Institut Puig Castellar
Santa Coloma de Gramenet

Comprensió d' OpenStack

CFGS Desenvolupament de Sistemes Informàtics en Xarxa

Autor:
Daniel Pérez Palacino
dperez@elpuig.xeill.net

Curs:
2ºn ASIX

Data d'entrega:
30/05/2017

Resum del projecte:

Comprensió d'OpenStack (CO) és un projecte que neix amb la idea de comprendre el funcionament de la solució de cloud computing OpenStack de codi obert.

Actualment OpenStack es utilitzada per diverses empreses del sector informàtic per diversos motius però la majoria de gent desconeix el funcionament i que és exactament a més de com utilitzar-ho, perquè molts diuen que es molt difícil de ser utilitzat per un usuari qualsevol (degut a la documentació, per exemple, que es molt tècnica i a la complexitat del programa) , i això es el que vull aconseguir, demostrar que OpenStack pot ser utilitzat per un usuari amb un mínim de coneixements informàtics i proporcionar una guia en català per a futurs estudiants d'aquesta matèria.

Abstract:

Understanding OpenStack (CO) is a project that was created with the idea to understand the operation of the solution OpenStack open source cloud computing.

Currently OpenStack is used by several companies in the computer industry for several reasons but most people do not know the operation and that is exactly of how to use it, because many say is very difficult to be used by any user (due to documentation, for example, which is highly technical and complex program), and this is what i want, demonstrate that OpenStack can be used by a user with minimal computer skills and provide guidance in Catalan for future students this matter.

Paraules clau/ Key words:

OpenStack, Màquina virtual, KVM, Virtualització, Cloud.

Índex

1. Introducció

1.1 Context i justificació del Projecte

A l'escola Puig Castellar s'utilitzen contínuament màquines virtuals tant per treballar a l'aula com a la gestió del servidor del propi centre:

-S'utilitzen màquines virtuals a l'aula per tal de posar en pràctica els coneixements adquirits sense posar en perill el maquinari.

-S'utilitzen màquines virtuals al servidor del centre per tal de gestionar diferents serveis (moodle, owncloud, web del centre...).

Amb la finalitat de agilitzar la gestió de les màquines virtuals neix un projecte per gestionar de manera eficaç y eficient les diferents virtualitzacions, tant màquines com contenidors.

1.2 Objectius del Projecte

- Comprendre que és OpenStack y tots els conceptes que l'envolten.
- Aprendre a assimilar la informació aportada en les diferents documentacions:
  - Documentació oficial.
  - Llibres.
  - Articles.
- Aprendre a utilitzar OpenStack de manera eficient.
- Aprendre a temporitzar el temps disponible de manera productiva i eficaç.
- Explicar d'una manera comprensible tot el coneixement adquirit per a que futurs estudiants puguin fer us d'aquesta eina.

1.3 Enfocament i mètode seguit

- Enfoco aquest projecte cap al camí de la superació per tal d'aconseguir resultats satisfactoris en camps que no hem treballat a classe.

- Partint dels coneixements que tinc sobre les àrees en què es desenvolupa el nostre projecte he triat l'opció de crear un projecte nou, orientat a qualsevol usuari amb motivacions d'aprendre sobre aquesta eina i que tingui un mínim de coneixements per tal d'aconseguir-ho.

1.4 Planificació del projecte

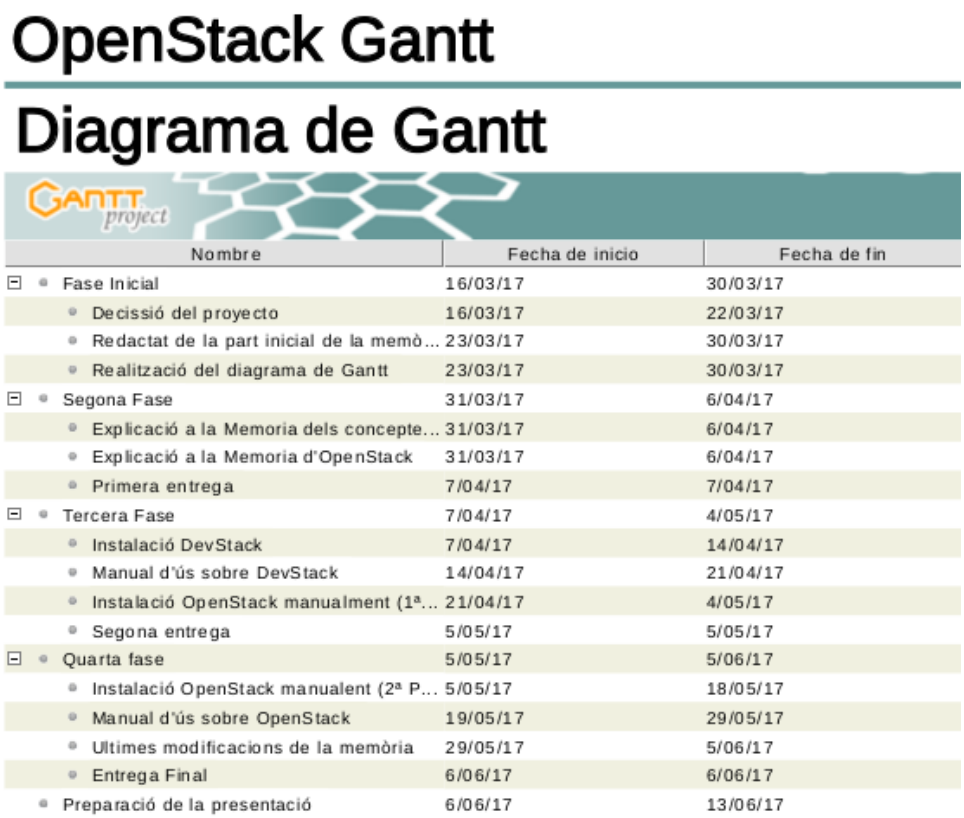
1.4.1. Tecnologia utilitzada

El projecte es desenvolupa utilitzant diverses màquines virtuals amb Ubuntu 16.04 Server les quals utilitzem per desplegar l'eina OpenStack.

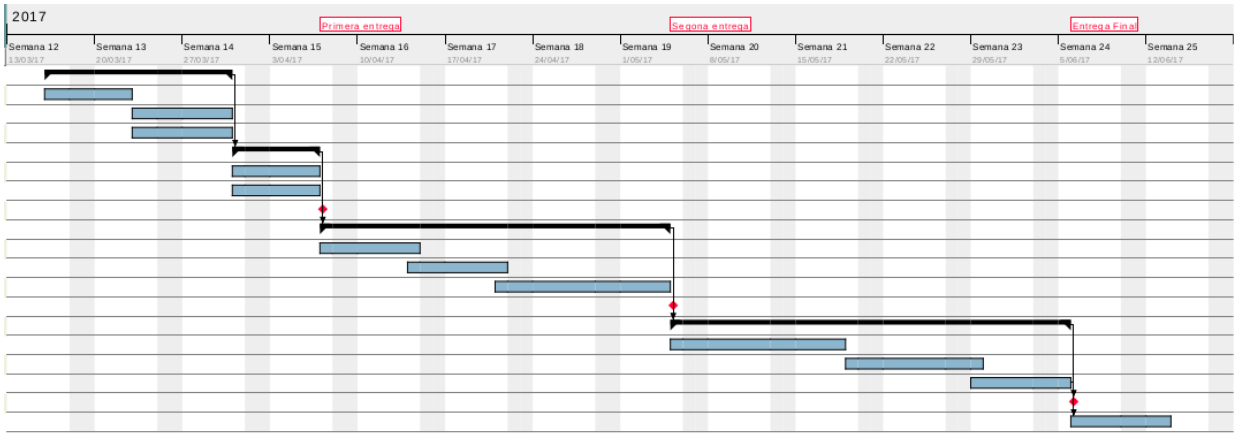
Màquina Virtual (KVM, VirtualBox). Per mitjà d'aquesta aplicació és possible instal·lar sistemes operatius addicionals, amb servidors, cadascú amb el seu propi ambient virtual.

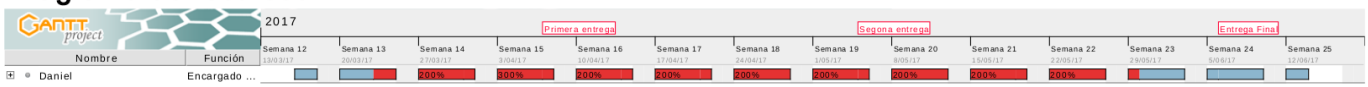
GanttProject. Per mitjà d'aquesta aplicació es possible gestionar de manera eficient el temps que es empleat en el projecte mitjançant unes gràfiques generades a mida de l'usuari.

1.4.2 Diagrama de GANTT.



Nombre	Fecha de inicio	Fecha de fin
Fase Inicial	16/03/17	30/03/17
Decissió del proyecto	16/03/17	22/03/17
Redactat de la part inicial de la memò...	23/03/17	30/03/17
Realització del diagrama de Gantt	23/03/17	30/03/17
Segona Fase	31/03/17	6/04/17
Explicació a la Memoria dels concepte...	31/03/17	6/04/17
Explicació a la Memoria d'OpenStack	31/03/17	6/04/17
Primera entrega	7/04/17	7/04/17
Tercera Fase	7/04/17	4/05/17
Instalació DevStack	7/04/17	14/04/17
Manual d'ús sobre DevStack	14/04/17	21/04/17
Instalació OpenStack manualment (1ª...	21/04/17	4/05/17
Segona entrega	5/05/17	5/05/17
Quarta fase	5/05/17	5/06/17
Instalació OpenStack manualent (2ª P...	5/05/17	18/05/17
Manual d'ús sobre OpenStack	19/05/17	29/05/17
Ultimes modificacions de la memòria	29/05/17	5/06/17
Entrega Final	6/06/17	6/06/17
Preparació de la presentació	6/06/17	13/06/17





Tarea

Nombre	Fecha de inicio	Fecha de fin
Fase Inicial	16/03/17	30/03/17
Decissió del proyecto	16/03/17	22/03/17
Redactat de la part inicial de la memòria	23/03/17	30/03/17
Realització del diagrama de Gantt	23/03/17	30/03/17
Segona Fase	31/03/17	6/04/17
Explicació a la Memoria dels conceptes bàsics	31/03/17	6/04/17
Explicació a la Memoria d'OpenStack	31/03/17	6/04/17
Primera entrega	7/04/17	7/04/17
Tercera Fase	7/04/17	4/05/17
Instalació DevStack	7/04/17	14/04/17
Manual d'ús sobre DevStack	14/04/17	21/04/17
Instalació OpenStack manualment (1ª Part)	21/04/17	4/05/17
Segona entrega	5/05/17	5/05/17
Quarta fase	5/05/17	5/06/17
Instalació OpenStack manualent (2ª Part)	5/05/17	18/05/17
Manual d'ús sobre OpenStack	19/05/17	29/05/17
Ultimes modificacions de la memòria	29/05/17	5/06/17
Entrega Final	6/06/17	6/06/17
Preparació de la presentació	6/06/17	13/06/17

1.5 Breu sumari del producte obtingut

El producte final compleix amb el que es contemplava en iniciar el projecte. He adquirit coneixements d'OpenStack tant a nivell teòric com a nivell pràctic. A més, gràcies a aquest projecte potser algun dia el centre disposi d'aquesta tecnologia per a proporcionar clouds i màquines virtuals als alumnes per a poder treballar des de fora de casa o per fer diferents pràctiques entre d'altres possibilitats.

Aquest projecte es el primer orientat a OpenStack en català vist que en tota la duració no s'ha trobat cap documentació/projecte en aquest idioma i en un futur es podria utilitzar com a referent per aprendre aquesta tecnologia a les aules.

1.6 Breu descripció dels següents capítols

En els pròxims capítols podem trobar:

- **Manual de l'usuari:** Una completa guia d'ús per als usuaris que volen aprendre com funciona OpenStack des de cero. Començant per conceptes bàsics per poder finalitzar en l'explicació completa, mòdul a mòdul, de l'eina OpenStack.
- **Gestió d'errors:** En aquest apartat documento amb detall els errors més greus que he trobat al llarg de la realització del projecte.
- **Conclusions:** En aquest apartat documento les conclusions a les que he arribat un cop he finalitzat el projecte
- **Webgrafia:** En aquest apartat documento les direccions web que més he utilitzat i que més m'han servit per realitzar el projecte.
- **Bibliografia:** En aquest apartat documento els diferents llibres que he utilitzat per a realitzar el projecte.
- **Annexos:** En aquest apartat documento els diferents materials que annexo al projecte, com per exemple les màquines virtuals creades.

2. Manual de l'usuari

En aquest capítol aprendrem a utilitzar OpenStack pas per pas. Repassarem els conceptes bàsics per a tenir una òptima comprensió d'aquesta eina. Ho dividirem en diferents capítols. Comencem:

2.1 Introducció

Al llarg dels anys, la computació corporativa ha vist molts desenvolupaments, que eventualment han portat el sorgiment de la computació al nuvol com la coneixem:

- A la dècada de 1990, la computació corporativa es va centrar en en els servidors endins de datacenters.
- A la dècada de 2000, la computació corporativa es va centrar en la virtualització.
- A la dècada de 2010 estem observant un augment significatiu de la computació al nuvol per aprofitar la computació corporativa.

El concepte de “cloud computing” es molt ampli, fins al punt de que podem afirmar que no existeix tal cosa com a “cloud computing”. Si demanem a un usuari final que expliqui què és i a continuació, preguntem a un administrador de sistemes la mateixa pregunta, obtindrem dos descripcions diferents. En general ni ha tres tipus de enfoc que ens podria definir que és aquest concepte:

- **IaaS**(**Infrastructure as a Service**)
- **PaaS (Platform as a Service)**
- **SaaS (Software as a Service)**

OpenStack pertany a la categoria de computació en el cloud **IaaS** però està evolucionant contínuament, ampliant el seu abast. En algunes ocasions, l'enfoc d'OpenStack va mes enllà de **IaaS**.

Per a poder conèixer OpenStack, primer hem d'adquirir uns conceptes bàsics que veurem a continuació.

2.2 Conceptes bàsics

2.2.1 ¿Què es la virtualització?

Quan parlem sobre virtualització, molta gent pensa en màquines virtuals. Una màquina virtual es un ordinador sobre el que el seu software i el seu hardware son creats mitjançant una solució software.

El propòsit de la virtualització es proporcionar una versió virtual, en lloc de física, dels següents recursos essencials:

- Hardware: Aquest es l'ús tradicional, on el sistema operatiu està instal·lat sobre una representació de software dels recursos del hardware.

- Emmagatzematge: També conegut com a «Software Defined Storage» (SDS). Es crea una capa entre els discs durs reals i els equips que accedeixen a aquests discs amb l'objectiu de fer-los més accessibles.

- Xarxa: També conegut com a «Software Defined Networking» (SDN). Els usuaris del nuvol poden crear una infraestructura de xarxa lògica sobre la xarxa física, el que fa que sigui més fàcil acomodar les seves necessitats.

Tots aquests tipus de virtualització s'utilitzen en OpenStack.

2.2.2 ¿Com funciona la virtualització?

La solució més comú utilitzada per arrencar màquines virtuals en un datacenter amb Linux es KVM. KVM es una virtualització basada en hipervisor, el que implica que la màquina virtual funciona a sobre d'un nucli de virtualització petit i altament optimitzat, que permet a la màquina virtual abordar el hardware de la manera més eficient.

Una altre solució de virtualització comú es VirtualBox. VirtualBox permet als desenvolupadors executar un sistema operatiu virtual a sobre del seu escriptori. No obstant, aquesta solució no es tan eficient com la virtualització basada en hipervisor.

El cloud IaaS (Infrastructure as a Service) està directament relacionada amb la virtualització basada en hipervisor.

2.2.3 ¿Que son els contenidors?

Els contenidors son un mètode de virtualització a nivell de sistema operatiu. Això permet generar múltiples instàncies de un sistema operatiu executant-se al mateix nucli alhora de manera que es fa un ús més eficient dels recursos.

Els contenidors van néixer de la necessitat dels usuaris d'executar una aplicació sense tenir que arrencar una màquina virtual per poder utilitzar-la.

A més, l'ús de contenidors facilita el treball als desenvolupadors ja que només s'ha de centrar en crear l'aplicació sense tenir en compte la plataforma a la que s'executarà. Les tecnologies de contenidors, com Docker, eliminen aquesta necessitat d'haver de tenir en compte les diferents característiques de les plataformes en les que s'executaran les aplicacions.

2.2.4 ¿Com funcionen els contenidors?

El punt de partida d'un contenidor es una imatge que contindrà tot el necessari per executar un contenidor:

- Aplicació.
- Dependències d'espai de l'usuari.
- Llibreries

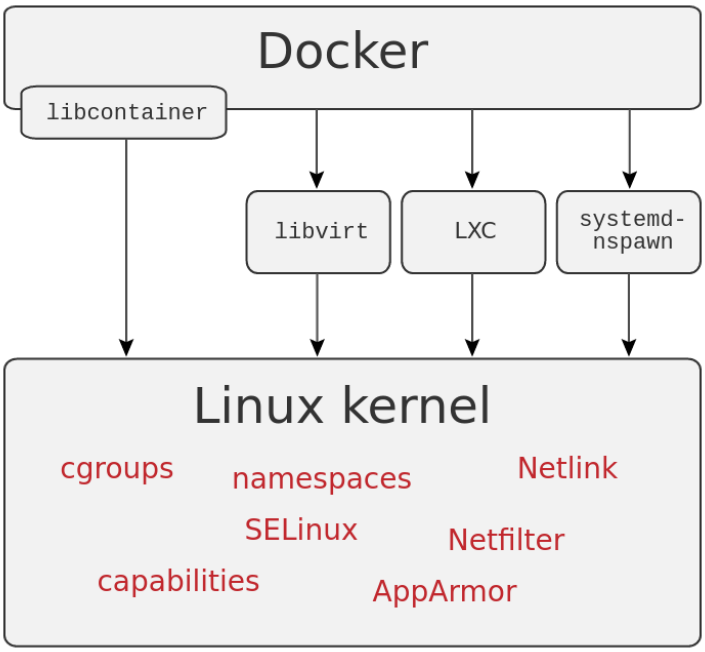
Els components de l'espai del kernel estan proporcionats pel sistema operatiu del host. Quan el contenidor es crea a partir d'una imatge, executa els seus processos en el kernel del host aprofitant els seus recursos.

2.2.5 ¿De quins components estan formats?

Cada contenidor en funcionament te tres components essencials:

- **Namespaces** (Espais de noms): Son recursos globals del sistema com la xarxa, PID, muntatges... de manera que el procés pensa que es l'únic que accedeix al recurs.
- **Cgroups:** S'utilitzen para reservar i assignar recursos aïllats als contenidors.
- **Union file system:** El sistema de fitxers fusiona diferents sistemes de fitxers en un sol, virtual.

Un dels contenidors més utilitzats s'anomena Docker. Com podem observar a la següent figura, el kernel de Linux està proporcionant diferents característiques (que ja hem vist abans) que son utilitzades per l'entorn de contenidors de Docker.



2.2.6 ¿Quins son els beneficis de utilitzar contenidors?

Els contenidors s'estan afermant com a la tecnologia de virtualització més eficient. Es poden implementar moltes instàncies alhora d'una aplicació per funcionar a la part superior d'un sol nucli. Aquest nucli proporciona entorns aïllats, i per tant disposen d'una alta seguretat.

Al mateix cop, això fa que s'executa un procés que garanteix la seguretat dels contenidors:

- Un usuari en un contenidor no pot accedir als recursos que s'assignen a un altre contenidor.

A causa d'això, molts contenidors es poden utilitzar a la part superior del mateix nucli aprofitant els recursos d'una manera més eficient. A més, el desplegament es molt més fàcil a causa de la mida relativament petit dels contenidors.

2.2.7 ¿Quines son les diferencies entre contenidors i màquines virtuals?

No n'hi ha una tecnologia millor que l'altre en aquest sentit, sinó que s'utilitzen segons els diferents contextos en els que treballem:

- Utilitzem contenidors si volem executar múltiples instàncies d'una sola aplicació. Per exemple, si volem tenir a un host real, es crearien diferents instàncies d'un servidor web per a proporcionar diferents serveis mitjançant contenidors.

- Utilitzem màquines virtuals si necessitem la flexibilitat d'executar múltiples aplicacions. Per exemple si per alguna casualitat estiguéssim treballant a alguna distribució de sistema operatiu privativa com per exemple Windows i necessitéssim utilitzar una serie de programari específic de Debian, llavors podríem crear una màquina virtual amb aquesta distribució i aquest programari per tal de no haver de fer-ho al nostre host real.

- Utilitzem màquines virtuals si hem de ser capaços de funcionar en qualsevol sistema operatiu. Per exemple, molts desenvolupadors crean programari específic per a un nucli i sistema operatiu i per tal de que pugui funcionar en totes les distribucions crean una màquina virtual que proporcionen amb el programari ja instal·lat.

A més de diferencies, també tenen una característica en comú que es la que a nosaltres ens interessa de cara a aquest projecte. Resulta que tots dos, contenidors i màquines virtuals, poden funcionar junts en un cloud IaaS.

2.2.8 ¿Quines limitacions tenen els contenidors?

Els contenidors, tot i que ofereixen grans beneficis en la seva utilització, també tenen limitacions:

- Per defecte, els contenidors no son multiplataforma. El contingut del contenidor depèn del nucli al que està lligat.

- Poden ser requerides algunes dependències per a que un contenidor s'executi a una plataforma específica.

- L'aïllament dels contenidors es pot considerar feble, ja que tots s'executen al mateix nucli. Si el nucli cau, tots els contenidors que s'estan executant a la part superior d'ell, cauran també.

- Hi ha massa potencial d'expansió. Com que es relativament fàcil implementar un nou contenidor, correm el risc d'acabar en un ambient on s'han desplegat un munt de contenidors on ningú sap que es el que s'està executant realment.

2.2.9 ¿Què es el Cloud Computing?

El Cloud Computing (computació en el núvol) es una computació basada en Internet que proporciona recursos de processament compartits i dades als ordinadors i altres dispositius.

Es permet l'accés sota demanda a un conjunt compartit de recursos informàtics, com ara xarxes, servidors, emmagatzematge, aplicacions i serveis, que normalment s'allotgen en els datacenters de tercers.

2.2.10 ¿Què entén l'usuari final com a Cloud Computing?

El Cloud Computing es una metàfora. Per a un usuari, els elements de xarxa que representen els serveis dels proveïdors son invisibles, com si estigués enfosquit per un nuvol.

La conclusió és que, des de la perspectiva de l'usuari, no importa realment on s'estan executant els recursos, sinó que estiguin disponibles per a ser utilitzats.

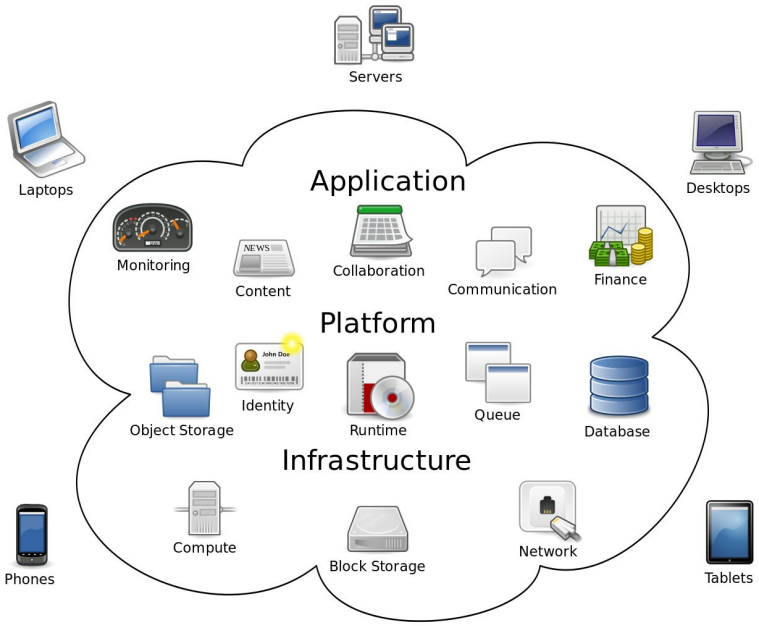
2.2.11 ¿Quins tipus diferents de Cloud Computing existeixen?

Depenent dels tipus de recursos, es poden distingir tres tipus de Cloud Computing.

- **IaaS**(**Infrastructure as a Service**): El proveïdor subministra una infraestructura (datacenters i xarxes) que es utilitzada per a proporcionar màquines virtuals i contenidors.

- **PaaS (Platform as a Service):** El proveïdor subministra la xarxa, servidors, emmagatzematge, sistema operatiu i el middleware (lògica d'intercanvi d'informació entre aplicacions) para allotjar una aplicació al Cloud.

- **SaaS (Software as a Service):** El proveïdor da accés a una aplicació dins del Cloud.



2.2.12 ¿Quins son els beneficis d'utilitzar Cloud Computing?

El cloud computing fa que sigui una eina molt més flexibles tant pels usuaris com pels administradors. Alguns dels seus beneficis son:

- Proporcionar un fàcil accés als actius (recursos) essencials. Això fa que sigui el més fàcil possible per als usuaris poder utilitzar aquests actius.
- Permet l'auto-desplegament. Això fa que els usuaris no necessitin realment un administrador per tal d'implementar un nou servidor.
- Proporciona escalabilitat. Quan s'executa fora dels recursos físics es més fàcil afegir nous recursos.

2.2.13 ¿Quins models de serveis ofereix IaaS?

El Cloud IaaS pot oferir-se de diferents maneres utilitzant diferents models de servei:

- **Cloud privat:** La creació d'una infraestructura de IaaS que es per a ús privat únicament, per part, per exemple d'una empresa/entitat.
- **Cloud públic**: La capacitat del Cloud es oferida com a servei de productes bàsics, com és el cas de la telefonia que ofereix un proveïdor de telecomunicacions. Els clients contracten una part de la infraestructura pública Cloud.
- **Cloud hibrid**: Aquest tipus de servei IaaS és un model que consisteix en unir els components públics i privats de IaaS.

2.2.14 Ús pràctic de Cloud Computing IaaS

Un usuari final accedeix a la web del Cloud per a crear una màquina virtual. Mentre que es fa la creació, l'usuari final pren decisions sobre xarxes, emmagatzematge i seguretat.

El Cloud IaaS ofereix una plataforma flexible per a fer funcionar contenidors i màquines virtuals d'una manera flexible:

- S'ofereix un portal/web d'autoservei.
- Es proporciona emmagatzematge i xarxes escalables.

2.2.15 ¿Es el Computing Cloud IaaS per a tothom?

La computació en el núvol IaaS funciona millor per a entorns que necessiten solucions flexibles. Tenir el seu propi Cloud IaaS no fa l'administració més fàcil. Això es deu al fet que amb la finalitat d'establir un gran Cloud IaaS, especialistes en molts temes diferents han de treballar junts per crear una solució eficient.

Si l'usuari no necessita escalabilitat i auto-servei no es necessari utilitzar aquesta tecnologia perquè es pot servir utilitzant virtualitzacions normals.

2.2.16 ¿Per què es OpenStack tan important quan parlem de Computing Cloud IaaS?

OpenStack es basa en estàndards oberts i té el suport dels principals actors de serveis Cloud.

Un dels elements principals de OpenStack es l'API oberta, que és un conjunt de definicions de rutines, protocols i eines per a la construcció de programari i aplicacions. A causa de que està tan ben desenvolupada l'API de OpenStack, genera una atracció cap als desenvolupadors que fan que aquesta eina sigui més fàcil d'utilitzar que altres tipus de solucions.

2.3 OpenStack

2.3.1 Els orígens d'OpenStack

OpenStack es va iniciar el 2010, com un projecte conjunt de Rackspace Hosting i la NASA:

- La NASA va contribuir amb la seva plataforma Nebula, que més tard es va convertir en Nova.
- Rackspace va contribuir amb la seva plataforma d'arxius cloud que més endavant es va convertir en Swift.

A l'abril de 2011, l'alliberament de *OpenStack Bexar* es va introduir en Ubuntu. Més tard aquest mateix any, Debian inclou *OpenStack Cactus* en la seva distribució. El 2012, Red Hat va anunciar un avançament de la seva distribució d' OpenStack. Des de llavors, molts altres van seguir el mateix camí, incloent Oracle, HP i Vmware.

2.3.2 La fundació OpenStack

La OpenStack Fundation promou el desenvolupament global, la distribució i l'adopció del sistema operatiu al núvol.

Proporciona recursos compartits per fer créixer el núvol OpenStack a més de permetre als proveïdors de tecnologia i desenvolupadors poder ajudar en la producció de programari en el núvol.

La fundació OpenStack disposa d'uns principis que es poden resumir en:

- Licencia Apache 2.0
- Procés de disseny obert
- Repositoris públics del codi font
- Tots els processos de desenvolupament han d'estar documentats i ser transparents
- Ha de ser orientat per adoptar estàndards oberts
- Disseny modular que permet flexibilitat mitjançant l'ús d'APIS.

2.3.3 ¿Què és OpenStack?

Es una solució de Cloud Computing del tipus laaS de codi obert.

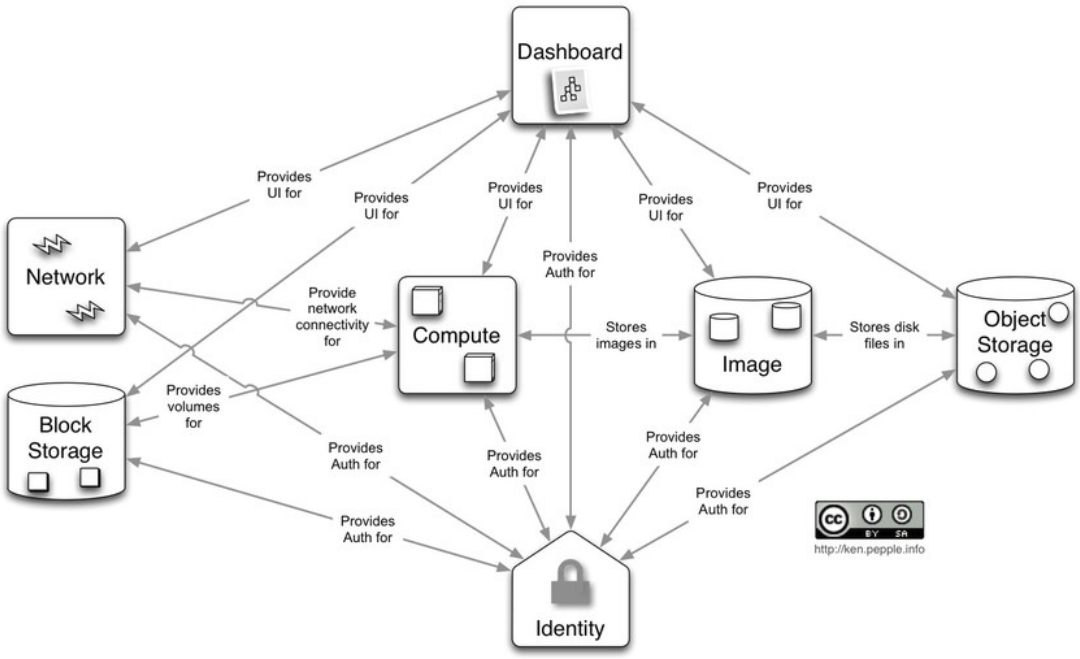
La seva missió es proveir una solució flexible tant per núvols públiques com privades siguin de la mida que siguin de manera que han de ser simples de implementar i massivament escalables.

Per complir amb aquests principis, OpenStack està dividit en diferents components que treballen en conjunt. Aquesta integració s'aconsegueix a través de les API's anteriorment anomenades que cada servei proporciona i consumeix.

Gracies a aquestes API's, els serveis poden comunicar-se entre ells i a més es possibilita que un servei sigui reemplaçat per un altre de característiques similars sempre que es respecti la forma de comunicació. Es a dir, OpenStack es extensible i s'ajusta a les necessitats de qui ho vulgui implementar.

2.3.4 ¿Quina estructura te OpenStack?

Al següent diagrama (creat per Ken Pepple) es pot veure la arquitectura d'OpenStack simplificada, assumint que s'utilitzen tots els serveis en la configuració més estàndard i sense il·lustrar com els clients del núvol poden interactuar amb ella.



Els diferents components que hem de conèixer de OpenStack per la seva importància son:

- **Ceilometer:** Ens ofereix serveis de telemetria amb els que podem monitoritzar l'ús de cada usuari a la nostra infraestructura.

- **Cinder:** Aquest mòdul es centra en el emmagatzematge de la forma més tradicional en que coneixem la paraula. Ens facilita l'accés a les dades allotjades als discs que es troben al núvol.

- **Glance:** Amb Glance disposarem d'un servei de gestió d'imatges que seran copies integres de les unitats de disc dur que tenim a la nostra disposició.

- **Heat:** Aquest mòdul ens permet emmagatzemar els requeriments d'una aplicació que proveïm des de el nostre núvol, en un fitxer que defineix els recursos necessaris per a dita aplicació. Ens ajuda a establir els requisits d'una infraestructura en funció de les aplicacions que allotgem en aquesta.

- **Horizon:** Aquest mòdul ens mostra mitjançant una interfície gràfica amb tota la gestió d'OpenStack, des de el que podrem veure que està passant al nostre núvol i gestionar qualsevol incidència que aparegui.

- **Keystone**: Aquest mòdul controlarà la identificació dels diferents usuaris que realitzen una connexió a la nostra infraestructura, i l'accés a segons que aplicacions o serveis.

- **Neutron:** Aquest mòdul s'encarrega de que cada mòdul de OpenStack es pugui comunicar amb altres i que estiguin interrelacionats.

- **Nova:** Aquest mòdul es considerat com el «motor» d'OpenStack. Es utilitzat per desplegar i administrar la quantitat de màquines virtuals i altres serveis que necessitem.

- **Swift:** Aquest mòdul encarregat d'emmagatzemar els fitxers del sistema, assegurar la seva integritat i replicar-los pels diferents discs que te la infraestructura, per a que aquests, sempre estiguin disponibles i accessibles de la forma més ràpida possible.

2.3.5 ¿Quines versions d'OpenStack existeixen?

Actualment existeixen moltes versions d'OpenStack des de el seu començament fins ara. A més, n'hi han algunes que estan en desenvolupament i altres que ja estan finalitzades. En el nostre projecte, intentarem utilitzar l'ultima versió finalitzada: Newton.

Series	Status	Initial Release Date	Next Phase	EOL Date
Queens	*Future*	TBD		TBD
Pike	Under Development	TBD		TBD
Ocata	Phase I – Latest release	2017-02-22	Phase II – Maintained release on 2017-08-28	2018-02-26
Newton	Phase II – Maintained release	2016-10-06	Phase III – Legacy release on 2017-08-28	2017-10-11
Mitaka	Phase III – Legacy release	2016-04-07	End of Life	2017-04-10
Liberty	EOL	2015-10-15		2016-11-17
Kilo	EOL	2015-04-30		2016-05-02
Juno	EOL	2014-10-16		2015-12-07
Icehouse	EOL	2014-04-17		2015-07-02
Havana	EOL	2013-10-17		2014-09-30
Grizzly	EOL	2013-04-04		2014-03-29
Folsom	EOL	2012-09-27		2013-11-19
Essex	EOL	2012-04-05		2013-05-06
Diablo	EOL	2011-09-22		2013-05-06
Cactus	Deprecated	2011-04-15		
Bexar	Deprecated	2011-02-03		
Austin	Deprecated	2010-10-21		

2.4 Formes de desplegament d' OpenStack

OpenStack es pot implementar mitjançant tres formes molt diferents:

- **Script:** Es pot implementar amb un Script com PackStack o DevStack
- **Manual:** Es pot implementar de forma manual seguint els passos descrits a la documentació oficial (http://docs.openstack.org/)
- **Desplegament a gran escala:** Es pot implementar utilitzant solucions d'implementació més avançades, com ara Triple O, director i altres...

