Pekka Hapuli

AVOIMEN LÄHDEKOODIN PALVELINVIRTUALISOINTIRATKAI SU OPEN SOURCE ARCHIVE -HANKKEEN TARPEISIIN

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma

Joulukuu 2014

MAMK
University of Applied Sciences

KUVAILULEHTI

MAMK University of Applied Sciences	**Opinnäytetyön päivämäärä** 05.12.2014
Tekijä(t) Pekka Hapuli	**Koulutusohjelma ja suuntautuminen** Tietojenkäsittelyn koulutusohjelma Tradenomi

Nimeke

Avoimen lähdekoodin palvelinvirtualisointiratkaisu Open Source Archive-hankkeen tarpeisiin

Tiivistelmä

Opinnäytetyöni on tehty OSA-hankkeelle (Open Source Archive), joka tarvitsi palvelimilleen virtualisointialustan.

Virtualisointi ja pilvipalvelut ovat vallanneet jo tovin markkinoita, ja vähitellen ne alkavat vallata myös pienempiä konesaleja. Työssäni tutustun virtualisoinnin historiaan, eli siihen mistä kaikki on lähtenyt ja miten eri tekniikat ovat syntyneet. Mitä vaihtoehtoja ja eritoten vapaan lähdekoodin vaihtoehtoja on tarjolla. Selvitin työssäni myös, mitä eroa on rauta-avusteisella virtualisoinnilla ja käyttöjärjestelmä-avusteisella virtualisoinnilla, unohtamatta kapselointia.

Kävin työssäni läpi pilvipalvelun ja pilvilaskennan määritelmät, erilaiset pilvityypit ja niiden ominaispiirteet. Työssäni asensin KVM-pohjaisen Libvirt-rajapintaa käyttävän OpenNebula -pilviympäristön. Työhöni liittyy myös vahvasti levyklusterointi.

Lopputuloksena syntyi järjestelmä, joka ei ollut täysin toimiva eikä näin ollen tuotantokäyttöön soveltuva. Työstä saatiin arvokasta kokemusta jotta pystyttiin valitsemaan uusi suunta ja uusi ympäristö.

Asiasanat (avainsanat)
Virtualisointi, KVM, QEMU, OpenNebula, Sunstone

Sivumäärä	**Kieli**	**URN**
35	Suomi	

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi Matti Juutilainen	**Opinnäytetyön toimeksiantaja** OSA-hanke

DESCRIPTION

MAMK University of Applied Sciences	**Date of the bachelor's thesis** 05 December 2014
Author(s) Pekka Hapuli	**Degree programme and option** Business information technology

Name of the bachelor's thesis

Open Source datacenter virtualization solution for Open Source Archived

Abstract

My thesis was made for the OSA (Open Source Arvice) project that needed a virtualization platform for its servers.

Virtualization and cloud platforms have been on market for a while, but now even medium and small sized data centers have wake up to the opportunities that these techniques could bring to them. In this thesis I have familiarized myself with the history of virtualization, techniques and tools and in particular what open source had to offer. I also got a little bit in to cloud computing and typical clouds.

The practical part dealt with the world of open source virtualization and proprietary hardware. I chose Opennebula as my main tool and chose the best possible open source parts to make it work. Even tough I used all the possible information in the practical part it turned out that Opennebula was not as ready and compatible with the fibre channel as I was led to believe.

In the end we had a virtualization platform that was not fully functional and therefore not ready to be used in production. I learned a lot and that information was used to choose a more stable and suitable environment.

Subject headings, (keywords)

Virtualization, KVM, QEMU, OpenNebula, Sunstone

Pages	**Language**	**URN**
35	Finnish	

Remarks, notes on appendices

Tutor	**Bachelor's thesis assigned by**
Matti Juutilainen	OSA Project

SISÄLTÖ

1 JOHDANTO

Opinnäytetyöni on tehty Open Source Archive -hankkeen parissa, jossa olin mukana harjoittelijana pystyttämässä virtualisointiympäristöä. (Open Source Archive) OSA-hanke tutkii ja kehittää avoimella lähdekoodilla toteutettuja digitaalisia arkistopalveluita ja sähköisen tiedon säilyttämisen työkaluja. Käsittelen tekstissä pääasiassa palvelinsaleihin soveltuvia avoimia ratkaisuja ja suljen muut ratkaisut ulkopuolelle vaikka niistä välillä saatankin jotain mainita. Käsittelen opinnäytetyössäni myös kevyesti käyttöjärjestelmien ytimien toimintaa, koska ilman kyseisen asian käsittelyä ei virtualisointia voi oikeasti ymmärtää. Esimerkiksi Andrew S. Tanenbaumin kirjoittama Operating Systems - Design and Implementation kirja auttaa ymmärtämään käyttöjärjestelmän ja ytimen toimintaa. Olen pyrkinyt tekstissäni selvittämään asiat mahdollisimman selkeästi sekä niin loogisessa järjestyksessä kuin mahdollista. Tästä huolimatta osa virtualisointiin liittyvistä asioista ei varmastikaan avaudu kaikille.

Virtualisointi ja pilvipalvelut ovat vallanneet jo tovin markkinoita ja vähitellen ne alkavat vallata myös pienempiä konesaleja. Virtualisoinnin alunperin kehittänyt Jim Rymarczyk sanoi vuonna 2011 haastetussa, että siitä on vahvasti tulossa alan normi. Syy on palvelinresurssien säästäminen sillä virtualisoidussa palvelinympäristössä pystytään ajamaan useampaa käyttöjärjestelmää samalla isäntäkoneella. Tämä helpottaa palvelinresurssien kohdentamista ja säästää rahaa. Virtualisoinnin etuina voidaan myös mainita skaalautuvuus ja vikasietoisuus. Huonona puolena näissä (eritoten avoimissa) ympäristöissä on se, että ne ovat melko monimutkaisia palapelejä. Rymarczyk onkin sitä mieltä, että integrointi olisi syytä tehdä kunnolla. (IBM Systems Lab Services and Training group, 2011.)

Opinnäytetyöni tavoitteena on luoda varmatoiminen ja hajautettu virtualisointiympäristö OSA-hankkeen tarpeisiin. Hanke hyödyntää nimensä mukaan mahdollisimman paljon avoimen lähdekoodin tekniikoita. Koska järjestelmän alustasta haluttiin helposti hallittava ja vikasietoinen sekä se haluttiin pystyttää nykyaikaiselle pohjalle, päädyttiin palvelinsalivirtualisointiin (datacenter virtualization). Lisäsyynä mainittakoon, että kaupallisille tuotteille ei tahdota tarjota yli 50:n vuoden tukea ilman huikeita kustannuksia, jos silloinkaan.

Opinnäytetyössäni tutkin ja testasin avointa konesaleihin suunnattua rauta-avusteista virtualisointiratkaisua nimeltä OpenNebula, joka antaa ainakin teoriassa valita mitä tekniikoita siihen halutaan liittää, mutta käytännössä kuitenkin Qemu-KVM ja Libvirt ovat parhaiten tuettuja. OpenNebulaan on myös tarjolla käytännöllinen OpenNebula Sunstone web –käyttöliittymä, jonka päätin asentaa käyttöä helpottamaan.

2 VIRTUALISOINTI

2.1 Kuinka virtualisointi keksittiin

Virtualisoinnin kehittivät alunperin General Electric (GE), Bell Labs ja International Business Machines (IBM). (History of Virtualization Everything VM, 2011) Usean yhtäaikaisen käyttäjän käyttöjärjestelmän käsite muodostui alun perin Oxfordin ensimmäisen laskennan professorin Christopher Stracheyn UNESCO -konfferenssissa luetusta paperista. Se käsitteli ”suoritinajan jakamista suurissa nopeissa tietokoneissa” (Time sharing in large fast computers). Hänen ajatuksensa muuttivat laskennan tutkimuksen suuntaa. (Virtualization: A long brief history, 2013)

1960-luvulla IBM:llä oli suuri määrä järjestelmiä, joista jokainen oli poikkeuksellisen erilainen kuin edeltäjänsä. Tämä aiheutti asiakkaille ongelmia yhteensopivuuden saralla. Tähän aikaan tietokoneilla ei voitu suorittaa monia ohjelmia samanaikaisesti (multitasking). Se ei ollut ongelma, koska suurin osa IBM:n asiakkaista oli tiedekuntia, joten ohjelmien suoritus voitiin hoitaa eräajona. IBM alkoi työstää S/360 keskustietokonettaan, jonka oli tarkoitus korvata suurin osa heidän muista järjestelmistään ja säilyttää taaksepäin yhteensopivuus. Järjestelmä suunniteltiin alunperin yhden käyttäjän sarja-ajo järjestelmäksi. (History of Virtualization Everything VM, 2011), (Virtualization: A long brief history, 2013)

Tilanne kuitenkin alkoi muuttua 1.7.1963, kun Massachusettsin Teknologian laitos (Massachusetts Institute of Technology) ilmoitti projekti MAC:stä (Mathematics and Computation, uudelleen nimettiin myöhemmin Multiple Access Computer:iksi). Projekti MAC sai kahden miljoonan dollarin rahoituksen DARPA:ta, joka tuki erityisesti käyttöjärjestelmien, tekoälyn ja laskennallisen teorian tutkimusta. (History of Virtualization Everything VM, 2011), (Virtualization: A long brief history, 2013)

Osana tutkimustaan MIT tarvitsi laitteiston, joka kykenisi useampaan kuin yhteen samanaikaiseen käyttäjään ja yritti näin ollen houkutella tietokonevalmistajia kuten GE:tä ja IBM:ää sitoutumaan kyseiseen projektiin. IBM ei kuitenkaan ollut vielä valmis lähtemään mukaan, koska heidän mielestään kysyntää ei ollut tarpeeksi. MIT ei myöskään halunnut käyttää valmista muunneltua järjestelmää joten he ottivat yhteyttä GE:hen, joka taas oli valmis sitoutumaan suoritinaikaa jakavan tietokoneen kehittämiseen. (History of Virtualization Everything VM, 2011)

Vasta tajuttuaan menetyksensä IBM:n kellot soivat ja he alkoivat ymmärtää, että kysyntää tämän tyyppiselle järjestelmälle tosiassa on, eritoten sen jälkeen kun IBM:n korviin kantautui Bell Labsin tarpeet. (History of Virtualization Everything VM, 2011)

Vastapainona MIT:n ja Bell Labs:in tarpeille IBM kehitti CP-40 keskustietokoneen, jota ei kuitenkaan koskaan myyty asiakkaille ja sitä käytettiin vain Bell Labsilla. Tämä on kuitenkin historiallisesti tärkeää sillä CP-40 järjestelmän pohjalta kehitettiin CP-67, ensimmäinen kaupallinen virtualisointia tukeva järjestelmä. CP-67:ssa käytettyyn käyttöjärjestelmään viitattiin nimellä CP/CMS (Control Program / Console Monitor System). CMS oli minimalistinen yhden käyttäjän vuorovaikutteinen käyttöjärjestelmä. CP taas oli keskustietokoneella pyörivä ohjelma jolla luotiin virtuaalikoneita, joilla pyöri CMS ja joiden kanssa käyttäjä oli vuorovaikutuksessa (hieman kuten pilvi ja web -sovellukset nykypäivänä). (History of Virtualization Everything VM, 2011), (Virtualization: A long brief history, 2013)

Vuorovaikutus oli CP-67:ssa tärkeä tekijä, koska aikaisemmin tietokoneelle vain syötettiin ohjelma, jonka jälkeen se hoiti laskennan ja tulosti vastauksen joko tulostimella tai näytölle. Vuorovaikutteinen käyttöjärjestelmä siis tarkoittaa sitä, että käyttäjällä on mahdollisuus vaikuttaa ohjelmaan ajon aikana. (History of Virtualization Everything VM, 2011), (Virtualization: A long brief history, 2013)

Ensimmäinen versio CP/CMS käyttöjärjestelmästä tosin tunnettiin nimellä CP-40 ja sitä käytettiin vain laboratoriossa. CP/CMS julkaistiin virallisesti 1968 ja ensimmäinen vakaa versio julkaistiin 1972. (History of Virtualization Everything VM, 2011), (Virtualization: A long brief history, 2013)

Perinteinen lähestymistapa suoritinaikaa jakavalle tietokoneelle oli jakaa resurssit (muisti, suoritinaika, jne.) käyttäjien kesken. Hyvä esimerkki menneeltä ajalta on MultiCS, joka oli luotu MIT:ssä osana project MAC:ia. MultiCS:ssän jatkokehitys tapahtui kuitenkin Bell Labsilla ja myöhemmin siitä kuoriutui Unix. (History of Virtualization Everything VM, 2011), (Virtualization: A long brief history, 2013)

CP:n lähestymistapa suoritinajan jakamiseksi osoittautui todella käteväksi ja antoi omalla tavallaan jokaiselle käyttäjälle oman tietokoneen (tyhmät päätteet). (History of Virtualization Everything VM, 2011), (Virtualization: A long brief history, 2013)

Virtuaalikoneen edut verrattuna suoritinaikaa jakavaan käyttöjärjestelmään on resurssien jako niin ettei kaikkea ole pakko jakaa tasaisesti kaikkien käyttäjien kesken. Lisäksi tämä tuo ainakin nykypäivänä mukanaan tietoturvaan, käyttöjärjestelmän arkkitehtuuriin, sekä vakauteen liittyvät seikat. (History of Virtualization Everything VM, 2011), (Virtualization: A long brief history, 2013)

