

Implementación de un IaaS sobre hardware del laboratorio

MEMORIA DEL PROYECTO



Daniel González y Ronaldo Campuzano

UNIVERSIDAD POLITÉCNICA DE MADRID | 3/04/2019



UNIVERSIDAD
POLITÉCNICA
DE MADRID

Índice de Contenido

1 - Introducción: Visión general del proyecto.....	5
2 - Tecnologías implicadas.....	6
2.1 - OpenStack.....	6
2.2 - Docker.....	7
2.3 - Kolla-Ansible.....	7
3 - Infraestructura y componentes del IaaS.....	8
3.1 - Infraestructura y funcionalidad de los nodos	8
3.2 - Componentes clave en la implementación de OpenStack distribuido	11
4 - Instalación y despliegue de la infraestructura OpenStack distribuida	11
4.1 - Consideraciones iniciales.....	11
4.2 - Configurar PXE y servidor web de Apache.....	12
4.3 - Configurar servidor DHCP en el controlador	13
4.4 - Configurar iptables persistentes y forwarding	13
4.5 - Instalar Ubuntu en nodos de cómputo vía PXE.....	14
4.6 - Configurar SSH.....	14
4.7 - Instalar y configurar Kolla-Ansible	15
4.7.1 - Scripts de instalación de dependencias de Kolla-Ansible.....	15
4.7.2 – Configuración de Kolla-Ansible: Docker, inventario multinode y globals.yml.....	15
4.8 - Desplegar infraestructura OpenStack.....	17
4.9 - Despliegue de red demo en plataforma OpenStack	18
5 - Conclusiones	19
6 - Bibliografía	20
A - Anexos	21
A.1 - Cuaderno de Equipo	21
A.2 - Manual de Instalación y Despliegue de OpenStack	26
A.2.1 - Consideraciones iniciales.....	26
A.2.1.1 - Renombrar Interfaces Ethernet-USB	27
A.2.1.2 - Duplicado de MAC en paquetes salientes	28
A.2.2 - Configurar PXE.....	28
A.2.2.1 - Descarga y montaje de UBUNTU-18.04.1-SERVER.....	28
A.2.2.2 Crear directorio tftpboot.....	28
A.2.2.3 - Modificar los menús de carga de tftpboot	29
A.2.3 -Configurar Servidor Web.....	30
A.2.3.1 - Instalar y configurar apache2.....	30
A.2.3.2 - Configurar ficheros de instalación	30
A.2.3.2.1 - Copiar la imagen del sistema en el directorio del servidor local.....	30
A.2.3.2.2 - Configurar rutina inicial de instalación.....	31

A.2.3.2.3 - Añadir fuente de la imagen en el servidor.....	32
A.2.4 - Configurar DHCP	32
A.2.4.1 - Instalar y configurar <i>dnsmasq</i>	32
A.2.4.2 - Comprobar que el servicio está activo	33
A.2.5 - Configurar iptables persistentes y forwarding	33
A.2.5.1 - Abrir puertos firewall en el servidor.....	33
A.2.5.2 - En el servidor host se añade el bit de forwarding	34
A.2.5.3 - Configurar reglas NAT de forma persistente	34
A.2.6 - Instalar Ubuntu vía PXE	34
A.2.7 - Configurar SSH.....	35
A.2.7.1 - Registrar los nombres de los Hosts.....	35
A.2.7.2 - Probar la conexión SSH para root y reicloud.....	36
A.2.8 - Instalar Kolla-Ansible.....	36
A.2.8.1 - Instalación en los Nodos de Cómputo	36
A.2.8.2 - Instalación en el Controlador.....	36
A.2.9 - Configuración del entorno Kolla	38
A.2.9.1 Preparación para despliegue de Docker	38
A.2.9.2 - Modificar el inventario multinode	39
A.2.9.2.1 - Código de configuración.....	39
A.2.9.3 - Modificar el fichero <i>globals.yml</i>	40
A.2.9.2.1 - Código de configuración.....	40
A.2.10 - Despliegue de OpenStack	41
A.2.10.1 - Despliegue de una Red demo.....	42
B - Troubleshooting	44
B.1 - Fallo de servicio DHCP	44
B.2 - Comprobación de instalación de Kolla-Ansible	44
B.3 - Fallo de ping Ansible	45
B.4 - Errores en despliegue de contenedores.....	45
B.5 - Despliegue de una Red demo personalizada.....	46

Índice de Figuras de la Memoria

Figura 1: Diagrama de topología de OpenStack distribuido.....	5
Figura 2: Esquema general de la plataforma OpenStack.	6
Figura 3: Arquitectura de Docker.	7
Figura 4: Arquitectura de Kolla-Ansible.	8
Figura 5: Esquema de conexión de <i>bridges</i> br-int/br.tun.....	10
Figura 6: Esquema de virtualización de <i>bridges</i> para los nodos de cómputo.	10
Figura 7: Contenedores Docker del controlador con los servicios de OpenStack.	11
Figura 8: Dashboard de OpenStack.	18
Figura 9: Diagrama de planificación inicial.....	21

Índice de Ilustraciones del Manual

Ilustración 1: Topología del escenario OpenStack.	26
Ilustración 2: Muestra de las particiones de disco en el Controlador luego de la instalación. ..	27
Ilustración 3: Montaje de la imagen de Ubuntu 18.04.....	28
Ilustración 4: Estado activo del servicio web apache2.	30
Ilustración 5: Estado del servicio de DHCP dnsmasq.	33
Ilustración 6: Estado del servicio de firewall ufw.	33
Ilustración 7: Comprobación de conexión entre controlador y nodos de cómputo.	41
Ilustración 8: Contenedores desplegados en el nodo controlador.	42
Ilustración 9: Dashboard de OpenStack (servicio Horizon).	42

1 - Introducción: Visión general del proyecto

OpenStack es una plataforma de computación en la nube que permite la gestión de grandes centros de datos basándose en el paradigma de *Infrastructure as a Service* (IaaS).

El principal objetivo de este proyecto ha sido proveer un servicio de IaaS, configurando y desplegando la infraestructura de OpenStack sobre un escenario físico distribuido sobre equipos finales del laboratorio utilizando servicios basados en *Docker*.

El escenario está formado por tres equipos finales conectados entre sí:

- Un servidor que actuará como controlador de la infraestructura OpenStack. También actúa como nodo de red y de configuración y despliegue de la infraestructura de OpenStack mediante Kolla-Ansible.
- Dos servidores de cómputo que se encargan de gestionar los servicios y máquinas virtuales desplegadas en OpenStack.

El diagrama de topología de la infraestructura de OpenStack distribuida se puede observar en la Figura 1.

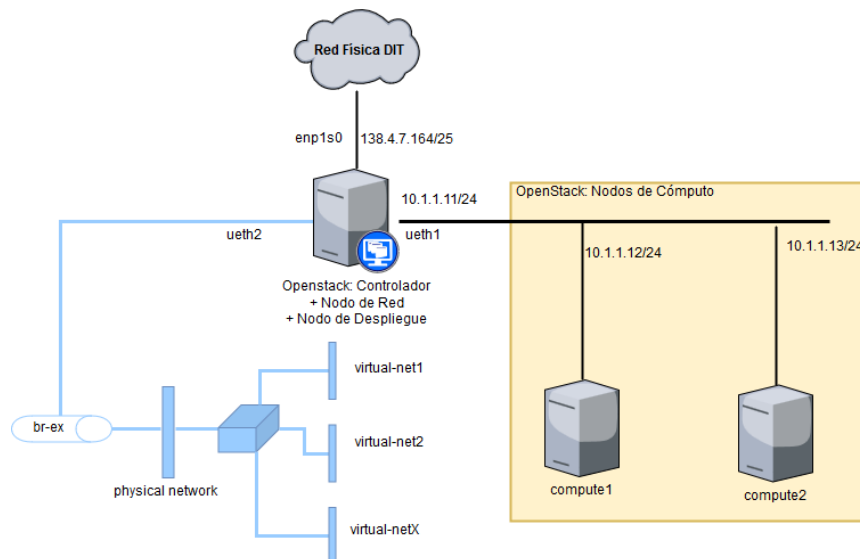


Figura 1: Diagrama de topología de OpenStack distribuido.

Para la implementación de la infraestructura OpenStack distribuida en un clúster de servidores dedicados se hace uso de la herramienta Kolla-Ansible que permite realizar el despliegue de los servicios de OpenStack a partir de contenedores Docker, utilizando *playbooks* de configuración de Ansible.

A continuación, se explica en detalle cada una de las tecnologías utilizadas para el despliegue de la plataforma IaaS y la funcionalidad detallada de cada nodo y servicio componente clave para la implementación de la infraestructura OpenStack distribuida. Posteriormente, se detalla cómo realizar paso a paso la instalación y despliegue de la plataforma OpenStack distribuida.

2 - Tecnologías implicadas

En este apartado de la memoria se mencionan y explican cada una de las tecnologías más importantes que están involucradas en la implementación de la plataforma IaaS distribuida.

2.1 - OpenStack

OpenStack es un sistema operativo o plataforma de computación en la nube de código abierto para controlar grandes conjuntos de recursos de computación, almacenamiento y redes a través de un centro de datos, todo administrado a través de un *framework* que le da a los administradores el control mientras que permite a sus usuarios aprovisionar recursos a través de una interfaz web. OpenStack sigue el modelo Infraestructura como Servicio (IaaS). Ver su esquema general en la Figura 2.

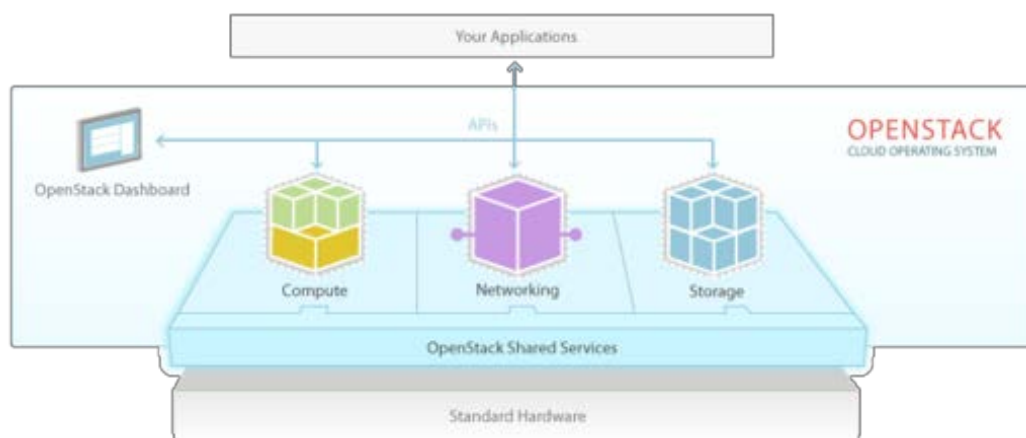


Figura 2: Esquema general de la plataforma OpenStack.

OpenStack tiene una arquitectura modular formada por varios componentes. Entre los componentes más importantes se encuentran:

- Compute (Nova): administra todas las máquinas virtuales y también contiene una Nova-API para gestionar los recursos de cómputo.
- Object Store (Glance): proporciona un repositorio para las imágenes de disco virtual.
- Object Store (Swift): proporciona un sistema de almacenamiento de objetos redundante y escalable. OpenStack es responsable de asegurar la replicación y la integridad de los datos almacenados en el clúster.
- Dashboard (Horizon): proporciona una interfaz web para controlar todos los servicios de OpenStack.
- Servicio de Identidad (Keystone): proporciona autenticación y autorización para todos los servicios de OpenStack.
- Networking (Neutron): anteriormente conocida como nova-network y Quantum. Proporciona la conectividad y recursos de red como un servicio de OpenStack.
- Block Storage (Cinder): anteriormente conocido como nova-volume. Proporciona dispositivos de almacenamiento a nivel de bloques persistentes para utilizar con instancias de máquinas virtuales generadas con OpenStack Compute.

- Orquestación (Heat): es un servicio para orquestar y lanzar múltiples aplicaciones compuestas en la nube utilizando plantillas de código, tanto a través de una API REST OpenStack nativa como una API de consulta compatible con CloudFormation.

2.2 - Docker

Docker es una plataforma de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores *software*, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Ver la arquitectura de los contenedores Docker en la Figura 3.

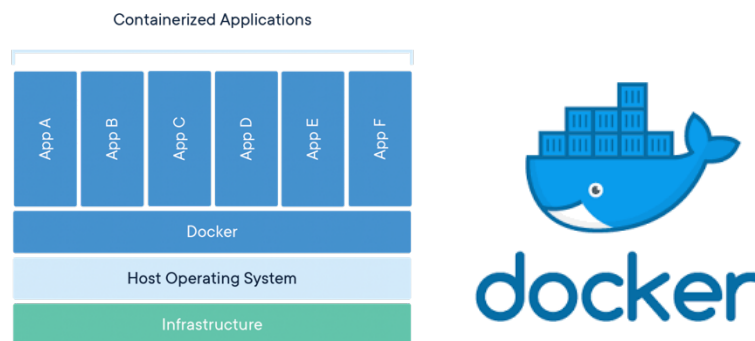


Figura 3: Arquitectura de Docker.

Docker implementa una API de alto nivel para proporcionar contenedores livianos que ejecutan procesos de manera aislada.

Construido sobre las facilidades proporcionadas por el *kernel* Linux, un contenedor Docker, a diferencia de una máquina virtual, no requiere incluir un sistema operativo independiente. En su lugar, se basa en las funcionalidades del *kernel* y utiliza el aislamiento de recursos (CPU, la memoria, el bloque E/S, red, etc.) y *namespaces* separados para aislar la vista de una aplicación del sistema operativo.

Mediante el uso de contenedores, los recursos pueden ser aislados, los servicios restringidos, y se otorga a los procesos la capacidad de tener una visión casi completamente privada del sistema operativo con su propio identificador de espacio de proceso, una estructura del sistema de archivos, y las interfaces de red. Contenedores múltiples comparten el mismo núcleo, pero cada contenedor puede ser restringido a utilizar solo una cantidad definida de recursos como CPU, memoria y E/S.

Usar Docker para crear y gestionar contenedores puede simplificar la creación de sistemas distribuidos, permitiendo que múltiples aplicaciones, las tareas de los trabajadores y otros procesos funcionen de forma autónoma en una única máquina física o en varias máquinas virtuales.

Mediante esta tecnología de virtualización ligera se desplegarán los diferentes servicios dependientes de OpenStack para este caso de estudio de la plataforma IaaS distribuida.

2.3 - Kolla-Ansible

Kolla-Ansible es un entorno proporciona contenedores y herramientas para la implementación y despliegue de plataformas OpenStack que son escalables, rápidas, confiables y actualizables. Esto

permite a los administradores con poca experiencia implementar OpenStack y modificar rápidamente su configuración para que se ajuste a los requisitos exactos del operador.

Su misión es realizar el despliegue de los servicios de OpenStack a partir de contenedores Docker, haciendo uso de *playbooks* de configuración de Ansible. Ver la arquitectura general de Kolla-Ansible en la Figura 4.

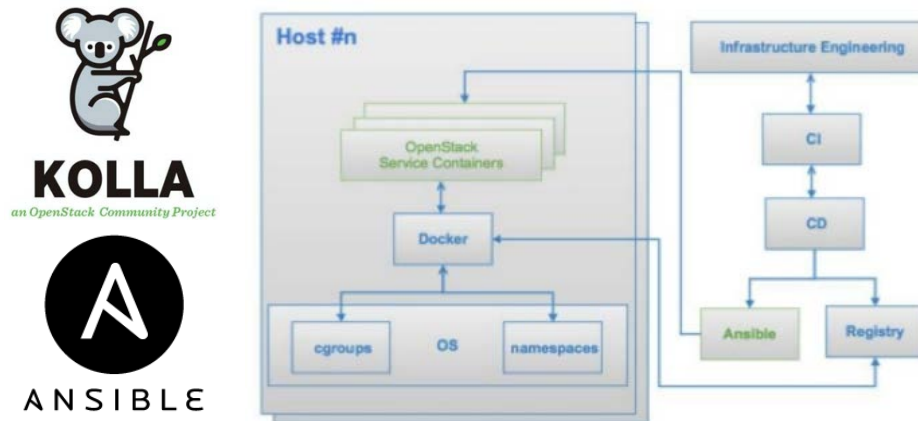


Figura 4: Arquitectura de Kolla-Ansible.

