Creació d'un clúster de computació amb OpenStack

Oscar Barrabés Vilafranca
Enginyeria en Informàtica

Miquel Colobran Huguet

Gener 2018

Aquest PFC està dedicat a totes les persones que m'han ensenyat alguna cosa al llarg de la meva vida.

En particular vull agrair als meus pares tots els esforços i la dedicació per ajudar-me a ser la persona que sóc. També a la Cristina, la meva companya, sense ella de ben segur no hauria ni començat a fer el PFC, m'ha animat sempre a continuar amb perseverança. Als "nens", tant en Biel com la Laia, que han hagut de tenir més paciència amb mi durant aquests mesos. Però sobretot a en Roger, ja que veient-lo fer el seu treball de recerca de Batxillerat m'ha motivat a fer aquest i a no defallir. Finalment també vull dedicar-li al meu soci i gran amic, en Joan, perquè cada dia aprenc d'ell i sempre hi és per donar-me un cop de mà.

Índex

Índex de figures

Resum

La informàtica ha evolucionat al llarg del temps d'una manera trepidant, l'aparició de la internet a principis dels anys 90 van aparèixer nous reptes que semblava que havien de quedar resolts amb la virtualització de servidors.

La realitat però, és que han aparegut nous paradigmes, amb nous conceptes com el de servir les "necessitats" de les empreses com a servei; parlem de les XaaS[1].

XaaS es refereix a tots els serveis que s'ofereixen al núvol i una de les formes en que es pot implementar és el IaaS[2].

IaaS permet consumir serveis tant d'emmagatzematge com de computació al núvol. És en aquest entorn on han aparegut més proveïdors que permeten posar en funcionament sistemes complexes en hores o fins i tot en minuts.

Alguns exemples són Amazon AWS, Microsoft Azure, Google Cloud Platform i Rackspace Cloud.

Amb l'aparició de conceptes com núvol privat i núvol públic, calien eines noves per tal de poder gestionar aquests nous serveis i és aquí on encaixa OpenStack.

[1] as a Service: com a servei

[2] Infraestructure as a Service: Infraestructura com a servei

Introducció

Justificació del PFC i context en el qual es desenvolupa

Al principi de la virtualització, el ROI de molts projectes quedava plenament justificat amb l'aprofitament en el seu sentit més ampli del hardware físic. Es podien compartir els recursos de computació i de memòria, així com l'espai que ocupen als racks.

Aquesta aproximació avui en dia ja no és del tot vàlida, ja que tot plegat està superat. Podem trobar fàcilment un clúster de virtualització de com a mínim dos o tres servidors a qualsevol PIME. Això ha originat que abans les empreses es referien al "servidor" i actualment en aquestes empreses malgrat per ells sigui el servidor, els seus serveis estan distribuïts i atomitzats en diferents "servidors virtuals".

Gavin McCance[3] del CERN va introduir el concepte de "pets vs cattle". El concepte "pets vs cattle" o mascotes vs ramat, ens obliga a buscar noves maneres de gestionar els nostres serveis (i ja no servidors).

Ja no parlem de servidors d'aplicacions o de versions dels sistemes operatius sinó que parlem de serveis. D'alguna manera, ja no existeixen barreres entre l'usuari i el món tècnic.

Tot i això, el present és encara híbrid i on sembla que únicament hi pugui haver Azure, Amazon o Google, hi encaixa perfectament una solució amb OpenStack. Amb OpenStack podem gestionar la infraestructura del nostre núvol privat d'una manera molt semblant a com ho fem al núvol públic.

OpenStack és un projecte de computació al núvol per proporcionar una infraestructura com a servei (IaaS).

Està pensat per cobrir les necessitats d'empreses on dinàmicament es puguin crear càrregues de treball especialitzades per fer càlculs o anàlisi de manera temporal o

[3] https://www.slideshare.net/gmccance/cern-data-centre-evolution

simplement poder gestionar moments amb pics de tràfic, processador o qualsevol altre recurs.

Objectius del PFC

L´objectiu del PFC és la implementació i gestió d´un clúster d'OpenStack i veure en quins entorns té sentit implementar-lo.

Els objectius parcials per assolir-lo són els següents:

- Conèixer la seva arquitectura
- La visió d'OpenStack segons els fabricants: Redhat, Suse, Ubuntu,
- Conèixer els principals components d'OpenStack:
    - Computació: nova
    - Xarxa: neutron
    - Emmagatzematge de dades: Swift (objectes) i cinder (block)
    - Imatges: glance
    - Gestió d'identitats: keystone
    - Dashboard i gestió web: horizon

Enfocament i mètode seguit

L'enfocament del projecte serà integral, ja que en els primers capítols entendrem l'espai que ocupa OpenStack dins de les diferents solucions de cloud i detallaré quins són els seus components.
A continuació desplegarem l'OpenStack en un entorn de proves amb tota la seva funcionalitat on es veurà com es gestiona i les avantatges que aporta.

Breu descripció dels altres capítols de la memòria

El PFC està estructurat amb el següent ordre:

En el següent capítol, apropament a OpenStack, es descriuran les necessitats actuals de les empreses i com encaixa OpenStack com a solució de cloud, d'on sorgeix i quines àrees cobreix.

Continuarem amb un capítol dedicat a l'arquitectura d'OpenStack, on es descriuran els principals components d'OpenStack i com s'interrelacionen entre ells.

Al quart capítol, es descriurà la visió d'OpenStack segons els fabricants i les principals diferències entre ells per tal de tenir una visió del que és el projecte.

El cinquè capítol, els principals components d'OpenStack s'identificarà cadascun dels principals components i es desenvoluparà una explicació del seu funcionament.

El sisè capítol, d'instal·lació i administració, es descriurà un desplegament d'OpenStack tenint en compte els diferents tipus d'implementació i com s'administra.

El setè capítol, us d'OpenStack, estarà dedicat a veure com s'utilitza i es posen en servei càrregues de treball a l'entorn d'OpenStack utilitzant els diferents components que disposa

Finalment un capítol dedicat a les conclusions.

Planificació del projecte

Creació d'un clúster de computació amb OpenStack			
	Setmana	**Activitat**	**Memòria**
1	20-24 setembre		
2	25-1 octubre	Recopilació informació + Lliurament Pla treball	Índex
3	2-8 octubre	PAC1 - Proposta de pla de treball del PFC	Fi Pla de treball
4	9-15 octubre	Capítol 1 – Introducció	Fi capítol 1
5	16-22 octubre	Capítol 2 - Apropament a OpenStack	Fi capítol 2
6	23-29 octubre	Capítol 3 – Arquitectura d'OpenStack + Proves laboratori	Fi capítol 3
7	30-5 novembre	Capítol 4 – La visió d'OpenStack segons els fabricants + Proves laboratori	Fi capítol 4 + laboratori
8	6-12 novembre	Capítol 5 - Els principals components d'OpenStack + Creació laboratori definitiu	Fi capítol 5

9	13-19 novembre	Lliurament PAC2:	Background + Proposta
10	20-26 novembre	Capítol 6 – Instal·lació i administració d'OpenStack + Preparació scripts administració	Laboratori definitiu
11	27-03 desembre	Capítol 6 – Instal·lació i administració d'OpenStack + Preparació scripts administració	Fi capítol 6
12	4-10 desembre	Capítol 7 – Us d'OpenStack	
13	11-17 desembre	Capítol 7 – Us d'OpenStack	Fi capítol 7
14	18-24 desembre	Lliurament PAC3: Capítol 8 - Conclusió	Memòria definitiva
15	25-31 desembre	Correccions i preparació material per entregar	
16	3 gener	Lliurament Memòria i Presentació	Lliurament
17	8 -14 gener	Debat	

Apropament a OpenStack

Evolució cap al servei

Des d'un punt de vista de desenvolupament de noves aplicacions al núvol, podem trobar tres tipus d'entorn molt diferenciats.

Principalment podem referir-nos a SaaS, Paas o IaaS.

SaaS (Software-as-a-Service) és un concepte que ha existit des de fa molt temps i sol fer referència a l'ús de serveis web. Normalment no cal cap tipus de software client instal·lat i s'executa a través del propi navegador.

Algun exemple poden ser el propi Gmail, Google Docs, Dropbox o CRMs.

Tot el que està relacionat amb el manteniment de la infraestructura com copies, actualitzacions, manteniments és responsabilitat del proveïdor del servei, fins i tot en cas de caigudes del servei. Tot això implica poc control sobre l'entorn per part dels administradors del sistema.

En el model PaaS (Platform-as-a-Service) es gestionen automàticament l'escalabilitat utilitzant més recursos si es requereixen. Òbviament a nivell de disseny de les aplicacions és important que estiguin optimitzades al màxim en quan a accessos a disc, temps de processament, espai requerit o qualsevol altre paràmetre que afecti a la plataforma.

Alguns exemples PaaS podria ser Google App Engine, Salesforce Heroku, Acquia, IBM SmartCloud, Microsoft Azure.

Finalment IaaS (Infraestructure-as-a-Service) que és el que més control permet, sobretot en la infraestructura. Físicament els recursos provenen de varis servidors i xarxes distribuïts en diversos centres de processament de dades. Del manteniment de la part d'infraestructura se n'encarrega el proveïdor de serveis.

Exemples IaaS poden ser Rackspace Cloud o vCloud de VMWare o Amazon Web Service (AWS) on podem controlar l'emmagatzemament amb S3 i les màquines virtuals amb EC2.

Es pot triar la configuració del sistema operatiu, memòria, processadors, ample de banda, connexió de xarxa, adreces IP i balancejadors, de manera virtual.

Una aplicació del model IaaS per a grans empreses poden ser les **clouds privades** que utilitzen recursos de la xarxa, computació i emmagatzematge de dades per executar les aplicacions pel funcionament del dia a dia.

Això permet el creixement segons necessitats de l'empresa i amb el núvol privat es garanteix la privacitat de les dades.

També podem trobar un altre model per a les empreses que és l'anomenat **Hosting Cloud** en el que les webs estan en servidors virtuals i on es pot disposar de redundància i escalabilitat per poder suportar pics de tràfic en la web.

Les principals avantatges d'un model IaaS són:

- Escalabilitat: els recursos requerits estan disponibles per quan el client els necessiti i de manera immediata.
- No hi ha inversió en hardware: en el cas de tenir un servei cloud és el propi proveïdor qui es cuida del manteniment i de la configuració del hardware.
- Tarificació segons demanda: es paguen únicament els serveis requerits.
- Independència de la localització: els recursos físics poden estar ubicats a qualsevol lloc del mon.
- Seguretat física en els CPDs: en el cas de clouds públics o privats però allotjats a les instal·lacions del proveïdor on es disposen de centres de processament de dades amb totes les mesures de seguretat necessàries.
- Sense punts únics de risc: poden fallar recursos com discos, targetes de xarxa o servidors però amb serveis configurats amb redundància. Això no ha d'afectar al servei.

OpenStack

OpenStack[4] va començar el 2010 com un projecte conjunt de Rackspace Hosting i la NASA. A partir de 2016, és gestionada per la Fundació OpenStack, que és una entitat sense ànim de lucre per promoure el programari OpenStack i la seva comunitat. Des de llavors més de 500 empreses s'han unit al projecte.

OpenStack és una plataforma de programari lliure i de codi obert per a la computació en núvol, majoritàriament implementada com a infraestructura de servei (IaaS), on servidors virtuals i altres recursos estan disponibles per a ser consumits per clients.

La plataforma de programari està formada per components interrelacionats que controlen diversos serveis de processament, emmagatzematge i recursos de xarxes.

Els usuaris el gestionen a través d'un tauler de control basat en web, a través d'eines de línia d'ordres o a través de serveis web REST.

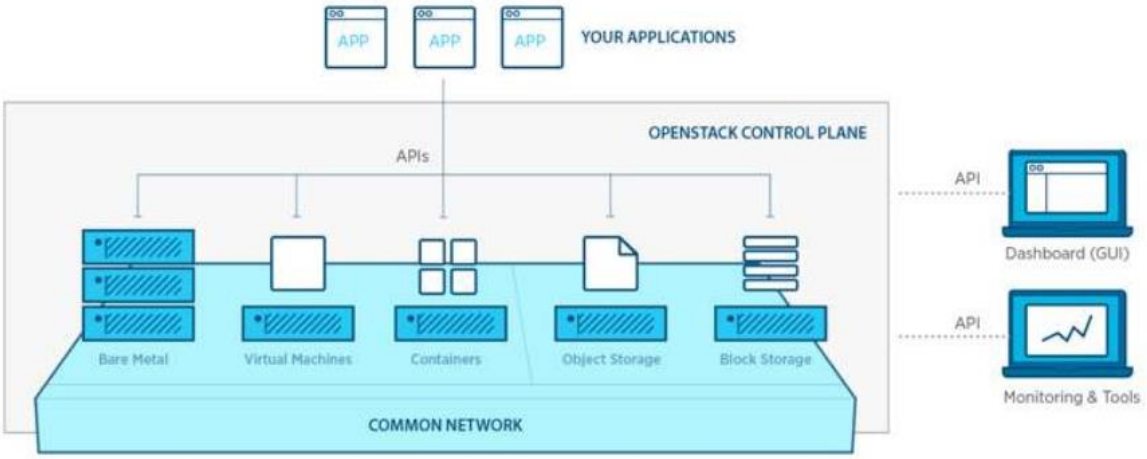

Figura 3 - 1 Esquema comunicació components OpenStack

[4] https://es.wikipedia.org/wiki/OpenStack

Arquitectura d' OpenStack

Unes de les principals característiques del clouds, tant públics com privats, és que han de ser simples d'implementar i escalables.

OpenStack està dividit en diferents components que treballen en conjunt i que s'interrelacionen entre ells mitjançant APIs[5]. Cada servei s'ofereix als altres i alhora en consumeix d'altres.

Les APIs poden ser cridades de dues formes:

- Directament amb python[6], consumint les classes de cada servei. Es permet fer scripts d'automatització com a alternativa a la línia de comandes.

- Mitjançant REST[7], utilitzen el protocol HTTP i són àmpliament utilitzades avui dia. Les APIs REST son cada vegada més comunes en les arquitectures de sistemes ja que faciliten la programació de tasques i la connexió senzilla entre diferents aplicacions. Les principals característiques de REST són:
    - Es defineix una o diverses URL's per a la comunicació entre el client i el servidor, que proporciona una sintaxi universal per identificar els recursos. En el cas d'OpenStack, la definició de les URL's es fa mitjançant endpoints que apunten cadascun d'ells a un servei, d'aquesta manera tindrem un endpoint o varis per cada servei dins d'OpenStack.

[5] API: https://developer.OpenStack.org/api-guide/quick-start/

[6] https://www.ibm.com/developerworks/cloud/library/cl-OpenStack-pythonapis/index.html

[7] https://en.wikipedia.org/wiki/Representational_state_transfer

GET /v3/projects

List projects:

```
curl -s \
 -H "X-Auth-Token: $OS_TOKEN" \
 "http://localhost:5000/v3/projects" | python -mjson.tool
```

Example response:


```
{
    "links": {
        "next": null,
        "previous": null,
        "self": "http://localhost:5000/v3/projects"
    },
    "projects": [
        {
            "description": null,
            "domain_id": "default",
            "enabled": true,
            "id": "3d4c2c82bd5948f0bcab0cf3a7c9b48c",
            "links": {
                "self": "http://localhost:5000/v3/projects/3d4c2c82bd5948f0bcab0cf3a7c9b48c"
            },
            "name": "demo"
        }
    ]
}
```


Figura 4 - 1 Exemple de crida API JSON

- Es defineixen una sèrie d'operacions utilitzant mètodes HTTP GET, POST, PUT i DELETE. S'utilitza habitualment un llenguatge de marques per a les peticions i les respostes. Inicialment el llenguatge més utilitzat era XML, però cada vegada s'utilitza més JSON. En el cas d'OpenStack com podem observar[8] és JSON

Aquesta arquitectura base fa que els serveis es puguin comunicar entres ells de manera fàcil i eficient i alhora els components puguin ser reemplaçats per altres amb la mateixa finalitat.

[8] https://docs.OpenStack.org/keystone/latest/api_curl_examples.html

Els principals components d'OpenStack són els següents:

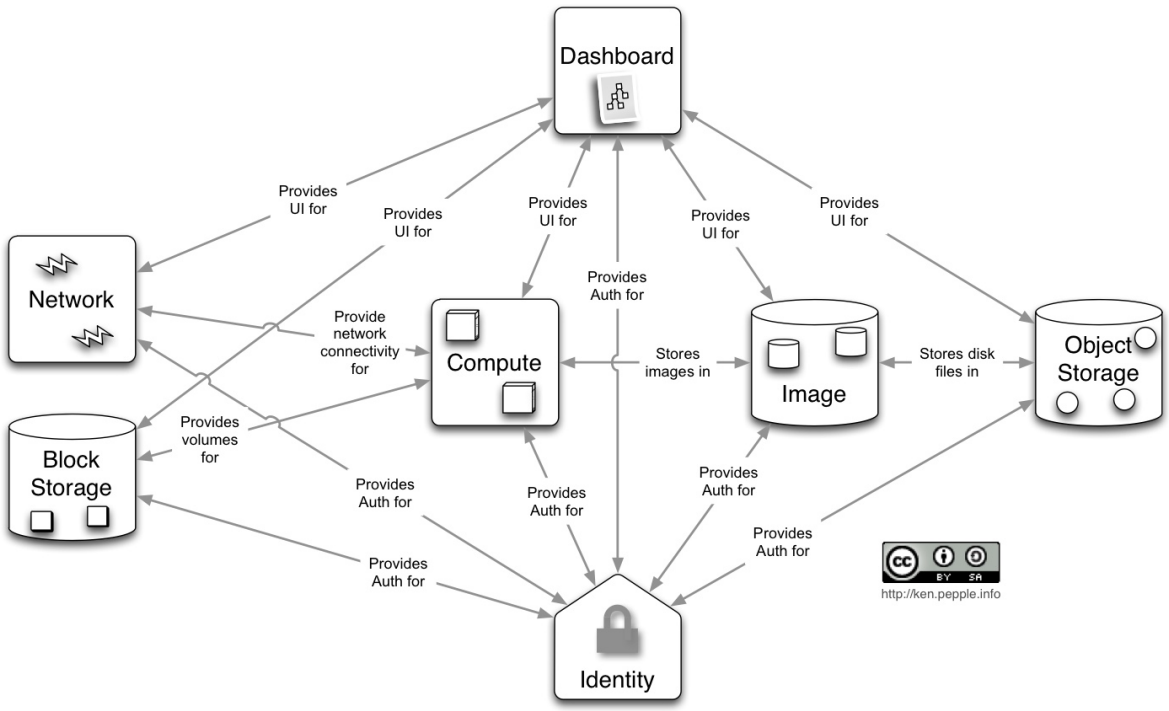

Figura 4 - 2 Components d'OpenStack

Dashboard proveeix la interfície als usuaris finals i a l'administrador. Malgrat també es pot interactuar amb el sistema cridant les API's directament, el més habitual i amigable és fer-ho amb la interfície web. El projecte que el lidera és Horizon[9].

Compute recupera imatges i metadades associades i transforma les comandes dels usuaris en màquines virtuals. No es tracta de l'hypervisor, sinó d'una capa superior que per sota interactua amb ell. El projecte que el lidera és Nova[10]

Network proveeix xarxes virtuals com a servei entre dispositius administrats per altres serveis d'OpenStack, com pot ser una màquina virtual de Nova. Permet als usuaris crear les seves pròpies xarxes i després vincular-les amb els dispositius que desitgin.

[9] https://docs.OpenStack.org/horizon/latest/

[10] https://docs.OpenStack.org/nova/latest/

Implementa totes les capes OSI i es distribueix en tot l'entorn. El projecte que el lidera és Neutron[11].

Image proveeix un catàleg i un repositori per a les imatges. El projecte que el lidera és Glance[12].

Block Storage proveeix emmagatzematge persistent a les VMs allotjades en el núvol. El projecte que el lidera és Cinder[13].

Object Store proveeix emmagatzematge d'objectes. No es tracta d'un sistema d'arxius sinó d'un contenidor en el qual es poden emmagatzemar arxius i recuperar-los després. El projecte que el lidera és Swift[14].

Identity proveeix autenticació i autorització per a tots els serveis d'OpenStack, i també un catàleg d'aquests serveis d'un núvol en particular. El projecte que el lidera és Keystone[15].

Servei	Projecte líder	Funció
Dashboard	Horizon	Interfície per l'usuari final
Compute	Nova	Recupera imatges i metadades
Network	Neutron	Proveeix xarxes virtuals
Image	Glance	Catàleg i repositori de les imatges
Block Storage	Cinder	Emmagatzematge persistent
Object Store	Swift	Emmagatzematge d'objectes
Identity	Keystone	Autenticació i autorització

[11] https://docs.OpenStack.org/neutron/latest/

[12] https://docs.OpenStack.org/glance/latest/

[13] https://docs.OpenStack.org/cinder/latest/

[14] https://docs.OpenStack.org/swift/latest/

[15] https://docs.OpenStack.org/keystone/latest/

La visió d'OpenStack segons els fabricants

OpenStack és un projecte on hi ha diferents fabricants i organitzacions. Tots ells conformen el consorci d'OpenStack[16] i és per aquest motiu que cada distribució d'OpenStack té les seves particularitats.

Red Hat i SUSE són els dos fabricants que estan treballant molt per l'ús i difusió de OpenStack i per tant s'han convertit en els líders en aquest àmbit.

Ubuntu va ser el primer fabricant en donar suport de manera comercial a OpenStack en els seus inicis però actualment s'utilitza més per a entorns petits o de laboratori. Finalment, Mirantis que és un fabricant que des del principi ha estat involucrat amb OpenStack.

Red Hat i SUSE

Actualment les seves distribucions de Linux són les més utilitzades al món empresarial. La seva funcionalitat és molt similar i cada cop són més importants pel projecte OpenStack.

Ambdós fabricants requereixen d'un registre a la seva pàgina web per tal de poder-se instal·lar OpenStack. Amb això es permet realitzar les descàrregues, registrar la instal·lació i configurar els repositoris per poder fer les actualitzacions.

SUSE disposa d'una ISO per a la instal·lació i una altra per les fonts. Red Hat permet fer la instal·lació sobre un Red Hat Enterprise Linux (RHEL) mitjançant repositoris i les seves eines d'instal·lació de paquets.

Pel que fa al llicenciament / subscripció de SUSE, varia segons el rol de cada servidor,. Es diferencien els preus[17] en funció del node:

[16] https://www.OpenStack.org/foundation/companies/

[17] https://www.suse.com/es-es/products/suse-OpenStack-cloud/how-to-buy/

- Node d'administració i de control, l'utilitzarem per poder administrar el cloud i desplegar la resta de nodes. Actualment té un cost de 8.300 EUR/any. Els addicionals tenen un cost de 2.080 EUR/any

- Node de còmput, destinat a allotjar les càrregues de treball, actualment té un cost de subscripció de 670 EUR/any per cadascun, tenint en compte que poden incloure fins a dues CPU's físiques. Cal afegir-hi el cost, si en té, de l'hypervisor (VMWare, HyperV, ...)

- Node d'emmagatzematge, és on s'allotjaran les dades, té un cost de 2.910 EUR/any

Pel que fa a Red Hat, els preus no son públics i no apareixen a la seva web. En el seu model de negoci, OpenStack forma part de Red Hat Cloud Infrastructure (RHCI), però per gestionar-lo cal afegir altres components: Red Hat Enterprise Virtualization (RHEV), System management i el cloud management (Satellite i CloudForms).

En ambdós fabricants, amb la subscripció s'inclou accés a la documentació tècnica tant per fer la instal·lació com la de suport tècnic, així com possibilitat d'assistència tècnica mitjançant l'obertura de casos a la seva web.

El cicle de vida d'OpenStack és un punt a tenir en compte. Els canvis de versió son cada 12 mesos i aquest és un dels punts més negatius en l'ús d'OpenStack en entorns empresarials ja que un canvi de versió pot tenir una afectació important als sistemes, que cada cop són més difícils d'aturar malgrat les aturades siguin programades. Per aquest motiu, tant SUSE com Red Hat estan intentant allargar-los a divuit o vint-i-quatre mesos.

Respecte a la dificultat en la instal·lació, Red Hat és molt simple. Per a un entorn de proves només cal una màquina amb Red Hat Enterprise Linux i mitjançant el gestor de paquets YUM i una connexió a internet, permet instal·lar un sistema amb tots el components d'OpenStack.

L'automatització de la instal·lació la fa mitjançant scripts en Puppet[18] que després utilitzarà per gestionar l'entorn amb l'eina Packstak[19].

[18] https://puppet.com/

Amb SUSE, tot i que es podria disposar de tota la infraestructura en una sola màquina física, es recomana separar en com a mínim tres màquines. Una d'elles dedicada al control, una altra per a la computació i la última per a l'emmagatzematge.

Basa la seva instal·lació en la creació d'un node d'administració que permeti desplegar mitjançant Crowbar[20]. Crowbar és un framework basat en Chef creat per SUSE i permet desplegar qualsevol component d'OpenStack.

A simple vista sembla que SUSE hagi de ser més complex d'instal·lar però amb el node d'administració funcionant permet crear tant el laboratori en màquines virtuals com desplegar la solució en màquines físiques.

Pel que fa a les funcionalitats, aquestes son molt parelles i similars, ja que al tenir cicles de vida tant curts, les dues adopten com a base les mateixes versions d'OpenStack i inclouen tots els serveis que més endavant es descriuran.

Una altra punt important és que SUSE està donant suport a gairebé la majoria de distribucions Linux inclòs Red Hat, facilitant la migració cap als seus entorns.

Hi ha una característica que fa que SUSE sigui interessant, i és que juntament amb Ubuntu son els dos frontends per obtenir la certificació COA[21] d'OpenStack.

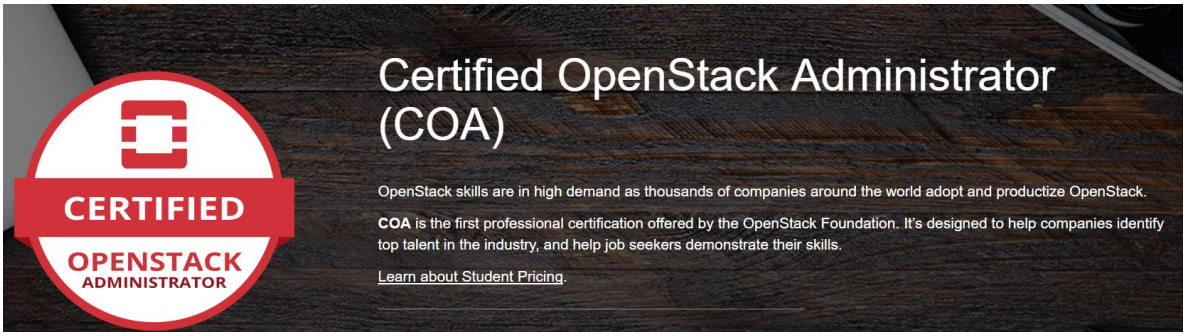

Figura 5 - 1 Certificació COA 1

[19] https://wiki.OpenStack.org/wiki/Packstack

[20] https://github.com/crowbar

[21] https://www.OpenStack.org/coa/

Ubuntu

El projecte d'OpenStack va molt lligat a Ubuntu des que el 2012 Canonical, fabricant d'Ubuntu, va donar-hi suport comercial. Tot i amb això, Red Hat i SUSE han obtingut molt més protagonisme en entorns empresarials. Ubuntu però, encara és un ferm candidat a utilitzar en el cas de voler avaluar OpenStack.

Un exemple del vincle entre OpenStack i Ubuntu, és que la fundació OpenStack utilitza el sistema de bugtracking de Canonical, anomenat Launchpad[22] , per registrar-hi tots els bug del projecte.

També Ubuntu acostuma a ser el primer fabricant en tenir disponibles els paquets de les actualitzacions de qualsevol dels projectes d'OpenStack .

Pel que fa a les subscripcions, Ubuntu[23] és molt similar als anteriors, i es poden tenir subscripcions per node (indiferentment del que sigui) des de 1.500 USD/any a 5.450 USD/any depenent del tipus de suport que es vulgui.

Pel que fa al cicle de vida, Ubuntu integra dins de les seves versions Long Time Support (LTS) les diferents versions d'OpenStack. Això a priori pot ser interessant, però fa que el resultat final no estigui tant integrat com el de la competència, per que ha de mantenir una compatibilitat molt gran entre tots el components d'OpenStack i els propis de Linux.

Ubuntu no permet posar els nodes en alta disponibilitat. Això fa que des del punt de vista empresarial, per entorns productius no sigui la millor elecció.

Pel que fa al seu desplegament, en aquest cas compta amb Devtack[24] que de manera similar a SUSE crea una màquina d'administració que mitjançant Juju[25] permet fer el desplegament de tot l'entorn.

---

[22] https://launchpad.net/OpenStack

[23] https://www.ubuntu.com/support/plans-and-pricing#OpenStack

[24] https://docs.OpenStack.org/devstack/latest/guides/single-machine.html

[25] https://jujucharms.com/OpenStack

Tot i això, hi ha tres coses que fan d'Ubuntu un distribució interessant per a OpenStack:

1.- La plataforma BootStack[26] que és una plataforma de pagament per ús de hardware proveïda per Canonical on es pot desplegar OpenStack sense haver de fer inversions de maquinari.
2.- Orange Box[27] que és un appliance amb deu nodes per poder instal·lar OpenStack i que està enfocat a persones que volen formar-se.
3.- El frontend web és el que tenim per defecte (juntament amb el de SUSE) quan ens volem certificar amb el COA.

Mirantis

Tot i que Mirantis no té una distribució d'OpenStack, aquest sempre ha esta molt lligat amb el projecte. Els inicis de Mirantis es van centrar en la formació en OpenStack i molta de la documentació està feta amb exemples de Mirantis.

Al principi OpenStack no disposava d'una interfície gràfica per desplegar l'entorn i es va crear Mirantis Fuel[28]. Mirantis Fuel és un producte per desplegar i gestionar OpenStack basat en Puppet. S'instal.la sobre les distribucions de Linux més conegudes: CENTOS, Red Hat, Debian, Ubuntu, ...

[26] https://www.ubuntu.com/cloud/managed-cloud

[27] https://insights.ubuntu.com/wp-content/uploads/DS_The_Orange_Box.pdf

[28] https://www.mirantis.com/software/OpenStack/fuel/

Els principals components d'OpenStack

Tal com hem vist ens els punts anteriors, OpenStack és una unió de molts projectes que es comuniquen mitjançant APIs. En aquest apartat s'explica amb més detall les funcions de cada component:

Keystone

És el servei de gestió de les identitats d'OpenStack. Proporciona el servei per identificar i donar accés a tots els components.

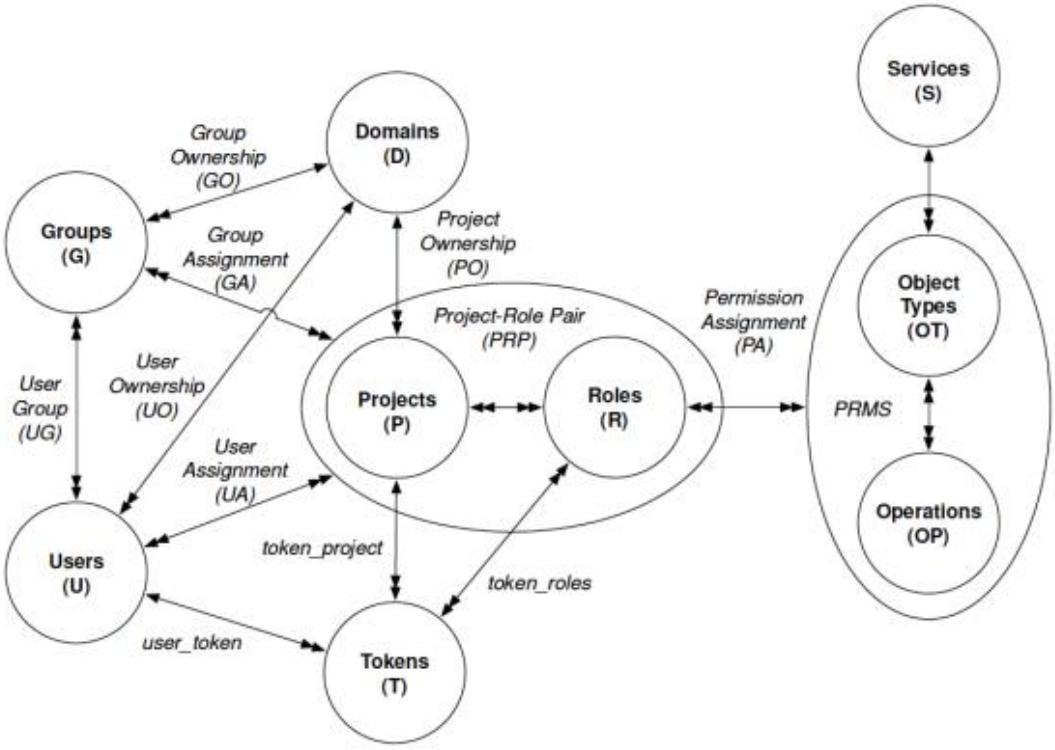

Figura 6 - 1 Components de Keystone

Els serveis que ofereix Keystone son:

- Autenticació que permet la validació d'usuari i contrasenya quan els usuaris del núvol entren en l'entorn d'OpenStack. Keystone pot utilitzar una base de dades per emmagatzemar aquests parells d'usuari / contrasenya o utilitzar una font externa, com ara un LDAP, que ens permetria poder tenir integrat el nostre entorn en sistemes ja existents.

- Servei d'Autorització (o Token). El servei d'autorització està molt relacionat amb el Servei d'autenticació mitjançant la creació de tokens d'autenticació per a usuaris i serveis autenticats que els permeten accedir a altres serveis d'OpenStack.

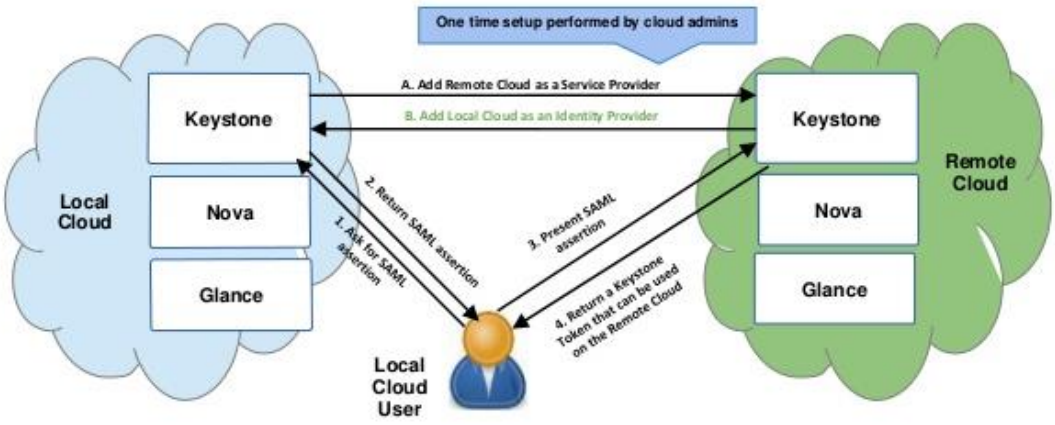

Figura 6 - 2 Autenticació amb tokens

- Catàleg de serveis. Proporciona un índex central de serveis del núvol i els seus punts finals (endpoints). Això simplifica la configuració de la resta de serveis d'OpenStack, perquè enlloc de configurar manualment cada servei per comunicar-se amb qualsevol altre, només cal utilitzar el Catàleg de serveis per cercar el servei relacionat i definir si té o no té accés.

- Servei de recursos que gestiona totes les dades relatives a l'estructura organitzativa d'OpenStack.

- Servei d'assignació que gestiona totes les dades. Relacionat amb rols i assignacions de funcions. Les funcions s'assignen als usuaris i són les que concedeixen o restringeixen l'accés a recursos específics del núvol.

- Servei de polítiques (Policy Service). Gestiona totes les autoritzacions que estan basades en regles entre els usuaris o grups als quals s'assignen els rols i les accions a les quals estan associades.

La raó per la qual Keystone o el Servei d'identitat d'OpenStack són tan importants és que simplifica la interacció entre tots els serveis en el núvol.

Sense el servei d'identitat d'OpenStack, cada usuari hauria de tenir accés a cada servei individualitzat, augmentant enormement les càrregues administratives, creant més possibilitats d'errades en les configuracions.

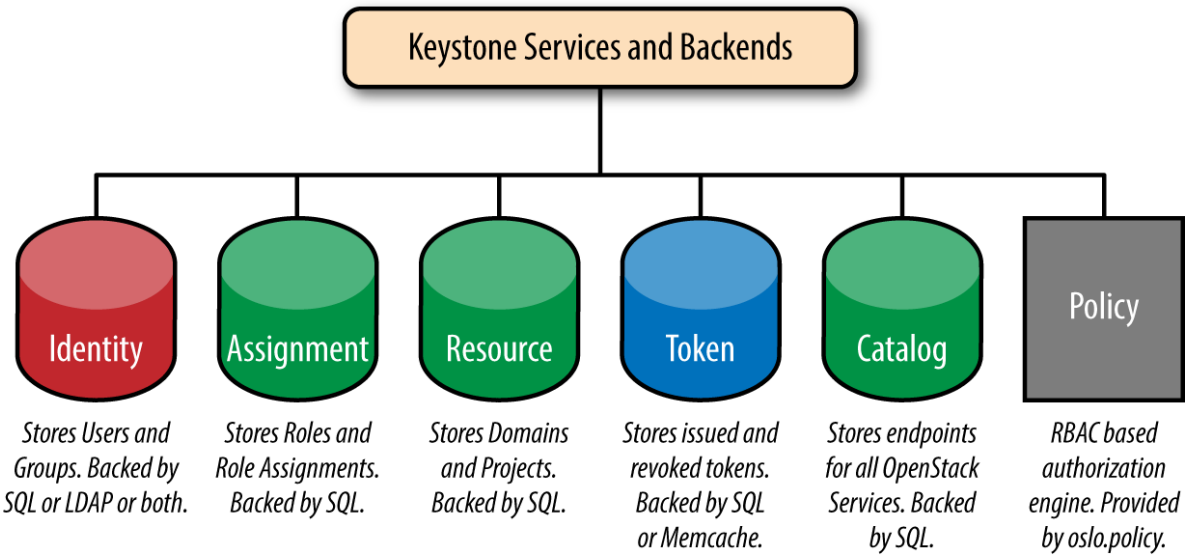

Figura 6 - 3 Serveis de Keystone

Què són els inquilins (tenants), els usuaris i els rols?

Anem a utilitzar l'exemple d'un edifici d'oficines per a explicar els conceptes d'inquilins, usuaris i rols a Openstack.

En els negocis, sovint és més rendible que una empresa llogui l'espai d'oficines en lloc de construir i mantenir els seus propis edificis. El mateix es pot dir dels recursos computacionals. En comptes de construir i mantenir els seus propis centres de dades, el lloguer d'espai en els CPDs també pot ser més rendible.

En el nostre exemple, un edifici d'oficines representa un núvol d'Openstack.

L'edifici està dividit en espais de diferents mides i amb diferents despatxos, sales de conferències, etc. En funció de la quantitat i el tipus d'espai que necessiten les empreses, poden llogar aquests diferents espais. Es converteixen en inquilins de l'edifici d'oficines.

Els empleats d'una empresa són els que realment utilitzen l'espai a l'edifici d'oficines. Es reuneixen a les sales, utilitzen els espais comuns de cuina i es relaxen en sales dedicades al descans.

Els empleats d'una empresa desenvolupen treballs diferents i, per les seves diferents responsabilitats, cada treballador necessiten accés als diferents recursos a l'espai d'oficines. Alguns empleats necessiten una oficina, alguns necessiten una taula i alguns, com el personal de neteja, necessiten accedir a les sales de subministrament de neteja. Aquest concepte seria el de rol de cada usuari.

Ara cal comparar això amb el nostre núvol en OpenStack. Les empreses que necessiten recursos informàtics o d'emmagatzematge poden llogar-los a un proveïdor de núvol d'Openstack. Una entitat anomenada inquilí o tenant es crea en el servei d'identitat OpenStack (o Keystone) que correspon a l'empresa en el nostre exemple, que llogarà espai al núvol.

En algunes de les eines OpenStack es fa referència a un inquilí o tenant com a projecte, però ambdós termes sempre fan referència al mateix tipus d'entitat.

Es defineix una quota en el Servei de computació OpenStack (o Nova) que restringeix la quantitat i el tipus de recursos que es poden utilitzar al núvol, com l'accés als espais de l'edifici. Aquesta quota s'associa a l'arrendatari o usuari. Els usuaris es creen al Servei d'identitat o Keystone i se'ls assigna a l'arrendatari o tenant.

Aquests comptes d'usuari són el que els nostres consumidors (l'equivalent als treballadors en l'exemple de les oficines) utilitzaran per iniciar sessió al núvol i consumir els recursos

del núvol en el context de tenant als quals estan assignats. Si es desitja, les accions i polítiques es poden definir als diferents serveis del núvol, per part de l'administrador, que descriuen quins tipus d'accés es permet.

Els rols es poden assignar als usuaris que, en última instància, els donen accés al subconjunt exacte de recursos del núvol que necessiten. El concepte ha anat evolucionant, i a partir de la versió 3 de l'API, els rols es creen i s'assignen a nivell de grup en comptes d'usuari, simplificant la gestió i la administració. Les funcionalitats dins d'Openstack son molt canviants i fan que d'una versió a una altra tot sigui molt diferent i cal estar molt al cas tant per migrar-los si s'utilitzen entorns amb diferents versions, de fet en aquest mateix projecte, a l'avaluar diferents entorns d'Openstack m'he trobat amb aquest problema.

També pot existir una entitat més gran anomenada Domini. Un domini pot abastar diversos tenants (o projectes) i usuaris. Un administrador de dominis pot gestionar tots aquests tenants, els seus usuaris i rols dins del domini.

En el nostre exemple de l'edifici d'oficines, el domini podria representar l'espai d'arrendament de l'empresa a l'edifici i els inquilins podrien representar departaments o unitats de negoci individuals de la companyia. Els usuaris poden associar-se amb diversos inquilins i poden tenir funcions diferents assignades a ells en els diferents inquilins amb els quals estan associats. En resum, l'accés als recursos d'Openstack es concedeix als inquilins i estan restringits per quotes.

Opcionalment, les accions i les polítiques es poden definir i associar-se als rols, que al final s'assignen als usuaris, definint de manera efectiva com aquests usuaris poden utilitzar els recursos en el núvol contractats pels inquilins. Els inquilins i els usuaris es poden agrupar en dominis que permeten l'administració centralitzada d'aquestes entitats per part d'administradors de dominis.

Aquest canvi, ha permès tenir com subentorns gestionats per administradors diferents podent delegar les funcions administratives a més d'un nivell.

Neutron

En un principi el servei de xarxa el proveïa el mateix projecte de computació Nova. El servei de xarxes de Nova era molt limitat i va sorgir el projecte Neutron que va estar dissenyat des d'un principi com un servei extensible.

D'aquesta manera diferents proveïdors (fabricants) de xarxes podien crear connectors a Neutron que permetien solucions definides pel programari de xarxes (SDN).

Així s'amplia en concepte d'OpenStack i apareixen connectors per exemple per VMWare NSX o per Cisco pels seus switchos nexus.

Neutron proporciona tant la capa 2 com la capa 3 del model OSI de xarxa. És un component que pot cobrir qualsevol necessitat que se'ns pugui plantejar a nivell de xarxes.

En Neutron es poden distingir dos tipus de xarxes:

- Les d'infraestructura, on comunicarem amb els diferents serveis: administració, Storage i les definides segons necessitat (cloud-SDN que veiem en el següent gràfic)
- Les de cloud, que son les que pròpiament consumiran els usuaris des de l'exterior.

