TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació

CREACIÓ AUTOMATITZADA DE HONEYNETS

Memòria

XAVIER ORIOL SELLÉS
TUTOR: LÉONARD JANER

CURS 2017 - 18

TecnoCampus
Mataró-Maresme

Abstract

This project has been made to provide a new tool against the threats of the net, which are becoming more frequents, helping to distract and get information from HoneyNets. The result is an application able to deploy and eliminate networks defined by the user easily and quickly. The installation of services for each host, in addition to a centralized monitoring system, are done automatically by the program.

Resum

Aquest projecte s'ha realitzat per a aportar una nova eina contra les amenaces en la xarxa, cada vegada més freqüents, ajudant en la distracció i recopilació d'informació a través de *HoneyNets*. El resultat és una aplicació capaç de desplegar i eliminar xarxes definides per l'usuari, de forma fàcil i ràpida. La instal·lació dels diferents serveis indicats per cada màquina, així com un sistema de monitorització centralitzat, es realitzen de forma automàtica pel programa.

Resumen

Este proyecto se ha realizado para aportar una nueva herramienta contra las amenazas en la red, cada vez más frecuentes, ayudando en la distracción y recopilación de información a través de *HoneyNets*. El resultado es una aplicación capaz de desplegar y eliminar redes definidas por el usuario, de forma fácil y rápida. La instalación de los diferentes servicios indicados por cada máquina, así como un sistema de monitorización centralizado, se realizan de forma automática por el programa.

Índex

1. Introducció

Aquest projecte tracta sobre la automatització de la creació de *Honeynets*. Per a això, primer cal tenir clar que és un *Honeypot*.

S'anomena *Honeypot* a una màquina creada expressament a mode d'esquer, per tal d'atraure als atacants cap a ella. El principal propòsit d'aquestes màquines és entretenir a l'atacant mentre es recopila informació sobre la seva activitat i forma d'actuar. Per un cantó, desvies a l'atacant de les màquines que estiguin en producció i, per tant, es disposa de més temps per a detectar-lo i frenar-lo abans de que arribí a causar perjudicis. Per altra banda, també es pot fer servir simplement per recaptar informació sobre els atacants, les formes d'actuar i els nous patrons o eines que utilitzen. Aquesta informació es pot utilitzar després per a crear solucions més elaborades o, inclús, per a ensenyar a sistemes d'intel·ligència artificial.

Ara ja es pot definir una *Honeynet* com un conjunt de *Honeypots* que interactuen entre ells. Aquesta vegada no se simula una sola màquina, sinó tota una xarxa sencera. Això permet fer-ho més creïble i atractiu de cara a un atacant, de la mateixa forma que possibilita obtenir més informació, ja que l'atacant té més espai i eines per a moure's.

La importància d'aquest projecte recau en un context en que cada vegada es realitzen més atacs en la xarxa i de forma més organitzada, això requereix d'informació al respecte per tal de poder-ho confrontar. A més, l'escalabilitat i la gestió de les màquines són aspectes cada vegada més requerits en tots els sistemes actuals.

El resultat d'aquest projecte és una eina que, a partir d'una configuració prèvia proporcionada per consola, ha de ser capaç de virtualitzar tota una xarxa de màquines que actuïn com a *Honeynet*. És a dir, tota aquesta xarxa s'ha de crear ja preparada per a que les diferents màquines interactuïn entre elles i, al mateix temps, s'extregui informació de les activitats i interaccions que succeeixen en cadascuna d'aquestes i en la comunicació entre elles. D'aquesta manera, l'única tasca restant per a crear una *Honeynet* creïble és omplir-la d'informació falsa però també creïble, informació que haurà d'estar relacionada amb el sistema o àmbit que es vol imitar.

En ser un projecte destinat a l'àmbit dels servidors i no dels clients, la importància recau en la seva funcionalitat. Considerant, a més, que la majoria de servidors no disposen d'un entorn gràfic, aquest aspecte no està contemplat dins l'abast del projecte exposat. La entrada de la configuració desitjada ha d'estar preparada per a poder-se realitzar a través de qualsevol consola de comandes.

Per altra banda, la xarxa resultant i les dimensions de la mateixa depenen dels requisits previs configurats i dels recursos dels que disposi l'usuari. La automatització en la creació de *Honeynets* està estretament lligada amb la escalabilitat d'aquesta. A més de la xarxa en sí amb el funcionament indicat anteriorment, el resultat ha de contenir la informació de tota la xarxa, així com l'accés a cadascuna de les màquines que la conformen.

A més, qualsevol xarxa creada amb aquesta eina disposa d'un mecanisme d'aturada, modificació i posada en marxa de la mateixa quan sigui necessari.

2. Marc teòric i anàlisi de referents

En aquest apartat es realitza un estudi de la situació actual i les necessitats d'informació que sorgeixen per tal de realitzar aquest projecte.

2.1 Context

Els atacs en la xarxa és un fet que augmenta cada dia de forma notable. No s'augmenta només en el nombre d'atacs, sinó que també augmenta la organització criminal que hi ha darrera, junt amb el volum de diners i persones que hi contribueixen. A més, la informació sobre les vulnerabilitats i la forma d'atacar, junt amb les eines, es troben actualment a l'abast de qualsevol persona. Això comporta que cada vegada es necessitin més eines i persones per tal de contrarestar-ho, a més de la informació i coneixements al respecte sobre com fer-ho.

Aquest fenomen està recolzat per la quantitat de codi i, en definitiva software, que intervé en el nostre dia a dia, junt amb les vulnerabilitats que això suposa. Cada vegada volem més facilitats, comunicacions, flexibilitat i, en resum, funcionalitats. Això suposa un apropament constant a la xarxa i al món lògic, món amb moltes funcionalitats per oferir però, a la vegada, també perills i responsabilitats que molts cops no es tenen en suficient consideració.

Es pot observar, per exemple, que cada vegada es disposa de menys màquines físiques com a servidors. Inclús aquestes s'han anat transformant en màquines lògiques creades a partir d'un volum gran de recursos. Per anar més lluny, cada vegada es té menys servidors a les empreses, sinó que estan migrant-se al "núvol", a on es connecta de forma completament remota. Això és un exemple de com la lògica (o software) i la xarxa estan guanyant terreny constantment, junt amb les vulnerabilitats que deixen en el seu camí.

Per fer front a aquests perills es necessita, sobretot, informació. Informació per a comprendre els perills en els que s'està exposat i la forma de protegir-se. Per això es necessiten sistemes capaços de recopilar informació constantment, en especial dels nous atacs que sorgeixen dia a dia (anomenats atacs de dia zero fins al moment de fer-se públics).

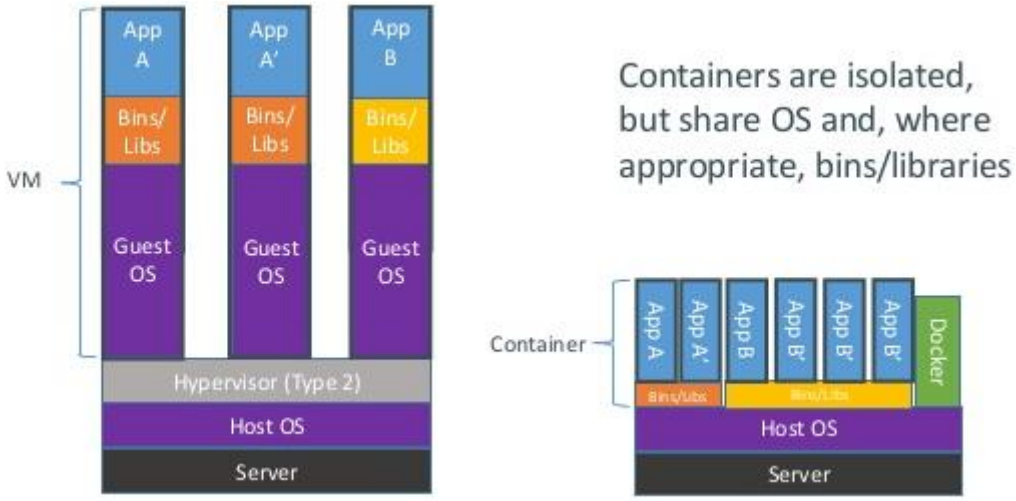
2.2 Antecedents

Pel que fa a la virtualització es poden identificar dos mètodes diferents segons la tecnologia utilitzada: aquelles basades en hipervisors i aquelles basades en contenidors.

En la virtualització mitjançant hipervisors, també anomenada virtualització completa, s'executa tot un sistema operatiu directament per sobre de la màquina amfitriona (anomenats hipervisors natius o *bare metal*), o per sobre d'un altre sistema operatiu (anomenats

hipervisors *hosted*). Amb aquesta tecnologia s'aïlla completament el sistema operatiu i s'emula el hardware que el farà córrer. Alguns exemples de productes són VMware [1], Oracle VirtualBox [2], Microsoft Hyper-V [3] o KVM (*Kernel-based Virtual Machine*) [4]. En la virtualització mitjançant contenidors, també anomenada virtualització a nivell de sistema operatiu, no s'intenta virtualitzar el hardware, sinó que s'aïllen els processos i recursos utilitzats. Això significa que tots els sistemes s'executen sota el mateix *kernel*, però tenen els seus propis sistemes de fitxers, processos, memòria, etc. Cada entorn virtual queda encapsulat en el que s'anomena contenidor. L'exemple principal d'aquesta tecnologia, actualment, és *Docker* [5].

A la Imatge 1 es pot veure de forma gràfica una comparació entre les dues tecnologies.



Imatge 1: Contenidors contra màquines virtuals. [6]

Finalment, i cada vegada més, les màquines s'estan creant en el núvol, d'aquesta forma es guanya en escalabilitat i s'allibera la responsabilitat del manteniment de la infraestructura. El principals proveïdors són Amazon amb *Amazon Web Services* (AWS) [7] i Microsoft amb *Azure* [8], junt amb alternatives enfocades en els desenvolupadors com per exemple *DigitalOcean* [9]. Finalment, també es pot considerar la alternativa *open source*, anomenada *OpenStack* [10]. Aquests, al seu torn, utilitzen alguna de les dues tecnologies anteriors de virtualització.

A nivell de sistema operatiu, i més concretament en l'àmbit de servidors, hi ha moltes distribucions Linux capaces de proporcionar fiabilitat, amb suficient temps i maduresa i que, actualment, estan corrent en entorns importants de producció. Es poden diferenciar les branques de *Debian* [11] *RedHat* [12] i *Suse* [13], les tres amb les seves respectives variants.

Totes aquestes distribucions són compatibles amb els sistemes de virtualització indicats anteriorment.

Per altra banda, també es necessita sistemes de detecció d'intrusions per tal de monitoritzar l'activitat de l'atacant. Aquests es poden diferenciar en dos tipus: sistemes de detecció d'intrusos a nivell de *host* (HIDS) o sistemes de detecció d'intrusos a nivell de xarxa (NIDS).

Els sistemes de detecció d'intrusos a nivell de host (HIDS) són software de monitorització que s'instal·la en una màquina per tal de poder detectar les intrusions en aquesta. Encara que el seu abast sigui una sola màquina, es pot centralitzar la monitorització en una altra independent, i a la primera simplement establir un agent encarregat d'escoltar i transmetre les dades a la segona. Un clar exemple d'aquests sistemes és *Ossec* [14].

Els sistemes de detecció d'intrusos a nivell de xarxa realitzen la mateixa tasca que els anteriors, però aquest cop el seu abast inclou tota una xarxa. Un clar exemple d'aquests sistemes és *Snort* [15].

També es poden trobar eines que, encara que no estiguin categoritzades exclusivament com detecció d'intrusos, també realitzen tasques de detecció i monitorització junt amb altres funcionalitats addicionals. Un exemple d'eina per aquest cas seria *Sysdig* [16].

Tal com es pot veure en els casos anteriors, hi ha antecedents i eines que cobreixen parts concretes necessàries per a la realització d'aquest projecte. A més, tractant-se d'un entorn de servidors on predomina el sistema operatiu Linux, moltes d'aquestes eines són de codi lliure, de tal forma que qualsevol persona les pot fer servir sense inconvenient.

Tot i això, cap d'aquestes eines i entorns cobreix la necessitat. Per un cantó, es té la virtualització, però simplement és la creació de màquines lògiques, en cap cas s'apliquen els mecanismes necessaris en una *Honeynet*. A més, la virtualització té en compte la màquina que crees en aquell moment, no tota una xarxa sencera i la interacció entre aquestes. Per altra banda, existeixen bones eines per a detectar intrusions i monitoritzar, però no disposen d'infraestructura. Així doncs, encara que configurar una màquina com a *Honeypot* es pot realitzar en no massa estona, haver de configurar totes les màquines d'una xarxa i la interacció entre aquestes de forma manual no és gens escalable.

En resum, es pot afirmar que es disposa de les eines necessàries per a cobrir els diferents aspectes del projecte, però no hi ha un projecte ja existent que atorgui la automatització i la escalabilitat desitjades.

2.3 Necessitats d'informació

Per a realitzar aquest projecte es necessita informació sobre els diferents aparts que ja han anat apareixent.

Per un cantó, es necessita informació sobre les diferents formes de virtualització i les característiques que ofereixen, així com els sistemes operatius a utilitzar. S'ha de tenir sempre en compte que un dels requisits de les *Honeynets* és imitar un entorn de producció de forma creïble, així que en tot moment s'ha de considerar la situació actual en l'àmbit dels servidors.

Per altra banda, s'ha de recopilar informació sobre les diferents eines que existeixen per a la detecció d'intrusos i la monitorització, sabent que aquestes han de passar desapercebudes de cara a un possible atacant. D'aquestes eines s'ha de saber les configuracions de les que disposen i la forma d'instal·lació, de tal forma que la seva automatització sigui viable.

Finalment, s'ha de buscar informació sobre les diferents vies d'entrada dels atacants i les vulnerabilitats existents. S'ha de tenir present que un accés massa fàcil pot ser sospitós per un atacant. Només d'aquesta forma se li pot treure el màxim partit a la *Honeynet* resultant.

3 Objectius i abast

Es poden identificar diferents objectius en aquest projecte, començant amb els objectius propis de les *Honeynet*:

- Recopilar informació sobre les noves vulnerabilitats descobertes dels servidors i les formes d'atacar-les. La pròpia xarxa ha de ser capaç de recopilar informació en tot moment a través d'eines de monitorització. Aquesta informació va destinada als administradors de les xarxes per tal de facilitat la labor de protegir les seves infraestructures.
- Desviar durant el màxim temps possible l'atenció d'un atacant cap a aquestes màquines, per tal que no arribi a les màquines en producció. Això s'aconsegueix simulant un entorn creïble i amb la seguretat adequada per a que s'entretingui una bona estona sense sospitar.

Per altra banda, es poden definir els objectius que aquest projecte en concret pretén afegir als anteriors:

- Facilitar el desplegament i escalabilitat d'una *Honeynet*, junt amb les eines de monitorització i una correcta comunicació entre les diferents màquines. Qualsevol persona, i en qualsevol moment, ha de poder fer-ho a través d'una línia de comandes i uns coneixements mínims sobre aquesta, sempre que es disposi d'accés a la infraestructura sobre la que es vol muntar. Amb això s'apropa la seguretat i la recopilació d'informació important al respecte a un major nombre de persones i el temps deixa de ser un problema.
- Facilitar l'aturada i modificació d'una *Honeynet* existent per tal de fer-la adaptable a les necessitats canviants dels usuaris. Això es realitza a través de comandes i del fitxer de configuració de la xarxa en qüestió. Qualsevol persona amb permisos sobre la màquina que ha creat la xarxa i el fitxer de configuració ha de poder realitzar aquest canvi.
- Proporcionar un sistema de control centralitzat per a veure en qualsevol moment l'estat de les màquines que conformen la xarxa. Qualsevol persona amb accés a la màquina de control ha de poder comprovar aquest estat a través de la línia de comandes des d'un sol punt. D'aquesta manera es pot actuar davant d'una intrusió o la necessitat de modificació de la xarxa.
- Proporcionar la topologia de la xarxa i l'accés a les diferents màquines un cop creades, per tal de facilitar la gestió futura de la mateixa a l'encarregat d'aquesta

tasca. Això s'aconsegueix a través d'un fitxer de configuració i les claus d'accés per SSH a cada una de les màquines.

El públic potencial o *target* d'aquest projecte són els administradors de sistemes o experts de seguretat, principalment. Als administradors de sistemes els interessa més aviat per a desviar l'atenció del atacants i, d'aquesta manera, disminuir el risc sobre les màquines en producció. Per altra banda, als experts en seguretat els interessa més aviat per la vessant d'investigació i recopilació de informació ja que, d'aquesta manera, poden desplegar *Honeynets* senceres en qualsevol lloc, de forma fàcil i amb poc temps.

A més, cal considerar aquelles persones amb poca experiència en el sector que volen un entorn fàcil i ràpid de muntar per a fer diverses proves, comprovar el funcionament de les eines utilitzades o inclús intentar atacar de forma segura.

4. Metodologia

Al llarg d'aquest apartat se suposa en diversos casos les alternatives comentades en la secció d'antecedents d'aquest mateix avantprojecte.

Al llarg de totes les alternatives i aspectes a considerar, sempre hi ha un criteri fonamental que està present: el resultat d'una *Honeynet* ha de ser creïble per tal de poder enganyar a l'atacant, si no tota la xarxa perd el sentit.

Pel que respecte a la plataforma de virtualització, els criteris més importants a la hora de realitzar la elecció són la flexibilitat mitjançant la línia de comandes o peticions HTTP a través d'una API i, a través d'aquests, la gestió conjunta de les màquines existents. Això és imprescindible de cara a automatitzar tots els processos, ja que les diferents tasques a realitzar s'han d'executar a través de crides per codi, no es pot dependre d'una interfície gràfica. Arrel d'aquest aspecte, es poden considerar diferents opcions de les nombrades anteriorment.

Referent a la virtualització per contenidors, s'ha de tenir en compte el software *Docker*. Aquest disposa d'una línia de comandes preparat per a configurar qualsevol cosa i, el mateix software, ofereix una capa de gestió per sobre i comuna a tots les maquines creades. A més pot incloure, a través d'eines com *docker-compose* [17], tota la lògica de configuració i execució a través de fitxers. Finalment, cal tenir en compte que és una alternativa gratuïta a la que qualsevol persona pot tenir accés i utilitzar-la en una infraestructura pròpia.