3 - Infraestructura y componentes del IaaS

En este apartado se describe la infraestructura del entorno IaaS desplegado y en concreto la funcionalidad realizada por cada equipo que forma parte del sistema. Además, se especifica la utilidad de los dos servicios o componentes claves para el despliegue de la infraestructura OpenStack distribuida: Kolla-Ansible y Docker.

3.1 - Infraestructura y funcionalidad de los nodos

Tal como se ha introducido, la infraestructura de nuestro escenario de OpenStack distribuido está formado por tres equipos terminales: un nodo controlador del sistema OpenStack (controller) y dos nodos de computación OpenStack (compute1 y compute2).

El equipo *controller*, además de actuar como nodo controlador del sistema OpenStack, también actúa como nodo de red y de configuración y despliegue de la infraestructura de OpenStack mediante Kolla-Ansible. Los roles asignados al nodo controlador se detallan a continuación:

- Como nodo controlador, permite ejecutar las principales aplicaciones de control y el interfaz de usuario del sistema OpenStack (Horizon Dashboard).
- Como nodo de red (Neutron Networking), se encarga de proporcionar los servicios de red necesarios para crear las redes virtuales donde se asocian las máquinas virtuales y permiten otras ciertas funciones como habilitar el acceso a la red exterior configurando y gestionando las IP flotantes y el servicio NAT.
- Como nodo de configuración y despliegue se encarga de gestionar las herramientas, servicios y *scripts* de configuración necesarios para poder implementar el escenario distribuido de OpenStack. Entre las funciones asociadas al despliegue podemos encontrar:

- Configurar PXE (Preboot eXecution Environment) y DHCP (en modalidad dnsmasq: servidores DHCP + DNS + TFTP con soporte para PXE) para posteriormente poder transferir, arrancar e instalar el sistema operativo en los nodos de cómputo a través de la red y a partir de la imagen ISO del S.O del equipo anfitrión.
- Activar el *bit* de *forwarding* en el *controller* para configurarlo como puerta de acceso a la red del DIT y tener acceso a Internet desde los *hosts* de computación.
- Configurar reglas NAT persistentes mediante iptables para permitir a los equipos terminales de la red interna salir al exterior.
- Activar el servicio SSH para permitir la gestión remota de los nodos de cómputo desde el *controller*.
- Instalar Kolla-Ansible para poder configurar y desplegar la plataforma distribuida de OpenStack.

Los equipos de cómputo (compute1 y compute2) se encargan de gestionar los servicios de los recursos de computación y de dar soporte a las máquinas virtuales desplegadas en OpenStack (Compute Nova).

Estos tres equipos están conectados entre sí a través de una red privada LAN conmutada con direccionamiento 10.1.1.0/24. El diagrama de topología se puede ver en la Figura 1 de la sección 1 - Introducción: Visión general del proyecto.

El equipo controlador dispone de las siguientes tres interfaces:

1. enp1s0: interfaz conectada a la red externa del DIT con IP pública 138.4.7.164/25 asociada para acceso a Internet.
2. ueth1: interfaz conectada a la red interna para la comunicación con los nodos de cómputo (compute1 y compute2) con una IP 10.1.1.11/24. Los tres equipos están conectados a través de un *switch*. Decir que esta interfaz se ha incluido con un adaptador USB Ethernet.
3. ueth2: interfaz conectada a la red del DIT, pero sin IP asignada para asociar el *bridge* “*br-ex*” de virtualización necesario para OpenStack y establecer así la conexión entre la red física y las redes y máquinas virtuales de OpenStack. Decir que esta interfaz se ha incluido con un adaptador USB Ethernet.

Los nodos de cómputo (compute1 y compute2) tienen una única interfaz (IP 10.1.1.12/24 para compute1 e IP 10.1.1.13/24 para compute2) que los conecta al *switch* de la red LAN interna para la comunicación con el controlador.

Una vez configurado y desplegado el entorno OpenStack distribuido, aparecerán nuevas interfaces asociadas a los equipos terminales relacionadas con la comunicación con los contenedores Docker que orquestan los servicios de OpenStack, interfaces para la comunicación con las máquinas y redes virtuales instanciadas en OpenStack y otras interfaces *bridge* de tipo openvswitch.

Entre las interfaces *bridges* de virtualización utilizadas por OpenStack encontramos:

- br-ex: permite asociar la red interna virtual de la plataforma OpenStack a la infraestructura de red externa. Esta interfaz *bridge* sólo la tiene el nodo controlador.

- **br-int:** al utilizar virtualización de redes conseguimos que las instancias de una misma red se comuniquen entre sí y compartan algunos elementos comunes, pero permaneciendo al margen del resto de redes virtuales. Esto debe producirse independientemente del *hypervisor* (nodo de computación) en el que se estén ejecutando las máquinas virtuales que vayamos a desplegar. Una de las formas de conseguir esto con OVS es crear un *switch* virtual en cada nodo que se denomina *bridge* de integración o *br-int*. Esta interfaz *bridge* la poseen los tres equipos, de tal forma que las máquinas virtuales alojadas en los nodos de cómputo se puedan asociar entre sí a través de las redes instanciadas desde nodo de red y controlador del sistema OpenStack
- **br-tun:** se encarga de comunicar cada nodo del sistema OpenStack con los demás a través de túneles GRE que garantizan la seguridad de las comunicaciones entre nodos. El *bridge* *br-tun* se encuentra conectado al *bridge* *br-int* a través de una interfaz especial denominada puerto *patch*. Esta interfaz *bridge* la poseen los tres equipos, de tal forma que se establece un tráfico de red seguro entre las máquinas virtuales mediante túneles GRE.

En la Figura 5 se representa la situación de los *bridges* de virtualización tras una instalación del servicio de red Neutron de OpenStack distribuido.

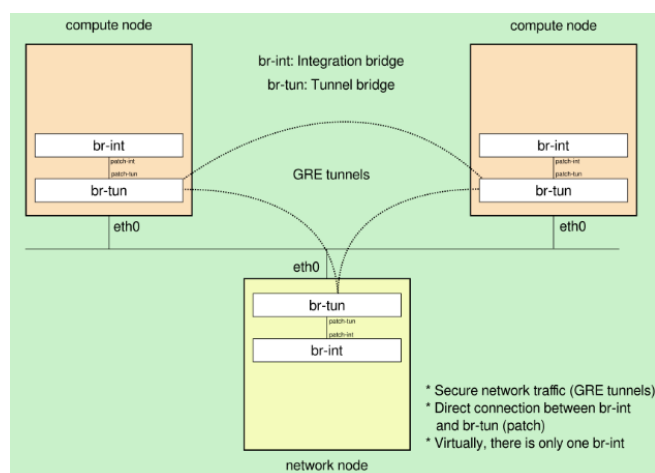


Figura 5: Esquema de conexión de *bridges* *br-int*/*br-tun*.

La forma en la que se interconectan todas las interfaces de tipo *bridge* en los nodos de cómputo a la hora de instanciar máquinas virtuales se muestra en la Figura 6.

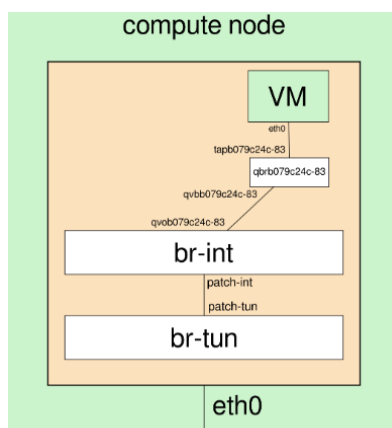


Figura 6: Esquema de virtualización de *bridges* para los nodos de cómputo.

(Para comprender mejor la estructura y funcionamiento de los *bridges* de virtualización utilizados en la arquitectura OpenStack distribuida, se recomienda consultar los siguientes enlaces: <https://elatov.github.io/2018/01/openstack-ansible-and-kolla-on-ubuntu-1604/> y <http://iesgn.github.io/cloud/cursos/u8/red-privada>).

3.2 - Componentes clave en la implementación de OpenStack distribuido

Entre los componentes y servicios clave en la implementación del IaaS se encuentra la herramienta Kolla-Ansible, la cual permite realizar el despliegue de los servicios de OpenStack a partir de contenedores Docker, haciendo uso de *playbooks* de configuración de Ansible. Los *playbooks* utilizados para el despliegue son:

- **bootstrap-servers** para instalar las dependencias de Kolla.
- **prechecks** para realizar comprobaciones previas a la implementación de OpenStack.
- **deploy** para realizar la configuración y despliegue de los contenedores Docker que conforman el escenario OpenStack
- **post-deploy** para generar el archivo “admin-openrc” donde se configuran las credenciales para el usuario administrador de OpenStack.

Una vez instalados estos *playbooks*, OpenStack estará activo y funcionando.

Los servicios o componentes de OpenStack como Neutron, Nova, Glance o Heat son desplegados en diferentes contenedores Docker. A continuación, se muestran los contenedores activos en el *controller* tras desplegar la infraestructura IaaS distribuida (ver la Figura 7).

```

root@cloud01:~# docker ps -a
CONTAINER ID        IMAGE                                     COMMAND                  CREATED             STATUS              PORTS              NAMES
31885d199aa8        kolla/centos-binary-horizon:queens      "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           horizon
a2882808eaf1        kolla/centos-binary-heat-engine:queens  "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           heat_engine
f215810a7fba        kolla/centos-binary-heat-api-cfn:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           heat_api_cfn
8728b3c5345c        kolla/centos-binary-heat-api:queens     "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           heat_api
5386c90fc41f        kolla/centos-binary-neutron-metadata-agent:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           neutron_metadata_agent
74853d0e7f49        kolla/centos-binary-neutron-l3-agent:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           neutron_l3_agent
c160f0ac39ca        kolla/centos-binary-neutron-dhcp-agent:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           neutron_dhcp_agent
f782166e21c        kolla/centos-binary-neutron-openvswitch-agent:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           neutron_openvswitch_agent
b3658838c45e        kolla/centos-binary-neutron-server:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           neutron_server
02301c1c476        kolla/centos-binary-openvswitch-vswitchd:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           openvswitch_vswitchd
2d4cd45dc406        kolla/centos-binary-openvswitch-db-server:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           openvswitch_db
63dd0956509e        kolla/centos-binary-nova-novncproxy:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           nova_novncproxy
2ea225d60e0f        kolla/centos-binary-nova-consoleauth:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           nova_consoleauth
97fabeb149e8        kolla/centos-binary-nova-conductor:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           nova_conductor
664c208ab4b6        kolla/centos-binary-nova-scheduler:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           nova_scheduler
60431c1c021        kolla/centos-binary-nova-api:queens     "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           nova_api
00802c19f7aa        kolla/centos-binary-nova-placement-api:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           placement_api
bcff3db37e5        kolla/centos-binary-glance-registry:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           glance_registry
ba8bd0f5fd4c        kolla/centos-binary-glance-api:queens   "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           glance_api
d1e3cf887076        kolla/centos-binary-keystone:queens     "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           keystone
3e033f7866d8        kolla/centos-binary-rabbitmq:queens     "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           rabbitmq
4443f754028f        kolla/centos-binary-mariadb:queens      "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           mariadb
875f8e065576        kolla/centos-binary-cron:queens         "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           cron
06c1194c5be1        kolla/centos-binary-kolla-toolbox:queens "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           kolla_toolbox
9a0ac7ce015c        kolla/centos-binary-fluentd:queens      "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           fluentd
9cdabceab6bd        kolla/centos-binary-memcached:queens    "dumb-init --single-..." 3 weeks ago         Up 3 weeks          8080-8081           memcached
    
```

Figura 7: Contenedores Docker del controlador con los servicios de OpenStack.

4 - Instalación y despliegue de la infraestructura OpenStack distribuida

En este apartado se realiza una descripción general y teórica de cómo realizar la instalación y despliegue de la infraestructura OpenStack distribuida comentada en apartados anteriores. Los pasos a seguir y su explicación para tal propósito son detallados en los siguientes subapartados.

4.1 - Consideraciones iniciales

En primer lugar, se debe disponer de los equipos, herramientas e infraestructura de red necesarios para la implementación de la plataforma IaaS OpenStack.

- Tres equipos PC: un equipo actuará de nodo controlador, nodo de red y nodo de configuración y despliegue del sistema OpenStack. Los otros dos serán los nodos de cómputo OpenStack.
- Disco duro o SSD limpio para los tres equipos.
- Monitores a los que conectar cada uno de los equipos PC.
- *Switch* para conectar los tres equipos de la infraestructura OpenStack con conexión por cableado Ethernet. Los tres equipos tendrán conectividad entre sí a partir de una LAN privada conmutada.
- A parte de la red interna con los nodos de cómputo, el nodo controlador tendrá dos conexiones externas: una interfaz conectada a la red externa del DIT con IP pública asociada y otra conectada a la red del DIT, pero sin IP asignada para asociar los *bridges* de virtualización necesarios para OpenStack. En caso de no disponer de tarjetas o interfaces de red suficientes en el equipo controlador, utilizar adaptadores USB Ethernet (para cambiar el nombre de las interfaces asociadas a estos adaptadores revisar la sección 1.1 del Anexo A.2).
- *Pen Drive USB bootable* con la distribución de Linux Ubuntu 18.04.1.

Una vez provistos de todos los componentes y habiendo montado y conectado la infraestructura de equipos y red, se empieza con la instalación del sistema operativo en los equipos finales.

Para los dos nodos de cómputo se procede de la siguiente manera:

- Acceder a la BIOS y cambiar el orden de preferencias de arranque, colocando el arranque por red (PXE) como primera opción (comentaremos el entorno PXE en el paso 4.2: Configurar PXE).
- Dentro de la BIOS, conseguir las direcciones MAC de las interfaces de red de los nodos de cómputo, ya que servirán para configuraciones posteriores del escenario.
- Apagar los nodos de cómputo para empezar la instalación del controlador

Para el nodo controlador, instalar la versión de Ubuntu 18.04.1 a través del *pen drive bootable*, definiendo la estructura de particiones deseada.

Para más detalles sobre la configuración necesaria para este paso, consultar la sección 1 del Anexo A.2.

4.2 - Configurar PXE y servidor web de Apache

PXE (*Preboot eXecution Environment*) es un entorno que permite instalar el sistema operativo en equipos agentes desde un nodo controlador a través de una red, de manera independiente de los dispositivos de almacenamiento de datos disponibles o de sistemas operativos instalados localmente.

En el equipo controlador se configuran los ficheros necesarios para que este nodo sirva de repositorio de una imagen del sistema operativo Ubuntu y de guía de instalación de esta a través de la red mediante menús gráficos simples.

De esta manera se realizará la instalación del sistema operativo en los nodos de cómputo a través de la red (modo de arranque PXE) por medio del controlador.

Para ello, es necesario previamente descargar de los repositorios de Ubuntu una imagen ISO del sistema operativo y montarla en el sistema controlador. Una vez hecho esto, habrá que configurar una serie de archivos de arranque de PXE (menús *tftpboot*) para guiar por medio de interfaces gráficas de usuario la instalación del sistema operativo en los nodos de cómputo.

Para más detalles sobre la configuración necesaria para PXE, consultar la sección 2 del Anexo A.2.

Posteriormente hay que desplegar un servidor web de Apache para que los *scripts* y ficheros de configuración que necesitan los nodos de cómputo para la instalación del sistema operativo a través de la red sean accedidos como recursos web con su interfaz gráfica asociada. Se debe cambiar el puerto 80 de escucha de Apache por un puerto 8080, ya que el puerto 80 será necesario en configuraciones posteriores de Ansible.

En los menús de configuración de PXE se hace referencia a un fichero *ks.cfg* accedido a través del servidor web de Apache. Este fichero se encarga de automatizar el proceso de instalación desde dar formato y crear particiones del disco de los nodos agentes, hasta el idioma y zona horaria del sistema según sea programado. Las configuraciones que no estén especificadas en este fichero se harán manualmente durante el proceso de instalación a través de la interfaz propia del instalador.