2.4.1 ¿Què és DevStack?

DevStack es la solució estàndard per a desplegar OpenStack mitjançant un Script que pot ser utilitzat en múltiples distribucions de Linux (Ubuntu, Fedora, OpenSuse, Debian, CentOS/RedHat...).

Aquesta solució proporciona una serie d'Scripts per a crear ràpidament un entorn complert d'OpenStack basada en l'actual estat de desenvolupament en arbre.

DevStack es molt popular entre els desenvolupadors ja que s'utilitza per al desenvolupament i les proves de funcionament/rendiment. Es molt important destacar que aquesta solució no s'ha creat per a ser utilitzat en la producció.

2.4.2 ¿Quines possibilitats ens ofereix DevStack per ser utilitzat?

DevStack permet diferents escenaris en els que es pot utilitzar:

- **OpenStack en una sola màquina virtual:** Es l'escenari que utilitzarem en aquest projecte degut a les limitacions existents.
- **OpenStack en una sola màquina real:** Es molt similar a l'escenari anterior però utilitzant una màquina real en lloc d'una màquina virtual.
- **OpenStack amb múltiples nodes:** Es l'escenari en el qual es poden utilitzar múltiples màquines virtuals o màquines reals.

2.5 Desplegant OpenStack Mitjançant Script (DevStack)

Per desplegar OpenStack versió Newton (És l'última versió totalment acabada) podem trobar la documentació a http://docs.openstack.org/developer/devstack/.

Els passos a seguir son els següents:

1. Tenir una màquina virtual de l'actual Ubuntu Server LTS (16.04) amb la configuració per defecte dins d'un KVM amb la tarja de xarxa per defecte (NAT). A més, la màquina ha de tenir com a mínim 4Gib de Ram i a ser possible més d'un processador assignat.

2. Un cop la màquina està iniciada, tenim que crear un usuari només per a l'ús d'OpenStack. DevStack s'ha d'executar com a un usuari no root amb el sudo habilitat. Per això, creem l'usuari “stack”, li donem aquests privilegis i entrem com a aquest usuari nou.

```
usuari@ubuntu:~$  sudo -s
root@ubuntu:~$  useradd -s /bin/bash -d /opt/stack -m stack
root@ubuntu:~$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee
/etc/sudoers.d/stack
root@ubuntu:~$ sudo su – stack
```

3. Actualitzem els paquets i el nostre Ubuntu i instal·lem l'administrador de paquets de Python “pip”.

```
stack@ubuntu:~$  sudo -s
root@ubuntu:~$  apt -y update
root@ubuntu:~$  apt -y upgrade
root@ubuntu:~$  apt install -y git vim python-pip
```

Durant la actualització, se'ns demanarà el lloc per instal·lar el Grub. Escollim el mateix lloc en el que està instal·lat, en el meu cas, /dev/sda.

4. Actualitzem i instal·lem els següents paquets de Python amb privilegis de superuser. Es possible que se'ns demani escollir alguna opció. Escollim la opció per defecte.

```
root@ubuntu:~$  pip install --upgrade setuptools
root@ubuntu:~$  apt-get install python-dev libffi-dev libssl-dev
root@ubuntu:~$  pip install --upgrade pip
root@ubuntu:~$  pip install --upgrade urllib3
root@ubuntu:~$  pip install --upgrade urllib3[secure]
root@ubuntu:~$  apt-get install -y python-jpype python-dev
root@ubuntu:~$  pip install --upgrade apptools appdirs
root@ubuntu:~$  exit
```

5. Descarreguem OpenStack versió Mitaka de github (en el directori personal del usuari stack:

```
stack@ubuntu:~$ git clone https://git.openstack.org/openstack-dev/devstack -b
stable/newton
```

6. Anem al directori descarregat anomenat devstack i creem el fitxer de configuració amb l'editor preferit amb el següent contingut (canviant el valor password per la contrasenya que volem nosaltres) :

```
stack@ubuntu:~/devstack# cd devstack
stack@ubuntu:~/devstack# vim local.conf

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

7. Dins del mateix directori, obrim el fitxer **stack.sh** amb el nostre editor, i afegim la línia **FORCE=yes** a sobre de la primera línia que no està comentada ja que com podem comprovar a la següent captura, l'script no funcionarà sense aquesta opció (A la versió Newton ve per defecte, a les anteriors no).

```
# override check and attempt installation with ``FORCE=yes ./stack``
if [[ ! ${DISTRO} =~ (trusty|vivid|wily|7.0|wheezy|sid|testing|jessie|f22|f23|rh
el7|kvmibm1) ]]; then
    echo "WARNING: this script has not been tested on $DISTRO"
    if [[ "$FORCE" != "yes" ]]; then
        die $LINENO "If you wish to run this script anyway run with FORCE=yes"
    fi
fi
```

8. Un cop hem arribat aquí, el que tenim que fer es executar l'script de manera que siguem un usuari amb privilegis de superuser però sense ser root i sense utilitzar la comanda sudo:

```
stack@ubuntu:~/devstack#  ./stack.sh
```

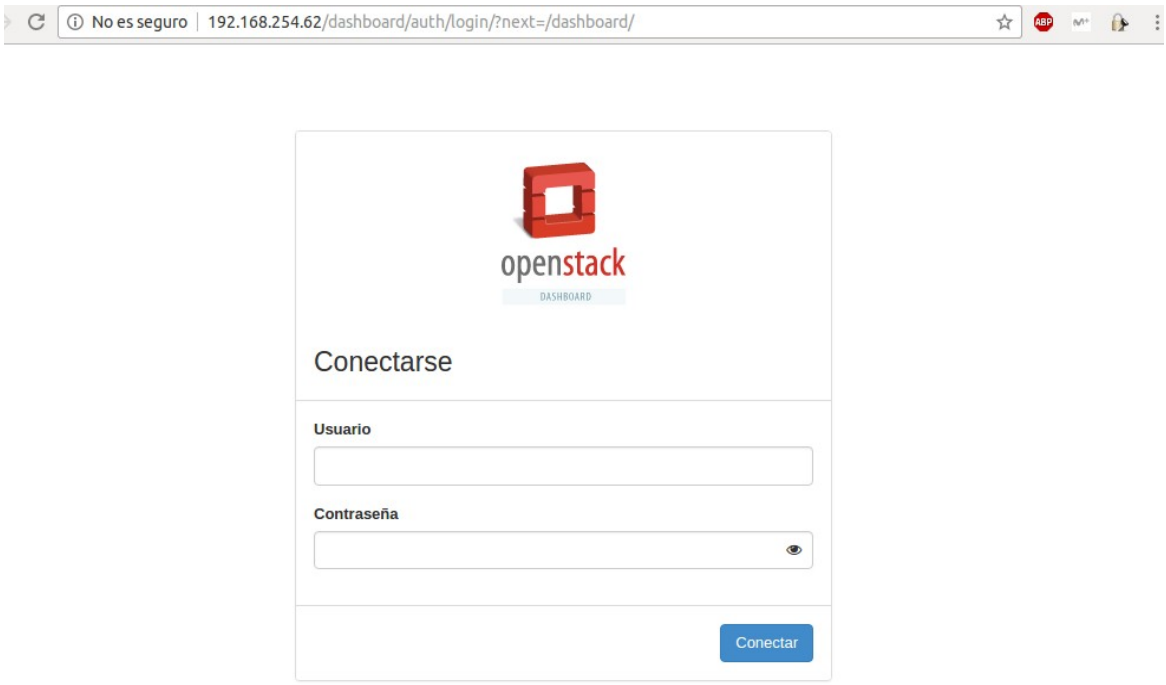
Durant el procés de la instal·lació, si ens pregunten quines contrasenyes volem que tingui l'administrador, la base de dades... li donem a la tecla enter a totes les preguntes, agafarà per defecte el valor que hem definit al fitxer local.conf.

```
########################################################################
ENTER A PASSWORD TO USE FOR THE DATABASE.
########################################################################
This value will be written to /home/usuario/devstack/.localrc.password file so y
ou don't have to enter it
again.  Use only alphanumeric characters.
If you leave this blank, a random default value will be used.
```

9. Un cop la instal·lació ha finalitzat correctament (en el meu ordinador ha trigat 13 minuts aproximadament), ens mostrarà per pantalla diferents estadístiques, el teu host, la direcció del mòdul Horizon per poder entrar a la administració via servidor web...

```
Total runtime         797

run_process            64
test_with_retry         4
apt-get-update          3
pip_install           142
restart_apache_server  16
wait_for_service       12
apt-get                 5
========================



This is your host IP address: 192.168.254.62
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.254.62/dashboard
Keystone is serving at http://192.168.254.62/identity/
The default users are: admin and demo
The password: admin
2017-04-18 14:22:13.839 | WARNING:
2017-04-18 14:22:13.839 | Using lib/neutron-legacy is deprecated, and it will be
 removed in the future
2017-04-18 14:22:13.839 | stack.sh completed in 797 seconds.
stack@soyuz:~/devstack$
```

10. Després, obrim el navegador amb la direcció IP de la màquina virtual i entrem amb els paràmetres de login de l'administrador definits al fitxer local.conf. D'aquesta manera tindrem accés a la interfície web OpenStack Horizon, que es tracta del mòdul Horizon explicat anteriorment.



2.6 Utilitzant Horizon a DevStack

Horizon es el mòdul que ens proveeix una interfície web que ens fa més fàcil l'administració d'OpenStack.

Abans de començar a treballar amb Horizon ens hem de familiaritzar amb uns conceptes essencials.

- **Service (Servei)**: Un servei es un projecte. Es un component essencial de OpenStack.
- **Tenant (Inquilí)**: Un inquilí es un client. Si OpenStack s'està utilitzant un cloud públic es poden crear múltiples inquilins.
- **Role (Rol)**: Un rol son un grup de permisos que son utilitzats per a permetre als usuaris poder fer determinades accions.
- **User**: Un usuari es una entitat administrativa que te assignat un rol específic.

Abans d'entrar podem veure ràpidament com es troba el servidor Apache que estem utilitzant.

```
stack@ubuntu:~$ systemctl status apache2.service
```

Això ens mostra que el servidor Apache està funcionant perfectament i que està allotjant l'Horizon també conegut com a dashboard.

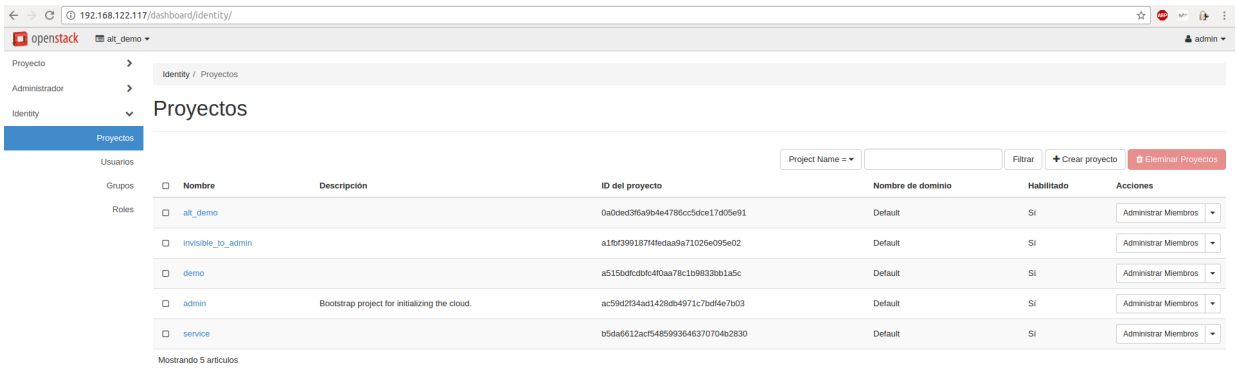
```
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since jue 2017-04-27 15:33:09 CEST; 46min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1282 ExecStart=/etc/init.d/apache2 start (code=exited, status=0/SUCCESS)
    Tasks: 137
   Memory: 1.0G
      CPU: 27.923s
   CGroup: /system.slice/apache2.service
           ├─1439 /usr/sbin/apache2 -k start
           ├─1451 (wsgi:horizon)    -k start
           ├─1452 (wsgi:horizon)    -k start
           ├─1453 (wsgi:horizon)    -k start
           ├─1454 (wsgi:keystone-pu -k start
           ├─1455 (wsgi:keystone-pu -k start
           ├─1456 (wsgi:keystone-pu -k start
           ├─1457 (wsgi:keystone-pu -k start
           ├─1458 (wsgi:keystone-pu -k start
           ├─1459 (wsgi:keystone-ad -k start
           ├─1460 (wsgi:keystone-ad -k start
           ├─1461 (wsgi:keystone-ad -k start
           ├─1462 (wsgi:keystone-ad -k start
           ├─1463 (wsgi:keystone-ad -k start
           ├─1464 /usr/sbin/apache2 -k start
           └─1465 /usr/sbin/apache2 -k start
```

Ara si que podem continuar i entrar a la interfície web. A la instal·lació de DevStack s'han afegit els usuaris admin i demo (el qual es un usuari sense privilegis).

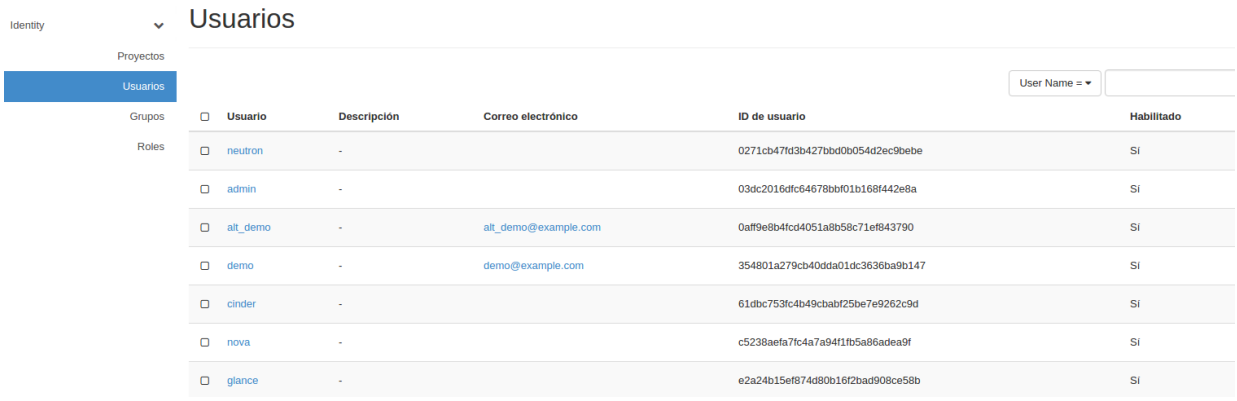
Cadascun d'aquests dos usuaris tenen vistes diferents. Si logines amb l'administrador, veuràs uns objectes i pestanyes diferents que si ho fas amb un usuari normal.

L'usuari normal veu un entorn específic per a aquell usuari, de manera que entre els diferents usuaris normals poden tenir diferents vistes.

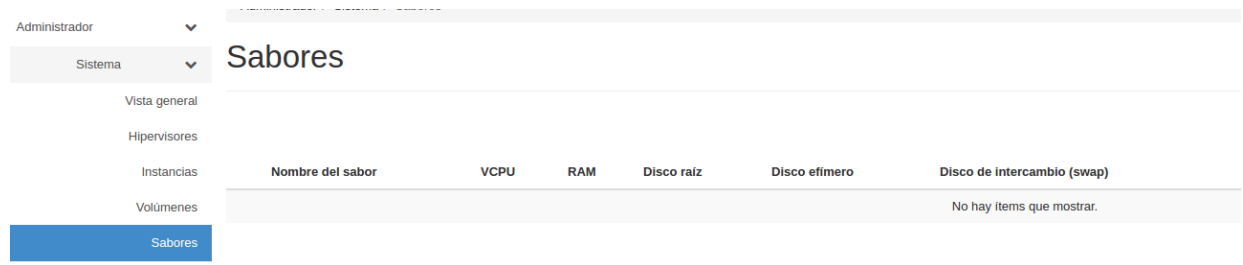
Loginem amb l'usuari Admin i se'ns mostrarà la següent vista:



La part més important de la vista de l'administrador es la taula d'identitats a la que es configura els usuaris, els grups, l'entorn que veu cada usuari...



Una altre opció que te l'administrador es la de gestionar la infraestructura dels clouds des de la pestanya Admin. Des de aquí es poden veure quins hipervisors (màquines) estan disponibles, es poden agregar hosts, grups d'hipervisors. A més te permet gestionar els «Sabors», que son els perfils del hardware, o les imatges de les que es disposen per iniciar instàncies.



Per últim tenim la pestanya de gestió de projectes. Aquesta opció està a la vista de l'administrador perquè aquest usuari pot veure y gestionar tot però no es una cosa que normalment es gestiona des de aquest usuari. Habitualment es separa l'entorn de l'usuari normal, que conté l'entorn en el que es van a gestionar les màquines virtuals, de l'entorn de l'administrador, que realment està destinat a proporcionar la infraestructura i a gestionar-la.

2.7 Desplegant OpenStack manualment

2.7.1 Configuració del maquinari

Per començar a instal·lar OpenStack de manera manual, hem de tenir clar les màquines que necessitem, l'esquema de xarxa...

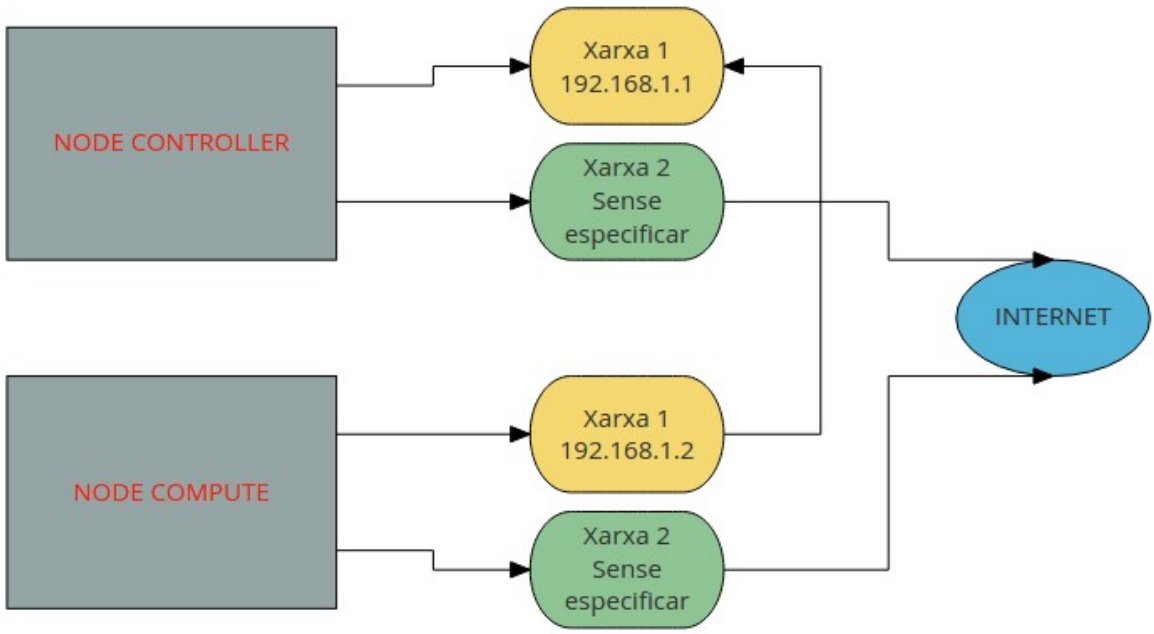
2.7.1.1 Nodes necessaris

Primerament crearem dos màquines virtuals a KVM, una anomenada Controller i l'altre anomenada Compute. Aquests dos nodes seran els encarregats de suportar OpenStack.

Aquests dos nodes disposaran de 4 GiB de memòria cadascun i de 20 GiB d'emmagatzematge el node Controller i 10 GiB el node Compute.

2.7.1.2 Esquema de xarxa

OpenStack pot treballar amb dues xarxes independents, de manera que el nostre esquema disposarà de dues targes de xarxa per cadascun dels nodes: Una serà la encarregada de la comunicació interna entre els dos nodes i l'altre que serà l'encarregada de proveir accés a internet.



De manera que el fitxer /etc/network/interfaces hauría de quedar així per als diferents nodes:

Node Controller

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto ens3
iface ens3 inet dhcp

auto ens7
iface ens7 inet static
address 192.168.1.1
netmask 255.255.255.0
```

Node Compute

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto ens3
iface ens3 inet dhcp

auto ens7
iface ens7 inet static
address 192.168.1.2
netmask 255.255.255.0
```

2.7.1.3 Configuració dels hosts

El següent que hem de fer es configurar el fitxer /etc/hosts per a que els nodes es trobin sense necessitat de posar la ip, a més de canviar el hostname de cada node. Els nostres hostsname seran controller i compute, un per a cada node definit anteriorment.

Node Controller

```
root@ubuntu:~$ hostnamectl set-hostname controller
```

```
127.0.0.1       localhost
127.0.1.1       controller
192.168.1.2     compute

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
~
```

Node Compute

```
root@ubuntu:~$ hostnamectl set-hostname compute
```

```
127.0.0.1       localhost
127.0.1.1       compute
192.168.1.1     controller

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
~
```

Un cop fet això, reiniciem les xarxes i fem un ping per comprovar que l'esquema de xarxa que hem creat funciona correctament.

```
root@compute:~# ping controller
PING controller (192.168.1.1) 56(84) bytes of data.
64 bytes from controller (192.168.1.1): icmp_seq=1 ttl=64 time=0.295 ms
64 bytes from controller (192.168.1.1): icmp_seq=2 ttl=64 time=0.265 ms
```

2.7.1.4 Configuració d'NTP

Un cop tenim definides les estructures dels nodes, les xarxes, els seus hosts i hem comprovat que tot funciona bé, configurarem un servidor d'hora NTP(Network Time Protocol) per a que els nostres nodes estiguin sincronitzats en el temps i evitar possibles errors de sincronització.

Per a realitzar això, utilitzarem el programa chrony, que és una implementació versàtil del NTP. Entre d'altres coses, pot sincronitzar el rellotge del sistema amb servidors NTP, de manera que els nostres nodes aniran a la mateixa hora exacta.

La instal·lació es diferent per a cada node, per això ho dividirem en dues parts:

Node Controller

1. Instal·lem el programa chrony

```
root@controller:~$ apt install chrony
```

2. Editem el fitxer de configuració **/etc/chrony/chrony.conf** de manera que afegim la següent línia:

   server **Server_NTP** iburst on substituirem Server_NTP per el servidor que volem utilitzar.

   En el nostre cas quedaria així:

```
server 0.europe.pool.ntp.org iburst
```

   Desem el fitxer i reiniciem el servei.

```
root@controller:~$ systemctl restart chrony.service
```

Node Compute

1. Instal·lem el programa chrony

```
root@controller:~$ apt -y install chrony
```

2. Editem el fitxer de configuració **/etc/chrony/chrony.conf** però escollirem com a servidor d'hora el del nostre node controller. Es pot posar per IP o per hostname.

```
server controller iburst
```

   Desem el fitxer i reiniciem el servei.

```
root@controller:~$ systemctl restart chrony.service
```

2.7.1.5 Configuració de repositoris

Per defecte els paquets de l'ultima versió d'OpenStack, la versió newton, que es la que nosaltres utilitzarem, no estan a Ubuntu 16.04, pel que hem d'afegir-los manualment al nostre node controller:

Node Controller i Compute

```
root@controller:~$ apt -y install software-properties-common
root@controller:~$ add-apt-repository cloud-archive:newton
```

Un cop fet això, ens deuria de sortir per pantalla aquesta confirmació de que tot ha anat bé:

```
root@controller:~# add-apt-repository cloud-archive:newton
 Ubuntu Cloud Archive for OpenStack Newton
 Más información: https://wiki.ubuntu.com/ServerTeam/CloudArchive
Pulse [Intro] para continuar o ctrl-c para cancelar

Leyendo lista de paquetes...
Creando árbol de dependencias...
Leyendo la información de estado...
Se instalarán los siguientes paquetes NUEVOS:
  ubuntu-cloud-keyring
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 163 no actualizados.
Se necesita descargar 5.086 B de archivos.
Se utilizarán 34,8 kB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu xenial/universe amd64 ubuntu-cloud-keyring all 2012.0
8.14 [5.086 B]
Descargados 5.086 B en 0s (47,5 kB/s)
Seleccionando el paquete ubuntu-cloud-keyring previamente no seleccionado.
(Leyendo la base de datos ... 59647 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../ubuntu-cloud-keyring_2012.08.14_all.deb ...
Desempaquetando ubuntu-cloud-keyring (2012.08.14) ...
Configurando ubuntu-cloud-keyring (2012.08.14) ...
Importing ubuntu-cloud.archive.canonical.com keyring
OK
Processing ubuntu-cloud.archive.canonical.com removal keyring
gpg: /etc/apt/trustdb.gpg: se ha creado base de datos de confianza
OK
```

Ara actualitzarem el sistema un altre cop per a que es descarregui tots els paquets relacionats amb OpenStack Newton:

```
root@controller:~$ apt -y update
root@controller:~$ apt -y dist-upgrade
```

Finalment instal·lem el paquet de python d'OpenStack per poder interactuar amb OpenStack i les seves api's.

```
root@controller:~$ apt -y install python-openstackclient
```

2.7.1.6 Instal·lació base de dades MariaDB

Tots els components d'OpenStack necessiten una base de dades per emmagatzemar informació. OpenStack ens permet utilitzar bases de dades, tant relacionals com no relacionals. En aquest projecte utilitzarem MariaDB que es més àgil que altres com PostgreSQL.

Node Controller

```
root@controller:~$ apt -y install mariadb-server python-pymysql
```

Un cop la instal·lació ha finalitzat, creem el fitxer **/etc/mysql/conf.d/openStack.conf** per tal de configurar la base de dades per a funcionar amb OpenStack afegint les següents línies:

```
[mysqld]
bind-address = 192.168.1.1
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8

[mysql]
default-character-set = utf8
```

Reiniciem el servei de Mysql:

```
root@controller:~$ systemctl restart mysql.service
```

2.7.1.7 Instal·lació de cua de missatges Rabbit MQ

Els components d'OpenStack necessiten estar informats en tot moment de l'estat entre els diferents serveis als nodes per poder coordinar les operacions, de manera que

Com que OpenStack utilitza una cua de missatges per fer això i disposa de múltiples serveis compatibles amb ell, utilitzarem un. En el nostre cas utilitzarem RabbitMQ al nostre node controller:

Node Controller

```
root@controller:~$ apt -y install rabbitmq-server
```

Creem l'usuari que utilitzarem a RabbitMQ on el penúltim paràmetre es el nom de l'usuari i l'ultim la contrasenya:

```
root@controller:~$ rabbitmqctl add_user openstack openstack
```

Per comprovar que l'usuari s'ha creat correctament utilitzarem la següent comanda:

```
root@controller:~$ rabbitmqctl list_users
```

```
root@controller:~# rabbitmqctl list_users
Listing users ...
guest   [administrator]
openstack       []
```

Un cop hem comprovat que l'usuari existeix, li assignem permisos d'escriptura i lectura:

```
root@controller:~$ rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

2.7.1.8 Instal·lació de sistema distribuït de caché Memcached

OpenStack utilitza un sistema de distribució de cache que li permet emmagatzemar dades i objectes per facilitar i agilitzar el treball. Sobretot s'utilitza per el primer mòdul que instal·larem, Keystone.

Node Controller

```
root@controller:~$ apt -y install memcached python-memcache
```

Si la instal·lació ha anat correctament, obrim el fitxer **/etc/memcached.conf** per anar a la configuració i especificar que accepti peticions de la nostre tarja de xarxa, de xarxa interna, perquè Memcached està configurat per defecte que només accepti peticions per localhost i necessitem que es comuniqui amb els altres nodes:

Com podem comprovar, a l'opció -l, que es la que indica en quina ip està escoltant, i per tant, en quina tarja, està posada en localhost.

```
# Specify which IP address to listen on. The default is to listen on all IP addr
esses
# This parameter is one of the only security measures that memcached has, so mak
e sure
# it's listening on a firewalled interface.
-l 127.0.0.1
```

La canviem per la IP que vam definir al començament per al node controller:

```
# Specify which IP address to listen on. The default is to listen on all IP addr
esses
# This parameter is one of the only security measures that memcached has, so mak
e sure
# it's listening on a firewalled interface.
-l 192.168.1.1
```

Un cop canviat, reiniciem el servei:

```
root@controller:~$ systemctl restart memcached.service
```

2.7.2 Instal·lació dels mòduls

Com ja vam explicar a l'apartat **2.3.4** a la que explicàvem la estructura, OpenStack disposa de diferents mòduls, que junts fan OpenStack. No es necessari instal·lar tots, només els que necessitem per a una determinada tasca o objectiu.

2.7.2.1 KeyStone (Serveis d'identitat)

KeyStone es el servei que utilitza OpenStack per a la autenticació i autorització entre els diferents serveis, basat en tokens. Disposa d'una API, actualment a la versió 3.8, que es utilitzada per els desenvolupadors per fer eines d'accés als recursos.

2.7.2.1.1 Coneixent KeyStone

Com sempre, necessitem uns coneixements bàsics per a poder entendre com funciona ja que disposa de diferents components específics:

- **Usuari**: Els usuaris, que disposen d'un login, poden ser sistemes, persones reals o serveis que utilitzen el nuvol d'OpenStack. KeyStone s'encarrega de validar les autenticacions d'aquests.

- **Rol**: Son utilitzats per a definit les accions que poden realitzar els usuaris. Aquests rols disposen d'un grup de privilegis per a cada usuari.

- **Token**: Es un bit utilitzat per a accedir als recursos. Els tokens es generen quan es confirma la autenticació de manera que cadascun d'ells te un abast que ens permet conèixer a quins recursos pot accedir l'usuari.

- **Credencial**: Son les dades que l'usuari utilitza per autenticar-se. Aquestes dades poden ser en forma de login i contrasenya, de token...

- **Autenticació**: Es l'acció de confirmar la identitat d'un usuari utilitzant els credencials que li han donat. Quan es realitza aquesta confirmació, es proveeix d'un token per ser utilitzat en les peticions de l'usuari de manera que no necessiti autenticar-se a cada moment.

- **Tenant**: Poden ser clients, organitzacions, comptes o projectes que s'utilitzen per a agrupar i aïllar els recursos dels que es disposen.

- **Endpoint**: Es l'adreçament IP o URI per a accedir als serveis.

Un cop tenim clar els components, realitzarem la instal·lació del mòdul.

2.7.2.1.2 Instal·lació de KeyStone

Primerament, hem de crear i configurar la base de dades que requereix aquest mòdul (tots els mòduls necessiten una), on desarà la informació.

Node Controller

```
root@controller:~$ mysql -u root -p
```

```
MariaDB [(none)]> CREATE DATABASE keystone;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'contrasenyaEscollida';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'192.168.1.1' IDENTIFIED BY 'contrasenyaEscollida';

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

On 'contrasenyaEscollida' serà la contrasenya que volem posar a la base de dades.

Continuem generant un token temporal per a la configuració inicial de KeyStone (Ens retornarà una cadena que hem de copiar per fer servir després).

```
root@controller:~$ openssl rand -hex 10
```

Hens ha retornat 3aa5d36bc51ec5c0be89.

Deshabilitem l'opció de que KeyStone arrenqui automàticament després de la seva instal·lació.

```
root@controller:~$ echo "manual" > /etc/init/keystone.override
```

Un cop fet, instal·lem els paquets necessaris per a utilitzar aquest mòdul entre els que s'inclouen el del propi mòdul i els necessaris per a que funcioni amb Apache2 i el mòdul wsgi que explicarem més endavant.

```
root@controller:~$ apt-get install keystone apache2 libapache2-mod-wsgi
```

2.7.2.1.3 Configuració de KeyStone

Per a configurar KeyStone, anirem al fitxer /etc/keystone/keystone.conf
Un cop endins, ens hem de moure per diferents seccions per tal de fer funcional aquest mòdul:

1. Anem a la secció [DEFAULT] i afegim el token que hem generat anteriorment:

```
admin_token = 3aa5d36bc51ec5c0be89
```

```
[DEFAULT]

#
# From keystone
#

# Using this feature is *NOT* recommended. Instead, use the `keystone-manage
# bootstrap` command. The value of this option is treated as a "shared secret"
# that can be used to bootstrap Keystone through the API. This "token" does not
# represent a user (it has no identity), and carries no explicit authorization
# (it effectively bypasses most authorization checks). If set to `None`, the
# value is ignored and the `admin_token` middleware is effectively disabled.
# However, to completely disable `admin_token` in production (highly
# recommended, as it presents a security risk), remove
# `AdminTokenAuthMiddleware` (the `admin_token_auth` filter) from your paste
# application pipelines (for example, in `keystone-paste.ini`). (string value)
admin_token = 3aa5d36bc51ec5c0be89
```

2. Anem a la secció [database] i afegim la configuració per accedir a la base de dades que hem definit anteriorment:

```
connection = mysql+pymysql://keystone:contrasenyaEscollida@controller/keystone
```

```
[database]

#
# From oslo.db
#

# DEPRECATED: The file name to use with SQLite. (string value)
# Deprecated group/name - [DEFAULT]/sqlite_db
# This option is deprecated for removal.
# Its value may be silently ignored in the future.
# Reason: Should use config option connection or slave_connection to connect
# the database.
#sqlite_db = oslo.sqlite
connection = mysql+pymysql://keystone:contrasenyaEscollida@controller/keystone
```

3. Anem a la secció [token] i configurem el proveïdor de token «fernet». KeyStone utilitza tokens Fernet que utilitzen claus privades compartides per a que aquests tokens no tinguin que ser emmagatzemats i no es produeixin rèpliques a la base de dades. Això redueix considerablement la carga de treball a la base de dades de Keystone.

```
provider = fernet
```

```
[token]

#
# From keystone
#

# This is a list of external authentication mechanisms which should add token
# binding metadata to tokens, such as `kerberos` or `x509`. Binding metadata is
# enforced according to the `[token] enforce_token_bind` option. (list value)
#bind =
provider = fernet
```

Un cop hem editat el fitxer configuració, desem els canvis i realitzem la sincronització amb la base de dades de Keystone.

```
root@controller:~$ su -s /bin/sh -c "keystone-manage db_sync" keystone
```

Si aquesta fase de sincronització ens dona error, anar a l'apartat de gestió d'errors 3.3 i 3.4.

Si hem aconseguit fer la sincronització, inicialitzem els token fernet amb la següent comanda:

```
root@controller:~$ keystone-manage fernet_setup --keystone-user keystone –
keystone-group keystone
```

Si ens dona la següent sortida per pantalla no es cap error, son avisos sobre la creació i inicialització d'aquests tokens, pel que haurà resultat un èxit.

