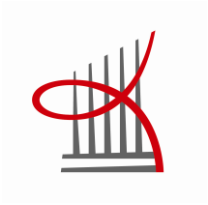


Virtualisoitu palvelinympäristö pilvipalvelun alustana

Ville Lehto

Opinnäytetyö
Elokuu 2013
Tietojenkäsittelyn koulutusohjelma
Tietoverkkopalvelut

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Tietoverkkopalvelut

LEHTO,VILLE:
Virtualisoitu palvelinympäristö pilvipalvelun alustana

Opinnäytetyö 50 sivua, joista liitteitä 2 sivua
Elokuu 2013

Opinnäytetyön tavoitteena oli virtualisointia ja pilvijärjestelmiä tutkimalla luoda käsitys virtualisointitekniikoista ja virtuaalisille alustoille asennettujen pilvipalveluiden toiminnasta. Pilvipalveluihin tutustumalla selvitettiin samalla pilvipalvelualustan suunnitteluun ja toteuttamiseen liittyviä haasteita. Lopuksi toimeksiantajalle Toyme Lab Oy:lle tuottamaa pilvipalvelusovellusta varten toteutettiin virtualisoitu palvelin vapaan lähdekoodin ohjelmistoja hyödyntäen.

Toyme Labin tuottama Toyme-järjestelmä edustaa pilvipalveluita, jotka yleistyvät yritysmaailmassa jatkuvasti. Tämä asettaa uusia vaatimuksia palvelualustojen tehokkuudelle, joustavuudelle ja tietoturvatasolle. Tätä varten toimeksiantaja halusi ottaa käyttöön virtualisoidun konesalin, jonka toteutuksesta yrityksessä ei ennestään ollut kokemusta.

Työssä tutkittiin virtualisointia käsitteenä ja esiteltiin virtuaaliseen ympäristöön liittyviä tekniikoita ja ohjelmistoja sekä eri virtuaalikonetyyppejä. Tutkimuksen kohteena olivat myös pilvipalveluiden eri mallit sekä niiden soveltuvuus organisaatioille. Samalla esiteltiin yleisimpiä vapaan lähdekoodin infrastruktuuripalveluja. Palvelinympäristön toteutus suoritettiin asentamalla hypervisor-ohjelmisto KVM Ubuntu 12.04-käyttöjärjestelmään. Tälle alustalle määriteltiin ensin virtualisointiin vaadittavat asetukset, jonka jälkeen luotiin eri tavoilla muutamia virtuaalikoneita ja tutkittiin niiden ominaisuuksia. Osa virtuaalikoneista otettiin käyttöön osaksi Toyme-palvelun testausta.

Opinnäytetyön tuloksena syntynyt virtualisoitu ympäristö on toimeksiantajalla osoittautunut onnistuneeksi ratkaisuksi ohjelmiston käyttöönottovaiheessa. Kehitystyö palvelimen käytön monipuolistamiseksi jatkuu edelleen opinnäytetyön valmistumisen jälkeen.

Asiasanat: pilvipalvelu, virtualisointi, ubuntu, kvm

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Data Network Services

LEHTO, VILLE:
Virtualized server as a Cloud Computing Platform

Bachelor's thesis 50 pages, appendices 2 pages
August 2013

The objective of the thesis was to find out how virtualized cloud computing services work by examining virtualization techniques and different cloud systems. Moreover, challenges in designing and implementing cloud service platforms were investigated. Finally, for the client Toyme Lab, a virtualized open-source based server was introduced for utilization with their cloud computing software.

As cloud computing services become continuously more common, they require further efficiency, flexibility and security from their platform. The client wished to prepare for this by implementing a virtualized data center, a system of which they had no experience.

In the thesis, virtualization was reviewed as a concept. Other subjects under analysis include virtualization techniques and software as well as cloud computing models and how they suit organizations' needs. Some of the most common open source infrastructure services were also introduced.

The server environment was built by installing KVM hypervisor software under Ubuntu Linux 12.04. After defining virtualization settings, a few virtual machines were created using different installation methods. Some of those machines were later utilized in the Toyme software testing process. The environment was found successful in its role as a software utilization tool, and work will continue in the future on diversifying the services and features of the server.

Key words: cloud computing, virtualization, ubuntu, kvm

SISÄLLYS

1 JOHDANTO 7
2 VIRTUALISOINTI 8
2.1 Mitä virtualisointi tarkoittaa? 8
2.2 Miksi virtualisointi kannattaa? 8
2.3 Virtuaalikoneet 9
2.4 Hypervisorit 11
2.5 KVM, QEMU ja libvirt 12
3 PILVIPALVELUT 14
3.1 Pilvipalveluista yleisesti 14
3.2 Käyttöönottotyypit 15
3.2.1 Public cloud 15
3.2.2 Private cloud 16
3.2.3 Community cloud 16
3.2.4 Hybrid cloud 17
3.3 Palvelumallit 18
3.3.1 Software as a Service 18
3.3.2 Platform as a Service 18
3.3.3 Infrastructure as a Service 19
3.4 IaaS – palveluja 19
3.4.1 OpenStack 20
3.4.2 Eucalyptus 22
3.4.3 Apache CloudStack 24
3.4.4 Nimbus 25
3.4.5 OpenNebula 26
4 YMPÄRISTÖN SUUNNITTELU 27
4.1 Toimeksiantajan esittely 27
4.2 Lähtökohdat, tavoitteet ja tarkoitus 27
5 TOTEUTUS 29
5.1 Laitteisto ja käyttöjärjestelmä 29
5.2 Ympäristön asennus 29
5.3 Hypervisorin asennus 32
5.4 Verkkoyhteyden siltaaminen ja palomuuri 33
5.5 Virtuaalikoneiden luominen 35
5.6 Virtuaalikoneiden käsittely 37
5.7 Palvelinohjelmistojen asennus 40
5.7.1 Apache Tomcat 40

5.7.2 Apache HTTP server 42
5.7.3 MySQL ja phpMyAdmin 42
6 POHDINTA 43
LÄHTEET 45
LIITTEET 49

LYHENTEET JA TERMIT

HTTP	Hypertext Transfer Protocol, web – palvelimissa käytetty tiedostonsiirtoprotokolla.
RAID	Redundant Array of Independent Disks, tekniikka, jota käytetään kiintolevyjen yhdistämiseksi loogisesti nopeuden, vikasietoisuuden tai molempien kasvattamiseksi.
SSH	Secure Shell, tietoliikenneprotokolla salattujen yhteyksien muodostamiseen.
TLS	Transport Layer Security, yleisesti web – sivujen suojaukseen käytetty salausprotokolla.
VPN	Virtual Private Network, tapa liittää verkko tai päätelaite yksityiseen verkkoon Internetin tai muun julkisen verkon yli.
XML	Extensible Markup Language, merkintäkieli, jota käytetään selkeän rakenteensa vuoksi formaattina tiedonsiirrossa ja dokumenttien tallennuksessa.

1 JOHDANTO

Pilvipalvelut ovat ehtineet lyhyessä ajassa mullistamaan Internetin käyttäjien ajattelutavan tuomalla esiin ominaisuuksia, joista ennen vuosituhannen vaihdetta ei osattu uneksiakaan. Kuluttajille palvelut kuten Spotify, Netflix, Steam tai Dropbox ovat jo itsestäänselvyyksiä, mutta yritysmaailmassa siirtyminen tehokkaaseen pilvipalveluiden hyödyntämiseen on vielä kesken, ellei vasta aluillaan. Tämä muutos näyttää kuitenkin väistämättömältä, ovathan eri palvelutyyppeinä toteutetut pilvijärjestelmät jo todistaneet ylivertaisuutensa kustannustehokkuuden, dynaamisuuden ja toimintavarmuuden suhteen perinteisiin IT – toimintamalleihin verrattuna.

Tärkeimpänä yksittäisenä teknologiana pilvipalveluiden yleistymisen mahdollistamiseen voidaan pitää virtualisointia, joka on murtanut mallin yksittäisestä resurssista yksittäiselle palvelulle ja nostanut sitä kautta huimasti resurssien hyödyntämisastetta sekä laskenut niiden hintoja avaten sitä kautta mahdollisuuksia uusille toimijoille perinteisesti kalliilla liiketoiminta-alueella. Virtualisointi kasvattaa myös palveluiden saatavuutta ja parantaa sitä kautta käyttäjäkokemuksia, mikä puolestaan alentaa entisestään kynnystä virtualisoitujen järjestelmien käyttöönotolle.

Tässä opinnäytetyössä tutustutaan tarkemmin virtualisointiin ja pilvipalveluihin käsitteinä, eritellään niihin liittyvät erityispiirteet ja luodaan katsaus yleisimpiin vapaan lähdekoodin järjestelmiin virtualisoidun palvelininfrastruktuurin pystyttämiseksi. Viimeiseksi toteutetaan virtuaalisen konesalin pystytys KVM – hypervisorin avulla. Työn tavoitteena on tutkia pilvipalveluiden hyötyjä organisaatiotasolla ja selvittää, miten virtualisoidun konesalin rakentaminen ja ylläpitäminen onnistuu vapaan lähdekoodin työkaluilla sekä millaisia hyötyjä se tuottaa toimeksiantajan kaltaiselle pienelle yritykselle. Tarkoitus on samalla edesauttaa toimeksiantajan kehitystyötä laajentamalla tietämystä virtualisoidun konesalin toiminnasta tulevien IT – investointien varalle.

2 VIRTUALISOINTI

2.1 Mitä virtualisointi tarkoittaa?

Tietojenkäsittelyssä virtualisoinnilla tarkoitetaan jonkin fyysisen komponentin muuttamista loogiseksi kohteeksi (Portnoy 2012, 2). Tekniikkana se on ollut käytössä jo 1960 – luvulta asti, jolloin IBM:n ensimmäisten isojen keskustietokoneiden siihen aikaan valtava laskentateho piti saada jaetuksi järkevästi loppukäyttäjille (WMTech 2011). Nykyisin termi yhdistetään yleisimmin tapaan, jolla laitealustan tai käyttöjärjestelmän fyysiset resurssit piilotetaan resurssien käyttäjiltä. Tilalla näytetään looginen ympäristö, joka uskoo käyttävänsä omia fyysisiä komponenttejaan. Tuloksena toteutunutta, järjestelmän sovelluskerroksella ajettavaa prosessia kutsutaan virtuaalikoneeksi (virtual machine). (IBM 2007.)

Virtuaalikoneiden lisäksi virtualisointimalli voidaan toteuttaa muun muassa seuraavissa käyttötapauksissa:

- Tiedon varastointi: usealle eri alustalle tallennetun tiedon yhdistäminen yhdeksi loogiseksi yksiköksi (logical unit, LUN)
- Tietoverkot: virtuaaliset lähiverkot (virtual local network, VLAN)
- Suurten konesalien resurssien jakaminen kerroksittain virtuaalisesti (Virtual Services Offering, VSO)
- Työpöytien virtualisointi (Virtual Desktop Infrastructure, VDI) (Ruest & Ruest 2009, 27-28.)

2.2 Miksi virtualisointi kannattaa?

Ennen virtualisoinnin yleistymistä ohjelmistot ja prosessit piti jakaa erillisille palvelimille. Tämä johti usein siihen, että suurissa konesaleissa tuotantopalvelinten käyttöaste saattoi jäädä hyvin pieneksi, jopa alle 5 prosenttiin. Alhainen käyttöaste aiheuttaa yrityksille runsaasti tappioita resurssien hukkaantumisen ja suurten ylläpitokustannusten muodossa. Virtualisoinnin ansiosta useiden palvelinten toiminnot voidaan yhdistää jakamalla ne virtuaalikoneisiin, jotka sijaitsevat samalla fyysisellä alustalla. Tämä palvelinten yhdistäminen (server consolidation), jota mitataan yhdistämissuhteella (consolidation ratio), tuottaa merkittäviä säästöjä sekä kasvattaa selvästi kustannustehokkuutta.

Esimerkiksi neljän fyysisen palvelimen yhdistäminen neljäksi virtuaalikoneeksi tarkoittaa yhdistämissuhdetta 4:1, ja säästää yrityksen resursseja kolmen fyysisen palvelimen verran. (Portnoy 2012, 9.)

International Journal of Scientific & Engineering Research – lehden vuonna 2010 tekemässä tutkimuksessa 500 fyysisen palvelimen virtualisoinnissa saavutettiin yhdistämissuhteella 10:1 käyttöasteen kohoaminen 5 prosentista 50 prosenttiin ja energiankulutukseen jopa 87 prosentin säästö (Uddin & Azizah 2010).

Virtualisointi tarjoaa myös mahdollisuuden kasvattaa sovellusten ja palveluiden saatavuutta sekä vähentää käyttökatkoja virtuaalikoneiden kahdentamisella ja siirtämisellä (migration) alustalta toiselle. Näillä toimenpiteillä virtuaalikone voidaan kopioida tai siirtää joko palvelimen sisällä tai kokonaan uudelle fyysiselle alustalle, ja ottaa käyttöön alkuperäistä konetta vastaavassa tilassa. Tämä tarjoaa erinomaiset mahdollisuudet testata uusia ohjelmistoja ja päivityksiä ilman, että vaarannetaan alkuperäisen järjestelmän toimintaa, sekä vikatilanteen sattuessa jatkaa virtuaalikoneen käyttöä uudessa ympäristössä parhaassa tapauksessa ilman katkoksia. Samoin varmuuskopiointi ja palautus onnistuvat useimmiten virtuaalikonekohtaisesti, esimerkiksi VMwaren vSphere- tai KVM:n snapshot- työkaluilla. (VMware 2012; Graziano 2011, 7; Warnke & Ritzau 2010, 222.)