Com a alternativa també gratuïta, està la opció de *OpenStack*. Aquest disposa de una API per poder gestionar tota la infraestructura a través de peticions HTTP. Actualment, ja hi ha proveïdors de infraestructura que utilitzen aquesta eina i disposa d'un cert grau de maduresa i comunitat. Al estar a l'abast de qualsevol persona, es pot utilitzar una infraestructura pròpia.

Com a opció més madura i de major confiança, junt amb el cost que comporta la seva llicència, es poden utilitzar els productes de *VMware*. Això inclou el *ESXi* [18] com a virtualització *bare-metal* i el *vCenter* [19] com a control del primer i proveïdor d'una API de gestió.

Tot i això, la alternativa de virtualització al núvol també comporta beneficis importants, per la major proximitat amb els entorns de producció actuals i la facilitat de portar les *Honeynet* resultants a la xarxa, de forma pública.

Per altra banda, en l'elecció del sistema operatiu, la tria realitzada és utilitzar entorn Linux, per estar corrents en la majoria de servidors actuals amb una gran fiabilitat, per una major documentació en aquest àmbit sobre la majoria de programes o serveis, i per la major predisposició a crear eines gratuïtes i de codi obert, més enllà del propi sistema operatiu. Preferiblement les distribucions de *CentOS* [20] o *Ubuntu* [21], per ser aquelles gratuïtes més utilitzades a nivell de servidors.

Finalment, en l'àmbit de detecció i monitorització hi ha moltes eines, començant pels sistemes de detecció d'intrusions basats en host o en xarxa, com per exemple *Ossec* i *Snort* respectivament. A més, hi ha moltes eines que també poden aportar funcionalitats addicionals o cobrir part del que ja cobrien les anteriors. Entre aquestes, es poden trobar exemples d'eines amb una funcionalitat concreta com per exemple *kippo* [22], simulador d'una consola SSH. Per altra banda, també hi ha eines de monitorització i anàlisis semblants als sistemes de detecció d'intrusos, com per exemple *sysdig*.

Un criteri molt important en l'àmbit de monitorització i detecció és la dificultat de detectar l'eina de monitorització utilitzada, ja que un cop detectada es poden eliminar els registres o simplement anar a buscar una altra màquina víctima.

La elecció realitzada és *Ossec* per la seva llicència *Open Source* i les eines creades al seu voltant per ampliar la funcionalitat, funcionalitat que ja de per si és molt flexible gràcies a la possibilitat de definir regles pròpies de detecció i actuació.

5. Planificació i pressupost

En aquest apartat s'exposa la viabilitat a la hora de realitzar aquest projecte, des de les diferents vessants a considerar.

5.1. Planificació

Per a fer aquest projecte es disposa d'aproximadament de 500 hores, des de novembre del 2017 fins a inicis de juny del 2018. Aquestes hores s'han de realitzar per a una sola persona, encara que adoptant diferents rols al llarg del projecte. Al llarg de tot aquest període s'ha de realitzar un seguiment de l'estat del projecte i, en cas de ser necessari, realitzar els canvis pertinents en la documentació i planificació originals.

S'ha realitzat una etapa inicial en que s'ha definit el context, objectius, abast del projecte, recursos i, finalment, la viabilitat. El temps dedicat a aquest aspecte han estat 30 hores realitzades pel cap de projecte. Aquesta etapa va finalitzar a inicis del mes de desembre.

Posteriorment, s'ha realitzat un anàlisi més exhaustiu de les diferents eines i tecnologies que es poden utilitzar. Per a aquesta etapa s'han destinat 100 hores més, repartides entre l'administrador de sistemes i l'especialista en seguretat. Aquesta etapa finalitza a mitjans d'abril. Aquesta etapa ha requerit més temps del que inicialment s'havia plantejat per problemes a la hora d'aconseguir la infraestructura. Totes les proves realitzades i problemes trobats estan explicats a l'apartat corresponent al desenvolupament d'aquest treball.

De mentre es realitzava l'etapa, s'ha iniciat la programació amb algunes d'aquestes eines, per tal de poder realitzar l'anàlisi de forma més confiable i avançar amb la elaboració del producte.

L'etapa d'implementació del projecte té una durada de 3 mesos, fins a mitjans de maig, en la que es destinen 300 hores en total. Al final d'aquest apartat es concreta encara més sobre les diferents fases d'implementació i el desglossament del temps en cadascuna d'elles. Al llarg de tot el procés d'implementació s'han de realitzar proves de funcionament per cadascuna de les funcionalitats implementades, amb una estimació de 40h a realitzar per part del tester. Les 260h restants són destinades a la programació i implementació del producte en sí.

Finalment, hi ha una última etapa del projecte que correspon amb l'últim mig mes restant. A aquesta etapa es destinen 30h i serveix per finalitzar la documentació necessària referent als resultats i preparar la defensa del projecte. Aquesta tasca correspon al cap de projecte.

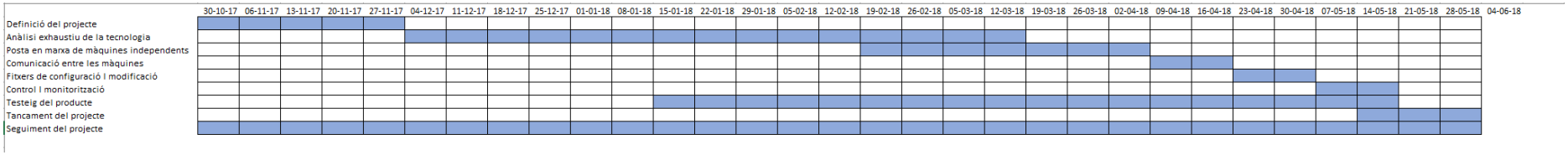
A més a més, es reparteixen 40h entre tot el projecte per la gestió, seguiment i documentació d'aquest mateix, també a càrrec del cap de projecte.

Es pot separar la etapa d'implementació del projecte comentada anteriorment en les següents fases:

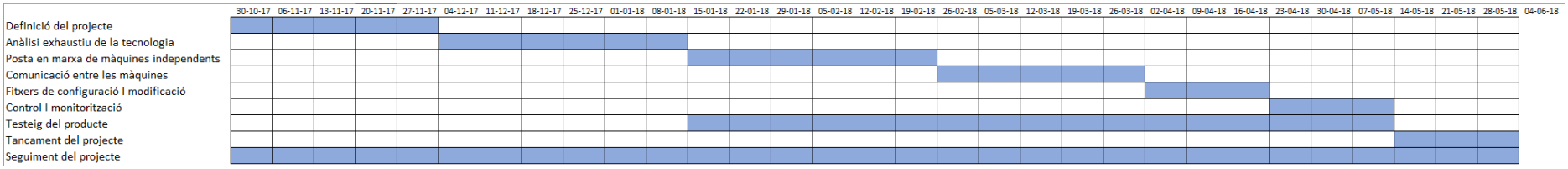
- Posta en marxa d'una màquina independent i els serveis necessaris, segons una configuració introduïda per l'usuari prèviament. El temps a dedicar és de 120h, ja que el volum de temps és conseqüència de la quantitat de configuracions i serveis diferents que es poden considerar.
- Funcionament com a xarxa entre les diferents màquines, és a dir la configuració d'aquestes per tal d'aconseguir una comunicació correcte entre elles. El temps a dedicar és de 80h.
- Realització dels fitxers de configuració com a resposta de la creació de la xarxa i la possibilitat d'aturar, modificar i fer córrer la xarxa novament a partir d'aquests. El temps a dedicar és de 30h.
- Afegir els sistemes de control i monitorització en totes les màquines, així com el sistema de visualització centralitzat. El temps a dedicar és de 30h.
- Proves de funcionament i qualitat, així com les correccions pertinents. El temps a dedicar és de 40h. Mentre que les fases anteriors es realitzen en l'ordre en que s'han explicat prèviament, aquesta està repartida durant tot el procés d'implementació.

Tots aquests terminis i tasques s'han resumit en la Imatge 2.

Per altra banda, en la Imatge 3 es pot observar la planificació inicial que es va realitzar, amb diferències respecte el calendari que s'ha explicat i que finalment s'ha seguit. El temps d'anàlisi de la tecnologia, sobretot pel que fa a la infraestructura, s'ha allargat més de l'esperat degut als diferents inconvenients trobats. Aquests inconvenients s'explicaran al llarg de la memòria en els apartats pertinents. En conseqüència, la implementació del projecte ha començat més tard, afectant negativament al temps emprat en això i les funcionalitats totals programades.



Imatge 2. Diagrama de Gantt. Absicess pels dies. Ordenades per les tasques. Planificació final



Imatge 3. Diagrama de Gantt. Absicess pels dies. Ordenades per les tasques. Planificació inicial

5.2. Pressupost

Primerament, considerant diferents persones per a realitzar aquest projecte, amb el seu càrrec i sou, ens trobem amb un cost aproximadament com el següent:

- Cap de projecte: 40€/h x 100h -> 4000€
- Especialista en seguretat: 30€/h x 60h -> 1800€
- Administrador de sistemes: 30€/h x 20h -> 600€
- Programador: 25€/h x 280h -> 7000€
- Tester: 20€/h x 40h -> 800€

Cost total personal: 14200€

En la consideració d'aquests preus, s'ha tingut en compte ofertes actuals amb els diferents rols identificats. La cerca ha estat acotada a persones amb experiència i s'ha extret un valor arrodonit de la mitja obtinguda. A més, s'ha procurat que no hi hagi una diferència massa gran com per poder generar malestars, sempre tenint en compte que, certament, els rols amb millor remuneració són aquells que necessiten més capacitat de lideratge i coneixements.

Referent a la maquinària de treball, es necessita un ordinador capaç de virtualitzar diferents màquines a la vegada, amb el que es pot assolir un preu de 1600€ en la seva compra. Tenint en compte que la vida útil són quatre any i el termini del projecte són 7 mesos, el cost proporcional per aquest ordinador és el següent:

1600€ x (7/48 mesos) -> 233.33€

Referent al consum energètic (0.123€/kWh) tenim el següent:

- Ordinador: 500h x 0.220Kw -> 110kwh -> 13.53€
- Llum: 500h x 0.060kw -> 30kwh -> 3.69€

Cost energètic total: 17.22€

Per altra banda, es pot realitzar una estimació sobre el cost de desenvolupar aquest sistema en el núvol. Per a aquesta hipòtesi, se suposa que l'entorn de test s'utilitza des de que es comença la implementació fins a que es presenta el projecte i s'atura quan no s'està utilitzant (un total de 300h aproximadament). A més, es suposa que les proves es realitzen sobre xarxes formades per 4 màquines de mitja. Es suposa també que es realitza una prova en el núvol cada hora, ja que la resta serà en màquines virtuals en local, sumant així 300 proves. Finalment, s'utilitza el preu mínim d'una màquina a OVH que és 0.012 € la hora. Partint d'aquestes premisses i a mode d'aproximació, s'obté el següent cost:

300proves x 4màquines x 0.012€/h -> 14.04€

Cal tenir en compte que el còmput d'hores sempre es realitza arrodonint cap amunt. Aquesta característica afecta molt negativament i de forma poc previsible a les proves momentànies de creació i eliminació de màquines.

Donada la poca fiabilitat del cost en un entorn de virtualització en el núvol, juntament amb la falta d'una decisió final sobre el mode de virtualització, dificulten el càlcul de costos en aquest apartat.

Finalment, el cost que té el manteniment d'una xarxa, de cara a l'usuari final que utilitzi el nostre producte, depèn completament de les magnituds de la xarxa en qüestió. Així doncs, tampoc es pot realitzar un càlcul acurat, sinó que s'ha de tornar a parlar d'aproximacions.

A mode de resum, els pressupostos queden representats en la Taula 1.

Cost del personal	14200€
Ordinador de treball	233.33€
Consum energètic	17.22€
Virtualització en el núvol	14.04€
Software utilitzat	0€
Total	14464.59€

Taula 1. Pressupost del projecte.

6. Anàlisi de viabilitat

6.1. Anàlisi de viabilitat tècnica

Tal com ja s'ha comentat al llarg d'aquest avantprojecte, les eines i tecnologies necessàries per a implementar les diferents funcionalitats d'aquest projecte ja existeixen. Així doncs, la tasca a realitzar és bàsicament d'automatització a través de software, és a dir codi. Per aquesta raó, aquest projecte es viable des d'un punt de vista tecnològic.

6.2. Anàlisi de viabilitat econòmica

Encara que els costos, sobretot en la part personal, augmentin una mica, es mou en una xifra perfectament assumible per a una empresa.

Per altra banda, hi ha alternatives de virtualització que no afegeixen cap cost addicional i software a utilitzar de forma lliure. Per aquestes motius, la viabilitat econòmica per a fer aquest projecte no és cap inconvenient.

Finalment, amb aquest producte es poden utilitzar dos models de negoci no excloents entre si: vendre el producte en sí o vendre la informació recopilada gràcies a aquest. Tot i això, un cop més, la realitat que hi ha darrera d'aquest projecte no és empresarial i, per tant, la intenció no és vendre-ho.

6.3. Anàlisi de viabilitat mediambiental

En cas de tenir una infraestructura pròpia, un CPD, simplement s'han de tenir les mesures adequades envers a aquest, com qualsevol empresa del sector. Per altra banda, si s'acaba desplegant al núvol, esdevé un aspecte que ni tan sols pertoca contemplar, ja que el propi proveïdor assumeix la responsabilitat.

Així doncs, aquest projecte també és viable des d'un punt de vista mediambiental.

6.4. Aspectes legals

Tal com s'ha comentat, les eines utilitzades en la implementació d'aquest projecte són eines d'utilització gratuïta o codi lliure en la seva majoria, així que es poden fer servir sense cap inconvenient. No hi ha necessitat de llicències ni recursos amb propietat intel·lectual, amb l'excepció dels productes de *VMware*. De totes formes, en cas d'utilitzar aquest últim, es farà amb les llicències corresponents.

Per altra banda, s'ha de tenir en compte que, en cas de simular serveis públics que demanin les dades personals, com per exemple un registre en una web, s'ha de redactar unes condicions d'ús i polítiques de privacitat d'obligatòria acceptació per part del client o atacant. Compte amb la llei de protecció de dades a la hora d'emmagatzemar informació.

7. Requeriments

Podem diferenciar els requeriments en dos tipus, funcionals i tecnològics.

7.1. Requeriments funcionals

A nivell funcional, es poden definir els següents requeriments:

- Crear de forma automatitzada, amb només una configuració prèvia, un conjunt de màquines que formen part d'una mateixa xarxa.
- Extreure informació sobre el comportament i estat de cadascuna de les màquines, així com les interaccions dels usuaris amb aquestes.
- Assolir una comunicació correcte entre les diferents màquines de la xarxa i els serveis de les que disposa cadascuna.
- Poder visualitzar l'estat de qualsevol de les màquines des d'un únic punt de control comú.
- Permetre definir el port SSH, si no es vol per defecte, abans de la creació de la màquina en qüestió.
- Poder definir un nou usuari en el moment de la configuració de cada màquina.
- Obtenir un fitxer de configuració per tal de poder engegar, modificar i aturar la *Honeynet* de forma fàcil.
- Obtenir un fitxer amb la relació de les màquines, adreces, dominis, claus d'accés i serveis que s'estan executant, així com els registres DNS que s'han d'afegir al servidor corresponent.
- Proporcionar un sistema de creació i gestió independent de la quantitat de màquines desitjades.
- Assolir el funcionament de tota la xarxa i els seus serveis en un temps màxim de deu minuts a partir de la definició d'aquesta.
- Restringir l'accés a les màquines en aquells ports en que no és necessari.
- Proporcionar una via per carregar les imatges pròpies que es vulguin utilitzar com a punt de partida en la configuració de les màquines.
- Proporcionar un sistema de gestió de totes les funcionalitats sense necessitat d'una interfície d'usuari.

7.2. Requeriments tecnològics

Els requeriments tecnològics no es poden acotar gaire, ja que la grandària d'una xarxa i els recursos utilitzats en aquesta depenen de la configuració desitjada per l'usuari. És un projecte basat en la automatització i la escalabilitat i, per tant, el rang de recursos tecnològics necessaris és molt gran i depèn en gran mesura de l'usuari, no del projecte en sí.

Realment, és difícil acotar-ho inclús a nivell d'una sola màquina, ja que els serveis que es volen fer córrer en aquesta marquen els recursos necessaris. Tot i això, es pot fer una estimació al voltant de 1GB de memòria, 10GB d'espai en disc i un processador per a cada màquina, tenint en compte que un Linux sense entorn gràfic necessita pocs recursos i que, en no ser un entorn de producció, el volum de feina que ha de suportar és baix.

Cal tenir en compte que es tracta d'un projecte purament software i, els recursos necessaris per la virtualització de les màquines depenen de la topologia de xarxa i els serveis que l'usuari vulgui imitar. Tal com s'ha dit en anteriors ocasions, les funcionalitats a implementar en aquest projecte són independents de la mesura de la xarxa resultant.

Tot i això, es pot identificar com a recursos tecnològics també les eines explicades en apartats anteriors que cobreixen parts concretes del projecte, més enllà dels requeriments tecnològics físics. Aquestes eines inclouen les diferents formes de virtualització, els sistemes operatius que fan funcionar les màquines, els serveis disponibles en cadascuna d'aquestes o les eines de monitorització i detecció d'intrusos.

8. Desenvolupament

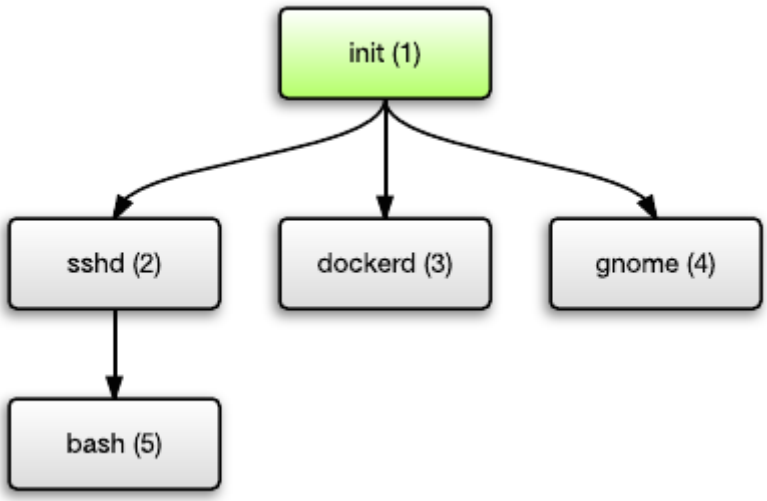
Al llarg d'aquest apartat s'explicarà el desenvolupament del treball: tot allò que s'ha implementat, de quina forma i quins problemes s'han trobat.