```
root@controller:~# keystone-manage fernet_setup --keystone-user keystone --keyst
one-group keystone
2017-05-07 17:31:02.269 3052 INFO keystone.common.fernet_utils [-] key_repositor
y does not appear to exist; attempting to create it
2017-05-07 17:31:02.282 3052 INFO keystone.common.fernet_utils [-] Created a new
 key: /etc/keystone/fernet-keys/0
2017-05-07 17:31:02.282 3052 INFO keystone.common.fernet_utils [-] Starting key
rotation with 1 key files: ['/etc/keystone/fernet-keys/0']
2017-05-07 17:31:02.283 3052 INFO keystone.common.fernet_utils [-] Current prima
ry key is: 0
2017-05-07 17:31:02.283 3052 INFO keystone.common.fernet_utils [-] Next primary
key will be: 1
2017-05-07 17:31:02.283 3052 INFO keystone.common.fernet_utils [-] Promoted key
0 to be the primary: 1
2017-05-07 17:31:02.284 3052 INFO keystone.common.fernet_utils [-] Created a new
 key: /etc/keystone/fernet-keys/0
```

A continuació fem el mateix amb els relacionats amb els credencials:

```
root@controller:~$ keystone-manage credential_setup --keystone-user keystone --keystone-group keystone
```

```
root@controller:~# keystone-manage credential_setup --keystone-user keystone --k
eystone-group keystone
2017-05-07 17:33:38.213 3094 INFO keystone.common.fernet_utils [-] key_repositor
y does not appear to exist; attempting to create it
2017-05-07 17:33:38.214 3094 INFO keystone.common.fernet_utils [-] Created a new
 key: /etc/keystone/credential-keys/0
2017-05-07 17:33:38.214 3094 INFO keystone.common.fernet_utils [-] Starting key
rotation with 1 key files: ['/etc/keystone/credential-keys/0']
2017-05-07 17:33:38.214 3094 INFO keystone.common.fernet_utils [-] Current prima
ry key is: 0
2017-05-07 17:33:38.215 3094 INFO keystone.common.fernet_utils [-] Next primary
key will be: 1
2017-05-07 17:33:38.215 3094 INFO keystone.common.fernet_utils [-] Promoted key
0 to be the primary: 1
2017-05-07 17:33:38.215 3094 INFO keystone.common.fernet_utils [-] Created a new
 key: /etc/keystone/credential-keys/0
```

2.7.2.1.4 Configuració de KeyStone amb Apache2 + Mod_WSGI

A continuació configurarem KeyStone amb Apache2 i el mòdul wsgi, que com vam dir anteriorment, explicarem.

El mòdul wsgi es un dels mòduls propis d'Apache que permet servir aplicacions fetes amb Python.

El primer que hem de fer es editar el fitxer de configuració /etc/apache2/apache2.conf on hem d'especificar com a ServerName el nostre node controller.

Afegim la següent línia:

```
ServerName controller
```

Després creem un virtualhost nou en el que configurarem l'accés de Keystone amb el mòdul wsgi. Per això, creem el fitxer **/etc/apache2/sites-available/wsgi-keystone.conf** on afegirem el següent per a que funcioni per els ports 5000 i 35357:

```
Listen 5000
Listen 35357
<VirtualHost *:5000>

WSGIDaemonProcess keystone-public1 processes=5 threads=1 user=keystone
group=keystone display-name=%{GROUP}
        WSGIProcessGroup keystone-public1
        WSGIScriptAlias / /usr/bin/keystone-wsgi-public
        WSGIApplicationGroup %{GLOBAL}
        WSGIPassAuthorization On
        ErrorLogFormat "%{cu}t %M"
        ErrorLog /var/log/apache2/keystone.log
        CustomLog /var/log/apache2/keystone_access.log combined

        <Directory /usr/bin>
                Require all granted
        </Directory>
</VirtualHost>

<VirtualHost *:35357>
WSGIDaemonProcess keystone-admin1 processes=5 threads=1 user=keystone
group=keystone display-name=%{GROUP}
        WSGIProcessGroup keystone-admin1
        WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
        WSGIApplicationGroup %{GLOBAL}
        WSGIPassAuthorization On
        ErrorLogFormat "%{cu}t %M"
        ErrorLog /var/log/apache2/keystone.log
        CustomLog /var/log/apache2/keystone_access.log combined

        <Directory /usr/bin>
                Require all granted
        </Directory>
</VirtualHost>
```

Un cop desat el fitxer, habilitem el fitxer, bé fent servint un enllaç simbòlic a sites-enabled d'Apache o ve utilitzant la comanda a2ensite:

Enllaç simbólic:

```
root@controller:~$ ln -s /etc/apache2/sites-available/wsgi-keystone.conf /etc/apache2/sites-enabled
```

Comanda a2ensite:

```
root@controller:~$ a2ensite wsgi-keystone.conf
```

Per finalitzar la configuració d'Apache, el reiniciem i eliminem la base de dades sqlite que utilitza Keystone per defecte.

```
root@controller:~$ systemctl restart apache2.service
```

```
root@controller:~$ rm -f /var/lib/keystone/keystone.db
```

2.7.2.1.5 Creació del servei Keystone + Endpoints

Necessitem crear el servei de Keystone, Identity, que proporciona un catàleg de serveis i les seves ubicacions. Cada servei que s'afegeix a OpenStack requereix una entitat de servei i diferents endpoints al catàleg.

Per defecte, la base de dades del servei de Keystone Identity no conté cap informació que accepti serveis d'autenticació i catàlegs. Tenim que utilitzar el token temporal que vam definir en apartats anteriors i uns endpoints de l'API per al servei per a que funcioni.

Per fer això, utilitzarem les següents comandes per a passar els diferents valors necessaris per a que estiguin establerts com a variables d'entorn.

```
root@controller:~$ export OS_TOKEN=ADMIN_TOKEN
```

On substituirem ADMIN_TOKEN pel token generat anteriorment:

```
root@controller:~$ export OS_TOKEN=3aa5d36bc51ec5c0be89
```

Després, configurem el endpoint de la URL:

```
root@controller:~$ export OS_URL=http://controller:35357/v3
```

I configurem la versió de la API:

```
root@controller:~$ export OS_IDENTITY_API_VERSION=3
```

Un cop fet això, creem el servei identity.

```
root@controller:~$ openstack service create --name keystone --description
"OpenStack Identity" identity
```

Si s'ha creat correctament, ens sortirà el següent missatge per pantalla en el que ens mostra el servei creat, si està habilitat, el nom, tipus...i l'id que es genera automàticament.

```
root@controller:~# openstack service create --name keystone --description "OpenS
tack Identity" identity
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Identity               |
| enabled     | True                             |
| id          | 0d012401b90047808e526e7e656ecccc |
| name        | keystone                         |
| type        | identity                         |
+-------------+----------------------------------+
```

El servei identity gestiona un catàleg d'endpoints d'APIs associat als serveis en l'entorn del nostre OpenStack. Els serveis utilitzen aquest catàleg per a determinar com es comuniquen amb els altres serveis del nostre entorn.

OpenStack utilitza tres variants d'endpoints d'APIS per a cada servei: admin, internal i public:

1. **Endpoint Admin**: Permet modificar els usuaris i tenants per defecte, mentre que els altres no.

2. **Endpoint Public**: Pot ser visible des de internet per a que els usuaris administrin els núvols

3. **Endpoint Internal**: Pot restringir els accessos per xarxa per a fer-ho accessible només dins de l'organització.

A més, OpenStack permet múltiples regions per a l'escalabilitat. Per simplificar, utilitza la xarxa d'administració per a totes les variacions d'endpoints i la RegionOne per defecte.

A continuació crearem els tres endpoints d'API per al nostre servei identity.

Endpoint public:

```
root@controller:~$ openstack endpoint create --region RegionOne identity public http://controller:5000/v3
```

```
root@controller:~# openstack endpoint create --region RegionOne identity public
http://controller:5000/v3
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | c6ed8f61d6034378a27b5cbeb4ed8f37 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 0d012401b90047808e526e7e656ecccc |
| service_name | keystone                         |
| service_type | identity                         |
| url          | http://controller:5000/v3        |
+--------------+----------------------------------+
```

Endpoint Internal:

```
root@controller:~$ openstack endpoint create --region RegionOne identity internal http://controller:5000/v3
```

```
root@controller:~# openstack endpoint create --region RegionOne identity interna
l http://controller:5000/v3
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 7eb929f8d7424e08bf71477a2490fb47 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 0d012401b90047808e526e7e656ecccc |
| service_name | keystone                         |
| service_type | identity                         |
| url          | http://controller:5000/v3        |
+--------------+----------------------------------+
```

Endpoint Admin:

```
root@controller:~$ openstack endpoint create --region RegionOne identity admin http://controller:35357/v3
```

```
root@controller:~# openstack endpoint create --region RegionOne identity admin h
ttp://controller:35357/v3
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 546ccd1c253e4f0589be5a472b330480 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 0d012401b90047808e526e7e656ecccc |
| service_name | keystone                         |
| service_type | identity                         |
| url          | http://controller:35357/v3       |
+--------------+----------------------------------+
```

2.7.2.1.6 Creació de dominis, projectes, usuaris i rols

Quan vam fer la instal·lació d'OpenStack mitjançant l'script DevStack, vam explicar l'apartat relacionat amb KeyStone en els que vam parlar dels projectes, usuaris, rols...

El servei identity que hem creat proporciona serveis d'autenticació per a cada servei d'OpenStack. Els serveis d'autenticació utilitzen una combinació de dominis, projectes (tenants), usuaris i rols.

A continuació o posarem en pràctica:

1. Creació d'un domini per defecte:

```
root@controller:~$ openstack domain create --description "Default Domain" default
```

```
root@controller:~# openstack domain create --description "Default Domain" defaul
t
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Default Domain                   |
| enabled     | True                             |
| id          | 82d7ae268d914dc8999bf852fcf3dd6e |
| name        | default                          |
+-------------+----------------------------------+
```

2. Creació del projecte Admin:

```
root@controller:~$ openstack project create --domain default --description "Admin Project" admin
```

```
root@controller:~# openstack project create --domain default --description "Admi
n Project" admin
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Admin Project                    |
| domain_id   | 82d7ae268d914dc8999bf852fcf3dd6e |
| enabled     | True                             |
| id          | 3df3a9e2446541868520ab7cab1d5933 |
| is_domain   | False                            |
| name        | admin                            |
| parent_id   | 82d7ae268d914dc8999bf852fcf3dd6e |
+-------------+----------------------------------+
```

3. Creació del usuari Admin:

```
root@controller:~$ openstack user create --domain default --password-prompt admin
```

```
root@controller:~# openstack user create --domain default --password-prompt admi
n
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | 82d7ae268d914dc8999bf852fcf3dd6e |
| enabled             | True                             |
| id                  | b74caa92bb5545c78e26bf51fab485f4 |
| name                | admin                            |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

4. Creació del rol Admin:

```
root@controller:~$ openstack role create admin
```

```
root@controller:~# openstack role create admin
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 8cabd7b7a7094e969372e49af476d049 |
| name      | admin                            |
+-----------+----------------------------------+
```

5. Assignem el rol Admin a l'usuari Admin i al projecte Admin.

```
root@controller:~$ openstack role add --project admin --user admin admin
```

A més, al nostre projecte utilitzarem un projecte service que contindrà un únic usuari per a cada servei que afegim al nostre entorn. Creem el projecte service.

```
root@controller:~$ openstack project create --domain default --description "Service Project" service
```

```
root@controller:~# openstack project create --domain default --description "Service Project" service
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Service Project                  |
| domain_id   | 82d7ae268d914dc8999bf852fcf3dd6e |
| enabled     | True                             |
| id          | 405f3aa4f59242a89994475e0b707c91 |
| is_domain   | False                            |
| name        | service                          |
| parent_id   | 82d7ae268d914dc8999bf852fcf3dd6e |
+-------------+----------------------------------+
```

L'usuari Admin se'n encarrega de la infraestructura dels núvols. Per tant, per a fer tasques regulars (sense necessitat d'administrador), s'han d'utilitzar projectes i usuaris sense privilegis. Com a exemple crearem un projecte i un usuari com a demostració de que el servei funciona correctament i ens sol·licita el token quan ens volem autenticar.

1. Creem el projecte demo:

```
root@controller:~$ openstack project create --domain default --description "Demo Project" demo
```

```
root@controller:~# openstack project create --domain default --description "Demo Project" demo
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Demo Project                     |
| domain_id   | 82d7ae268d914dc8999bf852fcf3dd6e |
| enabled     | True                             |
| id          | b2fc7cb72ed64a9db085976af6421515 |
| is_domain   | False                            |
| name        | demo                             |
| parent_id   | 82d7ae268d914dc8999bf852fcf3dd6e |
+-------------+----------------------------------+
```

2. Creem l'usuari demo:

```
root@controller:~$ openstack user create --domain default --password-prompt
demo
```

```
root@controller:~# openstack user create --domain default --password-prompt demo
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | 82d7ae268d914dc8999bf852fcf3dd6e |
| enabled             | True                             |
| id                  | d153971cfa4a4a0e854f8dd709ac3c2e |
| name                | demo                             |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

3. Creem el rol user:

```
root@controller:~$ openstack user create --domain default --password-prompt
demo
```

```
root@controller:~# openstack role create user
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 3a268671a34649569ee77627de6de5b2 |
| name      | user                             |
+-----------+----------------------------------+
```

4. Afegim el rol user a l'usuari demo i al projecte demo:

```
root@controller:~$ openstack role add --project demo --user demo user
```

2.7.2.1.7 Verificació de les operacions realitzades al servei Identity

A continuació verificarem que el servei identity funciona correctament:

1. En primer lloc deshabilitarem temporalment el mètode d'autenticació mitjançant tokens per raons de seguretat. Editem el fitxer **/etc/keystone/keystone-paste.ini** i esborrem la paraula `admin_token_auth` dels apartats [pipeline:public_api], [pipeline:admin_api] i [pipeline:api_v3].

2. Deshabilitem temporalment l'OS_TOKEN i l'OS_URL com a variables d'entorn per a poder sol·licitar nous tokens al servei identity o ens donarà un error:

```
root@controller:~$ unset OS_TOKEN OS_URL
```

3. Demanem un token per a l'usuari admin introduïnt la contrasenya quan sen's demani:

```
root@controller:~$ openstack --os-auth-url http://controller:35357/v3 --os-project-domain-name default --os-user-domain-name default --os-project-name admin --os-username admin token issue
```

```
root@controller:~# openstack --os-auth-url http://controller:35357/v3 --os-proje
ct-domain-name default --os-user-domain-name default --os-project-name admin --o
s-username admin token issue
Password:
+------------+----------------------------------------------------------------+
| Field      | Value                                                          |
+------------+----------------------------------------------------------------+
| expires    | 2017-05-07 18:44:06+00:00                                      |
| id         | gAAAAABZD1zmBQX-3ilaNvkkItb8dqHrBEayZolwl1hyVY9nKLhNsOXGFtfMz4O |
|            | vLVna96dyRqeiOZ9Lj82bDUQu1jTEarFkUuDwn6T4R3KnuVX6vVbfa07ImzSBSP |
|            | Qy-uSMXPJ_8aDdG48SoAu0pQOaWhj8zRvo7B7Md44aAggjdLk72r2K75o       |
| project_id | 3df3a9e2446541868520ab7cab1d5933                               |
| user_id    | b74caa92bb5545c78e26bf51fab485f4                               |
+------------+----------------------------------------------------------------+
```

4. Ara sol·licitem un token per a l'usuari demo:

```
root@controller:~$ openstack --os-auth-url http://controller:5000/v3 --os-project-domain-name default --os-user-domain-name default --os-project-name demo --os-username demo token issue
```

```
root@controller:~# openstack --os-auth-url http://controller:5000/v3 --os-projec
t-domain-name default --os-user-domain-name default --os-project-name demo --os-
username demo token issue
Password:
+------------+----------------------------------------------------------------+
| Field      | Value                                                          |
+------------+----------------------------------------------------------------+
| expires    | 2017-05-07 18:45:52+00:00                                      |
| id         | gAAAAABZD11QB0kn3C1Djp6S1AIU_l6G5t9A8OS3iEG9uqJYqt9eaAf7x1kPY4U |
|            | nOZj0DEGdklM0IG8Jxf860k2SFFF50c4sw5clkB13Jaf3ZmUgVjDwStp3d0GlYc |
|            | PemC9xDGUYniPdRc19yegjjVRsAUQu3sVpcUnIztQPy-oODKqMHzlqUBo       |
| project_id | b2fc7cb72ed64a9db085976af6421515                               |
| user_id    | d153971cfa4a4a0e854f8dd709ac3c2e                               |
+------------+----------------------------------------------------------------+
```

Aquesta comanda utilitza el port 5000 perquè es el port que només permet l'accés regular (a usuaris no administradors).

2.7.2.1.8 Creació d'entorns de clients d'OpenStack mitjançant scripts

A la secció anterior hem utilitzat una combinació de variables d'entorn amb diferents comandes per a interactuar amb el servei identity a través del client OpenStack dins del mateix node. Per a augmentar la eficiència de les operacions del client, OpenStack suporta una serie d'scripts d'entorn del client també coneguts com a fitxers OpenRC. Aquests Scripts normalment contenen opcions comuns per a tots els clients, però també permeten opcions úniques.

A continuació farem una demostració d'una creació d'scripts per als usuaris i projectes admin i demo. En un futur, referenciarem aquests scripts per a carregar els credencials apropiats per les operacions dels clients.

1. Creem el fitxer **admin-openrc** dins de la nostre carpeta personal i afegim el següent contingut:

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin
export OS_AUTH_URL=http://controller:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

2. Creem el fitxer **demo-openrc** dins de la nostre carpeta personal i afegim el següent contingut:

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=demo
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

Un cop els hem creat, els posarem en pràctica. Per a executar els clients com a un projecte i usuari especific podem carregar l'script a l'entorn del client abans d'executar-ho.

```
root@controller:~$ . admin-openrc
```

```
root@controller:~$ . demo-openrc
```

Ara ja podem demanar un token d'autenticació per a tots els nostres usuaris definits.

```
root@controller:~# openstack token issue
+------------+----------------------------------------------------------------------+
| Field      | Value                                                                |
+------------+----------------------------------------------------------------------+
| expires    | 2017-05-07 19:05:22+00:00                                            |
| id         | gAAAAABZD2Hi07bVsX-eeZoLzqZRtrHIhNkZWta5SCrH1WZuQxBWxIdRCHWeVat |
|            | Yjpxk2Jugj_QuIrphnOALqIJxq7a2tBMUhaCiWI-OypwhHeg6cGlUaU-         |
|            | Ox7yBHqaoaYN5-b61Iac0g_FpXNI15fljt-_XnxWMKpQ7LALQPvYoZOHe9w-    |
|            | fmjg                                                                 |
| project_id | b2fc7cb72ed64a9db085976af6421515                                     |
| user_id    | d153971cfa4a4a0e854f8dd709ac3c2e                                     |
+------------+----------------------------------------------------------------------+
```

Amb això hem finalitzat l'explicació de la instal·lació, configuració i ús del mòdul KeyStone.

2.7.2.2 Glance (Servei de Gestió d'Imatges)

Glance es el servei que utilitza OpenStack per a proporcionar registres, descobriments i entrega d'imatges de màquines virtuals. Disposa d'una API que li permet consultar metadades d'imatges de màquines virtuals i recuperar la imatge actual. Permet emmagatzemar imatges de màquines virtuals disponibles a través del servei d'imatges a una varietat d'ubicacions, des de sistemes de fitxers simples fins a sistemes d'emmagatzematge d'objectes.

Les imatges emmagatzemades es poden utilitzar com a plantilles per a un ràpid desplegament de nous servidors sense haver de realitzar la instal·lació del sistema operatiu. A més, permet ser utilitzat per a emmagatzemar un número il·limitat de backups.

2.7.2.2.1 Coneixent Glance

Les imatges gestionades per Glance es poden trobar en diferents estats, que hem de conèixer per tal de entendre aquest mòdul:

- **Queued**: Indica que l'identificador de la imatge ha sigut reservat en el registre del servei Glance però no s'han afegit/especificat les dades de la imatge.
- **Saving**: Indica que les dades de la imatge s'estan afegint en aquest moment al servei Glance.
- **Active**: Indica que la imatge està disponible/accessible al servei Glance. Això significa que les dades de la imatge s'han afegit al servei o bé que la mida de la imatge s'ha establert en 0 durant la creació.
- **Killed**: Indica que ha hagut un error durant l'afegiment de les dades de la imatge al servei i que no es accessible.

- **Deleted**: Indica que el servei Glance ja no disposa de la informació de la imatge i que no està disponible. A més, s'esborrarà automàticament per complert en poc temps.

- **Pending_delete**: Es molt similar al estat Deleted, però aquest indica que el servei Glance encara no ha esborrat les dades de la imatge però que està pendent de fer-ho i per tant la imatge es recuperable.

2.7.2.2.2 Instal·lació de Glance

Primerament, hem de crear i configurar la base de dades que requereix aquest mòdul (com vam dir anteriorment, tots els mòduls necessiten una), on desarà la informació.

Node Controller

```
root@controller:~$ mysql -u root -p
```

```
MariaDB [(none)]> CREATE DATABASE glance;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY 'contrasenyaEscollida';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'192.168.1.1' IDENTIFIED BY 'contrasenyaEscollida';

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

On 'contrasenyaEscollida' serà la contrasenya que volem posar a la base de dades.
Un cop fet això, sortim del client de la base de dades.

Ara necessitem utilitzar els credencials de l'usuari Admin que vam crear anteriorment per a accedir a OpenStack mitjançant el servei identity de KeyStone.

Anem al directori personal (on vam crear l'script d'autenticació) i executem el de l'admin de manera que quedi a les variables d'entorn.

```
root@controller:~$ . admin-openrc
```

Ara crearem l'usuari glance i afegirem el projecte en el que treballarà que vam crear anteriorment amb l'usuari Admin i Demo. A més, crearem el servei glance i els endpoints d'API per aquest servei :

1. Creem l'usuari glance:

```
root@controller:~$ openstack user create --domain default --password-prompt glance
```

```
root@controller:~# openstack user create --domain default --password-prompt glan
ce
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | 82d7ae268d914dc8999bf852fcf3dd6e |
| enabled             | True                             |
| id                  | 08222d9af4ae4d58b5c243b4ca5a5306 |
| name                | glance                           |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

2. Afegim el rol i el projecte que vam crear anomenats admin a l'usuari glance.

```
root@controller:~$ openstack role add --project service --user glance admin
```

3. Creem la entitat de servei glance.

```
root@controller:~$ openstack service create --name glance --description "OpenStack Image" image
```

```
root@controller:~# openstack service create --name glance --description "OpenSta
ck Image" image
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Image                  |
| enabled     | True                             |
| id          | 13d70c4cf40348509c6d477d55851a06 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+
```

4. Creem els tres endpoints d'API per al servei creat.

   Endpoint Public:

```
root@controller:~$ openstack endpoint create --region RegionOne image public http://controller:9292
```

```
root@controller:~# openstack endpoint create --region RegionOne image public htt
p://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | d431eaf99aae4012a7400c95849ad1d8 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 13d70c4cf40348509c6d477d55851a06 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

   Endpoint Internal:

```
root@controller:~$ openstack endpoint create --region RegionOne image internal http://controller:9292
```

```
root@controller:~# openstack endpoint create --region RegionOne image internal h
ttp://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 99e5768884e942f5bc9bbb93ddc05d57 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 13d70c4cf40348509c6d477d55851a06 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

Endpoint Admin:

```
root@controller:~$ openstack endpoint create --region RegionOne image admin
http://controller:9292
```

```
root@controller:~# openstack endpoint create --region RegionOne image admin http
://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 24613e82ff5143e4968865124aeac2d9 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 13d70c4cf40348509c6d477d55851a06 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

Un cop fet això, instal·lem el paquet necessari per a utilitzar aquest mòdul.

```
root@controller:~$ apt -y install glance
```

2.7.2.2.3 Configuració de Glance

Per a configurar Glance, primerament anirem al fitxer **/etc/glance/flance-api.conf.**
Un cop endins, ens hem de moure per diferents seccions per tal de fer funcional aquest mòdul:

1. Anem a la secció [database] i configurem l'accés del servei a la base de dades afegint la següent línia:

```
connection = mysql+pymysql://glance:contrasenyaEscollida@controller/glance
```

```
[database]

#
# From oslo.db
#
connection = mysql+pymysql://glance:contrasenyaEscollida@controller/glance
```

2. Després, a la secció [keystone_authtoken] configurem la integració de glance amb el servei identity de KeyStone per a la autenticació, on l'username i el password son els escollits anteriorment quan hem creat l'usuari glance.

```
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = glance
```

```
[keystone_authtoken]

#
# From keystonemiddleware.auth_token
#


auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = glance
```

3. A la secció [paste_deploy] li indiquem a glance que el flavor (sabor, concepte que vam mencionar a l'apartat de la instal·lació de DevStack) es keystone.

```
flavor = keystone
```

```
[paste_deploy]

#
# From glance.api
#
flavor = keystone
```

4. A la secció [glance_store] configurem el directori en el que s'emmagatzemaran les imatges de les màquines virtuals.

```
flavor = keystone
```

```
[glance_store]

#
# From glance.store
#
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

Ara editarem el fitxer **/etc/glance/glance-registry.conf.** De igual manera anirem de secció en secció per a configurar el fitxer similarment a l'anterior.

1. A la secció [database] afegim la configuració de la connexió a la base de dades.

```
connection = mysql+pymysql://glance:contrasenyaEscollida@controller/glance
```

```
[database]

#
# From oslo.db
#
connection = mysql+pymysql://glance:contrasenyaEscollida@controller/glance
```

2. A la secció [keystone_authtoken] configurem l'accés del servei amb keystone.

```
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = glance
```

```
[keystone_authtoken]

#
# From keystonemiddleware.auth_token
#

auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = glance
```

3. A la secció [paste_deploy] definim el flavor (sabor) de keystone.

```
flavor = keystone
```

```
[paste_deploy]

#
# From glance.registry
#

flavor = keystone
```

Un cop fet això, omplim la base de dades amb la informació per defecte de glance sincronitzant-ho per a que generi la estructura de les taules.

```
root@controller:~$ su -s /bin/sh -c "glance-manage db_sync" glance
```

```
root@controller:~# su -s /bin/sh -c "glance-manage db_sync" glance
/usr/lib/python2.7/dist-packages/oslo_db/sqlalchemy/enginefacade.py:1171: OsloDB
DeprecationWarning: EngineFacade is deprecated; please use oslo_db.sqlalchemy.en
ginefacade
  expire_on_commit=expire_on_commit, _conf=conf)
2017-05-08 12:20:11.216 4317 INFO migrate.versioning.api [-] 0 -> 1...
2017-05-08 12:20:11.218 4317 INFO glance.db.sqlalchemy.migrate_repo.schema [-] c
reating table images
2017-05-08 12:20:11.777 4317 INFO migrate.versioning.api [-] done
2017-05-08 12:20:11.777 4317 INFO migrate.versioning.api [-] 1 -> 2...
2017-05-08 12:20:11.780 4317 INFO glance.db.sqlalchemy.migrate_repo.schema [-] c
reating table image_properties
2017-05-08 12:20:12.544 4317 INFO migrate.versioning.api [-] done
2017-05-08 12:20:12.544 4317 INFO migrate.versioning.api [-] 2 -> 3...
2017-05-08 12:20:13.732 4317 INFO migrate.versioning.api [-] done
2017-05-08 12:20:13.733 4317 INFO migrate.versioning.api [-] 3 -> 4...
2017-05-08 12:20:14.132 4317 INFO migrate.versioning.api [-] done
2017-05-08 12:20:14.132 4317 INFO migrate.versioning.api [-] 4 -> 5...
2017-05-08 12:20:14.662 4317 INFO migrate.versioning.api [-] done
2017-05-08 12:20:14.663 4317 INFO migrate.versioning.api [-] 5 -> 6...
```

Per a finalitzar la instal·lació, reiniciem els serveis (no fa falta habilitar-los perquè ja ho estan per defecte).

```
root@controller:~$ systemctl restart glance-registry.service

root@controller:~$ systemctl restart glance-registry.service
```

Realitzem una comprovació de que els serveis estiguin executant-se sense cap problema i donem per finalitzada la instal·lació.

```
root@controller:~# systemctl status glance-api.service
● glance-api.service - OpenStack Image Service API
   Loaded: loaded (/lib/systemd/system/glance-api.service; enabled; vendor prese
   Active: active (running) since lun 2017-05-08 12:26:10 CEST; 1min 38s ago
  Process: 4781 ExecStartPre=/bin/chown glance:adm /var/log/glance (code=exited,
  Process: 4777 ExecStartPre=/bin/chown glance:glance /var/lock/glance /var/lib/
  Process: 4773 ExecStartPre=/bin/mkdir -p /var/lock/glance /var/log/glance /var
 Main PID: 4784 (glance-api)
    Tasks: 3
   Memory: 100.8M
      CPU: 2.339s
   CGroup: /system.slice/glance-api.service
           ├─4784 /usr/bin/python /usr/bin/glance-api --config-file=/etc/glance/
           ├─4797 /usr/bin/python /usr/bin/glance-api --config-file=/etc/glance/
           └─4798 /usr/bin/python /usr/bin/glance-api --config-file=/etc/glance/

may 08 12:26:11 controller glance-api[4784]:   return pkg_resources.EntryPoint.p
may 08 12:26:11 controller glance-api[4784]: /usr/lib/python2.7/dist-packages/pa
may 08 12:26:11 controller glance-api[4784]:   return pkg_resources.EntryPoint.p
may 08 12:26:11 controller glance-api[4784]: /usr/lib/python2.7/dist-packages/os
may 08 12:26:11 controller glance-api[4784]:   debtcollector.deprecate('Multiple
may 08 12:26:11 controller glance-api[4784]: 2017-05-08 12:26:11.397 4784 INFO g
```

```
root@controller:~# systemctl status glance-registry.service
● glance-registry.service - OpenStack Image Service Registry
   Loaded: loaded (/lib/systemd/system/glance-registry.service; enabled; vendor
   Active: active (running) since lun 2017-05-08 12:26:05 CEST; 1min 9s ago
  Process: 4735 ExecStartPre=/bin/chown glance:adm /var/log/glance (code=exited,
  Process: 4731 ExecStartPre=/bin/chown glance:glance /var/lock/glance /var/lib/
  Process: 4726 ExecStartPre=/bin/mkdir -p /var/lock/glance /var/log/glance /var
 Main PID: 4739 (glance-registry)
    Tasks: 3
   Memory: 99.5M
      CPU: 943ms
   CGroup: /system.slice/glance-registry.service
           ├─4739 /usr/bin/python /usr/bin/glance-registry --config-file=/etc/gl
           ├─4753 /usr/bin/python /usr/bin/glance-registry --config-file=/etc/gl
           └─4754 /usr/bin/python /usr/bin/glance-registry --config-file=/etc/gl

may 08 12:26:05 controller glance-registry[4739]:   return pkg_resources.EntryPo
may 08 12:26:05 controller glance-registry[4739]: /usr/lib/python2.7/dist-packag
may 08 12:26:05 controller glance-registry[4739]:   return pkg_resources.EntryPo
may 08 12:26:05 controller glance-registry[4739]: /usr/lib/python2.7/dist-packag
may 08 12:26:05 controller glance-registry[4739]:   return pkg_resources.EntryPo
may 08 12:26:05 controller glance-registry[4739]: 2017-05-08 12:26:05.977 4739 I
```

2.7.2.2.4 Verificació de les operacions realitzades al servei Glance

A continuació verificarem que el servei glance funciona correctament:

1. Agafem els credencials de l'usuari Admin a les variables d'entorn mitjançant l'script creat anteriorment per garantir l'accés a les comandes de l'admin.

```
root@controller:~$ . admin-openrc
```

2. Descarreguem una imatge d'un cirrOS (una distribució de linux molt petita per a fer test de núvols).

```
root@controller:~$ wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
```

3. Carreguem la imatge dins del servei amb un format de disc QCOW2 i una visibilitat pública per a que tots els projectes puguin accedir a ella.

```
root@controller:~$ openstack image create "cirros" --file cirros-0.3.4-x86_64-disk.img --disk-format qcow2 --container-format bare --public
```

```
usuario@controller:~$ openstack image create "cirros" --file cirros-0.3.4-x86_64
-disk.img --disk-format qcow2 --container-format bare --public
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | ee1eca47dc88f4879d8a229cc70a07c6                     |
| container_format | bare                                                 |
| created_at       | 2017-05-08T16:41:52Z                                 |
| disk_format      | qcow2                                                |
| file             | /v2/images/463faac1-cb0e-49ff-84fb-76fa364405d1/file |
| id               | 463faac1-cb0e-49ff-84fb-76fa364405d1                 |
| min_disk         | 0                                                    |
| min_ram          | 0                                                    |
| name             | cirros                                               |
| owner            | 3df3a9e2446541868520ab7cab1d5933                     |
| protected        | False                                                |
| schema           | /v2/schemas/image                                    |
| size             | 13287936                                             |
| status           | active                                               |
| tags             |                                                      |
| updated_at       | 2017-05-08T16:41:53Z                                 |
| virtual_size     | None                                                 |
| visibility       | public                                               |
+------------------+------------------------------------------------------+
```

4. Per confirmar que la imatge s'ha carregat correctament, utilitzarem la següent comanda:

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

```
usuario@controller:~$ openstack image list
+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| 463faac1-cb0e-49ff-84fb-76fa364405d1 | cirros | active |
+--------------------------------------+--------+--------+
```

2.7.2.3 Nova (Servei de computació)

Nova es el servei que utilitza OpenStack per allotjar i gestionar el sistema de computació al núvol. Aquest mòdul significa la major part de la infraestructura del sistema com a servei (IaaS).

2.7.2.3.1 Coneixent Nova

OpenStack Nova interactua amb OpenStack Identity per a la autenticació, amb OpenStack Glance per agafar les imatges i OpenStack Horizon (Dashboard) per a tenir una interfície de usuari i administrador.

L'accés a les imatges està limitat per els projectes i por els usuaris i les quotes estan limitades per el projecte (el numero de instancies per exemple).

El servei Nova es pot escalar horitzontalment en un hardware estàndard i pot descarregar imatges per a iniciar instancies.

A més, es molt important saber que Nova no es un hypervisor sinó que es qui gestiona els recursos dels hipervisors com per exemple de KVM.

2.7.2.3.2 Instal·lació de Nova

Primerament, hem de crear i configurar la base de dades que requereix aquest mòdul.

Node Controller

```
root@controller:~$ mysql -u root -p
```

```
MariaDB [(none)]> CREATE DATABASE nova_api;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' IDENTIFIED BY 'contrasenyaEscollida';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'192.168.1.1' IDENTIFIED BY 'contrasenyaEscollida';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' IDENTIFIED BY 'contrasenyaEscollida';



MariaDB [(none)]> CREATE DATABASE nova;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'contrasenyaEscollida';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'192.168.1.1' IDENTIFIED BY 'contrasenyaEscollida';

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

On 'contrasenyaEscollida' serà la contrasenya que volem posar a la base de dades. Un cop fet això, sortim del client de la base de dades.

Ara necessitem utilitzar els credencials de l'usuari Admin que vam crear anteriorment per a accedir a OpenStack mitjançant el servei identity de KeyStone.