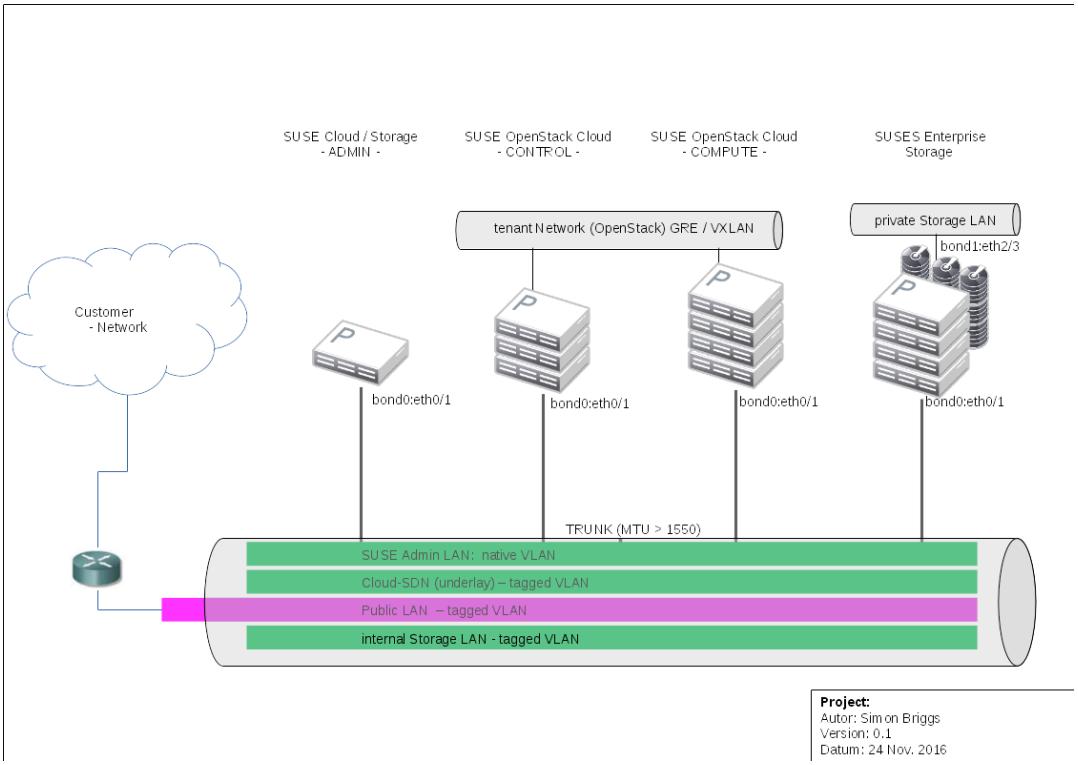

Figura 6 - 4 Tipus de xarxes

Les xarxes definides segons necessitat (cloud-SDN) es reparteixen en tota la infraestructura de manera distribuïda i tenen un component en cada node on s'implementen diferents protocols i/o capes del model OSI.

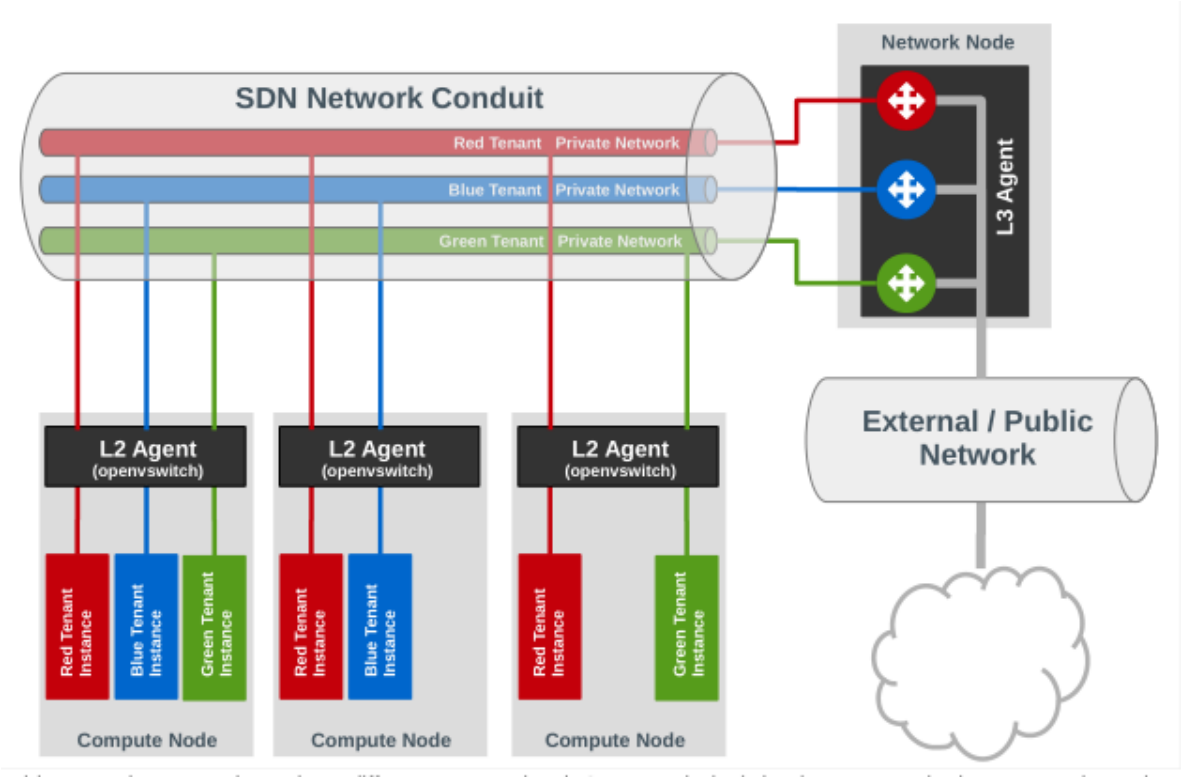

Figura 6 - 5 Components de Neutron

Els dos elements que conformen Neutron són els serveis de capa 2 i de capa 3.

Els serveis de capa 2 corren a mode d'agent en els nodes de computació i son els responsables de crear i mantenir les xarxes que interconnecten les diferents instancies de computació que tenen executant els usuaris.

Depenent de l'arquitectura final que es tingui, es podrà disposar de nodes de xarxa que s'encarregaran de totes les tasques de routing de les diferents xarxes, tot i que també es pot optar per tenir distribuïts aquest serveis.

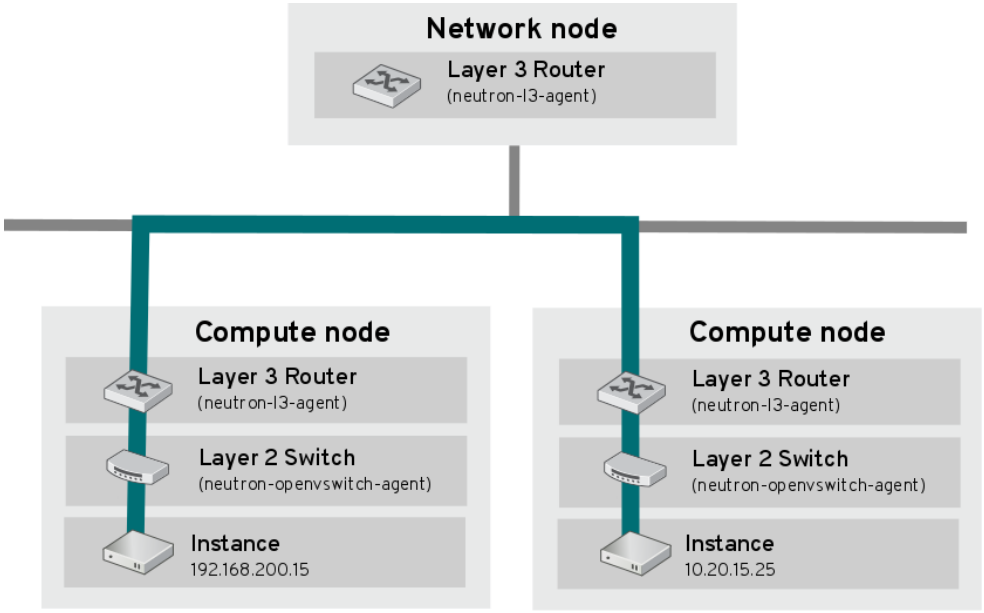

Figura 6 - 6 Terminologia de Neutron

Cal tenir en compte la terminologia que fa servir OpenStack relatiu a les xarxes [29]i fer una abstracció del terme en el món físic i adaptar-lo a núvol:

- Networks per OpenStack es refereix als "cables de xarxa" virtuals creats per als consumidors del núvol. El mecanisme d'implementació d'aquestes xarxes poden ser protocols com ara túnels GRE, VLAN o VXLAN.
- Subnets per OpenStack són les subxarxes IP associades i que s'executen en aquestes xarxes.

  És possible tenir diverses subxarxes associades a una única subnet però el més habitual és tenir només una subxarxa que s'executi en una subnet.
- Router per OpenStack el fa servir per anomenar els enrutadors que connecten subxarxes. Els enrutadors en OpenStack només poden tenir una interfície externa, però poden tenir diverses interfícies internes.

[29] https://ilearnstack.com/2013/10/07/networking-in-OpenStack-panoramic-view/

Els tenants creen els enrutadors per permetre que les seves instàncies es comuniquin amb el món extern i amb altres instàncies que es puguin connectar a altres xarxes o subxarxes que hagin creat.

- Gateway quan s'utilitza en el context d'OpenStack, fa referència a la interfície externa d'un enrutadors (és a dir, la interfície de la passarel·la del Router).
- Els pools d'assignació o Allocation pools en OpenStack són grups d'adreces que es poden utilitzar per a diferents coses, com els àmbits de DHCP en xarxes internes o grups d'IP flotants a xarxes externes.
- Les adreces IP flotants o Floating IP addresses en OpenStack, són una adreça IP que existeix a les xarxes d'OpenStack per consumir des de l'exterior del cloud. Es poden associar amb instàncies que s'executen en xarxes privades per permetre accedir a aquestes instàncies directament des del món extern.

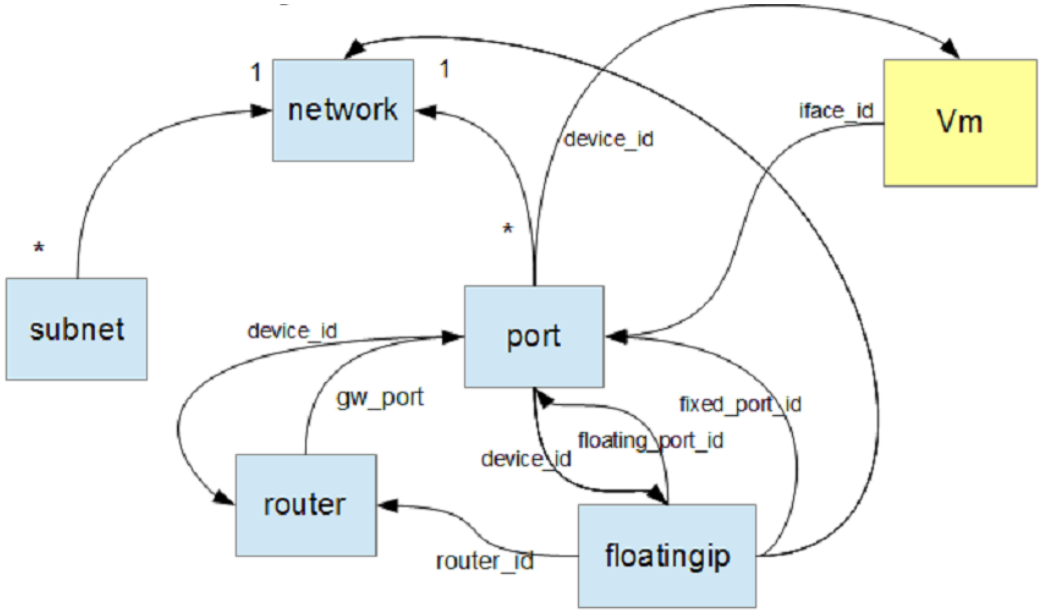

Figura 6 - 7 Relació entre components de Neutron

En OpenStack existeixen adreces IP flotants a les xarxes externes. Aquestes adreces es poden assignar als inquilins i els inquilins poden associar-les amb instàncies que s'executen a les seves xarxes privades.

El nombre d'adreces IP flotants que es poden assignar a un inquilí o tenant es pot restringir mitjançant quotes.

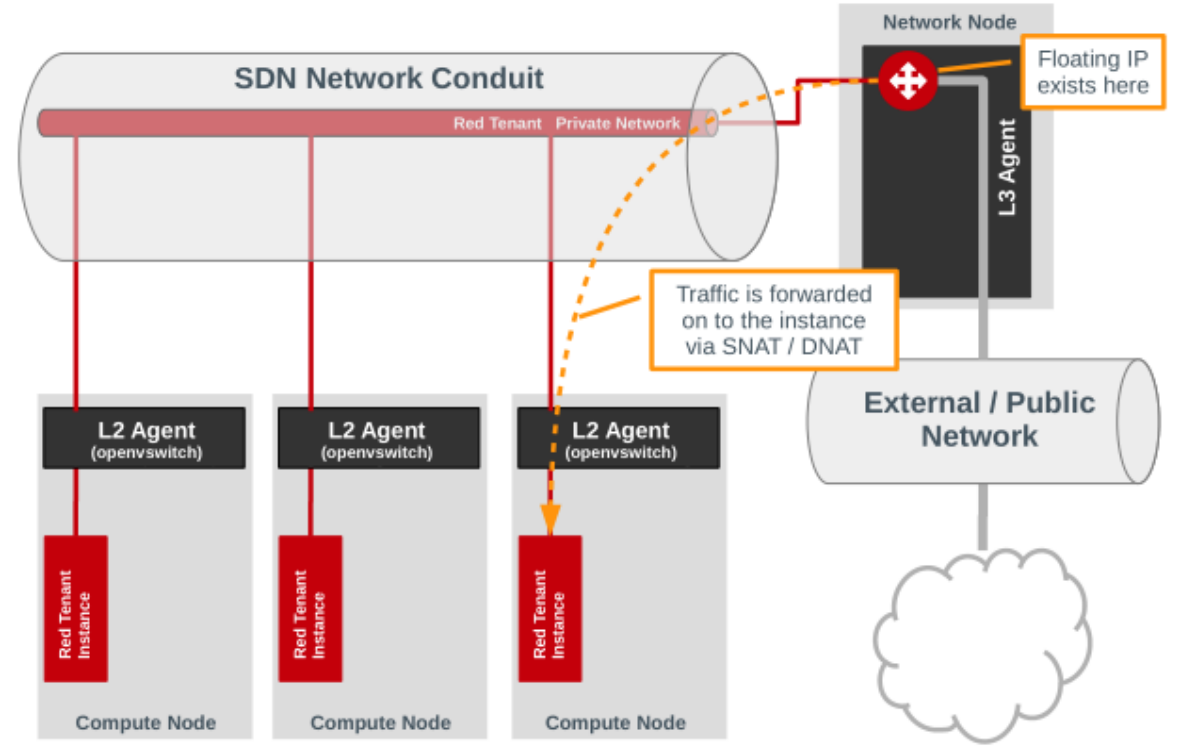

Figura 6 - 8 Esquema trànsit de xarxa entre instancia i IP flotant

Les adreces IP flotants, quan s'assignen a un inquilí, es creen a la interfície externa dels enrutadors virtual del projecte. Quan s'associa a una instància en una xarxa privada, el trànsit de xarxa és fa per SNAT / DNAT entre la instància i l'exterior permetent que el recurs sigui accedit pel món extern.

A OpenStack, el procés d'assignació d'una IP flotant a un inquilí s'anomena "creació d'una IP flotant", de fet, la IP no s'està creant perquè prové d'un grup d'IP flotants que l'usuari del núvol havia creat / definit prèviament. Només s'està "creant" a la interfície externa de l' enrutador d'inquilins.

Els grups de seguretat o security groups són un mètode per simplificar la creació de regles de Firewall per a les instàncies. Permeten definir regles de tallafocs per a diferents protocols / ports i, a continuació, agrupar-los. Aquests grups es poden associar amb instàncies.

Només cal definir les regles del tallafoc una vegada al grup de seguretat enlloc de definir-les cada vegada que es comença una nova instància de firewall.

El nombre de grups de seguretat creats per un inquilí i la quantitat de regles creades per un inquilí es poden restringir per mitjà de la quota.

Això és important perquè la creació i la gestió de les regles del tallafocs poden generar despeses de computació significatives al cloud quan hi ha un gran nombre d'inquilins amb grans quantitats de casos i usos. En general l'ús de les quotes és el que permet, en tots els recursos d'OpenStack limitar-ne l'ús i tarificar-los.

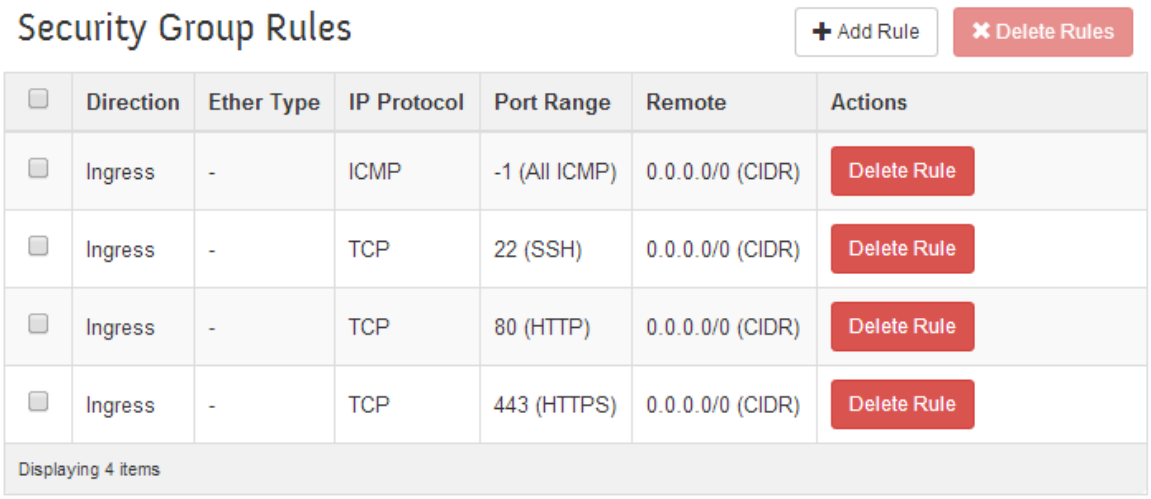

Figura 6 - 9 Grups de seguretat

Glance

El servei d'imatges d'OpenStack, també conegut com Glance, ofereix els serveis de registrar, descobrir i recuperar imatges de màquines virtuals.

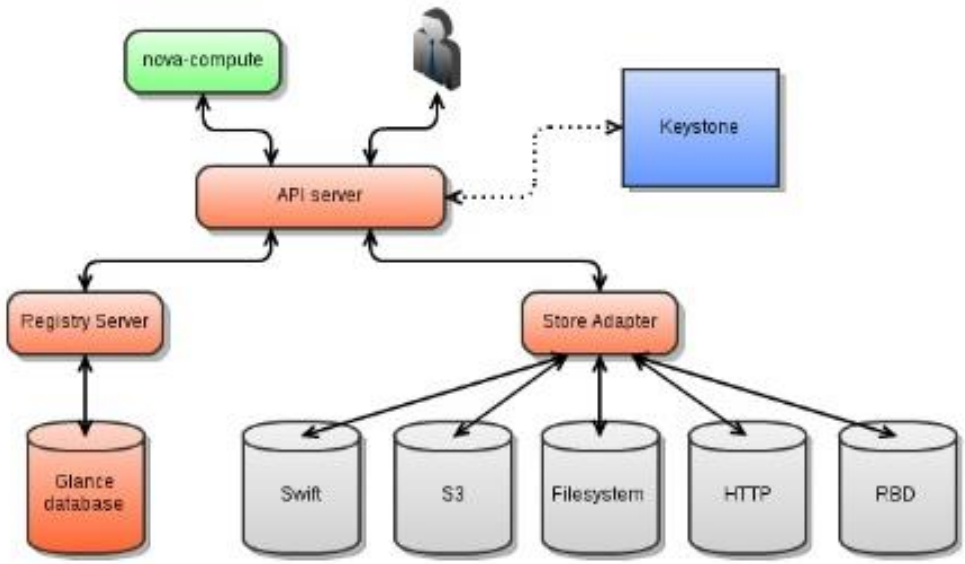

Figura 6 - 10 Esquema general de Glance

L'API d'OpenStack Image Service actua com a registrador de les imatges.

Quan un usuari vol emmagatzemar una imatge en el servei d'imatge, proporciona al registrador d'imatges el fitxer d'imatge juntament amb informació o metadades que la descriuen: el seu nom, el tipus de disc, la seva arquitectura de la CPU, el tipus d'hipervisor, etc.

El registrador d'imatges llavors emmagatzema la imatge en un contenidor i crea una entrada al registre que conté la ubicació del fitxer d'imatge i tota la informació addicional que s'ha proporcionat.

És important tenir en compte que, tot i que es poden canviar les metadades d'una imatge una vegada que es troba al registre d'imatges, la imatge és immutable i qualsevol canvi implica la recreació d'aquesta.

Quan el servei Compute vol llançar una instància de càrrega de treball, li diu al registrador d'imatges quina imatge necessita. El registrador d'imatges recupera la ubicació del fitxer de la imatge des del registre i el dóna al servei de computació. El node Compute que executarà la instància baixarà una còpia del fitxer d'imatge directament des de la ubicació proporcionada i la utilitzarà per iniciar la instància.

Les metadades addicionals emmagatzemades a l'entrada d'una imatge s'utilitzen per determinar la idoneïtat d'un fitxer d'imatge per a una situació específica. Per exemple, en el cas de disposar de diferents còpies d'una imatge específica (per exemple de Linux), cada una d'elles en un format d'imatge de disc diferent (per exemple el cas de ser per diferents hipervisors: VMWare, KVM, ...), s'utilitzen les metadades per identificar quina imatge és la correcta per cada situació.

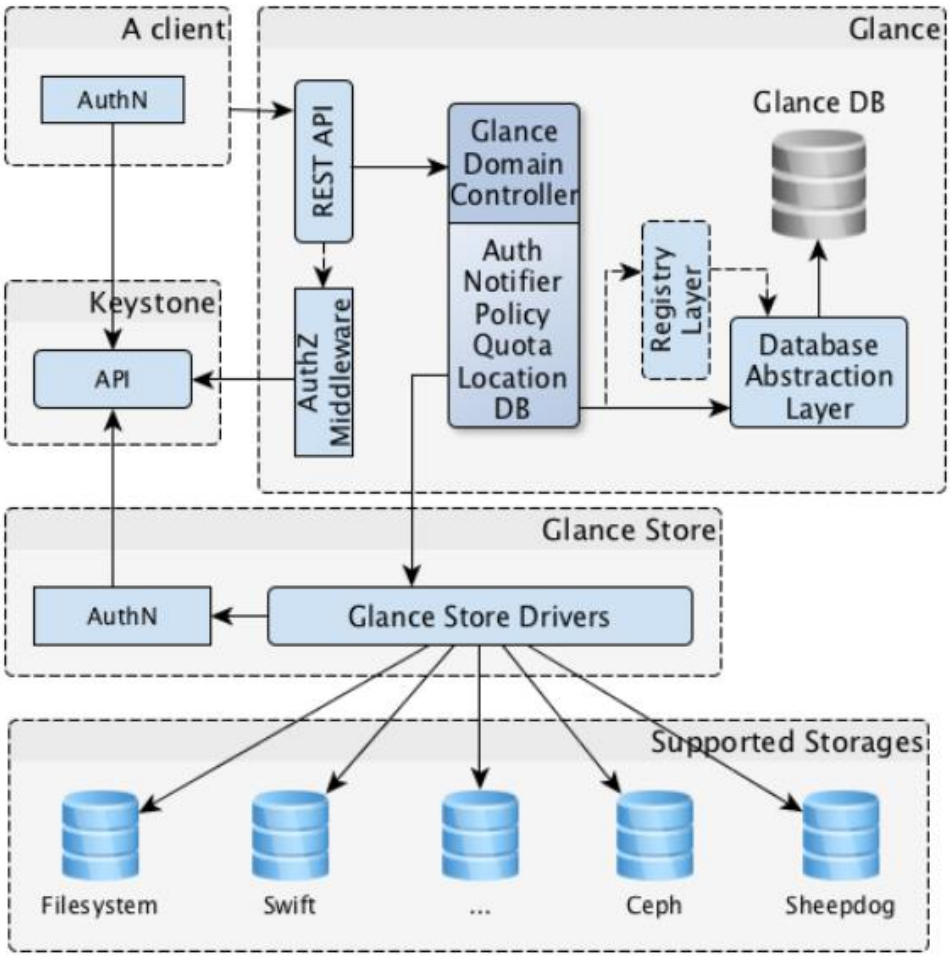

Figura 6 - 11 Esquema de funcionament de Glance

L'emmagatzematge utilitzat per guardar els fitxers d'imatge pot ser tan senzill com un disc connectat al servidor que executa els serveis de Glance, o pot ser més accessible, com ara una biblioteca d'objectes Swift o un dispositiu de bloc RADOS a Ceph.

Com que el registre d'imatges s'emmagatzema en una base de dades, es pot escalar el servei d'imatges afegint instàncies d'API addicionals o registradors d'imatges segons calgui.

Cada registrador d'imatges pot comunicar-se amb la base de dades de manera independent per afegir-ne de noves i recuperar o actualitzar informació sobre les existents.

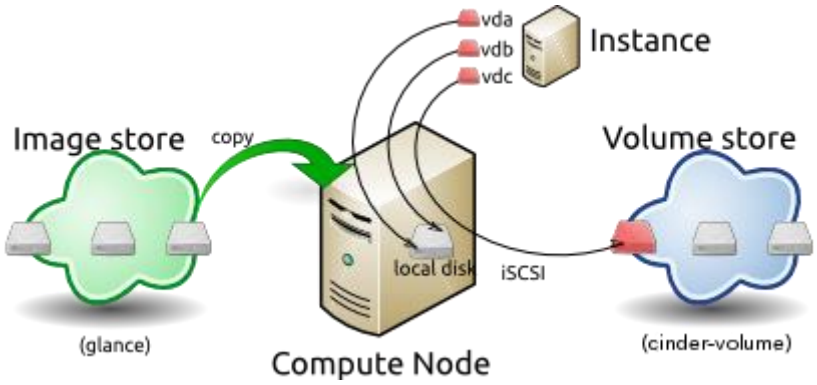

Figura 6 - 12 Esquema emmagatzematge d'Openstack

Nova

El servei Compute d'OpenStack, o Nova, és el que proporciona la provisió, implementació i retirada de instàncies de càrrega de treball en un núvol d'OpenStack. Nova va ser un dels primers serveis d'OpenStack, ja que és indispensable per poder executar-les.

Hi ha dues funcions principals que es proporcionen dins del servei Compute:

- Compute Controller: la seva principal funció és proporcionar serveis com ara determinar quina serà la instància de càrrega de treball i programar on s'inicia una instància. També proporciona l'API utilitzada per iniciar i gestionar les instàncies de càrrega de treball.

- Compute Node. comunica directament amb els hipervisors i gestiona el llançament i la gestió reals de les instàncies de càrrega de treball.

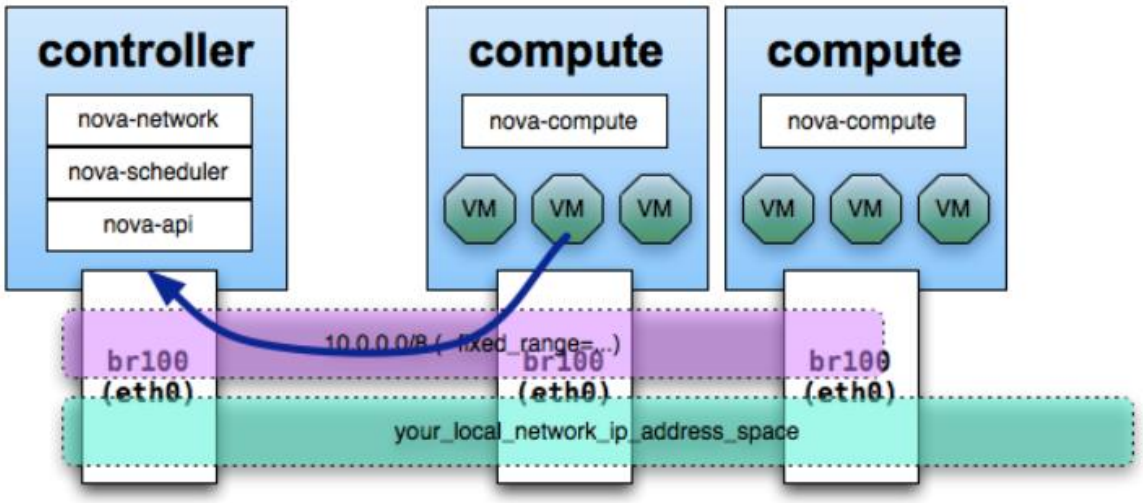

Figura 6 - 13 Comunicació Controller i Nova

Una instància de càrrega de treball és una combinació de dos elements:

- La configuració que descriu el que és la instància
- L'emmagatzematge de blocs que s'utilitzarà per als seus discs.

Els paràmetres de configuració bàsics utilitzats per les instàncies de càrrega de treball es defineixen en "sabors" (flavors). Aquests sabors defineixen paràmetres com quants VCPUs i la quantitat de memòria que tindrà i quants discs s'adjuntaran a ella i les seves mides. Normalment aquests sabors els defineixen els usuaris del núvol, però poden venir donades pels administradors.

Quan un usuari vol llançar una instància de càrrega de treball, primer selecciona el sabor que vol que sigui la instància.

A continuació, selecciona la imatge que vol arrencar des de la instància. Selecciona les xarxes que vol connectar a la instància i els grups de seguretat que vol que sigui membre.

Quan s'inicia la instància, el Compute Controller determina quin node Compute executarà la instància i, a continuació, lliurarà tota aquesta informació sobre la instància a aquest node.

El node Compute recupera el fitxer d'imatge o els fitxers especificats i indica al hipervisor que inicii la instància.

També indica a l'hypervisor que adjunti la instància a les xarxes especificades i els volums d'emmagatzematge persistents.

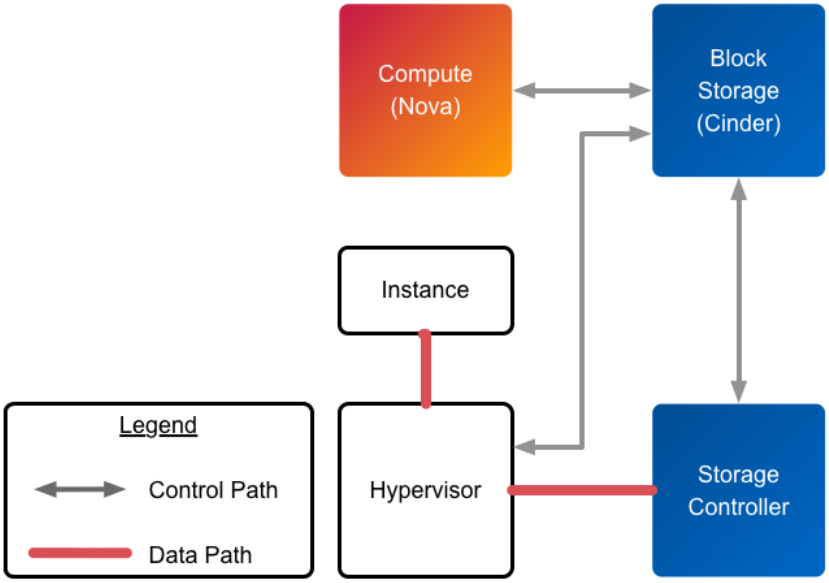

Figura 6 - 14 Relacions entre Nova i Cinder

Si les instàncies en execució necessiten migrar-se a diferents nodes de còmput, el controlador de Compute gestiona això actualitzant la seva base de dades.

Després es comunica amb els nodes de Compute afectats que es comuniquen amb els seus hipervisors.
Quan una instància ha de ser rescindida, el Compute Controller també gestiona això d'una manera similar, actualitzant la base de dades i comunicant-se amb el node de Compute relacionat.

Tota la informació sobre les instàncies vives de càrrega de treball s'emmagatzema en una base de dades i per tant es pot escalar el servei del controlador de Compute afegint nodes addicionals de controlador de computació.

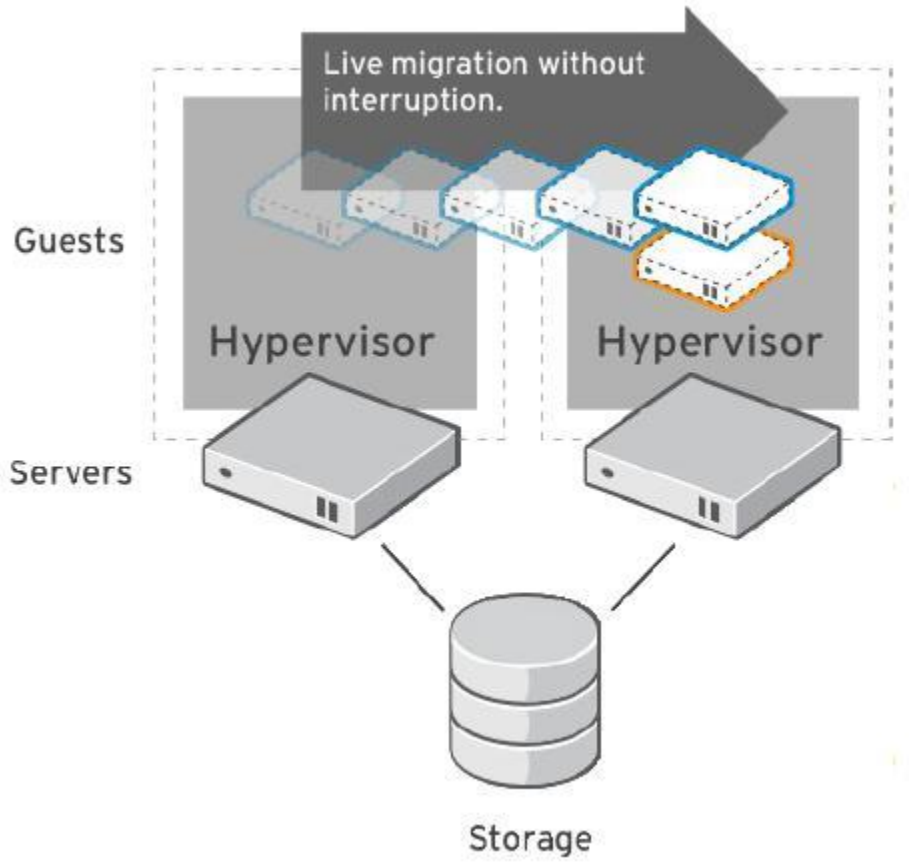

Figura 6 - 15 Migració d'instàncies entre nodes de Computació

En resum, el Servei de Computació d'OpenStack o Nova és responsable de gestionar el cicle de vida de les instàncies de càrrega de treball en un núvol d'OpenStack fent un seguiment de les instàncies vives, comunicant-se amb els hipervisors que executen aquestes instàncies.

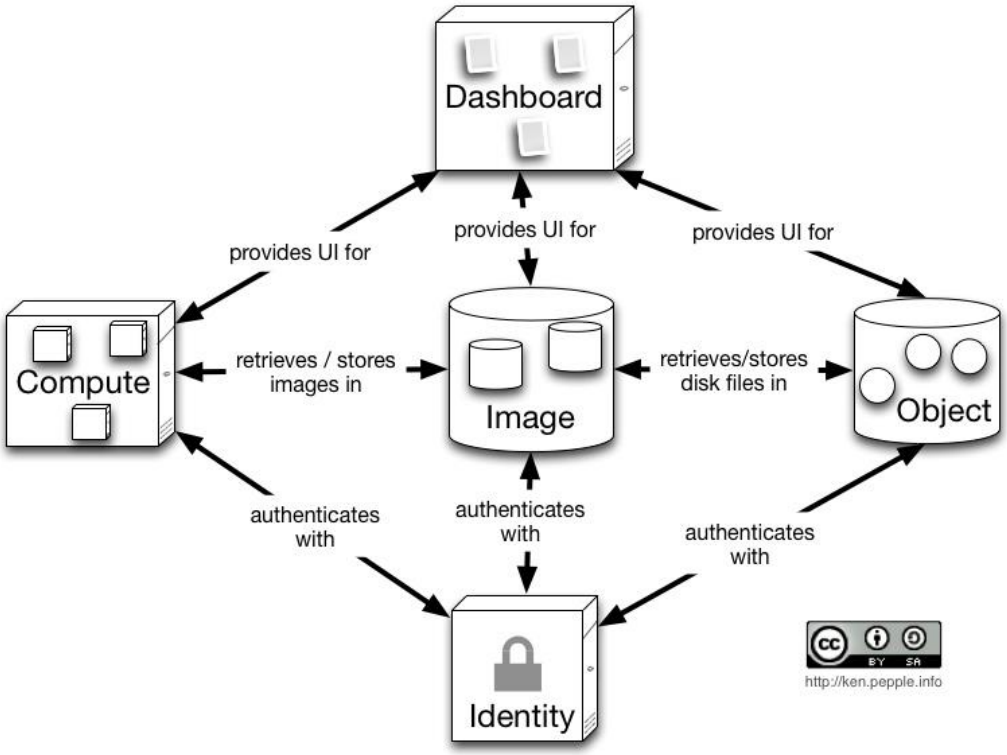

Figura 6 - 16 Relació entre Nova amb la resta de serveis

Depenent de l'hypervisor que utilitzem es comportarà d'una manera o altra, i com en tot a OpenStack hi haurà una API diferent per cadascun d'ells.

KVM o Xen

Els nodes del controlador són els responsables de programar i generalment gestionar tot el treball de computació realitzat pel núvol d'OpenStack.

Es comuniquen amb els serveis de computació que s'executen en els nodes de còmput. Els serveis de computació també interactuen amb els serveis d'hipervisors. Quan s'utilitzen els hipervisors KVM o Xen, el servei de comprovació Nova s'executa directament al node que executa l'hypervisor.

Els serveis del controlador Nova i els serveis de computació Nova es comuniquen mitjançant el servidor o servidors de la cua de missatges. A causa d'aquesta comunicació de missatges basada en cues, és possible escalar fàcilment els nodes del control o els nodes de còmput en funció de la capacitat addicional necessària.

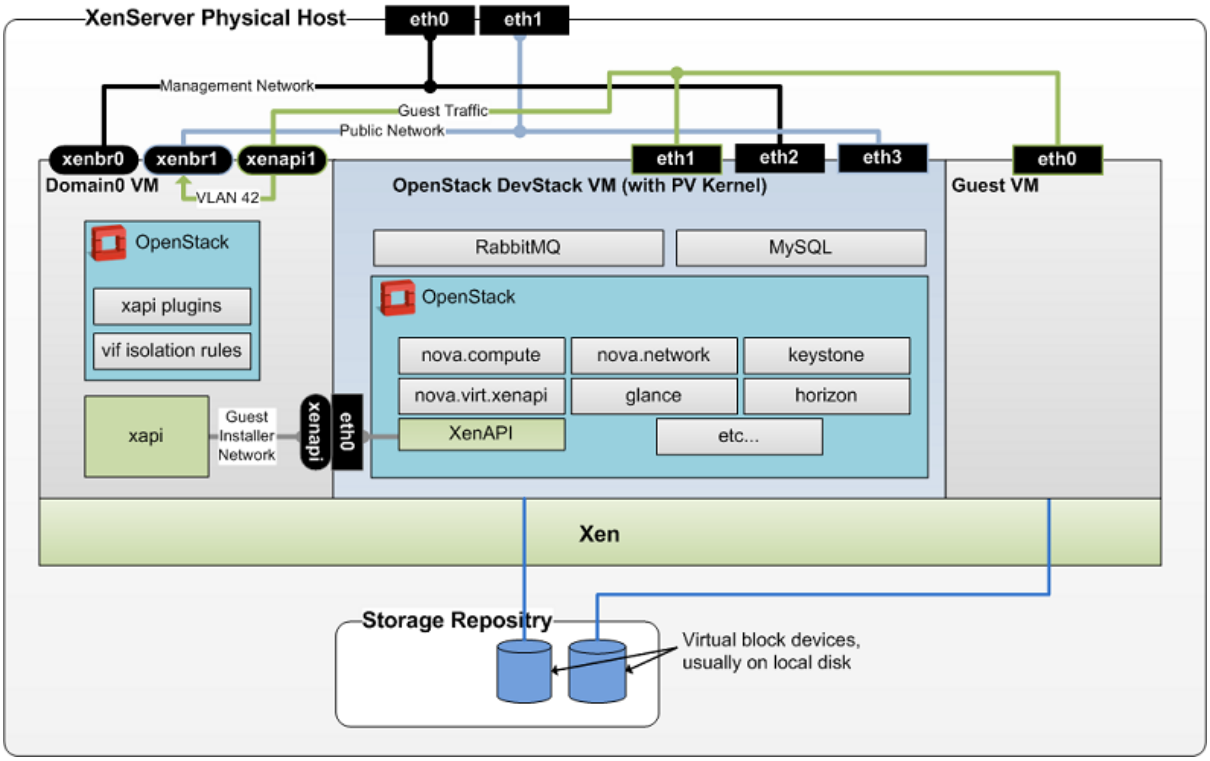

Figura 6 - 17 Esquema arquitectura hypervisor Xen

VMWare

Quan s'utilitza l'hypervisor VMWare, el servei de comprovació Nova no s'executa en els mateixos nodes que l'hypervisor, s'executa en un node separat o en un conjunt de nodes. Aquests nodes de comprovació es comuniquen amb els hipervisors de VMWare.

El controlador Nova Compute es comunica amb els nodes Compute de la mateixa manera, els nodes de computació només han d'interactuar amb els hipervisors a través de la xarxa en lloc de localment com amb KVM i Xen.

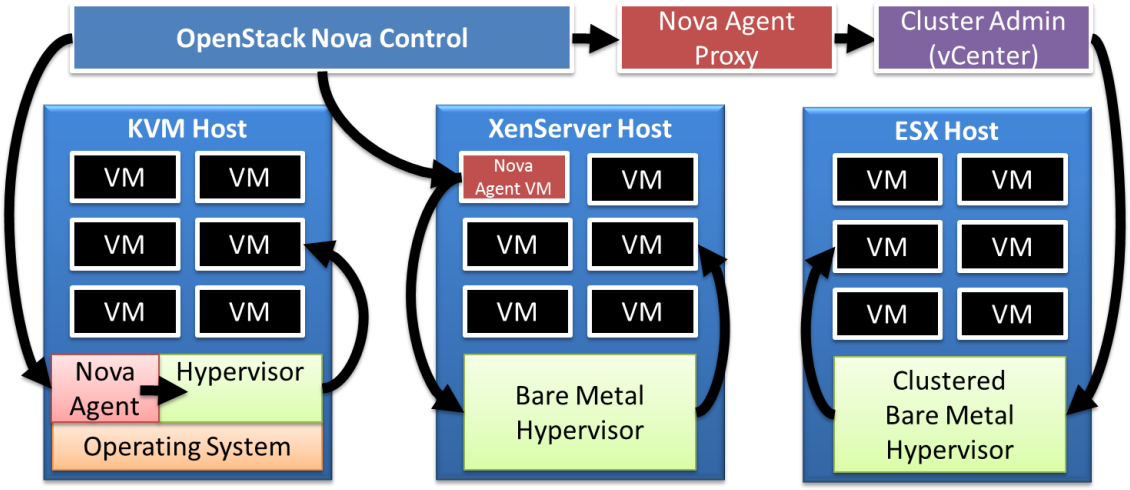

Figura 6 - 18 Esquema arquitectura hypervisor VMWare

Les instàncies de càrrega de treball al núvol, o simplement instàncies, son les màquines virtuals (VM's).