El desenvolupament del treball comença amb una cerca de la infraestructura a fer servir. Al llarg d'aquesta, s'han provat diferents alternatives i han sorgit varis problemes, fet que ha comportat que la primera etapa del treball hagi requerit més temps del plantejat inicialment.

8.1. Docker

Docker és un entorn de virtualització mitjançant contenidors en el que no s'arriba a executar tot un sistema operatiu sencer, a diferència dels sistemes de virtualització completa. La funció de *Docker* és executar uns serveis concrets de forma aïllada de la resta del sistema, de forma ràpida i còmoda mitjançant contenidors.

De forma general, un sistema operatiu està format per un conjunt de processos ordenats en forma d'arbre que s'executen i moren segons la situació i ordres del sistema. Ara bé, per a poder formar aquest arbre de processos ha d'haver un primer procés amb PID (*Process Identification Number*) 1. Es pot veure un esquema de processos en la Imatge 4.



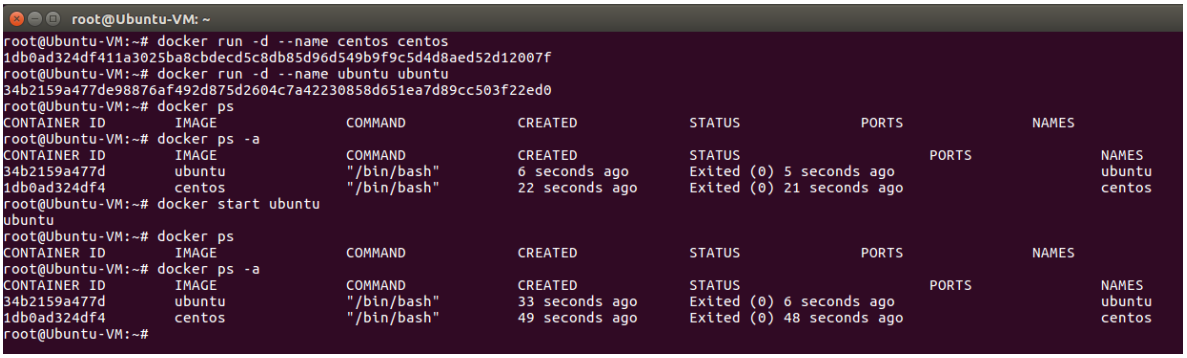
Imatge 4. Arbre de processos. Bibliografia [23]

Aquest procés amb PID 1 és executat pel *Kernel* en iniciar el sistema i es responsabilitza d'executar la resta de serveis del mateix. A més, s'encarrega de que tots els processos finalitzin correctament, encara que s'hagin interromput abans de forma inesperada.

Docker, per altra banda, no executa un procés pare amb aquestes responsabilitats, sinó que directament executa el servei pel qual s'hagi destinat el contenidor. En cas de finalitzar aquest servei i quedar-se sense processos en marxa, s'atura el contenidor sencer i la màquina amfitriona passa a ser l'encarregada de finalitzar correctament els processos associats. Per aquesta raó, *Docker* està orientat a una arquitectura de microserveis, però no a assolir totes les responsabilitats d'un sistema operatiu.

8.1.1. Executar imatges dels sistemes operatius

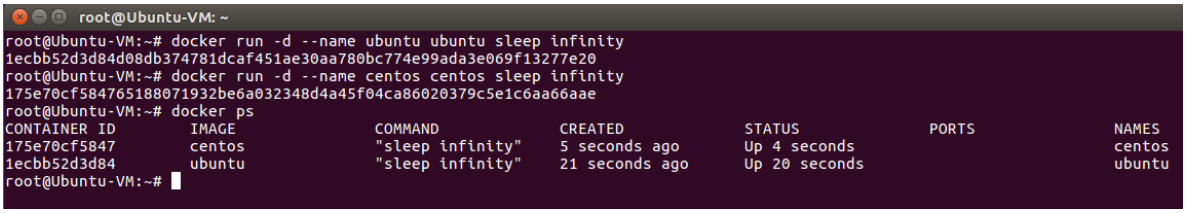
Un dels primers inconvenients es troba en intentar executar una imatge d'un sistema operatiu sense serveis addicionals. Tal com es pot veure en la Imatge 5, les imatges de *Docker* dels sistemes operatius oficials CentOS i Ubuntu no es poden executar.



Imatge 5. Executar imatges Docker de SO

La raó d'aquest fet és la absència de processos en els contenidors resultats, ja que no hi ha un procés pare executant-se de forma permanent i encarregat de posar en marxa i finalitzar correctament la resta de processos.

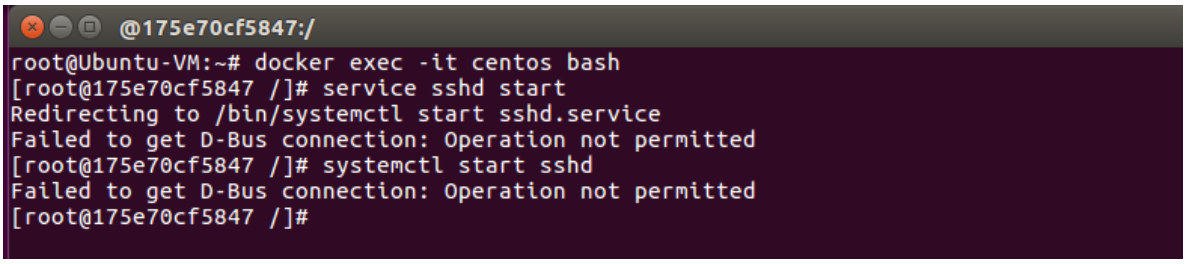
Aquest inconvenient es podria resoldre de forma bruta executant un arxiu amb un bucle infinit que mantingués el procés associat en marxa. De totes formes, la comanda *Linux* adequada per a resoldre-ho és "*sleep infinity*", tal com es pot veure en la Imatge 6.



Imatge 6. Executar imatges Docker amb sleep infinity

8.1.2. Imatges CentOS i control de serveis

En la imatge oficial de *CentOS* es troben problemes per gestionar els serveis del sistema operatiu, ja que no es poden utilitzar les principals eines destinades a aquest fi. Tal com es pot veure en la Imatge 7, no es pot gestionar un servei mitjançant l'eina *systemctl* [24]. Per altra banda, la comanda *service* [25] tampoc bé inclosa per defecte, encara que es pot instal·lar junt amb el paquet *initscripts* [26]. De totes formes, tampoc serveix ja que internament realitza una redirecció al cas anterior.



Imatge 7. Problemes systemctl CentOS

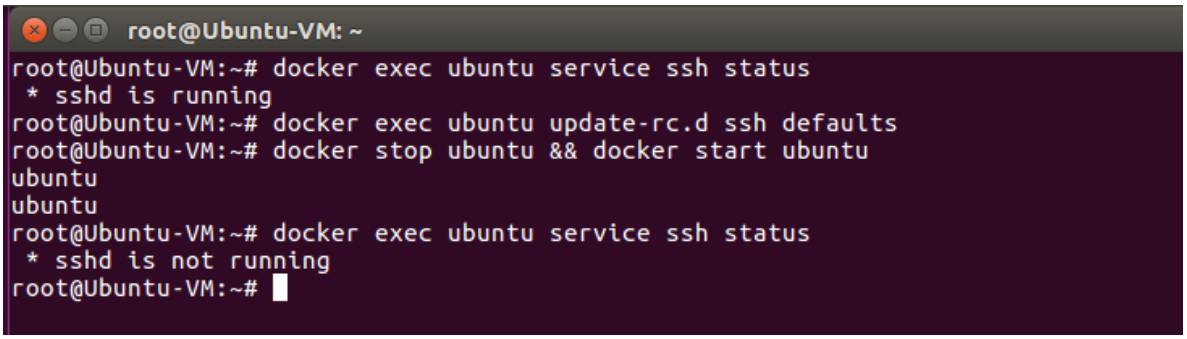
Segons la pàgina web oficial de la imatge *Docker* de *CentOS* [27], es pot fer funcionar una imatge amb el *systemd* [28], seguint uns passos indicats en la mateixa. Tot i això, després de crear un contenidor seguint les instruccions, no s'ha aconseguit fer funcionar i els problemes amb la gestió de serveis segueixen sent els mateixos.

Degut als problemes explicats amb la imatge de *CentOS* i l'ús majoritari del sistema operatiu *Ubuntu* per als contenidors *Docker*, s'ha escollit aquest últim per a seguir treballant amb *Docker*. Aquest us major facilita la compatibilitat i diversitat de contenidors ja preparats amb *Ubuntu* i es deu, en part, a la estreta relació entre la companyia encarregada de *Ubuntu*, *Canonical* [29], i la companyia de *Docker*. Es poden trobar diverses noticies que corroboren aquesta relació [30].

8.1.3. Serveis executats en l'inici del sistema

Un servei imprescindible que s'ha d'executar junt amb l'inici dels nostres sistemes és el *Secure Shell (SSH)* [31], servei per accedir de forma remota a una màquina. A través d'aquest, es poden realitzar la resta d'instal·lacions i configuracions necessàries, a part de garantir l'accés remot sempre que sigui necessari, per tal de poder fer una gestió manual per part de l'usuari administrador de les màquines.

En aquest cas, tornen a haver inconvenients relacionats amb els contenidors *Docker*, ja que els serveis no s'inicien junt amb el sistema. Es pot veure la demostració d'aquest cas en la Imatge 8.



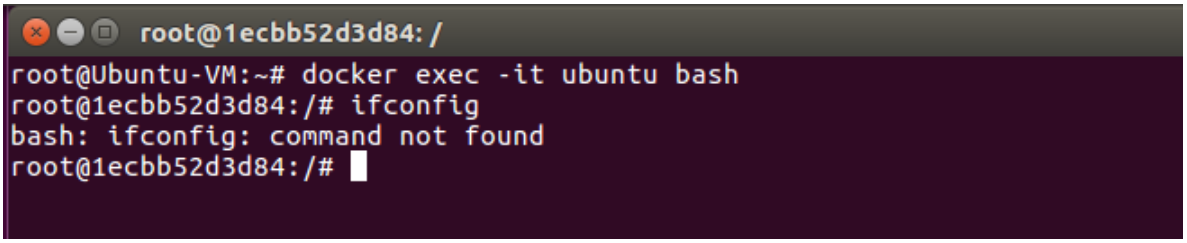
Imatge 8. Serveis no iniciats amb el sistema

Un cop més, el fet de que els serveis no s'iniciïn junt amb el sistema, és degut a l'absència d'un procés pare encarregat d'executar tota la resta. Els contenidors s'executen a partir d'una sola comanda que, al seu torn, pot executar un *script* que posi en marxa la resta de serveis. S'ha de tenir sempre en ment que l'arrancada d'un sistema operatiu sencer i la arrencada d'un contenidor segueixen uns procediments diferents.

8.1.4. Altres inconvenients

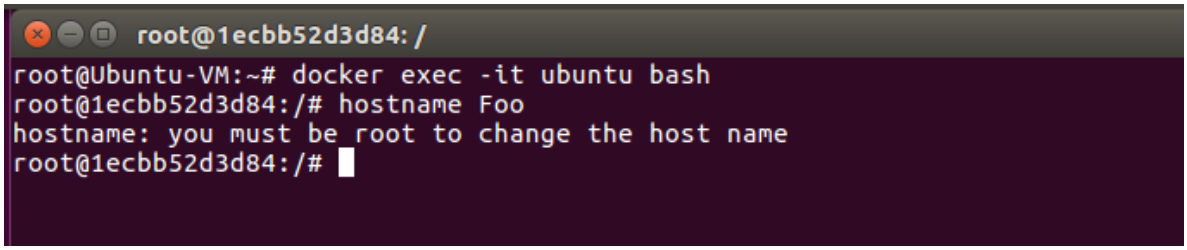
Els punts explicats anteriorment fan referència als inconvenients més grans trobats a l'entorn *Docker*. A més, encara es poden afegir alguns:

- Falta de utilitats molt bàsiques. Veure Imatge 9.



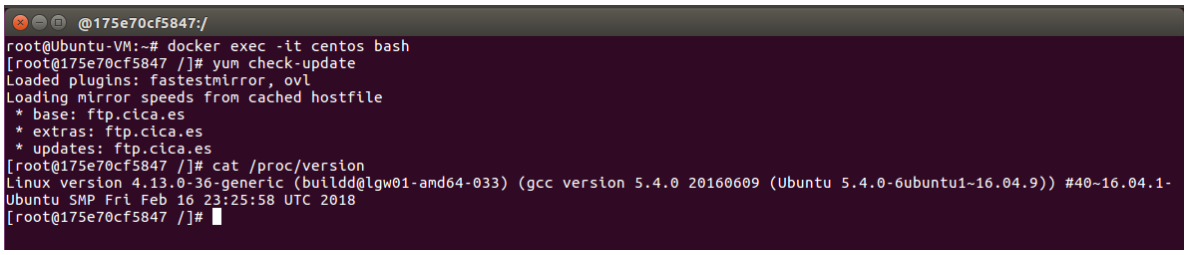
Imatge 9. Utilitat ifconfig desconeguda

- Impossibilitat de canviar el nom de la màquina des de l'usuari *root*. Veure Imatge 10.



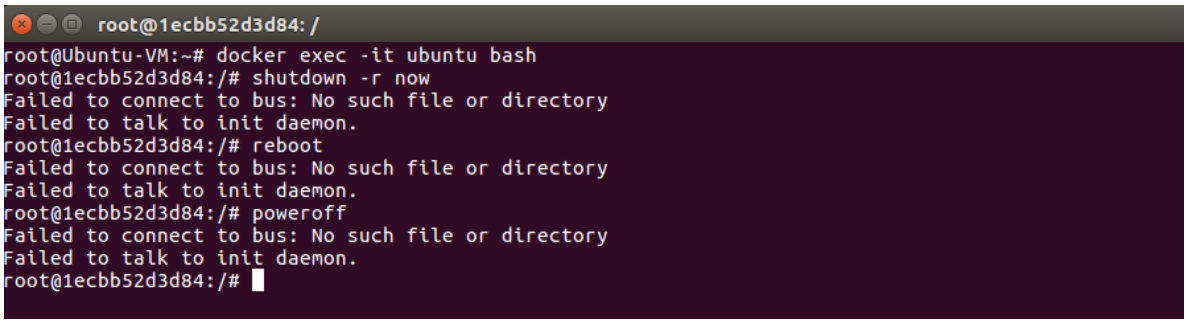
Imatge 10. Hostname sense permisos

- El sistema operatiu que t'informa el contenidor és el de l'amfitrió. Veure Imatge 11.



Imatge 11. CentOS informa el SO Ubuntu de l'amfitrió

- Impossibilitat de reiniciar el sistema des del contenidor. Veure Imatge 12.



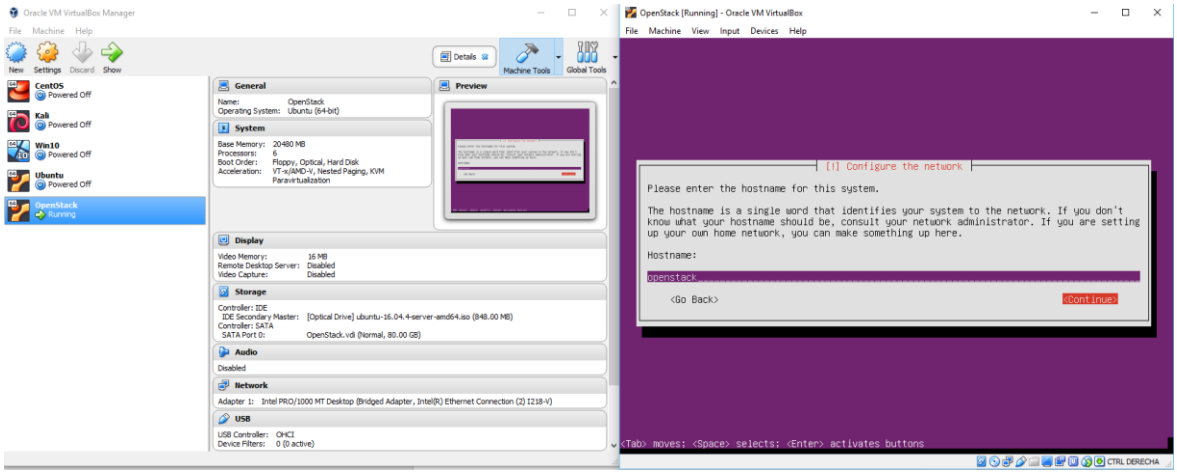
Imatge 12. Reinici del sistema no permès

Possiblement, a base d'afegir utilitats a una imatge i crear un arxiu de inicialització suficientment gran, es podrien resoldre bona part d'aquests inconvenients. Tot i això, s'estaria fent servir *Docker* per un propòsit pel qual no està pensat i la resolució hauria de ser concreta pel seu entorn, perdent compatibilitat del *software* amb altres plataformes i sistemes operatius, inclús amb altres imatges de *Docker* per a les quals no s'hagi programat. A més, una de les principals característiques d'una *HoneyNet* és la credibilitat i, des de dins la màquina, és difícil d'aconseguir.

Pels motius comentats al llargs d'aquests últims apartats, *Docker* no és un bon punt de partida pel projecte que es vol realitzar.

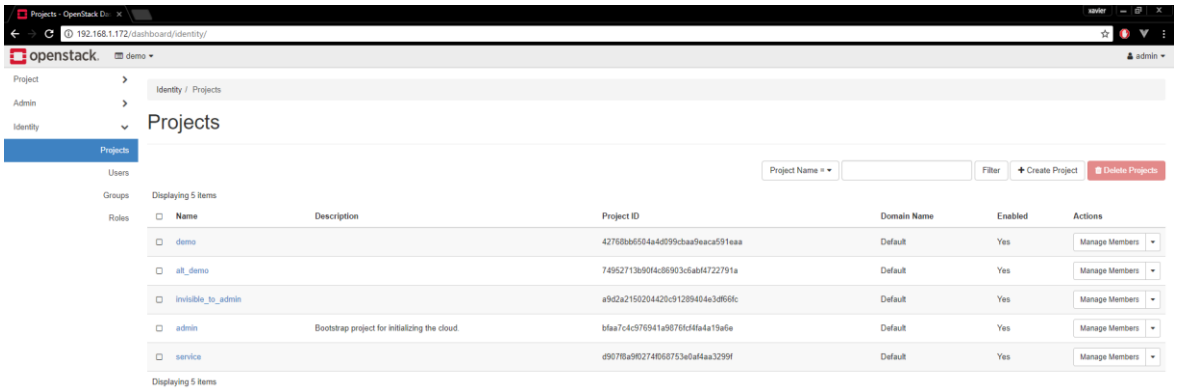
8.2. OpenStack propi

Per tal de realitzar les proves amb *OpenStack*, s'ha utilitzat una màquina local i virtualització de la infraestructura a través de *VirtualBox,* tal com es pot veure en la Imatge 13.