Anem al directori personal (on vam crear l'script d'autenticació) i executem el de l'admin de manera que quedi a les variables d'entorn.

```
root@controller:~$ . admin-openrc
```

Ara crearem l'usuari nova i afegirem el projecte en el que treballarà el qual vam crear anteriorment amb l'usuari Admin i Demo. A més, crearem el servei nova i els endpoints d'API per aquest servei :

1. Creem l'usuari nova:

```
root@controller:~$ openstack user create --domain default --password-prompt nova
```

```
root@controller:~# openstack user create --domain default --password-prompt nova
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | 82d7ae268d914dc8999bf852fcf3dd6e |
| enabled             | True                             |
| id                  | 3046a5cee66548d496773cd1d2dd199b |
| name                | nova                             |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

2. Afegim el rol i el projecte admin a l'usuari nova.

```
root@controller:~$ openstack role add --project service --user nova admin
```

3. Creem la entitat de servei nova.

```
root@controller:~$ openstack service create --name nova --description "OpenStack Compute" compute
```

```
root@controller:~# openstack service create --name nova --description "OpenStack
 Compute" compute
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Compute                |
| enabled     | True                             |
| id          | 631463ceeb154a5e903964e87dcfe2ec |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+
```

5. Creem els tres endpoints d'API per al servei creat.

Endpoint Public:

```
root@controller:~$ openstack endpoint create --region RegionOne compute public http://controller:8774/v2.1/%\(tenant_id\)s
```

```
root@controller:~# openstack endpoint create --region RegionOne compute public h
ttp://controller:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 5543f93a8deb431f8a3f3568d7795a5a        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 631463ceeb154a5e903964e87dcfe2ec        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+
```

Endpoint Internal:

```
root@controller:~$ openstack endpoint create --region RegionOne compute
internal http://controller:8774/v2.1/%\(tenant_id\)s
```

```
root@controller:~# openstack endpoint create --region RegionOne compute internal
 http://controller:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 9d49b89623f84ee5bb7512b117c30e90        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 631463ceeb154a5e903964e87dcfe2ec        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+
```

Endpoint Admin:

```
root@controller:~$ openstack endpoint create --region RegionOne compute
admin http://controller:8774/v2.1/%\(tenant_id\)s
```

```
root@controller:~# openstack endpoint create --region RegionOne compute admin ht
tp://controller:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 2be7806598f4405d81adb5a49fe89293        |
| interface    | admin                                   |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 631463ceeb154a5e903964e87dcfe2ec        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+
```

Un cop fet això, instal·lem el paquet necessari per a utilitzar aquest mòdul.

```
root@controller:~$ apt -y install nova-api nova-conductor nova-console nova-novncproxy nova-scheduler
```

2.7.2.3.3 Configuració de Nova

Per a configurar Nova, primerament anirem al fitxer **/etc/nova/nova.conf.**
Un cop endins, ens hem de moure per diferents seccions per tal de fer funcional aquest mòdul:

1. Anem a la secció [DEFAULT] i habilitem només las APIs compute i metadata indicant la següent línia.

```
enabled_apis = osapi_compute,metadata
```

```
[DEFAULT]
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
log-dir=/var/log/nova
state_path=/var/lib/nova
force_dhcp_release=True
verbose=True
ec2_private_dns_show_ip=True
enabled_apis=osapi_compute,metadata
```

Després, configurem l'accés a la cua de missatges RabbitMQ indicant-li l'usuari i contrasenya definits anteriorment a la seva instal·lació.

```
transport_url = rabbit://openstack:openstack@controller
```

Un cop realitzat, configurem el servei d'autenticació identity

```
auth_strategy = keystone
```

Després, configurem la opció «my_ip» per a utilitzar la direcció IP de la interfície de xarxa del node controlador. A més, habilitem el suport per al servei Networking.

```
my_ip = 192.168.1.1

use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

La secció [DEFAULT] hauria de quedar així:

```
[DEFAULT]
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
log-dir=/var/log/nova
state_path=/var/lib/nova
force_dhcp_release=True
verbose=True
ec2_private_dns_show_ip=True
enabled_apis=osapi_compute,metadata
transport_url = rabbit://openstack:openstack@controller
auth_strategy = keystone
my_ip = 192.168.1.1
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

2. A les seccions [api_database] i [database] configurem l'accés a la base de dades del servei nova.

[api_database]

```
connection = mysql+pymysql://nova:contrasenyaEscollida@controller/nova_api
```

[database]

```
connection = mysql+pymysql://nova:contrasenyaEscollida@controller/nova
```

```
[database]
#connection=sqlite:////var/lib/nova/nova.sqlite
connection = mysql+pymysql://nova:contrasenyaEscollida@controller/nova_api
[api_database]
#connection=sqlite:////var/lib/nova/nova.sqlite
connection = mysql+pymysql://nova:contrasenyaEscollida@controller/nova
```

3. Anem a la secció [keystone_authtoken] i finalitzem la configuració del servei d'autenticació identity on substituirem el paràmetre NOVA_PASS per la contrasenya que hem definit per a l'usuari nova.

```
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
```

```
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = nova
```

4. A la secció [vnc] configurem el proxy VNC per a utilitzar la direcció IP de la interfície de xarxa del node del controlador (utilitzant la variable definida anteriorment com a my_ip).

```
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip
```

```
[vnc]
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip
```

5. A la secció [glance] configurem a on buscarà el servei glance d'imatges.

```
api_servers = http://controller:9292
```

```
[glance]
api_servers = http://controller:9292
```

6. A la secció [oslo_concurrency] configurem el lock path.

```
lock_path = /var/lib/nova/tmp
```

```
[oslo_concurrency]
lock_path = /var/lib/nova/tmp
```

Un cop fet això, omplim les bases de dades amb la informació per defecte de Nova sincronitzant-ho per a que generi la estructura de les taules.

```
root@controller:~$ su -s /bin/sh -c "nova-manage api_db sync" nova

root@controller:~$ su -s /bin/sh -c "nova-manage db sync" nova
```

Per finalitzar reiniciem els serveis i els habilitem si no ho estiguessin.

```
systemctl restart nova-api.service nova-consoleauth.service nova-scheduler.service nova-conductor.service nova-novncproxy.service
```

I comprovem que estan funcionant sense cap error.

```
root@controller:~# systemctl status nova-api.service
● nova-api.service - OpenStack Compute API
   Loaded: loaded (/lib/systemd/system/nova-api.service; enabled; vendor preset:
   Active: active (running) since mar 2017-05-09 11:21:57 CEST; 37s ago
  Process: 28947 ExecStartPre=/bin/chown nova:adm /var/log/nova (code=exited, st
  Process: 28943 ExecStartPre=/bin/chown nova:nova /var/lock/nova /var/lib/nova
  Process: 28941 ExecStartPre=/bin/mkdir -p /var/lock/nova /var/log/nova /var/li
 Main PID: 28948 (nova-api)
    Tasks: 5
   Memory: 267.5M
      CPU: 3.944s
   CGroup: /system.slice/nova-api.service
           ├─28948 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nov
           ├─29009 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nov
           ├─29010 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nov
           ├─29017 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nov
           └─29018 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nov

may 09 11:22:01 controller nova-api[28948]: 2017-05-09 11:22:01.848 29010 INFO n
may 09 11:22:01 controller nova-api[28948]: 2017-05-09 11:22:01.856 29009 INFO n
may 09 11:22:02 controller sudo[29011]: pam_unix(sudo:session): session closed f
may 09 11:22:02 controller sudo[29014]:     nova : TTY=unknown ; PWD=/var/lib/no
```

```
root@controller:~# systemctl status nova-consoleauth.service
● nova-consoleauth.service - OpenStack Compute Console
   Loaded: loaded (/lib/systemd/system/nova-consoleauth.service; enabled; vendor
   Active: active (running) since mar 2017-05-09 11:21:56 CEST; 1min 9s ago
  Process: 28888 ExecStartPre=/bin/chown nova:adm /var/log/nova (code=exited, st
  Process: 28885 ExecStartPre=/bin/chown nova:nova /var/lock/nova /var/lib/nova
  Process: 28883 ExecStartPre=/bin/mkdir -p /var/lock/nova /var/log/nova /var/li
 Main PID: 28894 (nova-consoleaut)
    Tasks: 1
   Memory: 117.9M
      CPU: 2.068s
   CGroup: /system.slice/nova-consoleauth.service
           └─28894 /usr/bin/python /usr/bin/nova-consoleauth --config-file=/etc/

may 09 11:21:56 controller systemd[1]: Stopped OpenStack Compute Console.
may 09 11:21:56 controller systemd[1]: Starting OpenStack Compute Console...
may 09 11:21:56 controller systemd[1]: Started OpenStack Compute Console.
may 09 11:22:00 controller nova-consoleauth[28894]: Option "verbose" from group
may 09 11:22:00 controller nova-consoleauth[28894]: 2017-05-09 11:22:00.198 2889
may 09 11:22:00 controller nova-consoleauth[28894]: 2017-05-09 11:22:00.222 2889
```

```
root@controller:~# systemctl status nova-scheduler.service
● nova-scheduler.service - OpenStack Compute Scheduler
   Loaded: loaded (/lib/systemd/system/nova-scheduler.service; enabled; vendor p
   Active: active (running) since mar 2017-05-09 11:22:00 CEST; 1min 36s ago
  Process: 28989 ExecStartPre=/bin/chown nova:adm /var/log/nova (code=exited, st
  Process: 28984 ExecStartPre=/bin/chown nova:nova /var/lock/nova /var/lib/nova
  Process: 28982 ExecStartPre=/bin/mkdir -p /var/lock/nova /var/log/nova /var/li
 Main PID: 28994 (nova-scheduler)
    Tasks: 1
   Memory: 118.5M
      CPU: 2.042s
   CGroup: /system.slice/nova-scheduler.service
           └─28994 /usr/bin/python /usr/bin/nova-scheduler --config-file=/etc/no

may 09 11:22:00 controller systemd[1]: Stopped OpenStack Compute Scheduler.
may 09 11:22:00 controller systemd[1]: Starting OpenStack Compute Scheduler...
may 09 11:22:00 controller systemd[1]: Started OpenStack Compute Scheduler.
may 09 11:22:02 controller nova-scheduler[28994]: Option "verbose" from group "D
may 09 11:22:02 controller nova-scheduler[28994]: 2017-05-09 11:22:02.754 28994
may 09 11:22:02 controller nova-scheduler[28994]: 2017-05-09 11:22:02.909 28994
```

```
root@controller:~# systemctl status nova-conductor.service
● nova-conductor.service - OpenStack Compute Conductor
   Loaded: loaded (/lib/systemd/system/nova-conductor.service; enabled; vendor p
   Active: active (running) since mar 2017-05-09 11:21:57 CEST; 2min 13s ago
  Process: 28917 ExecStartPre=/bin/chown nova:adm /var/log/nova (code=exited, st
  Process: 28911 ExecStartPre=/bin/chown nova:nova /var/lock/nova /var/lib/nova
  Process: 28904 ExecStartPre=/bin/mkdir -p /var/lock/nova /var/log/nova /var/li
 Main PID: 28925 (nova-conductor)
    Tasks: 3
   Memory: 184.4M
      CPU: 3.677s
   CGroup: /system.slice/nova-conductor.service
           ├─28925 /usr/bin/python /usr/bin/nova-conductor --config-file=/etc/no
           ├─29003 /usr/bin/python /usr/bin/nova-conductor --config-file=/etc/no
           └─29004 /usr/bin/python /usr/bin/nova-conductor --config-file=/etc/no

may 09 11:21:57 controller systemd[1]: Stopped OpenStack Compute Conductor.
may 09 11:21:57 controller systemd[1]: Starting OpenStack Compute Conductor...
may 09 11:21:57 controller systemd[1]: Started OpenStack Compute Conductor.
may 09 11:22:00 controller nova-conductor[28925]: Option "verbose" from group "D
may 09 11:22:00 controller nova-conductor[28925]: 2017-05-09 11:22:00.764 28925
may 09 11:22:00 controller nova-conductor[28925]: 2017-05-09 11:22:00.785 28925
```

```
root@controller:~# systemctl status nova-novncproxy.service
● nova-novncproxy.service - OpenStack Compute novncproxy
   Loaded: loaded (/lib/systemd/system/nova-novncproxy.service; enabled; vendor
   Active: active (running) since mar 2017-05-09 11:21:57 CEST; 2min 33s ago
  Process: 28922 ExecStartPre=/bin/chown nova:adm /var/log/nova (code=exited, st
  Process: 28909 ExecStartPre=/bin/chown nova:nova /var/lock/nova /var/lib/nova
  Process: 28903 ExecStartPre=/bin/mkdir -p /var/lock/nova /var/log/nova /var/li
 Main PID: 28932 (nova-novncproxy)
    Tasks: 1
   Memory: 113.0M
      CPU: 1.234s
   CGroup: /system.slice/nova-novncproxy.service
           └─28932 /usr/bin/python /usr/bin/nova-novncproxy --config-file=/etc/n

may 09 11:21:57 controller systemd[1]: Starting OpenStack Compute novncproxy...
may 09 11:21:57 controller systemd[1]: Started OpenStack Compute novncproxy.
may 09 11:21:58 controller nova-novncproxy[28932]: Option "verbose" from group "
may 09 11:21:59 controller nova-novncproxy[28932]: 2017-05-09 11:21:59.568 28932
may 09 11:21:59 controller nova-novncproxy[28932]: 2017-05-09 11:21:59.569 28932
may 09 11:21:59 controller nova-novncproxy[28932]: 2017-05-09 11:21:59.569 28932
may 09 11:21:59 controller nova-novncproxy[28932]: 2017-05-09 11:21:59.569 28932
may 09 11:21:59 controller nova-novncproxy[28932]: 2017-05-09 11:21:59.569 28932
```

2.7.2.3.4 Configuració de Nova amb un node compute

OpenStack pot funcionar amb diferents nodes, no només el controller. En aquesta secció començarem a utilitzar un altre. Instal·larem i configurarem el servei nova (compute) de manera que pugui suportar desplegaments d'instàncies i de màquines virtuals.

Node Compute

Per començar, instal·larem els paquets necessaris:

```
root@compute:~$ apt -y install nova-compute
```

Després editarem el fitxer **/etc/nova/nova.conf** i anirem per les diferents seccions:

1. A la secció [DEFAULT] configurarem de manera similar al que hem explicat al node controller pel que no explicarem que fa cada cosa (ja està explicat). Afegirem les següents línies.

```
enabled_apis = osapi_compute,metadata
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

On «MANAGEMENT_INTERFACE_IP_ADDRESS» indicarem la ip que vam asignar al nostre node compute.

```
[DEFAULT]
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova
force_dhcp_release=True
libvirt_use_virtio_for_bridges=True
verbose=True
ec2_private_dns_show_ip=True
api_paste_config=/etc/nova/api-paste.ini
enabled_apis=ec2,osapi_compute,metadata
transport_url = rabbit://openstack:openstack@controller
auth_strategy = keystone
my_ip = 192.168.1.2
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

2. A la secció [keystone_authtoken] integrem nova amb keystone.

```
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
```

On NOVA_PASS es la contrasenya escollida per a l'usuari nova.

```
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = nova
```

3. A la secció [vnc] configurem l'accés remot per consola.

```
enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html
```

```
[vnc]
enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html
```

4. A la secció [glance] l'indiquem al servei nova la localització de glance (servei d'imatges)

```
api_servers = http://controller:9292
```

```
[glance]
api_servers = http://controller:9292
```

5. A la secció [oslo_concurrency] configurem el lock path.

```
lock_path = /var/lib/nova/tmp
```

```
[oslo_concurrency]
lock_path = /var/lib/nova/tmp
```

Un cop fet, el següent es indicar-li al nostre node compute quin tipus de virtualització va a utilitzar. Le indicarem que utilitzarem KVM (QEMU). Per això el primer que tenim que fer es verificar que el nostre node compute suporta la acceleració per hardware:

```
root@compute:~$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

```
root@compute:~# egrep -c '(vmx|svm)' /proc/cpuinfo
2
```

Si el valor que retorna es 1 o més gran significa que el teu node compute suporta l'acceleració per hardwre, el qual no requereix cap configuració addicional.

Per finalitzar reiniciem els serveis de nova-compute i de libvirt i comprovem que funcionen sense cap error.

```
root@compute:~$ systemctl restart libvirtd.service nova-compute.service
```

```
root@compute:~# systemctl status libvirtd
● libvirt-bin.service - Virtualization daemon
   Loaded: loaded (/lib/systemd/system/libvirt-bin.service; enabled; vendor pres
   Active: active (running) since mar 2017-05-09 12:05:33 CEST; 7s ago
     Docs: man:libvirtd(8)
           http://libvirt.org
 Main PID: 7678 (libvirtd)
    Tasks: 16
   Memory: 6.1M
      CPU: 152ms
   CGroup: /system.slice/libvirt-bin.service
           └─7678 /usr/sbin/libvirtd

may 09 12:05:33 compute systemd[1]: Starting Virtualization daemon...
may 09 12:05:33 compute systemd[1]: Started Virtualization daemon.
```

```
root@compute:~# systemctl status nova-compute.service
● nova-compute.service - OpenStack Compute
   Loaded: loaded (/lib/systemd/system/nova-compute.service; enabled; vendor pre
   Active: active (running) since mar 2017-05-09 12:05:33 CEST; 32s ago
  Process: 7701 ExecStartPre=/bin/chown nova:nova /var/lock/nova /var/log/nova /
  Process: 7699 ExecStartPre=/bin/mkdir -p /var/lock/nova /var/log/nova /var/lib
 Main PID: 7706 (nova-compute)
    Tasks: 22
   Memory: 132.8M
      CPU: 1.864s
   CGroup: /system.slice/nova-compute.service
           └─7706 /usr/bin/python /usr/bin/nova-compute --config-file=/etc/nova/

may 09 12:05:35 compute nova-compute[7706]:   </guest>
may 09 12:05:35 compute nova-compute[7706]: </capabilities>
may 09 12:05:35 compute nova-compute[7706]: 2017-05-09 12:05:35.489 7706 ERROR n
may 09 12:05:35 compute nova-compute[7706]: 2017-05-09 12:05:35.492 7706 WARNING
may 09 12:05:35 compute nova-compute[7706]: 2017-05-09 12:05:35.492 7706 INFO no
may 09 12:05:35 compute nova-compute[7706]: 2017-05-09 12:05:35.523 7706 WARNING
may 09 12:05:35 compute nova-compute[7706]: 2017-05-09 12:05:35.581 7706 INFO no
may 09 12:05:35 compute nova-compute[7706]: 2017-05-09 12:05:35.631 7706 INFO no
may 09 12:05:35 compute nova-compute[7706]: 2017-05-09 12:05:35.632 7706 INFO no
may 09 12:05:35 compute nova-compute[7706]: 2017-05-09 12:05:35.689 7706 INFO no
```

2.7.2.3.5 Verificació de les operacions realitzades al servei Nova

Node Controller

Per a verificar que el servei Nova funciona i que està connectat entre els dos nodes el primer que hem de fer es carregar els credencials de l'usuari admin amb l'script que vam crear.

```
root@controller:~$ . admin-openrc
```

Després, llistem els serveis dels que disposa el nostre mòdul nova i veurem que el nostre node compute (l'ultim de tots) està habilitat i funcionant.

```
root@controller:~# openstack compute service list
+----+--------------+------------+----------+---------+-------+--------------+
| ID | Binary       | Host       | Zone     | Status  | State | Updated At   |
+----+--------------+------------+----------+---------+-------+--------------+
|  5 | nova-        | controller | internal | enabled | up    | 2017-05-09T1 |
|    | conductor    |            |          |         |       | 0:09:07.0000 |
|    |              |            |          |         |       | 00           |
|  6 | nova-        | controller | internal | enabled | up    | 2017-05-09T1 |
|    | scheduler    |            |          |         |       | 0:09:08.0000 |
|    |              |            |          |         |       | 00           |
|  7 | nova-        | controller | internal | enabled | up    | 2017-05-09T1 |
|    | consoleauth  |            |          |         |       | 0:09:06.0000 |
|    |              |            |          |         |       | 00           |
|  8 | nova-compute | compute    | nova     | enabled | up    | 2017-05-09T1 |
|    |              |            |          |         |       | 0:09:10.0000 |
|    |              |            |          |         |       | 00           |
+----+--------------+------------+----------+---------+-------+--------------+
```

2.7.2.4 Neutron (Servei de xarxa)

Neutron es el mòdul d'OpenStack que permet crear i connectar dispositius de xarxa administrats per altres serveis d'OpenStack a les xarxes. A més, es poden implementar plugins per acomodar diferents equips de xarxa i software proporcionant flexibilitat a la arquitectura i el desplegament d'OpenStack. En resum, es qui s'encarrega de gestionar les xarxes a OpenStack(les xarxes de les màquines virtuals dins d'OpenStack, per exemple).

2.7.2.4.1 Coneixent Neutron

OpenStack Neutron gestiona totes las facetes de la xarxa per al VNI (Virtual Networking Infraestructure) i de les facetes de la xarxa per a PNI (Physical Networking Infraestructure) per a l'entorn d'OpenStack.

Neutron ens permet habilita projectes per a crear topologies de xarxes virtuals que inclouen elements com firewalls, load balancers i VPN's(virtual private network). A més, proporciona xarxes, subnets i routers com a abstracció d'objectes. Cada abstracció te una funcionalitat que imita la seva contrapart física: les xarxes contenen subxarxes i els encaminadors encaminen el tràfic entre diferents subxarxes i xarxes.

Qualsevol configuració de Neutron te almenys una xarxa externa. A diferencia de les altres xarxes, la externa no es només una xarxa virtualment definida sinó que representa una vista en una part de la xarxa física externa que es accessible des de fora de la instal·lació d'OpenStack. Això vol dir que les direccions IP a la xarxa externa son accessibles per a qualsevol físicament a la xarxa externa.

A més de les xarxes externes, qualsevol configuració de xarxa te una o més xarxes internes. Això significa que cada xarxa definida per software es connecta directament a les màquines virtuals. Només les màquines virtuals d'una xarxa interna o de les subxarxes connectades mitjançant un encaminador poden accedir directament a les màquines virtuals connectades a aquesta xarxa.

Un cop sabem això, hem de conèixer que per a que una xarxa externa accedeixi a les màquines virtuals i viceversa, es necessiten routers entre les xarxes. Cada encaminador te una passarel·la que està connectada a una xarxa externa i a una o més connectades a xarxes internes. De igual manera que un encaminador físic, les subxarxes poden accedir a les màquines d'altres subxarxes que estiguin connectats al mateix encaminador, i les màquines virtuals poden accedir a la xarxa externa a través de la porta d'enllaç del encaminador.

A més, es poden assignar Ips de xarxes externes als ports de la xarxa interna. Es poden associar direccions IP com a ports a màquines virtuals per a que els clients puguin accedir a les màquines virtuals des de fora.

Les xarxes també admeten grups de seguretat. Els grups de seguretat permeten als administradors definir regles de firewall en diferents grups. Una màquina virtual pot pertànyer a un o més grups de seguretat i Neutron aplica les regles per a aquests grups per bloquejar o desbloquejar ports, tipus de tràfics acceptat per a aquestes màquines virtuals...

2.7.2.4.2 Instal·lació de Neutron

Primerament, hem de crear i configurar la base de dades que requereix aquest mòdul on desarà la informació.

Node Controller

```
root@controller:~$ mysql -u root -p
```

```
MariaDB [(none)]> CREATE DATABASE neutron;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY 'contrasenyaEscollida';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'192.168.1.1' IDENTIFIED BY 'contrasenyaEscollida';

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

On 'contrasenyaEscollida' serà la contrasenya que volem posar a la base de dades.
Un cop fet això, sortim del client de la base de dades.

Ara necessitem utilitzar els credencials de l'usuari Admin que vam crear anteriorment per a accedir a OpenStack mitjançant el servei identity de KeyStone.

Anem al directori personal (on vam crear l'script d'autenticació) i executem el de l'admin de manera que quedi a les variables d'entorn.

```
root@controller:~$ . admin-openrc
```

Creem l'usuari neutron i afegirem el projecte en el que treballarà que vam crear anteriorment amb l'usuari Admin i Demo. A més, crearem el servei neutron i els endpoints d'API per aquest servei :

1. Creem l'usuari neutron

```
root@controller:~$ openstack user create --domain default --password-prompt neutron
```

```
root@controller:~# openstack user create --domain default --password-prompt neut
ron
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | 82d7ae268d914dc8999bf852fcf3dd6e |
| enabled             | True                             |
| id                  | b2375d5565d44467abfc2f0ceead49aa |
| name                | neutron                          |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

2. Afegim el rol i el projecte a l'usuari neutron.

```
root@controller:~$ openstack role add --project service --user neutron admin
```

3. Creem la entitat del servei neutron.

```
root@controller:~$ openstack service create --name neutron --description
"OpenStack Networking" network
```

```
root@controller:~# openstack service create --name neutron --description "OpenSt
ack Networking" network
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Networking             |
| enabled     | True                             |
| id          | 4c7dd80699a84b5ab35410702b5c5a33 |
| name        | neutron                          |
| type        | network                          |
+-------------+----------------------------------+
```

4. Creem els tres endpoints d'API per al servei creat.

Endpoint Public:

```
root@controller:~$ openstack endpoint create --region RegionOne network
public http://controller:9696
```

```
root@controller:~# openstack endpoint create --region RegionOne network public h
ttp://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | e8bad5d3f4154e0589f901ca312234af |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 4c7dd80699a84b5ab35410702b5c5a33 |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

Endpoint Internal:

```
root@controller:~$ openstack endpoint create --region RegionOne network
internal http://controller:9696
```

```
root@controller:~# openstack endpoint create --region RegionOne network internal
 http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 97ef9217d6b544799ab3db5a8c6d4226 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 4c7dd80699a84b5ab35410702b5c5a33 |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

Endpoint Admin

```
root@controller:~$ openstack endpoint create --region RegionOne network
admin http://controller:9696
```

```
root@controller:~# openstack endpoint create --region RegionOne network admin ht
tp://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 820f7963f47c41fd9d51984e27549478 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 4c7dd80699a84b5ab35410702b5c5a33 |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

Un cop hem arribat a aquest punt, haurem de decidir el camí a seguir. El servei neutron disposa de dues arquitectures a utilitzar:

1. **Provider networks**: Es la configuració més bàsica, la qual només permet afegir instancies a la xarxa de xarxa externa pel que només l'usuari admin pot gestionar la infraestructura.

2. **Self-service networks**: Es la configuració més completa, la qual disposa d'un ampli abast de possibilitats, es a dir, cada usuari pot gestionar la seva pròpia infraestructura de xarxa creant routers, subxarxes...i la possibilitat de generar IPs flotants amb les que proveir l'accés a internet a les màquines virtuals.

Nosaltres escollim la opció 2, pel que instal·lem els següents paquets:

```
root@controller:~$ apt -y install neutron-server neutron-plugin-ml2 neutron-linuxbridge-agent neutron-dhcp-agent neutron-l3-agent
```

2.7.2.4.3 Configuració de Neutron

Per a configurar Neutron, primerament anirem al fitxer **/etc/neutron/neutron.conf.**
Un cop endins, ens hem de moure per diferents seccions per tal de fer funcional aquest mòdul:

1. Anem a la secció [database] i configurem l'accés a la base de dades:

```
connection = mysql+pymysql://neutron:contrasenyaEscollida@controller/neutron
```

```
[database]

#
# From neutron.db
#
Connection = mysql+pymysql://neutron:contrasenyaEscollida@controller/neutron
```

2. Anem a la secció [DEFAULT] i habilitem el plugin Modular Layer 2, el plugin del servei de routers i el de solapament d'IPs

```
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
```

Després configurem el RabbitMQ:

```
transport_url = rabbit://openstack:openstack@controller
```

I integrem keystone al mòdul Neutron:

```
auth_strategy = keystone
```

A continuació configurem neutron per a notificar els canvis a la topologia de la xarxa de nova (compute).

```
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
```

Un cop hem fet això, la secció [DEFAULT] deuria de quedar així:

```
[DEFAULT]

#
# From neutron
#
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True

transport_url = rabbit://openstack:openstack@controller

auth_strategy = keystone

notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
```

3. Anem a la secció [keystone_authtoken] i configurem l'accés al servei identity (KeyStone).

```
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = NEUTRON_PASS
```

```
[keystone_authtoken]

#
# From keystonemiddleware.auth_token
#
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = neutron
```

4. Anem a la secció [nova] i configurem l'accés al servei nova (Compute).

```
auth_url = http://controller:35357
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
```

5. Anem a la secció [oslo_concurrency] i configurem el lock_path.

```
lock_path = /var/lib/neutron/tmpc
```

```
[oslo_concurrency]

#
# From oslo.concurrency
#
lock_path = /var/lib/neutron/tmp
```

Desem el fitxer i ja haurem fet la configuració inicial.

2.7.2.4.3.1 Configuració del plug-in Modular Layer 2(ML2)

El plug-in ML2 utilitza el mecanisme de bridge de linux per a construir infraestructures de xarxes virtuals per a les instancies (bridging i switching). Per a realitzar la configuració editem el fitxer **/etc/neutron/plugins/ml2/ml2_conf.ini:**

1. Anem a la secció **[ml2]** i habilitem el flat, les VLANs i les VXLANs.

```
type_drivers = flat,vlan,vxlan
```

Habilitem les VXLAN per a la xarxa self-service.

```
tenant_network_types = vxlan
```

Habilitem el bridge de linux amb els mecanismes de layer-2 population.

```
mechanism_drivers = linuxbridge,l2population
```

Habilitem el driver port_security.

```
extension_drivers = port_security
```

Un cop fet, el fitxer deuria quedar així:

```
[ml2]

#
# From neutron.ml2
#
type_drivers = flat,vlan,vxlan
tenant_network_types = vxlan
mechanism_drivers = linuxbridge,l2population
extension_drivers = port_security
```

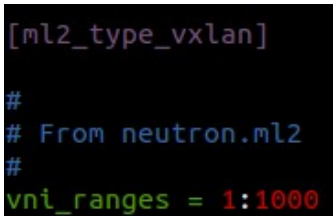
2. A la secció **[ml2_type_flat]** configurem el provider virtual.

```
flat_networks = provider
```

```
[ml2_type_flat]

#
# From neutron.ml2
#
flat_networks = provider
```

3. A la secció **[ml2_type_vxlan]** especifiquem el rang de xarxes per a la xarxa serlf-service, es a dir, el nombre de subxarxes possibles.

```
vni_ranges = 1:1000
```



4. A la secció [securitygroup] habilitem l'ipset per a incrementar la eficiència de les regles als grups de seguretat.

   Ipset es una extensió d'iptables que permet la creació de regles de firewall que responen a conjunts sencers de direccions ip de forma simultània. Aquests conjunts estan a les estructures de dades, indexats per augmentar l'eficiència sobretot en sistemes amb una gran quantitat de regles.

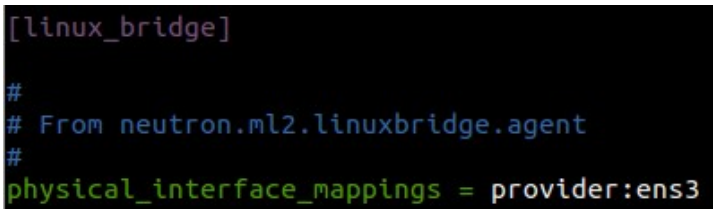
Desem el fitxer i sortim.

2.7.2.4.3.2 Configuració de l'agent Linux Bridge

L'agent Linux Bridge (bridging and switching) construeix una infraestructura de xarxes virtuals layer-2 per a instancies i gestionar grups de seguretat. Per a la seva configuració editem el fitxer **/etc/neutron/plugins/ml2/linuxbridge_agent.ini:**

1. A la secció **[linux_bridge]**, realitzem un mapeig entre la interfície física (la nostre tarja provider, que te accés a internet, ens3) i el provider virtual.

```
physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME
```



2. A la secció **[vxlan]** habilitem les xarxes VXLAN, configurem la IP de la interfície de xarxa física que s'encarregarà de gestionar-les (la no provider, ens7).

```
enable_vxlan = True
local_ip = OVERLAY_INTERFACE_IP_ADDRESS
l2_population = True
```

Reemplacem «OVERLAY_INTERFACE_IP_ADDRESS» per la adreça IP, al nostre cas es ens7 a la que vam assignar una IP estàtica per a que es comuniqui sense problemes amb els diferents nodes i serveis sense que la ip canvii.

```
[vxlan]

#
# From neutron.ml2.linuxbridge.agent
#
enable_vxlan = True
local_ip = 192.168.1.1
l2_population = True
```

3. A la secció **[securitygroup]** habilitem els grups de seguretat i configurem el driver del firewall iptables per al Linux Bridge.

```
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

```
[securitygroup]

#
# From neutron.ml2.linuxbridge.agent
#
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Desem el fitxer i sortim.

2.7.2.4.3.3 Configuració de l'agent layer-3

L'agent Layer-3 proveeix serveis d'encaminament i NAT per a les xarxes virtuals self-service. Per a la seva configuració editem el fitxer **/etc/neutron/l3_agent.ini**:

1. A la secció **[DEFAULT]** configurem el driver de Linux Bridge i el pont amb la xarxa externa.

```
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
```

```
[DEFAULT]

#
# From neutron.base.agent
#
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
```

2.7.2.4.3.4 Configuració de l'agent DHCP

L'agent DHCP proveeix un servei DHCP per a les xarxes virtuals. Per a la seva configuració editem el fitxer **/etc/neutron/dhcp_agent.ini**:

1. A la secció **[DEFAULT]** configurem el driver de Linux Bridge, el de Dnsmasq DHCP i habilitem la opció enable_isolated_metadata la qual permet a les instàncies accedir al servidor de metadades mitjançant la xarxa.

```
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = True
```

```
[DEFAULT]

#
# From neutron.base.agent
#
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = True
```

2.7.2.4.3.5 Configuració de l'agent metadata

L'agent metadata proveeix informació de configuració a les instancies. Per configurar-lo editem el fitxer **/etc/neutron/metadata_agent.ini:**

1. A la secció [**DEFAULT]** configurem el servidor de metadades.

```
nova_metadata_ip = controller
metadata_proxy_shared_secret = METADATA_SECRET
```

```
[DEFAULT]

#
# From neutron.metadata.agent
#
nova_metadata_ip = controller
metadata_proxy_shared_secret = METADATA_SECRET
```

S'ha d'escollir un valor per al secret de les metadades.

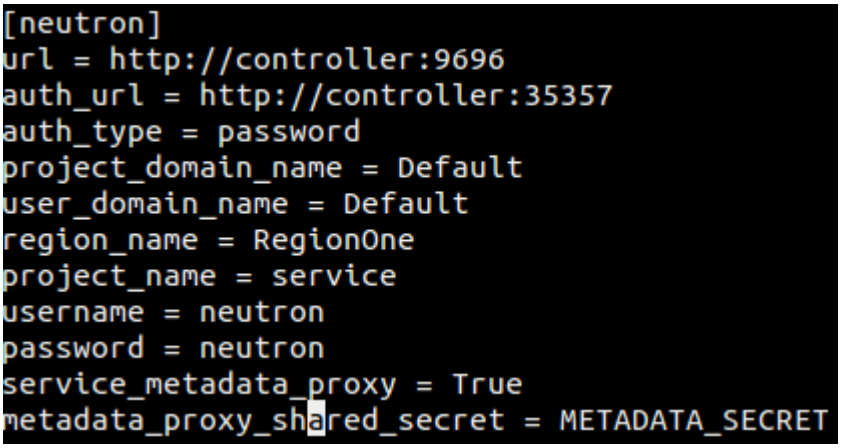
2.7.2.4.3.6 Configuració per integrar Nova amb Neutron

Per a integrar el servei Nova amb Neutron editarem el fitxer **/etc/nova/nova.conf:**

1. A la secció [neutron] configurem els paràmetres d'accés i habilitem el proxy de metadades.