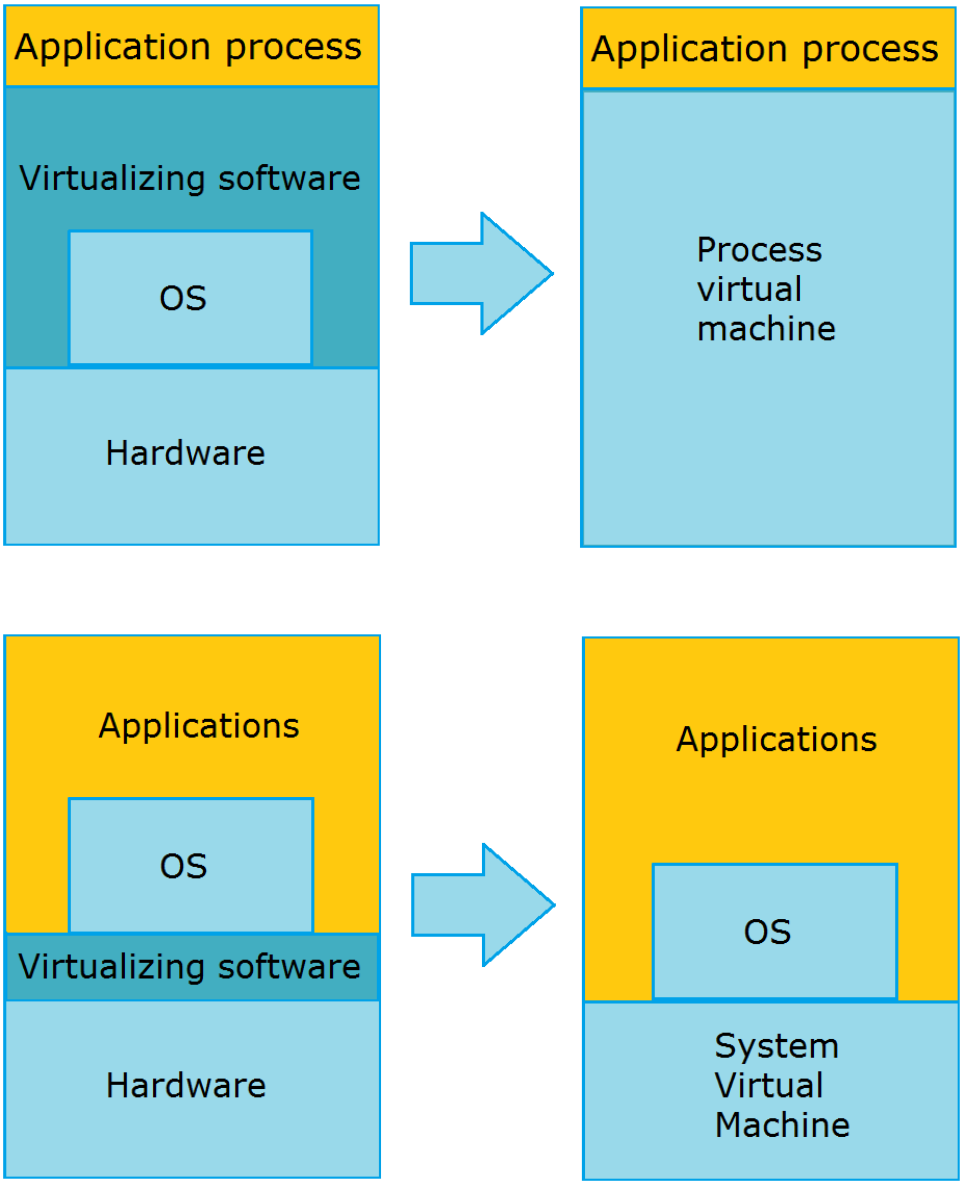
2.3 Virtuaalikoneet

Virtuaalikoneet käyttäytyvät täsmälleen samalla tavalla kuin fyysiset vastinkappaleensa hyödyntämällä virtuaalisen laitealustansa resursseja. Virtuaalikoneet ovat kuitenkin käytöltään joustavampia, sillä niille määriteltyjä resursseja ja jopa toimintaperiaatetta pystytään hallitsemaan käyttötarkoituksen mukaan. Virtuaaliympäristöt luokitellaan käytön perusteella kahteen malliin:

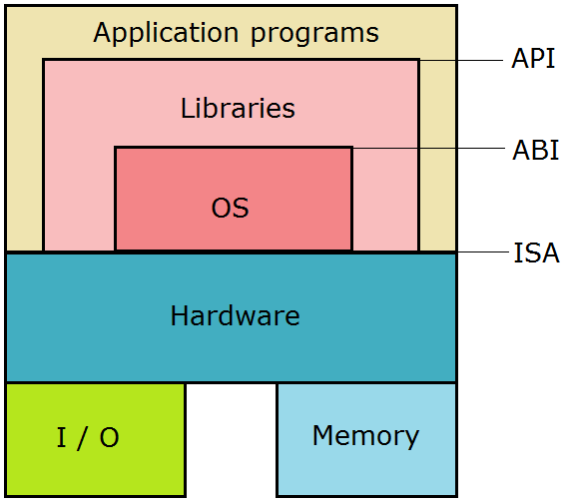
- Process Virtual Machine (PVM), yhtä ohjelmaa varten olemassa oleva, usein kertaluontoinen virtuaalinen prosessi
- System Virtual Machine (SVM), laajempaan käyttöön suunniteltu pysyvä ympäristö yleensä käyttöjärjestelmän alustaksi

Erot näiden kahden virtuaalikonetyypin välillä liittyvät tapaan, jolla virtualisoitu järjestelmä (guest) käyttäytyy alustansa (host) kanssa (kuva 1). SVM:n kohdalla virtuaaliko-

neella on yhteys laitteiston resursseja ohjaavaan instruction set architecture (ISA) – rajapintaan, joka mahdollistaa virtuaalikoneen päälle asennettavan käyttöjärjestelmän. PVM:ssa virtuaalikoneen toteuttava hypervisor - ohjelmisto jäljittelee itsessään käyttöjärjestelmää, jolloin virtuaalikoneella on pääsy vain useimmiten ohjelmistoissa hyödynnettäviin application binary interface (ABI) - ja application programming interface (API) – rajapintoihin (kuva 2). Tästä syystä PVM - mallia käytetään monissa ohjelmointikielissä sovelluksen suorittamiseen eristetyssä, käyttöjärjestelmäriippumattomassa ympäristössä, esimerkkeinä Java Virtual Machine ja Microsoftin .NET – pohjan mahdollistava Common Language Infrastructure. (Smith & Nair 2005.)



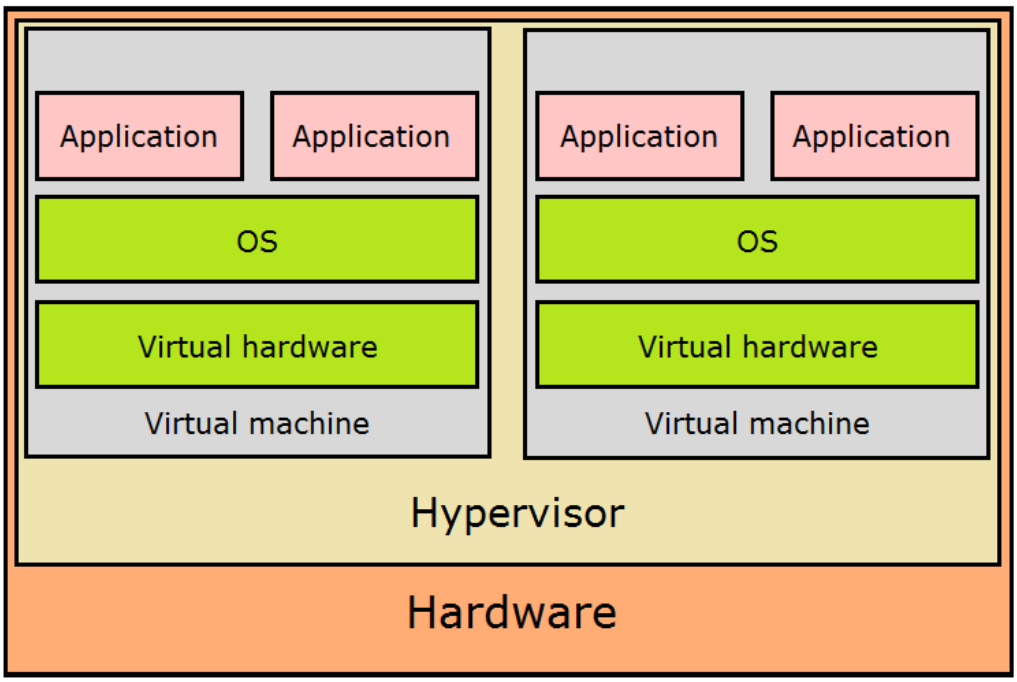
KUVA 1. Virtuaalikonetyyppien vertailu. Mukaillen (Smith & Nair 2005).



KUVA 2. Yksinkertaistettu tietokonejärjestelmän arkkitehtuuri rajapintoineen. Mukaillen (Smith & Nair 2005).

2.4 Hypervisorit

Hypervisoria, josta käytetään joskus myös nimitystä virtual machine manager (VMM), tarvitaan virtualisoinnissa alustaksi virtuaalikoneille (Portnoy 2012, 19). Se on ohjelmisto, jonka tarkoituksena on mahdollistaa fyysisten resurssien jakaminen usean eri virtuaalikoneen kesken luomalla niille virtuaalisen laitteistopohjan (kuva 3). Tämä tapahtuu joko tai suoraan rautatason päälle tyypin 1 tai välillisesti käyttöjärjestelmän alla toimivalla tyypin 2 hypervisorilla (Portnoy 2012, 21–24).



KUVA 3. Hypervisorin toiminta. Mukaillen (Portnoy 2012, 25).

Tyypin 1 hypervisorit pystyvät keskustelemaan suoraan allaan olevan laitteiston kanssa, mikä nopeuttaa selvästi virtuaalikoneiden suorittamia I/O – kyselyitä. Tästä syystä tyypin 1 ohjelmistoista käytetään myös termiä bare metal hypervisor. Näiden hypervisoreiden etuna tyyppiin 2 nähden on tehokkuuden lisäksi vakaus; tyypin 1 hypervisorin alle asennetut virtuaalikoneet eivät vaikuta toisiinsa, joten yhdessä virtuaalikoneessa esiintyvä virhe ei pääse sekoittamaan muiden virtuaalikoneiden toimintaa. Esimerkkejä tyypin 1 hypervisoreista ovat VMware ESX/ESXi, Citrix XenServer, ja Microsoftin HyperV. (Portnoy 2012, 21–22.)

Isäntäkoneen käyttöjärjestelmän päälle asennettua virtuaalikoneita ajavaa ohjelmistoa kutsutaan tyypin 2 hypervisoriksi. Näitä ratkaisuja ei pidetä yhtä tehokkaina kuin tyypin 1 hypervisoreita, koska hypervisorin ja raudan väliin jäävä ylimääräinen kerros, käyttöjärjestelmä, hidastaa merkittävästi hypervisorin toimintaa välittämällä kaikki pyynnöt osapuolten välillä. Myös tyypin 2 hypervisoreiden käyttövarmuus heikkenee isäntäkoneen käyttöjärjestelmän takia, sillä alustan vikatilanteet sekä huolto- ja päivityskatkot vaikuttavat suoraan asennettuihin virtuaalikoneisiin. Toisaalta tyypin 2 vahvuuksiin kuuluvat isäntäkäyttöjärjestelmän tarjoama laaja laitetuki sekä valmiiksi asennetut verkko- ja tallennusratkaisut. Tyypin 2 hypervisoreihin kuuluvat muun muassa VMware Workstation ja VirtualBox. (Portnoy 2012, 23-24.)

2.5 KVM, QEMU ja libvirt

Kernel-based Virtual Machine eli KVM on Qumranetin (nykyisin osa Red Hatia) tuottama GNU General Public Licensellä suojattu hypervisor. Sen käyttöön vaadittavat osat on sisällytetty Linuxin kerneliin versiosta 2.6.20 alkaen, ja käyttöönotto tapahtuu lataamalla se kernel-moduulina useimmiten suoraan jakelupaketin omalla pakettienhallintaohjelmistolla. KVM:n rooli hypervisorina on epäselvä, sillä vaikka se ajetaankin isäntäkäyttöjärjestelmästä, virtuaalikoneet suoritetaan suoraan kernelin päällä, jolloin fyysiset resurssit ovat käytettävissä ilman isäntäkäyttöjärjestelmän väliintuloa. (Graziano 2011, 10; Petersen 2010, 207.)

KVM tukee sekä täyttä laitteiston virtualisointia (full virtualization) että paravirtualisointia, jossa virtuaalikoneen kernelin tulee olla muokattu lähettämään prosessorille suunnatut pyynnöt tavalla, jonka hypervisor osaa tulkita. Täysi muokkaamattomien vir-

tuaalikoneiden ajaminen vaatii isäntäkoneen prosessorilta laitteiston virtualisoinnin mahdollistavan teknologian (Intel VT, AMD-V). Laitteiston virtualisointia KVM ei suorita itse, vaan hyödyntää siihen QEMU – emulaattoria. QEMU on vapaan lähdekoodin ohjelmisto, joka pystyy toimimaan myös itsenäisenä hypervisorina. Sen ominaisuuksiin kuuluvat muun muassa tuki laajalle joukolle prosessoriarkkitehtuureja kuten x86, PowerPC ja ARM sekä mahdollisuus muokata virtuaalikoneita käytettäväksi suoraan myös muilla hypervisor – alustoilla. (Warnke & Ritzau 2010, 15-16.)

KVM yhteistyössä QEMU:n kanssa tarjoaa nopean alustan ja kattavan laitteistotuen virtuaalikoneiden asennukseen ja suorittamiseen. Näiden kahden hypervisor-paketin lisäksi tarvitaan useimmiten erillinen hallintaohjelmisto, jolla virtuaalikoneiden luontia ja ylläpitoa pystytään yksinkertaistamaan käyttäjäystävällisemmäksi. Yksi tällainen ratkaisu on libvirt, joka tarjoaa käyttöliittymärajapinnan KVM:n avulla luotujen virtuaalikoneiden käsittelyyn. Sen toimintoihin muun muassa komentorivikäyttöliittymä *virsh*, virtuaalikoneiden luontityökalu *virt-install*, sekä virtuaalikoneisiin pääsyn mahdollistava työpöytäsovellus *virt-viewer*. Libvirt tukee myös muita tyypin 2 hypervisoreita ja se sisältää työkalut virtuaalikoneiden etähallintaan useiden eri käyttöjärjestelmien kautta. (Libvirt 2013a; Libvirt 2013b; TuxRadar 2010.)

3 PILVIPALVELUT

3.1 Pilvipalveluista yleisesti

Palvelinten virtualisointi sekä tehostuneet tietoverkkoratkaisut ovat 2000 – luvun aikana mahdollistaneet toimintamallin, jossa oman konesalin ylläpitämisen sijaan ohjelmistot, alustat ja mahdollisesti koko tietoinfrastruktuuri siirretään käytettäväksi yksityisen verkon tai Internetin yli, eli pilveen. IT-resurssien mukauttaminen palveluun perustuvaksi toimintamalliksi lisäävät yleensä resurssien saatavuutta, joustavuutta, mitattavuutta sekä integroitavuutta toisten järjestelmien kanssa. Pilvipalveluiden luonteeseen kuuluu, että palveluiden loppukäyttäjän ei yleensä tarvitse välittää, missä hänen käyttämänsä ohjelmistot sijaitsevat, vaan ne ovat saatavissa paikasta ja päätelaitteesta riippumatta. (Hurwitz, Bloor, Kaufman & Halper 2010, 9-12; Salo 2010, 17.)

Yhdysvaltojen mittaustekniikoita ja standardeja kehittävän NIST (National Institute of Standards and Technology) – instituutin mukaan pilvimalliseksi kutsutaan verkon yli käytettävää, helposti käyttöönotettavaa järjestelmää, jonka tarjoamiseen käytetään dynaamisia resursseja. Lisäksi pilvipalvelu täyttää seuraavat kriteerit:

- On demand self-service eli käytettävissä tarpeen mukaan itsepalveluna.
- Käytettävissä päätelaitteesta riippumatta missä ja milloin vain.
- Ympäristö on multi-tenant –mallinen, jolloin palveluun käytettävä laitteisto on jaettu kaikkien asiakkaiden kesken eikä dedikoituja palvelimia käytetä.
- Nopea joustavuus: palvelua voidaan laajentaa ja supistaa kysynnän mukaan.
- Palvelua voidaan mitata ja optimoida automaattisesti. Käytön seuranta ja raportointi on mahdollista läpinäkyvyyden lisäämiseksi. (Mell & Grance 2011.)