Aquestes instàncies estan formades per CPU virtuals, memòria, discs (que són de naturalesa efímera), volums (emmagatzematge persistent) i xarxes.

Les instàncies poden tenir mides múltiples i diferents d'aquests components computacionals.

Tal com es pot veure en el gràfic següent, una imatge de sistema operatiu pot tenir variants (flavors) que dependran del la CPU, memòria, disc, ... que tenen assignats.

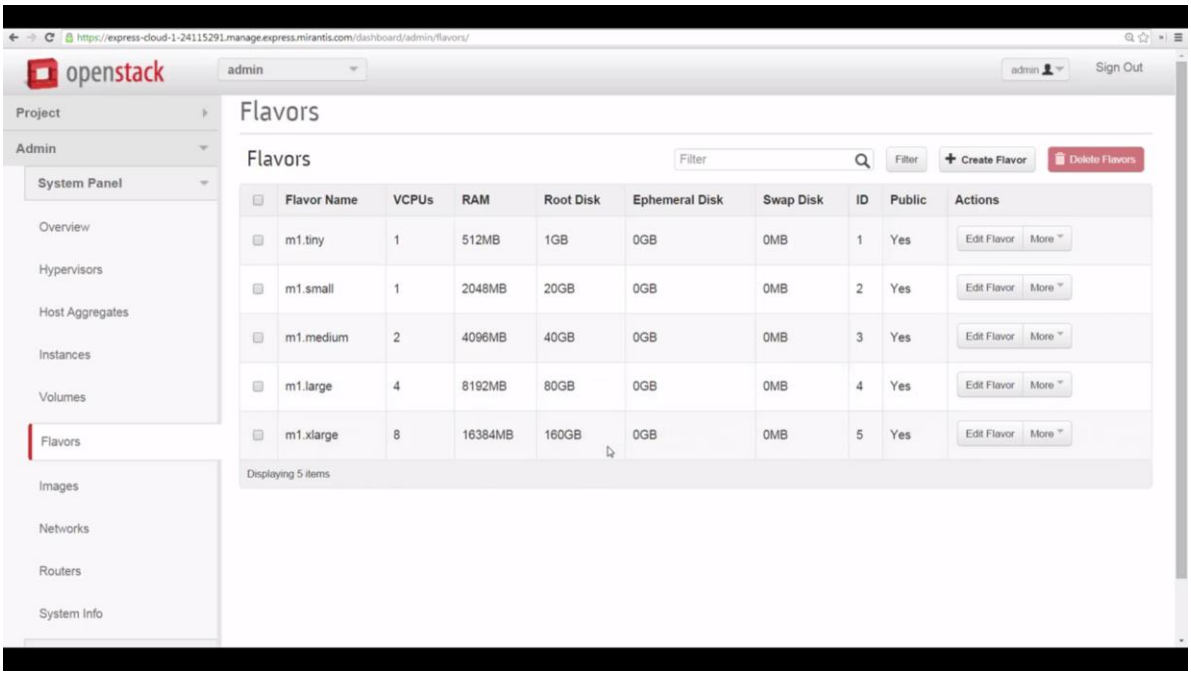

Figura 6 – 19 Exemple de Flavors a Horizon

Cinder

Hi ha dos tipus d'emmagatzematge de blocs en un entorn de núvol: efímer i persistent.

L'emmagatzemament efímer de bloc sol ser el fitxer d'imatge de la màquina virtual que s'utilitza com a disc arrel d'una instància de càrrega de treball quan s'està executant.

Es poden afegir discs efímers addicionals a una instància de càrrega de treball per a usar-los com a espai d'emmagatzematge o intercanvi addicional.

L'emmagatzematge efímer de blocs s'anomena "Ephemeral" perquè només existeix mentre existeixi la instància. Quan aquesta finalitza, es suprimeixen tots els seus discs efímers.

Si es volen mantenir dades més enllà de la vida útil d'una instància, caldrà l'emmagatzematge de bloc persistent que s'anomenen volums.

Els volums són un emmagatzematge de bloc persistent que es pot connectar a instàncies

de càrrega de treball en execució. Quan es finalitza una instància de càrrega de treball, el volum queda intacte i es pot adjuntar a una altra instància en el futur.

Els volums també es poden utilitzar com a discs root de les instàncies de càrrega de treball. Això es coneix com "boot from volume".

En aquest cas, quan es fa una instància nova, en lloc de fer una còpia d'un fitxer d'imatge i arrencar directament des d'ell, el contingut del fitxer d'imatge s'escriu en un volum i la instància de la càrrega de treball arrenca des del volum.

Això permet que un volum de root, amb tota la seva configuració intacta, sigui utilitzat per diferents instàncies de càrrega de treball al llarg del temps.

Un altre avantatge de l'arrencada des dels volums és que pot fer que la migració de les instàncies en execució des d'un node Compute sigui més fàcil.

El servei de bloc d'OpenStack o Cinder és el que proporciona aquests volums. L'API del servei de blocs actua com a intermediari entre els nodes Compute que executen les instàncies de càrrega de treball i els nodes d'emmagatzematge on resideixen els volums físicament.

El servei de bloc d'OpenStack admet una àmplia gamma de servidors d'emmagatzematge. Un simple motor d'emmagatzematge podria ser volums tradicionals LVM, que estan en els mateixos servidors de la infraestructura, que s'exporten com a destins iSCSI o LUNS.

Darrerament el projecte Ceph[30] està entrant amb força al mon OpenStack, ja que dona tota la funcionalitat que té Cinder i n'afegeix d'altres de molt potents. De fet Ceph per si sol està essent la solució d'emmagatzematge per molts projectes o es requereix escalabilitat tant en rendiment com en capacitat.

Implementa funcions molt interesants com emmagatzematge d'objectes, de bloc i d'arxius, així com de rèplica, ja que al ser un sistema amb clúster i distribuït, el porta integrat implícitament.

[30] http://ceph.com/

Una altra opció és utilitzar els controladors que permeten que les SANs i les NAS allotgin els volums. Els fabricants tradicionals d'emmagatzematge han desenvolupat les APIs per tal que Cinder parli amb ells i puguin servir de repositori.

La raó per la qual el servei de bloc d'OpenStack actua com a intermediari és que no està implicat en la connexió real entre les instàncies de càrrega de treball i els seus volums.

Quan el servei Compute vol llançar una nova instància de càrrega de treball que està connectada a un volum, es posa en contacte amb el Service Block per obtenir la URI del volum que necessita connectar.

El node Compute que executarà la instància de la càrrega de treball usarà aquesta URI per anar directament al backend on resideix físicament el volum i la connectarà a la instància.

Es poden afegir nodes de bloc segons necessitats sense cap impacte en el funcionament del nostre cloud. Això és així perquè la informació sobre volums i la seva ubicació física s'emmagatzema en una base de dades.

En resum, el servei de bloc OpenStack, o Cinder, s'utilitza per proporcionar instàncies de càrrega de treball amb accés a volums d'emmagatzematge persistents.

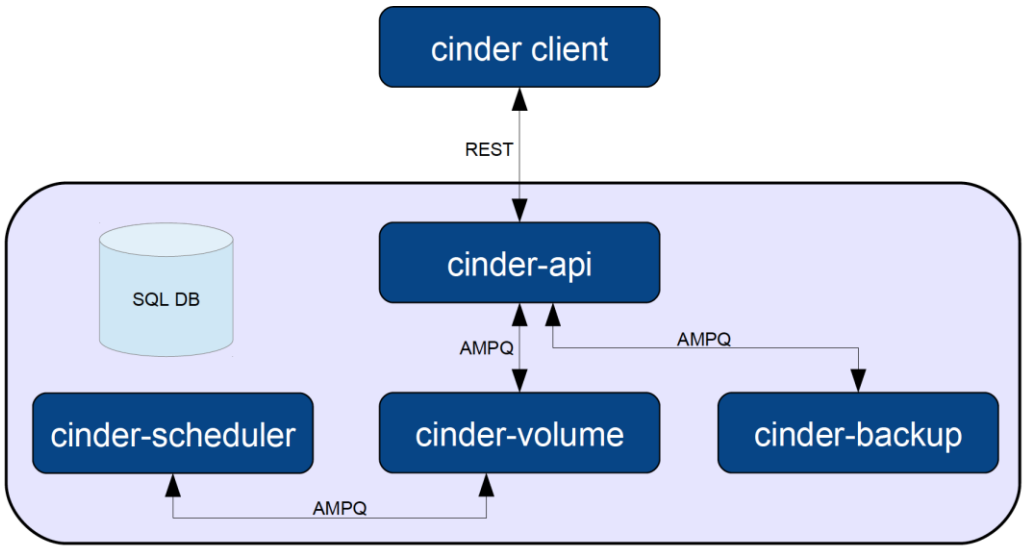

Figura 6 - 20 Esquema de funcionament de Cinder

Cinder + iSCSI

El servei Nova Compute es comunica amb l'API de Cinder que s'executa al servidor de Cinder per descobrir la ruta cap al volum que s'adjunta a la instància.

Quan s'utilitza un backend iSCSI, la ruta d'accés al volum es retorna com un LUN iSCSI.

El servei de comprovació Nova passa això al hipervisor que, a continuació, connecta la instància directament a la LUN iSCSI en el servidor iSCSI que pertoqui.

El servidor iSCSI pot executar-se en el mateix node que el servei Cinder API (el servidor Cinder) o es pot executar en el servidor o servidors de la infraestructura d'emmagatzematge.

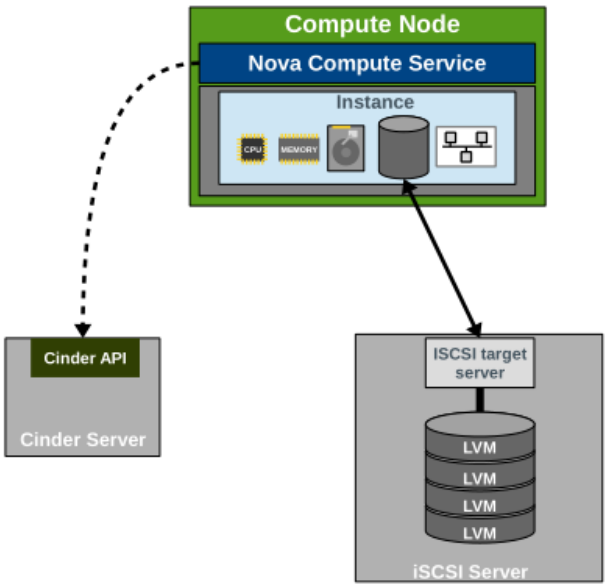

Figura 6 - 21 Esquema de funcionament de Cinder + iSCSI

Cinder i NFS

De la mateixa manera que s'utilitza un backend iSCSI, amb un backend de NFS, el servei Nova Compute es comunica amb l'API de Cinder que s'executa al servidor de Cinder per descobrir la ruta d'accés al volum que s'adjuntarà a la instància.

Tanmateix, quan s'utilitza un backend d'emmagatzematge NFS, la ruta d'accés al volum que es retorna indica un fitxer raw en un export de NFS.

El servei de comprovació Nova passa això al Hipervisor que després munta l'export de NFS i, a continuació, connecta la instància directament al fitxer raw en l'export de NFS.

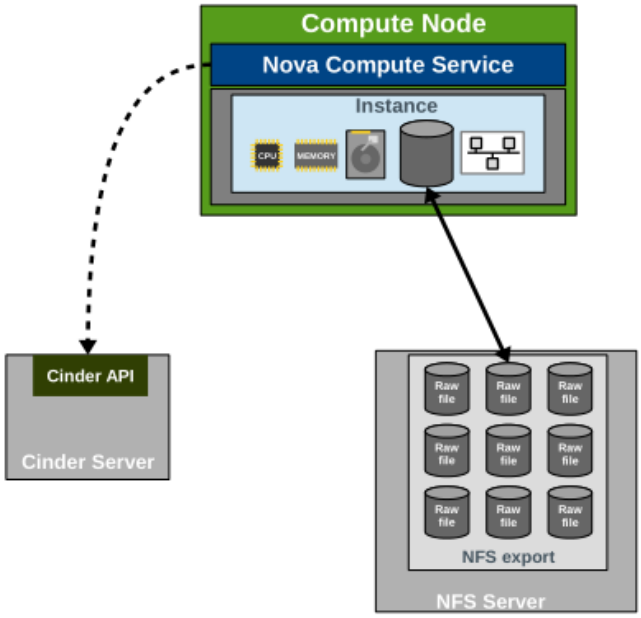

Figura 6 - 22 Esquema de funcionament de Cinder + NFS

Cinder i SAN/NAS

Cinder també disposa de controladors que li permeten treballar amb SAN i NAS tradicionals.

La connexió dels volums a les instàncies es realitza d'una manera similar al iSCSI. El servei de comprovació de Nova es comunica amb l'API de Cinder per descobrir la ruta cap al volum. El camí passa a l'hypervisor que connecta el volum SAN/NAS directament a la instància.

La majoria de fabricants: HP, EMC, NetApp han desenvolupat integracions amb OpenStack.

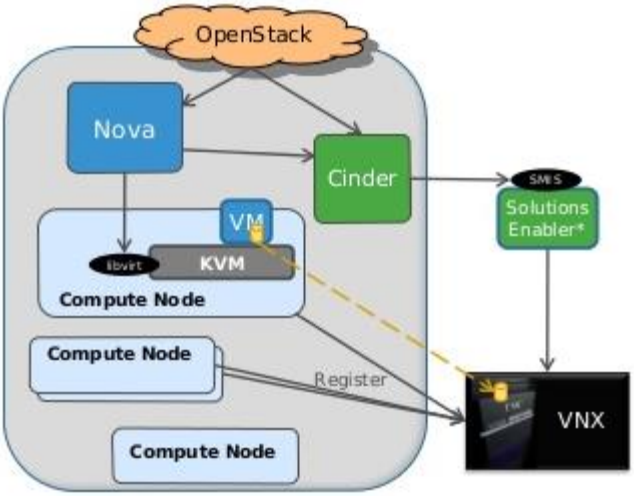

Figura 6 - 23 Esquema de funcionament de Cinder + SAN

Cinder i Ceph

Cinder també disposa d'un controlador de backend d'emmagatzematge que li permet utilitzar un Rados Block Devices (RBD), que resideix en un clúster Ceph com a volums.

El servei de comprovació Nova recupera la ruta d'accés a la RBD des de l'API de Cinder i l'envia a l'hypervisor. L'hypervisor utilitza controladors RBD natius per connectar el volum RBD directament a la instància.

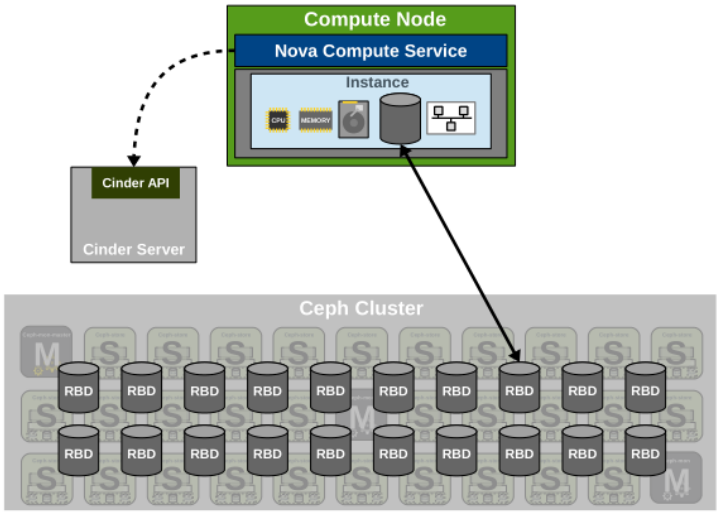

Figura 6 - 24 Esquema de funcionament de Cinder + Ceph

Quan s'inicia una instància d'un volum, el sistema operatiu i tota la seva configuració s'emmagatzemen en un volum en comptes d'un disc efímer, permetent conservar aquestes dades quan finalitzi la instància, això és conegut com a "boot from volume".

Posteriorment, l'usuari podria iniciar una nova instància, arrencant-la des d'aquest volum, i la instància seria idèntica a la instància original, podent-la clonar, replicar-la en moltes d'iguals o simplement continuar treballant sense perdre cap dada.

Un altre avantatge d'arrencar instàncies de volums està relacionat amb la migració en calent:
Quan una instància utilitza un disc efímer com el seu volum d'arrel, si aquesta instància s'ha de migrar a un altre node Hipervisor, la VM ha de quedar en pausa, els discs efímers d'aquesta instància es copien al nou hipervisor i després es treu la pausa de la VM.
En el cas d'emmagatzematge compartit, la migració simplement requereix que la memòria de VM es mogui al nou node i no cal fer res amb l'emmagatzematge, ja que aquest és accessible des de tots els nodes.

No s'ha de confondre amb el fet que si hi ha un servei funcionant i volem que les dades canviants es guardin, això no es pugui fer adjuntant qualsevol tipus d'emmagatzematge que tenim al sistema, llavors en el cas d'aturada de la instància al engegar-se tindríem un sistema net amb les dades de la instància anterior, que les hauríem d'adjuntar via un script o manualment.

Swift

L'emmagatzematge d'objectes guarda dades de manera diferent a com ho fa un sistema de fitxers.

Amb un sistema de fitxers, els fitxers s'emmagatzemen en un sistema jeràrquic de directoris i subdirectoris en un únic dispositiu de bloc. Quan accedim a un fitxer en un sistema de fitxers, hem de conèixer la ruta exacta a l'estructura del directori on resideix el fitxer.

Amb l'emmagatzematge d'objectes, els fitxers s'emmagatzemen en un espai de noms pla que pot abastar diversos servidors d'emmagatzematge amb cada fitxer amb el seu propi

identificador únic i accessible per la seva URL exclusiva. Per accedir a un fitxer en emmagatzematge d'objectes, simplement haurem de conèixer aquest URL i s'accedeix mitjançant una interfície REST basada en HTTP.

Un altre avantatge de l'emmagatzematge d'objectes és que és escalable de forma senzilla i massiva.

Pel que fa a la terminologia que fa servir Swift és la següent:

- Account: correspon a un usuari del clúster
- Container: conté objectes associats amb un compte
- Objecte: dades emmagatzemades al clúster

Per afegir espai a un contenidor d'objectes, simplement afegim servidors d'emmagatzematge addicionals. Atès que l'emmagatzematge d'objectes no es basa en una estructura de dades basada en el directori, no té les limitacions inherents a l'extensió a mides massives que fan els sistemes de fitxers tradicionals.

En un sistema de fitxers s'emmagatzemen les dades en blocs o sectors de disc i a continuació, confien en una estructura de dades per emmagatzemar les metadades del fitxer i assignar quins blocs al disc pertanyen a aquest fitxer.

En l'emmagatzematge d'objectes es guarden els fitxers, les metadades associades i l'identificador únic com a únic objecte. Això minimitza els costos generals requerits per emmagatzemar i accedir als fitxers.

Tanmateix, això suposa un desafiament, ja que amb un contenidor d'objectes, els fitxers no es poden modificar en el seu lloc. Amb un sistema de fitxers, els fitxers es poden modificar perquè només requereix assignar blocs de disc addicionals al fitxer. Amb l'emmagatzematge d'objectes, el fitxer s'ha de recuperar, en la seva totalitat, des del magatzem d'objectes abans que es pugui modificar. La versió modificada ha de tornar a pujar al contenidor d'objectes. Això vol dir que els fitxers que canvien sovint o les dades transaccionals no són bons candidats per a l'emmagatzematge d'objectes.

Els millors tipus de dades que es poden emmagatzemar en objectes són dades de caràcter estàtic o no estructurat, com imatges, vídeos o còpies de seguretat de dades.

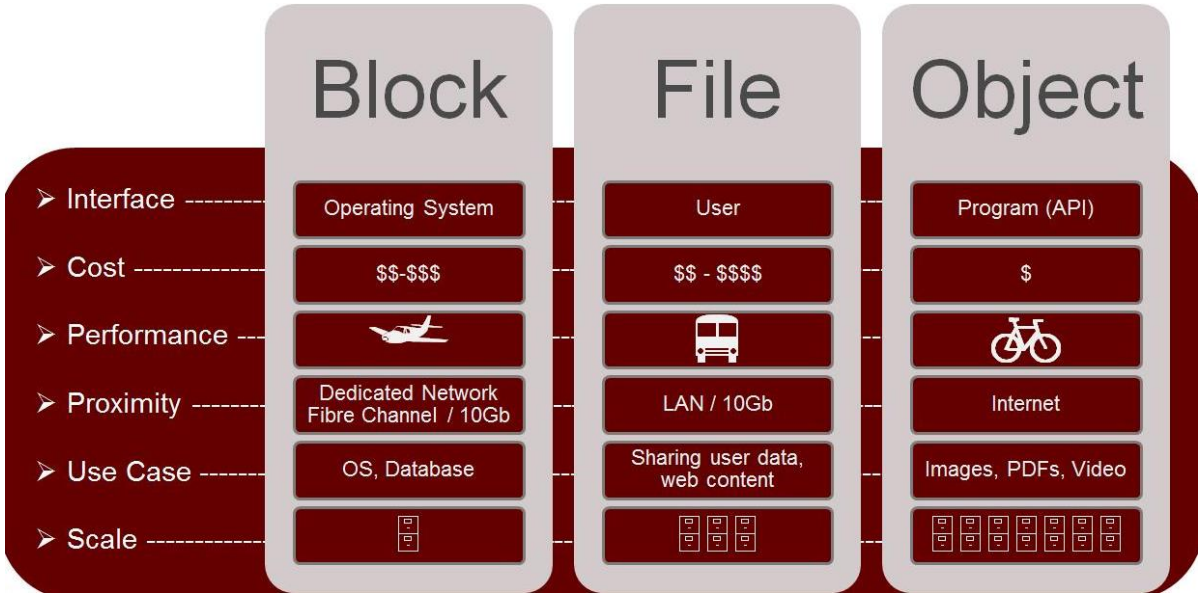

Figura 6 - 25 Usos de Swift

Així com en les arquitectures d'emmagatzematge tradicional, la redundància i/o la prevenció de les fallades es fa en base a sistemes RAID, en l'emmagatzematge d'objectes la redundància s'aconsegueix generalment a través de la replicació de dades com a còpies múltiples en diferents servidors d'emmagatzematge en un clúster.

Una característica comuna en aquestes solucions, és la reparació automàtica, o la replicació automàtica en cas de fallada. Atès que els contenidors d'objectes estan dissenyats per evitar errors, es poden construir utilitzant un maquinari més econòmic, ja que no cal que tinguin costoses controladores perquè el rendiment i la capacitat el donen els nodes, podent créixer en cas de necessitar una cosa o una altra.

Com que la majoria de les dades que es generen avui en dia són de naturalesa estàtica o no transaccional, els contenidors d'objectes es poden utilitzar per emmagatzemar aquestes dades de forma barata per tal de que siguin fàcilment i ràpidament accessibles.

En resum, l'emmagatzematge d'objectes és molt útil per a l'emmagatzematge de grans quantitats de dades que no canvien sovint.

És escalable de forma massiva i barata a causa del seu mètode d'emmagatzematge i d'accés a les dades i la seva capacitat per utilitzar el maquinari de productes bàsics

gràcies a la seva replicació i regeneració individual, ja que la redundància es dona a nivell de node.

L'accés a les dades en un contenidor d'objectes es fa a través d'una URL o una interfície RESTful basada en web, en comptes de muntar-lo com a sistema de fitxers.

Així doncs l'arquitectura de Swift la podem dividir ens dos nivells:

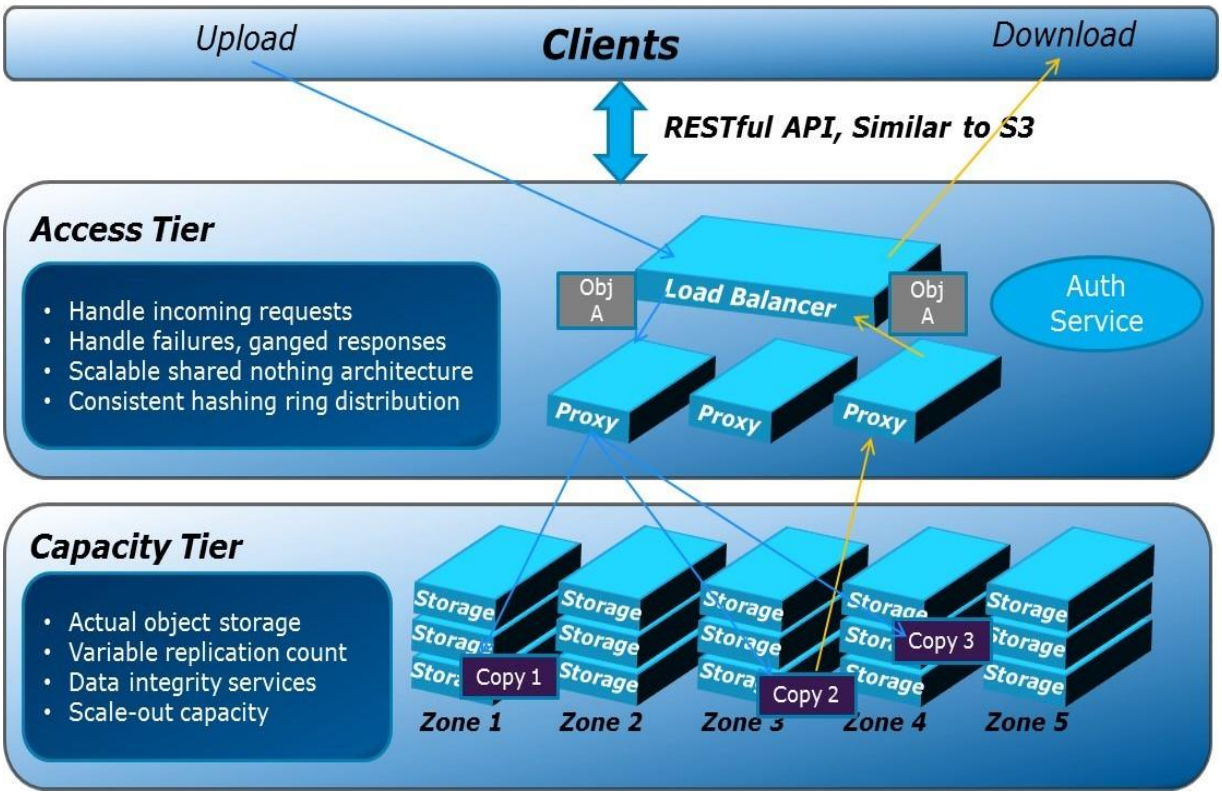

Figura 6 - 26 Arquitectura de Swift

1.- Nivell d'accés:

Balancejador de càrrega:

- Distribueix les sol·licituds en tots els servidors proxy

Servidors proxy

- Controla les sol·licituds de l'API entrants i enruta les peticions al servidor correcte
- Controla les falles

- Coordina marques de temps dels objectes
- Gestiona els objectes per facilitar l'escalat del sistema

El nivell d'accés resideix als servidors proxy de Swift. Un servidor Proxy Swift controla totes les sol·licituds d'API entrants per pujar o accedir a dades i encamina les respostes al servidor correcte en el nivell d'emmagatzematge. El servidor Proxy també controla qualsevol falla i gestiona la replicació de les dades. També coordina tots els segells temporals de totes les còpies de dades emmagatzemades al clúster.

Un servidor Proxy Swift utilitza una arquitectura de shared-nothing, que consisteix en una arquitectura distribuïda en el que cada node es independent i autosuficient i que fa que sigui molt senzill escalar quan es necessiti capacitat addicional en el nivell d'accés.

Tot el que es requereix és l'addició de servidors proxy de Swift addicionals. Si es vol un o més balancejadors de càrrega es poden col·locar davant dels servidors proxy Swift per presentar una única adreça com a punt final de l'API per accedir al clúster Swift. En definitiva qualsevol servidor Swift Proxy pot respondre a qualsevol sol·licitud d'API.

2.- Nivell d'emmagatzematge:

Pot haver-hi tres tipus diferents de servidors d'emmagatzematge:

- Servidors d'objectes: emmagatzemen, recuperen i eliminen objectes emmagatzemats en dispositius locals. Aquests objectes són les dades reals emmagatzemades al clúster Swift.
- Servidors de contenidors: gestionen les llistes d'objectes que contenen els contenidors
- Servidors de comptes: s'ocupen de les fitxes dels contenidors associats als comptes.

Igual que amb el nivell d'accés, l'escalat és bastant fàcil. Quan es necessita més capacitat es poden afegir servidors d'emmagatzematge addicionals. La replicació de dades en aquests nous servidors es gestiona automàticament i el sistema s'autobalanceja.

L'accés a un clúster Swift es basa en comptes. Un compte és anàleg a un usuari humà que necessita emmagatzemar dades al clúster.

Abans que un usuari pugui emmagatzemar qualsevol objecte, és a dir, dades, al clúster Swift, s'ha de crear un contenidor per conservar les dades.

Els contenidors són anàlegs als directoris en un sistema de fitxers tradicional. Un compte pot crear diversos contenidors per mantenir els seus objectes.

Les dades reals emmagatzemades al clúster Swift es denominen objectes.

Cada objecte conté les dades completes, les metadades i l'identificador únic d'aquest objecte i és accessible per la seva URL exclusiva.

Cada contenidor pot contenir diversos objectes.

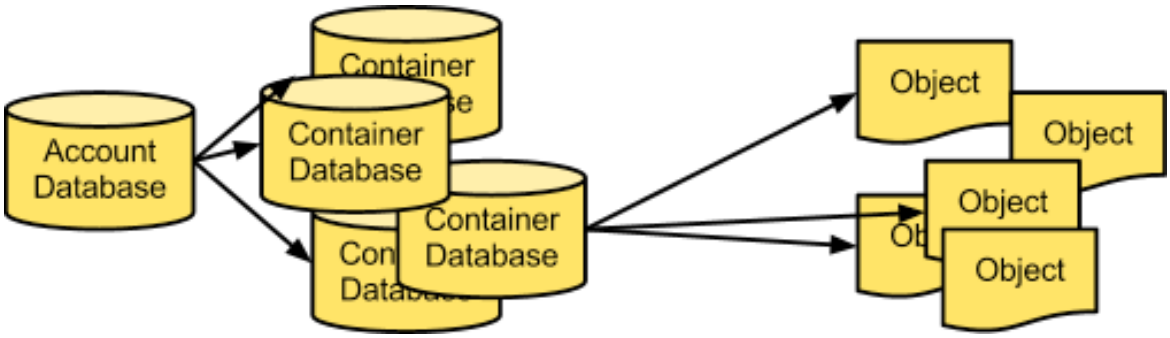

Figura 6 – 27 Emmagatzematge d'objectes en contenidors

Ring - Anells:

- Mapeja els noms lògics d'entitats a les seves ubicacions físiques al disc
- Manté mapes basats en zones, dispositius, particions i rèpliques
- Ajuntar rings per a comptes, contenidors i objectes
- Usat pels servidors proxy i altres processos de fons com la replicació

En un clúster Swift, tots els mapejos entre entitats (comptes, contenidors, objectes) i la seva ubicació física real al disc s'emmagatzemen i accedeixen en els anomenats Rings o Anells.

Cada tipus d'entitat té el seu propi anell corresponent que significa que hi ha un anell de compte, un anell de contenidor i un anell d'objecte. Aquests anells són accessibles i mantinguts pels servidors Swift Proxy.

Els processos de replicació en fons del clúster Swift també utilitzen aquests anells per fer el seguiment de quantes còpies de les dades existeixen i on existeixen aquestes còpies.

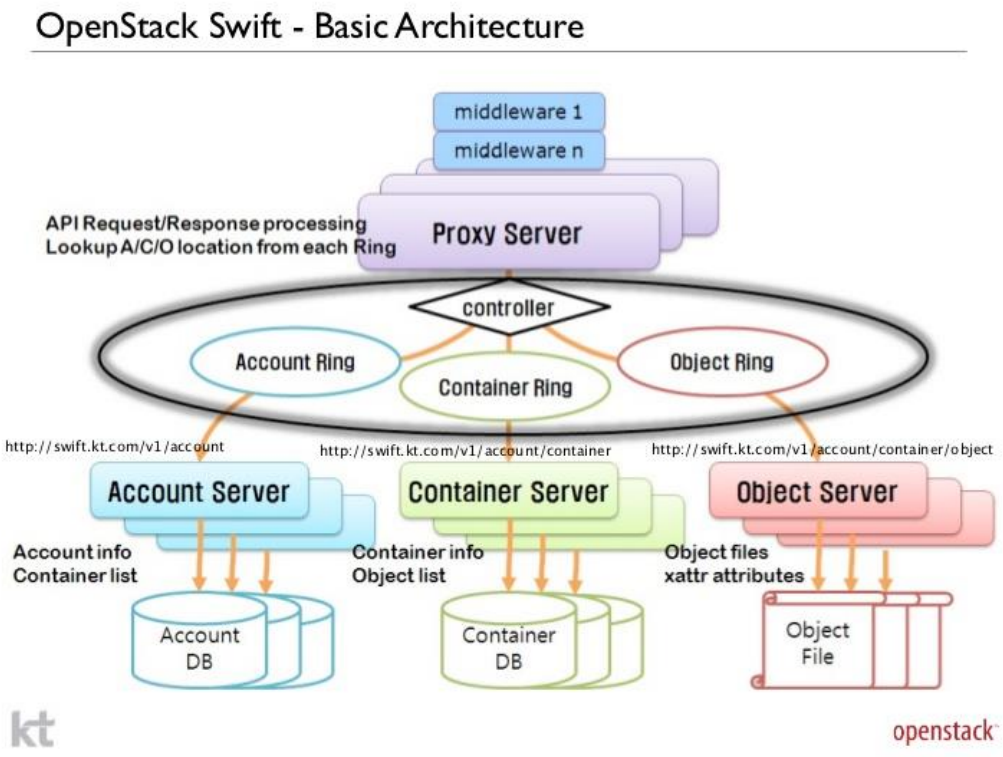

Figura 6 - 28 Arquitectura de Swift

Zone - Zona:

- Aïlla els límits de la fallada
- Conté servidors d'objectes, contenidors i comptes
- Les rèpliques de dades resideixen en diferents zones
- La fallada en qualsevol zona no afecta cap altra zona

Un clúster Swift es divideix en diferents zones que s'utilitzen per aïllar les fallades. Les rèpliques de dades resideixen en diferents zones. La idea és que la fallada en una zona no afectarà cap altra zona.

Tenir més de dues zones permetrà més de dues rèpliques de dades i fins i tot assegurarà que el fallada en més d'una zona no afectarà la validesa de les dades en tot el clúster (sempre que la quantitat de rèpliques sigui superior a dos).

Cada zona contindrà servidors de comptes, contenidors i objectes

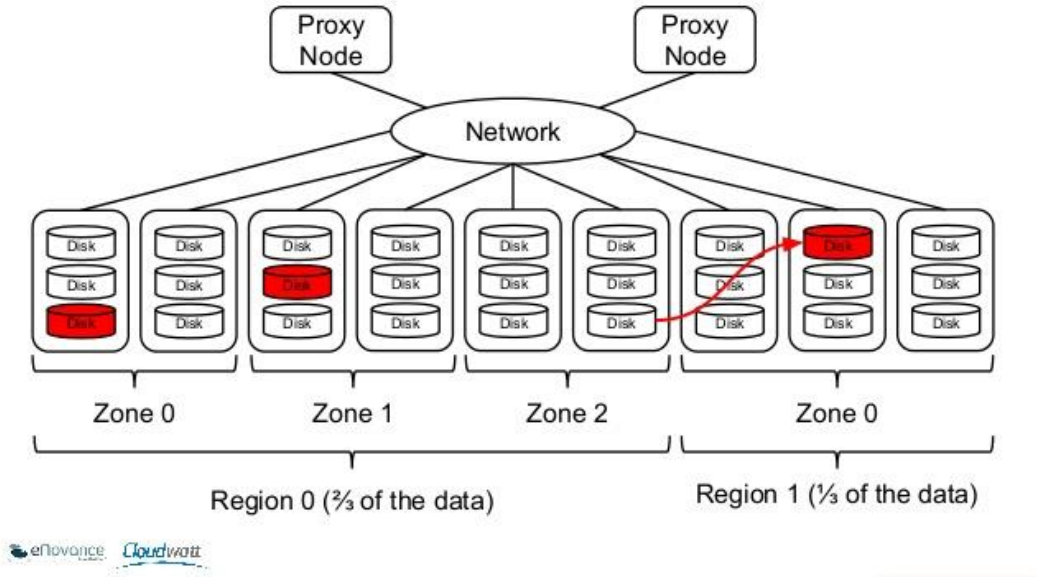

Figura 6 - 29 Redundància de les dades

Device - Dispositiu

- Disc on es guarden les dades

Partition - Partició:

- Recull de dades emmagatzemades (compte DB / contenidor DB / objectes)

Replica - Rèplica:

- còpia d'una partició en una zona

Un dispositiu en un clúster Swift fa referència a un disc real on es guarden les dades. Un servidor d'emmagatzematge generalment tindrà més d'un dispositiu, ja que no és usual tenir un sol disc en un node.

Una partició és una col·lecció de dades emmagatzemades, incloses les dades del compte, les dades del contenidor i els objectes. Les rèpliques són còpies de particions. Les rèpliques es troben en dispositius individuals dels servidors d'emmagatzematge del nivell d'emmagatzematge.

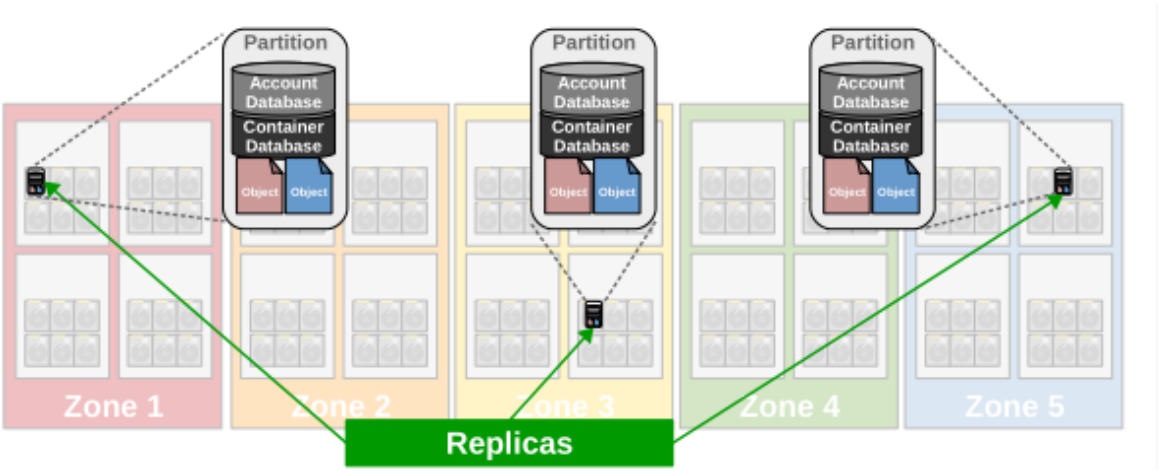

Figura 6 - 30 Rèpliques de les dades

Ceph

Tot i que no és un projecte oficial d'OpenStack, Ceph és molt utilitzat amb OpenStack, ja que Ceph[31] pot proporcionar tant l'emmagatzematge d'objectes Swift com l'emmagatzematge de bloc de Cinder i Glance.

Les seves característiques principals són:

- Emmagatzematge d'objectes
- Emmagatzematge de bloc, amb capacitat de snapshots i clons
- Sistema de fitxers compatible amb POSIX
- Altament escalable, teòricament a exabytes i "sense límits"

[31] https://es.wikipedia.org/wiki/Ceph_File_System

- Funciona amb maquinari heterogeni i de "baix cost"
- Fiable i tolerant a fallades

RADOS (Reliable Autonomous Distributed Object Store) és el motor que hi ha darrere de Ceph.

Per proporcionar accés als contenidors d'objectes implementa les següents eines: proporciona una biblioteca (librados) i un conjunt de serveis (RADOS Gateway, RBD i CephFS).

Librados: és una llibreria que permet des de diferents llenguatges (c++, python, ruby, php, java, ..) poder accedir al RADOS, pel que el fa molt atractiu als desenvolupadors.

RADOS Gateway: és un gateway REST orientat a l'accés via Amazon S3 i Swift.

RBD: és un sistema d'emmagatzematge de bloc que està integrat al kernel de Linux

Ceph-fs: és un sistema de fitxers distribuït compatible amb POSIX que suporta FUSE.

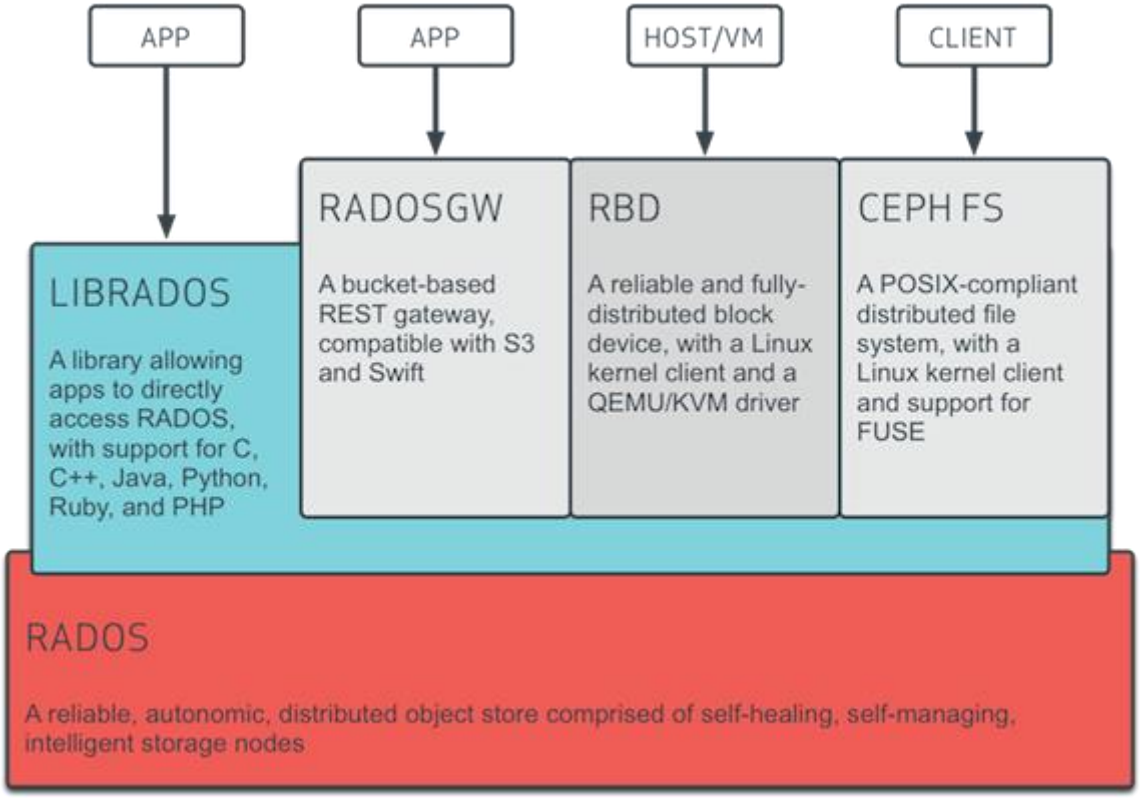

Figura 6 - 31 Arquitectura de Ceph

Sense entrar en detalls, l'arquitectura de ceph compta amb l'algoritme CRUSH que és utilitzat per distribuir i accedir a les dades que s'estan emmagatzemant en els nodes del clúster. És ràpid i determinista i proporciona una distribució estadísticament uniforme de les dades en tots els nodes del clúster

Horizon

El projecte Horizon, també conegut com OpenStack Dashboard, proporciona una interfície d'usuari basada en web al núvol d'OpenStack per a usuaris i administradors.