```
url = http://controller:9696
auth_url = http://controller:35357
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
```

   Reemplacem el secret de les metadades i la contrasenya de neutron per els valors que vam definir anteriorment.



Para la inicialització del servei neutron utilitzarem una serie d'scripts amb els que primer farem uns enllaços simbòlics en cas de que no existeixin.

```
root@controller:~$ ln -s /etc/neutron/plugins/ml2/ml2_conf.ini
/etc/neutron/plugin.ini
```

Sincronitzem la base de dades.

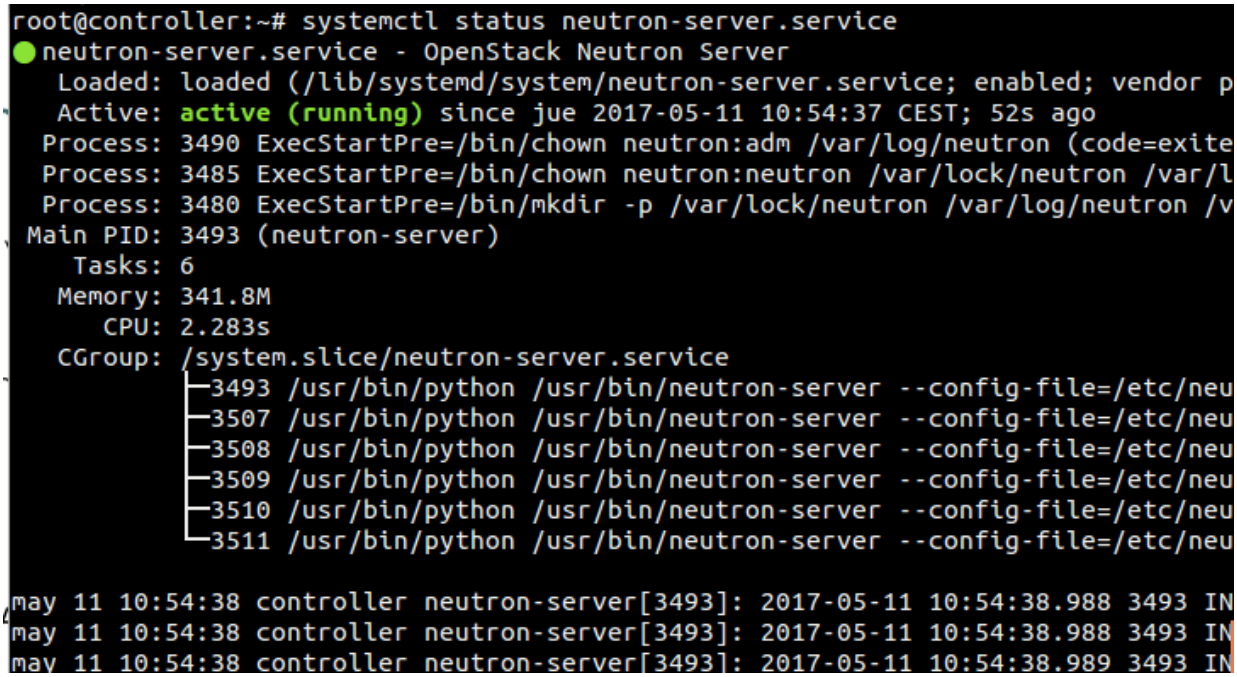
```
root@controller:~$ su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

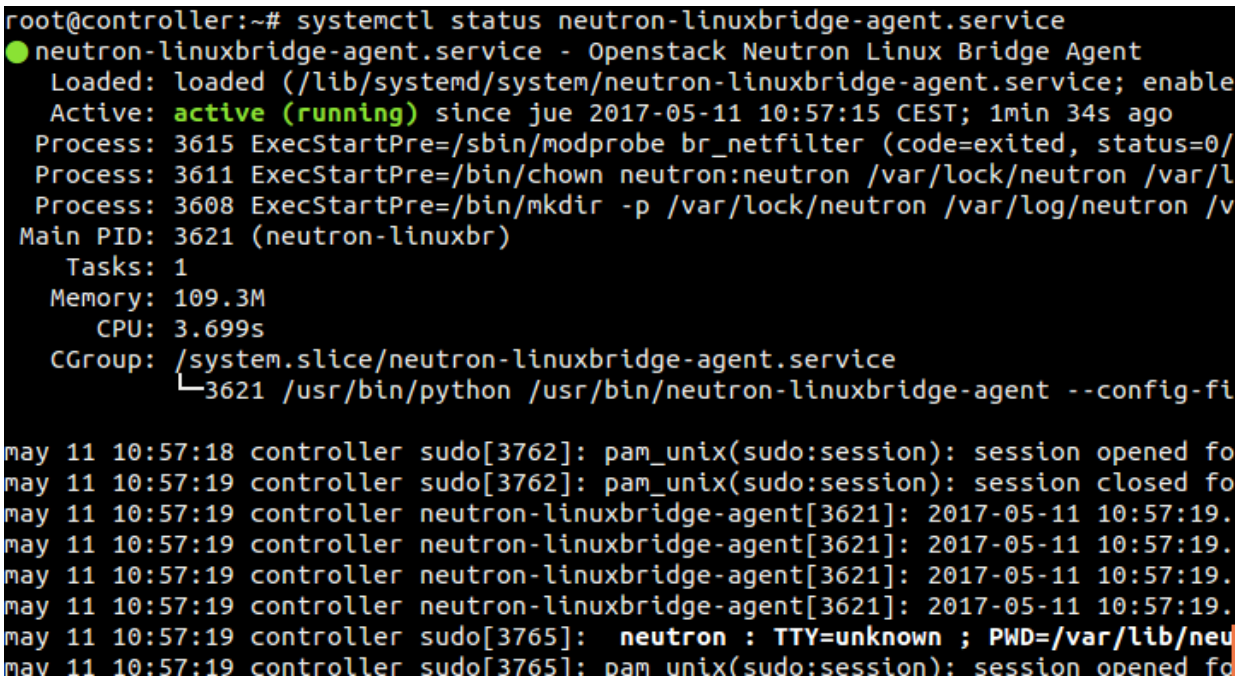
Reiniciem el servei nova-api.

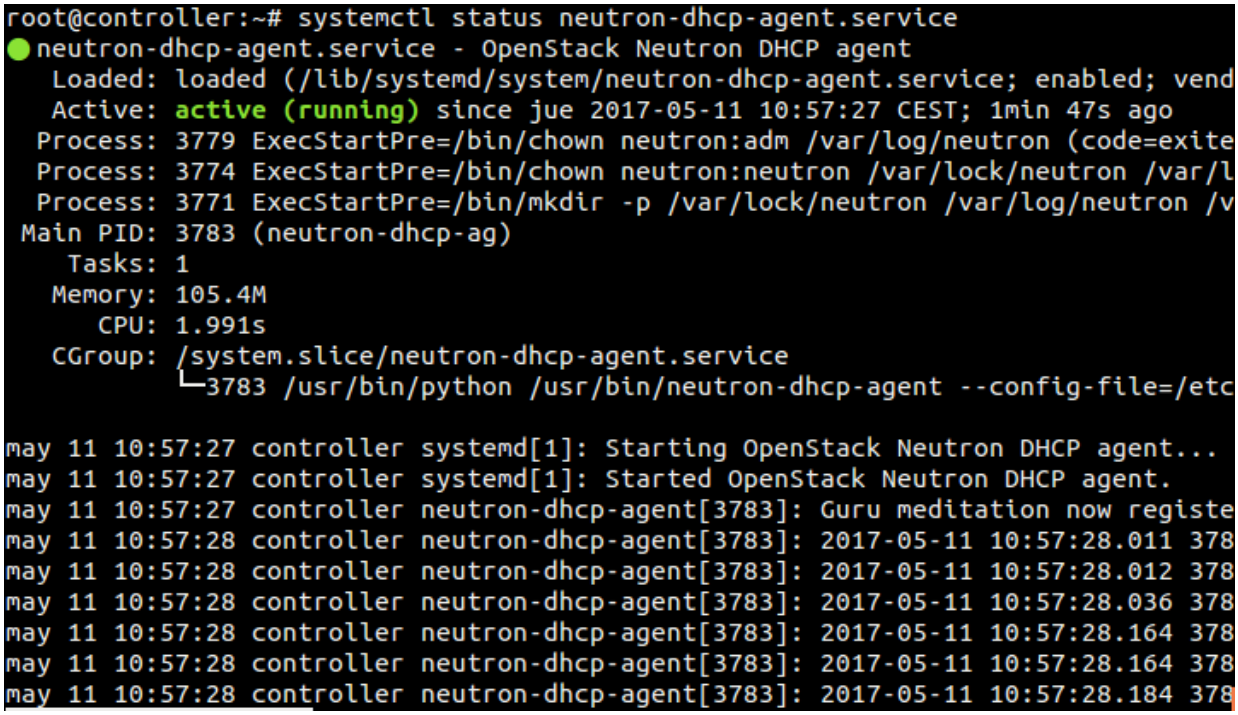
```
root@controller:~$ systemctl restart nova-api.service
```

Reiniciem els servei de neutron, i si cal, ho habilitem (per defecte ve habilitat) i comprovem que funcionen sense cap error.

```
root@controller:~$ systemctl restart neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp-agent.service neutron-metadata-agent.service neutron-l3-agent.service
```




root@controller:~# systemctl status neutron-linuxbridge-agent.service
neutron-linuxbridge-agent.service - Openstack Neutron Linux Bridge Agent
Loaded: loaded (/lib/systemd/system/neutron-linuxbridge-agent.service; enable
Active: active (running) since jue 2017-05-11 10:57:15 CEST; 1min 34s ago
Process: 3615 ExecStartPre=/sbin/modprobe br_netfilter (code=exited, status=0/
Process: 3611 ExecStartPre=/bin/chown neutron:neutron /var/lock/neutron /var/l
Process: 3608 ExecStartPre=/bin/mkdir -p /var/lock/neutron /var/log/neutron /v
Main PID: 3621 (neutron-linuxbr)
Tasks: 1
Memory: 109.3M
CPU: 3.699s
CGroup: /system.slice/neutron-linuxbridge-agent.service
└─3621 /usr/bin/python /usr/bin/neutron-linuxbridge-agent --config-fi
may 11 10:57:18 controller sudo[3762]: pam_unix(sudo:session): session opened fo
may 11 10:57:19 controller sudo[3762]: pam_unix(sudo:session): session closed fo
may 11 10:57:19 controller neutron-linuxbridge-agent[3621]: 2017-05-11 10:57:19.
may 11 10:57:19 controller neutron-linuxbridge-agent[3621]: 2017-05-11 10:57:19.
may 11 10:57:19 controller neutron-linuxbridge-agent[3621]: 2017-05-11 10:57:19.
may 11 10:57:19 controller neutron-linuxbridge-agent[3621]: 2017-05-11 10:57:19.
may 11 10:57:19 controller sudo[3765]: neutron : TTY=unknown ; PWD=/var/lib/neu


root@controller:~# systemctl status neutron-dhcp-agent.service
neutron-dhcp-agent.service - OpenStack Neutron DHCP agent
Loaded: loaded (/lib/systemd/system/neutron-dhcp-agent.service; enabled; vend
Active: active (running) since jue 2017-05-11 10:57:27 CEST; 1min 47s ago
Process: 3779 ExecStartPre=/bin/chown neutron:adm /var/log/neutron (code=exite
Process: 3774 ExecStartPre=/bin/chown neutron:neutron /var/lock/neutron /var/l
Process: 3771 ExecStartPre=/bin/mkdir -p /var/lock/neutron /var/log/neutron /v
Main PID: 3783 (neutron-dhcp-ag)
Tasks: 1
Memory: 105.4M
CPU: 1.991s
CGroup: /system.slice/neutron-dhcp-agent.service
└─3783 /usr/bin/python /usr/bin/neutron-dhcp-agent --config-file=/etc
may 11 10:57:27 controller systemd[1]: Starting OpenStack Neutron DHCP agent...
may 11 10:57:27 controller systemd[1]: Started OpenStack Neutron DHCP agent.
may 11 10:57:27 controller neutron-dhcp-agent[3783]: Guru meditation now registe
may 11 10:57:28 controller neutron-dhcp-agent[3783]: 2017-05-11 10:57:28.011 378
may 11 10:57:28 controller neutron-dhcp-agent[3783]: 2017-05-11 10:57:28.012 378
may 11 10:57:28 controller neutron-dhcp-agent[3783]: 2017-05-11 10:57:28.036 378
may 11 10:57:28 controller neutron-dhcp-agent[3783]: 2017-05-11 10:57:28.164 378
may 11 10:57:28 controller neutron-dhcp-agent[3783]: 2017-05-11 10:57:28.164 378
may 11 10:57:28 controller neutron-dhcp-agent[3783]: 2017-05-11 10:57:28.184 378

```
root@controller:~# systemctl status neutron-metadata-agent.service
● neutron-metadata-agent.service - OpenStack Neutron Metadata Agent
   Loaded: loaded (/lib/systemd/system/neutron-metadata-agent.service; enabled;
   Active: active (running) since jue 2017-05-11 10:57:16 CEST; 2min 22s ago
  Process: 3663 ExecStartPre=/bin/chown neutron:adm /var/log/neutron (code=exite
  Process: 3659 ExecStartPre=/bin/chown neutron:neutron /var/lock/neutron /var/l
  Process: 3657 ExecStartPre=/bin/mkdir -p /var/lock/neutron /var/log/neutron /v
 Main PID: 3666 (neutron-metadat)
    Tasks: 2
   Memory: 114.6M
      CPU: 1.114s
   CGroup: /system.slice/neutron-metadata-agent.service
           ├─3666 /usr/bin/python /usr/bin/neutron-metadata-agent --config-file=
           └─3746 /usr/bin/python /usr/bin/neutron-metadata-agent --config-file=

may 11 10:57:16 controller systemd[1]: Stopped OpenStack Neutron Metadata Agent.
may 11 10:57:16 controller systemd[1]: Starting OpenStack Neutron Metadata Agent
may 11 10:57:16 controller systemd[1]: Started OpenStack Neutron Metadata Agent.
may 11 10:57:16 controller neutron-metadata-agent[3666]: Guru meditation now reg
may 11 10:57:17 controller neutron-metadata-agent[3666]: 2017-05-11 10:57:17.809
may 11 10:57:17 controller neutron-metadata-agent[3666]: 2017-05-11 10:57:17.810
may 11 10:57:17 controller neutron-metadata-agent[3666]: 2017-05-11 10:57:17.819
may 11 10:57:17 controller neutron-metadata-agent[3666]: 2017-05-11 10:57:17.840
```

```
root@controller:~# systemctl status neutron-l3-agent.service
● neutron-l3-agent.service - OpenStack Neutron L3 agent
   Loaded: loaded (/lib/systemd/system/neutron-l3-agent.service; enabled; vendor
   Active: active (running) since jue 2017-05-11 11:16:41 CEST; 48s ago
  Process: 4065 ExecStartPre=/bin/chown neutron:adm /var/log/neutron (code=exite
  Process: 4061 ExecStartPre=/bin/chown neutron:neutron /var/lock/neutron /var/l
  Process: 4057 ExecStartPre=/bin/mkdir -p /var/lock/neutron /var/log/neutron /v
 Main PID: 4068 (neutron-l3-agen)
    Tasks: 1
   Memory: 106.1M
      CPU: 1.341s
   CGroup: /system.slice/neutron-l3-agent.service
           └─4068 /usr/bin/python /usr/bin/neutron-l3-agent --config-file=/etc/n

may 11 11:16:41 controller systemd[1]: Started OpenStack Neutron L3 agent.
may 11 11:16:42 controller neutron-l3-agent[4068]: Guru meditation now registers
may 11 11:16:42 controller neutron-l3-agent[4068]: 2017-05-11 11:16:42.815 4068
may 11 11:16:42 controller neutron-l3-agent[4068]: 2017-05-11 11:16:42.816 4068
may 11 11:16:42 controller neutron-l3-agent[4068]: 2017-05-11 11:16:42.824 4068
may 11 11:16:42 controller neutron-l3-agent[4068]: 2017-05-11 11:16:42.853 4068
may 11 11:16:42 controller neutron-l3-agent[4068]: 2017-05-11 11:16:42.874 4068
may 11 11:16:42 controller neutron-l3-agent[4068]: 2017-05-11 11:16:42.889 4068
```

2.7.2.4.4 Instal·lació i configuració de Neutron al node Compute

Node Compute

Instal·lem els components necessaris:

```
root@compute:~$ apt -y install neutron-linuxbridge-agent ebtables ipset
```

Per a configurar Neutron al nostre node Compute, primerament anirem al fitxer **/etc/neutron/neutron.conf** Un cop endins, ens hem de moure per diferents seccions per tal de fer funcional aquest mòdul:

1. A la secció **[database]** comentem qualsevol línia relacionada amb qualsevol connexió perquè els nodes compute no poden tenir accés directament a la base de dades, per seguretat.

2. A la secció **[DEFAULT]** configurem RabbitMQ amb la següent línia.

```
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

Després integrem neutron amb KeyStone com a servei d'autenticació per l'accés.

```
auth_strategy = keystone
```

```
[DEFAULT]

#
# From neutron
#
transport_url = rabbit://openstack:openstack@controller
auth_strategy = keystone
```

3. A la secció **[keystone_authtoken]** configurem l'accés a KeyStone.

```
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = NEUTRON_PASS
```

```
[keystone_authtoken]

#
# From keystonemiddleware.auth_token
#
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = neutron
```

4. A la secció **[oslo_concurrency]** configurem el lock path.

```
lock_path = /var/lib/neutron/tmp
```

```
[oslo_concurrency]

#
# From oslo.concurrency
#
lock_path = /var/lib/neutron/tmp
```

2.7.2.4.4.1 Configuració de l'agent Linux Bridge

Ara hem d'escollir la mateixa opció sobre la arquitectura que hem escollit al node controller, pel que farem la següent configuració editant el fitxer **/etc/neutron/plugins/ml2/linuxbridge_agent.ini** :

1. A la secció **[linux_bridge]**, realitzem un mapeig entre la interfície física (la nostre tarja provider, que te accés a internet, ens3) i el provider virtual.

```
physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME
```

```
[linux_bridge]

#
# From neutron.ml2.linuxbridge.agent
#
physical_interface_mappings = provider:ens3
```

2. A la secció **[vxlan]** habilitem les xarxes VXLAN, configurem la IP de la interfície de xarxa física que s'encargarà de gestionarles (la no provider, ens7).

```
enable_vxlan = True
local_ip = OVERLAY_INTERFACE_IP_ADDRESS
l2_population = True
```

3. Reemplacem «OVERLAY_INTERFACE_IP_ADDRESS» per la adreça IP, al nostre cas es ens7 a la que vam assignar una IP estàtica per a que es comuniqui sense problemes amb els diferents nodes i serveis sense que la IP canvii.

```
[vxlan]

#
# From neutron.ml2.linuxbridge.agent
#
enable_vxlan = True
local_ip = 192.168.1.2
l2_population = True
```

4. A la secció **[securitygroup]** habilitem els grups de seguretat i configurem el driver del firewall iptables per al Linux Bridge.

```
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

```
[securitygroup]

#
# From neutron.ml2.linuxbridge.agent
#
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

2.7.2.4.4.2 Configuració per integrar Nova amb Neutron al node Compute

Per a integrar nova amb neutron realitzarem els mateixos passos que al node controller editant el fitxer de configuració **/etc/nova/nova.conf**:

1. Afegim la següent secció **[neutron]**:

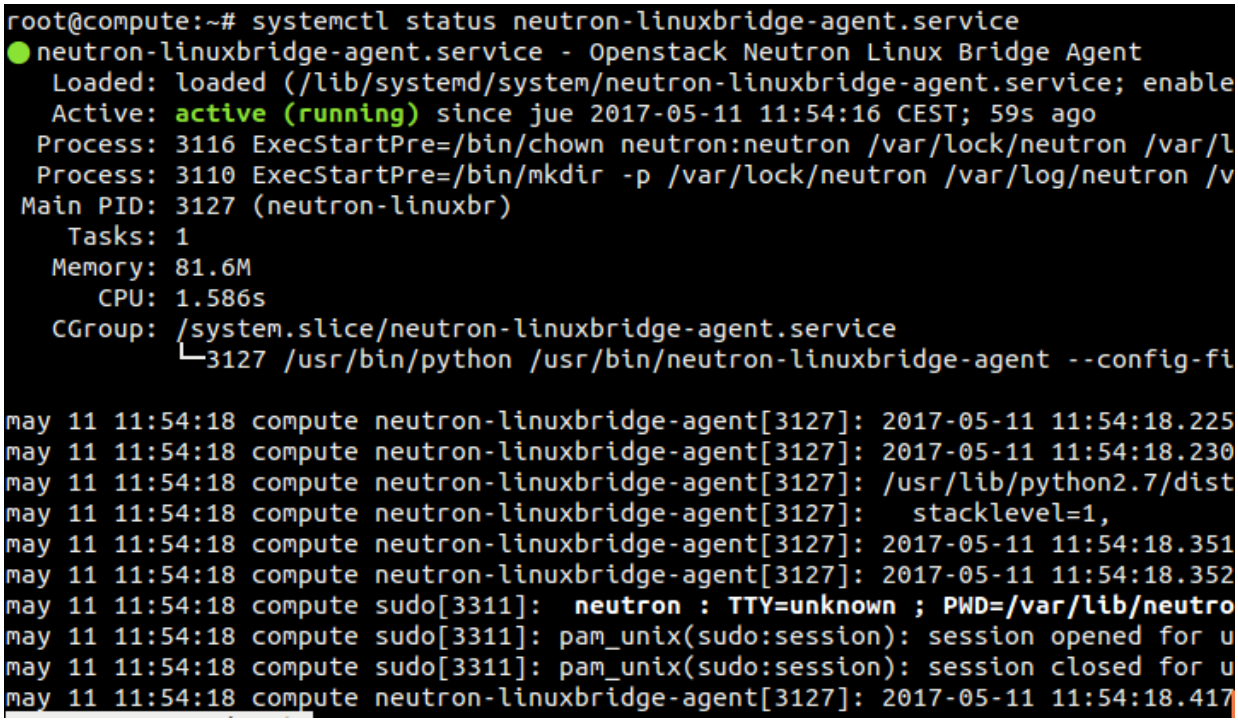
```
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
```

```
[neutron]
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = neutron
```

Per finalitzar, reiniciem el servei de nova i els de neutron, comprovant que funcionen sense cap problema.

```
root@compute:~$ systemctl restart nova-compute.service neutron-linuxbridge-agent.service
```

```
root@compute:~# systemctl status nova-compute.service
● nova-compute.service - OpenStack Compute
   Loaded: loaded (/lib/systemd/system/nova-compute.service; enabled; vendor pre
   Active: active (running) since jue 2017-05-11 11:54:31 CEST; 8s ago
  Process: 3318 ExecStartPre=/bin/chown nova:nova /var/lock/nova /var/log/nova /
  Process: 3315 ExecStartPre=/bin/mkdir -p /var/lock/nova /var/log/nova /var/lib
 Main PID: 3323 (nova-compute)
    Tasks: 22
   Memory: 131.0M
      CPU: 1.939s
   CGroup: /system.slice/nova-compute.service
           └─3323 /usr/bin/python /usr/bin/nova-compute --config-file=/etc/nova/

may 11 11:54:33 compute nova-compute[3323]:     <features>
may 11 11:54:33 compute nova-compute[3323]:       <disksnapshot default='on' tog
may 11 11:54:33 compute nova-compute[3323]:     </features>
may 11 11:54:33 compute nova-compute[3323]:   </guest>
may 11 11:54:33 compute nova-compute[3323]: </capabilities>
may 11 11:54:33 compute nova-compute[3323]: 2017-05-11 11:54:33.827 3323 WARNING
may 11 11:54:33 compute nova-compute[3323]: 2017-05-11 11:54:33.828 3323 INFO no
may 11 11:54:33 compute nova-compute[3323]: 2017-05-11 11:54:33.906 3323 INFO no
may 11 11:54:33 compute nova-compute[3323]: 2017-05-11 11:54:33.906 3323 INFO no
may 11 11:54:33 compute nova-compute[3323]: 2017-05-11 11:54:33.970 3323 INFO no
```



2.7.2.4.5 Verificació de les operacions realitzades al servei Neutron

Per a verificar que neutron funciona correctament als dos nodes farem el següent:

Node Controller

1. Carreguem els credencials d'autenticació de l'admin a les variables d'entorn.

```
root@controller:~$ . admin-openrc
```

2. Llistem les extensions carregades per a verificar el llançament dels processos del servei neutron.

```
root@controller:~$ neutron ext-list
```

```
root@controller:~# neutron ext-list
+---------------------------+-----------------------------------------------+
| alias                     | name                                          |
+---------------------------+-----------------------------------------------+
| default-subnetpools       | Default Subnetpools                           |
| network-ip-availability   | Network IP Availability                       |
| network_availability_zone | Network Availability Zone                     |
| auto-allocated-topology   | Auto Allocated Topology Services              |
| ext-gw-mode               | Neutron L3 Configurable external gateway mode |
| binding                   | Port Binding                                  |
| agent                     | agent                                         |
| subnet_allocation         | Subnet Allocation                             |
| l3_agent_scheduler        | L3 Agent Scheduler                            |
| tag                       | Tag support                                   |
| external-net              | Neutron external network                      |
| flavors                   | Neutron Service Flavors                       |
| net-mtu                   | Network MTU                                   |
| availability_zone         | Availability Zone                             |
| quotas                    | Quota management support                      |
| l3-ha                     | HA Router extension                           |
| provider                  | Provider Network                              |
| multi-provider            | Multi Provider Network                        |
| address-scope             | Address scope                                 |
| extraroute                | Neutron Extra Route                           |
| subnet-service-types      | Subnet service types                          |
| standard-attr-timestamp   | Resource timestamps                           |
| service-type              | Neutron Service Type Management               |
| l3-flavors                | Router Flavor Extension                       |
| port-security             | Port Security                                 |
| extra_dhcp_opt            | Neutron Extra DHCP opts                       |
| standard-attr-revisions   | Resource revision numbers                     |
| pagination                | Pagination support                            |
| sorting                   | Sorting support                               |
| security-group            | security-group                                |
| dhcp_agent_scheduler      | DHCP Agent Scheduler                          |
| router_availability_zone  | Router Availability Zone                      |
| rbac-policies             | RBAC Policies                                 |
| standard-attr-description | standard-attr-description                     |
| router                    | Neutron L3 Router                             |
| allowed-address-pairs     | Allowed Address Pairs                         |
| project-id                | project_id field enabled                      |
| dvr                       | Distributed Virtual Router                    |
+---------------------------+-----------------------------------------------+
```

3. Listem els agents entre els que es trobarà el Linux Bridge definit al nostre node compute que verificarà l'èxit de la instal·lació.

```
root@controller:~$ openstack network agent list
```

```
root@controller:~# openstack network agent list
+--------------------------------------+--------------------+------------+-------------------+-------+-------+---------------------------+
| ID                                   | Agent Type         | Host       | Availability Zone | Alive | State | Binary                    |
+--------------------------------------+--------------------+------------+-------------------+-------+-------+---------------------------+
| 012b315c-0f62-4eb5-81cf-2afd6458a31a | Linux bridge agent | controller | None              | True  | UP    | neutron-linuxbridge-agent |
| 5ae9d405-27fd-45f8-8c8b-25ea027b9c76 | Metadata agent     | controller | None              | True  | UP    | neutron-metadata-agent    |
| 74cd12d1-c929-40a0-87dc-08e40b341d43 | L3 agent           | controller | nova              | True  | UP    | neutron-l3-agent          |
| 8e2a2a3b-29fb-4850-9997-28ab42444c95 | Linux bridge agent | compute    | None              | True  | UP    | neutron-linuxbridge-agent |
| f113dcd4-d358-460b-898d-c19a5d45cb30 | DHCP agent         | controller | nova              | True  | UP    | neutron-dhcp-agent        |
+--------------------------------------+--------------------+------------+-------------------+-------+-------+---------------------------+
```

2.7.2.5 Horizon (Servei de dashboard)

OpenStack Horizon es un dels principals components d'OpenStack. Ofereix una interfície web que permet la administració d'usuaris, núvols i serveis de manera gràfica perquè integra els altres mòduls per a poder ser utilitzats mitjançant aquesta interfície per fer més fàcil l'administració.

En el nostre projecte, farem la seva instal·lació amb el servidor web Apache2 que vam instal·lar anteriorment al mòdul KeyStone.

2.7.2.5.1 Instal·lació d'Horizon

Instal·lem els paquets necessaris.

```
root@controller:~$ apt -y install openstack-dashboard
```

Després, el configurem editant el fitxer **/etc/openstack-dashboard/local_settings:**

1. Configurem horizon per a que s'executi al node controller afegint la següent línia.

```
OPENSTACK_HOST = "controller"
```

2. Configurem horizon per a que permeti accedir als hosts remots.

```
ALLOWED_HOSTS = ['*', ]
```

3. Configurem horizon per a que utilitze memcached i emmagatzemi en caché les sessions:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

CACHES = {
    'default': {
                                                                    'BACKEND':
'django.core.cache.backends.memcached.MemcachedCache',
         'LOCATION': '192.168.1.1:11211',
    }
}
```

4. Habilitem KeyStone per a tots els dominis.

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

5. Habilitem la versió 3 de l'API d'identity.

```
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
```

6. Configurem la versió de les APIs.

```
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 2,
}
```

7. Configurem per defecte el domini dels usuaris que crearem mitjançant horizon.

```
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
```

8. Configurem el rol per defecte dels usuaris que crearem mitjantçant horizon assignant-li el rol «user».

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

9. Configurem la zona horària que volem (per defecte nosaltres utilitzarem UTC).

```
TIME_ZONE = "TIME_ZONE"
```

Per finalitzar la instal·lació reiniciem Apache i el servei memcached i comprovem que funcionen sense cap error.

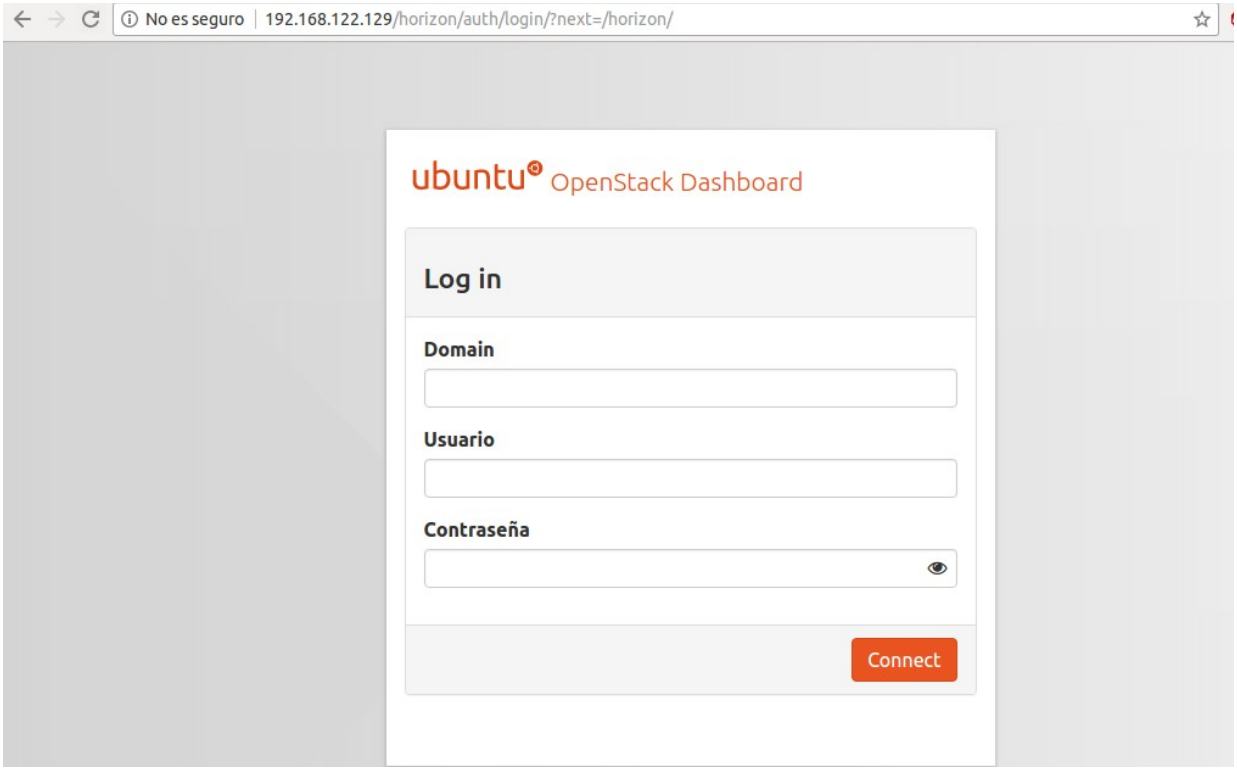
```
root@controller:~$ systemctl restart apache2.service memcached.service
```

```
root@controller:~# systemctl status apache2.service
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since jue 2017-05-11 12:31:35 CEST; 46s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 6212 ExecStop=/etc/init.d/apache2 stop (code=exited, status=0/SUCCESS)
  Process: 5698 ExecReload=/etc/init.d/apache2 reload (code=exited, status=0/SUCCESS)
  Process: 6268 ExecStart=/etc/init.d/apache2 start (code=exited, status=0/SUCCESS)
    Tasks: 174
   Memory: 71.4M
      CPU: 281ms
   CGroup: /system.slice/apache2.service
           ├─6286 /usr/sbin/apache2 -k start
           ├─6289 (wsgi:horizon)    -k start
           ├─6290 (wsgi:horizon)    -k start
           ├─6291 (wsgi:horizon)    -k start
           ├─6292 (wsgi:keystone-pu -k start
           ├─6293 (wsgi:keystone-pu -k start
           ├─6294 (wsgi:keystone-pu -k start
           ├─6295 (wsgi:keystone-pu -k start
           ├─6296 (wsgi:keystone-pu -k start
```

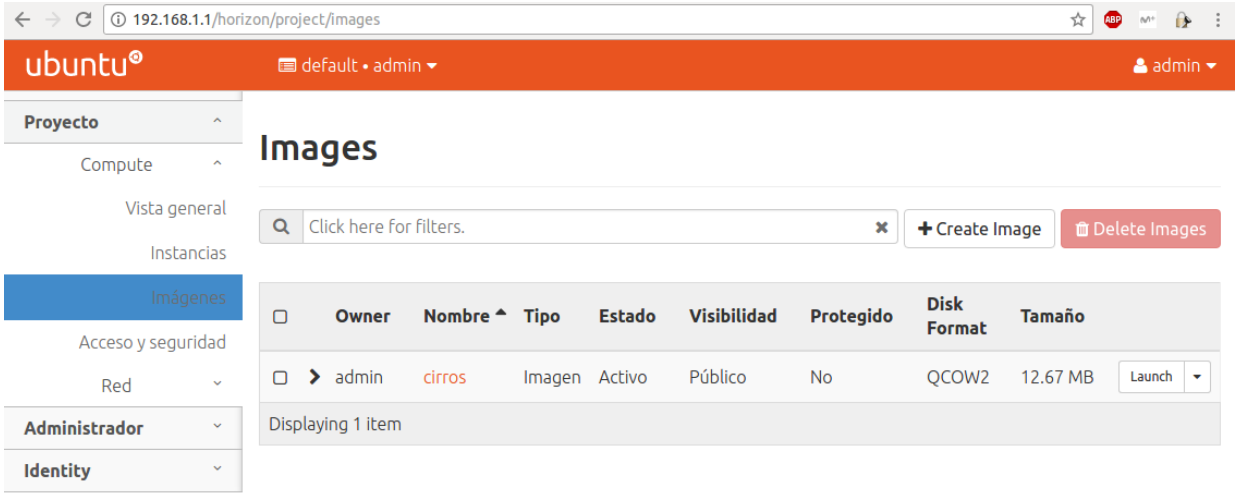
```
root@controller:~# systemctl status memcached.service
● memcached.service - memcached daemon
   Loaded: loaded (/lib/systemd/system/memcached.service; enabled; vendor preset: ena
   Active: active (running) since jue 2017-05-11 12:31:33 CEST; 1min 10s ago
 Main PID: 6218 (memcached)
    Tasks: 6
   Memory: 684.0K
      CPU: 5ms
   CGroup: /system.slice/memcached.service
           └─6218 /usr/bin/memcached -m 64 -p 11211 -u memcache -l 192.168.1.1

may 11 12:31:33 controller systemd[1]: Started memcached daemon.
```

2.7.2.5.2 Verificació de les operacions realitzades al servei Horizon

Per a verificar si la instal·lació que hem realitzat ha sigut satisfactòria entrarem a la interfície web amb la següent ruta: http://controller/horizon o bé per la IP de la xarxa provider: http://192.168.122.129/horizon on ens apareix la nova versió d'Horizon en la que ens demana l'usuari, la contrasenya i el domini.