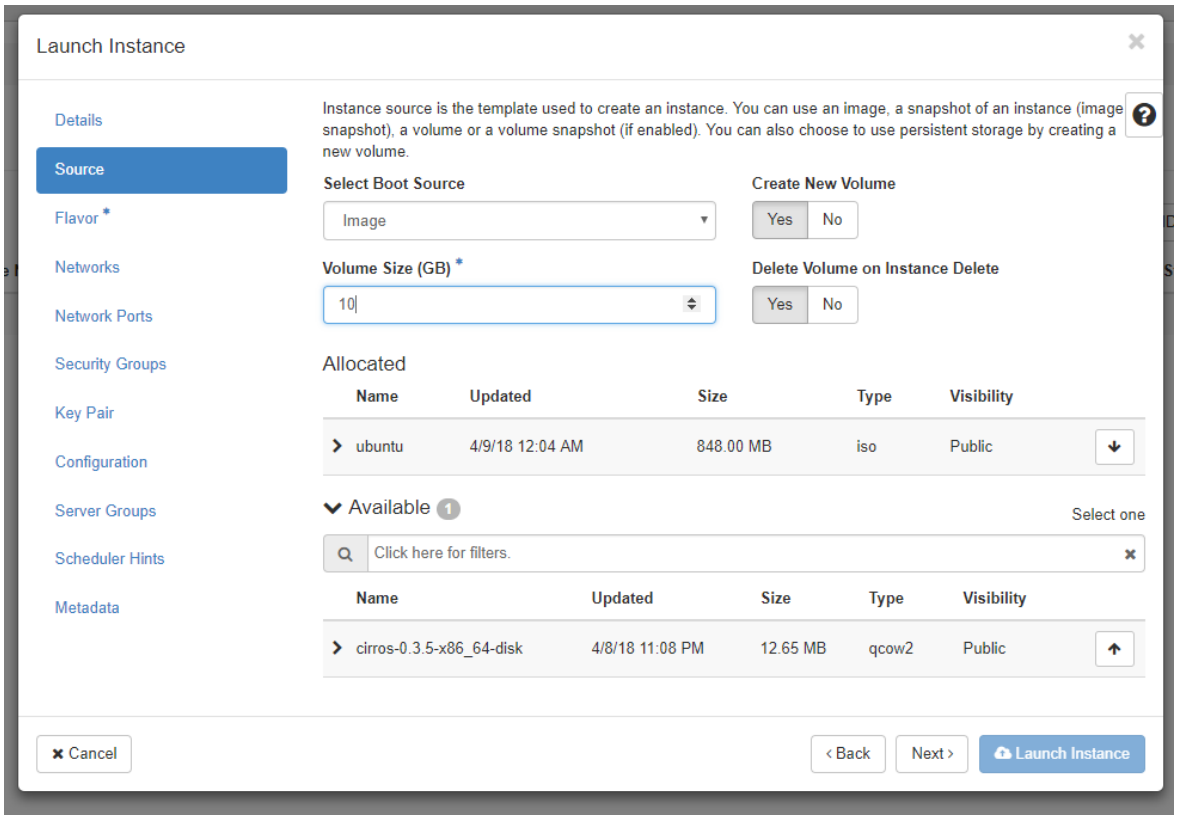
Imatge 13. Creació màquina virtual per OpenStack.

Un cop amb la màquina preparada, s'ha procedit amb la instal·lació pròpiament del *OpenStack*, seguint els passos de l'utilitari *DevStack* [32], trobat en la mateixa pàgina oficial. Després de posar-se en marxa, el panell d'administrador es visualitza com en la Imatge 14.



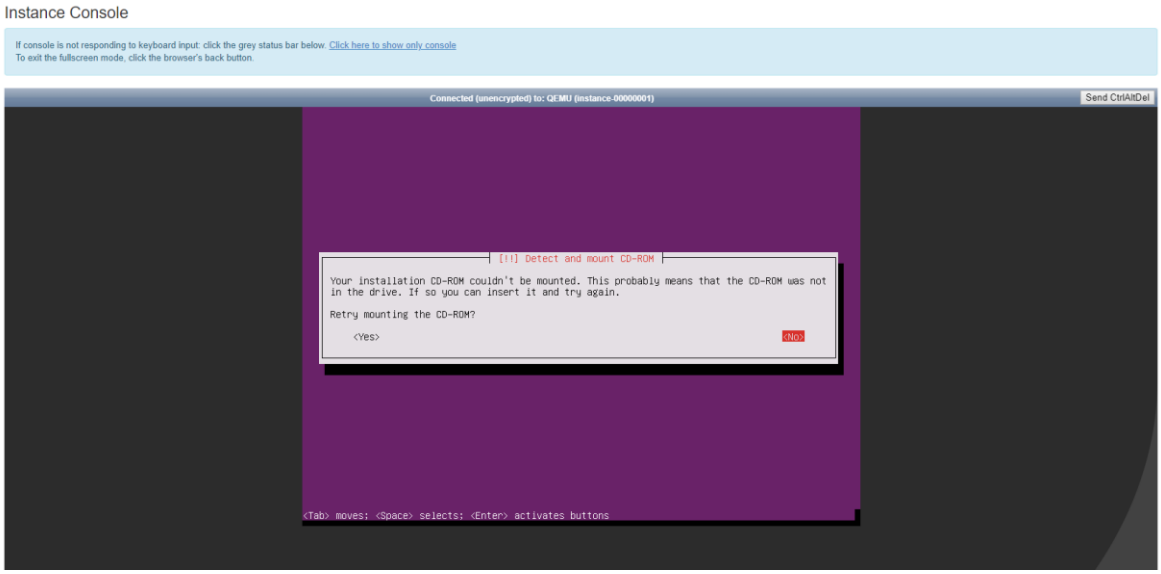
Imatge 14. Panell d'administració OpenStack

En aquest punt, navegant pel panell d'administrador, ja es poden crear imatges i instancies, tal com es mostra en la Imatge 15.



Imatge 15. Creació de instancia OpenStack.

Tot i això, es troba algun inconvenient en la creació de instàncies. Si el volum es crea junt amb la mateixa instància, tal com s'ha fet en la Imatge 15, aquest no es detectarà a la hora d'intentar instal·lar el sistema operatiu, portant a errors com els de la Imatge 16.

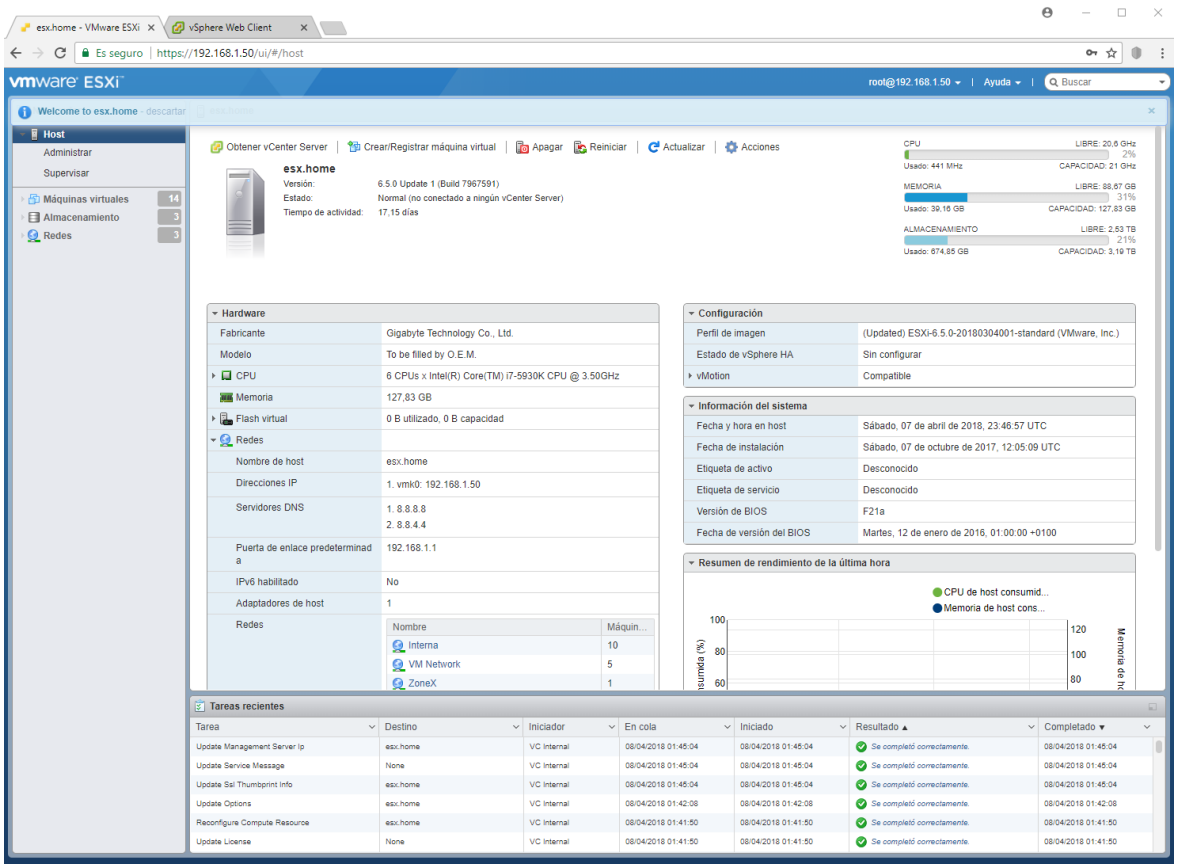


Imatge 16. Error volums creats amb la instància OpenStack.

En canvi, si primer es crea el volum, després la instància i, finalment, se li assigna, es pot procedir amb la instal·lació del sistema operatiu. Tot i això, la instal·lació tarda molt més temps del que necessita amb altres infraestructures. Possiblement hi ha paràmetres de configuració que permeten guanyar rendiment en aquest aspecte però, de totes formes, virtualitzar sobre virtualitzat no és el més òptim.

8.3. VMware

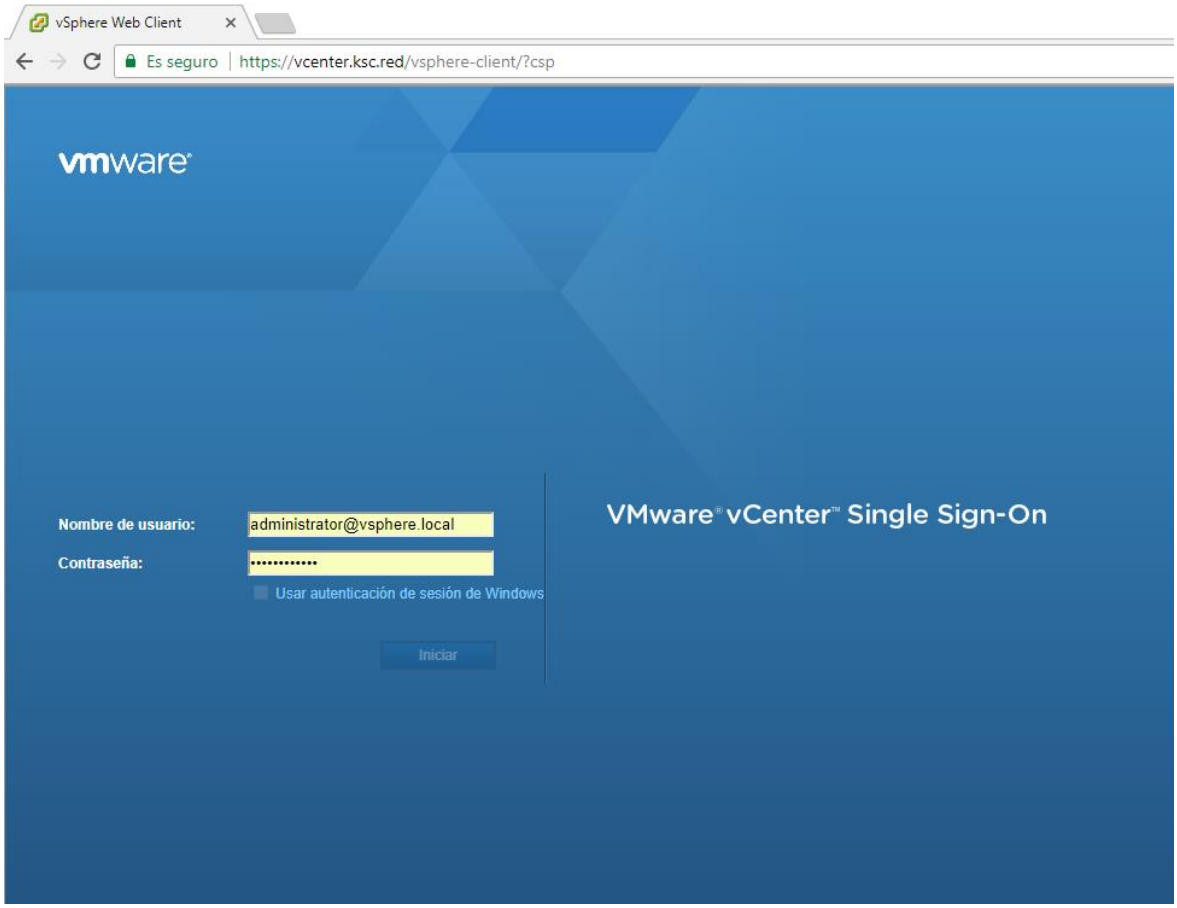
Per analitzar els productes de *VMware*, s'ha partit d'una infraestructura sobre la que està muntat un hipervisor *ESXi*, tal com es pot veure en la Imatge 17.



Imatge 17. Panell d'administració del ESXi

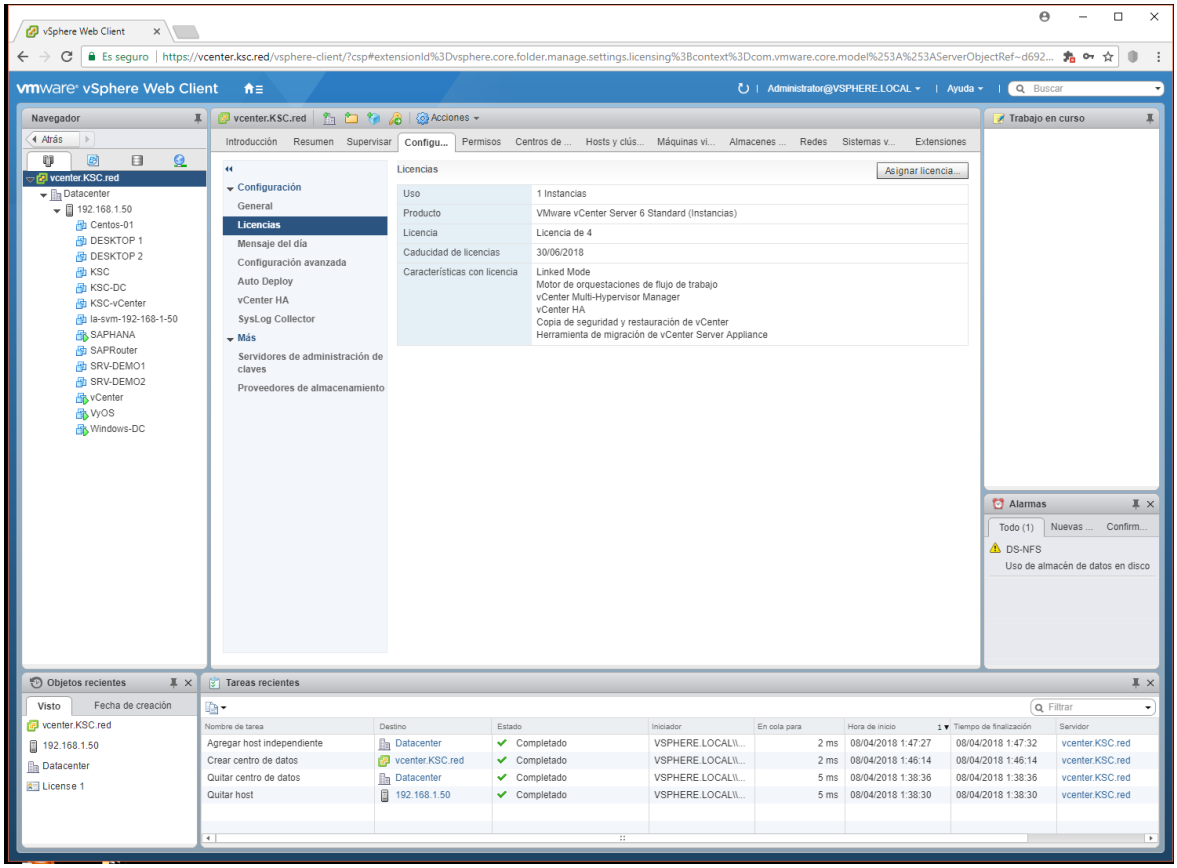
Ja que aquesta infraestructura està muntada sobre una xarxa privada, s'ha d'accedir a aquesta a través de VPN (*Virtual Private Network*) i el resultat no serà visible en la xarxa pública. Tot i això, amb una mínima flexibilitat en la configuració del codi, es pot utilitzar per muntar la topologia desitjada sobre direccions IP públiques, fet que no suposa cap inconvenient.

Per a gestionar la infraestructura de forma dinàmica, s'han afegit més productes de *VMware*, entre els quals es destaca el *vCenter*, tal com es pot veure en la Imatge 18. Aquest últim ofereix una API que es pot trobar de forma pública [33].



Imatge 18. Inici de sessió vCenter

Inicialment es van utilitzar les llicències temporals de prova que ofereix VMware, però no disposen de suficients permisos. Finalment, s'han pogut aconseguir llicències amb les que s'ofereixen més permisos, tal com es pot veure en la Imatge 19. Tot i això, encara no s'han pogut realitzar totes les proves desitjades.