Para más detalles sobre la configuración necesaria para el servidor de Apache, consultar la sección 3 del Anexo A.2.

4.3 - Configurar servidor DHCP en el controlador

En el controlador se configura el servicio *dnsmasq*, que es un servidor DHCP + DNS + TFTP con soporte para PXE para redes de área local.

En este paso se utilizan las direcciones MAC de los nodos de cómputo para asignarles dinámicamente direcciones IP desde el arranque PXE.

Se empieza instalando el servicio en el nodo controlador desde la terminal de Linux.

Posteriormente hay que editar su archivo de configuración “*/etc/dnsmasq.conf*” para especificar entre otras cosas el rango de direcciones del pool de asignación de DHCP, la dirección local del servidor DHCP y los *hosts* de cómputo con sus direcciones MAC e IP estática asociada por DHCP.

Una vez configurado *dnsmasq*, se debe activar su servicio haciendo uso de los comandos de *systemctl*.

Para más detalles sobre la configuración necesaria para este paso, consultar la sección 4 del Anexo A.2.

4.4 - Configurar iptables persistentes y forwarding

En el controlador es necesario crear *iptables* y activar el *bit* de *forwarding* para permitir que funcione como puerta de enlace predeterminada (*default gateway*) de los nodos de cómputo y realice al NAT para tráfico hacia el exterior.

Para habilitar y activar el *bit* de *forwarding* de forma permanente y que el controlador funcione de puerta de enlace a la red exterior para los nodos de cómputo, se debe editar el fichero “/etc/systemctl.conf” y especificar la opción “net.ipv4.ip_forward=1” y posteriormente reiniciar el sistema.

Para la configuración de NAT utilizaremos reglas de iptables persistentes (seguir la configuración de reglas indicadas en la sección 5.3 del Anexo A.2).

Para más detalles sobre la configuración necesaria para este paso, consultar la sección 5 del Anexo A.2.

4.5 - Instalar Ubuntu en nodos de cómputo vía PXE

Habiendo elegido previamente el modo de arranque por red (PXE) en el paso 4.1, encender los nodos de cómputo y comprobar que el proceso de asignación automática de direcciones IP por DHCP se realiza correctamente.

Cuando la IP sea asignada con éxito, aparecerá el menú de instalación del sistema operativo. Seleccionar “Install Ubuntu Server 18.04.1” para proceder con la instalación mediante el entorno PXE previamente configurado.

Dependiendo del contenido del fichero *kick-start* (“*ks.cfg*”) de automatización del proceso de instalación PXE, la interfaz gráfica del instalador esperará a que las opciones no automatizadas sean seleccionadas por el usuario para continuar.

Cuando el proceso de instalación termine, acceder con el usuario y contraseña configurados y comprobar que el sistema operativo se haya instalado correctamente en los nodos de cómputo y que efectivamente existe conectividad entre los equipos de la infraestructura distribuida (nodos de cómputo y controlador).

Para más detalles sobre la configuración necesaria para este paso, consultar la sección 6 del Anexo A.2.

4.6 - Configurar SSH

Configurar el servicio de SSH en los equipos para poder gestionar los sistemas de cómputo remotamente desde el nodo controlador. El proceso de configuración de SSH se lo hace tanto para el usuario *reicloud* como para *root*.

Para ello, primero hay que generar la clave en el controlador. A continuación, con el comando “ssh-copy-id”, se exporta la clave del controlador a los nodos de cómputo (las claves externas se copian en el archivo “.ssh/authorized_keys” de los nodos de acceso).

Posteriormente se debe cambiar el archivo de configuración de SSH en los nodos de cómputo (/etc/ssh/ssh_config) para permitir el acceso y la autenticación sin clave.

Finalmente, se deben reiniciar los servicios de SSH en los equipos. Comprobar que efectivamente existe conectividad SSH desde el controlador a los nodos de cómputo.

Si se desea asociar nombres de dominio a las direcciones IP de los nodos de cómputo, se debe especificar el par “dirección_IP - domain_name” en el archivo “/etc/hosts” del nodo controlador.

Para más detalles sobre la configuración necesaria para este paso, consultar la sección 7 del Anexo A.2.

4.7 - Instalar y configurar Kolla-Ansible

Como ya se ha comentado en apartados anteriormente, Kolla-Ansible es la herramienta elegida para realizar el despliegue de los servicios de OpenStack a partir de contenedores Docker, haciendo uso de una serie de *playbooks* de configuración de Ansible.

4.7.1 - Scripts de instalación de dependencias de Kolla-Ansible

Para poder desplegar la infraestructura OpenStack, el primer paso es descargar e instalar paquetes de dependencias para instalar Kolla-Ansible. Para automatizar este proceso de instalación de dependencias, se han desarrollado y adaptado una serie de *scripts*:

- `install1-nodes.sh`: *script* para actualizar e instalar dependencias en los nodos de cómputo (Python, Docker, openvswitch, net-tools y bridge-utils).
- `install2-ansible.sh`: *script* para actualizar e instalar las anteriores dependencias, pero en el nodo controlador y además instalar Ansible.
- `install3-kolla.sh`: *script* para instalar Kolla-Ansible y sus dependencias en el nodo controlador, además de los paquetes de Python para clientes básicos de OpenStack CLI: “python-openstackclient”, “python-glanceclient” y “python-neutronclient”.

4.7.2 – Configuración de Kolla-Ansible: Docker, inventario multinode y globals.yml

Una vez instaladas las dependencias de Kolla-Ansible en los nodos del sistema, realizar la configuración requerida para el despliegue de OpenStack.

El primer paso consiste en realizar la preparación para el despliegue de servicios con Docker. Para ello, configurar una carpeta de servicio de Docker, crear un archivo de configuración de Kolla dentro de la carpeta y especificar la URL que usa Kolla para descargar Docker válida para la versión de Ubuntu 18.04 (ver procedimiento de configuración de la sección 9.1 del Anexo A.2).

Posteriormente hay que configurar parámetros del archivo de configuración del inventario “multinode” generado en el controlador al instalar Kolla-Ansible. El inventario “multinode” permite elegir los roles de los equipos del sistema OpenStack distribuido y definir las credenciales de acceso.

Dentro del fichero de configuración del “multinode”, los roles están identificados mediante etiquetas “[...]”.

El nodo controlador toma los roles de [control], [network], [deployment] y [monitoring], por lo que hay que especificar que “localhost” se encarga de tales funciones.

Dentro de cada rol, se especifican una serie de identificadores de interfaces cuya equivalencia en nuestro escenario es la siguiente:

- `neutron_external`: correspondiente a la interfaz sin IP del controlador que conecta a la red del DIT utilizada para asociar con Kolla los *bridges* de virtualización necesarios para crear escenarios OpenStack.

- **network_interface**: interfaz que asocia al controlador a la red interna de conexión con los nodos de cómputo.
- **api_interface**: interfaz local con acceso a Internet, con IP pública asociada a la red externa del DIT.

Hay que indicar el nombre de las interfaces que se relacionan con cada uno de estos identificadores en cada uno de los roles definidos.

A parte, la etiqueta [external-compute] hace referencia a los nodos de cómputo que forman parte del sistema. En este rol, se especificarán los nodos “compute1” y compute2” e indican los parámetros de conexión y autenticación SSH para establecer conexión remota entre el controlador y los nodos de cómputo y poder realizar las configuraciones y comprobaciones desde los *playbooks* de Ansible. Además, en este rol se indican los nombres de las interfaces de red (**network_interface**) que conecta a los nodos de cómputo con el nodo controlador y orquestador.

Para saber cómo configurar en detalle el archivo “multinode”, acceder a la sección 9.2 del Anexo A.2.

Por último, hay que editar los parámetros del archivo de configuración principal de Kolla-Ansible “/etc/kolla/globals.yml” donde se personalizan ajustes de OpenStack como su versión, la forma de descarga, las interfaces de redes para el despliegue, la distribución de Docker o servicios adicionales. Algunos de los parámetros de configuración que son necesarios configurar en este archivo son los siguientes:

- **kolla_base_distro** podría ser Centos o Ubuntu, ambas funcionan bien.
- **kolla_install_type** puede ser *binary* o *source*, *binary* suele ser más segura.
- **openstack_release** debería ser *pike* o *queens*. Seleccionamos *queens*, última versión de OpenStack estable.
- **network_interface** es la interfaz del nodo de red de acceso a la red de conexión interna con los nodos de cómputo. En este caso, la del controlador. La definición de las interfaces de red para el despliegue ya se ha hecho en el archivo multinode al especificar los roles por nodo, por lo que no hace falta definir las en globals.yml (aunque también es posible hacerlo aquí).
- **enable_haproxy**: haproxy (High Availability Proxy) sirve para configurar un servidor *proxy* que pueda balancear la carga entre varios controladores del cluster desplegado para el sistema OpenStack. Puesto que no tenemos más que un equipo controlador y ningún otro de respaldo dejaremos inactivo este servicio. Si se configura “enable_haproxy” a “no”, no se realizará balanceo de carga. Si se configura a “yes” hará balanceo de carga con la IP que se haya especificado en el campo **kolla_internal_vip_address**.
- **kolla_internal_vip_address** equivale a una dirección IP para la cual se accede a los servicios de OpenStack. Si no se habilita **ha_proxy**, se debe dejar la misma que la interfaz de red de acceso a Internet. Si se habilita haproxy, se debe escribir una IP libre perteneciente al mismo segmento de red que funcionará como IP de balanceo.
- **openvswitch (neutron_plugin_agnet)** es el plugin usado para crear las redes y subredes en OpenStack mediante el servicio Neutron.

Para saber cómo configurar en detalle el archivo “globals.yml”, acceder a la sección 9.3 del Anexo A.2.

4.8 - Desplegar infraestructura OpenStack

Una vez instalado y configurado correctamente el entorno de despliegue Kolla-Ansible se procede a implementar y desplegar el escenario OpenStack distribuido.

Antes de todo hay que comprobar que la conexión remota entre el nodo controlador y los nodos de cómputo es exitosa para poder instalar y configurar las dependencias mediante Ansible. Si el resultado es exitoso, podremos proceder a la instalación y despliegue de OpenStack.

Previo al despliegue, es posible cambiar la contraseña de autenticación de administrador para el acceso a la plataforma OpenStack a partir de la variable “keystone_admin_password” del fichero “/etc/kolla/passwords.yml”.

A continuación, se comenta el proceso o pasos a seguir para el despliegue de OpenStack haciendo uso de los mencionados *playbooks* de configuración de Ansible. Decir que estos pasos son secuenciales, por lo que no se puede pasar a ejecutar el siguiente *playbook* sin haber tenido éxito en la instalación del *playbook* actual. El proceso de despliegue es el siguiente:

- Ejecutar el *playbook* **bootstrap-servers** de arranque para instalar las dependencias de despliegue de Kolla-Ansible mediante el siguiente comando:

```
$ sudo kolla-ansible -i multinode bootstrap-servers
```

- Ejecutar el *playbook* **prechecks** de pre-despliegue para realizar comprobaciones previas a la implementación de OpenStack mediante el siguiente comando:

```
$ sudo kolla-ansible -i multinode prechecks
```

- Ejecutar el *playbook* **deploy** para realizar la implementación y despliegue de OpenStack mediante el siguiente comando:

```
$ sudo kolla-ansible -i multinode deploy
```

- Ejecutar el *playbook* **post-deploy** para generar el archivo “openrc” donde se configuran las credenciales para el usuario administrador de OpenStack.

Una vez instalados estos *playbooks*, OpenStack debería estar activo y funcionando. Los contenedores Docker deberían estar ejecutándose en el controlador y en los nodos de cómputo. Un ejemplo del inventario de los contenedores Docker de servicios OpenStack se puede ver en la Figura 8 de la sección 3.2 - *Componentes clave en la implementación de OpenStack distribuido*.

Comprobar el funcionamiento de la plataforma OpenStack accediendo al *dashboard* a través de un navegador web, ingresando a la dirección IP especificada en **kolla_internal_vip_address** (ver la Figura 8).

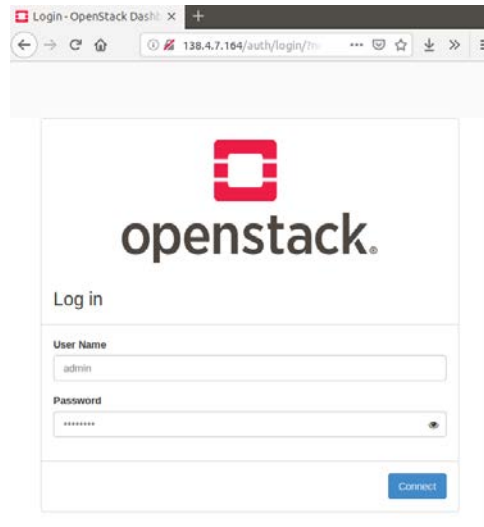


Figura 8: Dashboard de OpenStack.

Para cualquier duda sobre el despliegue final de OpenStack con Kolla-Ansible, acceder a la sección 10 del Anexo A.2. En caso de cualquier error en el despliegue, consultar la sección B - *Troubleshooting* del Anexo A.2 dedicada a tal propósito.

4.9 - Despliegue de red demo en plataforma OpenStack

Una vez realizado el despliegue de la plataforma OpenStack de manera exitosa, se puede proceder a implementar nuestro primer escenario.

Al realizar el “post-deploy” en el paso anterior, se genera un archivo ejecutable “/etc/kolla/admin-openrc.sh”. Este archivo permite exportar variables de entorno para poder utilizar las credenciales de admin y administrar así OpenStack desde la terminal. Para ejecutar el *script* **openrc** ejecutar el siguiente comando desde la terminal:

```
$ . /etc/kolla/admin-openrc.sh
```

A continuación, se debe ejecutar el archivo “**/usr/local/share/kolla-ansible/init-runonce**” que se genera automáticamente al instalar Kolla-Ansible para poder crear un entorno o escenario modelo de operación de OpenStack, definiendo redes y subredes, imágenes de máquinas virtuales, direcciones IP flotantes y otros tantos parámetros con comandos de OpenStack. Sin embargo, debe ser editado previamente para incluir todos los rangos de direcciones IP flotantes disponibles (EXT_NET_RANGE) y las imágenes de imágenes virtuales que se desean cargar para el nuevo escenario de OpenStack. También se debe especificar la IP pública del controlador con conexión a Internet (EXT_NET_CIDR) y la puerta de enlace de dicha red externa (EXT_NET_GATEWAY). Además, editar la sección de *neutron* para que incluya los múltiples rangos de direcciones en la subred que se pretenda crear. Para ejecutar el *script* **init-runonce** ejecutar el siguiente comando desde la terminal:

```
$ . /usr/local/share/kolla-ansible/init-runonce
```

Para cualquier duda sobre el despliegue de un escenario ejemplo en OpenStack con Kolla-Ansible, acceder a la sección 10.1 del Anexo A.2.

En caso de cualquier error en la instalación y despliegue de la plataforma OpenStack, consultar la sección B - *Troubleshooting* del Anexo A.2.

5 - Conclusiones

Con la realización de este proyecto hemos aprendido a configurar y desplegar una infraestructura de OpenStack distribuida entre equipos físicos a través de la utilidad Kolla-Ansible. De esta forma hemos automatizado el proceso de despliegue de un entorno IaaS multinodo.

Para facilitar la tarea a administradores o proveedores sobre el despliegue de escenarios OpenStack distribuidos, hemos diseñado una receta o manual de instalación detallado con los diferentes pasos a seguir para su configuración y despliegue completo.

Como valoración personal, el resultado obtenido con el proyecto desarrollado ha sido muy satisfactorio. Hemos mejorado nuestros conocimientos sobre el entorno de computación en la nube OpenStack y aprendido sobre otras herramientas complementarias que intervienen en su configuración y despliegue, como Kolla-Ansible que nos permite automatizar el proceso es escenarios de infraestructura distribuida.