Utilitzarem els credencials que vam crear a KeyStone amb admin o demo, i el domini es «default» i tindrem accés a la interfície. Com podem observar, la interfície es molt diferent de la que teníem quan vam instal·lar DevStack.



2.7.2.6 Cinder (Servei d'emmagatzematge de blocs)

OpenStack Cinder es el mòdul que proveeix dispositius d'emmagatzematge de blocks per a allotjar instancies. El mètode en el que l'emmagatzematge es proveeix i es consumeix bé determinat per el Block Storage driver, o drivers en el cas de múltiples configuracions. Existeixen un munt de drivers disponibles, NAS/SAN, NFS, iSCSI, Ceph...

A més, Cinder s'executa normalment als nodes controller.

2.7.2.6.1 Coneixent Cinder

Cinder afegeix magatzems persistents a una màquina virtual. Proporciona una infraestructura per a gestionar volums i interactuar amb Nova per a proveir volums per a instancies.

2.7.2.6.2 Instal·lació de Cinder

Primerament, hem de crear i configurar la base de dades que requereix aquest mòdul on desarà la informació.

Node Controller

```
root@controller:~$ mysql -u root -p
```

```
MariaDB [(none)]> CREATE DATABASE cinder;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'contrasenyaEscollida';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'192.168.1.1' IDENTIFIED BY 'contrasenyaEscollida';

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

On 'contrasenyaEscollida' serà la contrasenya que volem posar a la base de dades.

Agafem els credencials de l'usuari a les variables d'entorn.

```
root@controller:~$ . admin-openrc
```

A continuació crearem l'usuari cinder i afegirem el projecte en el que treballarà que vam crear anteriorment amb l'usuari Admin i Demo. A més, crearem el servei cinder i els endpoints d'API per aquest servei :

1. Creem l'usuari glance:

```
root@controller:~$  openstack user create --domain default --password-prompt
cinder
```

```
root@controller:~# openstack user create --domain default --password-prompt cind
er
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | 82d7ae268d914dc8999bf852fcf3dd6e |
| enabled             | True                             |
| id                  | 2856cfe0a14b4513846f85e01a78afea |
| name                | cinder                           |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

2. Afegim el rol i el projecte a cinder.

```
root@controller:~$  openstack role add --project service --user cinder admin
```

```
root@controller:~# openstack user create --domain default --password-prompt cind
er
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | 82d7ae268d914dc8999bf852fcf3dd6e |
| enabled             | True                             |
| id                  | 2856cfe0a14b4513846f85e01a78afea |
| name                | cinder                           |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

3. Creem les entitats de servei cinder i cinderv2 (requereix les dues entitats).

```
root@controller:~$  openstack service create --name cinder --description
"OpenStack Block Storage" volume
```

```
root@controller:~$  openstack service create --name cinderv2 --description
"OpenStack Block Storage" volumev2
```

```
root@controller:~# openstack service create --name cinder --description "OpenSta
ck Block Storage" volume
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Block Storage          |
| enabled     | True                             |
| id          | 98f1b7d64bbe40a8b9e7451396955187 |
| name        | cinder                           |
| type        | volume                           |
+-------------+----------------------------------+
root@controller:~# openstack service create --name cinderv2 --description "OpenS
tack Block Storage" volumev2
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Block Storage          |
| enabled     | True                             |
| id          | a6270652110a4f839ef7f6cfcfc992b9 |
| name        | cinderv2                         |
| type        | volumev2                         |
+-------------+----------------------------------+
```

4. Creem els tres endpoints d'API per als serveis creats.

   Endpoint Public:

```
root@controller:~$ openstack endpoint create --region RegionOne volume public http://controller:8776/v1/%\(tenant_id\)s
```

```
root@controller:~$ openstack endpoint create --region RegionOne volumev2 public http://controller:8776/v2/%\(tenant_id\)s
```

```
root@controller:~# openstack endpoint create --region RegionOne volume public ht
tp://controller:8776/v1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 41fa3edf3de14296895d6a40afdb629b        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 98f1b7d64bbe40a8b9e7451396955187        |
| service_name | cinder                                  |
| service_type | volume                                  |
| url          | http://controller:8776/v1/%(tenant_id)s |
+--------------+-----------------------------------------+
```

```
root@controller:~# openstack endpoint create --region RegionOne volumev2 public
http://controller:8776/v2/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | ef3a51d32d474f4495fd645f6da301dd        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | a6270652110a4f839ef7f6cfcfc992b9        |
| service_name | cinderv2                                |
| service_type | volumev2                                |
| url          | http://controller:8776/v2/%(tenant_id)s |
+--------------+-----------------------------------------+
```

Endpoint Internal:

```
root@controller:~$ openstack endpoint create --region RegionOne volume
internal http://controller:8776/v1/%\(tenant_id\)s
```

```
root@controller:~$ openstack endpoint create --region RegionOne volumev2
internal http://controller:8776/v2/%\(tenant_id\)s
```

```
root@controller:~# openstack endpoint create --region RegionOne volume internal
http://controller:8776/v1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 789c126ed6d247dd9ebdf9c5c465d797        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 98f1b7d64bbe40a8b9e7451396955187        |
| service_name | cinder                                  |
| service_type | volume                                  |
| url          | http://controller:8776/v1/%(tenant_id)s |
+--------------+-----------------------------------------+
```

```
root@controller:~# openstack endpoint create --region RegionOne volumev2 interna
l http://controller:8776/v2/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | cdc86214389c4e1b88cb5f932375a71e        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | a6270652110a4f839ef7f6cfcfc992b9        |
| service_name | cinderv2                                |
| service_type | volumev2                                |
| url          | http://controller:8776/v2/%(tenant_id)s |
+--------------+-----------------------------------------+
```

Endpoint Admin:

```
root@controller:~$ openstack endpoint create --region RegionOne volume admin http://controller:8776/v1/%\(tenant_id\)s
```

```
root@controller:~$ openstack endpoint create --region RegionOne volumev2 admin http://controller:8776/v2/%\(tenant_id\)s
```

```
root@controller:~# openstack endpoint create --region RegionOne volume admin htt
p://controller:8776/v1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | a0862568c40c4ff5b71677408c862879        |
| interface    | admin                                   |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 98f1b7d64bbe40a8b9e7451396955187        |
| service_name | cinder                                  |
| service_type | volume                                  |
| url          | http://controller:8776/v1/%(tenant_id)s |
+--------------+-----------------------------------------+
```

```
root@controller:~# openstack endpoint create --region RegionOne volumev2 admin h
ttp://controller:8776/v2/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 5492101395b44c12b0e742076be1832f        |
| interface    | admin                                   |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | a6270652110a4f839ef7f6cfcfc992b9        |
| service_name | cinderv2                                |
| service_type | volumev2                                |
| url          | http://controller:8776/v2/%(tenant_id)s |
+--------------+-----------------------------------------+
```

5. Instal·lem els paquets necessaris.

```
root@controller:~$ apt -y install cinder-api cinder-scheduler
```

2.7.2.6.3 Configuració de Cinder

Per a configurar Cinder, primerament anirem al fitxer **/etc/cinder/cinder.conf**
Un cop endins, ens hem de moure per diferents seccions per tal de fer funcional aquest mòdul:

1. A la secció **[database]** i configurem l'accés del servei a la base de dades afegint la següent línia:

```
connection = mysql+pymysql://cinder:contrasenyaEscollida@controller/cinder
```

```
[database]
connection = mysql+pymysql://cinder:contrasenyaEscollida@controller/cinder
```

2. A la secció **[DEFAULT]** configurem rabbitMQ.

```
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

Integrem cinder amb KeyStone per a poder autenticar-se:

```
auth_strategy = keystone
```

Després, configurem la opció «my_ip» per a utilitzar la direcció IP de la interfície de xarxa interna del node controlador.

```
my_ip = 192.168.1.1
```

La secció deuria quedar de la següent manera:

```
[DEFAULT]
rootwrap_config = /etc/cinder/rootwrap.conf
api_paste_confg = /etc/cinder/api-paste.ini
iscsi_helper = tgtadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
state_path = /var/lib/cinder
lock_path = /var/lock/cinder
volumes_dir = /var/lib/cinder/volumes
transport_url = rabbit://openstack:openstack@controller
auth_strategy = keystone
my_ip = 192.168.1.1
```

3. A la secció **[keystone_authtoken]** configurem el servei d'accés identity.

```
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = cinder
password = CINDER_PASS
```

```
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = cinder
password = cinder
```

4. A la secció **[keystone_authtoken]** configurem el servei d'accés identity.

```
lock_path = /var/lib/cinder/tmp
```

```
[oslo_concurrency]
lock_path = /var/lib/cinder/tmp
```

Un cop fet això, sincronitzem la base de dades de Cinder.

```
root@controller:~$ su -s /bin/sh -c "cinder-manage db sync" cinder
```

2.7.2.6.3.1 Configuració per integrar Nova amb Cinder

Per integrar Nova amb Cinder editem el fitxer de configuració **/etc/nova/nova.conf** i afegim la següent línia a la secció [cinder]:

```
os_region_name = RegionOne
```

```
[cinder]
os_region_name = RegionOne
```

Reiniciem els diferents serveis configurats i comprovem que funcionen correctament.

```
root@controller:~$ systemctl restart nova-api.service cinder-api.service cinder-scheduler.service
```

```
root@controller:~# systemctl status nova-api.service
● nova-api.service - OpenStack Compute API
   Loaded: loaded (/lib/systemd/system/nova-api.service; enabled; vendor preset:
   Active: active (running) since lun 2017-05-15 11:28:46 CEST; 6s ago
  Process: 3887 ExecStartPre=/bin/chown nova:adm /var/log/nova (code=exited, sta
  Process: 3884 ExecStartPre=/bin/chown nova:nova /var/lock/nova /var/lib/nova (
  Process: 3879 ExecStartPre=/bin/mkdir -p /var/lock/nova /var/log/nova /var/lib
 Main PID: 3889 (nova-api)
    Tasks: 5
   Memory: 276.5M
      CPU: 3.583s
   CGroup: /system.slice/nova-api.service
           ├─3889 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nova
           ├─3928 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nova
           ├─3929 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nova
           ├─3936 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nova
           └─3937 /usr/bin/python /usr/bin/nova-api --config-file=/etc/nova/nova

may 15 11:28:49 controller nova-api[3889]: 2017-05-15 11:28:49.575 3929 INFO nov
may 15 11:28:49 controller nova-api[3889]: 2017-05-15 11:28:49.619 3928 INFO nov
may 15 11:28:49 controller sudo[3930]: pam_unix(sudo:session): session closed fo
may 15 11:28:49 controller sudo[3933]:     nova : TTY=unknown ; PWD=/var/lib/nov
```

```
root@controller:~# systemctl status cinder-api.service
● cinder-api.service - OpenStack Cinder Api
   Loaded: loaded (/lib/systemd/system/cinder-api.service; enabled; vendor prese
   Active: active (running) since lun 2017-05-15 11:28:31 CEST; 40s ago
  Process: 3855 ExecStartPre=/bin/chown cinder:adm /var/log/cinder (code=exited,
  Process: 3852 ExecStartPre=/bin/chown cinder:cinder /var/lock/cinder /var/lib/
  Process: 3850 ExecStartPre=/bin/mkdir -p /var/lock/cinder /var/log/cinder /var
 Main PID: 3857 (cinder-api)
    Tasks: 3
   Memory: 144.0M
      CPU: 2.081s
   CGroup: /system.slice/cinder-api.service
           ├─3857 /usr/bin/python /usr/bin/cinder-api --config-file=/etc/cinder/
           ├─3873 /usr/bin/python /usr/bin/cinder-api --config-file=/etc/cinder/
           └─3874 /usr/bin/python /usr/bin/cinder-api --config-file=/etc/cinder/

may 15 11:28:33 controller cinder-api[3857]: 2017-05-15 11:28:33.837 3857 WARNIN
may 15 11:28:33 controller cinder-api[3857]: 2017-05-15 11:28:33.837 3857 WARNIN
may 15 11:28:33 controller cinder-api[3857]: 2017-05-15 11:28:33.837 3857 WARNIN
may 15 11:28:33 controller cinder-api[3857]: 2017-05-15 11:28:33.838 3857 WARNIN
may 15 11:28:33 controller cinder-api[3857]: 2017-05-15 11:28:33.838 3857 WARNIN
may 15 11:28:33 controller cinder-api[3857]: 2017-05-15 11:28:33.838 3857 WARNIN
```

```
root@controller:~# systemctl status cinder-scheduler.service
● cinder-scheduler.service - OpenStack Cinder Scheduler
   Loaded: loaded (/lib/systemd/system/cinder-scheduler.service; enabled; vendor
   Active: active (running) since lun 2017-05-15 11:28:47 CEST; 58s ago
  Process: 3902 ExecStartPre=/bin/chown cinder:adm /var/log/cinder (code=exited,
  Process: 3898 ExecStartPre=/bin/chown cinder:cinder /var/lock/cinder /var/lib/
  Process: 3892 ExecStartPre=/bin/mkdir -p /var/lock/cinder /var/log/cinder /var
 Main PID: 3908 (cinder-schedule)
    Tasks: 1
   Memory: 123.0M
      CPU: 1.583s
   CGroup: /system.slice/cinder-scheduler.service
           └─3908 /usr/bin/python /usr/bin/cinder-scheduler --config-file=/etc/c

may 15 11:28:46 controller systemd[1]: Starting OpenStack Cinder Scheduler...
may 15 11:28:47 controller systemd[1]: Started OpenStack Cinder Scheduler.
may 15 11:28:48 controller cinder-scheduler[3908]: Option "verbose" from group "
may 15 11:28:48 controller cinder-scheduler[3908]: 2017-05-15 11:28:48.221 3908
may 15 11:28:48 controller cinder-scheduler[3908]: 2017-05-15 11:28:48.274 3908
may 15 11:28:48 controller cinder-scheduler[3908]: 2017-05-15 11:28:48.291 3908
may 15 11:28:48 controller cinder-scheduler[3908]: 2017-05-15 11:28:48.403 3908
may 15 11:28:48 controller cinder-scheduler[3908]: 2017-05-15 11:28:48.631 3908
```

2.7.2.6.4 Instal·lació i configuració de Cinder al node Compute

OpenStack Cinder pot funcionar amb diferents nodes de manera que es pot fer escalable. Seguint els passos que realitzarem ara, el sistema es pot escalar horitzontalment amb tants nodes i blocks d'emmagatzematge com vulguem.

Para realitzar la següent instal·lació, crearem un volum físic per a LVM, podem utilitzar el disc que tenim o podem afegir-ne un altre. Nosaltres hem afegit un de 10 GiB anomenat vdb a la màquina de KVM.

Node Compute

Instal·lem els components necessaris:

```
root@compute:~$ apt -y install lvm2 cinder-volume targetcli python-keystone
```

Habilitem i encenem el següent servei de lvm.

```
root@compute:~$ systemctl enable lvm2-lvmetad.service
root@compute:~$ systemctl start lvm2-lvmetad.service
```

Creem el volum físic per a LVM:

```
root@compute:~$ pvcreate /dev/vdb
```

```
root@compute:~# pvcreate /dev/vdb
  Physical volume "/dev/vdb" successfully created
```

Creem el grup de volums per a LVM:

```
root@compute:~$ vgcreate cinder-volumes /dev/vdb
```

```
root@compute:~# vgcreate cinder-volumes /dev/vdb
  Volume group "cinder-volumes" successfully created
```

Cinder utilitza aquest grup de volums per a generar volums lògics que seran associats a instàncies. Per assegurar que només les instàncies poden tenir accés a aquest grup editem el fitxer **/etc/lvm/lvm.conf** i indiquem a la secció devices el següent:

```
filter = [ "a/vdb/", "r/.*/"]
```

```
devices {

        # Configuration option devices/dir.
        # Directory in which to create volume group device nodes.
        # Commands also accept this as a prefix on volume group names.
        # This configuration option is advanced.
        dir = "/dev"

        # Configuration option devices/scan.
        # Directories containing device nodes to use with LVM.
        # This configuration option is advanced.
        scan = [ "/dev" ]
        filter = [ "a/vdb/", "r/.*/"]
```

Per a configurar cinder al node compute editarem el fitxer **/etc/cinder/cinder.conf**.

1. A la secció **[database]** configurem l'accés a la base de dades del node controller.

   ```
   connection = mysql+pymysql://cinder:contrasenyaEscollida@controller/cinder
   ```

   ```
   [database]
   connection = mysql+pymysql://cinder:contrasenyaEscollida@controller/cinder
   ```

2. A la secció **[DEFAULT]** configurem rabbitMQ.

   ```
   transport_url = rabbit://openstack:RABBIT_PASS@controller
   ```

   Integrem cinder amb KeyStone per a poder autenticar-se:

   ```
   auth_strategy = keystone
   ```

   Després, configurem la opció «my_ip» per a utilitzar la direcció IP de la interfície de xarxa interna del node compute.

   ```
   my_ip = 192.168.1.2
   ```

A continuació habilitem el backend de LVM.

```
enabled_backends = lvm
```

Per últim configurem la localització del servei d'imatges Glance.

```
glance_api_servers = http://controller:9292
```

La secció deuria de quedar així:

```
[DEFAULT]
rootwrap_config = /etc/cinder/rootwrap.conf
api_paste_confg = /etc/cinder/api-paste.ini
iscsi_helper = tgtadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
state_path = /var/lib/cinder
lock_path = /var/lock/cinder
volumes_dir = /var/lib/cinder/volumes
transport_url = rabbit://openstack:openstack@controller
auth_strategy = keystone
my_ip = 192.168.1.2
enabled_backends = lvm
glance_api_servers = http://controller:9292
```

3. A la secció **[keystone_authtoken]** configurem el servei identity (KeyStone).

```
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = cinder
password = CINDER_PASS
```

```
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = cinder
password = cinder
```

4. A la secció **[lvm]** configurem el back end de LVM amb el driver i el protocol iSCSI

```
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
iscsi_protocol = iscsi
iscsi_helper = lioadm
```

```
[lvm]
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
iscsi_protocol = iscsi
iscsi_helper = lioadm
```

5. A la secció **[oslo_concurrency]** configurem el lock path.

```
lock_path = /var/lib/cinder/tmp
```

```
[oslo_concurrency]
lock_path = /var/lib/cinder/tmp
```

Per finalitzar, reiniciem els serveis configurats i comprovem que funcionen correctament.

```
root@compute:~$ systemctl restart tgt.service cinder-volume.service target.service
```

```
root@compute:~# systemctl status tgt.service cinder-volume.service target.service
● tgt.service - (i)SCSI target daemon
   Loaded: loaded (/lib/systemd/system/tgt.service; enabled; vendor preset: enabled)
   Active: active (running) since lun 2017-05-15 18:55:59 CEST; 7s ago
     Docs: man:tgtd(8)
  Process: 5445 ExecStop=/usr/sbin/tgtadm --op delete --mode system (code=exited, status=0/SUCCESS)
  Process: 5438 ExecStop=/usr/sbin/tgt-admin --update ALL -c /dev/null -f (code=exited, status=0/SUCCESS)
  Process: 5422 ExecStop=/usr/sbin/tgt-admin --offline ALL (code=exited, status=0/SUCCESS)
  Process: 5420 ExecStop=/usr/sbin/tgtadm --op update --mode sys --name State -v offline (code=exited, status=0/SUCCESS)
  Process: 5478 ExecStartPost=/usr/sbin/tgtadm --op update --mode sys --name State -v ready (code=exited, status=0/SUCCESS)
  Process: 5453 ExecStartPost=/usr/sbin/tgt-admin -e -c /etc/tgt/targets.conf (code=exited, status=0/SUCCESS)
  Process: 5451 ExecStartPost=/usr/sbin/tgtadm --op update --mode sys --name State -v offline (code=exited, status=0/SUCCESS)
 Main PID: 5448 (tgtd)
   Status: "Starting event loop..."
    Tasks: 1
   Memory: 288.0K
      CPU: 33ms
   CGroup: /system.slice/tgt.service
           └─5448 /usr/sbin/tgtd -f

may 15 18:55:59 compute systemd[1]: Starting (i)SCSI target daemon...
may 15 18:55:59 compute tgtd[5448]: tgtd: iser_ib_init(3434) Failed to initialize RDMA; load kernel modules?
may 15 18:55:59 compute tgtd[5448]: tgtd: work_timer_start(150) use signal based scheduler
may 15 18:55:59 compute tgtd[5448]: tgtd: bs_init(387) use signalfd notification
may 15 18:55:59 compute systemd[1]: Started (i)SCSI target daemon.
```

```
● cinder-volume.service - OpenStack Cinder Volume
   Loaded: loaded (/lib/systemd/system/cinder-volume.service; enabled; vendor preset: enabled)
   Active: active (running) since lun 2017-05-15 18:56:05 CEST; 1s ago
  Process: 5838 ExecStartPre=/bin/chown cinder:adm /var/log/cinder (code=exited, status=0/SUCCESS)
  Process: 5834 ExecStartPre=/bin/chown cinder:cinder /var/lock/cinder /var/lib/cinder (code=exited, status=0/SUCCESS)
  Process: 5831 ExecStartPre=/bin/mkdir -p /var/lock/cinder /var/log/cinder /var/lib/cinder (code=exited, status=0/SUCCESS)
 Main PID: 5840 (cinder-volume)
    Tasks: 1
   Memory: 119.2M
      CPU: 1.274s
   CGroup: /system.slice/cinder-volume.service
           └─5840 /usr/bin/python /usr/bin/cinder-volume --config-file=/etc/cinder/cinder.conf --log-file=/var/log/cinder/cinder-volume.log

may 15 18:56:05 compute systemd[1]: Starting OpenStack Cinder Volume...
may 15 18:56:05 compute systemd[1]: Started OpenStack Cinder Volume.
may 15 18:56:07 compute cinder-volume[5840]: Option "verbose" from group "DEFAULT" is deprecated for removal.  Its value may be silently ignored in
may 15 18:56:07 compute cinder-volume[5840]: 2017-05-15 18:56:07.030 5840 WARNING oslo_reports.guru_meditation_report [-] Guru meditation now regist
may 15 18:56:07 compute cinder-volume[5840]: 2017-05-15 18:56:07.082 5840 INFO root [-] Generating grammar tables from /usr/lib/python2.7/lib2to3/Gr
may 15 18:56:07 compute cinder-volume[5840]: 2017-05-15 18:56:07.099 5840 INFO root [-] Generating grammar tables from /usr/lib/python2.7/lib2to3/Pa
may 15 18:56:07 compute cinder-volume[5840]: 2017-05-15 18:56:07.226 5840 INFO cinder.rpc [req-2563574f-e2a6-4b4e-974f-995550804292 - - - - -] Autom
```

```
● target.service - LSB: The Linux SCSI Target service
   Loaded: loaded (/etc/init.d/target; bad; vendor preset: enabled)
   Active: active (exited) since lun 2017-05-15 18:56:00 CEST; 6s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 5424 ExecStop=/etc/init.d/target stop (code=exited, status=0/SUCCESS)
  Process: 5626 ExecStart=/etc/init.d/target start (code=exited, status=0/SUCCESS)
    Tasks: 0
   Memory: 0B
      CPU: 0
```

Existeix un bug a la versió Newton en la que a vegades el servei cinder-volume apareix com a apagat, però realment està engegat. Això es pot comprovar amb l'altre node i veure si el servei està up o down.

2.7.2.6.5 Verificació de les operacions realitzades al servei cinder

A continuació verificarem que el servei cinder funciona correctament:

Node Controller

1. Agafem els credencials de l'usuari Admin a les variables d'entorn mitjançant l'script creat anteriorment per garantir l'accés a les comandes de l'admin.

```
root@controller:~$ . admin-openrc
```

2. Llistem els components de cinder on apareix el node compute.

```
root@controller:~$ openstack volume service list
```

```
usuario@controller:~$ openstack volume service list
+------------------+-------------+------+---------+-------+----------------------------+
| Binary           | Host        | Zone | Status  | State | Updated At                 |
+------------------+-------------+------+---------+-------+----------------------------+
| cinder-scheduler | controller  | nova | enabled | up    | 2017-05-15T18:14:00.000000 |
| cinder-volume    | compute@lvm | nova | enabled | up    | 2017-05-15T18:13:34.000000 |
+------------------+-------------+------+---------+-------+----------------------------+
```

2.7.2.7 Creant una instància dins d'OpenStack

Per donar per finalitzada la instal·lació d'OpenStack crearem una instància, prèviament creant les xarxes necessàries i el router.

2.7.2.7.1 Creació d'una xarxa serlf-service

Node Controller

1. Com que la xarxa serà creada al projecte demo, carreguem els credencials de l'usuari demo.

```
root@controller:~$ . demo-openrc
```

2. Creem la xarxa self-service

```
root@controller:~$ openstack network create selfservice
```

```
usuario@controller:~$ openstack network create selfservice
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2017-05-16T07:01:28Z                 |
| description               |                                      |
| headers                   |                                      |
| id                        | 22743018-3092-44b4-b8b1-ad2a8c1bc1d4 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| mtu                       | 1450                                 |
| name                      | selfservice                          |
| port_security_enabled     | True                                 |
| project_id                | b2fc7cb72ed64a9db085976af6421515     |
| project_id                | b2fc7cb72ed64a9db085976af6421515     |
| revision_number           | 3                                    |
| router:external           | Internal                             |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      | []                                   |
| updated_at                | 2017-05-16T07:01:29Z                 |
+---------------------------+--------------------------------------+
```

Com estem amb els credencials d'un usuari sense privilegis, no podem indicar-li segons quins paràmetres, però podem editar el fitxer ml2_conf.ini per poder editar-ho manualment.

3. Creem una subxarxa self-service

```
root@controller:~$ openstack subnet create --network selfservice --dns-nameserver DNS_RESOLVER --gateway SELFSERVICE_NETWORK_GATEWAY --subnet-range SELFSERVICE_NETWORK_CIDR selfservice
```

```
usuario@controller:~$ openstack subnet create --network selfservice --dns-nameserver 8.
8.4.4 --gateway 172.16.1.1 --subnet-range 172.16.1.0/24 selfservice
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allocation_pools  | 172.16.1.2-172.16.1.254              |
| cidr              | 172.16.1.0/24                        |
| created_at        | 2017-05-16T07:06:15Z                 |
| description       |                                      |
| dns_nameservers   | 8.8.4.4                              |
| enable_dhcp       | True                                 |
| gateway_ip        | 172.16.1.1                           |
| headers           |                                      |
| host_routes       |                                      |
| id                | ffcbcca6-8060-47db-8c57-1192bf0da8d8 |
| ip_version        | 4                                    |
| ipv6_address_mode | None                                 |
| ipv6_ra_mode      | None                                 |
| name              | selfservice                          |
| network_id        | 22743018-3092-44b4-b8b1-ad2a8c1bc1d4 |
| project_id        | b2fc7cb72ed64a9db085976af6421515     |
| project_id        | b2fc7cb72ed64a9db085976af6421515     |
| revision_number   | 2                                    |
| service_types     | []                                   |
| subnetpool_id     | None                                 |
| updated_at        | 2017-05-16T07:06:15Z                 |
+-------------------+--------------------------------------+
```

4. Creem la xarxa externa:

```
root@controller:~$ openstack network create --share --external --provider-physical-network provider --provider-network-type flat provider
```

```
root@controller:~# openstack network create --share --external --provider-physical-netw
ork provider --provider-network-type flat provider
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2017-05-16T07:51:56Z                 |
| description               |                                      |
| headers                   |                                      |
| id                        | 8e022ca0-8fc0-42c9-a5b2-3a5689a66518 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | False                                |
| mtu                       | 1500                                 |
| name                      | provider                             |
| port_security_enabled     | True                                 |
| project_id                | 3df3a9e2446541868520ab7cab1d5933     |
| project_id                | 3df3a9e2446541868520ab7cab1d5933     |
| provider:network_type     | flat                                 |
| provider:physical_network | provider                             |
| provider:segmentation_id  | None                                 |
| revision_number           | 4                                    |
| router:external           | External                             |
| shared                    | True                                 |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      | []                                   |
| updated_at                | 2017-05-16T07:51:56Z                 |
+---------------------------+--------------------------------------+
```

5. Creem la subxarxa provider.

```
root@controller:~$ openstack subnet create --network provider --allocation-pool start=START_IP_ADDRESS,end=END_IP_ADDRESS --dns-nameserver DNS_RESOLVER --gateway PROVIDER_NETWORK_GATEWAY --subnet-range PROVIDER_NETWORK_CIDR provider
```

```
root@controller:~# openstack subnet create --network provider --allocation-pool start=1
92.168.122.50,end=192.168.122.100 --dns-nameserver 8.8.4.4 --gateway 192.168.122.1 --su
bnet-range 192.168.122.0/24 provider
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allocation_pools  | 192.168.122.50-192.168.122.100       |
| cidr              | 192.168.122.0/24                     |
| created_at        | 2017-05-16T07:59:54Z                 |
| description       |                                      |
| dns_nameservers   | 8.8.4.4                              |
| enable_dhcp       | True                                 |
| gateway_ip        | 192.168.122.1                        |
| headers           |                                      |
| host_routes       |                                      |
| id                | b8b7bbc6-9545-46ea-ba8c-2eecdbb9f2fb |
| ip_version        | 4                                    |
| ipv6_address_mode | None                                 |
| ipv6_ra_mode      | None                                 |
| name              | provider                             |
| network_id        | 8e022ca0-8fc0-42c9-a5b2-3a5689a66518 |
| project_id        | 3df3a9e2446541868520ab7cab1d5933     |
| project_id        | 3df3a9e2446541868520ab7cab1d5933     |
| revision_number   | 2                                    |
| service_types     | []                                   |
| subnetpool_id     | None                                 |
| updated_at        | 2017-05-16T07:59:54Z                 |
+-------------------+--------------------------------------+
```

6. La xarxa self-service es connecta a la xarxa provider mitjançant un router virtual que realitza NAT en les dues direccions. A continuació crearem el router.
7.

```
root@controller:~$ openstack router create router
```

```
usuario@controller:~$ openstack router create router
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| admin_state_up          | UP                                   |
| availability_zone_hints |                                      |
| availability_zones      |                                      |
| created_at              | 2017-05-16T07:09:58Z                 |
| description             |                                      |
| distributed             | False                                |
| external_gateway_info   | null                                 |
| flavor_id               | None                                 |
| ha                      | False                                |
| headers                 |                                      |
| id                      | 9d6aab05-5fff-4805-9fbd-2234a8ff7835 |
| name                    | router                               |
| project_id              | 3df3a9e2446541868520ab7cab1d5933     |
| project_id              | 3df3a9e2446541868520ab7cab1d5933     |
| revision_number         | 3                                    |
| routes                  |                                      |
| status                  | ACTIVE                               |
| updated_at              | 2017-05-16T07:09:58Z                 |
+-------------------------+--------------------------------------+
```

8. Afegim la subxarxa self-service com a interfície en el router.

```
root@controller:~$ neutron router-interface-add router selfservice
```

```
usuario@controller:~$ neutron router-interface-add router selfservice
Added interface d4954c56-5574-4dcd-80c8-438496f7821a to router router.
```

9. Afegim com a porta d'enllaç del router la xarxa provider (la que te accés a internet).

```
root@controller:~$ neutron router-gateway-set router provider
```

```
root@controller:~# neutron router-gateway-set router provider
Set gateway for router router
```

2.7.2.7.2 Verificació de la xarxa self-service

Per verificar que la xarxa ha estat creada satisfactòriament:

1. Agafem els credencials de l'usuari Admin.

```
root@controller:~$ . admin-openrc
```

2. Llistem els namespaces de les xarxes. Hem de veure un qrouter i 2 qdhcp.

```
root@controller:~$ ip netns
```

```
root@controller:~# ip netns
qdhcp-8e022ca0-8fc0-42c9-a5b2-3a5689a66518 (id: 2)
qrouter-9d6aab05-5fff-4805-9fbd-2234a8ff7835 (id: 1)
qdhcp-22743018-3092-44b4-b8b1-ad2a8c1bc1d4 (id: 0)
```

3. Llistem els ports en el router per veure les dues subxarxes connectades.

```
root@controller:~$ neutron router-port-list router
```

```
root@controller:~# neutron router-port-list router
+--------------------------------+------+-------------------+--------------------------------+
| id                             | name | mac_address       | fixed_ips                      |
+--------------------------------+------+-------------------+--------------------------------+
| 40408592-3f49-4098-bbf6-cf4468 |      | fa:16:3e:d9:3a:ba | {"subnet_id": "b8b7bbc6-9545   |
| 444b74                         |      |                   | -46ea-ba8c-2eecdbb9f2fb",      |
|                                |      |                   | "ip_address": "192.168.122.55"} |
| d4954c56-5574-4dcd-            |      | fa:16:3e:69:b0:36 | {"subnet_id": "ffcbcca6-8060   |
| 80c8-438496f7821a              |      |                   | -47db-8c57-1192bf0da8d8",      |
|                                |      |                   | "ip_address": "172.16.1.1"}    |
+--------------------------------+------+-------------------+--------------------------------+
```

4. Per últim, fem un ping a una direcció ip dins del rang de les IPs flotants assignades:

```
root@controller:~$ ping -c 3 192.168.122.50
```

```
root@controller:~# ping -c 3 192.168.122.50
PING 192.168.122.50 (192.168.122.50) 56(84) bytes of data.
64 bytes from 192.168.122.50: icmp_seq=1 ttl=64 time=0.067 ms
64 bytes from 192.168.122.50: icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from 192.168.122.50: icmp_seq=3 ttl=64 time=0.078 ms
```

2.7.2.7.3 Creació d'una instància a la xarxa self-service

A continuació llançarem una instància d'una imatge cirro emmagatzemada al servei d'imatges Glance dins de la xarxa que acabem de crear.