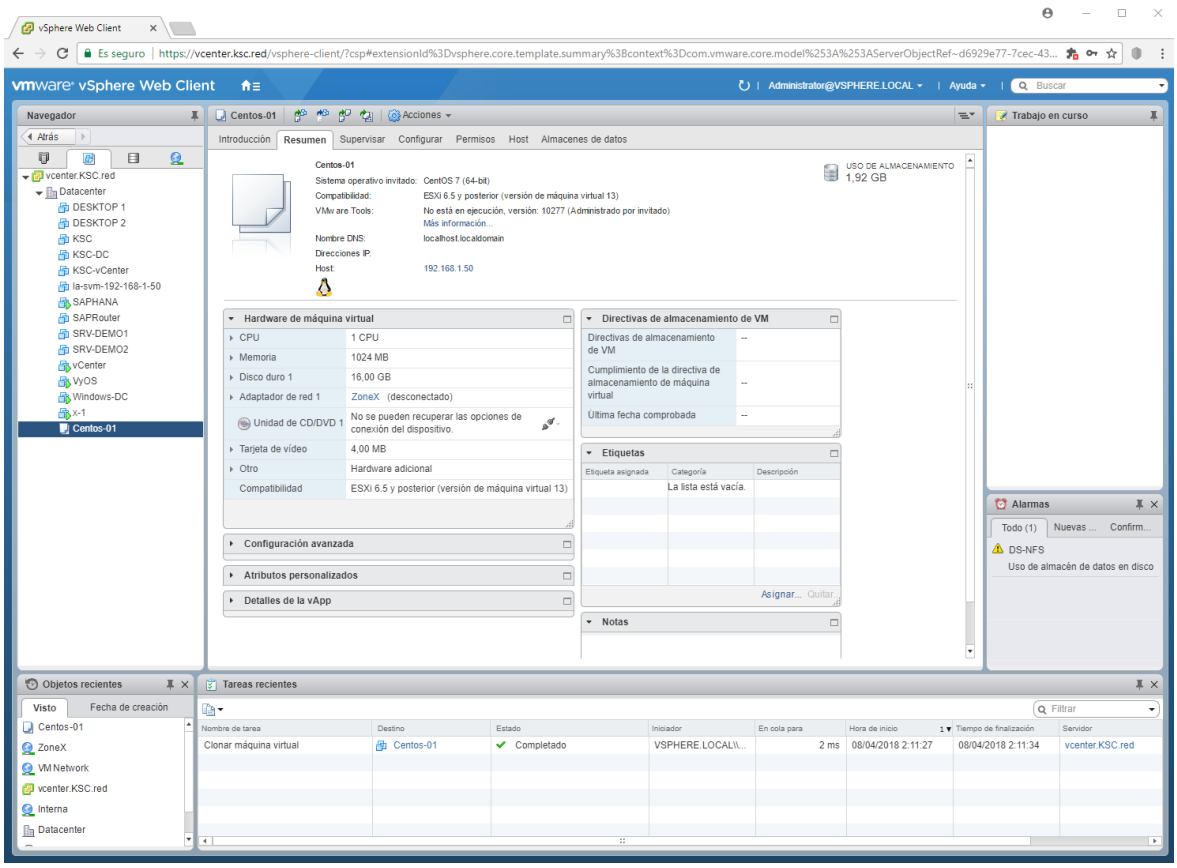


Imatge 19. Llicència de VMware.

Per altra banda, també en la Imatge 19, es pot observar com ja hi ha diverses màquines en funcionament, algunes prèvies a aquest treball. Entre aquests *hosts*, podem destacar els següents:

- *vCenter*: màquina on s'ha instal·lat el *vCenter* per a la gestió del *ESXi*.
- *Windows-DC*: controlador del domini. El *vCenter* s'autentifica a través d'aquesta màquina.
- *Vyos*: màquina que actua com a *router*, utilitzant el sistema operatiu *VyOS* [34]. S'ha establert una connexió VPN cap a aquest *router* i s'ha reservat la xarxa 192.168.3.0/24 per a la realització del treball.

Finalment, s'ha preparat una plantilla amb *CentOS* com a sistema operatiu, amb el servei SSH preparat per connectar-se tant bon punt la nova màquina es posi en marxa. Es pot veure en la Imatge 20.



Imatge 20. Plantilla CentOS creada pel vCenter

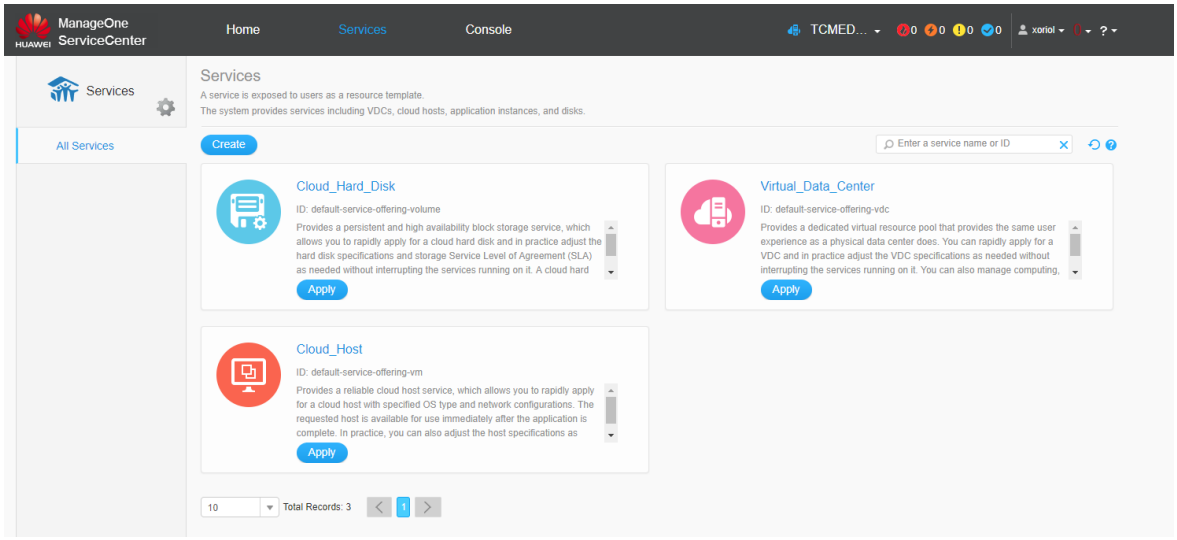
Tot i que aquesta opció s'ha madurat molt, finalment ha hagut inconvenients amb la VPN que no han permès treballar directament des de la mateixa xarxa de la infraestructura. D'aquesta manera, adaptar el programa a aquest escenari implicaria perdre de vista la utilitat de cara a les infraestructures actuals i casos reals en producció.

8.4. FusionSphere

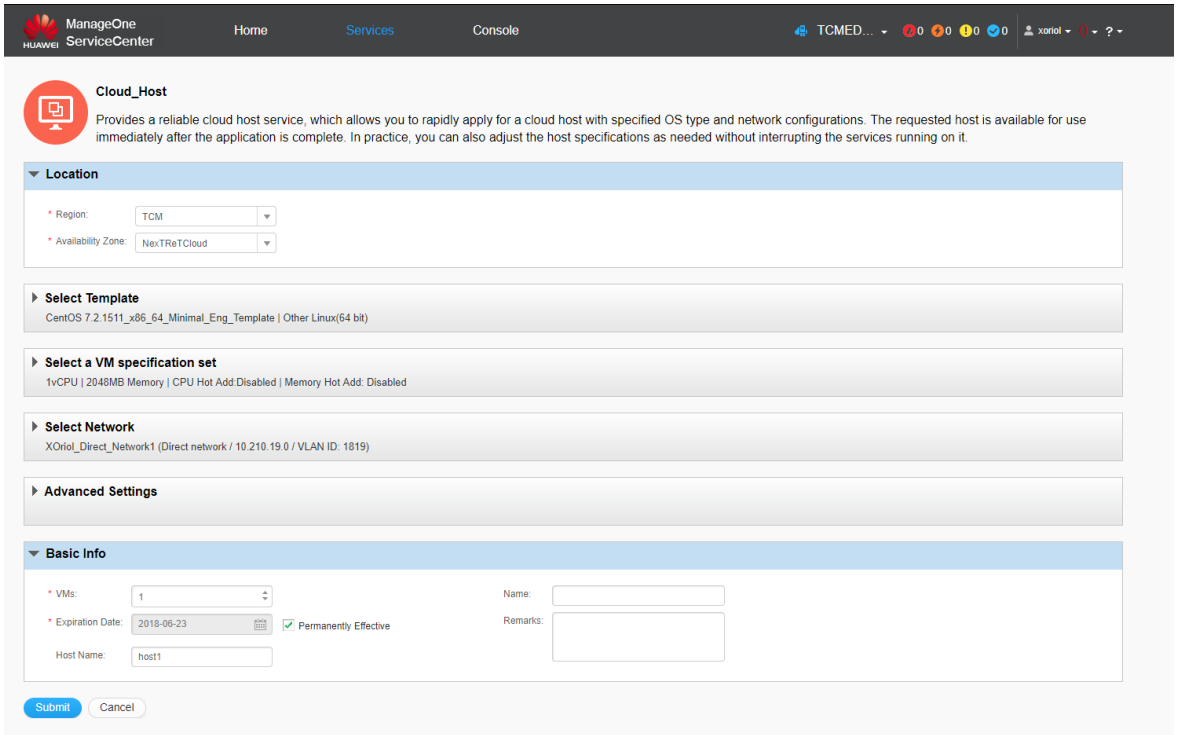
Referent a la virtualització al núvol, s'ha disposat d'una infraestructura i *Software* propis de la marca *Huawei*, anomenada *FusionSphere*. Es tracta d'una infraestructura nova, de la qual encara no es té gaire coneixement ni referències. Es pot trobar més informació al respecte en la seva pàgina web [35].

Un cop s'accedeix a la seva plataforma web de gestió, es troba amb la interfície mostrada en la Imatge 21. La opció corresponent a la creació d'una nova màquina és *Cloud_Host*, necessària per a comprovar el funcionament de la plataforma abans d'intentar implementar la funcionalitat a través de la API.

Per altra banda, en la Imatge 22 es pot veure els diferents elements a configurar per a la creació d'una nova màquina.



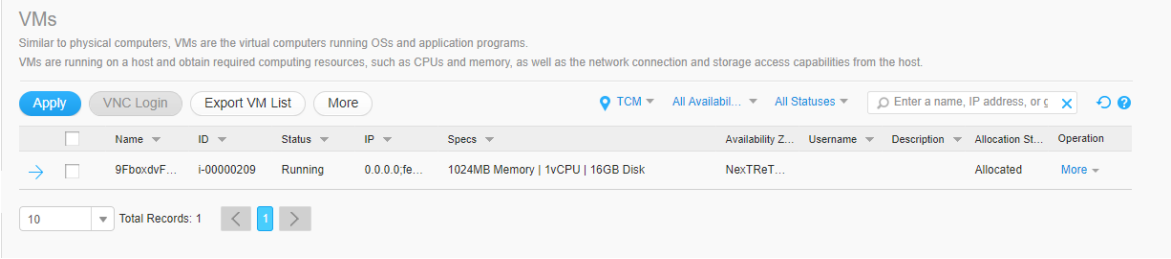
Imatge 21. Serveis de FusionSphere



Imatge 22. Creació d'una nova màquina amb FusionSphere

Després de provar les diferents opcions que ofereix la plataforma, tant en la creació com en el manteniment de la infraestructura, s'han trobat diferents errors. Alguns d'aquests fan referència a la pròpia plataforma web i compatibilitat amb els diferents navegadors, fet que dificulta la gestió i comoditat de la mateixa. Altres errors, en canvi, afecten directament a

les màquines creades i impossibiliten el treball amb les mateixes. Entre els inconvenients més greus, està el fet de que les màquines creades perden la IP assignada i la connexió a internet. Tal com es pot veure en la Imatge 23, la màquina creada i en marxa no disposa de direcció IP, ja que aquesta apareix com 0.0.0.0. A més, accedint a través de la interfície web, es pot comprovar que no hi ha connexió amb l'exterior.



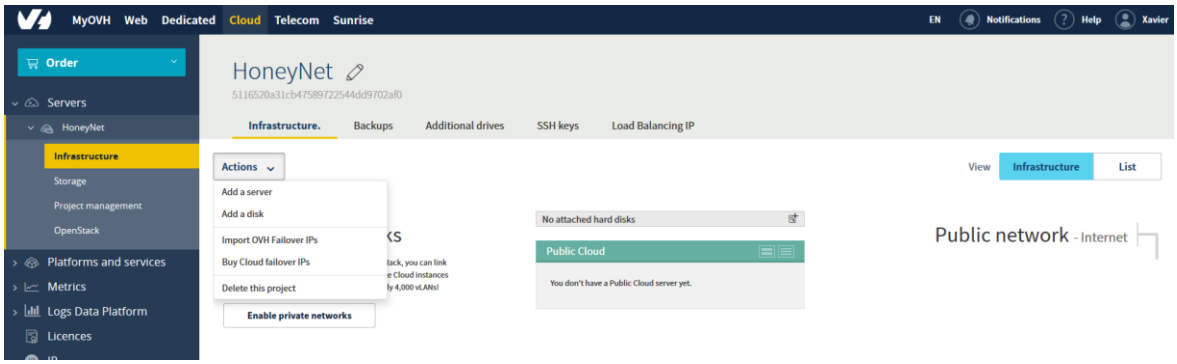
Imatge 23. Llistat de les màquines creades amb FusionSphere

8.5. Infraestructura utilitzada

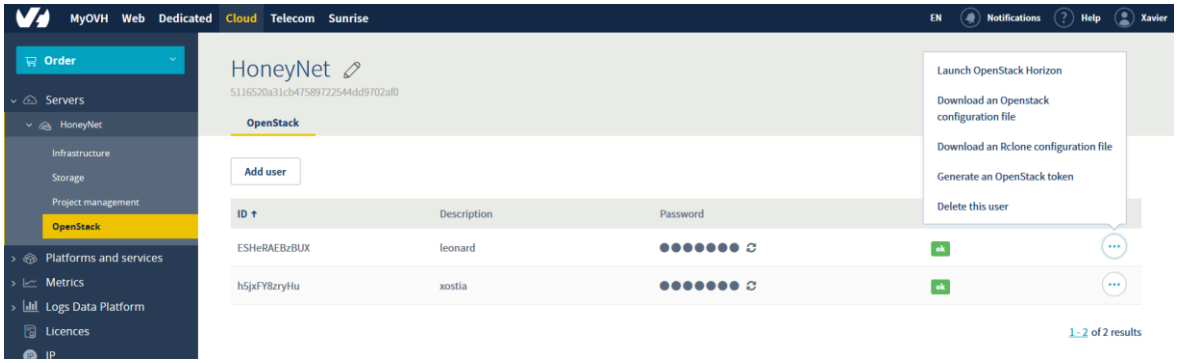
Finalment, considerant els diferents inconvenients comentats en les alternatives anteriors, s'ha optat per utilitzar el *Cloud* que ofereix el proveïdor de *OVH* [36] a la hora de desplegar les xarxes. Les màquines creades són principalment del model *S1-2* amb 2GB de RAM, 1vCore i 10GB SSD i la connexió *SSH* amb aquestes es realitza a partir de claus públiques i privades.

El *Cloud* de *OVH* està muntat sobre *OpenStack* i, per tant, s'ha tingut disponible tot un conjunt d'eines relacionades amb aquesta plataforma, com per exemple el *SDK* [37] utilitzat per a una gestió còmode des del nostre programa.

Tal com es pot veure en les següents imatges, *OVH* t'ofereix una capa de gestió per sobre de *OpenStack*. D'aquesta forma, afegeix noves funcionalitats, com la gestió del projecte en sí i la visualització dels costos del mateix, mentre que també intenta proporcionar de forma més agradable la gestió de la infraestructura i configuracions ja disponibles. En la Imatge 24 es pot veure la interfície web de *OVH* per a gestionar la infraestructura del *Cloud*, mentre que en la Imatge 25 es pot veure la interfície per a gestionar els usuaris de *OpenStack* i l'accés a aquesta plataforma.



Imatge 24. Panell de OVH per gestionar la infraestructura



Imatge 25. Panell de OVH per a gestionar els usuaris de OpenStack

De totes formes, per tal de reduir el cost en la implementació del nostre projecte, la majoria de proves unitàries s'han realitzat sobre màquines virtuals corrents en local amb *VirtualBox*. S'han creat plantilles imitant la configuració de les màquines que proveeix OVH per a més comoditat.

8.6. Llenguatges de programació

El programa principal s'ha implementat amb el llenguatge de programació *Python* [38], més concretament la versió 3.6 d'aquest. Això és degut a la simplicitat i potencia del llenguatge, junt amb la gran varietat d'eines i recursos al seu voltant. Sense anar més lluny, *OpenStack* està implementat principalment amb *Python* i el seu *SDK* oficial també, junt amb la documentació, tal com es pot veure en la Imatge 26 extreta de la documentació pels desenvolupadors [39].

OpenStack SDKs

Language	Name	Description	URL
Python	OpenStack SDK	Official Python-based library for OpenStack.	https://docs.openstack.org/openstacksdk/latest/
Python	Libcloud	A Python-based library that the Apache Foundation manages. Use it to work with multiple cloud types.	https://libcloud.readthedocs.org/en/latest/compute/drivers/openstack.html
Python	Shade	A Python-based library developed by OpenStack Infra team. Use it to operate multiple OpenStack clouds.	https://docs.openstack.org/infra/shade/
Java	jClouds	A Java-based library that the Apache Foundation manages. Use it to work with multiple cloud types.	https://jclouds.apache.org/guides/openstack/
Ruby	fog	A Ruby-based SDK. Use it to work with multiple clouds.	https://github.com/fog/fog-openstack/blob/master/docs/getting_started.md
node.js	pkgcloud	A Node.js-based SDK. Use it work with multiple clouds.	https://github.com/pkgcloud/pkgcloud/tree/master/docs/providers/openstack
PHP	php-opencloud	A PHP-based library. Use it to write PHP code that works with OpenStack clouds.	http://php-opencloud.readthedocs.org/en/latest/getting-started-with-openstack.html
.NET Framework	OpenStack SDK for Microsoft .NET	A .NET-based library. Use it to write C++ or C# code for Microsoft applications.	https://www.nuget.org/packages/openstack.net
Go	gophercloud	A go-based SDK. Use it to write Golang code that works with OpenStack clouds.	http://gophercloud.io/

Imatge 26.OpenStack opcions SDK

A més, per aquelles accions que s'han d'executar sobre les màquines de la xarxa resultant, s'han implementat *scripts* en *bash* [40], per tal d'imitar les ordres que donaria una persona davant la consola de les màquines remotes.