Horizon està dissenyat per ser fàcilment personalitzable i mitjançant fulles d'estil poder-lo adaptar a la imatge del proveïdor que està donant-lo com a servei o la organització on està funcionant.

Un altre avantatge de Horizon és que es pot ampliar amb complements d'altres fabricants tal i com s'apuntava en l'apartat de SDN de xarxes.

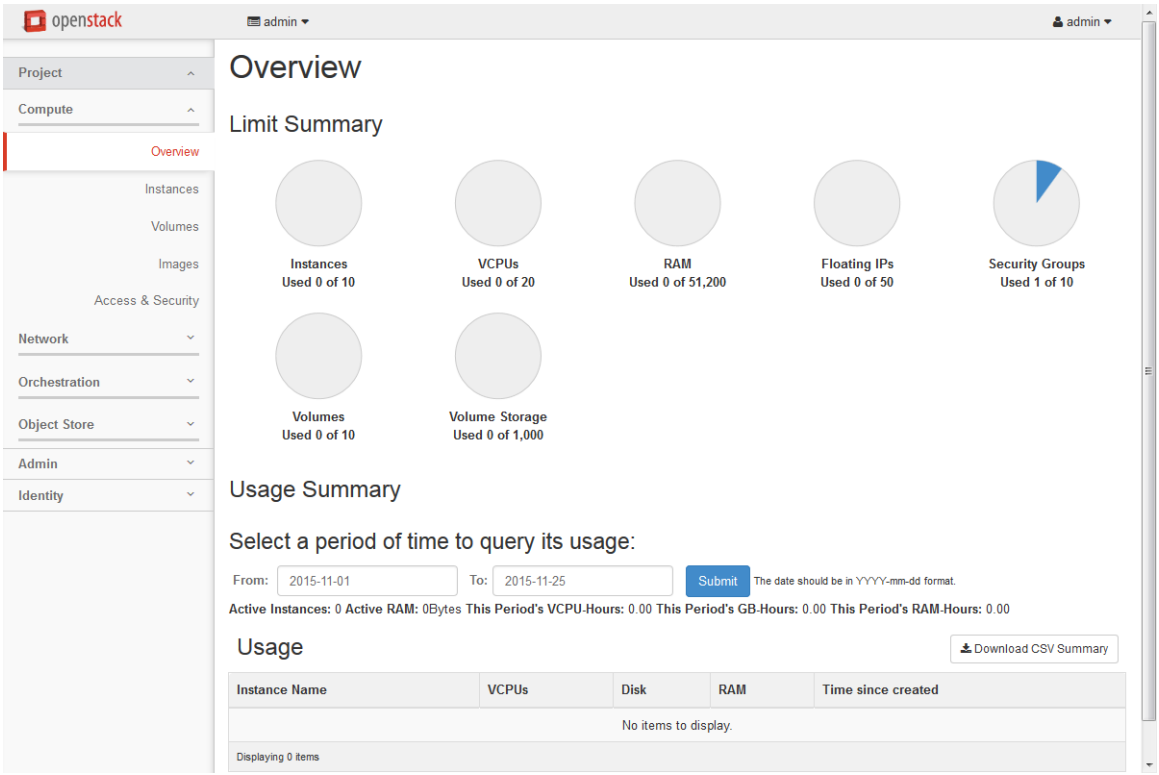

Figura 6 - 32 Dashboard d'Horizon

Ceilometer

El projecte Ceilometer, també conegut com el servei de telemetria, proporciona mètriques dels recursos en un núvol OpenStack.

Proporciona mètriques d'ús de tots els components d'OpenStack i com no podia ser d'una altra manera, té una API oberta que permet que qualsevol nou component que s'introdueixi a OpenStack pugui ser mesurat i integrat dins del sistema.

Utilitza el sistema de cues del propi OpenStack que utilitzen tots els components per comunicar-se. Com a backend té una base de dades mongo-db on emmagatzema totes les dades que va rebent per tal de poder graficar-les segons l'administrador o els usuaris que tinguin permisos els calgui.

Disposa d'un generador d'informes personalitzats de tots els components d'OpenStack.

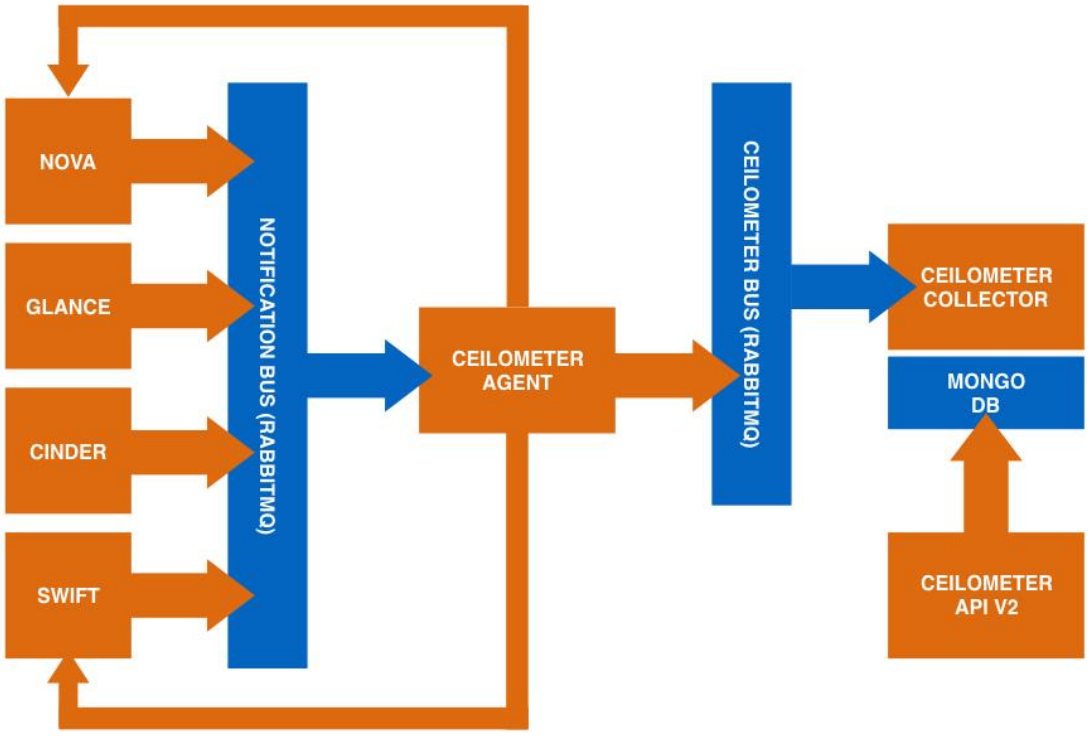

Figura 6 - 33 Arquitectura de Ceilometer

Instal·lació i administració d'OpenStack

En aquest capítol es realitzarà el desplegament d'OpenStack en forma de laboratori. A la primera part del capítol veurem com descarregar i instal·lar el node d'administració d'Openstack. A la segona part del capítol veurem com desplegar els nodes i els serveis del nostre cloud.

Disposem d'un servidor amb processador Intel® Xeon® E3-1220 amb 32 GB de RAM i un disc dur SSD de 256 GB.

Per a un desplegament ràpid en un entorn d'OpenStack, triarem Ubuntu, de fet segons els seus estudis el 50% de les instal·lacions corren sobre els seus sistemes operatius.
Si volem un ús més professional anirem cap a Red Hat o SUSE.

Tot i que ambdós tenen funcionalitats molt semblants, caldrà escollir per coneixement de les distribucions o de Puppet o Chef.

Amb aquesta configuració desplegarem OpenStack en una sola màquina i utilitzarem SUSE, que juntament amb Red Hat son les que estan més enfocades a entorns empresarials.

Com a hipervisor utilitzarem KVM.

Descàrrega del software

Accedim a la url https://www.suse.com/es-es/ i ens autentiquem:

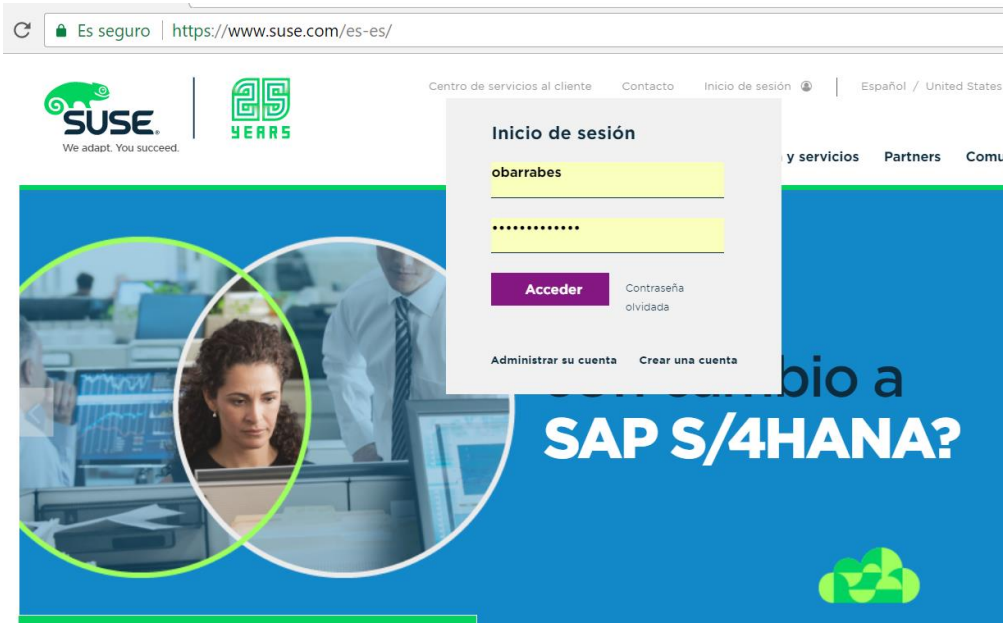

Figura 7 - 1 Plana d'inici de sessió del portal de SUSE

Anem a Descarregues gratuïtes i descarreguem el paquet SUSE OpenStack Cloud, tal i com es pot veure en la imatge següent:

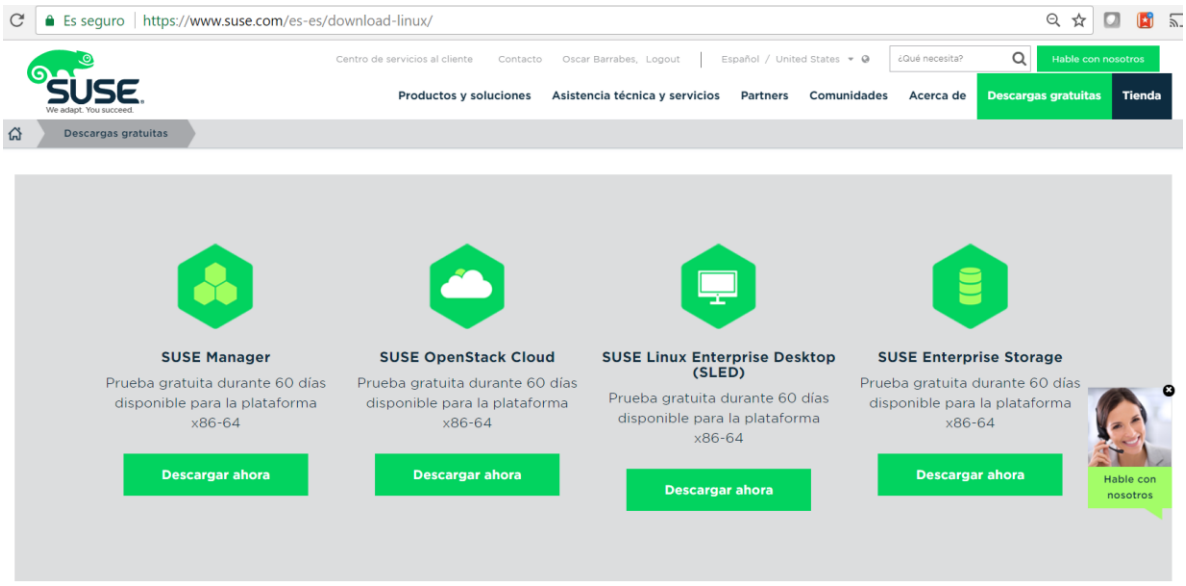

Figura 7 - 2 Descàrrega de SUSE OpenStack

Finalment ens apareix la pantalla següent, ens cal únicament la primera ISO, ja que les altres són els fonts i el debug:

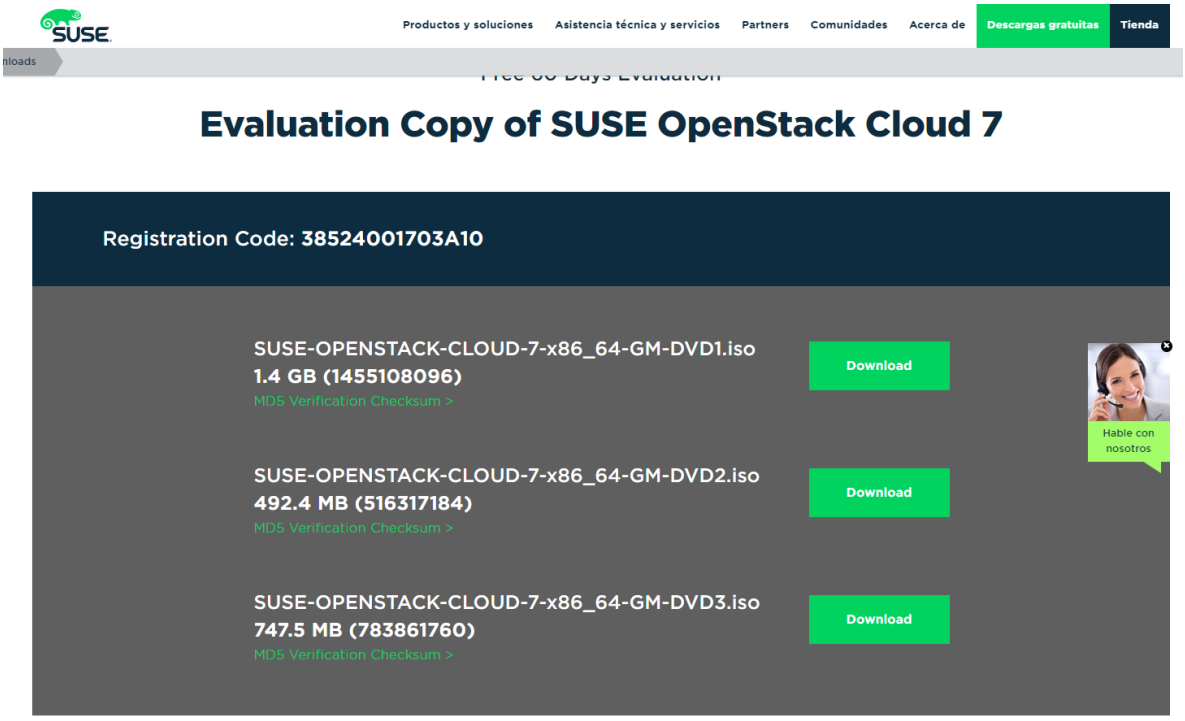

Figura 7 - 3 ISOs disponibles d'OpenStack SUSE

En aquesta mateix pantalla hi trobarem les guies d'instal·lació i la resta de documentació.

Instal·lació del software

Node d'administració

El node d'administració d'Openstack, des del que instal·larem la resta de la clúster, és el primer que cal instal·lar. Obtindrem una màquina virtual amb SUSE Enterprise Linux que instal·larem el crowbar que ens automatitzarà amb Chef el desplegament i configuració de la resta de node nostre núvol. Es pot veure de manera detallada la instal·lació a l'Annex 1.

Des d'aquest node sempre que calgui, podrem crear / eliminar nodes i configurar-los amb el rol que ens convingui.

El node d'administració tindrà un servidor PXE que servirà per enviar la imatge de SUSE a cada node i mitjançant els scripts de Chef configurar cadascun d'ells amb el rol que li assignem.

Desplegament dels nodes i serveis d'OpenStack

Nodes de control: en crearem tres, d'aquesta manera podrem provar la funcionalitat d'alta disponibilitat del nostre entorn, fent que els serveis corrin distribuïts i amb capacitat de fallada, cal recordar que és una de les avantatges de SUSE, ja que incorpora aquesta funcionalitat sense haver d'integrar res i es configura directament des del crowbar.

Nodes de còmput: en crearem dos, suficients per fer proves de desplegament de màquines i serveis.

Nodes de emmagatzematge: en crearem quatre i en desplegarem un ceph, d'aquesta manera tindrem tots els tipus d'emmagatzematge que ens calen en aquests nodes. Els distribuirem de la següent manera: en tindrem un que farà de gateway i els tres restants faran de osd, com que funcionen amb clúster, sempre ens en calen com a mínim tres.

Un cop hem desplegat totes les màquines i s'han inventariat dins del crowbar, podem instal·lar els nodes accedint via web a la consola de crowbar:

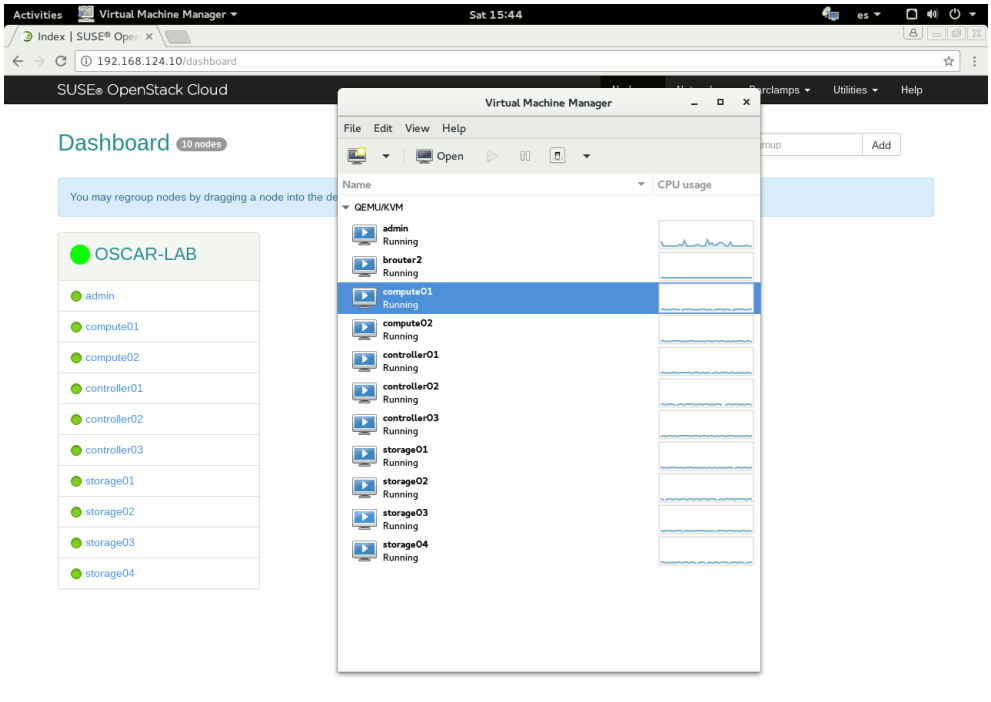

Figura 7 - 4 Dashboard de crowbar

En la següent taula podem veure un resum dels diferents serveis amb els tipus de nodes on s'executen:

Servei	Node de Control	Node d'Emmagatzematge	Node de Còmput
Servei de base de dades	SÍ		
Servei de missatgeria	SÍ		
Keystone	SÍ		
Ceph	SÍ	SÍ	
Glance	SÍ	SÍ	
Glance	SÍ	SÍ	
Neutron	SÍ	SÍ	SÍ
Nova	SÍ		SÍ
Horizon	SÍ		
Ceilometer	SÍ		

L'ordre que seguirem per fer la instal·lació, és el següent, a l'annex 2 es pot veure detalladament com es fa la instal·lació i configuració de cadascun d'ells:

1. Servei d'alta disponibilitat: no formen part del projecte d'OpenStack, mitjançant pacemaker[32], corosync[33] i hawk[34] que els integra directament SUSE en el seu desplegament d'Openstack ens permet tenir en alta disponibilitat el nostre cloud.

2. Servei de base de dades: és la base de dades on els components d'OpenStack emmagatzemaran tota la informació.

3. Servei de missatgeria: està basat en RabbitMQ, un software desenvolupat per la empresa Pivotal[35], que mitjançant el protocol AMQP (Advanced Message Queueing Protocol) y un servidor programat en Erlang, permet gestionar missatges amb cues de manera distribuïda. Tots els components d'Openstack per comunicar-se entre ells, utilitzen el sistema de cues.

4. Servei d'identitats / Keystone: un cop ja hem instal·lat on emmagatzemarem la informació dels components i el sistema de comunicació d'aquests, ens cal començar a implementar la seguretat i aquest servei és l'encarregat d'aquesta tasca.

5. Servei de Ceph: tal com ja he explicat en capítols anteriors, Ceph ens donar tota la funcionalitat per l'emmagatzematge d'imatges, de bloc i d'objectes.

6. Servei de Glance: basat sobre el Ceph que hem instal·lat al pas anterior, instal·larem el servei d'imatges dels sistemes operatius disponibles per desplegar al cloud.

7. Servei de Cinder: basat sobre el Ceph que hem instal·lat anteriorment, instal·larem servei d'emmagatzematge persistent del nostre cloud en format bloc

[32] https://clusterlabs.org/pacemaker.html

[33] https://clusterlabs.org/corosync.html

[34] https://hawk-ui.github.io/

[35] https://es.wikipedia.org/wiki/RabbitMQ

8. Servei de Neutron: el servei de xarxa definida per software, aquest servei al ser transversal per tota la infraestructura és dels pocs que pot executar-se en tots els nodes del nostre cloud.

9. Servei de Nova: el servei de còmput que en el nostre cas està sobre KVM i que és el que ens permet executar les màquines virtuals.

10. Servei d'Horizon: és el frontend web, que ens permetrà interactuar amb el nostre cloud

11. Servei Ceilometer: és el servei que ens permet extreure gràfiques d'ús del cloud.

Un cop hem instal·lat els diferents serveis, ja podrem accedir al nostre cloud:

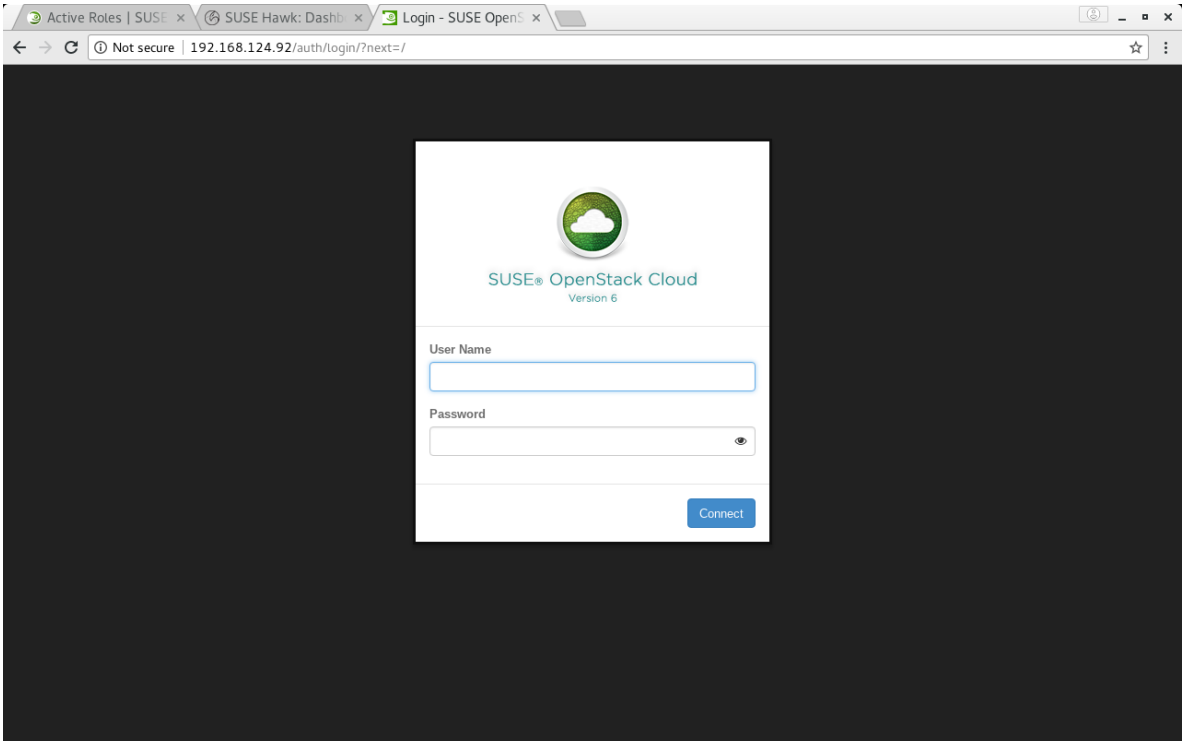

Figura 7 - 5 Login d'usuari de SUSE OpenStack

Us d'OpenStack

L'objectiu del capítol és conèixer com s'utilitza i es posen en servei càrregues de treball a l'entorn d'OpenStack utilitzant els diferents components que disposa. Totes les accions sobre OpenStack, es poden fer tant des de la interfície web com des de línies de comanda, es per això que el primer que veurem és el servei d'identitats, on veurem com accedir mitjançant els dos mètodes.

El servei d'identitats

Per accedir al nostre cloud via web, ho farem mitjançant la ip virtual que ens ha assignat el sistema en el capítol anterior, ens identificarem amb l'usuari / contrasenya.

Per fer-ho mitjançant línies de comandes, haurem d'utilitzar un script que podem descarregar-lo des de la mateixa web d'administració:

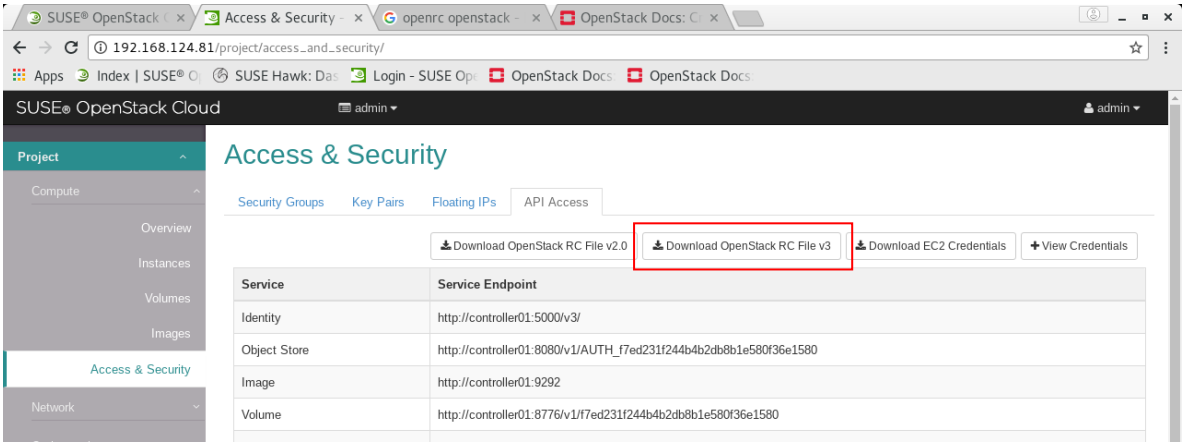

Figura 8 - 1 Descàrrega script variables entorn de línia de comandes

Obtindrem un fitxer que executat des de el node d'administració podrem interactuar amb el sistema:

```
tux@linux:~/Oscar> more admin-openrc.sh
#!/usr/bin/env bash

# To use an OpenStack cloud you need to authenticate against the Identity
# service named keystone, which returns a **Token** and **Service Catalog**.
```

```
# The catalog contains the endpoints for all services the user/tenant has
# access to - such as Compute, Image Service, Identity, Object Storage, Block
# Storage, and Networking (code-named nova, glance, keystone, swift,
# cinder, and neutron).
#
# *NOTE*: Using the 3 *Identity API* does not necessarily mean any other
# OpenStack API is version 3. For example, your cloud provider may implement
# Image API v1.1, Block Storage API v2, and Compute API v2.0. OS_AUTH_URL is
# only for the Identity API served through keystone.
export OS_AUTH_URL=http://controller01:5000/v3/

# With the addition of Keystone we have standardized on the term **project**
# as the entity that owns the resources.
export OS_PROJECT_ID=f7ed231f244b4b2db8b1e580f36e1580
export OS_PROJECT_NAME="admin"
export OS_USER_DOMAIN_NAME="Default"
if [ -z "$OS_USER_DOMAIN_NAME" ]; then unset OS_USER_DOMAIN_NAME; fi

# unset v2.0 items in case set
unset OS_TENANT_ID
unset OS_TENANT_NAME

# In addition to the owning entity (tenant), OpenStack stores the entity
# performing the action as the **user**.
export OS_USERNAME="admin"

# With Keystone you pass the keystone password.
echo "Please enter your OpenStack Password: "
read -sr OS_PASSWORD_INPUT
export OS_PASSWORD=$OS_PASSWORD_INPUT

# If your configuration has multiple regions, we set that information here.
# OS_REGION_NAME is optional and only valid in certain environments.
export OS_REGION_NAME="RegionOne"
# Don't leave a blank variable, unset it if it was empty
```

```
if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
tux@linux:~/Oscar>  OpenStack project list
+----------------------------------+------------+
| ID                               | Name       |
+----------------------------------+------------+
| f7ed231f244b4b2db8b1e580f36e1580 | admin      |
| bb64c58238e448a480d8087286c87e37 | service    |
| 64a024b12a164e6a9807912319a0a3a5 | OpenStack  |
| 1cf0b903fbf5472eb8ecfdc0ec104fea | dispersion |
+----------------------------------+------------+
tux@linux: ~/Oscar>
```

Un cop ja estem validats al sistema i per provar el sistema d'indentitats, podem procedir a crear un rol:

```
tux@linux: [admin@Default/admin (v3)]~/Oscar> openstack role create network_admins
+-------+----------------------------------+
| Field | Value                            |
+-------+----------------------------------+
| id    | 7cc7e083168a4b2297a00c005b345636 |
| name  | network_admins                   |
+-------+----------------------------------+
tux@linux: [admin@Default/admin (v3)]~/Oscar>
tux@linux: [admin@Default/admin (v3)]~/Oscar> openstack role list
+----------------------------------+------------------+
| ID                               | Name             |
+----------------------------------+------------------+
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_         |
| 8dd89755d88b4fd297eb2f2ec279ae5b | admin            |
| 62cb720943224c7bb3ea2faa149db68d | Member           |
| f9e88756b09b440293e29fd5b39db36d | ResellerAdmin    |
| 4d355f59ae9c4a75a1d0a84ff33f62ea | heat_stack_user  |
| bfc4a08b5db540d385d6b2cf30f36cdf | heat_stack_owner |
| 7cc7e083168a4b2297a00c005b345636 | network_admins   |
+----------------------------------+------------------+
tux@linux: [admin@Default/admin (v3)]~/Oscar>
```

Usuaris i grups

El primer que farem és crear un projecte (o tenant) d'on penjaran les càrregues de treball del nostre PFC:

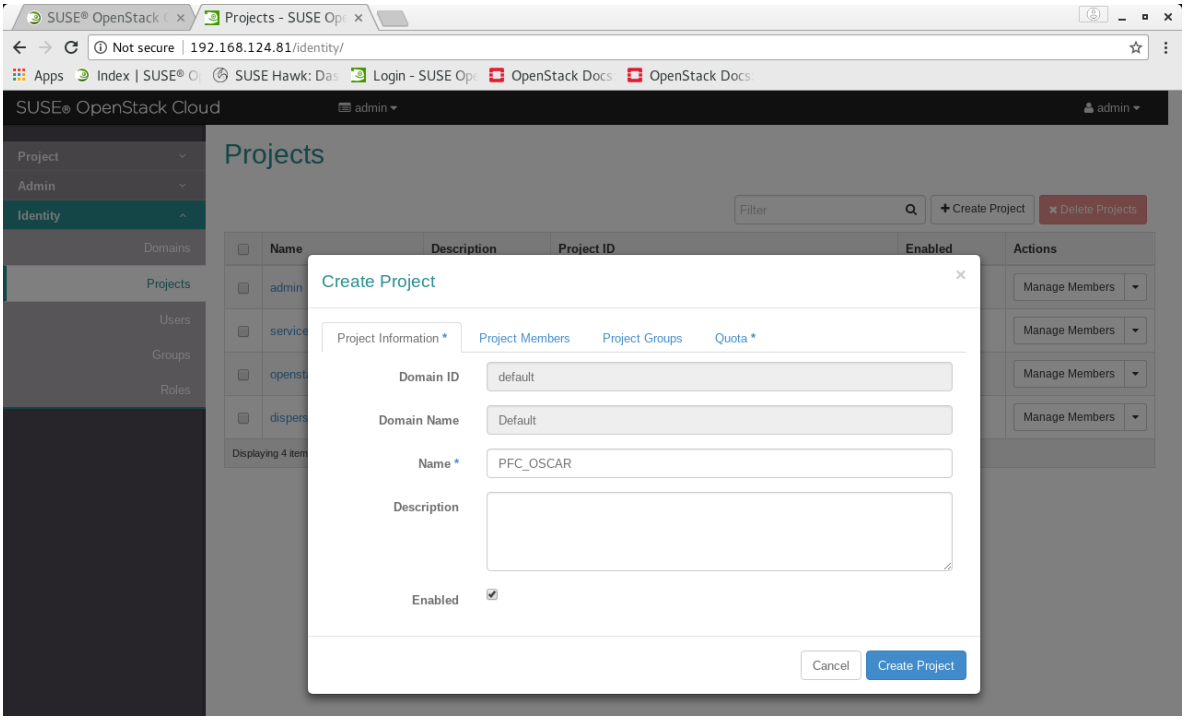

Figura 8 - 2 Creació de projecte

Ara ja podem crear un usuari i assignar-lo com a membre al projecte:

```
tux@linux: [admin@Default/admin (v3)]~/Oscar> openstack user create \
>    --project PFC_OSCAR \
>    --password Oscar123 \
>    --email obarrabes@uoc.eu \
>    Oscar
+-------------------+----------------------------------+
| Field             | Value                            |
+-------------------+----------------------------------+
| default_project_id | db653b636b794ed581a6c95efaf74938 |
| domain_id         | default                          |
| email             | obarrabes@uoc.eu                 |
```

```
| enabled           | True                             |
| id                | 36c83fb1c2c644ce9e7fe3300f2fb2a8 |
| name              | Oscar                            |
+-------------------+----------------------------------+
tux@linux: [admin@Default/admin (v3)]~/Oscar> openstack role add \
>    --project PFC_OSCAR \
>    --user Oscar \
>    Member
tux@linux: [admin@Default/admin (v3)]~/Oscar>
```

Ara si accedim a la consola web amb el nou usuari podrem veure que està assignat al projecte:

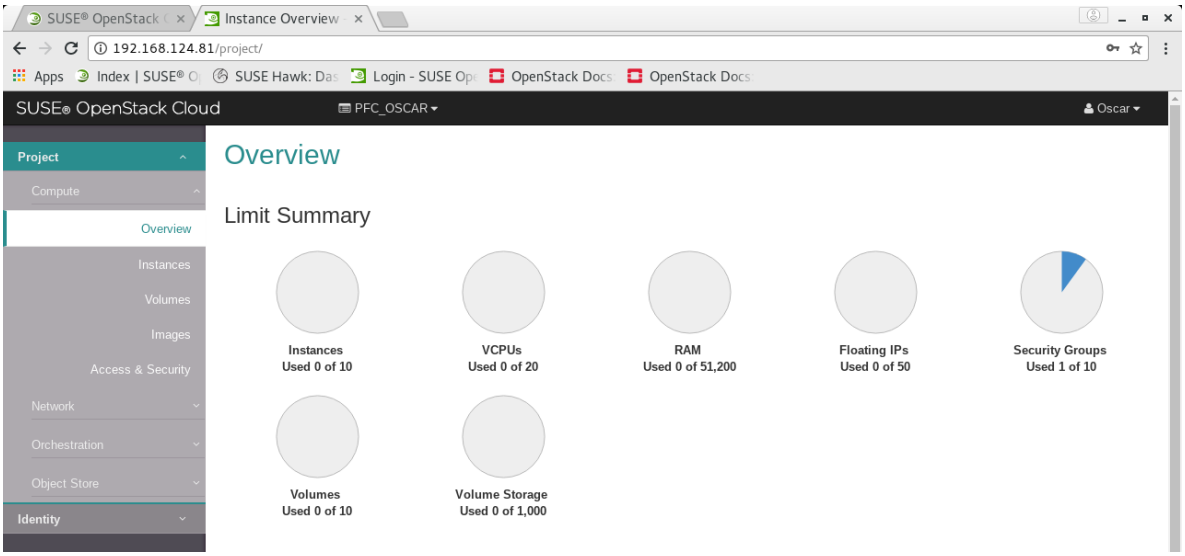

Figura 8 - 3 Vista de recursos consumits

Gestió d'imatges

La següent tasca que farem és carregar una imatge d'un sistema operatiu per poder utilitzar-la més endavant per generar càrregues de treball. Ho farem per línia de comanda:

```
tux@linux: [admin@Default/admin (v3)]/home/images> ll
total 354348
-rw-r--r-- 1 tux root 362847744 Dec  3 15:43 SLES12SP1-cloudimage.qcow2
```

```
tux@linux: [admin@Default/admin (v3)]/home/images>
tux@linux: [admin@Default/admin (v3)]/home/images> openstack image create --public --container-format bare --disk-format qcow2 --min-disk 2 --min-ram 512 --file SLES12SP1-cloudimage.qcow2 SLES12SP1-x86_64
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | fcdeb8b10730ac96bccc9a121ee030f4                     |
| container_format | bare                                                 |
| created_at       | 2017-12-12T17:06:16Z                                 |
| disk_format      | qcow2                                                |
| file             | /v2/images/2ac75599-0285-4806-b685-8f9781ee25e1/file |
| id               | 2ac75599-0285-4806-b685-8f9781ee25e1                 |
| min_disk         | 2                                                    |
| min_ram          | 512                                                  |
| name             | SLES12SP1-x86_64                                     |
| owner            | f7ed231f244b4b2db8b1e580f36e1580                     |
| protected        | False                                                |
| schema           | /v2/schemas/image                                    |
| size             | 362847744                                            |
| status           | active                                               |
| updated_at       | 2017-12-12T17:06:18Z                                 |
| virtual_size     | None                                                 |
| visibility       | public                                               |
+------------------+------------------------------------------------------+
tux@linux: [admin@Default/admin (v3)]/home/images>
```

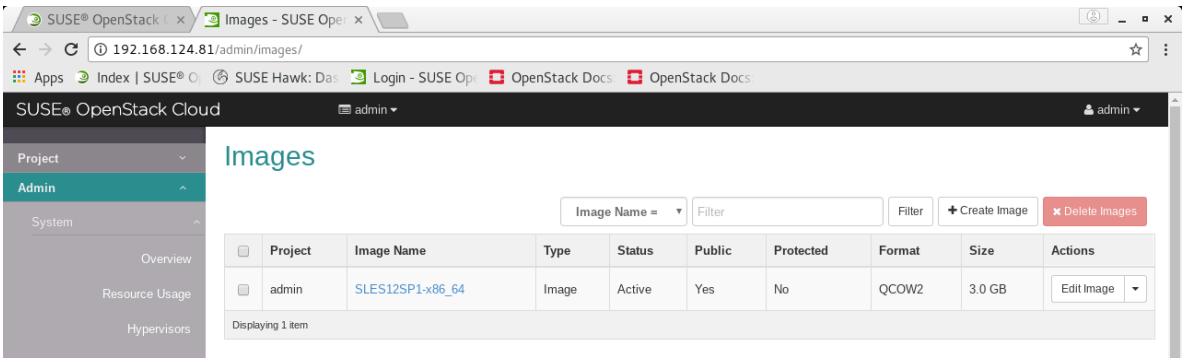

Figura 8 - 4 Imatges disponibles

Podem carregar diferents tipus d'imatges, en el nostre cas ho hem fet amb format qcow2 que ens permet utilitzar-la directament amb KVM[36].

Les xarxes definides per programari

En aquest punt crearem els elements de xarxa que ens calen pel nostre projecte.