2.2 Ohjelmien siirrettävyys

Vaikka Unix ei olekaan luotu ajamaan virtualisoituja käyttöjärjestelmiä eikä se myöskään ole ensimmäinen montaa yhtäaikaista käyttäjää tukeva käyttöjärjestelmä, niin se on silti loistava esimerkki yhdestä maailman eniten käytetyimmästä käyttäjätasoa virtualisoivasta käyttöjärjestelmästä. Unixissa useat käyttäjät jakavat resurssit, kuten suoritinajan, välimuistin ja massamuistin. Silti jokaisella käyttäjällä on oma erillinen profiilinsa (“käyttäjätunnus”), joka on erotettu muista profiileista. Riippuen käyttöjärjestelmän asetuksista käyttäjä voi tai ei voi asentaa itselleen sovelluksia. Myös tietoturva on hoidettu käyttäjäkohtaisesti. (History of Virtualization Everything VM, 2011)

Unix ei ollut ainoastaan edistysaskel usean käyttäjän käyttöjärjestelmien kohdalla, vaan myös edistysaskel sovellustason virtualisoinnissa. Unix ei ole kuitenkaan esimerkki sovellustason virtualisoinnista vaikka se toikin mukanaan suuremmat mahdollisuudet siirtää sovellus toiselle arkkitehtuurille. Ennen Unixia lähes kaikki käyttöjärjestelmät oli ohjelmoitu Assembly ohjelmointikielellä. Kun taas Unix oli ohjelmoitu C-ohjelmointikielellä, minkä johdosta vain pientä osaa käyttöjärjestelmästä (ydin / kernel) täytyi muuttaa, jotta sen sai pyörimään eri alustalla. Loppuosa käyttöjärjestel-

mästä oli helppo kääntää ilman - tai ainakin lähes ilman muutoksia. (History of Virtualization Everything VM, 2011)

2.3 Sovellustason virtualisointi

Vaikka Unix ja C-kääntäjä sallivat käyttäjän ajaa lähes mitä tahansa ohjelmaa millä tahansa alustalla, täytyi käyttäjän silti kääntää kaikki ohjelmat joita hän halusi ajaa. Todellisen siirrettävyyden nimissä tarvitaan sovellus, joka hyödyntää sovellustason virtualisointia. Tämän jälkeen vuonna 1990 Sun Microsystems astui mukaan kuvioihin projektinimellä Stealth, jossa olivat mukana Sunin C/C++ Sovellusrajapintoihin tympääntyneet insinöörit. He uskoivat, että on olemassa parempiakin tapoja kirjoittaa ja ajaa sovelluksia. Tämän jälkeen projekti koki monia nimenvaihdoksia ja sai lopulta nimekseen Java. Oli vuosi 1994 ja Java oli suunnattu Web -markkinoille, koska Sun näki siellä suuret markkinat. Tähän aikaan ei ollut vielä yleistä tapaa ajaa monipuolisia (ns. “rikkaita”) sovelluksia Internetin välityksellä. Tammikuussa 1996 julkaistu Java Development Kit (JDK) kuitenkin koetti aloittaa tämän kehityssuunnan. Java salli käyttäjän kirjoittaa sovelluksen vain kerran ja tämän jälkeen sitä pystyi ajamaan millä tahansa tietokoneella jolla oli Java Run-time Environment (JRE) asennettuna. Java Run-time Environment oli ja on edelleen ilmainen ympäristö vaikka onkin hiljattain siirtynyt Oraclen omistukseen. Javan tapauksessa koodi käännetään Java bittikoodiksi, jota vain JRE osaa lukea. JRE muodostuu monista osista, joista tärkeimpänä Java virtuaalikone (Java Virtual Machine). Sitä voi luonnehtia virtuaalikoneeksi jossa pyörii todella pieni käyttöjärjestelmä tarkoituksenaan ainoastaan suorittaa Javalla kirjoitettuja sovelluksia. (History of Virtualization Everything VM, 2011)

2.4 Laitteistovirtualisoinnin yleistyminen

Tammikuussa 1987 Insignia Solutions esitteli sovellus emulaattorin nimeltä SoftPC, joka mahdollisti DOS sovellusten ajamisen Unixissa. Tällaista ei kukaan ollut toteuttanut aiemmin. Näihin aikoihin PC, joka kykeni ajamaan MS DOS:ia maksoi noin 1500$. SoftPC maksoi noin 500$ ja oli näin ollen erittäin kelvollinen ratkaisu Unix työaseman omaaville henkilöille. (History of Virtualization Everything VM, 2011)

Vuonna 1989 Insignia Solutions julkaisi Mac version SoftPC:stä antaen mahdollisuuden ajaa ei ainoastaan DOS, vaan myös Windows sovelluksia. Vuonna 1994 yritys

alkoi myydä sovellustensa mukana ennalta ladattuja käyttöjärjestelmiä sisältäen SoftOS/2:sen ja SoftWindowsin. Tästä menestyksestä inspiroituneena muut yritykset alkoivat nostaa päätään ja vuonna 1997 Apple loi sovelluksen nimeltä Virtual PC jota myytiin Connectix nimisen yrityksen kautta. (History of Virtualization Everything VM, 2011)

"Published applications"

Unixin alkuaikoina sovelluksiin otettiin etäyhteys Telnet -protokollalla ja myöhemmin SSH:n (Secure shell) avulla. Kumpikin tukee sekä teksti- että graafista tilaa, joskaan graafinen puoli ei ole optimoitu. Telnetin ja SSH:n ratkaiseva ero on siinä, että SSH -yhteys on salattu. Käytännössä siis näillä työkaluilla useamman käyttäjän oli mahdollista käyttää sovellusvirtualisoinnin avulla samaa tietokonetta "samanaikaisesti". Windowsille ja OS/2 käyttöjärjestelmille etäkäyttötyökaluja oli tarjolla vain kolmansien osapuolien tuottamina ja ne eivät tukeneet useaa samanaikaista käyttäjää. (History of Virtualization Everything VM, 2011)

Eräät IBM:n insinöörit elättelivät kuitenkin ideaa vastaavasta työkalusta OS/2 käyttöjärjestelmälle. Valitettavasti IBM vanhollisena yrityksenä ei kuitenkaan jakanut samoja näkemyksiä ja vuonna 1989 Ed Lacobucci päättikin jättää IBM:n ja perustaa Citrus nimisen yrityksen. Yritys kuitenkin vaihtoi nimensä nopeasti, koska saman niminen yritys oli jo olemassa. Uudeksi nimeksi tuli Citrix, joka on kombinaatio sanoista citrus ja unix. (History of Virtualization Everything VM, 2011)

Citrix lisensoi OS/2 lähdekoodin Microsoftin kautta ja työsti kaksi vuotta useaa yhtäaikaista käyttäjää tukevaa käyttöliittymää. Käyttöliittymä saikin osuvasti nimen MULTIUSER. Vuonna 1991 Citrix joutui kuitenkin hylkäämään projektin, kun Microsoft ilmoitti lopettavansa OS/2 tuen. Citrix kuitenkin lisensoi Windowsin lähdekoodin Microsoftilta ja alkoi työstää vastaavaa sovellusta Windowsille. Vuonna 1993 Citrix hankki Novellin Netware Access Server nimisen sovelluksen, joka vastasi toiminnaltaan Citrixin OS/2 käyttöjärjestelmälle toteuttamaa sovellusta, joka tarjosi usean käyttäjän tuen. Tämän jälkeen Citrix lisensoi Windows NT:n lähdekoodin ja vuonna 1995 alkoi myydä tuotetta nimeltä WinFrame. WinFrame oli Windows NT 3.5 varustettuna etäkäyttöominaisuuksilla (salli etäajaa sovelluksia) ja usean yhtäaikaisen käyttäjän tuella. WinFramen Windows NT 4.0 kehitystyön aikana Microsoft päätti ettei enää aio myöntää tarvittavia lisenssejä Citrixille. Tästä johtuen Citrixin ainoa

taloudellisesti järkevä vaihtoehto oli lisensoida WinFrame Microsoftille, joka sitten sisällytettiin Windows NT 4.0:aan päätepalveluina (Terminal Services). (History of Virtualization Everything VM, 2011)

Virtualisoidut työpöytäkäyttöjärjestelmät (VDI/Virtual Desktop Infrastructures)

VDI on suhteellisen uusi tapa ajaa työpöytäkäyttöjärjestelmää (kuten Windows XP) virtuaalikoneella keskitetyssä ympäristössä. Työpöytäkäyttöjärjestelmien virtualisointi tuli massan tietoisuuteen vasta vuonna 2007, kun WMWare esitteli DVI tuotteensa. Toteutus itsessään muistuttaa todella paljon 60-luvun IBM:n keskustietokoneita virutaalikoneineen (tyhmät päätteet). Jokainen käyttäjä saa oman käyttöjärjestelmänsä ja jokainen käyttäjä voi touhuta järjestelmässään mitä tahansa häiritsemättä muita. Tämä on todella tehokas tapa jakaa resursseja käyttäjien kesken. Näillä näkyminen tämä on myös tehoa vaativien pelien tulevaisuus. Vaikka käyttäjien oli mahdollista ajaa virtualisoituja työpöytäkäyttöjärjestelmiä työasemillaan niin ne eivät yleistyneet rajusti ylläpitoon liittyvien ongelmien vuoksi. Tähän WMWare ja useat muut yritykset kuten myös Microsoft ja Citrix kehittivät ratkaisunsa (WMWaren työkalun nimeksi tuli Virtual Machine Manager), jotka aiheuttivat VDI järjestelmien räjähdysmäisen kasvun. (History of Virtualization Everything VM, 2011)

2.5 Virtualisointitekniikat

Monesti menee sekaisin tekniikka ja työkalu, joten muista, että tekniikka on nimenomaan taito, ei työkalu.

Täysi virtualisointi käyttäen binäärikäännöstä ja suoraa suoritusta

Virtualisointitekniikoita on paljon, mutta täysi virtualisointi käyttäen binäärikäännöstä ja suoraa suoritusta (full virtualization using binary translation and direct execution) on tällä hetkellä ainoa tapa ajaa mitä tahansa käyttöjärjestelmää toisen käyttöjärjestelmän päällä. Binäärikäännös tarkoittaa sitä, että binäärimuotoisia ”suorittimen toimintaohjeita” eli käskyjä tai kokonaisia käskykantoja käännetään (siis kuten kieltä) toiseen käskykantaan sopiviksi. Binäärikäännöksen tarkoitus virtualisoinnissa on korvata yhtyeensopimattomat konekäskyt (kernel instructions) käskyillä, jotka toimivat virtualisointiympäristössä toivotulla tavalla. “Suora suoritus” (Direct execution) tarkoittaa sitä, että käyttäjätason (user level) koodi suoritetaan suoraan prosessorilla ilman, että sitä tarvitsee käsitellä isäntäkäyttöjärjestelmän puolella.

Nämä kaksi tekniikkaa yhdistämällä saadaan aikaan täysi virtualisointikerros, jossa virtualisoitu käyttöjärjestelmä (Guest OS) on täysin erotettu laitteistosta. Koska virtuaalikoneet eivät ole omasta näkökulmastaan yhteydessä toistensa tai isäntäkoneen kanssa, ne näkevät toisensa fyysisinä laitteina. Tämä tapa myös yksinkertaistaa migrointia ja siirrettävyyttä sillä virtuaalikoneeseen ei tarvitse tehdä muutoksia, jotta sitä voidaan ajaa virtualisoidun raudan lisäksi myös oikealla raudalla. (Understanding Full Virtualization, Paravirtualization, and Hardware Assist, 2007)

Monesti kuulee puhuttavan myös termistä bare-metal ”ratkaisusta” joka tarkoittaa sitä, että käyttöjärjestelmä on suunniteltu virtualisoinnin ympärille ja sillä on käytännössä vain yksi tarkoitus. Tämmöisiä virtualisointiratkaisuja ovat mm. vSphere Hypervisor (uudelleen nimetty VMware-hypervisor) ja Xen Cloud Platform.