Como trabajo futuro, se podrían mejorar y optimizar ciertos aspectos de configuración del sistema OpenStack distribuido como los siguientes:

- Uno de las principales dudas que tenemos es justificar el uso de la interfaz ueth2 del nodo controlador, la cual no tiene direccionamiento IP y en principio es necesaria para asociar el *bridge* de virtualización “br-ex” de conexión entre la red virtual interna de OpenStack y la infraestructura de red externa. Se podría estudiar y justificar la posibilidad de prescindir de ella.
- Otra de los aspectos a mejorar es estudiar la utilización de *haproxy* (*High Availability Proxy*) disponible en la configuración del fichero “globals.yml” del entono Kolla-Ansible, que en principio sirve para configurar un servidor *proxy* que pueda balancear la carga entre varios nodos controladores del clúster desplegado para el sistema OpenStack. Puesto que solo disponíamos de un equipo controlador y ningún otro de respaldo, no pudimos probar dicha opción.

Decir que la infraestructura de OpenStack distribuida está actualmente disponible como servicio IaaS para su uso y despliegue de escenarios de virtualización sobre proyectos o casos de estudio relacionados con la computación en la nube.

Finalmente, el fundamento sobre el despliegue de esta plataforma IaaS OpenStack podría servir de apoyo y ayuda para otros proyectos futuros relacionados.

6 - Bibliografía

- Documento “PROYECTO REICLOUD: Retos Educativos para la Integración de competencias profesionales de *CLOUD computing* y virtualización en enseñanzas de grado y máster”, Autores: Víctor Quilón Ranera y Juan Carlos Díaz Torres, ETSIT-UPM, 2018-2019.
- Enlaces sobre configuración y despliegue de OpenStack con Kolla-Ansible:
 1. <https://docs.openstack.org/kolla-ansible/latest/user/quickstart.html>
 2. <https://releases.openstack.org/#series-independent-releases>
 3. <https://releases.openstack.org/queens/index.html>
 4. <https://releases.openstack.org/teams/openstacksdk.html>
- Enlaces informativos sobre los *bridges* de virtualización de OpenStack distribuido:
 1. <https://elatov.github.io/2018/01/openstack-ansible-and-kolla-on-ubuntu-1604/>
 2. <http://iesgn.github.io/cloud/cursos/u8/red-privada>
- Enlaces sobre servicios PXE y *dnsmasq*:
 1. [https://wiki.archlinux.org/index.php/PXE_\(Español\)](https://wiki.archlinux.org/index.php/PXE_(Español))
 2. [https://wiki.archlinux.org/index.php/Dnsmasq_\(Español\)](https://wiki.archlinux.org/index.php/Dnsmasq_(Español))
- Enlace con repositorio de imágenes Docker para los servicios de OpenStack desplegados con Kolla-Ansible: <https://hub.docker.com/u/kolla/>
- Enlace con repositorio de imágenes de máquinas virtuales para utilizar en OpenStack: <http://idefix.dit.upm.es/download/cnvr/ostack-images/>
- Enlaces de ayuda en incidencias:
 1. <https://www.omgubuntu.co.uk/2017/09/disable-network-connectivity-checking-ubuntu-17-10>
 2. <https://askubuntu.com/questions/831216/how-can-i-reinstall-grub-to-the-efi-partition>

A - Anexos

En este apartado se incluyen diferentes anexos para detallar el cuaderno de equipo del proyecto, así como un manual o guía detallada sobre cómo realizar la configuración y despliegue de la plataforma OpenStack Distribuida.

A.1 - Cuaderno de Equipo

Anexo con la planificación inicial del proyecto y el diario de trabajo semanal dedicado.

- Diagrama de planificación inicial del proyecto (ver la Figura 9):

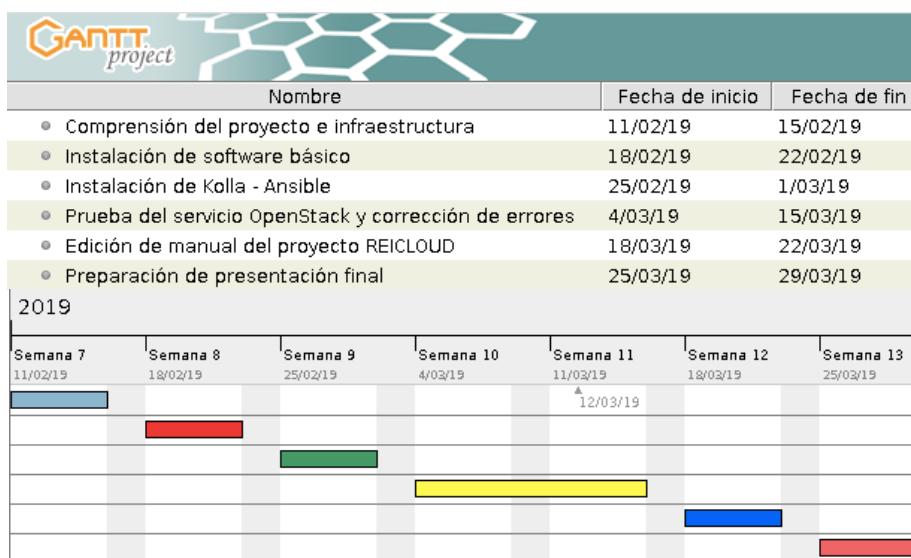


Figura 9: Diagrama de planificación inicial.

- Diario semanal final del proyecto:

Semana 1		Fecha: 11/02/2019 - 15/02/2019
Día 11/02/19	Resumen	<p>Previamente, el día 8/02/19, tuvimos una reunión inicial con los tutores para entender la idea general del Mini Proyecto: "Implementación de un IaaS sobre hardware de laboratorio con OpenStack". Se comentaron los objetivos y resultados específicas que se persiguen con la realización de este proyecto.</p> <p>Primera toma de contacto con el escenario de OpenStack preliminar del proyecto REICLOUD. Verificamos la infraestructura de red y la configuración existente en los equipos del sistema OpenStack distribuido.</p>
	Duración	4 horas
	Incidencias	---
Día 13/02/19	Resumen	<p>Inicialmente, continuamos revisando la configuración de los servidores de la infraestructura OpenStack existente.</p> <p>Una vez revisado el desarrollo preliminar del proyecto REICLOUD, se nos proporciona un disco SSD de 250 GB para utilizarlo en el controlador y comenzar con la instalación desde cero. Antes de todo, copiaríamos los archivos y <i>scripts</i> del equipo controlador que vimos convenientes, que nos serían útiles y aprovechables para la nueva instalación. En principio, mantendríamos los dos nodos de cómputo con su instalación actual.</p> <p>A partir de un <i>pen drive bootable</i> con la distribución de Linux UBUNTU 18.04.1, realizaríamos la instalación del sistema operativo en el nodo controlador. Una vez terminada</p>

		la instalación, configuraríamos PXE (Preboot eXecution Environment) y DHCP (en modalidad <i>dnsmasq</i> : servidores DHCP + DNS + TFTP con soporte para PXE) para posteriormente poder transferir, arrancar e instalar el sistema operativo en los nodos de cómputo a través de la red y a partir de la imagen ISO del S.O del equipo anfitrión (controlador), de manera independiente de los dispositivos de almacenamiento de datos disponibles (discos duros) o de los sistemas operativos previamente instalados. Configuramos reglas NAT persistentes mediante “iptables” en el controlador y activamos el <i>bit</i> de <i>forwarding</i> . El siguiente paso sería arrancar los nodos de cómputo en modo PXE para instalar el sistema operativo por red, comprobando que la conexión y configuración realizada no produce errores.
	Duración	4 horas
	Incidencias	<ol style="list-style-type: none"> 1. Cambiar nombre de las interfaces USB Ethernet (desde fichero “etc/udev/rules.d/70-persistent-net.rules”). 2. Determinar DNS de la red del departamento. 3. Problemas de pérdidas de conexión continuas a la red del DIT. 4. Modificar el archivo <i>kickstart</i> de PXE-TFTPBOOT (“ks.cfg”) para gestionar el disco de almacenamiento y el esquema de partición de los nodos de cómputo manualmente al realizar la instalación del S.O por red.
Día 13/02/19	Resumen	Procedemos con la nueva instalación de los sistemas operativos en los nodos de cómputo. Se nos proporcionan dos nuevos discos SSD de 250 GB para tal propósito. Accederíamos a BIOS de los equipos para obtener su MAC (necesaria para asignarles la IP de la red privada automáticamente desde DHCP en el controlador) y cambiaríamos el modo de arranque inicial a arranque por red PXE.
	Duración	5 horas
	Incidencias	En el archivo <i>kickstart</i> “ks.cfg” se especificaba la gestión automática de la elección del disco de almacenamiento y el esquema de partición. Cambiamos su configuración para poder elegir el nuevo disco de almacenamiento SSD y especificar manualmente la estructura de partición del disco en cada nodo de cómputo.

Semana 2		Fecha: 18/02/2019 - 22/02/2019
Día 18/02/19	Resumen	<p>Procedemos a configurar SSH en el sistema distribuido para poder gestionar los nodos de cómputo a través del nodo controlador. Para ello, primero generaríamos la clave en el controlador (mediante el comando “ssh-keygen”). Posteriormente cambiaríamos el archivo de configuración de SSH en los nodos de cómputo (/etc/ssh/ssh_config) para permitir la autenticación mediante usuario “reicloud” o “root”. Finalmente, con el comando “ssh-copy-id”, exportaríamos la clave del controlador a los nodos de cómputo (las claves externas se copian en el archivo “.ssh/authorized_keys” de los nodos de acceso).</p> <p>Una vez configurado SSH, procederíamos a instalar los paquetes de dependencias para instalar Kolla-Ansible. Reutilizaríamos y ajustaríamos <i>scripts</i> de configuración del disco duro anterior del controlador que fueron utilizados en la instalación previa del sistema OpenStack. Adaptaríamos estos <i>scripts</i>, que sirven para lo siguiente:</p> <ol style="list-style-type: none"> 1. install1-nodes.sh -> <i>script</i> para actualizar e instalar dependencias en los nodos de cómputo (Python, Docker, openvswitch, net-tools, bridge-utils). 2. install2-ansible.sh -> <i>script</i> para actualizar e instalar las anteriores dependencias, pero en el nodo controlador y además instalar Ansible. 3. install3-kolla.sh -> <i>script</i> para instalar Kolla-Ansible y sus dependencias en el nodo controlador, además de los paquetes de Python para servicios básicos de OpenStack CLI.
	Duración	6 horas
	Incidencias	Generar clave de SSH del controlador para el usuario root y exportarla de nuevo a los nodos de cómputo. Arreglar problema de arranque para el disco SSD en el controlador, reinstalando el gestor de arranque GRUB (ver el enlace https://askubuntu.com/questions/831216/how-can-i-reinstall-grub-to-the-efi-partition).
Días 20/02/19	Resumen	Comenzamos con la configuración del entorno Kolla-Ansible para el despliegue de OpenStack. Generamos la carpeta de servicios de <i>Docker</i> y determinamos la “ <i>docker_apt_url</i> ” para Ubuntu 18.04. Entendemos y configuramos parámetros de archivo

y 22/02/19		de configuración <i>inventory</i> “ <i>multinode</i> ” para elegir los roles de los equipos del sistema OpenStack y las credenciales de acceso. Comprendemos y configuramos parámetros del archivo de configuración principal de Kolla-Ansible “ <i>globals.yml</i> ” con ajustes de configuración de la versión de OpenStack, redes o servicios adicionales. Una vez realizada la configuración de estos archivos, procederíamos a pasar a la fase de implementación. Kolla-Ansible proporciona una serie de <i>playbooks</i> para instalar los servicios requeridos para el despliegue final de OpenStack. Comenzaríamos con los <i>playbooks</i> “bootstrap-servers” y “prechecks” para instalar las dependencias de despliegue de Kolla y realizar comprobaciones pertinentes en los nodos de la red.
	Duración	4.5 horas cada día
	Incidencias	Dificultades para entender ciertos parámetros de configuración de <i>globals.yml</i> como el “enable_haproxy”. Problemas y errores en la instalación de los <i>playbooks</i> . Verificar configuración del <i>multinode</i> para evitar proporcionar en claro la clave de usuario para el acceso a los nodos cómputo a la hora de configurar e instalar las dependencias y componentes de despliegue de Kolla-Ansible (evitar introducir el <i>password</i> del usuario “reicloud” con el parámetro “--extra_vars “ansible_sudo_pass=...” “ y acceder como usuario root sin proporcionar clave.

Semana 3		Fecha: 25/02/2019 - 1/03/2019
Días 25/02/19 y 27/02/19	Resumen	Determinamos que el parámetro “enable_haproxy” de <i>globals.yml</i> (<i>High Availability Proxy</i>) sirve para configurar un servidor <i>proxy</i> que pueda balancear la carga entre varios controladores del clúster desplegado para el sistema OpenStack. Puesto que no tenemos más que un equipo controlador y ningún otro de respaldo dejaremos inactivo este servicio. Para ello, se pone enable_haproxy: “no” y en el parámetro de “kolla_internal_vip_address” introducimos la propia IP pública del DIT del equipo controlador. Nos dedicamos a documentarnos sobre las diferentes versiones de OpenStack existentes y comprobar cuál de ellas daría menos problemas de instalación en el despliegue de los <i>playbooks</i> . Probamos con distintas versiones de OpenStack el <i>playbook</i> de despliegue de OpenStack “deploy”.
	Duración	4.5 horas cada día
	Incidencias	Con la definición del haproxy solucionaríamos ciertos problemas en la instalación de los <i>playbooks</i> . En el <i>playbook</i> “deploy” tuvimos ciertos fallos relacionados con la activación del <i>Docker</i> para el servicio de “nova-compute” en los nodos de la red que trataríamos de resolver en las siguientes sesiones. Continuamos con fallos continuos en la conexión a la red del DIT.

Semana 4		Fecha: 4/03/2019 - 8/03/2019
Día 4/03/19	Resumen	Solución al problema de activación del <i>Docker</i> para el servicio de “nova-compute” en el despliegue de OpenStack. Con el comando “systemctl daemon-reload” reiniciamos todos los <i>daemons</i> de servicios de tipo <i>systemd.generator</i> del sistema y se reactivan además los puertos de los <i>sockets</i> asociados. Finalmente se procedería a realizar la parte de arranque de OpenStack. En primer lugar, utilizamos el <i>playbook</i> “post-deploy” para generar el archivo “openrc” donde se configuran las credenciales para el usuario administrador. Se generó el archivo ejecutable “/etc/kolla/admin-openrc.sh” para poder utilizar las credenciales de admin y administrar así OpenStack desde la terminal. Seguidamente, pudimos entrar al <i>dashboard</i> de OpenStack desde el <i>web browser</i> con las credenciales (usuario y contraseñas) especificados por defecto en “/etc/kolla/admin-openrc.sh”. A continuación, procedimos a ejecutar un archivo que se genera al instalar Kolla-Ansible (“/usr/local/share/kolla-ansible/init-runonce”) para poder crear un entorno de operación de OpenStack de ejemplo definiendo redes y subredes, imágenes de VMs, direcciones de IPs flotantes, creación de instancias de VMs y otros tantos parámetros con comandos de OpenStack. Verificamos que se despliegan correctamente las VMs y las redes dentro del sistema OpenStack. Además, solventamos los problemas de pérdidas de conexión que teníamos en la red pública del DIT. El problema venía de que Ubuntu tiene la utilidad “ <i>Network Manager</i> ” habilitada por defecto que permite la verificación de conectividad periódica con servidores de Ubuntu para detectar posibles portales cautivos. Estos <i>pings</i> de fondo provocan en ocasiones problemas de conexión, generando paquetes que usan dos direcciones MAC diferentes para una misma IP. Para desactivar esta opción basta con acceder a <i>Settings > Privacy > network connectivity checking ->off</i> (ver el enlace

		https://www.omgubuntu.co.uk/2017/09/disable-network-connectivity-checking-ubuntu-17-10).
	Duración	6 horas
	Incidencias	<ol style="list-style-type: none"> 1. Tratar de cambiar la contraseña de acceso a OpenStack. 2. Descargar y generar nuevas imágenes de VMs. 3. Incluir nuevos rangos de IPs flotantes disponibles para las VMs de OpenStack.
Día 8/03/19	Resumen	<p>Determinamos que la <i>password</i> de usuario administración de OpenStack se puede modificar en el archivo “/etc/kolla/passwords.yml”. En este archivo hay un parámetro “keystone_admin_password” que permite elegir la <i>password</i> de autenticación de administrador a partir del servicio <i>keystone</i> de OpenStack. Borrando el fichero de credenciales de administración en “/etc/kolla/admin-openrc.sh” y lanzando de nuevo el <i>playbook</i> “post-deploy”, generamos de nuevo las credenciales de admin junto con la nueva <i>password</i>. También comprobamos podemos cambiar la <i>password</i> de admin desde el <i>dashboard Horizon</i> de OpenStack.</p> <p>Además, probamos a volver a realizar el despliegue de OpenStack. Para eliminar la implementación y despliegue de OpenStack mediante Kolla-Ansible utilizamos la opción de <i>playbook</i> “destroy”. Probamos a desplegar el escenario de OpenStack de nuevo para verificar que no ocurren errores con la configuración definida. También tratamos de probar la última versión disponible de OpenStack Rocky, en lugar de la anteriormente utilizada Queens.</p>
	Duración	5 horas
	Incidencias	Problemas al realizar el despliegue con la versión de OpenStack Rocky en la parte de activación del contenedor <i>Docker</i> para el servicio de <i>keystone</i> durante la ejecución del <i>playbook</i> “deploy”. Ante la dificultad de poder desplegar OpenStack con esta versión, nos decantamos por utilizar la versión Queens que parece más estable y no genera tantos problemas.