Crearem una xarxa privada, on es connectaran les nostres màquines vituals:

```
tux@linux: [admin@Default/admin (v3)]~/Oscar> neutron net-create PFC_OSCAR_network-01
Created a new network:
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| admin_state_up  | True                                 |
| id              | 1179dc6d-2d15-4b6b-a9f6-96f6fa5add82 |
| mtu             | 1358                                 |
| name            | PFC_OSCAR_network-01                 |
| router:external | False                                |
| shared          | False                                |
| status          | ACTIVE                               |
| subnets         |                                      |
| tenant_id       | db653b636b794ed581a6c95efaf74938     |
+-----------------+--------------------------------------+
tux@linux: [admin@Default/admin (v3)]~/Oscar> neutron subnet-create \
> --name PFC_OSCAR_subnet-01 \
> --allocation-pool start=10.0.1.100,end=10.0.1.150 \
> --gateway=10.0.1.1 \
> --enable_dhcp \
> --dns-nameserver 8.8.8.8 \
> PFC_OSCAR_network-01 10.0.1.0/24
```

[36] https://docs.OpenStack.org/image-guide/obtain-images.html

Created a new subnet:

```
+-------------------+-------------------------------------------+
| Field             | Value                                     |
+-------------------+-------------------------------------------+
| allocation_pools  | {"start": "10.0.1.100", "end": "10.0.1.150"} |
| cidr              | 10.0.1.0/24                               |
| dns_nameservers   | 8.8.8.8                                   |
| enable_dhcp       | True                                      |
| gateway_ip        | 10.0.1.1                                  |
| host_routes       |                                           |
| id                | bf36e224-da29-4d4a-a4fe-49bb3b753386      |
| ip_version        | 4                                         |
| ipv6_address_mode |                                           |
| ipv6_ra_mode      |                                           |
| name              | PFC_OSCAR_subnet-01                       |
| network_id        | 1179dc6d-2d15-4b6b-a9f6-96f6fa5add82      |
| subnetpool_id     |                                           |
| tenant_id         | db653b636b794ed581a6c95efaf74938          |
+-------------------+-------------------------------------------+
tux@linux: [admin@Default/admin (v3)]~/Oscar>
```

Un cop tenim la xarxa, crearem el router:

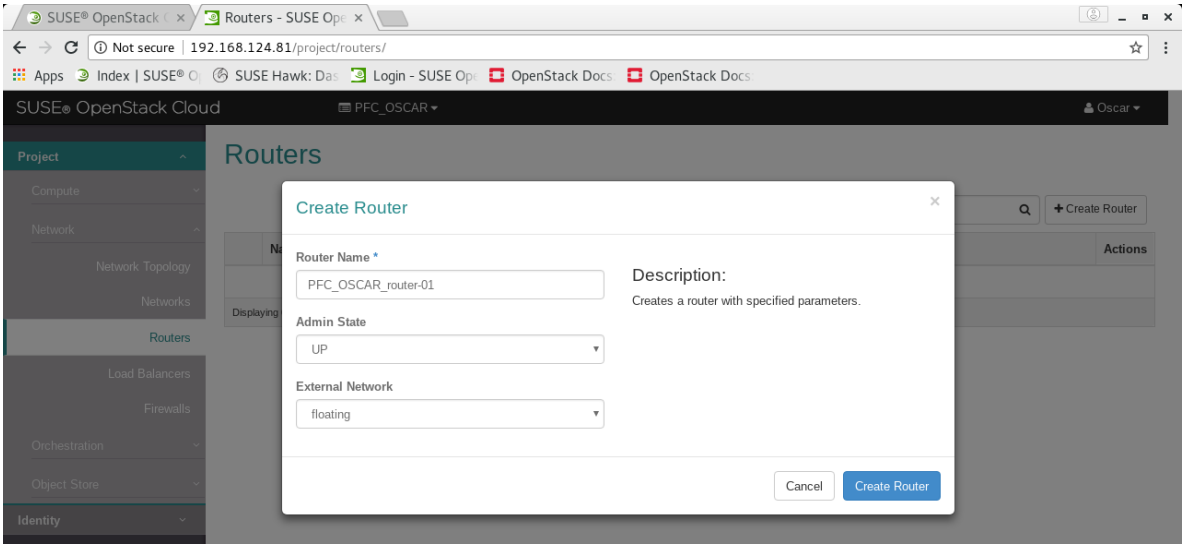

Figura 8 - 5 Creació d'un router

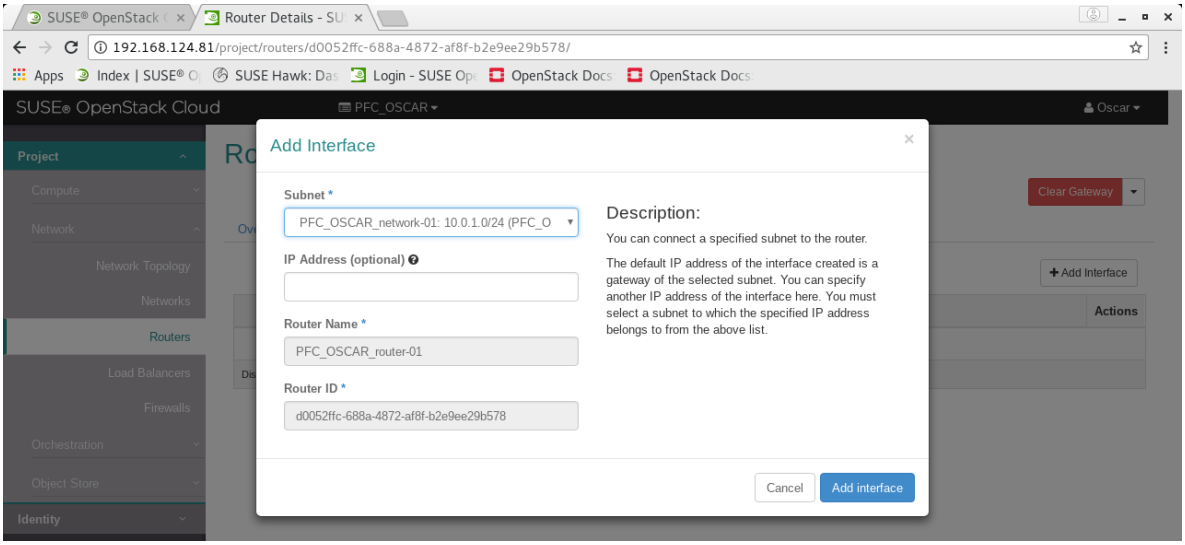

Figura 8 - 6 Creació d'un router II

Des de la interfície web, a l'opció de xarxes, hi ha una gràfica que ens mostra la topologia de la nostra xarxa, podem veure la xarxa interna connectada al router i la vola del món que indica l'exterior del cloud:

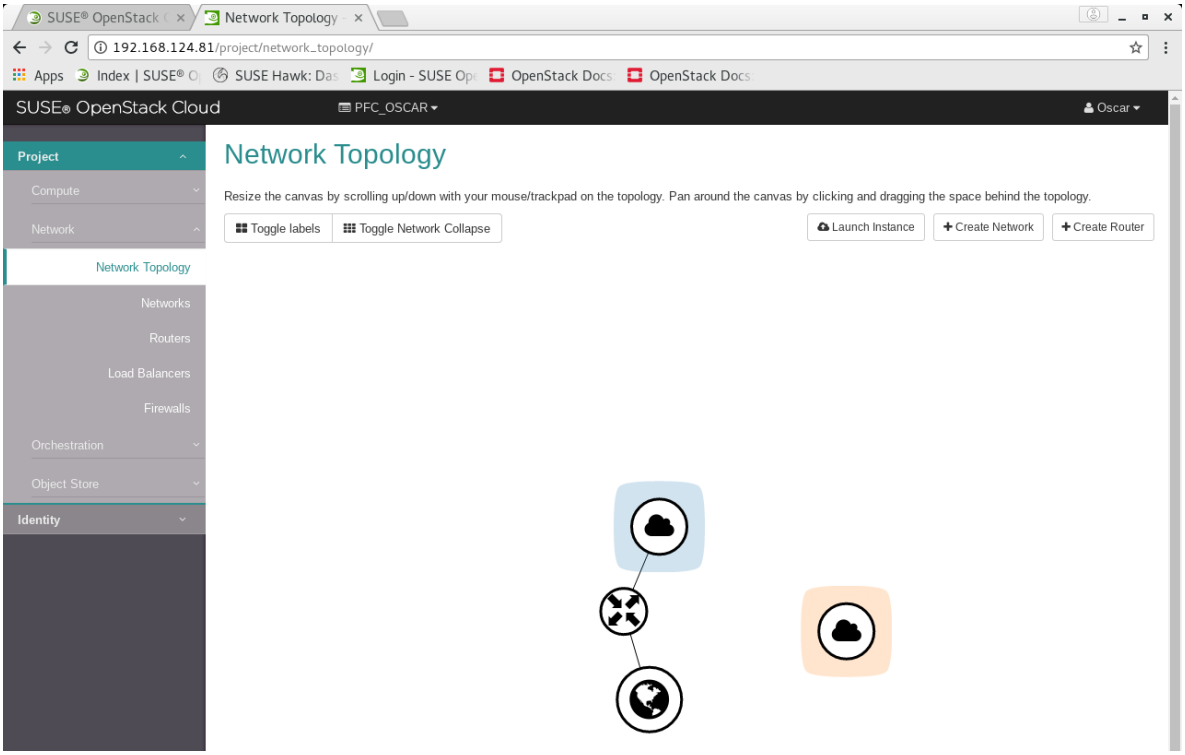

Figura 8 - 7 Topologia de xarxa

Ara crearem les regles per poder disposar d'ICMP i SSH sobre aquesta xarxa:

```
tux@linux: [admin@Default/admin (v3)]~/Oscar> neutron security-group-create PFC_OSCAR-ICMP_Rules \
> --description "Regles per ICMP"
Created a new security_group:
+---------------------+--------------------------------------------------------------------------------------------------------+
| Field               | Value                                                                                                  |
+---------------------+--------------------------------------------------------------------------------------------------------+
| description         | Regles per ICMP                                                                                        |
| id                  | a2e17b7b-caca-49ef-b14a-2ecdfed963f4                                                                   |
| name                | PFC_OSCAR-ICMP_Rules                                                                                   |
| security_group_rules | {"remote_group_id": null, "direction": "egress", "remote_ip_prefix": null, "protocol": null,          |
|                     | "tenant_id": "db653b636b794ed581a6c95efaf74938", "port_range_max": null, "security_group_id":          |
|                     | "a2e17b7b-caca-49ef-b14a-2ecdfed963f4", "port_range_min": null, "ethertype": "IPv4", "id":             |
|                     | "3ccef400-d301-4980-97a7-d396e586ad45"}                                                                |
|                     | {"remote_group_id": null, "direction": "egress", "remote_ip_prefix": null, "protocol": null,          |
|                     | "tenant_id": "db653b636b794ed581a6c95efaf74938", "port_range_max": null, "security_group_id":          |
|                     | "a2e17b7b-caca-49ef-b14a-2ecdfed963f4", "port_range_min": null, "ethertype": "IPv6", "id":             |
|                     | "d1a2fb7f-360a-4d5b-b4b2-4008a3ce6273"}                                                                |
| tenant_id           | db653b636b794ed581a6c95efaf74938                                                                       |
+---------------------+--------------------------------------------------------------------------------------------------------+
tux@linux: [admin@Default/admin (v3)]~/Oscar>
```

```
tux@linux: [admin@Default/admin (v3)]~/Oscar> neutron security-group-rule-create \
> --direction ingress \
> --protocol icmp \
> PFC_OSCAR-ICMP_Rules
Created a new security_group_rule:
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| id                | d261811d-812e-44f4-9015-90e983da4f53 |
| port_range_max    |                                      |
| port_range_min    |                                      |
| protocol          | icmp                                 |
| remote_group_id   |                                      |
| remote_ip_prefix  |                                      |
| security_group_id | a2e17b7b-caca-49ef-b14a-2ecdfed963f4 |
| tenant_id         | db653b636b794ed581a6c95efaf74938     |
+-------------------+--------------------------------------+
tux@linux: [admin@Default/admin (v3)]~/Oscar>
tux@linux: [admin@Default/admin (v3)]~/Oscar> neutron security-group-rule-create \
> --direction egress \
> --protocol icmp \
> PFC_OSCAR-ICMP_Rules
Created a new security_group_rule:
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| direction         | egress                               |
| ethertype         | IPv4                                 |
| id                | 212c87f4-4535-4af7-987e-dd0a61222e63 |
| port_range_max    |                                      |
| port_range_min    |                                      |
| protocol          | icmp                                 |
| remote_group_id   |                                      |
```

```
| remote_ip_prefix  |                                      |
| security_group_id | a2e17b7b-caca-49ef-b14a-2ecdfed963f4 |
| tenant_id         | db653b636b794ed581a6c95efaf74938     |
+-------------------+--------------------------------------+
tux@linux: [admin@Default/admin (v3)]~/Oscar> neutron security-group-create PFC_OSCAR-SSH_Rules \
> --description "Regles pel transit SSH"
Created a new security_group:
+----------------------+-------------------------------------------------------------------------------------------------------+
| Field                | Value                                                                                                 |
+----------------------+-------------------------------------------------------------------------------------------------------+
| description          | Regles pel transit SSH                                                                                |
| id                   | f64717ea-5df3-4ea4-bfc9-b6b67cae3a4b                                                                  |
| name                 | PFC_OSCAR-SSH_Rules                                                                                   |
| security_group_rules | {"remote_group_id": null, "direction": "egress", "remote_ip_prefix": null, "protocol": null,         |
|                      | "tenant_id": "db653b636b794ed581a6c95efaf74938", "port_range_max": null, "security_group_id":         |
|                      | "f64717ea-5df3-4ea4-bfc9-b6b67cae3a4b", "port_range_min": null, "ethertype": "IPv4", "id":            |
|                      | "a1db916c-9f49-4314-b913-7d1284e2f260"}                                                               |
|                      | {"remote_group_id": null, "direction": "egress", "remote_ip_prefix": null, "protocol": null,         |
|                      | "tenant_id": "db653b636b794ed581a6c95efaf74938", "port_range_max": null, "security_group_id":         |
|                      | "f64717ea-5df3-4ea4-bfc9-b6b67cae3a4b", "port_range_min": null, "ethertype": "IPv6", "id":            |
|                      | "35b0f446-fa9a-4056-9530-067a620c72f9"}                                                               |
| tenant_id            | db653b636b794ed581a6c95efaf74938                                                                      |
```

```
+---------------------+-------------------------------------------------------------------------------------
----+
tux@linux: [admin@Default/admin (v3)]~/Oscar>
tux@linux: [admin@Default/admin (v3)]~/Oscar> neutron security-group-rule-create \
> --direction ingress \
> --protocol tcp \
> --port_range_min 22 \
> --port_range_max 22 \
> PFC_OSCAR-SSH_Rules
Created a new security_group_rule:
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| id                | 3e80ca78-c70a-4aab-a785-7d64abc4b6cd |
| port_range_max    | 22                                   |
| port_range_min    | 22                                   |
| protocol          | tcp                                  |
| remote_group_id   |                                      |
| remote_ip_prefix  |                                      |
| security_group_id | f64717ea-5df3-4ea4-bfc9-b6b67cae3a4b |
| tenant_id         | db653b636b794ed581a6c95efaf74938     |
+-------------------+--------------------------------------+
tux@linux: [admin@Default/admin (v3)]~/Oscar>
```

També ho podem fer des de la interfície web, i ho farem per permetre trànsit http:

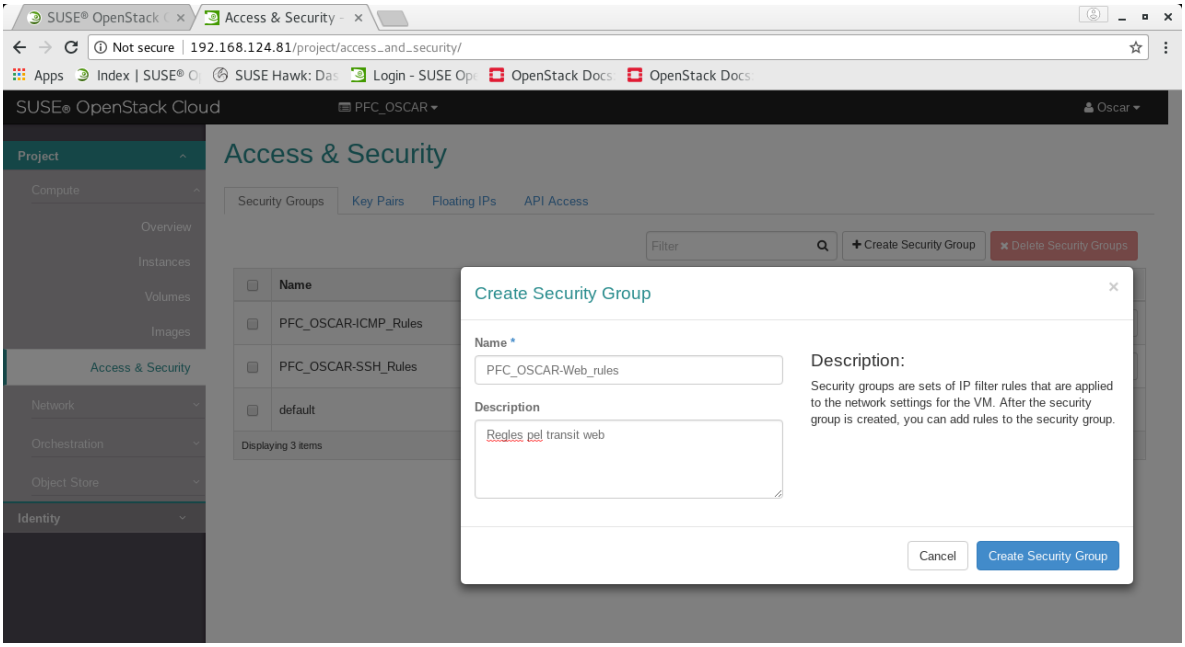

Figura 8 - 8 Creació d'un Security Group

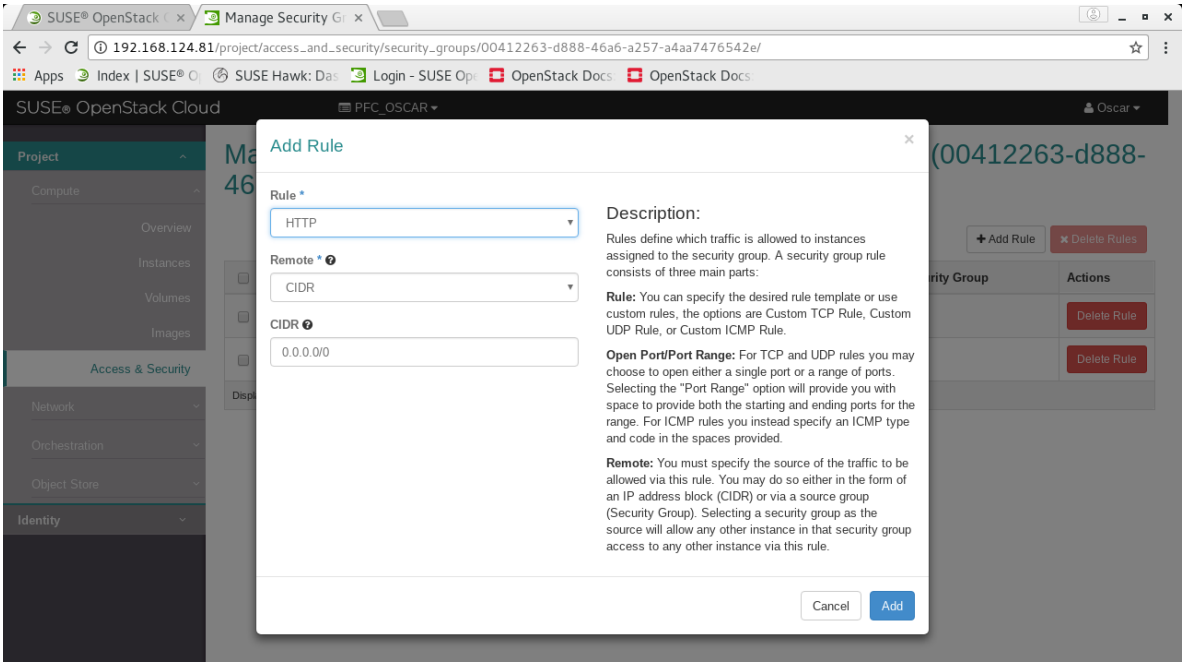

Figura 8 - 9 Afegir una regla a un Security Group

Afegim una regla pel trànsit HTTPS i un cop fet podem veure el detall del que hem fet:

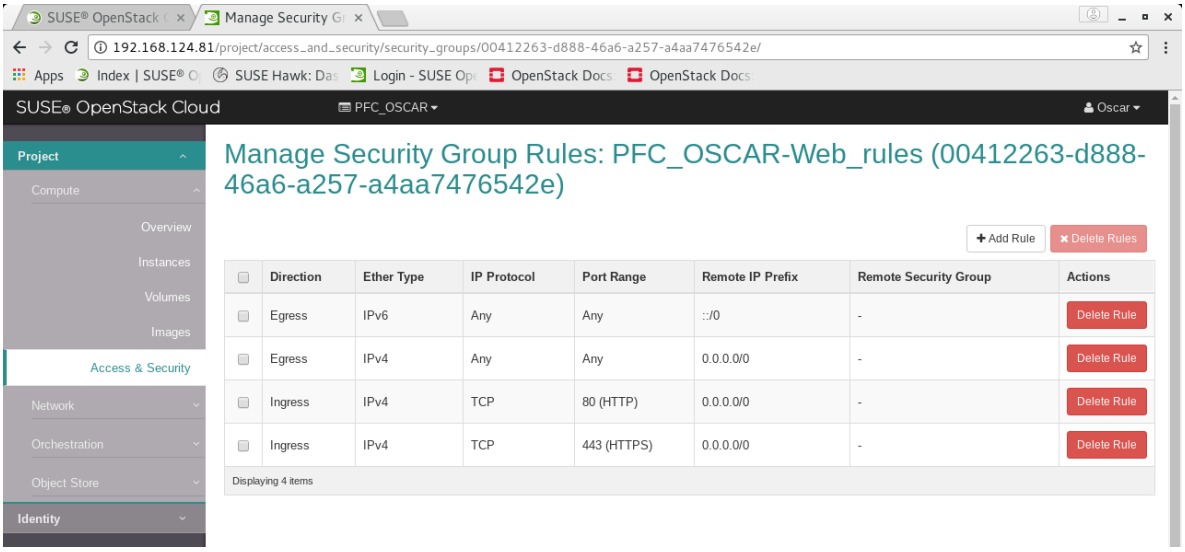

Figura 8 - 10 Vista general de les regles d'un Security Group

La darrera tasca al router és afegir-li una ip externa per poder accedir-hi des de l'exterior:

```
tux@linux: [admin@Default/admin (v3)]~/Oscar> openstack ip floating create floating
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| fixed_ip    | None                                 |
| id          | 9db8334c-6d13-43ae-97ff-150adc72af2f |
| instance_id | None                                 |
| ip          | 192.168.126.131                      |
| pool        | floating                             |
+-------------+--------------------------------------+
tux@linux: [admin@Default/admin (v3)]~/Oscar>
```

Les càrregues de treball

Per defecte a la instal·lació inicial ens crea uns flavors, podem crear-ne més i ajustar-los a les necessitats de les càrregues de treball. En crearem un per ajustant-lo a mínims, ja que el maquinari del nostre laboratori és limitat:

```
tux@linux:~/Oscar>OpenStack flavor create --ram 512  --disk 5 --vcpu 1 --public PFC_Oscar
```

```
+----------------------------+-----------------------------------+
| Field                      | Value                             |
+----------------------------+-----------------------------------+
| OS-FLV-DISABLED:disabled   | False                             |
| OS-FLV-EXT-DATA:ephemeral  | 0                                 |
| disk                       | 5                                 |
| id                         | 8a9c510a-dda4-48d5-a141-e68187d97718 |
| name                       | PFC_Oscar                         |
| os-flavor-access:is_public | True                              |
| ram                        | 512                               |
| rxtx_factor                | 1.0                               |
| swap                       |                                   |
| vcpus                      | 1                                 |
+----------------------------+-----------------------------------+
tux@linux:~/Oscar>
```

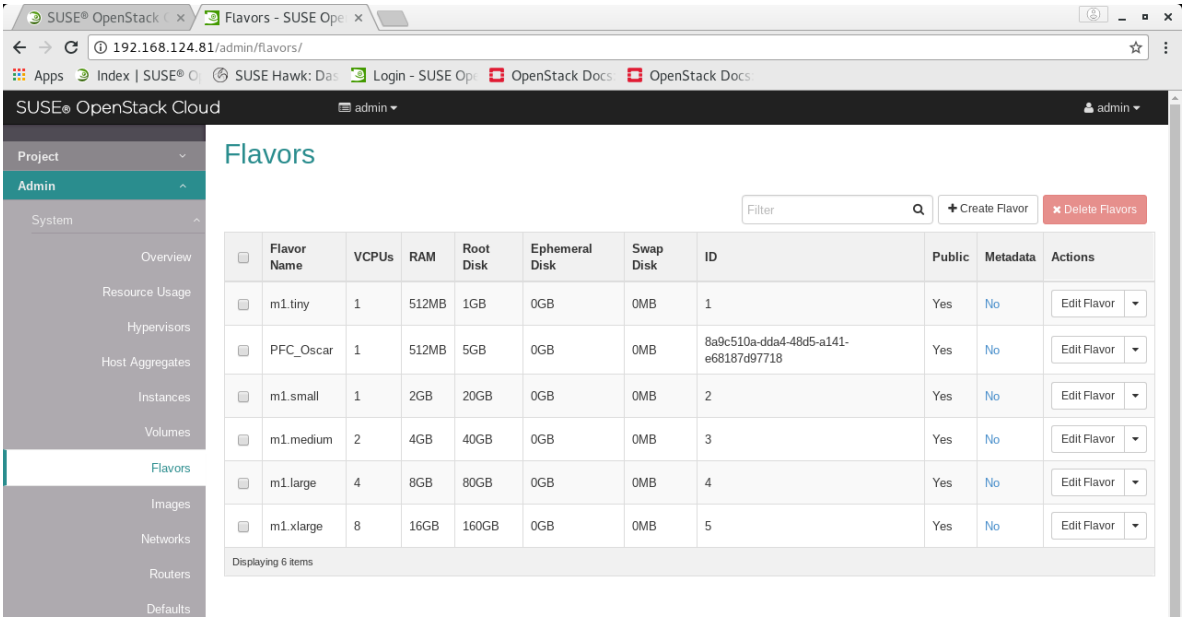

Figura 8 - 11 Flavors disponibles

Ara ja podem crear la nostra primera càrrega de treball:

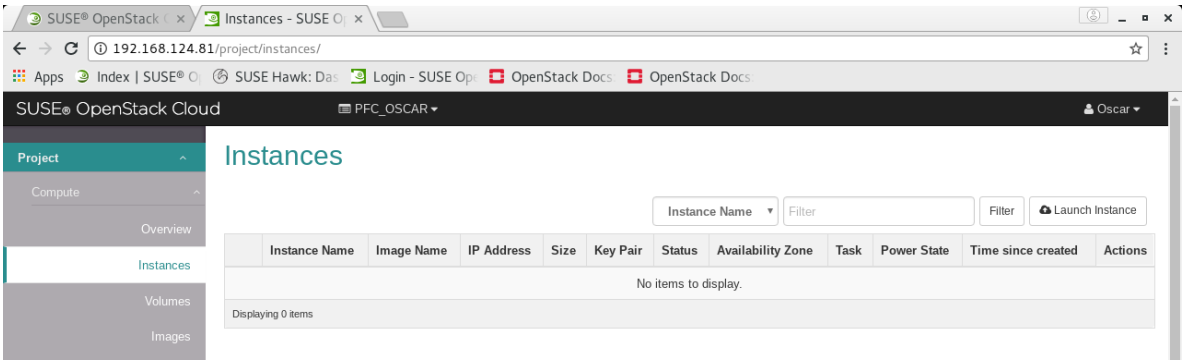

Figura 8 - 12 Vista general de les instàncies

Definim els paràmetres de la instància: nom, sabor, tipus d'engegada i la imatge, observem que ens mostra un detall dels recursos que ens consumirà, dels que té consumits i els que tenim assignats per quota:

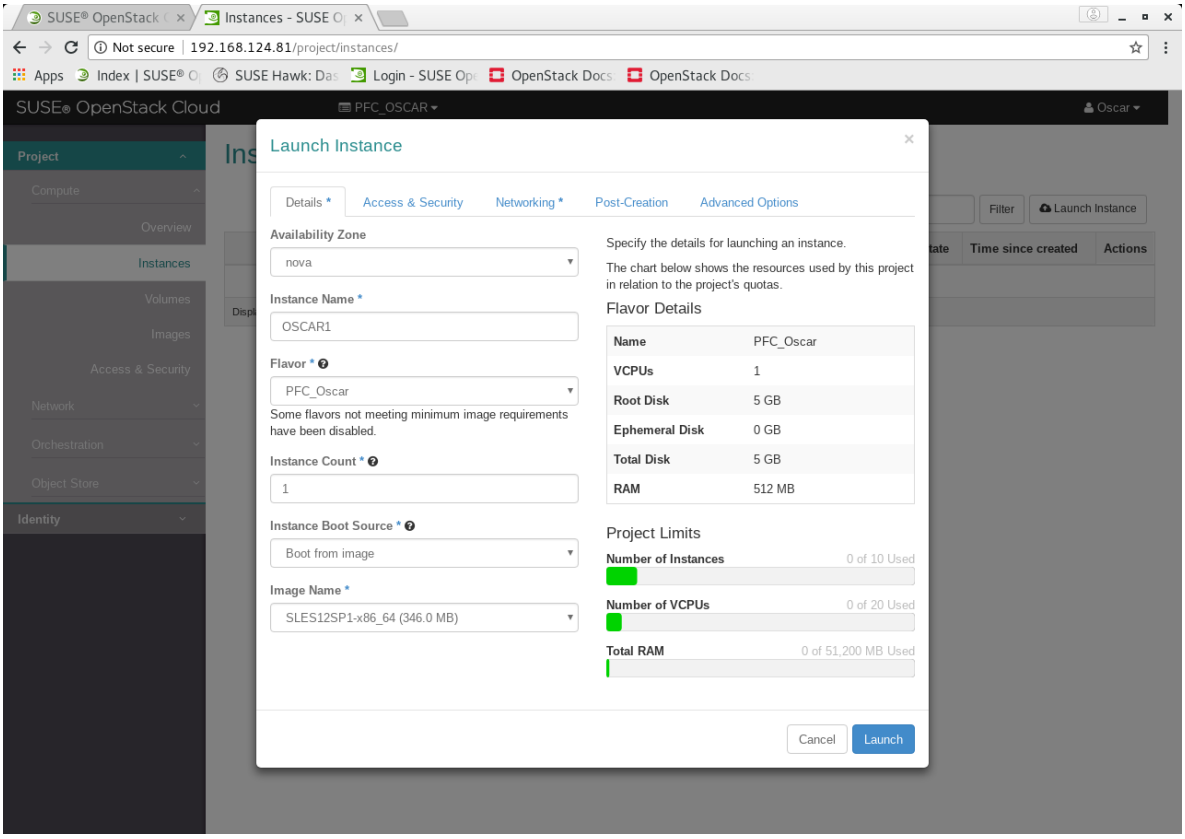

Figura 8 - 13 Desplegament d'una instància: dades generals

Definim les xarxes que volem que tingui, en el nostre cas la interna que hem creat:

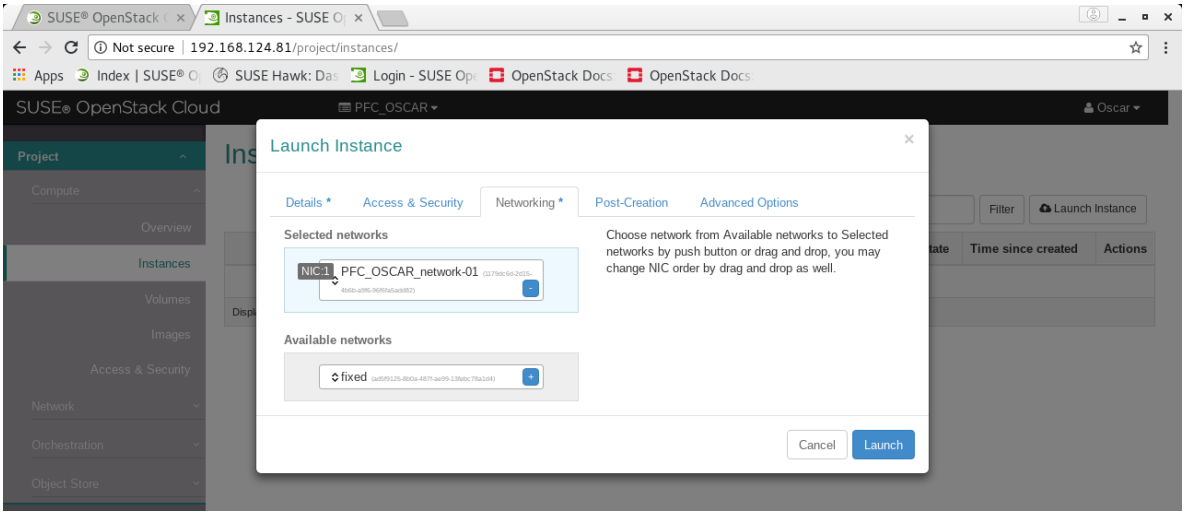

Figura 8 - 14 Desplegament d'una instància: networking

I ja podem executar-la:

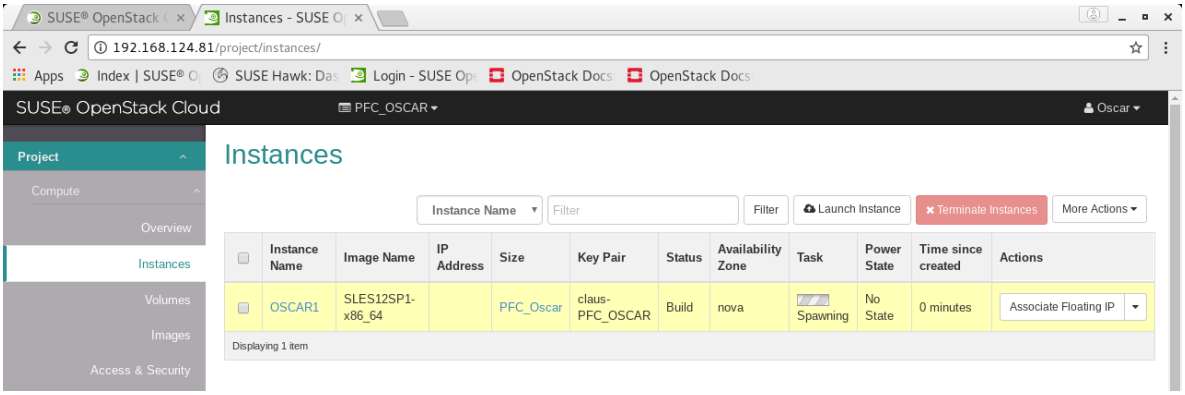

Figura 8 - 15 Desplegament d'una instància: execució

Al cap d'uns moments ja la tenim en marxa:

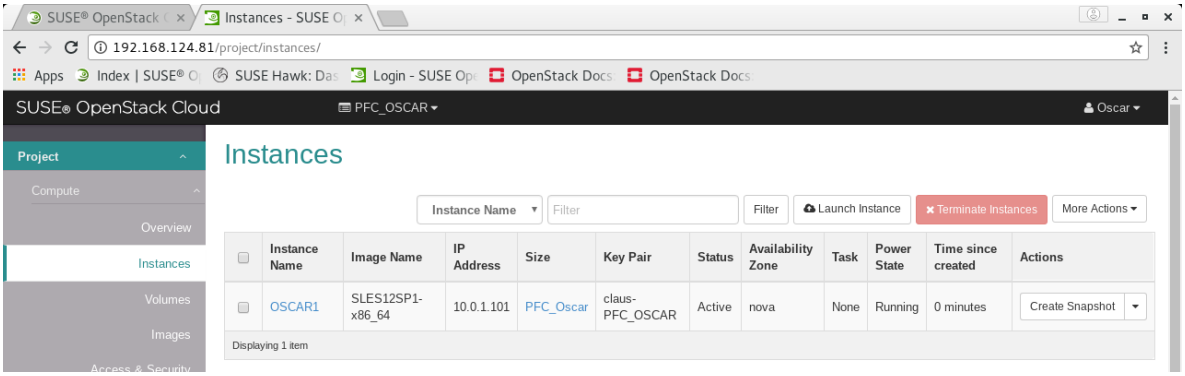

Figura 8 - 16 Instància en execució

Ara ja hi podem accedir. La manera més ràpida és des de la mateixa interfície web, on tenim una opció de consola:

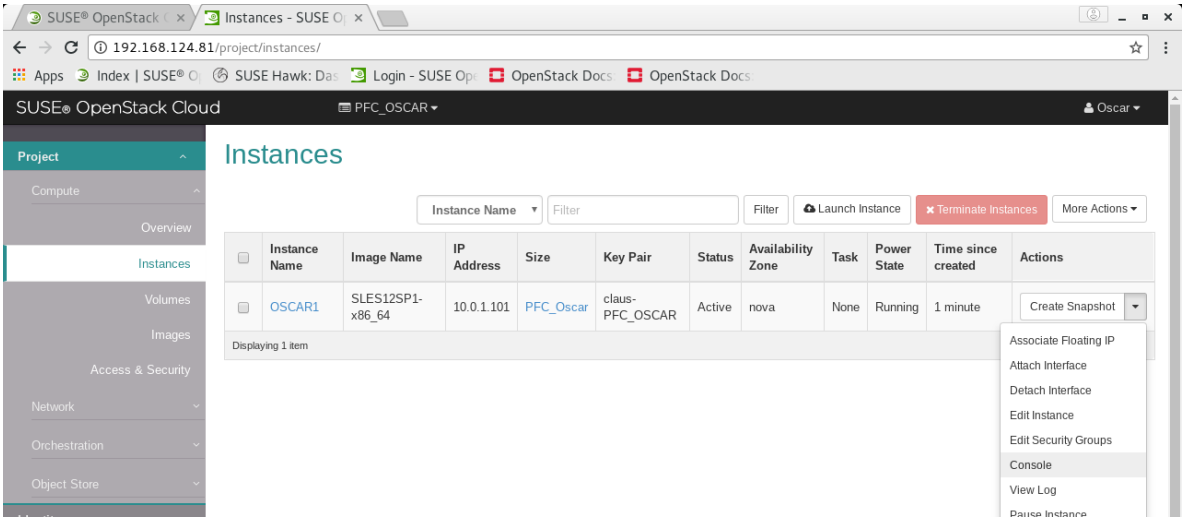

Figura 8 - 17 Accés a la consola des de la interfície web

Podem observar que tenim la instància funcionant:

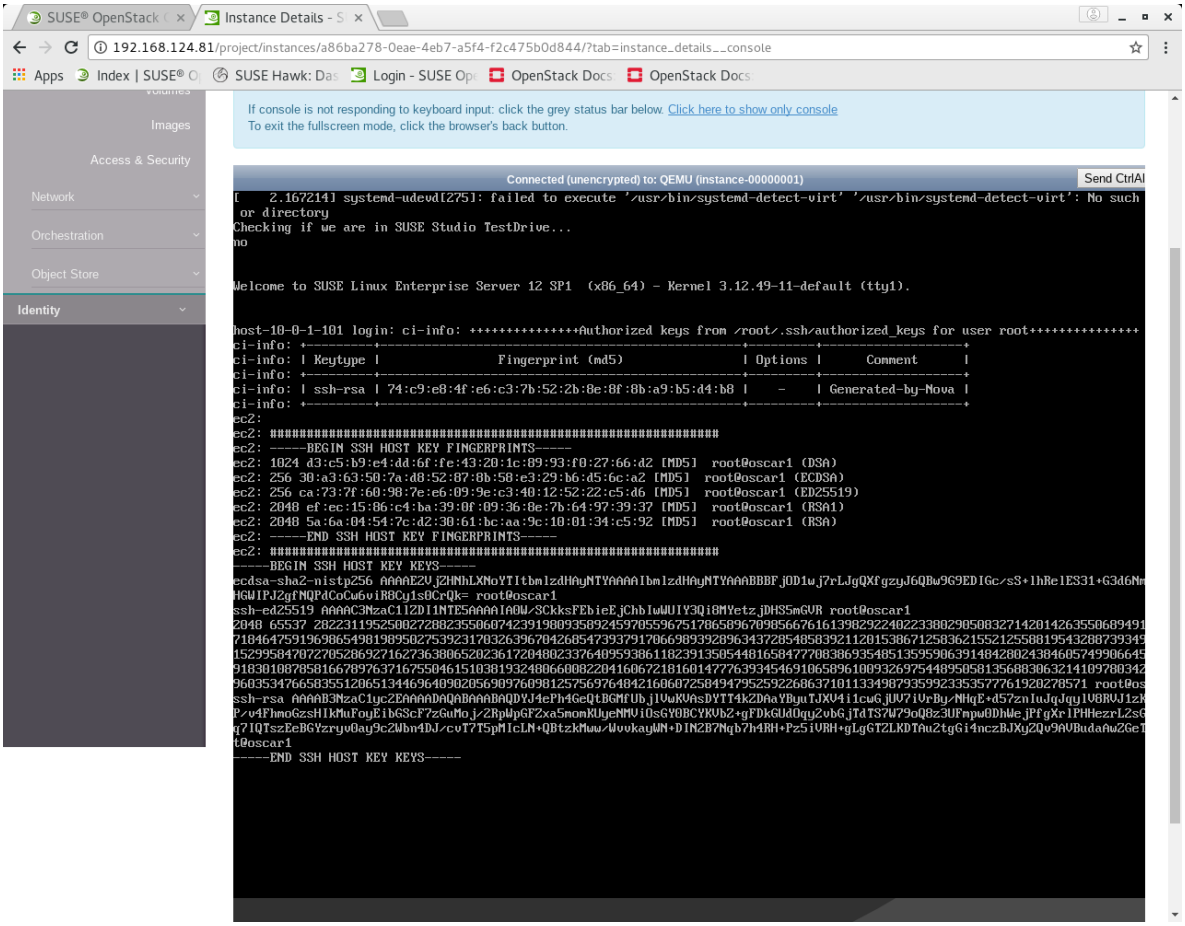

Figura 8 - 18 Consola de la instància

Si volem accedir des de l'exterior, tindrem que assignar una ip flotant i assignar els grups de seguretat per tal de poder connectar-nos per SSH o fer un ping:

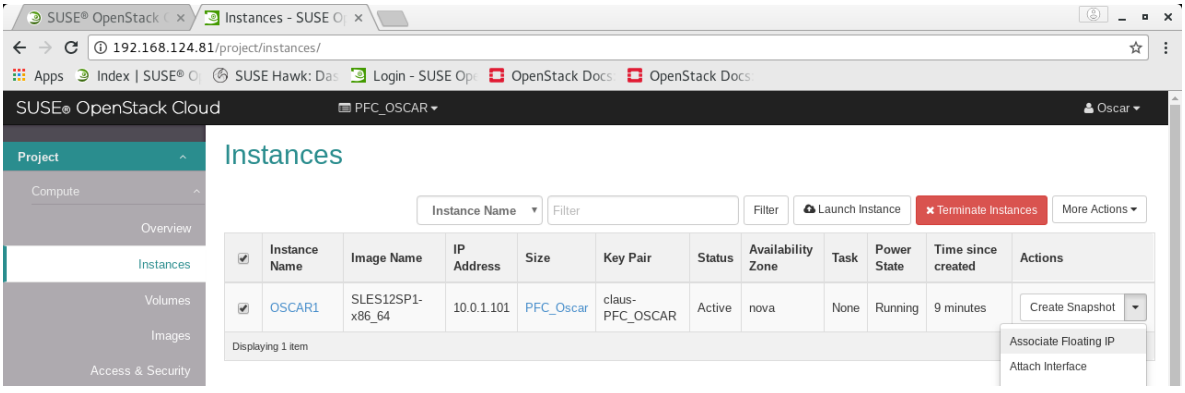

Figura 8 - 19 Menú d'associació d'una adreça flotant

Li assignem la ip 192.168.126.31, que és del pool d'adreces flotants disponibles:

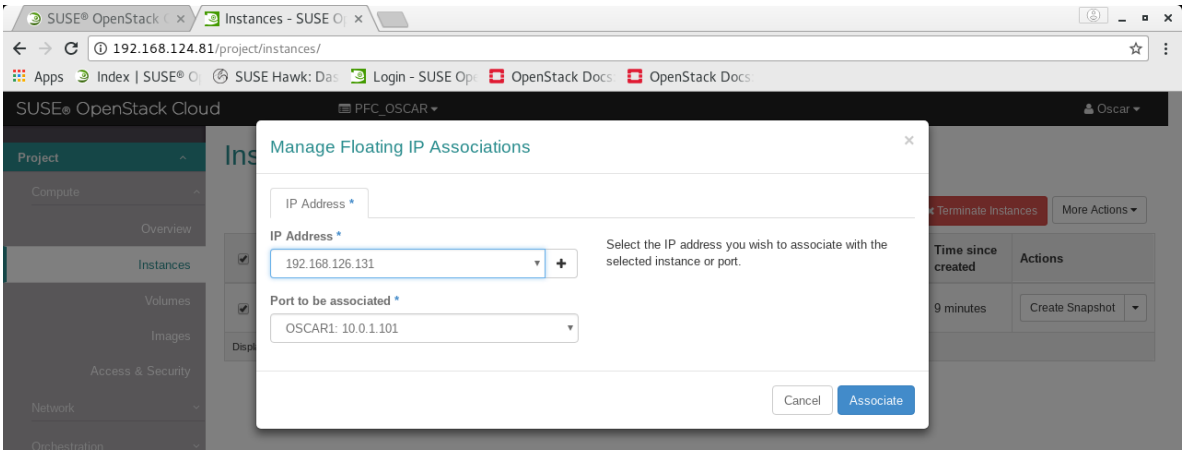

Figura 8 - 20 Associació d'una adreça flotant del pool de disponibles

I amb les comandes següents assignem les regles:

tux@linux:~/Oscar>OpenStack server add security group OSCAR1 PFC_OSCAR-ICMP_Rules

tux@linux:~/Oscar>OpenStack server add security group OSCAR1 PFC_OSCAR-SSH_Rules

Com podem veure ja podem accedir per fer ping i connectar per ssh:

```
tux@linux:~> sudo ping 192.168.126.131 -c 4
PING 192.168.126.131 (192.168.126.131) 56(84) bytes of data.
64 bytes from 192.168.126.131: icmp_seq=1 ttl=63 time=2.50 ms
64 bytes from 192.168.126.131: icmp_seq=2 ttl=63 time=8.76 ms
64 bytes from 192.168.126.131: icmp_seq=3 ttl=63 time=1.61 ms
64 bytes from 192.168.126.131: icmp_seq=4 ttl=63 time=1.07 ms

--- 192.168.126.131 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 1.070/3.488/8.765/3.089 ms
tux@linux:~> ssh root@192.168.126.131
Last login: Tue Dec 12 21:00:02 2017 from 192.168.126.254
oscar1:~ #
```

L'emmagatzematge de bloc

Un cop ja tenim càrregues de treball funcionant, ens cal dotar al sistema d'emmagatzematge de bloc per tal de proveir storage persistent a les nostres instàncies.

