCE_HOSTPORT=$SERVICE_HOST:9292

ENABLED_SERVICES=n-cpu,q-agt,c-vol,placement-client

NOVA_VNC_ENABLED=True

NOVNCPROXY_URL="http://$SERVICE_HOST:6080/vnc_lite.html"

VNCSERVER_LISTEN=$HOST_IP

VNCSERVER_PROXYCLIENT_ADDRESS=$VNCSERVER_LISTEN

#HOST_IPV6=2001:db8::7



# Logging

# -------


# By default ``stack.sh`` output only goes to the terminal where it runs.  It can

# be configured to additionally log to a file by setting ``LOGFILE`` to the full

# path of the destination log file.  A timestamp will be appended to the given name.

LOGFILE=$DEST/logs/stack.sh.log


# Old log files are automatically removed after 7 days to keep things neat.  Change

# the number of days by setting ``LOGDAYS``.

LOGDAYS=2
```

```
# Nova logs will be colorized if ``SYSLOG`` is not set; turn this off by setting
# ``LOG_COLOR`` false.
#LOG_COLOR=False



# Using milestone-proposed branches
# ---------------------------------


# Uncomment these to grab the milestone-proposed branches from the
# repos:
#CINDER_BRANCH=milestone-proposed
#GLANCE_BRANCH=milestone-proposed
#HORIZON_BRANCH=milestone-proposed
#KEYSTONE_BRANCH=milestone-proposed
#KEYSTONECLIENT_BRANCH=milestone-proposed
#NOVA_BRANCH=milestone-proposed
#NOVACLIENT_BRANCH=milestone-proposed
#NEUTRON_BRANCH=milestone-proposed
#SWIFT_BRANCH=milestone-proposed


# Using git versions of clients
# -----------------------------
# By default clients are installed from pip.  See LIBS_FROM_GIT in
```

```
# stackrc for details on getting clients from specific branches or

# revisions.  e.g.

# LIBS_FROM_GIT="python-ironicclient"

# IRONICCLIENT_BRANCH=refs/changes/44/2.../1


# Swift

# -----


# Swift is now used as the back-end for the S3-like object store. Setting the

# hash value is required and you will be prompted for it if Swift is enabled

# so just set it to something already:

SWIFT_HASH=66a3d6b56c1f479c8b4e70ab5c2000f5


# For development purposes the default of 3 replicas is usually not required.

# Set this to 1 to save some resources:

SWIFT_REPLICAS=1


# The data for Swift is stored by default in (``$DEST/data/swift``),

# or (``$DATA_DIR/swift``) if ``DATA_DIR`` has been set, and can be

# moved by setting ``SWIFT_DATA_DIR``. The directory will be created

# if it does not exist.

SWIFT_DATA_DIR=$DEST/data
```

7.7. Script para crear proyectos

```
source admin-openrc.sh


month=$(date +%m)

year=$(date +%Y)

fecha=$year$month


i=0


while IFS= read -r correo

do

  # echo "$correo"

  IFS='@' read -ra ADDR <<< "$correo"

  ((i++))

  echo "Generando  proyecto: ${ADDR[0]}"

  comando="openstack project create --domain default ${ADDR[0]}"

  comando2="$($comando)"

  echo "$comando2"

  echo "Configurando proyecto: ${ADDR[0]}"

  comando3="openstack role add --user admin --project ${ADDR[0]} admin"

  comando4="$($comando3)"

  echo "$comando4"
```

```
comando5="openstack role add --user demo --project ${ADDR[0]} admin"

  comando6="$($comando5)"

  echo "$comando6"



  echo "Generando usuario: ${ADDR[0]}"

  comando7="openstack user create --project ${ADDR[0]} --email $correo --password
${ADDR[0]}$fecha ${ADDR[0]}"

  comando8="$($comando7)"

  echo "$comando8"

  comando9="openstack role add --user ${ADDR[0]} --project ${ADDR[0]} member"

  comando10="$($comando9)"

  echo "$comando10"



  echo "Generando network: ${ADDR[0]}"

  comando11="openstack network create --project ${ADDR[0]} --no-share --internal
${ADDR[0]}"

  comando12="$($comando11)"

  echo "$comando12"

  comando13="openstack subnet create --project ${ADDR[0]} --subnet-range
10.0.$i.0/24 --network ${ADDR[0]} ${ADDR[0]}"

  comando14="$($comando13)"

  echo "$comando14"
```

```
echo "Generando router: ${ADDR[0]}"
  comando15="openstack router create --project ${ADDR[0]} ${ADDR[0]}"
  comando16="$($comando15)"
  echo "$comando16"
  comando17="openstack router set --external-gateway public ${ADDR[0]}"
  comando18="$($comando17)"
  echo "$comando18"
  comando19="openstack router add subnet ${ADDR[0]} ${ADDR[0]}"
  comando20="$($comando19)"
  echo "$comando20"
done < correos.txt
```

7.8. Script para eliminar proyectos

```
#!/bin/bash

source admin-openrc.sh


month=$(date +%m)

year=$(date +%Y)

fecha=$year$month


openstack server list --all-projects | grep 192 | awk '//{print $2}' >> instancias.txt


while IFS= read -r instancia

do

  #echo "$instancia"

  IFS='\n' read -ra ADDR <<< "$instancia"

  echo "Eliminando instancia: ${ADDR[0]}"

  comando="openstack server delete --wait  ${ADDR[0]}"

  comando2="$($comando)"

  echo  "$comando2"

done < instancias.txt


openstack floating ip list | grep 192 | awk '//{print $4}' >> floatingip.txt
```

```
while IFS= read -r floatingip
do
  #echo "$floatingip"
  IFS='\n' read -ra ADDR <<< "$floatingip"
  echo "Liberando IP Flotante: ${ADDR[0]}"
  comando3="openstack floating ip delete ${ADDR[0]}"
  comando4="$($comando3)"
  echo "$comando4"
done < floatingip.txt


while IFS= read -r correo
do
  # echo "$correo"
  IFS='@' read -ra ADDR <<< "$correo"
  echo "Eliminando network: ${ADDR[0]}"
  comando5="openstack router remove subnet ${ADDR[0]} ${ADDR[0]}"
  comando6="$($comando5)"
  echo "$comando6"
  comando7="openstack subnet delete ${ADDR[0]}"
  comando8="$($comando7)"
  echo "$comando8"
```

```
comando9="openstack network delete ${ADDR[0]}"
  comando10="$($comando9)"
  echo "$comando10"
  echo "Eliminando router: ${ADDR[0]}"
  comando11="openstack router delete ${ADDR[0]}"
  comando12="$($comando11)"
  echo "$comando12"
  echo "Eliminando usuario: ${ADDR[0]}"
  comando13="openstack user delete ${ADDR[0]}"
  comando14="$($comando13)"
  echo "$comando14"
  echo "Eliminando proyecto: ${ADDR[0]}"
  comando15="openstack project delete ${ADDR[0]}"
  comando16="$($comando15)"
  echo "$comando16"
done < correos.txt


rm instancias.txt
rm floatingip.txt
```

Universidad de Alcalá