Toinen tapa määritellä pilvipalveluja on OSSM (On-demand, Scalable, Self-service and Metered) – malli (Hardiman 2012). Pilvipalveluiden katsotaan voivan toteutua kolmena palvelumallina (Software as a Service, Platform as a Service, Infrastructure as a Service) ja neljänä käyttöönottomallina (private cloud, community cloud, public cloud, hybrid cloud) (Mell & Grance 2011). Pilvipalvelutyypit esitellään tarkemmin luvussa 3.2.

On demand – malli tuottaa perinteiseen konesalimalliin verrattuna huomattavia säästöjä, sillä usein palveluiden käyttötarve jakaantuu epätasaisesti, eikä omia palvelimia tarvitse niiden takia pitää käytettävissä jatkuvasti, kuten tähän asti. Pilvipalveluiden käyttöönotto merkitsee yritykselle usein myös siirtymistä uudenlaiseen toimintatapaan, jossa säästöjä haetaan virtualisoinnilla, ylläpidon ulkoistamisella, palvelinkapasiteetista luopumisella sekä kiinteiden kustannusten muuttamista muuttuviksi kustannuksiksi palveluostojen muodossa. (Heino 2010, 171-182.)

3.2 Käyttöönottotyypit

Pilvipalvelun käyttöönotto voidaan toteuttaa usealla eri periaatteella riippuen palvelun luonteesta sekä vaadittavasta saatavuus-, yksityisyys- ja tietoturvatasosta sekä rahoituksesta. Palveluiden saatavuus turvataan yleensä palvelutasoa koskevalla sopimuksella (Service Level Agreement, SLA), jossa on määritelty palvelun minimikäyttöaste sekä huolto- ja päivityskatkojen maksimikesto sekä ajankohdat. Sopimuksen toteutumista voidaan mitata palvelutasotavoitteilla (Service Level Object, SLO). (Heino 2010, 35-36.)

3.2.1 Public cloud

Julkisessa pilvipalvelumallissa palveluntarjoajan resurssit ovat käytettävissä Internetin yli päätelaitteesta riippumatta, jolloin palvelu on avoin kaikille tai saavutettavissa ainoastaan yrityksen lähiverkosta VPN - yhteyden yli. Public cloud – tyyppiset ratkaisut ovat helpon saatavuutensa ansiosta loppukäyttäjille vaivattomia, mutta eivät sovi arkaluontoista, kuten henkilö-, pankki-, potilas- tai yritystietoa käsittelevien järjestelmien sijoituspaikaksi. Nämä palvelut halutaan yleensä tietoturvan ja saatavuuden varmistamiseksi suorittaa hallitusti yrityksen sisällä. Julkinen pilvipalvelumalli sopiikin hyvin yritysten tukiprosesseille ja muille ei-bisneskriittisille osa-alueille. Palveluntarjoaja vastaa palveluun käytettävän laitteiston ja yhteyksien toiminnasta sekä kustannuksista, ja laskutus tapahtuu kuukausi- tai tuntiperusteisesti tai kapasiteettiin sidottuna. Asiakas ei siis omista palvelua, eikä voi kirjata sitä pääomaansa. (Furth & Escalante 2010, 67; Heino 2010, 54-55.)

3.2.2 Private cloud

Private cloud on pilvipalvelumalli yrityksen sisäiseen käyttöön. Tässä mallissa palvelun toteuttajana ja omistajana voivat toimia tulkinnasta riippuen vain organisaatio itse (Lakshmi & Vadhiyar 2011; Heino 2010, 55; Salo 2010, 32) tai vaihtoehtoisesti ulkoinen toimija tai organisaatio yhteistyössä ulkoisen toimijan kanssa (Mell & Grance 2011). Palvelu rajataan yrityksen sisäverkkoon palomuurin taakse, mutta voidaan halutessa aukaista etäkäyttäjille VPN – yhteydellä. Private cloudia käyttävät useimmiten suuret yritykset, jotka haluavat valjastaa tietokeskustensa laskentatehon virtualisoinnin ja palvelinten yhdistämisen avulla pilvipalvelukäyttöön sekä välttää sitä kautta tehokkaasti julkisten pilvipalveluiden sisältämät luotettavuus- ja tietoturvariskit. (Furth & Escalante 2010, 67-68.)

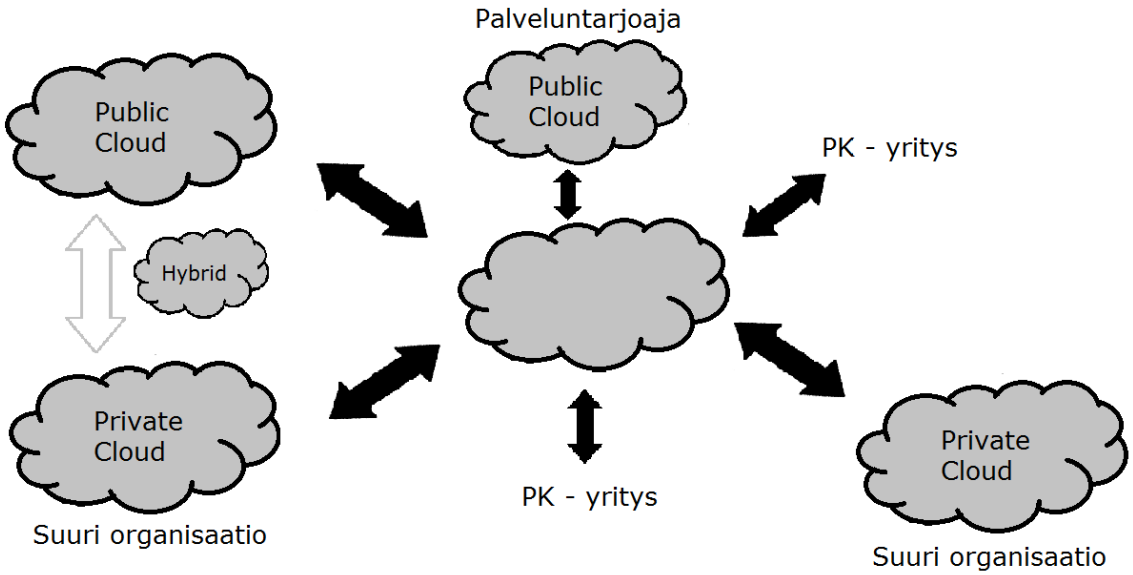
3.2.3 Community cloud

Community cloud – malli tarjoaa mahdollisuuden jakaa yksityisen pilvipalvelun resurssit usealle eri käyttäjätahoille. Yleensä jaettua pilvipalvelumallia käyttävillä organisaatioilla on yksi tai useampia yhdistävä tekijä, kuten toimiala tai korkeat tietoturvavaatimukset, jotka luovat intressin yksityisen tietojärjestelmien yhteiskäyttöön julkisten palveluratkaisujen sijaan. Näin yhteisöjen tarpeet saadaan täytettyä huomattavasti edullisemmin kuin erillisten palvelujen hankkimisella. Palvelun rakentamisessa voidaan hyödyntää käyttäjätahojen omien, usein vajaakäytölle jääneiden tietojärjestelmien ja -verkkojen yhdistämistä. (Raj 2013; Mell & Grance 2011.)

Yleiseksi ongelmaksi yhteisten pilvipalvelujen käytössä nousevat usein omistussuhteet ja hallinta; selkeää vastuuta ylläpidosta tai palveluja koskevasta päätöksenteosta ei välttämättä osata asettaa käyttäjäosapuolten välillä. Jaettua järjestelmiä pidetään myös tietoturvatasoltaan heikompana ratkaisuna kuin omaa, yksityistä pilvipalvelua. (Raj 2013, 234.)

3.2.4 Hybrid cloud

Kahden pilvipalvelun käyttöönottotyypin yhdistämisestä, yleensä private tai community cloud – tyyppisen palvelun yhdistäminen julkiseen palveluun, käytetään nimitystä hybric cloud. Hybridimallissa julkista pilvipalvelua hyödynnetään sisäisen järjestelmän jatkeena yhdistämällä se Internetin välityksellä standardoituja tai itse kehitettyjä rajapintoja käyttäen. Tällä tavalla voidaan tehokkaasti rajata tietyt tehtävät suoritettavaksi julkisessa ja osa yksityisessä pilvessä, esimerkiksi käyttäjätietokannat ja muut kriittiset palvelut halutaan säilyttää organisaation sisällä, mutta testausvaiheen ohjelmisto voidaan saattaa julkiseen pilveen omien resurssien säästämiseksi. Hybridipilven avulla varaudutaan myös äkilliseen kapasiteetin kysynnän nousuun (cloud bursting), sillä julkisen pilvipalvelun laajentaminen onnistuu yleensä nopeasti. Hybrid cloud on selvästi nouseva trendi Enterprise – tason yrityksissä. Toisaalta sitä voidaan myös pitää eräänlaisena siirtymävaiheena yksityisestä pilvimallista julkiseen. (Raj 2013, 235; Subramanian 2011; Furth & Escalante 2010, 68-69.)



KUVA 4. Pilvipalvelujen käyttöönottotyypit. Mukaillen (Raj 2013, 235).

3.3 Palvelumallit

3.3.1 Software as a Service

Software as a Service – mallissa asiakas hankkii yksittäisen ohjelmiston palveluna, jolloin palvelun käyttö tapahtuu Internetin yli, yleensä selainpohjaisesti. Tässä mallissa asiakas ei omista palvelualustan laitteistoa eikä ohjelmistoja (Mell & Grance 2011). SaaS on suunniteltu korvaamaan perinteiset ohjelmistot laitteisto-, lisenssi- ja hallintakuluineen tehokkaammalla palveluntarjoajan ratkaisulla, jossa laskutus tapahtuu aika- ja/tai käyttäjäperustaisesti, vapauttaen samalla palvelun kehitykseen ja ylläpitoon vaadittavia henkilöstöresursseja. (Furth & Escalante 2010, 346; Salo 2010, 29.).

Ohjelmistojen ulkoistaminen palvelumuotoisiksi tuo useita etuja asiakasyrityksille. Koska palveluntarjoajan tuottama sovellus on usein samankaltainen kaikkien asiakkaiden kesken, kehitystyö on nopeaa ja asiakkailta opittuja toimintatapoja voidaan helposti hyödyntää muiden asiakkaiden tuotteissa. SaaS – ohjelmistojen tulee luonteensakin puolesta olla helppokäyttöisiä ja toimintavarmoja, ja ne on yleensä helppo integroida muihin järjestelmiin. Markkinoilta löytyy runsaasti valmiita SaaS – ratkaisuja organisaatioiden eri toimintoihin, kuten toiminnanohjaukseen, asiakas- ja sisällönhallintaan, toimistotyökaluiksi, tai sisällön tuottamiseen. (Salo 2010, 29-31.)

3.3.2 Platform as a Service

Platform as a Service – mallinen palvelu toimii alustana sovellusten kehittämiseen. PaaS – ympäristö sisältää laitealustan lisäksi elementit ohjelmistojen tuottamiseen ja testaukseen tarjoamalla ohjelmistonipun (software stack), johon kuuluvat ohjelmointiympäristö, sovelluspalvelin sekä tietoturva- ja varmuuskopiointityökalut (Furth & Escalante 2010, 5-6). Kuten SaaS – palveluissa, asiakas hallinnoi vain palvelun sisältöä välittämättä alle asennetusta infrastruktuurista. Tällöin alustalla toteutetut ohjelmat käyttävät normaalisti myös muita palveluntarjoajan tuottamia palveluita, esimerkiksi tietokantaa, ja ovat palvelun tarjoamien kirjastojen ja rajapintojen kautta laajennettavissa huomattavasti paikallista alustaa monipuolisemmin (Salo 2010, 28). PaaS – alustoista ainakin Windows Azure ja Google App Engine sisältävät myös mahdollisuuden hyödyntää palvelua myös paikallisessa ohjelmointiympäristössä erillisen liitännäisen kautta (Google 2013; Windows Azure 2013).

Palvelumallisten kehitysympäristöjen käyttöönotto tarkoittaa, ettei yritysten tarvitse hukata resursseja oman ohjelmointi- ja testausympäristön rakentamiseen. Pilviympäristössä ei myöskään tarvitse kantaa huolta kapasiteetin ylittymisestä, vaan ne skaalantuvat itsestään tarpeen mukaan, joka hyödyttää varsinkin aloittelevia yrityksiä. Haittapuolina PaaS - ratkaisuissa voidaan pitää rajoittunutta tukea ohjelmointikielille sekä tietoturvariskejä, jotka syntyvät yrityksen ohjelmistojen lähdekoodin sijoittamisesta kolmannen osapuolen hallitsemalle alustalle. (Salo 2010, 28-29.)

3.3.3 Infrastructure as a Service

Infrastructure as a Service tarkoittaa yksinkertaisimmillaan virtuaalisen laitteistopohjan hankkimista palvelumuotoisena. IaaS – järjestelmä hankitaan yleensä ulkoiselta toimijalta, ja se käsittää lisäksi palvelujen suorittamiseen vaadittavat komponentit: virtuaalikoneen/konesalin, käyttöjärjestelmän, varastotilan, sekä verkon, joiden varaan asiakasyritys voi asentaa omia ohjelmistojaan ja palvelujaan (Diversity Limited 2011). Virtualisoinnin ansiosta IaaS – mallisen palvelun hallinnointi ja mukauttaminen onnistuvat asiakkaalta todella joustavasti. Se voi yleensä vapaasti lisätä, poistaa ja kopioida virtuaalikoneita sekä muuttaa niille määriteltyjä resursseja kuten prosessoritehoa, muistia ja kiintolevytilaa haluamansa mukaan. Laskutus tapahtuu käytettyjen resurssien perusteella, eikä yhteydenpitoa palveluntarjoajan kanssa yleensä tarvita missään vaiheessa. (Salo 2010, 25-26.)

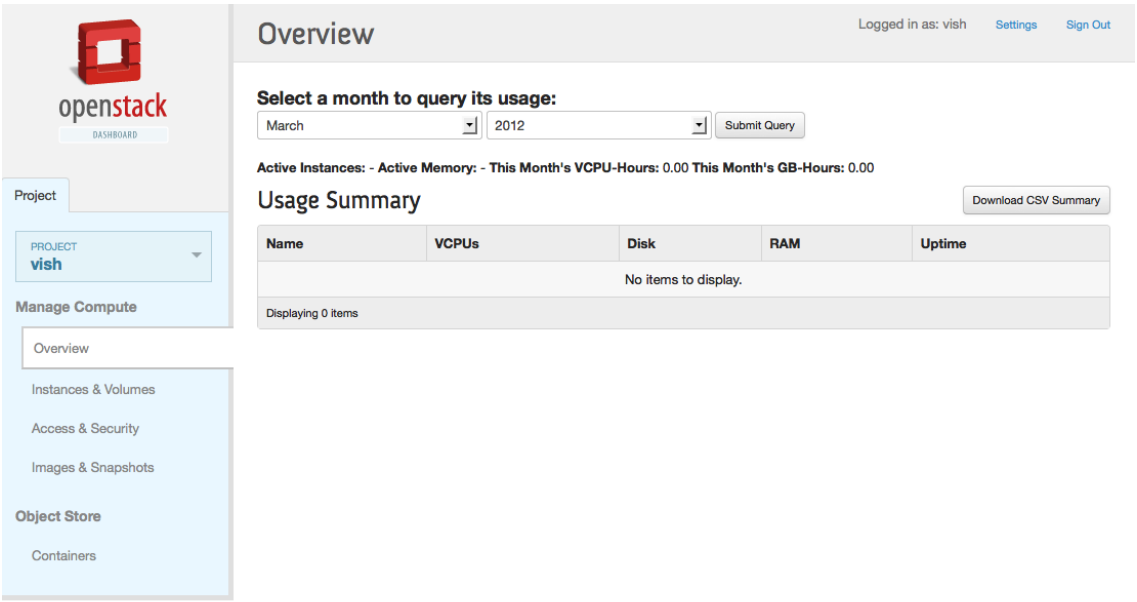
3.4 IaaS – palveluja

Infastructure as a Service – mallisiin pilvipalveluihin on nykyisin tarjolla useita valmiita maksullisia sekä vapaan lähdekoodin ratkaisuja. Osa näistä palveluista käyttää omia hypervisoreitaan, kun taas toisia pystytään yhdistelemään yhteen tai useampaan hypervisoriin. Tunnetuimpiin kaupallisiin palvelualustoihin kuuluvat Amazon Web Services, HP Cloud, Windows Azure ja IBM SmartCloud Enterprise (Heino 2010, 105-128). Tässä luvussa tutustutaan tarkemmin vapaan lähdekoodin IaaS – palveluihin ja niiden sisältöihin.