Per aquest fi, crearem un volum des de Horizon que residirà a Cinder y que posteriorment mitjançant comandes adjuntarem a la instància que ja tenim en marxa:

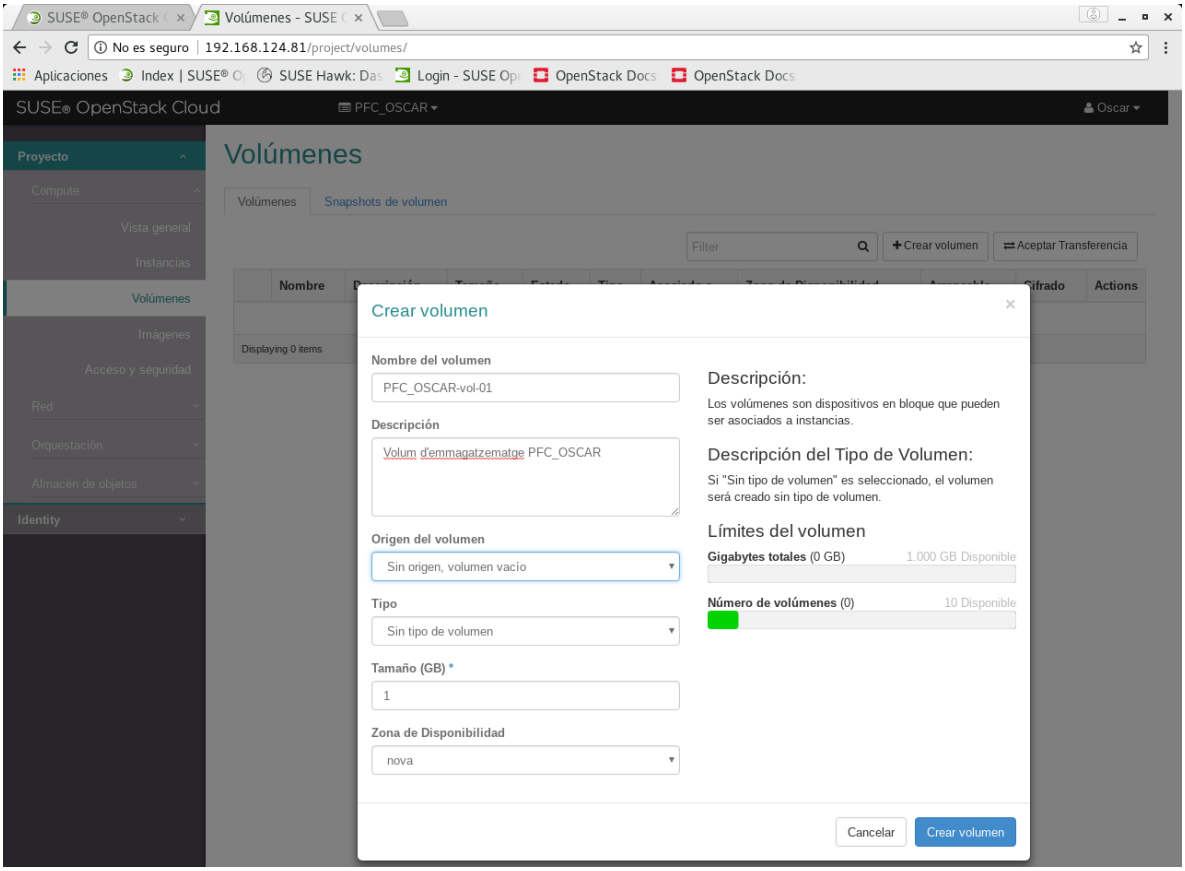

Figura 8 - 21 Creació d'un volum de Cinder

Un cop creada apareix a la llista de volums:

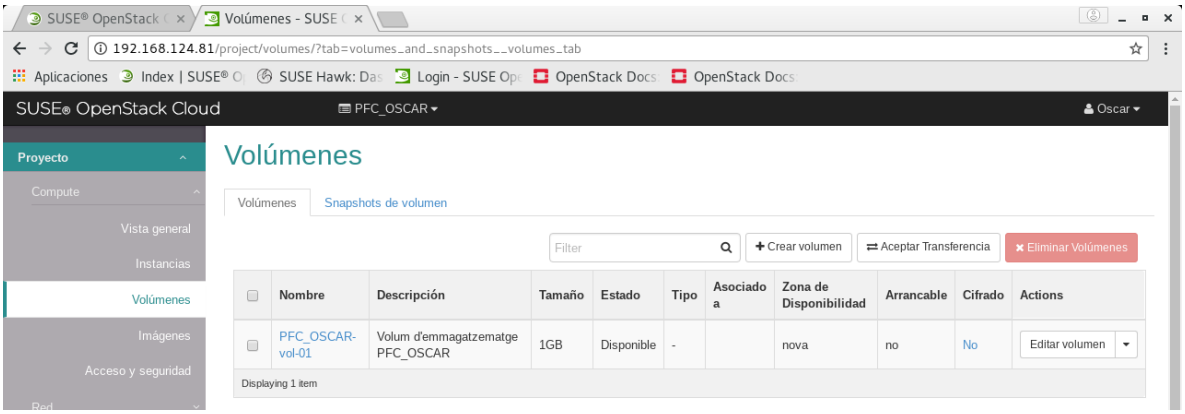

Figura 8 - 22 Vista general dels volums del projecte

Ara l'adjuntarem a la instància amb les següents comandes:

```
tux@linux:~/Oscar>openstack volume list
+--------------------------------------+-----------------+-----------+------+-------------+
| ID                                   | Display Name    | Status    | Size | Attached to |
+--------------------------------------+-----------------+-----------+------+-------------+
| 418aedfd-3077-459a-96d9-63587652ca8a | PFC_OSCAR-vol-01 | available |    1 |             |
+--------------------------------------+-----------------+-----------+------+-------------+
tux@linux:~/Oscar>openstack server list
+--------------------------------------+--------+--------+-----------------------------------------------+
| ID                                   | Name   | Status | Networks                                      |
+--------------------------------------+--------+--------+-----------------------------------------------+
| 6671c1b9-c0b4-44a4-aa77-ad4331986474 | OSCAR1 | ACTIVE | PFC_OSCAR_network-01=10.0.1.105, 192.168.126.131 |
+--------------------------------------+--------+--------+-----------------------------------------------+
tux@linux:~/Oscar>openstack server add volume OSCAR1 PFC_OSCAR-vol-01
tux@linux:~/Oscar>
```

Si des de Horizon refem la consulta dels volums veurem que la tenim assignada a la instància:

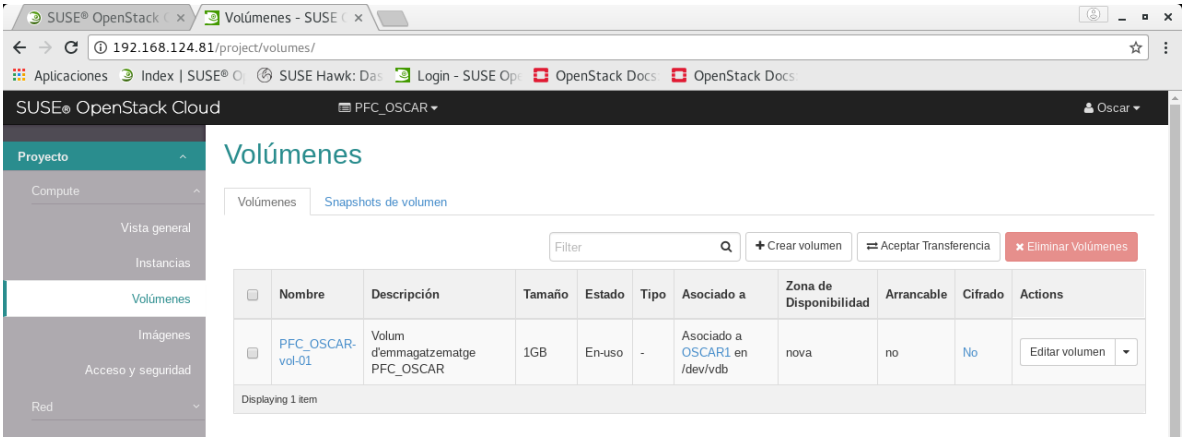

Figura 8 - 23 Assignació d'un volum a una instància

Si accedim a la instància veurem que tenim disponible el disc:

```
tux@linux:~/Oscar>ssh root@192.168.126.131
Last login: Sat Dec 16 10:28:11 2017 from 192.168.126.254
oscar1:~ # lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vda    254:0    0   5G  0 disk
└─vda1 254:1    0   5G  0 part /
vdb    254:16   0   1G  0 disk
oscar1:~ #
```

El procediment per desassignar el volum a la instància és similar i el farem amb la comanda:

```
tux@linux:~/Oscar>openstack server remove volume OSCAR1 PFC_OSCAR-vol-01
tux@linux:~/Oscar>
tux@linux:~/Oscar>ssh root@192.168.126.131
Last login: Sat Dec 16 10:28:58 2017 from 192.168.126.254
oscar1:~ # lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vda    254:0    0   5G  0 disk
└─vda1 254:1    0   5G  0 part /
oscar1:~ #
```

Un cop ja hem creat volums i els hem assignat a les instàncies crearem snapshots d'aquest volum, això ens permetrà poder generar una còpia idèntica i poder-la guardar o continuar treballant amb una i/o altra.

Per veure el funcionamet del snapshot. Crearem la partició, el sistema de fitxers i un fitxer dins del volum:

```
tux@linux:~/Oscar>ssh root@192.168.126.131
Last login: Sat Dec 16 10:33:45 2017 from 192.168.126.254
oscar1:~ # lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vda    254:0    0   5G  0 disk
└─vda1 254:1    0   5G  0 part /
vdb    254:16   0   1G  0 disk
oscar1:~ # mount /dev/vdb /mnt/
oscar1:~ # cd /mnt/
oscar1:/mnt # ll
-bash: ll: command not found
oscar1:/mnt # touch primerfitxer
oscar1:/mnt # ls
lost+found  primerfitxer
oscar1:/mnt #
```

Ara crearem un snapshot sobre el volum, es recomanable que al fer el snapshot el volum estigui desmuntat i per anar bé que cap instància en tingui accés, ja que sinó podríem deixar el sistema de fitxers corromput:

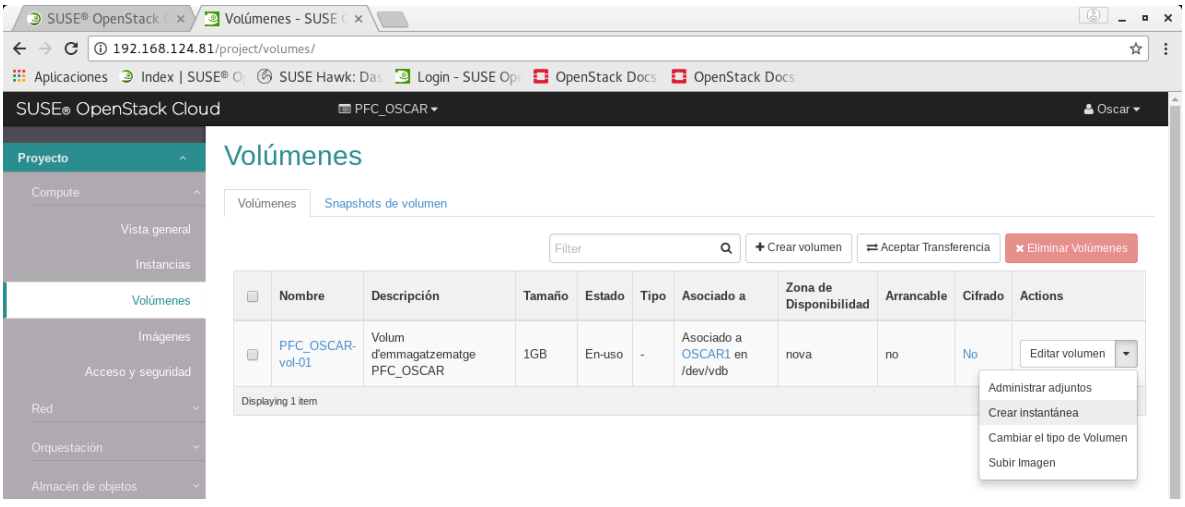

Figura 8 - 24 Creació d'un Snapshot del volum

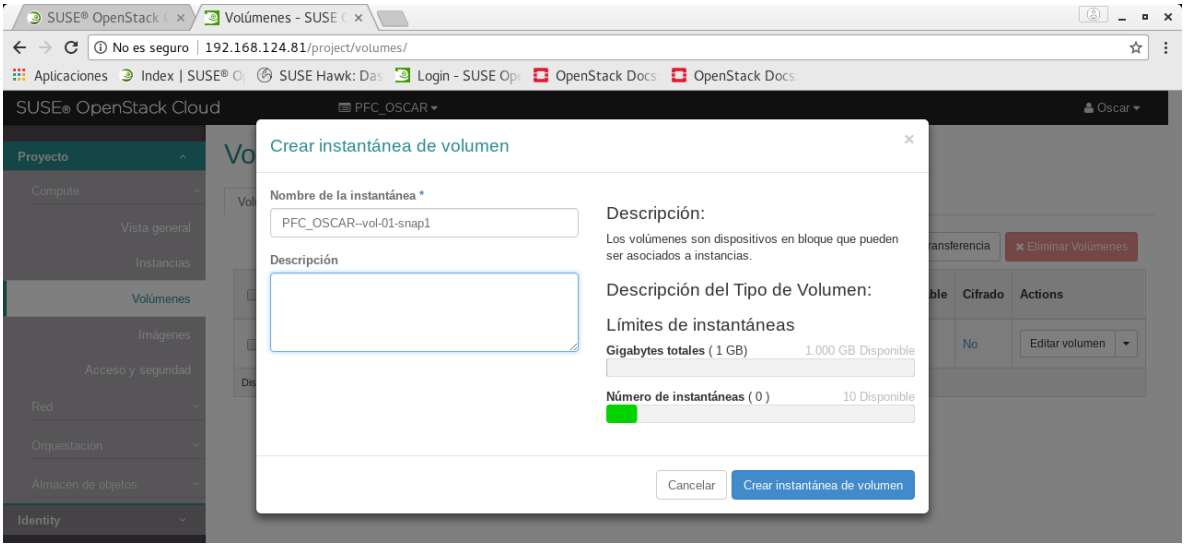

Figura 8 - 25 Creació d'un Snapshot del volum: paràmetres

Els snapshots no són accessibles directament pel sistema, en cas de voler fer-lo assignar a una instància ens cal convertir-lo a volum:

```
tux@linux:~/Oscar>openstack snapshot list
+--------------------------------------+-----------------------+-------------+-----------+------+
| ID                                   | Name                  | Description | Status    | Size |
+--------------------------------------+-----------------------+-------------+-----------+------+
| 5c997d76-7ea5-45ad-bfc9-e926a81524bf | PFC_OSCAR--vol-01-snap1 |             | available |    1 |
+--------------------------------------+-----------------------+-------------+-----------+------+
tux@linux:~/Oscar>openstack volume create --size 2 --snapshot 5c997d76-7ea5-45ad-bfc9-e926a81524bf \
> --description "Creació de volum a partir PFC_OSCAR-vol-01-snap1" \
> PFC_OSCAR-vol-02
+---------------------+------------------------------------------------+
| Field               | Value                                          |
+---------------------+------------------------------------------------+
| attachments         | []                                             |
| availability_zone   | nova                                           |
| bootable            | false                                          |
| created_at          | 2017-12-16T10:56:45.567461                     |
| display_description | Creació de volum a partir PFC_OSCAR-vol-01-snap1 |
| display_name        | PFC_OSCAR-vol-02                               |
```

```
| encrypted           | False                                |
| id                  | bae85c4b-37ba-436d-84af-81d2e59b14a1 |
| multiattach         | false                                |
| properties          |                                      |
| size                | 2                                    |
| snapshot_id         | 5c997d76-7ea5-45ad-bfc9-e926a81524bf |
| source_volid        | None                                 |
| status              | creating                             |
| type                | None                                 |
+---------------------+--------------------------------------+
tux@linux:~/Oscar>openstack volume list
+--------------------------------------+-----------------+-----------+------+-------------+
| ID                                   | Display Name    | Status    | Size | Attached to |
+--------------------------------------+-----------------+-----------+------+-------------+
| bae85c4b-37ba-436d-84af-81d2e59b14a1 | PFC_OSCAR-vol-02 | available |    2 |             |
| 418aedfd-3077-459a-96d9-63587652ca8a | PFC_OSCAR-vol-01 | available |    1 |             |
+--------------------------------------+-----------------+-----------+------+-------------+
tux@linux:~/Oscar>
```

Ara si adjuntem a la instància els dos volums, veurem que es comporten de manera independent:

```
tux@linux:~/Oscar>ssh root@192.168.126.131
Last login: Sat Dec 16 10:38:04 2017 from 192.168.126.254
oscar1:~ # lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vda    254:0    0   5G  0 disk
└─vda1 254:1    0   5G  0 part /
vdb    254:16   0   2G  0 disk
vdc    254:32   0   1G  0 disk
oscar1:~ # mkdir /mnt/b
oscar1:~ # mkdir /mnt/c
oscar1:~ # mount /dev/vdb /mnt/b
oscar1:~ # mount /dev/vdc /mnt/c
```

```
oscar1:~ # touch /mnt/b/segonfitxer
oscar1:~ # ls -al /mnt/c/
total 24
drwxr-xr-x 3 root root  4096 dic 16 10:40 .
drwxr-xr-x 4 root root  4096 dic 16 10:59 ..
drwx------ 2 root root 16384 dic 16 10:40 lost+found
-rw-r--r-- 1 root root     0 dic 16 10:40 primerfitxer
oscar1:~ # ls -al /mnt/c/
total 24
drwxr-xr-x 3 root root  4096 dic 16 10:40 .
drwxr-xr-x 4 root root  4096 dic 16 10:59 ..
drwx------ 2 root root 16384 dic 16 10:40 lost+found
-rw-r--r-- 1 root root     0 dic 16 10:40 primerfitxer
oscar1:~ # df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1       4,9G  808M  3,9G  17% /
devtmpfs        237M  8,0K  237M   1% /dev
tmpfs           245M     0  245M   0% /dev/shm
tmpfs           245M  4,5M  241M   2% /run
tmpfs           245M     0  245M   0% /sys/fs/cgroup
/dev/vdb        976M  1,3M  908M   1% /mnt/b
/dev/vdc        976M  1,3M  908M   1% /mnt/c
oscar1:~ #
```

Fins aquí, hem vist com treballar amb instàncies i com tenir volums per tal de poder-hi emmagatzemar dades persistents, ara veurem com crear instàncies a partir de volums amb "Boot from volume" per tal de que els canvis que fem sobre la màquina virtual siguin persistents.

Primer crearem el volum a partir de la imatge:

```
tux@linux:~/Oscar>openstack image list
+--------------------------------------+---------------------+
| ID                                   | Name                |
+--------------------------------------+---------------------+
| 324f6306-f4e6-4f98-8df7-1825dccebf1a | SLES12SP1-x86_64    |
| 9868b3ba-dade-4afa-bb06-3c008b5ea8ce | xenialpup64-7.5-uefi |
```

```
+--------------------------------------+---------------------+
tux@linux:~/Oscar>openstack volume create \
--image 324f6306-f4e6-4f98-8df7-1825dccebf1a --size 5 SLES12SP1-vol
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attachments         | []                                   |
| availability_zone   | nova                                 |
| bootable            | false                                |
| created_at          | 2017-12-16T11:04:44.918326           |
| display_description | None                                 |
| display_name        | SLES12SP1-vol                        |
| encrypted           | False                                |
| id                  | f0e3a93a-d5e4-45fb-86e8-8c9c917b852d |
| image_id            | 324f6306-f4e6-4f98-8df7-1825dccebf1a |
| multiattach         | false                                |
| properties          |                                      |
| size                | 5                                    |
| snapshot_id         | None                                 |
| source_volid        | None                                 |
| status              | creating                             |
| type                | None                                 |
+---------------------+--------------------------------------+
tux@linux:~/Oscar>openstack volume list
+--------------------------------------+-----------------+-----------+------+------------------------------+
| ID                                   | Display Name    | Status    | Size | Attached to                  |
+--------------------------------------+-----------------+-----------+------+------------------------------+
| f0e3a93a-d5e4-45fb-86e8-8c9c917b852d | SLES12SP1-vol   | available |    5 |
|
| bae85c4b-37ba-436d-84af-81d2e59b14a1 | PFC_OSCAR-vol-02 | in-use    |    2 |
Attached to OSCAR1 on /dev/vdb |
| 418aedfd-3077-459a-96d9-63587652ca8a | PFC_OSCAR-vol-01 | in-use    |    1 |
Attached to OSCAR1 on /dev/vdc |
+--------------------------------------+-----------------+-----------+------+------------------------------+
tux@linux:~/Oscar>
```

Ara ja que disposem del volum podrem crear la instància:

```
tux@linux:~/Oscar>openstack network list
+--------------------------------------+--------------------+--------------------------------------+
| ID                                   | Name               | Subnets                              |
+--------------------------------------+--------------------+--------------------------------------+
| 1179dc6d-2d15-4b6b-a9f6-96f6fa5add82 | PFC_OSCAR_network-01 | bf36e224-da29-4d4a-a4fe-49bb3b753386 |
| 163fbcba-8197-4715-b352-fe58d4fd01da | floating           | bfbb9631-e397-4e92-88bc-7972f8e15f04 |
| 4f3ad586-afa7-4d9d-bd88-086dee5bba05 | external-2         | a5a88f47-7735-4387-be88-0385c1c5a618 |
| ad5f9125-8b0a-487f-ae99-13febc78a1d4 | fixed              | 5880b9e9-dd4c-4b4f-b41f-678e9f1fe507 |
+--------------------------------------+--------------------+--------------------------------------+
tux@linux:~/Oscar>openstack server create \
> --flavor PFC_Oscar \
> --volume SLES12SP1-vol \
> --key-name PFC_OSCAR-key02 \
> --nic net-id=1179dc6d-2d15-4b6b-a9f6-96f6fa5add82 \
> OSCAR2
+--------------------------------------+------------------------------------------------+
| Field                                | Value                                          |
+--------------------------------------+------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                         |
| OS-EXT-AZ:availability_zone          |                                                |
| OS-EXT-STS:power_state               | 0                                              |
| 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                            | GtwdZr4JcUFC                                   |
| config_drive                         |                                                |
| created                              | 2017-12-16T11:13:45Z                           |
```

```
| flavor                          | PFC_Oscar (8a9c510a-dda4-48d5-a141-e68187d97718)   |
| hostId                          |                                                    |
| id                              | e2d8c224-a5a4-4e8d-baa5-f0c5e4fd836d               |
| image                           |                                                    |
| key_name                        | PFC_OSCAR-key02                                    |
| name                            | OSCAR2                                             |
| os-extended-volumes:volumes_attached | [{u'id': u'f0e3a93a-d5e4-45fb-86e8-8c9c917b852d'}] |
| progress                        | 0                                                  |
| project_id                      | db653b636b794ed581a6c95efaf74938                   |
| properties                      |                                                    |
| security_groups                 | [{u'name': u'default'}]                            |
| status                          | BUILD                                              |
| updated                         | 2017-12-16T11:13:45Z                               |
| user_id                         | 36c83fb1c2c644ce9e7fe3300f2fb2a8                   |
+---------------------------------+----------------------------------------------------+
tux@linux:~/Oscar>
```

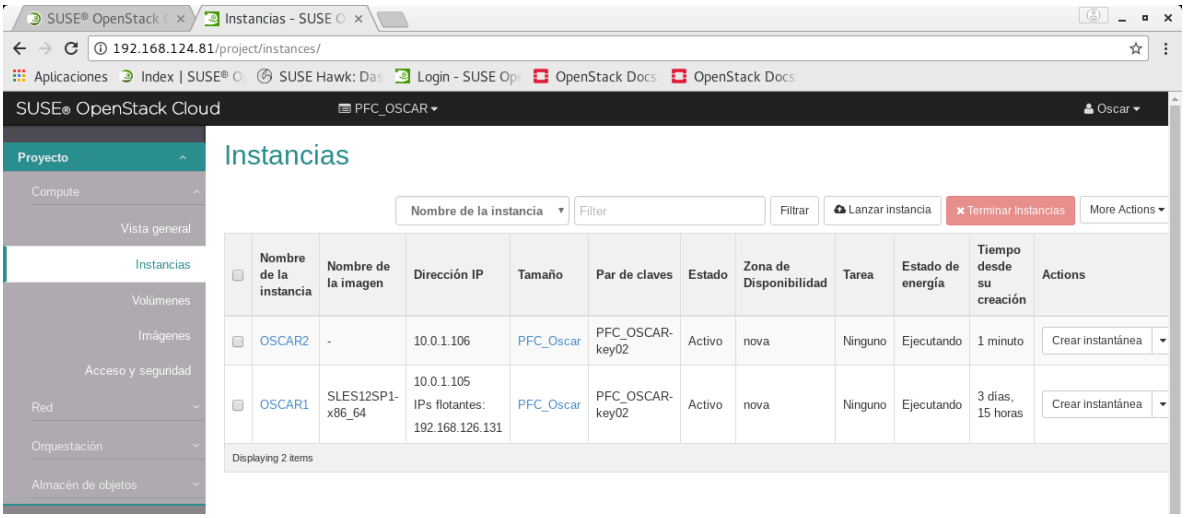

Figura 8 - 26 Vista general de les instàncies

Ara ja podem accedir a la nova instància, com que no li hem assignat ip flotant, ho farem mitjançant la màquina virtual:

```
tux@linux:~/Oscar>ssh root@192.168.126.131
Last login: Sat Dec 16 10:58:33 2017 from 192.168.126.254
```

```
oscar1:~ # ping 10.0.1.106
PING 10.0.1.106 (10.0.1.106) 56(84) bytes of data.
64 bytes from 10.0.1.106: icmp_seq=1 ttl=64 time=2.67 ms
64 bytes from 10.0.1.106: icmp_seq=2 ttl=64 time=1.31 ms
^C
--- 10.0.1.106 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.319/1.995/2.672/0.677 ms
oscar1:~ #
```

L'emmagatzematge d'objectes

Fins ara hem vist com treballar amb instàncies tant de manera efímera com persistent i adjuntar volums amb emmagatzematge de blocs.

Tal com hem vist en capítols anteriors, a OpenStack tenim la possibilitat d'emmagatzemar objectes. Aquest servei és independent a la resta i no complementa cap d'altra, això vol dir que es comporta com un emmagatzemament que pot ser consumit o bé pels usuaris que executin càrregues de treball al nostre OpenStack o per d'altres que puguin necessitar aquest servei. Aquest servei d'objectes és compatible amb Amazon S3, per lo que el fa interessant ja sigui per tenir entorns de backup, de proves o simplement migrar dades d'un cap a l'altre.

El primer que farem és crear el contenidor per poder emmagatzemar aquests objectes:

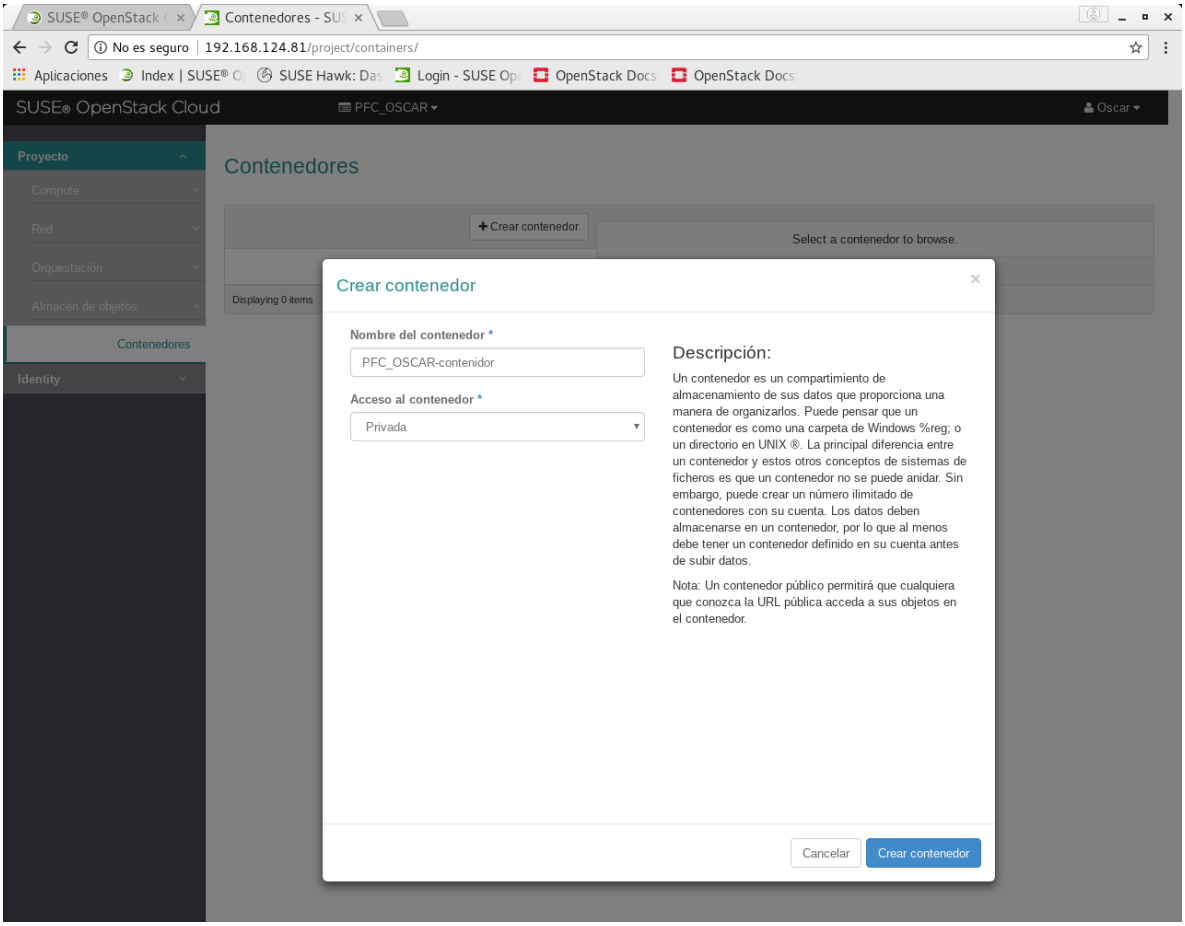

Figura 8 - 27 Creació d'un contenidor d'objectes Swift

Des de comandes podem gestionar els objectes:

```
tux@linux:~/Documents/opensuse>ll
total 22364
-rw-r--r-- 1 tux users 3245206 dic 16 13:14 book.gnomeuser_color_en.pdf
-rw-r--r-- 1 tux users 4977797 dic 16 13:14 book.opensuse.reference_color_en.pdf
-rw-r--r-- 1 tux users 5530367 dic 16 13:13 book.opensuse.startup_color_en.pdf
-rw-r--r-- 1 tux users 3187678 dic 16 13:13 book.security_color_en.pdf
-rw-r--r-- 1 tux users 1334992 dic 16 13:13 book.sle.tuning_color_en.pdf
-rw-r--r-- 1 tux users 4586491 dic 16 13:13 book.virt_color_en.pdf
tux@linux:~/Documents/opensuse>openstack container list
+----------------------+
| Name                 |
+----------------------+
```

```
| PFC_OSCAR-contenidor |
+----------------------+
tux@linux:~/Documents/opensuse>openstack object create \
>     PFC_OSCAR-contenidor \
>     boo*.pdf
+-----------------------------------+--------------------+----------------------------------+
| object                            | container          | etag                             |
+-----------------------------------+--------------------+----------------------------------+
|    book.gnomeuser_color_en.pdf                         |    PFC_OSCAR-contenidor    |
63c4b0ab4fa20758237ec092e4b66292 |
|         book.opensuse.reference_color_en.pdf          |         PFC_OSCAR-contenidor         |
aa1accbc0433429ee25b0a7f0d0662de |
|       book.opensuse.startup_color_en.pdf              |       PFC_OSCAR-contenidor       |
fc8d0891d922f5ae2c4a4b14e851c894 |
|    book.security_color_en.pdf                         |    PFC_OSCAR-contenidor    |
12f0c825c579d1f70ee8add915a50769 |
|     book.sle.tuning_color_en.pdf                      |    PFC_OSCAR-contenidor    |
18f838b4bc1b9d2741f33653537021b0 |
|   book.virt_color_en.pdf                              |    PFC_OSCAR-contenidor    |
10230e03ffbfd49c342658ca788988aa |
+-----------------------------------+--------------------+----------------------------------+
tux@linux:~/Documents/opensuse>
```

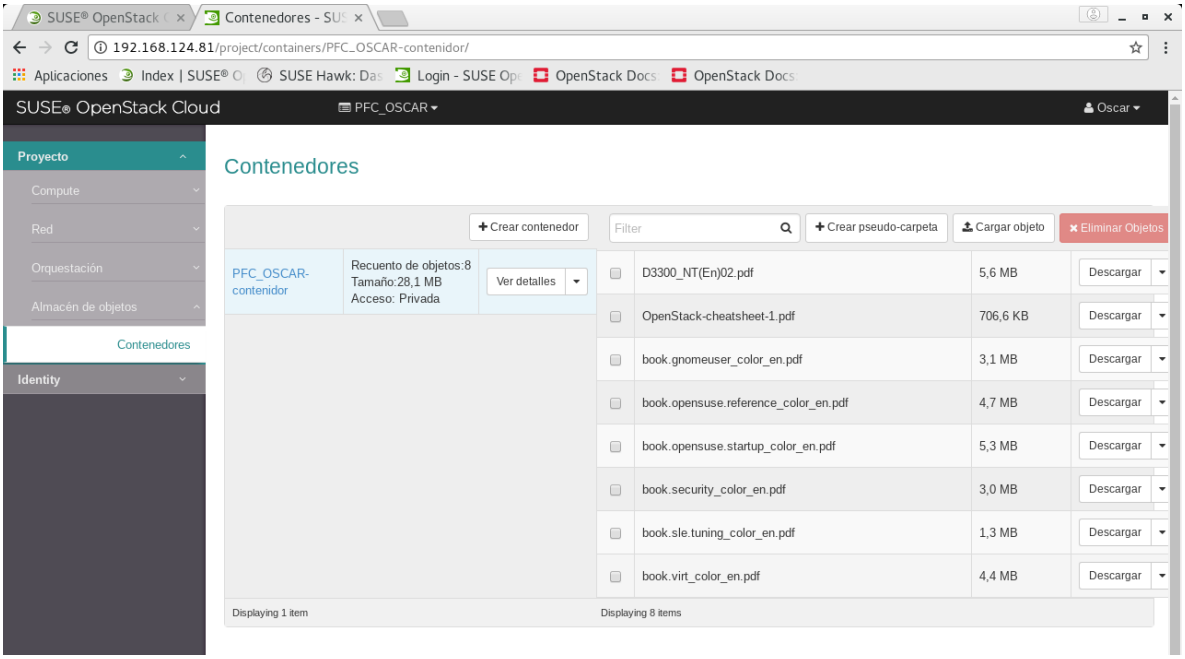

Figura 8 - 28 Vista general dels objectes del contenidor

Hem vist com carregar objectes dins d'un contenidor, ara podem gestionar-los des del mateix Horizon o podem fer-ho per comandes, assignant permisos o fins i tot accedit a ells per http.

Per veure les propietats del contenidor:

tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar> swift stat PFC_OSCAR-contenidor

Account: AUTH_db653b636b794ed581a6c95efaf74938
Container: PFC_OSCAR-contenidor
Objects: 8
Bytes: 29416842
Read ACL:
Write ACL:
Sync To:
Sync Key:
Accept-Ranges: bytes
X-Storage-Policy: Policy-0
X-Timestamp: 1513425545.38879
X-Trans-Id: tx5b27ceb2beaf4f9a80d67-005a351563

```
    Content-Type: text/plain; charset=utf-8
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar>
```

Podem observar que tant les ACL's de lectura com d'escriptura estan en blanc, i que si accedim als objectes via http sense autenticar retornarà un error perquè no són objectes públics:

```
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar> nova endpoints | grep -A 8 swift
| swift     | Value                                                                    |
+-----------+--------------------------------------------------------------------------+
| global    | True                                                                     |
| id        | 71f3f5af4e784a2a8a9c1a178e3abcde                                         |
| interface | public                                                                   |
| region    | RegionOne                                                                |
| region_id | RegionOne                                                                |
| url       | http://controller01:8080/v1/AUTH_db653b636b794ed581a6c95efaf74938 |
+-----------+--------------------------------------------------------------------------+
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar> wget http://controller01:8080/v1/AUTH_db653b636b794ed581a6c95efaf74938/PFC_OSCAR-contenidor/OpenStack-cheatsheet-1.pdf
--2017-12-16 13:52:43-- http://controller01:8080/v1/AUTH_db653b636b794ed581a6c95efaf74938/PFC_OSCAR-contenidor/OpenStack-cheatsheet-1.pdf
Resolviendo controller01 (controller01)... 192.168.124.81
Conectando con controller01 (controller01)[192.168.124.81]:8080... conectado.
Petición HTTP enviada, esperando respuesta... 401 Unauthorized

La autentificación usuario/contraseña falló.
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar>
```

Mitjançant les ACL's donarem permisos per fer el contenidor públic:

```
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar> swift post -r ".r:*" PFC_OSCAR-contenidor
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar> swift stat PFC_OSCAR-contenidor
       Account: AUTH_db653b636b794ed581a6c95efaf74938
```


```
      Container: PFC_OSCAR-contenidor
        Objects: 8
          Bytes: 29416842
       Read ACL: .r:*
      Write ACL:
        Sync To:
       Sync Key:
  Accept-Ranges: bytes
     X-Trans-Id: txcc729b894b35401d8c96f-005a3519fe
X-Storage-Policy: Policy-0
     X-Timestamp: 1513425545.38879
   Content-Type: text/plain; charset=utf-8
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar> wget
http://controller01:8080/v1/AUTH_db653b636b794ed581a6c95efaf74938/PFC_OSCAR-
contenidor/OpenStack-cheatsheet-1.pdf
--2017-12-16 14:05:09--
http://controller01:8080/v1/AUTH_db653b636b794ed581a6c95efaf74938/PFC_OSCAR-
contenidor/OpenStack-cheatsheet-1.pdf
Resolviendo controller01 (controller01)... 192.168.124.81
Conectando con controller01 (controller01)[192.168.124.81]:8080... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 723573 (707K) [application/pdf]
Grabando a: “OpenStack-cheatsheet-1.pdf”

OpenStack-cheatsheet-1.pdf
100%[=============================================================
=>] 706,61K  --.-KB/s    in 0,02s

2017-12-16 14:05:09 (35,7 MB/s) - “OpenStack-cheatsheet-1.pdf” guardado
[723573/723573]

tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar>
```

Orquestració: automatització del desplegament

Fins aquest apartat hem vist com gestionar el nostre núvol Openstack amb Horizon i com fer-ho amb línies de comanda. Ara és el torn de Heat, que ens permetrà automatitzar el cicle de vida de qualsevol component d'OpenStack.