Semana 5		Fecha: 11/03/2019 - 15/03/2019
Día 15/03/19	Resumen	Se nos proporcionaron diferentes rangos de IPs públicas disponibles que podríamos utilizar como IPs flotantes para las VMs instanciadas en OpenStack. Además, se nos proporcionó el repositorio desde donde descargar las imágenes válidas a utilizar para las instancias de máquinas virtuales de prueba que instalemos en OpenStack. Con estos nuevos datos, tratamos de desarrollar un nuevo <i>script</i> ejemplo con la configuración de entorno de operación de OpenStack similar al de “usr/local/share/kolla-ansible/init-runonce”. En este nuevo <i>script</i> definimos los nuevos rangos de IPs flotantes y las nuevas imágenes (<i>trusty</i> , <i>cirros</i> y <i>xenial</i>) a incluir y utilizar en nuestros escenarios de OpenStack. Además, hicimos pruebas de <i>debug</i> del servicio OpenStack, creando y borrando manualmente desde terminal nuevas imágenes, redes/subredes e instanciando nuevas VMs, comprobando que la plataforma IaaS distribuida funcionaba correctamente. Comprobamos como las máquinas virtuales se iban asociando de forma equilibrada a cada uno de los dos nodos de cómputo.
	Duración	6 horas
	Incidencias	Probar y mejorar el nuevo <i>script</i> ejemplo de despliegue de entorno de operación en OpenStack.

Semana 6		Fecha: 18/03/2019 - 22/03/2019
Día 22/03/19	Resumen	Semana dedicada a la edición del manual o receta del proyecto, especificando claramente y con detalle los pasos seguidos durante su desarrollo. También nos dedicamos a repasar diferentes configuraciones del escenario y verificar que la plataforma de OpenStack seguía teniendo un funcionamiento óptimo.

	Duración	8 horas distribuidas a lo largo de la semana
	Incidencias	---

Semana 7		Fecha: 25/03/2019 - 29/03/2019
Día 29/03/19	Resumen	Semana dedicada a la elaboración de la memoria y la presentación finales del proyecto. También nos dedicamos a repasar diferentes configuraciones del escenario y verificar que la plataforma de OpenStack seguía teniendo un funcionamiento óptimo.
	Duración	8 horas distribuidas a lo largo de la semana
	Incidencias	Ocurrieron fallos de última hora en la infraestructura IaaS relacionados con la instanciación de las máquinas virtuales en el escenario de OpenStack y los <i>bridges</i> de virtualización OVS. Estos problemas provocaban error de asociación de las VMs a la red interna de OpenStack. Se solucionaron reiniciando servicios dependientes y el equipo controlador.

A.2 - Manual de Instalación y Despliegue de OpenStack

Guía detallada sobre cómo realizar la configuración y despliegue de la plataforma OpenStack distribuida mediante Kolla-Ansible.

A.2.1 - Consideraciones iniciales

El escenario donde se realizará despliegue de OpenStack a través de Kolla-Ansible se puede ver en la Ilustración 1.

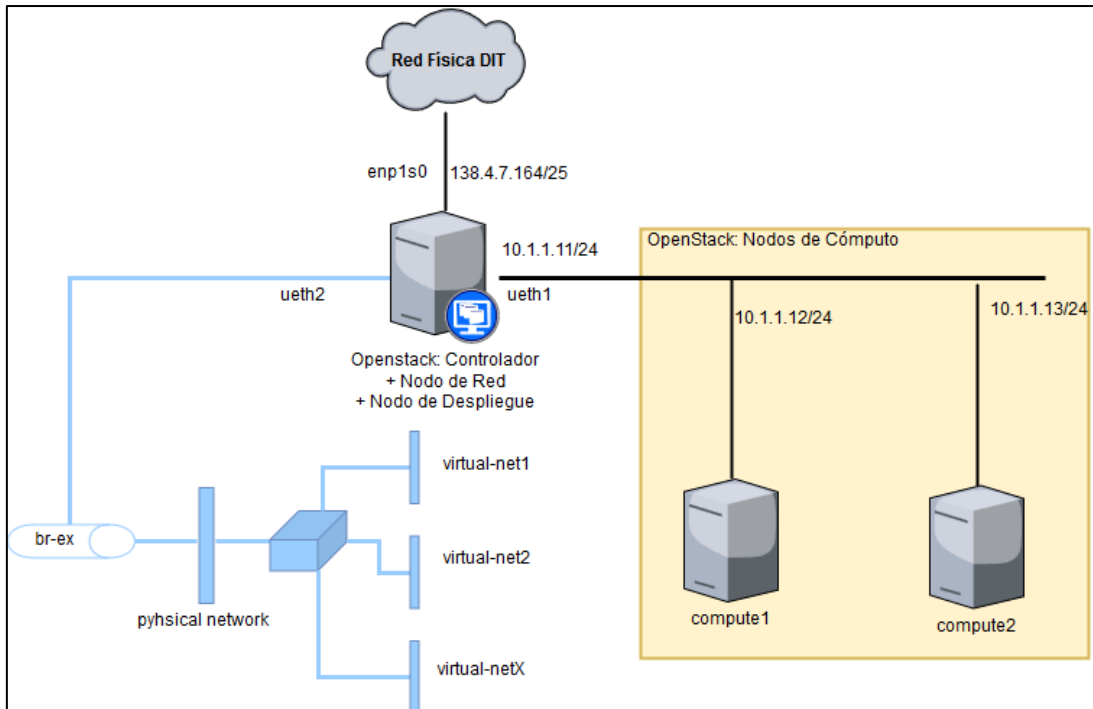


Ilustración 1: Topología del escenario OpenStack.

En primer lugar, se debe tener gestión independiente de todos los equipos involucrados, es decir, se deben conectar todos los ordenadores (controlador y nodos de cómputo) a un monitor. Además, se debe tener las siguientes consideraciones:

En los nodos de cómputo (*compute1* y *compute2*)

- Acceder a la BIOS y cambiar el orden de preferencias de arranque, colocando el arranque por red (PXE) como primera opción.
- Dentro de la BIOS, conseguir las direcciones MAC de las interfaces de red de los nodos de cómputo, ya que servirán para configuraciones posteriores del escenario.
- Apagar los nodos de cómputo para empezar la instalación del controlador
- En el controlador instalar la versión de Ubuntu 18.04.1, a través de un dispositivo *bootable*, configurando la siguiente estructura de particiones (ver la Ilustración 2):
 - 100 MB de **/boot/efi** para el arranque (fat32)
 - 100 GB de **/home** para la carpeta de usuario (ext4)
 - 4GB de swap
 - El resto de espacio para la partición raíz (/) (ext4)

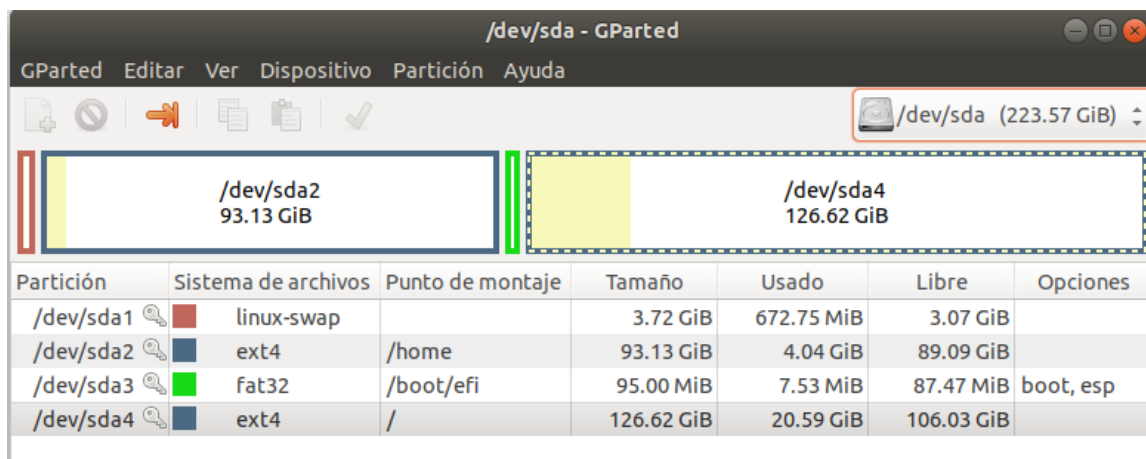


Ilustración 2: Muestra de las particiones de disco en el Controlador luego de la instalación.

- Configurar como nombre de usuario **reicloud** con su contraseña **nub3re6l**.
- Crear una contraseña para usuario **root** con el comando:
\$ sudo passwd
- Crear un directorio llamado **data_reicloud** dentro del directorio de usuario (**/home/reicloud/**) para almacenar imágenes de sistemas operativos, scripts de configuración, etc.

A.2.1.1 - Renombrar Interfaces Ethernet-USB

Cuando el sistema operativo del **controlador** haya sido instalado correctamente, se recomienda cambiar el nombre a las interfaces Ethernet-USB.

Al tener interfaces de red a través de adaptadores Ethernet-USB, los nombres de estas interfaces son aleatorios lo que ocasiona que sean poco legibles y difíciles de recordar. Por lo tanto, se propone otorgar un alias a todas las interfaces USB que tenga el controlador basados en su dirección MAC, para poder utilizar su nombre en configuraciones posteriores.

- Obtener las direcciones MAC de las interfaces Ethernet-USB.

```
$ ifconfig -a
```

Estas interfaces suelen tener nombres como *enx18d6c714f4e8* o *enx18d6c714f67c*.

- Crear o editar el fichero **/etc/udev/rules.d/70-persistent-net.rules** incluyendo las siguientes dos entradas:

```
# USB Ethernet
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="18:d6:c7:14:f4:e8", ATTR{dev_id}=="0x0",
ATTR{type}=="1", NAME="ueth1"

SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="18:d6:c7:14:f6:7c", ATTR{dev_id}=="0x0",
ATTR{type}=="1", NAME="ueth2"
```

En este caso, la interfaz *enx18d6c714f4e8* pasa a ser llamada **ueth1** y la interfaz *enx18d6c714f67c* pasa a ser llamada **ueth2**.

- Reiniciar las interfaces de red (*networking*) y el equipo y comprobar que el cambio de nombre de las interfaces se haya efectuado correctamente mediante los siguientes comandos:

```
$ sudo /etc/init.d/networking restart  
$ sudo reboot  
$ ifconfig -a
```

A.2.1.2 - Duplicado de MAC en paquetes salientes

Puede ocurrir que existan paquetes que usan dos MAC para una misma IP. El centro de monitorización del departamento detecta estos paquetes y la conexión a internet falla aproximadamente cada cinco minutos. Estos paquetes son del servicio *network connectivity* de Ubuntu: <https://www.omgubuntu.co.uk/2017/09/disable-network-connectivity-checking-ubuntu-17-10>

Se debe desactivar esta rutina de conectividad en el **controlador**:

- Acceder a ajustes -> privacidad
- Desactivar la opción “comprobación de conectividad de red”.

A.2.2 - Configurar PXE

PXE es un entorno que permite instalar el sistema operativo en equipos agentes desde un nodo controlador a través de una red, de manera independiente de los dispositivos de almacenamiento de datos disponibles o de los sistemas operativos instalados localmente.

En el **controlador** se configuran los ficheros necesarios para que este nodo sirva de repositorio de una imagen del sistema operativo Ubuntu y de guía de instalación de esta a través de la red mediante menús gráficos simples.

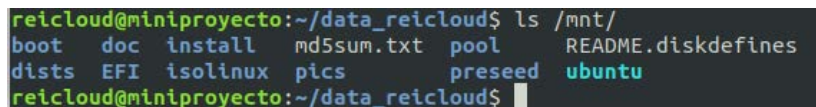
De esta forma, se realizará la instalación del sistema operativo en los nodos de cómputo a través de la red (modo de arranque PXE) por medio del controlador.

A.2.2.1 - Descarga y montaje de UBUNTU-18.04.1-SERVER

Dentro del directorio de datos (*/home/reicloud/data_reicloud*) ejecutar los siguientes comandos

```
$ wget  
http://www.cdimage.ubuntu.com/ubuntu/releases/18.04/release/ubuntu-  
18.04.1-server-amd64.iso  
$ sudo mount -o loop ubuntu-18.04.1-server-amd64.iso /mnt/  
$ ls /mnt/
```

Después de montar la imagen de Ubuntu 18.04 en el directorio */mnt/* deben mostrarse los elementos de la Ilustración 3.



```
reicloud@miniproyecto:~/data_reicloud$ ls /mnt/  
boot  doc  install  md5sum.txt  pool  README.diskdefines  
dists  EFI  isolinux  pics  preseed  ubuntu  
reicloud@miniproyecto:~/data_reicloud$
```

Ilustración 3: Montaje de la imagen de Ubuntu 18.04.

A.2.2.2 Crear directorio tftpboot

```
$ sudo mkdir /var/lib/tftpboot  
$ sudo cp -rf /mnt/install/netboot/* /var/lib/tftpboot/
```

A.2.2.3 - Modificar los menús de carga de tftpbboot

Los menús de tftpbboot son las interfaces gráficas que guían la instalación del sistema operativo en los nodos de cómputo. Estos archivos hacen referencia al uso de un servidor web para acceder a los archivos de configuración necesarios (*ks.conf*, *local-sources.seed*) que se detallarán más adelante. A continuación, se muestra el contenido de los tres ficheros a modificar. Se debe reemplazar su contenido por el detallado en este apartado.

Fichero “txt.cfg”:

```
$ sudo nano /var/lib/tftpbboot/ubuntu-installer/amd64/boot-  
screens/txt.cfg
```

```
default install label local  
menu label Boot from ^Local drive menu default  
localboot 0xffff label install  
menu label ^Install Ubuntu Server 18.04.1  
kernel ubuntu-installer/amd64/linux  
append ks=http://10.1.1.11:8080/ks.cfg vga=788 auto=true  
url=http://10.1.1.11:8080/ubuntu/preseed/local-sources.seed  
initrd=ubuntu-installer/amd64/initrd.gz --- quiet label cli  
menu label ^Command-line install kernel ubuntu-installer/amd64/linux  
append ks=http://10.1.1.11:8080/ks.cfg auto=true  
url=http://10.1.1.11:8080/ubuntu/preseed/local-sources.seed  
tasks=standard pkgsel/language-pack-patterns= pkgsel/install-language-  
support=false vga=788 initrd=ubuntu-installer/amd64/initrd.gz --- quiet
```

Fichero “menu.cfg”:

```
$ sudo nano /var/lib/tftpbboot/ubuntu-installer/amd64/boot-  
screens/menu.cfg
```

```
#menu hshift 13  
#menu width 49  
#menu margin 8  
  
menu title Installer boot menu  
include ubuntu-installer/amd64/boot-screens/stdmenu.cfg  
include ubuntu-installer/amd64/boot-screens/txt.cfg  
include ubuntu-installer/amd64/boot-screens/gtk.cfg  
menu begin advanced  
menu title Advanced options  
include ubuntu-installer/amd64/boot-screens/stdmenu.cfg  
label mainmenu  
menu label ^Back..  
menu exit  
include ubuntu-installer/amd64/boot-screens/adtxt.cfg include ubuntu-  
installer/amd64/boot-screens/adgtk.cfg  
menu end  
label help  
menu label ^Help  
text help  
Display help screens; type 'menu' at boot prompt to return to this  
menu  
endtext  
config ubuntu-installer/amd64/boot-screens/prompt.cfg
```

Fichero “syslinux.cfg”:

```
$ sudo nano /var/lib/tftpboot/ubuntu-installer/amd64/boot-  
screens/syslinux.cfg
```

```
# D-I config version 2.0  
# search path for the c32 support libraries (libcom32, libutil etc.)  
path ubuntu-installer/amd64/boot-screens/  
include ubuntu-installer/amd64/boot-screens/menu.cfg default ubuntu-  
installer/amd64/boot-screens/vesamenu.c32  
prompt 0  
timeout 100
```

A.2.3 -Configurar Servidor Web

En el **controlador** se despliega un servidor web **apache2** para que los scripts de instalación y ficheros de configuración que necesitan los nodos de cómputo para la instalación del sistema operativo a través de la red sean accedidos como recursos web.