3.4.1 OpenStack

OpenStack lähti liikkeelle NASA:n tietojärjestelmien yhdistämisprojekti Nebulan ongelmista riittävän skaalantuvan ja monipuolisen palveluinfrastruktuurin löytämiseksi. Yhdessä palveluntarjoaja Rackspacen kanssa aloitettu projekti tähtäsi Nebulan entisen projektipäällikön Ray O'Brienin (2012) mukaan yhteisöön, joka rakentaisi näiden kahden organisaation erillisistä ratkaisuista, Nebulasta ja RackSpacen Cloud Files - tietokantaprojektista, yhden vapaan lähdekoodin pilvipalvelualustan. Sittemmin OpenStackista on kasvanut yli 100 jäsenyrityksen konsortio, joista merkittävimpinä osallistujina tietotekniikka- ja verkkoyhtiöt AT&T, HP, IBM ja Nebula sekä suurimmat Linux-jakelupakettien valmistajat Canonical, Red Hat ja Suse. (OpenStack 2013a; Jha, D, Murari, Raju, Gherian & Girikumar 2012, 7.)

OpenStack kutsuu itseään pilvikäyttöjärjestelmäksi laajojen tietoresurssien hallinnointiin (OpenStack 2013b). Käyttörajapintana toimii web-pohjainen hallintapaneeli Horizon, jonka avulla pystytään suorittamaan palvelun kaikkia toimintoja, kuten virtuaalikoneita, käyttäjiä, verkkolevyjä ja avainpareja (kuva 5). Horizonin alla toimii neljä palvelukokonaisuutta, joista osa jakaantuu useampaan komponenttiin. Taulukko 1 esittelee palvelun osat ja niiden käyttötarkoitukset. (Jha, D, Murari, Raju, Gherian & Girikumar 2012, 7, 13.)



KUVA 5. OpenStackin Horizon-hallintapaneeli. (OpenStack 2013c)

TAULUKKO 1. OpenStackin rakenne

Palvelun osa	Komponentit	Käyttötarkoitus
Nova	API Server Message Queue Compute Workers Network Controller Volume Worker Scheduler	OpenStackin "aivot". Virtuaalikoneiden, verkon, kulunvalvonnan ja skaalautuvuuden hallinta. Toimii rajapintana hypervisorille Libvirtin välityksellä.
Glance	Glance-control Glance-registry	Virtuaalikoneiden levykuvien luonti, hallinta ja varastointi.
Swift	Swift Account Swift Container Swift Object Swift Proxy The Ring	OpenStackin tallennusohjelmisto (object storage).
Keystone	Token Service Catalog Service Policy Service	Identiteetin hallinta. Suorittaa käyttäjien autentikoinnin ja käyttölupakyselyt muille palvelun osille.
Horizon	-	Web-hallintapaneeli.

OpenStackin arkkitehtuurissa Nova hoitaa komponenttiensa kanssa palvelun ydinprosessit käyttäen hyödykseen muita palvelun osia varastointiin, levykuvien hallintaan ja identiteetin tarkistukseen. Nova suorittaa myös tehtävien ajastukseen ja jonotukseen liittyvät toimenpiteet. Uusiman OpenStack-version, Grizzlyn, tuettujen hypervisoreiden listaan kuuluvat KVM, LXC, QEMU, UML, VMware vSphere, Xen, PowerVM, ja Hyper-V. (OpenStack 2013d; Jha, D, Murari, Raju, Gherian & Girikumar 2012, 7-8.)

3.4.2 Eucalyptus

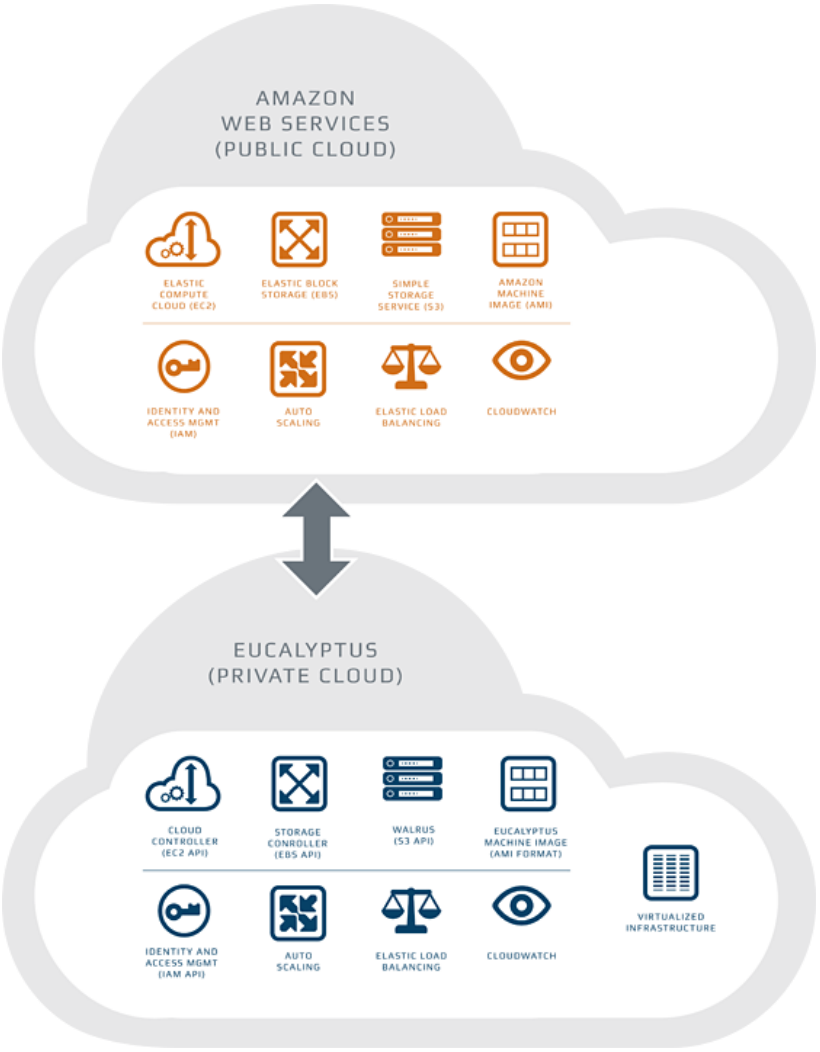
Eucalyptus (lyhenne sanoista Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems) on IaaS – ohjelmisto, joka erikoistuu Amazon Web Services – integroitujen pilvipalvelujen luomiseen (Eucalyptus 2013a). Eucalyptus kohosi merkittävään asemaan vapaan lähdekoodin pilvialustojen joukossa, kun se vuonna 2009 sisällytettiin Ubuntun 9.10 – julkaisun pilvipalveluversioon, Ubuntu Enterprise Cloudiin (Eucalyptus 2013b). Yhteistyö Canonicalin kanssa päättyi kuitenkin jo neljä julkaisua myöhemmin, kun Eucalyptus korvattiin versiossa 11.10 OpenStackilla, ja nimi muutettiin samalla Ubuntu Cloud Infrastructureksi (Morgan 2011). Sittemmin myös virallinen tuki Xen – hypervisorille loppui. Nykyisin Eucalyptus tukee virallisesti vain Red Hat Enterprise Linux - ja CentOS- isäntäkäyttöjärjestelmiä KVM- sekä VMware ESXi- ja vCenter- hypervisoreilla kehityssuunnan käännyttyä pääosin AWS – pohjaisiin hybrid cloud – ratkaisuihin. (Eucalyptus 2013c).

Eucalyptuksen uusimmassa 3.3.0.1 – versiossa lähes kaikki komponentit on hiottu yhteensopiviksi AWS:n eri osien kanssa, jolloin pilven eri toimintoja, kuten käyttäjätietoja, verkkoa ja hypervisoreja, voidaan hallinnoida AWS:n kautta (kuva 6). Lisäksi Eucalyptus tukee AWS:n rajapintoja, mikä mahdollistaa ulkoisille järjestelmille mahdollisuuden kommunikoida molemmilla tavoilla toteutetuille pilvipalveluille samoilla työkaluilla. (Eucalyptus 2013a; Eucalyptus 2013d.)

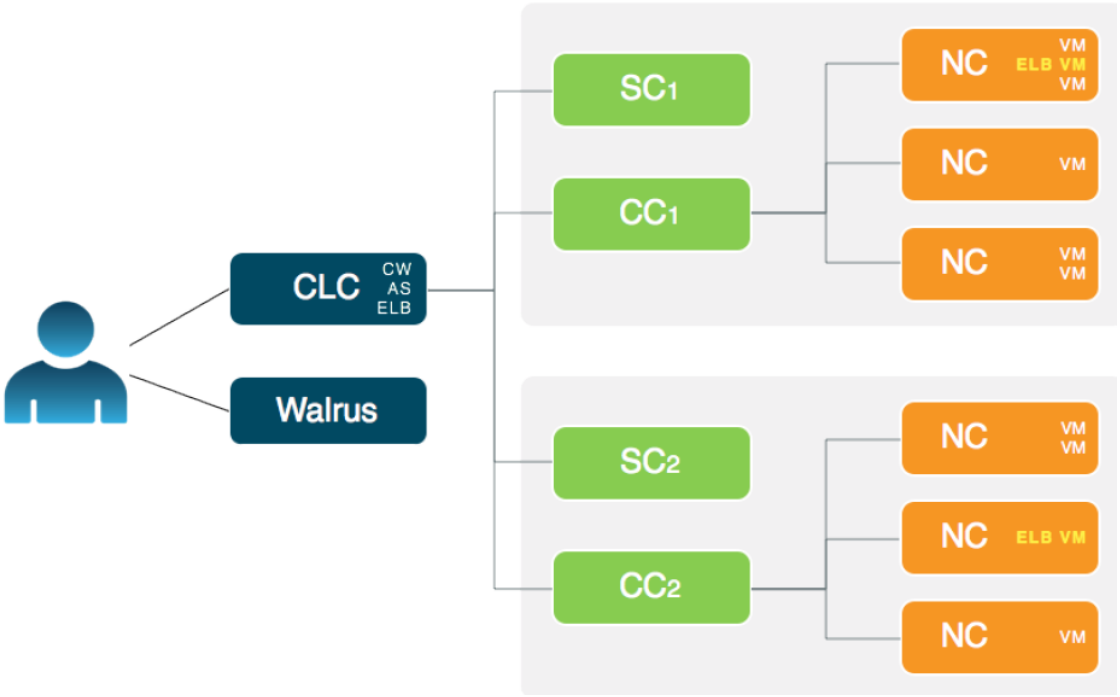
Eucalyptus käyttää hierarkkista mallia, jossa palvelun osat niputetaan kokonaisuuksiksi, joita hallinnoidaan ylemmän tason komponenteilla (kuva 7). Eucalyptus sisältää seuraavat komponentit:

- Cloud Controller (CLC): Pilven hallintapaneeli, jolla ohjataan koko palvelun toimintaa antamalla käskyjä alemmille palvelun osille. Cloud Controlleria käytetään web – pohjaisella Eucalyptus Administrator Consolella tai Amazon EC2 – yhteensopivilla komentorivikäskyillä.
- Cluster Controller (CC): Yhteydessä sekä Cloud Controlleriin että paikalliseen Node Controller – nippuun eli klusteriin. Välittää käskyt Cloud Controllerilta NC – klusterille ja hallinnoi klusterin verkkoa.

- Node Controller (NC): Asennetaan virtuaalikoneita suorittavalle alustalle. Hallinnoi paikallisia virtuaalikoneita, levynkuvia ja verkkoa hypervisorin kautta. Suorittaa Cluster Controllerilta tulevat käskyt.
- Walrus: Tallennusohjelmisto (object storage) levynkuvien ja käyttäjätietojen säilytykseen. Yhdistettävissä Amazonin Simple Storage Serviceen.
- Storage Controller (SC): Käytetään verkkolevyjen luomiseen virtuaalikoneille. Verkkolevyjä voidaan tallentaa snapshot – ominaisuuden avulla Walrusiin.
- VMware Broker: Lisäosa, jolla parannetaan Eucalyptuksen yhteensopivuutta VMwaren hypervisoreiden kanssa ja mahdollistetaan palvelun ominaisuudet myös VMwaren alle asennetuille virtuaalikoneille. Vaatii palvelun maksullisen version. (Eucalyptus 2013e.)



