



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE MÁSTER

*Prueba de concepto y prototipo de una nube privada protegida
para las redes del MDEF basada en OpenStack.*

**Máster en Gestión y Dirección de Sistemas y Tecnologías de la Información y
las Comunicaciones y de Seguridad de la Información**

ALUMNO: José Antonio Castilla Barea

DIRECTORES: Miguel Rodelgo Lacruz

CURSO ACADÉMICO: 2017-2018



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE MÁSTER

*Prueba de concepto y prototipo de una nube privada protegida
para las redes del MDEF basada en OpenStack.*

**Máster en Gestión y Dirección de Sistemas y Tecnologías de la Información y
las Comunicaciones y de Seguridad de la Información**

RESUMEN

El objetivo principal del trabajo es realizar la implementación completa de un sistema de nube privada con OpenStack.

Tras realizar un somero estudio del origen y la historia del “Proyecto OpenStack” y una búsqueda de posibles alternativas a OpenStack, se han analizado los componentes del Proyecto para proponer un diseño de nube privada adaptado a las especiales características de protección de la información que requiere el ámbito del Ministerio de Defensa (MDEF).

Me propongo, realizando una primera implantación básica del sistema a modo de prototipo, que funcione, ofrecer una guía de implantación y un conjunto de resultados sobre su bastionado que sirva de base para futuras implementaciones.

Este trabajo será una de las primeras (quizás la primera) experiencia documentada de aprovisionamiento de una nube privada en las Fuerzas Armadas (FAS) para ofrecer infraestructura como servicio a demanda. Lo complejo de OpenStack y la adaptación de sus productos para cumplir los exigentes estándares de seguridad que imponen las normas en el MDEF ponen límite al alcance de este trabajo, y dentro del mismo a la profundidad y detalle conseguido, por lo que ha supuesto un reto de dimensiones considerables.

PALABRAS CLAVE

“computación en la nube”, “nube privada”, “virtualización”, “IaaS”, “OpenStack”, “*cloud computing*”, “*cloud*”, “*virtualization*”.

AGRADECIMIENTOS

A Ana, mi querida esposa, por haber aguantado mi falta de atención por causa de la realización de este trabajo. A mis hijas María y Clara, por haber hecho a su madre más leve este trago. En general a mi familia porque sin su apoyo no lo hubiera realizado.

A mi profesor D. Miguel Rodelgo Lacruz, excelente en su paciencia y en su confianza al haberme dejado libertad de decisión y aportado luz cuando más la necesitaba.

CONTENIDO

Contenido	1
Índice de Figuras	5
Índice de Tablas.....	7
1 Introducción y objetivos	8
1.1 Motivación para la realización de este trabajo	8
1.1.1 Existencia de normativa.....	8
1.1.2 OpenStack es el futuro.....	10
1.1.3 OpenStack, el reto.....	10
1.2 Objetivos del TFM	11
1.3 Estructura de esta memoria	11
1.4 Convencionalismos en este documento	12
1.4.1 Tipos de texto	12
1.4.2 Figuras y tablas	13
1.4.3 Referencias en el texto.....	13
2 Estado del arte	15
2.1 Computación en la nube (<i>cloud computing</i>)	15
2.1.1 Características de la computación en la nube	15
2.1.2 Modelos de servicio en la nube	16
2.1.3 Modelos de implementación.....	18
2.2 Virtualización.....	19
2.2.1 ¿Qué es la virtualización?	19
2.2.2 Tipos de virtualización y tipos de hipervisores	20
2.2.3 ¿Para qué virtualizar? Ventajas y desventajas de la virtualización	22
2.3 OpenStack	24
2.3.1 Que es OpenStack [13]	24
2.3.2 Los componentes del proyecto OpenStack.....	25
2.3.3 Distribuciones de OpenStack.....	30
2.4 Securización, bastionado o <i>hardening</i>	32
2.4.1 El entorno regulatorio	32
2.4.2 Guías de seguridad.....	35
2.4.3 Lo que se puede hacer	36
3 Desarrollo del TFM	38
3.1 Un modelo de nube para los sistemas clasificados del ET	38

3.1.1 Descripción de sistemas actuales. Un nodo maestro como base de partida.	38
3.1.2 ¿Una nube privada?, ¿para qué?	39
3.1.3 Descripción del prototipo y su integración en el nodo maestro.....	42
3.2 Las primeras instalaciones all-in-one.....	45
3.2.1 Instalación all-in-one sobre CentOS con PackStack	46
3.2.2 Instalación all-in-one sobre RHEL con PackStack.....	50
3.2.3 Sobre las instalaciones con PackStack	51
3.3 Ubuntu OpenStack <i>all-in-one</i>	53
3.3.1 Instalación all-in-one sobre Ubuntu con Conjure-up.....	53
3.3.2 Instalación all-in-one sobre Ubuntu con DevStack	58
3.4 Virtualización de un clúster de OpenStack.	59
3.4.1 Nodo de control y de red.	61
3.4.2 Nodo de computación.	90
3.4.3 Referencias para la instalación de nodos de almacenamiento.	96
3.5 Ubuntu OpenStack en clúster (enfocado a producción).....	97
3.5.1 Situación	97
3.5.2 La planificación de la infraestructura de nube.....	98
3.5.3 Mínima configuración del clúster.	102
3.5.4 Instalación del clúster con MaaS Server y conjure-up	103
3.5.5 Instalación del clúster con DevStack	106
3.5.6 Instalación del clúster paso a paso.....	108
4 Resultados / Validación / Prueba.....	110
4.1 Resultado del estudio y recopilación de información y documentación.....	110
4.1.1 De los antecedentes y fundamentos	110
4.1.2 Del estado del Arte	110
4.2 Resultados de las instalaciones	111
4.2.1 Instalaciones all-in-one	111
4.2.2 Instalaciones multinodo	114
4.3 Pruebas de funcionamiento y desempeño	116
4.4 Determinación de configuraciones de seguridad	117
5 Conclusiones y líneas futuras	118
5.1 Conclusiones sobre los estudios previos.....	118
5.2 Conclusiones sobre los diseños de nube privada	119
5.3 Conclusiones sobre el prototipado	119
5.4 Conclusiones sobre la securización.....	121

5.5 Posibles líneas futuras de trabajo.	121
6 Bibliografía.....	123
Anexo I: Acrónimos utilizados.....	129
Anexo II: Conceptos y definiciones	132
Anexo III: Guía de securización.....	137
aIII.1. Objeto	137
aIII.2. Seguridad durante la instalación del sistema operativo	137
aIII.2.1. Imágenes del sistema operativo	138
aIII.2.2. Dispositivo de arranque	138
aIII.2.3. Primeros pasos de la instalación	138
aIII.2.4. Configuración de la cuenta de usuario	140
aIII.2.5. Particionado de los discos	140
aIII.2.6. Instalación del sistema base y configuraciones finales.....	144
aIII.3. Primeras configuraciones de seguridad del sistema operativo.....	144
aIII.3.1. Modificar las características de montaje del directorio /dev/shm.....	144
aIII.3.2. Sustituir el directorio /var/tmp por un enlace simbólico al /tmp.....	145
aIII.3.3. Modificar las características de montaje del directorio /tmp	145
aIII.3.4. Modificar opciones de montaje del sistema de ficheros	145
aIII.3.5. Primera configuración del firewall.	146
aIII.3.6. Configuración de la gestión de paquetes.	149
aIII.3.7. Configuración de las contraseñas (autenticación).....	153
aIII.3.8. Configuración de restricciones de acceso.	155
aIII.3.9. Configuración de restricciones de acceso al sistema de ficheros	159
aIII.3.10. Configuración de restricciones de acceso por red a nivel de aplicación.....	159
aIII.4. Reducción de la superficie de ataque del sistema.	160
aIII.5. Limitaciones en permisos de carpetas y archivos	162
aIII.5.1. Propiedad de los archivos ‘passwd’, ‘shadow’, ‘group’ y ‘gshadow’	162
aIII.5.2. Permisos de los archivos ‘passwd’ y ‘group’	162
aIII.5.3. Permisos de los archivos ‘shadow’ y ‘gshadow’	162
aIII.5.4. Directorios con permisos de escritura.....	162
aIII.5.5. Permisos de escritura universal.....	162
aIII.5.6. Permisos SUID/SGID activados en ficheros ejecutables.....	163
aIII.5.7. Reparar ficheros huérfanos	163
aIII.5.8. Directorios con permisos de escritura globales sin un propietario	164
aIII.6. Limitaciones en ejecutables del sistema.....	164

aIII.7. Securizar las cuentas existentes en el sistema	165
aIII.7.1. Lo primero es eliminar las cuentas innecesarias	165
aIII.7.2. No debe permitirse ningún usuario sin contraseña	165
aIII.7.3. Hay que comprobar que nadie más que root tiene UID '0'	165
aIII.7.4. Creación de un grupo único a cada usuario	165
aIII.7.5. Securizar ficheros de configuración de sesión para cuentas de usuario	166
aIII.7.6. Eliminar el historial de acciones de los usuarios	166
aIII.8. Evitar el montaje de dispositivos usb de almacenamiento masivo	166
aIII.9. Securitización de parámetros de red	167
aIII.9.1. Fortalecimiento de la pila TCP/IP	167
aIII.9.1. Deshabilitar IPv6	167
aIII.10. Configuración segura de servicios	167
aIII.10.1. Servicio SSH	167
aIII.10.2. Servicio CROND	169
aIII.10.3. NTP	169
aIII.11. Servicios de seguridad específicos	169

ÍNDICE DE FIGURAS

Figura 1-1. Logotipo del Centro Universitario de la Defensa	13
Figura 2-1. Diferencias en el alcance y el control entre los modelos de servicios en la nube [5]....	17
Figura 2-2. Tipos y características de las nubes [6]	19
Figura 2-3. Arquitectura de virtualización utilizando un hipervisor de tipo 1 [10]	21
Figura 2-4. Arquitectura de virtualización utilizando un hipervisor de tipo 2 [10]	21
Figura 2-5. Ecosistema de OpenStack (figura simplificada) [13]	24
Figura 2-6. Mapa de los componentes de OpenStack [27].....	29
Figura 3-1. Esquema de un posible nodo autoprotegido de un sistema clasificado.	38
Figura 3-2. Arquitectura lógica del ecosistema OpenStack [54].....	43
Figura 3-3. Pantalla de fin de instalación de OpenStack PackStack.	48
Figura 3-4. Resultado del comando <code>ip addr</code> tras la instalación de OpenStack.	49
Figura 3-5. Contenido del fichero <code>ifcfg-enp0s3</code>	49
Figura 3-6. Contenido del fichero <code>ifcfg-br-ex</code>	50
Figura 3-7. Acceso a OpenStack desde CLI.....	50
Figura 3-8. Pantalla final del script <code>conjure-up</code>	56
Figura 3-9. Portal Horizon, página de login.	57
Figura 3-10. Portal Horizon tras hacer login.	57
Figura 3-11. Final de la instalación de OpenStack con DevStack.	59
Figura 3-12. Creación del dominio ‘example’ y de los proyectos ‘service’ y ‘myproject’	69
Figura 3-13. Creación de un usuario, un rol y vinculación de rol a proyecto.	70
Figura 3-14. Solicitud de token de autenticación.	70
Figura 3-15. <i>Script</i> de configuración de variables de entorno y solicitud de <i>token</i>	71
Figura 3-16. Creación del usuario glance.....	72
Figura 3-17. Creación del servicio ‘glance’.	73
Figura 3-18. Configuración de los API <i>endpoints</i> para el servicio de imágenes.	73
Figura 3-19. Listado de nodos de computación en el nodo de control.....	80
Figura 3-20. Resultado del descubrimiento de host de computación.	80
Figura 3-21. Servicios Nova activos.	81
Figura 3-22. Catálogo de endpoints.	82
Figura 3-23. Relación de imágenes cargadas y activas.	82
Figura 3-24. Estado de las API de ‘cell’ y ‘placement’.	83
Figura 3-25. Networking opción 1. Redes de proveedor [74].....	84
Figura 3-26. Prueba de la instalación de neutrón.	89

Figura 3-27. Fuentes NTP en el nodo de control.	92
Figura 3-28. Fuentes NTP en el nodo de computación.	93
Figura A3-1. Instalación de Ubuntu. Carátula de inicio.....	138
Figura A3-2. Instalación de Ubuntu. Configuración de red.	139
Figura A3-3. Instalación de Ubuntu. Particionado.....	143
Figura A3-4. Servicios del nodo de control.	161

ÍNDICE DE TABLAS

Tabla 1-1. Ejemplo de tabla.....	13
Tabla 2-1. Ventajas y desventajas de la virtualización.....	23
Tabla 2-2. Historial de versiones de OpenStack, desde la que se encuentra en desarrollo hasta la primera descontinuada [14].....	25
Tabla 2-3 Componentes en la versión Rocky [15]	26
Tabla 2-4. Equivalencias entre grados de clasificación de la información.	34
Tabla 3-1. Configuraciones de OpenStack según su propósito.....	40
Tabla 3-2. (Continuación de la tabla 3-1). Configuraciones de OpenStack según su propósito.	41
Tabla 3-3. Principales webs de DevStack.	52
Tabla 3-4. Datos de instalación del SO del nodo de control.	61
Tabla 3-5. Lista de servicios que requieren contraseña y sus referencias en la documentación oficial de OpenStack [67].....	64
Tabla 4-1. Cómputo de horas de trabajo invertidas.....	110
Tabla 4-2. Resumen del resultado de las instalaciones all-in-one.....	114
Tabla 4-3. Resumen del resultado de las instalaciones multinodo.....	116
Tabla A3-1. Particionado de un disco de 150 Gb.....	142
Tabla A3-2. Configuración de Iptables.	148
Tabla A3-3. Archivos de configuración de módulos PAM.....	154
Tabla A3-4. Servicios en Ubuntu [44]	162
Tabla A3-5. Ficheros con permisos suid/sgid activados [44]	163
Tabla A3-6. Usuarios asociados a servicios [44]	165
Tabla A3-7. Soluciones para otras medidas de seguridad.....	170

1 INTRODUCCIÓN Y OBJETIVOS

En este capítulo se contextualiza el trabajo, se motiva y se detallan los objetivos que pretende conseguir. Este capítulo finaliza con una sección que describe la organización de la presente memoria.

1.1 Motivación para la realización de este trabajo

El master en el que se desarrolla el trabajo final (TFM) está enfocado a la dirección TIC, pero nos ha ofrecido a los alumnos la oportunidad de acercarnos lo suficiente a la tecnología como para tomar contacto con las más innovadoras.

Hemos visto como la computación en la nube nos brinda oportunidades en forma de servicios a coste contenido para soportar nuestras funciones y puede ser una herramienta útil en las misiones de las Fuerzas Armadas (FAS), pero solo hemos visto esta tecnología *cloud* desde el punto de vista del cliente, y el primer impulso que me motiva ha sido entenderla desde el del proveedor de servicios (ISP), ¿resultará muy difícil desplegar o implementar una nube segura en nuestros sistemas? Esa es la cuestión que a nivel personal me hacía al enfrentar este trabajo.

1.1.1 Existencia de normativa

La segunda cuestión es si hay opción a que en el Ministerio de Defensa (MDEF) se doten sistemas en las nubes públicas de diferentes proveedores de servicios, se instale un sistema de nube privada, o ambos construyendo una nube híbrida, (más adelante se explicarán estos conceptos). En el presente TFM propongo una prueba de concepto, la cuestión planteada tiene sentido si la normativa da pie a que el *Cloud Computing* se utilice en el MDEF y por nuestras FAS.

Permítanme la licencia de adelantar la respuesta a esta cuestión. Desde la orientación a servicios fijada en “La Política CIS/TIC” [1], plenamente desarrollada en las líneas estratégicas mediante el “PECIS” [2] y normalizada a nivel técnico la tecnología aplicable por la Arquitectura Global (AG) [3]..., las opciones de computación en la nube son una realidad a nuestro alcance para satisfacer las necesidades del MDEF respecto a la cual tenemos la responsabilidad de estudiar sus posibilidades.

La Política de los Sistemas y Tecnologías de la Información y las Comunicaciones del Ministerio de Defensa, (Orden DEF/2639/2015, de 3 de diciembre), viene a sentar las bases en cuanto a tecnologías y sistemas de información para el desarrollo de las capacidades CIS en el Ministerio. Establece en el artículo 5 los principios rectores de la misma, siendo el primero (y no por ello el más importante), la Orientación a Servicios “*donde la información se identifica como un recurso estratégico sobre el que se debe buscar la superioridad para facilitar el cumplimiento y alcanzar el éxito de los cometidos encomendados al Ministerio y de las misiones de las FAS*” [1].

Básicamente el cumplimiento de estos principios viene a permitir que la información, por su carácter estratégico, sea fiable y accesible con la debida protección, en todo momento y lugar, para cualquier usuario que la precise, conforme a su perfil autorizado, y a los requisitos de dicha información. Esto es la finalidad de la Política.

Para alcanzar esta finalidad se establecen los “ejes estratégicos (EE)”. El tercer EE es la Normalización Técnica, y es por el que:

1. Se desarrollaran las Arquitecturas (conforme al marco NAF de la OTAN).
2. Se priorizan “*soluciones comunes aplicables, ya en uso en la AGE, la OTAN y la UE o países aliados, para mejorar la interoperabilidad y abaratar costes. De no ser posible, se adquirirán productos comerciales ya desarrollados (COTS, «Commercial Off-The-Shelf») de probada efectividad y calidad, y finalmente, solo si es estrictamente necesario, se abordará la creación de una solución propietaria*” [1].

3. Se establece el Catálogo Unificado de Estándares CIS/TIC del Ministerio de Defensa (CUE) incluido en la AG, buscando la mayor interoperabilidad, seguridad, resiliencia, etc...
4. Se organizarán los servicios conforme a una taxonomía propia del Ministerio de Defensa, siguiendo el modelo OTAN (*NATO C3 Classification Taxonomy*).
5. *“Se tenderá a la aplicación de tecnologías emergentes y contrastadas en el ámbito nacional e internacional del sector, que favorezcan la eficacia, interoperabilidad y eficiencia del empleo de los sistemas y la provisión de servicios” [1].*
6. *“Se adoptarán modelos y metodologías de referencia estandarizada de buenas prácticas en la gestión de los servicios y en la obtención, operación y mantenimiento de los sistemas” [1].*

La Arquitectura Global de Sistemas y Tecnologías de Información y Comunicaciones del Ministerio de Defensa (AG) y el Plan Estratégico de los Sistemas y Tecnologías de la Información y las Comunicaciones (PECIS) ya han sido dictados; la primera se aprueba por la Instrucción 58/2016, de 28 de octubre, y el segundo por la Instrucción 33/2018, de 6 de junio, ambas del Secretario de Estado de Defensa.

Los Ejes Estratégicos del PECIS (EE) son los recogidos en el artículo 6.2 de la Política CIS/TIC. El PECIS desarrolla los citados ejes estratégicos en objetivos estratégicos (OE), éstos en líneas de acción (LA), y éstas a su vez en actuaciones.

El EE 1 del PECIS, *“avanzar hacia una única infraestructura integral de información para la defensa (I3D), gestionada por el CESTIC” [2],* plantea:

“En la situación objetivo, conforme a lo establecido en la Política CIS/TIC y la AG CIS/TIC, se avanzará hacia una única infraestructura I3D que ofrecerá todos los Servicios CIS/TIC proporcionados por los CIS permanentes. La I3D será una red privada destinada a los servicios específicos de la defensa y seguridad nacional, para lo que estará dotada de los más altos estándares de calidad, disponibilidad, redundancia, seguridad y resiliencia. Así mismo, la I3D dispondrá de un núcleo protegido que asegure la disponibilidad de determinados Servicios CIS/TIC, con el alcance necesario que posibilite el funcionamiento de los sistemas de defensa, incluso en situaciones adversas o ante cualquier tipo de incidente que afecte a la misma” [2].

Dentro del citado EE 1 se establece el OE 1.2., *“diseñar y desplegar las Capacidades de Infraestructura de Información de la I3D”*, por el que se pretende *“La unificación y la racionalización, (teniendo en cuenta los requisitos protección de la información a transmitir, procesar y almacenar), de todas las plataformas de información (Centros de Proceso de datos -CPD’s- y equipos de sistemas de información) que transmiten, procesan y almacenan la información del Departamento en una única infraestructura de información, accesible para todos los usuarios, que proporcionará servicios de información de forma transversal a todo el Departamento” [2].*

Y el PECIS al desarrollar el contenido de este Objetivo dice textualmente:

“La consolidación de los CPD’s permitirá optimizar los recursos de procesamiento y almacenamiento del Departamento y estandarizar las actividades y los procesos de operación y, en consecuencia, una gestión más óptima de la información. Para ello, se proporcionarán, entre otros, servicios de procesamiento virtual y servicios en la nube (cloud services), optimizando el espacio físico y las prestaciones que se precisan” [2].

A través de las líneas de acción y actuaciones de los objetivos (no solo del enumerado) se dará respuesta a los requisitos operativos de las Capacidades CIS/TIC del MDEF, recogidos en el Apéndice VI de la AG CIS/TIC. Algunos de estos requisitos apuntan directamente a soluciones de virtualización y en la nube, como por ejemplo el RO-41 que reza:

“La infraestructura de información se debe implantar, en la medida de lo posible, en sus nodos permanentes y desplegados, mediante plataformas que ofrezcan entornos virtuales que permitan el uso de los recursos de una forma más eficiente” [3].

Por supuesto otros muchos Requisitos Operativos de las Capacidades pueden ser satisfechos mediante la adopción de tecnología de virtualización y en la nube, pero también por otros tipos de sistemas. Solo un meticuloso desarrollo de ingeniería puede decidir cómo cada sistema satisfecerá los requisitos de las Capacidades para las que se establezcan, desarrollo de ingeniería que es a su vez un requisito más de los recogidos en el Apéndice VI de la AG CIS/TIC (RO-99).

Continuando con la AG CIS/TIC en su Apéndice VIII. Taxonomía de Servicios CIS/TIC del Ministerio de Defensa se clasifican Servicios CIS/TIC del Ministerio de Defensa (respetando la clasificación que de los mismos hace la OTAN, y por tanto en correspondencia con los de la Alianza).

Dentro de los servicios que se clasifican están los “*Servicios de Información de Núcleo*” [3], bajo cuyo orden se alinean entre otros “*los Servicios de Infraestructura*” [3] donde se “*incluyen los servicios de procesamiento, almacenamiento y red de alto nivel que pueden usarse para implantaciones de centros de datos o servicios en la nube (cloud computing)*” [3].

Examinando esta clasificación de servicios vemos como dentro de los Servicios de Gestión de la Infraestructura se trata de unos Servicios de Provisión de Infraestructura los cuales deben permitir la gestión de las solicitudes, tiempo de ejecución y provisión de los recursos virtuales y escalables de infraestructura para todos los usuarios y localizaciones, concepto perfectamente identificable con el de Infraestructura como Servicio (IaaS), y si abundáramos más en esta parte de la clasificación vemos como se previenen servicios para dotar de almacenamiento, accesos a redes, procesamiento virtual, etc...

Por último, en el Catálogo Unificado de Estándares CIS/TIC del Ministerio de Defensa (CUE), incluido en la AG CIS/TIC como apéndice IX, se incluyen los estándares necesarios para desarrollar Servicios *Cloud*, tanto para virtualización, como para *Cloud Computing* y por último gestión de la infraestructura.

Concluyendo, las opciones de computación en la nube están perfectamente prevenidas en nuestra normativa del Departamento.

1.1.2 OpenStack es el futuro

La siguiente motivación, tras haber llegado a la conclusión anterior, la encuentro en el propio producto, OpenStack, ¿por qué OpenStack? Porque es un mundo relativamente nuevo (últimos años, desde el 2010) que a nivel técnico se encuentra a la vanguardia de las TIC.

OpenStack es capaz de manejar y controlar una gran cantidad de recursos informáticos, para procesado, almacenamiento y de red, todo un centro de datos, que se administrarían con un portal propio o mediante la API de OpenStack.

OpenStack tiene un ecosistema fuerte que funciona con tecnologías pensadas para entornos empresariales pero que son de código abierto soportado por una amplísima comunidad de desarrolladores, y no faltan empresas que apostando por OpenStack han desarrollado sus propias distribuciones ofreciendo soporte, consultoría e incluso desarrollos a medida (se ofertan en el denominado Marketplace de OpenStack), por lo que es ideal para infraestructuras heterogéneas.

Cientos de las empresas y organismos más grandes del mundo confían en OpenStack para gestionar y soportar sus actividades a diario, reduciendo los costes respecto a otros productos notablemente más caros como Azure de Microsoft.

1.1.3 OpenStack, el reto

Pero aparte de todo lo anterior no puedo dejar de sincerarme, quería intentar probar la tecnología *cloud*, instalándola ‘*on site*’, lo que es lo mismo, crear una nube privada, y OpenStack me permite

hacerlo gratis pues para este trabajo carezco de financiación posible. Efectivamente podría haber utilizado Microsoft Azure, o VMware vCloud, pero estaban fuera de mi alcance. El coste de los productos junto al tiempo disponible me ha abocado a usar OpenStack. Estos mismos condicionantes son un atractivo que otros pueden encontrar para utilizar este producto y no otro. Otra opción, quizás más sencilla podría haber sido CloudStack, un proyecto de Apache parecido a OpenStack.

1.2 Objetivos del TFM

El objetivo principal del trabajo es realizar la implementación completa de un sistema de nube privada con OpenStack y asegurarlo en lo posible sin perder de vista que es un prototipo y el tiempo para la realización de este trabajo muy limitado.

Para ello me propongo cumplir los siguientes objetivos

1. Realizar un somero estudio del origen y la historia del “Proyecto OpenStack” y una búsqueda de posibles alternativas a OpenStack.
2. Proponer un diseño de nube privada, a modo de prototipo, o el uso de un desarrollo basado en OpenStack, adaptado a las especiales características de protección de la información que requiere el ámbito del Ministerio de Defensa.
3. Probar diferentes formas de instalar o desplegar una nube privada basada en OpenStack.
4. Realizar una primera implantación básica del sistema a modo de prototipo, que funcione, para poder probar la tecnología.
5. Ofrecer una guía de implantación para una nube privada segura, o al menos un conjunto de buenas prácticas.

1.3 Estructura de esta memoria

La estructura del trabajo se ha dividido en seis capítulos y anexos que se describen a continuación:

Capítulo 1: Introducción y objetivos.

Es el presente capítulo. A modo de introducción se describe que ha motivado al alumno, autor, para realizar este trabajo, y qué objetivos se ha planteado alcanzar. También se describe como está organizada la memoria y los convencionalismos para ordenar las referencias, formatear el trabajo en un estilo adecuado, etc.

Capítulo 2: Estado del arte.

Comienza definiendo que es el *cloud computing*, clasifica las nubes y seguidamente se definen los servicios. Se repasan los tipos de virtualización para enfocarse en los que usa OpenStack. Se presenta el ecosistema de OpenStack y su funcionamiento, y por último se realizan los planteamientos básicos para elegir las configuraciones de seguridad del sistema basándose en guías.

Capítulo 3: Desarrollo del TFM.

Este capítulo probablemente resulte al lector algo heterogéneo, pues se expone como se ha realizado el trabajo y mezcla estudio, investigación, práctica y decisiones (en ocasiones tomadas al vuelo cuando algo no ha funcionado). Se exponen como se han instalado los productos (siguiendo diferentes procedimientos) y que configuraciones se han probado.

Capítulo 4: Resultados / Validación / Prueba.

Se repasan los trabajos realizados y los resultados obtenidos de los estudios y prácticas efectuadas.

Capítulo 5: Conclusiones y líneas futuras.

En este capítulo, se hace balance global del trabajo, aportando ideas sobre posibles líneas de trabajo futuras, que en lo posible mejoren las conclusiones y extiendan el alcance del presente TFM, que bien pudiera servir de referencia inicial para otros futuros alumnos.

Capítulo 6: Bibliografía.

Contiene las referencias consultadas y que se relacionan de acuerdo al formato IEEE. Para incluirlas se ha utilizado la aplicación Zotero.

En esta sección figuran todas las referencias, sean recursos web, libros, artículos, etc., incluyendo la información de autores, título de la obra, nombre de la publicación, año, edición y enlace más fecha de último acceso en el caso de referencias a recursos online.

Anexo I. Acrónimos utilizados.

Anexo II. Conceptos y definiciones.

Anexo III. Guía de securización.

Este anexo ofrece una relación de buenas prácticas para mejorar la seguridad del sistema de nube basado en OpenStack.

1.4 Convencionalismos en este documento

1.4.1 Tipos de texto

El texto está escrito usando la tipografía Times New Roman (12 pt) en color negro. Para los fragmentos de texto dentro de las celdas de las tablas se usa tipo Times New Roman (11 pt).

Para llamar la atención sobre alguna palabra o frase en particular estas se subrayan, aunque pertenezcan a alguna cita, y nunca se usan ni negritas ni cursivas para llamar la atención.

Las citas textuales se entrecomillan, (comillas dobles), y se transcriben textualmente en cursiva, con el mismo tipo y tamaño de fuente.

Los términos en inglés se escriben en cursiva.

Las negritas se utilizan solo en los títulos de primer y segundo nivel, (de los capítulos y anexos), en las cabeceras de las filas y columnas de las tablas, y en los pies de figuras y tablas.

Los títulos y los pies de figuras y tablas tienen un estilo específico diferente del contenido:

- Títulos de primer nivel (capítulos y anexos) precedidos de un número de orden, en azul (RGB 27, 27, 127), con tipo Times New Roman (20 pt).
- Títulos de segundo nivel precedidos de dos números de orden, en negro, con tipo Times New Roman (14 pt) en negrita.
- Títulos de tercer nivel precedidos de tres números de orden, en negro, con tipo Times New Roman (14 pt) en cursiva.
- Pies de figuras y tablas precedidos de un código de orden, en negro, con tipo Times New Roman (10 pt) en negrita.

En el texto se incluyen fragmentos de código, órdenes de Shell y fragmentos de archivos de configuración de aplicaciones. Estos fragmentos están escritos con Times New Roman (12 pt) en color azul sobre fondo gris.

En las órdenes de Shell se diferencian entre su ejecución por parte de *root* o de usuario no privilegiado mediante el *prompt*. # es el *prompt* de *root* y \$ el de usuario no privilegiado.

1.4.2 Figuras y tablas

Las figuras y tablas siempre aparecen centradas en la página e incluyen un pie debajo describiendo su contenido. Se han numerado con la notación num_capítulo-num_figura_dentro_capítulo y se referenciarán en el texto, como se puede ver en la Figura 1-1. Tras el título, si la figura está tomada de alguna fuente se indicará la misma con su referencia.



Figura 1-1. Logotipo del Centro Universitario de la Defensa

Las tablas se han numerado de igual manera que las figuras, como se aprecia en la

Tabla 1-1.

Datos en fila 6	Datos en col 2 y fila 6
Columna 1	Columna 2
Datos en fila 1	Datos en col 2
Datos en fila 2 y columna 1	Datos en col 2 y fila 2
Datos	Datos en col 2
Datos en fila4	Datos en col 2
Datos en fila 5	Datos en col 2 y fila 5
Datos en fila 6	Datos en col 2 y fila 6

Tabla 1-1. Ejemplo de tabla.

1.4.3 Referencias en el texto

Toda la información que se ha obtenido de alguna publicación, sea cual sea su tipo, o de alguna página web está referenciada a la bibliografía listada en el capítulo 0, de acuerdo con el estilo IEEE proporcionado por la aplicación Zotero.

Todas las inserciones entrecomilladas y en cursiva son frases literales de las referencias incluidas y mostradas junto a ellas o explícitamente descritas.

El estilo IEEE se corresponde con el siguiente formato (mismo que el establecido por la Dirección del Master):

- La referencia junto al texto se muestra mediante un número entre corchetes. Por ejemplo [3].

- Las referencias bibliográficas en el capítulo 6 se citan de la siguiente manera: primero los autores o *sponsors* separados por comillas. En segundo lugar el título de la obra, publicación o similar. En tercer lugar el año de la publicación. En cuarto lugar la página o páginas citadas (si ha lugar). Por ejemplo:
 - [3]Ministerio de Defensa, Instrucción 58/2016, de 28 de octubre, del Secretario de Estado de Defensa, por la que se aprueba la Arquitectura Global de Sistemas y Tecnologías de Información y Comunicaciones del Ministerio de Defensa. 2016.
 - [30] «OpenStack on Ubuntu is your scalable private cloud, by Canonical | Ubuntu». [En línea]. Disponible en: <https://www.ubuntu.com/OpenStack>. [Accedido: 17-sep-2018].

2 ESTADO DEL ARTE

En este capítulo se define que es el *cloud computing*, se clasifican las nubes y seguidamente se definen los servicios. Se repasan los tipos de virtualización para enfocarse en los que usa OpenStack. Se presenta el ecosistema de OpenStack y su funcionamiento, y por último se realizan los planteamientos básicos para elegir las configuraciones de seguridad del sistema basándose en guías.

2.1 Computación en la nube (*cloud computing*)

La información vertida en este apartado se encuentran en muchísimas publicaciones, pero se resume muy bien en la guía o publicación especial del *National Institute of Standards and Technology* (NIST) *Special Publication* (SP) 800-145, «*The NIST Definition of Cloud Computing*», de septiembre de 2011.

El NIST define la computación en la nube como un modelo, (utiliza dicho término), en representación de cualquier sistema informático que se ajuste a unas características determinadas.

Haciendo una traducción libre, el NIST viene a definirla como un modelo que permite a los usuarios el acceso independientemente de su ubicación a unos recursos informáticos compartidos (redes, servidores, almacenamiento, aplicaciones y servicios), que pueden aprovisionarse por ellos de la manera que más les convenga, bajo demanda, y ponerse en operación con el mínimo esfuerzo administrativo y la menor participación del proveedor de servicios [4].

La computación en la nube presenta cinco características principales, ofrece tres modelos de servicio y cuatro implementaciones.

2.1.1 Características de la computación en la nube

Autoservicio (a demanda del consumidor).

Esta característica permite que un cliente, o un usuario, (en definitiva un consumidor), pueda dotarse o aprovisionarse de recursos y/o capacidades informáticas, como capacidad de proceso (el tiempo del servidor) y el almacenamiento en red, de acuerdo con sus necesidades, por su cuenta y riesgo con mínima o ninguna interacción humana con ningún agente o administrador del proveedor de servicios (podría decirse unilateralmente y automáticamente).

Acceso universal.

Los recursos y capacidades están disponibles en red (ya sean estas de área local, metropolitanas o más extensas) y se accede a ellos a través de mecanismos estandarizados desde ordenadores comunes (portátiles o sobremesa), tablets, teléfonos móviles, etc.

Agrupación de recursos.

El proveedor de servicios agrupa los recursos informáticos (almacenamiento, procesamiento, memoria y ancho de banda de red) que dispone con la finalidad de dar servicio simultáneamente a muchos usuarios o clientes, para lo que los asigna dinámicamente (ya sean estos recursos físicos o virtuales) y en función de la demanda los reasignará siempre cumpliendo con los acuerdos de nivel de servicio (SLA)¹.

¹ En esta situación los usuarios o clientes tendrán una sensación de independencia respecto a su ubicación y desde donde se le asignan los recursos que digamos contrata, pues en realidad no sabe desde donde le están prestado el servicio, aunque, (y esto es importante desde el punto de vista de la seguridad), podría acordarse con el proveedor de servicios desde donde se prestarán estos, por ejemplo, país, provincia o CPD.

Flexibilidad²

La computación en la nube permite que recursos informáticos compartidos se asignen o liberen de un determinado usuario o grupo de ellos en función de su demanda y de su contrato, esta asignación tiende a ser lo más automática posible. Esta característica permite que el servicio sea escalable en función a la demanda, e incluso llegan a parecer a ojos del usuario ilimitadas ya que pueden asignarse en cualquier cantidad y en cualquier momento.

Monitorización.

Los sistemas en la nube controlan y optimizan automáticamente el uso de los recursos informáticos compartidos gracias a elementos específicamente dedicados a monitorizar el uso en términos de almacenamiento, de capacidad de proceso, ancho de banda y cuentas de usuario activas, siempre que sean parámetros relevantes para el servicio que se ofrece. Además la monitorización es indispensable en los procesos de tarificación.

2.1.2 Modelos de servicio en la nube

Software como servicio (SaaS).

En esta modalidad el proveedor instala las aplicaciones (por ejemplo ofimáticas) en un sistema en la nube y los clientes reciben la capacidad de utilizarlas. Lógicamente la instalación permitirá que múltiples usuarios las utilicen de manera que sus datos y procesos no se mezclen en absoluto, se mantengan por tanto aislados (suele emplearse el término en inglés *isolation*).

Los usuarios acceden a las aplicaciones servidas mediante sus clientes ligeros, típicamente desde los navegadores, aunque pueden servirse interfaces ligeras más específicas. Son ejemplos el correo web (como Gmail), o Office 365, suite que permite correr las aplicaciones Word, Excel, Access, PowerPoint, etc., desde un navegador.

El cliente no administra ni controla la infraestructura que desde la nube le sirve las aplicaciones (red, servidores físicos o virtuales, sistemas operativos, unidades de almacenamiento, etc.), por tanto en esta modalidad de servicio la carga administrativa en el cliente es la menor posible. Lo único que el cliente debe administrar es la configuración de la aplicación misma, por ejemplo cuando se almacenan perfiles para incrustar metadatos o configuraciones similares.

Respecto a la infraestructura en la nube, por ella se entiende: *“la colección de hardware y software que permite las cinco características esenciales de la computación en la nube. La infraestructura de la nube se puede ver como un sistema de dos capas, una física y una de abstracción. La capa física se compone de los recursos de hardware necesarios para admitir los servicios en la nube, que se proporcionan, y generalmente incluye componentes de servidor, almacenamiento y red. La capa de abstracción consiste en el software implementado a través de la capa física, que manifiesta las características esenciales de la nube. Conceptualmente, la capa de abstracción se encuentra encima de la capa física”* [4] (traducido del documento de la referencia).

Plataforma como servicio (PaaS).

En esta modalidad el cliente recibe del proveedor la capacidad de instalar aplicaciones en un sistema en la nube mediante la instalación de las bibliotecas y servicios que la aplicación precisa para funcionar y que permiten configurar desde el idioma hasta los accesos y herramientas de la propia aplicación. Todos los elementos proporcionados al cliente y necesarios para la aplicación o aplicaciones deben ser compatibles con la infraestructura del proveedor.

² En muchos documentos se prefiere la palabra elasticidad, como dando sensación de elongación o encogimiento en función de la demanda, y es la palabra que se usa por los expertos de OpenStack. Yo prefiero usar flexibilidad pues es el término que me parece más correcto en castellano.

La manera más habitual de prestar esta modalidad de servicio es el contenedor (por ejemplo los *dockers*), que son como máquinas virtuales muy ligeras que solo contienen lo preciso para que la aplicación para la que se ha configurado pueda instalarse y funcionar, por tanto no contienen el peso de un sistema operativo completo, entre otras características.

El cliente no gestiona ni administra la infraestructura del proveedor de la nube, (la red, los servidores, los sistemas operativos o el almacenamiento), pero tiene control sobre las aplicaciones implementadas y las configuraciones del contenedor que la aloja (aunque este punto puede ser delegado en el proveedor).

Para este caso la capa de abstracción comienza en el contenedor mismo, digamos que se hace más delgada para el proveedor y más pesada para el cliente.

Infraestructura como servicio (IaaS).

El servicio que se presta al cliente es capacidad de proceso, almacenamiento, redes y otros recursos informáticos donde pueden instalar y ejecutar cualquier *software*, como si de un sistema completo se tratara, con sus recursos de red, su disco duro, su procesador y memoria y su sistema operativo. Se le sirven máquinas virtuales.

Esta modalidad se podría confundir con el *hosting*, pero no es lo mismo precisamente por las características vistas en el apartado anterior. Cuando se contrata un servidor virtual en un ISP este no se adapta a la demanda y la de los otros clientes y no se aprovisiona automáticamente, ni tiene en su conjunto las características esenciales del *cloud computing*. Por supuesto no puede confundirse en absoluto con el *housing*.

El cliente no gestiona la infraestructura subyacente de la nube del proveedor y que este utiliza para servir la infraestructura demandada, pero tiene control sobre los sistemas operativos, el almacenamiento y las aplicaciones implementadas; y posiblemente un control limitado de los componentes de red seleccionados (por ejemplo, *firewalls host*).

En la Figura 2-1 a continuación se muestra la diferente perspectiva del proveedor y del cliente en cuanto a su alcance a lo largo de las diferentes capas del sistema en la nube, podríamos decir que las dos más bajas son la capa física mientras que las tres superiores son la abstracción. Dependiendo del modelo de servicio la responsabilidad de la gestión cae más del lado del proveedor o del cliente.

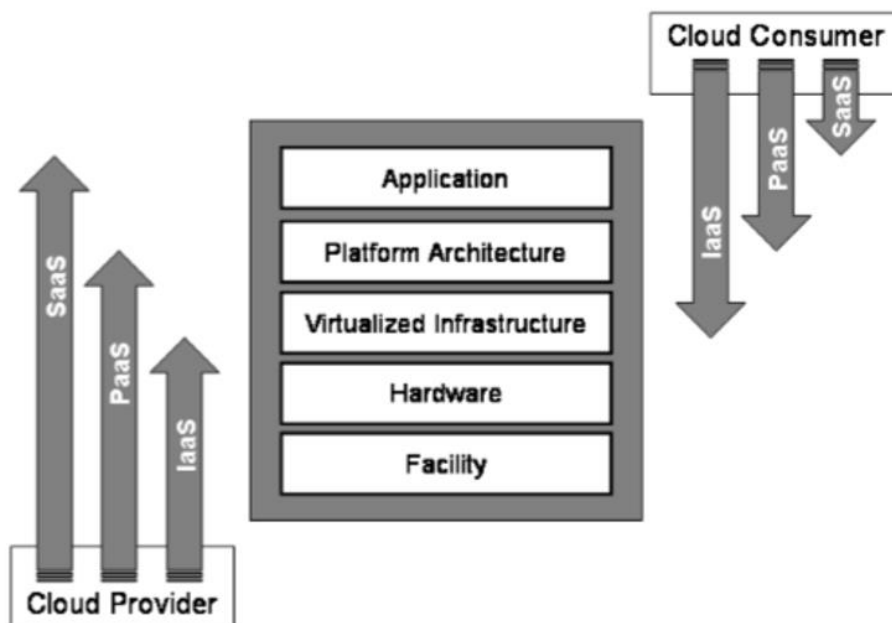


Figura 2-1. Diferencias en el alcance y el control entre los modelos de servicios en la nube [5]

2.1.3 Modelos de implementación

Nube privada.

Una organización instala su propia infraestructura de nube para su uso exclusivo.

Uno de los aspectos es la propiedad, es de la organización, pues ya sea por adquisiciones o mediante alquiler el efecto es el mismo. Dentro del aspecto de la propiedad cabe el *housing*, incluso el *hosting*.

Por tanto esta infraestructura puede ser gestionada y operada por la organización misma o por un tercero, con el cual contrate la administración el alojamiento de equipos físicos o el alquiler de máquinas.

Esta implementación puede hacerse incluso en la intranet corporativa, no siendo (dependiendo del despliegue de la organización), necesario ni el acceso a Internet.

Nube comunitaria.

En realidad es una modalidad de nube privada pero que satisface las necesidades de una comunidad de propietarios.

Lo normal es que los socios de la comunidad compartan buena parte de los aspectos de su negocio, o sus requisitos funcionales, por ejemplo su misión, visión y valor, sus requisitos de seguridad, políticas, normativas y regulaciones sectoriales, etc.

Esta implementación puede hacerse incluso en la intranet corporativa de uno de los socios, o si todos están juntos en un edificio o campus, en la red existente, no siendo (dependiendo del despliegue de los socios), necesario ni el acceso a Internet, aunque en este caso la dependencia de la Red de Redes suele ser mayor que en el caso anterior.

Nube pública.

La nube implementada se ha construido para un uso abierto de una amplia gama de clientes, para el público en general.

Puede ser propiedad de una organización empresarial, académica o gubernamental, o alguna combinación de ellos y ser gestionada por un agente o un proveedor de servicios.

En este modelo se despliega la infraestructura en las instalaciones del proveedor de la nube.

Nube híbrida.

Esta implementación es una combinación de dos o más modelos anteriores que permanecen como nubes separadas, pero que comparten tecnología basada en estándares para hacer posible la portabilidad de datos y aplicaciones (por ejemplo, para poder hacer balanceo de carga en términos de almacenamiento, capacidad de proceso, etc. entre nubes).

La figura 2-2 muestra los tipos y características de las nubes.



Figura 2-2. Tipos y características de las nubes [6]

2.2 Virtualización

Sin la virtualización es imposible la computación en la nube, por eso se describe en este capítulo como tecnología subyacente. De una manera simplificada podríamos decir que cuando a una infraestructura de virtualización se le incorporan los elementos necesarios para dotarla de las cinco características esenciales del *cloud* descritas en el apartado anterior, nace la nube.

La virtualización se inventó el siglo pasado en los albores de la informática, en los años 60 de manos de IBM, quizás buscando la eficiencia de los *mainframe* que gracias a ella multi-procesaban y podían ejecutar varias aplicaciones al mismo tiempo.

Posteriormente en los años 80 la arquitectura x86 vino a dominar el mercado, con un conjunto de instrucciones de procesador más eficiente y un menor coste, un abaratamiento real de la tecnología, y la llegada de los ordenadores personales. Éstos se conectaron muchos entre si y el modelo cliente-servidor se extendió y trajo la computación distribuida mediante la que las máquinas colaboraban entre ellas para procesar en paralelo (más o menos).

Durante los años 90 el modelo se consolidó y la virtualización pasó a un segundo plano, hasta el final de la década, que volvió a ponerse de moda de nuevo ya que la evolución de los procesadores los potenció y empezaban a desaprovechar recursos. Con ello se llegó en los años 90 al mismo punto de los años 60.

Popek y Goldberg formularon un conjunto de condiciones [7] que debe cumplir una arquitectura de procesador para soportar virtualización eficientemente. Los sistemas x86 presentaban hasta 17 problemas o incumplimientos de estas condiciones, al contrario que otras arquitecturas del momento (de IBM y Motorola), pero VMware presentó en 1999 un sistema basado en x86 que incluía una capa de abstracción que interceptaba las instrucciones problemáticas para el procesador y las adaptaba y convertía en instrucciones seguras, dejando el resto de instrucciones trabajar de forma original. VMware había creado y presentado al mercado el primer hipervisor en x86, tomando el control [8].

2.2.1 ¿Qué es la virtualización?

La virtualización es la emulación de uno o más ordenadores en un equipo físico único. Realmente lo que se emula es un *hardware* mediante un *software*, un programa denominado hipervisor. El *hardware* emulado se comporta como un ordenador físico y permite la instalación de un sistema operativo (SO) como si lo fuera.

A cada *hardware* emulado se le denomina invitado (*guest*) mientras al ordenador físico que lo emula gracias al hipervisor se le denomina anfitrión (*host*). En un *host* pueden emularse más de un *guest*. El equipo *host* reparte sus recursos *hardware* entre todos los *guest*, esto significa que un solo equipo puede

actuar como varios equipos diferentes que pueden trabajar juntos (formando *clusteres*³) o ser totalmente independientes. Los *guest* son las denominadas máquinas virtuales (VM) y tienen su propio sistema operativo (Linux, Windows, etc.) instalado, y todo gracias a los avances en la tecnología *hardware* que ha permitido el aumento de la potencia de computo.

2.2.2 Tipos de virtualización y tipos de hipervisores

Hablamos de dos tipos de virtualización, la completa y la paravirtualización. Además se habla de la virtualización de escritorio o de contenedores, que es algo parecido pero no exactamente lo mismo.

Virtualización completa.

Es la que permite crear MV,s con sistemas operativos sin modificar.

El SO del ordenador es el encargado de gestionar el *hardware*, para eso contiene los programas con las funciones primitivas necesarias y los *drivers*. Cuando al SO se le presenta un *hardware* virtual, falso, recreación de uno real, el *software* que lo recrea debe interceptar las instrucciones del SO que van dirigidas al *hardware* físico, el real. Muchas instrucciones pueden dejarse pasar sin más intervención pero otras deben modificarse para asegurar el funcionamiento del *hardware* real, manejando la concurrencia, capturando y liberando los recursos, etc.

El hipervisor es la capa de abstracción que comunica el SO del *guest* y el *hardware* físico. De cara al SO gestiona sus solicitudes igual que lo haría el *hardware* físico, pero de cara al *hardware* físico se puede comportar de maneras diferentes dependiendo de su arquitectura, lo que da lugar a la clasificación de los hipervisores.

Existen a su vez dos subtipos, la virtualización clásica y la traducción binaria. En la clásica los procesadores físicos del *host* deben tener una arquitectura que cumpla con las condiciones de Popek y Goldberg [7], mientras que la traducción binaria es lo que inventó VMware en 1999 para virtualizar sobre x86. Para virtualización clásica utilizamos los hipervisores del tipo 1 y para la traducción binaria los del tipo 2.

Hipervisor de tipo 1

Son los llamados nativos, *unhosted* o *bare-metal*. Dispuesta una máquina física, este tipo de hipervisor se instala como el SO de la máquina ejecutándose directamente en el *hardware* como se representa en la Figura 2-3. El hipervisor se carga en el arranque de la máquina, antes de que pueda ejecutarse cualquier MV, y controlan todas las instrucciones que hacen las MV,s.

“Un hipervisor de tipo 1 o bare metal se ejecuta en modo kernel directamente sobre el hardware de la máquina y arranca antes que el o los sistemas operativos que se ejecutan sobre él. El hipervisor se comporta como un sistema operativo dedicado o ejecuta una máquina virtual privilegiada (normalmente denominada dominio 0) desde la que se puede controlar el hipervisor y el resto de máquinas virtuales. Se suele utilizar fundamentalmente para servidores” [9].

Son soluciones de virtualización de este tipo Microsoft Hyper-V, Citrix XenServer y VMware ESXi-Server (actualmente VMware vSphere) por ejemplo.

Buena parte de la literatura reserva el término hipervisor para este tipo mientras que a los otros los denomina *virtual machine monitor*, monitores de máquina virtual, (VMM).

³ Desde 2017 la RAE admite la castellanización de los términos *cluster* por *clúster* y *clusters* por *clústeres*.



Figura 2-3. Arquitectura de virtualización utilizando un hipervisor de tipo 1 [10]

El tipo 1 a su vez se subclasifica en monolíticos y de microkernel, siendo los primeros más clásicos y más lentos pues el hipervisor debe tener sus propios *drivers* para el sistema operativo *guest*. En los de microkernel el hipervisor es un software muy sencillo que solo particiona el sistema físico entre los diversos sistemas virtualizados y no requieren drivers específicos para acceder al hardware ya que los *guest* usan los del *host*. NIST, publicación SP 800-125 [11].

Hipervisor de tipo 2

Son hipervisores que se instalan como una aplicación en un SO previo, y que por tanto se carga antes de poder arrancar el hipervisor, como se representa en la Figura 2-4, por lo que se les llama *hosted* (hospedados). Son ejemplos de este tipo VirtualBox (de Oracle Inc.), VMware WorkStation o Linux KVM (hay distribuciones en las que KVM se instala como *bare-metal*). Las MV,s se ejecutan por encima del hipervisor, y no actúan por si solas sobre el *hardware* físico, como si el conjunto formara una estructura de tres capas, SO-hipervisor-MV,s.

“Un hipervisor de tipo 2 o *hosted* se ejecuta en modo usuario como un proceso o programa más del SO anfitrión, es decir, sobre un sistema operativo” [9].



Figura 2-4. Arquitectura de virtualización utilizando un hipervisor de tipo 2 [10]

Paravirtualización

La paravirtualización implica modificar el código fuente del SO del *guest*, por lo que este debe estar disponible, y eso, por ejemplo, no es posible en sistemas Windows salvo para Microsoft.

El SO se modifica para sustituir instrucciones que son directas al *hardware* físico por otras que llaman al *hardware* virtual, al hipervisor en definitiva, teniendo que recompilarse ciertas librerías y

usarse en lugar de las primitivas. Las MV,s tienen un alto rendimiento y apenas se diferencian de las físicas.

“Permite ejecutar el sistema operativo del guest en modo privilegiado (anillo 0) sin simular el hardware mientras que el hipervisor garantiza el aislamiento gestionando las instrucciones sensibles” [9].

Virtualización a nivel de sistema operativo

No es en realidad un tipo de virtualización propiamente dicha. Consiste en permitir varias máquinas muy ligeras (no son ordenadores completos) de los mínimos recursos necesarios para poder ejecutar las aplicaciones previstas para ellas, siempre en modo de usuario no privilegiado. A cada una de estas máquinas se le denomina contenedor.

Todos los contenedores se ejecutan en el mismo sistema operativo e incluyen las aplicaciones y todo el conjunto de paquetes de los que depende, así su grado de aislamiento respecto al *host* no es muy alto por lo que no pueden ejecutarse contenedores Windows sobre Linux y viceversa.

Los contenedores ofrecen alto rendimiento en términos de velocidad de proceso y pocos requisitos de espacio (comparten almacenamiento del *host*).

Son ejemplos de contenedores Docker (del cual recibe el nombre) y VMware ThinApp.

2.2.3 ¿Para qué virtualizar? Ventajas y desventajas de la virtualización

Evidentemente virtualizamos por las ventajas que representa, y en cualquier caso si los inconvenientes superan a las ventajas, en un entorno o situación determinada, no lo haríamos.

No hablamos de ventajas y desventajas sin más, sino que debemos analizar el coste de asumir las desventajas más el ahorro que suponen las ventajas, para valorar el impacto que supone utilizar la virtualización.

Por ejemplo podemos encontrar que virtualizar una granja de servidores ahorra en *hardware* (ventaja) pero aumenta el coste de las licencias al incluir las del *software* de virtualización (desventaja), al mismo tiempo necesitamos administrar una capa *software* adicional (desventaja), por tanto probablemente tengamos que echar cuentas y dependiendo del número de servidores optar por una infraestructura clásica o virtual.

La Tabla 2-1, a continuación, muestra ventajas frente a desventajas, no son todas, pero son las más llamativas. La tabla permite enfrenar unas a otras:

Ventajas	Desventajas
Reduce el coste en <i>hardware</i> , en espacio físico, el consumo eléctrico, la necesidad de refrigeración y la instalación de las redes cableadas.	Aumenta el coste del <i>software</i> . Impide o dificulta la utilización de <i>hardware</i> especializado o no soportado. Penaliza posibilidades del <i>hardware</i> como la aceleración de vídeo.
Mejora la seguridad: - Facilita los planes de continuidad mejorando la recuperación de desastres. - Aumenta la disponibilidad.	Perjudica la seguridad: - Adicionalmente hay que planificar la seguridad de la capa de virtualización. - Puede aumentar la amenaza sobre la confidencialidad. - Puede perjudicarse la integridad de los sistemas por la mayor dependencia del <i>software</i>.
Facilita el mantenimiento: - Mediante consolas de gestión se mantienen las MV,s de forma centralizada evitando acceder a cada una de ellas independientemente. - Facilita la implementación de cambios, las pruebas y las migraciones.	Desperdicio de recursos (almacenamiento, ciclos de procesador, memoria). Aumenta la necesidad y dependencia de personal especializado. Aumenta la complejidad.
Reduce los tiempos de despliegue e instalación. Elimina instalación física (cableado, electricidad, etc.)	Penaliza el rendimiento de las MV,s frente a las físicas.
Aumenta la flexibilidad de la infraestructura: Balanceo de carga entre MV,s	Dificulta la administración de los recursos: - Implica planeamiento del reparto de recursos entre MV,s. (evitar desperdicio de recursos). - El <i>host</i> es un punto de fallo único. Se puede gestionar gracias a las características de alta disponibilidad (HA). - Dependencia frente al SO del <i>host</i>.

Tabla 2-1. Ventajas y desventajas de la virtualización.

Las infraestructuras de virtualización se planifican para entornos de aplicación específicos, de acuerdo con sus necesidades y requisitos, en las que se va a intentar balancear los pros y contras mediante análisis de coste-beneficio, si los costes asociados a la virtualización superan a los beneficios, o al coste de la misma infraestructura sin virtualizar, nuestra decisión debería ser no virtualizar.

La instalación de una nube privada para ofrecer IaaS sigue el mismo criterio. La instalaríamos si el coste-beneficio sobre la instalación de sistemas ya sea físicos o virtuales mejora el balance a favor de la primera.

2.3 OpenStack

La información vertida sobre este apartado se ha obtenido de la página web oficial del proyecto OpenStack [12].

OpenStack® es marca registrada de “OpenStack Foundation”. Aunque el código OpenStack® está disponible bajo la licencia de software de código abierto de Apache, usar el nombre y el logotipo de OpenStack® está restringido por la Ley. “*The OpenStack project is provided under the Apache 2.0 license*” [12].

Nace a raíz de una colaboración entre la NASA y el proveedor de computación en la nube Rackspace.

Entre mayo y junio del año 2010 Rackspace liberó su proyecto de computación en la nube *Cloud Files*, un almacenamiento como servicio (SaaS), al mismo tiempo que la NASA liberaba Nebula un proyecto de IaaS. Ambos pasaron a ser proyectos de código abierto (*open source*) y se convirtieron respectivamente en los componentes básicos del recién nacido OpenStack: Nova y Swift.

Creada la OpenStack Foundation, grandes empresas como AMD, Intel, Cisco, Dell, HP, IBM, Red Hat, Canonical o Yahoo se sumaron al proyecto a través de la fundación.

2.3.1 Que es OpenStack [13]

OpenStack es un sistema operativo de *cloud* que permite a los administradores del sistema gestionar una gran cantidad y variedad de recursos informáticos, de proceso, de almacenamiento y de red en un CPD.

OpenStack se administra a través de un panel de control web, el mismo que permite a los usuarios aprovisionarse de recursos como servicio a través de una interfaz ligera basada en un navegador web.

El OpenStack Marketplace proporciona productos y servicios relacionados con OpenStack, como distribuciones, sistemas embebidos (*appliances, hardware y software* instalado), nubes públicas, consultoría y formación.

La Figura 2-5 muestra como todos los elementos se enlazan y a través de la API de OpenStack son accedidos por las aplicaciones de clientes (APP), con el *Dashboard*⁴ (GUI) y con las herramientas de monitorización.