Käyttöjärjestelmäavusteinen tai rinnakkaisvirtualisointi

Käyttöjärjestelmäavusteinen virtualisointi (OS Assisted virtualization / paravirtualization) tarkoittaa sitä, että virtualisoitavan käyttöjärjestelmän ydintä on muokattu niin, että yhteensopimattomat konekäskyt on korvattu hyperkutsuiksi (hypercall) kutsutuilla järjestelmäkutsuilla (systemcall), jotka kutsuvat käyttöjärjestelmän sijaan hypervisoria. Koska ydintä joudutaan muokkaamaan, ei suljetun koodin käyttöjärjestelmien ajaminen yleensä onnistu (Windows). Lisäksi yhteensopivuus on huono ja siirrettävyys heikko. Esimerkiksi XEN tuki aikoinaan vain käyttöjärjestelmäavusteista virtualisointia (lähinnä siitä syystä ettei laitteistoavusteista virtualisointia ollut vielä tarjolla). (Understanding Full Virtualization, Paravirtualization, and Hardware Assist, 2007)

Laitteistoavusteinen virtualisointi

Tällä hetkellä markkinoilla on kaksi kilpailevaa teknologiaa, Intelin VT-x ja AMD:n AMD-V. Kummatkin perustuvat samaan ideaan etuoikeutetuista konekäskyistä. Ne lisäävät suorittimen protection domain 0:n jälkeen uuden ringin (useimmiten tätä nimitetään ring -1:si), joka luo yhteensopivuuden yhdeksälle uudelle konekäskylle, jotka mahdollistavat Hypervisorin ajamisen niin, että sillä on pääsy laitteistoon (teoriassa lähes sama asia kuin potection domain 0:ssa ajaminen, mutta yhteensopivuussyistä toteutettu näin). (Understanding Full Virtualization, Paravirtualization, and Hardware Assist, 2007)

Lisätietoa protection domaineista löytyy kirjasta Operating Systems Design and Implementation, 3rd Edition, sivulta 537 kappaleesta 5.5.1. Protection Domains

”Suojaustila / suojauskehä” (Protection domains / protection rings tai virallisemmalta nimeltään hierarchical protection domains) on käyttöjärjestelmään rakennettu keino erottaa softa ja rauta toisistaan niin ettei esimerkiksi virukset pääse suoraan käsiksi rautaan. Prosessin tehdessä järjestelmäkutsun (system call), vaihtuu suoritus ydintilaan (kernel domain) jossa nimensä mukaan ajetaan vain käyttöjärjestelmän ydintä. (Andrew S. Tanenbaum & Albert S. Woodhull - Operating Systems Design and Implementation 3rd Edition, 2006)

Eri suoritinarkkitehtuureissa suojaustilojen määrä voi vaihdella ja tästä syystä esimerkiksi Windowsissa on vain kaksi tilaa, koska se on alun perin tehty kahta suojaustilaa tukevalle suorittimelle. Ydin- ja ohjelmistotilat. Esimerkiksi x86 arkkitehtuurissa tiloja on neljä ja ne ovat: ydin, laitteistoajuri, rajoitetumpi laitteistoajuri, ohjelmisto. (William Stallings - Operating Systems Internals and Design Principles 7th Edition, 2012)

Kapselointi (containers)

Virtualisoinnin yhteydessä kuulee usein puhuttavan erilaisista “kapselointiympäristöistä” virtualisointiympäristöinä. Teknisesti kapselointi ei kuitenkaan ole sama asia kuin virtualisointi. Kapselointi jakaa isäntäjärjestelmän ytimen ja näin ollen se on pikemminkin verrattavissa chroot-ympäristöön, koska ydintä ei varsinaisesti virtualisoida. Käyttöjärjestelmiä, jotka eivät kykene käyttämään isäntäkoneen ydintä ei voida ajaa kyseisessä ympäristössä. Hyvänä esimerkkinä tästä on Microsoftin Windows. (Understanding Full Virtualization, Paravirtualization, and Hardware Assist, 2007)

Esimerkkejä kapselointiympäristöistä ovat mm. LXC (LinuX Containers), OpenVZ Linuxvserver, FreeBSD jail, AIX Workload Partitions ja Solaris Containers.

Virtualisoinnin sovellutukset

Palvelinvirtualisointi on yleensä ensimmäinen tietoisesti käyttöönotettu virtualisointitekniikka. Jokaisessa palvelimessa voidaan ainakin teoriassa käyttää joko palvelinvirtualisointia tai kapselointia. Syitä käyttää palvelinvirtualisointia on monia, mutta yleensä syynä ovat palvelinten pienet käyttöasteet, jolloin kannattaa ajaa useita virtuaalisia palvelimia saman raudan päällä. Tämä johtaa siihen, että fyysisiä palvelimia

tarvitaan vähemmän jolloin sähkönkulutus, palvelinsalin lämmöntuotto, tilantarve, ylläpidon työmäärä ja huoltokustannukset laskevat. Lisäksi palvelinten käyttöönotto ja poisto nopeutuu, jolloin ikivanha rauta ei jää pyörimään nurkkiin ja kuluttamaan sähköä. Joissakin tapauksissa tarvitaan myös vähemmän sovellus yms. lisenssejä, jolloin lisenssikustannukset laskevat. Yleensä myös kuorman tasaus ja vikasietoisuus hoituu helpommin, jolloin käytettävyys paranee.

Sovellusvirtualisointi (software container) on useimmiten suurissa yrityksissä ja oppilaitoksissa käytetty tekniikka, jota ei erityisemmin mainosteta IT-tuen ulkopuolella. Sovellusvirtualisointia kannattaa käyttää, koska se mahdollistaa lisenssien siirron eri käyttäjille, jolloin kustannukset laskevat. Lisäksi sovellusristiriitojen välttäminen on helppoa, koska sovellukset eivät tee muutoksia itse käyttöjärjestelmän rekisteriin, jolloin voidaan ajaa useita eri versioita samasta sovelluksesta samassa koneessa. Sovelluksia voidaan ajaa työasemien lisäksi myös palvelimella, jolloin tehokkaan raudan tarve laskee ja kustannukset laskevat. Myös sovellusten ylläpito ja päivitysten jakelu on hoidettavissa samasta paikasta. Lisäksi jakelu, käyttöönotto, päivittäminen ja poistaminen on helppoa. Käytännössä sovellusvirtualisointi ei eroa kovinkaan paljoa käyttöjärjestelmien kapseloinnista ja englanniksi nimitykset ovatkin useimmiten samat ja sotkeutuvat helposti keskenään. (VIRTUALIZATION DEFINED - EIGHT DIFFERENT WAYS, 2008.)

Työasemavirtualisointi (VDI) on tavallaan jatkoa palvelinvirtualisoinnille. Jokaisen käyttäjän työpöytä koostuu käyttöjärjestelmästä, sovelluksista, asetuksista ja tiedostoista. Työasemavirtualisoinnin etuihin voisi laskea tiedostojen ja tietojen tallentumisen palvelimelle, jolloin hävikki pienenee vikatilanteessa tai varkauden sattuessa. Lisäksi työasemavirtualisointi helpottaa myös laajamittaisten korjaus- ja päivitystöiden suorittamista. Tietoja ei myöskään tarvitse siirrellä, jolloin päätelaitetta vaihdettaessa ei tule ylimääräistä työtä siirrellessä sovelluksia, asetuksia ja tiedostoja. Myös käyttöjärjestelmän ja sovellusten saatavuus paranee mikäli virtuaalikoneella sattuu jokin vahinko ja käyttöjärjestelmä tai sovellukset eivät enää toimikaan on helppoa palauttaa käyttöjärjestelmä varmuuskopiosta tai vedoksesta (snapshot) ilman aikaa vieviä uudelleenasennuksia. Työasemavirtualisointi laskee myös laitteistokustannuksia, koska työasema on vain “kuvayhteys” palvelimeen ja varsinainen tietojenkäsittely tapahtuu palvelimella. Siksi työasema ei vanhene nopeasti käyttötarkoitusta silmälläpitäen. Lisäksi ylläpito helpottuu, koska käyttöjärjestelmän vaihdokset ja päivitykset nopeutu-

vat, koska muutoksia ei tarvitse tehdä jokaiselle koneelle erikseen ja käyttäjäprofiileihin pääsee käsiksi kaikkialta. Lisäksi niiden siirtotyö on paljon pienempi ellei jopa lähes olematon. Myös valvonta ja hallinta helpottuu, koska virtualisoinnin myötä jokaisesta työasemasta ei tarvitse luoda omaa levykuvaansa. Voidaan luoda levykuvia, jotka sopivat tietylle ryhmälle käyttäjiä ja näin saadaan ylläpidettävien käyttöjärjestelmien määrää ainakin teoriassa pudotettua sellaisiin määriin ettei suuryrityksessä tarvita jokaista 5-10 käyttäjän ryhmää varten omaa tukihenkilöä. (VIRTUALIZATION DEFINED - EIGHT DIFFERENT WAYS, 2008)

2.6 Hypervisorit

Hypervisor (toiselta nimeltään virtual machine monitor eli VMM) on sovellus, joka hallitsee useita käyttöjärjestelmiä tai useita instansseja samasta käyttöjärjestelmästä yhden fyysisen tietokoneen päällä. Hypervisor myös hallitsee ja jakaa järjestelmän resursseja kuten suoritinaikaa ja muistia virtuaalikoneille. Esittelen jälkimmäisenä tämän hetken suositummat hypervisorit.

XEN

XEN on vuonna 2003 julkaistu Cambridgen yliopiston kehittämä microkernel malliin perustuva avoimen lähdekoodin hypervisor. Sitä ylläpitää ja kehittää XEN-projektin ympärille muodostunut yhteisö. Alunperin XEN tuki vain käyttöjärejstelmäavusteista virtualisointia, Nykyään se tukee myös laitteistoavusteista virtualisointia, jonka vuoksi mitä tahansa käyttöjärejstelmää voidaan ajaa XENin päällä muuttamattomana, mikäli suorittimessa on tuki AMD-V:lle tai VT-x:lle. XENin tarjoamat käyttöjärjestelmäavusteisesti virtualisoidut järjestelmäajurit voivat nopeuttaa joitakin käyttöjärjestelmiä huomattavasti. (Understanding Full Virtualization, Paravirtualization, and Hardware Assist, 2007), (KVM or Xen? Choosing a Virtualization Platform, 2010)

KVM

KVM on Open Virtualization Alliancen (OVA) kehittämä vuonna 2007 julkaistuun Linux kernelin versioon 2.6.23 sisällytetty avoimen lähdekoodin hypervisor. Se muuttaa linux kernelin hypervisoriksi, mutta toimiakseen vaatii suorittimelta laitteistoavusteisen virtualisoinnin tuen. Nykyään QEMU ja KVM on paketoitu yhteen jolloin laitteistoavusteisen virtualisoinnin puutteessa voidaan palata käyttämään QEMU:n tar-

joamaa emulointia. Se tukee osittain käyttöjärjestelmäavusteisesti virtualisoituja laiteajureita joillekin laitteille käyttäen VirtIO ohjelmointirajapintaa. Se tarjoaa vastaavanlaisen tuen kuin XEN poistaen turhia rasitteita laitteistovirtualisoinnista. (KVM or Xen? Choosing a Virtualization Platform, 2010), (QEMU project, 2014)

VMware ESX/ESXi

VMware ESX on kaupallinen vuonna 2001 julkaistu hypervisor, johon WMware viittaa vmkernel nimellä. ESX on omalaatuinen ratkaisu ja vaikka se käyttää Linux kerneliä ladatakseen kokoelman virtualisointikomponentteja kuten WMware vmkernel ytimen, ei se silti ole open sourcea muilta kuin hyväksikäytetyiltä osiltaan. Käynnistyksen jälkeen vmkernel ladataan Linuxista jonka jälkeen se ottaa ohjat. (Understanding Full Virtualization, Paravirtualization, and Hardware Assist, 2007)

Hyper-V

Vuonna 2008 julkaistu Hyper-V on hypervisor, jonka voi hankkia kahdella eri tavalla. Sen voi hankkia Hyper-V server pakettina tai Windows Server:in mukana. Ero näissä on se, että Hyper-V Server on suunnattu niille, jotka siirtävät Windows palvelimiaan fyysiseltä raudalta virtuaalisiksi ja näin ollen omistavat jo lisenssit. Hyper-V server on bare metal ratkaisu, joka käyttää Windows ajurirajapintaa. Koska siitä on yritetty jättää kaikki ylimääräinen pois, on sen hallinta on hoidettava komentoriviltä tai Windows käyttöjärjestelmästä käyttäen Remote Server Administration työkalua. (Hyper-V Server 2012 or Windows Server 2012 with Hyper-V?, 2013)

Windows Serverin kaupalliseen käyttöön suunnatut lisenssit taas sallivat virtualisoidut Windowsit suoraan. Koska kyse on kokonaisesta käyttöjärjestelmästä, voi hallintatyökalut asentaa suoraan palvelimelle ja käyttää esimerkiksi RDP:n (Remote Desktop Protocol) kautta etänä. Valitettavasti rajoittamattoman määrän lisenssejä tarjoaa ainoastaan Datacenter-edition. Standard tarjoaa kaksi ja Essentials yhden lisenssin virtuaalikoneelle. Ei kaupalliseen käyttöön suunnattu Windows Server Foundation edition ei tarjoa virtualisointia. (Hyper-V Server 2012 or Windows Server 2012 with Hyper-V?, 2013)

3 PILVI

Pilvi määrittyy USA:n National Institute of Standards and Technologyn (NIST) mukaan seuraavista piirteistä: Tietokoneella tai mobiililaitteella käytettävä automatisoitu itsepalvelu, jossa resurssit on jaettu useiden käyttäjien sekä sovellusten kesken ja ne ovat tarpeen mukaan helposti uudelleenjaettavissa. Lisäksi tämä kaikki tarjotaan ”mitattuna palveluna” eli resursseja (esim. muistia, suoritinaikaa, levytilaa) voi hankkia tarpeen mukaan helposti lisää. (The NIST Definition of Cloud Computing, 2011)

Pilvityypit

Pilvityyppejä ovat USA:n National Institute of Standards and Technologyn (NIST) mukaan neljä ja ne on määritetty seuraavasti.