KUVA 6. Eucalyptuksen AWS – yhteensopivuus (Eucalyptus 2013d)



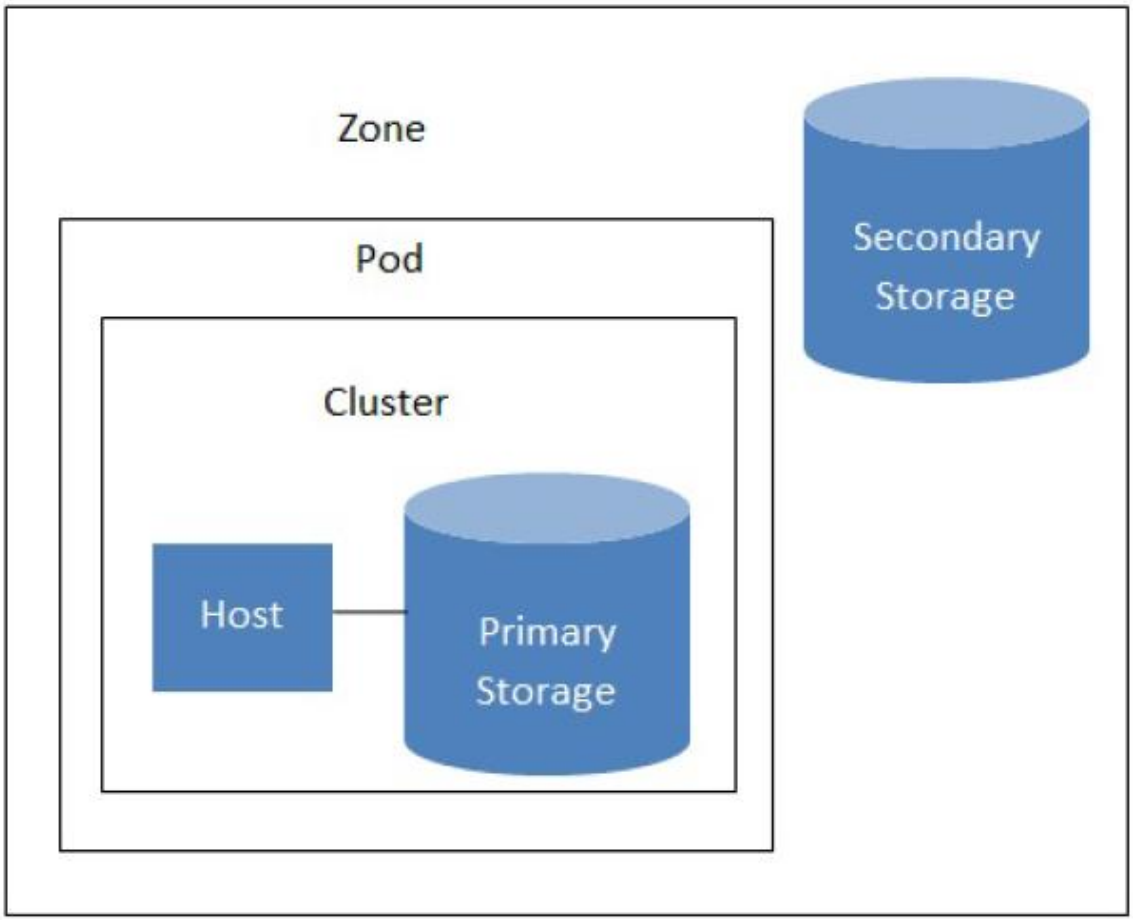
KUVA 7. Eucalyptuksen komponenttien hierarkia high availability -mallissa (Eucalyptus 2013d)

3.4.3 Apache CloudStack

Apache CloudStack on Java – pohjainen IaaS – järjestelmä yksityisten, julkisten ja hybridimallisten pilvipalvelujen käyttöönottoon. Cloud.com – yhtiö julkaisi vuonna 2010 suurimman osan ohjelmiston lähdekoodista GNU General Public License v3 – lisenssillä. Sen jälkeen yhtiö myytiin Citrixille, joka puolestaan luovutti 2012 CloudStackin Apache Software Foundationille. Projekti lisensoitiin uudelleen Apache Software License 2.0 – lisenssillä ja nostettiin maaliskuussa 2013 Apachen tärkeimpien projektien joukkoon samalla kun kehitystyötä ohjattiin OpenStackin tapaan yhteisöllisempään suuntaan. (Brockmeier 2013; CloudStack 2013a.)

CloudStackin ominaisuuksiin kuuluvat laaja tuki hypervisoreille, alustan vikasietoisuus ja keskitetty hallinta, pilven ylläpito-ohjelmistot ja virtuaalikoneiden automaattinen konfigurointi, web-hallintasivusto, monipuoliset hallintarajapinnat sekä korkean käytettävyyden työkalut, muun muassa MySQL – replikointi ja verkkoyhteyksien niputus. CloudStackin arkkitehtuurin yksikkönä toimii alue (Zone), joka jakaantuu kapseleihin (Pod) ja klustereihin (Cluster) (kuva 8). Kapselit toimivat Eucalyptuksen Cluster Controllerin tapaan virtuaaliverkon jakajina käyttäen 2-tason kytkintä reititykseen. Tallen-

nusohjelmistot on jaettu primaarisiin (virtuaalilevyt) ja sekundaarisiin (templaatit, levynkuvat ja snapshotit). (CloudStack 2013b.)



KUVA 8. CloudStackin infrastruktuurin rakenne (CloudStack 2013b)

3.4.4 Nimbus

Nimbus – projekti on osoitettu erityisesti tieteellisille yhteisöille ja yliopistoille. Nimbuksen pääkomponentit ovat IaaS – alusta Nimbus Infrastructure ja Nimbus Platform, joka sisältää perinteisen IaaS – ympäristön lisäksi PaaS – elementtejä. Sen cloudinit.d– ja Context Broker – työkaluilla pyritään automatisoimaan virtuaalikoneiden luontia ja mahdollistamaan kehittyneiden pilviohjelmistojen käyttö virtuaalikoneissa ilman ylimääräistä ylläpitotyötä. Cloudinit.d mallintaa Unix – järjestelmien init.d – työkalua pilviohjelmistojen konfigurointiin. (Nimbus 2013a; Nimbus 2013b.)

Nimbuksen ylläpitäjät toimivat aktiivisesti pilvipalvelujen kehittämiseksi. Yhteisö julkaisee pilvipalveluihin liittyviä tutkimuksia ja ylläpitää myös sciencecloud.org - sivustoa, joka kerää yhteen yliopistojen omia pilvipalveluratkaisuja ja keskustelua virtuaalikoneisiin, ohjelmistoihin ja muuhun pilviympäristöjen kehittämiseen liittyen. (Nimbus 2013c; Science Clouds 2013.)

3.4.5 OpenNebula

OpenNebula on yksi vanhimmista vapaan lähdekoodin IaaS – alustoista. Se tähtää puhtaasti yksityiseen pilvimalliin, jossa loppukäyttäjillä on pääsy suoraan pilven kaikkiin toimintoihin. Lisäksi OpenNebulan erikoisuuksiin kuluu hajautettu tiedostojärjestelmä (Network File System, NFS) kaikkien palvelun käyttämien tiedostojen tallennukseen. Nämä ominaisuudet tekevät OpenNebulasta haastavan ylläpidettävän. Myös OpenNebulaan kuuluu mahdollisuus osittaiseen Amazon EC2 – integrointiin. (Popović, Jovanović, Jovanović & Popović 2011.)

4 YMPÄRISTÖN SUUNNITTELU

4.1 Toimeksiantajan esittely

Projektin toimeksiantaja, Toyme Lab Oy, on vuonna 2000 perustettu tamperelainen ohjelmistoalan yritys, jonka palveluksessa työskentelee kirjoitushetkellä 4 henkilöä. Yrityksen tärkein tuote on Java – pohjainen Toyme - palvelu, jota voidaan yrityksissä käyttää aloitteiden ja kehitysehdotusten, työturvallisuushavaintojen, reklamaatioiden, sekä tuotantopoikkeamien ja havaintojen kirjaamiseen ja raportointiin.

Toyme on pilvipalvelu, joka tarjotaan asiakkaalle SaaS - periaatteella. Alusta palvelun ylläpitoon on hankittu erillisiltä palveluntarjoajilta.

4.2 Lähtökohdat, tavoitteet ja tarkoitus

Tarve nykyaikaiselle, tehokkaan testaus- ja demotoiminnan mahdollistavalle alustalle oli toimeksiantajalla esiintynyt jo kauan. Tähän mennessä asiakasprojektien käyttöönotossa testausvaihe on usein jäänyt puutteelliseksi. Tällöin yksinkertaisetkin päivitys-muokkaus- ja korjaustoimenpiteet, jotka olisi helposti havaittu testausvaiheessa, on pitänyt toteuttaa tuotantokäytössä olevaan ohjelmistoon. Tämä aiheutti ylimääräisiä katkoksia ja heikensi käytettävyyttä.

Uuden palvelimen tehtävänä on määrittelyn mukaan:

- Tarjota alusta demoprojektien esittelyyn
- Mahdollistaa asiakkaalle oman järjestelmänsä testaus joustavasti
- Toimia ”hiekkalaatikkona” uusien tekniikoiden käyttöönotossa
- Tarjota mahdollisuus testata sovelluksen laajennusmahdollisuuksia kuten integrointi ulkoisiin järjestelmiin, käyttäjämoduuleiden eräajon automatisointi, tietoturva
- Toimia mahdollisena varmuuskopioiden tallennuspaikkana
- Laajentaa ymmärrystä virtualisoidun palveluinfrastruktuurin toiminnasta

Palvelimen lähtökohdaksi otettiin oman virtuaalisen konesalin rakentaminen, jotta eri palvelut saadaan jaoteltua virtuaalikoneittain. Tavoitteena oli vapaan lähdekoodin oh-

jelmistoja hyödyntäen rakentaa joustava, helposti ylläpidettävä ja laajennettava infrastruktuuri, joka tarjoaa toimeksiantajalle mahdollisuuden uusien käyttöjärjestelmien, sovellusalustojen, ja ohjelmointitekniikoiden testaukselle ja käyttöönotolle sekä asiakkaalle turvallisen testausympäristön.

Konesalin virtualisointi päätettiin toteuttaa KVM – hypervisorilla. Valinta oli alustavan vertailun perusteella selkeä, sillä KVM:n kehitys on edennyt pitkälle ja se täyttää hypervisor – ohjelmistolle annetut kriteerit:

- Ilmainen
- Helppokäyttöinen
- Yksinkertainen asentaa, sisältyy Linuxin kerneliin
- Monipuolinen tuki virtuaalikoneiden käyttöjärjestelmille
- Laaja dokumentointi

5 TOTEUTUS

5.1 Laitteisto ja käyttöjärjestelmä

Palvelimen rakennuksessa käytettiin seuraavia laiteresursseja:

- Emolevy: ASRock Z68 Extreme3 Gen3
- Prosessori: Intel Core i5 2320
- Muistit: 4 GB 1333 MHz DDR3 x3, yht. 12GB
- Kiintolevyt: Seagate Barracuda SATA-600 500 GB x2

Prosessorin valinnassa on otettava huomioon tuki virtualisointialusta KVM:n vaatimalle virtualisointitekniikalle. Intelin tapauksessa lista Intel VT - tuetuista malleista löytyy suoraan valmistajan sivuilta (Intel 2013). Muistia käyttävät isäntäkäyttöjärjestelmän lisäksi virtuaalikoneet, joten sitä on asennettava riittävästi. Palvelinympäristössä kiintolevyjen varmistus tapahtuu asentamalla ne RAID – pakkaan, jonka tasoksi valittiin RAID 1. Tässä mallissa kiintolevyt peilataan eli niille kirjoitetaan samat tiedot, jolloin varmistetaan käyttöjärjestelmän toiminnan jatkuvuus myös laitteistovian yhteydessä.

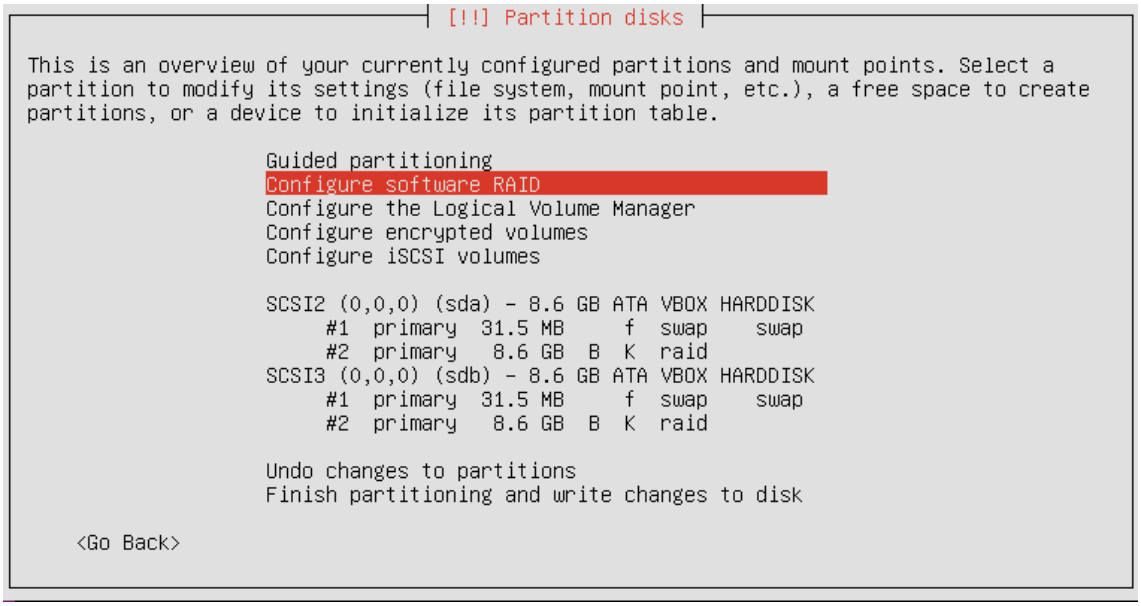
Käyttöjärjestelmäksi palvelimeen valittiin Ubuntu 12.04 Server. Perustelut valinnalle olivat helppokäyttöisyys sekä kattava dokumentointi KVM:n asennukselle ja ylläpidolle. Ubuntu on työpöytäkäytössä myös osoittautunut luotettavaksi ja helpoksi päivittää.

5.2 Ympäristön asennus

Asennuksen aluksi oli tarkistettava emolevyn virtualisointitekniikan tila. Käytetyssä ASRockin emolevyn UEFI – käyttöliittymässä asetus löytyy kohdasta Advanced → CPU Configuration → Intel Virtualization Technology. Asetuksen tilana on oltava ”Enabled”. Samalla UEFI:n kautta määritetään ensisijainen käynnistysmedia, tässä tapauksessa USB – portti, kohdasta Boot → Boot option #1. UEFI – asetuksista poistuessa tietokone uudelleenkäynnistetään, jonka jälkeen näkyviin avautuu Ubuntu Serverin asennusvalikko. Käyttöjärjestelmän asennus aloitetaan valitsemalla ”Install Ubuntu Server”.

Ennen käyttöjärjestelmän varsinaista asennusta RAID 1 – pakan luontia testattiin emolevyn Intel Raid Storage – työkalulla hardware RAID – mallisena. Tästä mallista kuitenkin luovuttiin, sillä hardware RAID sisälsi käyttöjärjestelmän kautta asennettavaan software RAID – tekniikkaan verrattuna useita heikkouksia, esimerkiksi epävarma toimivuus, SWAP – muistin määrän valinnan puuttuminen, heikko pakan tilan seuranta ja hallittavuus sekä pakollinen uudelleenkäynnistys pakan muutoksia, kuten levyjen lisäämistä, varten.