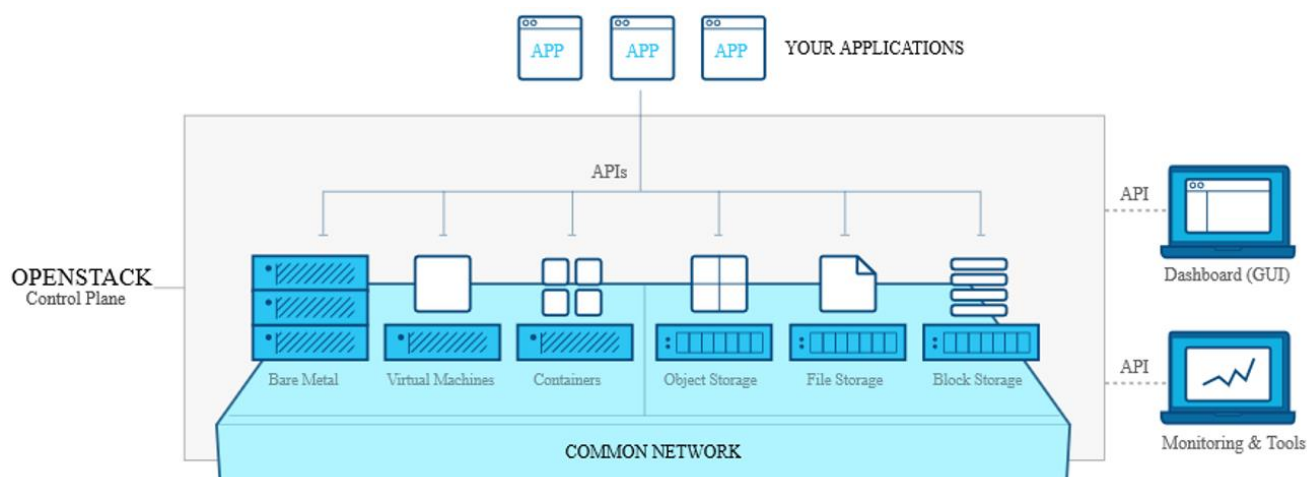


Figura 2-5. Ecosistema de OpenStack (figura simplificada) [13]

⁴ Componente Horizon.

Cada versión de OpenStack se desarrolla periódicamente en un ciclo que dura seis meses aproximadamente. El siguiente es un ejemplo del ciclo de una de las versiones (la última, denominada Stein) de OpenStack:

- Entre el 10 y el 14 de septiembre de 2018 tuvo lugar la reunión de equipos de proyecto en Denver, Estados Unidos.
- Desde el 13 al 15 de noviembre de 2018 se llevará a cabo la presentación de Stein en la Cumbre de OpenStack de Berlín
- El 10 de abril de 2019 tendrá lugar el lanzamiento de Stein.

La última versión de OpenStack en el momento de la escritura de este trabajo es Rocky, es la décimo octava versión del producto. Normalmente se encuentran mantenidas (bajo soporte) las cuatro últimas versiones, por tanto el producto recibirá actualizaciones durante dos años (aunque los periodos no son exactos). Desde la versión Newton hacia atrás todas están descontinuadas.

Versión	Estado	Lanzamiento	Fin ciclo de vida	Descontinuada
Stein	Development	2019-04-10 (estimada)	2019-04-10 (estimada)	
Rocky	Mantenida	2018-08-30	2019-02-23 (estimada)	
Queens	Mantenida	2018-02-28	2019-08-25 (estimada)	
Pike	Mantenida	2017-08-30	2019-03-03 (estimada)	
Ocata	Mantenida	2017-02-22	2018-08-27 (estimada)	
Newton	Fin ciclo vida	2016-10-06		2017-10-25

Tabla 2-2. Historial de versiones de OpenStack, desde la que se encuentra en desarrollo hasta la primera descontinuada [14]

2.3.2 Los componentes del proyecto OpenStack

OpenStack no es sí mismo un único proyecto, lo componen varios proyectos más o menos independientes (cada uno va a su ritmo serializándose en fechas distintas y en su desarrollo trabajan comunidades diferentes) pero relacionados, de manera que cada uno se focaliza en controlar unos recursos de computación determinados, la capacidad de proceso, el almacenamiento, la red, etc. Todos se conectan a un centro de datos y se administran mediante un portal o panel de control, y que además proporciona a los usuarios una interfaz web.

A medida que han ido sucediéndose las versiones de OpenStack ha ido creciendo el número de componentes que lo integran, así en la primera versión, Austin de octubre de 2010, solo estaban presentes Nova y Swift, y en la última estable, Rocky de agosto de 2018, los componentes son los que se muestran en la tabla 2-3.

Servicio	Componente
Compute	Nova; Zun; Qinling
Bare Metal	Ironic; Cyborg
Storage	Swift; Cinder; Manila
Networking	Neutron; Octavia; Designate
Shared Services	Keystone; Glance; Barbican; Karbor; Searchlight
Orchestration	Heat; Senlin; Mistral; Zaqr; Blazar; Aodh
Workload Provisioning	Magnum; Sahara; Trove
Application Lifecycle	Masakari; Murano; Solum; Freezer
API Proxies	EC2API
Web Frontend	Horizon
Monitoring Tools	Ceilometer; Panko; Monasca
Optimization/policy tools	Watcher; Vitrage; Congress; Rally
Billing/Business Logic	Cloudkitty
Multi-Region Tools	Tricircle
Containers	Kuryr; Fuxi
NFV	Tacker
Swift add-ons	Storlets
Neutron plug-ins	Dragonflow

Tabla 2-3 Componentes en la versión Rocky [15]

A continuación los trataremos brevemente, no los vamos a presentar todos, solo los más relevantes.

Servicio de computación (*Compute*): Nova

Por servicio de computación entendemos el que maneja los recursos y se encarga de los cálculos del sistema operativo *Cloud*, ejecutando varias instancias (virtuales que se sirven como servicio) de tipos diferentes. El servicio puede ejecutarse en varios servidores del clúster.

Nova es uno de los tres controladores de la estructura de OpenStack, y su desarrollo empezó con la primera versión, Austin, por lo que es de los dos componentes más veteranos.

Como controlador de estructura *cloud computing* se encarga de la gestión y automatización de los *pools* o agrupaciones de recursos, siendo el componente principal de la prestación de IaaS.

Nova se relaciona o interactúa con la tecnología de virtualización completa, con los hipervisores de tipo 1 Xen, Microsoft Hyper-V y VMware vSphere, y de tipo 2 con KVM (a día de hoy completamente integrado en Linux y que distribuciones como Ubuntu definen como *bare-metal*).

También interactúa con la virtualización de sistema operativo, con los contenedores Linux como LXC. Toda la información sobre Nova se encuentra en la web del proyecto OpenStack desde la que se accede a la wiki del proyecto [16].

Aunque Nova puede ejecutar contenedores, Zun implementa un servicio computacional más específico y que soporta más cantidad de tecnologías de contenedor. Quinling proporciona un servicio orientado a manejar funciones que se incorporarían a los puntos de servicio satisfechos a los usuarios.

Los servicios de *bare metal*, son podríamos decir los más próximos al *hardware*, es como se denominaban a los hipervisores tipo 1 (*unhosted* o *bare metal*), por tanto se encargan de proporcionar en OpenStack un servicio de virtualización completa con una relación más directa con el *hardware*, así Ironic permite instanciar, a demanda, como servicios (con sus dependencias), *hardware* virtual, máquinas virtuales y contenedores. Cyborg es un proyecto muy nuevo que ha sido lanzado en Rocky y trabaja como un acelerador de *hardware*.

Servicio de Almacenamiento de Objetos (*Object Storage*): Swift

Swift es el componente que ejerce de almacén de objetos. El componente crea una nube de almacenamiento como pueda ser DropBox.

Como sistema de almacenamiento proporciona redundancia y garantiza la integridad manejando la replicación de los datos entre los discos disponibles. Los archivos y objetos se guardan en varios discos repartidos por los servidores que alojan el ecosistema OpenStack (y que conforman un clúster) y basta con añadir más discos a un servidor, o una unidad específica de almacenamiento (típicamente una SAN, pero puede ser incluso un NAS), o más servidores al clúster para que Swift los utilice, haciendo el sistema de almacenamiento escalable horizontalmente. Si se produce un fallo en un disco o un servidor, OpenStack replica su contenido desde los otros servidores (nodos) y almacenará los datos en nuevas ubicaciones.

Swift proporciona un sistema de almacenamiento que el servicio de computación utiliza como plantilla para instancias de máquinas virtuales.

Junto a Nova este componente es el otro más veterano y está presente en OpenStack desde el principio.

Toda la información sobre Swift se puede consultar en la referencia [17].

Servicio de almacenamiento de bloques (*Block Storage*): Cinder

Cinder proporciona dispositivos de almacenamiento a nivel de bloque persistentes, son como discos duros (volúmenes lógicos) que se asignan a las máquinas virtuales o a usuarios, y por tanto es totalmente diferente a Swift que es un almacenamiento dinámico.

Cinder permite crear, asignar y desconectar (o desasignar) los volúmenes a los servidores, para lo que se integra con Nova. Por otra parte desde Horizon (el *dashboard* que se presenta más adelante) los usuarios pueden gestionar volúmenes como espacio propio de almacenamiento.

Una utilidad muy interesante es que Cinder puede gestionar no solo el almacenamiento que le proporciona el *hardware* donde se instala (que gestiona directamente el SO Linux del servidor), además puede utilizar almacenamiento en Internet, en las nubes públicas.

Este almacenamiento de bloques es, frente al de objetos, el apropiado para soportar las bases de datos o los sistemas de archivos expandibles, en los que el rendimiento es un parámetro importante. Además al tratarse de volúmenes es fácil mediante *snapshots* (instantáneas) realizar copias de seguridad que pueden restaurarse y utilizarse para crear nuevos volúmenes.

Toda la información sobre Cinder se puede consultar en la referencia [18].

Manila es un componente que gestiona el acceso a sistemas de archivos compartidos o distribuidos de manera coordinada.

Servicios de red (Networking): Neutron

Neutron es para la gestión de redes IP. Proporciona modelos para diferentes necesidades como direccionamiento estático y dinámico, segmentación mediante VLAN, para grupos de máquinas, aplicaciones y para usuarios. Los usuarios pueden crear sus propias redes conectando sus servidores y dispositivos a una o más y pueden controlar el tráfico.

Neutron se encarga de que la red no tenga cuellos de botella, por ejemplo reserva direcciones IP que asigna dinámicamente redirigiendo tráfico a cualquiera de sus recursos informáticos, lo que supone supervivencia frente a desastres.

Neutrón utiliza la tecnología de redes definidas por software (SDN) lo que le permite, además de ser notablemente escalable, la extensión de servicios adicionales como *firewalls*, IDS, balanceadores de red y concentradores VPN.

En esta línea hay dos componentes, Octavia que permite crear IDS en las redes y Designate que implementa servidores DNS.

Toda la información sobre Neutron se puede consultar en la referencia [19].

Servicios compartidos. El servicio de identidad: Keystone, y el servicio de imágenes: Glance

Keystone es un directorio centralizado de usuarios para gestionar el acceso a los demás servicios de OpenStack. Proporciona servicios de autenticación en todo el sistema operativo de la nube (que implementa OpenStack) y se puede integrar con los servicios de directorio existentes en la plataforma física o virtual sobre la instalamos nuestro ecosistema de OpenStack para producir nuestra nube, como LDAP. Puede autenticar mediante nombre de usuario y contraseña, *tokens*, etc.

Keystone proporciona además una lista de todos los servicios existentes en la nube de OpenStack para que los usuarios y las aplicaciones puedan consultar qué recursos les son accesibles.

Glance proporciona servicios de descubrimiento, de inscripción y de entrega de imágenes (tipo ISO, y también directamente discos de máquinas virtuales) que se encuentran almacenadas a modo de plantilla. La API de Glance proporciona una interfaz REST estándar para consultar información sobre las imágenes de disco almacenadas y permite a los clientes transmitir las imágenes a nuevos servidores.

También se puede utilizar para almacenar y catalogar un número ilimitado de copias de seguridad.

Toda la información sobre Keystone se puede consultar en la referencia [20] y sobre Glance en la [21].

Servicios de orquestación⁵: Heat; y de telemetría: Ceilometer

Heat es un servicio que produce la dirección necesaria para que los recursos de infraestructura sobre los que se instala la nube de OpenStack trabajen coordinadamente y sincronizadamente para que funcionen múltiples aplicaciones compuestas en la nube utilizando plantillas.

Heat trata las plantillas (archivos de texto) como código a través de una API REST OpenStack nativa y una API de consultas compatibles con CloudFormation.

Heat también proporciona un servicio de autoescalado que se integra con los servicios de telemetría de OpenStack, Ceilometer, por lo que puede incluir un grupo de escala como recurso en una plantilla.

Ceilometer proporciona un único punto de contacto para que todos los componentes de OpenStack (incluidos los que se desarrollen en el futuro) puedan tarificar por su uso a los clientes, o puedan establecerse diferentes arcos por uso.

⁵ Orquestación es a mi juicio una mala traducción pero que se encuentra en demasiadas publicaciones. Quiere hacer referencia al trabajo colaborativo y sincronizado, y por tanto debiera traducirse por funcionamiento coordinado y sincronizado.

Las medidas y resultados de los contadores es trazable y auditable y se entrega a agentes que recolectan los datos y que son independientes de todo el sistema.

Toda la información sobre Heat se puede consultar en la referencia [22] y sobre Ceilometer en la [23].

Servicios de trabajo. Base de datos: Trove

Bajo la denominación de servicios de trabajo se agrupan utilidades que proporcionan capacidades de aplicación para el ecosistema.

Trove [24] es una base de datos que funciona como un servicio de aprovisionamiento de motores de bases de datos relacionales y no relacionales (DBaaS).

Sahara [25] es un marco de trabajo de BigData (por ejemplo, si en una instalación de nube privada lo que se pretendiera es procesamiento de BigData usaríamos Sahara). Magnum [26] es un aprovisionador de contenedores para aplicaciones (parecido a Docker).

Portal Web o Frontend: Horizon

También denominado *Dashboard* de OpenStack. Horizon proporciona a los administradores y usuarios una interfaz gráfica para el acceso, la provisión y automatización de los recursos basados en la nube. Contiene todas las herramientas necesarias para la gestión y monitorización de los recursos aprovisionados.

Horizon proporciona además una API nativa de OpenStack o una API de compatibilidad EC2 para automatizar el acceso o construir herramientas para gestionar recursos.

La Figura 2-6 muestra los componentes que actualmente se encuentran continuados en el en el ciclo de vida de OpenStack. Los resaltados en **negrita** implementan los servicios core de la suite.

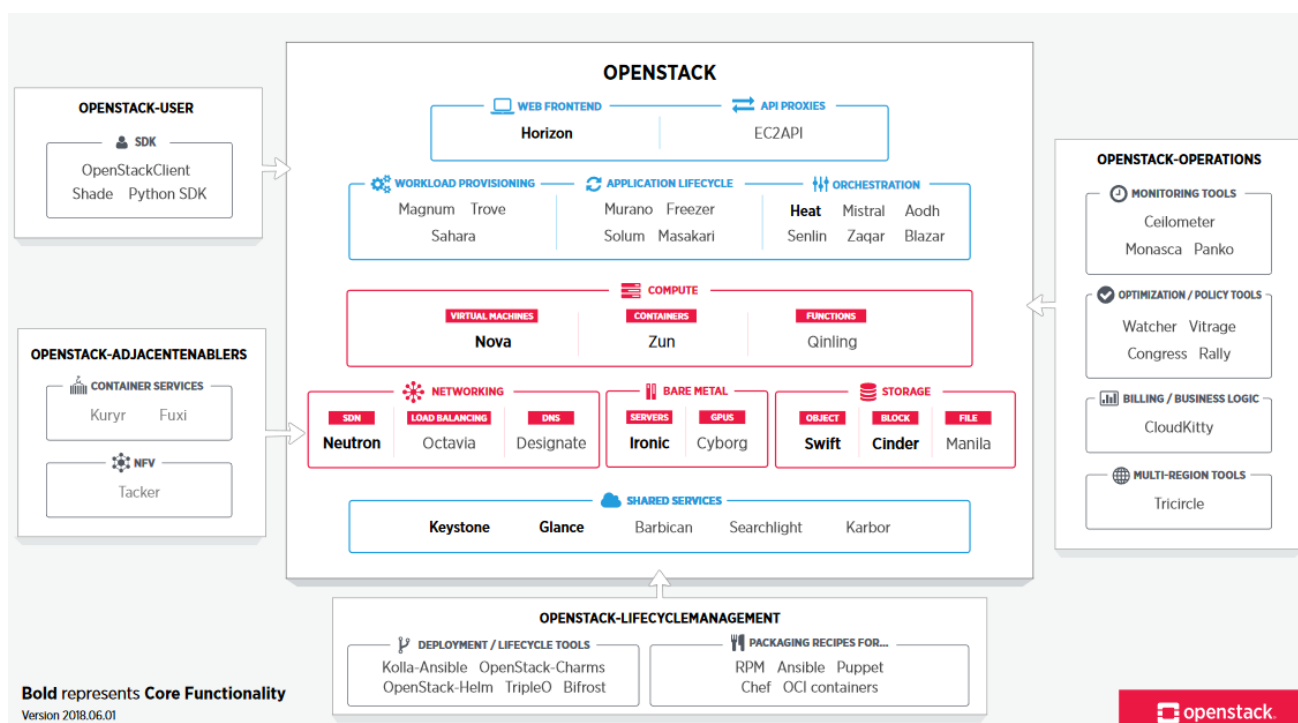


Figura 2-6. Mapa de los componentes de OpenStack [27]

2.3.3 Distribuciones de OpenStack

OpenStack, a la vista de sus componentes, resulta muy modular y por supuesto puede instalarse ya sea sobre un conjunto de máquinas físicas, (incluso sobre un ordenador portátil) como sobre una infraestructura virtual.

OpenStack requiere Linux como sistema operativo de base, y podríamos instalar cada componente por separado, y podríamos arreglarlo o customizarlo de una manera particular pues el código fuente está disponible, claro, luego tendríamos que compilarlo y desplegarlo, y ¿cuánto tiempo hace ya que nos hemos olvidado de compilar los *kernels* y las aplicaciones?, por eso las distribuciones son probablemente la mejor de las opciones.

Damos un repaso de las distribuciones a mi juicio más interesantes:

Red Hat OpenStack Platform [28]

Para un desarrollo serio y empresarial probablemente la distribución de Red Hat fuera la elección, de pago, más acertada. Su distribución es altamente estable y su soporte es de lo mejor.

Red Hat esponsoriza la comunidad RDO que lleva varios proyectos relacionados con OpenStack, y si lo que se quiere es una instalación lo más actualizada posible, es la opción, pero al igual que OpenStack sigue un ritmo demasiado acelerado para un entorno empresarial.

El proyecto de RDO de implementación de OpenStack sobre Red Hat es gratuito, pero como dicen en la misma comunidad si se busca soporte a nivel empresarial o información sobre certificación de partners, Red Hat también ofrece Red Hat OpenStack Platform.

Es decir, puede instalarse una versión gratuita de OpenStack sobre Red Hat, pero las muchas opciones de personalización y el apoyo técnico que ofrece Red Hat OpenStack Platform totalmente soportado merecen la pena para un entorno empresarial, donde las actualizaciones y la formación resultan vitales.

Por otra parte RDO también ofrecen implementaciones OpenStack en CentOS y Fedora (ambas distribuciones derivadas del Red Hat Linux original), y lleva adelante varios proyectos:

- PackStack: Es un instalador que nos permite crear una prueba de concepto de nube⁶ implementando OpenStack en un dispositivo *all-in-one*, de cara a desarrolladores, fundamentalmente, y a quienes quieren probar el sistema *cloud*. Posteriormente se pueden agregar más nodos a la nube implementada. PackStack funcionará en instalaciones sobre CentOS y sobre Red Hat Enterprise Linux (RHEL).
- TripleO Quickstart: es una herramienta de implementación rápida y de administración de OpenStack sobre una máquina *all-in-one*, igual que PackStack. Está totalmente probada sobre CentOS 7, pero en principio debe poder funcionar sobre cualquier distro RHEL 7 o superior.
- TripleO: a diferencia del anterior TripleO Quickstart, permite desplegar una nube en producción para lo que utiliza el conjunto de herramientas del mismo OpenStack (Keystone, Horizon, Nova...). TripleO se orienta a entornos con muchos servidores, ya que para despliegues más limitados existen otras herramientas de despliegue [29]. TripleO facilita la instalación, despliegue y gestión de la nube creada en un conjunto de servidores físicos, sobre los que operan un entorno virtual.

⁶ Lo cual concuerda muy bien para los propósitos de este trabajo, aunque finalmente no será la opción elegida.

Ubuntu OpenStack [30]

Ubuntu, lanzada en 2004, es una distribución de Linux basada en Debian, y es probablemente la más instalada por usuarios medios y como sistema operativo de base en servidores web, en todo el mundo (con cuota de mercado de un 17% tendente a crecer) [31].

La empresa británica Canonical Ltd. se hizo cargo de Ubuntu en 2009 con el lanzamiento del Centro de Software de Ubuntu.

Canonical como patrocinador de Ubuntu lo mantiene, y estando su modelo de negocio asociado a servicios vinculados a este sistema operativo, ha lanzado versiones específicas:

- Ubuntu desktop, orientado a equipos cliente.
- Ubuntu Server está orientado a servidores.
- Ubuntu Business Desktop Remix, orientado a empresas.
- Ubuntu TV, para televisores.
- Ubuntu Tablet, para tablets.
- Ubuntu Phone y Ubuntu for Android, dirigidas a la telefonía móvil....

Canonical publica versiones de Ubuntu cada 6 meses (sincronizadas con OpenStack, lo que demuestra la implicación de Canonical con este proyecto) y les da soporte (actualizaciones de seguridad, parches y actualizaciones de programas incluidos) durante 9 meses.

Además, dirigido al mundo más empresarial y tanto para servidores (server) como para clientes (desktop), ofrece versiones *Long Time Support* (LTS) cada dos años (los años pares), y estas reciben soporte durante 5 años a partir de la fecha del lanzamiento. En la actualidad la versión soportada más antigua (hasta abril de 2019) es la 14.04 lanzada en abril de 2014, y la más moderna es la 18.04 lanzada en abril de 2018 y cuyo soporte está previsto hasta abril de 2023.

También existen muchas variantes de Ubuntu, por ejemplo considerando diferentes escritorios, como Kubuntu (escritorio KDE) o Ubuntu MATE (escritorio MATE), o para entornos escolares como Edubuntu, y para otros propósitos.

Y en este caldo de cultivo tratamos de Ubuntu OpenStack, que no es una distribución, sino más bien un servicio, *“Canonical diseña, construye, opera y admite nubes privadas OpenStack en Ubuntu. Entendemos la importancia de la seguridad, la estabilidad, el rendimiento y la eficiencia económica para la infraestructura de nube privada”* [30].

Dentro de los servicios que ofrece tiene varias variantes desde la gestión completa hasta la obtención del producto y su instalación soportada por un tiempo inicial, pasando por migraciones a OpenStack sobre Ubuntu desde otras plataformas, como desde VMware, respecto la cual promete lo mismo por un 20% del precio.

Canonical nos ofrece Ubuntu OpenStack gratis, pues su negocio está en el servicio, y comparando con Red Hat su soporte puede ser igualmente bueno, aunque no abandere la excelencia (de acuerdo con lo recogido en Internet), de hecho no tiene una versión del producto específica.

Para instalar una nube privada sobre un clúster de servidores físicos se realiza una instalación denominada *Metal as a Service* (MaaS) [32] y el proceso viene guiado por una utilidad denominada conjure-up [33]. Canonical nos ofrece soporte para resolver nuestros problemas desde la instalación hasta la gestión y administración, y es por eso por lo que cobra.

Nos ofrecen dos opciones para la instalación [34], una sobre un clúster de servidores y otra para una instalación en un ordenador (*all-in-one*), en ambos casos prevista para instalaciones previas de una versión 16.04 LTS o 18.04 LTS (actualmente, pues arranca de versiones más antiguas).

La instalación sobre un clúster puede llegar a incluir alta disponibilidad y utiliza la virtualización KVM como *bare metal* mientras que sobre un equipo único está pensada para desarrolladores y usan virtualización LXD, *Linux Container Daemon*, (es una particularización de LXC), para crear varios contenedores de máquina en el PC donde se ejecuten los diversos servicios de OpenStack, imitando un clúster real.

VMware Integrated OpenStack [35]

VMware tiene su propio producto *cloud* que se integra con vSphere, VMware vCloud, pero finalmente ha sucumbido a OpenStack pues ha facilitado instalar OpenStack dentro de su infraestructura vSphere.

Es necesario tener instalado un clúster de servidores ESXi como mínimo versión vSphere Enterprise Plus o vSphere Estándar con NSX Advanced, y sobre el hipervisor se puede instalar OpenStack gratuito, siguiendo las instrucciones de VMware [36].

A pesar de que se pueda instalar OpenStack gratuito, al obligarnos a pagar las licencias del vSphere, unido a los costes por soporte en OpenStack, la instalación final resulta muy cara, (como dice Canonical hasta cinco veces más cara).

Mirantis OpenStack [37]

Su patrocinador es Mirantis Inc. Mirantis OpenStack es una distribución bajo la herramienta de desarrollo y administración Fuel. En foros se dice que es una de las mejores distribuciones OpenStack a día de hoy.

Para probarla podemos instalar una versión sobre máquinas virtuales, que implica virtualizar sobre virtualizado (*nested virtualization*) para lo que podemos seguir una guía como la de la referencia [38], ofrecida por “dbigcloud”, es fácil de instalar y cuenta con *plugins* propios para la gestión y una comunidad bastante activa detrás.

Puede instalarse sobre cualquier Linux como SO, y es una de las distribuciones que cuenta con la alianza de SUSE, que por otra parte tiene su propia línea de producto SUSE OpenStack Cloud.

IBM Cloud Manager with OpenStack

IBM ha creado su propia distribución. Permite el despliegue en KVM, ESXi, HyperV y Xen. Ofrece servicios profesionales a empresas y su despliegue es complicado.

Oracle OpenStack

Está basada en su propia distribución de Linux (Oracle Linux) y emplea Docker para el despliegue. Su orientación es a desarrolladores pensando en que estos utilicen los productos de Oracle Inc. En sus trabajos.

Otras distribuciones OpenStack

Además de las indicadas también existen otras muchas distribuciones que pueden consultarse en el Marketplace de OpenStack, en el siguiente enlace <https://www.Openstack.org/marketplace/distros/> [39].

2.4 Securización, bastionado o *hardening*.

2.4.1 El entorno regulatorio

No se trata de disertar largamente sobre leyes y normativas, basta decir que podríamos dividir la información de la Administración en dos ámbitos, la clasificada y la que no lo está.

La información clasificada, si es del nivel de Reservado o Secreto (clasificaciones nacionales) está sujeta a la Ley 9/1968, modificada por la Ley 48/78, sobre Secretos Oficiales, y su Decreto de desarrollo 242/1969.

La Ley 11/2002, de 6 de mayo, reguladora del Centro Nacional de Inteligencia (CNI), asigna a este Organismo “*Velar por el cumplimiento de la normativa relativa a la protección de la información clasificada*” [40]. A estos efectos el Secretario de Estado Director del CNI es el responsable último de proteger la información clasificada nacional (niveles de Reservado y Secreto).

Por otra parte, “*En España, mediante Acuerdo de Consejo de Ministros de 21 de mayo de 2012, basado en los precedentes acuerdos de Consejo de Ministros de 1982, 2002 y 2005, se designa Autoridad Nacional de Seguridad para la protección de la información clasificada originada por la OTAN, la UE y la ESA a los Ministros de la Presidencia, de Defensa y de Asuntos Exteriores y de Cooperación, de forma conjunta, quienes por Acuerdo de Consejo de Ministros, delegan dicha función en el Secretario de Estado Director del Centro Nacional de Inteligencia*” [40].

También, toda información clasificada en el ámbito de organizaciones como OTAN, UE o la Agencia Espacial Europea (ESA) que se clasifica con un nivel equivalente a los de Reservado y Secreto nacional, está sujeta a iguales medidas de protección de la mano de diferentes tratados internacionales suscritos por España ya que dichos tratados imponen los mismos niveles de seguridad.

Es decir, visto lo anterior, el Secretario de Estado Director del CNI se constituye como la Autoridad Nacional Para la Protección de la Información Clasificada (ANPIC), sea ésta de ámbito nacional o de las organizaciones internacionales.

Partiendo de la existencia de esta Autoridad, la ANPIC, otras leyes subordinan a su acción la protección de la información clasificada en poder de sectores o agentes sociales, como por ejemplo la Ley 24/2011, de 1 de agosto, de contratos del sector público en los ámbitos de la defensa y de la seguridad, que obliga a las empresas a regirse por las disposiciones que dicte la ANPIC cuando por contrato tengan necesidad de acceder a información clasificada.

La ANPIC ha dictado una normativa básica para la protección de la información clasificada, independientemente de si es nacional o de organizaciones internacionales con las que exista acuerdo al efecto, concretamente las normas de la referencia [40], y que como se dice en dicho documento “*Estas Normas de la Autoridad Nacional para la Protección de la Información Clasificada constituyen el desarrollo normativo con el que se regula el manejo de la información clasificada en España y se da respuesta a las obligaciones contraídas en el ámbito internacional por nuestro país con otros estados u organizaciones internacionales*”⁷ [40].

En España además de los niveles de clasificación de información establecidos por la Ley de Secretos Oficiales, en determinados ministerios hay otros niveles de clasificación, por ejemplo en el de Defensa están el de Confidencial y el de Difusión Limitada, y estas son consideradas materias de reserva interna. Estos otros niveles de reserva interna se extienden al ámbito de las Administraciones Públicas por las Normas de la ANPIC.

Las organizaciones internacionales y muchos países con los que existe “acuerdo⁸” en suma tienen los mismos cuatro niveles de clasificación, desde Secreto a Difusión Limitada, la tabla muestra los niveles de equivalencia entre clasificaciones:

⁷ Por organizaciones internacionales en las Normas de la ANPIC se contemplan la OTAN, UE y ESA, pero dentro del ámbito internacional se consideran además los países con los cuales España ha suscrito un acuerdo bilateral.

⁸ Normalmente los acuerdos son negociados por la ANPIC.

Ámbito	Máximo nivel ←.....→ Mínimo.			
España	SECRETO	RESERVADO	CONFIDENCIAL	DIFUSIÓN LIMITADA
OTAN	COSMIC Top Secret	NATO Secret	Nato Confidential	Nato Restricted
UE	Top Secret	Secret	Confidential	Restricted
ESA	ESA Top Secret	ESA Secret	ESA Confidential	ESA Restricted

Tabla 2-4. Equivalencias entre grados de clasificación de la información.

¿Y la información que no está clasificada? Hay que decir que la información Confidencial o Difusión Limitada no está bajo el paraguas de la Ley de Secretos Oficiales y por tanto puede estar sujeta a otras leyes.

La Ley 19/2013, de 9 de diciembre, de transparencia, acceso a la información pública y buen gobierno, establece el acceso universal (de todo el mundo) a la información pública⁹, en los términos previstos en la Constitución, y con ciertos límites basados en la seguridad nacional, la defensa y la seguridad pública. También dispone esta Ley que “*Se regirán por su normativa específica, y por esta Ley con carácter supletorio, aquellas materias que tengan previsto un régimen jurídico específico de acceso a la información*”. Tal es el caso de la información sujeta a la protección de datos de carácter personal, o la de infraestructuras críticas o incluso la que se ampara por el Esquema Nacional de Seguridad (ENS).

En suma, hay información sujeta a regímenes de protección fuera del ámbito de la ANPIC, por ejemplo, de los datos de carácter personal se ocupa la Agencia Española de Protección de Datos, velando por la aplicación del Reglamento Europeo y la normativa propia, pero cuando se trata de Administraciones Públicas el ENS como veremos a continuación extiende el brazo de la ANPIC.

La Ley 11/2007, de 22 de junio, de acceso electrónico de los ciudadanos a los Servicios Públicos estableció el ENS.

Posteriormente el Real Decreto 3/2010, de 8 de enero, desarrolla la política de seguridad en la utilización de medios electrónicos en el ámbito de aplicación del ENS. En este RD se establece en el artículo “29. *Guías de seguridad. Para el mejor cumplimiento de lo establecido en el Esquema Nacional de Seguridad, el Centro Criptológico Nacional, en el ejercicio de sus competencias, elaborará y difundirá las correspondientes guías de seguridad de las tecnologías de la información y las comunicaciones*”. También se asigna al CCN a través de su Centro de Respuestas (CERT) toda la gestión de incidentes y su normalización. Siendo el Centro Criptológico Nacional (CCN) un organismo superpuesto al CNI, su Director es el mismo Secretario de Estado del CNI ‘y del CCN’, que no es otro que la ANPIC, que por medio de sus Guías de la serie 800 maneja como se ha de establecer la seguridad de la información para cumplimiento del ENS.

Cumplir con el ENS garantiza por otra parte una buena seguridad de cara a la configuración de los sistemas de información en los que se maneje datos de carácter personal de los ciudadanos e incluso relativos a protección de infraestructuras críticas.

El círculo está cerrado por tanto en torno a los sistemas que manejan la información de la Administración del Estado, la información que está clasificada queda sujeta a las Normas de la Autoridad Nacional para la protección de la información clasificada [40] y la información que no es clasificada puede ser accedida libremente o no, en función de lo que disponga el organismo dueño de la misma, y tanto su seguridad como este aspecto mismo está sujeto a otras normativas, siendo el ENS la norma a mayores que debe cumplir todo organismo público cuando no se trata de información clasificada. Y

⁹ En los términos establecidos en la Constitución, y que precisamente desarrolla esta Ley.

todas las medidas de seguridad que hay que instalar las determina la misma Autoridad, la ANPIC, ya sea por un camino o por otro.

2.4.2 Guías de seguridad.

Partiendo de lo anterior, la ANPIC establece que las guías de seguridad aplicables para la seguridad de los sistemas de información y las tecnologías relacionadas son las editadas por el CCN-CERT, y que se conocen como las Guías CCN-STIC.

Todo sistema que maneje información clasificada de grado Difusión Limitada o equivalente y superior (Confidencial, Reservado o Secreto) debe resultar previamente acreditado para ello. La principal guía del proceso de acreditación es la CCN-STIC-101, de julio de 2016 [41]. Esta guía define el procedimiento de acreditación (de acuerdo con la Política de Seguridad) para sistemas clasificados.

La guía CCN-STIC-205 dice en su punto 6. PRINCIPIOS GENERALES, en el párrafo 34 “*Cuando existan productos en el mercado (COTS), se preferirán a desarrollos específicos*” [42].

Los requisitos de seguridad STIC (los propiamente determinados para los sistemas de información, *hardware* y *software*) se especifican en la guía CCN-STIC-301 [43], que es de difusión limitada, por lo que en lo sucesivo durante este trabajo, incluso cuando esta guía fuera la fuente de información para argumentar cualquier medida de seguridad, no la referenciaré más.

Las guías del CCN de la serie 400 son guías generales, varias de las cuales detallan conceptos aplicables en el bastionado de sistemas, entre otras:

CCN-STIC-400. MANUAL STIC.

CCN-STIC-408. SEGURIDAD PERIMETRAL. CORTAFUEGOS.

CCN-STIC-412. REQUISITOS DE SEGURIDAD DE ENTORNOS Y APLICACIONES WEB.

CCN-STIC-415. IDENTIFICACIÓN Y AUTENTICACIÓN ELECTRÓNICA.

CCN-STIC-419. CONFIGURACIÓN SEGURA CON IPTABLES.

CCN-STIC-422. DESARROLLO SEGURO DE APLICACIONES WEB.

Las guías del CCN de la serie 500 son guías de seguridad específicas para productos Microsoft, y las de la serie 600 de otros productos. De estas series pueden ser aplicables varias. De las de la serie 500 aplicarán a las imágenes que pudieran ser puestas a disposición de los usuarios en nuestra *cloud* como IaaS, pues las MV,s que aprovisionaran deberían estar securizadas aplicándolas, por ejemplo si se instala un servidor W2012 para SQL SERVER aplicarían sucesivamente las guías de ambos productos.

De la serie 600 hay guías que entran dentro del ámbito de este TFM, por ejemplo: CCN-STIC-612. SECURIZACIÓN DE SISTEMAS BASADOS EN DEBIAN (AGO/2014) [44].

Hay otras guías de securización de distros de Linux, pero son demasiado antiguas, del año 2006 por lo que no las consideraré. Y hay guías de securización de *Firewalls*, *IDS*, *Proxies*, *Apache*, *Tomcat*, etc., la mayoría antiguas y desfasadas pero de las que se pueden extrapolar medidas.

Para otros sistemas de la Administración que manejen información sin clasificar les es de aplicación el Esquema Nacional de Seguridad (ENS), según se establece en las Leyes 39/2015, de 1 octubre, (de Procedimiento Administrativo Común a las Administraciones Públicas) y la 40/2015, de 1 octubre, (de Régimen Jurídico del Sector Público).

Todas las guías de seguridad para cumplimiento del ENS se encuadran en la serie 800 de las guías CCN-STIC, y de ellas alguna es precisamente para entornos *cloud*: CCN-STIC-823. UTILIZACIÓN DE SERVICIOS EN LA NUBE. [45]

No siempre se cuenta con una guía CCN-STIC para el robustecimiento de determinados sistemas de información, cuando eso no ocurre hay que intentar buscar guías aplicables, por ejemplo en las siguientes fuentes:

- CERT, nacionales y de referencia¹⁰:
 - INCIBE-CERT [46], es un CERT de referencia para las empresas y los ciudadanos y trabaja en diversas colaboraciones con el Centro Nacional de Protección de Infraestructuras y Ciberseguridad (CNPIC) del Ministerio del Interior. Las guías del INCIBE-CERT se encuentran en la referencia [47]. Por el momento no hay guías aplicables en este TFM, pero podría haberlas en un futuro.
 - El Mando de Ciberdefensa (MCCD) es el CERT de referencia para el MDEF. No tiene guías pero el canal está abierto para en caso necesario pedirles que establezcan criterios y medidas de seguridad que el instalador de determinados sistemas debe implementar para securizarlos.
 - Desde el Portal de Administración Electrónica se ofrecen soluciones de seguridad [48], pero básicamente refiere y utiliza las ya comentadas fuentes del CCN.
- Fuentes internacionales. Estas deben ser tenidas en consideración antes de aplicarlas para determinar si son adecuadas o no. Por ejemplo las siguientes:
 - Guías OTAN. Accesibles para determinadas unidades y organismos, pero hay muchas que no son clasificadas y pueden difundirse para otros propósitos que no sean el bastionar un sistema de la Alianza. En el momento de realizar este TFM no hay guía aplicable a los productos de OpenStack.
 - Guías publicaciones del NIST. Pueden encontrarse en la sección de publicaciones de la referencia [49], de las cuales algunas publicaciones ya han sido referidas en este trabajo.
 Entre las publicaciones del NIST se destacan las *Special Publications* (SP) y las guías del Federal Desktop Core Configuration (FDCC). El FDCC viene a ser un órgano del NIST que establece las líneas base y las configuraciones de seguridad para los sistemas gubernamentales de los Estados Unidos. (Viene a ser como el CCN Español).
 El FDCC ofrece una lista de configuraciones de seguridad recomendadas por el NIST para microordenadores de propósito general que están conectados directamente a las redes gubernamentales de los EEUU. Entre ellas están todas las configuraciones para sistemas Microsoft, y para otros sistemas operativos y productos de software.
 El NIST tiene un programa dedicado al *cloud computing* que puede consultarse en la referencia [50].
- Fuentes en Internet. La principal fuente siempre debe ser el fabricante u organización que está detrás de un producto determinado. Estos suelen ofrecer conjuntos de buenas prácticas para configurar sus productos. Tampoco pueden desecharse otros interlocutores válidos que podemos encontrar en la Red, como el conocido Oráculo [51].

2.4.3 Lo que se puede hacer...

Para el caso que nos ocupa, que podríamos resumir es bastionar una nube privada basada en OpenStack, lo que se puede y debería hacerse es (a modo de decálogo):

¹⁰ El CERT de referencia para las Administraciones Públicas es el CCN-CERT.

1º Decidir la arquitectura y la procedencia de todos los componentes a instalar, desde los SO de base hasta los productos de OpenStack basándonos en el soporte y la existencia de guías de seguridad o buenas prácticas.

2º Comprobar cada uno de los requisitos de seguridad establecidos en la guía CCN-STIC-301 y establecer una estrategia para cumplirlos y satisfacer los que sean aplicables. Para esto como mínimo se habrá de recurrir a las medidas contempladas en los siguientes párrafos.

3º En cada servidor o máquina física o virtual sobre la que se despliegue la infraestructura necesaria para la nube hay que actualizar y mantener actualizado su *firmware*, BIOS o UEFI, siguiendo las instrucciones de su propio manual y estableciendo en lo posible contraseñas de acceso a las utilidades y en caso de necesidad habilitando comunicaciones seguras (protocolos ssh y/o https) de acceso a interfaces de administración remota.

4º La instalación del sistema se realizará guardándose de no conectarlo a Internet, hasta que no haya sido totalmente bastionado. En función de su criticidad (de acuerdo al nivel de clasificación de la información que maneje y al modo de operación del sistema) habrá o no obligación de mantener el sistema aislado de Internet.

Por tanto la instalación de los SO y los componentes antes de su bastionado hay que hacerla en principio sin conexión a Internet.

De acuerdo con si el sistema puede o no conectarse a la Red será necesario instalar un sistema de actualizaciones *off-line*.

5º Securitizar el SO de base, por ejemplo si se instala la distro de Ubuntu OpenStack hay que aplicar las recomendaciones de la guía CCN-STIC-612, adaptándola en lo posible a una versión soportada como es la Ubuntu 18.04 LTS.

6º Si la búsqueda de guías específicas para los productos de OpenStack ha resultado infructuosa en las fuentes consultadas, (siguiendo con el mismo ejemplo) recurrir a los manuales de OpenStack Foundation donde para cada producto hay una sección de seguridad, y en base a las buenas prácticas recomendadas aplicar cuantas configuraciones sean posibles para cumplir con los requisitos establecidos en la guía CCN-STIC-301.

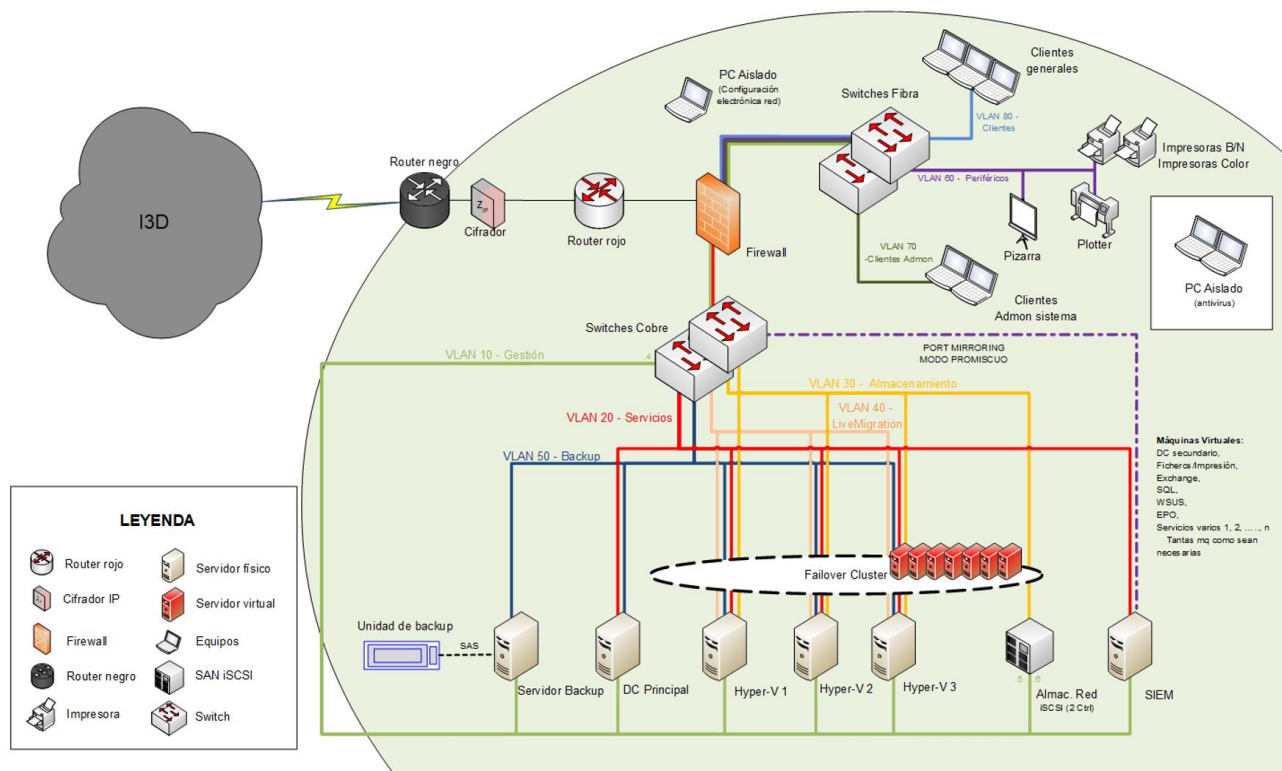
7º Si es posible, establecer mecanismos de cifrado de los soportes de información y los discos duros del sistema.

8º Si es posible, cifrar las comunicaciones entrantes y salientes.

9º Proteger el sistema ante código malicioso.

10º Ejecutar un plan de pruebas que permita establecer el grado de seguridad alcanzado y mejorarlo con el tiempo si es posible. Dentro del plan deberá realizarse un análisis de vulnerabilidades.

Aparte de este decálogo habría que aplicar otras muchas medidas de procedimiento como en el alta y baja de usuarios, en el uso de pendrives (memorias USB), etiquetados de información y uso de metadatos, criterios para las máquinas virtuales que la nube dispensará como IaaS, etc. ¡Todo dependerá del grado de paranoia que estemos sufriendo!



de cobre en *stack* donde se configuran diferentes VLAN (para gestión, servicios, almacenamiento, migración en vivo y *backup*).

El *rack* de servidores se conecta a través del *firewall*, cuyas políticas proporcionan el concepto de autoprotección, con el *router* que da acceso al exterior y con las tres redes, VLAN, de clientes y periféricos.

¿Qué caracteriza a este sistema? Pues, aparte de su alta disponibilidad y redundancia, que ofrecería sus servicios por medio de servidores virtuales los cuales son ‘hosteados’ en la granja.

Hablando hipotéticamente, cada servicio tiene un conjunto de máquinas y estas son accedidas por clientes, hay servicios de *core* que siempre estarán presentes, como controlador de dominio secundario, ficheros e impresión, actualizaciones (WSUS), EPO, etc., pero la mayoría serán servicios productivos, los que demandan los usuarios del sistema a través de los clientes.

Servicios típicos serían: correo electrónico, bases de datos, entidad de certificación, portales web (Sharepoint, Joomla, WISE), y algunos más específicos (construidos por aplicaciones cliente-servidor, o por aplicaciones web, o por una combinación de ambas) como SIMACET, IGESIT, SIS, LOGFAS, TOPFAS, JCHAT, videoconferencia, JOCWatch, JEMM, etc.

En sistemas como el descrito el aprovisionamiento de un nuevo servicio implicaría que unos determinados nuevos servidores virtuales sean ‘hosteados’, si caben en el nodo, y sean administrados y gestionados.

Podría haber complicado el sistema maestro expuesto con mayor segmentación, zonas desmilitarizadas (DMZ), etc., lo cual implicaría aun mayor carga administrativa.

Por finalizar la foto, varios de estos nodos se interconectan a la I3D, y entre todos prestan variedad de servicios, algunos iguales y otros exclusivos al conjunto de todos los usuarios del sistema.

No se van a describir en este trabajo estos servicios, algunos son aplicaciones OTAN, y otros nacionales, por ejemplo el SIS implica la instalación de una máquina virtual que es un servidor TOMCAT que corre unas decenas de *servelets* y presentan un portal a través de un Apache, otra máquina virtual soporta la presentación de información geográfica aportando los mapas al TOMCAT, y por último una instancia de base de datos para almacenar información relacional y espacio de almacenamiento específico en el servidor de ficheros del nodo.

Varios de esos servicios se prestan a los usuarios del nodo y no a los de otros nodos, pero entre los servicios de varios nodos se replican datos, se comparte información y por supuesto hay casos en los que usuarios de un nodo acceden a los servicios de otro nodo.

Recuerdo que el sistema descrito no es real, y aunque se expone como si fuera un sistema militar, no difiere de lo que montan bancos y otras empresas, más de lo que cambiarían los servicios, los cuales no describo por ser información que no importa para los objetivos de este TFM.

3.1.2 ¿Una nube privada?, ¿para qué?

Podríamos instalar una nube privada en un nodo o varias nubes en varios nodos. Además podríamos incluir en el clúster de una nube equipamiento no ya de su nodo, sino de los nodos adyacentes, el almacenamiento de varios nodos podría verse como un solo espacio, y las máquinas virtuales con las que se prestan los servicios ser aprovisionadas a demanda, no teniendo que tener máquinas redundantes en varios nodos. Las nubes permitirían optimizar los recursos, limitarían la carga administrativa de los gestores de los nodos y en definitiva proporcionarían al sistema las características esenciales del *cloud* (descrita en el capítulo dos), diferenciándose de un sistema de *hosting*.

La configuración de la nube privada que podríamos instalar estaría condicionada por el uso que le quisiéramos dar, en realidad para el propósito de este trabajo, que no es más (ni menos) que realizar una prueba de concepto, casi podríamos elegir cualquier configuración.

Por configuración se refiere a los componentes de OpenStack que se instalarían. Algunas de las configuraciones posibles son las que se muestran en la Tabla 3-1 a modo de ejemplo:

	Computación de alto rendimiento	Web Hosting	Clouds privadas¹¹
Nova (Computación)	X	X	X
Swift (Almacenamiento de objetos)			X
Cinder (Almacenamiento de bloques)	X		X
Glance (Servicio de imágenes)	X	X	X
Neutron (Red)		X	X
Heat (Orquestación)	X		
Trove (Base de Datos)			
Keystone (Autenticación)	X	X	X
Ceilometer (Telemetría)	X	X	
Designate (DNS)			X
Sahara (Elastic MapReduce)			
Ironi (Bare-Metal)			
Horizon (Interfaz Web)	X	X	

Tabla 3-1. Configuraciones de OpenStack según su propósito.

¹¹ Se refiere a un cloud para que los usuarios aprovisionen IaaS y ellos mismos se instalen sus servicios.

	Web y eCommerce	Big Data	DBaaS ¹²	Multimedia ¹³
Nova (Computación)	X	X	X	X
Swift (Almacenamiento de objetos)			X	X
Cinder (Almacenamiento de bloques)	X		X	
Glance (Servicio de imágenes)	X	X	X	
Neutron (Red)	X	X	X	X
Heat (Orquestación)				
Trove (Base de Datos)	X		X	
Keystone (Autenticación)	X	X	X	X
Ceilometer (Telemetría)				
Designate (DNS)			X	
Sahara (Elastic MapReduce)		X		
Ironic (Bare-Metal)		X	X	
Horizon (Interfaz Web)	X	X	X	

Tabla 3-2. (Continuación de la tabla 3-1). Configuraciones de OpenStack según su propósito.

En cada una de estas configuraciones (las de las tablas 3-1 y 3-2, anteriores) pueden ser añadidos algunos otros proyectos (componentes) de OpenStack, pero con los especificados serían configuraciones funcionales.

La modularidad de OpenStack muy alta y aunque la información para decidir las composiciones anteriores está sacada del sitio oficial, las de las anteriores tablas difiere un poco, simplificando las configuraciones de ejemplo que se dan en el site. Ver la referencia [13].

Para este trabajo hay que elegir una arquitectura de las posibles, revisando las configuraciones o arquitecturas de ejemplo de la referencia [13], parece lo más adecuado comenzar con la configuración “*Compute Starter Kit*” [52], sobre la que realizaríamos una primera escalada a la configuración “*Video Processing and Content Delivery*” [53], y a continuación una segunda escalada instalando los componentes Glance y Horizon debería ser suficiente para ofrecer IaaS, prescindiendo de la telemetría para lo que sería necesario instalar Celiometer.

Con esta configuración la prueba de concepto que realizaremos será la de integrar una nube privada para que los usuarios puedan aprovisionar IaaS donde instalen sus servicios ellos mismos.

¹² Base de datos como servicio.

¹³ Video y audio streaming.

Decisión 1. Arquitectura OpenStack.

Se instalará un prototipo OpenStack con la siguiente arquitectura de componentes (todos o algunos):

- Nova (Computación)
- Swift (Almacenamiento de objetos)
- Cinder (Almacenamiento de bloques)
- Glance (Servicio de imágenes)
- Neutron (Red)
- Keystone (Autenticación)
- Horizon (Interfaz Web)

En realidad, y tratándose de sistemas clasificados, la securización y la posterior inspección de seguridad es tan estricta que este grado de libertad a los usuarios no es deseable. En los sistemas actualmente acreditados los servicios están muy tasados y el sistema de *hosting* se muestra eficiente, por lo que en conclusión, satisface las necesidades mientras que una instalación *cloud* para IaaS puede aumentar la superficie de ataque e introducir nuevas variables en la seguridad, y ¿es realmente necesaria la funcionalidad que ofrece una nube?

La principal ventaja que observo es que una *cloud* proporcionaría probablemente más elasticidad sobre la infraestructura, un uso más optimizado del *hardware*, mejoraría la disponibilidad y favorecería la recuperación ante desastres. En cualquier caso las imágenes volcadas en Glance deberían ser muy específicas, de manera que si existen altos requerimientos de seguridad cualquier cosa no pueda ser instalada en la nube privada.

3.1.3 Descripción del prototipo y su integración en el nodo maestro

El prototipo de nube que planteamos lo queremos integrar como componente del nodo maestro descrito en el apartado 3.1.1.

De acuerdo con la decisión 1 tomada la arquitectura lógica de la nube privada es la mostrada en la figura 3-2:

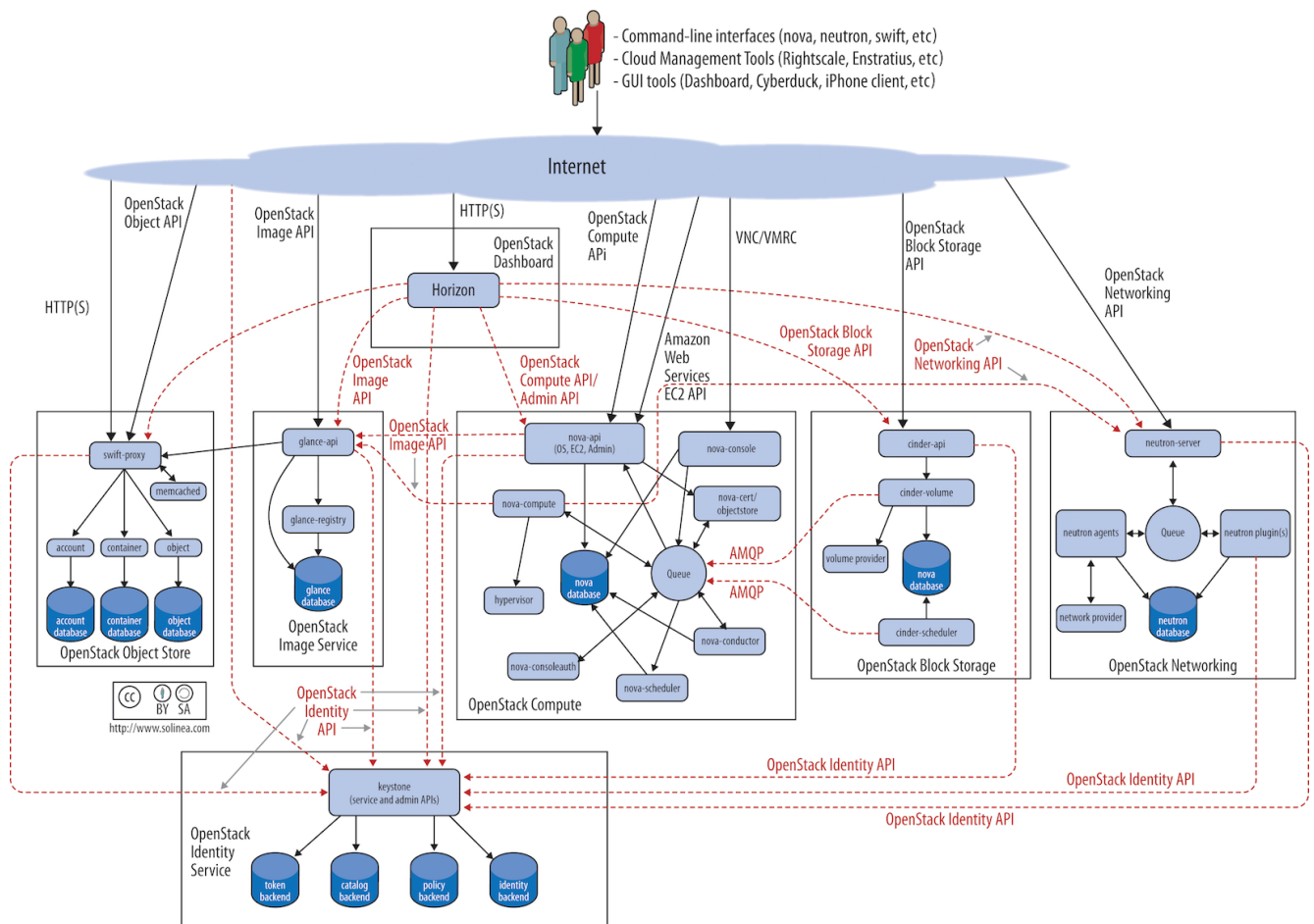


Figura 3-2. Arquitectura lógica del ecosistema OpenStack [54].

Se pueden hacer básicamente dos instalaciones, la primera totalmente virtualizada, (virtualizar sobre virtualizado *-nested virtualization-*), es decir los servidores de la nube se instalarían sobre servidores virtuales hospedados en el nodo maestro, y para esto deberíamos ver la manera de emplear el hipervisor de la granja como subyacente, lo que nos lleva a pensar en opciones propietarias como la que ofrece VMware o Microsoft Azure, que son las alternativas serias a OpenStack.

La segunda instalación consistiría en conectar los servidores físicos (los necesarios) al nodo maestro y sobre ellos instalar OpenStack (u otra solución, como CloudStack). En esta opción, pensando en OpenStack, se intentaría integrar con el nodo maestro por ejemplo el almacenamiento (máquinas virtuales existentes podrían servir de plantilla para la IaaS), recursos de red, y sobre todo la autenticación conectando Keystone con el LDAP o el Directorio Activo del nodo maestro.