Yksityinen pilvi (private cloud) on yksityisen organisaation pilvi. Se voi olla joko organisaation tai kolmannen osapuolen yksin tai yhdessä omistama, hallitsema ja operoima. Se voi sijaita lähes missä vain, myös useammassa paikassa. (The NIST Definition of Cloud Computing, 2011)

Yhteisöpilvi (community cloud) on useiden samanlaista pilviympäristöä tarvitsevien organisaatioiden pilvi. Organisaatioita voi yhdistää mm. suuntautuminen, tietoturva vaatimukset ja käytännöt. Se voi olla joko yhden tai useamman organisaation tai kolmannen osapuolen omistama, hallitsema ja operoima. Se voi sijaita lähes missä vain, myös useammassa paikassa. (The NIST Definition of Cloud Computing, 2011)

Julkinen pilvi (public cloud) on nimensä mukaan julkinen pilvi avoimeen käyttöön. Se voi olla joko yhden tai useamman yrityksen, akateemisen järjestön tai hallituksen organisaation omistama, hallitsema ja operoima. Sen täytyy sijaita tarjoajan tiloissa. (The NIST Definition of Cloud Computing, 2011)

Hybridipilvi (hybrid cloud) on ympäristö, joka koostuu kahdesta tai useammasta erillisestä yksityisestä, yhteisöllisestä tai julkisesta pilvestä, joilla on omat olemuksensa. Ne on tuotu yhteen joko standardeilla tai kaupallisilla teknologioilla, jotka mahdollistavat datan ja sovellusten siirrettävyyden. (The NIST Definition of Cloud Computing, 2011)

Pilvipalvelut

Alunperin termi pilvi on kuvannut erilaisia verkkoja ja Internetiä. Mainosmiehet ovat kuitenkin ruvenneet käyttämään sitä miten sattuun ja nykyään termiä pilvi käytetään useasti, kun tarjotaan palveluna internetin yli softaa (Software as a Service, eli SaaS), rautaa (Hardware as a Service, HaaS), ympäristöjä (Infrastructure as a Service, eli IaaS) tai alustoja (Platform as a Service, PaaS).

Näihin palveluihin käyttäjä pääsee yleensä kiinni nettiselaimella. Kuluja syntyy yleensä käytön mukaan. Suurin osa pilvipalveluista on pitkälle automatisoituja. (The NIST Definition of Cloud Computing, 2011.)

Pilvilaskenta

Verkon yli myytävät palvelinympäristöt ovat usein virtualisoituja. Tämä helpottaa niiden ylläpitoa, skaalaamista ja siirtelyä fyysisillä laitteilla huomaamattomasti ja tehokkaasti. Fyysiset palvelimet voivat sijaita eri palvelinsaleissa eri puolilla maailmaa. Tästä syystä palvelinvirtualisoitua ympäristöä kutsutaan usein pilvipalveluksi / pilviympäristöksi, vaikka aina näin ei ole. (The NIST Definition of Cloud Computing, 2011)

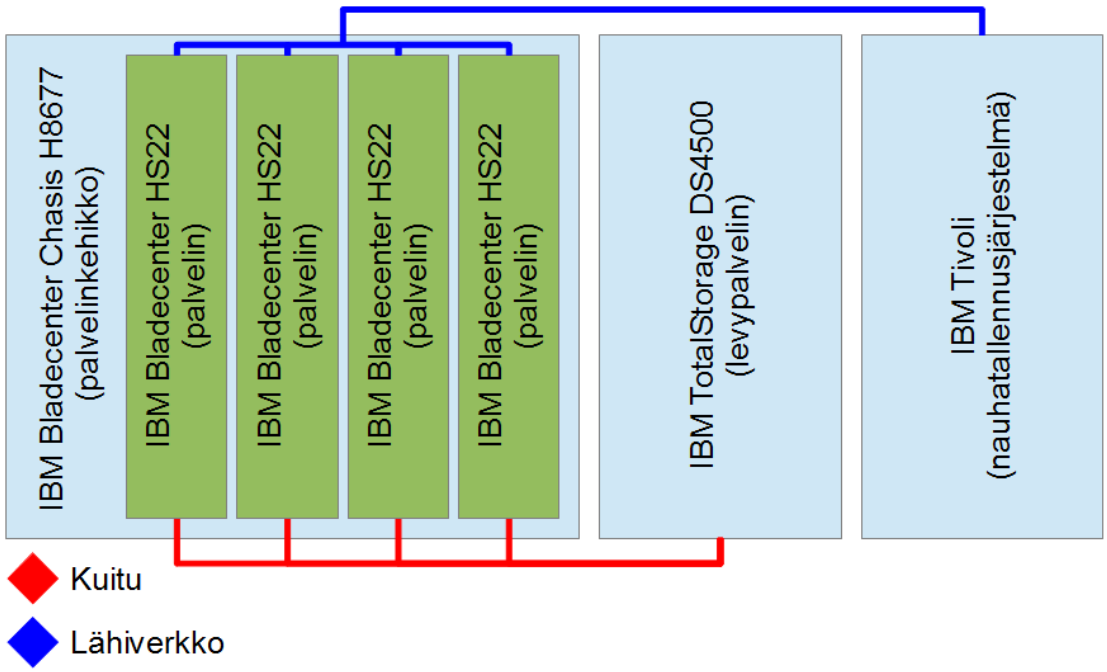
4 YMPÄRISTÖ

Sovellusvalinnat

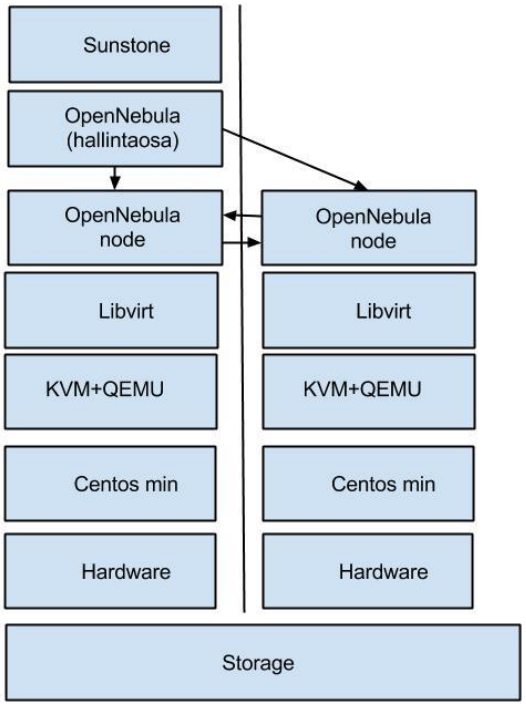
Sovellusvalintani pohjautuvat testeihin ja OSA-hankkeessa esitettyihin vaatimuksiin. Olin ennen opinnäytetyöni aloittamista harjoittelussa samassa hankkeessa. Testasin siinä erilaisia virtualisointiratkaisuja. Huomasin nopeasti ettei vanha rauta ollut paras mahdollinen testaamiseen ja tämän vuoksi esimerkiksi XCP osoittautui ongelmalliseksi. Vaatimukset muodostuivat vasta ymmärryksen mukana ja tästä johtuen päädyin OpenVZ:n kautta OpenNebulaan lähinnä hyvän hypervisor tuen vuoksi. Ei sekään täydellinen ollut, mutta se toimi kyseisellä raudalla. Päädyin KVM-pohjaiseen ratkaisuun, koska se pystyi hyödyntämään QEMU:a ja näin ollen pystyin ajamaan mitä tahansa käyttöjärjestelmää testiympäristössä. Lisäksi se on opennebulassa erittäin hyvin tuettu.

Laitteisto

Minulla oli käytössäni neljä IBM blade -palvelinta, jotka ovat asennettuna H8677 palvelinkehikkoon. Kehikosta on tuplatut yhteydet sekä verkkoon, että kuidun kautta DS4500 levypalvelimeen. Palvelimilta on myös yhteys Tivoli -nauhatallennusjärjestelmään, vaikka sitä ei hyödynnetäkään palvelinsalivirtuaalisoinnin yhteydessä (ainakaan vielä).



KUVA 1. Miten laitteet kytkeytyvät toisiinsa fyysisesti



KUVA 2. Järjestelmän kerroksien sijainnit ja kommunikointi.

5 ASENNUS

Asennus oli melko pitkä prosessi ja varauduin siihen kupillisella kahvia. Koska en ole verkon enkä palomuurien ylläpitäjä jouduin hankkimaan ylläpitäjien yhteystiedot. Lisäksi olisi ollut hyvä että minulla olisi ollut vähintään yksi IP-osoitte per solmu käytettävissäni. Käyttöjärjestelmä valinnat kannattaa miettiä läpi kunnolla ja tarkistaa tukeeko joku tietty Linux-jakelu suoraan kyseistä rautaa.

Isäntäkäyttöjärjestelmän valinta

Isäntäkäyttöjärjestelmäksi valitsin alunperin Ubuntu-Linuxin, koska se on paljon käytetty ja toimivaksi todettu, mutta jouduin pikaisesti pyörtämään päätökseni sillä siitä ei löydy läheskään yhtä hyvää Enterprise raudan tukea kuin Red Hat Linuxiin pohjautuvasta CentOSta. Valitettavasti vaikka ympäristö onkin pääasiassa avoin niin IBM:än nauhatallennusjärjestelmä -Tivoliin ei löydy ohjelmistoja muille kuin SLES ja Red Hat Linuxeille. Asensin CentOS:in virallisen ohjeen mukaan, ohje löytyy CentOSin nettisivulta.

5.1 Käytetyt sovellukset

QEMU-KVM

Nykyään QEMU ja KVM on yhdistetty QEMU-KVM paketiksi, jolloin laitteistoavusteisen virtualisoinnin puutteessa voidaan palata käyttämään QEMU:n tarjoamaa emulointia. (QEMU project, 2014)

Libvirt

Libvirtiksi kutsuttu kolmiosainen paketti sisältää hallintasovelluksen, virtualisointiratkaisut yhdistävän rajapinnan ja libvirtd-daemonin. Daemon huolehtii palvelinpäässä ajettavista virtuaalikoneista ja niihin tehtävistä muutoksista (käynnistys, sammutus, migraatio, configurointi sekä verkon ja tallennusjärjestelmän hallinta). Hallintasovellus on nimeltään virsh, se on C:llä kirjoitetu ja päätteessä ajettava. Sen rinnalle asennetaan useimmiten jokin graafinen sovellus kuten virt-manager.

Libvirt tukee useita eri hypervisoreita sekä tekniikoita. Tästä johtuen moni hallintasovellus hyödyntääkin sitä. Tällä hetkellä tuettuja hypervisoreita ovat ainakin: XEN, QEMU, KVM , LXC , OpenVZ, VirtualBox ja VMware ESX. (Libvirt project, 2013.)

OpenNebula

Vuonna 2008 julkaistu OpenNebula on työkaluohjelmisto hajautettujen sekakoosteisten palvelinsalivirtualisointiympäristöjen hallintaan. Sekakoosteisella tarkoitetaan sitä, että OpenNebula tukee yksityisiä, julkisia ja hybridi IaaS (Infrastructure as a service) ympäristöjä yhtä aikaa. OpenNebula pystyy hoitamaan tallennusjärjestelmän, verkon, virtualisoinnin, monitoroinnin ja tietoturvan moniportaisessa järjestelmässä (kuten laskentaklusterissa). OpenNebula ympäristöt luokitellaan pilviympäristöiksi. OpenNebula tarjoaa myös Ruby:llä toteutetun web-käyttöliittymän käyttäjille ja ylläpitäjille, mikä helpottaa yksityis- ja hybridiympäristöjen hallintatehtävissä. (OpenNebula Project (OpenNebula.org), 2013)

5.2 Asennusprosessi

Ensiksi täytyi asentaa käyttöjärjestelmä. Koska isäntäkäyttöjärjestelmän ei tässä tapauksessa ole hyvästä sisältää mitään ylimääräistä niin päädyin asentamaan Centos minimal:in, joka sisältää lähinnä vain perustan kaikelle tulevalle. Asennusprosessi seurasi pitkälti virallista asennusohjetta, joskin kohdistettuna kuitulevyyn. [http://wiki.centos.org/TipsAndTricks].