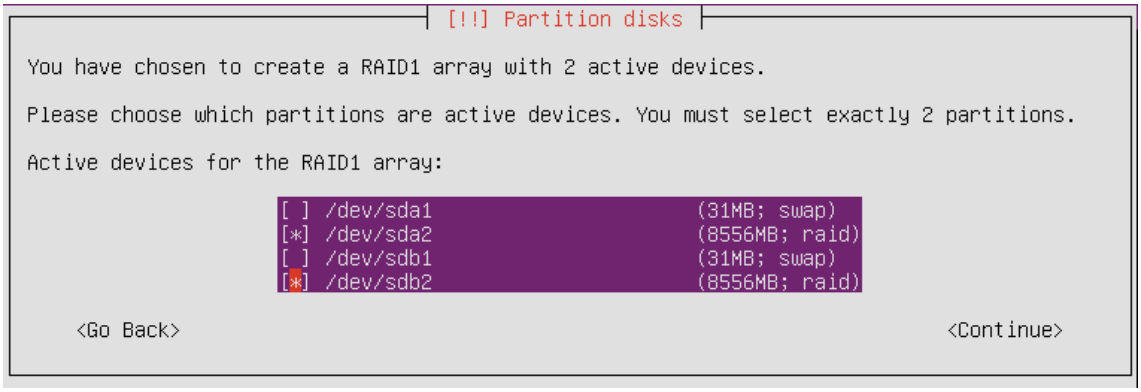
Käyttöjärjestelmän asennusvalikossa seuraa kieli- näppäimistö- aika- ja käyttäjäasetusten jälkeen kiintolevyjen osiointi. Vaihtoehto ”Manual” mahdollistaa kiintolevyjen osioinnin halutulla tavalla. Molemmille kiintolevyille luodaan kaksi osiota: pieni SWAP – osio sekä lopun kiintolevystä käyttävä RAID – osio (kuva 9). Tämä tapahtuu valitsemalla kiintolevyn tyhjä osio, jonka jälkeen aukeavasta valikosta ”Create a new partition”. Seuraavaksi määritellään osion koko, tyyppi (”Primary”) ja sijainti (”Beginning”). Osion hallinnassa levyjärjestelmäksi valitaan SWAP, tai RAID – osion kohdalla ”Physical volume for RAID”. RAID – osio määritellään samalla käynnistysosioksi kohdassa ”Bootable Flag”.



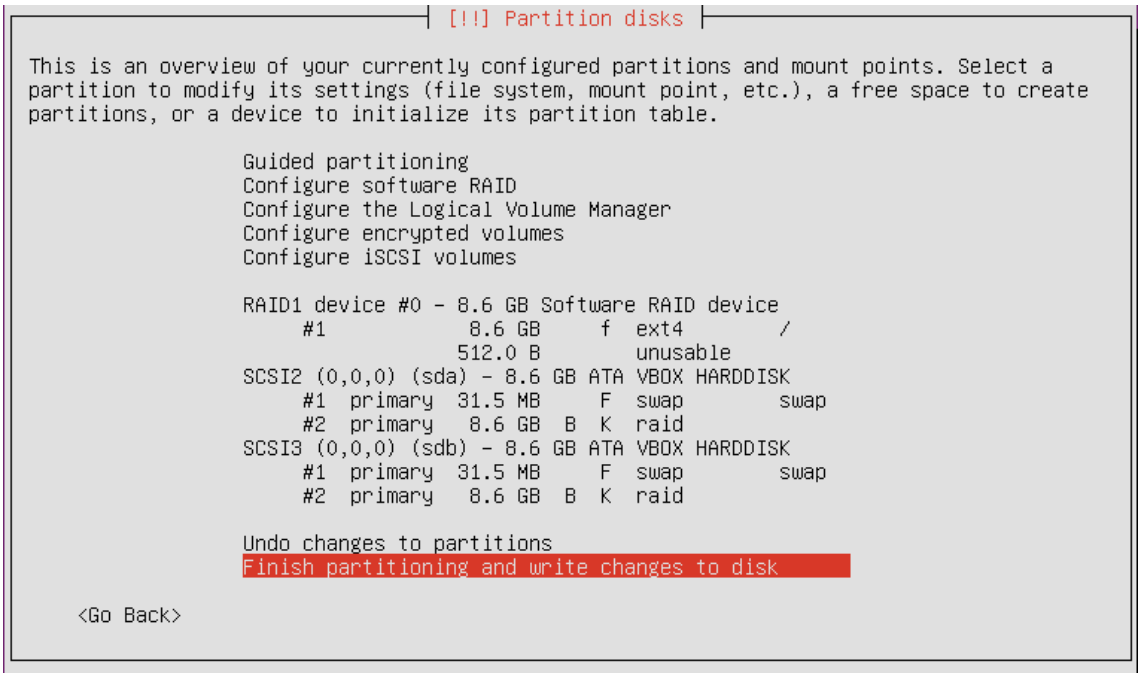
KUVA 9. Kiintolevyosiointi ennen RAID:n asennusta

Software RAID:n asennus käynnistetään valitsemalla ”Configure software RAID” → ”Create MD device”. Asennuksessa RAID – pakalle määritellään tyyppi (”RAID 1”), osallistuvien laitteiden määrä (2) sekä valittavat osiot (edellisessä kappaleessa luodut RAID – osiot) (kuva 10). Asennuksesta palataan takaisin osiovalikkoon valitsemalla

”Finish”. Osiovalikossa luotu RAID – pakka näkyy uutena loogisena levynä, jonka osioita voidaan muokata normaalien kovalevyjen tapaan. RAID – levyn osio asetetaan käynnistysosioksi EXT4 – levyjärjestelmällä samalla, kun pakalle määritetään asennuskansio. Tämän jälkeen levyn osiointi voidaan viedä loppuun valitsemalla ”Finish partitioning and write changes to disk” (kuva 11). Asennus kysyy vielä toimenpiteitä pakan osaan ilmenevän ongelman varalta ennen levyn kirjoittamisen aloittamista. Asennuksen aikana voi näppäinyhdistelmällä ctrl + alt + F2 siirtyä konsoliin, jossa pakan tilaa pystyy seuraamaan *cat /proc/mdstat* ja *mdam --detail /dev/md0* – komennoilla (kuva 12). Levyjen synkronointi *cat /proc/mdstat* – tulosteessa tulee seurata loppuun ennen grup – käynnistysvalikon asennusta. Tämän jälkeen asennuksen voi viedä loppuun normaalisti.



KUVA 10. RAID – osioiden valinta



KUVA 11. Valmis RAID - osiointi

```
~ # mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Tue Aug 20 14:15:03 2013
     Raid Level : raid1
     Array Size : 8351680 (7.96 GiB 8.55 GB)
  Used Dev Size : 8351680 (7.96 GiB 8.55 GB)
   Raid Devices : 2
  Total Devices : 2
    Persistence : Superblock is persistent

    Update Time : Tue Aug 20 14:52:32 2013
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

           Name : ubuntu1204:0  (local to host ubuntu1204)
           UUID : 65225f7d:179cc6dc:cd29acfd:e6717b00
         Events : 19

    Number   Major   Minor   RaidDevice State
       0       8        2        0      active sync   /dev/sda2
       1       8       18        1      active sync   /dev/sdb2
```

KUVA 12. Tuloste RAID – pakasta mdam – komennolla

5.3 Hypervisorin asennus

Ennen KVM:n asennusta käyttöjärjestelmässä tulee vielä varmistaa virtualisointiteknii-kan tuki. Tämä tapahtuu tarkastelemalla /proc/cpuinfo – tiedostoa, jonka flags - rivillä ovat lueteltuna kaikki prosessorin tukemat ominaisuudet kirjainyhdistelminä. Tuki lait-teistopohjaiselle virtualisoinnille ilmoitetaan lyhenteellä wmx (Intel) tai svm (AMD). Koska kyseessä on Intelin prosessori, on helpoin tapa tutkia *grep wmx /proc/cpuinfo* – komennon tulostetta (kuva 13)

```
demo@demo:~$ grep vmx /proc/cpuinfo
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi
 mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtop
ology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse
4_1 sse4_2 popcnt tsc_deadline_timer aes xsave avx lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow
 vnmi flexpriority ept vpid
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi
 mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtop
ology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse
4_1 sse4_2 popcnt tsc_deadline_timer aes xsave avx lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow
 vnmi flexpriority ept vpid
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi
 mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtop
ology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse
4_1 sse4_2 popcnt tsc_deadline_timer aes xsave avx lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow
 vnmi flexpriority ept vpid
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi
 mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtop
ology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse
4_1 sse4_2 popcnt tsc_deadline_timer aes xsave avx lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow
 vnmi flexpriority ept vpid
demo@demo:~$
```

KUVA 13. Virtualisointituen todennus *grep wmx /proc/cpuinfo* – tulosteesta

KVM:n ja sitä tukevien libvirt – pakettien asennus aloitetaan päivittämällä pakettilistaus komennolla *sudo apt-get update*. Paketit asentuvat *sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils* – komennolla. Samassa yhteydessä asentuvat sekä vmbuilder – työkalu Ubuntu – pohjaisten virtuaalikoneiden luomista varten sekä bridge-utils – paketti, jonka avulla on mahdollista luoda siltaavia verkkoliitäntöjä virtuaalikoneiden käyttöön. Lopuksi kirjautunut käyttäjä lisätään kvm- ja libvirtd – ryhmiin: *sudo adduser <käyttäjätunnus> kvm* ja *sudo adduser <käyttäjätunnus> libvirtd* sekä suoritetaan uudelleenkirjautuminen muutosten vahvistamiseksi. Tämän jälkeen itse hypervisorin asennus on valmis. Toiminnan voi tarkistaa libvirt – pakkauksen mukana tulevan *virsh* – käyttöliittymän avulla muodostamalla yhteys paikalliseen qemu – instanssiin ja antamalla virtuaalikoneiden listauskäsky komennolla *sudo virsh -c qemu:///system list***.** Tulosteena komennosta syntyy onnistuneen asennuksen seurauksena tyhjä listaus, jonka otsikkoina Id, Name ja State.

5.4 Verkkoyhteyden siltaaminen ja palomuuri

Virtuaalikoneiden verkkoasetuksia määritettäessä valitaan hypervisor – alustasta riippumatta yleensä jokin kolmesta vaihtoehdosta:

- Sisäinen verkko, jossa virtuaalikoneet ovat keskenään verkossa virtuaalisen kytkimen välityksellä. Ei yhteyttä isäntäkäyttöjärjestelmään tai sen ulkopuolelle.
- Yhteys isäntäkoneen verkkoon NAT – käännöksellä.
- Suora yhteys isäntäkoneen verkkokortin kautta käyttäen siltausta, jolloin yhdistetyt laitteet näkyvät verkossa.

Jotta ainakin yksi virtuaalikoneista saataisiin näkyväksi ulkoverkosta, on isäntäkoneen verkkoyhteys siis sillattava luvussa 5.3 asennetulla bridge-utils – pakkauksella. Tätä varten on muokattava käyttöjärjestelmän verkkoasetustiedostoa komennolla *sudo vi /etc/network/interfaces*. Tiedostossa luodaan uusi looginen br0 – liitäntä, jolle lisätään eth0 – portin käyttämät IP – osoitemääritykset sekä erilliset siltausasetukset (kuva 14). Samalla poistetaan eth0 – liitännästä määritykset ja asetetaan sen tilaksi ”manual”. Tiedoston tallentamisen jälkeen muutokset tulevat voimaan */etc/init.d/networking restart* – komennolla. Muutosten voimaantulon voi todentaa *ifconfig* – tulosteesta, jossa näkyvissä ovat juuri luotu br0 – portti (kuva 15) sekä libvirtin NAT – käännösportti virbr0.

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet manual

auto br0
iface br0 inet static
        address 192.168.100.2
        netmask 255.255.255.0
        network 192.168.100.0
        broadcast 192.168.100.255
        gateway 192.168.100.1
        # dns-* options are implemented by the resolvconf package, if installed
        dns-nameservers 192.168.100.1
        bridge_ports eth0
        bridge_fd 9
        bridge_hello 2
        bridge_maxage 12
        bridge_stp off
```

KUVA 14. Verkkoliitäntöjen asetustiedosto /etc/network/interfaces, jossa julkiset IP – osoitteet vaihdettu yksityisiksi.

```
demo@demo:~$ ifconfig
br0       Link encap:Ethernet  HWaddr [redacted]
          inet addr:[redacted]  Bcast:[redacted]  Mask:[redacted]
          inet6 addr: [redacted] Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:206562 errors:0 dropped:0 overruns:0 frame:0
          TX packets:136506 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:145658056 (145.6 MB)  TX bytes:16441467 (16.4 MB)
```

KUVA 15. Br0 – siltausportti ifconfig - tulosteessa

Palomuuriohjelmistona käytetään Unbuntun mukana asentuvaa ufw – sovellusta, joka toimii yksinkertaisena käyttöliittymänä Linuxin kernelin iptables – palomuurisäännöille. Ufw otetaan käyttöön *sudo ufw enable* – komennolla. Sääntölistaa muokataan komennoilla *sudo ufw [delete] [insert NUM] allow|deny|reject|limit [in|out on INTERFACE] [log|log-all] [proto protocol] [from ADDRESS [port PORT]] [to ADDRESS [port PORT]]* ja se saadaan näkyviin antamalla *sudo ufw status numbered* - komento.

Br0 - liitäntä aiheuttaa ongelmia palomuuriin, sillä oletuksena ufw:n sääntölista koskee vain siltaamattomia liitäntöjä, ja torjuu kaiken sillatun liikenteen. Tämä ongelma ratkaistaan sallimalla sillattu liikenne tiedostossa /etc/ufw/before.rules lisäämällä rivi *”-I FORWARD -m physdev –physdev-is-bridged -j ACCEPT*” viimeiseksi ennen COMMIT – käskyä. Kyseisessä tiedostossa luetellut säännöt toteutetaan ennen ufw:n käyttöliittymässä määriteltyjä sääntöjä.

5.5 Virtuaalikoneiden luominen

Ubuntun palvelinversiossa virtuaalikoneiden luomisprosessi suoritetaan kahdella eri tavalla: *vmbuilder* (asennettu luvussa 5.3) Ubuntu – pohjaisille ja *virt-install* muilla käyttöjärjestelmillä varustetuille virtuaalikoneille. Työpöytäympäristössä asennustyökalujen hallintaan voidaan käyttää graafista *virt-manager* – ohjelmaa. Tällä kertaa asennus kuitenkin suoritettiin komentorivipohjaisesti.

Ensimmäinen virtuaalikone asennettiin *vmbuilder*:lla käyttäen Ubuntu Server 12.04 – käyttöjärjestelmää. Asennuksen onnistumiseksi suoritetaan valmistavat toimenpiteet, joista ensimmäisenä oman hakemiston luominen /vm/ - kansion alle:
mkdir /vm/vm1/.

Asennuskansioon kopioidaan ensin *vmbuilder*:n käyttämät libvirtin mallitiedostot *templates* – alikansioon:
mkdir /vm/vm1/templates/
cp /etc/vmbuilder/libvirt/ /vm/vm1/templates/*

Halutessa voidaan luoda konfigurointitiedosto virtuaalilevyn osoinnille ja Unix-hakemistoille:
vi /vm/vm1/vmbuilder.partition

root 25000
swap 80000

/var/ 50000

Kahdella ensimmäisellä rivillä määritellään root- ja swap- osioiden koot megatavuina. Katkoviivoilla erotetaan uusi levynkuva, johon sijoitetaan */var/* - hakemisto. Tämän jälkeen on vielä mahdollista eritellä halutut toiminnot erillisessä komentosarjatiedostossa *boot.sh*, joka suoritetaan virtuaalikoneen käynnistyksen yhteydessä. Tässä tiedostossa kannattaa suorittaa *openssh-server* – paketin asennus, jotta virtuaalikoneen ssh-avaimesta saadaan yksilöllinen. Paketin asennus onnistuu myös virtuaalikoneen luomiseen käytettävässä komennossa, mutta tällöin ssh - avain jaetaan muiden virtuaalikoneiden kanssa. Samalla asetetaan virtuaalikoneeseen luotavan käyttäjän salasana umpeutuvaksi, jolloin se on vaihdettava ensimmäisellä kirjautumiskerralla:
vi /vm/vm1/boot.sh

Tiedoston sisältö:

apt-get update
apt-get install -qqy --force-yes openssh-server
passwd –e testuser

Valmisteluja seuraa varsinainen *vmbuilder* – komento, joka on suoritettava virtuaalikoneelle osoitetussa hakemistossa. Asennus ei vaadi käyttöjärjestelmän levynkuvaa, mutta esimerkkikomennossa se on määritelty *--iso* **–** parametrissa.

cd /vm/vm1/
sudo vmbuilder kvm ubuntu --suite=precise --flavour=virtual --arch=amd64 --iso=/home/demo/ubuntu-12.04.1-server-amd64.iso --mirror=http://fi.archive.ubuntu.com/ubuntu -o --libvirt=qemu:///system --ip=x.x.x.x --gw=y.y.y.y --part=vmbuilder.partition --templates=templates --user=testuser --name=user --pass=password --addpkg=vim-nox --addpkg=unattended-upgrades --addpkg=acpid --firstboot=/vm/vm1/boot.sh --mem=4096 --hostname=ubuntu2 --bridge=br0

Komennolla luodaan virtuaalikone, jolle asetetaan yksi 64 – bittinen virtuaalinen prosessoriydin, 4 gigatavua muistia, ja verkkokortti yhdistettäväksi isäntäkäyttöjärjestelmän br0 – liitäntään. Samalla määritetään verkkokortin IP – asetukset, käyttäjätiedot, Ubuntun käyttämä apt-repository sekä asennettavat pakkaukset.

Vertailun vuoksi KVM - alustalle asennettiin Linux – jakeluista myös CentOS 6 v*irt-install* – työkalulla, joka osoittautui jonkin verran *vmbuilder*:a yksinkertaisemmaksi ratkaisuksi. *Virt-install* asennetaan komennolla:
sudo apt-get install virtinst

Asennusta varten täytyy ensin luoda *vmbuilder*:n tapaan asennuskansio uudelle virtuaalikoneelle ja siihen erillinen qemu:n käyttämä qcow2 – levynkuva, jonka kooksi määritellään 12 gigatavua:
mkdir /vm/vm2/
sudo qemu-img create -f qcow2 -o preallocation=metadata /vm/vm2/centos6.qcow2 12G

Seuraavaksi suoritetaan *virt-install* – asennuskomento. Asennuksessa käytetään hyväksi CentOS:n omaa asennusohjelmaa, joka saadaan näkyviin *--vnc* – parametrilla:

sudo virt-install --connect=qemu:///system --name=centos6 --vcpus=1 --disk path=/vm/vm2/centos6.qcow2,format=qcow2,bus=virtio,cache=none --ram=1024 --os-type=linux --os-variant=rhel6 --network=bridge:br0,model=virtio --cdrom=/home/demo/CentOS-6.4-x86_64-bin-DVD1.iso --accelerate –vnc

5.6 Virtuaalikoneiden käsittely

Virsh on KVM:ssa tärkein työkalu virtuaalikoneiden käsittelyyn. Sen avulla pystytään suorittamaan kaikki virtuaalikoneisiin kohdistuvat toimenpiteet tarkastelusta ja laitteiston muokkaamisesta aina verkkojen luontiin, tilakaappauksiin ja virtuaalikoneen kopioimiseen toisille alustalle (migration). Asennettujen virtuaalikoneiden listaus tapahtuu komennolla *list --all* (kuva 16).