Cabría una tercera opción, y es instalar una infraestructura de nube totalmente independiente, y sobre ella prestar los mismos recursos y capacidades que presta el nodo maestro.

Desde el punto de vista de la seguridad todo sistema debe diseñarse con componentes *hardware* y *software* que cumplan ciertas condiciones, que obviando las de seguridad física (como limitaciones en las emanaciones de los equipos), en lo referente a estos elementos podemos resumir:

- El hardware debe permitir la configuración segura de los accesos de bajo nivel, por ejemplo en los servidores HP las interfaces iLO, o las iDRAC de Dell. También deben estar actualizados a la última versión el *firmware* de los dispositivos, y parcheados con las últimas actualizaciones de seguridad, por lo que deben ser versiones soportadas por fabricante.

- Todo componente *software* debe estar bien licenciado. No solo por *compliance*, por cumplir con la legalidad, en el caso de *software opensource* si el licenciamiento no da derecho a actualizaciones y es necesario adquirir algún tipo de soporte, debe hacerse.
- Todo componente *software* debe estar bajo soporte de su fabricante y garantizada la distribución de los parches o actualizaciones de seguridad, (por eso son requeridos fabricantes de reconocido prestigio). Cuando termina el soporte termina su posible uso.
- Debe existir una guía de seguridad aplicable al producto, consistente en las configuraciones de seguridad y buenas prácticas para disminuir la superficie de ataque (las vulnerabilidades), preferentemente del CCN en el caso de sistemas nacionales, o de OTAN para sistemas de este ámbito, o de alguna otra procedencia (NIST por ejemplo), o proporcionada por el fabricante.
- Debe elaborarse una documentación de seguridad donde se consignen las medidas aplicadas, y en su caso todas las desviaciones de las guías de seguridad deben estar justificadas.

De las posibilidades gratuitas que se nos ofrecen para disfrutar de OpenStack, frente instalar los servicios de OpenStack uno a uno en una de las distribuciones Linux soportadas, y frente a las de la comunidad RDO, Ubuntu OpenStack cumple lo siguiente:

- OpenStack se instala de forma nativa sobre Ubuntu Server en sus versiones LTS, con lo que este SO actualmente está totalmente soportado por Canonical (cuyo prestigio es reconocido) desde su versión 14.04 LTS y si elegimos la versión 18.04 LTS tiene soporte hasta abril de 2023.
- Las versiones LTS y las server son más adecuadas que las de corto periodo y las de escritorio porque por construcción su superficie de ataque es más pequeña (a base de prescindir de todo lo superfluo).
- Las versiones de OpenStack están sincronizadas en su lanzamiento con las de Ubuntu, y son continuadas (mantenidas y soportadas) por OpenStack Foundation durante dos años. Antes de discontinuar una versión Canonical introduce lo necesario en su sistema Operativo Ubuntu para que pueda actualizarse OpenStack a las versiones continuadas.
- Los sistemas clasificados se acreditan por un periodo máximo de tres años, por lo que los periodos de soporte de Ubuntu y de OpenStack parecen suficientemente adecuados.
- El CCN dispone de una guía de seguridad (CCN-STIC 612 del año 2014) para sistemas Linux basados en Debian, y si bien está redactada para aplicarse directamente en Ubuntu Server 12.04 LTS puede adaptarse¹⁴ para las versiones más modernas. Adicionalmente tanto OpenStack Foundation como Canonical ofrece un conjunto de buenas prácticas en su documentación que pueden complementar las medidas de la guía.

Todo lo anterior lleva a adoptar la siguiente decisión:

Decisión 2. Selección de la distro.

Utilizaremos Ubuntu OpenStack y/o una instalación de OpenStack paso a paso sobre Ubuntu. La usaremos porque permite tanto la instalación de un sistema *all-in-one* y la instalación de un clúster orientado a producción. Facilita además la expansión sobre la instalación inicial añadiendo fácilmente otros componentes.

¹⁴ Procedería que caso de implantarse Ubuntu Server LTS como sistema operativo en sistemas clasificados el CCN mantuviera la guía más actualizada, para lo que habría que solicitárselo, normalmente lo hacen cuando la demanda es fuerte.

Para realizar instalaciones sin conexión a Internet sería necesario instalar servidores *mirror* aparte de los destinados a las pruebas. Por falta de tiempo y por escasez de medios se prescindirá para todas las instalaciones de servidores *mirror* que permitieran descargar los paquetes y trasladarlos a las máquinas sin conexión a Internet. Esto motiva la decisión 3:

Decisión 3. Conexión a Internet.

Todas las máquinas físicas y virtuales que se instalen en el ámbito de este TFM se conectarán a Internet, aun sabiendo que para sistemas destinados a producción en entornos clasificados, o simplemente seguros, no es una condición permitida ni deseable, y deberían instalarse *mirrors off-line* para este propósito.

Como no disponemos un campo de pruebas con un nodo maestro nos centraremos en construir un sistema OpenStack totalmente independiente pasando por tres fases:

1ª Instalar un OpenStack *all-in-one*, para pruebas y desarrollo. Los epígrafes 3.2 y 3.3 explican las prácticas realizadas dentro de esta fase.

2ª Instalar una infraestructura clúster basada en servidores virtuales que pudiera ser hosteada en un nodo maestro. El epígrafe 3.4 explica esta instalación tal y como se ha llevado a cabo.

3ª Instalar una infraestructura clúster independiente basada en servidores físicos y que pudiera interconectarse al nodo maestro o directamente a la I3D. Se explica en el epígrafe 3.5.

En todas estas instalaciones prestaré especial atención a la configuración de la seguridad para aportar los inputs necesarios (Anexo III) para acreditar el sistema OpenStack para el manejo de la información hasta cierto grado de clasificación.

3.2 Las primeras instalaciones all-in-one.

Durante este trabajo el primer acercamiento práctico a OpenStack lo he realizado instalando dos versiones en modalidades *all-in-one*, enfocadas a montar un entorno de pruebas y desarrollo.

Estas instalaciones permiten desplegar el ecosistema en una única máquina cuyos recursos deben ser suficientes, lo recomendado es un procesador i7 (a ser posible que no sea tipo U), 16 Gb de RAM y un disco SSD. Pueden hacerse instalaciones en máquinas más modestas, pero difícilmente podrán ser escalables (una instalación *all-in-one* podría incluso escalar a producción) y si falta capacidad tampoco funcionará del todo bien como entorno de desarrollo.

De las instalaciones que he probado la que me ha resultado más cómoda ha sido OpenStack DevStack, pero PackStack, que es la instalación simplificada del proyecto RDO para instalar OpenStack en CentOS, tampoco es complicada. Por supuesto es *opensource* y gratis.

La instalación se completa en cuatro etapas, recopilación de la información necesaria, configuración del *hardware* e instalación del SO, configuración del SO e instalación de OpenStack y por último configuración de OpenStack.

Para este trabajo la necesidad obliga y realicé la instalación en una máquina virtual con VirtualBox, por tanto la capacidad del sistema resultante es muy limitada, y su rendimiento no es en absoluto óptimo, OpenStack funcionará pero con muy poca capacidad de prestar servicio a usuarios (tampoco es el objetivo de esta instalación obtener un sistema en producción, sería un entorno de pruebas o desarrollo). Las características de la máquina virtual son:

- Tipo (sistema operativo): Linux.
- Versión: Red Hat (64-bit)
- RAM: 6 Gb.

- Disco: 50GB VDI reservado dinámicamente.
- Interfaz de red: Una en modo bridge y en modo promiscuo con la opción permitir todo.

3.2.1 Instalación all-in-one sobre CentOS con PackStack

El sistema operativo para esta instalación es CentOS 7 y se descarga del sitio oficial [55], elegimos la iso minimal: CentOS-7-x86_64-Minimal-1804.iso.

Lo primero que choca es que las iso no se descargan de los *mirror* oficiales de la distribución, avisa la página de que por causas de conservación del ancho de banda disponible, ¿resulta inseguro? Realizo la descarga desde el *mirror* de la Universidad de Vigo:

http://centos.uvigo.es/7.5.1804/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso

A continuación se monta la iso en la unidad óptica de la MV y se realiza la instalación. Solo hay que tener en cuenta lo siguiente:

- Únicamente separaremos la partición /boot y la SWAP del resto (/). Esto no es lo que hay que hacer desde el punto de vista de la seguridad, pero tampoco hay que conectarse a Internet y muchas cosas más. A los efectos de probar OpenStack lo haremos así.
- Desactivaremos el KDUMP, que es un mecanismo de volcado de memoria cuando se producen fallos del kernel y consume memoria.
- Configuraremos la red con una IP fija. Si estamos usando una conexión doméstica, al tener la interfaz en modo *bridge*, es como si la máquina virtual estuviera en la misma red doméstica, y la puerta de enlace y los servidores DNS deben ser el mismo *router* (cada cual debe adaptar estos datos a su caso particular).
- Ponemos el nombre del *host*: OpenStack.localdomain.

Tras realizar la instalación comprobamos la conexión de red, desde el *host* y desde la máquina, a nivel local y hacia Internet.

Después de algunos quebraderos de cabeza, revisando la documentación de RHEL *networking* [56] he podido ver que es necesario deshabilitar el servicio NetworkManager para que funcione la red de OpenStack. Este servicio es un gestor de red presente en RHEL y sus derivados CentOS y Fedora.

NetworkManager se usa en entornos de escritorio (enfocado a clientes más que a servidores) pero además durante la instalación figura como dependencia en algunos paquetes, por lo que incluso no habiendo instalado ningún escritorio puede que esté instalado y activo, aunque con una instalación minimal no debería estar. Lo comprobamos con el siguiente comando:

```
# systemctl -t service
```

... o directamente con el comando:

```
# Service NetworkManager status
```

(# es el prompt de root y \$ el de usuario no privilegiado).

En las instalaciones que he realizado siempre encontré NetworkManager.service cargado, activo y corriendo. Ojo, que una de las cosas que más me ha confundido es que con el comando `chkconfig` al listar no lo encontraba como activo y es por la configuración nativa de systemd.

Para poder deshabilitarlo primero de todo lo tenemos que detener:

```
# service NetworkManager stop
```

A continuación lo eliminamos del arranque:

```
# chkconfig NetworkManager off.
```

También debemos configurar la interfaz para que no use el NetworkManager, para lo que editamos el archivo `/etc/sysconfig/network-scripts/ifcfg-enp0s3` y lo modificamos introduciendo justo delante de la línea de la IP el parámetro:

```
NM_CONTROLLED="no"
```

Nos aseguramos de que el servicio Network está iniciado:

```
# service network start
```

... y habilitado con el comando:

```
# chkconfig network on.
```

Pueden producirse otros problemas con la instalación de OpenStack a raíz del *firewall* y de la *suite* de seguridad SELINUX. Como ahora nos interesa progresar en la instalación de OpenStack y no preocuparnos en este momento en que el *firewall* del SO nos corte alguna conexión necesaria, paramos y deshabilitamos el FirewallD y lo mismo con SELINUX.

FirewallD es una aplicación para manejar desde la línea de comandos Netfilter (es el cortafuegos en el núcleo de Linux) que hace más fácil configurar las reglas que Iptables. Por defecto está instalada en CentOS y Fedora. Tecleamos los comandos:

```
# systemctl disable firewalld  
# systemctl stop firewalld.
```

SELINUX es un módulo de seguridad para soportar políticas de seguridad, como las asociadas al control de acceso. Para deshabilitarlo es necesario editar su archivo de configuración `/etc/selinux/config` y modificar o establecer:

```
SELINUX=disabled.
```

Por último reiniciamos la máquina. Por supuesto todas estas modificaciones hacen inseguro el sistema, y hay que planteárselas si estuviéramos instalando un entorno de producción. En el contexto de este trabajo precisamente se trata de estudiar las implicaciones de las configuraciones realizadas en el estado de seguridad, pero lo que ahora interesa es progresar en la instalación. Una vez todo instalado y funcionando habría que ir robusteciendo el sistema y verificando que no se pierden funcionalidades. Por otra parte si no estuviéramos haciendo una instalación basada en una distribución, en unos *script* y programas, que despliegan el ecosistema OpenStack sin prácticamente tener que tomar decisiones y con opciones por defecto, no actuaríamos de esta manera.

A continuación instalaremos OpenStack PackStack (Rocky) desde los repositorios de CentOS:

```
# yum install -y centos-release-openstack-rocky  
# yum install -y openstack-packstack  
# yum update -y  
# reboot
```

Y ya solo nos queda desplegar el ecosistema mediante PackStack. Daremos la siguiente orden:

```
# packstack --allinone --provision-demo=n --os-neutron-ovs-bridge-mappings=extnet:br-ex --os-neutron-ml2-type-drivers=vxlan,flat --os-heatinstall=y
```

Se instalará OpenStack en modo *all-in-one* (sin modo demo), con Heat, un componente adicional para la arquitectura que teníamos pensada, pero al que PackStack saca partido.

La instalación lleva una hora aproximadamente. Cuando termine recibiremos un mensaje si todo ha ido bien, como se muestra en la figura 3-3, o recibiremos mensajes de error, bastante complicados de depurar pues son del proceso automatizado de PackStack (por ejemplo, a mi me dieron causados por el servicio NetworkManager).

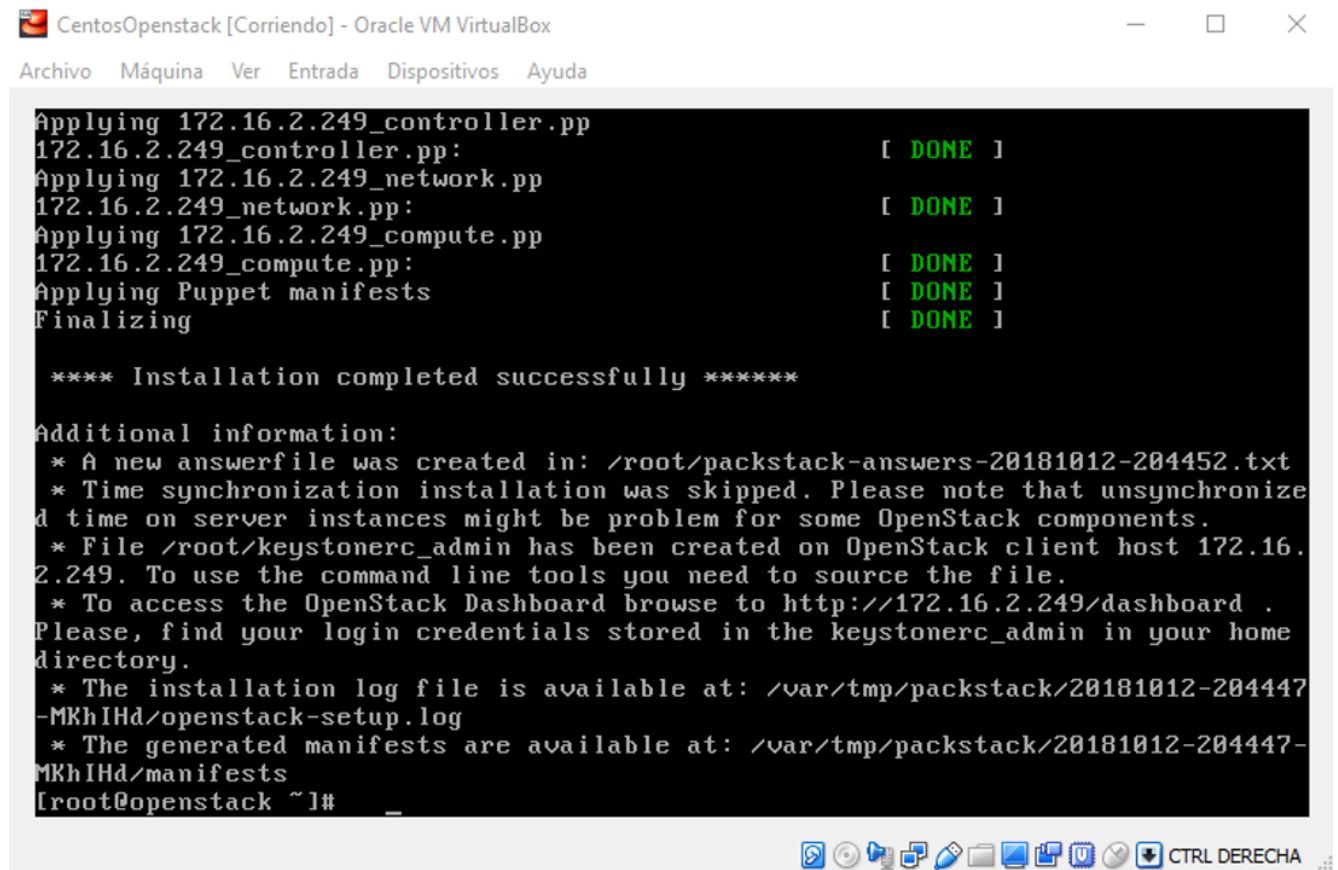


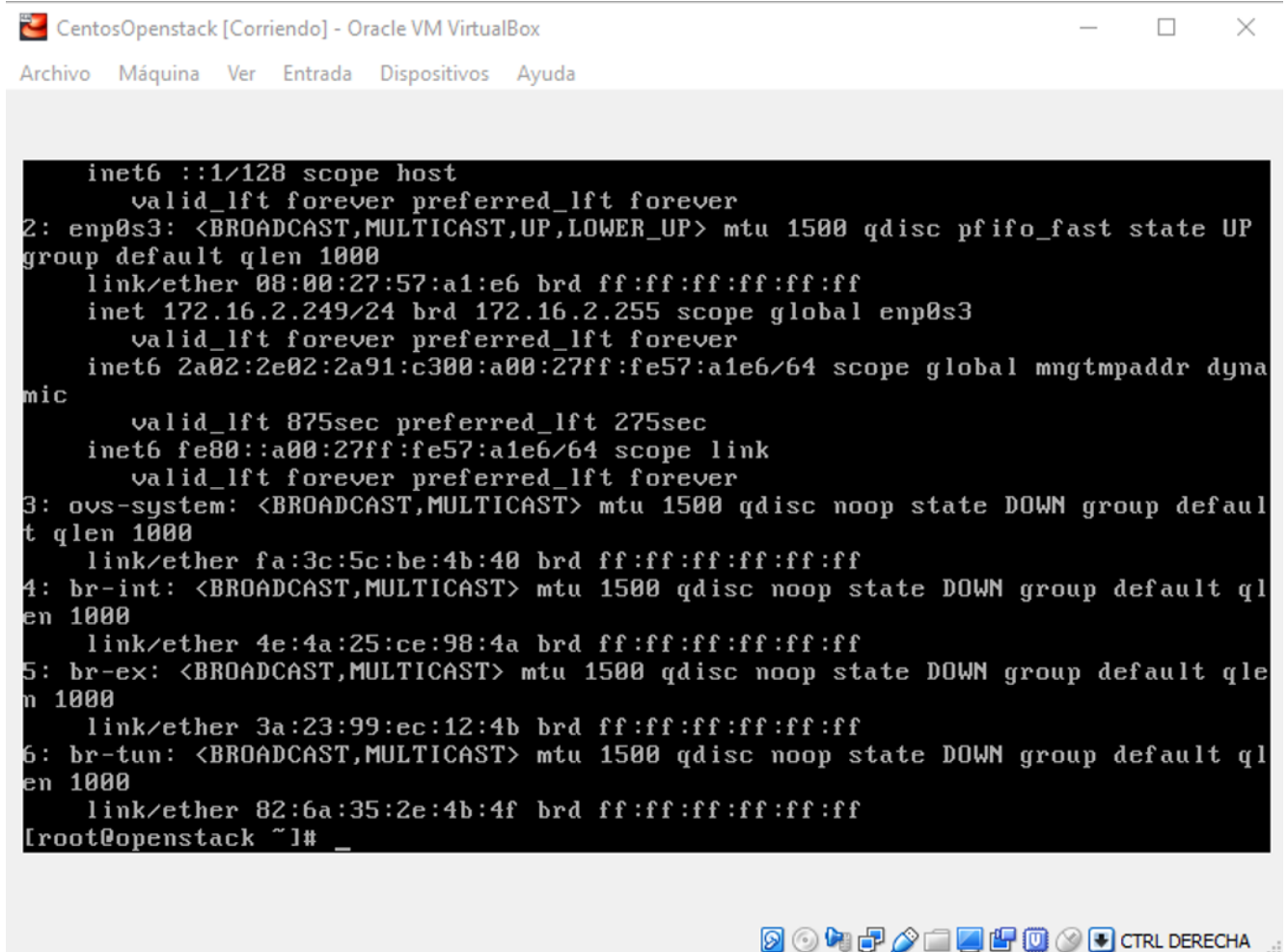
Figura 3-3. Pantalla de fin de instalación de OpenStack PackStack.

En la pantalla fin de la instalación se nos remite al fichero `/root/keystonerc_admin` para ver las credenciales de acceso, lo listamos:

```
# cat /root/keystonerc_admin
```

... y accedemos a las credenciales.

Tras la instalación es necesario reconfigurar las interfaces de red. PackStack al instalar OpenStack habrá creado cuatro tarjetas de red, las cuales aparecerán como desactivadas, son las que se muestran en la figura 3-4 y que están numeradas de la 3 a la 6:



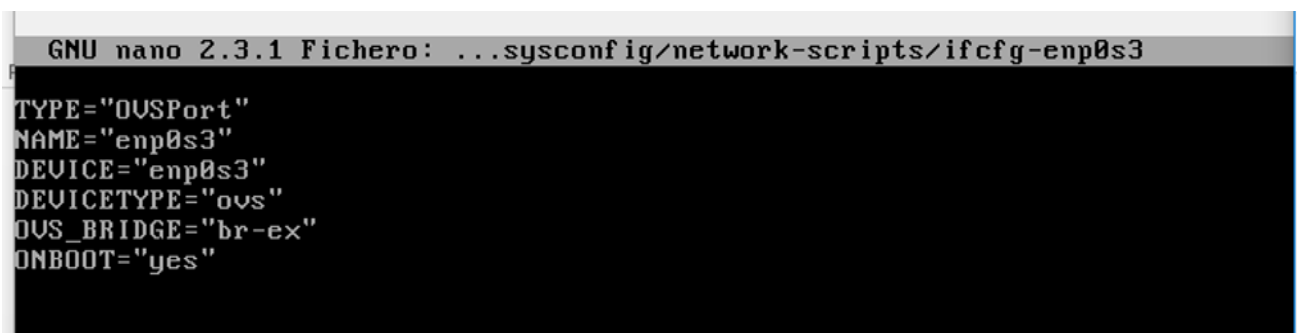
```

inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:57:a1:e6 brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.249/24 brd 172.16.2.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 2a02:2e02:2a91:c300:a00:27ff:fe57:a1e6/64 scope global mngtmpaddr dyna
mic
    valid_lft 875sec preferred_lft 275sec
    inet6 fe80::a00:27ff:fe57:a1e6/64 scope link
    valid_lft forever preferred_lft forever
3: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group defaul
t qlen 1000
    link/ether fa:3c:5c:be:4b:40 brd ff:ff:ff:ff:ff:ff
4: br-int: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default ql
en 1000
    link/ether 4e:4a:25:ce:98:4a brd ff:ff:ff:ff:ff:ff
5: br-ex: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default gle
n 1000
    link/ether 3a:23:99:ec:12:4b brd ff:ff:ff:ff:ff:ff
6: br-tun: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default ql
en 1000
    link/ether 82:6a:35:2e:4b:4f brd ff:ff:ff:ff:ff:ff
[root@openstack ~]#
  
```

Figura 3-4. Resultado del comando ip addr tras la instalación de OpenStack.

Las tarjetas creadas están identificadas como ovs-system, br-int, br-ex y br-tun. Tenemos que reconfigurar la tarjeta de red de la máquina, enp0s3 y la tarjeta br-ex de manera que los datos pasen entre las dos mediante un puente interno en la máquina, y la tarjeta que quede expuesta a la red de área local sea br-ex. Para esto hay que modificar los ficheros de configuración de las interfaces.

El fichero /etc/sysconfig/network-scripts/ifcfg-enp0s3 debe contener lo mostrado en la figura 3-5:



```

GNU nano 2.3.1 Fichero: ...sysconfig/network-scripts/ifcfg-enp0s3
TYPE="OVSPort"
NAME="enp0s3"
DEVICE="enp0s3"
DEVICETYPE="ovs"
OVS_BRIDGE="br-ex"
ONBOOT="yes"
  
```

Figura 3-5. Contenido del fichero ifcfg-enp0s3.

El fichero /etc/sysconfig/network-scripts/ifcfg-br-ex debe contener lo mostrado en la figura 3-6:

```
GNU nano 2.3.1 Fichero: .../sysconfig/network-scripts/ifcfg-br-ex
DEVICE="br-ex"
DEVICETYPE="ovs"
TYPE="OVSBridge"
BOOTPROTO="static"
IPADDR="172.16.2.249"
NETMASK="255.255.255.0"
GATEWAY="172.16.2.1"
DNS1="172.16.2.1"
IPV4_FAILURE_FATAL="no"
IPV6INIT="no"
ONBOOT="yes"
```

Figura 3-6. Contenido del fichero ifcfg-br-ex.

Ya hemos creado nuestra nube privada y podemos acceder a ella desde la red local de nuestra organización, por ejemplo desde el ordenador anfitrión de la MV CentOS que aloja la nube, desde un navegador accedemos a <http://172.16.2.249/> (la ip de la CentOS) y nos remite al portal Horizon, o como se observa en la figura 3-7, directamente en línea de comandos desde Putty, o en la Shell de la MV.

```
[root@openstack ~]# source keystone_admin
[root@openstack ~(keystone_admin)]# openstack flavor list
+-----+-----+-----+-----+-----+-----+-----+
| ID | Name       | RAM | Disk | Ephemeral | VCPUs | Is Public |
+-----+-----+-----+-----+-----+-----+-----+
| 1  | m1.tiny    | 512 | 1    | 0          | 1      | True      |
| 2  | m1.small   | 2048 | 20   | 0          | 1      | True      |
| 3  | m1.medium  | 4096 | 40   | 0          | 2      | True      |
| 4  | m1.large   | 8192 | 80   | 0          | 4      | True      |
| 5  | m1.xlarge  | 16384 | 160  | 0          | 8      | True      |
+-----+-----+-----+-----+-----+-----+-----+
[root@openstack ~(keystone_admin)]#
```

Figura 3-7. Acceso a OpenStack desde CLI.

3.2.2 Instalación all-in-one sobre RHEL con PackStack

Tras la instalación anterior he realizado otra muy similar sobre RHEL, también con el instalador PackStack. La principal diferencia está en la obtención del sistema operativo, ya que tenemos que registrarnos en la página oficial de Red Hat en la dirección:

https://www.redhat.com/wapps/ugc/register.html?_flowId=register-flow&_flowExecutionKey=e1s1

Concluido el registro, nos logueamos y nos suscribimos a las versiones de prueba de OpenStack y del sistema operativo mismo en la dirección <https://access.redhat.com/downloads>, donde debemos suscribir las de Red Hat OpenStack Platform y Red Hat Enterprise Linux. La versión de RHEL 7.5 fue lanzada en abril de 2018 pero en general nos vale cualquier iso de RHEL 7.X, hay que tener en cuenta que las versiones estables tienen soporte durante 10 años. Ya decíamos en el capítulo 2 que si tuviéramos que instalar OpenStack en serio, probablemente la mejor elección sería RHEL. Elegimos la última iso Red Hat Enterprise Linux 7.5 Binary DVD.

Antes de salir de la web de RHEL hay que pinchar en las suscripciones activas y seleccionar la de OpenStack y tomar nota del dato grupo maestro (*Master Pools*, no confundir con *Derived Pools*) que se encuentra en Id,s de grupo (*Pool IDs*). Es un número en hexadecimal de 32 caracteres que más adelante usaremos al configurar el sistema operativo.

Haremos la instalación sobre una máquina virtual como la utilizada antes en CentOS. Tras la instalación del SO comenzamos con las configuraciones básicas, empezando por el nombre del equipo y, muy importante, configurando la suscripción mediante el número en hexadecimal *Master Pools*. Los comandos para verificar que el nombre del equipo está bien configurado serían:

```
# hostname
```

```
... o
```

```
# hostname-f
```

El segundo comando debe dar el mismo resultado que el anterior, si da *localhost* hay que solucionarlo editando el archivo */etc/hosts*, y posteriormente teclear los dos comandos siguientes:

```
# hostnamectl set-hostname openstack.localdomain
```

```
# hostnamectl set-hostname --transient openstack.localdomain
```

Volvemos a comprobar con *hostname-f* y todo debe ir correcto.

A continuación la suscripción:

```
# subscription-managerregister
```

Introducimos nuestros datos de login en el sitio de Red Hat en el que nos dimos de alta. Una vez logueados listamos nuestras suscripciones:

```
# subscription-managerlist
```

Y nos responderá que no estamos suscritos a nada. Por eso vamos a introducir el código de nuestro Master Pools:

```
# subscription-manager attach --pool="nuestro número en hexadecimal"
```

Y por último comprobamos si hemos tenido éxito:

```
# subscription-managerlist --available --all
```

Ahora con:

```
# subscription-managerlist
```

... debe reportarnos las dos suscripciones la de OpenStack y la de RHEL.

A continuación deshabilitaremos los repositorios de Red Hat para luego habilitar solo los que nos interesan para nuestra instalación:

```
# subscription-manager repos --disable=*
```

```
# subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-server-extras-rpms --  
enable=rhel-7-server-rh-common-rpms --enable=rhel-ha-for-rhel-7-server-rpms --enable=rhel-7-  
server-optional-rpms
```

A partir de aquí continuaríamos parando el *firewall*, deshabilitando *NetworkManager*, parando *SELINUX*, y digamos que los pasos son los mismos que en la instalación anterior sobre CentOS.

El número de errores en esta instalación de RHEL ha sido mayor que en la de CentOS, por eso me he extendido en explicar aquella y no en esta, pero insisto, si queremos algo profesional RHEL OpenStack es la referencia.

3.2.3 Sobre las instalaciones con PackStack

Ya tenemos una nube privada en servicio, pero ¿qué ha pasado?

Aunque no nos pararemos ahora en valorar los resultados y conclusiones sobre estas primeras instalaciones, pues el capítulo siguiente muestra todas las conclusiones y resultados, el momento merece plantearse que hemos hecho.

KVM es el único hipervisor que utilizan los sistemas Red Hat y los basados en él como CentOS y Fedora. KVM se incluye como dos módulos del kernel, por tanto todo el código del hipervisor ya estaba incluido en el SO.

KVM convierte a Linux en un hipervisor de tipo 1, *bare metal*, al ser parte del núcleo se relaciona directamente con el módulo administrador de memoria, con el planificador de procesos, con la pila de entrada o salida (E/S), con los controladores de dispositivos, con la gestión de seguridad, con la pila de red, etc., todo para ejecutar las máquinas virtuales, de manera que cada una se implementa como un proceso de Linux programado por el planificador estándar del SO con *hardware* virtual dedicado: tarjeta de red, adaptador de gráficos, CPU, memoria y discos.

Dejando CentOS y Fedora aparte, Red Hat tiene dos productos capaces de virtualizar con KVM, Red Hat Enterprise Linux (RHEL) y Red Hat Virtualization (RHEV), las cuales ejecutan versiones diferentes de KVM.

El KVM de RHEL tiene todas las funcionalidades pero capacidades de gestión básicas y permite ejecutar hasta cuatro máquinas virtuales aisladas en un único *host*.

El KVM de RHEV permite la gestión empresarial de máquinas virtuales ilimitadas y se ofrecen productos de gestión del CPD por lo que se despliega sobre infraestructuras de varias máquinas y las implementaciones más grandes pueden requerir RHEV y OpenStack.

Por último, para implementar nuestra primera nube privada hemos recurrido a un instalador, PackStack (son una serie de *scripts* lanzados con muchos parámetros de configuración), claudicando para su uso condiciones de seguridad y de performance. Esto nos ocurrirá cada vez que utilicemos un instalador con un producto tan complejo como es OpenStack, por lo que tras la instalación convertir el sistema en realmente seguro y productivo puede ser tan trabajoso o más que hacer instalaciones de los productos uno a uno de manera totalmente interactiva entre el sistema y el ingeniero.

PackStack es un proyecto de RDO, igual que TripleO, que es otro de los instaladores que podríamos haber utilizado, pero el instalador (aunque es algo más que un instalador) más utilizado es DevStack, que también son una serie de *scripts* para desplegar OpenStack completamente, con las últimas versiones de los módulos.

Detrás de DevStack está la propia fundación OpenStack Foundation, a través de su programa QA (OpenStack Quality Assurance) [57]. El proyecto proporciona y mantiene el instalador de los módulos de OpenStack (los principales pues hay módulos de terceros). Utiliza repositorios en GitHub (donde están las fuentes del proyecto) y algunos *sites* específicos, como se muestra en la tabla 3-3:

Documentación	http://devstack.org	[58]
Preguntas más frecuentes	https://www.openstack.org/appdev/	[59]
Fuentes	https://git.openstack.org/cgit/openstack-dev/devstack/	[60]
GitHub	https://github.com/openstack-dev/devstack	[61]
DevStack. Script para instalación automática de Ubuntu y DevStack sin ninguna configuración manual.	https://github.com/makelinux/devstack-install-on-iso	[62]
Plataforma de lanzamiento	https://launchpad.net/devstack	[63]

Tabla 3-3. Principales webs de DevStack.

El proyecto DevStack incluye demostraciones que desde línea de comandos permiten comprobar el inicio y estado de ejecución de los servicios de OpenStack. También incluye múltiples ejemplos de configuración y ejecución de servicios, muchas de estas configuraciones prácticamente no se utilizan. DevStack también trata el SO de la interfaz CLI de los módulos de OpenStack.

De manera rápida, primero disponemos una máquina (física o virtual) con una instalación Linux mínima. DevStack funcionará sin problemas con las versiones de RHEL 7, con las últimas de Fedora y CentOS, con OpenSUSE, Debian y con las Ubuntu LTS desde la versión 12.04, (aunque el proyecto intenta centrarse en funcionar bien sobre las dos últimas Ubuntu LTS¹⁵). A continuación instalamos DevStack y lo lanzamos con opciones por defecto, se realizarán cambios de calado en el SO por tanto hay que tener cuidado de no probarlo en máquinas en producción.

3.3 Ubuntu OpenStack *all-in-one*.

Hasta aquí he llegado con otras distros, pero de acuerdo con las decisiones ya tomadas (y justificadas) al principio de este capítulo, Ubuntu es el SO que utilizaré de aquí en adelante. Ahora me enfrentaré a la instalación de un sistema *all-in-one*, y que por tanto comparte las principales características con las descritas en el apartado anterior; es idóneo para entornos de prueba y desarrollo, pero lógicamente, cuando el propósito es servir infraestructura como servicio, poco podemos ofrecer con una sola máquina.

Una de las características de la instalación sobre Ubuntu es que la instalación *all-in-one* puede ser escalada a producción con facilidad, agregándola a más nodos OpenStack que corran módulos más específicos o igualmente de propósito general.

Voy a probar dos instalaciones diferentes, una con DevStack, y otra siguiendo las instrucciones de Canonical mediante un script de instalación llamado Conjure-up [33]. Conjure-up proporciona un *wizard* para instalar paso a paso OpenStack, tanto en un clúster de servidores usando KVM como en una sola máquina usando contenedores LXD (*Linux Container Daemon*).

En cualquier caso volvemos a lo mismo, la mayoría de parámetros configurables serán por defecto, y partiré de que el SO no tendrá configuradas opciones de seguridad, de cara a robustecerlo, todo para que los scripts de instalación me funcionen sin problemas, por lo que para llevar el sistema a producción ninguna de estas opciones me parece lo mejor desde el punto de vista de la seguridad y del rendimiento (como todo es cuestión de preferencias, pues una vez funcionando el ecosistema podemos hacer un *hardening* del conjunto, aunque será mucho más complicado).

Las características aconsejables del equipo son las mismas que ya especificué en el punto anterior: procesador i7, 16 Gb de RAM y disco SSD.

3.3.1 Instalación *all-in-one* sobre Ubuntu con Conjure-up

En primer lugar probaré el procedimiento de Canonical con Conjure-up [34]. Una vez realizada la instalación por este procedimiento OpenStack se ejecutará en un clúster simulado a base de contenedores LXD con lo que los componentes del ecosistema se comportarán de manera similar a cuando se ejecutan en un CPD.

He intentado realizar esta instalación hasta dos veces sobre una máquina virtual de VirtualBox en un *host* i5U con 16Gb de RAM y HDD normal. La primera vez he configurado la MV con 6 Gb de RAM y un solo procesador de dos núcleos, y la segunda con 12 Gb de RAM y procesador de 4 núcleos. El resultado es que ha sido imposible terminar con éxito. Hay que decir que debería esperarse esto pues están documentados múltiples problemas con LXD en VirtualBox, y probablemente si hubiera hecho la

¹⁵ Ubuntu 16.04 LTS es el más probado por el equipo de proyecto y por tanto el que mejor resultado pueda darnos.

instalación aprovechando la virtualización KVM hubiera resultado como las anteriores sobre CentOS y RHEL.

Para esta prueba realizaré una instalación en una máquina física, un portátil que se acerca a las características aconsejables: procesador i5U (algo escaso) con 16Gb de RAM y disco SSD.

Primero instalamos Ubuntu server 18.04 LTS. Solo algunas indicaciones:

1) Descargar imagen para la instalación de Ubuntu server del sitio oficial: <https://www.ubuntu.com/#download> (ubuntu-18.04.1-live-server-amd64.iso) o alternativamente desde <http://cdimage.ubuntu.com/releases/18.04/release/> otra combinación de imagen e instalador, (depende de las preferencias, por ejemplo ubuntu-18.04.1-server-amd64.iso). Para quemar la iso en un USB *bootable* utilizar ‘rufus’, como está explicado en el anexo III de este documento.

2) Cancelar la configuración de red por DHCP y seleccionar configuración manual, continuar con los datos de la instalación, nombre de la máquina, cuenta de usuario, etc. Hasta llegar al particionado, donde hay que seleccionar particionado manual LVM y separar /boot y SWAP del resto. Cuando pida seleccionar paquetes adicionales, seleccionar OpenSSH. Tras esto finalmente completar la instalación del SO.

3) Deberá configurarse la interfaz de red, que en Ubuntu 18.04 se hace de forma permanente mediante el archivo de texto situado en /etc/netplan/01-netcfg.yaml, cuyo contenido debe parecerse al siguiente:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s31f6:
      dhcp4: no
      dhcp6: no
      addresses: [192.168.1.100/24, ]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
  wifis:
    wlp2s0:
      dhcp4: no
      dhcp6: no
      addresses: [192.168.1.101/24, ]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
      Access-points:
        "SSID de mi AP"
        password: "mi psk wpa2"
```

Se guarda el archivo con los cambios y se aplican en netplan:

```
$ sudo netplan apply
```

Para que funcione la interfaz wifi debe estar instalado el paquete wpa_supplicant que Ubuntu Server 18.04 LTS no trae por defecto.

4) Una vez instalado el sistema hay que instalar conjure-up de acuerdo a las instrucciones de la referencia [34]. Usaremos LXD, como ya se ha explicado, por lo que primero de todo tendremos que instalar LXD (mínima versión 3.0.0) y configurarlo para conjure-up:

```
$ sudo snap install lxd
```

... (en el momento de redactar este apartado se instaló la versión 3.6 de lxd).

Snap es un método de empaquetado para la instalación que trae de la Snapcraft los paquetes necesarios para las app. Snapcraft [64] es como la Store de Microsoft. Los paquetes snap traen incorporadas todas las fuentes y dependencias necesarias para la aplicación y éstas se instalan con aislamiento unas de otras en el SO. Para instalar LXD el método recomendado es usar snap.

A continuación, por si hubiera una instalación anterior de LXD basada en dev, se migra la estructura de contenedores existente:

```
$ sudo lxd.migrate
```

Aunque la migración no sea necesaria, al intentarla nos solicitará eliminar los antiguos contenedores, y es lo que hay que hacer. Reiniciamos el sistema.

Tras la migración lanzamos el init de LXD: (si no estamos como root, usar sudo):

```
$ sudo lxd init
```

Nos realizará una serie de preguntas, es necesario configurar una interfaz de red en modo bridge y un pool de almacenamiento. Para conjure –up es importante respetar que el bridge se denomine ‘lxdbr0’ y el pool ‘default’.

```
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (btrfs, ceph, dir, lvm, zfs) [default=zfs]:
Create a new ZFS pool? (yes/no) [default=yes]:
Would you like to use an existing block device? (yes/no) [default=no]:
Size in GB of the new loop device (1GB minimum) [default=100GB]:
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, "auto"; or "none") [default=auto]:
10.8.8.1/24
Would you like LXD to NAT IPv4 traffic on your bridge? [default=yes]:
What IPv6 address should be used? (CIDR subnet notation, "auto" or "none") [default=auto]: none
Would you like LXD to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
```

A todas las preguntas respondemos con un enter aceptando las opciones por defecto, hasta llegar a la que nos interroga por la IP, que responderemos con la IP 10.8.8.1/24; y luego cuando nos pregunta qué dirección IPv6 debemos usar le decimos que ninguna, *none*.

Por último realizamos la instalación de conjure-up:

```
$ sudo apt update && sudo apt upgrade
$ sudo apt install snapd
$ source /etc/profile.d/apps-bin-path.sh
$ sudo snap install conjure-up --classic
```

5) Y una vez hallamos instalado conjure-up ya estamos listos para instalar y desplegar OpenStack. Lanzamos la utilidad:

```
$ conjure-up
```

Se nos mostrará el primer menú donde elegimos ‘*OpenStack with LXD*’.

Se nos presentarán cinco cuestiones:

- 1) Seleccionar una acción del script conjure: ‘*OpenStack with Nova LXD*’.
- 2) La nube en la que se desea desplegar (decimos que es una nueva) y el servidor: ‘*Configure a New Cloud to live on localhost*’.
- 3) Cuales son el bridge y el pool: ‘*lxdbr0*’ y ‘*default*’.
- 4) Cual es la ruta de la clave pública SSH: ‘*~/ssh/id_rsa.pub*’.
- 5) Otras configuraciones: las dejamos por defecto.

Pulsamos en *Deploy* y conjure-up comenzará su magia, en total desplegará 20 servicios. La instalación tarda aproximadamente una hora y utiliza bastante espacio en disco y E/S porque está ejecutando todos los procesos para todos los nodos del clúster en los contenedores en local.

Al final, si todo ha ido bien, nos presentará una pantalla con información donde están las credenciales de acceso y cuál es la ruta del portal Horizon, como se muestra en la figura 3-8:

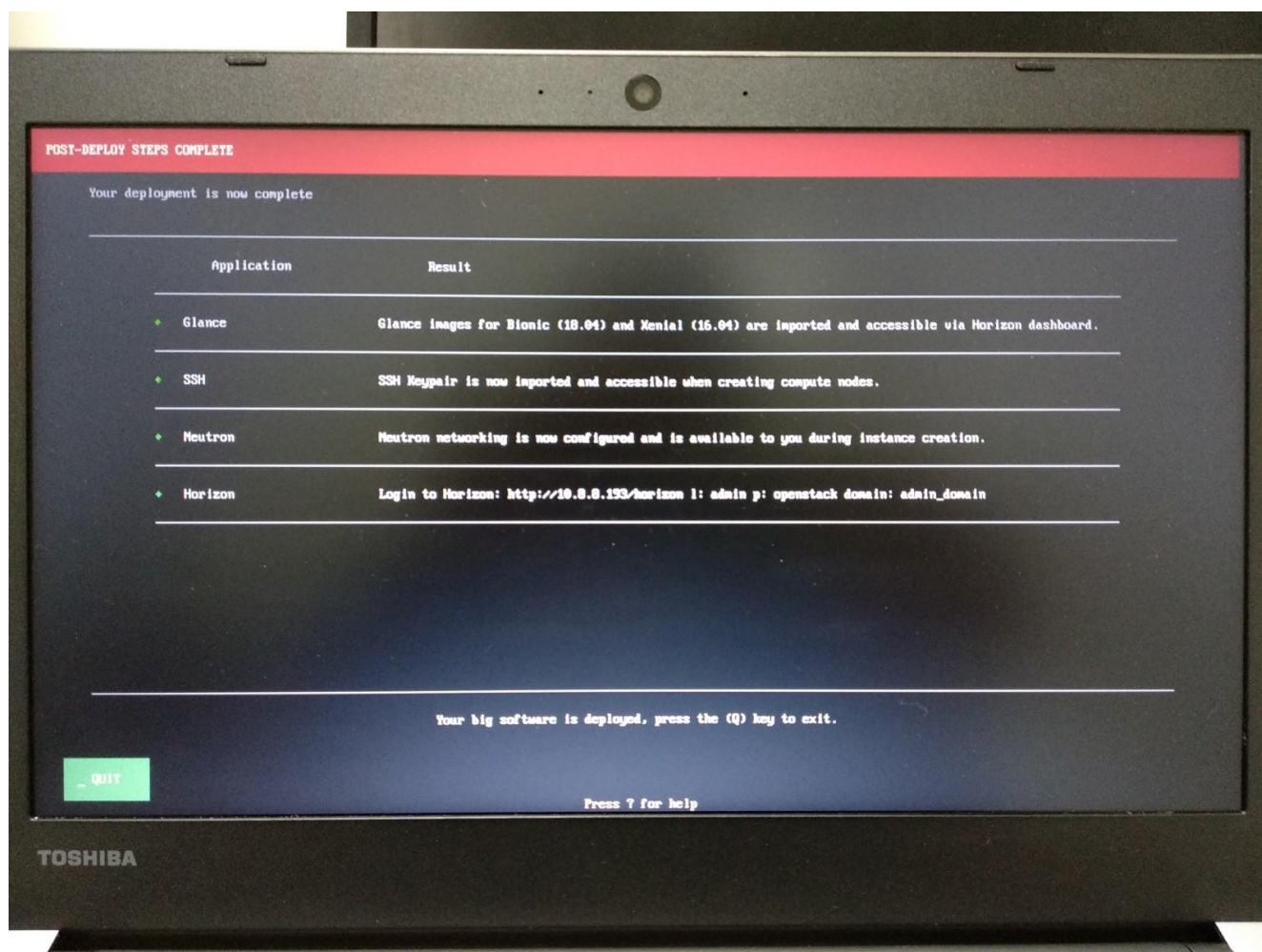


Figura 3-8. Pantalla final del script conjure-up.

La figura no es muy afortunada, pero puede verse más o menos la línea donde dice:

◆ Horizon Login to Horizon: <http://10.8.8.193/horizon> l: admin p: openstack domain: admin_domain

Como hemos instalado el ecosistema en una red interna en nuestro *localhost* ahora solo es accesible desde un navegador en la misma máquina. Debemos configurar el SO para que enrute internamente desde la interfaz al exterior, la que se encuentra en la red LAN, en mi caso es la tarjeta wifi en la IP 192.168.1.249 (estática), hacia la red interna 10.8.8.0/24 donde está el ecosistema OpenStack.

Para eso editaremos el archivo `/etc/sysctl.conf` y descomentamos la línea:

`net.ipv4.ip_forward=1.`

A continuación en el *router* de la red LAN hay que crear una regla de enrutamiento que redirija todo el tráfico hacia la red 10.8.8.0/24 hacia la IP de la máquina que contiene OpenStack, 192.168.1.249, en mi caso. (Cada *router* se configura de una manera pero todos son parecidos).

Ahora si, vamos a un navegador de una máquina de la red y abrimos la url `http://10.8.8.193/horizon`, se mostrará la página de *login* de Horizon como se muestra en la figura 3-9:

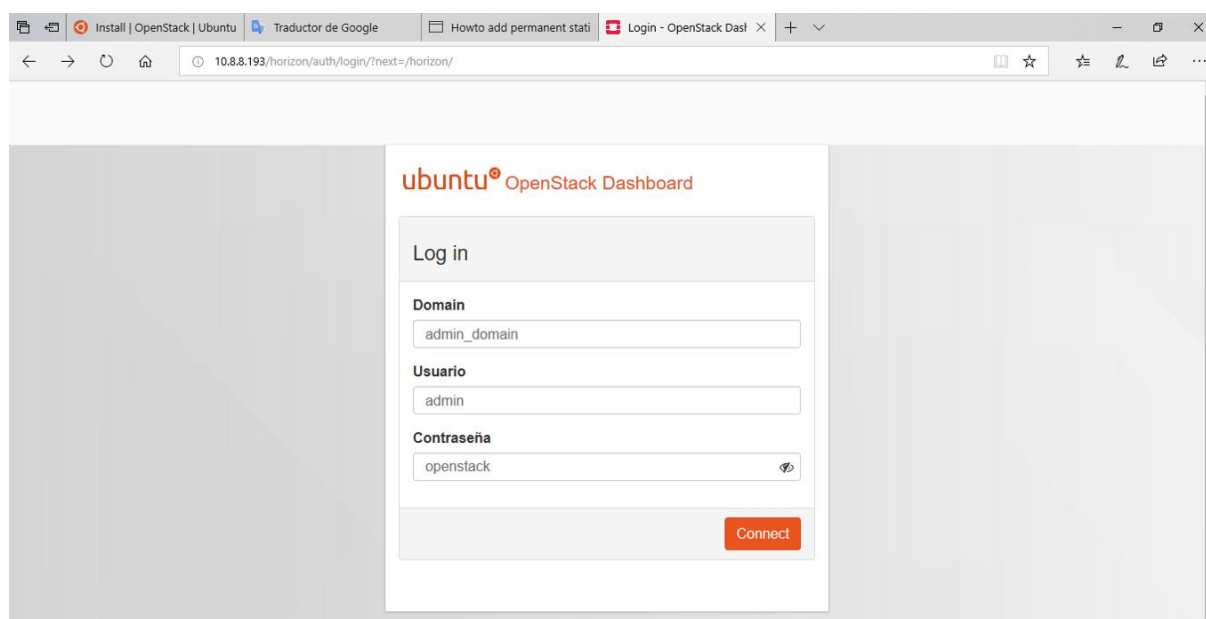


Figura 3-9. Portal Horizon, página de login.

La figura 3-10 muestra el resultado de hacer login.

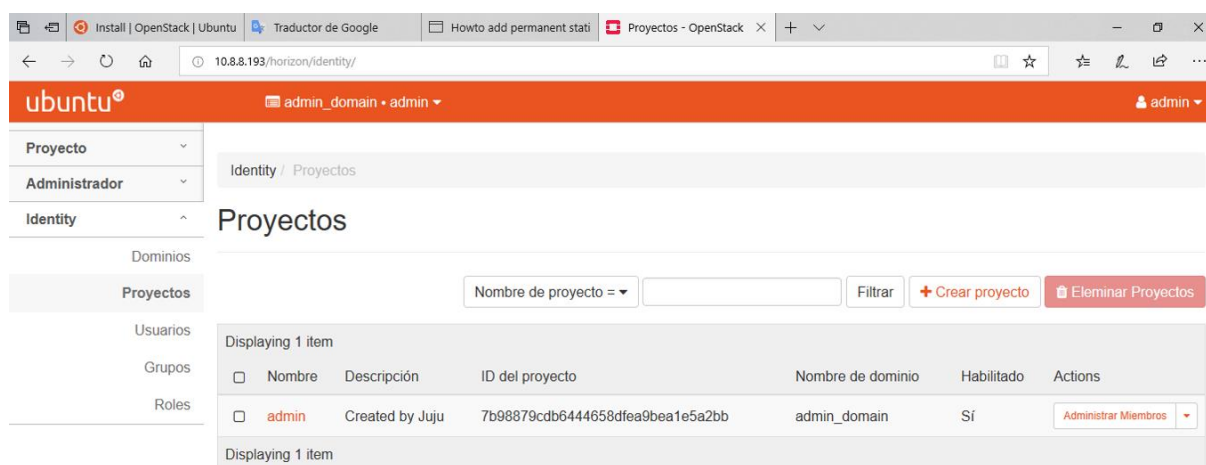


Figura 3-10. Portal Horizon tras hacer login.

Igual que en las pruebas de instalación realizadas con CentOS y RHEL, ya tenemos una nube privada en servicio accesible a la red LAN. La securización de esta máquina será muy complicada, como muestra

al listar las interfaces de red veremos que hay más de 40, todas creadas por *conjure-up*. Sí, es posible robustecer el SO pero probablemente a cada uno de los módulos de OpenStack nos cueste aplicarles las medidas de seguridad recomendadas, y además hay que hacer *hardening* del sistema de contenedores LXD.

3.3.2 Instalación *all-in-one* sobre Ubuntu con DevStack

La siguiente instalación que quería hacer es OpenStack sobre Ubuntu con DevStack. Para esta instalación intentaré usar una máquina virtual sobre un *host* i5U con 16 Gb de RAM y HDD normal, y la MV la especificaré con 6 Gb de RAM y un solo procesador de dos núcleos.

Comenzamos como las veces anteriores por crear la máquina virtual e instalar el SO. Seguiremos las instrucciones del sitio oficial de OpenStack DevStack [58] para realizar una instalación mínima del ecosistema.

Recordando lo ya dicho anteriormente, DevStack son una serie de *scripts* que se utilizan para desplegar rápida e inatendidamente un ecosistema OpenStack y también para realizar pruebas funcionales sobre el ecosistema desplegado.

Hay que crear un usuario ‘stack’ (no root), a continuación se le otorgan permisos agregándolo al grupo de sudoers (usuario stack con sudo habilitado), y cambiamos en la Shell a este nuevo usuario para realizar el clonado de DevStack desde github. Antes de clonar el repositorio comprobamos que no necesitamos actualizar git en nuestro sistema, y a continuación clonamos, con lo que crearemos el directorio /opt/stack/DevStack donde estará el repositorio de scripts.

```
$ sudo useradd -s /bin/bash -d /opt/stack -m stack
$ echo "stack ALL = (ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
$ sudo su - stack
$ sudo apt install git
$ git clone https://git.openstack.org/openstack-dev/devstack
$ cd devstack
```

El repositorio devstack contiene un *script* que instala OpenStack y plantillas para archivos de configuración, pero hay que crear un archivo nombrado local.conf conteniendo como mínimo cuatro contraseñas:

```
[[local | localrc]]
ADMIN_PASSWORD = secreto
DATABASE_PASSWORD = $ ADMIN_PASSWORD
RABBIT_PASSWORD = $ ADMIN_PASSWORD
SERVICE_PASSWORD = $ ADMIN_PASSWORD
```

Iniciamos la instalación lanzando el *script* stack.sh:

```
$ ./stack.sh
```

La instalación finalizará tras un ratito y como en las instalaciones anteriores ya tendremos OpenStack funcionando, como se muestra en la figura 3-11 se da información sobre la instalación, direcciones de los servicios, credenciales de acceso, y como anécdota el tiempo que ha llevado la instalación: 44,86 minutos.