Red Hat klusteroinnin ja GFS2 osion luonti tapahtui seuraavasti. Asetin seuraavat domain nimet /etc/hosts –tiedostossa, joka solmulla:

10.21.221.2 node1
10.21.221.3 node2
10.21.221.4 node3
10.21.221.5 node4

Asensin: mod_cluster kuormituksen tasaimen (load balancer), RGManager vikasietoisuuspalvelun (failover services), gfs2 tiedostojärjestelmän luonti ja hallintatyökalut, lvm2 klusterilaajennuksen, cluster manager klusterin jäsen- ja päätöksentekotyökalun, ricci klusterin hallinta- ja asetustyökalun sekä luci web -käyttöliittymän seuraavasti:

Asennus solmuille joille ei tule hallintaa komennolla:
yum groupinstall "High Availability" "Resilient Storage"

Asennus hallintasolmulle:
yum groupinstall "High Availability Management" "High Availability"
(install modcluster rgmanager gfs2-utils lvm2-cluster cman ricci luci)

Näin luodaan tiedostojärjestelmä:
mkfs.gfs2 -p lock_dlm -t osacluster:GFS -j 4 /dev/mapper/mpatha
This will destroy any data on /dev/mapper/mpatha.
It appears to contain: data

Are you sure you want to proceed? [y/n] y
Device: /dev/mapper/mpatha
Blocksize: 4096
Device Size 2000.4 GB (1952146476 blocks)
Filesystem Size: 2000.4 GB (1952146476 blocks)
Journals: 4
Resource Groups: 4
Locking Protocol: "lock_dlm"
Lock Table: " osacluster:GFS"

UUID: ef9eda0d-a3da-b62d-f694-d738f5787285

Riccille salasanan asetus jokaisella solmulla tapahtuu seuraavasti:

```
passwd  ricci
Changing password for user ricci.
 New password: salasana
 passwd: all authentication tokens updated successfully.
```

Tämän jälkeen käynnistin riccin komennolla:

```
service ricci start
```

```
service ricci start
Starting system message bus:                [  OK  ]
 Starting oddjobd:                          [  OK  ]
 generating SSL certificates...  done
 Generating NSS database...  done
 Starting ricci:                            [  OK  ]
```

Asetin riccin käynnistymään bootissa seuraavasti:

```
chkconfig ricci on
```

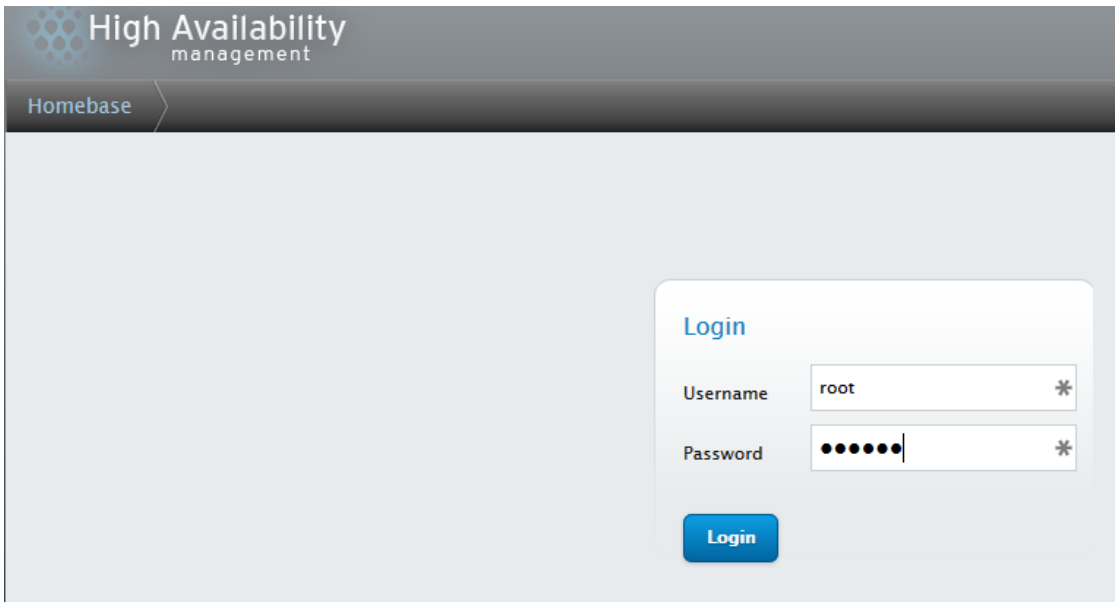
Käynnistin lucin hallintasolmulla seuraavasti:

```
service luci start
Start luci...                               [  OK  ]
 Point your web browser to https://node4:8084 (or equivalent) to access luci
```

Sammuta verkonhallintatyökalu (Network Manager) ja käynnistä ”vastaava” klusterin hallintatyökalu (Cluster Manager)

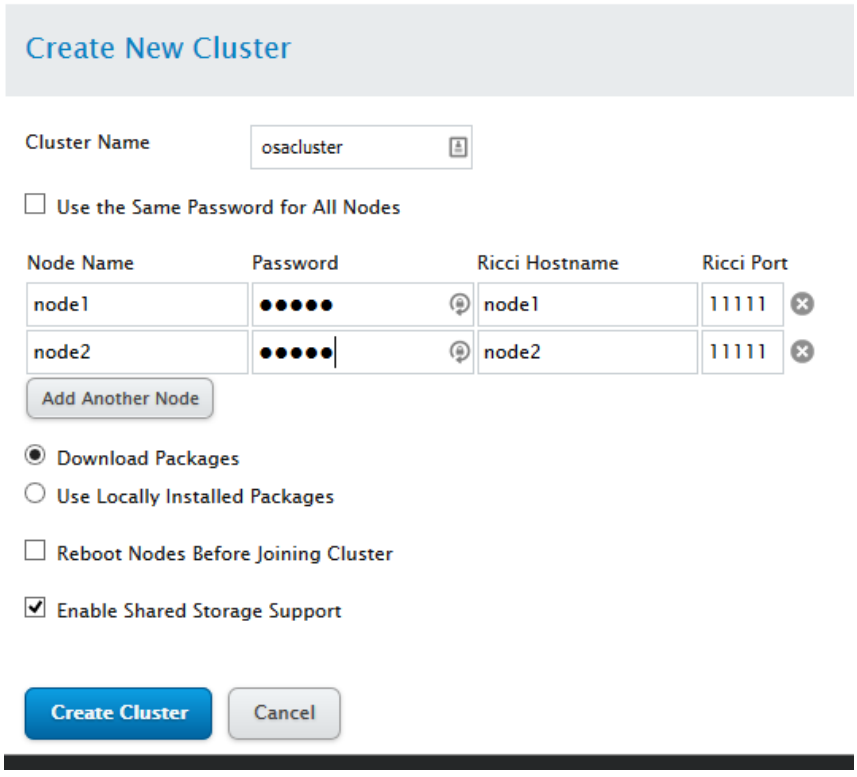
```
service NetworkManager stop
service cman start
```

Kirjauduin luciin osoitteessa https://node4:8084

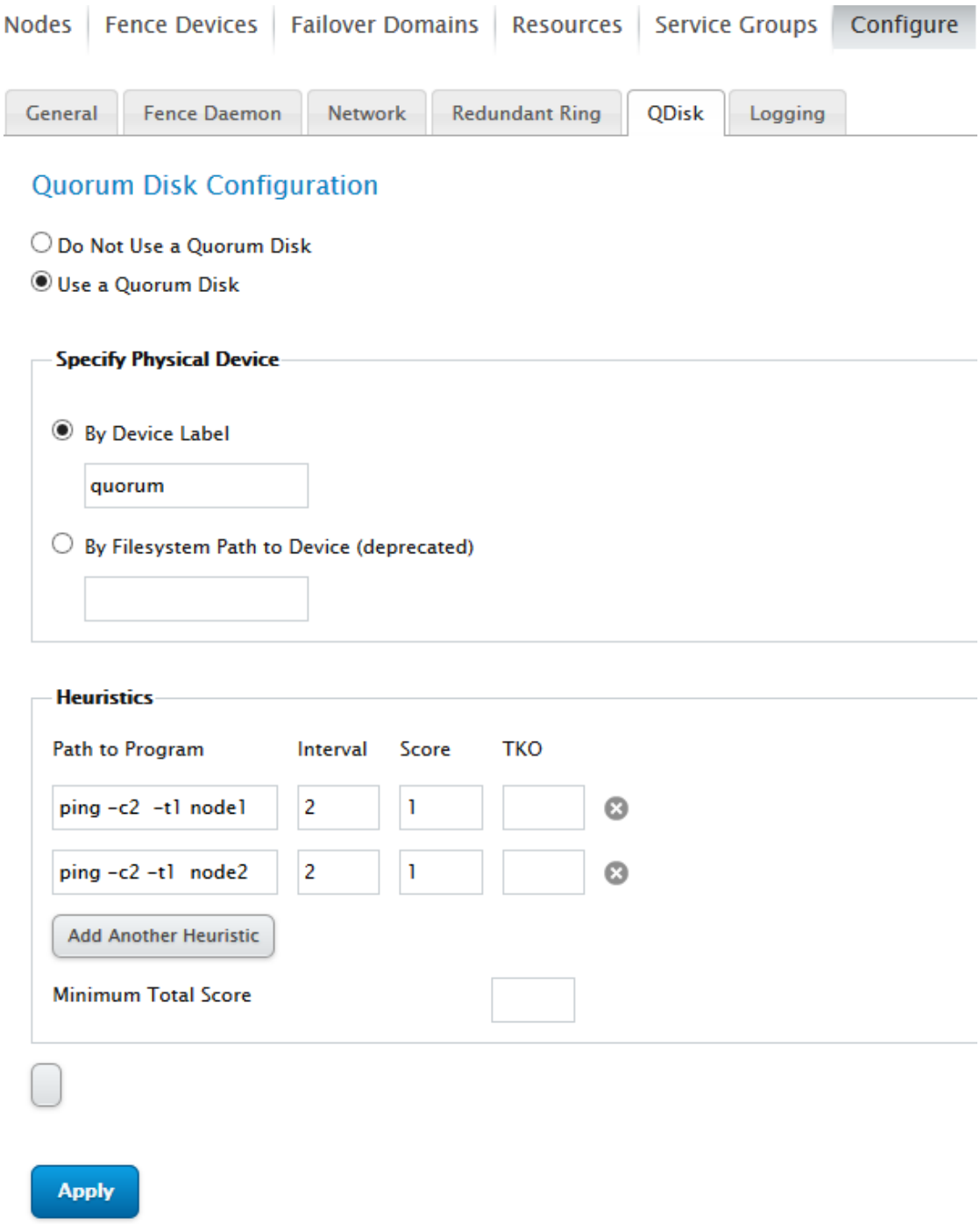


KUVA 3. Ruudunkaapaus lucin kirjautumisruudusta

Valitsin Manage clusters, Clusters, Create ja tein kuvan mukaiset asetukset

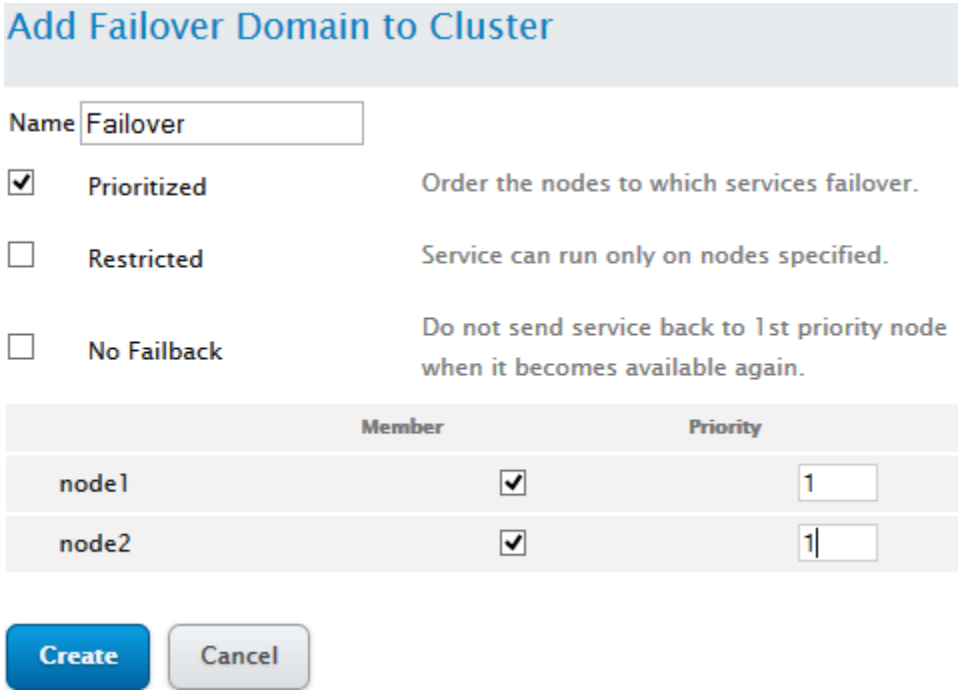


KUVA 4. Klusterin luonti



KUVA 5. Quorum asetukset

Klusterin luonnin jälkeen tee vikasietoisuusasetukset.



KUVA 6. failover prioriteettiasetukset

Loin hakemiston opennebulan datastorea varten mkdir /var/lib/one/datastore

Lisäsin klusteroidun gfs2 levyn resursseihin

Ressources, add, GFS2

Name:	osacluster
Mount Point:	/var/lib/one/datastore
Device, FS Label, or UUID:	/dev/mapper/mpatha
Filesystem Type:	GFS2
Mount Options:	
Filesystem ID (optional):	
Force Unmount:	[x]
Enable NFS daemon and locking workaround:	[ ]
Reboot Host Node if Unmount Fails	[ ]

Loin myös service groupin

Valitsin Service Group, Add ja syötin seuraavat tiedot:

Service Name [osacluster]

Automatically Start This Service [x]

Run Exclusive [ ]

Failover Domain [Failover]
Recovery Policy [Relocate]

Tämän jälkeen klikkasin ”Add Resource”, hetken odotuksen jälkeen clusteri käynnistyi ilmoittaen statuksekseen ”Running on node4”.

Varmistin, että /etc/lvm/lvm.conf tiedostossa on asetettuna ”locking_type = 3”

Klusterin toimintaan saannin jälkeen täytyi asentaa OpenNebula.

OpenNebulan asennus onnistui suhteellisen mutkattomasti. Ensiksi täytyi ottaa käyttöön testausvaiheessa ollut “OpenNebula Cloud for CentOS - testing”-pakettilähde, varmistaa, että lähde on onnistuneesti lisätty komennolla “yum repolist” ja asentaa yum:lla paketit opennebula-server, opennebula-sunstone ja opennebula-node-kvm.