1. Agafem els credencials de l'usuari admin.

```
root@controller:~$ . admin-openrc
```

2. Creem el flavor m1.nano (una plantilla).

```
root@controller:~$ openstack flavor create --id 0 --vcpus 1 --ram 64 --disk 1 m1.nano
```

```
root@controller:~# openstack flavor create --id 0 --vcpus 1 --ram 64 --disk 1 m1.nano
+----------------------------+---------+
| Field                      | Value   |
+----------------------------+---------+
| OS-FLV-DISABLED:disabled   | False   |
| OS-FLV-EXT-DATA:ephemeral  | 0       |
| disk                       | 1       |
| id                         | 0       |
| name                       | m1.nano |
| os-flavor-access:is_public | True    |
| properties                 |         |
| ram                        | 64      |
| rxtx_factor                | 1.0     |
| swap                       |         |
| vcpus                      | 1       |
+----------------------------+---------+
```

3. Generem un parell de claus agafant els credencials de l'usuari demo.

```
root@controller:~$ ssh-keygen -q -N ""
```

```
root@controller:~# ssh-keygen -q -N ""
Enter file in which to save the key (/root/.ssh/id_rsa):
```

```
root@controller:~$ openstack keypair create --public-key /root/.ssh/id_rsa.pub mykey
```

```
root@controller:~# openstack keypair create --public-key /root/.ssh/id_rsa.pub mykey
+-------------+-------------------------------------------------+
| Field       | Value                                           |
+-------------+-------------------------------------------------+
| fingerprint | 2f:a8:4a:61:e0:99:ff:2b:91:ad:04:7a:e9:2e:f4:95 |
| name        | mykey                                           |
| user_id     | b74caa92bb5545c78e26bf51fab485f4                |
+-------------+-------------------------------------------------+
```

4. Afegim regles de grup de seguretat tant per fer pings com per a accedir mitjançant ssh.

```
root@controller:~$ openstack security group rule create --proto icmp ID_GROUP
```

```
root@controller:~# openstack security group rule create --proto icmp 6ec7bd8d-7947-467c-9c0b-ec
16e9b086f2
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| created_at        | 2017-05-16T08:29:18Z                 |
| description       |                                      |
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| headers           |                                      |
| id                | 4058dee9-47e2-4399-9a60-684c9906a75c |
| port_range_max    | None                                 |
| port_range_min    | None                                 |
| project_id        | 3df3a9e2446541868520ab7cab1d5933     |
| project_id        | 3df3a9e2446541868520ab7cab1d5933     |
| protocol          | icmp                                 |
| remote_group_id   | None                                 |
| remote_ip_prefix  | 0.0.0.0/0                            |
| revision_number   | 1                                    |
| security_group_id | 6ec7bd8d-7947-467c-9c0b-ec16e9b086f2 |
| updated_at        | 2017-05-16T08:29:18Z                 |
+-------------------+--------------------------------------+
```

```
root@controller:~$ openstack security group rule create --proto tcp --dst-port 22 ID_GROUP
```

```
root@controller:~# openstack security group rule create --proto tcp --dst-port 22 6ec7bd8d-7947
-467c-9c0b-ec16e9b086f2
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| created_at        | 2017-05-16T08:31:44Z                 |
| description       |                                      |
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| headers           |                                      |
| id                | 33e5a4da-4777-417d-8982-426bec688129 |
| port_range_max    | 22                                   |
| port_range_min    | 22                                   |
| project_id        | 3df3a9e2446541868520ab7cab1d5933     |
| project_id        | 3df3a9e2446541868520ab7cab1d5933     |
| protocol          | tcp                                  |
| remote_group_id   | None                                 |
| remote_ip_prefix  | 0.0.0.0/0                            |
| revision_number   | 1                                    |
| security_group_id | 6ec7bd8d-7947-467c-9c0b-ec16e9b086f2 |
| updated_at        | 2017-05-16T08:31:44Z                 |
+-------------------+--------------------------------------+
```

5. Llancem la instància afegint els paràmetres que hem especificat abans, com per exemple el grup de seguretat o la xarxa a la que estarà, la imatge que vam crear cirros...

```
root@controller:~$ openstack server create --flavor m1.nano --image cirros --nic net-id=SELFSERVICE_NET_ID --security-group default --key-name mykey selfservice-instance
```

```
root@controller:~# openstack server create --flavor m1.nano --image cirros --nic net-id=2274301
8-3092-44b4-b8b1-ad2a8c1bc1d4 --security-group 6ec7bd8d-7947-467c-9c0b-ec16e9b086f2 --key-name
mykey selfservice-instance
+-------------------------------------+-----------------------------------------------------+
| Field                               | Value                                               |
+-------------------------------------+-----------------------------------------------------+
| OS-DCF:diskConfig                   | MANUAL                                              |
| OS-EXT-AZ:availability_zone         |                                                     |
| OS-EXT-SRV-ATTR:host                | None                                                |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None                                                |
| OS-EXT-SRV-ATTR:instance_name       |                                                     |
| OS-EXT-STS:power_state              | NOSTATE                                             |
| OS-EXT-STS:task_state               | scheduling                                          |
| OS-EXT-STS:vm_state                 | building                                            |
| OS-SRV-USG:launched_at              | None                                                |
| OS-SRV-USG:terminated_at            | None                                                |
| accessIPv4                          |                                                     |
| accessIPv6                          |                                                     |
| addresses                           |                                                     |
| adminPass                           | FCRn8ozRLEGd                                        |
| config_drive                        |                                                     |
| created                             | 2017-05-16T08:37:03Z                                |
| flavor                              | m1.nano (0)                                         |
| hostId                              |                                                     |
| id                                  | 51a68e46-4c31-422c-9e78-c533b26c20b9                |
| image                               | cirros (463faac1-cb0e-49ff-84fb-76fa364405d1)       |
| key_name                            | mykey                                               |
| name                                | selfservice-instance                                |
| os-extended-volumes:volumes_attached | []                                                  |
| progress                            | 0                                                   |
| project_id                          | 3df3a9e2446541868520ab7cab1d5933                    |
| properties                          |                                                     |
| security_groups                     | [{u'name': u'6ec7bd8d-7947-467c-9c0b-ec16e9b086f2'}] |
| status                              | BUILD                                               |
| updated                             | 2017-05-16T08:37:04Z                                |
| user_id                             | b74caa92bb5545c78e26bf51fab485f4                    |
+-------------------------------------+-----------------------------------------------------+
```

6. Comprovem l'estat de la instància creada.

```
root@controller:~$ openstack server list
```

```
root@controller:~# openstack server list
+----------------------+---------------------+--------+-----------------------+------------+
| ID                   | Name                | Status | Networks              | Image Name |
+----------------------+---------------------+--------+-----------------------+------------+
| 0bc23483-ebde-4f7e-  | selfservice-instance | ACTIVE | selfservice=172.16.1. | cirros     |
| 9ed6-dbd885368339    |                     |        | 4                     |            |
+----------------------+---------------------+--------+-----------------------+------------+
```

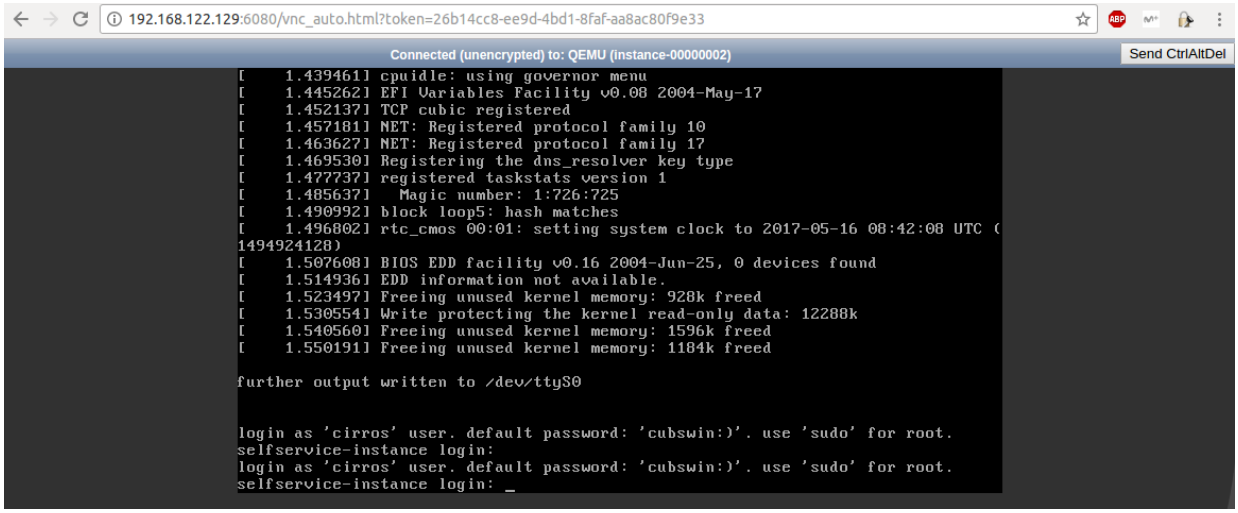
2.7.2.7.4 Accedint a la instància via consola virtual

Per a que els clients puguin accedir a la instància, utilitzem la següent comanda que ens proporcionarà la direcció via web que te assignada.

```
root@controller:~$ openstack console url show selfservice-instance
```

```
root@controller:~# openstack console url show selfservice-instance
+-------+-----------------------------------------------------------------------------------+
| Field | Value                                                                             |
+-------+-----------------------------------------------------------------------------------+
| type  | novnc                                                                             |
| url   | http://controller:6080/vnc_auto.html?token=26b14cc8-ee9d-4bd1-8faf-aa8ac80f9e33 |
+-------+-----------------------------------------------------------------------------------+
```

Accedim a la direcció que se'ns proporciona i ens apareix el terminal de la nostra màquina cirros.

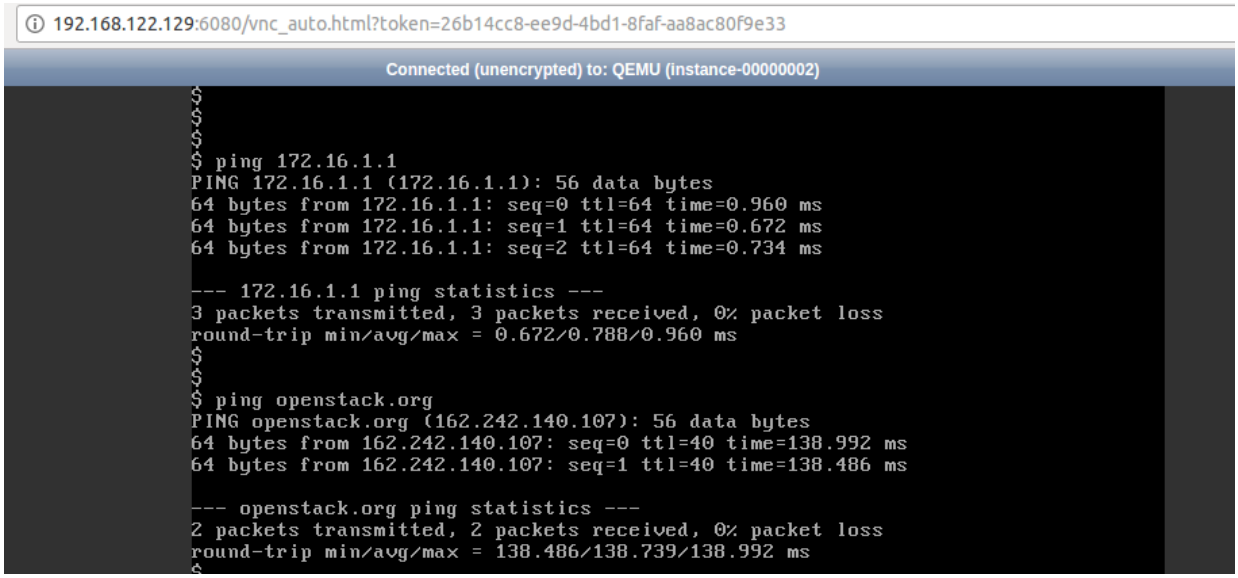


Entrem amb els credencials que ens surt i ja podrem accedir a la nostre màquina cirros. El funcionament es el mateix per crear una màquina amb un ubuntu o un centos

La màquina CirrOS es un linux molt petit però ens permet fer pings, per exemple. Comprovarem que connecta amb el router i que a més, pot sortir a internet sense cap problema.

```
cirros:~$ ping 172.16.1.1
```

```
cirros:~$ ping openstack.org
```



2.7.2.7.5 Accedint a la instància via remota

Per a accedir de manera remota sense necessitat de la web, per ssh, fem el següent:

1. Creem la ip flotant a la xarxa provider.

```
root@controller:~$ openstack floating ip create provider
```

```
root@controller:~# openstack floating ip create provider
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| created_at          | 2017-05-16T09:01:46Z                 |
| description         |                                      |
| fixed_ip_address    | None                                 |
| floating_ip_address | 192.168.122.52                       |
| floating_network_id | 8e022ca0-8fc0-42c9-a5b2-3a5689a66518 |
| headers             |                                      |
| id                  | e786ea17-0ce5-4b71-80d7-47e028b7b798 |
| port_id             | None                                 |
| project_id          | 3df3a9e2446541868520ab7cab1d5933     |
| project_id          | 3df3a9e2446541868520ab7cab1d5933     |
| revision_number     | 1                                    |
| router_id           | None                                 |
| status              | DOWN                                 |
| updated_at          | 2017-05-16T09:01:46Z                 |
+---------------------+--------------------------------------+
```

2. Associem la IP flotant amb la instància.

```
root@controller:~$ openstack server add floating ip selfservice-instance 192.168.122.52
```

```
root@controller:~# openstack server add floating ip selfservice-instance 192.168.122.52
```

3. Mirem l'estat de la nostra IP flotant a la instància.

```
root@controller:~$ openstack server list
```

```
root@controller:~# openstack server list
+--------------------+---------------------+--------+----------------------+------------+
| ID                 | Name                | Status | Networks             | Image Name |
+--------------------+---------------------+--------+----------------------+------------+
| 0bc23483-ebde-4f7e-| selfservice-instance | ACTIVE | selfservice=172.16.1.| cirros     |
| 9ed6-dbd885368339  |                     |        | 4, 192.168.122.52    |            |
+--------------------+---------------------+--------+----------------------+------------+
```

4. Verifiquem la connectivitat amb la IP flotant de la instància.

```
root@controller:~$ ping 192.168.122.52
```

```
root@controller:~# ping 192.168.122.52
PING 192.168.122.52 (192.168.122.52) 56(84) bytes of data.
64 bytes from 192.168.122.52: icmp_seq=1 ttl=63 time=0.749 ms
64 bytes from 192.168.122.52: icmp_seq=2 ttl=63 time=0.802 ms
64 bytes from 192.168.122.52: icmp_seq=3 ttl=63 time=0.665 ms
^C
--- 192.168.122.52 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
```

5. Accedim remotament a la instància via ssh. Com ja està afegit el parell de claus, no fa falta posar els credencials.

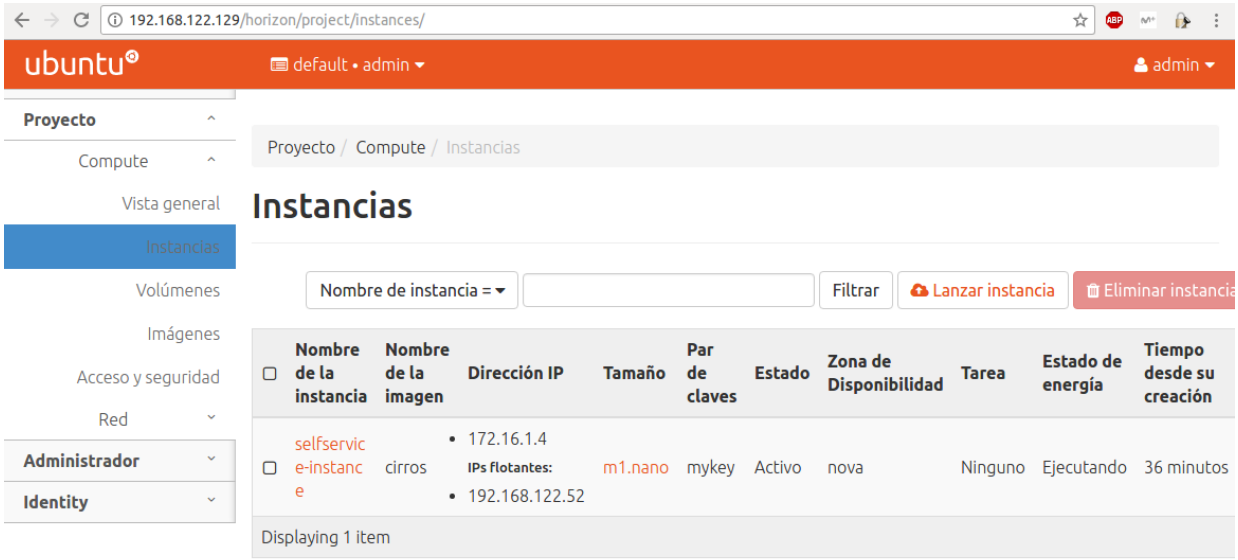
```
root@controller:~$ ssh cirros@192.168.122.52
```

```
root@controller:~# ssh cirros@192.168.122.52
The authenticity of host '192.168.122.52 (192.168.122.52)' can't be established.
RSA key fingerprint is SHA256:0D5xmh/ZiRjwHnYJWQhLehftoZLTXnsDYrsNc1qhFao.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.122.52' (RSA) to the list of known hosts.
$
$
```

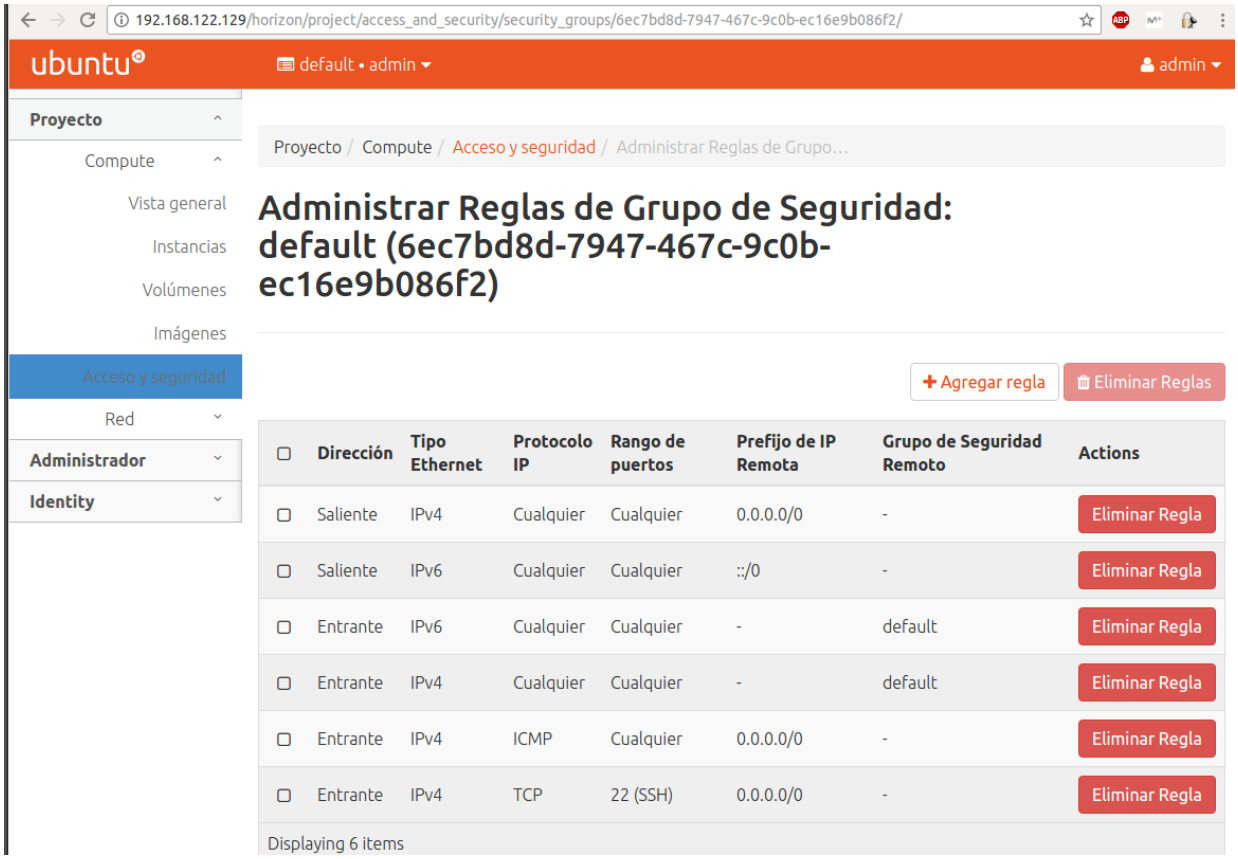
2.7.2.7.6 Verificant la creació de la instància a Horizon

Per a finalitzar el manual de l'usuari, on hem realitzat una instal·lació «bàsica» de tots els components necessaris per a poder llançar una instància, veurem la nostra instància creada a la interfície web de OpenStack.

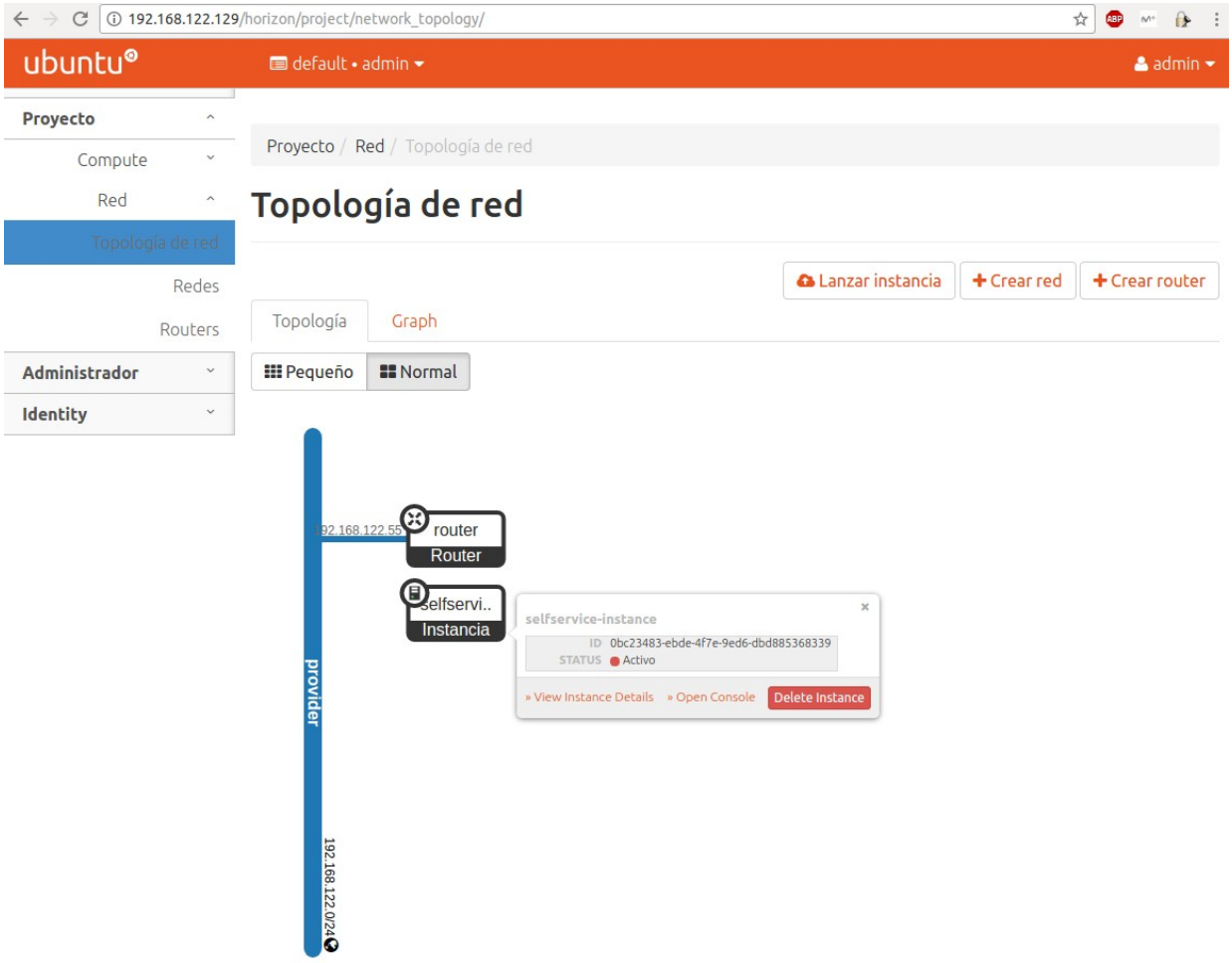
Accedim a la web http://controller/horizon/, accedim amb els credencials de l'admin per exemple, anem a l'apartat instàncies i apareix la nostra instància creada i funcionant.

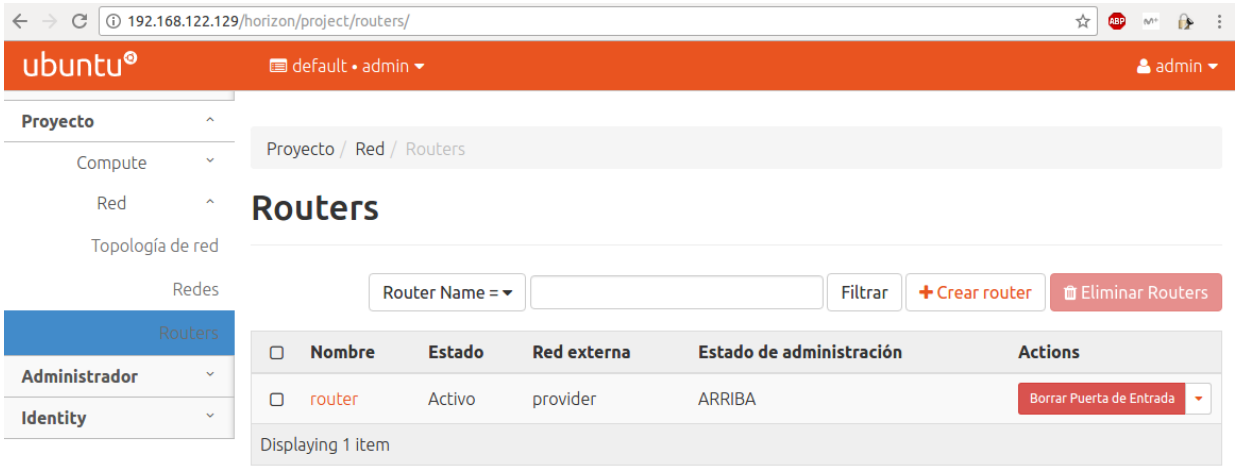


A més, podem veure els grups de seguretat amb les regles i les xarxes que hem definit, els routers....



Dins de la topologia de la xarxa, podem veure un esquema on es veu el router i la instància creada.





2.7.2.8 Ampliant OpenStack

OpenStack es podria ampliar amb més mòduls. El treball que he realitzat només es tracta d'una instal·lació bàsica d'OpenStack on ens permet crear instàncies de manera eficient. Hem creat un OpenStack funcional que es pot aplicar a un maquinari real, dins del centre, el qual pot servir màquines virtuals per a que els alumnes treballin a casa des d'un entorn web o des d'un terminal, sense necessitat d'instal·lar res a casa seva.

Aquest OpenStack funcional es pot ampliar amb el següents mòduls, que no he realitzar per falta de temps perquè son instal·lacions independents en les que tindríem que editar les configuracions que hem realitzat per tal d'integrar tots el mòduls, es a dir, no seria com la instal·lació dels mòduls que hem fet que podria millorar considerablement la instal·lació.

1. Ironic (Bare Metal Service)
2. Magnum (Container Infraestructure Management Service)
3. Trove (Database Service)
4. Barbican (Key Manager Service)
5. Zaqar (Messaging Service)
6. Swift (Object Storage Services)
7. Heat (Orchestation Service)
8. Manila (Shared File Systems Service)
9. Aodh (Telemetry Alarming Services)
10. Ceilometer (Telemetry Data Collection Service)

3. Gestió d'errors

3.1 Error en el desplegament de DevStack versió Newton

Aquest es el primer error que ha aparegut al projecte. En el moment del desplegament d'OpenStack (al punt **2.5**), concretament a la execució de l'Script, aquest va donar un error que fa que es detingui i no finalitzi la instal·lació correctament.

La sortida per pantalla de l'error era el següent:

```
, oslo.concurrency, oslo.context, oslo.serialization, python-dateutil, oslo.log,
 paramiko, python-mimeparse, urllib3
  Found existing installation: appdirs 1.4.3
    Uninstalling appdirs-1.4.3:
      Successfully uninstalled appdirs-1.4.3
  Rolling back uninstall of appdirs
Exception:
Traceback (most recent call last):
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pip/ba
secommand.py", line 215, in main
    status = self.run(options, args)
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pip/co
mmands/install.py", line 342, in run
    prefix=options.prefix_path,
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pip/re
q/req_set.py", line 784, in install
    **kwargs
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pip/re
q/req_install.py", line 851, in install
    self.move_wheel_files(self.source_dir, root=root, prefix=prefix)
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pip/re
q/req_install.py", line 1064, in move_wheel_files
    isolated=self.isolated,
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pip/wh
eel.py", line 247, in move_wheel_files
    prefix=prefix,
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pip/lo
cations.py", line 140, in distutils_scheme
    d = Distribution(dist_args)
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/setupt
ools/dist.py", line 321, in __init__
    _Distribution.__init__(self, attrs)
  File "/usr/lib/python2.7/distutils/dist.py", line 287, in __init__
    self.finalize_options()
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/setupt
ools/dist.py", line 389, in finalize_options
    ep.require(installer=self.fetch_build_egg)
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pkg_re
sources/__init__.py", line 2324, in require
    items = working_set.resolve(reqs, env, installer, extras=self.extras)
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pkg_re
sources/__init__.py", line 862, in resolve
    new_requirements = dist.requires(req.extras)[::-1]
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pkg_re
sources/__init__.py", line 2568, in requires
    dm = self._dep_map
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pkg_re
sources/__init__.py", line 2815, in _dep_map
    self.__dep_map = self._compute_dependencies()
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pkg_re
sources/__init__.py", line 2824, in _compute_dependencies
    for req in self._parsed_pkg_info.get_all('Requires-Dist') or []:
  File "/opt/stack/tempest/.tox/tempest/local/lib/python2.7/site-packages/pkg_re
sources/__init__.py", line 2806, in _parsed_pkg_info
```

Després d'un temps cercant l'origen del problema, vaig donar amb diferents possibles solucions les quals van ser provades sense èxit fins que vaig trobar en un petit comentari d'un fòrum la solució correcta:

Resulta que a l'última versió de DevStack versió Newton en Ubuntu 16.04 hi han diferents errors/bugs que encara no estan solucionats però hi ha formes poc elegants per poder avançar. En aquest cas, el problema estaba relacionat amb les llibreries de python y les seves dependències.

La idea era actualitzar al màxim el meu Ubuntu, baixar i instal·lar python-pip (una eina per la instal·lació i administració dels packets de Python). Després, actualitzar els paquets que em donaven els problemes a més d'instal·lar diferents paquets de Python que son essencials per a que OpenStack em deixés avançar.

Un cop fet això, el problema va deixar d'existir i vaig finalitzar la instal·lació correctament. El post trobat es el següent (també es pot trobar a la bibliografia):

Tim Serewicz (h-tim-2) wrote on 2017-04-13: #17

```
This continues to be a show-stopper at least with Newton on Ubuntu 16.04.
Simply running stack.sh twice will not fix it. While not an elegant
workaround I have found these steps work:

    sudo -i
    apt-get update ; apt-get -y upgrade ; apt-get install -y git vim
python-pip
    ## You will need to answer a question in the middle of the upgrade.
Use enter to choose default

    pip install --upgrade setuptools
    pip install --upgrade urllib3
    pip install --upgrade urllib3[secure]
    apt-get install -y python-jpype python-dev
    pip install --upgrade apptools appdirs

    exit #back to Ubuntu user

    git clone https://git.openstack.org/openstack-dev/devstack -b
stable/newton

    cd devstack/
    vim local.conf
    ## Populate with various settings you usually use

    ./stack.sh
    #This wil fail after about ten minutes

    sudo -H pip install --upgrade appdirs
    ./stack.sh
    # This time ./stack.sh should complete and work

Hopefully this at least can get devstack working for you. If I can find a
more automated fashion, please let me know.
```

3.2 Error intentant virtualitzar endins d'una virtualització (nested virtualization)

Des de el començament vaig decidir crear el projecte utilitzant el software VirtualBox. Un cop feta la instal·lació i en el moment de començar a utilitzar DevStack, em vaig adonar de que VirtualBox no podria realitzar virtualitzacions niuades, pel que no podria gestionar màquines virtuals dins de la meva màquina virtual amb OpenStack.

La virtualització niuada és l'acció d'executar un hipervisor dins d'una màquina virtual, es a dir, un hipervisor endins d'un hipervisor.

Degut a a aquest problema, vaig tenir que migrar a un altre software gestor de màquines virtuals que si que permetia aquesta opció: KVM.

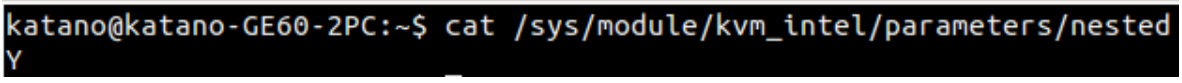
KVM es una de les opcions que està explicada a la memòria per a utilitzar OpenStack. La idea es migrar de VirtualBox a KVM.

Vaig intentar clonar la màquina però KVM no permet imatges amb el format «.ova». Llavors vaig intentar canviar el format del disc dur de «.vdi» a «raw» o «qcow2». Això em va donar més problemes ja que la configuració de com estava instal·lat DevStack no coincidia amb la manera que en que OpenStack te les targes de xarxa.