A.2.3.1 - Instalar y configurar apache2

```
$ sudo apt install -y apache2
```

Se debe cambiar el puerto por defecto por el cual escucha el servidor web (puerto 80) al 8080, ya que el puerto 80 es necesario para la configuración de Ansible.

```
$ sudo nano /etc/apache2/ports.conf
```

```
Listen 8080
```

```
$ sudo nano /etc/apache2/sites-enabled/000-default.conf
```

```
<VirtualHost *:8080>
```

```
$ sudo service apache2 restart
```

```
$ sudo service apache2 status (ver la Ilustración 4)
```

```
reicloud@mintproyecto:~$ sudo service apache2 status  
● apache2.service - The Apache HTTP Server  
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)  
   Drop-In: /lib/systemd/system/apache2.service.d  
            └─apache2-systemd.conf  
   Active: active (running) since Mon 2019-03-04 12:05:52 CET; 3 weeks 3 days ago  
   Main PID: 1868 (apache2)  
     Tasks: 55 (limit: 4915)  
    CGroup: /system.slice/apache2.service  
            └─1868 /usr/sbin/apache2 -k start  
              9366 /usr/sbin/apache2 -k start  
              9367 /usr/sbin/apache2 -k start
```

Ilustración 4: Estado activo del servicio web apache2.

A.2.3.2 - Configurar ficheros de instalación

A.2.3.2.1 - Copiar la imagen del sistema en el directorio del servidor local

```
$ sudo mkdir /var/www/html/ubuntu
```

```
$ sudo cp -rf /mnt/* /var/www/html/ubuntu/
```

A.2.3.2.2 - Configurar rutina inicial de instalación

En los menús de configuración se hace referencia a un fichero *ks.cfg* accedido a través del servidor web. Este fichero se encarga de automatizar el proceso de instalación desde dar formato y crear particiones del disco, hasta el idioma y zona horaria del sistema según sea programado. Las configuraciones que no estén especificadas en este fichero se harán durante el proceso de instalación a través de la interfaz propia del instalador.

Fichero “ks.cfg”:

```
$ sudo nano /var/www/html/ks.cfg
```

```
#Generic Kickstart template for Ubuntu
#Platform: x86 and x86-64

#System language
lang es_ES

#System keyboard
keyboard es

#System mouse
mouse

#System timezone
timezone Europe/Madrid

#Root password
#rootpw --disabled

#Initial user (user with sudo capabilities)
user reicloud --fullname "Reicloud Node" --password nub3re6l

#Reboot after installation
reboot

# Install OS instead of upgrade
install

#Dpkg repositories
url --url http://es.archive.ubuntu.com/ubuntu/

#Partition clearing information
#clearpart --all --initlabel

#Basic disk partition
#part swap --size 4096
#part / --fstype ext4 --size 1 --grow --asprimary
#part /boot --fstype ext4 --size 256 --asprimary

#Network information
#network --bootproto=dhcp --device=enp3s0 --hostname=computeOne

%pre
route add default gw 10.1.1.11
echo "nameserver 10.1.1.11" >> /etc/resolv.conf
%end
```

Consideraciones importantes:

- Si en los nodos de cómputo se dispone de un disco duro vacío o que contenga información irrelevante y se desea vaciar, se deben descomentar las líneas bajo los apartados de *Partition clearing information* (una línea) y *Basic disk partition* (tres líneas). Estos comandos automatizan el borrado de la información del disco (en caso de que hubiese) y creará las particiones necesarias.
- Si en los nodos de cómputo se dispone de múltiples discos, o se desea hacer la instalación en una partición del disco existente sin borrar su información, los comandos mencionados anteriormente en los dos apartados deben permanecer comentados. Por lo tanto, la creación de particiones se la hará a través del gestor de instalación de Ubuntu.

A.2.3.2.3 - Añadir fuente de la imagen en el servidor

Otro archivo al que se hace referencia es a *local-sources.seed* el cual contiene la ruta al instalador y a los paquetes a instalar en los nodos de cómputo.

```
$ sudo nano /var/www/html/ubuntu/preseed/local-sources.seed
```

```
d-i live-installer/net-image string  
http://10.1.1.11:8080/ubuntu/install/filesystem.squashfs  
# Individual additional packages to install  
d-i pkgsel/include string openssh-server build-essential
```

A.2.4 - Configurar DHCP

En el **controlador** se configura el servicio *dnsmasq* que es un servidor DHCP + DNS + TFTP con soporte para PXE para redes de área local. El archivo por defecto es muy extenso como para editarlo, por lo tanto, se reemplaza por uno ajustado a nuestro escenario. En el apartado A.2.4.1 se crea una copia del estado actual del archivo de configuración antes de reemplazarlo.

En este paso usaremos las direcciones MAC que se obtuvieron en el apartado A.2.1, para asignarles automáticamente las direcciones IP a los nodos de cómputo.

A.2.4.1 - Instalar y configurar *dnsmasq*

```
$ sudo apt install dnsmasq
```

```
$ sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.backup
```

```
$ sudo nano /etc/dnsmasq.conf
```

```
interface=ueth1  
bind-interfaces  
domain=reicloud.upm.es  
  
dhcp-range=eth0,10.1.1.12,10.1.1.20,255.255.255.0,1h  
dhcp-option=3,10.1.1.11  
dhcp-option=6,10.1.1.11  
dhcp-option=6,8.8.8.8  
server=10.1.1.11  
dhcp-option=28,10.0.0.255  
dhcp-option=42,0.0.0.0  
  
dhcp-boot=/var/lib/tftpboot/pxelinux.0  
  
pxe-prompt="Press F8 for menu.",1
```



```
pxe-service=x86PC, "Install Ubuntu 18.04 from network server 10.1.1.11",  
pxelinux  
  
enable-tftp  
tftp-root=/var/lib/tftpboot  
  
dhcp-host=f4:39:09:00:c4:c4,compute1,10.1.1.12,infinite  
dhcp-host=f4:39:09:00:c4:a5,compute2,10.1.1.13,infinite  
  
dhcp-host=00:1e:4f:93:48:93,network,10.1.1.14,infinite
```

A.2.4.2 - Comprobar que el servicio está activo

```
$ service dnsmasq restart
```

```
$ service dnsmasq status (ver la Ilustración 5)
```

```
retcloud@miniproyecto:~$ sudo service dnsmasq status  
● dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server  
   Loaded: loaded (/lib/systemd/system/dnsmasq.service; enabled; vendor preset: enabled)  
   Active: active (running) since Mon 2019-03-04 12:07:15 CET; 3 weeks 3 days ago  
 Main PID: 6445 (dnsmasq)  
    Tasks: 1 (limit: 4915)  
   CGroup: /system.slice/dnsmasq.service  
           └─6445 /usr/sbin/dnsmasq -x /run/dnsmasq/dnsmasq.pid -u dnsmasq -7 /etc/dnsmasq.d,.dpkg-dist,.c
```

Ilustración 5: Estado del servicio de DHCP dnsmasq.

A.2.5 - Configurar iptables persistentes y forwarding

En el **controlador** es necesario crear *iptables* y activar el *forwarding* para permitir que funcione como puerta de enlace predeterminada (*default Gateway*) de los nodos de cómputo y realice al NAT para tráfico hacia el exterior.

A.2.5.1 - Abrir puertos firewall en el servidor

```
$ sudo ufw allow http  
$ sudo ufw allow 8080  
$ sudo ufw allow 53/tcp  
$ sudo ufw allow 53/udp  
$ sudo ufw allow 67/udp  
$ sudo ufw allow 69/udp  
$ sudo ufw allow 4011/udp  
$ sudo ufw enable  
$ sudo service ufw restart  
$ sudo netstat -tulpn
```

Comprobar el estado del servicio (ver la Ilustración 6):

```
retcloud@miniproyecto:~$ sudo ufw status  
Estado: activo  
  
Hasta      Acción     Desde  
-----  
80         ALLOW      Anywhere  
8080       ALLOW      Anywhere  
53/tcp     ALLOW      Anywhere  
53/udp     ALLOW      Anywhere  
67/udp     ALLOW      Anywhere  
69/udp     ALLOW      Anywhere  
4011/udp   ALLOW      Anywhere  
80/tcp     ALLOW      Anywhere  
80 (v6)    ALLOW      Anywhere (v6)  
8080 (v6)  ALLOW      Anywhere (v6)  
53/tcp (v6) ALLOW      Anywhere (v6)  
53/udp (v6) ALLOW      Anywhere (v6)  
67/udp (v6) ALLOW      Anywhere (v6)  
69/udp (v6) ALLOW      Anywhere (v6)  
4011/udp (v6) ALLOW      Anywhere (v6)  
80/tcp (v6) ALLOW      Anywhere (v6)
```

Ilustración 6: Estado del servicio de firewall ufw.

A.2.5.2 - En el servidor host se añade el bit de forwarding

Habilitamos el *bit* de *forwarding* permanentemente:

```
$ sudo nano /etc/sysctl.conf
```

```
net.ipv4.ip_forward=1  
net.ipv4.conf.default.forwarding=1  
net.ipv4.conf.all.forwarding=1
```

A.2.5.3 - Configurar reglas NAT de forma persistente

Borrar las reglas existentes para evitar errores:

```
$ sudo iptables -F INPUT ACCEPT  
$ sudo iptables -F FORWARD ACCEPT  
$ sudo iptables -F OUTPUT ACCEPT  
$ sudo iptables -t nat -F  
$ sudo iptables -t mangle -F  
$ sudo iptables -F  
$ sudo iptables -X
```

Crear las nuevas reglas reemplazando las siguientes variables de ser necesario:

- enp1s0: <Interfaz Internet>
- ueth1: <Interfaz switch local>

```
$ sudo iptables -A FORWARD -o enp1s0 -i ueth1 -s 10.1.1.0/24 -j ACCEPT  
$ sudo iptables -A FORWARD -j ACCEPT  
$ sudo iptables -t nat -F POSTROUTING  
$ sudo iptables -t nat -A POSTROUTING -o enp1s0 -j MASQUERADE  
  
$ sudo apt-get install -y iptables-persistent
```

(Seleccionar sí en guardar las reglas actuales)

```
$ sudo iptables-save | sudo tee /etc/iptables.sav  
$ sudo iptables-restore < /etc/iptables.sav  
$ sudo iptables-save > /etc/iptables/rules.v4
```

Activar forwarding con el siguiente comando:

```
$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

o haciendo un *reboot* una vez realizada la ejecución de los comandos en la sección A.2.5.2.

A.2.6 - Instalar Ubuntu vía PXE

Encender los **nodos de cómputo** y comprobar que el proceso de petición y respuesta DHCP es realizado correctamente. Si falla la asignación de IP, acceder a la sección A.2.1.

Cuando la IP sea asignada con éxito, aparecerá el menú de instalación. Seleccionar “Install Ubuntu Server 18.04.1” para proceder con la instalación.

Dependiendo del contenido del fichero **ks.cfg** el instalador esperará a que las opciones no automatizadas sean seleccionadas para continuar con la instalación, o simplemente procederá a continuar con la instalación.

Si durante el proceso de instalación, existen errores con mensajes tipo “This file may be corrupt” revisar errores de caracteres (espacios, saltos de línea, comillas) en los ficheros **local-sources.seed** y en los ficheros del apartado A.2.3.

Cuando el proceso de instalación termine, acceder con el usuario y contraseña ya conocido y comprobar conectividad entre los nodos de cómputo y el controlador.

A.2.7 - Configurar SSH

En el controlador configurar ssh ejecutando el *script* “install0-ssh-controller.sh”.

```
$ sudo ./install0-ssh-controller.sh
```

Fichero “install0-ssh-controller.sh”:

```
#Generar las claves
ssh-keygen -b 4096
ssh-copy-id reicloud@10.1.1.12
ssh-copy-id reicloud@10.1.1.13
sudo su
ssh-keygen -b 4096
ssh-copy-id reicloud@10.1.1.12
ssh-copy-id reicloud@10.1.1.13
```

En los **nodos de cómputo**, editar el fichero de configuración de ssh para permitir acceso sin contraseñas.

```
$ sudo nano /etc/ssh/sshd_config
```

```
PermitRootLogin without-password
PasswordAuthentication no
AllowUsers reicloud root
```

```
$ sudo service sshd restart
```

```
$ sudo service ssh restart
```

A.2.7.1 - Registrar los nombres de los Hosts

```
$ sudo nano /etc/hosts
```

```
10.1.1.12    computel
10.1.1.12    compute2
```

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

Comprobar que la resolución de nombres funcione con una prueba de ping.

```
$ ping computel
```

```
$ ping compute2
```

Si la resolución de nombres falla, reiniciar el controlador para que los cambios entren en vigor.

A.2.7.2 - Probar la conexión SSH para root y reicloud

```
$ ssh computel
```

```
$ sudo ssh compute2
```

A.2.8 - Instalar Kolla-Ansible

A.2.8.1 - Instalación en los Nodos de Cómputo

En cada uno de los **nodos de cómputo** ejecutar el script **install1-nodes.sh**. Pasar el fichero desde el **controlador** hacia los nodos de cómputo con el comando “**scp**”.

```
$ scp ./install1-nodes.sh reicloud@computel:/home/reicloud/
```

```
$ scp ./install1-nodes.sh reicloud@compute2:/home/reicloud/
```

Desde cada nodo ejecutar el script.

```
$ sudo ./install1-nodes.sh
```

Fichero install1-nodes.sh:

```
#!/bin/bash
#NODE INSTALLER
echo "<====Installing Updates and Upgrades====>"
sudo apt-get update
sudo apt-get upgrade -y
sudo apt install -y openvswitch-switch
sudo apt install -y bridge-utils
sudo apt install -y net-tools
echo "<====Installing Python-Pip-Easy_Install =====>"
sudo apt-get install -y python-pip
sudo pip install -U pip
sudo apt-get install -y python-dev build-essential libffi-dev gcc
libssl-dev python-selinux
sudo wget
https://files.pythonhosted.org/packages/e7/16/da8cb8046149d50940c61
10310983abb359bbb8cbc3539e6bef95c29428a/setuptools-40.6.2-py2.py3-
none-any.whl
sudo pip install setuptools-40.6.2-py2.py3-none-any.whl
echo "<=====>"
echo "<===== Installing Keys =====>"
sudo apt-get install -y curl
sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --
recv-keys 58118E89F3A912897C070ADBF76221572C52609D
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
echo "<===== Finish Installation=====>"
```

Para comprobar la instalación, acceder a la sección B.2 del *Troubleshooting*.

A.2.8.2 - Instalación en el Controlador

En el controlador ejecutar el *script* “install2-ansible.sh”.