Tämän jälkeen tuli ongelmaksi se ettei portissa 9869 pyörivään hallintaliittymään päässyt kiinni selaimella. Ongelman syyksi selvisi /etc/one/sunstone-server.conf –tiedosto, joka rajasi yhteydet vain paikalliseen koneeseen. Tiedostosta täytyi muuttaa rivi “:host: 127.0.0.1” muotoon “:host: 0.0.0.0” sekä sammuttaa iptables softapalomuuri “service iptables stop” komennolla jolloin yhteydet kaikkialta on sallittu (tämä kuitenkin rajoittui verkkojen välissä olevien palomuurien vuoksi sisäverkkoon). Otin myös palomuurin pois alussa käynnistyvistä ohjelmista komennolla “chkconfig iptables off”.

Nyt palvelimella täytyi käynnistää itse opennebula. Lisäksi käynnistin ensimmäiselle palvelimelle sunstonen tehdäkseni siitä hallintasolmun. Kokeilin myös kirjautua sunstonen kautta ja huomasin ettei salasana olekaan vakio, vaan se löytyy oneadmin käyttäjän (käyttäjä jolla opennebulaa yleensä ajetaan) kotihakemiston alta polusta /var/lib/one/.one/one_auth. Huomasin myös, että hallintatunnustakin voi muokata kyseisestä tiedostosta.

Tämän jälkeen sama tehtiin toisillekin palvelimille, tässä vaiheessa huomasin, että solmut tarvitsevat oneadmin tunnukselle salasanattoman, ssh-avaimella toimivan ja esteettömän pääsyn toisilleen. Avaimen luonti solmukoneille tapahtui komennolla ssh-keygen -t rsa (suoritetaan jokaisella solmulla erikseen) ja tämän jälkeen avaimen li-

sääminen tunnettujen listaan tapahtui komennolla ssh-copy-id -i ~/.ssh/id_rsa.pub oneadmin@ip (suoritettuna oneadmin tunnuksella).

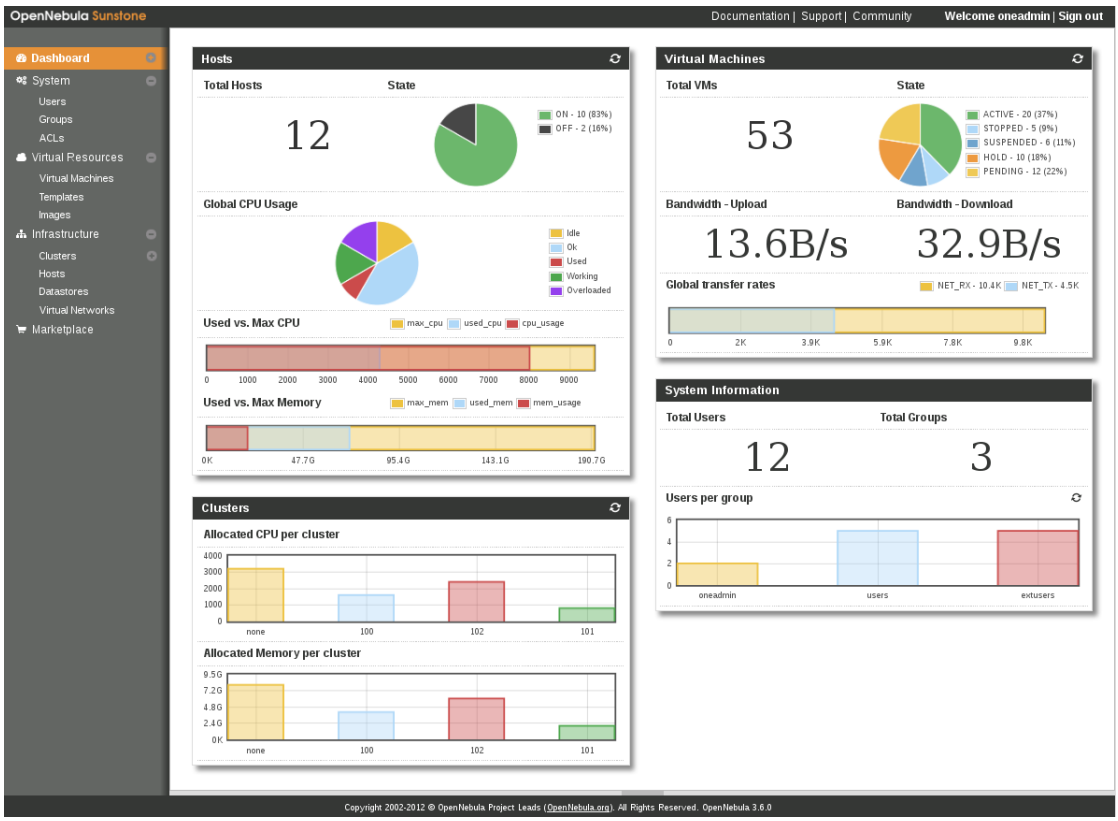
6 OPENNEBULAN KÄYTTÖ

Opennebulaa voi käyttää kahdella tapaa, päätteestä tai selaimella Sunstone käyttöliittymän kautta. Yritän tässä esitellä perusasiat läpi, mutta jos aiot testata Opennebulaa tai eritoten, jos aiot asentaa sen tuotantoympäristöksi niin kannattaa käydä silmäilemässä viralliset ohjeet läpi.

Sunstonessa voi tehdä lähes kaiken minkä päätesovelluksessakin.

Sunstone hallintapaneeli

Sunstonen hallintapaneelissa näkee järjestelmän tilan nopeasti ja helposti. voit mm. tarkkailla: solmukoneiden kokonaismäärää, tilaa, suorittimien rasitusta, muistin käyttöä, virtuaalikoneiden määrää, kaistankäyttöä, klustereiden tilaa ja muuta näihin liittyvää. OpenNebulassa voit myös luoda virtuaalikoneiden lisäksi virtuaalisia verkkoja ja rajoittaa niihin pääsyn tiettyihin käyttäjiin tai ryhmiin. Hienosäätöjä oikeuksiin voi tehdä kohdassa ”ACLs”.



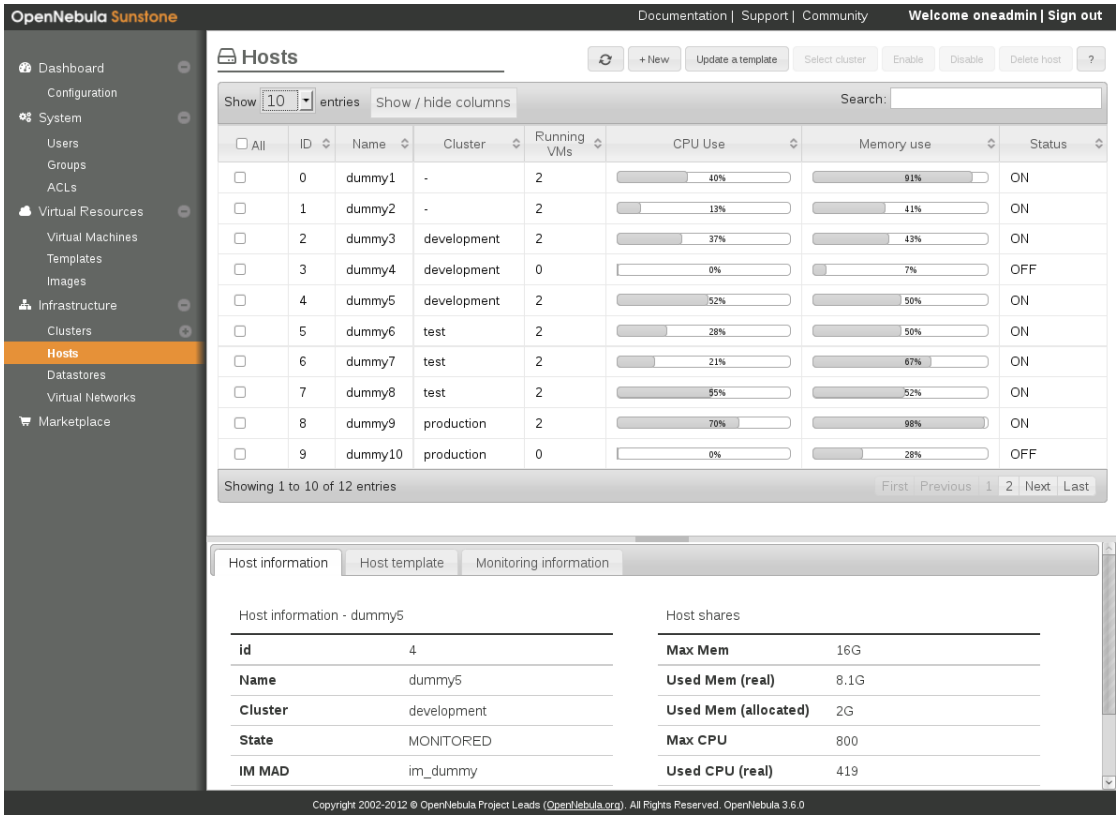
KUVA 7. OpenNebula Sunstone hallintapaneeli

Samat tiedot saisi näkyviin päätteessä komennoilla onehost list, onevm list, onecluster list ja niin edelleen, mutta web -käyttöliittymä on asiakaspäässä helpompi ja nopeampi omaksua.

OpenNebulan perusajatus on, että se tukee useita tekniikoita, hypervisoreita ja tallennusratkaisuja. Lisäksi asetukset olisi tarkoitus pystyä aina tekemään OpenNebulan tai Sunstone käyttöliittymän kautta ilman, että sinun tarvitsee varsinaisesti päästä käsiksi virtuaalikoneeseen. Tämän vuoksi koneet täytyy valmistella. Suurin osa avoimen lähdekoodin käyttöjärjestelmistä onkin ladattavissa valmiina kauppapaikasta eli ”marketplacesta”. OpenNebulan dokumentaatiossa on myös ohjeet koneiden valmisteluun. Tästä syystä johtuen OpenNebula hyödyntää myös Libvirt-rajapintaa.

Levykuviin pääsee käsiksi ”images” kohdasta. Siellä voi olla asennuskuvia sekä valmistelemattomien että valmisteltujen koneiden kuvia. Jotta pystyt käyttämään koneita mahdollisimman vaivattomasti, kannattaa valmistella levykuvat ja verkko kunnolla niin, että virtuaaliverkko voidaan asettaa käytettäväksi templaatissa (pohjassa) jolloin koneita voidaan luoda useampia kerralla ilman, että asetuksiin tarvitsee tehdä käsin muutoksia. Jos et kuitenkaan pysty luomaan / käyttämään virtuaaliverkkoa, niin sinun

täytyy tehdä asetukset koneisiin käsin tai suunnitella oma scripti. Tämä kuitenkin lisää huomattavasti ”turhan” työn määrää. Lisäksi solmukoneisiin täytyy useimmiten luoda verkkosillat. Sunstonesta koneisiin pääsee käsiksi html5:llä toteutetun VNC etäkäyttö-työkalun avulla (tämä ei jokaisella selaimella toiminut toivotulla tavalla, sillä kursorin kuva ei ollut oikealla kohdalla). Levykuvat näkyvät saman tyyppisenä listana kuin kaikki muutkin resurssit. Päätteessä ne saa näkyviin komennolla oneimage --list.



KUVA 8. Solmut

Solmukoneita pääsee tarkastelemaan kohdasta ”Hosts” [kuva opennebulan ohjeesta] Päätteessä listan solmukoneista ja niiden tiloista saa komennolla onehost --list ja tarkemmat konekohtaiset tiedot komennolla onehost show <koneen tunniste>.

Verkon luonti

Virtuaaliverkkoa varten tarvitaan verkkosilta. Loin sen muokkaamalla verkkoasetuksista kahta tiedostoa.