8.7. Estructura de carpetes i fitxers

El programa s'ha organitzat en diferents carpetes i directoris, dels quals podem resumir les seves funcionalitats de la següents manera:

- *index.py*: fitxer principal per executar per tal d'iniciar el programa, conté la lògica del menú per tal de recollir les accions que l'usuari vol realitzar.
- *env.py*: conté la configuració de la nostra aplicació i està ignorat pel control de versions, ja que conté credencials i paràmetres privats. El fitxer *env_example.py* conté la mateixa configuració però amb els valors buits, per tal que l'usuari pugui crear el fitxer *env.py* a partir d'aquest.
- *providers/*: directori que conté la lògica referent als diferents proveïdors d'infraestructura de la nostra aplicació, actualment només *OpenStack*. Proporciona la gestió pròpiament de la infraestructura, incloent creació i eliminació de màquines.
- *network/*: directori que conté la lògica de la xarxa creada i els seus servidors, representar-la a nivell de codi i aportant funcions a realitzar sobre aquesta.

- *services/*: directori que conté la lògica dels diferents serveis a posar en marxa sobre les màquines de la xarxa.
- *scripts/*: directori que conté els *scripts bash* que seran executats posteriorment en les diferents màquines de la xarxa, per tal d'instal·lar els serveis indicats o modificar les configuracions existents.
- *helpers/*: directori que conté varis fitxers amb funcions d'ajuda encarregades de realitzar diferents tasques concretes.
- *outputs/*: directori que conté els arxius creats pel propi programa, resultants de les configuracions introduïdes per l'usuari i la creació de les xarxes. Els arxius de configuració es guarden com fitxers en format *.json* [41]. Tots els fitxers d'aquest directori estan ignorats pel sistema de control de versions.
- *pkeys/*: directori on l'usuari haurà de col·locar les claus privades que posteriorment farà servir per connectar-se a les màquines de la xarxa. Aquestes claus privades hauran de tenir les seves corresponents públiques, amb el mateix nom, afegides al proveïdor de *OpenStack*. Tots els fitxers d'aquest directori estan ignorats pel sistema de control de versions.

8.8. Llibreries externes utilitzades

Per tal d'implementar les funcionalitats requerides, s'han utilitzat dues llibreries externes:

- *Paramiko* [42], per tal de gestionar les connexions, tant per *SSH* com *SFTP*, contra les diferents màquines de la xarxa. D'aquesta manera, es pot executar comandes o transferir fitxers còmodament.
- *OpenStack SDK* per tal de gestionar la infraestructura i recursos de *OpenStack* des de *Python*. En la Imatge 27 es pot veure la facilitat de connexió amb el nostre entorn *Cloud* i, un cop obtinguda la connexió, la gestió es realitza a partir de funcions ja preparades per la llibreria.

```python
def get_client(self):
    config = self.get_config_parameters()
    return openstack.connect(
        auth_url=config['auth']['url'],
        username=config['auth']['username'],
        password=config['auth']['password'],
        project_name=config['project']['name'],
        region_name=config['project']['region'],
        app_name='HoneyNet',
        app_version='2.0',
    )
```

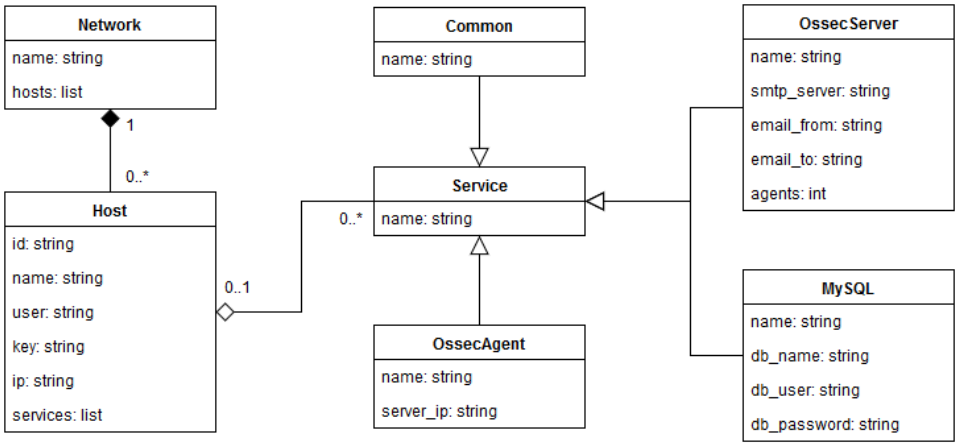
Imatge 27. Connexió contra la API de OpenStack

8.9. Representació de la xarxa

Una xarxa es representa amb diferents classes corresponents a diferents nivells de concreció. La classe més general, corresponent a tota la xarxa sencera, és la classe *Network*. Al seu torn, entre les propietats d'aquesta classe, es troba un llistat de totes les màquines que hi pertanyen, cadascuna representada per un objecte de la classe *Host*. Finalment, entre les propietats de cada màquina, hi ha un llistat de tots els serveis dels quals ha de disposar, representats com a objectes que hereten de la classe *Service*.

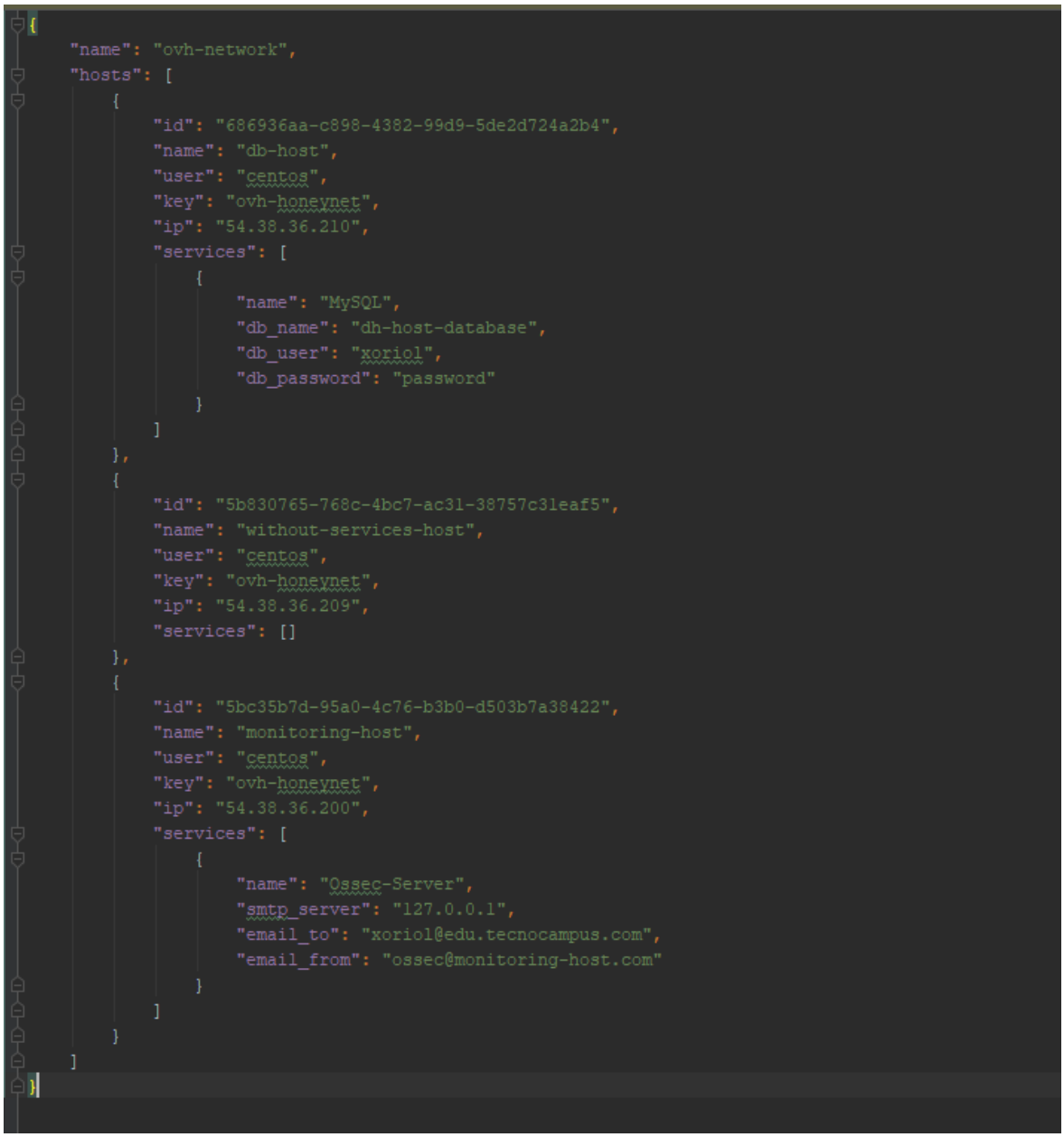
Per a demanar a l'usuari la configuració de la xarxa desitjada, s'utilitzen diverses funcions del fitxer d'ajuda *Inputs.py*, fitxer que desglossa bona part de la interacció que l'usuari ha de fer amb el programa. Un cop l'usuari ha definit les màquines amb els serveis desitjats, el programa demana crear-ne una última a mode de servidor de monitorització.

La configuració de cada una de les xarxes es guarda en un fitxer *.json*, podent ser recuperada en posteriors execucions del programa. En la Imatge 28 es pot veure el diagrama de classes que representa una xarxa.



Imatge 28. Model de classes per representar una xarxa

Per altra banda, en la Imatge 29 es pot veure un exemple de configuració de xarxa, composta per 3 servidors, en format *JSON*.



Imatge 29. Exemple de configuració JSON d'una xarxa

8.10. Proveïdors i creació de màquines

Per tal d'aconseguir flexibilitat a l'hora d'escollir el proveïdor d'infraestructura que proporciona les màquines de la xarxa, s'ha implementat dos fitxers claus:

- *Provider.py*: fitxer que implementa la classe principal que ha d'heretar qualsevol proveïdor d'infraestructura, ja que conté els mètodes bàsics per la creació i

eliminació de la xarxa. Aquells mètodes que depenen del proveïdor en qüestió retornen l'error *NotImplementedError*, esperant ser implementats en les classes filles.

- *ProviderFactory.py*: fitxer amb una funció principal encarregada d'obtenir, a partir d'una referència, la instància d'un proveïdor o un altre. Aquesta instància ha de ser l'objecte d'una classe filla de *Provider* que implementi els mètodes restants.

Cal recordar que, encara que s'ha escollit *OpenStack* com a proveïdor durant la implementació del projecte, aquest no forma part pròpiament del projecte. S'ha de donar la facilitat d'adaptació al proveïdor que més convingui a l'usuari i, aquesta solució, permet separar la implementació exclusiva del proveïdor de la resta de la lògica del programa. En la Imatge 30 es pot veure un fragment del codi de la classe *Provider*, amb els mètodes que les classes filles hauran d'implementar obligatòriament, ja que en cas contrari el programa llença un error indicant-ho.

```
class Provider(object):

    def __init__(self, name):
        self.name = name

    def set_up_host(self, host):
        raise NotImplementedError

    def remove_host(self, host):
        raise NotImplementedError

    def set_up_host_and_inform(self, host):
        print('Creating server ' + host.name + '...')
        self.set_up_host(host)
        print('Server ' + host.name + ' created with IP ' + host.ip)

    def remove_host_and_inform(self, host):
        print('Removing server ' + host.name + '...')
        self.remove_host(host)
        print('Server ' + host.name + 'removed')

    def create_network_hosts(self, network):
        threads = []
        for host in network.hosts.values():
            thread = Thread(target=self.set_up_host_and_inform, args=([host]))
            thread.start()
            threads.append(thread)
        for thread in threads:
            thread.join()
```

Imatge 30. Classe Provider amb mètodes sense implementar

En el moment de creació de les màquines de la xarxa, el programa llença un nou fil d'execució (*Thread*) per cada màquina, per tal de crear-les en paral·lel i, d'aquesta manera, mantenir un temps total acotat independentment de la quantitat.

8.11. Serveis de les màquines

Per a implementar un servei o programa a posar en marxa sobre les màquines de la xarxa, indicades per l'usuari, cal crear una nova classe que hereti de *Service*. Al seu torn, aquesta classe hereta de *Thread*, proporcionant així una execució paral·lela al programa principal. Cal recordar que la configuració d'un servei afecta a una màquina remota i, per tant, no ha d'interferir amb les tasques que s'estan realitzant a altres màquines o amb el mateix programa principal. Finalment, en cas d'haver dependències entre serveis, la classe *Thread* implementa els seus propis mecanismes de control i espera, facilitant la coordinació des del programa principal.

Tot i el comportament indicat anteriorment, els serveis d'una mateixa màquina es posen en marxa un darrera l'altra, ja que si no podria haver problemes d'accessos concurrents a fitxers o altres serveis que no ho acceptin, com per exemple el gestor de paquets.

En la Imatge 31, es pot veure la implementació de la classe *Service*. A més del mètode *run()* obligatori en tots els *Threads*, també implementa un mètode per indicar la maquina destinatària i un altre per obtenir el diccionari d'atributs a guardar com a part de la configuració de xarxa.

```python
import threading


class Service(threading.Thread):

    def __init__(self, name):
        threading.Thread.__init__(self)
        self.name = name
        self.host = None

    def run(self):
        raise NotImplementedError

    def set_target(self, host):
        self.host = host

    def get_dict(self):
        return {'name': self.name}
```

Imatge 31. Classe Service pare de tots els serveis

Per altra banda, també s'ha implementat el fitxer *ServiceFactory.py* que proporciona les funcions per obtenir la instància d'un servei a partir d'una referència a aquest o d'un diccionari de dades.

8.12. Serveis de monitorització

Per tal de realitzar la monitorització de la nostra xarxa, s'ha utilitzat el sistema de detecció d'intrusions *Ossec*. S'han creat dos serveis més, seguint la estructura de la resta de serveis, per a implementar aquesta funcionalitat. Un dels serveis correspon al servidor de monitorització, mentre que l'altre correspon a l'agent de monitorització.

El servei de monitorització no es pot escollir explícitament com a servei a la hora de configurar cada màquina, sinó que el programa demana sempre una última màquina expressament per aquest. Per altra banda, s'instal·la automàticament l'agent de monitorització en tota la resta de màquines, agent encarregat de transmetre la informació al servidor de monitorització central.

El programa demana definir la direcció origen i destí dels correus d'avís que enviarà el servidor de monitorització, davant d'algunes accions determinades en qualsevol de les màquines de la xarxa.

8.13. Connexió remota i execució de *scripts* locals

La possibilitat d'executar comandes o fitxers locals en remot és fonamental a la hora d'instal·lar i configurar els serveis desitjats en les diferents màquines. S'ha trobat alguns aspectes amb els que tenir compte a la hora de realitzar les implementacions corresponents, recolzades en la llibreria *Paramiko*.

Per un cantó, la connexió contra les màquines remotes no sempre s'aconsegueix al primer intent, sobretot just després de la creació d'aquesta. Això s'ha solucionat implementant un sistema d'intents de connexió amb un marge de temps entre aquests, tal com es pot veure amb la Imatge 32, en la funció *get_connection()*.

```python
class Host(object):

    max_connection_tries = 5
    connection_interval = 10

    def __init__(self, name, key, user='centos', identifier=None, ip=None):
        self.id = identifier
        self.name = name
        self.user = user
        self.key = key
        self.ip = ip
        self.services = {}

    def get_connection(self):
        client = paramiko.SSHClient()
        client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        key_path = Files.get_private_key_path(self.key)
        for tries in range(self.max_connection_tries):
            try:
                client.connect(self.ip, port=22, username=self.user, key_filename=key_path)
                return client
            except Exception as error:
                print("Attempt "+str(tries+1)+" of "+str(self.max_connection_tries)+": Cannot connect to "+self.name)
                print("Error message: " + str(error))
                time.sleep(self.connection_interval)
        raise paramiko.SSHException

    def add_service(self, service):
        name = service.name
```

Imatge 32. Classe Host amb funció per obtenir una connexió

Un altre aspecte a considerar és la diferència de codificació entre diferents sistemes operatius. Això provoca que un fitxer, creat des d'una màquina *Windows*, no es pugui executar directament en una màquina *Linux*, ja que retornaria un error en el salt de línia, tal com es pot veure en la Imatge 33.

```
[centos@localhost ~]$ /tmp/mysql_set_up.sh
-bash: /tmp/mysql_set_up.sh: /bin/bash^M: bad interpreter: No such file or directory
[centos@localhost ~]$
```

Imatge 33. Error execució fitxer Windows en Linux

Per tal de poder executar el fitxer, es necessari netejar els salts de línia abans. Això es pot realitzar fàcilment gràcies a l'editor *sed* [43], disponible per defecte en molts sistemes *Linux*. La comanda exacta és "*sed -i -e 's/\r$//' [file]*".

A mode de resum, en la Imatge 34 es poden veure els diferents passos necessaris per a l'execució d'un *script* en una màquina remota:

- S'obre una connexió contra el servidor objectiu.
- Es copia el *script* en la carpeta temporal del servidor.
- Es canvien el permisos per tal que el *script* sigui executable.
- Es netegen els finals de línia per eliminar incompetències amb altres sistemes operatius.

- S'executa el *script* amb els arguments indicats com a paràmetres.
- S'elimina el fitxer del directori de temporals.
- Es tanca la connexió.

```
def execute_local_script(script_name, connection, arguments=[]):
    local_path = Files.get_script_path(script_name)
    destination_path = '/tmp/' + script_name
    sftp = connection.open_sftp()
    sftp.put(local_path, destination_path)
    sftp.chmod(destination_path, stat.S_IRWXU)
    connection.exec_command("sed -i -e 's/\\r$//' " + destination_path)
    command = destination_path + ' ' + ' '.join(arguments)
    output = execute_command_and_get_output(command, connection)
    sftp.remove(destination_path)
    sftp.close()
    return output
```

Imatge 34. Funció per executar un script local en una màquina remota

Finalment, un altre inconvenient és la dificultat de depurar els resultat obtinguts de la màquina remota, ja que qualsevol sortida és un text, sense discriminació del seu contingut. La llibreria de *Paramiko* permet diferenciar una sortida estàndard en cas correcte del que no ho és, però no permet diferenciar una advertència d'un error, per exemple. Així doncs, per fer una bona depuració, s'hauria d'implementar manualment tot el control per cada un dels diferents serveis.