```
$ sudo ./install2-ansible.sh
```

Fichero “install2-ansible.sh”:

```
#!/bin/bash
echo "<===== Installing Updates and Upgrades =====>"
sudo apt-get update
sudo apt-get upgrade -y
sudo apt install -y openvswitch-switch
sudo apt install -y bridge-utils
sudo apt install -y net-tools
echo "<=====Installing Python-Pip-Easy_Install =====>"
sudo apt-get install -y python-pip
sudo pip install -U pip
sudo apt-get install -y python-dev build-essential libffi-dev gcc
libssl-dev python-selinux
echo "<===== Installing Ansible =====>"
sudo apt-get install -y ansible
sudo pip install -U ansible
echo "<=====Installing Updates and Upgrades OK=====>"
echo "<=====Please, follow the Next Steps=====>"
echo "Edit the file /etc/ansible/ansible.cfg"
echo "Add the following lines after de tag [defaults]"
printf '%s\n%s\n%s\n%s\n' '[defaults]' 'host_key_checking=False'
'pipelining=True' 'forks=100'
```

Editar el fichero ansible.cfg que se encuentra dentro del directorio “/etc/ansible” y añadir en [defaults] las siguientes configuraciones.

```
$ nano /etc/ansible/ansible.cfg
```

```
[defaults]
host_key_checking=False
pipelining=True
forks=100
```

Continuar con la ejecución del script “install3-kolla.sh”.

```
$ sudo ./install3-kolla.sh
```

Fichero “install3-kolla.sh”:

```
#!/bin/bash
echo "<===== Installing Kolla-Ansible-6.0.0 =====>"
sudo pip install 'kolla-ansible==6.0.0'
echo "<===== Copying Some Files =====>"
sudo cp -r /usr/local/share/kolla-ansible/etc_examples/kolla /etc/
sudo cp /usr/local/share/kolla-ansible/ansible/inventory/* .
echo "<===== Installing Keys =====>"
sudo apt-get install -y curl
sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --
recv-keys 58118E89F3A912897C070ADBF76221572C52609D
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
echo "<=====Generating Kolla-Passwords =====>"
sudo kolla-genpwd
echo "<===== Installing Python-Easy_Install =====>"
sudo wget
https://files.pythonhosted.org/packages/e7/16/da8cb8046149d50940c61
```

```
10310983abb359bbb8cbc3539e6bef95c29428a/setuptools-40.6.2-py2.py3-
none-any.whl
sudo pip install -y setuptools-40.6.2-py2.py3-none-any.whl
echo "<===== Installing Python OpenStack-Client =====>"
sudo pip install python-openstackclient python-glanceclient python-
neutronclient
echo "<=====Install Kolla-Ansible-6.0.0 OK=====>"
```

Para comprobar que la instalación se ha hecho correctamente, acceder a la sección de *Troubleshooting B.2*

A.2.9 - Configuración del entorno Kolla

A.2.9.1 Preparación para despliegue de Docker

Crear la carpeta del servicio de Docker

```
$ sudo mkdir -p /etc/systemd/system/docker.service.d
```

Crear el archivo de configuración de kolla para el servicio de Docker. Añadir EOF a la configuración,

```
$ tee /etc/systemd/system/docker.service.d/kolla.conf <<-'EOF'
```

```
[Service]
MountFlags=shared
EOF
```

Cambiar un parámetro en el fichero `precheck.yml` tal como se resalta en negro a continuación:

```
$ sudo nano /usr/local/share/kolla-
ansible/ansible/roles/neutron/tasks/precheck.yml
```

```
- name: Checking if 'MountFlags' for docker service is set to 'shared'
command: systemctl show docker
register: result
changed_when: false
failed_when: result.stdout.find('MountFlags=shared') == -1 when:
- (inventory_hostname in groups['neutron-dhcp-agent'])
or inventory_hostname in groups['neutron-l3-agent']
or inventory_hostname in groups['neutron-metadata-agent'])
- ansible_os_family == 'RedHat' or ansible_distribution == 'Ubuntu'
```

Cambiar la URL que usa kolla para descargar Docker en el fichero: `/usr/local/share/kolla-ansible/ansible/roles/baremetal/defaults/main.yml`.

```
$ sudo nano /usr/local/share/kolla-
ansible/ansible/roles/baremetal/defaults/main.yml
```

Este fichero tiene por defecto como **`docker_apt_url`**: <https://apt.dockerproject.org>, válido para Ubuntu 16, pero no para Ubuntu 18.04, por lo tanto, hay que cambiarlo a:

```
docker_apt_url:  "{{ 'http://obs.linaro.org/ERP:/17.12/Debian_9' if
ansible_architecture == 'aarch64' else
'https://download.docker.com/linux/ubuntu' }}"
docker_apt_key_file:  "{{ 'Release.key' if ansible_architecture ==
'aarch64' else 'gpg' }}"
docker_apt_key_id:  "{{ 'C32DA102AD89C2BE' if ansible_architecture ==
'aarch64' else 'F76221572C52609D' }}"
```

Además, editar el fichero **docker_apt_repo.j2** localizado en el directorio **/usr/local/share/kolla-ansible/ansible/roles/baremetal/templates/** para que tenga la versión **stable** en lugar de **main** ya que se trata de Ubuntu 18.04, como se muestra a continuación:

```
{% if ansible_architecture == 'aarch64' %}
deb {{ docker_apt_url }} ./
{% else %}
deb {{ docker_apt_url }} {{ ansible_distribution_release | lower }} stable
{% endif %}
```

Además de este cambio, en el apartado "debian_pkg_install" hay que cambiar la línea:

```
- "{{ {'docker-ce' if ansible_architecture == 'aarch64' else 'docker-
engine=1.12.*' }}"
```

por:

```
- docker-ce.
```

A.2.9.2 - Modificar el inventario multinode

El inventario "**multinode**" es el fichero donde se eligen los roles de nuestro OpenStack, es decir, el rol asociado a cada nodo.

La única parte que se debe modificar es la primera ya que en nuestro escenario el nodo **controlador** cumple con multifunciones, mientras que los **nodos de cómputo** cumplen únicamente el rol de *compute*:

Controlador + red + despliegue: Este nodo despliega los ficheros Ansible. Además, funciona como controlador y nodo de red.

Compute 1 y Compute 2: Nodos de computación donde se ejecutarán los contenedores Docker de *nova* y los necesarios para conectarse con el controlador.

A.2.9.2.1 - Código de configuración

Como el controlador es quien despliega los ficheros ansible, la etiqueta [control] equivale a localhost (el mismo controlador). Las interfaces Ethernet-USB están asociadas a **neutron_external** (donde kolla pondrá el br-ex) y a **network_interface** que es la interfaz que pertenece a la red interna con los nodos de cómputo. Luego tenemos la interfaz **api_interface** que se asocia con la interfaz local hacia la red externa del DIT.

```
[control]
localhost    ansible_connection=local
localhost    neutron_external_interface=ueth2
network_interface=ueth1    api_interface=enp1s0
```

El controlador cumple también con funciones de nodo de red, por lo tanto, la etiqueta [network] también la asociamos con localhost.

```
[network]
localhost    ansible_connection=local
localhost    neutron_external_interface=ueth2
network_interface=ueth1    api_interface=enp1s0
```

La etiqueta [external-compute] hace referencia a los nodos de cómputo que forman parte del sistema. En este caso son los ordenadores compute1 y compute2 (comprobar que los hosts estén registrados en "/etc/hosts" tal como se indicó en el apartado A.2.7.1). Además, indicar los parámetros SSH para establecer conexión remota entre el controlador y los nodos de cómputo.

```
[external-compute]
compute1 ansible_connection=ssh ansible_user=root ansible_become=true
        ansible_ssh_private_key_file=/root/.ssh/id_rsa
compute2 ansible_connection=ssh ansible_user=root ansible_become=true
        ansible_ssh_private_key_file=/root/.ssh/id_rsa

compute1 network_interface=enp2s0
compute2 network_interface=enp2s0
[compute:children]
inner-compute
external-compute
```

El servicio de monitorización también estará en el controlador.

```
[monitoring]
localhost ansible_connection=local
localhost neutron_external_interface=ueth2
network_interface=ueth1 api_interface=enp1s0
```

La etiqueta [deployment] indica qué ordenador va a desplegar los ficheros ansible (en este caso el controlador)

```
[deployment]
localhost ansible_connection=local
localhost neutron_external_interface=ueth2
network_interface=ueth1 api_interface=enp1s0
```

Comentar el nodo de almacenamiento

```
[storage]
#storage01
```

A.2.9.3 - Modificar el fichero globals.yml

El fichero "**globals.yml**" es donde se personalizan ajustes de OpenStack como su versión, la forma de descarga, la distribución de los Docker que usa, o qué interfaces de red forman parte del despliegue.

A.2.9.2.1 - Código de configuración

Editar el fichero incluyendo las siguientes configuraciones en caso de no existir:

```
$ sudo nano /etc/kolla/globals.yml
```

```
kolla_base_distro: "centos"
kolla_install_type: "binary"
openstack_release: "queens"
kolla_internal_vip_address: "138.4.7.164"
neutron_plugin_agent: "openvswitch"
enable_haproxy: "no"
enable_openvswitch: "{{ neutron_plugin_agent != 'linuxbridge' }}"
glance_enable_rolling_upgrade: "no"
```

- **kolla_base_distro** podría ser Centos o Ubuntu, ambas funcionan bien.
- **kolla_install_type** puede ser *binary* o *source*, *binary* suele ser más segura.
- **openstack_release** debería ser *pike* o *queens*. Seleccionamos *queens*.
- **network interface** es la interfaz de red del nodo de red, en nuestro caso, la del

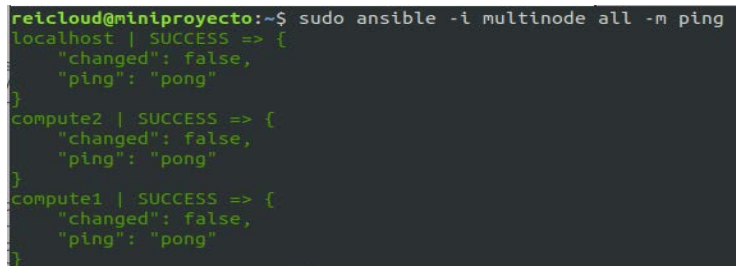
controlador.

- **kolla_internal_vip_address** equivale a una dirección IP para la cual se accede a los servicios. Si no se habilita **ha_proxy**, se debe dejar la misma que la interfaz de red **enp1s0**. Si se habilita **ha_proxy**, se debe escribir una IP libre perteneciente al mismo segmento de red.
- **openvswitch** es el plugin usado para neutron.
- **enable_haproxy**: si se configura a “no”, no se realizará balanceo de carga. Si se configura a “yes” hará balanceo de carga con la IP que se haya especificado en el campo **kolla_internal_vip_address**.

A.2.10 - Despliegue de OpenStack

En el **controlador** y desde el directorio donde se tenga el fichero **multinode**, comprobar la conexión con los nodos de cómputo ejecutando el siguiente comando (ver la Ilustración 7):

```
$ sudo ansible -i multinode all -m ping
```



```
reicloud@mintproyecto:~$ sudo ansible -i multinode all -m ping
localhost | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
compute2 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
compute1 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

Ilustración 7: Comprobación de conexión entre controlador y nodos de cómputo.

Si el resultado de ping es exitoso, proceder al despliegue de OpenStack. En caso de que el ping no sea exitoso, revisar a la sección B.3 del *Troubleshooting*.

Antes del despliegue, se puede cambiar la contraseña que usará la plataforma de OpenStack. Para ello se debe editar el fichero **/etc/kolla/passwords.yml** y reemplazar el valor de la variable **keystone_admin_password** por una contraseña nueva. Ejemplo:

```
$ sudo nano /etc/kolla/globals.yml
```

```
keystone_admin_password: nub3re6l
```

A continuación, se menciona el proceso de despliegue de OpenStack. Considerar que estos pasos son secuenciales y no se puede pasar al siguiente sin haber tenido éxito en el paso previo.

- Ejecutar el Bootstrap-server:

```
$ sudo kolla-ansible -i multinode bootstrap-servers
```
- Ejecutar el Prechecks:

```
$ sudo kolla-ansible -i multinode prechecks
```
- Ejecutar el Despliegue:

```
$ sudo kolla-ansible -i multinode deploy
```
- Ejecutar el Post Despliegue:

```
$ sudo kolla-ansible post-deploy
```

Hasta este punto, la plataforma OpenStack debería estar funcionando sin inconvenientes. Los contenedores Docker deberían estar ejecutándose en el controlador y en los nodos de cómputo.

Se puede listar los contenedores del controlador con “sudo docker ps -a” (ver la Ilustración 8).

```
felci@kolla:~$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
31885d199aa8	kolla/centos-binary-horizon:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		horizon
a2802860eaf1	kolla/centos-binary-heat-engine:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		heat_engine
7e153b7e7fba	kolla/centos-binary-heat-api-cfn:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		heat_api_cfn
87283c5345c	kolla/centos-binary-heat-api:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		heat_api
5386c98fca1f	kolla/centos-binary-neutron-metadata-agent:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		neutron_metadata_agent
74653ade7f49	kolla/centos-binary-neutron-l3-agent:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		neutron_l3_agent
c1660f9c39ca	kolla/centos-binary-neutron-dhcp-agent:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		neutron_dhcp_agent
fc79216de21c	kolla/centos-binary-neutron-openvswitch-agent:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		neutron_openvswitch_agent
b3658839c45e	kolla/centos-binary-neutron-server:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		neutron_server
823e1ce1c476	kolla/centos-binary-openvswitch-vswitchd:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		openvswitch_vswitchd
2d4cd45dca00	kolla/centos-binary-openvswitch-db-server:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		openvswitch_db
63dda995509e	kolla/centos-binary-nova-novncproxy:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		nova_novncproxy
5ea525ed0e0f	kolla/centos-binary-nova-consoleauth:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		nova_consoleauth
9fabeeb149e8	kolla/centos-binary-nova-conductor:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		nova_conductor
664c208ab4b0	kolla/centos-binary-nova-scheduler:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		nova_scheduler
6841c1c1ed21	kolla/centos-binary-nova-api:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		nova_api
6a80a2c19faa	kolla/centos-binary-nova-placement-api:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		placement_api
bc7fc3db37e5	kolla/centos-binary-glance-registry:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		glance_registry
ba8bdf5fd4c	kolla/centos-binary-glance-api:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		glance_api
01e3cf88707a	kolla/centos-binary-keystone:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		keystone
5e031ff86e08	kolla/centos-binary-rabbitmq:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		rabbitmq
44a3f754628f	kolla/centos-binary-narladb:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		narladb
075f8e065576	kolla/centos-binary-cron:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		cron
06c1184c5be1	kolla/centos-binary-kolla-toolbox:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		kolla_toolbox
99aeac7e0d5c	kolla/centos-binary-fluentd:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		fluentd
8cda6ce0b8d	kolla/centos-binary-memcached:queens	"dumb-init --single..."	3 weeks ago	Up 3 weeks		memcached

Ilustración 8: Contenedores desplegados en el nodo controlador.

Además, se comprueba el funcionamiento accediendo al dashboard a través de un navegador web, ingresando a la dirección IP especificada en `kolla_internal_vip_address`. En nuestro caso “http://138.4.7.164” (ver la Ilustración 9).

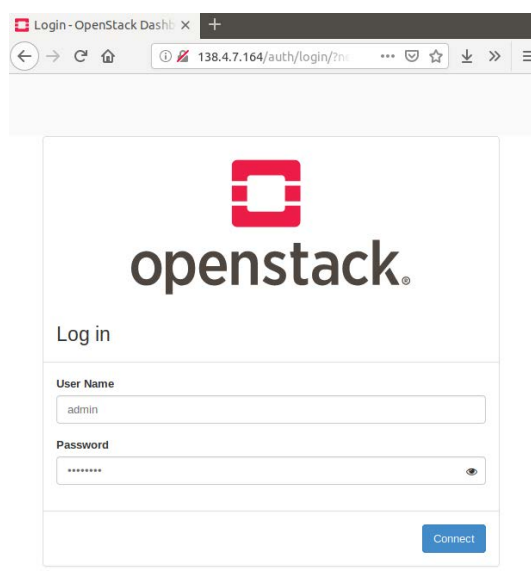


Ilustración 9: Dashboard de OpenStack (servicio Horizon).

En caso de que algún paso de despliegue falle, se puede usar una serie de herramientas detalladas en la sección B.4 del *Troubleshooting* para destruir y volver a desplegar toda la plataforma de OpenStack o para encontrar la causa del fallo.

A.2.10.1 - Despliegue de una Red demo

Exportar las variables de entorno para poder utilizar las credenciales de admin y administrar así OpenStack desde la terminal.