Tiedoston /etc/sysconfig/network-scripts/ifcfg-eth0 sisältö:

DEVICE="eth0"
BOOTPROTO=none
ONBOOT=yes

HWADDR=5C:F3:FC:3F:37:0C
BRIDGE=virbr0

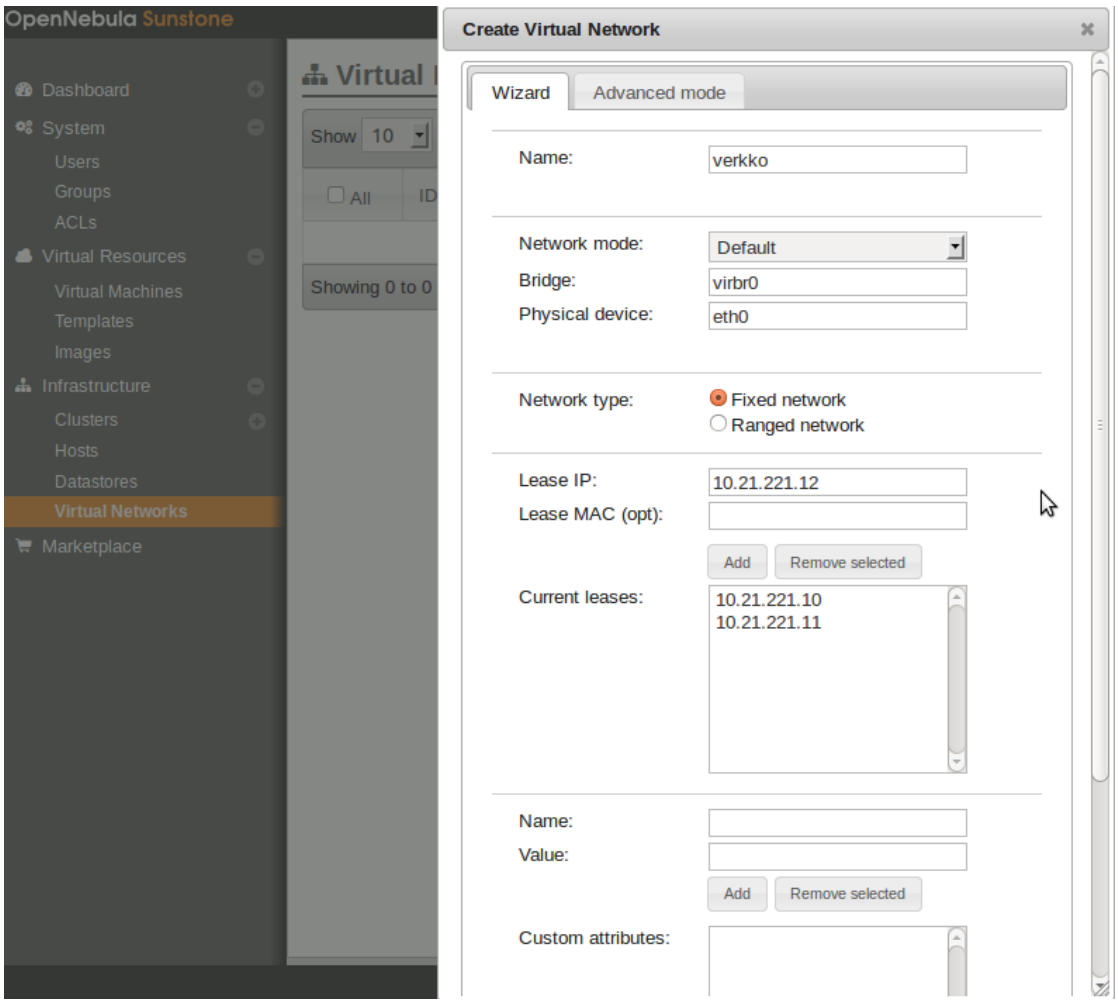
Tiedoston /etc/sysconfig/network-scripts/ifcfg-virbr0 sisältö:

DEVICE=virbr0
TYPE=Bridge
ONBOOT=yes
DELAY=0
BOOTPROTO=static
IPADDR=10.11.112.34
NETMASK=255.255.255.0
GATEWAY=10.11.112.1
DNS1=195.148.233.203
DNS2=195.148.235.200

Muutosten jälkeen joudutaan verkkotoiminnot käynnistämään uudelleen komennolla:
service network reboot

Sillatun virtuaaliverkon voi luoda sunstonessa seuraavasti:
Valitse vasemmalta sivupalkista Virtual Networks ja klikkaa nappia New.

Anna verkolle nimi. Tässä tapauksessa annoin nimeksi verkko. Käytin tässä vaiheessa vielä Network modessa default, eli ”vakio” asetusta. Sillan nimi oli virbr0 ja fyysinen verkkolaite eth0. Kyseessä on fixed verkko ja sen kanssa on omat taistelunsa. Lease IP:t ovat osoitteita, jotka voidaan antaa koneiden käytettäväksi kyseisestä verkosta. Listauksen tarkoitus on estää kiusanteko päällekkäisillä osoitteilla.

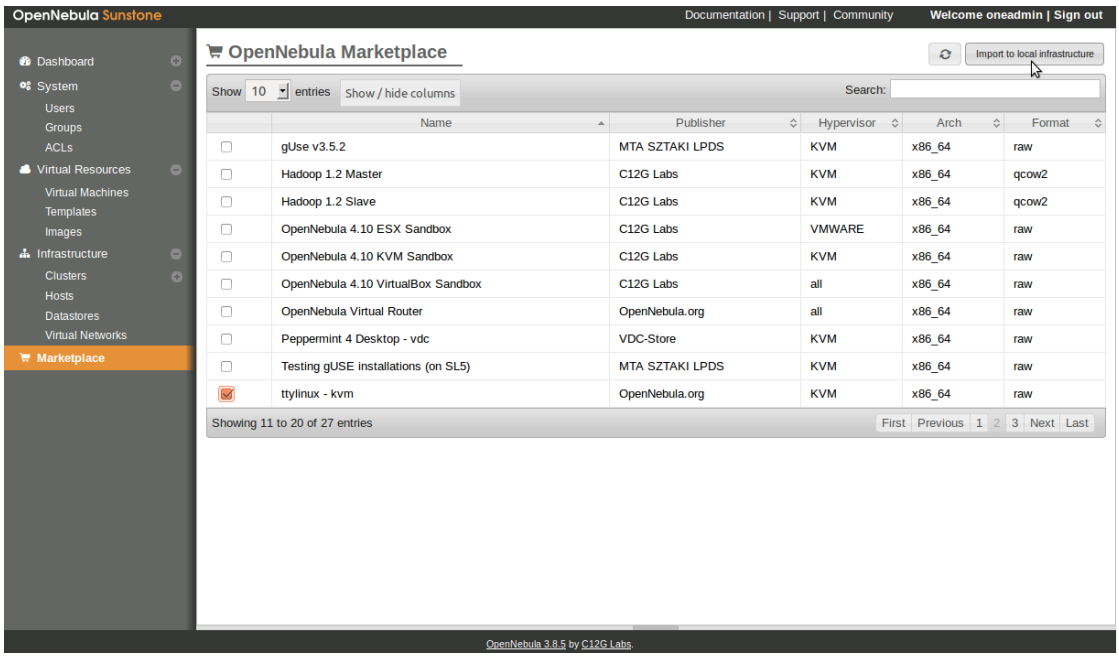


KUVA 9. Virtuaaliverkon luonti

Kuva 10. Create -nappi

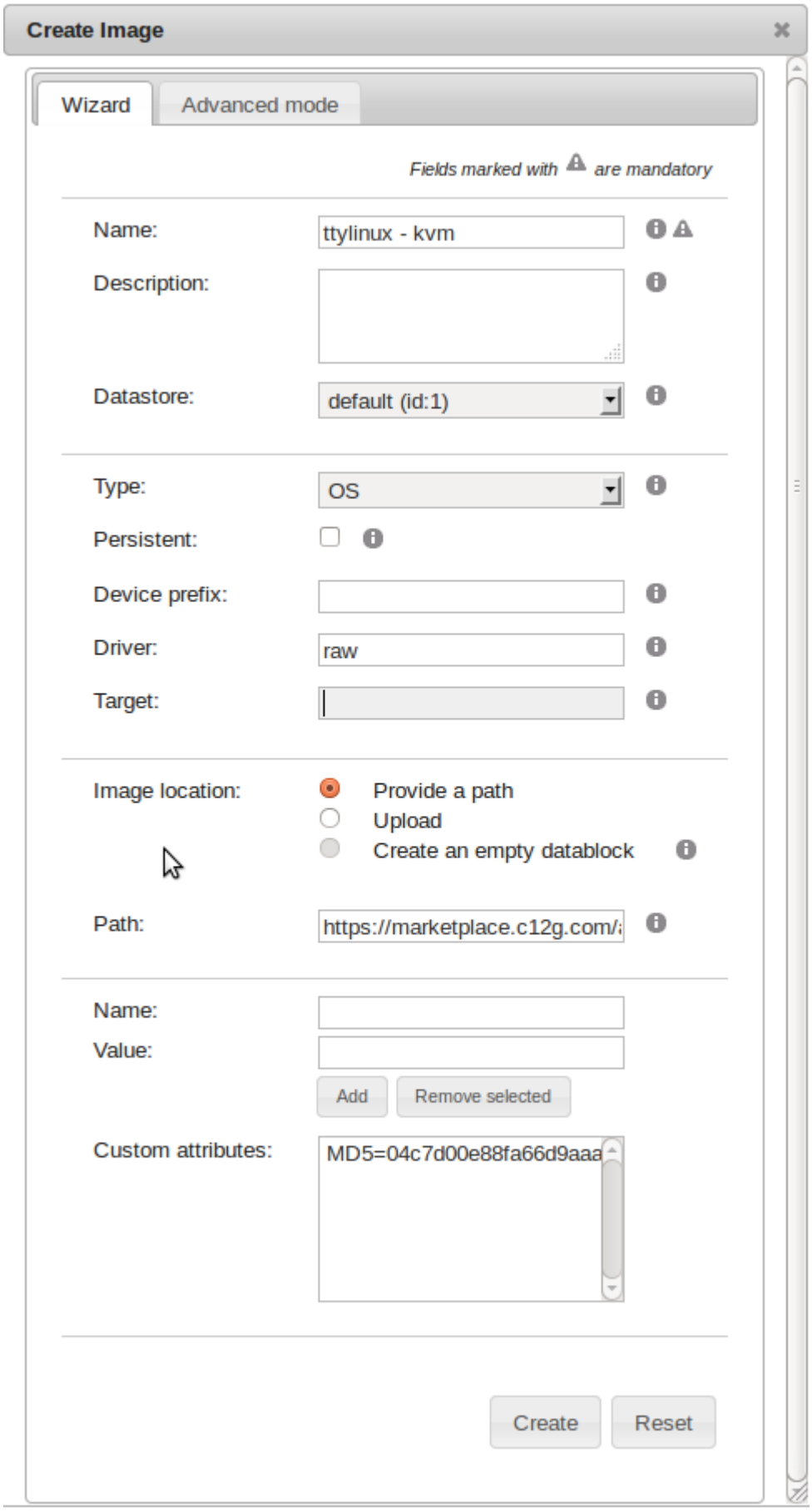
Valmiita koneita marketplacesta

Marketplace on toimiessaan OpenNebulan yksi parhaista ominaisuuksista



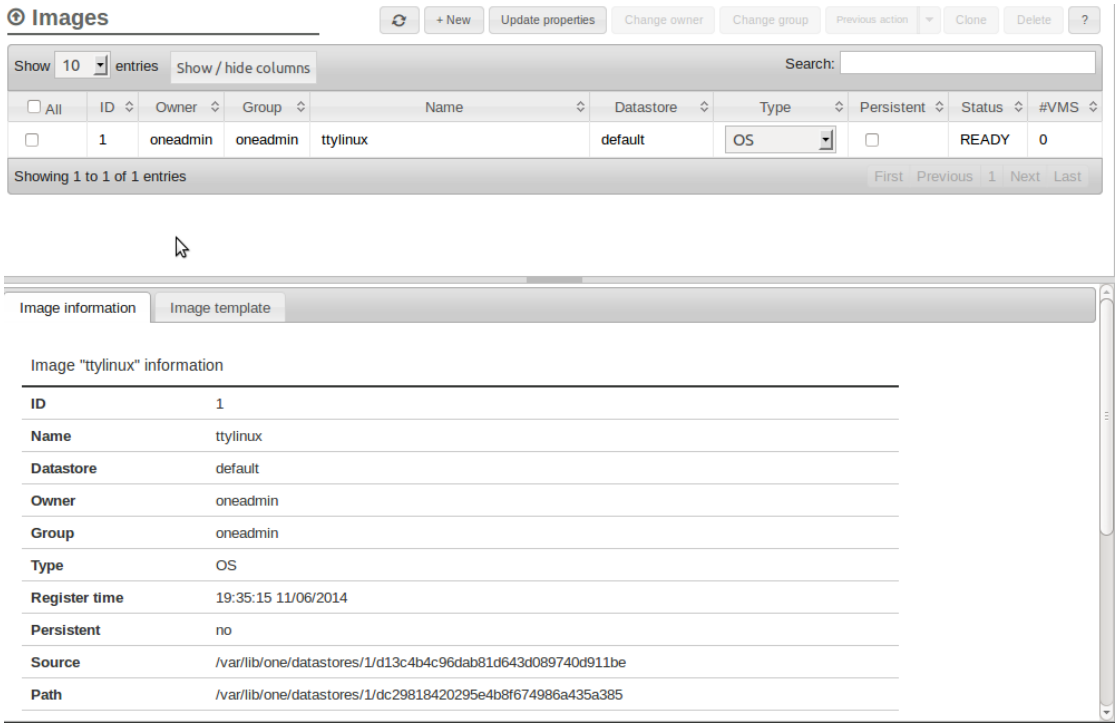
KUVA 11. Marketplace ttylinux – kvm valittuna

Marketplacesta löytyy monia käyttöjärjestelmiä (pääasiassa open sourcea) valmiina. Tarvitsee vain katsoa että valitsee oikean ympäristön imagen, rastittaa haluamansa ja klikkaa ”import to local infastructure”, täyttää lomakeen ja painaa create.