Tal com ja hem vist en capítols anteriors, Heat utilitza el llenguatge HOT[37]. L'esquelet dels templates (plantilles) que són els fitxers de definició consten de les següents seccions:

- Version: és obligatòria, és en format data i dependent d'aquesta indica al motor de Heat en quina versió està escrita, disposant de més o menys funcionalitats[38]
- Description: també és una secció obligatòria i és una petita descripció del que fa la plantilla
- Parameter_groups: en cas d'haver-hi paràmetres especifica com s'agruparan i en l'ordre que es proporcionaran
- Parameters: en cas d'haver paràmetres els declararem en aquesta secció
- Resources: és la darrera secció obligatòria i conté les instruccions per crear els recursos que ha de crear la plantilla
- Output: en cas d'existir paràmetres de sortida els declararem en aquesta secció
- Conditions: Aquesta secció opcional inclou condicions que calen per poder executar la plantilla, sol ser utilitzada per verificar si hi ha recursos necessaris per que es completi o per verificar possibles prerequisits de la plantilla

Un cop ja hem vist l'estructura, ja podem començar a utilitzar-los. El primer que farem, és definir una plantilla per desplegar una càrrega de treball nova:

```
heat_template_version: 2015-10-15
description: Plantilla per desplegar una instància
resources:
  OSCAR3:
    type: OS::Nova::Server
```

[37] https://docs.openstack.org/heat/latest/template_guide/hello_world.html

[38] https://docs.openstack.org/heat/latest/template_guide/hot_spec.html#heat-template-version

```
  properties:
    key_name: PFC_OSCAR-key02
    image: SLES12SP1-x86_64
    flavor: PFC_Oscar
    networks:
      - network: PFC_OSCAR_network-01
```

El contingut d'aquesta plantilla el copiarem a un fitxer que l'anomenarem desplegar-instancia.template

A l'acció d'executar la plantilla es diu crear un stack i és el que farem:

```
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar/template> heat stack-create -f desplegar-instancia.template PFC_OSCAR-stack-1
+--------------------------------------+------------------+--------------------+--------------------------+--------------+
| id                                   | stack_name       | stack_status       | creation_time            | updated_time |
+--------------------------------------+------------------+--------------------+--------------------------+--------------+
| 48da0824-2771-4b0c-a31d-a5d869640886 | PFC_OSCAR-stack-1 | CREATE_IN_PROGRESS | 2017-12-16T19:30:51.075593 | None         |
+--------------------------------------+------------------+--------------------+--------------------------+--------------+
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar/template>
```

Des de Horizon podem veure el resultat:

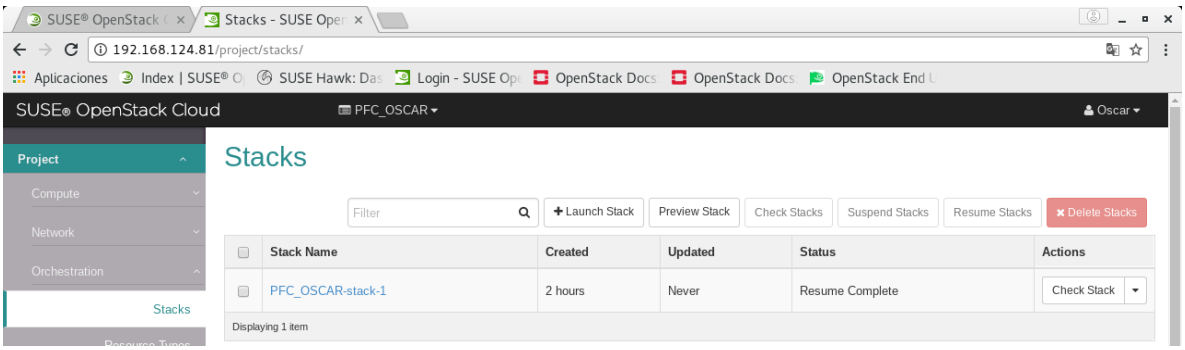

Figura 8 - 29 Vista general dels stacks

Utilitzant l'exemple del template anterior, introduirem l'ús de paràmetres:

```
heat_template_version: 2015-10-15
description: Plantilla per desplegar una instancia amb parametres
parameters:
  clau_ssh:
    type: string
    label: clau SSH
    description: Nom de la clau publica SSH
    default: PFC_OSCAR-key02
  imatge:
    type: string
    label: Nom imatge
    description: Imatge de SO
    default: SLES12SP1-x86_64
  sabor:
    type: string
    label: Flavor de la instancia
    description: Sabor instancia
    default: PFC_Oscar
resources:
 OSCAR4:
   type: OS::Nova::Server
   properties:
     key_name: { get_param: clau_ssh }
     image: { get_param: imatge }
     flavor: { get_param: sabor }
     networks:
       - network: PFC_OSCAR_network-01
```

Si executem el stack:

```
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar/template> heat stack-create -f desplegar-instancia-parametres.template PFC_OSCAR-stack-2
+--------------------------------------+------------------+--------------------+--------------------------+--------------+
```

```
| id                                   | stack_name         | stack_status       | creation_time             | updated_time |
+--------------------------------------+--------------------+--------------------+---------------------------+--------------+
|  48da0824-2771-4b0c-a31d-a5d869640886  |  PFC_OSCAR-stack-1  | RESUME_COMPLETE    | 2017-12-16T19:30:51.075593 | None         |
|  25979f7a-2563-4c71-a45a-d1864b56d2d1  |  PFC_OSCAR-stack-2  | CREATE_IN_PROGRESS | 2017-12-16T21:40:31.124079 | None         |
+--------------------------------------+--------------------+--------------------+---------------------------+--------------+
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar/template>
```

El proper template mostra el desplegament de tot el que hem vist en aquest capítol, des de les xarxes, l'enrutador, la ip flotant, els volums persitents i la instancia:

```
heat_template_version: 2015-10-15
description: Plantilla per desplegar tot l entorn
parameters:
  clau_ssh:
    type: string
    label: clau SSH
    description: Nom de la clau publica SSH
    default: PFC_OSCAR-key02
  imatge:
    type: string
    label: Nom imatge
    description: Imatge de SO
    default: SLES12SP1-x86_64
  sabor:
    type: string
    label: Flavor de la instancia
    description: Sabor instancia
    default: PFC_Oscar
  xarxa_externa:
    type: string
    label: Xarxa externa
```

```
    description: Xarxa externa
    default: floating
  OSCAR-stack_subxarxa_CIDR:
    type: string
    label: Subxarxa de l estack
    description: Subxarxa interna de l estack
    default: 10.0.1.0/24
  mida_volum:
    type: number
    label: Mida del volum
    description: Mida del volum pesistent del servidor
    default: 1
resources:
  OSCAR5:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: clau_ssh }
      image: { get_param: imatge }
      flavor: { get_param: sabor }
      networks:
        - port: { get_resource: OSCAR5_port }
  OSCAR5_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_resource: OSCAR-stack_xarxa }
      security_groups:
        - { get_resource: OSCAR-stack_ssh_secgroup }
      fixed_ips:
        - subnet_id: { get_resource: OSCAR-stack_subxarxa }
  OSCAR-stack_xarxa:
    type: OS::Neutron::Net
  OSCAR-stack_subxarxa:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: OSCAR-stack_xarxa }
```

```
    cidr: { get_param: OSCAR-stack_subxarxa_CIDR }
  OSCAR-stack_router:
    type: OS::Neutron::Router
    properties:
      external_gateway_info: { network: { get_param: xarxa_externa } }
  OSCAR-stack_router_interface:
    type: OS::Neutron::RouterInterface
    properties:
      router_id: { get_resource: OSCAR-stack_router }
      subnet: { get_resource: OSCAR-stack_subxarxa }
  OSCAR-stack_ssh_secgroup:
    type: OS::Neutron::SecurityGroup
    properties:
      description: Security Group for SSH Traffic
      rules:
        - remote_ip_prefix: 0.0.0.0/0
          protocol: tcp
          port_range_min: 22
          port_range_max: 22
        - remote_ip_prefix: 0.0.0.0/0
          protocol: icmp
  OSCAR5_floating_ip:
    type: OS::Neutron::FloatingIP
    properties:
      floating_network: { get_param: xarxa_externa }
  association:
    type: OS::Neutron::FloatingIPAssociation
    properties:
      floatingip_id: { get_resource: OSCAR5_floating_ip }
      port_id: { get_resource: OSCAR5_port }
  OSCAR5_vol_id:
    type: OS::Cinder::Volume
    properties:
      size: { get_param: mida_volum }
  OSCAR5_volum:
```

```
    type: OS::Cinder::VolumeAttachment
    properties:
      instance_uuid: { get_resource: OSCAR5 }
      volume_id: { get_resource: OSCAR5_vol_id }
      mountpoint: /dev/vdb
```

Un cop executat el stack:

```
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar/template> heat stack-create -f desplegar-tot.template PFC_OSCAR-stack-3
+--------------------------------------+------------------+--------------------+--------------------------+----------+
| id                                   | stack_name       | stack_status       | creation_time            | updated_time |
+--------------------------------------+------------------+--------------------+--------------------------+----------+
| 808c993e-50ff-47eb-9f48-e3827788b14a | PFC_OSCAR-stack-3 | CREATE_IN_PROGRESS | 2017-12-16T22:59:42.916160 | None     |
+--------------------------------------+------------------+--------------------+--------------------------+----------+
tux@linux: [Oscar@Default/PFC_OSCAR (v3)]~/Oscar/template>
```

Podem veure que està funcionant al nostre entorn:

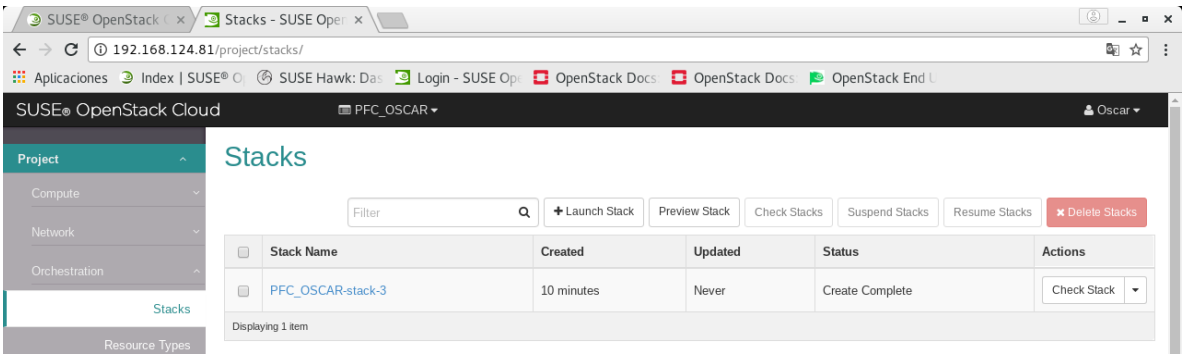

Figura 8 - 30 Vista general dels stacks II

I si accedim al stack, podem veure de manera gràfica tot el desplegament:

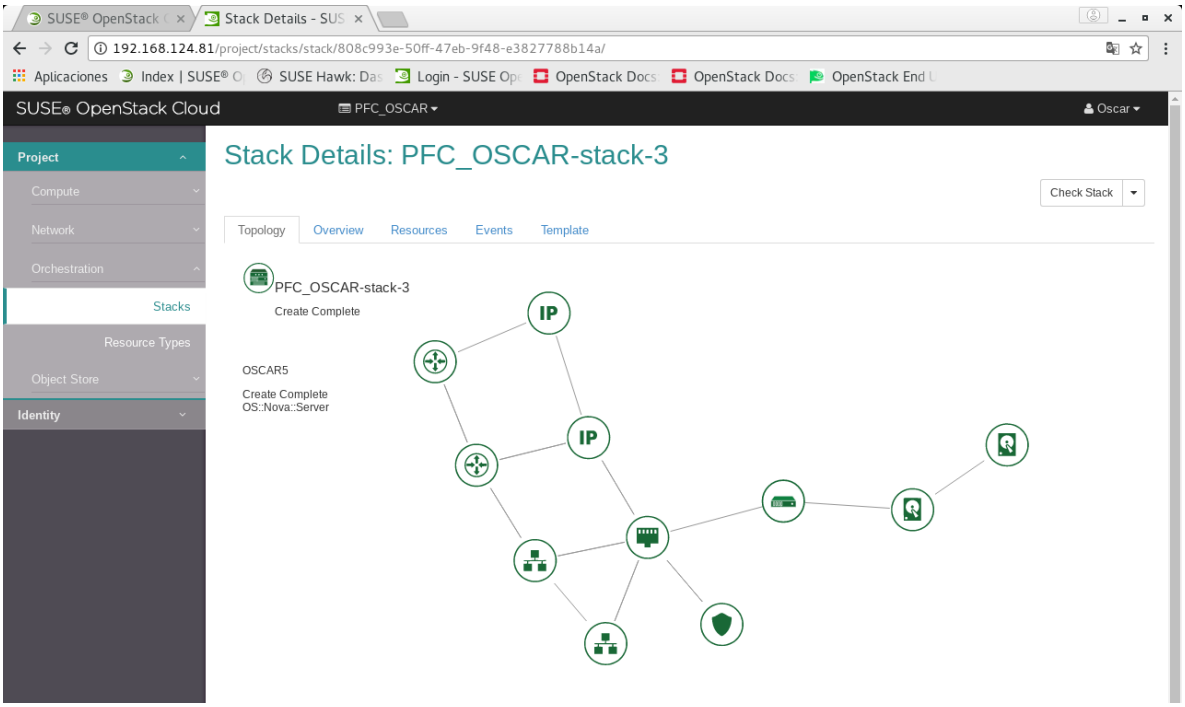

Figura 8 - 31 Detall gràfic del stack amb els seus components

Hem vist que hem creat els paràmetres amb valor per defecte dins del template, però podem assignar-los per la línia de comanda:

```
heat stack-create -P "OSCAR-stack_subxarxa_CIDR=10.0.2.0/24;mida_volum=2" \
-f desplegar-tot.template \
PFC_OSCAR-stack-4
```

D'aquesta manera podem variar el desplegament de manera molt àgil i fàcil. Des de la topologia de xarxa podem veure els dos stacks executant-se:

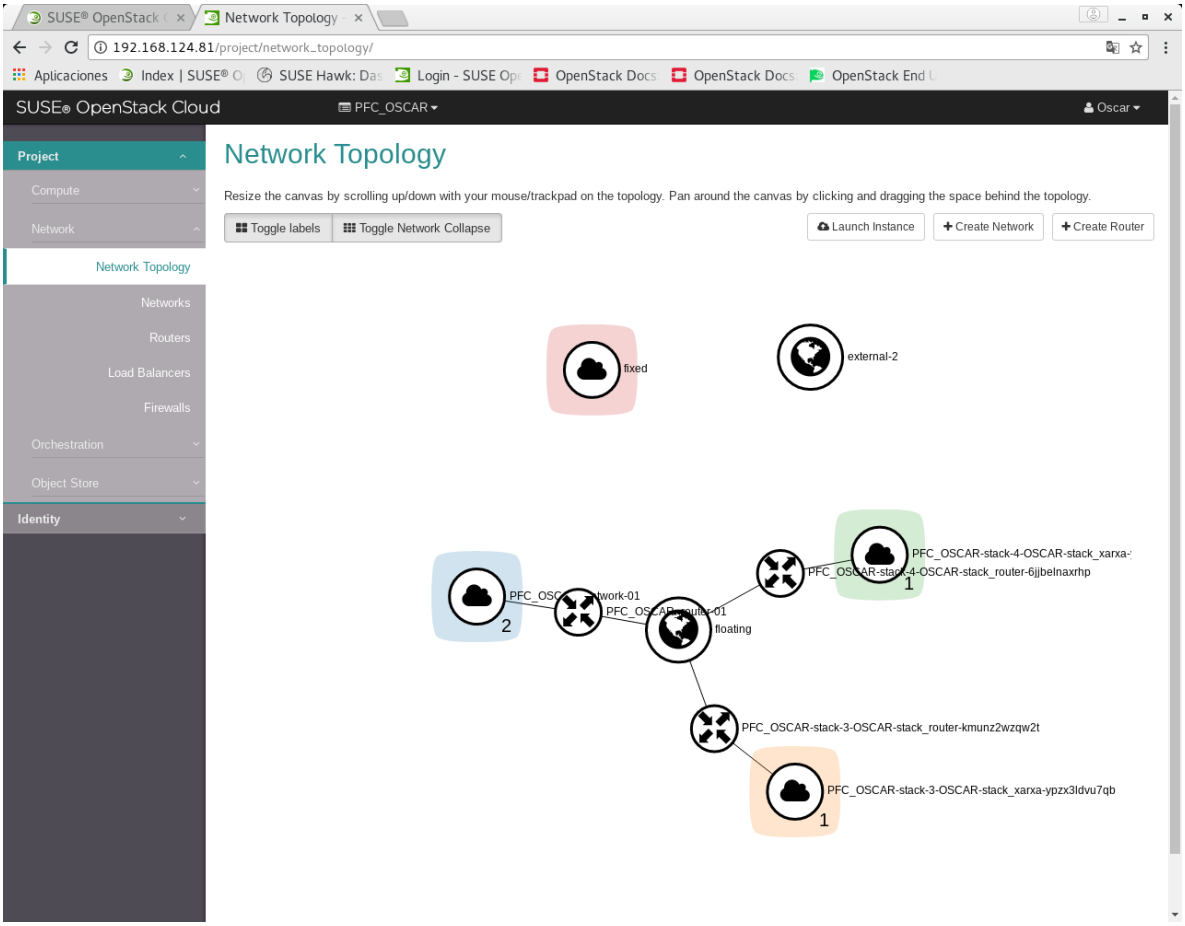

Figura 8 - 32 Topologia de xarxa del projecte

Mesures d'ús

En tot sistema és molt important tenir mètriques d'ús, ja sigui, per exemple per poder avaluar els recursos que s'utilitzen al sistema, per poder fer estimacions de creixement o per facturar el consum al nostre client.

Amb OpenStack això és possible amb el component Ceilometer. L'administrador en qualsevol moment pot veure per períodes de temps el consum que ha tingut qualsevol component del sistema i l'ús que n'ha fet cada usuari o projecte.

Malgrat incorpora unes gràfiques molt bàsiques, el que realment és interessant la possibilitat de poder descarregar aquestes dades i poder-les tractar amb algun altre programari per poder explotar la informació.

Mitjançant les següents comandes podrem extreure informació del nostre sistema:

ceilometer statistics --meter memory

On obtindrem mesures de memòria

D'igual forma des de Horizon podem veure aquests gràfics:

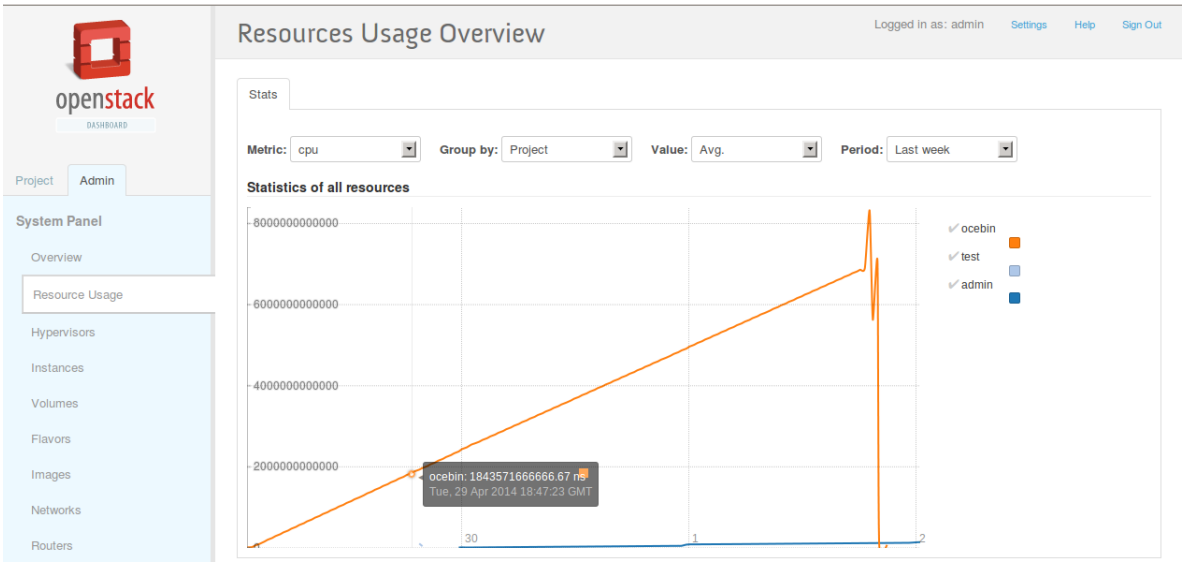

Figura 8 - 33 Detall d´ús de CPU

Conclusió

L'objectiu principal del projecte ha estat la implementació i la gestió d´un clúster d'OpenStack i veure en quins entorns tenia sentit implementar-lo.

OpenStack és la plataforma Cloud Computing de software lliure que més ha crescut en els darrers anys. Està orientada a oferir núvols privats i públics amb infraestructures com a serveis (IaaS). Grans companyies hi estan invertint molts diners, com IBM, Dell, Red Hat o Mirantis i això fa que esdevingui possiblement la plataforma més important de software lliure.

Un dels avantatges més importants en OpenStack en comparació amb altres plataformes existents és que amb OpenStack s'ofereixen serveis i no màquines virtuals, això vol dir que permet escalar els recursos en funció de les necessitats.

Un dels serveis més potents i inclou moltes opcions d'implementació és Neutron. Permet encapsular sobre les seves xarxes internes tota la lògica per tal de comunicar les xarxes que es defineixen a l'entorn i com queden aïllades entre elles.

Un dels principals problemes observats a OpenStack és la gran quantitat d'actualitzacions que apareixen i que dificulten el seu manteniment. El fet de que sigui un producte nou i dinàmic fa que sigui difícil estabilitzar-lo en un entorn de productiu. Per exemple, la versió d'abril del 2016 avui dia ja està a End of Live (EOL) i la d'agost 2017 ho estarà al setembre del 2019[39].

Una de les dificultats que he trobat ha estat entendre les diferencies entre les distribucions d'OpenStack pel que fa al llicenciament i costos. Públicament està poc clar quin és el cost real de la posta en marxa d'OpenStack en funció dels nodes i de les capacitats.
En un entorn real caldria aprofundir més en aquest aspecte ja que hi poden haver diferències importants entre els fabricants.

[39] https://releases.openstack.org/

He triat la distribució de SUSE per dos motius: juntament amb Ubuntu son les dues úniques distribucions que permeten la certificació Certified OpenStack Administrator (COA) i també per la facilitat en la instal·lació i administració d'OpenStack.

La part de clusteritzar els serveis, possiblement degut a la potència de l'equip on he desplegat l'entorn, m'ha portat molts maldecaps, ja que el clúster s'ha degradat força vegades, havent de repetir el desplegament molts cops.

Pel que fa a la implementació i la gestió d'Openstack, he pogut aprendre l'arquitectura de la plataforma i veure quines opcions de disseny permet. El més complicat ha estat entendre amb profunditat els components i la seva interrelació.

Annex 1

En aquest apartat detallo pas a pas el procés de desplegament de l'OpenStack. Indicarem la instal·lació del node d'administració que és el punt de partida per instal·lar tot el clúster.

1. Accedim a la url https://www.suse.com/es-es/ i ens autentiquem:

2. Anem a Descarregues Gratuïtes i descarreguem el paquet SUSE OpenStack Cloud, tal i com es pot veure en la imatge següent:

Figura A1 - 1 Descarregar OpenStack

3. Descarreguem el primer paquet, que és la ISO dels binaris:

Figura A1 - 2 ISO amb els binaris

4. Instal·lació del SUSE Enterprise Linux:

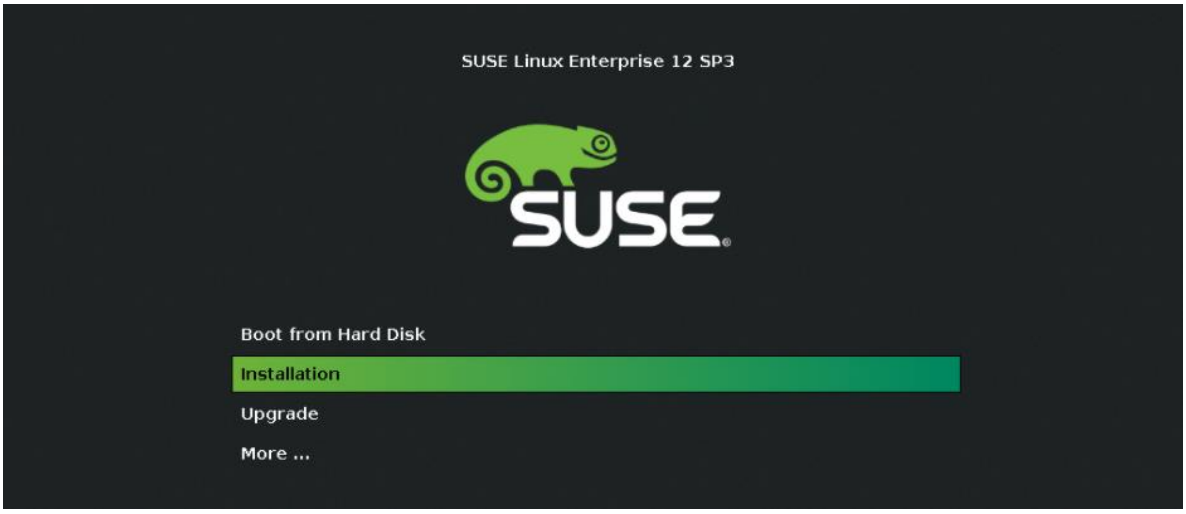

Figura A1 - 3 Instal.lació de SUSE Linux

5. Triem idioma:

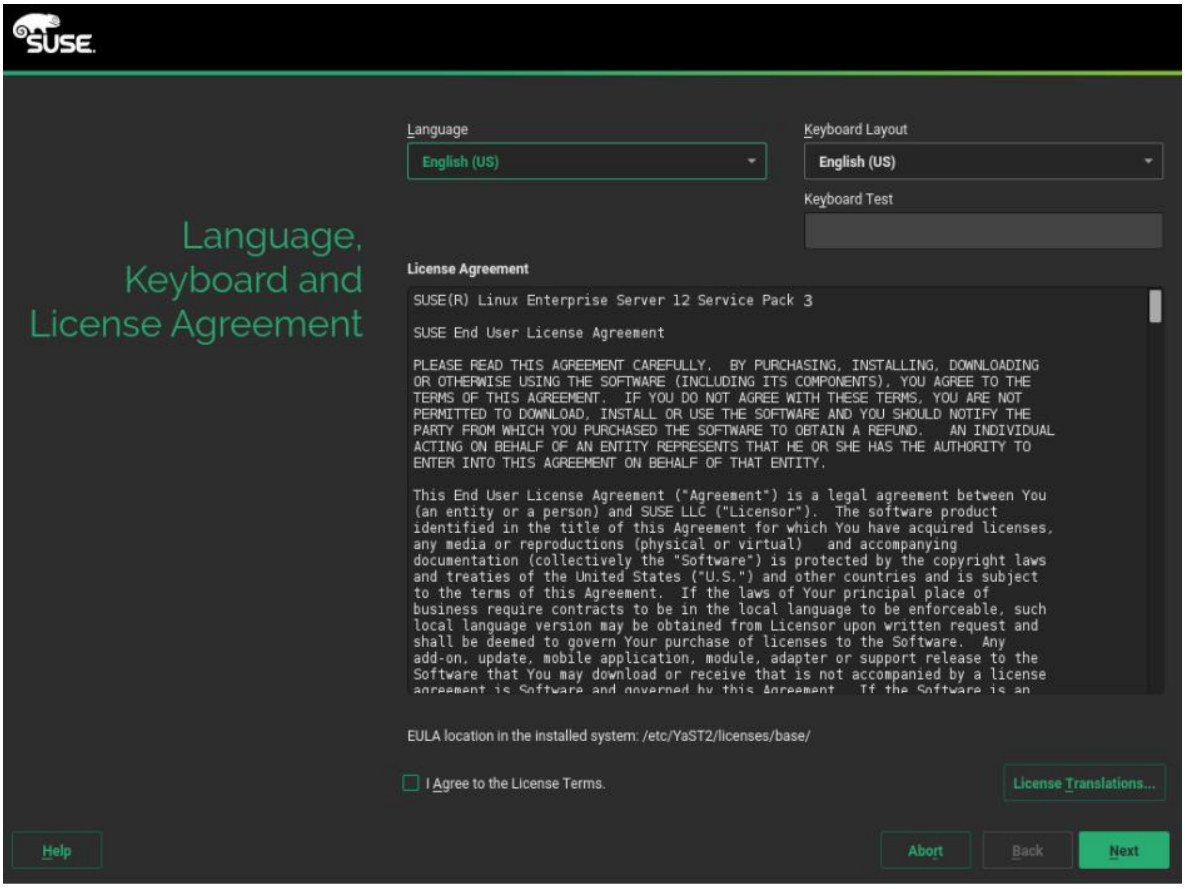

Figura A1 - 4 Idioma, teclat i acord de llicència

6. Seguim la instal·lació per defecte fins a la pantalla final de la instal·lació:

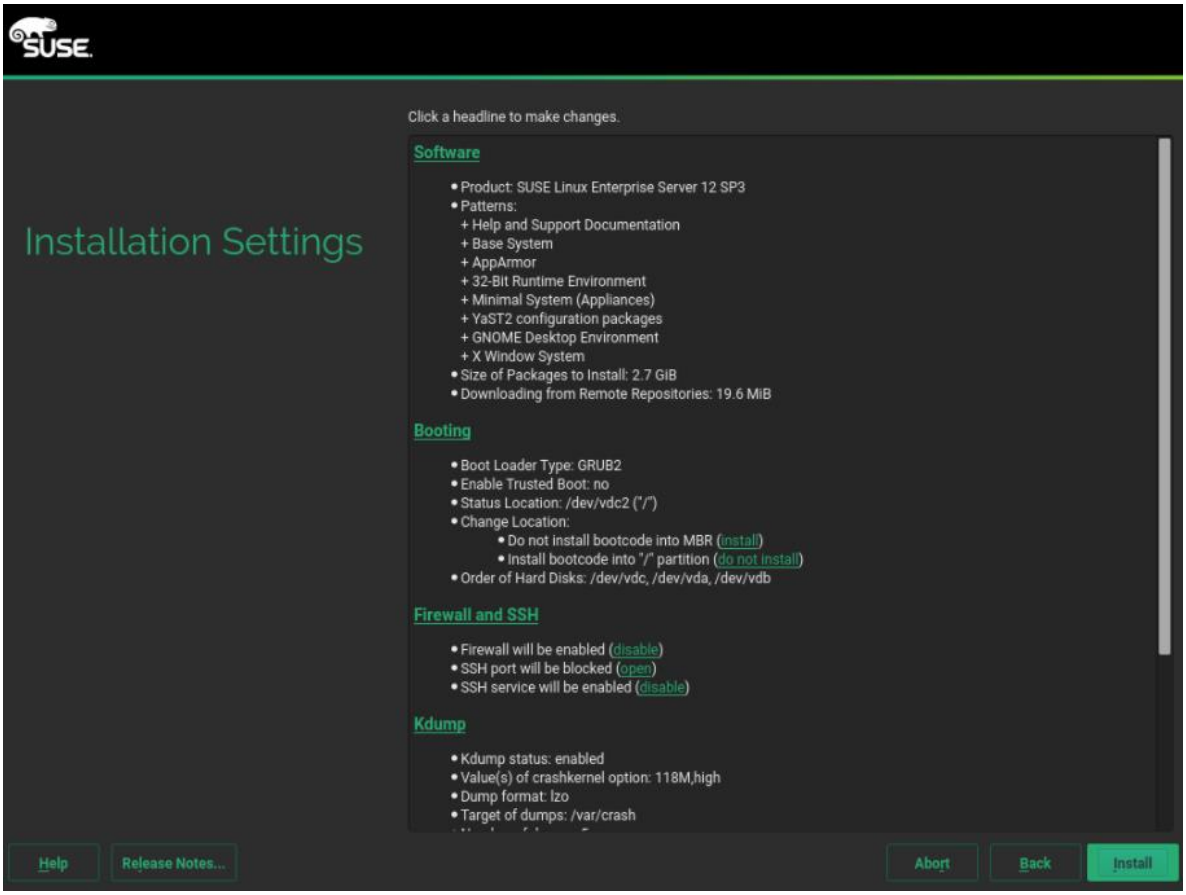

Figura A1 - 5 Resum de la instal.lació

Un cop acabada la instal·lació ens autentiquem al sistema i instal·lem l'extensió d'OpenStack Cloud:

1. Des de la línia de comandes, iniciem YaST i a Software / Add System Extensions or Modules, seleccionem la de Public Cloud Module:

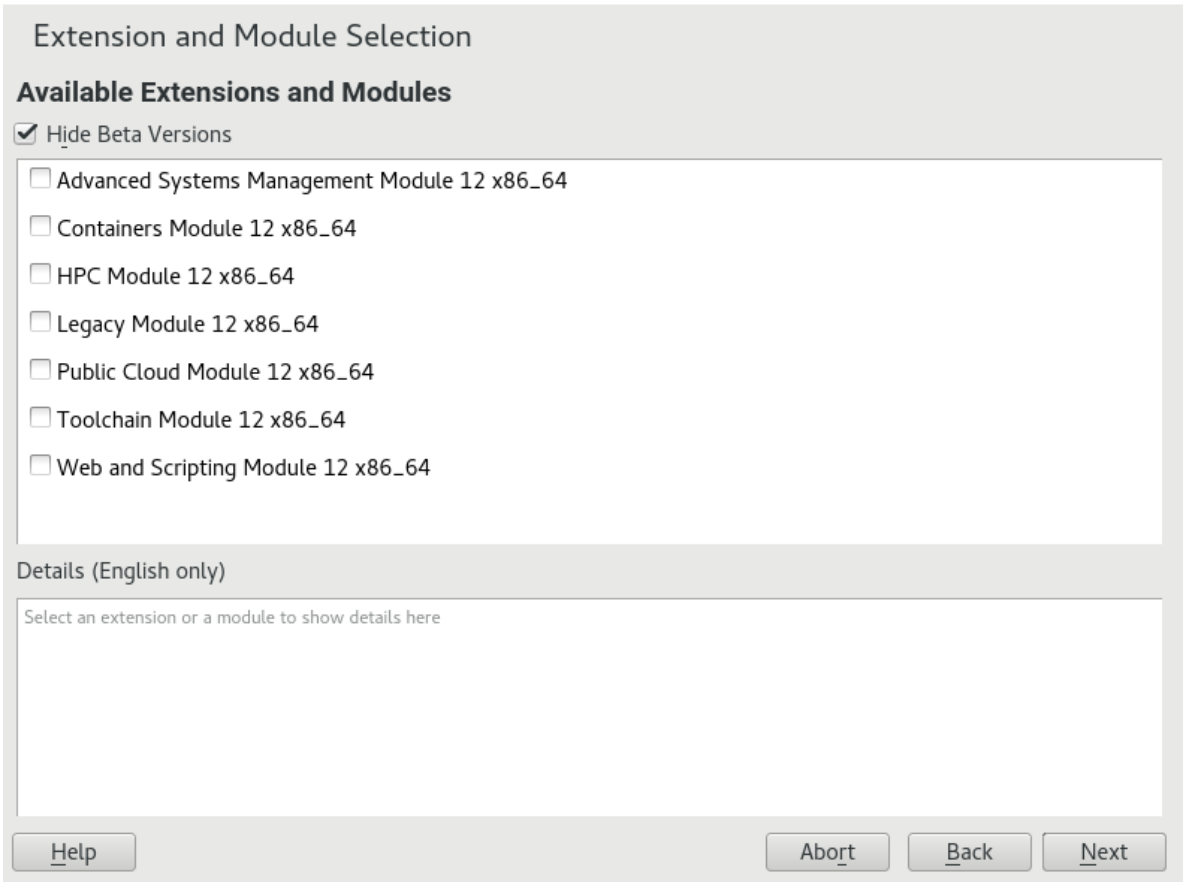

Figura A1 - 6 Instal.lació de la extensió d'OpenStack

2. Copiem els repositoris en local de la següent manera, d'aquesta manera ja no dependrem de les ISOs per fer els desplegaments següents:

SUSE Linux Enterprise Server 12 SP2 DVD#1

```
mkdir -p /srv/tftpboot/suse-12.2/x86_64/install
mount /dev/dvd /mnt
rsync -avP /mnt/ /srv/tftpboot/suse-12.2/x86_64/install/
umount /mnt
```

SUSE OpenStack Cloud 7 DVD#1

```
mkdir -p /srv/tftpboot/suse-12.2/x86_64/repos/Cloud
mount /dev/dvd /mnt
rsync -avP /mnt/ /srv/tftpboot/suse-12.2/x86_64/repos/Cloud/
umount /mnt
```

3. Acabem la instal·lació del Crowbar amb la comanda:

```
yast crowbar
```

4. Apareix la pantalla següent per la configuració de les xarxes:

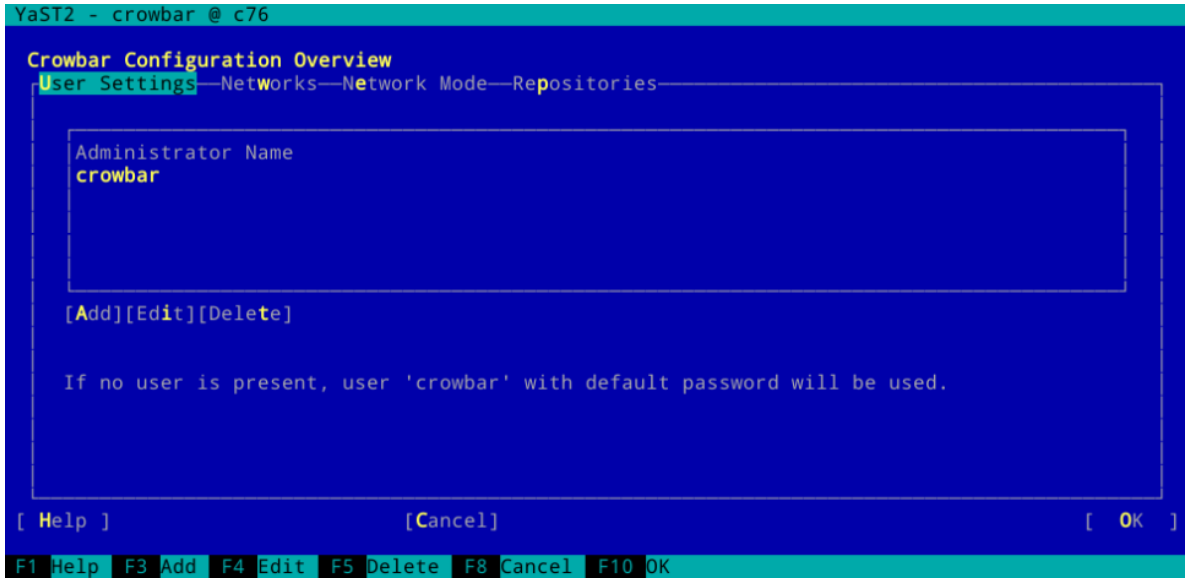

Figura A1 - 7 Configuració d'OpenStack: general

5. Aquesta és la pantalla de la configuració de la xarxa:

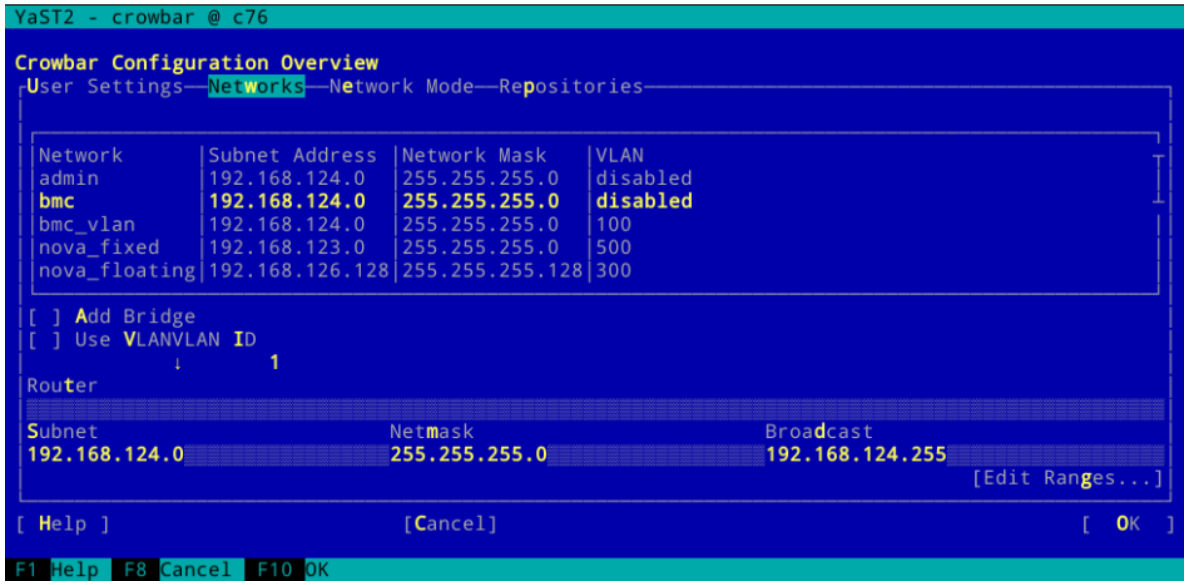

Figura A1 - 8 Configuració d'OpenStack: xarxes

El primer que cal tenir en compte és el disseny de les xarxes que utilitzarem pel laboratori. Després de consultar diferents whitepapers[40] [41] [42] i les guies d'instal·lació, triarem la configuració estàndard que utilitza SUSE.

Tal i com es mostra l'esquema següent:

La xarxa d'administració serà la 192.168.124.0/24.

El node d'administració tindrà la ip 192.168.124.10/24. En el diagrama apareix de color negre.

Disposarem de tres VLANs:

VLAN200: 192.168.1.125/0 estarà dedicada a la xarxa d'emmagatzematge, ja sigui pels nodes de Ceph o Swift i no és accessible pels usuaris del nostre cloud. És la que apareix en color taronja.

VLAN300: 192.168.126.0/24 per la xarxa nova-floating que tindrà un pool d'adreces públiques per a ser assignades a les càrregues de treball. Apareixen en color blau.

VLAN400: 192.168.130.0/24 per la xarxa de SDN de Neutron, aquesta xarxa és privada i l'utilitza Neutron per configurar els diferents elements en els nodes (túnels, Firewall, switchos, ...). No està representada, ja que està per sobre de totes i "corre" per sota de totes les altres.

VLAN500: 192.168.123.0/24 per la xarxa nova-fixed, per assignar directament a les màquines virtuals. Apareix en color vermell.

40 https://www.suse.com/docrep/documents/qo3ntby5nx/sgi_and_suse_OpenStack_cloud_reference_architecture_wp.pdf

41 https://www.suse.com/docrep/documents/r0ujxti4h7/suse_OpenStack_cloud_reference_architecture_with_dell_hardware_white_paper.pdf

42 https://www.suse.com/docrep/documents/j2nd2zd2zg/HP_ProLiant_for_Private_Cloud.pdf

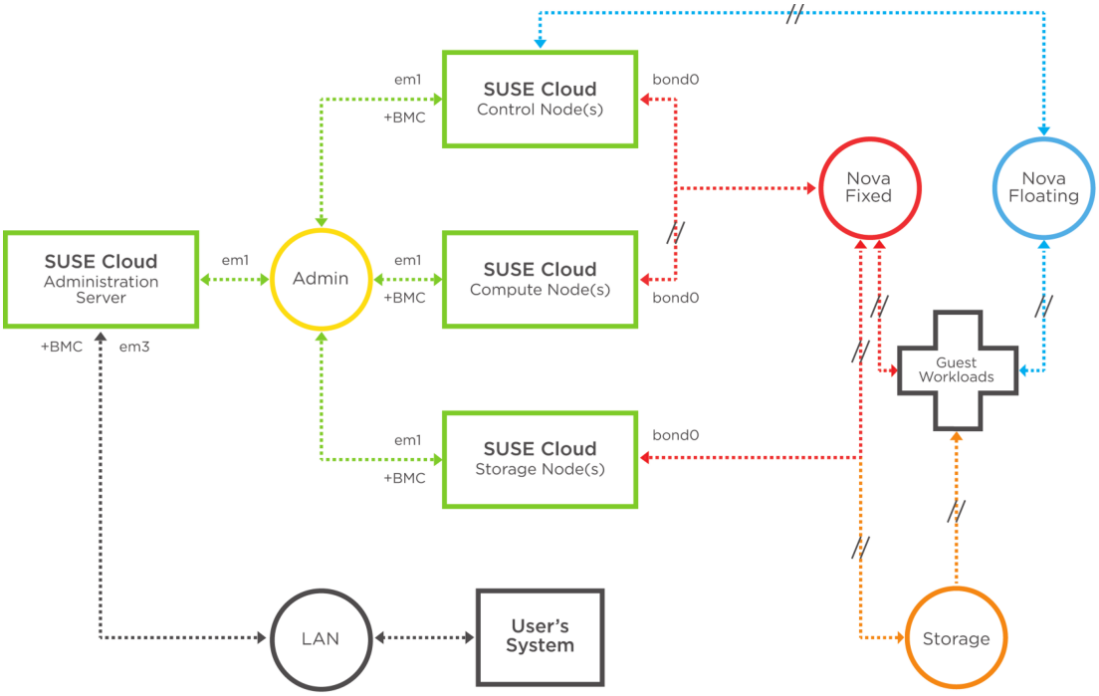

Figura A1 - 9 Esquema de xarxes

6. El darrer pas és configurar els repositoris prèviament descarregats

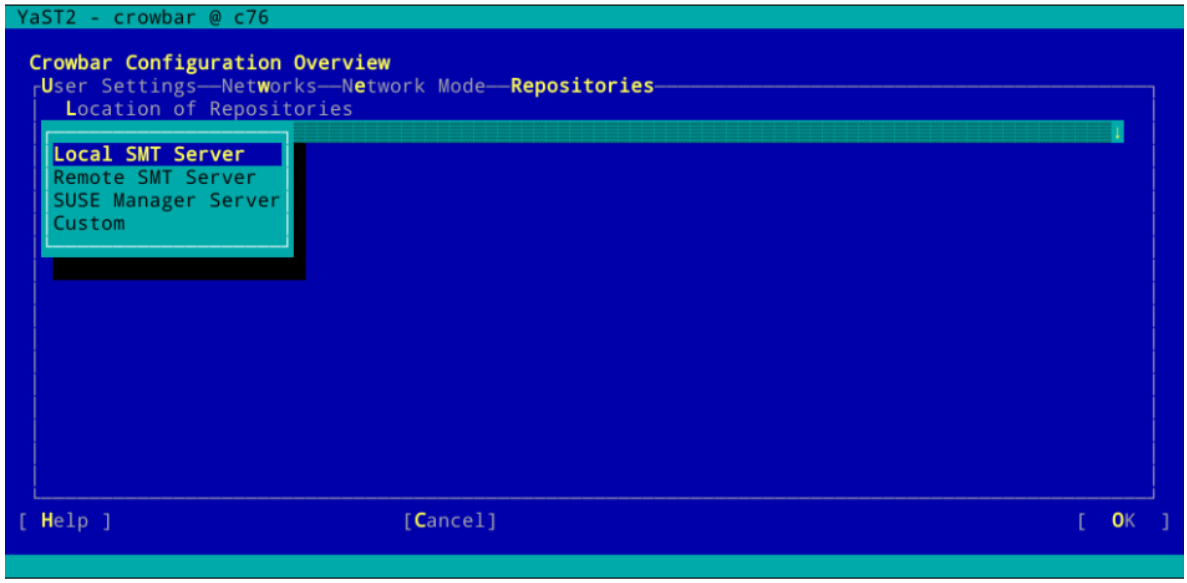

Figura A1 - 10 Configuració d'OpenStack: repositoris

7. Un cop ja hem acabat la part de configuració, executarem la comanda:

install-suse-cloud

Que instal·larà tot el node d'administració segons els paràmetres que hem introduït.