```
$ . /etc/kolla/admin-openrc.sh
```

Si la ejecución falla, se debe otorgar permisos de ejecución al *script*:

```
$ sudo chmod +x /etc/kolla/admin-openrc.sh
```

Se debe ejecutar el archivo “`/usr/local/share/kolla-ansible/init-runonce`” que se genera automáticamente al instalar Kolla-Ansible para poder crear un entorno modelo de operación de OpenStack definiendo redes y subredes, imágenes de máquinas virtuales, direcciones IP flotantes, y otros tantos parámetros con comandos de OpenStack. Sin embargo, se debe editarlo previamente para incluir todos los rangos de direcciones IP flotantes que el DIT ha asignado al proyecto.

Las direcciones IP proporcionadas por el DIT para el *pool* de IP flotantes son las siguientes:

- 138.4.7.216 rc-vm1
- 138.4.7.217 rc-vm2
- 138.4.7.218 rc-vm3
- 138.4.7.139 rc-vm4
- 138.4.7.140 rc-vm5
- 138.4.7.141 rc-vm6
- 138.4.7.221 rc-vm7
- 138.4.7.222 rc-vm8

Abrir el fichero **init-runonce** y editar las secciones referentes a la IP pública del controlador con conexión a la red externa del DIT (EXT_NET_CIDR), el grupo de direcciones IP flotantes disponibles por rangos (EXT_NET_RANGE) y la puerta de enlace predeterminada de dicha red externa (EXT_NET_GATEWAY). Además, editar la sección de *neutron* para que incluya los múltiples rangos de direcciones en la subred que se pretende crear.

```
$ sudo nano /usr/local/share/kolla-ansible/init-runonce
```

```
# This EXT_NET_CIDR is your public network,that you want to connect to
the internet via.
EXT_NET_CIDR='138.4.7.164/25'
EXT_NET_RANGE1='start=138.4.7.216,end=138.4.7.218'
EXT_NET_RANGE2='start=138.4.7.139,end=138.4.7.141'
EXT_NET_RANGE3='start=138.4.7.221,end=138.4.7.222'
EXT_NET_GATEWAY='138.4.7.129'
...

echo Configuring neutron.
openstack network create --external --provider-physical-network physnet1
\
--provider-network-type flat public1
openstack subnet create --no-dhcp \
--allocation-pool ${EXT_NET_RANGE1} --allocation-pool
${EXT_NET_RANGE2} --allocation-pool ${EXT_NET_RANGE3} \
--network public1 \
--subnet-range ${EXT_NET_CIDR} --gateway ${EXT_NET_GATEWAY} ext-
subnet1
```

Agregar permisos de ejecución al script.

```
$ sudo chmod +x /usr/local/share/kolla-ansible/init-runonce
```

Ejecutar el script “init-runonce”:

```
$ . /usr/local/share/kolla-ansible/init-runonce
```

Si se desea cambiar el fichero “init-runonce” por uno personalizado con las imágenes creadas por el departamento, acceder a la sección B.5 del *Troubleshooting*.

B - Troubleshooting

B.1 - Fallo de servicio DHCP

- Comprobar estado del servicio
`$ sudo service dnsmasq status`
- Reiniciar el servicio:
`$ sudo service dnsmasq restart`
- Si el problema continúa y el servicio no arranca, se debe a un error de configuración en el fichero **/etc/dnsmasq.conf**.
- Revisar que el fichero contenga los nombres correctos de las interfaces y las direcciones MAC de los nodos de cómputo escritas correctamente.

B.2 - Comprobación de instalación de Kolla-Ansible

Las versiones de los componentes de OpenStack, Kolla-Ansible, y Python cambian constantemente, generando problemas de compatibilidad. Por lo tanto, se sugiere revisar que las versiones de todos los componentes sean compatibles.

Para conocer la compatibilidad entre las versiones de OpenStack y sus componentes individuales revisar el siguiente enlace:

- <https://releases.openstack.org/#series-independent-releases>
- <https://releases.openstack.org/teams/openstacksdk.html>

Dentro del **controlador** ejecutar los siguientes comandos para comparar las versiones que se tienen con las especificadas en el documento de OpenStack.

```
$ docker --version
Docker version 18.09.0, build 4d60db4

root@controller:/home/reicloud# pip show kolla-ansible
Name: kolla-ansible
Version: 6.0.0
Summary: Ansible Deployment of Kolla containers
Home-page: https://docs.openstack.org/kolla-ansible/latest/
Author: OpenStack
Author-email: openstack-dev@lists.openstack.org
License: Apache License, Version 2.0
Location: /usr/local/lib/python2.7/dist-packages
Requires: oslo.utils, setuptools, oslo.config, PyYAML, pbr, Jinja2, six, docker, cryptography, netaddr
Required-by:

$ python --version
Python 2.7.15rc1

$ pip --version
pip 18.1 from /usr/local/lib/python2.7/dist-packages/pip
(python2.7)
$ pip show pip Name: pip Version: 18.1
Summary: The PyPA recommended tool for installing Python packages.
Home-page: https://pip.pypa.io/
```

```
Author: The pip developers
Author-email: pypa-dev@groups.google.com
License: MIT
Location: /usr/local/lib/python2.7/dist-packages
Requires:
Required-by:

$ pip show docker
Name: docker Version: 3.5.1
Summary: A Python library for the Docker Engine API. Home-page:
https://github.com/docker/docker-py Author: None
Author-email: None
License: Apache License 2.0
Location: /usr/local/lib/python2.7/dist-packages
Requires: backports.ssl-match-hostname, docker-pycreds,
ipaddress,
reuests, six, websocket-client
Required-by: kolla-ansible

$ sudo ansible --version ansible 2.7.1
config file = /etc/ansible/ansible.cfg
configured module search path =
[u'/root/.ansible/plugins/modules', u'/usr/share/ansible/
plugins/modules']
ansible python module location = /usr/local/lib/python2.7/dist-
packages/ansible
executable location = /usr/local/bin/ansible
python version = 2.7.15rc1 (default, Nov 12 2018, 14:31:15) [GCC
7.3.0]
```

B.3 - Fallo de ping Ansible

La causa más común de este fallo es una mala configuración del inventario **multinode** en la etiqueta de [external-compute]

- Comprobar que exista conectividad básica entre en controlador y los nodos de cómputo
- Comprobar que exista conexión SSH entre los dispositivos, tanto con usuario *reicloud* como *root*.
- Revisar los parámetros de la conexión SSH hacia los nodos de cómputo en el fichero **multinode**.

B.4 - Errores en despliegue de contenedores

Para eliminar los contenedores, archivos de configuración de los componentes (nova, neutron, etc), y las imágenes Docker descargadas.

```
$ sudo /usr/local/share/kolla-ansible/tools/cleanup-containers
$ sudo /usr/local/share/kolla-ansible/tools/cleanup-host
```

```
$ sudo /usr/local/share/kolla-ansible/tools/cleanup-images --all
```

Para destruir el escenario y borrar todos los contenedores Docker y archivos de configuración.

```
$ sudo kolla-ansible -i multinode destroy --yes-i-really-really-mean-it
```

Después de destruir el escenario, algunos puertos se quedan ligados a los procesos o demonios previamente activos. Es necesario reiniciar todos los demonios asociados antes de lanzar el escenario nuevamente.

```
$ sudo systemctl daemon-reload
```

Si se quiere borrar todos los contenedores e imágenes dentro de los nodos de cómputo, ejecutar lo siguiente:

- Para los contenedores

```
$ sudo docker rm $(sudo docker ps -aq)
```
- Para las imágenes

```
$ sudo docker image rm $(sudo docker image ls -a)
```

Para ver los logs de los servicios que se despliegan en Dockers, en el **controlador** acceder al contenedor **fluentd** y revisar los logs de cada servicio:

```
$ sudo docker exec -it fluentd /bin/bash
```

Dentro del Docker, acceder al directorio `/var/log/kolla` con los logs asociados a los servicios de OpenStack.

B.5 - Despliegue de una Red demo personalizada

Se ha creado un script llamado **script-init.sh** el cual puede ser ejecutado en lugar del **init-runonce**, que es la demo predeterminada por Kolla-Ansible. Este script incluye las imágenes preparadas por el departamento de la Escuela para distribuciones cirros, trusty y xenial.

Fichero “script-init.sh”:

```
#!/bin/bash
#
# This script is meant to be run once after running start for the first
# time. It downloads the images needed to deploy instances in OpenStack

ARCH=$(uname -m)
IMAGE_PATH=/home/reicloud/data_reicloud/
IMAGE_URL=http://idefix.dit.upm.es/download/cnvr/ostack-images/
IMAGE_TYPE=linux

IMAGE1=cirros-0.3.4-x86_64-disk-vnx.qcow2
IMAGE_NAME1=cirros

IMAGE2=trusty-server-cloudimg-amd64-disk1-vnx.qcow2
IMAGE_NAME2=trusty-server

IMAGE3=xenial-server-cloudimg-amd64-disk1-vnx.qcow2
IMAGE_NAME3=xenial-server

# This EXT_NET_CIDR is your public network, that you want to connect to
# the internet via.
EXT_NET_CIDR='138.4.7.164/25'
EXT_NET_RANGE1='start=138.4.7.216,end=138.4.7.218'
EXT_NET_RANGE2='start=138.4.7.139,end=138.4.7.141'
```

```
EXT_NET_RANGE3='start=138.4.7.221,end=138.4.7.222'
#If needed, add another entry with this format:
"EXT_NET_RANGEx='start=x.x.x.x,end=y.y.y.y'"

EXT_NET_GATEWAY='138.4.7.129'

# Sanitize language settings to avoid commands bailing out
# with "unsupported locale setting" errors.
unset LANG
unset LANGUAGE
LC_ALL=C
export LC_ALL
for i in curl openstack; do
    if [[ ! $(type ${i} 2>/dev/null) ]]; then
        if [ "${i}" == 'curl' ]; then
            echo "Please install ${i} before proceeding"
        else
            echo "Please install python-${i}client before proceeding"
        fi
        exit
    fi
done

# Test for credentials set
if [[ "${OS_USERNAME}" == "" ]]; then
    echo "No Keystone credentials specified. Try running source openrc"
    exit
fi

# Test to ensure configure script is run only once
if openstack image list | grep -q cirros; then
    echo "This tool should only be run once per deployment."
    exit
fi

echo Checking for locally available cirros image.

# Let's first try to see if the image is available locally
# nodepool nodes caches them in $IMAGE_PATH
#Download the images that you need
if ! [ -f "${IMAGE_PATH}/${IMAGE1}" ]; then
    echo None found, downloading cirros image.
    curl -L -o ${IMAGE_PATH}/${IMAGE1} ${IMAGE_URL}/${IMAGE1}
else
    echo Using cached cirros image from the nodepool node.
fi

if ! [ -f "${IMAGE_PATH}/${IMAGE2}" ]; then
    echo None found, downloading cirros image.
    curl -L -o ${IMAGE_PATH}/${IMAGE2} ${IMAGE_URL}/${IMAGE2}
else
    echo Using cached cirros image from the nodepool node.
fi

if ! [ -f "${IMAGE_PATH}/${IMAGE3}" ]; then
    echo None found, downloading cirros image.
    curl -L -o ${IMAGE_PATH}/${IMAGE3} ${IMAGE_URL}/${IMAGE3}
else
    echo Using cached cirros image from the nodepool node.
fi
```



```

EXTRA_PROPERTIES=
if [ ${ARCH} == aarch64 ]; then
    EXTRA_PROPERTIES="--property hw_firmware_type=uefi"
fi

#Create the images from the files downloaded
echo Creating glance image.
openstack image create --disk-format qcow2 --container-format bare --
public \
    --property    os_type=${IMAGE_TYPE}    ${EXTRA_PROPERTIES}    --file
${IMAGE_PATH}/${IMAGE1} ${IMAGE_NAME1}
openstack image create --disk-format qcow2 --container-format bare --
public \
    --property    os_type=${IMAGE_TYPE}    ${EXTRA_PROPERTIES}    --file
${IMAGE_PATH}/${IMAGE2} ${IMAGE_NAME2}
openstack image create --disk-format qcow2 --container-format bare --
public \
    --property    os_type=${IMAGE_TYPE}    ${EXTRA_PROPERTIES}    --file
${IMAGE_PATH}/${IMAGE3} ${IMAGE_NAME3}

#Create a new network. In this case, the EXTERNAL NETWORK needed to
floating IPs. It is necessary to create a network and a subnet belonging
to that network
echo Configuring neutron.
openstack network create --external --provider-physical-network physnet1
\
    --provider-network-type flat public1
openstack subnet create --no-dhcp \
    --allocation-pool    ${EXT_NET_RANGE1}    --allocation-pool
${EXT_NET_RANGE2} --allocation-pool ${EXT_NET_RANGE3} \
    --network public1 \
    --subnet-range ${EXT_NET_CIDR} --gateway ${EXT_NET_GATEWAY} ext-
subnet1
#-----
#Create a new network. It is a demo network
openstack network create --provider-network-type vxlan demo-net
openstack subnet create --subnet-range 10.0.0.0/24 --network demo-net \
    --gateway 10.0.0.1 --dns-nameserver 8.8.8.8 demo-subnet

#Create a new router. It is a demo router
openstack router create demo-router
openstack router add subnet demo-router demo-subnet
openstack router set --external-gateway public1 demo-router

# Get admin user and tenant IDs
ADMIN_USER_ID=$(openstack user list | awk '/ admin / {print $2}')
ADMIN_PROJECT_ID=$(openstack project list | awk '/ admin / {print $2}')
ADMIN_SEC_GROUP=$(openstack security group list --project
${ADMIN_PROJECT_ID} | awk '/ default / {print $2}')

# Sec Group Config
openstack security group rule create --ingress --ethertype IPv4 \
    --protocol icmp ${ADMIN_SEC_GROUP}
openstack security group rule create --ingress --ethertype IPv4 \
    --protocol tcp --dst-port 22 ${ADMIN_SEC_GROUP}
# Open heat-cfn so it can run on a different host
openstack security group rule create --ingress --ethertype IPv4 \

```

```
--protocol tcp --dst-port 8000 ${ADMIN_SEC_GROUP}
openstack security group rule create --ingress --ethertype IPv4 \
--protocol tcp --dst-port 8080 ${ADMIN_SEC_GROUP}
#Configuring nova public key in order to using them inside the virtual
machines.
if [ ! -f ~/.ssh/id_rsa.pub ]; then
    echo Generating ssh key.
    ssh-keygen -t rsa -f ~/.ssh/id_rsa
fi
if [ -r ~/.ssh/id_rsa.pub ]; then
    echo Configuring nova public key and quotas.
    openstack keypair create --public-key ~/.ssh/id_rsa.pub mykey
fi
#Configure the "quota" allowed to the user.
# Increase the quota to allow 40 ml.small instances to be created

# 40 instances
openstack quota set --instances 40 ${ADMIN_PROJECT_ID}

# 40 cores
openstack quota set --cores 40 ${ADMIN_PROJECT_ID}

# 96GB ram
openstack quota set --ram 96000 ${ADMIN_PROJECT_ID}

# add default flavors, if they don't already exist
if ! openstack flavor list | grep -q ml.tiny; then
    openstack flavor create --id 1 --ram 512 --disk 1 --vcpus 1 ml.tiny
    openstack flavor create --id 2 --ram 2048 --disk 20 --vcpus 1
ml.small
    openstack flavor create --id 3 --ram 4096 --disk 40 --vcpus 2
ml.medium
    openstack flavor create --id 4 --ram 8192 --disk 80 --vcpus 4
ml.large
    openstack flavor create --id 5 --ram 16384 --disk 160 --vcpus 8
ml.xlarge
fi

DEMO_NET_ID=$(openstack network list | awk '/ demo-net / {print $2}')

cat << EOF

Done.

To deploy a demo instance, run:

openstack server create \\\
--image ${IMAGE_NAME} \\\
--flavor ml.tiny \\\
--key-name mykey \\\
--nic net-id=${DEMO_NET_ID} \\\
demo1

EOF
```