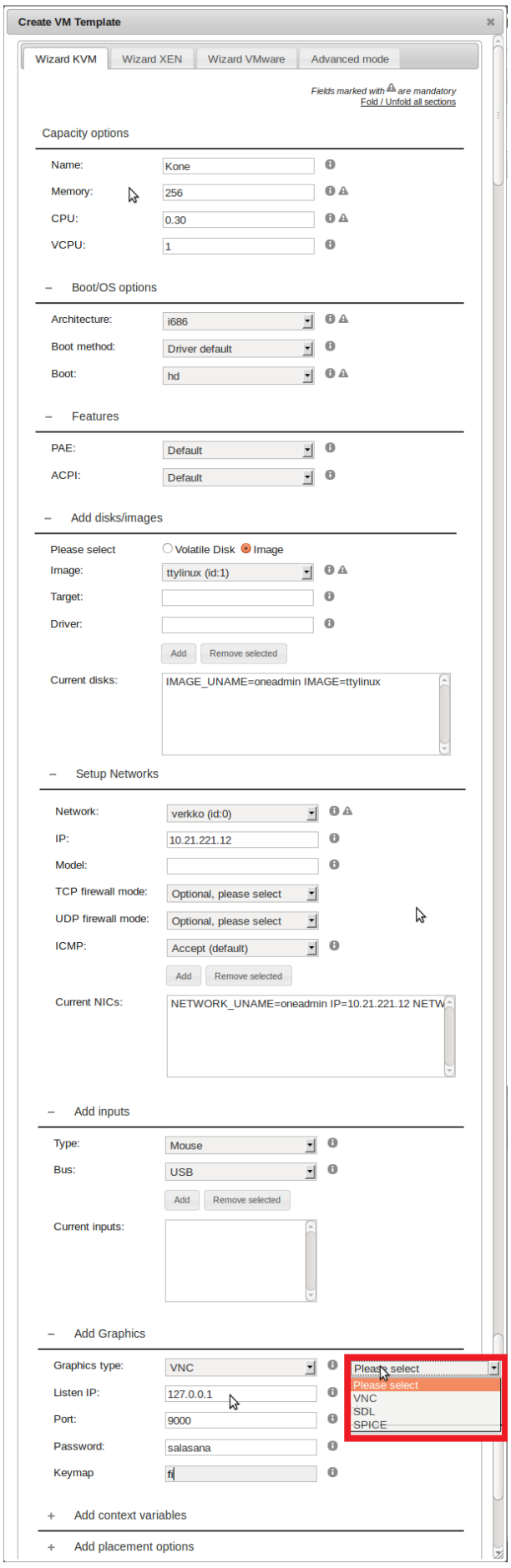
KUVA 12. Opennebula imagen marketplace luontilomake

Tuontilomakkeen sisältö riippuu ympäristöstä ja käyttäjän oikeuksista. Perusasennuksen järjestelmänvalvojan ei yleensä tarvitse täyttää mitään, kunhan ympäristö on oikea.



KUVA 13. Käyttövalmis image

Kun image on tuotu, voidaan keskittyä templaattien tekemiseen. Templaateissa määritellään koneen resurssit ja asetukset.

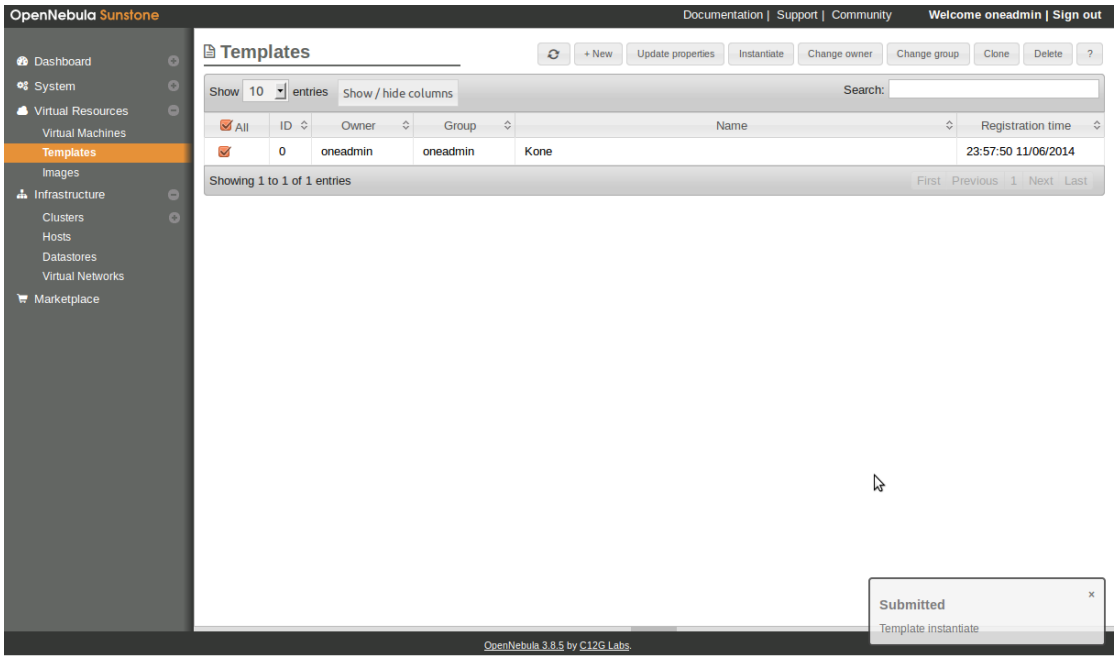


KUVA 14. Templaatin tekeminen valintojen maailmassa

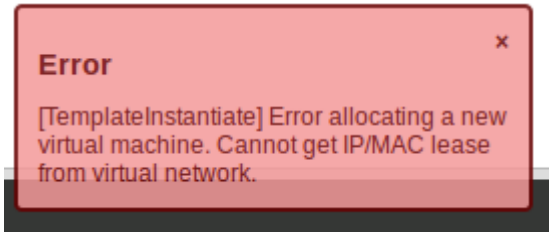
Templaatin tekemisessä kannattaa tutustua OpenNebulan käyttöohjeeseen. Yllä olevassa kuvassa on laitettu vain lähes välttämättömät asetukset. Templaatin tekeminen ja ”templatiointi” ovat eri asioita ja jälkimmäinen ei ole aivan helpoimmasta päästä.

Templaatin ”käyttöönotto” (instantiate)

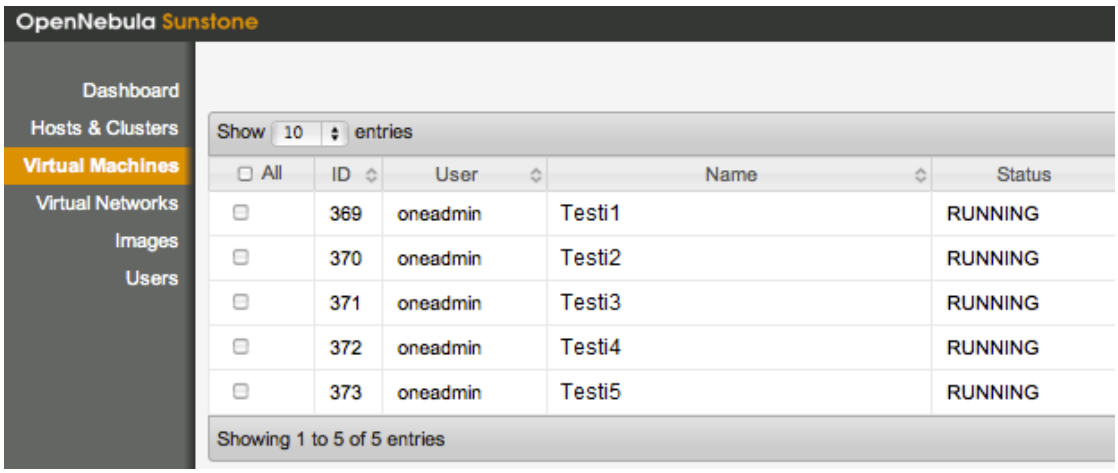
Templaatin käyttöönotto tai toisin sanoin koneen luominen tapahtuu niin että valitset templaattilistastasta haluamasi templaatin ja klikkaat instantiate. Opennebula luo tämän jälkeen templaatin pohjalta uuden koneen jollekin vapaalle solmulle sen mukaan missä on riittävästi tyhjää. Koska verkko on ”fixed”, niin uutta konetta luodessa templaattiin täytyy muuttaa IP, jotta alla olevan kuvan kertomaa IP ongelmaa ei ilmenisi. Asian voi myös hoitaa määrittelemällä IP-alue, mutta tämä ei välillä toiminut ja syykin jäi selvittämättä isompien ongelmien takia.



KUVA 15 Templaatin käyttöönotto



KUVA 16. Opennebula kertoo näin mikäli koitat käyttää IP:tä, joka ei ole leaselistalla tai joka on jo käytössä



KUVA 17. Oikein järjestetyssä ympäristössä voit luoda koneita tarvittaessa erissä

7 TULOKSET

Lopputuloksena oli järjestelmä, joka ei ollut täysin toimiva eikä näin ollen tuotantokäyttöön soveltuva. Työstä saatiin myös arvokasta kokemusta, jotta pystyttiin valitsemaan uusi suunta ja uusi ympäristö. Ongelmia tuli eniten keskeneräisen OpenNebula kuitulevyajurin kanssa. Tämän vuoksi OpenNebula ei osannut välillä käynnistyä tai se kadotti koneet. Myös palvelimien uudelleenkäynnistämisessä tuli ongelmia, jotka monesti kerkesin kuvitella ratkaisseeni. Sama toistui myös käytettäessä Red Hat klusterointia OpenNebulan alla. Ratkaisu osoittautui todella huonoksi jo pelkästään siitä syystä, että tehoa meni paljon hukkaan ja syntyi pullonkauloja. Koetin myös päivittää OpenNebulan versioon 4 ja kokeilla ajureiden uusimpia testiversioita ennen lopullista ajanpuutteesta johtunutta kuolemanlaukausta ja vaihtoa eri ympäristöön.

8 JOHTOPÄÄTÖKSET

Vikasietoisuus ei ollut riittävä kuitulevyä (fibre channel) käytettäessä. Tosin mikään virtualisointiratkaisu ei näytä tukevan sitä hyvin. Opin aiheesta reilusti kantapään kautta, koska en alkuvaiheessa ymmärtänyt miten fibre channel varsinaisesti eroaa muista tallennusratkaisuista. Yksittäisessä koneessa OpenNebula kuitenkin toimii kuten pitääkin. Tein asennuksesta myös ohjeet, mutta niiden avulla onnistuu vain yksittäisen OpenNebula koneen pystyttäminen. Klusterointia ei valitettavasti kannata vaivautua asentamaan.

Opinnäytetyön aiheena virtualisointi on melko mielenkiintoinen, mutta todella laaja alue. Lisäksi siihen liittyy aivan järkyttävä määrä muita monimutkaisia asioita. Opinnäytetyöni käytännönosuuden tekstistä ei tullut aivan niin hyvä asian laajuudesta, monimutkaisuudesta ja käytännön ongelmista johtuen.

LÄHTEET

Allan Murphy. 2008. VIRTUALIZATION DEFINED - EIGHT DIFFERENT WAYS. http://www.f5.com/pdf/white-papers/virtualization-defined-wp.pdf Päivitetty: 2008 Luettu: 28.07.2014

CentOS project. 2014. CentOS Tips and Tricks. WWW-dokumentti. http://wiki.centos.org/TipsAndTricks Päivitetty 27.10.2014 Luettu 27.10.2013

Everything VM. 2011. History of Virtualization. WWW-dokumentti. http://www.everythingvm.com/content/history-virtualization. Päivitetty 01.08.2011. Luettu 20.04.2014.

IBM Systems Lab Services and Training group. 2011. Jim Rymarczyk, IBM Fellow talks about virtualization. Technical University: IBM Systems Lab Services and Training group. Video. Päivitetty 19.7.2011. Katsottu 07.05.2014.

Joe Brockmeier. 2010. KVM or Xen? Choosing a Virtualization Platform. WWW-dokumentti. http://www.linux.com/news/enterprise/systems-management/327628-kvm-or-xen-choosing-a-virtualization-platform Päivitetty 12.12.2010 Luettu 07.05.2014.

Libvirt project. 2013. The virtualization API. Päivitetty 2013 Luettu 27.10.2013. Saatavissa: http://libvirt.org/

Microsoft. Ei ilmoitettu. Server and Cloud Platform. WWW-dokumentti. http://www.veeam.com/blog/hyper-v-server-2012-or-windows-server-2012-with-hyper-v.html. Päivitetty 14.05.2013. Luettu 07.05.2014.

OpenNebula Project (OpenNebula.org). 2013. Opennebula 3.8 guides. WWW-dokumentti. http://archives.opennebula.org/documentation:archives:rel3.8 Päivitetty 20.12.2013. Luettu 27.10.2013.

Peter Mell, Timothy Grance. 2011. The NIST Definition of Cloud Computing. PDF - dokumentti. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf Päivitetty 2011. Luettu 15.08.2014.

QEMU project. 2014. QEMU open source processor emulator. Päivitetty: 25.11.2014 Saatavissa: http://wiki.qemu.org/

Simon Rimblock. 2013. Virtualization: A long brief history. WWW-dokumentti. http://www.servethehome.com/virtualization-long-history. Päivitetty 28.02.2013. Luettu 28.07.2014.

Tanenbaum, Woodhull 2006. Operating Systems Design and Implementation 3rd Edition. Massachusetts: Prentice Hall.

William Stallings 2012. Operating Systems Internals and Design Principles 7th Edition:New Jersey: Prentice Hall.

WMware, Inc. 2007. Understanding Full Virtualization, Paravirtualization, and Hardware Assist. http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf

OpenNebula Project (OpenNebula.org). KUVA 7. OpenNebula Sunstone hallinta-paneeli. 05.10.2014. Saatavuus: http://archives.opennebula.org/_media/documentation:rel3.6:sunstone:view_dashboard.png

OpenNebula Project (OpenNebula.org). KUVA 8. Solmut. 05.10.2014. Saatavuus: http://archives.opennebula.org/_media/documentation:rel3.6:sunstone:view_hosts.png

LIITE 2(1).

Monisivuinen liite