8. Finalment verifiquem la instal·lació accedint via web a la ip del node d'administració i ens ha d'aparèixer en verd la pantalla següent:

Figura A1 - 11 Dashboard de crowbar

Annex 2

En aquest annex, detallo com administrar a partir del node d'administració el nostre clúster d'OpenStack.

Per tenir alta disponibilitat configurarem els nodes amb Pacemaker[43] per crear el clúster i instal·larem Hawk[44] per disposar d'una consola web per monitoritzar el nostre clúster.

1. Accedim des de la consola de crowbar als barclamps disponibles
2. Editem el barclamp de Pacemaker

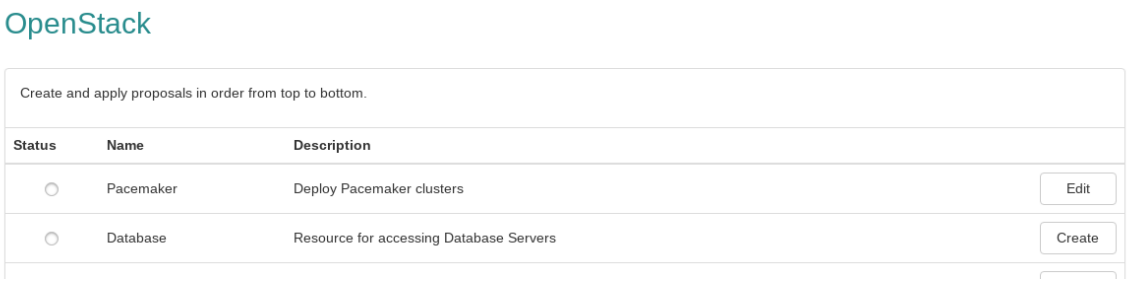

Figura A2 - 1 Llista de barclamps

3. Al crear el clúster aprofitarem i afegirem com a nodes remots els de còmput. Amb aquesta configuració dotarem als nostres nodes de còmput d'alta disponibilitat sense haver de tenir el tercer node de quòrum (típic d'un clúster) amb la pertinent pèrdua de recursos pel còmput.

[43] https://clusterlabs.org/pacemaker.html

[44] https://hawk-ui.github.io/

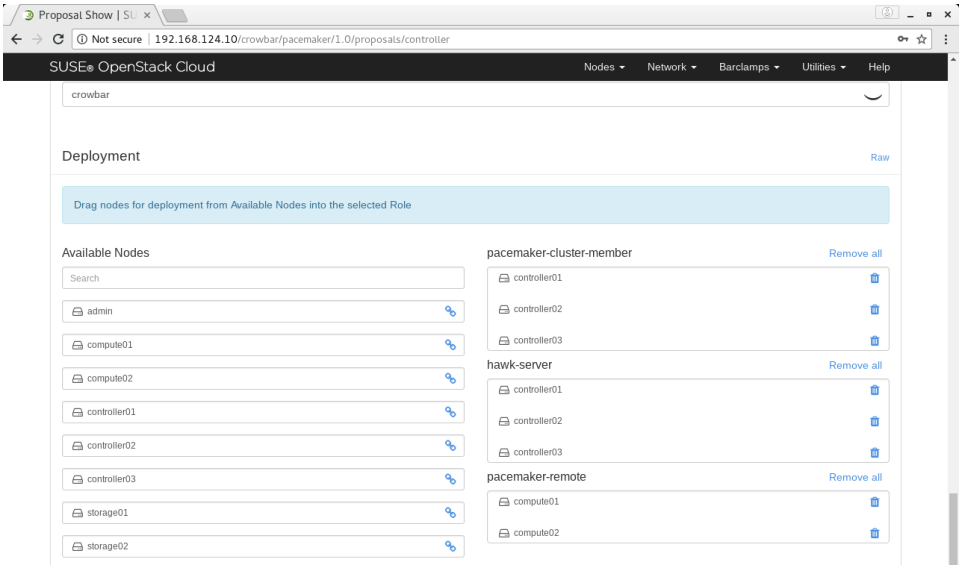

Figura A2 - 2 Detall de configuració del barclamp de pacemaker

4. Un cop creat el clúster i els seus serveis funcionant, començarem a clusteritzar els serveis d'OpenStack.

5. Primer de tot desplegarem sobre els nodes de control els serveis de base de dades (Database que és PostgreSQL) i de cues (RabbitMQ).

   Crowbar a part de tenir un frontend web, també disposa de línia de comandes. El servei de base de dades el desplegarem d'aquesta manera:

   Ens connectarem a un controlador i crearem un filesystem compartit on emmagatzemarem la nostra base de dades que serà accessible per tots els nodes de control:

```
root@admin:~ # ssh root@controller01
Last login: Sat Dec  9 08:54:28 2017 from admin.example.com
root@d52-54-00-63-a1-01:~ # ls /dev/disk/by-id | grep db
scsi-0QEMU_QEMU_HARDDISK_qemu-db
scsi-SQEMU_QEMU_HARDDISK_qemu-db
root@d52-54-00-63-a1-01:~ # mkfs.xfs /dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_qemu-db
meta-data=/dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_qemu-db isize=256    agcount=4, agsize=65536 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                       bsize=4096   blocks=262144, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
root@d52-54-00-63-a1-01:~ #
```

Figura A2 - 3 Creació del filesystem compartit per la base de dades

Des del node d'administració cridarem la comanda de crowbar:
crowbarctl proposal create database default

i ens obrirà un editor on configurarem les opcions de "storage" per dir-li el filesystem que he, creat i el nom del clúster que volem el servei.

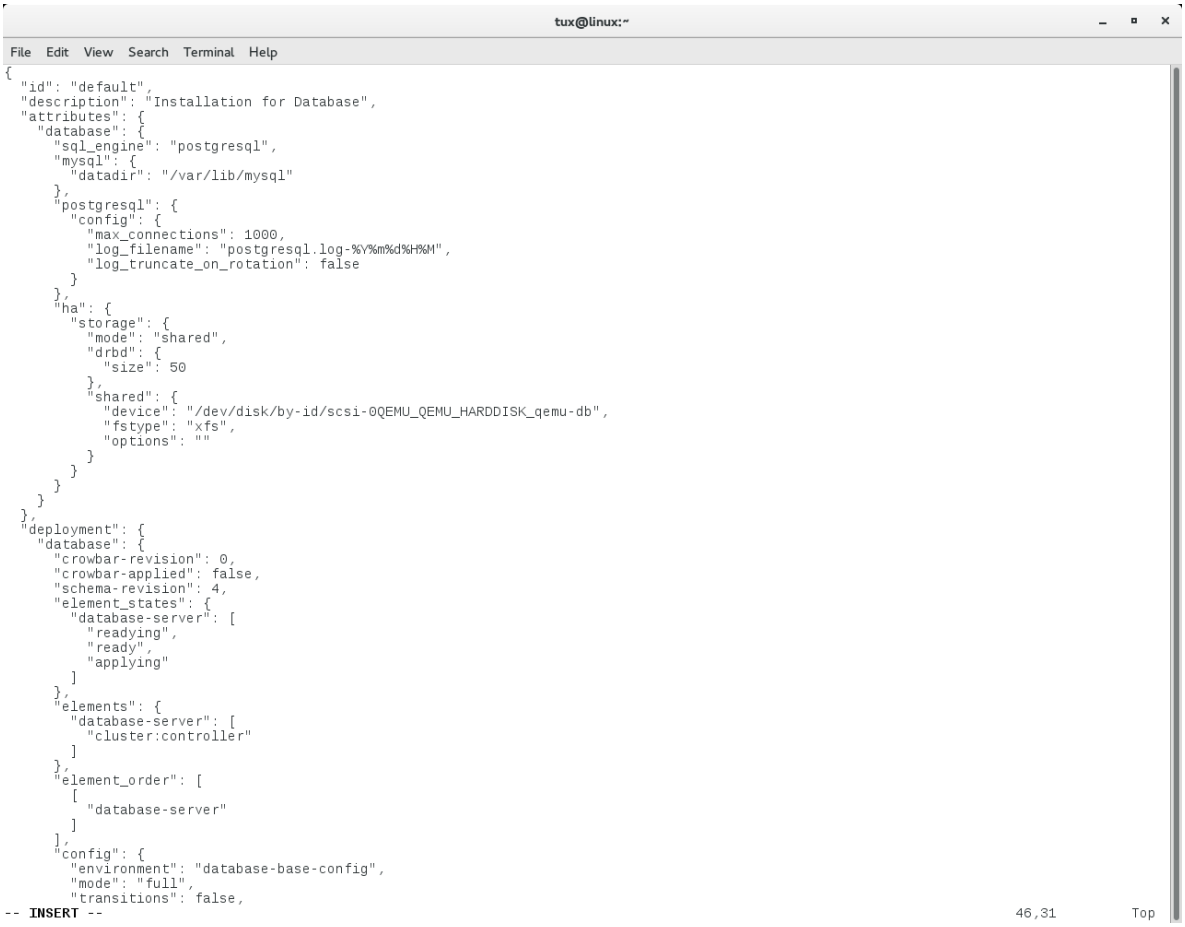

Figura A2 - 4 Detall de la recepta de Chef de la creació de la base de dades

Un cop hem modificat les opcions fem la comanda:
crowbarctl proposal commit database default

Que ens desplegarà el servei de base de dades al nostre clúster.

6. A l'igual que hem fet amb la base de dades ho fem amb el servei de cues RabbitMQ que es configura en alta disponibilitat i de manera similar.

7. El següent Barclamp que desplegarem és el servei d'identitats (Keystone), i anirà clusteritzat com els anteriors. Al ser el servei d'identitats, l'API pot utilitzar per la seva comunicació SSL, però al nostre laboratori per evitar problemes amb els certificats autosignats, ho farem per HTTP.

8. Fins aquí si accedim a la consola de Hawk, veurem que tenim els serveis funcionant i l'estat en que estan al clúster:

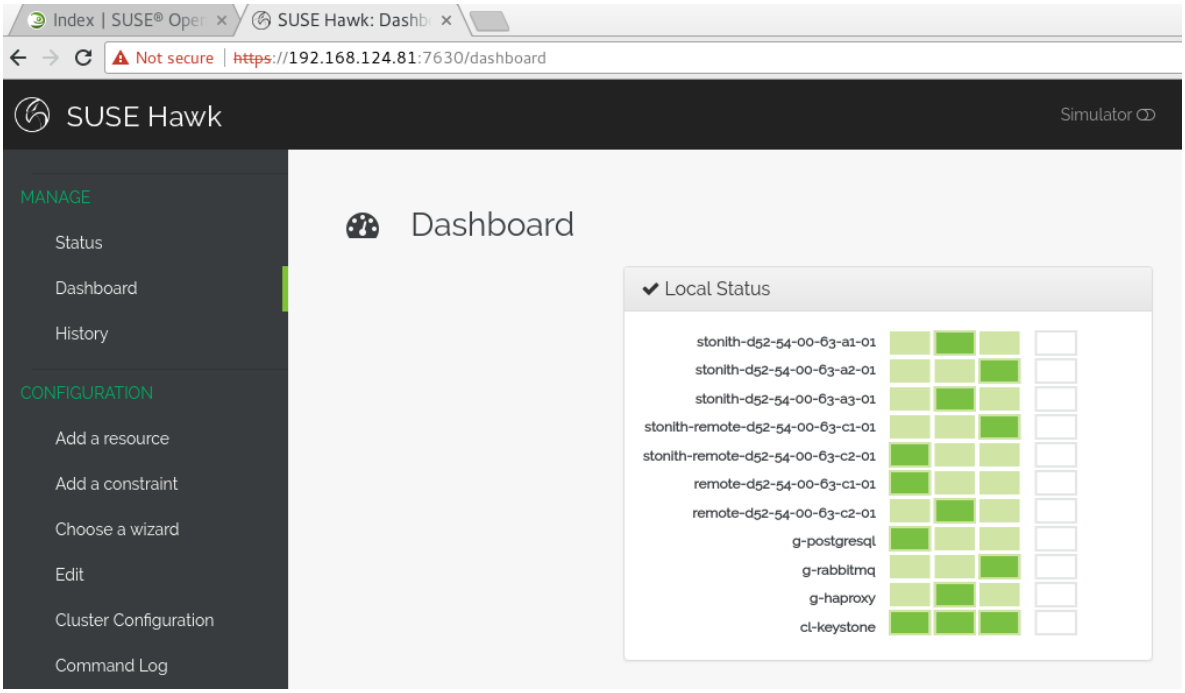

Figura A2 - 5 Dashboard de Hawk

9. Emmagatzematge

En aquest punt desplegarem la part d'emmagatzematge i ho farem amb Ceph i el desplegarem amb la següent estructura:

ceph-mon: que és un servei en clúster encarregat de monitoritzar el sistema de fitxers distribuït de ceph. Aquest servei correrà distribuït entre els tres primers nodes d'emmagatzematge del nostre laboratori.

ceph-osd: és el servei encarregat de l'emmagatzematge d'objectes de ceph. Aquest servei correrà distribuït entre els tres primers nodes d'emmagatzematge del nostre laboratori.

ceph-radosgw: és el servei encarregat de donar respostes a les peticions de la API REST del servei d'emmagatzematge. Aquest malgrat també pot estar clusteritzat i accedit mitjançant balancejadors web, al nostre laboratori serà un servei que residirà dins del node quatre d'emmagatzematge.

Ceph-calamari: és el servei encarregat de la monitorització i gestió de ceph, aquest servei no ens cal, ja que el gestionarem integrat dins d'OpenStack.

El seu desplegament és com en la resta de barclamps, el creem, modifiquem les dades per configurar-lo i el despleguem:

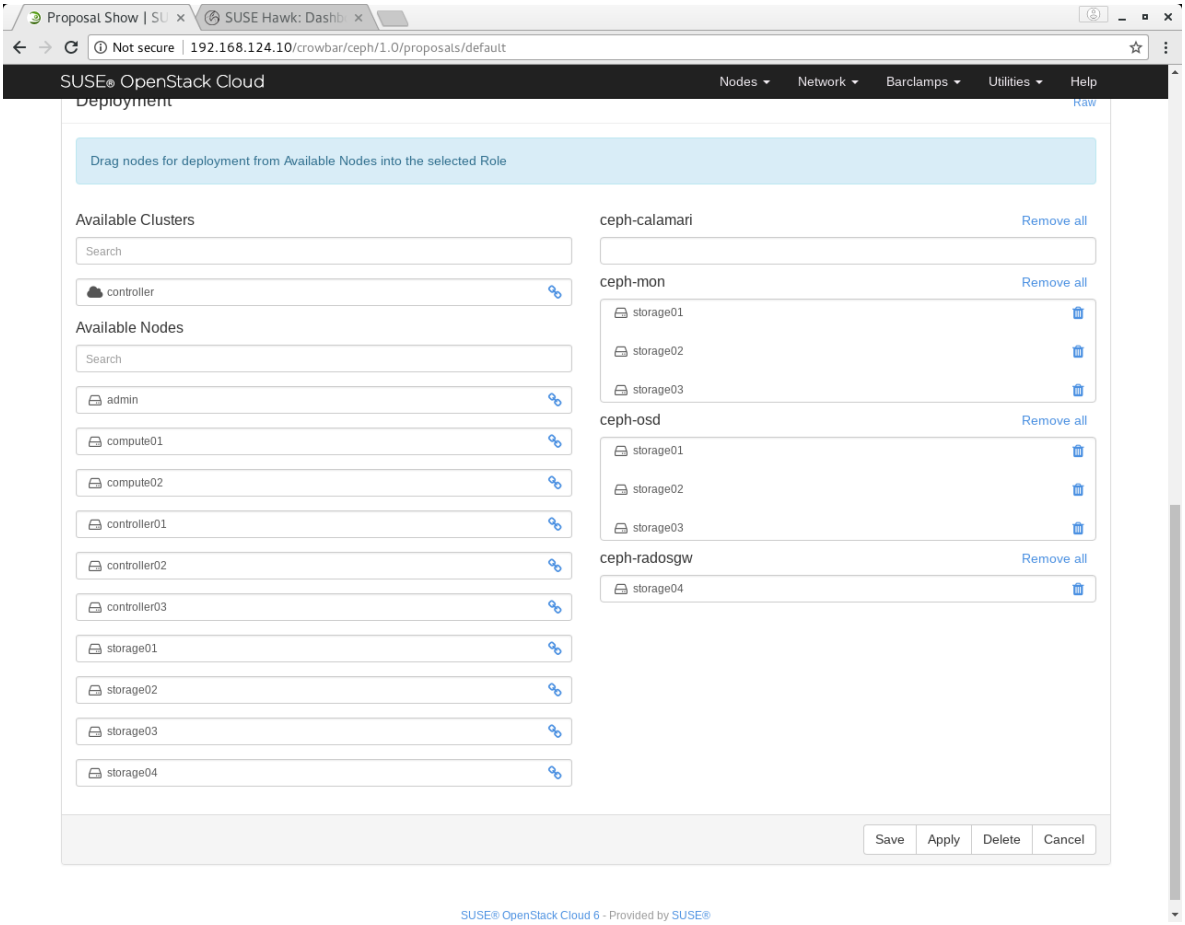

Figura A2 - 6 Detall de configuració del barclamp de Ceph

Un cop tenim l'emmagatzematge disponible, ja podem desplegar el servei d'imatges (glance) que residirà sobre el clúster controller i emmagatzemarà les imatges a ceph mitjançant el servei de bloc (Rados), en aquest cas ho farem per comandes des del node d'administració:

crowbarctl proposal create glance default

Modificarem els paràmetres i llençarem el desplegament:

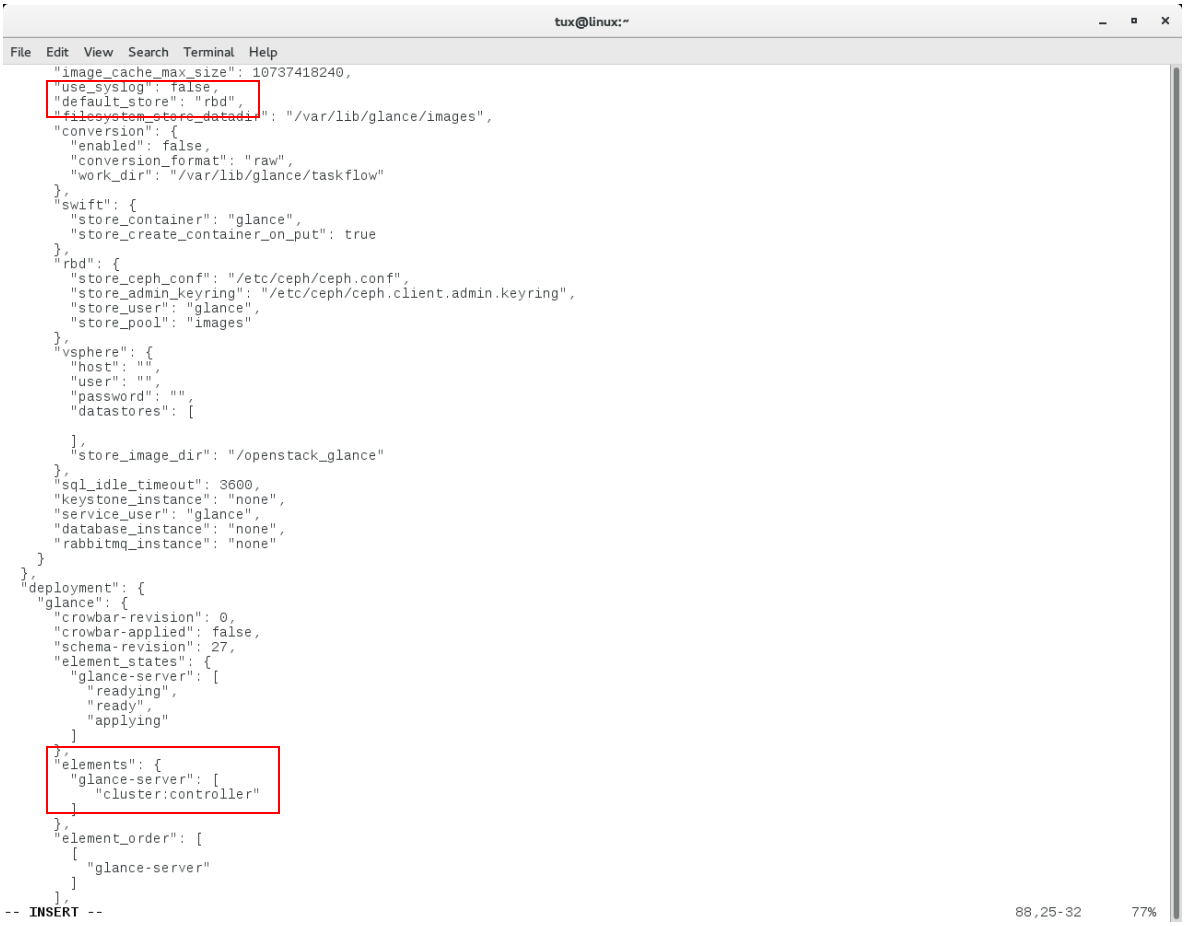

Figura A2 - 7 Detall de la recepta de Chef pel desplegament de Glance

root@admin:~ # crowbarctl proposal commit glance default

Successfully queued default proposal

root@admin:~ #

Aquesta comanda encua la petició a RabbitMQ i es processa, per veure l'estat podem o ve consultar la interfície web o amb la comanda

root@admin:~ # watch crowbarctl node status

Veurem com evoluciona el desplegament en els diferents nodes:

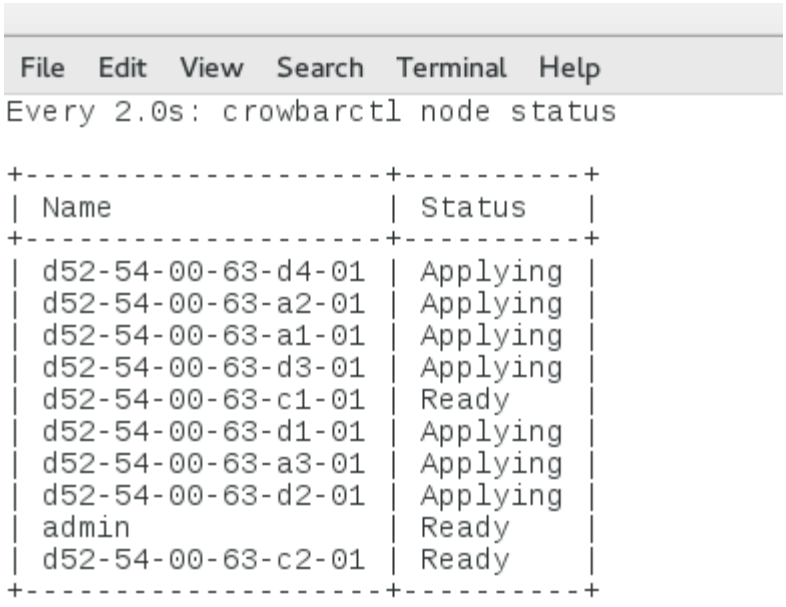

Figura A2 - 8 Estat de crowbar des de la línia de comandes

El darrer barclamp d'emmagatzematge per proveir servei de bloc és Cinder que el farem córrer sobre el servei Rados de Ceph (que és el servei nadiu de Ceph per bloc)

El servei el distribuirem de la següent manera:

cinder-controller: l'encarregar d'orquestrar l'emmagatzematge de bloc, el farem residir sobre el clúster dels controladors

cinde-volume: és on pròpiament guardarem les dades i en aquest cas utilitzarem els quatre nodes d'emmagatzematge.

Un cop hem desplegat aquest darrer servei d'emmagatzematge, si fem una ullada sobre la consola de Hawk veiem el següent:

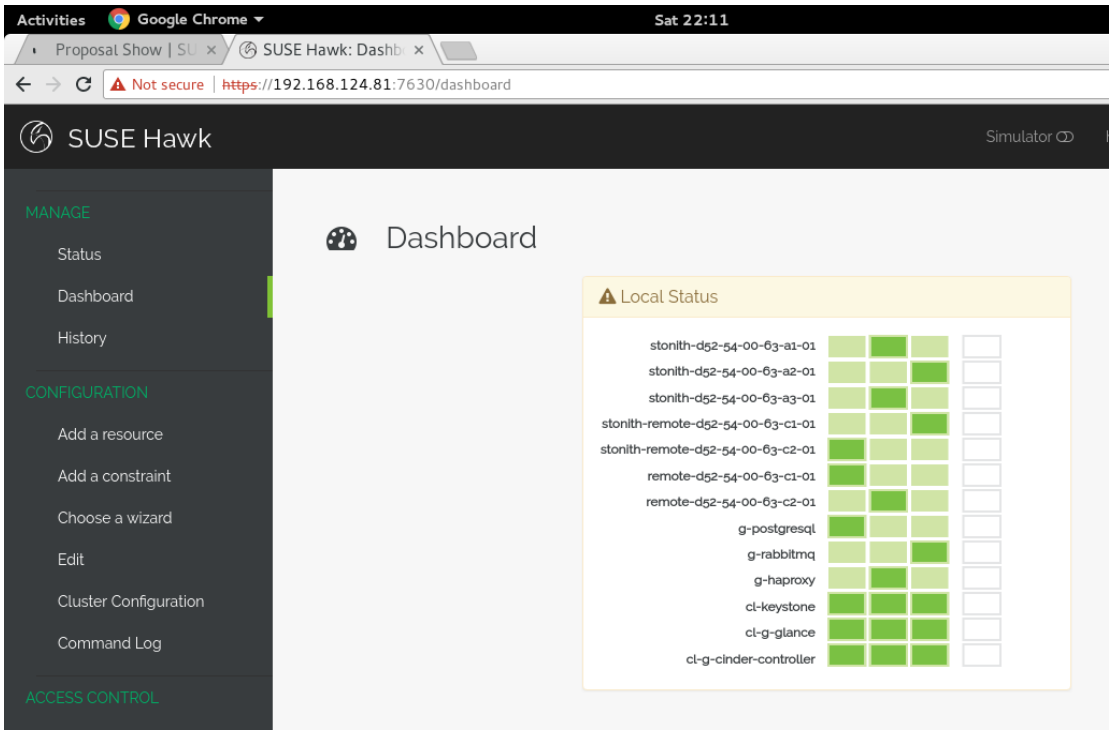

Figura A2 - 9 Dashboard de Hawk

Podem observar que les tres columnes verdes fan referència als tres nodes controladors que tenim al laboratori i el verd més fort indica que el servei està corrent en aquell node. Si les tres columnes del servei estan en verd fort (Keystone, Glance i Cinder-controller) indica que el servei és actiu en els tres nodes.

10. La xarxa

Un cop tenim els serveis d'administració bàsics (Database, RabbitMQ, Keystone) i els d'emmagatzematge sobre Ceph (Glance i Cinder) desplegarem la part de xarxa amb Neutron que ens permetrà "posar les carreteres" per comunicar les càrregues de treball amb qualsevol altra component d'OpenStack.

El primer que farem és desplegar el servei amb les següents característiques:

Configurarem el barclamp amb la openvswitch[45] que és un projecte que s'integra a opesntack i ens permet mitjançant túnels GRE comunicar els diferents serveis i dotar de connectivitat els nodes de còmput amb les diferents xarxes del nostre cloud. Si observem el gràfic següent veurem la seva arquitectura:

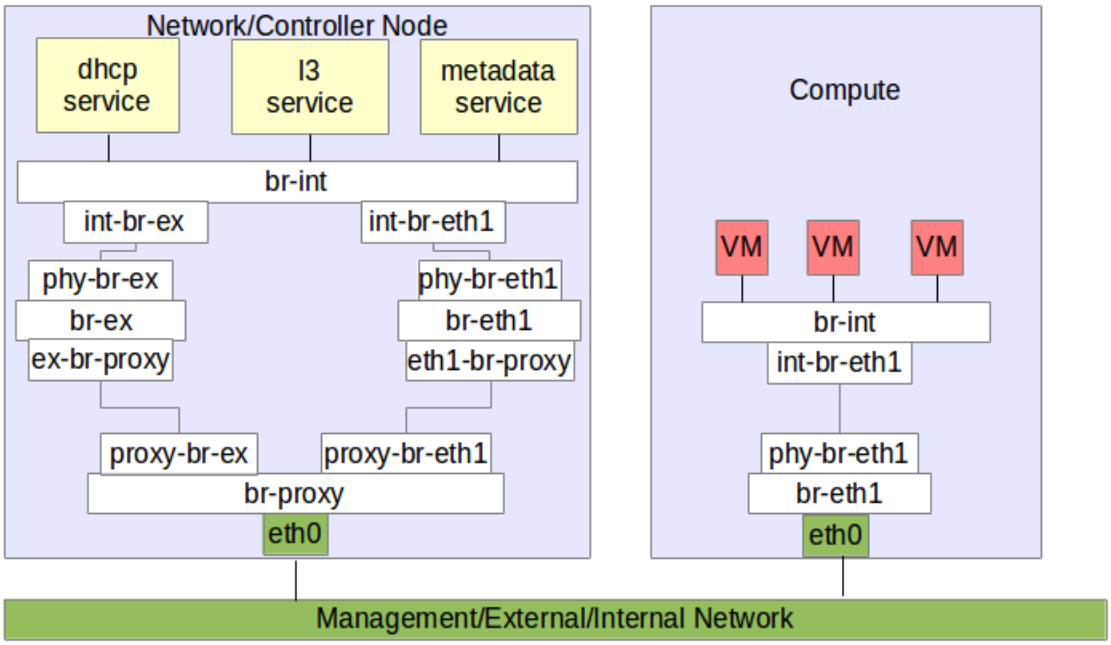

Figura A2 - 10 Arquitectura de Neutron

En instal·lacions molt grans es poden tenir nodes/clústers dedicats a Neutron. En el nostre cas, al clúster d'administració hi residirà la part de control i gestió de openvswitch on tindrem l'enrutament de les xarxes i el servei de DHCP i en els nodes de còmput hi residirà un agent que tancarà els túnels GRE entre les màquines virtuals i el controladors.

La configuració del barclamp serà la següent:

[45] http://openvswitch.org/

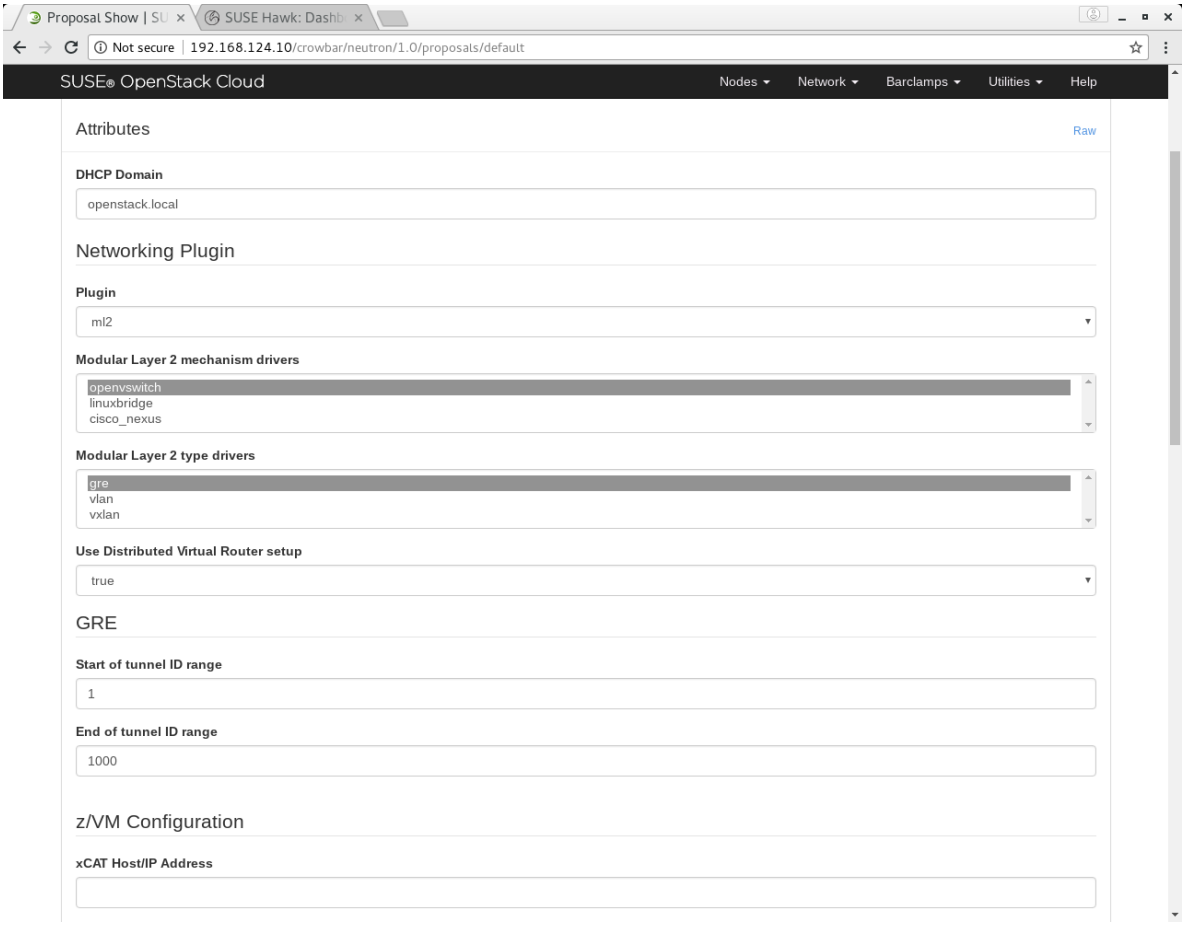

Figura A2 - 11 Detall de la configuració del barclamp de Neutron

11. La computació

Una de les darreres peces del nostre laboratori serà el servei de computació Nova que en el nostre cas estarà composat de dos nodes.

Com ja hem vist, podem triar entre diferents hipervisors, en el nostre cas utilitzarem KVM. El barclamp configurarà dins de cada node remot de computació una instal·lació de KVM i l'agent de Neutron. En les darreres versions d'OpenStack, també suporta l'ús de contenidors Docker.

Un cop acabat el desplegament si anem a Hawk veurem el següent:

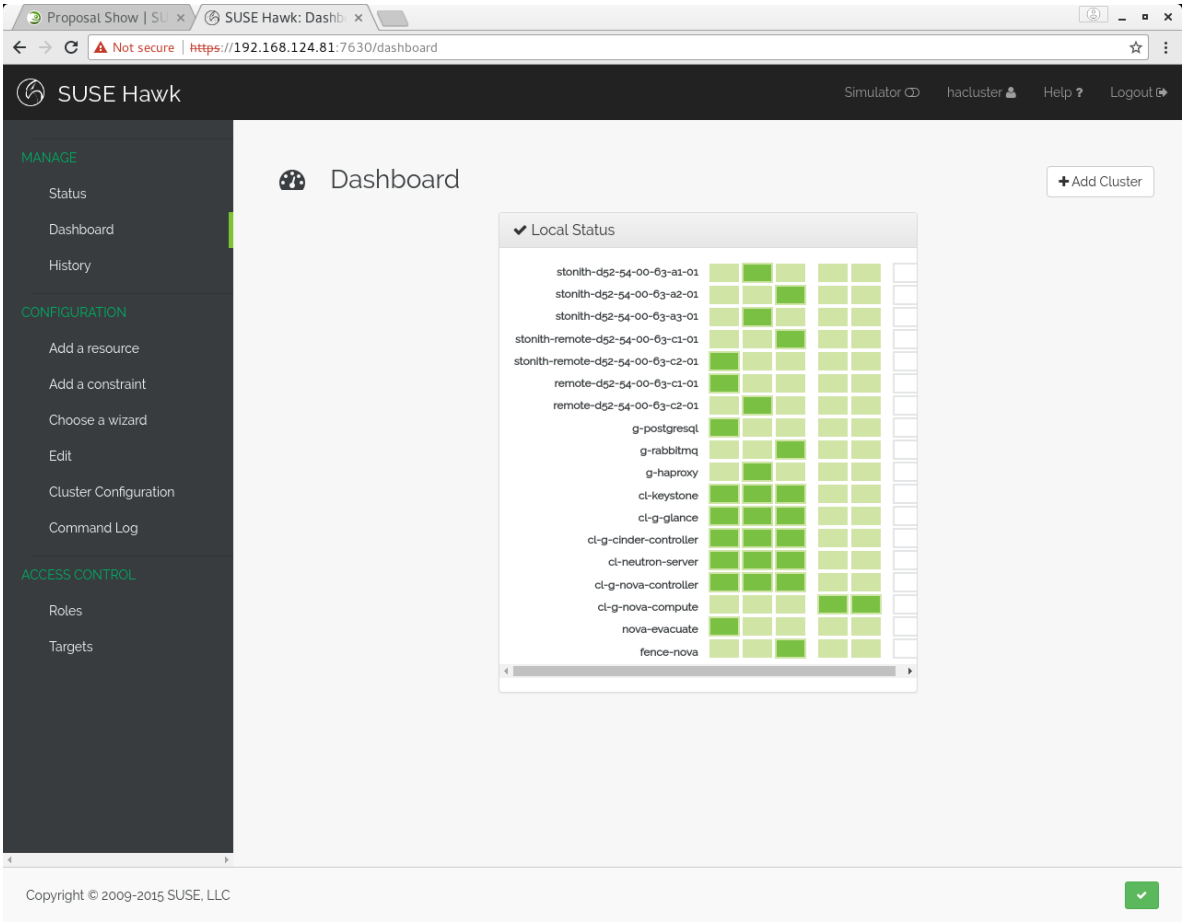

Figura A2 - 12 Dashboard de Hawk

Podem observar que en les dues darreres columnes verdes tenim el clúster de còmput i que tenim els serveis de Nova funcionant.

12. Dashboard

En aquest punt ho tenim gairebé tot, però ens falta saber com poder accedir a l'entorn per gestionar-lo. L'encarregat d'aquesta tasca és Horizon.

La configuració del barclamp és molt senzilla, ja que únicament li direm que el volem sobre el clúster de control i iniciarem la seva instal·lació.

Un cop acabat ja podrem accedir al nostre entorn per tal de poder començar a utilitzar-lo. Per fer-ho únicament ens caldrà saber l'adreça ip que ens ha assignat el sistema al clúster de Horizon.

Des del node d'administració ens connectarem a un controlador i farem les següents comandes:

```
root@admin:~ # ssh root@controller01
Last login: Sat Dec  9 12:31:54 2017 from admin.example.com
root@d52-54-00-63-a1-01:~ # crm configure show | grep -A 1 "primitive vip-admin-cluster"
primitive vip-admin-cluster-controller IPaddr2 \
        params ip=192.168.124.92 \
root@d52-54-00-63-a1-01:~ #
```

Després podrem accedir: http://192.168.124.92

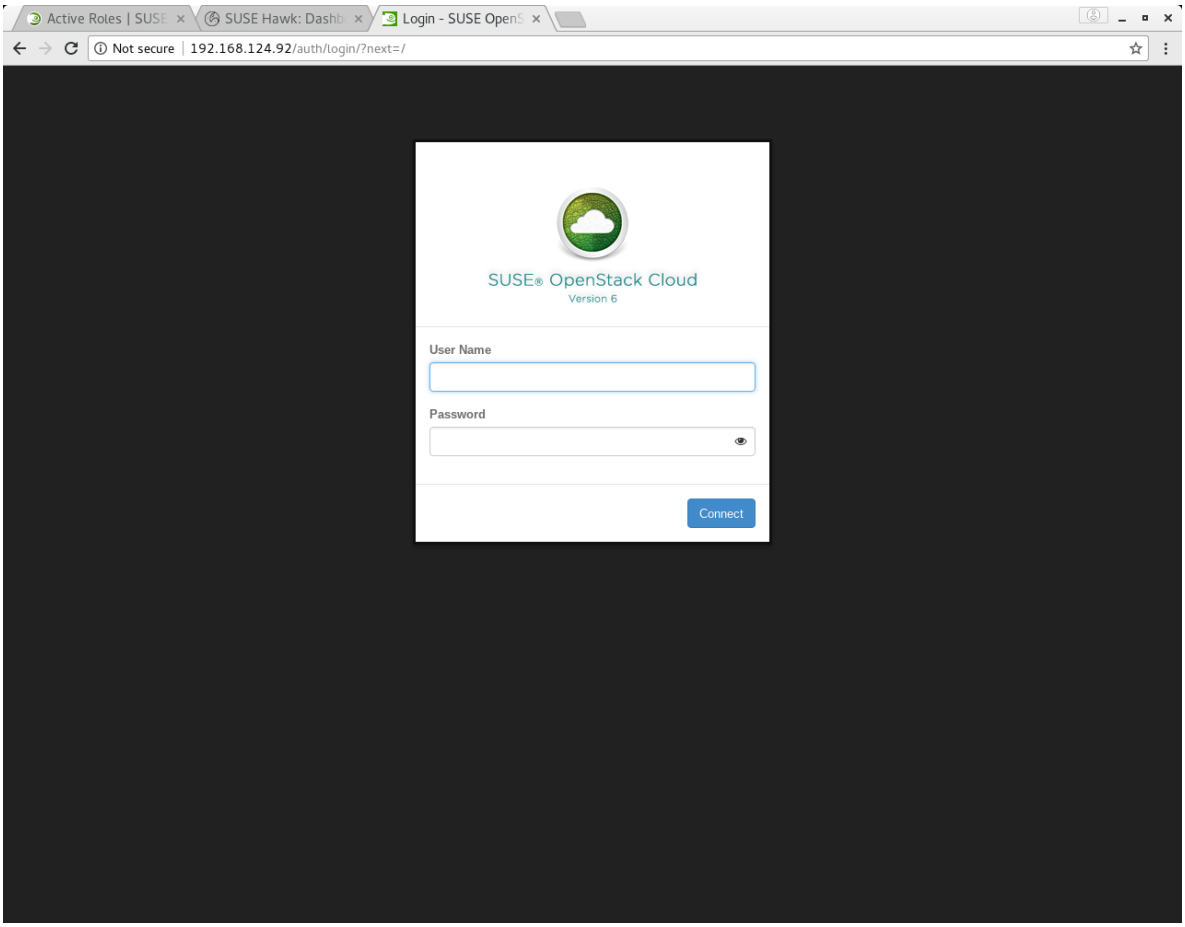

Figura A2 - 13 Pantalla de login d'OpenStack

13. Mètriques i orquestració

Abans de començar instal·larem els dos darrers components del nostre laboratori.

Ceilometer ens permetrà tenir mètriques del nostre entorn i poder avaluar-ne tant l'ús per poder prendre decisions com si ens cal escalar algun dels components.

Heat que és el component d'OpenStack encarregat de l'orquestració i que mitjançant plantilles podrem automatitzar el desplegament de qualsevol recurs del nostre núvol (xarxes, enrutadors, volums, instàncies, creació d'imatges, ...) i crear relacions entre ells.

La sintaxi de les plantilles és Heat Orchestation Template - HOT[46] que fa servir com a referència el llenguatge YAML[47].

El desplegament d'aquests dos component és molt similar als anteriors, ja que també estan en alta disponibilitat i únicament ens caldrà indicar que volem que s'executi sobre el clúster controller.

[46] https://docs.OpenStack.org/heat/pike/template_guide/hot_spec.html

[47] http://yaml.org/