```
test_with_retry      3
apt-get-update       2
osc                  197
wait_for_service     29
git_timed            918
dbsync               83
pip_install          514
apt-get              343
-----
Unaccounted time     573
=====
Total runtime        2692

This is your host IP address: 192.168.1.150
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.1.150/dashboard
Keystone is serving at http://192.168.1.150/identity/
The default users are: admin and demo
The password: secret

WARNING:
Using lib/neutron-legacy is deprecated, and it will be removed in the future

Services are running under systemd unit files.
For more information see:
https://docs.openstack.org/devstack/latest/systemd.html

DevStack Version: rocky
Change: 14782ceb6d3e44010842095b8d12b45b868b80d9 Merge "Fixed git show for upper-constraints.txt in
lib/tempest" into stable/rocky 2018-09-24 09:00:27 +0000
OS Version: Ubuntu 18.04 bionic

2018-10-16 11:13:13.795 | stack.sh completed in 2692 seconds.
stack@ubuntuopenstack:~/devstack$ _
```

Figura 3-11. Final de la instalación de OpenStack con DevStack.

DevStack habrá instalado los módulos Keystone, Glance, Nova, Cinder, Neutrón y Horizon (faltaría Swift para completar nuestro prototipo, el que decidimos en la decisión 1). También habrá creado las interfaces de red y se podrá acceder a Horizon para administrar VM, redes, volúmenes, etc.

Además con los *scripts* de DevStack se puede generar un canal ‘openrc’ para administrar OpenStack por línea de comandos. Las pruebas contenidas en la distribución de DevStack se encuentran en /opt/stack/tempest.

3.4 Virtualización de un clúster de OpenStack.

Tras las experiencias de instalación de OpenStack *all-in-one* pasaremos a mayores, a la instalación de un clúster. En este apartado abordaremos una instalación sobre máquinas virtuales en VirtualBox. Para la arquitectura que nos propusimos con la decisión 1 será necesario instalar los siguientes nodos en el clúster: Un nodo de control, uno o varios nodos de computación y un nodo de networking. Adicionalmente habría que instalar (no hay tiempo) los nodos de almacenamiento de bloques y de objetos.

Esta instalación mínima sobre servidores virtualizados está prevista en la documentación de OpenStack. Evidentemente si los servidores virtuales corren en una plataforma como es un portátil, el resultado será un prototipo que no es apto para un sistema en producción. Pero el proceso de instalación que iremos completando es exactamente igual que el de una instalación en servidores físicos de características suficientes para la puesta en producción de la nube privada que pretendemos.

Como mínimo deberemos instalar dos servidores (virtuales), uno para actuar como nodo de control y otro como un nodo de computación, en ellos entraremos más en profundidad en este epígrafe.

Tras instalar estos servidores virtuales podemos ‘escalar’ (agrandar) el clúster de servidores virtuales añadiendo otros nodos de computación y nodos de almacenamiento de bloques y de objetos, y añadir telemetría, módulos específicos para BigData, etc... (ya hemos hablado de la modularidad y escalabilidad de OpenStack). También sería posible añadir más nodos de control, pero esto es una acción enfocada a la extensión en un área geográfica extensa de una implementación *cloud*, digamos que no tiene sentido hacerlo en un prototipo basado en servidores virtuales independientes.

En todo caso a lo más que llegaríamos en este prototipo es a instalar un nodo de almacenamiento de bloques y uno de almacenamiento de objetos por lo que finalmente se compondría de cuatro servidores virtuales, pero por falta de tiempo, y porque el producto (4 servidores virtuales) resultará muy ‘voluminoso’ para hacerlo funcionar en un portátil, tan solo apuntaremos las referencias para poder instalarlos partiendo de las guías oficiales de OpenStack. Una opción para instalar estos componentes opcionales es hacerlo en un segundo portátil también sobre máquinas virtuales y conectarlos debidamente.

Además de los servidores virtuales el clúster tendrá dos redes:

- Red de Internet. El router de acceso a Internet que estoy usando para estas prácticas tiene la siguiente configuración LAN para la oficina:
 - Red: 192.168.1.0/24
 - Puerta de enlace: 192.168.1.1
 - Rango para DHCP: 192.168.1.33 a 192.168.1.199
 - DNS del ISP. El router solo resuelve nombres cuando los clientes se configuran por DHCP
 - Reglas de *routing*: IPv4; Dist 10.8.8.0/24; *Gateway* 192.168.1.249. Con esta regla el *router* redirige todo el tráfico que recibe con origen/destino de la red interna 10.8.8.0/24 a la dirección de la red LAN 192.168.1.249, en la que pondré mi portátil para que lleguen los paquetes a las MV,s.

Utilizaré el rango de direcciones 192.168.1.220 a 192.168.1.240, para conectar los servidores virtuales a Internet directamente.
- Red interna. Esta red enlazará los nodos del clúster:
 - Red: 10.8.8.0/24
 - Puerta de enlace: 10.8.8.1 (ojo que las MV,s que vaya creando no pueden tener configuradas dos *Gateway*, y no configuraré el de esta red hasta que no precise enviar paquetes de la red de administración fuera de ella, por ejemplo a Internet.
 - DHCP: Desactivado

Lo primero en un servidor es instalar el SO, Linux Ubuntu Server 18.04 LTS de acuerdo con los datos de la tabla 3-4:

Idioma:	Español
Teclado:	Español
Zona geográfica:	Europa/Madrid
Nombre de la máquina (hosts):	controller
Nombre de usuario:	Jose
Login del usuario:	jose
Password del usuario:	josejose
Particionado:	SWAP.....6.1Gb
	/boot 2.0Gb
	/ 41.9Gb

Tabla 3-4. Datos de instalación del SO del nodo de control.

Ni las credenciales, ni el particionado cumplen requisitos de seguridad, pero para evolucionar rápido en la práctica asumimos los datos de la tabla. En general tras la instalación del sistema operativo desde DVD o USB y sin conexión a Internet haríamos un bastionado del mismo, y cuando ya estuviera seguro, de acuerdo con el propósito del sistema lo conectaríamos a Internet o a una red interna donde estuvieran los repositorios para instalación de cualquier componente. Todo esto lo asumimos como lo que debería ser y no estamos cumpliendo.

3.4.1 Nodo de control y de red.

Este nodo ejecuta servicios generales (o de soporte) y servicios específicos.

Los generales son:

- NTP, servidor de tiempos para el clúster.
- Base de datos SQL. Optaremos por una instalación de MaríaDB.
- Mensajería. Mediante Rabbit.
- Un servidor web. El más típico, Apache2.

Los servicios específicos son:

- Servicio de identidad. Instalación de Keystone.
- Servicio de imágenes. Instalación de Glance.
- Gestión de nodos de computación.
- Gestión de nodos de *networking*.
- Agentes de red:
 - *Plugin ML2*.
 - *Linux bridge agent*.
 - DHCP.
 - Metadatos.
 - Utilidades de red Linux
- Portal Horizon.
- Opcionalmente puede correr partes de los servicios de almacenamiento de bloques (Cinder), almacenamiento de objetos (Swift), servicios de orquestación (Heat) y de servicios de telemetría (Celiometer).

En este caso vamos a instalar el nodo de control sobre una VM en VirtualBox y de acuerdo con la documentación oficial de OpenStack de referencia [65], para una prueba de concepto bastaría con configurarla con 1 CPU, 4 Gb de RAM, 5 Gb de disco y 2 tarjetas de red. La MV la configuraré con algo más de capacidad: 2 CPU con aceleración predeterminada, 6 Gb de RAM, 50 Gb de HDD y 2 tarjetas de red.

Consideramos instalado y securizado el SO y configuraremos las interfaces de red. De acuerdo con la decisión 3 una la configuraremos en modo ‘adaptador puente’ con ip estática en el rango de la red de acceso a Internet, y la otra en una red aislada interna para el clúster.

Red de Internet. Configuración del nodo de control (controller): La IP del nodo será 192.168.1.220; máscara /24; puerta de enlace 192.168.1.1; DNS de google 8.8.8.8 y 8.8.4.4.

Red interna. Configuración del nodo de control: La IP del nodo será 10.8.8.11; máscara /24; no le pongo puerta de enlace en esta red, ni resolución.

A continuación configuramos las interfaces de red como ya hemos hecho anteriormente en los sistemas all-in-one mediante el archivo de texto situado en /etc/netplan/01-netcfg.yaml, cuyo contenido debe parecerse al siguiente:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
      dhcp6: no
      addresses: [10.8.8.11/24, ]
#   gateway4: 10.8.8.1           (dejo estas líneas comentadas)
#   nameservers:
#     addresses: [10.8.8.1]
    enp0s8:
      dhcp4: no
      dhcp6: no
      addresses: [192.168.1.220/24, ]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
```

Se guarda el archivo con los cambios y se aplican en netplan:

```
$ sudo netplan apply
```

Ya debemos tener conexión a Internet. Lo primero será actualizar nuestro sistema y de paso instalar traceroute. Instalar aplicaciones innecesarias desde el punto de vista de la seguridad no es lo que hay que hacer si se quiere mantener la menor superficie de ataque, pero necesitare traceroute para depurar problemas de configuración de red.

Para la resolución de nombres en el nodo de control editamos el archivo /etc/hosts y declaramos las entradas:

```
# 127.0.1.1      localhost
10.8.8.11      controller
10.8.8.21      network
10.8.8.31      compute1
10.8.8.32      compute2
10.8.8.41      block1
10.8.8.51      object1
10.8.8.52      object2
```

Siguiente paso. El siguiente paso es instalar el servicio NTP.

Para sincronizar correctamente los servicios entre los nodos, Openstack.org recomienda Chrony para implementar este servicio. Realmente el nodo controlador debería sincronizarse horariamente contra un servidor más preciso que el mismo, y el resto de nodos del clúster deben sincronizarse con el nodo controlador. Instalamos:

```
# apt install chrony
```

A continuación editamos el archivo /etc/chrony/chrony.conf file y en la línea donde dice:

```
server NTP_SERVER iburst
```

... ponemos en lugar de NTP_SERVER el nombre del servidor NTP (o la dirección IP) contra el que se sincronizará nuestro nodo controlador. Podrían especificarse varios servidores NTP, y en ese caso habría que poner varios, uno en cada línea.

También pueden ser *pools* de servidores, la línea o directiva es:

```
pool NTP_DIRECCIÓN iburst maxsources NUM
```

Como no tengo servidores NTP en mi intranet¹⁶ dejaré los *pools* que por defecto se han creado en la instalación de Chrony, y que están en Internet:

```
pool ntp.ubuntu.com      iburst maxsources 4
pool 0.ubuntu.pool.ntp.org iburst maxsources 1
pool 1.ubuntu.pool.ntp.org iburst maxsources 1
pool 2.ubuntu.pool.ntp.org iburst maxsources 2
```

Para permitir que los nodos del clúster se sincronicen con el Chrony añadimos la siguiente línea:

```
allow 10.8.8.0/24
```

Por último reseteamos el servicio NTP:

```
# service chrony restart
```

Chrony se comunicará por el puerto UDP 123.

Siguiente paso. Hay que habilitar el repositorio de la versión de OpenStack en el nodo controlador para obtener fácilmente los paquetes necesarios:

```
# apt install software-properties-common
```

... (si hemos actualizado el sistema hace poco probablemente ya esté en su versión más reciente).

```
# add-apt-repository cloud-archive:rocky
```

Para finalizar este paso instalamos el cliente Python de OpenStack.

```
# apt update && apt upgrade
```

...(si hemos actualizado el sistema hace poco no es necesario)

```
# apt install python-openstackclient
```

Siguiente paso. Instalar un servidor de base de datos SQL siguiendo la referencia [66].

```
# apt install mariadb-server python-pymysql
```

Con lo que habremos instalado los dos paquetes, el servidor MariaDB y el script de Python.

Creamos el archivo /etc/mysql/mariadb.conf.d/99-OpenStack.cnf con el siguiente contenido:

```
[mysqld]
bind-address = 10.8.8.11
default-storage-engine = innodb
innodb_file_per_table = on
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

¹⁶ Este es otro de los motivos para instalar el sistema con conexión a Internet, no solo habría que instalar servidores *mirror*, también un servidor NTP con reloj de precisión en nuestra red corporativa.

... y finalizamos reseteando el servidor y lanzando el *script* de securización del servidor (para el usuario *root* del servidor de BD usará *password = toor*).

```
# service mysql restart
# mysql_secure_installation
```

Antes de continuar con el siguiente paso, daremos un vistazo a las contraseñas que los distintos servicios usarán para sus bases de datos. La referencia a seguir es la documentación oficial de la sección *environment-security* [67], en la que se muestra la tabla 3-5:

Password name	Description
Database password (no variable used)	Root password for the database
ADMIN_PASS	Password of user admin
CINDER_DBPASS	Database password for the Block Storage service
CINDER_PASS	Password of Block Storage service user cinder
DASH_DBPASS	Database password for the Dashboard
DEMO_PASS	Password of user demo
GLANCE_DBPASS	Database password for Image service
GLANCE_PASS	Password of Image service user glance
KEYSTONE_DBPASS	Database password of Identity service
METADATA_SECRET	Secret for the metadata proxy
NEUTRON_DBPASS	Database password for the Networking service
NEUTRON_PASS	Password of Networking service user neutron
NOVA_DBPASS	Database password for Compute service
NOVA_PASS	Password of Compute service user nova
PLACEMENT_PASS	Password of the Placement service user placement
RABBIT_PASS	Password of RabbitMQ user OpenStack

Tabla 3-5. Lista de servicios que requieren contraseña y sus referencias en la documentación oficial de OpenStack [67]

Ni que decir tiene que en un sistema con vocación de seguro las contraseñas deben serlo, y las mostradas por defecto no se podrían utilizar, pero para facilitar la instalación no las modificaré.

Los módulos del ecosistema de OpenStack admiten diferentes métodos de seguridad, control de accesos por contraseña, cifrado de los mensajes de control, y otras políticas. En cuanto las contraseñas en la guía de instalación de OpenStack se utilizan *SERVICE_PASS* para hacer referencia a las contraseñas de la cuenta de servicio y *SERVICE_DBPASS* para hacer referencia a las contraseñas de la base de datos, como puede verse en la tabla 3-5.

Los otros servicios que hemos denominado generales, como son el de base de datos SQL o el de mensajería utilizan contraseñas en el control de accesos a los mismos.

La guía de instalación de OpenStack para facilitar el proceso solo especifica las contraseñas tal y como están referidas en la tabla 3-5.

Siguiente paso. Instalar el servidor de cola de mensajería.

OpenStack utiliza una cola de mensajes para coordinar las operaciones y la información de estado entre los servicios. El servicio de cola de mensajes normalmente se ejecuta en el nodo de control. La aplicación para este propósito aconsejada es RabbitMQ, así que procederemos a instalarlo en el nodo de control.

```
# apt install rabbitmq-server
# rabbitmqctl add_user openstack RABBIT_PASS
# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

Tras la instalación de la cola hemos creado el usuario ‘openstack’ y le damos los permisos necesarios.

Siguiente paso. Instalar el servicio Memcached.

Keystone utiliza este servicio para almacenar *tokens*. El servicio Memcached normalmente se ejecuta en el nodo del controlador donde se instala Keystone.

En un sistema orientado a producción la autenticación gestionada por el servicio de identidad Keystone debe configurarse combinando el método de autenticación, con el cifrado y las reglas oportunas del *firewall*.

```
# apt install memcached python-memcache
```

Editamos el archivo de configuración `/etc/memcached.conf` y configuramos el servicio para usar la ip del nodo de control (ip de la red interna de gestión: 10.8.8.11) sustituyendo la especificada por defecto que debe ser la 127.0.0.1.

```
-l 10.8.8.11
```

Por último reiniciamos el servicio:

```
# service memcached restart
```

Siguiente paso. Instalar el servicio Etcd.

Este servicio puede ser requerido por los módulos de OpenStack, así que lo instalaré. Se utiliza como un almacén de pares clave-valor y para otras cuestiones relacionadas con la gestión de los servicios específicos (depende de los que se usen).

```
# apt install etcd
```

Editamos el archivo de configuración `/etc/default/etcd` y configuramos las directivas como sigue:

```
ETCD_NAME="controller"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS="http://10.0.0.11:2379"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://10.0.0.11:2380"
ETCD_INITIAL_CLUSTER="controller=http://10.0.0.11:2380"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_ADVERTISE_CLIENT_URLS="http://10.0.0.11:2379"
```

Iniciamos el servicio:

```
# systemctl enable etcd
# systemctl start etcd
```

Hasta aquí ya tenemos instalado el entorno necesario para que podamos comenzar a desplegar los servicios de OpenStack. Para esto hay que seguir la guía de instalación de la versión correspondiente, en nuestro caso Rocky [68].

Siguiente paso. Instalar el servicio Keystone, siguiendo la guía de la referencia [69].

Keystone es el punto único para la autenticación de los usuarios y para autorizarlos ante los diferentes servicios de su catálogo (del catálogo de servicios que provee la nube, el cual es único y administrado por Keystone), por lo tanto es el primer servicio con el que éstos interactúan. De la misma manera los módulos de OpenStack instalados en el sistema utilizan Keystone para comprobar que los usuarios son quienes dicen ser y para interactuar entre ellos mismos. Keystone permite la integración con LDAP y con Active Directory.

El catálogo de servicios es el conjunto de servicios disponibles en una implementación concreta de OpenStack. Los servicios pueden ser de tres tipos: administrativo, interno o público.

Cada servicio, materializado por su módulo, puede tener uno o varios puntos finales, a modo de interfaces que se conectan a diferentes redes normalmente separadas de acuerdo con las políticas de seguridad. Pueden configurarse las siguientes redes separadas:

- Red de API pública: con visibilidad desde Internet o desde una red corporativa serviría para que los clientes o usuarios pudieran administrar sus nubes.
- Red de API de gestión: Para el uso restringido a los administradores de la infraestructura del *cloud*.
- Red de API interna: donde se conectan los *hosts* que contienen servicios de OpenStack.

En nuestra implementación conectaremos todas las interfaces a la red de gestión o administración que hemos creado en el nodo controlador, la red 10.8.8.0/24.

Otro de los parámetros configurables es el de ‘región’, disponiendo de varias regiones los servicios pueden resultar más escalables pues se publican dentro de ella para sus usuarios lo que permite diferenciar instancias del servicio por regiones.

En nuestra implementación solo configuraremos una región que denominaremos ‘RegionOne’.

Dentro del catálogo de servicios manejado por Keystone, cada tríada de ‘servicio-región-interfaz’ viene a ser un servicio de los publicados en la implementación (en particular) de OpenStack. Por ejemplo, el servicio de computación 1 sería Nova-RegionOne-10.8.8.31.

El servicio de Keystone se compone de:

- Servidor. Proporciona de forma centralizada los servicios de autenticación y autorización mediante una interfaz RESTful.
- *Drivers*. Se integran en el servidor y se utilizan para acceder a repositorios externos de gestión de identidades, como LDAP o cualquier base de datos.
- Módulos. Son *middleware* se ejecutan en distintos componentes de OpenStack que utilizan el servicio de identidad. Estos módulos interceptan las solicitudes de servicio, extraen las credenciales de usuario y las envían al servidor para su autorización. La conexión entre los módulos y sus componentes se realiza a través de una interfaz que se configura como *Gateway* de un servidor web específico levantado por Python (Python Web Server).

Comenzamos a instalarlo. Esta fase me ha dado varios problemas haciendo paso a paso lo que dice la referencia [69], por lo que he cambiado algunos comandos.

Lo primero es configurar la base de datos del servicio para lo que abrimos sesión en el servidor MariaDB del nodo de control y creamos la BD (hay que estar seguros de hacerlo desde el *prompt* del usuario y con privilegios de sudo).

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

Nos pide la contraseña de sudo, y a continuación la de root (usuario de BD) y entramos en el prompt del MariaDB server:

```
MariaDB [(none)]> CREATE DATABASE keystone;  
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost'  
IDENTIFIED BY 'KEYSTONE_DBPASS';  
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED  
BY 'KEYSTONE_DBPASS';  
MariaDB [(none)]>exit
```

A continuación instalamos Keystone. Ganamos privilegios de *root*, pero manteniéndonos en la ruta del usuario no privilegiado, no afecta a la instalación en sí, pero si al volcado de la base de datos que haremos posteriormente:

```
$ sudo su
```

... (no hacemos \$ sudo su -).

El servidor Apache 2 y un módulo son necesarios para este componente así que hay que instalarlos además de Keystone. El comando que se utiliza en la guía de instalación es:

```
# apt install keystone apache2 libapache2-mod-wsgi
```

Para nuestro caso da igual utilizar apt-get, es otro programa de instalación diferente. Maneja las dependencias de manera diferente, por lo que lo utilizo en este caso. No le especifico que me instale ‘apache2’ y ‘libapache2-mod-wsgi’ porque me los resolverá como dependencias de keystone (desde el punto de vista de la seguridad lo recomendable es instalar los paquetes sin sus dependencias y las dependencias una a una):

```
# apt-get install keystone
```

Podemos ver el estado del servicio Apache2 y comprobar que está activo y en /etc/apache2/mods-enabled/ vemos que el módulo wsgi está instalado. También en /etc/apache2/sites-enabled/ vemos como hay un archivo de configuración keystone.conf que lo que hace es levantar un virtual server a la escucha en el puerto 5000. Esta configuración de Apache2 la ha hecho la instalación misma de Keystone, por lo que al último paso de la instalación que consiste en configurar el nombre del servidor en /etc/apache2/apache2.conf con la directiva ‘ServerName controller’ ya no la realizaremos.

Configuramos el servicio de identidad editando /etc/keystone/keystone.conf. En el apartado [database] comentaremos la cadena de conexión y crearemos la nuestra:

```
[database]  
# ...  
#connection = sqlite:///var/lib/keystone/keystone.db  
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone  
  
En el apartado [token] descomentamos el provider:  
  
[token]  
# ...  
provider = fernet
```

Lanzamos la orden para crear las tablas de la base de datos Keystone, es decir completamos la base de datos del servicio:

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

En este caso el *script* ‘keystone-manage db_sync’ se ejecuta como ‘su’ con perfil de ‘root’.

Entramos en MariaDB server y comprobamos que se han creado las 44 tablas en la base de datos keystone.

Iniciamos los repositorios de claves Fernet:

```
# keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
# keystone-manage credential_setup --keystone-user keystone --keystone-group keystone
```

Arrancamos el servicio (el comando contiene la cadena de opciones).

```
# keystone-manage bootstrap --bootstrap-password ADMIN_PASS --bootstrap-admin-url
http://controller:5000/v3/ --bootstrap-internal-url http://controller:5000/v3/ --bootstrap-public-url
http://controller:5000/v3/ --bootstrap-region-id RegionOne
```

Por último configuraríamos el nombre del servidor en /etc/apache2/apache2.conf añadiendo la directiva:

```
ServerName controller
```

... y resetearíamos el servicio:

```
# service apache2 restart
```

Esto como hemos apuntado antes no es estrictamente necesario.

Desde el navegador del portátil podemos ver el controller (ojo a la resolución de nombres y el enrutamiento) o bien con un navegador Lynx desde la misma máquina.

Ya para finalizar configuraremos las variables de entorno para realizar comprobaciones, y añadir artefactos a la base de datos con la cuenta de admin:

```
$ export OS_USERNAME=admin
$ export OS_PASSWORD=ADMIN_PASS
$ export OS_PROJECT_NAME=admin
$ export OS_USER_DOMAIN_NAME=Default
$ export OS_PROJECT_DOMAIN_NAME=Default
$ export OS_AUTH_URL=http://controller:5000/v3
$ export OS_IDENTITY_API_VERSION=3
```

Al final estas variables las declararemos por comodidad ejecutando un *script* pues son dependientes de la Shell y la sesión y están en el ámbito del usuario no privilegiado.

Todo debe haber ido bien, haremos algunas comprobaciones y tareas.

Aunque cuando arrancamos el servicio Keystone ya creamos el dominio ‘default’ crearemos otro dominio a modo de ejemplo, y además crearemos un par de proyectos (un proyecto viene a ser un servicio que se ofrecerá a clientes finales) como se ve en la figura 3-12:

```
jose@controller:~$ openstack domain create --description "DominioEjemplo" example
+-----+-----+
| Field | Value |
+-----+-----+
| description | DominioEjemplo |
| enabled | True |
| id | 7ac6efb7590e4c5f8768916b735a08ce |
| name | example |
| tags | [] |
+-----+-----+
jose@controller:~$ openstack project create --domain default --description "Service Project" service
+-----+-----+
| Field | Value |
+-----+-----+
| description | Service Project |
| domain_id | default |
| enabled | True |
| id | 9fdb2232277845679ae2e31cc52ac2b2 |
| is_domain | False |
| name | service |
| parent_id | default |
| tags | [] |
+-----+-----+
jose@controller:~$ openstack project create --domain default --description "Demo Project" myproject
+-----+-----+
| Field | Value |
+-----+-----+
| description | Demo Project |
| domain_id | default |
| enabled | True |
| id | 84cf559f4d154962b960b5231d07b79d |
| is_domain | False |
| name | myproject |
| parent_id | default |
| tags | [] |
+-----+-----+
jose@controller:~$ _
```

Figura 3-12. Creación del dominio ‘example’ y de los proyectos ‘service’ y ‘myproject’.

Crearemos a continuación un usuario, ‘myuser’, y hay que establecer su contraseña pues con su valor vacío no se podría hacer login, (*password* = MYUSER_PASS). A continuación crearemos un rol (*role*), denominado ‘myrole’ y lo añadiremos al proyecto ‘myproject’ con el usuario ‘myuser’; todo ello como se muestra en la figura 3-13:

```
jose@controller:~$ openstack user create --domain default --password-prompt myuser
User Password:
Repeat User Password:
+-----+
| Field | Value |
+-----+
| domain_id | default |
| enabled | True |
| id | 000064f2043b4886a663f3e7bd549657 |
| name | myuser |
| options | {} |
| password_expires_at | None |
+-----+
jose@controller:~$ openstack role create myrole
+-----+
| Field | Value |
+-----+
| domain_id | None |
| id | 8c0d3021f3e64b27afadf757252c3b44 |
| name | myrole |
+-----+
jose@controller:~$ openstack role add --project myproject --user myuser myrole
jose@controller:~$
```

Figura 3-13. Creación de un usuario, un rol y vinculación de rol a proyecto.

Como puede verse en la figura 3-13 la última orden de vinculación no produce ninguna salida en la terminal.

Continuamos con verificaciones, vamos a solicitar al servicio de identidad un token de autenticación. Para ello primero ‘vaciamos’ primero las variables de entorno OS_AUTH_URL y OS_PASSWORD:

```
$ unset OS_AUTH_URL OS_PASSWORD
```

... y a continuación damos la orden para obtener un token de autenticación para el usuario, ‘myusers’ token. La figura 3-14 a continuación muestra la orden y el resultado.

```
jose@controller:~$ openstack --os-auth-url http://controller:5000/v3 --os-project-domain-name Default --os-user-domain-name Default --os-project-name myproject --os-username myuser token issue
Password:
+-----+
| Field | Value |
+-----+
| expires | 2018-10-23T07:57:22+0000 |
| id | gAAAAABbzS2SXDzrI6sFUWGu9nHj6EnMnYIURfY5i1NUpekYzC5t9_rMikPTZpfIiwkmbed0s4av0x6.jyppsCE7icZAjDtYjWM10eZnxdD1lyY5SVZ91iqT1YzpG1XX5-9PsNnzHQ3X8LXAxM36XoB70fipxKhFAMBxxT_CPwbcN2mRkLuxcsEY |
| project_id | 84cf559f4d154962b960b5231d07b79d |
| user_id | 000064f2043b4886a663f3e7bd549657 |
+-----+
jose@controller:~$ _
```

Figura 3-14. Solicitud de token de autenticación.

Por último, ya dijimos antes que en vez de establecer variables de entorno cada vez que vayamos a hacer alguna tarea administrativa automatizaríamos esto mediante *scripts*. Para esto seguiremos exactamente las instrucciones de la referencia [70] y crearemos los *scripts* ‘admin-openrc’ y ‘demo-openrc’. Para usarlos se lanzan desde el *prompt* de usuario no privilegiado, por ejemplo:

```
$ . admin-openrc
```

Y a continuación requerimos su token de autenticación:

\$ OpenStack token issue

La secuencia se reproduce en la figura 3-15 a continuación:

```
jose@controller:~$ . admin-openrc
jose@controller:~$ openstack token issue
+-----+-----+
| Field      | Value |
+-----+-----+
| expires    | 2018-10-23T08:23:13+0000 |
| id         | gAAAAABbzshq3t0i3BseUXPqPeE20F6NIT1LfNOh7sxKEJmsJSfjzQm-1usv1WlbC36oqOPDw1511YkctVu5 |
| project_id | 068d866c8b714c7c859d373096e33e2c |
| user_id    | 3370d40fc575406a9fb2f87413d7030e |
+-----+-----+
jose@controller:~$
```

Figura 3-15. Script de configuración de variables de entorno y solicitud de *token*.

Con esto damos por finalizada la instalación del servicio Keystone y comenzamos el siguiente paso.

Siguiente paso. Instalar el servicio Glance, siguiendo la guía de la referencia [71].

Glance presta a los usuarios el servicio de descubrimiento, registro y recuperación de las imágenes de máquinas virtuales.

Presenta una interfaz API REST para consultar los metadatos de las imágenes de MV,s almacenadas en el sistema OpenStack y recuperar una imagen para producción.

Las imágenes de MV,s pueden estar almacenadas en el sistema de archivos o en un nodo ‘almacén de objetos’ que corre el componente Swift.

Glance es un servicio fundamental para poder ofrecer IaaS. A través de su API recibirá las solicitudes no solo de los usuarios finales, también las que le envía el componente Nova (de computación), y por supuesto si utiliza el almacén de objetos, de Swift.

Como aún no tenemos instalado Swift (como es normal) apuntaremos a que el almacén de objetos es el servidor de archivos, concretamente en el directorio por defecto: /var/lib/glance/images/. El nodo controlador lo he creado con un disco de 50 Gb, por lo que el espacio disponible no es mucho.

Glance incluye los siguientes componentes:

- Interfaz API REST. Acepta las llamadas para descubrir, recuperar y almacenar imágenes de MV,s.
- Registro. Se preocupa de mantener (almacenar y procesar) la información precisa sobre las imágenes de MV,s en metadatos. Este servicio de Registro nunca debe quedar expuesto o disponible a los usuarios, es un servicio interno para que Glance funcione.
- Base de datos. Se utiliza una base de datos SQL para almacenar los metadatos de las imágenes.
- Repositorio o almacén de imágenes. Como ya hemos dicho se admiten varios tipos, el propio sistema de archivos, los almacenes de objetos (Swift), almacenamiento de bloques (Cinder), y otros almacenes de datos en Internet, en sistemas VMware, etc...

- Servicio de definición de metadatos. Es una interfaz común para que los administradores, los usuarios y los otros módulos de OpenStack puedan definir datos personalizados (los que ellos necesiten) de las imágenes, artefactos, volúmenes, sabores y agregados.

El primer paso para instalar Glance es crear la base de datos. Lanzamos el cliente de MariaDB que ya tenemos instalado en el nodo controlador y comenzamos: (hay que recordar los problemas que tuvimos con la creación de la base de datos de Keystone, y seguiré los mismos criterios).

```
$ sudo mysql -u root -p
MariaDB [(none)]> CREATE DATABASE glance;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost'
IDENTIFIED BY 'GLANCE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@ '%' IDENTIFIED BY
'GLANCE_DBPASS';
MariaDB [(none)]> exit
```

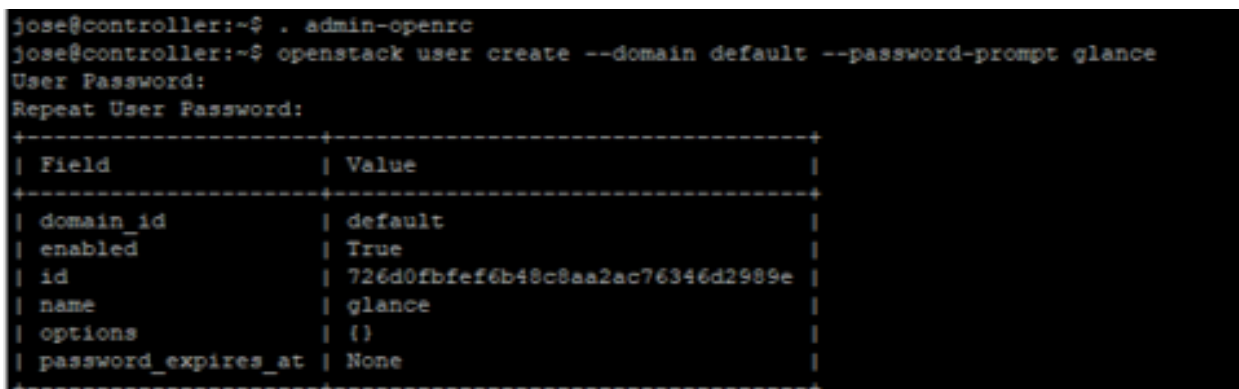
A continuación cargamos las variables de entorno del usuario 'admin'.

```
$ . admin-openrc
```

Ahora creamos el usuario glance:

```
$ openstack user create --domain default --password-prompt glance
```

Nos solicita la contraseña para este nuevo usuario dos veces. La contraseña que usaremos es 'GLANCE_PASS' (cuidado de no confundir con la del usuario de base de datos 'GLANCE_DBPASS'). La figura 3-16 muestra la acción.



```
jose@controller:~$ . admin-openrc
jose@controller:~$ openstack user create --domain default --password-prompt glance
User Password:
Repeat User Password:
+-----+-----+
| Field          | Value                                     |
+-----+-----+
| domain_id      | default                                 |
| enabled        | True                                    |
| id             | 726d0fbfef6b48c8aa2ac76346d2989e       |
| name           | glance                                 |
| options        | {}                                      |
| password_expires_at | None                                  |
+-----+-----+
```

Figura 3-16. Creación del usuario glance.

A continuación añadimos el rol de admin al proyecto de servicio y al usuario glance como se muestra en la figura 3-17 la ejecución del comando no produce salida en la Shell.

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

A continuación creamos un servicio con nombre 'glance'. La salida se muestra en la figura 3.17.

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

```
jose@controller:~$ openstack role add --project service --user glance admin
jose@controller:~$ openstack service create --name glance --description "OpenStack Image" image
+-----+
| Field      | Value                                     |
+-----+-----+
| description | OpenStack Image                         |
| enabled     | True                                    |
| id          | d3add8f7264f4a9a823c4f2ce67e5e51      |
| name        | glance                                  |
| type        | image                                   |
+-----+-----+
jose@controller:~$
```

Figura 3-17. Creación del servicio 'glance'.

A continuación creamos los 'API endpoints' del servicio de imágenes, mediante los comandos que se dictan. Los resultados podemos verlos en la figura 3-18.

```
$ openstack endpoint create --region RegionOne image public http://controller:9292
$ openstack endpoint create --region RegionOne image internal http://controller:9292
$ openstack endpoint create --region RegionOne image admin http://controller:9292
```

```
jose@controller:~$ openstack endpoint create --region RegionOne image public http://controller:9292
+-----+
| Field      | Value                                     |
+-----+-----+
| enabled     | True                                    |
| id          | bbe19d4aacc2426a97a82d86862dc864      |
| interface    | public                                  |
| region      | RegionOne                              |
| region_id   | RegionOne                              |
| service_id  | d3add8f7264f4a9a823c4f2ce67e5e51      |
| service_name | glance                                  |
| service_type | image                                   |
| url         | http://controller:9292                 |
+-----+-----+
jose@controller:~$ openstack endpoint create --region RegionOne image internal http://controller:9292
+-----+
| Field      | Value                                     |
+-----+-----+
| enabled     | True                                    |
| id          | 56bd16f3172241e1b6040660c2d76701      |
| interface    | internal                                |
| region      | RegionOne                              |
| region_id   | RegionOne                              |
| service_id  | d3add8f7264f4a9a823c4f2ce67e5e51      |
| service_name | glance                                  |
| service_type | image                                   |
| url         | http://controller:9292                 |
+-----+-----+
jose@controller:~$ openstack endpoint create --region RegionOne image admin http://controller:9292
+-----+
| Field      | Value                                     |
+-----+-----+
| enabled     | True                                    |
| id          | 766e30ee0bfe42218407567533c1cca8      |
| interface    | admin                                   |
| region      | RegionOne                              |
| region_id   | RegionOne                              |
| service_id  | d3add8f7264f4a9a823c4f2ce67e5e51      |
| service_name | glance                                  |
| service_type | image                                   |
| url         | http://controller:9292                 |
+-----+-----+
jose@controller:~$
```

Figura 3-18. Configuración de los API endpoints para el servicio de imágenes.

A continuación instalaremos el módulo Glance. Escalamos privilegios pero sin dejar el perfil del usuario no privilegiado y usará apt-get en vez de apt:

```
$ sudo su
# apt-get install glance
```

Configuramos el servicio de imágenes editando `/etc/glance/glance-api.conf`. En el apartado `[database]` comentaremos la cadena de conexión y crearemos la nuestra:

```
[database]
# ...
#connection = sqlite:///var/lib/glance/glance.sqlite
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone
backend = sqlalchemy (no olvidar dejar esto descomentado)
```

En el apartado `[token]` descomentamos el provider:

```
[token]
# ...
provider = fernet
```

En el apartado `[glance_store]` configuramos el sistema de archivos local como almacén de imágenes:

```
[glance_store]
# ...
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

Editamos ahora el archivo `/etc/glance/glance-registry.conf` y modificamos las secciones como se indica:

```
[database]
# ...
#connection = sqlite:///var/lib/glance/glance.sqlite
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone
backend = sqlalchemy (no olvidar dejar esto descomentado)

[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = GLANCE_PASS (que no es la contraseña del usuario de BD)

[paste_deploy]
# ...
flavor = keystone
```

Lanzamos la orden para crear las tablas de la base de datos glance, es decir completamos la base de datos del servicio de imágenes:

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

En este caso el *script* ‘glance-manage db_sync’ se ejecuta como ‘su’ con perfil de ‘root’.

Entramos en MariaDB server y comprobamos que se han creado las 15 tablas en la base de datos glance.

Reseteamos el servicio:

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

Desde el navegador del portátil podemos ver en el controller las url de las API de Glance (ojo a la resolución de nombres y el enrutamiento) o bien con un navegador Lynx desde la misma máquina.

Vamos a verificar el funcionamiento cargando una imagen de máquina virtual. Usaremos una imagen de ‘cirros’ porque pesa muy poco tanto en tamaño como en consumo de recursos.

Cargamos las variables de entorno:

```
$ . admin-openrc
```

Descargamos el archivo con la imagen de cirros con wget:

```
$ wget http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img
```

Subimos la imagen a glance con formato de disco QCOW2, formato ‘bare’ para el contenedor y con parámetro de visibilidad ‘public’ para que se vea desde cualquier proyecto.

```
$ openstack image create "cirros" --file cirros-0.4.0-x86_64-disk.img --disk-format qcow2 --
container-format bare --public
```

Se nos mostrará una salida con los parámetros y datos de la creación y con el comando:

```
$ openstack image list
```

... nos lista la imagen recién creada.

```
+-----+-----+-----+
| ID                               | Name | Status |
+-----+-----+-----+
| 38047887-61a7-41ea-9b49-27987d5e8bb9 | cirros | active |
+-----+-----+-----+
```

Siguiente paso. Instalar los componentes del servicio de computación Nova que deben estar en el nodo de control.

El servicio de computación se instala sobre otra máquina como nodo de computación, pero es en realidad un servicio distribuido entre las máquinas del nodo.

Los nodos de computación (puede haber varios) interactúan con el servicio de identidad (Keystone) en el nodo de control para la autenticación y autorizaciones del acceso. Interactúa con el servicio de imágenes (Glance) también en el nodo de control, para proporcionar al usuario final discos y MV,s como servicio (IaaS). Interactúa con el portal Horizon para que los usuarios accedan a los servicios y los administradores administren.

Los nodos de computación establecen los límites pues el acceso a las imágenes está limitado por proyectos y por usuarios mediante cuotas. Por ejemplo la cuota por proyecto limita el número de instancias que se pueden lanzar. Se pueden añadir nuevos nodos de computación a un clúster, por lo que son escalables.

Nova tiene los siguientes componentes:

- Nova-api (lo instalaremos en el nodo de control). Viene a ser la API de cálculo final de los usuarios, acepta sus peticiones e inicia las actividades de orquestación como lanzar una instancia.
- Nova-api-metadata. Sirve los metadatos a las instancias que las solicitan. El servicio se usa normalmente cuando se ejecuta en modo de *host* múltiple con instalaciones de nova-network.

- Nova-compute. Es un demonio que crea y finaliza las instancias de máquinas virtuales a través de la API de hipervisor (del que la que sea, XenAPI para XenServer/XCP, libvirt para KVM o QEMU, VMwareAPI para VMware,...).
En nuestro caso interactuará con libvirt aceptando las peticiones de una cola y ejecutando comandos del sistema, como lanzar una instancia de KVM y actualizar su estado en la base de datos.
- Nova-placement-api (lo instalaremos en el nodo de control). Explora el inventario y uso de cada proveedor.
- Nova-scheduler (lo instalaremos en el nodo de control). Adquiere las solicitudes de instancia de máquina virtual de la cola y determina en qué *host* se ejecuta.
- Nova-conductor (lo instalaremos en el nodo de control). Es un módulo *middleware* entre el servicio nova-compute y la base de datos. Elimina los accesos directos a la base de datos en la nube realizados por el servicio nova-compute y no se debe implementar en el mismo nodo donde corre (nova-compute).
- Nova-novncproxy daemon (lo instalaremos en el nodo de control). Proporciona un *proxy* para acceder a las instancias en ejecución a través de una conexión VNC. Soporta clientes novnc.
- Nova-spicehtml5proxy daemon. Proporciona un *proxy* para acceder a las instancias en ejecución a través de una conexión SPICE. Es compatible con el cliente HTML5 basado en navegador.
- Nova-xvvpncproxy daemon. Proporciona un *proxy* para acceder a las instancias en ejecución a través de una conexión VNC. Admite un cliente Java específico de OpenStack.
- *The queue* (la cola). Ya lo hemos visto antes, pasa mensajes entre demonios. Generalmente implementado con RabbitMQ.
- Base de datos SQL. Nova puede admitir cualquier base de datos que admita SQLAlchemy. Las bases de datos comunes son SQLite3 para trabajos de prueba y desarrollo, MySQL, MariaDB y PostgreSQL. Nosotros estamos usando MariaDB como ya es sabido.

A continuación instalaremos los componentes en el nodo controlador para lo que seguiremos la referencia [72]. Lo primero es crear las bases de datos. Lanzamos el cliente de MariaDB que ya tenemos instalado en el nodo controlador y comenzamos: (hay que recordar los problemas que tuvimos con la creación de la base de datos de keystone, y seguiré los mismos criterios).

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

Crearemos 4 bases de datos y a continuación estableceremos los permisos del usuario nova sobre ellas:

```
MariaDB [(none)]> CREATE DATABASE nova_api;
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> CREATE DATABASE nova_cell0;
MariaDB [(none)]> CREATE DATABASE placement;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost'
IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' IDENTIFIED BY
'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED
BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY
'NOVA_DBPASS';
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'localhost' IDENTIFIED BY 'PLACEMENT_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%' IDENTIFIED BY 'PLACEMENT_DBPASS';
MariaDB [(none)]> exit
```

Siguiente paso. Cargamos las variables de entorno del usuario ‘admin’.

```
$ . admin-openrc
```

Ahora creamos el usuario nova (no confundirlo con el usuario de BD):

```
$ openstack user create --domain default --password-prompt nova
```

Nos solicita la contraseña para este nuevo usuario dos veces. La contraseña que usaremos es ‘NOVA_PASS’ (cuidado de no confundir con la del usuario de base de datos ‘NOVA_DBPASS’). La salida del comando es parecida a la de la figura 3-16.

A continuación añadimos el rol de admin al proyecto de servicio y al usuario nova como hicimos anteriormente con el usuario glance, por ejemplo.

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

A continuación creamos un servicio con nombre ‘nova’.

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

A continuación creamos los ‘API endpoints’ del servicio de computación, mediante los comandos que se dictan.

```
$ openstack endpoint create --region RegionOne compute public http://controller:8774/v2.1
$ openstack endpoint create --region RegionOne compute internal http://controller:8774/v2.1
$ openstack endpoint create --region RegionOne compute admin http://controller:8774/v2.1
```

Ahora creamos el usuario placement (no confundirlo con el usuario de BD):

```
$ openstack user create --domain default --password-prompt placement
```

Nos solicita la contraseña para este nuevo usuario dos veces. La contraseña que usaremos es ‘PLACEMENT_PASS’ (cuidado de no confundir con la del usuario de base de datos ‘PLACEMENT_DBPASS’). La salida del comando es parecida a la de la figura 3-16.

A continuación añadimos el rol de admin al proyecto de servicio y al usuario nova como hicimos anteriormente con el usuario glance, por ejemplo.

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

A continuación creamos un servicio con nombre ‘placement’.

```
$ openstack service create --name placement --description "Placement API" placement
```

A continuación creamos los ‘API endpoints’ del servicio de computación, mediante los comandos que se dictan.

```
$ openstack endpoint create --region RegionOne placement public http://controller:8778
$ openstack endpoint create --region RegionOne placement internal http://controller:8778
$ openstack endpoint create --region RegionOne placement admin http://controller:8778
```

A continuación instalaremos los componentes nova-api, nova-conductor, nova-novncproxy, nova-scheduler y nova-placement-api. Escalamos privilegios pero sin dejar el perfil del usuario no privilegiado y usaré apt-get en vez de apt:

```
$ sudo su
# apt-get install nova-api nova-conductor nova-novncproxy nova-scheduler nova-placement-api
```

Configuramos los módulos instalados editando los archivos de configuración. Editamos /etc/nova/nova.conf. Modificamos los siguientes apartados:

```
[api]
# ...
auth_strategy = keystone

[api_database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova

[DEFAULT]
# ...
transport_url = rabbit://OpenStack:RABBIT_PASS@controller
# ...
my_ip = 10.0.0.11
# ...
use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver
# Ojo, quitar la opción log_dir de la sección [DEFAULT].

[glance]
# ...
api_servers = http://controller:9292

[keystone_authtoken]
# ...
auth_url = http://controller:5000/v3
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = NOVA_PASS

[placement_database]
# ...
```

```
connection = mysql+pymysql://placement:PLACEMENT_DBPASS@controller/placement

[oslo_concurrency]
# ...
lock_path = /var/lib/nova/tmp

[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS

[vnc]
enabled = true
# ...
server_listen = $my_ip
server_proxyclient_address = $my_ip
```

Lanzamos la orden para crear las tablas de la base de datos nova y placement, es decir completamos las bases de datos del servicio:

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

En este caso el *script* 'nova-manage api_db sync' se ejecuta como 'su' con perfil de 'root'.

Puede que en este paso se den algunos avisos por obsolescencia de algunas acciones del *script* al aplicarlo sobre OpenStack Rocky, los ignoraremos. Entramos en MariaDB server y comprobamos que se han creado las tablas en las bases de datos.

Con otro *script* registramos la base de datos cell0:

```
# su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
```

Creamos el 'cell' cell1:

```
# su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova
```

Creamos las tablas en la base de datos nova:

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

Puede que en este paso se den algunos avisos por obsolescencia de algunas acciones del *script* al aplicarlo sobre OpenStack Rocky, los ignoraremos. Verificamos estas tres últimas acciones:

```
# su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova
```

También debería funcionar la orden directa: # nova-manage cell_v2 list_cells. La respuesta es una tabla con las dos celdas creadas, cell0 y cell1.

El servicio nova-consoleauth no lo hemos instalado, en la documentación de la versión Rocky no consta que debamos hacerlo. En ciertas comprobaciones y verificaciones parece indicarse que debería

estar instalado, de momento progresaré en la instalación hasta que me sea requerido por alguna operación que necesite hacer y lo ignoraré en las validaciones.

Reseteamos los servicios:

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

Desde el navegador del portátil podemos ver en el controller las url de las API creadas (ojo a la resolución de nombres y el enrutamiento) o bien con un navegador Lynx desde la misma máquina.

Para continuar con la configuración del componente hay que instalar el nodo de computación, y cuando este esté preparado volveremos sobre este nodo. (Pasar al apartado 3.4.2).

Tras haber realizado la instalación del nodo compute1 continuamos añadiéndolo a la base de datos 'cell'

Cargamos las variables de entorno y confirmamos que hay un nodo de computación en la base de datos:

```
$ . admin-openrc
$ openstack compute service list --service nova-compute
```

Como puede verse en la figura 3-19 ya existe el nodo compute1:

```
exit
jose@controller:~$ . admin-openrc
jose@controller:~$ openstack compute service list --service nova-compute
+-----+-----+-----+-----+-----+-----+-----+
| ID | Binary          | Host      | Zone | Status | State | Updated At          |
+-----+-----+-----+-----+-----+-----+-----+
| 7  | nova-compute    | compute1  | nova | enabled | up    | 2018-10-25T08:39:58.000000 |
+-----+-----+-----+-----+-----+-----+-----+
jose@controller:~$
```

Figura 3-19. Listado de nodos de computación en el nodo de control.

Hacemos un descubrimiento de los *host* de computación:

```
# sudo su
# su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova
```

Lo que produce el resultado mostrado en la figura 3-20:

```
jose@controller:~$ sudo su
root@controller:/home/jose# su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova
Found 2 cell mappings.
Skipping cell0 since it does not contain hosts.
Getting computes from cell 'cell1': 12b8c365-fc8c-4579-88b7-fb958fe5d798
Checking host mapping for compute host 'compute1': 127aa3b0-7fal-4ad1-8c37-ceb4dac2b5f4
Creating host mapping for compute host 'compute1': 127aa3b0-7fal-4ad1-8c37-ceb4dac2b5f4
Found 1 unmapped computes in cell: 12b8c365-fc8c-4579-88b7-fb958fe5d798
root@controller:/home/jose#
```

Figura 3-20. Resultado del descubrimiento de host de computación.

Estos pasos hay que hacerlos cada vez que añadimos un nuevo nodo de computación, o programar el descubrimiento periódico en el archivo `/etc/nova/nova.conf`:

```
[scheduler]
discover_hosts_in_cells_interval = 300
```

Hacemos las verificaciones oportunas para comprobar el funcionamiento (estas también hay que hacerlas en el nodo de computación).

Listamos los servicios de Nova que están activos para ver si los procesos han sido correctamente lanzados y registrados:

```
$ . admin-openrc
$ openstack compute service list
```

La figura 3-21 muestra el resultado:

```
exit
jose@controller:~$ openstack compute service list
```

ID	Binary	Host	Zone	Status	State	Updated At
1	nova-scheduler	controller	internal	enabled	up	2018-10-25T08:56:20.000000
6	nova-conductor	controller	internal	enabled	up	2018-10-25T08:56:25.000000
7	nova-compute	computel	nova	enabled	up	2018-10-25T08:56:19.000000

```
jose@controller:~$
```

Figura 3-21. Servicios Nova activos.

La salida mostrará dos servicios en el nodo de control y uno en el nodo de computación. De acuerdo con la documentación debería aparecer un tercer servicio en el nodo de control, el servicio nova-consoleauth, pero la secuencia de instalación de la versión de OpenStack Rocky no lo implementa, ya se ha hecho una llamada de atención a este respecto anteriormente. De estar instalado este servicio, y activo debería aparecer en la salida de la orden anterior una línea parecida a esta:

```
| 1 | nova-consoleauth | controller | internal | enabled | up | 2016-02-09T23:11:15.000000 |
```

A continuación listaremos los *endpoint* en el servicio de identidad Keystone para verificar la conectividad con este servicio:

```
$ openstack catalog list
```

La salida se corresponde con la mostrada en la figura 3-22:

```
jose@controller:~$ openstack catalog list
```

Name	Type	Endpoints
keystone	identity	RegionOne
		internal: http://controller:5000/v3/
		RegionOne
		admin: http://controller:5000/v3/
		RegionOne
		public: http://controller:5000/v3/
		RegionOne
		public: http://controller:5000/v3/
placement	placement	RegionOne
		public: http://controller:8778
		RegionOne
		admin: http://controller:8778
		RegionOne
		internal: http://controller:8778
		RegionOne
		internal: http://controller:8778
nova	compute	RegionOne
		public: http://controller:8774/v2.1
		RegionOne
		internal: http://controller:8774/v2.1
		RegionOne
		internal: http://controller:8774/v2.1
		RegionOne
		admin: http://controller:8774/v2.1
glance	image	RegionOne
		internal: http://controller:9292
		RegionOne
		admin: http://controller:9292
		RegionOne
		public: http://controller:9292
		RegionOne
		public: http://controller:9292

```
jose@controller:~$
```

Figura 3-22. Catálogo de endpoints.

Los *endpoints* activos difieren con la instalación que se haga, los de la figura son los configurados en los pasos anteriores y coinciden.

Ahora vamos a listar las imágenes cargadas que nos permiten crear máquinas virtuales:

```
$ openstack image list
```

```
jose@controller:~$ openstack image list
```

ID	Name	Status
c7cc6cal-90ce-4b84-a404-23lee323ccf0	cirros	active

```
jose@controller:~$
```

Figura 3-23. Relación de imágenes cargadas y activas.

Y como se observa en la figura 3-23 está activa la imagen de cirros que cargamos durante la instalación y de la que podemos solicitar MV,s.

Por último comprobamos que las API de ‘cell’ y ‘placement’ están trabajando bien:

```
# sudo su
# nova-status upgrade check
```

```
jose@controller:~$ sudo su
[sudo] contraseña para jose:
root@controller:/home/jose# nova-status upgrade check
Deprecated: Option "enable" from group "cells" is deprecated
+-----+
| Upgrade Check Results |
+-----+
| Check: Cells v2 |
| Result: Success |
| Details: None |
+-----+
| Check: Placement API |
| Result: Success |
| Details: None |
+-----+
| Check: Resource Providers |
| Result: Success |
| Details: None |
+-----+
| Check: Ironi Flavor Migration |
| Result: Success |
| Details: None |
+-----+
| Check: API Service Version |
| Result: Success |
| Details: None |
+-----+
| Check: Request Spec Migration |
| Result: Success |
| Details: None |
+-----+
root@controller:/home/jose#
```

Figura 3-24. Estado de las API de 'cell' y 'placement'.

Siguiente paso. Consiste en instalar el servicio Neutron en el controller para lo que seguiremos la guía en la referencia [73].

Existen dos opciones de *networking*:

La opción 1, (denominada redes de proveedores) implementa el servicio de redes OpenStack de la manera más simple posible, proveyendo servicios de capa 2 (*bridging/switching*) y segmentación en VLAN,s.

Básicamente se comporta interconectando las VLAN,s con las redes físicas disponibles cuya infraestructura (la pila de red) se encarga de hacer el enrutamiento en la capa 3. Para que las instancias dispongan de direcciones IP instala un servicio DHCP para ellas.

Esta opción no puede prestar servicios avanzados de redes a los usuarios finales pues no levanta servicios virtuales de capa 3 (enrutamiento) y por tanto tampoco puede prestar servicios avanzados como *Load-Balancer-as-a-Service* (LBaaS) y *FireWall-as-a-Service* (FWaaS).

La figura 3-25 muestra un esquema de la opción 1:

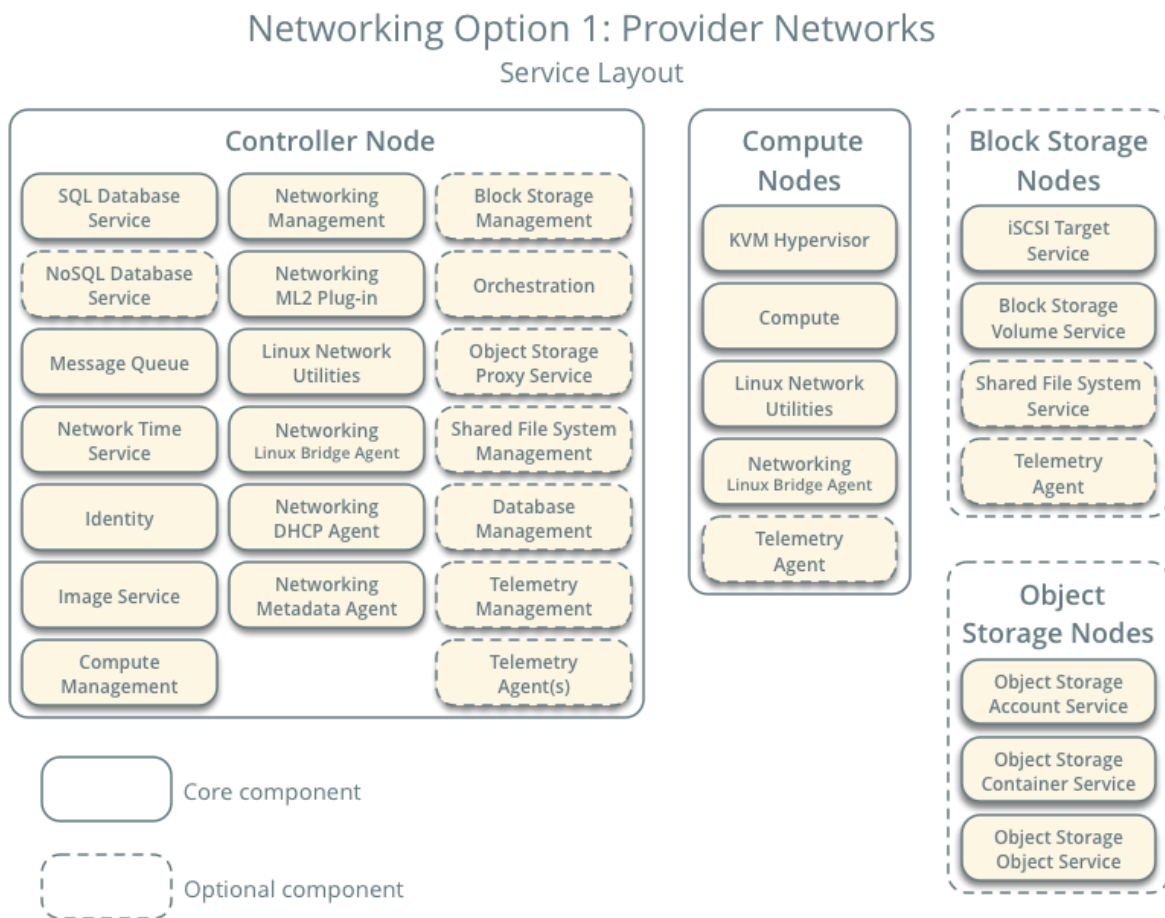


Figura 3-25. Networking opción 1. Redes de proveedor [74].

La opción 2 (que podemos denominar redes autoabastecidas o aprovisionadas), aumenta las capacidades de proveer servicios hasta la capa 3 (*routing*), que habilitan la posibilidad de crear redes LAN virtual extensible (VXLAN).

Lo que hace es que enruta las redes virtuales a las redes físicas mediante la traducción de direcciones de red (NAT). Además, esta opción proporciona la base para servicios avanzados como LBaaS y FWaaS.

Nosotros en esta experiencia trabajaremos con la opción 1, por ser la más sencilla y porque en principio solo buscamos que nuestro prototipo preste IaaS.

Lo primero es asegurarnos de que los servidores de nuestro ecosistema están debidamente configurados en cuanto a interfaces de red y resolución de nombres se refiere. Esto ya lo hemos tratado y no lo volveremos a explicar.

Comenzamos pues con la instalación propiamente dicha. Los pasos ya son conocidos.

Lo primero es crear la base de datos, el usuario y los permisos y añadir los *endpoints*. Lanzamos el cliente de MariaDB que ya tenemos instalado en el nodo controlador y comenzamos: (hay que recordar los problemas que tuvimos con la creación de la base de datos de keystone, y seguiré los mismos criterios).

```
$ sudo mysql -u root -p
MariaDB [(none)]> CREATE DATABASE neutron;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost'
IDENTIFIED BY 'NEUTRON_DBPASS';
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED  
BY 'NEUTRON_DBPASS';  
MariaDB [(none)]>exit
```

Ahora cargamos las variables de entorno del usuario 'admin'.

```
$ . admin-openrc
```

Ahora creamos el usuario neutron:

```
$ openstack user create --domain default --password-prompt neutron
```

Nos solicita la contraseña para este nuevo usuario dos veces. La contraseña que usaremos es 'NEUTRON_PASS' (cuidado de no confundir con la del usuario de base de datos 'NEUTRON_DBPASS'). La respuesta es como la ofrecida en creaciones de usuarios anteriores, como keystone o glance.

A continuación añadimos el rol de admin al proyecto de servicio y al usuario neutrón. La ejecución del comando no produce salida en la Shell.

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

A continuación creamos un servicio con nombre 'neutron'. La respuesta es como la dada en creaciones de servicios anteriores, como keystone o glance.

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

A continuación creamos los 'API endpoints' del servicio de red, mediante los comandos que se dictan. Los resultados son como en casos anteriores.

```
$ openstack endpoint create --region RegionOne network public http://controller:9696  
$ openstack endpoint create --region RegionOne network internal http://controller:9696  
$ openstack endpoint create --region RegionOne network admin http://controller:9696
```

Procedemos a instalar los componentes de neutrón. De acuerdo con nuestra elección de 'opción 1' de arquitectura de red, hay que instalar en el controller los siguientes componentes:

- neutron-server
- neutron-plugin-ml2
- neutron-linuxbridge-agent
- neutron-dhcp-agent
- neutron-metadata-agent

Escalamos privilegios pero sin dejar el perfil del usuario no privilegiado y usará apt-get en vez de apt:

```
$ sudo su  
# apt-get install neutron-server neutron-plugin-ml2 neutron-linuxbridge-agent neutron-dhcp-agent  
neutron-metadata-agent
```

Y a continuación tenemos que configurar los componentes: la base de datos, los mecanismos de autenticación, la cola de mensajes, las notificaciones, los *plugin*,...

Editamos el archivo /etc/neutron/neutron.conf y completamos:

```
[database]  
# ...  
connection = mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron
```

Todas las demás cadenas de conexión deben estar comentadas o eliminadas.

En la sección [DEFAULT] hay que configurar el plugin ML2 y deshabilitar cualquier otro; configuramos los accesos a RabbitMQ y a keystone; las notificaciones Nova.