8.14. Desplegament d'una xarxa

A la hora de desplegar una xarxa, el programa ha de realitzar les següents accions:

1. Demana al proveïdor d'infraestructura la quantitat de màquines definides en la configuració de la xarxa que es vol desplegar. Aquesta acció es realitza en paral·lel per cada una de les màquines.
2. Actualitza la configuració local de les màquines per tal de reflectir els identificadors i direccions IP proporcionats per a cada una de les màquines remotes creades.
3. Actualitza els servidors i instal·la algunes utilitats bàsiques que seran necessàries per a la resta de serveis. Aquesta acció es realitza en paral·lel per cada una de les màquines.

4. Instal·la els serveis definits per l'usuari, un darrera l'altre, en cada màquina de la xarxa. També inclou el servei de monitorització en la màquina corresponent. Aquesta acció es realitza en paral·lel per cada una de les màquines.
5. Instal·la els agents de monitorització en totes les màquines de la xarxa, excepte aquella que actua com a servidor de monitorització central. Aquesta acció es realitza en paral·lel per cada una de les màquines.
6. Afegeix cada un dels agents de monitorització al servidor central, configurant així la transmissió d'informació des de l'agent fins la servidor. El servidor envia un correu per cada agent afegit correctament.

Tal com es pot veure en la descripció de les tasques anteriors, la majoria d'aquestes es realitzen de forma paral·lela per cada una de les màquines. Això comporta que el temps total de desplegament augmenti mínimament per cada màquina afegida, proporcionant una gran escalabilitat al programa.

8.15. Possibles accions a realitzar

Les interaccions amb l'usuari són, en gran part, eleccions entre un llistat d'opcions, casos en que les opcions estan numerades i l'usuari ha d'indicar el número. Això s'ha fet així perquè és més còmode i dona menys marge d'error un número que una frase.

En el cas concret del menú principal, s'han extret les opcions en un fitxer a part anomenat *actions.py,* tal com es pot veure en la Imatge 35.

```
CREATE_NEW_NETWORK_CONFIGURATION = 1
LOAD_EXISTING_NETWORK_CONFIGURATION = 2
DEPLOY_NETWORK = 3
DESTROY_NETWORK = 4
REMOVE_NETWORK_CONFIGURATION = 5
EXIT = 9

options = {
    CREATE_NEW_NETWORK_CONFIGURATION: {
        'label': 'CREATE a new network CONFIGURATION',
    },
    LOAD_EXISTING_NETWORK_CONFIGURATION: {
        'label': 'LOAD an existing network CONFIGURATION (must be in outputs folder)',
    },
    DEPLOY_NETWORK: {
        'label': 'DEPLOY a NETWORK based on the configuration'
    },
    DESTROY_NETWORK: {
        'label': 'DESTROY loaded NETWORK',
    },
    REMOVE_NETWORK_CONFIGURATION: {
        'label': 'REMOVE CONFIGURATION file'
    },
    EXIT: {
        'label': 'EXIT the program. Bye bye.'
    }
}
```

Imatge 35. Opcions del menú principal en el fitxer actions.py

Les diferents accions que l'usuari pot realitzar amb el programa són:

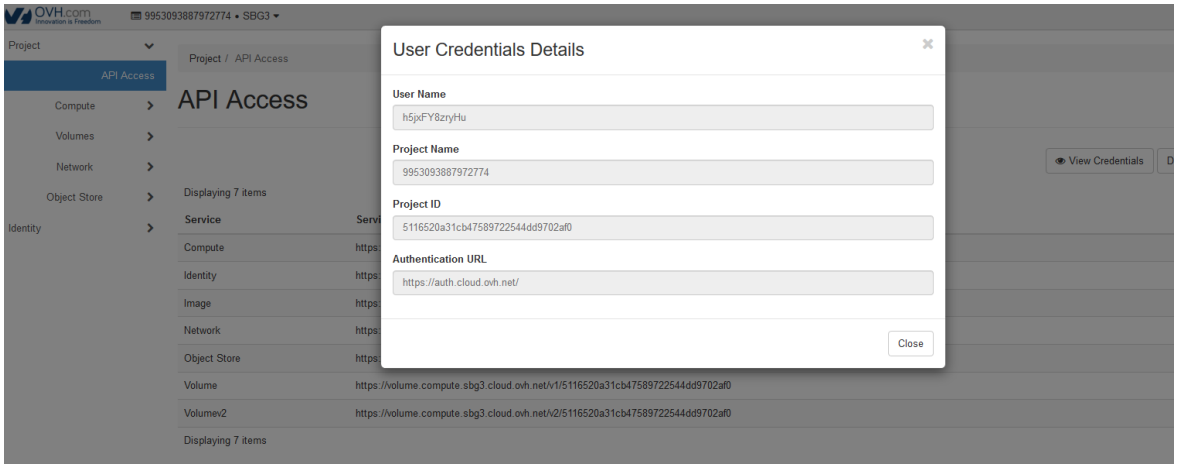
- Crear una nova configuració de xarxa.
- Carregar una configuració de xarxa ja existent.
- Desplegar la xarxa remota.
- Eliminar les màquines de la xarxa.
- Eliminar la configuració local de la xarxa.
- Sortir del programa.

En ser una xarxa pensada per ser atacada i recopilar informació, a mode d'esquer, s'ha de contemplar el cas de voler eliminar i tornar-la a aixecar després d'un atac amb èxit que provoqui un mal funcionament. En conseqüència, la configuració de la xarxa i el seu desplegament, així com l'eliminació de les màquines i l'eliminació de la configuració, han de ser accions independents.

8.16. Configuracions prèvies a l'execució

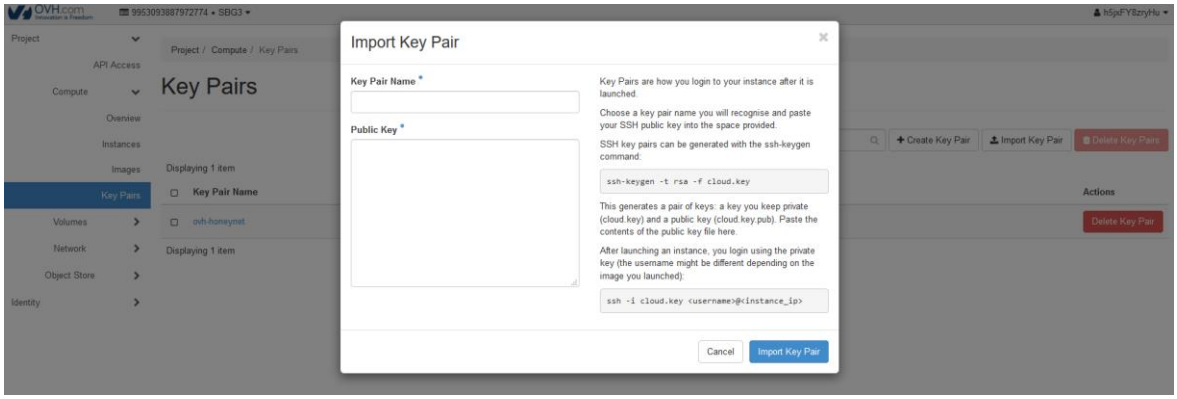
Abans d'utilitzar el programa per a desplegar xarxes remotes, s'ha de realitzar algunes configuracions prèvies, tant en local com en el *OpenStack* remot.

El primer que s'ha de fer és crear el fitxer *env.py* a partir del fitxer *env_example.py* i omplir amb els valors de la configuració corresponents. En el cas de *OpenStack*, es poden trobar les credencials d'accés a la *API* en l'apartat *API Access* i, un cop en aquest, *View Credentials*, tal com es pot veure en la Imatge 36.



Imatge 36. Credencials i accés a la API de OpenStack

Per altra banda, també cal crear una parella de claus pública/privada per tal d'accedir a les màquines via *SSH*. Respecte la clau privada, aquesta ha d'estar en el directori *pkeys/* de la aplicació i amb el mateix nom de fitxer que la clau pública del servidor. Des de *OpenStack*, es poden gestionar les claus en l'apartat *Key Pairs*, dins del punt de menú *Compute*. En la Imatge 37 es pot veure la interfície per importar una clau pública, dins l'apartat comentat anteriorment.



Imatge 37. Importació de clau pública al panell de OpenStack

9. Execució de la aplicació

Al llarg d'aquest apartat es mostrarà la execució del programa per les diferents accions que l'usuari pot realitzar.

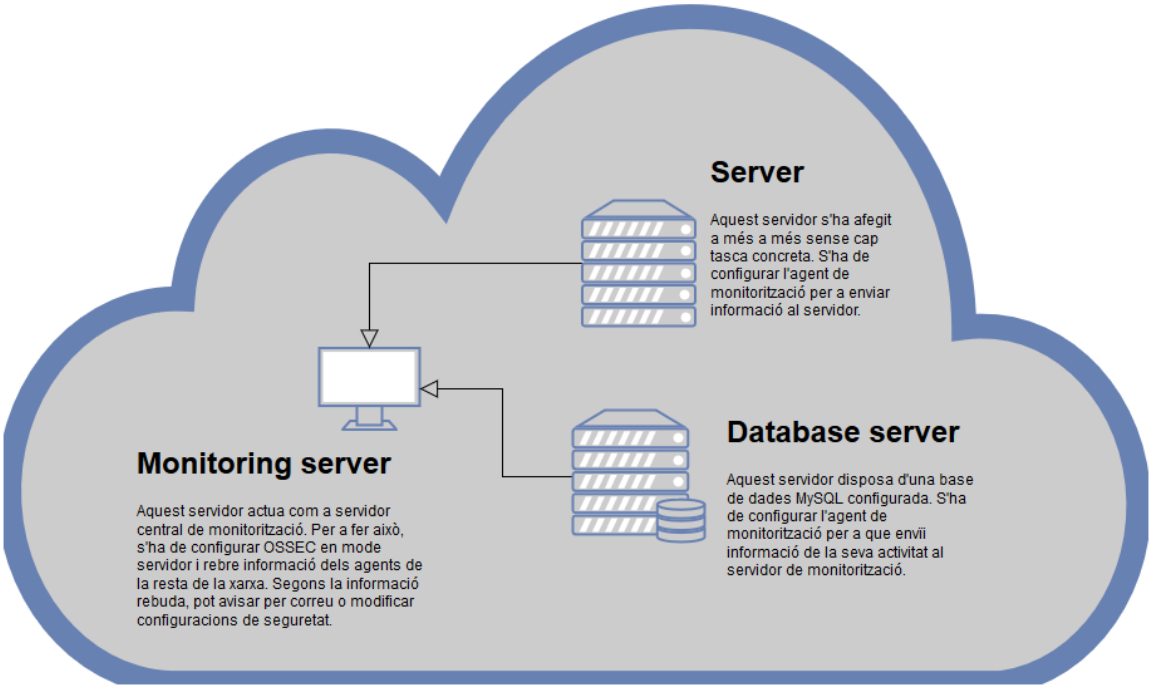
Un cop executat el programa, l'usuari es troba davant del menú principal, mostrat en la Imatge 38. Sempre que s'acabi d'executar una acció, l'usuari tornarà en el menú principal, mentre no escolli la opció per a sortir del programa.

```
PS X:\MisCosas\Universitat\Tecnocampus\TFG\HoneyNets> python index.py
Type the number of action to execute:
        1. CREATE a new network CONFIGURATION
        2. LOAD an existing network CONFIGURATION (must be in outputs folder)
        3. DEPLOY a NETWORK based on the configuration
        4. DESTROY loaded NETWORK
        5. REMOVE CONFIGURATION file
        9. EXIT the program. Bye bye.
Your choice:
```

Imatge 38. Menú principal del programa.

9.1. Definició de la xarxa

El primer que cal fer per a crear una xarxa és pensar amb la configuració desitjada. A mode d'exemple, s'ha utilitzat una topologia amb 3 servidors: un servidor de base de dades, un servidor sense serveis explícits i el servidor de monitorització. Per a més claredat, consultar la Imatge 39.



Imatge 39. Exemple de topologia de xarxa

9.2. Crear configuració de xarxa

A la hora de crear una configuració de xarxa, el programa pregunta sobre diferents paràmetres. En la Imatge 40, el programa sol·licita un nom per a la xarxa i pregunta pels paràmetres del primer servidor. En la Imatge 41, el programa pregunta pels serveis que es volen configurar en la màquina, fins que l'usuari indiqui que no vol afegir-ne més. Finalment, en la Imatge 42, l'usuari indica que no vol més màquines i, llavors, el programa sol·licita una última a mode de servidor de monitorització.

```
Type a name for the network: demo-network
Shall I add another host to the network (Yes/No)? y
Type a name for the server: db-host
Choose one of the available keys:
        0. ovh-honeynet
Type the key identification number: 0
Shall I add another service to this host (Yes/No)? y
Choose one of the available services:
```

Imatge 40. Nom de la xarxa i primera màquina

```
Shall I add another service to this host (Yes/No)? y
Choose one of the available services:
        0. MySQL
        1. RETURN BACK
Type the option identification number: 0
Type the database name: demo_database
Type a username for connecting to database: demo_user
Type a password for that user: demo_password
Shall I add another service to this host (Yes/No)? n
Shall I add another host to the network (Yes/No)? y
```

Imatge 41. Afegir serveis a la màquina, en aquest cas MySQL

```
Shall I add another host to the network (Yes/No)? n
Let's configure monitoring host:
Type a name for the server: monitor_server
Choose one of the available keys:
       0. ovh-honeynet
Type the key identification number: 0
Let's configure monitor service:
Type a destination email address to receive security notifications: xoriol@edu.tecnocampus.cat
Type an origin email address: ossec@demo_network.com
Configuration file saved in outputs/demo-network.json


Network configuration loaded: demo-network
Type the number of action to execute:
       1. CREATE a new network CONFIGURATION
       2. LOAD an existing network CONFIGURATION (must be in outputs folder)
```

Imatge 42. Servidor de monitorització i guardat de la configuració

Tal com es pot veure en la Imatge 42, el programa guarda un fitxer amb la configuració resultant. El nom del fitxer és igual al nom de la xarxa.

9.3. Carregar una configuració existent

En la Imatge 43, es pot veure com el programa mostra les configuracions ja existents i demana a l'usuari que indiqui quina vol carregar.

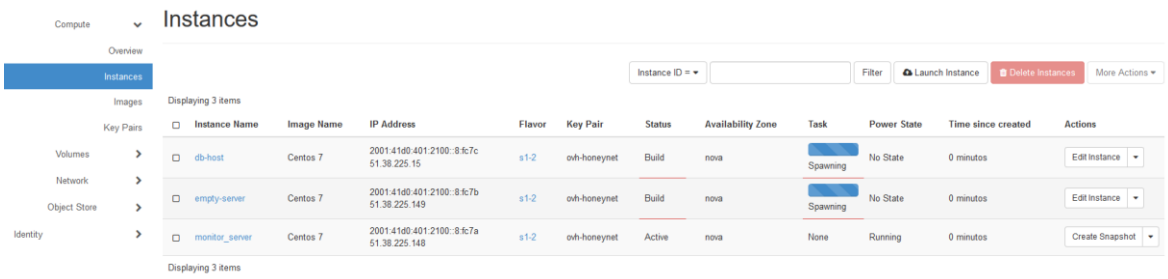
```
Choose one of the available networks:
        0. demo-network
        1. net
        2. network-local
        3. ovh-net2
        4. ovh-network
Type the network identification number: 0
Network demo-network loaded.


Network configuration loaded: demo-network
Type the number of action to execute:
        1. CREATE a new network CONFIGURATION
        2. LOAD an existing network CONFIGURATION (must be in outputs folder)
```

Imatge 43. Carrega d'una xarxa ja existent

9.4. Desplegar la xarxa carregada

Durant el desplegament de la xarxa, no hi ha cap interacció amb l'usuari. El programa va realitzant les diferents tasques ja comentades en apartats anteriors i avisant per pantalla de l'avanç del procés. Un cop acaba, la xarxa ja esta muntada i tots els serveis, incloent la monitorització, estan configurats i en marxa.

En la Imatge 44 es pot veure el panell de *OpenStack*, mostrant com les diferents màquines de la xarxa s'estan creant, mentre que en la Imatge 45 es pot veure l'exemple de la sortida del programa per al desplegament de la xarxa.