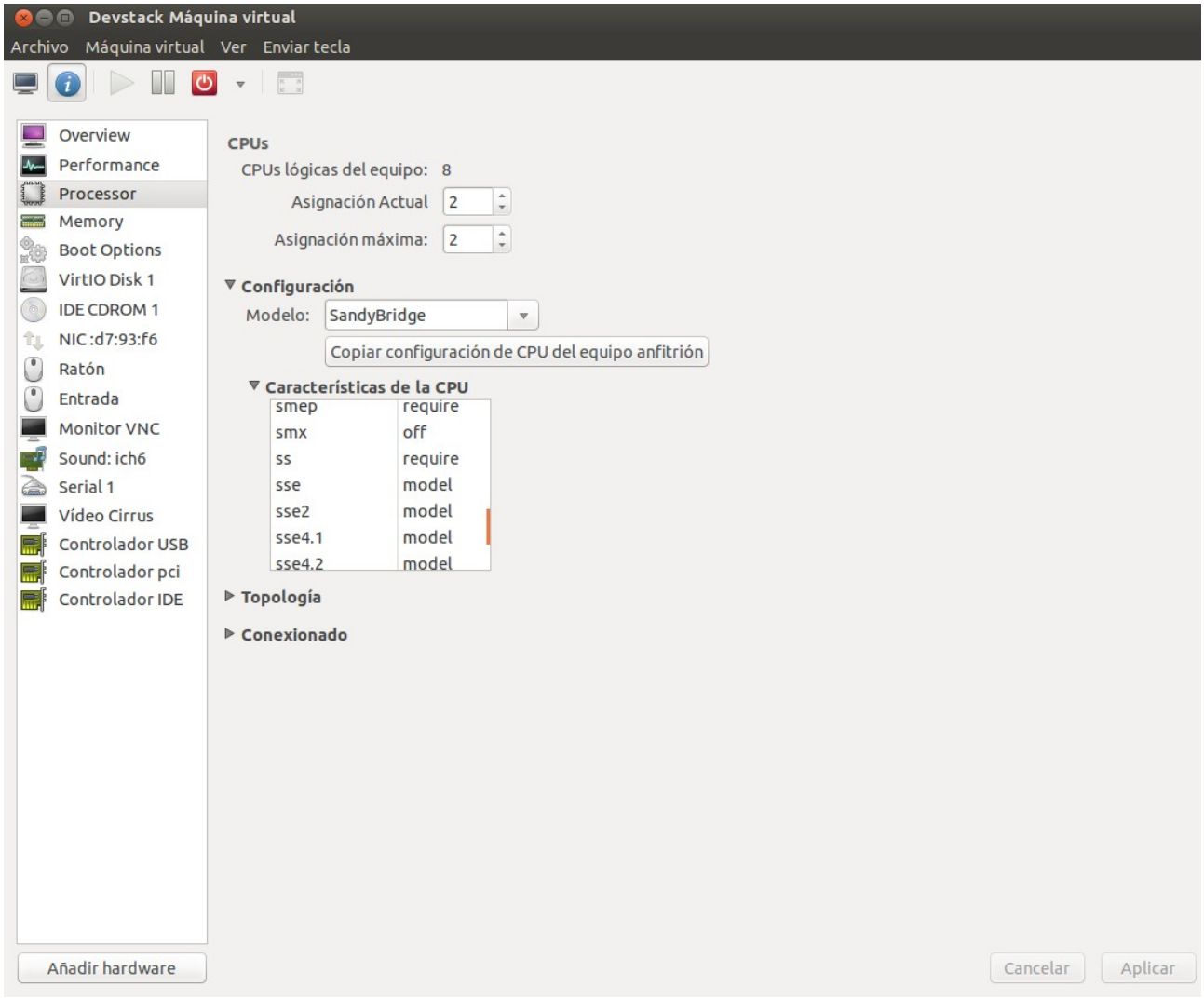
Al final vaig instal·lar de cero una màquina a KVM amb Ubuntu 16.04 per després instal·lar DevStack. Una de les diferencies més notable es que KVM ja crea per defecte un pont (bridge) entre la màquina amfitriona i la màquina virtual, pel que podem posar la tarja en mode NAT sense cap problema de comunicació amb la meva màquina amfitriona.

Un cop vaig tenir tota la màquina com la tenia a VirtualBox vaig procedir a la configuració d'aquesta per tal de fer funcionar la virtualització endins d'una altre virtualització:

1. En primer lloc vaig mirar si el sistema de la nostra màquina real suportava la opció de la virtualització niuada. Si la comanda ens retornava un Y o un nombre major que 1, significa que la màquina suporta aquesta opció.



2. Després vaig configurar KVM mitjançant el virt-manager. Vaig seleccionar «Copiar configuració de la CPU de l'equip amfitrió» dins de la pestanya «Processor». Això el que fa es agafar el model i configuració de la CPU de la teva màquina amfitriona per tal de que la màquina virtual tingui la mateixa i així entenguin la virtualització niuada.



3. Un cop fet això, encenem la nostra màquina virtual i comprovem que ha agafat els valors de la cpu real.

```
stack@ubuntu:~$ cat /proc/cpuinfo
```

```
usuario@devstack:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 42
model name      : Intel Xeon E312xx (Sandy Bridge)
stepping        : 1
microcode       : 0x1
cpu MHz         : 2593.994
cache size      : 4096 KB
physical id     : 0
siblings        : 1
core id         : 0
```

4. Un cop vaig fer la comprovació, vaig veure si ja estava suportada la opció de virtualització niuada.

```
usuario@devstack:~$ sudo virt-host-validate
  QEMU: Verificando for hardware virtualization                              : PASA
  QEMU: Verificando if device /dev/kvm exists                                : PASA
  QEMU: Verificando if device /dev/kvm is accessible                         : PASA
  QEMU: Verificando if device /dev/vhost-net exists                          : PASA
  QEMU: Verificando if device /dev/net/tun exists                            : PASA
  QEMU: Verificando for cgroup 'memory' controller support                   : PASA
  QEMU: Verificando for cgroup 'memory' controller mount-point               : PASA
  QEMU: Verificando for cgroup 'cpu' controller support                      : PASA
  QEMU: Verificando for cgroup 'cpu' controller mount-point                  : PASA
  QEMU: Verificando for cgroup 'cpuacct' controller support                  : PASA
  QEMU: Verificando for cgroup 'cpuacct' controller mount-point              : PASA
  QEMU: Verificando for cgroup 'devices' controller support                  : PASA
  QEMU: Verificando for cgroup 'devices' controller mount-point              : PASA
  QEMU: Verificando for cgroup 'net_cls' controller support                  : PASA
  QEMU: Verificando for cgroup 'net_cls' controller mount-point              : PASA
  QEMU: Verificando for cgroup 'blkio' controller support                    : PASA
  QEMU: Verificando for cgroup 'blkio' controller mount-point                : PASA
  QEMU: Verificando for device assignment IOMMU support                      : ADVERTENCIA (No
ACPI DMAR table found, IOMMU either disabled in BIOS or not supported by this hardware platform)
   LXC: Verificando Para Linux >= 2.6.26                                     : PASA
   LXC: Verificando for namespace ipc                                        : PASA
   LXC: Verificando for namespace mnt                                        : PASA
   LXC: Verificando for namespace pid                                        : PASA
   LXC: Verificando for namespace uts                                        : PASA
   LXC: Verificando for namespace net                                        : PASA
   LXC: Verificando for namespace user                                       : PASA
   LXC: Verificando for cgroup 'memory' controller support                   : PASA
```

Amb això vaig poder continuar amb el projecte encara que em va fer perdre gaire temps.

3.3 Error a la sincronització de KeyStone amb MySql

Durant la configuració del mòdul KeyStone em vaig trobar amb un error quan intentava sincronitzar la base de dades que havia creat a MySql amb KeyStone.

La sortida del error era el següent:

```
root@controller:~# su -s /bin/sh -c "keystone-manage db_sync" keystone
2017-05-05 17:07:49.022 5020 WARNING oslo_db.sqlalchemy.engines [-] SQL connecti
on failed. 10 attempts left.
2017-05-05 17:07:59.033 5020 WARNING oslo_db.sqlalchemy.engines [-] SQL connecti
on failed. 9 attempts left.
^Croot@controller:~# 2017-05-05 17:08:09.044 5020 WARNING oslo_db.sqlalchemy.eng
ines [-] SQL connection failed. 8 attempts left.
```

En el que intentava fins a 10 cops la sincronització en el moment en que l'usuari keystone s'intentava connectar per localhost.

Vaig fer diferents proves com per exemple entrar des de l'usuari keystone (al que li vaig tenir que assignar-li el bash a /etc/passwd que per defecte no tenia cap i no podí iniciar sessió amb ell) i vaig provar a connectar amb la base de dades al terminal, cosa que funcionava a la perfecció.

Cercant informació em vaig adonar de que el problema podia venir perquè KeyStone estava configurat per a accedir a la base de dades mitjançant la IP de la tarja de xarxa interna (192.168.1.1) la qual no estava especificada a MySql, pel que si accedia des de l'usuari Keystone al terminal podia entrar (perquè entrava per la tarja de Loopback) però no era possible entrar per cap altre tarja de xarxa.

Vaig entrar a la configuració de Mysql a **/etc/mysql/mariadb.conf.d/50-server.cnf** i vaig confirmar que MySql només escoltava pel Loopback, així que vaig donar amb el problema.

```
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 127.0.0.1

#
```

Vaig canviar el rang que escoltava MySql per a que escoltés en qualsevol IP (encara que només necessitaríem la nostra xarxa interna).

```
#
user            = mysql
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
port            = 3306
basedir         = /usr
datadir         = /var/lib/mysql
tmpdir          = /tmp
lc-messages-dir = /usr/share/mysql
skip-external-locking

# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 0.0.0.0
```

3.4 Segon Error a la sincronització de KeyStone amb MySql

Durant la sincronització de KeyStone va sortir un altre error tot just solucionat l'anterior. La sincronització es queixava de que una clau que vaig assignar dins del fitxer de configuració era mot llarga (ocupava massa bytes). En poc temps em vaig adonar de que el problema estava en la codificació de MySql respecte a utf8.

```
root@controller:~# su -s /bin/sh -c "keystone-manage db_sync" keystone
2017-05-05 17:45:47.408 3525 ERROR oslo_db.sqlalchemy.exc_filters [-] DBAPIError
 exception wrapped from (pymysql.err.InternalError) (1071, u'Specified key was t
oo long; max key length is 767 bytes') [SQL: u'\nCREATE TABLE migrate_version (\
n\trepository_id VARCHAR(250) NOT NULL, \n\trepository_path TEXT, \n\tversion IN
TEGER, \n\tPRIMARY KEY (repository_id)\n)\n\n']
2017-05-05 17:45:47.408 3525 ERROR oslo_db.sqlalchemy.exc_filters Traceback (mos
t recent call last):
2017-05-05 17:45:47.408 3525 ERROR oslo_db.sqlalchemy.exc_filters   File "/usr/l
ib/python2.7/dist-packages/sqlalchemy/engine/base.py", line 1139, in _execute_co
ntext
2017-05-05 17:45:47.408 3525 ERROR oslo_db.sqlalchemy.exc_filters     context)
2017-05-05 17:45:47.408 3525 ERROR oslo_db.sqlalchemy.exc_filters   File "/usr/l
```

Vaig intentar provar possibles solucions. Per exemple vaig canviar la cadena del fitxer de configuració i no funcionava. Vaig posar entre cometes la cadena per si no la agafés bé i seguia igual.

Després, em vaig fixar en la mida de la meva cadena i el màxim que deixava posar, i l'error no podia venir d'aquest fitxer perquè no sobrepassava la mida.

Buscant possibles solucions, vaig donar amb un post, com al primer error, en el qual es deia que es un bug dins de la versió Newton d'OpenStack. Resulta que la versió Newton a ubuntu 14.04 no disposa d'aquest error però a la versió 16.04 si.

Mitchell Zubarev-Foxworth (mitch-foxworth) wrote on 2016-05-23: #7

```
Confirmed @namgon's fix (this is building Openstack Liberty on 16.04; I
was seeing the problem populating the keystone database):

1) Replace all instances of 'utf8mb4' with 'utf8' in /etc/mysql/mariadb.
conf.d/*

2) Add the below to /etc/mysql/conf.d/mysqld_openstack.cnf:
[client]
default-character-set = utf8

[mysqld]
bind-address = 172.16.3.32
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8

[mysql]
default-character-set = utf8

3) Drop the keystone database
4) Restart mysql service
5) Run 'keystone-manage db_sync'

For some reason I was still getting the error when /etc/mysql/conf.d/
mysqld_openstack.cnf had the above values, and modifying the files in
/etc/mysql/mariadb.conf.d/ was also necessary to fix.
```

El primer que vaig fer va ser reemplaçar a tots els fitxers de configuració dins de **/etc/mysql/mariadb.conf.d** totes les referencies a utf8mb4 que hi havia per utf8.

```
#
# This group is read by the client library
# Use it for options that affect all clients, but not the server
#

[client]
# Default is Latin1, if you need UTF-8 set this (also in server section)
default-character-set = utf8

# This group is *never* read by mysql client library, though this
# /etc/mysql/mariadb.cnf.d/client.cnf file is not read by Oracle MySQL
# client anyway.
# If you use the same .cnf file for MySQL and MariaDB,
# use it for MariaDB-only client options
[client-mariadb]
```

Vaig canviar la configuració al fitxer **/etc/mysql/conf.d/openStack.conf** que inicialment estava així:

```
[mysqld]
bind-address = 192.168.1.1
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
character-set-server = utf8
~
```

Per aquest altre:

```
[mysqld]
bind-address = 192.168.1.1
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8

[mysql]
default-character-set = utf8
```

Vaig esborrar la base de dades que vaig crear al principi, de keystone, vaig reiniciar MySql i vaig tornar a crear un altre cop la base de dades donant els permisos a l'usuari keystone.

Un cop fet això, la base de dades es va sincronitzar a la perfecció.

```
root@controller:~# su -s /bin/sh -c "keystone-manage db_sync" keystone
2017-05-07 17:19:55.850 2903 INFO migrate.versioning.api [-] 66 -> 67...
2017-05-07 17:20:05.658 2903 INFO migrate.versioning.api [-] done
2017-05-07 17:20:05.658 2903 INFO migrate.versioning.api [-] 67 -> 68...
2017-05-07 17:20:05.716 2903 INFO migrate.versioning.api [-] done
2017-05-07 17:20:05.716 2903 INFO migrate.versioning.api [-] 68 -> 69...
2017-05-07 17:20:05.749 2903 INFO migrate.versioning.api [-] done
2017-05-07 17:20:05.750 2903 INFO migrate.versioning.api [-] 69 -> 70...
2017-05-07 17:20:05.808 2903 INFO migrate.versioning.api [-] done
2017-05-07 17:20:05.808 2903 INFO migrate.versioning.api [-] 70 -> 71...
2017-05-07 17:20:05.842 2903 INFO migrate.versioning.api [-] done
2017-05-07 17:20:05.842 2903 INFO migrate.versioning.api [-] 71 -> 72...
2017-05-07 17:20:05.892 2903 INFO migrate.versioning.api [-] done
2017-05-07 17:20:05.892 2903 INFO migrate.versioning.api [-] 72 -> 73...
2017-05-07 17:20:06.731 2903 INFO migrate.versioning.api [-] done
2017-05-07 17:20:06.732 2903 INFO migrate.versioning.api [-] 73 -> 74...
```

3.5 Error amb la configuració d'Apache per a KeyStone

Durant la configuració d'Apache amb el mòdul Wsgi em va sortir el problema següent al moment de finalitzar aquesta configuració i reiniciar Apache:

```
root@controller:~# systemctl restart apache2.service
Job for apache2.service failed because the control process exited with error cod
e. See "systemctl status apache2.service" and "journalctl -xe" for details.
```

```
root@controller:~# systemctl status apache2.service
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: failed (Result: exit-code) since dom 2017-05-07 17:55:31 CEST; 5s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 3471 ExecStop=/etc/init.d/apache2 stop (code=exited, status=0/SUCCESS)
  Process: 3551 ExecStart=/etc/init.d/apache2 start (code=exited, status=1/FAILURE)

may 07 17:55:31 controller apache2[3551]:  * The apache2 configtest failed.
may 07 17:55:31 controller apache2[3551]: Output of config test was:
may 07 17:55:31 controller apache2[3551]: AH00526: Syntax error on line 4 of /etc/apache2/sites-enabled/wsgi-keystone.conf:
may 07 17:55:31 controller apache2[3551]: Name duplicates previous WSGI daemon definition.
may 07 17:55:31 controller apache2[3551]: Action 'configtest' failed.
may 07 17:55:31 controller apache2[3551]: The Apache error log may have more information.
may 07 17:55:31 controller systemd[1]: apache2.service: Control process exited, code=exited status=1
may 07 17:55:31 controller systemd[1]: Failed to start LSB: Apache2 web server.
may 07 17:55:31 controller systemd[1]: apache2.service: Unit entered failed state.
may 07 17:55:31 controller systemd[1]: apache2.service: Failed with result 'exit-code'.
```

El que indicava es que hi havia un error de sintaxis a la línia 4 del fitxer de configuració d'Apache en el que especificava els nous virtualhosts, però repassant-ho un i altre cop no trobava cap error de sintaxi.

Cercant informació em vaig de que el problema era que tenia que declarar un nom per els daemon WSGI diferents als que tenia posats per defecte (keystone-public i keystone-admin) ja que per algun motiu aquests estaven agafats de serie i no es podien utilitzar, pel que vaig canviar-ho als dos virtualhosts definits:

```
<VirtualHost *:5000>
        WSGIDaemonProcess keystone-public1 processes=5 threads=1 user=keystone group=keystone display-name=%{GROUP}
        WSGIProcessGroup  keystone-public1
```

```
<VirtualHost *:35357>
        WSGIDaemonProcess keystone-admin1 processes=5 threads=1 user=keystone group=keystone display-name=%{GROUP}
        WSGIProcessGroup  keystone-admin1
```

3.6 Error amb la sincronització de Neutron amb MySql

En el moment de fer la sincronització de Neutron amb la base de dades em va aparèixer el següent error:

```
    return meth(self, multiparams, params)
  File "/usr/lib/python2.7/dist-packages/sqlalchemy/sql/ddl.py", line 68, in _execute_
on_connection
    return connection._execute_ddl(self, multiparams, params)
  File "/usr/lib/python2.7/dist-packages/sqlalchemy/engine/base.py", line 968, in _exe
cute_ddl
    compiled
  File "/usr/lib/python2.7/dist-packages/sqlalchemy/engine/base.py", line 1146, in _ex
ecute_context
    context)
  File "/usr/lib/python2.7/dist-packages/sqlalchemy/engine/base.py", line 1337, in _ha
ndle_dbapi_exception
    util.raise_from_cause(newraise, exc_info)
  File "/usr/lib/python2.7/dist-packages/sqlalchemy/util/compat.py", line 200, in rais
e_from_cause
    reraise(type(exception), exception, tb=exc_tb)
  File "/usr/lib/python2.7/dist-packages/sqlalchemy/engine/base.py", line 1139, in _ex
ecute_context
    context)
  File "/usr/lib/python2.7/dist-packages/sqlalchemy/engine/default.py", line 450, in d
o_execute
    cursor.execute(statement, parameters)
sqlalchemy.exc.OperationalError: (sqlite3.OperationalError) Cannot add a NOT NULL colu
mn with default value NULL [SQL: u'ALTER TABLE address_scopes ADD COLUMN ip_version IN
TEGER NOT NULL']
```

Es queixava de la connexió al sqlite, resulta que havia afegit a la configuració de neutron la connexió amb la base de dades MySql però no havia comentat la línia que ve per defecte, la de la connexió amb sqlite.

```
Hi,

The problem is in /etc/neutron/neutron.conf, in section [database] yo must
uncomment this line: connection = sqlite:////var/lib/neutron/neutron.
sqlite

I solved the problem so.
```

Vaig comentar la línia i problema resolt.

```
# The SQLAlchemy connection string to use to connect to the database. (string
# value)
# Deprecated group/name - [DEFAULT]/sql_connection
# Deprecated group/name - [DATABASE]/sql_connection
# Deprecated group/name - [sql]/connection
connection = sqlite:////var/lib/neutron/neutron.sqlite
```

3.7 Segon Error amb la sincronització de Neutron amb MySql

Després de solucionar l'anterior error vaig intentar tornar a sincronitzar la base de dades i em va sortir aquest error:

```
    module = load_module_py(module_id, path)
  File "/usr/lib/python2.7/dist-packages/alembic/util/compat.py", line 79, in lo
ad_module_py
    mod = imp.load_source(module_id, path, fp)
  File "/usr/lib/python2.7/dist-packages/neutron/db/migration/alembic_migrations
/env.py", line 120, in <module>
    run_migrations_online()
  File "/usr/lib/python2.7/dist-packages/neutron/db/migration/alembic_migrations
/env.py", line 106, in run_migrations_online
    with DBConnection(neutron_config.database.connection, connection) as conn:
  File "/usr/lib/python2.7/dist-packages/neutron/db/migration/connection.py", li
ne 32, in __enter__
    self.engine = session.create_engine(self.connection_url)
  File "/usr/lib/python2.7/dist-packages/oslo_db/sqlalchemy/engines.py", line 11
4, in create_engine
    url = sqlalchemy.engine.url.make_url(sql_connection)
  File "/usr/lib/python2.7/dist-packages/sqlalchemy/engine/url.py", line 186, in
 make_url
    return _parse_rfc1738_args(name_or_url)
  File "/usr/lib/python2.7/dist-packages/sqlalchemy/engine/url.py", line 235, in
 _parse_rfc1738_args
    "Could not parse rfc1738 URL from string '%s'" % name)
sqlalchemy.exc.ArgumentError: Could not parse rfc1738 URL from string ''
```

Resulta que OpenStack es molt sensible a les majúscules i minúscules i en el moment d'indicar li en el fitxer de configuració la connexió a la base de dades, vaig escriure la primera lletra de la següent línia en majúscules i no podia parsejar-la.

```
Connection = mysql+pymysql://neutron:contrasenyaEscollida@controller/neutron
```

3.8 Error al loginarse a la interfície web d'Horizon

Vaig tenir un error quan vaig fer la instal·lació d'Horizon i vaig intentar accedir a d'interfície web introduint els credencials de usuari, contrasenya i domini. Em va sortir un error que no em deixava accedir, i mirant el log d'Apache2 vaig veure el problema.

```
Thu May 11 10:41:30.758278 2017] [wsgi:error] [pid 6632:tid 139756925179648] Login successful for user "demo".
Thu May 11 10:41:32.200310 2017] [wsgi:error] [pid 6632:tid 139756925179648] Internal Server Error: /horizon/auth/login/
Thu May 11 10:41:32.200337 2017] [wsgi:error] [pid 6632:tid 139756925179648] Traceback (most recent call last):
Thu May 11 10:41:32.200339 2017] [wsgi:error] [pid 6632:tid 139756925179648]   File "/usr/lib/python2.7/dist-packages/django/core/handlers/base.py", line 132, in get_response
Thu May 11 10:41:32.200341 2017] [wsgi:error] [pid 6632:tid 139756925179648]     response = wrapped_callback(request, *callback_args, **callback_kwargs)
Thu May 11 10:41:32.200343 2017] [wsgi:error] [pid 6632:tid 139756925179648]   File "/usr/lib/python2.7/dist-packages/django/views/decorators/debug.py", line 76, in sensitive_post_parameters_wrapper
Thu May 11 10:41:32.200345 2017] [wsgi:error] [pid 6632:tid 139756925179648]     return view(request, *args, **kwargs)
Thu May 11 10:41:32.200347 2017] [wsgi:error] [pid 6632:tid 139756925179648]   File "/usr/lib/python2.7/dist-packages/django/utils/decorators.py", line 110, in _wrapped_view
Thu May 11 10:41:32.200349 2017] [wsgi:error] [pid 6632:tid 139756925179648]     response = view_func(request, *args, **kwargs)
Thu May 11 10:41:32.200350 2017] [wsgi:error] [pid 6632:tid 139756925179648]   File "/usr/lib/python2.7/dist-packages/django/views/decorators/cache.py", line 57, in _wrapped_view_func
Thu May 11 10:41:32.200352 2017] [wsgi:error] [pid 6632:tid 139756925179648]     response = view_func(request, *args, **kwargs)
Thu May 11 10:41:32.200354 2017] [wsgi:error] [pid 6632:tid 139756925179648]   File "/usr/lib/python2.7/dist-packages/openstack_auth/views.py", line 103, in login
Thu May 11 10:41:32.200356 2017] [wsgi:error] [pid 6632:tid 139756925179648]     **kwargs)
Thu May 11 10:41:32.200357 2017] [wsgi:error] [pid 6632:tid 139756925179648]   File "/usr/lib/python2.7/dist-packages/django/views/decorators/debug.py", line 76, in sensitive_post_parameters_wrapper
Thu May 11 10:41:32.200359 2017] [wsgi:error] [pid 6632:tid 139756925179648]     return view(request, *args, **kwargs)
Thu May 11 10:41:32.200361 2017] [wsgi:error] [pid 6632:tid 139756925179648]   File "/usr/lib/python2.7/dist-packages/django/utils/decorators.py", line 110, in _wrapped_view
Thu May 11 10:41:32.200363 2017] [wsgi:error] [pid 6632:tid 139756925179648]     response = view_func(request, *args, **kwargs)
Thu May 11 10:41:32.200381 2017] [wsgi:error] [pid 6632:tid 139756925179648]   File "/usr/lib/python2.7/dist-packages/django/views/decorators/cache.py", line 57, in _wrapped_view_func
Thu May 11 10:41:32.200384 2017] [wsgi:error] [pid 6632:tid 139756925179648]     response = view_func(request, *args, **kwargs)
Thu May 11 10:41:32.200386 2017] [wsgi:error] [pid 6632:tid 139756925179648]   File "/usr/lib/python2.7/dist-packages/django/contrib/auth/views.py", line 51, in login
Thu May 11 10:41:32.200387 2017] [wsgi:error] [pid 6632:tid 139756925179648]     auth_login(request, form.get_user())
Thu May 11 10:41:32.200389 2017] [wsgi:error] [pid 6632:tid 139756925179648]   File "/usr/lib/python2.7/dist-packages/django/contrib/auth/__init__.py", line 110, in login
Thu May 11 10:41:32.200390 2017] [wsgi:error] [pid 6632:tid 139756925179648]     request.session.cycle_key()
Thu May 11 10:41:32.200392 2017] [wsgi:error] [pid 6632:tid 139756925179648]   File "/usr/lib/python2.7/dist-packages/django/contrib/sessions/backends/base.py", line 285, in cycle_key
Thu May 11 10:41:32.200394 2017] [wsgi:error] [pid 6632:tid 139756925179648]     self.create()
Thu May 11 10:41:32.200395 2017] [wsgi:error] [pid 6632:tid 139756925179648]   File "/usr/lib/python2.7/dist-packages/django/contrib/sessions/backends/cache.py", line 48, in create
Thu May 11 10:41:32.200397 2017] [wsgi:error] [pid 6632:tid 139756925179648]     "Unable to create a new session key. "
Thu May 11 10:41:32.200398 2017] [wsgi:error] [pid 6632:tid 139756925179648] RuntimeError: Unable to create a new session key. It is likely that the cache is unavailable.
```

Resulta que a la documentació oficial d'OpenStack i ha un error quan es configura el fitxer **/etc/openstack-dashboard/local_settings.** A la documentació oficial escriuen en l'apartat de caches i del memcached la següent línia:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

CACHES = {
    'default': {
                                                                    'BACKEND':
'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': 'controller:11211',
    }
}
```

Em vaig donar compte de que a la línia «LOCATION» no estava agafant el valor de la IP controller que està definida a /etc/hosts peque estava entre cometes i intentava accedir amb el nom controller, no amb la IP que te assignat, pel que ho vaig solucionar afegint la IP directament.

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

CACHES = {
    'default': {
                                                                    'BACKEND':
'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '192.168.1.1:11211',
    }
}
```

3.9 Error final després de la instal·lació completada

El projecte va resultar ser un èxit, funcionava tot, les instàncies creades estaven comunicades amb internet i amb localhost, els clients podien accedir a les seves màquines fins que va sorgir el següent error.

Després d'una prova rutinària en el que comprovava que les instàncies funcionaven correctament, vaig apagar les màquines. Després, per algun motiu la màquina principal no encenia. Resulta que el kernel s'havia actualitzat perquè a la versió d'Ubunbu 16.04 està activat l'autoupdate del kernel. En algun moment durant la prova rutinària s'havia instal·lat, malament, i després era impossible engegar-la.

Després de moltes hores intentant engegar-la per poder fer la demostració a la presentació del projecte (instal·lant una versió nova del kernel, entre d'altres coses), vaig donar amb una mitja solució que em permetia executar OpenStack.

La idea es mitjançant un live-CD d'Ubuntu a la màquina virtual per a que al començament pogués accedir al terminal de la versió de prova i allà executar una serie de comandes que em permetien fer un chroot aixecant alhora tots els serveis de Systemd.

Les comandes el que feien era muntar tots els dispositius necessaris, aixecar els serveis de Systemd i ja podia treballar amb OpenStack un altre cop.

Amb aquest mètode i havia alguns serveis que no es volien aixecar, com per exemple l'iSCSI. Vaig realitzar una prova en la que vaig veure que OpenStack funcionava al 99%, només i ha un petit error que no em permet accedir a les instàncies que tinc des de l'exterior, es a dir, les instàncies no podien accedir a internet, i els nostres clients no podien accedir des de fora a les instàncies.

Com no havia pogut realitzar copies de seguretat (no disposava d'un dispositiu d'emmagatzematge per copiar màquines d'aquesta mida) i no podia fer snapshots a KVM (resulta que vaig escollir que els discs durs tinguessin un format RAW en lloc de QCOW2 i això no permetia fer snapshots) vaig pensar en fer una altre instal·lació d'OpenStack però no la vaig poder fer per falta de temps ja que em va sorgir al final quan em faltava retocar la memòria i entregar el projecte, pel que la màquina controller entregada necessita els següents passos per a executar-se:.

```
mkdir /mnt/root
mount /dev/mapper/ubuntu--vg--root /mnt/root
mount /dev/vda1 /mnt/root/boot
apt install systemd-container
systemd-nspawn --network-interface=ens3 --network-interface=ens7 -b -D /mnt/root
```

4. Conclusions

Un cop he finalitzat el projecte i comprenc OpenStack, com funciona i per a que es pot utilitzar, crec que es una de les eines més potents que hi ha actualment per poder gestionar instàncies de virtualitzacions. Té un ampli abast de possibilitats per realitzar amb aquest software.

El projecte en si ha sortit tal i com esperava, o inclòs millor, perquè com ho he realitzat només jo, amb el suport del professorat, no sabia si per temps ho podria acabar, perquè en el moment en el que em donés un error important i no em deixés avançar, el projecte hauria mort però la planificació amb el Gantt ha sigut molt exacta.

He tingut moltes dificultats de tot tipus, molts errors que he solucionat i mols altres programes que he necessitat aprendre per a utilitzar-los d'una manera eficaç sense els quals el projecte no hauria funcionat, com per exemple l'hipervisor KVM.

Per posar una pega al projecte, m'hauria agradat no només fer una instal·lació manual bàsica, sinó instal·lar tots els mòduls, un per un, per crear un servidor del tot funcional, a màquines reals, però estic més que satisfet amb el que he fet.

5. Webgrafia

http://www.gonzalonazareno.org/cloud/material/bk-admin-openstack.pdf

https://www.amazon.es/OpenStack-Cloud-Computing-Cookbook-Third/dp/1782174788/ref=asap_bc?ie=UTF8

http://vmartinezdelacruz.com/mi-primer-semana-en-openstack/

http://bibing.us.es/proyectos/abreproy/90140/fichero/Memoria.pdf

http://ebook.konfigurasi.net/Openstack/ (un montón de libros sobre openstack)

https://openwebinars.net/blog/los-9-componentes-de-openstack-que-deberias-conocer/

http://docs.openstack.org/

http://vmartinezdelacruz.com/en-pocas-palabras-como-funciona-openstack/

https://oliverveits.wordpress.com/2016/06/21/getting-started-with-openstack-using-devstack/

https://docs.openstack.org/project-install-guide/newton/

https://bugs.launchpad.net/devstack/+bug/1667545

https://www.linux-party.com/menu-up/hemeroteca/99-cloudcomputing/8815-introduccion-a-openstack-segunda-parte-como-instalar-y-configurar-openstack-en-un-servidor

https://wiki.hackzine.org/sysadmin/kvm-import-ova.html

https://www.howtoforge.com/tutorial/kvm-on-ubuntu-14.04/

https://fedoraproject.org/wiki/How_to_enable_nested_virtualization_in_KVM

http://www.rdoxenham.com/?p=275

https://bugs.launchpad.net/openstack-manuals/+bug/1575688

http://stackoverflow.com/questions/34716683/specified-key-was-too-long-max-key-length-is-767-bytes

https://docs.openstack.org/newton/install-guide-ubuntu/keystone-install.html

https://groups.google.com/forum/#!topic/modwsgi/tWdsjGCmckw

https://ask.openstack.org/en/question/94306/sql-connection-failed-10-attempts-left/

https://ask.openstack.org/en/question/93463/unable-to-connect-to-keystone-database/

http://www.itzgeek.com/how-tos/linux/ubuntu-how-tos/openstack-liberty-on-ubuntu-14-04-lts-configure-keystone-1.html

https://docs.openstack.org/newton/install-guide-rdo/keystone-services.html

https://docs.openstack.org/newton/install-guide-rdo/keystone-users.html

https://docs.openstack.org/newton/install-guide-rdo/keystone-verify.html

https://ask.openstack.org/en/question/94306/sql-connection-failed-10-attempts-left/

https://ask.openstack.org/en/question/93463/unable-to-connect-to-keystone-database/

http://www.itzgeek.com/how-tos/linux/ubuntu-how-tos/openstack-liberty-on-ubuntu-14-04-lts-configure-keystone-1.html

https://docs.openstack.org/newton/install-guide-rdo/keystone-services.html

https://docs.openstack.org/newton/install-guide-rdo/keystone-users.html

https://groups.google.com/forum/#!topic/modwsgi/51Q4F1jJhRA

http://stackoverflow.com/questions/39317200/name-duplicates-previous-wsgi-daemon-definition

https://docs.openstack.org/newton/install-guide-rdo/keystone-verify.html

https://docs.openstack.org/newton/install-guide-rdo/keystone-openrc.html

https://docs.openstack.org/newton/install-guide-rdo/glance.html

https://docs.openstack.org/newton/install-guide-rdo/common/get-started-image-service.html

https://docs.openstack.org/newton/install-guide-rdo/glance-install.html

https://docs.openstack.org/newton/install-guide-rdo/glance-verify.html

http://download.cirros-cloud.net/0.3.4/

https://docs.openstack.org/newton/install-guide-rdo/common/get-started-compute.html

https://docs.openstack.org/newton/install-guide-rdo/nova-controller-install.html

https://docs.openstack.org/newton/install-guide-rdo/nova-compute-install.html

https://docs.openstack.org/newton/install-guide-rdo/nova-verify.html

https://docs.openstack.org/newton/install-guide-rdo/neutron.html

https://docs.openstack.org/newton/install-guide-rdo/common/get-started-networking.html

https://docs.openstack.org/newton/install-guide-rdo/neutron-concepts.html

https://docs.openstack.org/newton/install-guide-rdo/neutron-controller-install.html

https://docs.openstack.org/newton/install-guide-rdo/neutron-controller-install-option2.html

https://bugs.launchpad.net/neutron/+bug/1577861

https://ask.openstack.org/en/question/91657/runtimeerror-unable-to-create-a-new-session-key-it-is-likely-that-the-cache-is-unavailable-authorization-failed-the-request-you-have-made-requires/

6. Bibliografia

OpenStack Cloud Computing Cookbook 1rst edition

OpenStack Cloud Computing Cookbook, 3rd Edition_d

6. Anexos

S'adjunta amb la documentació del projecte les dues màquines virtuals, controller i compute per a poder comprovar el seu bon funcionament. A més, també s'adjunta la màquina creada a VirtualBox amb DevStack.