```
[DEFAULT]
# ...
core_plugin = ml2
service_plugins =
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
# ...
auth_strategy = keystone
# ...
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true
```

En la sección [keystone_authtoken] hay que configurar el acceso al servicio de identidad:

```
[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
```

Cualquier otra entrada de esta sección debe estar comentada o eliminada.

En la sección [nova] hay que configurar las notificaciones Nova:

```
[nova]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
```

Editamos el archivo /etc/neutron/plugins/ml2/ml2_conf.ini y completamos:

Este plugin establece los mecanismos de capa 2 (*bridging and switching*) de la infraestructura de red virtual para las instancias. En la sección [ml2] del archivo configurar:

```
[ml2]
# ...
type_drivers = flat,vlan
# ...
tenant_network_types =
# ...
mechanism_drivers = linuxbridge
```

```
# ...  
extension_drivers = port_security
```

Puede ocurrir que al quitar algún valor de los establecidos por defecto en ‘type_drives’ se produzca algún problema de inconsistencia en la base de datos.

En la sección [ml2_type_flat] del archivo configurar:

```
[ml2_type_flat]  
# ...  
flat_networks = provider
```

En la sección [securitygroup] del archivo configurar:

```
[securitygroup]  
# ...  
enable_ipset = true
```

Configuramos el agente ‘Linux bridge’, que es el que construye la capa 2 de la infraestructura virtual de red y maneja los grupos de seguridad. Se edita el archivo /etc/neutron/plugins/ml2/linuxbridge_agent.ini y se completan las secciones como sigue:

```
[linux_bridge]  
physical_interface_mappings = provider:NOMBRE DE LA INTERFAZ A INTERNET  
[vxlan]  
enable_vxlan = false  
[securitygroup]  
# ...  
enable_security_group = true  
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

La interfaz del sistema con el proveedor de Internet puede coincidir con la variable \$IFACE

Hay que estar seguro de que kernel del SO sea compatible con los filtros de bridge para lo que hay que verificar que los siguientes valores de sysctl estén configurados en 1:

- net.bridge.bridge-nf-call-iptables:

```
# sysctl net.bridge.bridge-nf-call-iptables
```

- net.bridge.bridge-nf-call-ip6tables:

```
# sysctl net.bridge.bridge-nf-call-ip6tables
```

Las respuestas deben ser 1.

Para habilitar el puente de red, se debe habilitar el módulo de kernel ‘br_netfilter’. Con la siguiente orden se comprueba que está disponible:

```
# ls -R /lib/modules/$(uname -r) | grep br_netfilter
```

Con lsmod podemos listar los módulos y quien los está usando y veríamos que el módulo bridge está usando br_netfilter.

Configuramos el agente ‘DHCP’ de la red virtual. Editamos /etc/neutron/dhcp_agent.ini y completamos:

```
[DEFAULT]  
# ...  
interface_driver = linuxbridge  
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
```

```
enable_isolated_metadata = true
```

Seguidamente configuraremos el agente de metadatos, (que es el encargado de la información de configuración de los artefactos que usan la red, tal como credenciales de las instancias y otros), para lo que editamos el archivo `/etc/neutron/metadata_agent.ini` y en la sección `[DEFAULT]` establecemos:

```
[DEFAULT]
# ...
nova_metadata_host = controller
metadata_proxy_shared_secret = METADATA_SECRET
```

Evidentemente deberíamos reemplazar el secreto compartido por uno de nuestra conveniencia.

Tras esto configuramos el servicio de computación para usar el servicio de red. Editamos el archivo `/etc/nova/nova.conf` y establecemos en la sección `[neutron]` los parámetros de acceso, habilitamos el proxy de metadatos y establecemos el secreto compartido:

```
[neutron]
# ...
url = http://controller:9696
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = true
metadata_proxy_shared_secret = METADATA_SECRET
```

Por último para finalizar la instalación deberemos completar la base de datos de Neutrón (es importante no hacer esto antes de haber configurado los archivos anteriores):

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

Resetemos la API de Nova y los servicios de red:

```
# service nova-api restart
# service neutron-server restart
# service neutron-linuxbridge-agent restart
# service neutron-dhcp-agent restart
# service neutron-metadata-agent restart
```

Si hubiéramos seleccionado la opción 2 de arquitectura de los servicios de red también deberíamos reiniciar el agente L3 (de capa 3) de neutrón:

```
# service neutron-l3-agent restart.
```

Para verificar el funcionamiento correcto del servicio de red listamos los agentes de neutrón:

```
$ . admin-openrc
$ openstack network agent list
```

Los agentes se listan como se muestra en la figura 3-26:

```
jose@controller:~$ openstack network agent list
```

ID	Agent Type	Host	Availability Zone	Alive	State	Binary
473cd277-afef-4f6d-b4a4-ff131c031ab1	Linux bridge agent	computel	None	:-)	UP	neutron-linuxbridge-agent
84a5320c-4ea2-4f9b-83b2-e1e4b8d46ff7	Metadata agent	controller	None	:-)	UP	neutron-metadata-agent
b640f9ea-5486-4141-9056-91200b11671c	Linux bridge agent	controller	None	:-)	UP	neutron-linuxbridge-agent
d2e94d50-a665-4202-ae47-74a5a78a6bd0	DHCP agent	controller	nova	:-)	UP	neutron-dhcp-agent

```
jose@controller:~$
```

Figura 3-26. Prueba de la instalación de neutrón.

Debe haber tres agentes en el nodo de control y un agente en el de computación.

Siguiente paso. Consiste en instalar el *Dashboard* o portal Horizon. Seguiremos la guía de instalación de la referencia [75].

El único servicio requerido para instalar Horizon es Keystone usando Apache y Memcached para la instalación, configuración y operación.

En primer lugar hay que realizar la instalación de los paquetes:

```
$ sudo su
# apt-get install openstack-dashboard
```

Editamos el archivo `/etc/openstack-dashboard/local_settings.py` y completamos las siguientes acciones:

Configuramos Horizon para usar los servicios de OpenStack instalados en el controller:

```
OPENSTACK_HOST = "controller"
```

En la sección de configuración permitimos a los *hosts* de nuestro ecosistema el acceso. Se realiza con el parámetro `ALLOWED_HOSTS = ['one.example.com', 'two.example.com']`, por ejemplo. En nuestro caso usaremos:

```
ALLOWED_HOSTS = ['*']
```

Con esto aceptamos cualquier *host*, aunque esto es potencialmente inseguro y no debe usarse en un sistema en producción. Deberíamos incluir los nombres de los *hosts* que figuran en nuestro archivo `/etc/hosts`. En el caso de Ubuntu no se debe editar el parámetro bajo la sección de configuración de Ubuntu.

Configuramos el servicio Memcache:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': 'controller:11211',
    }
}
```

Cualquier otra entrada debe estar comentada o ser eliminada.

Habilitamos la API del servicio de identidad versión 3:

```
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
```

Habilitamos soporte para dominios:

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

Configuramos las versiones correctas de as API:

```
OPENSTACK_API_VERSIONS = {  
    "identity": 3,  
    "image": 2,  
    "volume": 2,  
}
```

Configuramos el dominio por defecto (Default) para la creación de usuarios a través del portal:

```
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
```

Configuramos el rol (role) por defecto (user) para los usuarios que creamos a través del portal:

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

Como habíamos elegido la opción 1 de arquitectura de red (al instalar Neutron), deshabilitamos el soporte para capa 3 de los servicios de red:

```
OPENSTACK_NEUTRON_NETWORK = {  
    'enable_router': False,  
    'enable_quotas': False,  
    'enable_ipv6': False,  
    'enable_distributed_router': False,  
    'enable_ha_router': False,  
    'enable_lb': False,  
    'enable_firewall': False,  
    'enable_vpn': False,  
    'enable_fip_topology_check': False,  
}
```

Configuramos la zona horaria (es opcional) con el parámetro `TIME_ZONE = "TIME_ZONE"`

ES	+4024-00341	Europe/Madrid	Spain (mainland)	Canonical	+01:00	+02:00
----	-------------	---------------	------------------	-----------	--------	--------

```
TIME_ZONE = "Europe/Madrid"
```

Configuramos el servidor Apache para que Horizon sea navegable. Editamos el fichero `/etc/apache2/conf-available/openstack-dashboard.conf`, y si no está, incluimos la línea:

```
WSGIApplicationGroup %{GLOBAL}
```

Terminamos la instalación recargando la configuración o reiniciando el servidor Apache2:

```
# service apache2 reload
```

Por último verificamos que Horizon está correctamente instalado, para lo que desde un navegador de un cliente en la red accedemos a la url del portal: `http://controller/horizon`. Las credenciales serán las de los usuarios ‘admin’ o ‘demo’ que hemos establecido cuando instalamos el controller.

El nodo controlador está totalmente instalado, y con el nodo de computación terminado nuestra nube privada es funcional. Ahora conviene realizar pruebas funcionales.

3.4.2 Nodo de computación.

En el segundo servidor virtual de nuestro clúster de OpenStack instalaremos el nodo de computación. No podemos esperar mucho de este nodo, bueno en general de este clúster, pues de acuerdo con los datos de la guía “*OpenStack Architecture Design Guide*” [76], una primera cuestión importante es que la CPU de un nodo de computación debe soportar la virtualización, y aunque en teoría VirtualBox, que es la opción gratis, soporta la virtualización sobre virtualización, es decir sus CPU,s virtuales tienen la opción de activar VT-x/AMD-V, cuando interrogamos al SO del servidor virtual nos informa que el procesador virtual no soporta. Bueno lo importante es que funcione el prototipo.

En el caso de la máquina virtual que emplearemos (los cálculos no son tan redondos) está configurada con 2 CPU,s virtuales con VT-x/AMD-V, por lo que podríamos virtualizar hasta 64 CPU,s (siendo *nested virtualization* lo dudo). También hay que pensar que cada instancia de máquina virtual no va a usar un solo procesador virtual, los ‘sabores’ acostumbrados definen 4 núcleos virtuales por instancia, por lo que nuestras 64 CPU,s virtuales se traducen en 16 MV,s.

Siendo este prototipo totalmente virtual la dependencia con el *host* que aloja los servidores hace que estos cálculos no sean fiables, y para saber el rendimiento del prototipo una vez instalado veremos cuál es la capacidad resultante para ofrecerla como IaaS, (por ejemplo determinar el número de instancias de máquina virtual de un determinado tipo que podemos lanzar).

El componente de OpenStack para computación es Nova, viene a ser un gestor de la virtualización, realmente quien administra el acceso de la máquina virtual al hardware subyacente es el hipervisor. Es importante la relación de este componente con el almacenamiento donde las instancias de MV,s se mantienen funcionando.

Para nuestro prototipo de clúster virtualizado optaremos por que el almacenamiento sea el del nodo de cálculo pero este almacenamiento no forme parte de un sistema de archivos compartido (hay otras dos opciones o posibilidades) lo cual tiene sus ventajas y desventajas, pero ahora no entraremos en ellas.

En nuestra implementación de nodo de computo virtual solo habilitaremos dos redes, una la usaremos para la gestión y almacenamiento, y otra para todo lo demás, fundamentalmente para conectarse a Internet y para poner la nube a disposición de la red corporativa.

En este nodo de computación se ejecuta un cliente NTP, para conectarse al servidor de tiempos del clúster instalado en el nodo de control y el servicio específico de computación.

En este caso vamos a instalar el nodo de computación sobre una VM en Virtual Box y de acuerdo con la documentación oficial de OpenStack de referencia [65], para una prueba de concepto bastaría con configurarla con 1 CPU, 2 Gb de RAM, 10 Gb de disco y 1 tarjetas de red. La MV la configuraré con algo más de capacidad: 2 CPU con aceleración predeterminada, 4 Gb de RAM, 50 Gb de HDD y 2 tarjetas de red.

Configuraremos las interfaces de red igual que hicimos con el nodo de control, una en modo ‘adaptador puente’ con ip estática en el rango de la red de acceso a Internet, y la otra en una red aislada interna para el clúster.

Red de Internet. Su configuración está descrita al principio de este epígrafe 3.4. Configuración del nodo de computación (compute1): La IP del nodo será 192.168.1.230; máscara /24; puerta de enlace 192.168.1.1; DNS de google 8.8.8.8 y 8.8.4.4.

Red interna. Su configuración está descrita al principio de este epígrafe 3.4. Configuración del nodo de computación (compute1): La IP del nodo computer1 será 10.8.8.31; máscara /24; no le pongo puerta de enlace en esta red ni resolución.

Para la resolución de nombres en el nodo de computación editamos el archivo /etc/hosts y declaramos las entradas (mismas que las del nodo de control):

```
# 127.0.1.1      compute1
10.8.8.11      controller
10.8.8.21      network
10.8.8.31      compute1
10.8.8.32      compute2
10.8.8.41      block1
10.8.8.51      object1
10.8.8.52      object2
```

Primeros pasos de la instalación. Comenzaremos por clonar el nodo de control y revertirlo a una configuración inicial de modo que partimos de una máquina virtual con SO Ubuntu Server 18.04 LTS con la configuración de red completada, las guest additions instaladas, y con traceroute, Chrony, repositorio Rocky y cliente python OpenStack.

La clonación tarda un rato y no debemos olvidarnos de reiniciar las MAC de las tarjetas de red. Cuando termine y antes de arrancar, reconfiguramos el hardware virtual (2 CPU con aceleración predeterminada, 4 Gb de RAM, 50 Gb de HDD y 2 tarjetas de red).

Cuando por fin accedamos a la nueva máquina hay que hacer un cambio de nombre de controller a compute1 (se edita el archivo /etc/hostname y se modifica), para que la máquina se identifique a sí misma, a continuación revisamos la configuración de red y el archivo /etc/hosts para comprobar que es correcto. Por último comprobamos la conectividad entre el nodo de control, el de computación, el router de la red, salida a Internet y el host (el portátil que estoy utilizando).

Lo siguiente es cambiar la configuración de Chrony para que actúe como cliente NTP solamente. Editamos el archivo /etc/chrony/chrony.conf y hacemos los siguientes cambios:

Comentamos o eliminamos todas las entradas de clave de servidor (*server key*) dejando solamente una que haga referencia al nodo de control:

```
server controller iburst
```

Comentamos las salidas hacia los pools:

```
# pool ntp.ubuntu.com iburst maxsources 4
# pool 0.ubuntu.pool.ntp.org iburst maxsources 1
# pool 1.ubuntu.pool.ntp.org iburst maxsources 1
# pool 2.ubuntu.pool.ntp.org iburst maxsources 2
```

Comentamos el permiso para sincronizar a los host de la red:

```
# allow 10.8.8.0/24
```

Reseteamos el servicio

```
# service chrony restart
```

Hacemos una verificación de la correcta sincronización horaria entre el nodo de control y el de computación, para lo cual en el nodo de control (controller) listamos las fuentes NTP:

```
# chronyc sources
```

Debe respondernos con el número de fuentes de las que coge la hora como se muestra en la figura 3-27.

```
jose@controller:~$ chronyc sources
210 Number of sources = 9
Name/IP address          Stratum Poll Reach LastRx Last sample
=====
* chilipepper.canonical.com 2 6 377 5 -1799us[-1978us] +/- 49ns
* golen.canonical.com      2 6 377 6 -3373us[-3550us] +/- 57ns
* alphyh.canonical.com     2 6 377 5 +9950us[+9771us] +/- 148ns
* pugot.canonical.com      2 6 377 5 -2292us[-2470us] +/- 47ns
* 213.251.52.234           2 6 277 9 -12ns[-12ns] +/- 78ns
? ntp.redinadrid.es        0 6 0 - - +0ns[ +0ns] +/- 0ns
? 2001:ac8:23::30          0 6 0 - - +0ns[ +0ns] +/- 0ns
? shackleton.red.uv.es     0 6 0 - - +0ns[ +0ns] +/- 0ns
? bjaaland.red.uv.es       0 6 0 - - +0ns[ +0ns] +/- 0ns
jose@controller:~$
```

Figura 3-27. Fuentes NTP en el nodo de control.

Hacemos lo mismo en el nodo compute1:

```
# chronyc sources
```

Ahora la única fuente debe ser el controller, como se muestra en la figura 3-28.

```
jose@compute1:~$ sudo service chrony restart
jose@compute1:~$ sudo chronyc sources
210 Number of sources = 1
=====
Source Name/IP address    Stratum Poll Reach LastRx Last sample
=====
^= controller              3      6   77    7  -17us( +98us) +/- 42ns
jose@compute1:~$ _
```

Figura 3-28. Fuentes NTP en el nodo de computación.

Siguiente paso. A continuación haremos la instalación del componente Nova en nodo compute1.

El servicio admite varios hipervisores para implementar máquinas virtuales (VM). Utilizaremos el hipervisor Quick EMUlator (QEMU) con la extensión VM basada en el kernel (KVM).

Posteriormente para añadir otros nodos de computación los prepararíamos de manera similar a esta solo que configuraríamos su red con los datos correctos.

A continuación instalaremos el módulo Nova. Escalamos privilegios pero sin dejar el perfil del usuario no privilegiado y usará apt-get en vez de apt:

```
$ sudo su
# apt-get install nova-compute
```

Configuramos el servicio de imágenes editando /etc/nova/nova.conf y hacemos los cambios pertinentes:

```
[DEFAULT]
# ...
transport_url = rabbit://OpenStack:RABBIT_PASS@controller
# ...
my_ip = 10.8.8.31
# ...
use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver
# comentar la opción log_dir de esta sección.

[api]
# ...
auth_strategy = keystone

[glance]
# ...
api_servers = http://controller:9292

[keystone_authtoken]
# ...
auth_url = http://controller:5000/v3
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = NOVA_PASS
```

```
[oslo_concurrency]
# ...
lock_path = /var/lib/nova/tmp

[placement]
#os_region_name = OpenStack
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS

[vnc]
# ...
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html
```

También habrá que configurar (más adelante) la sección [neutron] conforme a la guía de instalación del módulo Neutron, pero después de haber instalado Neutron en el nodo de control (o en el de red en su caso).

Para finalizar la instalación del componente Nova seguiremos los siguientes pasos:

Determinaremos si nuestro *hardware* soporta aceleración para las máquinas virtuales:

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

Si reporta un valor 1 o superior la aceleración está soportada, si reporta 0 no la soporta. Caso de reportar 0, (que es lo que ocurre en la máquina virtual que estoy utilizando a pesar de que está habilitada la opción en la configuración de la máquina en Virtual Box) deberemos configurar libvirt para usar QEMU en vez de KVM, lo hacemos editando el fichero `/etc/nova/nova-compute.conf`, y haciendo los siguientes cambios en la sección [libvirt]:

```
[libvirt]
# ...
virt_type = qemu
```

A continuación reseteamos el servicio Nova:

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

Si al resetear falla el inicio del servicio hay que revisar el log en `/var/log/nova/nova-compute.log`. El mensaje de error `'AMQP server on controller: 5672 is unreachable likely'` indica que el *firewall* del nodo de control (controller) está deteniendo el tráfico del puerto 5672 y hay que abrirlo. Después el servicio Nova debe iniciarse sin problemas.

Por el momento regresamos al nodo de control (controller) y continuamos en el punto 3.4.1 tras la instalación de Nova.

Ya de vuelta, creamos los *scripts* de usuario para cargar variables de entorno en este nodo como se explica en la referencia [70], y verificamos cómo funciona el sistema. Las comprobaciones son las mismas que realizamos en el nodo de control, y los resultados deben ser los mismos, excepto la última verificación, que las API de ‘cell’ y ‘placement’ están trabajando bien, pues los *scripts* de Python darán errores pues llaman a los parámetros de instalación de los *endpoints* en el nodo de control.

Tras todo esto debemos volver sobre la instalación del nodo controlador y completar la instalación de Neutron, tras lo cual y antes de instalar nada más regresaremos y continuaremos con la instalación de este nodo configurando Neutron.

Siguiente paso. Instalar los componentes de Neutron. El nodo de computación maneja la conectividad y los grupos de seguridad de las instancias que levantan los usuarios. Lo primero es instalar los componentes necesarios, que dependen de la opción de arquitectura de red seleccionada. Para la Opción 1 solo hay que instalar el agente neutron-linuxbridge-agent.

```
$ sudo su
# apt-get install neutron-linuxbridge-agent
```

A continuación lo configuramos (mecanismos de autenticación, cola de mensajes y *plugins*) editando el archivo /etc/neutron/neutron.conf

En primer lugar hay que tener en cuenta que no se usará la sección [database] pues los nodos de computación no acceden directamente a la base de datos de Neutron. Todas las cadenas de conexión deben estar comentadas.

En la sección [DEFAULT] configuraremos la cola de mensajes, y el acceso al servicio de identidad, igual que hicimos en el controller, y en la sección [keystone_authtoken] el acceso al servicio de identidad.

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
```

Cualquier otra entrada en la sección [keystone_authtoken] debe estar comentada o eliminada.

Continuamos configurando los componentes de Neutron en consonancia con la arquitectura de la red, la opción 1 en nuestro caso.

Configuramos el agente ‘linuxbridge_agent’ para lo que editamos el archivo /etc/neutron/plugins/ml2/linuxbridge_agent.ini y completamos:

```
[linux_bridge]
physical_interface_mappings = provider:NOMBRE DE LA INTERFAZ A INTERNET
```

Donde NOMBRE DE LA INTERFAZ A INTERNET es el nombre de la interfaz de conexión a la red de Internet. La interfaz del sistema con el proveedor de Internet puede coincidir con la variable \$IFACE

```
[vxlan]
enable_vxlan = false
[securitygroup]
# ...
enable_security_group = true
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Hay que estar seguro de que kernel del SO sea compatible con los filtros de *bridge* para lo que hay que verificar que los siguientes valores de sysctl estén configurados en 1:

- net.bridge.bridge-nf-call-iptables
- net.bridge.bridge-nf-call-ip6tables

Para habilitar el puente de red, se debe habilitar el módulo de kernel ‘br_netfilter’.

A continuación configuramos el servicio nova para usar el servicio neutrón. Editamos el archivo /etc/nova/nova.conf y completamos:

```
[neutron]
# ...
url = http://controller:9696
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
```

Por último reseteamos el servicio de computación y el agente linuxbridge:

```
# service nova-compute restart
# service neutron-linuxbridge-agent restart
```

Con todo el nodo de computación estará finalizado. Junto al controller tenemos una nube funcional. Ahora es el momento de instalar otros nodos complementarios, como los de almacenamiento, más de control y más de computación.

3.4.3 Referencias para la instalación de nodos de almacenamiento.

Para instalar un nodo de almacenamiento de bloques (Cinder) deberemos seguir las referencias:

- <https://docs.openstack.org/cinder/rocky/install/> [77]
 - <https://docs.openstack.org/cinder/rocky/install/get-started-block-storage.html> [78]
 - <https://docs.openstack.org/cinder/rocky/install/index-ubuntu.html> [79]
 - <https://docs.openstack.org/cinder/rocky/install/cinder-controller-install-ubuntu.html> [80]
 - <https://docs.openstack.org/cinder/rocky/install/cinder-storage-install-ubuntu.html> [81]

- <https://docs.openstack.org/cinder/rocky/install/cinder-backup-install-ubuntu.html> [82]
- <https://docs.openstack.org/cinder/rocky/install/cinder-verify.html> [83]

Para instalar un nodo de almacenamiento de bloques (Cinder) deberemos seguir las referencias:

- <https://docs.openstack.org/swift/rocky/install/> [84]
 - https://docs.openstack.org/swift/rocky/install/get_started.html [85]
 - <https://docs.openstack.org/swift/rocky/install/controller-install-ubuntu.html> [86]
 - <https://docs.openstack.org/swift/rocky/install/storage-install-ubuntu-debian.html> [87]
 - <https://docs.openstack.org/swift/rocky/install/initial-rings.html> [88]
 - <https://docs.openstack.org/swift/rocky/install/finalize-installation-ubuntu-debian.html> [89]
 - <https://docs.openstack.org/swift/rocky/install/verify.html> [90]

3.5 Ubuntu OpenStack en clúster (enfocado a producción).

3.5.1 Situación

Dado el tiempo requerido para las anteriores experiencias, para la realización de la instalación de un clúster en servidores físicos y su configuración, así como para la toma de datos y la transcripción a la memoria serían necesarias de dos a tres semanas. Al principio de este trabajo los primeros días de septiembre de 2018 se localizaron los equipos que se necesitaban para esta experiencia y se solicitó su préstamo.

Los servidores se han entregado al alumno el día 25 de octubre finalmente, pero ya fuera de plazo útil para implementar el clúster. Se recibieron 4 servidores y un *switch* gestionable. Los servidores, de la marca Supermicro, modelo 813M2 con las siguientes características:

- Procesador Intel Xenon X3430 (2,4GHz). 1 procesador de 4 núcleos (64 bits). VT-x. Sin hyperthreading.
- RAM: 8 Gb DDR3.
- IPMI BMC.
- NIC, 2 on board.
- NIC Switch LAN de 4 puertos.
- 4 HDD 500 Gb.

Por falta de tiempo y recursos físicos no se ha podido realizar esta implementación forzando las siguientes decisiones:

Decisión 4

No se realizará la implementación física, en su lugar se abordará esta experiencia desde un punto de vista teórico.

Decisión 5

No se realizarán pruebas de rendimiento, en su lugar se describirá una posible solución de cómo realizarlas.

Decisión 6

Las configuraciones de seguridad que no se puedan completar se describirán en su contexto en el anexo III.

3.5.2 La planificación de la infraestructura de nube.

Un clúster OpenStack enfocado a producción puede estar virtualizado en una o más granjas de servidores, pero sus servidores virtuales deberán estar optimizados para la función ya que habrá habido una fase de diseño que entre muchas cosas maneja los datos de demanda, de la capacidad existente y de la capacidad final que se pretende.

Los nodos de control pueden ser varios, por ejemplo tanto como las regiones configuradas, ya que cuanto mayor sea el número de nodos que compongan el sistema, más carga tendrán que soportar los nodos controladores.

No obstante lo anterior, lo normal es que destinemos *hardware* a la creación de la nube, y en eso estamos, en la creación de un clúster físico para la infraestructura *cloud*.

Pensando en la disponibilidad, (clúster *failover* y *High Ability* HA -con tolerancia a fallos y alta disponibilidad-) y que el sistema resulte eficaz, OpenStack se debería instalar en al menos 2 servidores físicos de generosas dimensiones conectados a una SAN de discos con interfaces iSCSI de 10 Gb (o más) redundantes y *hardware* de red específico dedicado con capacidades de enrutar en la capa 3.

Estos dos servidores pueden ejecutar todos los servicios involucrados, aunque algunos servicios de OpenStack están pensados para ser ejecutados en una única instancia. Aunque en la instalación sobre servidores virtuales hemos decidido un *hardware* más limitado para el ‘compute1’ que para el ‘controller’, en la realidad las exigencias sobre el *hardware* en los nodos de computación se observa que son superiores con respecto al nodo controlador.

Las capacidades mínimas para el nodo de control que apunta la documentación oficial son: 1 o 2 CPUs, 8 Gb de RAM, 100 Gb Disco y dos tarjetas de red; y para el nodo de computación: 2 o 4 CPUs, 8 + Gb de RAM, 100 + Gb Disco y dos tarjetas de red.

En el clúster físico, como mínimo habrá un nodo de control y un nodo de computación, el resto de nodos serán opcionales dependiendo de la implementación.

Una cuestión delicada es el nodo de red. Actualmente, OpenStack admite tanto el sistema de redes heredado (Nova-Network) como el nuevo sistema de redes OpenStack (Neutrón). Uno y otro tienen sus ventajas y desventajas. Ambas son opciones válidas y compatibles que se ajustan a los diferentes modelos de implementación de red descritos en las guías de operaciones de OpenStack que se pueden consultar en la referencia [91].

Cuando se utiliza el servicio de red, los nodos manejan el enrutamiento a menos que se seleccione el patrón de enrutadores virtuales dinámicos. La ejecución de enrutamiento directamente en los nodos mezcla la ‘producción’ y la ‘gestión’, lo que puede dar problemas. No confiar el enrutamiento a la propia infraestructura de OpenStack implica crear una capa para enrutamiento, de infraestructura adicional, de capa 3.

Cuando no hay un nodo específico para el enrutamiento, donde instalaríamos Neutron, este componente se instala en el nodo de control normalmente, aunque he visto algún caso en el que se dice que se instala sobre uno de computación.

Puede ser necesario conectarse a las siguientes redes:

- Red de instalación: Se utiliza para instalar el *software* base en los nodos de OpenStack. Es aconsejable una red de gigabit y normalmente no se necesita enrutamiento.

Para una instalación con vistas a producción de sistemas clasificados los repositorios de *software* deben implementarse en una ubicación local, constituirse como *mirrors* para facilitar que los nodos de OpenStack se instalen sin conectar a Internet. Los requisitos de seguridad mandan en la decisión que se adopte, también es una posibilidad instalar los sistemas operativos de base de los servidores desde *mirrors* en local o desde un DVD o similar, bastionarlo y considerándolo ya como seguro para conectarlo a Internet, hacerlo y a continuación instalar los componentes de OpenStack.

- Red de gestión (interna): se utiliza como la red de comunicación interna entre los nodos de control y computación de OpenStack. También se puede utilizar como red para el almacenamiento, es decir entre los nodos de computación y los de almacenamiento. Igual que la anterior es aconsejable una red de gigabit y normalmente no se necesita enrutamiento. Lo que si es aconsejable es contar con redundancia de esta red.
- Red de clientes (*guest*): Es como la red *host only* de VirtualBox, una red que permite la comunicación entre las instancias de cada cliente. Esta red es opcional y debe estar aislada de todas las demás redes por seguridad. Una interfaz de 1 Gb es suficiente pero en caso de implementaciones de alta disponibilidad (HA) debe estar redundada.
- Red de almacenamiento: Esta es una red dedicada entre los nodos de almacenamiento y los de computación, normalmente requiere 10 Gb y configuración redundante para HA. En el caso de una instalación seria esta red es esencial pues es muy demandante y puede ocasionar cuellos de botella e incluso fallos en los nodos si se comparte para otros propósitos que no sean el de almacenamiento.
- Red pública (externa): esta red se utilizaría para comunicarse externamente desde las máquinas virtuales con Internet (o la intranet corporativa). Estas direcciones de red son gestionadas por los agentes de Neutron en el nodo de control (incluso Neutron puede sustituirse por un dispositivo de redes diseñadas por *software* SDN). Hay implementaciones en que es necesario y no opcional que los nodos de computación tengan presencia en esta red.

Los servidores físicos para los nodos deben tener el número necesario de interfaces de red (tarjetas), y del tipo correcto en consonancia con el diseño que se haga. Como mínimo se necesitarán dos tarjetas.

Otro asunto es el almacenamiento, mejor dicho los nodos de almacenamiento. En el nodo de control se ejecuta el componente de OpenStack Glance, y este utiliza el sistema de archivos local del nodo como configuración por defecto, pero también puede utilizar nodos de almacenamiento de bloques basados en Cinder, o de objetos basados en Swift.

Los nodos de computación también necesitan almacenamiento, disco sobre el que se ejecutan las instancias de MV,s. Hay tres posibilidades:

- Utilizando un sistema de archivos compartido en el que el propio sistema de archivos del nodo de computación no participa.
En esta opción, los discos que almacenan las MV,s en ejecución están alojados en servidores fuera de los nodos de computación. Esto tiene la ventaja de que puedo detener el nodo, incluso borrarlo mientras no tenga ninguna instancia de MV,s en marcha, desde el punto de vista de la nube es un nodo 'sin estado', además si un nodo falla es más fácil recuperar sus instancias, y el mantenimiento es más fácil. Hay otras ventajas pero para mi esta es la mayor.
También tiene desventajas, como el uso intensivo de la red sobre el nodo de computo, pues la demanda sobre las instancias por parte de los usuarios recaen directamente sobre el nodo de computación; también resulta un sistema menos escalable, y otras.

- Utilizando un sistema de archivos compartido en el que el propio sistema de archivos del nodo de computación sí participa.
Cada nodo de computación aporta una capacidad de disco más o menos grande, y un sistema de archivos distribuido vincula los discos de cada nodo de computación en un conjunto de volúmenes lógicos. Esta modalidad también tiene ventajas y desventajas, por ejemplo es bueno cuando un nodo de computación requiere más recursos, pues estos se aprovisionan automáticamente a costa del almacenamiento general, pero también es más difícil satisfacer a clientes que quieren que sus MV,s estén en un sitio determinado, aumenta el uso de red, y otras desventajas.
- Utilizando el sistema de archivos del nodo de computación pero sin que forme parte de un sistema de archivo compartido.
Cada nodo de computación debe contar con el suficiente almacenamiento para las instancias que aloja. Hay ventajas grandes, el tráfico de red entre el nodo y el almacén se minimiza, es dedicado o interno del nodo, de existir, porque normalmente la comunicación se hace a nivel de la capa 2. Las demandas sobre el nodo de computación no afecta a otros nodos de computación y por ende a sus MV,s.
También hay desventajas, por ejemplo en caso de fallo el nodo de computación es un punto único, los datos asociados con las instancias que se ejecutan en ese nodo se pierden. Las migraciones de instancias de un nodo a otro son más complicadas, no es escalable en almacenamiento.

La tercera de estas opciones es la que implementamos en el clúster virtualizado, pero para el caso de utilizar servidores físicos deberemos planteárnoslo, pues para configurar las características de alta disponibilidad (HA) se necesita un almacenamiento compartido. Una de las características más deseables de un clúster con HA es la migración en vivo de las instancias virtuales levantadas en un nodo a otro nodo cuando el primero falla.

Sin tener un almacenamiento compartido nuestro clúster no operará fino las migraciones en vivo, aunque esta se puede realizar con almacenamiento no compartido en las últimas versiones de OpenStack, utilizando una función conocida como migración de bloques en vivo KVM. En todo caso usaremos un sistema de archivos distribuido NFS (por defecto para Linux) que en principio debe permitir esta funcionalidad.

La selección del almacenamiento depende de muchos factores, operativos normalmente, como requisitos de lectura/escritura, de seguridad, cargas de trabajo, perspectivas futuras, etc.

El *hardware* tiene repercusión en la selección del tipo de almacenamiento, o mejor dicho al revés, es decir hay *hardware* que maximiza la computación y otro que maximiza el almacenamiento. Los avances en la tecnología nos hacen ir cada vez más a la hiper-convergencia, *hardware* que ya es muy potente en todas las prestaciones y se utiliza para todo, bien configurado solo hay que añadir más iguales para escalar la producción, y este quizás sea el modo más rentable de construir nodos de computación de OpenStack.

Los nodos de computación en un clúster de OpenStack son los encargados de producir IaaS, pues la entrega del servicio se produce en la fase de operación, y es cuando los clientes reciben el valor que les aporta el servicio (como se dice en ITIL). Para la entrega del servicio se diseñan grupos de recursos informáticos dentro de los cuales están los servidores/nodos de computación, y lógicamente los de almacenamiento que sean.

Al diseñar estos grupos de recursos informáticos, hay que considerar la cantidad de procesadores, la cantidad de memoria, los requisitos de la red, la cantidad de almacenamiento requerido para cada hipervisor y los requisitos para los *hosts* completos que se aprovisionan para el sistema OpenStack instalado.

También hay que plantearse (planificar) si la IaaS se proporcionará en un solo grupo o en varios, en una zona o en varias, en definitiva hay que ver si es necesario crear varios perfiles de cálculo.

Una gran cantidad de nodos en la nube OpenStack aumenta mucho la carga administrativa, en términos de tráfico de red, mensajes en las colas, accesos a las base de datos en el o los nodos de control, etc. Por eso lo ideal es disponer de un cierto número de nodos de control, un cierto número de nodos de red (*networking*), un cierto número de nodos de computación, y lo mismo con el almacenamiento. En definitiva es necesario planear cuidadosamente la composición y despliegue de la nube.

Los nodos de computación pueden incorporarse dinámicamente a las nubes de OpenStack, es lo que se conoce como escalado horizontal, de manera que la capacidad del nuevo nodo se agregaría a la ya existente en la nube. Esto no es más que una expansión de la infraestructura y, como todas, requiere planificación, por ejemplo quizás sea necesario agregar más recursos de red.

Por otra parte puede aumentarse la capacidad de un nodo de computación hasta los límites del *hardware*, añadirle más memoria, más discos (unidades SAN conectada a iSCSI), más procesadores en los zócalos disponibles, o procesadores de más núcleos, es lo que se denomina escalado vertical. Después de una actualización de esta naturaleza, al instalar nuevamente el nodo en el clúster, los procesos automáticos de control de OpenStack volverán a analizarlos descubriendo los nuevos recursos y anotándolos en la base de datos del nodo de control (o en la de varios si existen).

Teniendo en cuenta unas dimensiones de computo de los nodos (al menos las nominales, nº de CPU, RAM, almacenamiento,...) es importante de cara al diseño de la nube saber calcular que recursos virtuales pueden exponerse al servicio.

Una primera cuestión importante es que la CPU de un nodo de computación debe soportar la virtualización, VirtualBox, que es la opción gratis, soporta la virtualización sobre virtualización, es decir sus CPU,s virtuales tienen la opción de activar VT-x/AMD-V¹⁷.

Otra cuestión es el número de núcleos (*cores*) del procesador, hoy en día son normales CPU,s de servidores con 24 cores que si soportan *hyperthreading* esos 24 doblan a 48 a efectos de cálculo, (realmente escalan a un rendimiento de 1,3 a 2 veces sobre las CPU,s que no lo implementan), y que con relación de sobre-dimensionamiento 16 a 1 podríamos decir que soportan hasta $2 \times 24 \times 16 = 768$ procesadores virtuales.

El sobre-dimensionamiento consiste en que OpenStack permite sobrepasar la CPU y la RAM en los nodos de computación. Esto permite aumentar el número de instancias que se ejecutan en la nube a costa de reducir el rendimiento de las instancias.

Las relaciones de sobre-dimensionamiento, son parámetros configurables en OpenStack y establecen los límites de instancias que OpenStack permitirá funcionar simultáneamente. Por ejemplo la relación entre procesadores virtualizados y físicos existentes en el nodo de computación por defecto está establecida en 16:1, es decir por cada CPU física pueden levantarse 16 virtuales, la de memoria es de 1,5:1, por cada 1 Gb de RAM física puede considerarse 1,5 Gb de RAM virtual.

Por ejemplo, si un nodo físico tiene 48 GB de RAM, el programador (*nova-scheduler*) asigna instancias a ese nodo hasta que la suma de la RAM asociada con las instancias alcanza los 72 GB (como nueve instancias, en el caso de que cada instancia tenga 8 GB de RAM).

Se debe seleccionar la proporción de asignación de CPU y RAM adecuada para cada caso de uso en particular.

¹⁷ Para que funcione es necesario habilitar el soporte para la virtualización del *host* sobre el que hemos instalado el hipervisor, de otra manera nunca los procesadores virtuales soportarían virtualización. Otra cosa que parece afectar es si se proporciona a las máquinas virtuales SO de 64 o 32 bits y la cantidad de memoria RAM que debe mantenerse por debajo de los 3 Gb. Por supuesto las opciones del procesador virtual deben estar habilitadas. Yo he intentado todo esto en varias instalaciones de Ubuntu y no he tenido éxito, de todas maneras creo que es solo cuestión de tiempo e investigación.

También tiene mucha importancia la capacidad de almacenamiento en disco, y como no la velocidad de acceso, no será lo mismo 7000 rpm que 10000, y no será lo mismo discos SSD que HDD. Una capacidad de disco insuficiente puede tener un efecto negativo en el rendimiento general, incluido el uso de la CPU y la memoria.

Para el clúster virtualizado hemos empleado un servidor virtual con 2 CPU,s virtuales sin VT-x/AMD-V, por lo que podríamos virtualizar hasta 32 CPU,s virtuales, lo cual dudo tratándose de *nested virtualization*.

También hay que pensar que cada instancia de máquina virtual no va a usar un solo procesador virtual, los ‘sabores’ acostumbrados definen 4 núcleos virtuales por instancia, por lo que nuestras 32 CPU,s virtuales se traducen en 8 MV,s.

Independientemente de la relación de sobredimensionamiento de cores y de RAM, no se puede arrancar una instancia de MV,s en ningún nodo físico con menos recursos brutos de lo que requiere la MV. En el caso del punto 3.4 donde hemos explicado como levantar un clúster en servidores virtuales la capacidad del portátil que utilizo ya es bastante limitada, así que no se podrá pedir mucho a estos nodos virtuales.

Respecto a la memoria, lo normal es que en una nube diferentes clientes dispongan diferentes recursos, unos más demandantes que otros, es decir, normalmente los accesos a memoria son dedicados y no uniformes (NUMA), es decir, no toda la memoria se reparte entre las solicitudes, sino que parte está dedicada a ciertos procesadores y en consecuencia el número de CPU,s virtuales que pueden funcionar a la vez decrece (es como si instalamos una CPU para el 50% de la memoria y el resto para el resto de CPU), y es importante planificar los recursos balanceando las CPU,s con la memoria disponible. Hay un servicio de Nova que se dedica a esto y en el caso de haber nodos NUMA el ‘NUMATopologyFilter’ debe estar activo en el Nova-Scheduler.

Nova viene a ser un gestor de la virtualización, realmente quien administra el acceso de la máquina virtual al hardware subyacente es el hipervisor. Nova, en mayor o menor grado, admite los siguientes hipervisores: Ironic, KVM, LXC (contenedores que ya usamos en la implementación del *all-in-one* sobre Ubuntu, realmente su implementación conocida como LXD), QEMU, VMware ESX/ESXi, Xen (usando libvirt), XenServer, Hyper-V, PowerVM, UML, Virtuozzo y zVM.

Deberíamos adoptar el hipervisor sobre el que más experiencia de uso tuviéramos, pero si no tenemos preferencia conviene saber que KVM es el más extendido en la comunidad de OpenStack. También es posible utilizar varios hipervisores en una nube agregando nodos o celdas, pero no se puede utilizar más de un hipervisor en un mismo nodo.

Para todas estas cuestiones relacionadas con el diseño final de una implementación hay mucha información disponible en la guía “*OpenStack Architecture Design Guide*” [76].

3.5.3 Mínima configuración del clúster.

Para construir un clúster OpenStack enfocado a una nube en producción el punto de partida es una implementación sobre 4 servidores físicos, un *switch* y un *router* con acceso a Internet.

No se usa una cabina de discos SAN por lo que no se requieren interfaces iSCSI o similares para una red específica de almacenamiento, por tanto no se pretende un sistema de almacenamiento complejo, lo que si es casi preceptivo es compartir el almacenamiento de los servidores físicos. Posteriormente podría expandirse el almacenamiento buscando implementar alta disponibilidad.

Los cuatro servidores, el *switch* y el *router* ocuparían como máximo 5U en un *rack* y una PDU, digamos que junto a los elementos auxiliares como una SAI, un teclado KVM, etc... ocuparía una cuarta parte de un *rack* con comodidad. Esta configuración es lo ideal para implementar por primera vez OpenStack.

Las especificaciones mínimas para los servidores son [<http://www.ubuntu.com/>]:

- Procesador Intel o AMD con soporte de virtualización VT-x o AMD-v (Servidores bare-metal)
- 8 GB de RAM
- Interfaz de administración de plataforma inteligente (IPMI) basada en BMC.
- Una tarjeta de red dual (dual NIC) o dos tarjetas de red

El *Baseboard Management Controller* (BMC), es un microcontrolador especializado integrado en la placa base del servidor. El BMC gestiona la interfaz entre el *software* de gestión del sistema y el *hardware* de la plataforma.

IPMI es un conjunto de especificaciones de interfaz de computadora para un subsistema de computadora autónomo que proporciona capacidades de administración y monitoreo independientemente de la CPU, el *firmware* (BIOS o UEFI) y el sistema operativo del sistema *host*.

Con esto los servidores Supermicro que se solicitaron cumplen las características necesarias y admiten una configuración de alta disponibilidad simple. Por supuesto esta infraestructura es escalable y puede llegarse a los 200 nodos. Para configuraciones de alta disponibilidad completa, con ‘*live migration*’, el número mínimo de servidores serían 12, por tanto la instalación del clúster de 4 servidores es el punto de arranque para una nube en producción. Canonical ofrece asesoría y soporte, y promete precios reducidos.

¿Cómo podríamos instalar el clúster? Hay varias posibilidades, en realidad las hemos visto y experimentado en las experiencias anteriores virtualizando servidores.

3.5.4 Instalación del clúster con MaaS Server y conjure-up

Siendo Ubuntu el SO elegido, Canonical nos ofrece una instalación, la que usamos para crear el *all-in-one* con Conjure-up, pero esta vez en vez de utilizar virtualización LXC/LXD usaríamos virtualización de KVM como *bare metal* [34].

Lo primero es enracar los equipos y conectar las NIC al *switch*, (las dos de cada servidor). Si los servidores no fueran iguales elegiríamos el más escaso de recursos para instalar el MaaS Server. Para construir el nodo de control utilizaríamos MaaS de acuerdo con la referencia [34].

MaaS es el acrónimo de metal como servicio. Es una aplicación de Canonical (de código abierto y gratuito) que permite aprovisionar en la nube servidores físicos como si fueran virtuales [92].

A parte de las ventajas administrativas están las ventajas de cara a completar infraestructuras sumando servidores físicos y virtuales como si fueran del mismo tipo. MaaS convierte el ‘metal’ en un recurso parecido a las instancias de una nube. Los servidores físicos se agrupan en un *pool* o grupo único de recursos, así las máquinas físicas pueden aprovisionarse como recurso de una infraestructura y cuando dejan de ser necesarias liberarse para otro propósito.

MaaS integra todas las herramientas necesarias para administrar:

- Interfaz web de usuario (optimizada para dispositivos móviles)
- Soporte de instalación de Ubuntu, CentOS, Windows y RHEL
- Gestión de direcciones IP de código abierto (IPAM)
- Soporte completo a través de API/CLI
- Alta disponibilidad HA (opcional)
- Soporta IPv6
- Inventario de componentes
- DHCP y DNS para los otros dispositivos de la red.
- VLAN,s.
- Servidor NTP para toda la infraestructura.

La interfaz web sirve para la mayoría de las funciones, mientras que las API de CLI y REST facilitan la configuración y la automatización a gran escala.

Íbamos diciendo que lo primero es instalar el nodo de MaaS, tratándose de MaaS podemos descargar una ISO ya instalada o podemos partir del servidor con Ubuntu 18.04 LTS e instalar MaaS como cualquier otra aplicación. Primero configuramos el *router* para que la red LAN tipo C creada con el *switch* acceda a Internet a través del *router*, y procedemos con la instalación.

```
$ sudo apt install maas
$ sudo maas init
```

Y a continuación desde un navegador (por ejemplo mediante un cliente conectado a la red) hacemos login en la dirección del ‘MaaS Server’: <http://<serverip>/MAAS/>. En el navegador se mostrará un asistente para ultimar la instalación y a continuación se deben importar las imágenes para Ubuntu. Esto tardaría un poco pero se puede continuar con la instalación del clúster mientras finaliza.

El siguiente paso es generar claves públicas auto-firmadas o importar algunas que ya se tengan para las comunicaciones por SSH entre las máquinas del clúster. MaaS facilitará que se pueda iniciar sesión con las claves SSH normales en cualquiera de las máquinas del clúster.

A partir de aquí y una vez que configuremos las subredes, MaaS se encargará de la instalación del resto de servidores del clúster. Para esto los servidores deben estar configurados para el arranque por red mediante un servicio PXE. MaaS controla las máquinas a través de IPMI u otro BMC, como son la iLO de HP, las iDRAC de Dell, las iBMC de Huawei o las UCS de Cisco, por eso se planteó como requisito disponer de esas interfaces en los servidores, lo cual no es estrictamente necesario si no se utiliza MaaS.

El siguiente paso sería configurar las subredes y el servicio DHCP. MaaS proporcionará el servicio DHCP y el servicio DNS para una red aislada tipo C (mascara /24) configurada previamente en el *switch*. Desde la interfaz web, en la pestaña ‘Subnets’ se verifica que la puerta de enlace y el servicio DNS están en la IP correcta.

Para el rango dinámico (direcciones por DHCP), hay que reservar dos direcciones IP por cada tarjeta de red del clúster, y al haber dos tarjetas por equipo serían 4 Ip,s por cada uno mas una para la interfaz IPMI, en total 20 direcciones para los 4 servidores. Podrían reservarse hasta 60 direcciones pensando en la escalabilidad, pero no debe utilizarse las 254 direcciones disponibles para DHCP, ya que necesitarán direcciones estáticas para servicios OpenStack y para uso de los usuarios de la nube.

Otra vez desde la pestaña ‘Subnets’ de la interfaz web se pulsa sobre la VLAN ‘untagged’ y se selecciona ‘Take action’ y luego ‘Provided dhcp’.

A continuación, tras haber realizado la configuración de la subred, verificaremos si las imágenes de Ubuntu que se estaban descargando han terminado de sincronizarse, se ve desde la pestaña ‘Images’ y deben aparecer todas con estado ‘Synced’. No se puede continuar el proceso sin finalizar la descarga de imágenes.

Una vez se dispone de las imágenes, para el resto de servidores del clúster (que serán los *host* de OpenStack) configuramos su arranque desde red (PXE) por defecto y deshabilitamos todas las demás opciones, y con las dos tarjetas de red conectadas al *switch*, los arrancamos. Los tres servidores aparecerán en la pestaña ‘Nodes’ de la interfaz web de MaaS al cabo de un ratito.

Si alguno no apareciera, lo primero que habría que hacer es conectarle un monitor y volver a intentar el arranque y ver como se efectúa este desde PXE por si estuviera dando algún error.

Seguidamente habría que editar cada servidor y completar los parámetros y corregirlos en su caso, aunque deberían haberse rellenado automáticamente.

A continuación se seleccionan todos los servidores y se pulsa sobre ‘Take action’ y ‘Commission’, y cuando todos estén con estado ‘Ready’ se verificaría la configuración de la red en cada servidor, desde

la pestaña de ‘*interfaces*’ por cada uno. La primera tarjeta debe ser la PXE y estar en la subred con dirección IP configurada ‘*Auto assign*’. La segunda tarjeta debe estar conectada a la subred con IP ‘*Unconfigured*’.

Hay posibilidad de desplegar un conjunto de pruebas de *hardware* en cada servidor, en el menú ‘*Take action*’ y luego ‘*Test hardware*’ con las pruebas que se quieran ejecutar. Algunas pruebas son profundas y pueden llevar mucho tiempo.

A continuación instalamos y ejecutamos conjure-up en el MaaS Server:

```
$ sudo snap install conjure-up --classic
$ conjure-up
```

Todo irá como ya relatamos al instalar el *all-in-one* sobre un servidor virtual Ubuntu por este procedimiento solo que con algunas decisiones diferentes en los menús que conjure nos ofrece.

La primera será que el hipervisor seleccionado será KVM en vez de LXD, (opción 2 *OpenStack with NovaKVM*, en vez de opción 1 *OpenStack with NovaLXD*). Seguidamente se nos solicita una entrada bajo la instrucción ‘*Configure a new cloud*’ y debemos entrar ‘*maas*’.

Después nos pedirá ‘*Add MAAS endpoint and credentials*’; el endpoint se especifica con la url de la interfaz web (<http://<<IP-Maas-Server>>/MAAS/>) y la clave es la que se encuentra en la página de la cuenta ‘*admin*’ del MAAS Server.

A continuación podrán configurarse los servicios de OpenStack para lo que en la terminal que muestra conjure-up habría que elegir la opción ‘*configure*’ de cada uno y cuando estén todos tal y como deseamos pulsaríamos el botón ‘*Deploy*’. En este paso está la clave de la instalación, es decir los componentes se desplegarán en nuestros servidores de acuerdo con la orientación de la nube que deseemos, si está más enfocada a la computación, a los recursos de red o al almacenamiento.

Existen múltiples arquitecturas de referencia [52] para despliegues multinodo de OpenStack. La arquitectura de nube que podemos implementar es multinodo simple, pues contamos con cuatro servidores, así la propuesta en este trabajo se basa en incluir un nodo de control, que incluya soporte *multi-tenant*¹⁸, un nodo de red (en el mismo servidor que el nodo de control, aunque podríamos separarlos), un nodo de computación (escalable con varios más) un nodo de almacenamiento de bloques y un nodo de almacenamiento de objetos, (estos también escalables con varios más de cada tipo).

Ya hemos dicho que para implementar una arquitectura orientada a la alta disponibilidad hay que escalar en número de servidores e implementar de entrada un segundo nodo de control, separar los nodos de red de los de control para ofrecer servicios virtualizados de red más escalables y complejos, y sumar más nodos de computación y almacenaje, buscando la redundancia de todos los elementos.

Combinar los nodos de control y de red en un único servidor físico tiene la pega de que si las máquinas virtuales o contenedores de los *tenants*, conectadas a las redes virtuales creadas y gestionadas por Neutron, incrementan el volumen de tráfico, por ejemplo el que envían o reciben desde Internet para manejar objetos muy grandes, típico en procesos de BigData, se puede llegar a la máxima capacidad de trabajo del procesador y/o la memoria, porque en el mismo servidor físico se llevaría a cabo el encaminamiento intensivo de paquetes y las funciones de filtrado de tráfico típicas del nodo de red, además de la sobrecarga que imponen los servicios de OpenStack que se ejecutan en el Nodo Controlador. Separarlos por tanto es una decisión de diseño.

¹⁸ Un *tenant* viene a ser la asociación del usuario propietario y sus instancias. Un entorno multi-tenant es un sistema en el que varios usuarios (propietarios) utilizan una misma instancia, por ejemplo un tenant sería un usuario y su base de datos, un multi-tenant es varios usuarios comparten una base de datos.

Igual que en el clúster virtualizado que instalamos en la experiencia anterior, en el nodo de control se ejecutarían el gestor de bases de datos SQL, la cola de mensajes, el servidor NTP, los servicios Keystone, Glance, Horizon, y componentes de gestión de Nova, Neutrón, Cinder y Swift.

Opcionalmente sobre este mismo nodo instalaríamos los componentes de orquestación (Heat), del servicio de telemetría (Ceilometer) y otros en su caso. Cuantos más servicios más necesidad de separar el nodo de red del de control.

En el nodo de red, si se separa del de control (decisión no tomada en nuestro caso), se instalan los *plugins* y agentes del servicio de red (Neutron) encargados de proporcionar la creación de redes, subredes y enrutadores virtuales, de realizar la conmutación, el enrutamiento, el filtrado de tráfico, la asignación DHCP a las instancias de los *tenants*, etc.

En los Nodos de Computación, se instala el hipervisor y los componentes de Nova que permiten aprovisionar las máquinas virtuales o contenedores de los *tenants*. Además en estos nodos también se instalan instancias de los plugins y agentes de red de Neutron requeridos para proporcionar conectividad de red a las máquinas virtuales y contenedores, así como para configurar grupos de seguridad para *firewall*.

Una opción es instalar en los nodos de computación agentes de Ceilometer para disponer de mediciones para facturación y estadísticas de uso de los recursos

Los nodos de almacenamiento en bloques (Cinder) y de almacenamiento de objetos (Swift) son opcionales. Los nodos de bloques contienen los discos duros sobre los cuales se crean los volúmenes que cinder proporciona a las instancias de contenedores o máquinas virtuales creadas por Nova (que por defecto utiliza el sistema de archivos como ya hemos explicado) como solución de almacenamiento permanente. Igualmente los nodos de objetos contienen los discos duros utilizados para almacenar los objetos, contenedores y cuentas de Swift.

Bueno, finalmente, si todo va bien nuestro clúster terminará bien instalado y listo para producción. Solo unas últimas consideraciones, ¿qué ha pasado? El MaaS Server actuando como servicio PXE ha arrancado los servidores, de manera integrada con una herramienta de gestión de orquestación de servicios desarrollada por Canonical llamada ‘JuJu’, para lo cual está especialmente diseñado, ha desplegado todos los servicios de OpenStack. MaaS administra las máquinas y JuJu administra los servicios que se ejecutan en esas máquinas hasta completar su instalación y configuración. Igual ocurrió cuando instalamos la versión *all-in-one* virtual con conjure-up, si se listan los contenedores se observa cómo están nominados por JuJu.

3.5.5 Instalación del clúster con DevStack

En una de las anteriores instalaciones *all-in-one* sobre Ubuntu que hicimos ya empleamos DevStack. La descripción de la instalación realizada en este punto es parecida, pero sobre varios servidores separadamente, por lo que hay que ser organizados.

Partimos de los mismos medios para la instalación física ya descritos. En los 4 servidores instalaríamos el SO Ubuntu 18.04 LTS y lo bastionaríamos. A continuación, como se describió en el punto anterior enracaríamos los equipos y conectaríamos las NIC al *switch*, (las dos de cada servidor). En el router permitimos la conexión a internet (o a una red de *mirrors*, pero a continuación solo nos referiremos a Internet) desde la LAN de servidores.

En el *switch* creamos dos redes locales (VLAN) una será la red de internet o de la intranet, a la que llamaremos externa, y otra la red aislada del clúster. Por ejemplo, red externa: 192.168.1.0/24 y red aislada: 10.8.8.0/24.

El router debe conectar por un lado en la red externa y por otro a la red de acceso de Internet o de la intranet. Si la IP del router en la red externa es 192.168.1.1 esta actuará de puerta de enlace para que los servidores puedan llegar a Internet (lógicamente el *router* debe estar configurado).

Las interfaces de los servidores se configuran de manera estática:

- Servidor 1. Destinado a nodo de control y nodo de red.
 - Interfaz 1 en la red externa: IP 192.168.1.11/24; gateway 192.168.1.1; DNS mismas del *router*.
 - Interfaz 2 en la red interna: IP 10.8.8.11/24; *gateway* sin configurar; DNS sin configurar.

En este servidor puede activarse el enrutamiento entre la red interna y la externa y configurar en el *router* una regla para que los paquetes dirigidos desde/hacia Internet se dirijan a la Ip del servidor en la red externa. Con esto todos los servidores tendrían conexión a Internet desde la red interna.

- Servidor 2. Destinado a nodo de computación.
 - Interfaz 1: Sin configurar.
 - Interfaz 2 en la red interna: IP 10.8.8.31/24; *gateway* sin configurar; DNS sin configurar.
- Servidor 3. Destinado a nodo de almacenamiento de bloques.
 - Interfaz 1: Sin configurar. Podría conectarse a una cabina de discos que soportara el sistema de archivos compartido. Es solo una previsión.
 - Interfaz 2 en la red interna: IP 10.8.8.41/24; *gateway* sin configurar; DNS sin configurar.
- Servidor 2. Destinado a nodo de almacenamiento de objetos.
 - Interfaz 1: Sin configurar.
 - Interfaz 2 en la red interna: IP 10.8.8.51/24; *gateway* sin configurar; DNS sin configurar.

Recordando lo ya dicho anteriormente, DevStack son una serie de *scripts* que se utilizan para desplegar rápida e inatendidamente un ecosistema OpenStack y también para realizar pruebas funcionales sobre el ecosistema desplegado.

Seguiremos las instrucciones del sitio oficial de OpenStack DevStack [58] para realizar la instalación que ya conocemos. Concretamente en...

<https://docs.Openstack.org/DevStack/latest/guides/multinode-lab.html>

... se dictan los pasos para instalar un clúster multinodo con DevStack.

En los servidores hay que crear un usuario 'stack' (no root), a continuación se le otorgan permisos agregándolo al grupo de sudoers (usuario stack con sudo habilitado):

```
$ sudo useradd -s /bin/bash -d /opt/stack -m stack
$ echo "stack ALL = (ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
```

... y cambiamos en la Shell a este nuevo usuario para configurar el usuario stack (en cada servidor) para acceder por SSH mediante una clave:

```
$ sudo su - stack
$ sudo mkdir ~/.ssh; chmod 700 ~/.ssh
$ sudo echo "ssh-rsa AAAAB3Nza...jose@midom.local" > ~/.ssh/authorized_keys
```

Desde este mismo usuario realizamos (en cada servidor) el clonado de DevStack desde github (o desde nuestros propios repositorios). Antes de clonar el repositorio comprobamos que no necesitamos actualizar git en nuestro sistema, y a continuación clonamos, con lo que crearemos el directorio /opt/stack/DevStack donde estará el repositorio de scripts.