Imatge 44. Creació de màquines OpenStack

```
Creating server db-host...
Creating server empty-server...
Creating server monitor_server...
Server monitor_server created with IP 51.38.225.148
Server db-host created with IP 51.38.225.15
Server empty-server created with IP 51.38.225.149
All hosts created.
Configuration file outputs/demo-network.json updated
Installing services...
Attempt 1 of 5: Cannot connect to monitor_server
Error message: [Errno None] Unable to connect to port 22 on 51.38.225.148
Upgrading and installing common utilities on host monitor_server...
Attempt 1 of 5: Cannot connect to db-host
Attempt 1 of 5: Cannot connect to empty-server
Error message: [WinError 10060] A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected h
Error message: [WinError 10060] A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected h
Upgrading and installing common utilities on host empty-server...
Upgrading and installing common utilities on host db-host...
Common utilities installed on host monitor_server
Common utilities installed on host empty-server
Common utilities installed on host db-host
Installing service MySQL on host db-host...
Installing service Ossec-Server on host monitor_server...
Service Ossec-Server installed on host monitor_server
Service MySQL installed on host db-host
All services installed
Installing monitoring agents...
Installing service Ossec-Agent on host db-host...
Installing service Ossec-Agent on host empty-server...
Service Ossec-Agent installed on host empty-server
Service Ossec-Agent installed on host db-host
Adding agent for host db-host on server monitor_server...
Added agent for host db-host on server monitor_server
Importing agent monitoring key on host db-host...
Agent key imported on host db-host
Adding agent for host empty-server on server monitor_server...
Added agent for host empty-server on server monitor_server
Importing agent monitoring key on host empty-server...
Agent key imported on host empty-server
Monitoring agents installed
ALL NETWORK AND SERVICES DEPLOYED !!! ENJOY IT !!!
```

Imatge 45. Sortida per consola del desplegament de la xarxa

Per altra banda, en la Imatge 46, la Imatge 47 i la Imatge 48, es pot veure com els serveis corresponents s'han instal·lat i configurat correctament.

```
xaviorse@XOStia:~$ ssh centos@51.38.225.149 -i .ssh/ovh-honeynet
The authenticity of host '51.38.225.149 (51.38.225.149)' can't be established.
ECDSA key fingerprint is SHA256:KLPvaDg1IK9LdxeeANbszqnUQv0foymaXjFudyEEI9w.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '51.38.225.149' (ECDSA) to the list of known hosts.
[centos@empty-server ~]$ hostname -I
51.38.225.149
[centos@empty-server ~]$
```

Imatge 46. Connexió per SSH correcte

```
xaviorse@XOStia:~$ mysql -u demo_user -h 51.38.225.15 -pdemo_password
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 6
Server version: 5.5.56-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

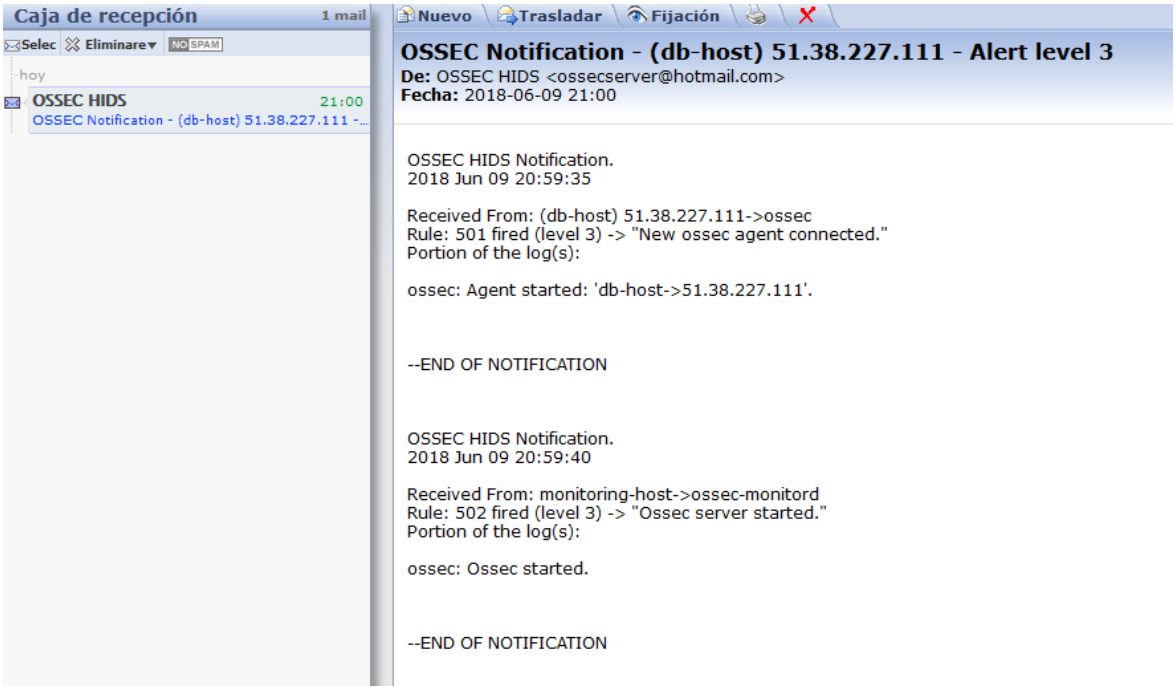
Imatge 47. Connexió amb la base de dades correcte

```
xaviorse@XOStia:~$ ssh centos@51.38.225.148 -i .ssh/ovh-honeynet
The authenticity of host '51.38.225.148 (51.38.225.148)' can't be established.
ECDSA key fingerprint is SHA256:KLPvaDg1IK9LdxeeANbszqnUQv0foymaXjFudyEEI9w.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '51.38.225.148' (ECDSA) to the list of known hosts.
[centos@monitor-server ~]$ sudo /var/ossec/bin/ossec-control status
ossec-monitord is running...
ossec-logcollector is running...
ossec-remoted is running...
ossec-syscheckd is running...
ossec-analysisd is running...
ossec-maild is running...
ossec-execd is running...
[centos@monitor-server ~]$
```

Imatge 48. Servei de monitorització correcte

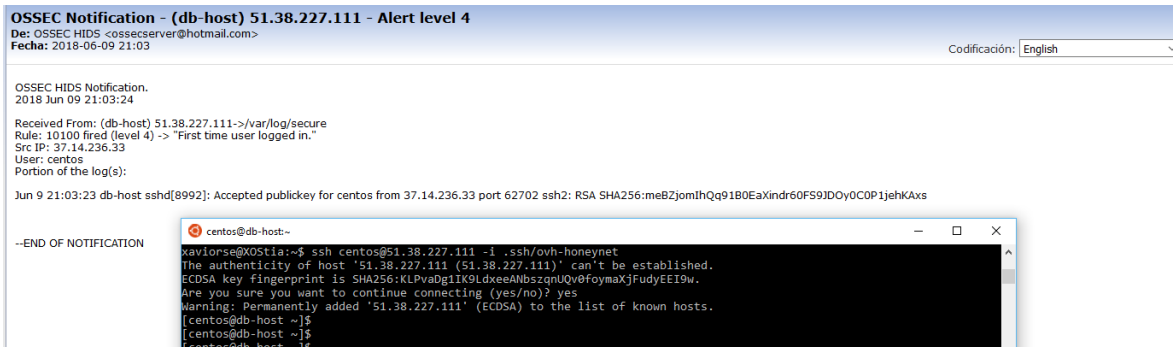
9.5. Monitorització

Un cop la xarxa ja ha estat desplegada, els primers missatges que es reben confirmen que, tant el servidor com els agents de monitorització, s'han iniciat correctament. En la Imatge 49 es pot veure un exemple de correu per a aquest cas.



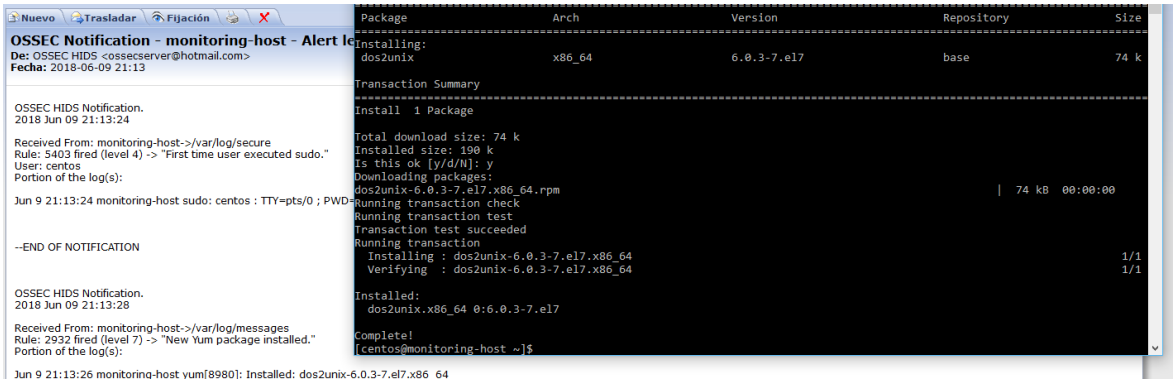
Imatge 49. Correu conforme s'ha configurat s'ha posat en marxa un nou agent

Per altra banda, en cas d'haver un inici de sessió en qualsevol de les màquines de la xarxa, el servidor de monitorització també envia un correu notificant-ho. Aquest cas es pot veure en la Imatge 50.



Imatge 50. Identificació i avís d'inici de sessió en un agent

Una altra altivitat crítica i que, per tant, ha d'estar controlada pel servidor és la instal·lació de paquets en qualsevol de les màquines. La forma d'actuar per defecte és la mateixa que en els casos anteriors, tal com es pot comprovar en la Imatge 51.



Imatge 51. Identificació i avís per instal·lació de paquets

Tot i que els casos anteriors són situacions freqüents i bàsiques per demostrar el correcte funcionament del sistema, *Ossec* disposa de moltes altres regles per a informar, i inclús modificar el sistema, davant de diferents situacions o atacs. A més, proporciona els mecanismes per a crear regles personalitzades de detecció i actuació, oferint així unes possibilitats de monitorització molt amplies i flexibles.

Cal tenir en compte que les regles d'actuació estan configurades únicament en el servidor central i, automàticament, s'apliquen també sobre tots els agents. En conseqüència, canviar el comportament de la monitorització en tota la xarxa és una tasca ràpida i simple, sobretot si es disposa prèviament d'un fitxer amb les regles personalitzades que es desitgen.

9.6. Eliminar la xarxa remota

Aquesta opció elimina les màquines remotes de la xarxa carregada, la *Imatge* 52 és un exemple de la resposta del programa.

```
Removing the network demo-network...
Removing server db-host...
Removing server empty-server...
Removing server monitor_server...
Server monitor_serverremoved
Server db-hostremoved
Server empty-serverremoved
Network demo-network deleted.
```

Imatge 52. Màquines remotes eliminades

9.7. Eliminar l'arxiu local de configuració

Aquesta opció elimina la configuració local carregada, indicant el fitxer concret, tal com mostra la *Imatge* 53.

```
Network configuration loaded: demo-network
Type the number of action to execute:
        1. CREATE a new network CONFIGURATION
        2. LOAD an existing network CONFIGURATION (must be in outputs folder)
        3. DEPLOY a NETWORK based on the configuration
        4. DESTROY loaded NETWORK
        5. REMOVE CONFIGURATION file
        9. EXIT the program. Bye bye.
Your choice: 5

Removed configuration file outputs/demo-network.json
```

Imatge 53. Eliminació de la configuració de xarxa local

10. Anàlisi de resultats, conclusions i possibles ampliacions.

Finalment s'ha obtingut una aplicació capaç de crear configuracions de xarxa de forma interactiva, guardar-les, carregar-les de nou, desplegar-les i eliminar-les. La aplicació és capaç de crear totes les màquines amb els seus respectius serveis i, a més, afegir un sistema de monitorització centralitzat en un últim servidor, instal·lant agents en la resta. Així doncs, es pot dir que s'ha aconseguit un primer programa capaç d'automatitzar la creació de *HoneyNets* i, per tant, el projecte ha complert el seu objectiu principal.

Tot i això, els inconvenients trobats a la hora d'aconseguir la infraestructura han endarrerit molt la implementació del projecte i, en conseqüència, la varietat de serveis disponibles és molt baixa, a més d'haver requeriments definits inicialment que no s'han assolit.

Respecte el temps de desplegament, aquest es veu poc influenciat per la quantitat de màquines que té la xarxa, mentre que depèn molt de la quantitat d'actualitzacions disponibles. Així doncs, la antiguitat de la imatge a partir de la qual es munta el sistema operatiu condiciona enormement el temps de desplegament de la xarxa, podent ser la fase d'actualització la més duradora i amb diferència.

En tractar-se d'un programa que afecta a màquines remotes i serveis mantinguts per companyies externes, una actualització en aquests serveis podria arribar a causar l'incorrecte funcionament del *script* relacionat. En cas d'error, s'hauria de comprovar el funcionament del servei de forma manual i adaptar el *script* corresponent a la nova forma de configuració. Detectar aquests errors és difícil, però la conseqüència és mínima ja que no són màquines de producció i el propi programa et permet eliminar i tornar a crear la xarxa molt fàcilment un cop solucionat. Cal afegir que, durant el temps de desenvolupament d'aquest projecte, no m'he trobat amb el cas d'una actualització crítica.

Entre les possibles millores es poden incloure aquells requeriments funcionals que es van definir inicialment i no s'han implementat:

- Permetre definir el port SSH, si no es vol per defecte, abans de la creació de la màquina en qüestió.
- Poder definir un nou usuari en el moment de la configuració de cada màquina.
- Proporcionar una via per carregar les imatges pròpies que es vulguin utilitzar com a punt de partida en la configuració de les màquines.
- Permetre la modificació de la xarxa un cop aquesta ja està creada.

A més, es poden implementar més serveis i proveïdors per tal de proporcionar més flexibilitat a l'usuari que utilitzarà l'aplicació.

11. Bibliografia

[1] *VMware* [en línia] [consulta: 18 de desembre del 2017]. Disponible a https://www.vmware.com/es.html

[2] *VirtualBox* [en línia] [consulta: 18 de desembre del 2017]. Disponible a https://www.virtualbox.org/

[3] *Microsoft Hyper-V* [en línia] [consulta: 18 de desembre del 2017]. Disponible a https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/

[4] *Kernel Virtual Machine* [en línia] [consulta: 18 de desembre del 2017]. Disponible a http://www.linux-kvm.org/page/Main_Page

[5] *Docker* [en línia] [consulta: 18 de desembre del 2017]. Disponible a https://www.docker.com/

[6] *Containers vs VMs* [en línia] [consulta: 18 de desembre del 2017]. Disponible a https://www.sdxcentral.com/cloud/containers/definitions/containers-vs-vms/

[7] *Amazon Web Services* [en línia] [consulta: 18 de desembre del 2017]. Disponible a https://aws.amazon.com/es/

[8] *Microsoft Azure* [en línia] [consulta: 18 de desembre del 2017]. Disponible a https://azure.microsoft.com/en-us/

[9] *DigitalOcean* [en línia] [consulta: 18 de desembre del 2017]. Disponible a https://www.digitalocean.com/

[10] *OpenStack* [en línia] [consulta: 8 d'abril del 2018]. Disponible a https://www.openstack.org/

[11] *Debian* [en línia] [consulta: 20 de desembre del 2017]. Disponible a https://www.debian.org/

[12] *RedHat* [en línia] [consulta: 20 de desembre del 2017]. Disponible a https://www.redhat.com/en

[13] *Suse* [en línia] [consulta: 20 de desembre del 2017]. Disponible a https://www.suse.com/es-es/

[14] *Ossec* [en línia] [consulta: 18 de desembre del 2017]. Disponible a https://ossec.github.io/

[15] *Snort* [en línia] [consulta: 18 de desembre del 2017]. Disponible a https://www.snort.org/

[16] *Sysdig* [en línia] [consulta: 18 de desembre del 2017]. Disponible a https://www.sysdig.org/

[17] *Docker Compose* [en línia] [consulta: 20 de desembre del 2017]. Disponible a https://docs.docker.com/compose/

[18] *ESXi* [en línia] [consulta: 8 d'abril del 2018]. Disponible a https://www.vmware.com/products/esxi-and-esx.html

[19] *vCenter* [en línia] [consulta: 8 d'abril del 2018]. Disponible a https://www.vmware.com/products/vcenter-server.html

[20] *CentOS* [en línia] [consulta: 20 de desembre del 2017]. Disponible a https://www.centos.org/

[21] *Ubuntu* [en línia] [consulta: 20 de desembre del 2017]. Disponible a https://www.ubuntu.com/

[22] *Kippo* [en línia] [consulta: 20 de desembre del 2017]. Disponible a https://github.com/desaster/kippo

[23] *PID1* [en línia] [consulta: 8 d'abril del 2018]. Disponible a https://blog.phusion.nl/2015/01/20/docker-and-the-pid-1-zombie-reaping-problem/

[24] *Systemctl* [en línia] [consulta: 8 d'abril del 2018]. Disponible a https://www.unix.com/man-page/centos/1/systemctl/

[25] *Service* [en línia] [consulta: 8 d'abril del 2018]. Disponible a https://www.unix.com/man-page/centos/8/service/

[26] *Initscripts* [en línia] [consulta: 8 d'abril del 2018]. Disponible a https://www.centos.org/docs/5/html/5.4/technical-notes/initscripts.html

[27] *Centos Container* [en línia] [consulta: 8 d'abril del 2018]. Disponible a https://hub.docker.com/_/centos/

[28] *Systemd* [en línia] [consulta: 8 d'abril del 2018]. Disponible a http://man7.org/linux/man-pages/man1/init.1.html

[29] Canonical [en línia] [consulta: 8 d'abril del 2018]. Disponible a https://www.canonical.com/

[30] *Docker and Canonical* [en línia] [consulta: 8 d'abril del 2018]. Disponible a https://www.docker.com/docker-news-and-press/docker-and-canonical-partner-cs-docker-engine-millions-ubuntu-users

[31] *SSH* [en línia] [consulta: 9 d'abril del 2018]. Disponible en https://www.ssh.com/ssh/

[32] DevStack [en línia] [consulta: 9 d'abril del 2018]. Disponible en https://docs.openstack.org/devstack/latest/

[33] *API vCenter* [en línia] [consulta: 9 d'abril del 2018]. Disponible a https://code.vmware.com/apis/62/vcenter-management#/

[34] *VyOS* [en línia] [consulta: 9 d'abril del 2018]. Disponible a https://vyos.io/es/

[35] *FusionSphere* [en línia] [consulta: 11 d'abril del 2018]. Disponible a http://e.huawei.com/es/products/cloud-computing-dc/cloud-computing/fusionsphere/fusionsphere

[36] *OVH* [en línia] [consulta: 27 de maig del 2018]. Disponible a https://www.ovh.es/public-cloud/instancias/

[37] *OpenStack SDK* [en línia] [consulta: 27 de maig del 2018]. Disponible a https://docs.openstack.org/openstacksdk/latest/index.html#

[38] *Python* [en línia] [consulta: 27 de maig del 2018]. Disponible a https://www.python.org/

[39] *OpenStack* pels desenvolupadors [en línia] [consulta: 27 de maig del 2018]. Disponible a https://developer.openstack.org/

[40] *Bash* [en línia] [consulta: 27 de maig del 2018]. Disponible a https://www.gnu.org/software/bash/

[41] *JSON* [en línia] [consulta: 28 de maig del 2018]. Disponible a https://www.json.org/

[42] *Paramiko* [en línia] [consulta: 28 de maig del 2018]. Disponible a http://www.paramiko.org/

[43] *Sed Stream Editor* [en línia] [consulta: 29 de maig del 2018]. Disponible a https://www.gnu.org/software/sed/manual/sed.html

[44] Álvarez Martín, Carlos; González Pérez, Pablo: *Hardening de servidores GNU / Linux*. Tercera edició. ISBN: 978-84-617-1518-3.

[45] García Rambla, Juan Luis; Alonso, Chema; González, Pablo: *Ataques en redes de datos IPv4 e Ipv6*. Tercera edició. ISBN: 978-84-617-9278-8.