```
$ sudo apt install git
```

```
$ git clone https://git.Openstack.org/OpenStack-dev/DevStack
$ cd DevStack
```

El repositorio DevStack contiene un *script* que instala OpenStack y plantillas para archivos de configuración, pero hay que crear un archivo nombrado `local.conf` que contendrá todas las opciones de instalación de cada servidor y que será diferente por cada uno.

Primero lanzaremos DevStack en el nodo de control. El archivo `local.conf` debe contener:

```
[[local | localrc]]
HOST_IP = 10.8.8.11
FLAT_INTERFACE = eth0
FIXED_RANGE = 10.4.128.0 / 20
FIXED_NETWORK_SIZE = 4096
FLOATING_RANGE = 10.8.8.128 / 25
MULTI_HOST = 1
LOGFILE = / opt / stack / logs / stack.sh.log
ADMIN_PASSWORD = secreto
DATABASE_PASSWORD = $ ADMIN_PASSWORD
RABBIT_PASSWORD = $ ADMIN_PASSWORD
SERVICE_PASSWORD = $ ADMIN_PASSWORD
```

En la configuración multinodo, las primeras 10 o más direcciones IP en la subred privada suelen estar reservadas. Hay que editar el *script* `local.sh` y agregar a siguiente línea para que se ejecute después de cada ejecución de `stack.sh`:

```
for i in `seq 2 10`; do /opt/stack/nova/bin/nova-manage fixed reserve 10.4.128.$i; done
```

Por último lanzamos el script `stack.sh`

```
$ ./stack.sh
```

Cuando termine, se mostrará un resumen de la ejecución, incluidas las URL de los *endpoints* creados, las cuentas y las contraseñas. Esta información se almacenará además en el archivo `stack.sh.log`.

A continuación hay que lanzar DevStack en los siguientes nodos, en cada caso con su archivo `local.conf` específico, tan solo hay que seguir las instrucciones en la referencia [93] y la documentación relacionada de DevStack para la instalación de cada componente de OpenStack.

3.5.6 Instalación del clúster paso a paso

La principal diferencia que existe entre la instalación paso a paso y con DevStack es que DevStack está orientado para un entorno de desarrollo y pruebas, por lo que no es recomendable realizar un despliegue final, y mucho menos lanzarlo para sumar servicios en servidores ya en producción.

DevStack no lo instala todo como servicio, por lo que ciertas aplicaciones no se inician automáticamente al iniciar el sistema.

DevStack realiza una instalación por defecto, aunque sea configurable mediante el archivo `local.conf`.

DevStack sobrepasa algunas medidas de seguridad (como la realización de copias de seguridad previas a ciertas acciones), permite editar componentes sin tener permisos de root, etc.

Por otro lado, la instalación manual, aunque es más laboriosa, permite crear un sistema totalmente a medida.

Esta instalación no difiere en nada de la que realizamos anteriormente en el clúster virtual, salvo en la instalación física de los servidores y de la red, pero que sería igual que la comentada para instalar con DevStack (que es muy similar a la del clúster virtual por no decir idéntica).

Podemos dar por terminado este epígrafe 3.5, que si bien no ha resultado de la experiencia real completada, es una aproximación teórica muy fiel. Solo resta decir que antes de comenzar la instalación es cómodo habilitar entre los servidores canales SSH con claves cruzadas e instalar un ordenador como cliente de la red para acceder por SSH a los diferentes servidores y trabajar desde él mejor que desde el KVM del *rack*.

4 RESULTADOS / VALIDACIÓN / PRUEBA

En este apartado se presentarán e interpretarán los resultados obtenidos.

En el planteamiento inicial del trabajo se estableció un alcance muy extenso, se pensó en desarrollar un conjunto de estudios, experiencias y pruebas demasiado ambicioso buscando el máximo aprovechamiento. Con menos de dos meses por delante, sin infraestructura ni apoyo económico, realmente sería casi milagroso llegar a hacer todo lo que se planteó:

- Estudio y recopilación de información y documentación.
 - Antecedentes y fundamentos.
 - Estado del Arte.
- Instalaciones de OpenStack diversas (prototipados)
 - Instalaciones *all-in-one*
 - Instalaciones multinodo
- Pruebas de funcionamiento y desempeño.
- Determinación de configuraciones de seguridad
- Escritura de la memoria.

Todo el trabajo ha llevado más de 180 horas de carga, que han supuesto casi 5 horas de media por día lectivo.

La tabla 4-1 ofrece el resumen de tiempos consumidos en cada parte del trabajo:

Tarea	Horas
Antecedentes y fundamentos.	10
Estado del Arte	50
Instalaciones all-in-one	20
Instalaciones multinodo	30
Pruebas de funcionamiento y desempeño	20
Determinación de configuraciones de seguridad	20
Escritura de una memoria de calidad	30
Total	180

Tabla 4-1. Cómputo de horas de trabajo invertidas.

En los epígrafes a continuación se enumeran los resultados de cada esfuerzo realizado.

4.1 Resultado del estudio y recopilación de información y documentación.

4.1.1 De los antecedentes y fundamentos

Se abordó el estudio de la normativa que preside la gestión, el uso y el desarrollo de las soluciones basadas en las TIC buscando el anclaje de la tecnología *cloud*. La conclusión ha sido que hay lugar a su aplicación y puede constituir uno de los ejes estratégicos CIS.

También se han estudiado los principios en los que se sustenta la seguridad aplicada a los sistemas TIC buscando poder esbozar una guía de bastionado, de la cual una aproximación puede ser el Anexo III de este trabajo.

En el capítulo 1 se exponen los resultados de este estudio que ha ocupado unas 10 horas de trabajo efectivo.

4.1.2 Del estado del Arte

Se ha estudiado la base tecnológica del *cloud computing*, sus características y modelos, la base de la virtualización, el ecosistema OpenStack y los procesos de bastionado.

Este estudio ha sido una de las fracciones del trabajo más demandantes pues la documentación oficial que la OpenStack Foundation pone a disposición de la audición es ingente, de un volumen que nos sobrepasa, unido a los foros y la wiki específica de la comunidad. Solo a la vista de las fuentes de referencias expuestas en el capítulo 6 puede hacerse una idea de la cantidad de documentación consultada y estudiada.

Una parte importante de los conocimientos adquiridos se exponen en el capítulo 2, pero no son más que la punta del hilo al tratarse de los fundamentos. Mucho más complicado ha resultado estudiar la documentación enfocada a la aplicación práctica, a la instalación, uso y *hardering* de OpenStack y a lo largo del capítulo 3 y en el resto de capítulos y anexos se explican multitud de detalles y conceptos, y se hacen muchas reflexiones hasta poder implementar prototipos de OpenStack.

El tiempo dedicado a este estudio ha sido de unas 50 horas.

4.2 Resultados de las instalaciones

4.2.1 Instalaciones *all-in-one*

Para las instalaciones de este tipo, enfocadas a sistemas para desarrollo y pruebas funcionales, los sistemas operativos no han sufrido un proceso de *hardering*, aunque en su caso para probar componentes desarrollados para entornos de producción específicos debería hacerse. Por otra parte son sistemas creados conectados a Internet, lo que no es aconsejable para sistemas críticos y no se permite en sistemas clasificados.

Por supuesto todas las modificaciones sobre los SO encaminadas a desactivar los *firewalls*, SELINUX, las opciones de montaje del sistema de archivos, a disponer contraseñas en claro en archivos de configuración ,etc., hacen inseguro el sistema, y hay que planteárselas si estuviéramos instalando un entorno de producción.

En el contexto de este trabajo, precisamente se trata de estudiar las implicaciones de las configuraciones e instalaciones realizadas en el estado de seguridad del sistema resultante, y desde ese punto de vista estas instalaciones habría que ir robusteciéndolas y haciendo pruebas funcionales sobre ellas tras aplicarles medidas de seguridad y sobre todo tras disminuir la superficie de ataque mediante la eliminación de paquetes que se instalan por defecto pero que no se van a utilizar.

Por otra parte haciendo una instalación basada en una distribución, en unos *scripts* y programas, que despliega el ecosistema OpenStack sin prácticamente tener que tomar decisiones y con opciones por defecto, podemos desplegar una nube privada de capacidades limitadas en un sistema con solo conectar a su LAN la máquina que la alberga, la cual podría ser un servidor capacidades notables. Como prueba de concepto ha permitido demostrar que la instalación de una nube privada es muy fácil en los sistemas militares.

Se han realizado las instalaciones que figuran en la tabla 4-2. Se aportan unas conclusiones relativas a cada una de ellas y el resultado obtenido.

Instalación	OpenStack PackStack en CentOS 7.5 sobre máquina virtual (VirtualBox).
Resultados	Tras la instalación se ha obtenido un sistema <i>all-in-one</i> funcional de capacidad muy limitada y bajo rendimiento por estar virtualizado sobre VirtualBox en un ordenador portátil, OpenStack funciona pero con muy poca capacidad de prestar servicio a usuarios. Es útil como sistema para pruebas y desarrollo. La nube privada generada es accesible desde la intranet a la cual se conecta el <i>host</i> y la MV. El ecosistema OpenStack instalado incluye: • Virtualización KVM. • Servicios genéricos: OpenSSH, Rabbit, MySQL y Apache. • Servicios específicos: Keystone, Neutron, Horizon, Nova, Glance, Cinder, Swift y Heat La securización se hace complicada pues PackStack requiere que el SO no le ponga trabas al despliegue y los datos necesarios los asume por defecto en su mayoría. Habría que estudiar en detalle la comandística de lanzamiento de los <i>scripts</i> y modificar los archivos de configuración de PackStack para hacer más segura la instalación y para que los servicios instalados sean los que específicamente se planeen.
Evidencia	Máquina virtual (operativa). La MV estará a disposición del tribunal de calificación del TFM en el momento de la defensa.
Instalación	OpenStack PackStack en RHEL 7.5 sobre máquina virtual (VirtualBox).
Resultados	Tras la instalación se ha obtenido un sistema <i>all-in-one</i> funcional de capacidad muy limitada y bajo rendimiento por estar virtualizado sobre VirtualBox en un ordenador portátil, OpenStack funciona pero con muy poca capacidad de prestar servicio a usuarios. Es útil como sistema para pruebas y desarrollo. La nube privada generada es accesible desde la intranet a la cual se conecta el <i>host</i> y la MV. El sistema operativo RHEL está sujeto a licencia de evaluación por 30 días y OpenStack igual pero por 60 días. El ecosistema OpenStack instalado incluye: • Virtualización KVM. • Servicios genéricos: OpenSSH, Rabbit, MySQL y Apache. • Servicios específicos: Keystone, Neutron, Horizon, Nova, Glance, Cinder, Swift y Heat La securización se hace complicada pues PackStack requiere que el SO no le ponga trabas al despliegue y los datos necesarios los asume por defecto en su mayoría. Habría que estudiar en detalle la comandística de lanzamiento de los <i>scripts</i> y modificar los archivos de configuración de PackStack para hacer más segura la instalación y para que los servicios instalados sean los que específicamente se planeen.

Evidencia	Máquina virtual (operativa). La licencias de RHEL y de OpenStack son de evaluación y concedidas a nombre del alumno tras registro en el <i>site</i> oficial de Red Hat. El ordenador portátil que se utilizó para esta instalación fue posteriormente formateado para instalar el sistema <i>all-in-one</i> sobre máquina física con Conjure-up, y por tanto es una de las máquinas virtuales desechadas.
Instalación	OpenStack en Ubuntu Server 18.04 LTS desplegado mediante Conjure-Up sobre máquina virtual (VirtualBox).
Resultados	<u>Prueba fallida</u>. Tras la instalación del SO los primeros problemas se presentan cuando se instala LXD (virtualización de sistema operativo basado en contenedores) como consecuencia de la virtualización sobre virtualización. Se toma la decisión de hacer una instalación completa sobre un ordenador portátil que cumple los requerimientos mínimos de <i>hardware</i>.
Evidencia	No procede.
Instalación	OpenStack en Ubuntu Server 18.04 LTS desplegado mediante Conjure-Up sobre máquina física.
Resultados	Tras la instalación se ha obtenido un sistema <i>all-in-one</i> funcional de capacidad limitada y buen rendimiento a pesar de los muchos servicios instalados. Es útil como sistema para pruebas y desarrollo. La nube privada generada es accesible desde la intranet a la cual se conecta el <i>host</i>. El ecosistema OpenStack instalado incluye: • Virtualización LXD (contenedores). • Servicios genéricos: NTP, OpenSSH, Rabbit, MySQL y Apache. • Servicios específicos: Ceph, Keystone, Neutron, Horizon, Nova y Glance La securización se hace muy complicada pues Conjure-Up requiere que el SO no le ponga trabas al despliegue y los datos necesarios los asume por defecto en su mayoría. Habría que estudiar en detalle la comandística de lanzamiento de los <i>scripts</i>, de Conjure-Up y JuJu, y modificar los archivos de configuración para hacer más segura la instalación y para que los servicios instalados sean los que específicamente se planeen.
Evidencia	Máquina física con la instalación. La máquina física estará a disposición del tribunal de calificación del TFM en el momento de la defensa.
Instalación	OpenStack en Ubuntu Server 18.04 LTS desplegado mediante DevStack sobre máquina virtual (Virtual Box).

Resultados	Tras la instalación se ha obtenido un sistema <i>all-in-one</i> funcional de capacidad limitada y bajo rendimiento por estar virtualizado sobre VirtualBox en un ordenador portátil, OpenStack funciona pero con muy poca capacidad de prestar servicio a usuarios. Es útil como sistema para pruebas, que se ven potenciadas con el conjunto de pruebas funcionales y de desarrollo incluidas en DevStack. La nube privada generada es accesible desde la intranet a la cual se conecta el <i>host</i> y la MV. El ecosistema OpenStack instalado incluye: • Virtualización KVM. • Servicios genéricos: OpenSSH, Rabbit, MySQL y Apache. • Servicios específicos: Keystone, Neutron, Horizon, Nova, Glance y Cinder. La securización se hace complicada pues DevStack requiere que el SO no le ponga trabas al despliegue y los datos necesarios los asume por defecto en su mayoría. Habría que estudiar en detalle la comandística de lanzamiento de los <i>scripts</i> y modificar los archivos de configuración de DevStack para hacer más segura la instalación y para que los servicios instalados sean los que específicamente se planeen.
Evidencia	Máquina virtual (operativa). La MV estará a disposición del tribunal de calificación del TFM en el momento de la defensa.

Tabla 4-2. Resumen del resultado de las instalaciones all-in-one.

Este conjunto de experiencias ha supuesto unas 20 horas de trabajo práctico (labs) solo dedicado a instalaciones de SO y componentes de OpenStack. Se han llegado a instalar sobre varios ordenadores portátiles hasta 11 MV,s y dos veces una máquina física, de las que tras pruebas de configuración no todas han sobrevivido.

4.2.2 Instalaciones multinodo

Las instalaciones multinodo están enfocadas a producción, a diferencia de las anteriores, por lo que la planificación y el diseño detallado de la infraestructura de soporte de la nube cobran una importancia capital.

Uno de los aspectos menos tratados en este trabajo es el de la configuración de las redes en la nube, la segmentación es más necesaria cuanto más extensa es nuestra infraestructura, y cuantos más componentes se instalen aisladamente, y cuantos más requisitos de alta disponibilidad y resiliencia se establezcan.

Las pruebas de *hardening* que se han podido realizar han sido precisamente sobre las máquinas virtuales del sistema clúster virtualizado, algunas de las cuales han terminado desechadas al haber quedado inoperativas tras aplicarle alguna configuración.

En general cabe hacer sobre estas instalaciones las mismas consideraciones respecto a la seguridad que ya se han hecho. La planificación y diseño que se ha aludido también debe detenerse en la seguridad del sistema. Lo primero es instalar los servidores *mirror*, en los cuales la organización debe mantener los paquetes fuente y añadir cada vez que se publiquen las actualizaciones. A continuación las instalaciones deben hacerse desde estos *mirrors*, sin conexión a Internet. El siguiente paso es bastionar los SO, desde su instalación con criterios de seguridad, y configurar los *firewalls* de *host* conforme a las políticas establecidas, por último hay que instalar los componentes uno a uno y aplicar las opciones de seguridad que se exponen en la documentación oficial de OpenStack.

Hay muchas más automatizaciones de la instalación de las que se han estudiado en este trabajo, como Mirantis con Fuel, TripleO, Ansible, etc. La instalación basada en una distribución, en unos *script* y programas, que despliegan el ecosistema OpenStack sin prácticamente tener que tomar decisiones y con opciones por defecto, no es lo deseable, pues se pierde el control sobre buena parte de las acciones como ya se explicó respecto a la instalación multinodo con DevStack.

Inicialmente se había planeado la instalación de los siguientes sistemas multinodo, siempre utilizando como SO de base Ubuntu Server 18.04 LTS:

- Clúster virtualizado desplegado con DevStack.
- Clúster virtualizado desplegado paso a paso.
- Clúster físico desplegado con MaaS Server y Conjure-up
- Clúster físico desplegado paso a paso.

Solo la instalación del ‘clúster virtualizado desplegado paso a paso’ ha llevado más de 5 días completos, (más de 25 horas y eso que no se han instalado nodos de almacenamiento) y por otra parte los servidores físicos de los que en un principio se iba a disponer no han estado a tiempo de poder realizar las instalaciones físicas, por tanto estas experiencias se han sustituido por unos ejercicios teóricos que se han expuesto en el capítulo 3.

Hay que decir que la instalación de clusteres virtualizados es también una opción para el despliegue de un sistema en producción y así se ha explicado. Por otra parte instalar sobre máquinas virtuales no es diferente de instalar sobre físicas más allá de la configuración de las BIOS de todos los elementos, y los detalles físicos, enracado, cableado, la energía, etc...

En la tabla 4-3 se resumen los resultados que se han podido obtener:

Instalación	Clúster virtualizado desplegado paso a paso.
Resultados	Se prefirió anticipar esta instalación a una basada en DevStack porque como ya se ha dicho la configuración de opciones de seguridad y el control sobre el proceso se ven favorecidos frente a una atendida por <i>scripts</i>. Tras la instalación se ha obtenido un sistema funcional de capacidad muy limitada y bajo rendimiento por estar virtualizado sobre VirtualBox en un ordenador portátil, OpenStack funciona pero con muy poca capacidad de prestar servicio a usuarios. Es útil como sistema para pruebas y desarrollo. Para mejorar este aspecto se podría haber construido el sistema sobre varios ordenadores portátiles de manera que los nodos virtualizados se repartieran en ellos. La nube privada generada es accesible desde la intranet a la cual se conecta el <i>host</i> y las MV,s que soportan los diferentes nodos. El ecosistema OpenStack instalado incluye: • Virtualización QEMU KVM. • Servicios genéricos: OpenSSH, Rabbit, MySQL y Apache. • Servicios específicos: Keystone, Neutron, Horizon, Nova y Glance (coincide con un ‘<i>starter kit</i>’ explicado en la documentación oficial).
Evidencia	Máquinas virtuales (operativas). Las MV,s estarán a disposición del tribunal de calificación del TFM en el momento de la defensa.
Instalación	Clúster virtualizado desplegado con DevStack.

Resultados	Esta instalación no se ha realizado por falta de tiempo. En realidad solo ofrecería el aprendizaje de la instalación es si misma pues para continuar con el TFM basta con disponer de las instalaciones ya realizadas. En su lugar se ha hecho un ejercicio teórico sobre como transcurriría la instalación que se expone al tratar de las instalaciones físicas, pues no difiere de una virtual y sería una experiencia más cercana a la realidad.
Evidencia	No procede.
Instalación	Clúster físico desplegado con MaaS Server y Conjure-up.
Resultados	Esta instalación no se ha realizado por falta de tiempo y de los servidores físicos. En su lugar se ha hecho un ejercicio teórico sobre como transcurriría la instalación.
Evidencia	No procede.
Instalación	Clúster físico desplegado paso a paso.
Resultados	Esta instalación no se ha realizado por falta de tiempo y de los servidores físicos. Esta instalación se realiza exactamente igual que la instalación del clúster virtualizado descrita con anterioridad
Evidencia	No procede.

Tabla 4-3. Resumen del resultado de las instalaciones multinodo.

4.3 Pruebas de funcionamiento y desempeño

Se han realizado pruebas básicas de funcionamiento (como carga y creación de imágenes, generación de usuarios y lanzamiento de instancias) desde el portal Horizon y desde la interfaz CLI, todas satisfactorias.

No se han realizado pruebas de desempeño, como era la intención inicial, por falta de tiempo, pero al menos en el presente apartado, a continuación, se aporta una descripción de cómo podrían transcurrir estas.

Una vez que el sistema está instalado y operativo es interesante conocer la relación de rendimiento del sistema virtualizado respecto al equipo en términos de sobrecarga de la CPU, la memoria RAM la red y el disco.

Lo primero es establecer las referencias. Hay que estudiar como rinde una máquina física y compararla con una instancia igual virtual lanzada por OpenStack. Hay herramientas como Phoronix Test Suite [94] que trabaja bien para este propósito sobre Ubuntu18.04 y que es *opensource*. Podemos lanzar una instancia y analizar el anfitrión mismo y la instancia.

Las pruebas a realizar y su resultado esperado pueden ser:

Test de rendimiento de la CPU: La herramienta medirá el rendimiento del procesador en el equipo anfitrión y en el instanciado. Los resultados obtenidos dependerán del número de *cores* de cada uno y de si tienen activada la opción de *hypethreading*. Es de esperar más del doble de rendimiento del procesador anfitrión respecto al virtual.

Test de rendimiento de la RAM: Consiste en un test de lectura, escritura y escalado. Es de esperar que no haya diferencias sustanciales siendo más o menos los mismos resultados pues la gestión de la memoria depende del hipervisor y son instrucciones primitivas. Las leves pérdidas de rendimiento se producen por la gestión más deficiente que el procesador virtual realiza de la memoria.

Test de rendimiento de disco: Realizamos pruebas de lectura y escritura. El resultado esperado es solo de un rendimiento ligeramente superior en el disco del anfitrión, igual que con el test de memoria.

Test de rendimiento de red: El test consiste en la transferencia de un fichero de un determinado tamaño desde la máquina mediante *loopback* (funcionalidad que ofrece la herramienta). Los resultados esperados son de rendimiento bastante inferior de la máquina instanciada respecto al anfitrión, ya que este debe realizar todas las operaciones que haría la electrónica de la tarjeta de red.

4.4 Determinación de configuraciones de seguridad

Lo primero es bastionar el sistema operativo y a continuación aplicar guías o ‘buenas prácticas’ a los productos, aplicaciones y librerías que se instalen.

En el anexo III se exponen, a modo de guía, los pasos necesarios para el *hardening*. El resultado es una aproximación al bastionado.

5 CONCLUSIONES Y LÍNEAS FUTURAS

En el capítulo 1 se establecían los objetivos de este TFM. El objetivo principal del trabajo es realizar la implementación completa de un sistema de nube privada con OpenStack y asegurarlo en lo posible sin perder de vista que es un prototipo y el tiempo para la realización de este trabajo muy limitado.

Para esto se proponía cumplir los siguientes objetivos

1. Realizar un somero estudio del origen y la historia del “Proyecto OpenStack” y una búsqueda de posibles alternativas a OpenStack.
2. Proponer un diseño de nube privada, a modo de prototipo, o el uso de un desarrollo basado en OpenStack, adaptado a las especiales características de protección de la información que requiere el ámbito del Ministerio de Defensa.
3. Probar diferentes formas de instalar o desplegar una nube privada basada en OpenStack.
4. Realizar una primera implantación básica del sistema a modo de prototipo, que funcione, para poder probar la tecnología.
5. Ofrecer una guía de implantación para una nube privada segura, o al menos un conjunto de buenas prácticas.

A continuación, tomando como línea argumental la exposición de objetivos se resumen las conclusiones del trabajo realizado.

5.1 Conclusiones sobre los estudios previos

En el capítulo 1 se hacen la exposición de motivos y objetivos del TFM.

Respecto a los objetivos, este alumno, autor del trabajo, considera que se han alcanzado y satisfecho notablemente, ahora es momento de que las valore el profesor director del trabajo y el tribunal del TFM.

Respecto a las motivaciones se llegó a la conclusión de que las opciones de computación en la nube están perfectamente prevenidas en nuestra normativa del Departamento. Desde la orientación a servicios fijada en “La Política CIS/TIC” [1], plenamente desarrollada en las líneas estratégicas mediante el “PECIS” [2] y normalizada a nivel técnico la tecnología aplicable por la Arquitectura Global (AG) [3]..., las opciones de computación en la nube son una realidad a nuestro alcance para satisfacer las necesidades del MDEF respecto a la cual tenemos la responsabilidad de estudiar sus posibilidades.

Del origen e historia del “Proyecto OpenStack”, brevemente explicado en el capítulo 2 cuando se comentaba sobre los diferentes componentes y versionado de OpenStack y de cómo estos empezaron a desarrollarse por la NASA y por Rackspace en 2010, podemos llegar a la conclusión de que OpenStack es un producto altamente fiable, tanto por los agentes implicados con la fundación Openstack.org, como por la gestión de sus versiones y los largos periodos de soporte de los productos; además el extendido uso por todo el mundo y en importantísimas organizaciones lo avalan.

Respecto a las posibles alternativas a OpenStack, estas no se han estudiado en el TFM, solo se han apuntado en el capítulo 2, así se han nombrado las soluciones de Amazon Web Services (AWS), de Microsoft Azure, de VMware Cloud, CloudStack, etc..., aludiendo en cada caso la opinión que merecen frente a OpenStack. La conclusión que se obtiene es que OpenStack es una solución de rendimiento igual o mayor que todas ellas y que se ofrece gratis, aunque el soporte resulta esencial si se quiere llegar a implementar seriamente en el MDEF.

También se han estudiado diferentes alternativas para la instalación, tanto de paquetes de despliegue como de sistemas operativos determinando que tanto las soluciones aportadas por Red Hat Inc. (estado unidense) como por Canonical Ltd. (británica) son las mejores, inclinándose este autor por la británica.

Con todo se da por satisfecho el ‘objetivo 1’.

5.2 Conclusiones sobre los diseños de nube privada

La conclusión obvia es un diseño de prototipo. Es imposible llegar a esta conclusión sin valorar previamente las obtenidas del estudio de la documentación de base y de las de las experiencias de instalación llevadas a cabo.

Por seguir ordenadamente el planteamiento de los objetivos, y centrándonos en el número 2, la conclusión es que el prototipo es ‘un sistema OpenStack multinodo’ compuesto por un nodo de control y red, y un nodo de computación, escalable horizontalmente mediante la adición de nuevos nodos de todos los tipos, para lo que se plantearía previamente la necesidad.

En cuanto a la opción de construcción del nodo es la de hacerlo en beneficio de la seguridad ‘paso a paso’ de acuerdo con la documentación oficial de Openstack.org.

Con todo se da por satisfecho el ‘objetivo 2’.

5.3 Conclusiones sobre el prototipado

Bajo este epígrafe se exponen las conclusiones de los objetivos 3 y 4, la instalación de los prototipos *all-in-one*, y multinodo y sus resultados.

Por prototipado entendemos la construcción de prototipos mediante la realización de las diversas instalaciones.

Las instalaciones realizadas son las expuestas en el capítulo 3 y cuyos resultados se presentan en el capítulo 4. Concretamente son:

- Prototipos all-in-one. Estos prototipos son adecuados para sistemas enfocados a pruebas y desarrollo, aunque pueden ser elevados a producción.
 - OpenStack PackStack sobre CentOS. La instalación resulta sencilla pero la seguridad se ve condicionada por la instalación.
 - OpenStack PackStack sobre RHEL. La instalación resulta sencilla pero la seguridad se ve condicionada por la instalación.
 - OpenStack mediante Conjure-up sobre Ubuntu.
 - Instalación sobre máquina virtual. Instalación fallida.
 - Instalación sobre máquina física. La instalación resulta sencilla pero la seguridad se ve condicionada por la instalación.
 - OpenStack DevStack sobre Ubuntu. Resulta el mejor sistema para pruebas pues DevStack incluye un buen número de ellas. La instalación resulta sencilla pero la seguridad se ve condicionada por la instalación.
- Prototipos multinodo: Los sistemas multinodo se enfocan directamente a producción.
 - Clúster virtualizado desplegado con DevStack. Se ha enfocado como un ejercicio teórico. La instalación resulta sencilla pero la seguridad se ve condicionada por la instalación.
 - Clúster virtualizado desplegado paso a paso. Se ha enfocado como ejercicio práctico y se ha instalado el prototipo. La instalación resulta compleja pero la seguridad no se ve condicionada por la instalación pudiéndose aplicar las medidas conforme se progresa en la instalación.
 - Clúster físico desplegado con MaaS Server y Conjure-up. Se ha enfocado como un ejercicio teórico. La instalación resulta medianamente compleja y la seguridad se ve condicionada por la instalación.

- Clúster físico desplegado paso a paso. Se ha enfocado como un ejercicio teórico. La instalación no difiere de la del o clúster virtualizado desplegado paso a paso, por lo que las conclusiones son extrapolables.

En este trabajo fin de Máster se han comparado métodos de implementación de una infraestructura *Cloud*. Conforme ha ido transcurriendo la ejecución del trabajo se han ido tomando decisiones para poder alcanzar los objetivos planteados.

La primera conclusión que se obtiene de las experiencias de instalación es que el desarrollo de implementaciones específicas mediante documentación genérica (aunque sea la oficial) no puede considerarse nunca como un trámite. La documentación adolece de detalles y solo se adapta a unas circunstancias determinadas que pueden no darse, como que dictan procedimientos para versiones más antiguas o menos actualizadas, aun dentro de la misma versión, de SO.

En otras ocasiones la documentación da por sentado que quien la sigue durante una instalación comprende los efectos que el SO puede tener al aplicar una determinada instrucción, por ejemplo cuando esas instrucciones se ejecutan desde el contexto de un usuario o desde el de otro.

El resultado es que la instalación no progresa como describe la documentación y los tiempos se alargan.

Por tanto crear y mantener una infraestructura de este tipo no es algo trivial y se necesita tener los conceptos muy claros.

La siguiente conclusión es que las diferentes instalaciones realizadas como prueba de concepto ha permitido demostrar que el despliegue de una nube privada es fácil en los sistemas militares, cuando estas son triviales, pero que otra cuestión es instalar o adaptar un sistema multinodo, por tanto distribuido, a una infraestructura TIC ya instalada. Habría requisitos como la integración de bases de datos del tipo LDAP o Active Directory, de uso de ciertos gestores de bases de datos, de redes y comunicaciones, de administración, y fundamentalmente de seguridad.

Las instalaciones basadas en paquetes específicos como PacStack, DevStack, TripleO, Fuel, etc... o en distribuciones como Mirantis, utilizan unos *scripts* y programas, que despliega el ecosistema OpenStack sin prácticamente tener que tomar decisiones y con opciones por defecto. Gracias a ellas podemos desplegar una nube privada de capacidades limitadas en un servidor, y con solo conectarlo a la red LAN una organización puede tener su propia nube privada.

Finalmente se ha podido realizar una instalación y configuración de OpenStack, (que es una tecnología novedosa cada vez más utilizada), multinodo con nodos de control, computación y almacenamiento, flexible y escalable que puede ofrecer servicios de infraestructura como servicio.

Aunque la instalación se ha realizado sobre servidores virtuales, (construyendo estos en una granja que les permitiera y asegurara un hardware virtual de dimensiones apreciables en términos absolutos de cantidad de *cores* de procesador, de memoria RAM y disco, podría subirse a producción), se ha realizado un ejercicio teórico de la instalación sobre una plataforma viable, de coste limitado, de cuatro servidores físicos.

Haber desplegado el prototipo sobre los servidores físicos y dejar conectada la nube privada a la WanPG hubiera sido el colofón a la ejecución de este trabajo pero no ha podido ser realizado por las siguientes razones.

- La principal, la falta de tiempo suficiente (menos de dos meses) para la realización de un TFM con un contenido tan extenso.
- Los servidores físicos que se localizaron han estado disponibles solo la última semana por lo que no habría habido tiempo material de hacer la instalación, y si se hubiera podido completar no se hubieran podido someter a pruebas diversas.

El impacto de la falta de esta experiencia en los objetivos de este TFM ha sido:

- No todas las configuraciones de seguridad han podido probarse.
- No se han podido realizar pruebas de rendimiento y ofrecer en consecuencia indicadores que sirvieran de referencia para diseños futuros.
- No obstante lo anterior, la virtualización del clúster lograda constituye un experimento muy similar a efectos de implementación. Dicha implementación ha permitido abordar el epígrafe 3.5 desde el plano de la teoría y ha permitido probar muchas de las configuraciones y cuando no, ha permitido el ejercicio teórico de ver donde es necesario aplicar alguna medida de seguridad.

Con todo el prototipo buscado se ha construido y ha permitido probar la tecnología, con lo que los objetivos números 3 y 4 se dan por satisfechos.

5.4 Conclusiones sobre la securización

La primera conclusión es que la securización afecta a todo el proceso de instalación y al mantenimiento posterior y no puede hacerse sin un diseño anterior planificado y sin una organización detrás que se involucre en la gestión de la seguridad.

En este Trabajo de Fin de Master se han puesto en jaque los conocimientos básicos del alumno sobre la tecnología, desde los SO hasta los conceptos de virtualización y computación dedicada, como los sistemas de almacenamiento distribuido y de redes, pasando por conocimientos de administración y seguridad de sistemas y de programación y *scripting*.

La realización de este trabajo ha sido una experiencia muy completa, creando, experimentando e instalando una nube privada a pesar de la gran cantidad de documentación que proporciona Openstack.org y que se ha tenido que ir seleccionando para enfocar este proyecto.

Se puede afirmar que el trabajo previo de investigación que hay que realizar es un trabajo adicional al de la instalación en sí, ya que cada nube es diferente, cada una de ellas tiene sus propias condiciones específicas, de red, de computación y de almacenamiento.

Iniciarse en OpenStack no es muy complicado una vez que se han adquirido previamente conocimientos básicos de computación en la nube.

Se puede decir que OpenStack es una tecnología válida para implementar una infraestructura que dé servicio a los sistemas militares de propósito general pues los planteamientos de seguridad pueden satisfacerse.

En el Anexo III se presentan las buenas prácticas para realizar el bastionado como base de partida, por lo que el objetivo 5 se da por satisfecho.

5.5 Posibles líneas futuras de trabajo.

Son líneas futuras que extienden el alcance de este trabajo: el estudio de nuevas propuestas de implementaciones combinando diferentes sistemas de virtualización con OpenStack, también combinando la instalación de diferentes módulos de OpenStack buscando una nube más optimizada de acuerdo a su propósito.

Una de las implementaciones realizadas ha consistido en desplegar los módulos de OpenStack sobre un sistema de virtualización ligera de sistema operativo (contenedores) como es LXD/LXC. Habría que estudiar la integración de OpenStack con Docker y Kubernetes en un trabajo posterior.

Las implementaciones realizadas no han incluido la instalación de todos los módulos de OpenStack y cada combinación de estos resulta en una instalación diferenciada por su propósito, por ejemplo el

despliegue de clústeres para procesamiento de BigData utilizando las tecnologías de Apache Hadoop con Sahara. Estos también son estudios y experiencias para posibles trabajos posteriores.

Pero como incremento futuro se propone el trabajo de terminar por instalar una infraestructura OpenStack física multinodo para ofrecer IaaS, conectada a la WanPG (la red de propósito general del MDEF), al mismo tiempo progresar en la documentación de estas experiencias para soportar la solución de problemas futuros y la operación del sistema, todo ello contando con el beneplácito y apoyo del CESTIC, de manera que pueda adquirirse conocimientos de administración, de seguridad y de mantenimiento de una infraestructura semejante. Un atractivo importante es poder ofrecer IaaS para muchos propósitos a las UCO,s a muy bajo coste.

6 BIBLIOGRAFÍA

En esta sección figuran todas las referencias, sean recursos web, libros, artículos, etc., incluyendo la información de autores, título de la obra, nombre de la publicación, año, edición y enlace más fecha de último acceso en el caso de referencias a recursos online.

[1] Ministerio de Defensa, Orden DEF/2639/2015, de 3 de diciembre, por la que se establece la Política de los Sistemas y Tecnologías de la Información y las Comunicaciones del Ministerio de Defensa. 2015.

[2] Ministerio de Defensa, Instrucción 33/2018, de 6 de junio, del Secretario de Estado de Defensa, por la que se aprueba el Plan Estratégico de los Sistemas y Tecnologías de la Información y las Comunicaciones del Ministerio de Defensa. 2018.

[3] Ministerio de Defensa, Instrucción 58/2016, de 28 de octubre, del Secretario de Estado de Defensa, por la que se aprueba la Arquitectura Global de Sistemas y Tecnologías de Información y Comunicaciones del Ministerio de Defensa. 2016.

[4] NIST Special Publication 800-145, «The NIST Definition of Cloud Computing». sep-2011.

[5] NIST Special Publication 800-144, «Guidelines on Security and Privacy in Public Cloud Computing». dic-2011.

[6] «ALMACENAMIENTO EN LA NUBE. TIPOS – Conectamelilla». [En línea]. Disponible en: <https://conectamelilla.es/almacenamiento-en-la-nube-tipos/>. [Accedido: 18-sep-2018].

[7] «Popek and Goldberg virtualization requirements», Wikipedia. 28-ago-2018.

[8] P. por: J. M. Gris, «Historia de la virtualización | Blog Virtualización & Cloud Computing», Blog Virtualización y Cloud Computing en Español, 01-mar-2012. .

[9] Miguel Rodelgo Lacruz, «Virtualización. (Apuntes de clase asignatura “Sistemas de computación”). MGSTICS. CUD. UVIGO.» 2018.

[10] «Hypervisor», Wikipedia, la enciclopedia libre. 18-sep-2018.

[11] NIST Special Publication 800-125, «Guide to Security for Full Virtualization Technologies». ene-2011.

[12] «Open source software for creating private and public clouds.», OpenStack. [En línea]. Disponible en: <https://www.openstack.org/>. [Accedido: 20-sep-2018].

[13] «What is OpenStack?», OpenStack. [En línea]. Disponible en: <https://www.openstack.org/software/>. [Accedido: 20-sep-2018].

[14] «OpenStack Releases: OpenStack Releases». [En línea]. Disponible en: <https://releases.openstack.org/index.html>. [Accedido: 20-sep-2018].

[15] «What is OpenStack? (Componentes)», OpenStack. [En línea]. Disponible en: <https://www.openstack.org/software/project-navigator/openstack-components#openstack-services>. [Accedido: 21-sep-2018].

[16] «Nova - OpenStack». [En línea]. Disponible en: <https://wiki.openstack.org/wiki/Nova>. [Accedido: 21-sep-2018].

[17] «Swift - OpenStack». [En línea]. Disponible en: <https://wiki.openstack.org/wiki/Swift>. [Accedido: 21-sep-2018].

[18] «Cinder - OpenStack». [En línea]. Disponible en: <https://wiki.openstack.org/wiki/Cinder>. [Accedido: 21-sep-2018].

- [19] «OpenStack Docs: Welcome to Neutron's documentation!» [En línea]. Disponible en: <https://docs.openstack.org/neutron/latest/>. [Accedido: 21-sep-2018].
- [20] «Keystone - OpenStack». [En línea]. Disponible en: <https://wiki.openstack.org/wiki/Keystone>. [Accedido: 21-sep-2018].
- [21] «Glance - OpenStack». [En línea]. Disponible en: <https://wiki.openstack.org/wiki/Glance>. [Accedido: 21-sep-2018].
- [22] «Heat - OpenStack». [En línea]. Disponible en: <https://wiki.openstack.org/wiki/Heat>. [Accedido: 22-sep-2018].
- [23] «Telemetry - OpenStack». [En línea]. Disponible en: <https://wiki.openstack.org/wiki/Telemetry>. [Accedido: 22-sep-2018].
- [24] «Trove - OpenStack». [En línea]. Disponible en: <https://wiki.openstack.org/wiki/Trove>. [Accedido: 22-sep-2018].
- [25] «Sahara - OpenStack». [En línea]. Disponible en: <https://wiki.openstack.org/wiki/Sahara>. [Accedido: 22-sep-2018].
- [26] «Magnum - OpenStack». [En línea]. Disponible en: <https://wiki.openstack.org/wiki/Magnum>. [Accedido: 22-sep-2018].
- [27] K.-A. OpenStack-Charms y O.-H. T. Bifrost, «DEPLOYMENT / LIFECYCLE TOOLS», p. 1.
- [28] «Red Hat OpenStack Platform». [En línea]. Disponible en: <https://www.redhat.com/en/technologies/linux-platforms/openstack-platform>. [Accedido: 23-sep-2018].
- [29] «Herramientas de despliegue: TripleOOpenCloud.es | OpenCloud.es». .
- [30] «OpenStack on Ubuntu is your scalable private cloud, by Canonical | Ubuntu». [En línea]. Disponible en: <https://www.ubuntu.com/openstack>. [Accedido: 17-sep-2018].
- [31] «Server technologies Web Usage Distribution». [En línea]. Disponible en: <https://trends.builtwith.com/server>. [Accedido: 24-sep-2018].
- [32] «MAAS | Metal as a Service». [En línea]. Disponible en: https://maas.io/?_ga=2.148023920.612395669.1537805186-38883017.1537805186. [Accedido: 24-sep-2018].
- [33] «conjure-up.io». [En línea]. Disponible en: https://conjure-up.io/?_ga=2.244515906.612395669.1537805186-38883017.1537805186. [Accedido: 24-sep-2018].
- [34] «Install | OpenStack | Ubuntu | Ubuntu». [En línea]. Disponible en: <https://www.ubuntu.com/openstack/install>. [Accedido: 24-sep-2018].
- [35] «OpenStack : Boost Developer Productivity with Open APIs», VMWare. [En línea]. Disponible en: <https://www.vmware.com/products/openstack.html>. [Accedido: 23-sep-2018].
- [36] «Guía de instalación y configuración de VMware Integrated OpenStack - VMware Integrated OpenStack 2.0», p. 60.
- [37] «OpenStack | Software | Mirantis», Mirantis | Pure Play Open Cloud. [En línea]. Disponible en: <https://www.mirantis.com/software/openstack/>. [Accedido: 24-sep-2018].
- [38] D. R. Sánchez, «INSTALANDO MIRANTIS OPENSTACK», p. 29.

- [39] «Commercial Distributions and Hardware Appliances of OpenStack Private Cloud», OpenStack. [En línea]. Disponible en: <https://www.openstack.org/marketplace/distros/>. [Accedido: 24-sep-2018].
- [40] Ministerio de la Presidencia. Secretaría General Técnica-Secretariado del Gobierno-Centro de Publicaciones, «Normas de la Autoridad Nacional para la Protección de la Información Clasificada». .
- [41] Centro Criptológico Nacional (CCN), «CCN-STIC-101. ACREDITACIÓN DE SISTEMAS DE LAS TIC QUE MANEJAN INFORMACIÓN CLASIFICADA.» jun-2016.
- [42] Centro Criptológico Nacional (CCN), «CCN-STIC-205. ACTIVIDADES DE SEGURIDAD EN EL CICLO DE VIDA DE LOS SISTEMAS TIC .» dic-2007.
- [43] Centro Criptológico Nacional (CCN), «CCN-STIC-301. REQUISITOS STIC .» nov-2016.
- [44] Centro Criptológico Nacional (CCN), «CCN-STIC-612. SECURIZACIÓN DE SISTEMAS BASADOS EN DEBIAN .» ago-2014.
- [45] Centro Criptológico Nacional (CCN), «CCN-STIC-823. UTILIZACIÓN DE SERVICIOS EN LA NUBE.» dic-2014.
- [46] «INCIBE-CERT», INCIBE-CERT. [En línea]. Disponible en: <https://www.incibe-cert.es/>. [Accedido: 02-oct-2018].
- [47] «Guías y Estudios», INCIBE-CERT, 25-ene-2016. [En línea]. Disponible en: <https://www.incibe-cert.es/guias-y-estudios>. [Accedido: 02-oct-2018].
- [48] «PAe - Seguridad». [En línea]. Disponible en: https://administracionelectronica.gob.es/pae_Home/pae_Estrategias/pae_Seguridad_Inicio.html#.W7OTxLjta03. [Accedido: 02-oct-2018].
- [49] C. C. Editor, «Home | CSRC». [En línea]. Disponible en: <https://csrc.nist.gov/>. [Accedido: 02-oct-2018].
- [50] T. A. Allen, «NIST Cloud Computing Program - NCCP», NIST, 15-nov-2010. [En línea]. Disponible en: <https://www.nist.gov/programs-projects/nist-cloud-computing-program-nccp>. [Accedido: 02-oct-2018].
- [51] «Oráculo Servicio Técnico», Oráculo Servicio Técnico. [En línea]. Disponible en: <https://oraculoserviciotecnico.wordpress.com/>. [Accedido: 02-oct-2018].
- [52] «What is OpenStack? sample-configs#compute-starter-kit», OpenStack. [En línea]. Disponible en: <https://www.openstack.org/software/sample-configs#compute-starter-kit>. [Accedido: 26-sep-2018].
- [53] «What is OpenStack? sample-configs#video-processing-and-content-delivery», OpenStack. [En línea]. Disponible en: <https://www.openstack.org/software/sample-configs#video-processing-and-content-delivery>. [Accedido: 26-sep-2018].
- [54] «OpenStack Docs: Design». [En línea]. Disponible en: <https://docs.openstack.org/arch-design/design.html>. [Accedido: 17-oct-2018].
- [55] «Download CentOS». [En línea]. Disponible en: <https://www.centos.org/download/>. [Accedido: 12-oct-2018].
- [56] «7.4. Common Networking Configuration». [En línea]. Disponible en: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/5/html/Installation_and_Configuration_Guide/sect-Common_Networking_Configuration.html#. [Accedido: 12-oct-2018].

- [57] «QA - OpenStack». [En línea]. Disponible en: <https://wiki.openstack.org/wiki/QA>. [Accedido: 14-oct-2018].
- [58] «OpenStack Docs: DevStack». [En línea]. Disponible en: <https://docs.openstack.org/devstack/latest/>. [Accedido: 16-oct-2018].
- [59] «OpenStack resources for app developers including OpenStack SDKs», OpenStack. [En línea]. Disponible en: <https://www.openstack.org/appdev/>. [Accedido: 31-oct-2018].
- [60] «openstack-dev/devstack - System for quickly installing an OpenStack cloud from upstream git for testing and development.» [En línea]. Disponible en: <https://git.openstack.org/cgit/openstack-dev/devstack/>. [Accedido: 31-oct-2018].
- [61] System for quickly installing an OpenStack cloud from upstream git for testing and development.: openstack-dev/devstack. openstack-dev mirrors, 2018.
- [62] Constantine, Automatic installation of Devstack/OpenStack on virtual Ubuntu-Server-14.04.3: makelinux/devstack-install-on-iso. 2018.
- [63] «devstack in Launchpad». [En línea]. Disponible en: <https://launchpad.net/devstack>. [Accedido: 31-oct-2018].
- [64] «Snapcraft - Snaps are universal Linux packages», Snapcraft. [En línea]. Disponible en: <https://snapcraft.io/>. [Accedido: 14-oct-2018].
- [65] «OpenStack Docs: Environment». [En línea]. Disponible en: <https://docs.openstack.org/install-guide/environment.html>. [Accedido: 17-oct-2018].
- [66] «OpenStack Docs: SQL database for Ubuntu». [En línea]. Disponible en: <https://docs.openstack.org/install-guide/environment-sql-database-ubuntu.html>. [Accedido: 18-oct-2018].
- [67] «OpenStack Docs: Security». [En línea]. Disponible en: <https://docs.openstack.org/install-guide/environment-security.html>. [Accedido: 18-oct-2018].
- [68] «OpenStack Docs: Rocky Installation Guides». [En línea]. Disponible en: <https://docs.openstack.org/rocky/install/>. [Accedido: 18-oct-2018].
- [69] «OpenStack Docs: Keystone Installation Tutorial». [En línea]. Disponible en: <https://docs.openstack.org/keystone/rocky/install/>. [Accedido: 18-oct-2018].
- [70] «OpenStack Docs: Create OpenStack client environment scripts». [En línea]. Disponible en: <https://docs.openstack.org/keystone/latest/install/keystone-openrc-ubuntu.html>. [Accedido: 23-oct-2018].
- [71] «OpenStack Docs: Glance Installation». [En línea]. Disponible en: <https://docs.openstack.org/glance/rocky/install/>. [Accedido: 21-oct-2018].
- [72] «OpenStack Docs: Install and configure controller node for Ubuntu». [En línea]. Disponible en: <https://docs.openstack.org/nova/rocky/install/controller-install-ubuntu.html>. [Accedido: 23-oct-2018].
- [73] «OpenStack Docs: Networking service». [En línea]. Disponible en: <https://docs.openstack.org/neutron/rocky/install/>. [Accedido: 29-oct-2018].
- [74] «OpenStack Docs: Overview». [En línea]. Disponible en: <https://docs.openstack.org/neutron/rocky/install/overview.html>. [Accedido: 29-oct-2018].
- [75] «OpenStack Docs: Installation Guide». [En línea]. Disponible en: <https://docs.openstack.org/horizon/rocky/install/>. [Accedido: 30-oct-2018].

- [76] «OpenStack Docs: OpenStack Architecture Design Guide». [En línea]. Disponible en: <https://docs.openstack.org/arch-design/index.html>. [Accedido: 24-oct-2018].
- [77] «OpenStack Docs: Cinder Installation Guide». [En línea]. Disponible en: <https://docs.openstack.org/cinder/rocky/install/>. [Accedido: 30-oct-2018].
- [78] «OpenStack Docs: Cinder Block Storage service overview». [En línea]. Disponible en: <https://docs.openstack.org/cinder/rocky/install/get-started-block-storage.html>. [Accedido: 30-oct-2018].
- [79] «OpenStack Docs: Cinder Installation Guide for Ubuntu». [En línea]. Disponible en: <https://docs.openstack.org/cinder/rocky/install/index-ubuntu.html>. [Accedido: 30-oct-2018].
- [80] «OpenStack Docs: Install and configure controller node». [En línea]. Disponible en: <https://docs.openstack.org/cinder/rocky/install/cinder-controller-install-ubuntu.html>. [Accedido: 30-oct-2018].
- [81] «OpenStack Docs: Install and configure a storage node». [En línea]. Disponible en: <https://docs.openstack.org/cinder/rocky/install/cinder-storage-install-ubuntu.html>. [Accedido: 30-oct-2018].
- [82] «OpenStack Docs: Install and configure the backup service». [En línea]. Disponible en: <https://docs.openstack.org/cinder/rocky/install/cinder-backup-install-ubuntu.html>. [Accedido: 30-oct-2018].
- [83] «OpenStack Docs: Verify Cinder operation». [En línea]. Disponible en: <https://docs.openstack.org/cinder/rocky/install/cinder-verify.html>. [Accedido: 30-oct-2018].
- [84] «OpenStack Docs: Object Storage Install Guide». [En línea]. Disponible en: <https://docs.openstack.org/swift/rocky/install/>. [Accedido: 30-oct-2018].
- [85] «OpenStack Docs: Object Storage service overview». [En línea]. Disponible en: https://docs.openstack.org/swift/rocky/install/get_started.html. [Accedido: 30-oct-2018].
- [86] «OpenStack Docs: Install and configure the controller node for Ubuntu». [En línea]. Disponible en: <https://docs.openstack.org/swift/rocky/install/controller-install-ubuntu.html>. [Accedido: 30-oct-2018].
- [87] «OpenStack Docs: Install and configure the storage nodes for Ubuntu and Debian». [En línea]. Disponible en: <https://docs.openstack.org/swift/rocky/install/storage-install-ubuntu-debian.html>. [Accedido: 30-oct-2018].
- [88] «OpenStack Docs: Create and distribute initial rings». [En línea]. Disponible en: <https://docs.openstack.org/swift/rocky/install/initial-rings.html>. [Accedido: 30-oct-2018].
- [89] «OpenStack Docs: Finalize installation for Ubuntu and Debian». [En línea]. Disponible en: <https://docs.openstack.org/swift/rocky/install/finalize-installation-ubuntu-debian.html>. [Accedido: 30-oct-2018].
- [90] «OpenStack Docs: Verify operation». [En línea]. Disponible en: <https://docs.openstack.org/swift/rocky/install/verify.html>. [Accedido: 30-oct-2018].
- [91] «OpenStack Docs: Rocky Administrator Guides». [En línea]. Disponible en: <https://docs.openstack.org/rocky/admin/#network-topology>. [Accedido: 24-oct-2018].
- [92] «What is MAAS? | MAAS documentation». [En línea]. Disponible en: <https://docs.maas.io/2.4/en/>. [Accedido: 29-oct-2018].
- [93] «OpenStack Docs: Multi-Node Lab». [En línea]. Disponible en: <https://docs.openstack.org/devstack/latest/guides/multinode-lab.html>. [Accedido: 29-oct-2018].

- [94] «Phoronix Test Suite - Linux Testing & Benchmarking Platform, Automated Testing, Open-Source Benchmarking». [En línea]. Disponible en: <https://www.phoronix-test-suite.com/>. [Accedido: 29-oct-2018].
- [95] «Curso OpenStack». [En línea]. Disponible en: http://iesgn.github.io/emergya/curso/u2/admin_basica_horizon. [Accedido: 29-oct-2018].
- [96] «OpenStack Docs: OpenStack Security Guide». [En línea]. Disponible en: <https://docs.openstack.org/security-guide/>. [Accedido: 10-oct-2018].
- [97] «The leading operating system for PCs, IoT devices, servers and the cloud | Ubuntu». [En línea]. Disponible en: <https://www.ubuntu.com/#download>. [Accedido: 10-oct-2018].
- [98] «Index of /releases/18.04/release». [En línea]. Disponible en: <http://cdimage.ubuntu.com/releases/18.04/release/>. [Accedido: 10-oct-2018].
- [99] «Rufus - Cree unidades USB arrancables fácilmente». [En línea]. Disponible en: <https://rufus.akeo.ie/>. [Accedido: 10-oct-2018].
- [100] «netfilter/iptables project homepage - The netfilter.org project». [En línea]. Disponible en: <https://netfilter.org/>. [Accedido: 07-nov-2018].
- [101] Centro Criptológico Nacional (CCN), «CCN-STIC-419. CONFIGURACIÓN SEGURA CON IPTABLES.» dic-2007.
- [102] «ufw in Launchpad». [En línea]. Disponible en: <https://launchpad.net/ufw>. [Accedido: 11-nov-2018].
- [103] «UncomplicatedCortafuegos - Ubuntu Wiki». [En línea]. Disponible en: <https://wiki.ubuntu.com/UncomplicatedCortafuegos>. [Accedido: 11-nov-2018].
- [104] «OpenStack Docs: Firewalls and default ports». [En línea]. Disponible en: <https://docs.openstack.org/install-guide/firewalls-default-ports.html>. [Accedido: 11-nov-2018].
- [105] «OpenStack Docs: Compute service node firewall requirements». [En línea]. Disponible en: <https://docs.openstack.org/nova/rocky/admin/default-ports.html>. [Accedido: 11-nov-2018].
- [106] «Amanda Open Source Backup». [En línea]. Disponible en: <http://amanda.zmanda.com/>. [Accedido: 12-nov-2018].

ANEXO I: ACRÓNIMOS UTILIZADOS

AG. Arquitectura Global (Arquitectura Global de Sistemas y Tecnologías de Información y Comunicaciones del Ministerio de Defensa).

AGE. Administración General del Estado (de España).

ANPIC. Autoridad Nacional Para la Protección de la Información Clasificada (en España).

API. Acrónimo en inglés. *Application Programming Interface*. Interfaz de programación de aplicaciones.

APP. Acrónimo en inglés. Es un acortamiento de la palabra inglesa application.

C3. Acrónimo en inglés. *Consultation, Command and Control*. Consulta, Mando y Control.

CCN. Centro Criptológico Nacional.

CERT. Acrónimo en inglés. *Computer Emergency Response Team*. Equipo de Respuesta ante Emergencias Informáticas o Incidentes.

CESTIC. Centro de Sistemas y Tecnologías de la Información y las Comunicaciones.

CIS. Acrónimo en inglés. *Communications and Information Systems*. Sistemas de Información y comunicaciones.

CNI. Centro Nacional de Inteligencia.

CNPIC. Centro Nacional de Protección de Infraestructuras y Ciberseguridad

COTS. Acrónimo en inglés. *Commercial Off-The-Shelf*. Producto comercial ya desarrollado.

CPD. Centro de Proceso de Datos

CUE. Catálogo Unificado de Estándares CIS/TIC del Ministerio de Defensa.

DBaaS. Acrónimo en inglés. *Data Base as a Service*. Base de datos como servicio.

DC. Acrónimo en inglés. *Domain controller*. Controlador de dominio.

DNS. Acrónimo en inglés. *Domain Name System*. Sistema de resolución de nombres de dominio.

EC2. Referido a compatibilidad de una API

EE. Eje Estratégico, referido a la Política CIS/TIC y al PECIS.

ENS. Esquema Nacional de Seguridad

ESA. Acrónimo en inglés. *European Space Agency*. Agencia Espacial Europea.

FAS. Fuerzas Armadas de España.

FDCC. Acrónimo en inglés. *Federal Desktop Core Configuration*

FaaS. Acrónimo en inglés. *Firewall as a Service*. Cortafuegos como servicio.

I3D. Infraestructura Integral de Información para la Defensa.

IDS. Acrónimo en inglés. *Intrusion Detection System*. Sistema de detección de intrusiones.

INCIBE. Instituto Nacional de Ciberseguridad.

IPS. Acrónimo en inglés. *Intrusion Prevention System*. Sistema de prevención de intrusiones.

ISO. Acrónimo en inglés. *International Organization for Standardization*. Organización Internacional de Normalización.

GUI. Acrónimo en inglés. *Graphical User Interface*. Interfaz gráfica de usuario.

HA. Acrónimo en inglés. *High Availability*. Alta disponibilidad.

IaaS. Acrónimo en inglés. *Infrastructure as a Service*. Infraestructura como servicio.

IP. Acrónimo en inglés. *Internet Protocol*. Protocolo de Internet.

ISP. Acrónimo en inglés. *Internet service provider*. Proveedor de servicios de internet.

KVM. Acrónimo en inglés. *Kernel-based Virtual Machine*. Máquina virtual basada en el núcleo, referido a sistemas Linux.

LA. Línea de Acción, referido a la Política CIS/TIC y al PECIS.

LDAP. Acrónimo en inglés. *Lightweight Directory Access Protocol*. Protocolo Ligero/Simplificado de Acceso a Directorios.

LTS. Acrónimo en Inglés. *Long Time Support*. Soporte de larga duración. Referido a versiones de Ubuntu.

LXC. Acrónimo en Inglés. *Linux Containers*. Contenedores basados en Linux.

LXD. Acrónimo en inglés. *Linux Container Daemon*. Demonio para contenedores en Linux.

MaaS. Acrónimo en Inglés. *Metal as a Service*. *Hardware* virtual (o plataforma) como servicio.

MCCD. Mando de Ciberdefensa.

MDEF. Ministerio de Defensa de España.

MV. Máquina virtual.

NAS. Acrónimo en Inglés. *Network Attached Storage*. Almacenamiento conectado en red.

NASA. Acrónimo en inglés. *National Aeronautics and Space Administration*.

NIST. Acrónimo en inglés. *National Institute of Standards and Technology*.

NAF. Acrónimo en inglés. *NATO Architecture Framework*.

NATO. Acrónimo en inglés. Ver OTAN.

OE. Objetivo Estratégico, referido a la Política CIS/TIC y al PECIS.

OTAN. Organización del Tratado del Atlántico Norte. (NATO. Acrónimo en inglés).

PaaS. Acrónimo en inglés. *Platform as a Service*. Plataforma como servicio.

PC. Acrónimo en inglés. *Personal computer*. Ordenador personal.

PECIS. Plan Estratégico de los Sistemas y Tecnologías de la Información y las Comunicaciones.

PXE. Acrónimo en inglés. *Preboot eXecution Environment*. Entorno de ejecución de pre-arranque.

RD. Real Decreto.

REST. Acrónimo en inglés. *Representational state transfer*. Transferencia de estado representacional.

RHEL. Acrónimo en inglés. *Red Hat Enterprise Linux*. Versión Enterprise de Linux Red Hat.

RO. Requisito Operativo, referido a la Arquitectura Global (AG).

SaaS. Acrónimo en inglés. *Software as a service*. *Software* como servicio.

SAN. Acrónimo en inglés. *Storage Area Network*. Red de área de almacenamiento.

SDN. Acrónimo en inglés. *Software Define Network*. Redes definidas por *software*.

SIEM. Acrónimo en inglés. *Security Information and Event Management*. La gestión de eventos e información de seguridad.

SLA. Acrónimo en inglés. *Service Level Agreement*. Acuerdo de nivel de servicio.

SO. Sistema operativo.

SP. Acrónimo en Inglés. *Special Publication*. Publicación especial, referido a publicaciones del NIST.

StaaS. Acrónimo en inglés. *Storage as a Service*. Almacenamiento como servicio.

STIC. Seguridad de las Tecnologías de la Información y las Comunicaciones.

TFM. Trabajo Fin de Master.

TIC. Tecnologías de la Información y las Comunicaciones.

UE. Unión Europea.

VLAN. Acrónimo en inglés. *Virtual LAN*. Red de área local virtual.

VM. Acrónimo en inglés. *Virtual machine*. Máquina virtual.

VMM. Acrónimo en inglés. *Virtual machine monitor*. Monitor de máquina virtual.

VPN. Acrónimo en inglés. *Virtual Private Network*. Red privada virtual

WSUS. Acrónimo en inglés. *Windows Server Update Services*. Servicio de actualizaciones para servidores Windows.

ANEXO II: CONCEPTOS Y DEFINICIONES

Agregados. Referido al *cloud computing*. OpenStack. Agregados de *host*: La agregación de *hosts* divide una zona de disponibilidad en unidades mediante la agrupación de los *hosts*. En esta pestaña podemos crear nuevos agregados de *host* y obtener información sobre las zonas de disponibilidad [95].

API Endpoint. Referido al *cloud computing*. OpenStack. Un punto de acceso a un servicio o API *Endpoint* es una referencia que incluye entre otros parámetros, la dirección IP y puerto en el que escucha el servicio y la región del mismo. Se refiere a una URI.

API REST. REST es un estilo de arquitectura *software* para sistemas distribuidos. API REST es por tanto una interfaz que está diseñada bajo esta arquitectura. El término describe cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo SOAP.

Es posible diseñar sistemas de servicios web de acuerdo con el estilo arquitectural REST de Fielding (el ingeniero que acuñó el término) y también es posible diseñar interfaces XMLHTTP de acuerdo con el estilo de llamada a procedimiento remoto (RPC), pero sin usar SOAP. Estos dos usos diferentes del término REST causan cierta confusión en las discusiones técnicas, aunque RPC no es un ejemplo de REST.

Artefactos. Referido al *cloud computing*. OpenStack. Conjuntos de componentes de una servicio: Imágenes, redes, *routers*, etc...

Autenticación. Acto de confirmar la identificación de un usuario.

Bond. *Bonding* es una forma de obtener enlaces redundantes en *bridges*, tanto en aparatos de alta gama, como en máquinas con *software* libre.

CIS.

1. Acrónimo en inglés. *Communications and Information Systems*. Sistemas de Información y comunicaciones.
2. Acrónimo en inglés. *Computer (-based) Information Systems*. Sistemas de Información basados en computadoras.
3. *CIS/TIC*. Referido de forma agrupada a las Tecnologías y a los Sistemas de Información y Comunicaciones.

Cloud. Tecnologías que habilita al usuario el consumo de servicios pagando solo por lo que se consume, de forma escalable, elástica, flexible y por medio de un portal de auto aprovisionamiento

Cloud Computing. Computación en la nube. Hace referencia a una tecnología que mediante el acceso remoto por medio de Internet (o de una red propietaria) a aplicaciones y programas (a *software*), al almacenamiento de archivos y al procesamiento de datos, se convierte en una alternativa a una computadora personal o servidor local o sistema más complejo.

Componentes del Servicio de Red (Neutron). OpenStack. Pueden resumirse en un servidor Neutron accesible mediante una API e instalado en el Nodo Controlador, un *plugin* de conmutación instalado en el Nodo Controlador y en el Nodo de Red, el correspondiente agente para dicho *plugin* instalado en el Nodo de Red y en cada Nodo de Computación, así como un agente de capa 3, un agente DHCP y un agente de metadatos instalados en el Nodo de Red.

Container-format. Formato que indica si la imagen está almacenada en un formato de fichero que también contiene metadatos acerca de la máquina virtual. El Servicio de Imágenes y otros servicios de OpenStack actualmente no soportan este parámetro por lo tanto todas las máquinas virtuales almacenadas en Glance tendrán asignado a este parámetro el valor '*bare*' que implica que la imagen no

contiene un envoltorio de metadatos. Otros valores posibles aunque no soportados son ‘ovf’, ‘aki’, ‘ari’ y ‘ami’.

COTS.

1. Aplicable al *software*, se refiere a sistemas comerciales, no desarrollados específicamente para cumplir con normas o estándares de un ámbito específico.
2. Se denomina así a programas y aplicaciones informáticas producidos en serie y que han estado o están disponibles en el mercado, en contraposición a los programas y aplicaciones informáticos especializados que cubren las mismas funciones que los programas en serie , pero que se desarrollan internamente o con ayuda de consultores externos.

Credenciales. Datos conocidos solo por el usuario que demuestra que es él.

Daemons. Tipo especial de proceso informático que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario. Este tipo de programas continua en el sistema, es decir, que puede ser ejecutado en forma persistente o reiniciado si se intenta matar el proceso dependiendo de configuración del demonio y políticas del sistema.

Docker. Solución de virtualización ligera a nivel de sistema operativo que ofrece una manera simple y eficiente de desplegar contenedores Linux que ofrecen servicios a partir de aplicaciones débilmente acopladas. Es un sistema análogo a LXC.

EC2 (Elastic Cloud Computing). Es un servicio web que proporciona capacidad informática con tamaño modificable en la nube. Está diseñado para facilitar a los desarrolladores recursos informáticos escalables basados en web.

Endpoint. Dirección de red accesible, normalmente descrita por una URI, donde se puede tener acceso al servicio.

Formato de disco. (Disk-format). Es cada una de las preparaciones del disco (soporte físico) para almacenar datos e incluye aspectos como la recuperación de los mismos y la corrección de errores.

Desde el punto de vista del *cloud computing* el significado es análogo pero se aplica al almacenamiento virtual. Existen muchos formatos diferentes:

- Raw: es el formato más simple y es soportado de manera nativa por los hipervisores KVM y Xen. Puede considerarse como el equivalente binario de un fichero de dispositivo de bloque.
- qcow2 (QEMU *copy-on-write* versión 2): este formato es comúnmente utilizado con el hipervisor KVM. Las imágenes que usan este formato tiene un tamaño más pequeño que las que utilizan el formato Raw. Además este formato permite crear *snapshots*.
- ami/aki/ari: formato inicialmente soportado por Amazon EC2 compuesto por tres ficheros:
 - ami (*Amazon Machine Image*): imagen de máquina virtual en formato raw.
 - aki (*Amazon Kernel Image*): un fichero kernel que el hipervisor carga inicialmente para arrancar la imagen. Para una máquina Linux, este sería el fichero “vmlinuz”
 - ari (*Amazon Ramdisk Image*): un fichero ramdisk opcional montado en tiempo de arranque de la máquina virtual. Para una máquina Linux este sería el fichero “initrd”.
- UEC tarball (*Ubuntu Enterprise Cloud*): un comprimido construido con “gzip” o “tar” que contiene ficheros ami, aki y ari.
- VMDK (*Virtual Machine Disk*): formato de imagen utilizado por el hipervisor ESXi de VMware.
- VDI (*Virtual Disk Image*): formato de imagen utilizado por Oracle VirtualBox. Ninguno de los hipervisores disponibles en OpenStack soporta este formato por lo que la imagen debe ser convertida a otro formato antes de poder utilizarse para arrancar una máquina virtual.

- VHD (*Virtual Hard Disk*) y VHDX: formato utilizado por Microsoft Hyper-V.
- OVF (*Open Virtualization Format*): formato de empaquetado definido por el grupo de estandarización *Distributed Management Task Force* (DMTF) y contiene uno o varios ficheros de imágenes, un fichero XML de metadatos y otros ficheros adicionales.
- ISO: formato de imagen formateado con el sistema de ficheros de solo lectura ISO 9660 usado para CDs y DVDs. No suelen ser el primer formato de imagen en el que se piensa, sin embargo, es capaz de contener un sistema de ficheros con un sistema operativo instalado que permite arrancar una máquina virtual.
- Docker: formato de imagen utilizado para crear contenedores Docker.

Framework. Estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, que puede servir de base para la organización y desarrollo de *software*. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Front-end. Parte del *software* que interactúa con el o los usuarios.

Hadoop. *Framework* de *software* que soporta aplicaciones distribuidas bajo una licencia libre.¹ Permite a las aplicaciones trabajar con miles de nodos y petabytes de datos.

Hosting. Puede definirse como el uso que se hace de todos o parte de los recursos de un grupo de servidores y sistemas de comunicaciones (memoria, espacio del disco duro, ancho de banda, etc.), previamente acordado por contrato y para un propósito. A diferencia del *cloud computing* los límites acordados del servicio se pagan se utilicen o no y quedan reservados para el cliente. Cuando hablamos de *hosting* nos referimos generalmente al alojamiento de sitios web en un centro de datos.

Housing. Es el alquiler de un espacio en un CPD para que los clientes pueden implantar sus propios servidores en un entorno dotado de servicios de soporte como electricidad, contraincendios, refrigeración, seguridad y alta conectividad. Los servidores son gestionados por los clientes, no por el proveedor del servicio.

Imagen. Referido al *cloud computing*. OpenStack. Una imagen de una máquina virtual consiste en un fichero que contiene la imagen de un disco duro con un sistema operativo pre-instalado y que sirve como plantilla para generar el contenido del disco duro de la instancia que se desea crear con Nova

Instancia. Clon de una imagen que se crea a demanda del usuario en uno de los nodos del *cloud*.

IPs flotantes. Dirección IP pública que puede asociarse a diferentes instancias con el fin de acceder a ellas desde fuera.

KVM. Máquina virtual basada en el núcleo. Es una solución para implementar virtualización completa con Linux.

LDAP. Son las siglas de *Lightweight Directory Access Protocol* (en español Protocolo Ligero/Simplificado de Acceso a Directorios) que hacen referencia a un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

Linux network namespaces. Instalación de Linux que permite tener un conjunto de interfaces y entradas a la tabla de enrutamiento simples.

Metadatos. El concepto de metadatos es análogo al uso de índices para localizar objetos en vez de datos. Por ejemplo, en una biblioteca se usan fichas que especifican autores, títulos, casas editoriales y lugares para buscar libros. Así, los metadatos ayudan a ubicar datos.

NATO C3 Classification Taxonomy. Es un marco de trabajo, una herramienta, que provee la manera de sincronizar las actividades propias de Consulta, Mando y Control en la Alianza.

Network namespace (espacio de nombre de red). Es una replicación lógica del *stack* de red que incluye sus propios dispositivos e interfaces de red, sus propias tablas de rutas, su propio proceso *Iptables* con sus propias reglas para filtrado de tráfico y traducción de direcciones.

Nodo. Punto de intersección, conexión o unión de varios elementos que confluyen en el mismo lugar. Hace referencia al sistema donde se instalan varios componentes con un mismo propósito como partes de un todo.

Nova-network. Antiguo componente de OpenStack, anterior a Neutron y para los mismos propósitos aunque no tan versátil. Puede continuar usándose en las últimas versiones de OpenStack aunque en la mayoría de los grandes despliegues de producción es remplazado por el proyecto de red de Neutron. Nova-network permite a los proveedores de servicios crear redes capa 2 a partir de utilizar la tecnología de Linux Bridge (*brctl*). Neutron permite crear redes capa 3.

Orquestación. Gestión de sistemas y automatización de infraestructura de la nube.

Preboot eXecution Environment (PXE). Entorno de ejecución de pre-arranque, es un entorno para arrancar e instalar el sistema operativo en computadoras a través de una red, de manera independiente de los dispositivos de almacenamiento de datos disponibles (como discos duros) o de los sistemas operativos instalados

QEMU. Emulador de procesadores basado en la traducción dinámica de binarios (conversión del código binario de la arquitectura fuente en código entendible por la arquitectura huésped). QEMU también tiene capacidades de virtualización dentro de un sistema operativo, ya sea GNU/Linux, Windows, o cualquiera de los admitidos.

Qpid. Sistema de mensajería que implemente AMQP (Advanced Message Queuing Protocol). Proporciona gestión de transacciones, colas, distribuciones, seguridad, gestión y soporte de multiplataformas heterogéneas.

Regiones. Referido al *cloud computing*. OpenStack. Es una agrupación de servicios en Keystone.

Representational state transfer o REST. Es un estilo de arquitectura software para sistemas hipermedia distribuidos como la *World Wide Web*. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

Rol. Personalidad que un usuario asume y que este sabe a qué cosa específicas puede acceder con ese rol.

Sabores. Referido al *cloud computing*. OpenStack. Conjunto de parámetros que definen el *hardware* virtual con el que Nova arrancará una imagen.

Servicio. Proporciona uno o más *endpoints* a través de los cuales, los usuarios pueden acceder a operaciones de rendimiento y accesos de recursos.

SIEM. La gestión de eventos e información de seguridad es un enfoque de gestión de la seguridad que busca proporcionar una visión holística de la seguridad de la tecnología de la información de una organización.

Tenant. Referido al *cloud computing*. OpenStack. Es un contenedor utilizado para agrupar y aislar recursos. Los *tenants* igualmente agrupan y aíslan objetos de identidad. Cada usuario puede ser asignado a varios *tenants* y tener diferentes roles en cada *tenant*, igualmente un usuario puede tener varios roles en un mismo *tenant*. (Un rol no es más que un conjunto de permisos y privilegios para realizar un conjunto de operaciones específicas).

Token. Un texto de bits arbitrarios que es usado para acceder a los recursos. Cada *token* dice que servicios se pueden acceder.

Usuario. Representación digital de una persona, sistema o servicio que quiere usar los servicios de la nube de OpenStack.

Volúmenes. Referido al *cloud computing*. OpenStack. Unidad de disco lógico.

ANEXO III: GUÍA DE SEGURIZACIÓN.

aIII.1. Objeto

El presente anexo está orientado a la seguridad de la instalación, configuración y mantenimiento de un sistema Ubuntu OpenStack.

Este documento proporciona a los administradores de sistemas una guía que les permite configurar de forma segura los servidores sobre los que se despliega una nube privada basada en OpenStack, aumentando su resistencia ante ataques externos y disminuyendo los posibles daños cuando éstos sean realizados sobre las máquinas.

Inicialmente se instala el sistema operativo Ubuntu Server 18.04 LTS en las máquinas o nodos y se securiza de acuerdo con las recomendaciones de la guía CCN-STIC-612 [44]. Precisamente porque esta guía se encuentra desactualizada, pues se refiere a la versión de Ubuntu Server 12.04 LTS, la cual ya está fuera de soporte, se ha elaborado este anexo III para ofrecer una adaptación de la misma y al mismo tiempo ver como se debe afrontar un *hardening* de un sistema.

La seguridad es una mayor parte de gestión y planificación y una menor de configuración para que posteriormente se ejecuten de manera periódica todas las tareas y actividades que permitan mantener el sistema a salvo de ataques y errores.

El orden de instalación del sistema, como ya se ha alumbrado sería:

- Instalar físicamente el *hardware* que soportará el sistema de nube, y a continuación garantizar la seguridad física y electrónica del sistema.
- Instalar los SO de todos los servidores y robustecerlos conforme a este anexo III.
- Por último se realiza la instalación de componentes de OpenStack y se securizan de acuerdo a las recomendaciones de la *OpenStack Security Guide* [96], y a las recomendaciones de las guías de instalación de los diversos componentes ampliamente referenciadas durante este TFM.

No se van a elaborar anexos explicando el *hardening* de cada componente, o servicio a servicio, pues las documentaciones referenciadas se encuentran actualizadas debidamente, y pueden seguirse, aunque como ya se apuntó en una de las conclusiones no es algo trivial hacerlo.

Solo se ha elaborado este anexo como esfuerzo de actualización de la guía CCN-STIC-612 precisamente para que el presente TFM pueda constituir un punto de partida para trabajos futuros.

aIII.2. Seguridad durante la instalación del sistema operativo

A continuación se describen los puntos relevantes y decisiones a tomar en cuanto a la seguridad durante el proceso de instalación de sistemas Ubuntu Server 18.04 LTS.

Inicialmente se debe realizar una instalación mínima del SO y luego añadir exclusivamente los servicios y paquetes necesarios de acuerdo con las decisiones de diseño del sistema, manteniendo la mínima superficie de ataque posible (lo más reducida que se pueda). El instalar Ubuntu Server 18.04 LTS y no otra versión, responde a esta filosofía.

Un sistema conectado a Internet está expuesto a los ataques desde el exterior, por tanto la instalación debe realizarse desde DVD o un dispositivo de memoria USB *bootable*. Cuando el sistema se haya instalado y securizado para estar expuesto a los riesgos procedentes de Internet, si su propósito es conectarse, podrá hacerse.

El sistema se instalará manteniéndolo aislado de Internet y redes locales de inferior nivel de seguridad. Para sistemas clasificados, nivel de seguridad equivale a nivel de clasificación.

aIII.2.1. Imágenes del sistema operativo

Para la instalación de Ubuntu. Lo más inmediato es descargar la versión server del sitio oficial: <https://www.ubuntu.com/#download> [97].

El archivo descargado se corresponde con la imagen “ubuntu-18.04.1-live-server-amd64.iso”, la cual contiene el instalador más nuevo, y que al ser una versión *server* está en línea de comandos. Este instalador produce en ciertos sistemas un problema de *loop* bastante recurrente, reiniciándose constantemente, si se da el caso, en vez de esta versión puede descargarse la misma distro pero con el instalador anterior en la url: <http://cdimage.ubuntu.com/releases/18.04/release/> [98].

Con la imagen “ubuntu-18.04.1-server-amd64.iso” no se producen los problemas de reinicio del instalador.

aIII.2.2. Dispositivo de arranque

A continuación puede “quemarse” la iso en un DVD, o utilizar un *pendrive bootable*.

Para la creación de un *pendrive bootable*, Canonical aconseja descargar la utilidad rufus portable de la web oficial: <https://rufus.akeo.ie/> [99]. En el momento de la redacción de este documento la versión es la rufus-3.3.

aIII.2.3. Primeros pasos de la instalación

Durante la instalación, tras la elección del idioma del instalador se ofrecen las opciones que se muestran en la figura A3-1.



Figura A3-1. Instalación de Ubuntu. Carátula de inicio.

De las tres opciones de instalación, la primera es para realizar una instalación mínima de un servidor, las dos siguientes son para instalar un *server* orientado a MaaS, ‘Metal as a Service’, una opción que

Canonical pone a nuestra disposición para instalar un sistema orientado a ser controlador de una infraestructura de nube y que ya se ha explicado en el capítulo 3 de este TFM.

Seleccionamos ‘Instalar Ubuntu Server’ la primera de las opciones de acuerdo con la intención de mantener reducida la superficie de ataque realizando una instalación mínima del sistema operativo.

A continuación seleccionamos la ubicación para el sistema que implica el idioma de instalación del SO, y el idioma del teclado. El sistema tras detectar y montar el dispositivo de instalación (DVD o USB) cargará un fichero de pre-configuración (debconf), detectará los controladores de discos virtuales del fabricante del *hardware*, cargará componentes del instalador y detectará el *hardware* de red. En este momento hay que detener el proceso de configuración automática de la red, para lo que hay que estar muy atento (realizando hasta cuatro pulsaciones de cancelación), y seleccionar la opción ‘No configurar la red en este momento’, como se muestra en la figura A3-2.

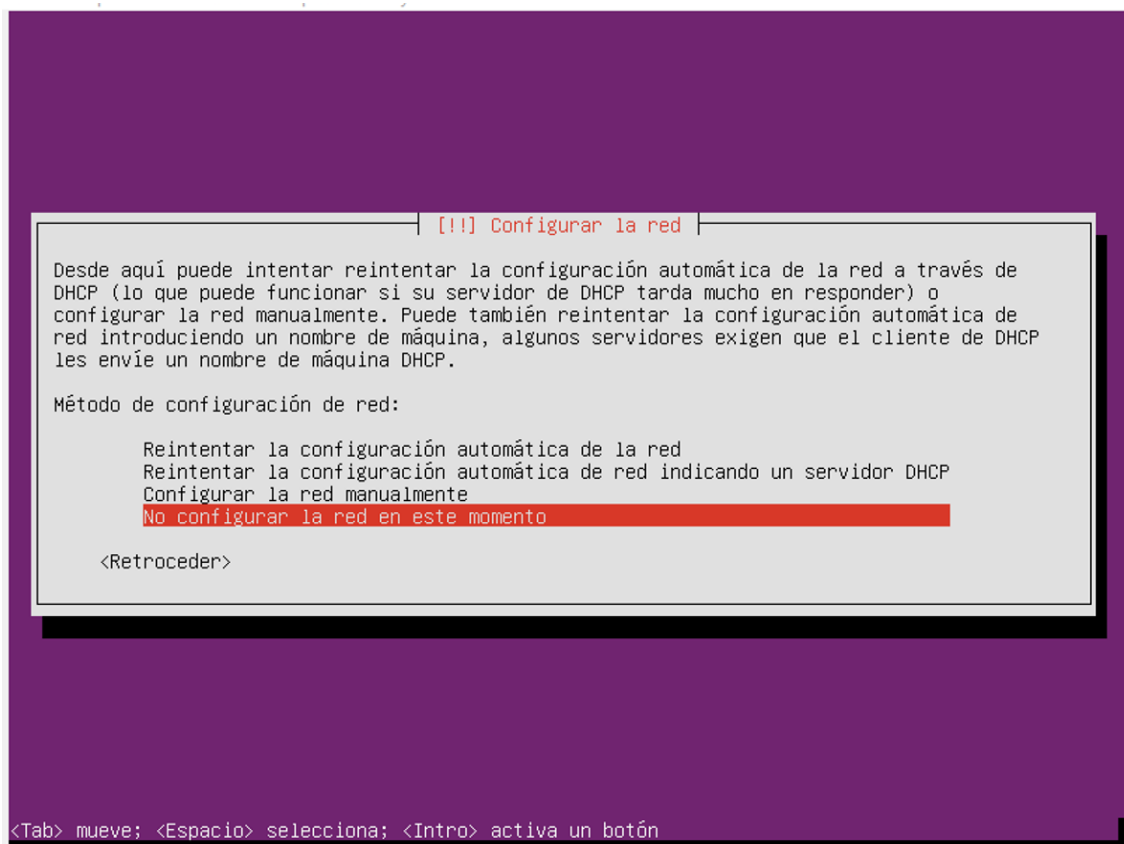


Figura A3-2. Instalación de Ubuntu. Configuración de red.

Durante la instalación no debemos configurar la red y hay que cancelar la detección automática por DHCP para la configuración porque cualquier indisponibilidad del servidor DHCP afectaría a los servicios del sistema. Lo mejor es realizar la configuración manualmente una vez instalado el sistema, cuando esta sea necesario.

Todas las configuraciones e instalaciones complementarias que puedan realizarse tras la instalación mínima del sistema se harán después, en el momento que se requieran.

Tras las opciones de red se da un nombre al sistema y comienza la configuración de la cuenta de usuario.

aIII.2.4. Configuración de la cuenta de usuario

Lo primero que se nos solicita es un nombre completo del usuario (usuario diferente a *root* o para realizar tareas que no sean administrativas). Este nombre se usa por el sistema para identificar al usuario en envíos de correo y similares para lo que un nombre completo es lo adecuado.

Luego se establece el *login* de dicho usuario. Con este *login* se creará la cuenta de usuario, y debe comenzar por una letra minúscula seguida de más números y letras minúsculas. Una buena política es no utilizar un *login* que identifique a personas físicas ni de pistas sobre la organización, para evitar ataques de ingeniería social. Tampoco deben utilizarse usuarios genéricos que varias personas físicas utilicen en el sistema compartiendo las contraseñas.

A continuación se nos solicita introducir la contraseña del usuario. Todas las contraseñas en el sistema deben establecerse de acuerdo con una política de contraseñas impuesta por la organización:

- Longitud mínima de la contraseña.
- Juego de caracteres admitido.
- Exigir combinar caracteres en mayúsculas, minúsculas, números y caracteres especiales.

Son buenos criterios usar contraseñas de al menos 10 caracteres, mayúsculas, minúsculas, números y caracteres especiales. Otra buena práctica es utilizar frases que los usuarios recuerdan mucho mejor, terminadas con un número y con alguna mayúscula. En general las frases por su mayor longitud y semántica como contraseñas son más difíciles de averiguar ante ataques de fuerza bruta.

“No se deben usar contraseñas genéricas o por defecto, así como tampoco se deben utilizar palabras presentes en diccionarios de cualquier lengua, ni contraseñas relacionadas con información pública, personal, sobre el Organismo o sobre la función de la máquina. En la medida de lo posible se deberán escoger contraseñas que sean aceptadas y fáciles de recordar por los usuarios sin que ello perjudique en la calidad de las claves” [44].

Después de la instalación se realizarán configuraciones adicionales sobre las cuentas de usuario y sus contraseñas.

Tras establecer la contraseña de la cuenta se configurará la zona horaria para el reloj del sistema, se detectarán los discos y se iniciará el particionado de los mismos en volúmenes.

aIII.2.5. Particionado de los discos

El particionado de los discos debe hacerse buscando el mayor control posible del sistema de ficheros, y aumentar el rendimiento y seguridad del sistema.

Los directorios en los que algún usuario tenga permisos de escritura deberán estar en un volumen lógico separado. Por ejemplo los directorios: /home, /tmp y /var/tmp.

Los directorios que contengan datos dinámicos y su tamaño pueda variar sustancialmente deberán estar en una partición o volumen lógico separado. Por ejemplo los directorios: /var y /var/log.

Los directorios que tengan datos estáticos que no vayan a ser modificados estarán en un volumen lógico separado y se montará dicha partición con permisos de sólo lectura.

Para el particionado se ejecutará la opción de realizar una partición manual mediante volúmenes lógicos (LVM). El cifrado de volúmenes es una opción a considerar por la organización durante el diseño del sistema de nube privada. A continuación al decir partición o volumen lógico nos estamos refiriendo a lo mismo.

Todos los volúmenes lógicos se deben formatear con *journaling ext4*, excepto la SWAP que se formatea como SWAP.

Memoria SWAP. El tamaño de este volumen dependerá del *hardware*, desde sus posibilidades de expansión hasta del tipo de disco. En general no será necesario reservar para este espacio más de 8 Gb, y el tamaño inicial debería ser:

- Si el equipo tiene menos de 1 Gb de RAM, la SWAP deberá ser como mínimo del doble de la memoria RAM y hasta 2 Gb como máximo.
- Si el equipo tiene más de 1 Gb y menos de 2 Gb de RAM, la SWAP deberá ser como mínimo de la cantidad de la memoria RAM más 2 Gb, con un máximo de 4 Gb.
- Si el equipo tiene más de 2 Gb de RAM, la SWAP deberá ser del doble de la RAM, pero por encima de 8 Gb hay que hacer un estudio más pormenorizado de la necesidad de SWAP, y además debería contarse con un disco SSD, pues el sistema puede volverse muy lento.

Posteriormente puede aumentarse el tamaño de este volumen, o activar particiones de intercambio adicional, aunque la mejor opción siempre será aumentar la RAM del sistema.

Directorio /boot. Este directorio contiene el arranque del núcleo del sistema, por lo que debe instalarse en un volumen lógico del sistema aislado de los demás. Suele bastar con un tamaño de unos 300 Mb, pero si hay disco suficiente puede reservarse hasta 1 Gb, poco más.

Debe formatearse la partición con *journaling ext4* como se ha dicho antes, pero hay que tener en cuenta que el arranque del sistema está condicionado por la BIOS del mismo y esto puede afectar al formato de la partición y a algunos aspectos relacionados. Las BIOS clásicas están ligadas a la arquitectura x86 del procesador y funcionan a 16 bits en *real-mode* lo que limita sus capacidades, sobre todo a los procesadores de 64 bits.

Actualmente las máquinas funcionan mediante *Extensible Firmware Interface* Unificado (UEFI), que es un estándar que sustituye a la BIOS. Primero EFI incorporó un gestor de arranque específico que superaba las limitaciones de las BIOS clásicas llegando a hacer innecesario implementaciones de cargadores de arranque específicos de fabricantes, como LILO o el propio cargador de Windows. EFI disponía de una interfaz en línea de comandos. Posteriormente las implementaciones EFI fueron unificadas en UEFI, se dotaron de interfaces gráficas y de utilidades para asistencia remota.

Hay que estudiar la documentación del *hardware* pues puede ser necesario que se realicen particiones específicas por ejemplo del tipo EF00 y se formateen con FAT32. Luego el instalador montará la partición en /boot/efi, o de alguna otra forma más específica que habría que indicarle accediendo a la BIOS tras la instalación.

Partición raíz '/. La raíz contendrá los directorios /etc, /bin, /sbin, /lib, y /dev para que el sistema pueda arrancar. Para estos directorios pueden bastar 20 Gb, pero puede darse formato en este volumen a todo el sobrante de haber reservado previamente el espacio para el resto de volúmenes lógicos. También puede decidirse dejar espacio sin particionar en el disco, para redimensiones posteriores, pero si se usa LVM es precisamente para favorecer esto, no siendo necesario.

Directorio /tmp. En este directorio todos los usuarios del sistema tienen acceso de escritura. Para evitar que los usuarios del sistema (o un atacante que logre suplantar a uno) puedan llenar todo el disco causando una denegación, es importante que se encuentre en un volumen lógico separado. Para este volumen pueden reservarse entre 1 y 5 Gb.

Posteriormente a la instalación debe configurarse una tarea automática (por ejemplo mediante el programador Cron/Crontab y un *script*) que borre los datos de la partición cada cierto tiempo.

Directorio /home. Un usuario podría, adrede o accidentalmente, escribir grandes cantidades de información llenando totalmente la partición en la que se encuentre el directorio, causando denegación, por lo que se creará un volumen lógico separado. Para este directorio debe reservarse el espacio en

función de las cuotas de uso que se asignen a los usuarios, pues cada uno tendrá un espacio reservado para él en /home. Es normal reservar varias decenas de Gb.

Directorio /usr. Este directorio es para archivos de aplicaciones y usuarios y su tamaño dependerá de lo que se instale en el sistema. Hay dos motivos para montarla separada del directorio raíz '/', el primero es que el volumen debe mantenerse como de solo lectura y cuando sea necesario el administrador habilitar permisos estrictos de ejecución y en su caso de escritura. El segundo es que pueda redimensionarse sin afectar al resto del particionado. Al igual que /home es normal reservar varias decenas de Gb.

Directorio /var. Este directorio se utiliza para datos dinámicos del sistema; *logs*, bases de datos, cola (*spool*) de impresión, correos, etc. Al ser datos dinámicos debe separarse la partición para evitar denegaciones por llenado del espacio del disco. En Ubuntu (y en general en los sistemas basados en Debian) se almacenan los paquetes descargados en el directorio /var/cache/apt/archives. Debe reservarse un espacio generoso para el volumen lógico creado para el directorio /var, como del doble que el usado para /home o /usr, que serán similares.

El subdirectorio /var/tmp puede ser necesario instalarlo en una partición específica. Pero en principio con separar /var parece suficiente en la mayoría de los casos.

Los tamaños son relativos al tamaño del disco y a la funcionalidad del servidor en casi todos los casos, por lo que deben tenerse en cuenta estos aspectos a la hora de dimensionar los volúmenes correspondientes. La tabla A3-1 da una idea del particionado respecto a un disco de 150 Gb de espacio en un sistema con 2 Gb de RAM.

Partición	Tamaño (Gb)	% del tamaño del disco
SWAP	4 (doble de la RAM)	
/boot	1	1,5%
/ (raíz)	20	12,5%
/tmp	5	7%
/home	20	12,5%
/usr	20	12,5%
/var	80	50%

Tabla A3-1. Particionado de un disco de 150 Gb.

Tras estas consideraciones volvemos al proceso de instalación. El instalador presenta las opciones para el particionado de discos y se nos ofrecen tres posibilidades de particionado guiado y una manual que es la que debemos elegir.

Al aceptar manual se nos presenta una serie de opciones para iniciar el particionado; las opciones dependerán de los discos instalados y de los volúmenes de esos discos.

Inicialmente el disco, todo su espacio, está en formato RAW, que el sistema ve, y dimensiona el disco, pero no tiene un formato con el que el SO pueda trabajar, por tanto hay que definir en el disco unidades de volumen. Cada volumen será como un disco duro de un tamaño determinado.

En un disco puede haber hasta cuatro volúmenes o particiones primarias y solo una partición extendida o secundaria (si se configura una extendida solo puede haber tres primarias). Los SO se instalan sobre particiones primarias, y las secundarias se utilizan para almacén de datos pudiéndose fragmentar en particiones lógicas que funcionarían como dispositivos independientes.

Todo el espacio libre del disco podemos configurarlo dentro de un volumen LVM (acrónimo de *Logical Volume Manager*) que es una implementación para el kernel de Linux que permite la gestión dinámica del tamaño de las particiones sobre lo que queda libre, es decir puedo configurar un volumen LVM de 50 GB y dentro de él crear particiones por ejemplo de 5 GB y asignarles punto de montaje a cada una, así en el futuro puedo empuqueñecer una y agrandar otra, el LVM me lo permite.

También podría dejar un trozo del LVM sin usar y usarlo más tarde para redimensionar. Todo esto permite que el sistema sobreviva desde el punto de vista de la disponibilidad ante problemas de tamaño y también porque las partes pueden ser respetadas cuando un problema sobre una de ellas ocurre pudiendo acceder a las otras.

La configuración de uno o más volúmenes LVM depende del uso futuro del sistema y es una cuestión de diseño al plantearse que componentes de OpenStack se instalarán.

Realmente interesa separar las particiones tal y como hemos explicado anteriormente con LVM, así que de las opciones manuales seleccionaremos todo el disco duro como único volumen primario: elegimos nuestro disco (sda). Antes de crear la tabla de particiones se nos muestra un aviso, decimos que si, y el instalador presenta el resumen de lo que hay por el momento, por ejemplo algo como lo siguiente:

SCSI3 (0,0,0) (sda) – 64.4 GB VMware, VMware Virtual S
Pri / log 64.4 GB ESPACIO LIBRE

A continuación creamos el grupo de volúmenes LVM, al que damos nombre, y una a una las particiones, que tenemos que especificar mediante su tamaño, el formato (*journalist ext4*), el punto de montaje y las opciones. (Las opciones pueden modificarse a posteriori una vez el sistema esté instalado).

La figura A3-3 muestra un resumen de las particiones creadas en un sistema con un disco de 10 Gb.

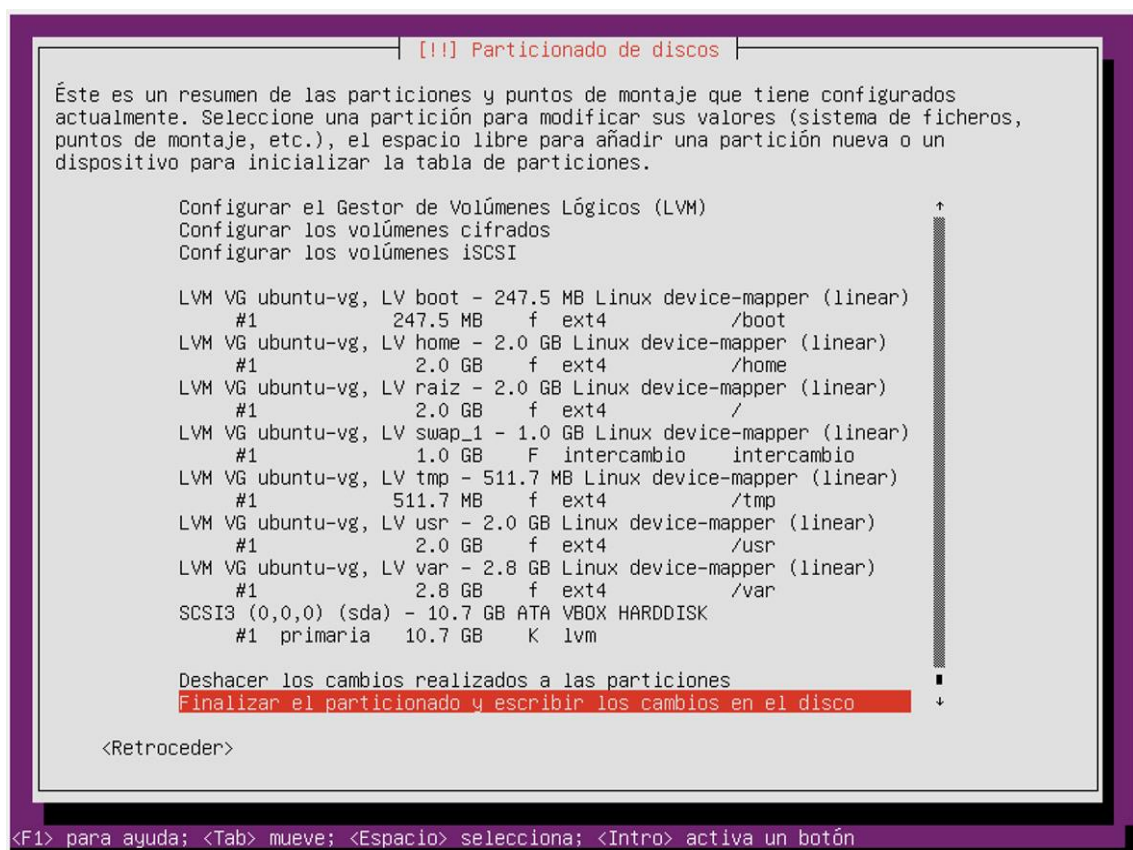


Figura A3-3. Instalación de Ubuntu. Particionado.

Como puede observarse en la figura, una vez asignada la utilización (ext4 o intercambio para SWAP) y el punto de montaje de los 6 volúmenes lógicos, seleccionamos finalizar el particionado y escribir los cambios en el disco y pulsamos ENTER, con lo que tras hacerse inicia la instalación del sistema base.

aIII.2.6. Instalación del sistema base y configuraciones finales.

La instalación se realiza sin intervención, y a continuación se nos solicita configurar el gestor de paquetes. Cancelaremos todas estas configuraciones, y seleccionaremos que las actualizaciones no se realizarán automáticamente (sin actualizaciones automáticas). Posteriormente trataremos de las actualizaciones del sistema.

Continuamos con la instalación de paquetes adicionales y programas. No se instalará nada pues haremos siempre instalaciones mínimas.

A continuación se nos preguntará si debemos instalar el cargador de arranque GRUB, contestamos que si y seleccionamos instalarlo en /dev/sda. Con esto se completa la instalación, desmontamos el USB con la imagen (o del *hardware* virtual desmontamos el disco imagen desde la interfaz de VirtualBox) y arrancamos el sistema y hacemos comprobaciones sobre los *login* de *root* y del usuario creado.

aIII.3. Primeras configuraciones de seguridad del sistema operativo.

Un usuario malintencionado o un hacker puede utilizar directorios temporales, como /tmp, /var/tmp y /dev/shm, para almacenar y ejecutar código. Esto ocurre por las características de montaje de estos directorios y por los permisos de los mismos.

aIII.3.1. Modificar las características de montaje del directorio /dev/shm.

Se trata de impedir la ejecución de código malicioso (en general de cualquier código por cualquier usuario) desde este directorio (/dev/shm).

Uno de los puntos de montaje sobre el que se monta el archivo tmpfs es /dev/shm. Éste es un archivo temporal de almacenamiento del sistema de archivos que utiliza la memoria RAM para el almacenamiento de respaldo. Puede funcionar como una memoria compartida de aplicación que facilita la comunicación entre procesos (IPC).

Listamos los puntos de montaje:

```
$ mount -l | less
```

Entre los puntos listados se encuentra:

```
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
```

... que quiere decir ‘archivo tmpfs montado en /dev/shm tipo tmpfs con las opciones lectura y escritura, ejecución acorde con los privilegios del usuario y no interpreta dispositivos especiales de bloques en el sistema de archivos montado’.

Las opciones de montaje listadas se corresponden con los permisos de lectura y escritura (rw), necesarios para el uso del fichero tmpfs. Además ya vienen establecidas las opciones nodev y nosuid, pero nos falta añadir noexec, que es la que impide específicamente la ejecución de binarios. Con noexec ningún binario podrá ejecutarse en el punto de montaje en el que se especifica.

Para modificar las opciones de montaje hay que modificar el archivo /etc/fstab. La estructura de este archivo usa una sintaxis especial. Cada línea está formada por seis campos (cada uno con su significado), separados por espacios en blanco (espacio, tabuladores o una combinación). Procedemos a editar el archivo fstab y al final insertamos la línea:

```
tmpfs    /dev/shm    tmpfs defaults,nodev,nosuid,noexec    0    0
```

Donde los campos están separados por tabuladores. Forzamos el montaje de todas las unidades reiniciando el sistema, y comprobamos los puntos de montaje otra vez con la misma orden anterior:

```
$ reboot
```

... y tras hacer login:

```
$ mount -l
```

... y vemos que la línea de tmpfs en /dev/shm muestra:

```
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,noexec)
```

... habiéndose añadido la opción noexec.

Hay otras entradas tmpfs pero todas muestran estas mismas opciones, excepto una correspondiente al punto de montaje /run que precisamente requiere permitir la ejecución.

aIII.3.2. Sustituir el directorio /var/tmp por un enlace simbólico al /tmp

Ubicar un directorio temporal dentro de /var, cuando hemos especificado una partición específica para los archivos temporales separada de las demás, para la misma finalidad no tiene mucho sentido. Por otra parte los permisos de los archivos temporales generados por el sistema se pueden controlar mejor desde un punto centralizado.

Para impedir la ubicación de temporales en el directorio /var, eliminaremos el directorio /var/tmp sustituyéndolo por un enlace simbólico al /tmp, ya que conlleva las mismas implicaciones de seguridad que el directorio /tmp.

```
$ sudo rm -rf /var/tmp
```

```
$ sudo ln -s /tmp /var/tmp
```

aIII.3.3. Modificar las características de montaje del directorio /tmp

Se debe impedir la ejecución de código arbitrario en el directorio /tmp, igual que en /dev/shm, para lo cual se deberían añadir las mismas opciones de montaje al archivo tmpfs en /tmp que las que hemos añadido a tmpfs en /dev/shm, pero al listar los puntos de montaje en nuestra versión de Ubuntu (18.04 LTS) no se realiza el montaje de tmpfs en /tmp, así que una vez comprobado no es necesario.

aIII.3.4. Modificar opciones de montaje del sistema de ficheros

Se trata de extender criterios como los ya tratados en los epígrafes anteriores aIII.3.1 y aIII.3.3 relativos a las opciones de montaje de esos dos directorios en particular, al resto del sistema de archivos. Todo, como ya se ha dicho se hace modificando el fichero '/etc/fstab'.

Con carácter general se debe añadir opción 'nodev' a particiones que no son raíz '/'. Por defecto se permite montar dispositivos como ficheros en todos los puntos del sistema de ficheros lo cual no tiene sentido más que en la raíz porque contiene el directorio '/dev'.

Añadir la opción 'nodev' en las particiones cuyo punto de montaje no es la raíz y usan el sistema de ficheros ext4, que de acuerdo a nuestra instalación serían: /boot, /home, /temp, /usr y /var.

Como ya hemos dicho, para modificar las opciones de montaje hay que modificar el archivo /etc/fstab que usa una sintaxis especial, con líneas formadas por seis campos (cada uno con su significado), separados por espacios en blanco (espacio, tabuladores o una combinación). Esta opción se debe añadir en la cuarta columna del fichero 'fstab' y permite prevenir que los usuarios creen dispositivos no autorizados en particiones no destinadas a ello.

Añadir opción 'nosuid' a las particiones que no son raíz ni /usr. La opción 'nosuid' deshabilita la posibilidad de que en ese punto de montaje donde se especifica no puedan existir ficheros SUID, los

cuales solo deben estar en la partición raíz que contiene /bin y en /usr. Se añade en ‘fstab’ como a se ha explicado.

Añadir opción ‘nodev’, ‘nosuid’ y ‘noexec’ a particiones de medios extraíbles.

Se deben añadir estas opciones para impedir la ejecución de binarios y para que no puedan contener ficheros SUID ni montar dispositivos adicionales.

aIII.3.5. Primera configuración del firewall.

Una medida de seguridad básica es controlar las conexiones entrantes y salientes permitiendo solo las explícitamente autorizadas.

El núcleo de Linux tiene un firewall de estado, Netfilter. La herramienta para gestionarlo en línea de comandos es Iptables. Iptables permite establecer las reglas de filtrado básicamente con tres cadenas:

- INPUT filtra los paquetes de red con destino el sistema.
- OUTPUT filtra los paquetes de red con origen en el sistema
- FORWARD filtra los paquetes de red que cruzan desde una de las redes con la que tiene conexión el sistema hacia otra diferente.

La referencia para construir reglas bien es la documentación del proyecto Netfilter [100]. Adicionalmente puede aplicarse la guía CCN-STIC-419 CONFIGURACIÓN SEGURA CON IPTABLES [101], aunque resulta un poco anticuada pues es del 2007.

Ubuntu Server trae incorporada la aplicación UFW (Uncomplicated FireWall) que es una herramienta sencilla, desde luego más que Iptables, para gestionar Netfilter.

Ubuntu no configura por defecto ninguna regla en el *firewall*, salvo las que lo configuran abierto a todo tráfico. Daremos los siguientes pasos:

- Eliminar todas las reglas existentes.
- Configurar la política de denegación por defecto.
- Permitir el tráfico de entrada y salida para la interfaz de *loopback* ya que si se bloquea su tráfico puede impedir el correcto funcionamiento de algunos servicios.
- Permitir el tráfico de entrada y salida cuya conexión ya se ha establecido.

Tras esta primera configuración se deben añadir al *firewall* las reglas necesarias para permitir conexiones de entrada y salida exclusivamente a los servicios necesarios de acuerdo con la planificación realizada. Por ejemplo:

- Para permitir la resolución DNS.
- Para permitir la descarga de paquetes de los repositorios oficiales de Ubuntu (si procede la actualización en línea en el sistema).
- Para permitir las conexiones al servidor de SSH de la máquina para una determinada dirección IP.
- Para permitir la conexión con el servidor NTP.

Estas reglas pueden añadirse más tarde a medida que se vayan instalando los diferentes servicios y aplicaciones que las requieran.

Para hacer persistente las reglas establecidas se guarda la configuración mínima en un fichero para usarla cada vez que se inicie el sistema:

```
# iptables-save > /etc/cortafuegos.conf
```

A continuación se edita el fichero */etc/rc.local* y se le añade la línea:

```
iptables-restore < /etc/cortafuegos.conf
```

... y no olvidar al final:

```
exit 0
```

Tras reiniciar la máquina se observa como la configuración persiste. Cada vez que se modifiquen las reglas de Iptables habrá que volver a guardar el archivo *cortafuegos.conf*.

La tabla A3-2 muestra la secuencia de órdenes en el sistema para realizar la configuración del firewall.

Orden en el terminal	Salida en el terminal
# iptables -L	Chain INPUT (policy ACCEPT) target prot opt source destination Chain FORWARD (policy ACCEPT) target prot opt source destination Chain OUTPUT (policy ACCEPT) target prot opt source destination
# iptables -F	
# iptables -P INPUT DROP	
# iptables -P OUTPUT DROP	
# iptables -P FORWARD DROP	
# iptables -A INPUT -i lo -j ACCEPT	
# iptables -A OUTPUT -o lo -j ACCEPT	
# iptables -A INPUT -m state -- ESTABLISHED,RELATED -j ACCEPT	
# iptables -A OUTPUT -m state -- ESTABLISHED,RELATED -j ACCEPT	
# iptables -L	Chain INPUT (policy DROP) target prot opt source destination ACCEPT all -- anywhere ACCEPT all -- anywhere state RELATED,ESTABLISHED Chain FORWARD (policy DROP) target prot opt source destination Chain OUTPUT (policy DROP) target prot opt source destination ACCEPT all -- anywhere ACCEPT all -- anywhere state RELATED,ESTABLISHED
# iptables-save > /etc/cortafuegos.conf)	

Orden en el terminal	Salida en el terminal
<code># sudo nano /etc/rc.local</code>	Añadir líneas: <pre>[...] iptables-restore < /etc/cortafuegos.conf [...] exit 0</pre>
Guardar y salir de nano.	

Tabla A3-2. Configuración de Iptables.

A continuación configuraríamos las interfaces de red con la utilidad Netplan tal y como se ha explicado en el capítulo 3 de este TFM. La configuración de las interfaces debe estar de acuerdo con las decisiones de diseño del sistema que estamos construyendo. También deben consignarse las direcciones conocidas en el archivo `/etc/hosts`.

Una vez hayamos configurado la red y el *firewall* directamente con Iptables o con UFW, para comprobar que está bien, es conveniente realizar un escaneo de puertos con una herramienta de auditoría como NMAP, tanto dentro de la red interna como desde fuera. El resultado debe guardarse como línea base de configuración de puertos abiertos, cerrados y filtrados.

Una vez lograda esta configuración de base, cada vez que instalemos un servicio que nos sea necesario y que precise de conexiones de entrada y salida habrá que añadir las reglas oportunas. Veamos algunos ejemplos:

Permitir la resolución DNS (suponiendo que la IP del DNS sea 192.168.1.50):

```
# iptables -A OUTPUT -o eth0 -p udp -d 192.168.1.50 --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
# iptables -A INPUT -i eth0 -p udp -s 192.168.1.50 --sport 53 -m state --state ESTABLISHED -j ACCEPT
```

Permitir las conexiones al servidor de SSH de la máquina para una determinada dirección IP:

```
# iptables -A INPUT -p tcp -s 192.168.1.2 --dport 22 -m state --state NEW -j ACCEPT
```

Permitir la conexión con el servidor NTP (línea 9-10).

```
# iptables -A OUTPUT -o eth0 -p udp -d 91.189.94.4 --dport 123 -m state --state NEW,ESTABLISHED -j ACCEPT
# iptables -A INPUT -i eth0 -p udp -s 91.189.94.4 --sport 123 -m state --state ESTABLISHED -j ACCEPT
```

La verdad es que ‘iptables’ es poco amigable. La herramienta UFW (*Uncomplicated FireWall*) es típica en sistemas Ubuntu y facilita bastante el trabajo con Netfilter, Está instalada por defecto como ya se dijo antes:

```
# ufw status verbose
```

Por ejemplo para especificar la anterior política de denegación por defecto basta con las órdenes:

```
# ufw default deny incoming
# ufw default deny outgoing
```

Y para escribir reglas que hagan lo mismo que hicimos antes con Iptables, por ejemplo (suponiendo que la interfaz de salida es eth1):

- Permitir la resolución DNS:

```
# ufw allow in on eth1 proto udp from 192.168.1.50 port 53
```

```
# ufw allow out on eth1 proto udp to 192.168.1.50 port 53
```

- Permitir las conexiones al servidor de SSH de la máquina para una dirección IP:

```
# ufw allow in on eth1 proto tcp from 192.168.1.2 to any port 22
```

- Permitir la conexión con el servidor NTP.

```
# ufw allow in on eth1 proto udp from 91.189.94.4 port 123
```

```
# ufw allow out on eth1 proto udp to 91.189.94.4 port 123
```

Además cuando está instalado UFW, este permite por defecto tráfico ICMP, multicast, broadcast, averiguación DHCP, entre otros. Si no es deseable (y tratándose de una configuración de seguridad toda conexión permitida debe estar explícitamente justificada, y no vale decir que es por comodidad) dichas reglas deben comentarse en el fichero `/etc/ufw/before.rules`.

Para aprender a manejar UFW toda la información necesaria se puede encontrar en las referencias [102] y [103].

Por supuesto, en este apartado merece atención especial las necesidades de filtrado de los servicios de OpenStack [104]. Será necesario establecer reglas para:

- ... que funcione el *networking* entre los nodos de control, computación y de almacenamiento: Neutron.
- ... los servicios generales: DHCP, NTP, MariaDB, Memcached, Rabbit, Apache2,
- ... el servicio de identidad: Keystone.
- ... el servicio de computación: Nova [105].
- ... el servicio de imágenes: Glance.
- ... el portal Horizon.
- ... otros: Cinder, Swift, Heat, Celiometer, etc.

En la ‘*OpenStack Security Guide*’ [96], en el capítulo o apartado correspondiente a cada servicio pueden verse las necesidades de tráfico entre componentes.

También es necesario controlar como se conectan con el exterior las instancias, lo hacen a través de Neutrón y cada una lleva asociada una configuración del firewall del componente.

aIII.3.6. Configuración de la gestión de paquetes.

Todo sistema seguro debe mantenerse actualizado. Gestionar, instalar, desinstalar y realizar actualizaciones en las distribuciones Linux en general, y en particular Ubuntu, tiene sus propios métodos.

Todas las distribuciones basadas en Debian, como Ubuntu, utilizan el sistema de gestión de paquetes APT, (*Advanced Packaging Tool*). APT utiliza las fuentes o repositorios de la distribución consignados en el fichero `/etc/apt/sources.list`.

Además de APT¹⁹ hay otras utilidades de actualizaciones que podrían instalarse en el sistema, como Aptitude o incluso Yum (aunque este último no es de la familia Debian). En principio evitaremos instalar Aptitude, ni ninguna otra que el SO no trae de forma nativa.

Ubuntu (y las distros basadas en Debian) traen la utilidad Debconf para configurar parámetros de la aplicación que se esté instalando basándose en un régimen de preguntas al usuario, de hecho es lo que se hace durante la instalación misma del SO.

¹⁹ También está disponible apt-get.

Debconf puede configurarse en cuatro niveles de interacción con el usuario: crítico, alto, medio y bajo; y en función del nivel se harán más o menos preguntas al usuario. A nivel más bajo más interacción y por tanto más preguntas, y la consecuencia es que se harán menos configuraciones por defecto.

Una vez instalado el sistema, inicialmente Debconf está en nivel alto, y si vamos bajando el nivel puede que nos veamos en problemas por falta de conocimientos, pero es recomendable configurarlo a nivel medio, mediante ‘dpkg’:

```
# dpkg-reconfigure debconf
```

Los repositorios para las actualizaciones del SO y otras aplicaciones deben ser los oficiales (de cada fabricante) de las versiones estables y soportadas. No deben utilizarse repositorios no oficiales de la distribución o los fabricantes pues estos no garantizan las actualizaciones de seguridad. Cualquier otro repositorio que se necesite por cualquier causa, de versiones no estables, de aplicaciones mejoradas por terceros, que incluyan el código fuente de los paquetes no debe usarse salvo que se documente la necesidad.

Ubuntu organiza sus repositorios oficiales en cuatro partes o secciones, solo se deben usar las secciones *main* y *restricted*:

- *main*: contiene *software* con soporte oficial, con el código fuente disponible y que se puede modificar y distribuir con la misma licencia.
- *restricted*: contiene *software* con soporte oficial pero la licencia no es totalmente libre, por lo que aunque el código fuente esté disponible (en algunos casos será así y en otros no), no se puede modificar y distribuir con la misma licencia.
- *universe*: contiene *software* sin soporte oficial mantenido por una comunidad de usuarios (o entusiastas).
- *multiverse*: contiene *software* propietario.

Canonical mantiene los repositorios oficiales de Ubuntu y tiene los siguientes tipos:

- Repositorio de *software*: contiene el *software* de la distribución.
- Repositorio de actualizaciones: con las actualizaciones de *software*.
- Repositorio de actualizaciones de seguridad: con las actualizaciones de seguridad de todos los paquetes disponibles.

Además de estos tres hay otros repositorios, por ejemplo el de *partners* de Canonical que mantienen otros paquetes, o el de *backports*. El *software* de *partners* o terceros es reconocido por Canonical pero no son parte de Ubuntu, por lo que conviene no incluirlos a no ser que algún tipo de *software* que instalemos lo precise. El *software* del repositorio *backports* puede ser novedoso pero no está probado como el de la versión estable y no recibe actualizaciones de seguridad, por lo que no debe ser utilizado. (El repositorio se identifica en la misma url por *versionubuntu-backports*)

En función de los requisitos de seguridad del sistema pueden darse varios casos. Para cada uno de estos casos es necesario configurar la utilidad APT de manera diferente (mediante modificación del fichero */etc/apt/sources.list*), pero las siguientes medidas se aplicarán a todos los casos:

- No incluir repositorios de código fuente.
- No incluir los repositorios *universe* y *multiverse*.
- No incluir repositorios de *software* en *backport*.
- No incluir repositorios de *software* de *partners* a no ser estrictamente necesario.
- Incluir solo el repositorio principal, el de actualizaciones y el de actualizaciones de seguridad.

Los casos son:

- Caso 1. El sistema no puede conectarse a Internet ni a ningún otro sistema por lo que debe gestionar los paquetes desde los repositorios instalados en local, en un disco duro extraíble o en una ubicación de la red.

Aunque no es objeto de este documento explicar cómo se instala un repositorio en local, daré algunas indicaciones:

- Para esto hay que disponer en primer lugar de un servidor Ubuntu conectado a Internet desde el que nos descargaremos las actualizaciones de los repositorios habituales.
- Este servidor debe tener instalado el paquete ‘apt-mirror’, y un disco duro USB conectado y montado.
- En este servidor conectado a Internet, al que llamaremos *mirror*, deberemos configurar apt-mirror mediante el fichero de configuración /etc/apt/mirror.list. Deben añadirse los repositorios necesarios que se usarán en los servidores sin Internet para actualizarlos.
- En el *mirror* crear el repositorio local y limpiarlo.
- En el *mirror* ejecutar desde *cron* la actualización del repositorio local.
- En el *mirror* copiar los ficheros del directorio /var/spool/apt-mirror/mirror/ en el disco duro USB, copiando sólo aquellos ficheros que hayan sido actualizados. Luego este USB puede volcarse en una ubicación de la red si se desea.
- En el sistema sin conexión, que se desea actualizar deben configurarse en el archivo /etc/apt/sources.list las fuentes: (bionic es el nombre de la distro 18.04 LTS)

```
deb <RUTA_O_URL_DEL_REPOSITORIO> bionic main restricted
deb <RUTA_O_URL_DEL_REPOSITORIO> bionic-updates main restricted
deb <RUTA_O_URL_DEL_REPOSITORIO> bionic-security main restricted
```

- Caso 2. El sistema no puede conectarse a Internet pero sí a un servidor de actualizaciones o *mirror* especialmente dedicado y que se encuentra en la red corporativa.

No es objeto de este documento entrar en las condiciones de seguridad del *mirror*—su nivel de seguridad debe ser equivalente al del sistema—, y tampoco en si puede estar o no conectado a Internet.

El *mirror* se instalaría como se ha explicado en el caso 1, pero en este caso además se le instalaría un servidor Apache para que los ficheros estén accesibles por HTTP, y dentro de la carpeta raíz del servidor web crearíamos los enlaces simbólicos necesarios a /var/spool/apt-mirror/mirror/de.archive.ubuntu.com/Ubuntu.

En el sistema sin conexión, que se desea actualizar deben configurarse en el archivo /etc/apt/sources.list las fuentes: (bionic es el nombre de la distro 18.04 LTS)

```
deb http://nombre o ip del mirror/ubuntu/ bionic main restricted
deb http://nombre o ip del mirror /ubuntu/ bionic-updates main restricted
deb http://nombre o ip del mirror /ubuntu/ bionic-security main restricted
```

- Caso 3. El sistema puede conectarse a Internet pero a través de un *proxy*. Para poder actualizar Ubuntu server 18.04 LTS si el sistema está tras un *proxy* hay que hacer cambios en el directorio /etc/apt/apt.conf.d, donde crearíamos un archivo específico para incluir el *proxy* y que llamaríamos apt_proxy.conf. Su contenido debería ser algo como:

```
Acquire::http::Proxy "http://user:password@proxy:port";
Acquire::ftp::Proxy "ftp://user:password@proxy:port";
```

Podrían ser necesarias modificaciones en los archivos `/etc/environment` y `/etc/bash.bashrc`. Por último bastará con la adecuada configuración de los repositorios en el archivo `/etc/apt/sources.list`. (bionic es el nombre de la distro 18.04 LTS. Se incluyen los repositorios de actualizaciones de seguridad en España y el del site `'security.ubuntu.com/Ubuntu'`).

```
deb http://es.archive.ubuntu.com/ubuntu/ bionic main restricted
deb http://es.archive.ubuntu.com/ubuntu/ bionic-updates main restricted
deb http://es.archive.ubuntu.com/ubuntu/ bionic-security main restricted
deb http://security.ubuntu.com/ubuntu bionic-security main restricted
```

- Caso 4. El sistema se puede conectar directamente a los servidores de Canonical y fabricantes en Internet.
En este caso la conexión es directa y bastará con la adecuada configuración de los repositorios en el archivo `/etc/apt/sources.list`. (bionic es el nombre de la distro 18.04 LTS. Se incluyen los repositorios de actualizaciones de seguridad en España y el del site `"security.ubuntu.com/Ubuntu"`).

```
deb http://es.archive.ubuntu.com/ubuntu/ bionic main restricted
deb http://es.archive.ubuntu.com/ubuntu/ bionic-updates main restricted
deb http://es.archive.ubuntu.com/ubuntu/ bionic-security main restricted
deb http://security.ubuntu.com/ubuntu bionic-security main restricted
```

Para todos los casos anteriores, una vez configurados los repositorios en `sources.list` se requiere actualizar la base de datos de paquetes que el sistema guarda para la utilidad APT con el contenido de los repositorios configurados exclusivamente:

```
# apt-get update
```

... y a continuación hacer una actualización del sistema y de la distro:

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

Cada vez que se instale un paquete de aplicación, APT (o Aptitude, si se decidió su instalación) buscará sus dependencias para instalarlas también. Esos paquetes identificados como dependencias pueden ser necesarios (sin los cuales la aplicación no funcionaría), recomendados (son prescindibles) o sugeridos (añaden algo, funcionalidad...). Como norma solo se instalarán las dependencias necesarias, para lo que cualquier orden de instalación debe incluir la opción `'--no-install-recommends'`.

Cuando se desinstala un paquete o aplicación:

```
# apt-get remove --purge paquetex
```

... solo se eliminan los componentes de la aplicación y (con `--purge`) los archivos de configuración de esa aplicación, quedando en el sistema librerías y paquetes que fueron instaladas como dependencias necesarias. Es necesario eliminar todas las dependencias necesarias que otros programas no usan y digamos han quedado huérfanas:

```
# apt-get autoremove
```

Volviendo a las actualizaciones, en los sistemas con especificaciones de máxima seguridad, éstas deben realizarse manualmente, (no automatizarlas), tras haber evaluado si al instalarlas habría alguna repercusión en el sistema. Para eso algún administrador debe estar al tanto de las actualizaciones que se publican y es una buena práctica crear un *script* que avise de su disponibilidad (podría hacerse con el

paquete “cron-apt”, pero este no está en los repositorios main o restricted de la versión bionic, lo he comprobado). Para hacer el *script*:

- Creamos un fichero (script .sh) en /etc/cron.daily/ con algún nombre identificable, avisadeactualizaciones.sh, con el siguiente contenido:

```
(apt-get update && apt-get -s -o Dir::Etc::SourceList=/etc/apt/fuentesegura.list upgrade) |  
mail -s "Notificación diaria de actualizaciones de $(hostname)" correo@administrador.es
```

Para probar el script sin envío por correo electrónico voy a enviarlo a un archivo de texto:

```
(apt-get update && apt-get -s -o Dir::Etc::SourceList=/etc/apt/fuentesegura.list upgrade) >  
/home/jose/act.txt
```

- Le damos permisos de ejecución:

```
# chmod +x /etc/cron.daily/avisadeactualizaciones.sh
```

- Creamos un fichero de fuentes en /etc/apt/ (mismo directorio donde se encuentra sources.list) y lo llamamos fuentesegura.list, con la ruta del repositorio (o repositorios) concreto que nos interesa que es el de actualizaciones de seguridad:

```
deb <RUTA_O_URL_DEL_REPOSITORIO> bionic-security main restricted
```

- Si el sistema tiene conexión a Internet podrían ser:

```
deb http://es.archive.ubuntu.com/ubuntu/ bionic-security main restricted  
deb http://security.ubuntu.com/ubuntu bionic-security main restricted
```

aIII.3.7. Configuración de las contraseñas (autenticación).

Todas las contraseñas en el sistema deben establecerse de acuerdo con una política de contraseñas impuesta por la organización, la cual debe establecer los criterios de complejidad, los plazos de caducidad y la calidad de las mismas.

Los criterios de complejidad de las contraseñas pueden ser:

- Longitud mínima de la contraseña. Al menos 10 caracteres.
- Juego de caracteres admitido.
- Exigir combinar caracteres en mayúsculas, minúsculas, números y caracteres especiales.
- Otra buena práctica (alternativa) es utilizar frases que los usuarios recuerdan mucho mejor, terminadas con un número y con alguna mayúscula y/o carácter especial. En general las frases por su mayor longitud y semántica como contraseñas son más difíciles de averiguar.

En canto a la caducidad de las contraseñas, se establecen plazos para forzar cambios, etc:

- Forzar cambio de contraseña cada 30 a 360 días. Para sistemas clasificados los periodos están establecidos en las guías CCN-STIC.
Editar el fichero /etc/login.defs y fijar el valor de “PASS_MAX_DAYS”.
Un periodo demasiado pequeño hará que los usuarios construyan sus contraseñas con poca variación entre unas y otras, por ejemplo una palabra y añadan números consecutivos, lo cual no es seguro.
- Forzar un mínimo de días entre cambios de contraseña, por ejemplo de 2 a 7 días. Para sistemas clasificados los periodos están establecidos en las guías CCN-STIC.
Editar el fichero /etc/login.defs y fijar el valor de “PASS_MIN_DAYS”.
Un número de días demasiado grande puede comprometer la seguridad.

- Forzar el sistema para dar un aviso a los usuarios cuando acceden al sistema, unos días antes (por ejemplo 7 días), de que su contraseña está a punto de expirar. Editar el fichero `/etc/login.defs` y fijar el valor de “PASS_WARN_AGE”.
- Forzar el sistema para no permitir más de un número intentos incorrectos de *login* en el sistema, por ejemplo 3 o 5 intentos. Editar el fichero `/etc/login.defs` y fijar el valor de “LOGIN_RETRIES”. Sobre esta directiva prima la análoga configurada en el módulo PAN, pero si no estuviera configurado correctamente el parámetro “PAM_MAXTRIES” se aplicaría.
- Hay que forzar estos cambios en `/etc/login.defs` para los usuarios que se han dado de alta con anterioridad (hay que hacerlo usuario por usuario) mediante la instrucción *chage*:

```
# chage -M 180 -m 2 -W 7 <usuario>
```

Otros parámetros pueden establecerse en el archivo `/etc/login.defs` como el tiempo permitido para hacer *login* en segundos, y el algoritmo de encriptación para los hashes de las contraseñas, pero es preferible utilizar el módulo PAM (*Pluggable Authentication Modules*) para todas estas cuestiones.

PAM es un conjunto de aplicaciones para gestionar la autenticación y cada una tiene su configuración específica. Los archivos de configuración están en `/etc/pam.d.` como se muestra en la tabla A3-3.

Módulo	Archivo de configuración
login	<code>/etc/pam.d/login</code>
pam_cracklib	<code>/etc/pam.d/common-password</code>
pam_unix	<code>/etc/pam.d/common-password</code>

Tabla A3-3. Archivos de configuración de módulos PAM.

‘pam_cracklib’ analiza la calidad de las contraseñas realizando comprobaciones como que las contraseñas no sean similares a palabras de diccionarios, que tengan una determinada longitud o que no sea la misma que la anterior contraseña.

Para poder hacer comprobaciones hay que instalar el paquete “libpam-cracklib”.

Editar el fichero “`/etc/pam.d/common-password`” y modificar o añadir la línea:

```
password requisite pam_cracklib.so retry=3 minlen=12 lcredit=-1 ucredit=-1 dcredit=-1 ocredit=-1 difok=3
```

‘retry’ sirve para establecer el número de intentos fallidos de login que se permiten.

‘minlen’ para definir la complejidad de la contraseña. La complejidad es un sumatorio donde cada carácter en la contraseña aporta 1 punto y lcredit, ucredit, dcredit y ocredit aportan su valor en puntos. Por ejemplo:

- minlen=12; lcredit=-1; ucredit=-1; dcredit=-1; ocredit=-1
- Por tanto longitud de la contraseña = 12 -1-1-1-1 = 8 caracteres.

‘lcredit’ para definir la cantidad de letras minúsculas que deben ser incluidas.

‘ucredit’ para definir la cantidad de letras mayúsculas.

‘dcredit’ para definir el número de dígitos.

‘ocredit’ para definir la cantidad de caracteres especiales (otros caracteres que no son letras mayúsculas, minúsculas y números).

‘difok’ para definir el número de caracteres diferentes entre la nueva contraseña y la anterior.

Las opciones `ucredit`, `lcredit`, `dcredit` y `ocredit` pueden tener números negativos o positivos, por ejemplo:

- `lcredit=-3`: como mínimo debe tener 3 caracteres en minúsculas.
- `lcredit=+3`: como máximo debe tener 3 caracteres en minúsculas.

La configuración de la línea anterior implicaría que se usen contraseñas de ocho o más caracteres con al menos una minúscula, mayúscula, número y carácter especial. Además obliga a que al menos tres caracteres sean diferentes con respecto a la última contraseña y permite hasta tres intentos de accesos fallidos.

Para limitar la repetición de contraseñas se configura la opción `'remember'` del módulo `'pam_unix'` en el fichero `/etc/pam.d/common-password`, con el número de contraseñas anteriores que el usuario no puede reutilizar. Por ejemplo:

```
password [success=1 default=ignore] pam_unix.so obscure use_authtok try_first_pass sha512  
remember=5
```

aIII.3.8. Configuración de restricciones de acceso.

Hay que limitar en el sistema el acceso a la BIOS /UEFI, al gestor de arranque GRUB, a la consola, al sistema de ficheros y el acceso por red.

Hay que establecer condiciones de acceso para los usuarios, de manera que por ejemplo no se acceda al sistema como `root` (administrador) salvo para realizar tareas administrativas cuando sea estrictamente necesario, y también para otros usuarios (los normales) hay que establecer condiciones, por ejemplo para el uso de `'su'` (condiciones en `'sudoers'`).

Otras cuestiones también importantes son la configuración de tiempos de inactividad y las reacciones del sistema cuando sucede, los avisos del sistema al realizar `login`, etc.

Las medidas adecuadas serían al menos las que se explican en los siguientes párrafos.

Limitar (restringir) el acceso a la BIOS. Se debe fijar una contraseña para el acceso a la configuración de la BIOS/UEFI y así evitar cambios no autorizados. Algunas BIOS/UEFI permiten mayor granularidad en los permisos, pudiendo exigir contraseñas para diferentes cosas, por ejemplo para el arranque del sistema, lo cual puede limitar su funcionamiento. Estas opciones deberán ser particularmente consideradas.

Siempre hay que configurar el disco duro como primer dispositivo de arranque de la máquina, desactivando el arranque desde red, DVD o dispositivos USB, incluso conviene la desactivación o desinstalación de lectores, interfaces de red y USB que no sean necesarios.

Limitar (restringir) el acceso al GRUB. El gestor de arranque por defecto instalado en Ubuntu es GRUB, es el programa que inicia el kernel o núcleo y da paso a los parámetros necesarios hacia él. Permite elegir entre varios Kernels instalados.

Si en la máquina se ha creado el usuario `root`, GRUB solicitará la contraseña de `root` antes dar acceso a la máquina, pero si no se hubiera creado, se podría obtener acceso como `root` al sistema sin necesidad de ninguna autenticación.

GRUB debe ser protegido para evitar la modificación de su configuración sin permiso. Para eso seguimos los siguientes pasos:

1º Seleccionamos una contraseña y generamos su hash:

```
# grub-mkpasswd-pbkdf2
```

2º Editamos el fichero `/etc/grub.d/40_custom` b y le añadimos la siguiente línea:

```
set superusers="superroot" password_pbkdf2 superroot <hash generado>
```

3º Editamos el fichero `/etc/default/grub` y deshabilitaremos el terminal gráfico para editar el dicho fichero:

```
GRUB_TERMINAL=console
```

4º Actualizamos GRUB:

```
# update-grub
```

Configurar el modo ‘SINGLE-USER’. Uno de los parámetros que se pueden pasar al kernel durante el arranque es el modo ‘single-user’. Hay dos entradas en el menú de GRUB, una el arranque normal de la máquina y otra el arranque en modo “single-user”, que es un modo monousuario para administrar el sistema en casos excepcionales.

En Ubuntu, al acceder al modo ‘single-user’ se tiene acceso al sistema como usuario *root* sin autenticación, lo que se evita estableciendo una contraseña para *root*, previa escalada a privilegiado (# sudo su) con el comando ‘passwd’, tras lo cual además se habilitará el *login* directo del usuario *root*.

Para proteger el arranque del modo “single-user” desde GRUB, se editará el archivo `/etc/grub.d/10_linux` (que contiene su configuración). Y se sustituirá el siguiente código:

```
if ${recovery} ; then
title="$(gettext_quoted "%s, with Linux %s (%s)")"
printf "menuentry '${title}' ${CLASS} {\n" "${os}" "${version}" "$(gettext
"${GRUB_RECOVERY_TITLE}")"
else
title="$(gettext_quoted "%s, with Linux %s")"
printf "menuentry '${title}' ${CLASS} {\n" "${os}" "${version}"
fi
```

Por

```
if ${recovery} ; then
title="$(gettext_quoted "%s, with Linux %s (%s)")"
printf "menuentry '${title}' --users superroot ${CLASS} {\n" "${os}" "${version}" "$(gettext
"${GRUB_RECOVERY_TITLE}")"
else
title="$(gettext_quoted "%s, with Linux %s")"
printf "menuentry '${title}' ${CLASS} {\n" "${os}" "${version}"
fi
```

Donde el único cambio es incluir `--users superroot` que es el usuario que establecimos al limitar el acceso a GRUB.

Configurar las consolas para desconectar a los usuarios después de un periodo de inactividad. Editar el fichero `/etc/profile` y añadir al final:

```
TMOUT=300
readonly TMOUT
export TMOUT
```

Tras reiniciar, el sistema desconectará a los usuarios tras 300 segundos (5 minutos) de inactividad.

Configurar con un tiempo máximo de tiempo de espera las sesiones a través de una conexión remota, como por ejemplo por SSH.

Es parecido al punto anterior. Hay que plantearse esto para cada una de las maneras de acceder remotamente al sistema, concretamente para sesiones SSH se puede configurar editando el archivo `/etc/ssh/sshd_config` y estableciendo los siguientes valores:

```
ClientAliveInterval 500
ClientAliveCountMax 0
```

El funcionamiento es que con el primero establecemos el periodo de inactividad en segundos, transcurrido el cual si el usuario no interactúa con la *Shell* expirará su sesión; con el segundo se configura que el servidor ssh interroge al cliente remoto si está activo o no un número de veces antes de cerrar su sesión, por ejemplo si lo establecemos en 3, tras 500 segundos pregunta una vez, a los 1000 por segunda vez y a los 1500 directamente cierra sesión, por lo tanto con el valor 0 se cierra sesión sin preguntar.

Configurar el banner de la pantalla de login del sistema. Cuando se inicia una terminal en el sistema se muestra antes de la solicitud de login el contenido del fichero `/etc/issue` y el de `/etc/issue.net` o una combinación de ambos.

Estos contenidos deben modificarse para que por una parte no muestre información sensible como la versión del SO (no hay porque dar pistas) y al mismo tiempo para que muestre un aviso legal.

Editamos los archivos y los modificamos para que solo se emita el siguiente mensaje: ‘Este sistema contiene información que solo debe ser accedida por personal autorizado. El uso del sistema no autorizado será perseguido y las actividades en el sistema serán registradas’.

Desactivar peticiones de acciones de bajo nivel al sistema. Se trata de desactivar la tecla de petición de sistema (Magic SysRq key) la cual permite a los usuarios realizar acciones a través del usuario *system* como enviar una señal de finalización de procesos o de reinicio. Para desactivar esta opción editamos el archivo `/etc/sysctl.conf` y añadimos la línea siguiente:

```
kernel.sysrq = 0
```

Desactivar ‘ctrl+alt+supr’. Pulsando esta combinación se puede reiniciar la máquina sin identificarse en el sistema. Conviene desactivar esta opción, para lo cual en Ubuntu 18.04 deberemos ejecutar los siguientes dos comandos:

```
$ sudo systemctl mask ctrl-alt-del.target
$ sudo systemctl daemon-reload
```

Esto no desactiva por completo la combinación de teclas Ctrl-Alt-Del. Systemd tiene una función en la que al presionar esa combinación de teclas más de siete veces en dos segundos se inicia un reinicio inmediato. No he encontrado como impedir esto.

Limitaciones de login del usuario root y de usuarios con privilegios de administrador. Normalmente el usuario *root* no debería poder hacer *login* en el sistema, pues se pueden cometer errores como el borrado de archivos accidental, accesos no auditados, ejecución de código fuera de contexto seguro, etc.

Los administradores deben iniciar sesión como usuario no privilegiado y escalar, incluso a *root* cuando sea necesario, en función de la actividad que necesiten realizar. Para tareas administrativas normales escalarían privilegios mediante el comando ‘su’, para lo que su usuario deberá tener permisos de ejecución en este comando.

Tras la instalación de Ubuntu Server 18.04 LTS el usuario *root* estará deshabilitado por defecto (pero existe, está creado) y se habrá creado un usuario con privilegios de administración, por ejemplo el usuario ‘superadmin’. Consecuentemente hay que crear un usuario sin privilegios de administración para el administrador (o tantos como administradores), por ejemplo creamos ‘admin’:

```
$ sudo adduser admin
```

Lo que se hace a continuación es permitir a este usuario ‘admin’ la ejecución de ‘su’, que permite que ‘admin’ use los privilegios de otros usuarios tras facilitar la contraseña de los segundos, por ejemplo para que ‘admin’ use los privilegios de ‘superadmin’ introduciría:

\$ su superadmin

El sistema solicitará la contraseña de ‘superadmin’ y otorgará los privilegios a ‘admin’. Por supuesto es absolutamente imprescindible limitar que usuarios pueden escalar privilegios a *root*.

Para limitar esto creamos el grupo ‘wheel’ (si no lo está):

\$ sudo addgroup wheel

Añadimos al grupo “wheel” el usuario sin privilegios ‘admin’:

\$ sudo adduser admin wheel

Editamos el archivo /etc/pam.d/su y comprobamos que la directiva siguiente existe y no está comentada (si no estuviera la introducimos):

Auth required pam_wheel.so

Los usuarios que pertenezcan al grupo ‘wheel’ serán los que puedan escalar privilegios.

Es conveniente usar “sudo” para registrar las acciones del usuario *root*. Creamos el grupo ‘admin’ (si no lo está, en Ubuntu lo normal es que esté por defecto):

\$ sudo addgroup admin

Todos los usuarios administradores (que pueden ejecutar como *root*) deben pertenecer a este grupo, lo comprobamos listando los usuarios con sus grupos de pertenencia uno a uno y finalmente listando todos los usuarios del grupo. Si alguno no pertenece lo agregamos.

Con el comando ‘sudo’ se controla los usuarios que pueden ejecutar comandos usando otras cuentas y además permite la auditoría de los comandos ejecutados por los usuarios con privilegios.

Lo primero es instalar el paquete sudo si no estuviera, no es lo normal pues Ubuntu lo incluye por defecto.

Editamos el fichero /etc/sudoers, lo cual es peligroso pues puede ocasionar errores de configuración cuya consecuencia pudiera ser que usuarios malintencionados o sin permisos adecuados pudieran lograr acceso al sistema o con privilegios que no les corresponden:

visudo -s -f /etc/sudoers

Con ‘visudo’ se verifica antes de guardar el fichero si la sintaxis es correcta. Sudoers no debe editarse con otra utilidad. Comprobamos que en el fichero no aparece o se encuentra comentada la directiva:

NOPASSWD

Esta directiva permite ejecutar como administrador a cualquiera que conozca el usuario pero sin pedirle contraseña. Seguidamente añadimos la siguiente línea:

%admin ALL=(ALL) ALL

Por último, si un administrador accede remotamente al sistema debe hacerlo con un usuario diferente al que usa cuando accede sobre el mismo sistema. Para este usuario se debe establecer el mismo criterio de usar ‘su’ y además configurar las posibilidades con el comando ‘sudo’.

Desde luego nunca un usuario privilegiado debe poder acceder en remoto lo que exigirá configuraciones adicionales de SSH.

El usuario *root* solo debería poder hacer *login* en el contexto de una *Shell*, en una consola, para eso es necesario modificar el fichero /etc/securetty que el programa ‘login’ utiliza.

Cada terminal de consola es un dispositivo para el sistema que se encuentra a disposición en las rutas /dev/console y /dev/ttyX. A estos dispositivos se puede acceder con ‘ctrl-alt-F1’ hasta ‘ctrl-alt-F6’.

Hay que limitar el acceso de consola dejando únicamente las consolas `ttyX`, `vc/X`, y el puerto serie `ttySX` comentando en el fichero `/etc/securetty` todas las opciones que no se deseen.

Limitaciones de login y a la consola para cuentas *no-root* del sistema. Las cuentas del sistema son las que el SO utiliza para diversas tareas y no se asocian a ninguna persona. De estas cuentas hay algunas que necesitan ciertos permisos privilegiados (*root*) pero hay otras que no los necesitan (*no-root*). Para limitar sus posibilidades editamos el archivo `/etc/passwd` y obtenemos todos los nombres de usuario, sus `UIDs` y las shells asignadas, con el comando `awk`:

```
# awk -F: '{print $1 ":" $3 ":" $7}' /etc/passwd
```

Las cuentas con un `UID` (`$3`) menor a 500 son del sistema. Para cada una de ellas y adicionalmente para las cuentas de usuario que no se les permita acceder al sistema ejecutar los siguientes comandos, el primero para bloquearla y el segundo para deshabilitar su *Shell*.

```
# usermod -L <cuenta>
# usermod -s /usr/sbin/nologin <cuenta>
```

Nunca se debe modificar la cuenta de *root* pues el sistema se podría volver inaccesible.

aIII.3.9. Configuración de restricciones de acceso al sistema de ficheros

El sistema de ficheros es un nivel al que se pueden establecer restricciones de acceso mediante el sistema de permisos de los objetos de Linux. Se pueden establecer los permisos tanto a los grupos como a los usuarios. Como mínimo un usuario debe pertenecer a un grupo.

Cada fichero tiene un conjunto de permisos, lectura, escritura y ejecución, configurados para el usuario propietario del mismo, para el grupo propietario del fichero y para los demás usuarios del sistema.

Objeto a objeto hay que establecer una asignación de permisos adecuada.

aIII.3.10. Configuración de restricciones de acceso por red a nivel de aplicación

En Ubuntu para filtrar conexiones a servicios además del cortafuego (`netfilter`) se dispone de `TCP Wrapper`. Es más eficiente el cortafuego a la hora de filtrar conexiones, pero eso no quita que se configure bien `TCP Wrapper`.

`TCP Wrapper` es una librería para que las aplicaciones compatibles dispongan de un control de acceso y registro (*logging*).

`TCP Wrapper` soporta servicios que utilizan la librería “`libwrap`”. Podemos ver cuáles son esos servicios tecleando:

```
$ apt-cache rdepends libwrap0
```

Uno de los servicios típicos es por ejemplo `sshd`.

`TCP Wrapper` funciona con dos archivos de configuración: `/etc/hosts.allow` y `/etc/hosts.deny`. Primero busca en el fichero `/etc/hosts.allow` una regla que cumpla con la conexión. Si la encuentra, se permite la conexión, si no la encuentra, busca en el fichero `/etc/hosts.deny` una regla que cumpla con la conexión. Si la encuentra, se bloquea la conexión. Si no hay ninguna regla para esa conexión, entonces la conexión se permite (permite por defecto).

Las reglas simples guardan la sintaxis: ‘`demonio: cliente`’. `Demonio`: es el nombre del servicio al que la conexión va destinado. `Cliente`: es el nombre de *host* o dirección IP del cliente. Tanto el ‘`demonio`’ como el ‘`cliente`’ pueden contener uno o varios valores separados por coma.

Lo primero es ver cuáles son los servicios públicos y si usan la librería “`libwrap`”, probemos con `sshd` (para que funcione debemos tener instalado el paquete ‘`binutils`’):

```
# strings /usr/sbin/sshd | grep libwrap
```

Que arroja el resultado: ‘libwrap.so.0’ y ‘libwrap refuse returns’, luego tiene dependencias hacia la librería y soporta TCP Wrapper, (no es el caso de apache2, por ejemplo).

También podríamos haber usado ‘ldd’. Lo mismo ocurre con el servicio FTP:

```
$ ldd $(which sshd) | grep libwrap
```

Resultado: Libwrap.so.0 => /lib/i386-linux-gnu/libwrap.so.0 (0xb766e000)

```
$ ldd $(which vsftpd) | grep libwrap
```

Resultado: Libwrap.so.0 => /lib/i386-linux-gnu/libwrap.so.0 (0xb7706000)

Como ejemplo editaremos los archivos /etc/hosts.allow y /etc/hosts.deny y añadiremos las reglas necesarias para permitir el acceso SSH y FTP solamente desde la dirección 192.168.1.150 y desde la máquina local ‘localhost’ y denegar cualquier otra. En el archivo ‘/etc/hosts.deny’ añadimos:

```
sshd,vsftpd : ALL
ALL : ALL
```

... y en ‘/etc/hosts.allow’:

```
sshd,vsftpd : 192.168.1.150,LOCAL
```

No hace falta reiniciar nada. Como regla general, por tanto:

En ‘/etc/hosts.allow’

```
serviciopublico1 , serviciopublico2 : ALL
servicioprivado : .dominio.local
ALL: localhost
```

... y en ‘/etc/hosts.deny’

```
ALL: ALL
```

Estas reglas rechazan conexiones a todos los servicios TCP Wrapper activos desde cualquier otro host que no sea localhost, acepta las conexiones a los servicios “serviciopublico1” y “serviciopublico2” desde cualquier lugar, y acepta las conexiones al servicio “servicioprivado” sólo desde el dominio local.

Para terminar, las conexiones rechazadas y errores se monitorizan mediante el registro que TCP Wrapper realiza por defecto en el fichero ‘/var/log/auth.log’. Conviene comprobar que esto es así para lo cual debe estar en el fichero de configuración ‘/etc/rsyslog.d/50-default.conf’ la línea siguiente:

```
auth,authpriv.* /var/log/auth.log
```

aIII.4. Reducción de la superficie de ataque del sistema.

Cuando instalamos un Ubuntu Server 18.04 LTS la superficie de ataque es bastante reducida de por sí, esto quiere decir que solo encontraremos activos en el sistema los servicios imprescindibles, pero no obstante el diseñador del sistema para el cual hemos instalado el servidor debe plantearse cuales son realmente necesarios y cuáles no. Todos los que no tengan una necesidad explícitamente justificada deben desactivarse, pasándolos a un estado detenido o incluso desinstalándolos del servidor.

Listamos todos los servicios y su estatus:

```
# service --status-all
```

La figura A3-4 muestra un pantallazo de un servidor que alberga a un nodo de control de OpenStack:

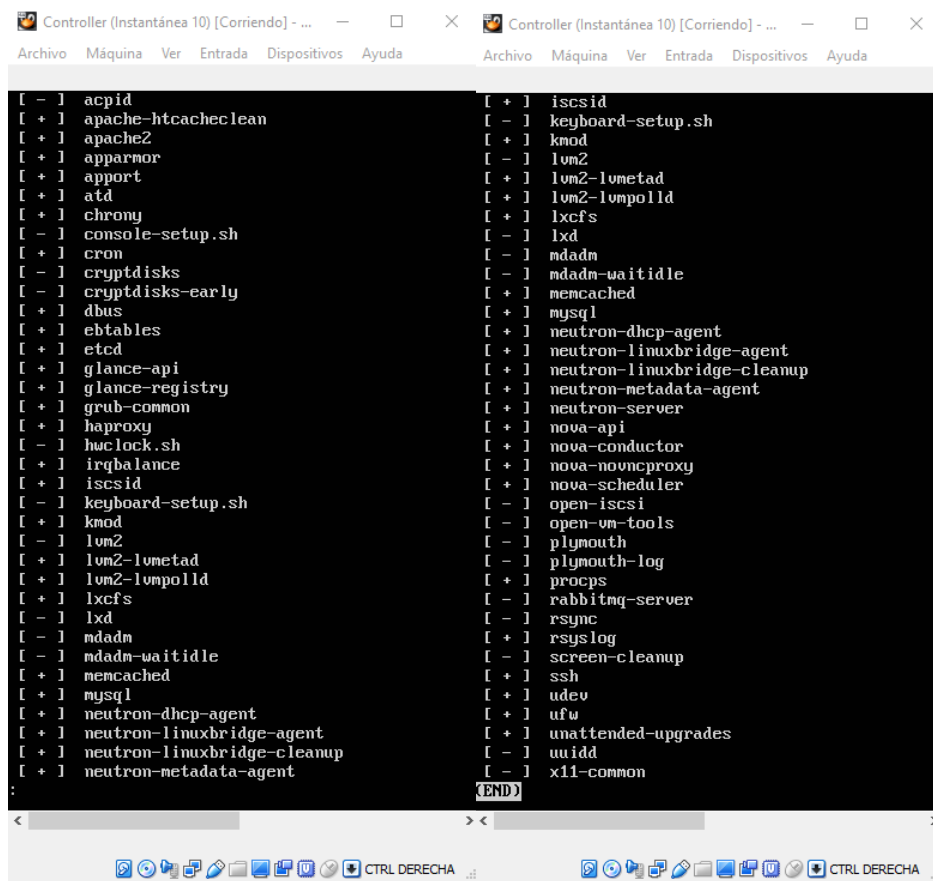


Figura A3-4. Servicios del nodo de control.

Para deshabilitar un servicio:

```
# update-rc.d -f <nombre_servicio> remove
```

En la tabla A3-4 se muestra un listado de servicios habilitados en los sistemas Ubuntu, donde se indica el nombre de servicio, su descripción y una acción sobre si dejarlo activo o no:

Nombre del servicio	Descripción	Acción
acpid	Configuración avanzada e interfaz de energía. Se utiliza para gestión de la energía.	Deshabilitar.
cron	Ejecuta trabajos programados	Dejar activo.
dns-clean	Limpia la información de DNS en conexiones por red telefónica (dial-up).	Deshabilitar.
rc.local	Inicio de los scripts locales	Dejar activo.
grub-common	Muestra información en el menú de GRUB.	Dejar activo.
ondemand	Configura el gestor de escala de frecuencia de CPU debajo demanda.	Dejar activo.
pppd-dns	Limpia el fichero /resolv.conf cuando el sistema falla.	Deshabilitar.
rmnologin	Elimina el fichero /etc/nologin al inicio.	Dejar activo.
rsync	Permite la copia de ficheros a o desde máquinas remotas.	Deshabilitar.
rsyslog	Permite el envío de mensajes.	Dejar activo.

ssh	Servidor de SSH.	Deshabilitar en caso de tener acceso físico.
stopbootlogd	Para el demonio de registro de <i>logs</i> enviados a la consola.	Dejar activo.
watchdog	Inicia el demonio de “watchdog”.	Deshabilitar.
wd_keepalive	Inicia el demonio “keepalive” de “watchdog”.	Deshabilitar.

Tabla A3-4. Servicios en Ubuntu [44]

aIII.5. Limitaciones en permisos de carpetas y archivos

El sistema se puede ver comprometido como consecuencia de ciertos permisos unidos a errores en software o de administración. Para limitar estos efectos en lo posible es necesario:

aIII.5.1. Propiedad de los archivos ‘passwd’, ‘shadow’, ‘group’ y ‘gshadow’

Hay que asegurarse que el propietario de los archivos ‘passwd’, ‘shadow’, ‘group’ y ‘gshadow’ es root:

```
# cd /etc
# chown root:root passwd shadow group gshadow
```

aIII.5.2. Permisos de los archivos ‘passwd’ y ‘group’

Solo el propietario puede leer y escribir en estos archivos, y el resto solo leer, por tanto hay que modificar los permisos de los archivos ‘passwd’ y ‘group’ a 644 (rw- r----):

```
# chmod 644 passwd group
```

aIII.5.3. Permisos de los archivos ‘shadow’ y ‘gshadow’

Solo el propietario puede leer en ellos, y el resto nada, por tanto hay que modificar los permisos de los archivos ‘shadow’ y ‘gshadow’ a 400 (r-- --- ---):

```
# chmod 400 shadow gshadow
```

aIII.5.4. Directorios con permisos de escritura

Verificar que ninguno de los directorios con permisos de escritura tiene el bit ‘sticky’ activado. Para localizar los directorios:

```
# find puntodemontaje -xdev -type d \( -perm -0002 -a ! -perm -1000 \) -print
```

Dónde puntodemontaje se refiere al de una partición. Si el comando anterior muestra cualquier información, corregir el permiso del directorio con el comando:

```
# chmod +t /directorio
```

aIII.5.5. Permisos de escritura universal

El sistema no debería contener ficheros con permisos de escritura universales, por tanto hay que modificarlo en los ficheros que lo tengan:

```
# find puntodemontaje -xdev -type f -perm -0002 -print
```

Si se muestra cualquier información, corregir el permiso del directorio con:

```
# chmod o-w file
```

aIII.5.6. Permisos SUID/SGID activados en ficheros ejecutables

Hay ficheros ejecutables con permisos SUID/SGID activados, lo que quiere decir que se ejecutan con los permisos del propietario del fichero o grupo en vez de con los permisos del usuario o grupo que lo ejecuta. Estos suelen ser ficheros propiedad de *root*, con lo cual siempre se ejecutan con los máximos permisos, lo cual puede hacer que si un atacante infiltre código en uno de los procesos abiertos por estos ejecutables automáticamente logre permisos de *root*. Para localizarlos, igual que antes, encada partición:

```
# find puntodemontaje -xdev \( -perm -4000 -o -perm -2000 \) -type f -print
```

La tabla A3-5 muestra los ficheros que normalmente tienen permisos SUID/SGID activados:

Fichero	SUID/SGID	Perteneciente	¿Deshabilitar?
/usr/lib/openssh/ssh-keysign	uid root	SSH	Sí
/usr/sbin/exim4	uid root	Correo	Sí
/bin/mount	uid root	Sistema	No
/bin/umount	uid root	Sistema	No
/bin/ping	uid root	Red	No
/usr/bin/su	uid root	Autenticación	No
/sbin/unix_chkpwd	uid root	Autenticación PAM	No
/usr/bin/chage	uid root	Expiración de contraseñas	Sí
/usr/bin/chsh	uid root	Información de usuario	Si
/usr/bin/crontab	uid / gid root	Cron	Sí
/usr/bin/gpasswd	uid root	Autenticación	No
/usr/bin/locate	gid slocate	Base de datos local	No
/usr/bin/newgrp	uid root	Autenticación	No
/usr/bin/passwd	uid root	Autenticación	No
/usr/bin/rcp	uid root	RSH	Sí
/usr/bin/rlogin	uid root	RSH	Sí
/usr/bin/rsh	uid root	RSH	Sí
/usr/bin/wall	gid tty	Mensajes de consola	Sí
/usr/bin/write	gid tty	Mensajes de consola	Sí

Tabla A3-5. Ficheros con permisos suid/sgid activados [44]

Para deshabilitar el permiso SUID/SGID ejecutar:

```
# chmod -s file
```

aIII.5.7. Reparar ficheros huérfanos

Encontrar y arreglar ficheros sin propietario o grupo válido:

```
# find punto de montaje -xdev \( -nouser -o -nogroup \) -print
```

De cada fichero encontrado, debe verificarse si es o no necesario y en caso de no poder eliminarse asignarle un propietario o grupo.

aIII.5.8. Directorios con permisos de escritura globales sin un propietario

Demasiadas veces tendemos a arreglar problemas de permisos de un objeto asignando permisos a todos, pero eso hace que un atacante o un incauto llene un directorio hasta ocupar toda la partición (si no tiene cuotas de disco asignadas) lo que podría ocasionar un desbordamiento. También pueden darse situaciones en las que un usuario borre información de otro:

```
# find puntodemontaje -xdev -type d -perm -0002 -uid +1000 -print
```

Encontraremos los directorios escribibles por cualquiera y comprobaremos que su propietario es *root* o un usuario de sistema, pues son los usuarios con el UID por debajo de 1000. Cualquier resultado, se deberá analizar por qué el propietario no es *root* u otro usuario de sistema y cambiarlo en su caso.

aIII.6. Limitaciones en ejecutables del sistema

Los problemas asociados a los ejecutables son la fuga de información debido a la ejecución de binarios y los desbordamientos de búfer.

Normalmente cuando se ejecuta un comando del sistema, estos se encuentran mediante una variable de entorno, \$PATH. Esta variable contiene las rutas donde están los ejecutables que son varias, normalmente: /bin, /sbin, /usr/bin, /usr/sbin, etc.

Es muy necesario listar el contenido de la variable \$PATH y ver que no contiene ninguna ruta absoluta como un '.' o una cadena vacía, pues en ellas un hacker podría alojar un ejecutable malicioso, por ejemplo con un nombre común para nosotros, como 'ls' o 'cat'.

También es muy conveniente deshabilitar 'core dumps'. Cuando un ejecutable es finalizado por causa de un error el SO toma una imagen de la memoria y la vuelca en un fichero *core dump*, a menos que estemos desarrollando algún *software* esto no es necesario, y estos ficheros pueden contener información sensible u ocupar mucho espacio en el disco.

Si se recuerda ya desactivamos esta opción para el proceso de instalación del sistema Ubuntu 18.04.

Para deshabilitar los *core dumps* a todos los usuarios editamos el archivo /etc/security/limits.conf y le añadimos:

```
/etc/security/limits.conf * hard core 0
```

Para asegurar que los *core dumps* nunca se realicen para aplicaciones con el bit SUID activado, editamos el fichero /etc/sysctl.conf y añadimos:

```
/etc/sysctl.conf fs.suid_dumpable = 0
```

A continuación conviene establecer protecciones contra el desbordamiento de buffer. Para esto no hay una receta única y hay que estudiar cómo se comporta el SO si es de 32 o de 64 bits con relación a su versión y la arquitectura del procesador sobre el que corre el sistema.

Desde el So puede obtenerse información del sistema, concretamente del procesador, y ver si soportan funcionalidades como PAE o XD o NX:

```
# cat /proc/cpuinfo
```

Si están soportadas hay paquetes como linux-image-generic-pae que se pueden instalar para establecer estas protecciones.

Algunas BIOS/EFI/UEFI tienen una opción para activar o desactivar esta funcionalidad.

Otra de las protecciones es ‘usar direcciones de pila de forma aleatoria’, que simplificando consiste en que los ejecutables no se inician en la misma dirección de la pila cada vez, puede descolocar a algún atacante, pero esto ya es muy fácil de salvar por los *hackers*. Para esto editamos el archivo `/etc/sysctl.conf` y activamos o incluimos la línea:

```
/etc/sysctl.conf kernel.randomize_va_space = 1
```

aIII.7. Securizar las cuentas existentes en el sistema

aIII.7.1. Lo primero es eliminar las cuentas innecesarias

La instalación del sistema previene la creación de cuentas de usuario para servicios que posiblemente no se instalen nunca. Esas cuentas deben eliminarse.

La tabla A3-6 muestra algunas de estas cuentas que hay que plantearse si se pueden o no eliminar:

Usuario	Descripción
proxy	Usuario asociado a algunos servidores Proxy (como squid).
www-data	Usuario asociado a servidores web (como Apache).
list	Usuario asociado al servicio de listas de distribución de correo electrónico.
irc	Usuario asociado a algunos servidores de IRC.
gnats	Usuario asociado al servicio de control de bugs (errores) del sistema operativo GNU.
uucp	Usuario asociado al protocolo UUCP (Unix to Unix CoPy).
news	Usuario asociado al servicio de news (Usenet News).
lp	Usuario asociado al servicio de impresión.

Tabla A3-6. Usuarios asociados a servicios [44]

aIII.7.2. No debe permitirse ningún usuario sin contraseña

Lo comprobamos, la orden no debería tener ninguna salida:

```
# awk -F: '($2 == "") {print}' /etc/shadow
```

Los usuarios que incumplan deben ser bloqueados para no hacer *login* ni acceder a *Shell* alguna.

aIII.7.3. Hay que comprobar que nadie más que root tiene UID ‘0’

Las cuentas de usuario se almacenan en `/etc/passwd`. Este fichero está en formato ASCII y puede leerlo cualquiera. Esto es así porque las aplicaciones del sistema lo precisan. El valor más importante del fichero `/etc/passwd` es el UID. El UID ‘0’ es el de *root* y no está sometido al sistema de permisos (en kernels no parcheados). Ese usuario podrá realizar cualquier acción sobre el sistema sin ningún tipo restricción. Es de extrema importancia que *root* sea usado solo en ocasiones imprescindibles, ya lo hemos explicado.

```
# awk -F: '($3 == "0") {print}' /etc/passwd
```

Sólo debe devolver un resultado, el del usuario *root*, todo otro UID debe ser modificado, y el usuario bloqueado hasta realizar las investigaciones pertinentes.

aIII.7.4. Creación de un grupo único a cada usuario

Cada nueva cuenta que se crea se hace corresponder con la creación por defecto de un grupo único primario con mismo nombre que el de la cuenta. Esto es para conseguir mayor protección contra ficheros que son creados con grupos que tienen el permiso de escritura activado. Cuando se ejecute el comando “*useradd*” no se debe especificar el parámetro ‘-g’ para no sobrescribir el grupo por defecto.

aIII.7.5. Securizar ficheros de configuración de sesión para cuentas de usuario

Esto sobre todo hay que hacerlo para las cuentas *root* y de los administradores del sistema.

La mayoría de los ficheros de configuración se encuentran en el directorio del usuario correspondiente, y son los responsables de la configuración de la sesión que abre en el sistema.

Es importante que los directorios ‘home’ de los usuarios no tengan permisos de escritura para miembros del grupo o permisos de lectura para todos los usuarios. Es decir no deben compartirse sin más estos directorios. Podemos comprobar por cada usuario los permisos de su directorio /home:

```
# ls -ld /home/<USUARIO>
```

Si es necesario se retirarían los permisos de escritura o de lectura a todos los usuarios:

```
# chmod g-w /home/<USUARIO>
# chmod o-rwx /home/<USUARIO>
```

También hay que ver los permisos de los ficheros ocultos:

```
#ls -la /home/<USUARIO>
```

Para los ficheros ocultos que no estén bien configurados, los configuramos:

```
# chmod go-w /home/<USUARIO>/<FICHERO>
```

Cuando los usuarios crean los ficheros en sus directorios la asignación de los permisos se hace conforme al valor ‘umask’, el cual debe establecerse que sea por defecto 077:

Para cada usuario hay que editar el archivo .profile (donde ya se encuentra la directiva umask comentada) y .bashrc (donde probablemente no estará), que está en su /home y modificar o establecer el valor de umask por 077.

El primer archivo es para las sesiones en líneas de comando (normalmente Bash), y el segundo para el entorno gráfico. No tendría sentido cambiarlo solo en un entorno y no en el otro.

En ambos archivos, donde se quiera, pero lo mejor es al final o al principio del todo se añade la directiva umask valor:

```
UMASK=077
```

Tras reiniciar la sesión del usuario el valor surte efecto. Si se crean archivos con ‘sudo’ el entorno es el de la sesión y no debe esperarse que se aplique el umask de *root*.

Para que el cambio afecte a todos los usuarios se cambia el valor en el archivo /etc/profile.

aIII.7.6. Eliminar el historial de acciones de los usuarios

El archivo ‘.bash_history’ en el *home* del usuario almacena todas las acciones realizadas. Para eliminar el historial se editan los archivos ‘.bashrc’ del usuario y ‘/etc/profile’ la siguiente directiva:

```
shred ~/.bash_history
```

Con esto con cada reinicio se eliminará el historial.

aIII.8. Evitar el montaje de dispositivos usb de almacenamiento masivo

El sistema nunca deberá montar de manera automática dispositivos extraíbles tales como CDs/DVDs o dispositivos o memorias extraíbles. Es necesario deshabilitar el módulo del kernel para dispositivos USB de modo que no sea posible montar ninguna memoria USB.

Para ello hay que editar el fichero /etc/modprobe.d/blacklist.conf y añadir al final la siguiente línea:

```
blacklist usb-storage
```

aIII.9. Securización de parámetros de red

aIII.9.1. Fortalecimiento de la pila TCP/IP

Para establecer los parámetros que afectan al modo de operar del kernel de Linux se utiliza ‘sysctl’. Para diferentes configuraciones editamos el archivo /etc/sysctl.conf y añadimos o modificamos los parámetros necesarios.

En los servidores (equipos individuales) que no actúan como *routers*, *firewalls* o *gateways*:

```
net.ipv4.ip_forward = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
```

En equipos individuales o en equipos que funcionan como router:

```
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
net.ipv4.icmp_echo_ignore_broadcasts = 1
net.ipv4.icmp_ignore_bogus_error_messages = 1
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
```

aIII.9.1. Deshabilitar IPv6

Ubuntu soporta IPv6. Solo si ciertas características de nuestra nube privada se van a usar sería procedente mantener este protocolo activo.

Para deshabilitarlo, en el fichero sysctl.conf añadiremos:

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

aIII.10. Configuración segura de servicios

Toda la configuración de cualquier servicio debe plantearse desde el punto de vista de la seguridad, desde los más simples incluidos en el SO, los generales como Rabbit, Memcache, Apache2, hasta los más complejos de OpenStack.

Los servicios innecesarios ya han sido deshabilitados, a continuación ofreceremos indicaciones de como robustecer algunos de los que quedan. Como ya se dijo al principio del anexo III solo se trata en éste de documentar cómo poder realizar las acciones de robustecimiento del SO Ubuntu en su versión 18.04 pues la guía STIC 612 [44], en la que nos estamos basando, está desactualizada (se refiere a la versión 12.04). El robustecimiento de servicios como los de OpenStack o Apache2 debe confiarse a la aplicación de otras guías que expliquen las buenas prácticas y que ya se han referido en el primer apartado de este anexo III y a lo largo del TFM.

aIII.10.1. Servicio SSH

En Ubuntu el nombre del paquete del servidor de SSH se llama ‘openssh-server’. Tener un servidor de SSH ejecutándose en el sistema abre una puerta a los *hackers*. En consecuencia no se debe instalar a

menos que se tenga que administrar remotamente el sistema, o que ciertos servicios usen conexiones por SSH para comunicaciones entre sistemas de un nodo. Nunca se debe usar la versión 1 del protocolo SSH, hay que utilizar la 2

La primera medida es limitar que direcciones IPs pueden acceder a dicho servicio mediante el cortafuego. La segunda medida consiste en configurar de forma segura el servidor SSH editando el fichero `/etc/ssh/sshd_config`. Veamos las directivas:

Revisar que el protocolo SSH usado por OpenSSH es la versión 2. Ver la línea del fichero `/etc/ssh/sshd_config`:

```
/etc/ssh/sshd_config Protocol 2
```

Para dar acceso sólo a los usuarios permitidos, añadir lo siguiente:

```
AllowUsers USUARIO1 USUARIO2
```

Establecer una mayor longitud de clave:

```
ServerkeyBits 4096
```

Deshabilitar la emulación RSH:

```
IgnoreRhosts yes
```

Deshabilitar la autenticación basada en hosts:

```
HostbasedAuthentication no
```

Desactivar la posibilidad de acceder al sistema con el usuario root por SSH para evitar que se realicen intentos de fuerza bruta a dicha cuenta:

```
PermitRootLogin no
```

Deshabilitar el acceso a usuarios con contraseñas vacías:

```
PermitEmptyPasswords no
```

Deshabilitar la autenticación mediante usuario y contraseña y habilitar la identificación mediante clave pública/privada:

```
PubkeyAuthentication yes
PasswordAuthentication no
UsePAM no
```

La generación de las claves se realizará siempre en el sistema cliente, y se copiará el contenido de la clave pública al fichero `“.ssh/authorized-keys”` del directorio personal del usuario con el que se autenticará:

```
# ssh-keygen
```

Se solicitará el nombre del fichero y la *passphrase* hasta dos veces.

Se deberá cambiar el puerto por defecto del servidor SSH por otro que no sea ‘bien conocido’ (el número es un ejemplo):

```
Port 100001
```

También debe configurarse un tiempo máximo de tiempo de espera las sesiones a través de una conexión remota, lo cual ya se explicó con anterioridad en el apartado aIII.3.8. Configuración de restricciones de acceso.

aIII.10.2. Servicio CROND

Cualquier usuario no debe tener el privilegio de programar tareas, por lo que se deberá denegar a todos los usuarios el uso de crontab a excepción de a los administradores, por ejemplo el usuario root. Esta configuración se realiza editando los archivos `/etc/cron.allow` y `/etc/cron.deny` (si no están creados pueden crearse).

aIII.10.3. NTP

Además de la necesidad de sincronismo que impone OpenStack todos los sistemas del clúster (en general todos los de la organización) deben sincronizarse horariamente. Esto es importante para la gestión centralizada de logs y para la correlación de eventos cuando se está investigando algún error, anomalía o intrusión en los ficheros de registro.

Ya se ha explicado como instalar y configurar Crony y mantener todos los sistemas sincronizados en el capítulo 3 de este TFM.

aIII.11. Servicios de seguridad específicos

Un sistema no estará totalmente bastionado si no se monitoriza, si no se protege detrás de un IDS/IPS, si no se le instala algún *software* antivirus (capaz contra *rootkits* y *webshells*), sin política y utilidad para *backups*, borrado seguro, cuotas de disco, y mucho más. Todo depende del grado de paranoia que se establezca en nuestra organización y plasme su política.

No vamos a entrar en todas estas medidas de seguridad que no son una configuración propiamente dicha del SO, pero a modo de indicación se enumerarán cuáles son las posibles soluciones en la tabla A3-7 a continuación:

Medida	Implementación
Monitorización del sistema	Se deberá configurar la monitorización del sistema mediante la configuración de <code>'rsyslog'</code> (ya incluido en Ubuntu). Con las herramientas <code>"logcheck"</code> y <code>"logwatch"</code>, que hay que instalar y configurar se monitorizan de forma más amigables los logs recopilados.
Sistema IDS	Si un ataque progresa y alguien consigue acceder al sistema, es vital detectar dicha intrusión y evaluar correctamente los daños causados. Para esto se emplean sistemas de detección de intrusos (IDS), tanto a nivel de red como a nivel de host. Dentro de los IDS de <i>host</i> se encuentran los comprobadores de integridad de ficheros. Su misión es detectar cualquier modificación, por pequeña que sea, en los directorios y ficheros del sistema ya sea realizada por parte de un intruso o por el administrador del sistema. Hay dos posibles herramientas muy útiles: Tripwire es un IDS que realiza comprobaciones de integridad a los ficheros y directorios del sistema. OSSEC es otro IDS de host. Realiza análisis de logs, comprobaciones de integridad del sistema, detección de rootkits y puede actuar reactivamente como IPS. Por último una adecuada configuración de Snort e Iptables puede ser el mejor y más potente IDS.

Antivirus	Donde más lógica resulta su instalación es en los servidores de ficheros y correo, pues los antivirus detectan mayoritariamente malware cuyo objetivo son sistemas Windows. Clamav es un escáner de virus disponible en sistemas basados en Debian y que puede utilizarse con los servidores de correo y de ficheros más comunes.
Detección de <i>rootkits</i>	Un rootkit tiene como finalidad obtener el control del sistema sin que sea descubierta escondiéndose a sí misma y escondiendo otros elementos del sistema como programas, procesos, y ficheros. Hay varias posibilidades como son Rkhunter, Unhide y Chkrootkit. La última es la aconsejada para Ubuntu. Permiten acciones como comprobar la presencia anomalías en archivos y carpetas por comparación de resúmenes md5 y anomalías, o localizar procesos y puertos TCP/UDP ocultos por <i>rootkits</i>, por el <i>kernel</i> de Linux o mediante otras técnicas.
Detección de <i>webshells</i>	Por ejemplo: PHP Web Shell Detector, NeoPI y Linux Malware Detector.
<i>Backups</i>	La realización de copias de seguridad es esencial para cualquier sistema. Deben hacerse de acuerdo con la política de seguridad de la organización Amanda es una herramienta para facilitar las tareas de creación y recuperación de copias de seguridad en sistemas Unix, tanto copias de seguridad incrementales como cifradas [106].
Borrado seguro	Puede usarse la utilidad ‘Shred’ que en una ejecución por defecto sobrescribe el fichero o dispositivo indicado 25 veces. El comando shred está incluido en el paquete ‘coreutils’, que no hay que instalar.
Cuotas de disco	Las cuotas de disco se usan para limitar el espacio de los diferentes usuarios. Se pueden materializar instalando y configurando el paquete ‘Quota’.
Control de acceso obligatorio MAC, (Mandatory Access Control)	SELinux es una utilidad para realizar control de acceso obligatorio MAC incluida en el kernel de Linux. SELinux añade una capa de permisos basados en roles que define el acceso y derechos de interacción de cada usuario, aplicación, proceso y fichero del sistema, que son los cuatro aspectos de la política de accesos que permite establecer. AppArmor es otra implementación MAC, que en su caso usa el módulo LSM (<i>Linux Security Module</i>). AppArmor es una alternativa a SELinux más fácil de configurar y mantener que protege de forma proactiva al sistema operativo y sus aplicaciones de amenazas, y ataques. A diferencia de SELinux que se basa en etiquetar el sistema de ficheros, AppArmor trabaja con rutas de ficheros. Las políticas de AppArmor definen que aplicaciones pueden acceder a los distintos recursos del sistema y con qué privilegios. AppArmor tiene varias políticas por defecto, y un modo de aprendizaje.

Tabla A3-7. Soluciones para otras medidas de seguridad.