```
demo@demo:~$ sudo virsh
[sudo] password for demo:
Welcome to virsh, the virtualization interactive terminal.

Type:  'help' for help with commands
       'quit' to quit

virsh # list --all
 Id Name                 State
----------------------------------
  1 ubuntu2              running
  2 centos6              running
  - ubuntu3              shut off

virsh #
```

KUVA 16. Virtuaalikoneiden listaus virsh - konsolissa

Virsh käyttää virtuaalikoneista nimitystä *domain*. Muita tärkeimpiä virtuaalikoneisiin kohdistuvia virsh – komentoja on lueteltu taulukossa 2. Yhdistäminen toiseen hypervisoriin tapahtuu komennolla *connect <hostname|url>*.

TAULUKKO 2. Yleisimpiä Virsh – komentoja virtuaalikoneille

Komento	Selitys
start	Käynnistää virtuaalikoneen
suspend	Keskeyttää virtuaalikoneen suorittamisen (varaa silti muistia)
resume	Palauttaa virtuaalikoneen *suspend* - tilasta
shutdown	Antaa ACPI - sammutuskäskyn virtuaalikoneelle
destroy	Katkaisee virran virtuaalikoneesta
save	Tallentaa virtuaalikoneen laitteiston tilan XML - tiedostoon
restore	Palauttaa *save* – komennolla tallennetun tilan
edit	Näyttää virtuaalikoneen XML – asetustiedoston
create	Luo tyhjän virtuaalikoneen valmiista XML - tiedostosta
undefine	Poistaa virtuaalikoneen
console	Yhdistää virtuaalikoneen konsoliporttiin, mikäli sellainen on määritelty XML - asetustiedostossa

KVM säilyttää virtuaalikoneiden tietoja omissa XML – tiedostoissa (liite 1). Tiedostoa muokkaamalla virtuaalikoneeseen voidaan esimerkiksi muuttaa muistin tai prosessoriydinten määrää tai lisätä laitteistoa. Yleinen Linux - virtuaalikoneisiin lisättävä laite on virtuaalinen konsoliportti, jotta koneen hallinta onnistuu myös paikallisesti SSH – yhteyden katkettua. Tätä varten täytyy <*devices*> - elementin sisälle lisätä rivit:

```
<serial type='pty'>
    <target port='0'/>
</serial>
<console type='pty'>
    <target type='serial' port='0'/>
</console>
```

Portti otetaan Ubuntu - virtuaalikoneessa käyttöön lisäämällä sille oma asetustiedosto /etc/init/ - kansioon:

sudo vi /etc/init/ttyS0.conf

Tiedoston sisältö:

start on stopped rc RUNLEVEL=[2345]

stop on runlevel [!2345]

respawn

exec /sbin/getty -L 38400 ttyS0 vt102

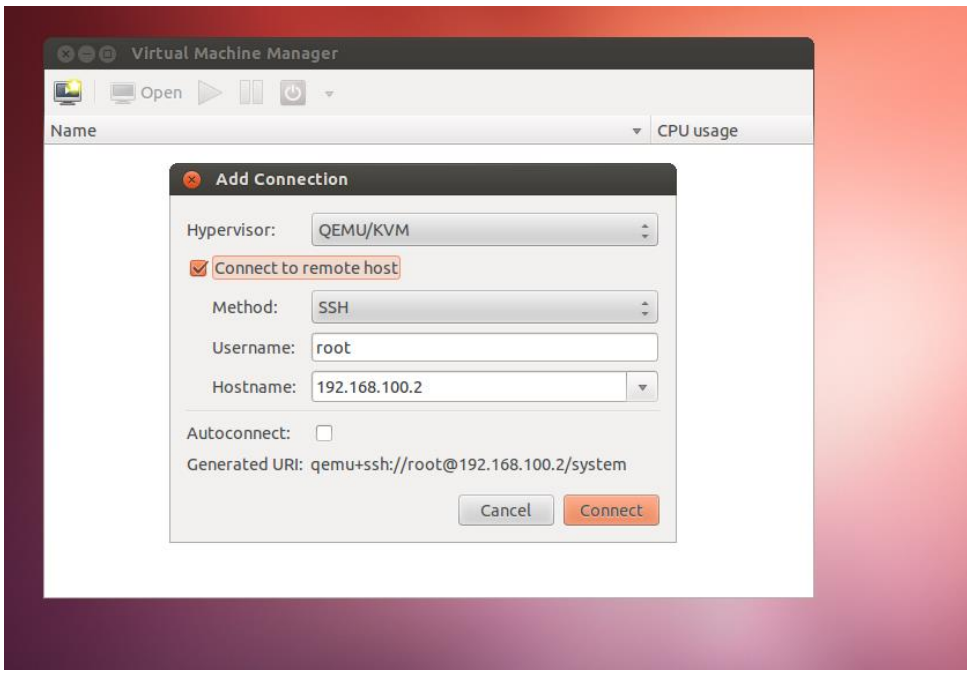
Tallentamisen jälkeen virtuaalikone sammutetaan. Isäntäkoneessa on vielä käynnistettävä libvirt uudelleen muutosten saattamiseksi voimaan:

sudo service libvirt-bin <stop|start>

Virtuaalikoneen käynnistämisen jälkeen konsoliyhteyden voi muodostaa *virsh*:n *console* – komennolla:

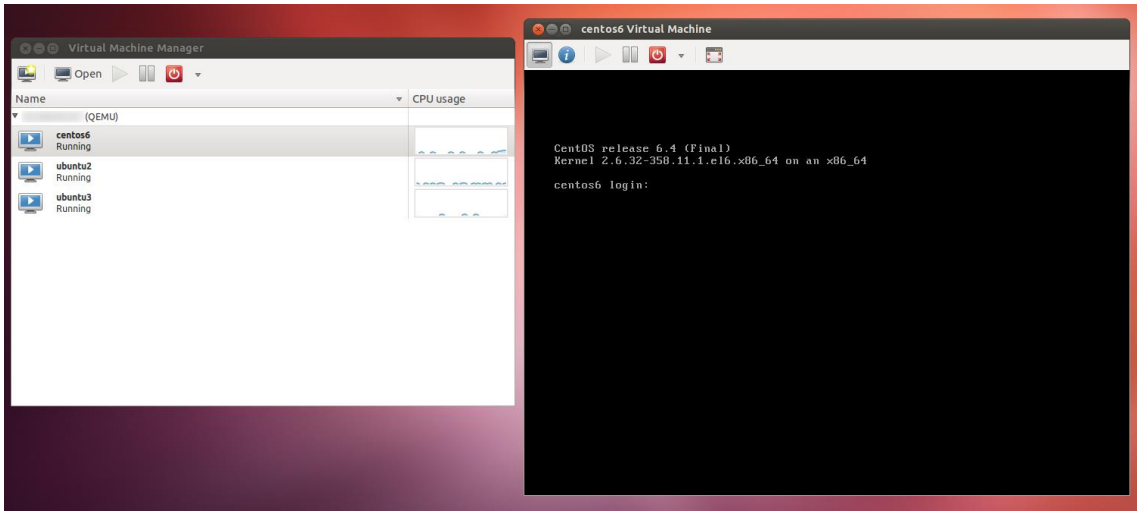
sudo virsh console ubuntu3

Hypervisoriin yhdistäminen onnistuu myös graafisella Virtual Machine Manager (pakkaus *virt-manager*) – ohjelmalla, joka on asennettu valmiiksi Ubuntun 12.04 – työpöytäversioon. Yhteys muodostetaan valitsemalla File – valikosta Add Connection, josta aukeavaan laatikkoon syötetään hypervisor – palvelimen tiedot (kuva 17).



KUVA 17. Virtual Machine Managerin yhdistäminen hypervisoriin

Yhdistämisen jälkeen aukeavat näkyviin hypervisorin alle asennetut virtuaalikoneet. Virtualikoneeseen yhdistäminen tapahtuu valitsemalla koneen nimi valikosta, jolloin aukeaa uusi ikkuna virtuaalikoneen työpöydälle tai graafiselle konsoliyhteydelle (kuva 18).



KUVA 18. Virtuaalikonelistaus ja graafinen konsoli Virtual Machine Managerissa

5.7 Palvelinohjelmistojen asennus

Yksi Ubuntu - virtuaalikoneista valjastettiin simuloimaan toimeksiantajan Toyme - palveluun käytettävää tuotantoympäristöä. Tätä varten palvelimeen asennettiin tärkeimmät web – palvelinohjelmistot, joiden päälle palvelu voidaan rakentaa.

5.7.1 Apache Tomcat

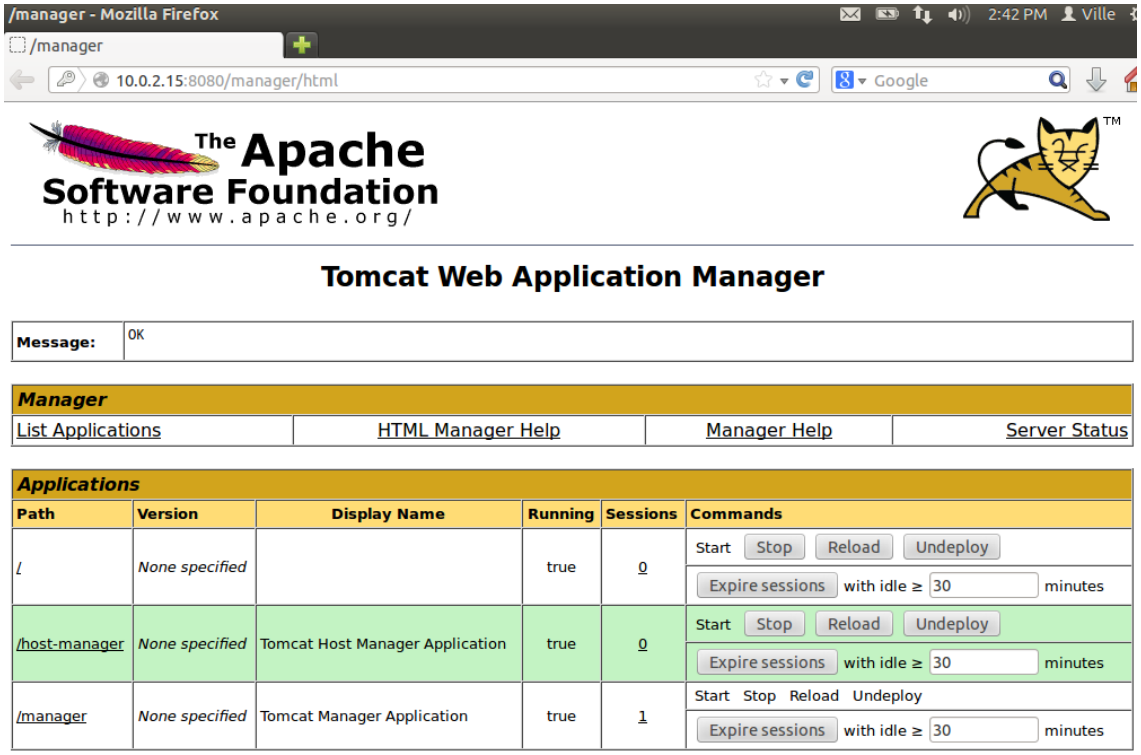
Apache Tomcat on Apache Software Foundationin kehittämä HTTP – palvelin Java Servlet ja JavaServer Pages (JSP) – tekniikoilla toteutettujen sovellusten alustaksi. Kirjoitushetkellä viimeisin vakaa Tomcat – versio 7.0.42 tukee Javan versioita 1.6 ja 1.7 sekä Java Servlet – määrittelyn versiota 3.0 (Apache Tomcat 2013). Se asennetaan Ubuntussa komennolla *sudo apt-get install tomcat7* ja käynnistetään tämän jälkeen komennolla *sudo service tomcat7 start*. Lisäksi on asennettava sovellusten hallintatyökalu Tomcat Manager komennolla *sudo apt-get install tomcat7-admin*. Ennen Manager – sovelluksen käyttöönottoa on vielä luotava uusi käyttäjäprofiili Tomcatin asennuskansion käyttäjätiedostoon: *sudo vi/etc/tomcat7/tomcat-users.xml*.

Tiedostoon lisätään käyttäjärooli ”manager” ja sitä käyttävä käyttäjätunnus:

<role rolename="manager"/>

<user username="admin" password="password" roles="manager"/>

Tomcatin uudelleenkäynnistyksen jälkeen manager – hallintasivuun voidaan ottaa yhteys selaimella antamalla osoite http://<ip|domain>:8080/manager/html (kuva 19)



KUVA 19. Tomcat Manager – hallintapaneeli

Manager – hallintasivulla uuden sovelluksen lisääminen onnistuu lataamalla se WAR – pakkauksena joko selaimella, palvelimen omasta hakemistosta, tai ulkoisesta URL – osoitteesta.

5.7.2 Apache HTTP server

Tomcatin rinnalle asennettu Apache HTTP server on maailman suosituin HTTP - palvelinohjelmisto. Vaikka Tomcat itsessäänkin riittäisi web – sivujen hallintaan, on Apachen asentaminen sen suositeltavaa muun muassa seuraavista syistä:

- Ohjaus HTTP – portista 80 Tomcatin porttiin 8080 ja takaisin (onnistuu tosin myös Tomcatissa asettamalla Connector kuuntelemaan porttia 80)
- Helpompi sivustojen pääsynhallinta
- Yksinkertaisempi TLS – sertifikaattien asennus
- Monipuolisemmat ominaisuudet, kuten rewrite ja proxypass – toiminnot

Apache asennetaan Ubuntussa komennolla *sudo apt-get install apache2*. Asennuksen jälkeen palveluun varten luodaan oma site - asetustiedosto /etc/apache2/sites-available – kansioon: *sudo vi /etc/apache2/sites-available/demo*. Porttiohjaus 80 → 8080 suoritetaan aktivoimalla ensin ProxyPass – toiminto komennolla *sudo a2enmod proxy* ja ottamalla se sitten käyttöön juuri luodun tiedoston <VirtualHost> - elementtiin lisättävällä ProxyPreserveHost On – rivillä. Itse ohjauskäskyt ovat muotoa:

ProxyPass / http://127.0.0.1:8080/
ProxyPassReverse / http://127.0.0.1:8080/

Luotu tiedosto aktivoidaan komentorivillä *a2ensite* – komennolla. Samalla poistetaan default – tiedosto käytöstä:
sudo a2ensite demo
sudo a2dissite default

5.7.3 MySQL ja phpMyAdmin

Palvelussa käytetään tietokantasovellus MySQL:a, joka asennetaan komennolla *apt-get install mysql-server*. Asennuksessa määritellään sovelluksen root – käyttäjän salasana. Samalla MySQL:n hallintaa varten asennettiin phpMyAdmin – työkalu komennolla *sudo apt-get install phpmyadmin*. PhpMyAdminia varten luvussa 5.7 asennettuun porttiohjaukseen on tehtävä poikkeus, jotta sitä ei ohjata Tomcatin porttiin 8080:
ProxyPass /phpmyadmin !

6 POHDINTA

Virtualisointi ja sen mukanaan tuomat hyödyt ovat opiskelujeni aikana nousseet niin merkittävään rooliin, että virtualisointitekniikoihin ja palveluntarjoajiin syventyminen tuntui luontevalta aiheelta opinnäytetyölle. Työelämässä pilvipalvelun tuottaminen puolestaan on aukaissut näkymiä mahdollisuuksille, joita toimivat ja helppokäyttöiset pilvijärjestelmät tehokkaan virtuaalialustan päälle asennettuna tuovat yritysmaailmaan. Opinnäytetyössä tutustuttiin virtualisoinnin periaatteeseen, eri virtuaalikonetyyppeihin sekä hypervisorien toimintaan. Pilvipalveluiden kohdalla työssä keskityttiin erittelemään pilvipalveluiden toimintamalleja ja niiden käyttöä organisaatioissa tekniseltä ja taloudelliselta kannalta.

Omalla alustalla ylläpidetyn, vakaan tietojärjestelmän siirtämisestä pilveen löytyy edelleen tekijöitä, jotka nostavat hankintakynnyksen korkealle. Vaikka pilvipalvelujen hinnat ovat laskeneet ja käyttöönottotyypit monipuolistuneet, ovat tietoturva, käyttöönottoprosessin hankaluus, järjestelmän altistuminen ulkoisen palveluntarjoajan hoidettavaksi tai esimerkiksi huonot kokemukset aiemmasta pilvipalvelusta vieläkin haittoina tai esteinä uuteen toimintamalliin siirryttäessä. Lieneekin totta, että koko IT – infastruktuuria ei olekaan syytä muuttaa pilvimalliseksi. Kannattaa kuitenkin muistaa, että olemassa olevia järjestelmiä voidaan tehostaa esimerkiksi virtualisoimalla niiden osia tai liittämällä niitä rajapintojen avulla pilvipalveluihin hybrid cloud – periaatteella.

Oman hajautetun IaaS – ympäristön rakentaminen vapaan lähdekoodin palvelukokonaisuuksia hyödyntäen on projekti, johon Suomessa riittävät resurssit vain suurimmilla yrityksillä. Pienemmässä mittakaavassa pilvipalveluiden käyttöönoton ja ylläpitämisen voidaan katsoa olevan jo niin kannattavaa, että sen pitäisi kuulua jokaisen pk – yrityksen IT – rutiineihin. Toimeksiantajalla tätä rutiinia kehitettiin toteuttamalla oma virtuaalinen konesali ja tutkimalla sen ominaisuuksia demokäytössä. Projekti onnistuikin hyvin, sillä itse konesalin pystytys ja virtuaalikoneiden lisääminen onnistuivat melko yksinkertaisesti nopealla aikataululla.

Suurimmat haasteet opinnäytetyössä liittyivät KVM:n dokumentaatioon, joka osoittautui useampaan kertaan laajaksi mutta monimutkaiseksi. Asennusprosessissa eteen tulleiden ongelmien tutkiminen edellytti monesti niin perusteellista Linux – ympäristön

osaamista, että ratkaisussa oli tyydyttävä yksinkertaisimpaan vaihtoehtoon. Tästä huolimatta tuntuma hypervisoreihin ja käyttöliittymärajapintoihin parani jatkuvasti, jolloin myös erot virtualisointialustojen välillä selkenivät. Yksi pettymys opinnäytetyön toteuttamisessa oli myös se, että KVM:n erikoisominaisuuksien, kuten tilakaappausten hallinnan tai migration – toiminnon, tutkimiseen ei valitettavasti jäänyt tarpeeksi aikaa. Lisäksi yksinkertaisen virtuaaliympäristön hallintaoppaan kirjoittaminen ei ehtinyt mukaan tähän työhön. Projektin lopputuloksena syntyneen palvelimen voi katsoa täyttäneen tavoitteensa, sillä se on jo avannut yrityksissä mahdollisuuden tehokkaaseen asiakasprojektien demoamiseen sekä uusien käyttöjärjestelmien ja palveluiden testaamiseen sekä helpottanut tulevan tuotantokonesalin valintaa.

LÄHTEET

Furth, B. & Escalante, A. 2010. Handbook of Cloud Computing. New York: Springer Science+Business Media LLC

Heino, P. 2010. Pilvipalvelut. Helsinki: Talentum.

Hurwitz, J., Bloor R., Kaufman M. & Halper F. 2010. Cloud Computing For Dummies. Hoboken, NJ: Wiley Publishing, Inc.

Petersen, R. 2010. Fedora 14 Administration and Security. Alameda: Surfing Turtle Press.

Portnoy, M. 2012. Virtualization Essentials. Indianapolis: John Wiley & Sons.

Raj, P. 2013. Cloud Enterprise Architecture. Boca Raton: CRC Press

Ruest, D. & Ruest, N. 2009. Virtualization: A Beginner's Guide. New York: McGraw-Hill.

Salo, I. 2010. Cloud Computing – palvelut verkossa. Helsinki: WSOYpro.

Warnke, R. & Ritzau, T. 2010. QEMU & KVM. 4. Painos. Norderstedt: Books on Demand.

Smith, J. E. & Nair, R. 2005. The Architecture of Virtual Machines. Computer 38 (5), 32-38.

Uddin, M. & Azizah A. R. 2010. Server Consolidation: An Approach to Make Data Centers Energy Efficient & Green. International Journal of Scientific & Engineering Research 1 (1).

Graziano, C.D. 2011. A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project. Iowa State University. Degree of master of science. Opinnäytetyö.

Popović, O., Jovanović, Z., Jovanović, N. & Popović, R. 2011. A Comparison and Security Analysis of the Cloud Computing Software Platforms. IEEE Explore.

Diversity Limited. 2011. Understanding The Cloud Computing Stack. Luettu 24.7.2013. http://www.rackspace.com/knowledge_center/sites/default/files/whitepaper_pdf/Understanding-the-Cloud-Computing-Stack.pdf

Jha, A., D, J., Murari, K., Raju M., Gherian, V. & Girikumar, Y. 7.5.2012. OpenStack Beginner's Guide (for Ubuntu - Precise). CSS Corporation. Luettu 26.7.2013. http://cssoss.files.wordpress.com/2012/05/openstackbookv3-0_csscorp2.pdf

Lakshmi, J. & Vadhiyar, S. 2011. Cloud Computing: A Bird's Eye View. Bangalore: Indian Institute of Science. Luettu 26.7.2013 http://www.serc.iisc.ernet.in/~jlakshmi/Research/CloudsandQoS/Cloud%20Computing-BirdsEyeView-Oct2011.pdf

Mell, P. & Grance, T. 2011. The NIST Definition of Cloud Computing. Gaithersburg, MD: National Institute of Standards and Technology. Luettu 25.7.2013 http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Subramanian, K. 2011. Hybrid Clouds. Trend Micro Inc. Luettu 19.7.2013. http://la.trendmicro.com/media/wp/hybrid-clouds-whitepaper-en.pdf

Apache Tomcat. 2013. Apache Tomcat Versions. Apache Software Foundation. Luettu 23.8.2013. http://tomcat.apache.org/whichversion.html

Brockmeier, J. 25.3.2013. The Apache Software Foundation Announces Apache CloudStack Has Become A Top-Level Project. The Apache Software Foundation. Luettu 5.8.2013. https://blogs.apache.org/cloudstack/entry/the_apache_software_foundation_announces

CloudStack. 2013a. About Apache CloudStack. The Apache Software Foundation. Luettu 5.8.2013. http://cloudstack.apache.org/about.html

CloudStack. 2013b. CloudStack Administrator's Guide. The Apache Software Foundation. Luettu 6.8.2013. http://cloudstack.apache.org/docs/en-US/Apache_CloudStack/4.1.1/pdf/Admin_Guide/Apache_CloudStack-4.1.1-Admin_Guide-en-US.pdf

Eucalyptus. 2013a. Eucalyptus FAQ. Eucalyptus Systems Inc. Luettu 1.8.2013. http://www.eucalyptus.com/faq

Eucalyptus. 2013b. Eucalyptus Powers the Ubuntu Enterprise Cloud in Ubuntu 9.10. Eucalyptus Systems Inc. Luettu 1.8.2013. http://www.eucalyptus.com/news/10-26-2009

Eucalyptus. 2013c. Eucalyptus Cloud Compatibility Matrix. Eucalyptus Systems Inc. Luettu 1.8.2013. http://www.eucalyptus.com/eucalyptus-cloud/iaas/compatibility

Eucalyptus. 2013d. AWS and Eucalyptus Compatibility. Eucalyptus Systems Inc. Luettu 1.8.2013.
http://www.eucalyptus.com/aws-compatibility

Eucalyptus. 2013e. Eucalyptus 3.3.0.1 Installation Guide. Eucalyptus Systems Inc. Luettu 2.8.2013.
http://www.eucalyptus.com/docs/eucalyptus/3.3/install-guide-3.3.0.pdf

Google. 2013. Google App Engine. Luettu 23.7.2013.
https://cloud.google.com/products/

Hardiman, N. 2012. What we mean when we talk about cloud computing. TechRebublic. Luettu 18.7.2013
http://www.techrepublic.com/blog/the-enterprise-cloud/what-we-mean-when-we-talk-about-cloud-computing/

IBM. 2007. Virtualization in Education. Luettu 17.6.2013.
http://www-07.ibm.com/solutions/in/education/download/Virtualization%20in%20Education.pdf

Intel 2013. About Intel® Virtualization Technology. Intel Corporation. Luettu 20.8.2013.
http://ark.intel.com/Products/VirtualizationTechnology

Libvirt. 2013a. Index. Luettu 8.7.2013.
http://libvirt.org/index.html

Libvirt. 2013b. Applications using libvirt. Luettu 8.7.2013.
http://libvirt.org/apps.html

Morgan, T. 13.10.2011. Ubuntu Server 11.10 leaps onto OpenStack clouds. The Register. Luettu 1.8.2013.
http://www.theregister.co.uk/2011/10/13/ubuntu_server_11_10/

Nimbus. 2013a. Nimbus Frequently Asked Questions – Nimbus 1.0 documentation. University of Chicago. Luettu 8.8.2013.
http://www.nimbusproject.org/doc/nimbus/faq/

Nimbus. 2013b. Cloudinit.d tutorial. University of Chicago. Luettu 8.8.2013.
http://www.nimbusproject.org/doc/cloudinitd/latest/tutorial.html

Nimbus. 2013c. Publications. University of Chicago. Luettu 8.8.2013.
http://www.nimbusproject.org/papers/

O'Brien, R. 29.5.2012. NASA and OpenStack 2012. Nebula Cloud Computing Platform. National Aeronautics and Space Administration. Luettu 26.7.2013.
http://nebula.nasa.gov/blog/2012/05/29/nasa-and-openstack-2012/

OpenStack. 2013a. Companies Supporting The OpenStack Foundation. Luettu 26.7.2013.
http://www.openstack.org/foundation/companies/

OpenStack. 2013b. About OpenStack. The OpenStack Foundation. Luettu 26.7.2013. http://www.openstack.org/software/

OpenStack. 2013c. Installing the OpenStack dashboard. The OpenStack Foundation. Luettu 26.7.2013. http://docs.openstack.org/essex/openstack-compute/install/apt/content/ch_install-dashboard.html

OpenStack 2013d. Selecting a Hypervisor. The OpenStack Foundation. Luettu 31.7.2013. http://docs.openstack.org/trunk/openstack-compute/admin/content/selecting-a-hypervisor.html

Science Clouds. 2013. Contact. University of Chicago. Luettu 8.8.2013. http://www.nimbusproject.org/papers/

TuxRadar. 2010. Howto: Linux and Windows virtualization with KVM and Qemu. Luettu 8.7.2013. http://www.tuxradar.com/content/howto-linux-and-windows-virtualization-kvm-and-qemu

Windows Azure. 2013. Download the Windows Azure SDK for Java. Luettu 23.7.2013. http://www.windowsazure.com/en-us/develop/java/java-home/download-for-windows/

VMTech. 2011. History of Virtualization. Luettu 17.6.2013. http://www.vmtech.com.au/virtualizationhistory.html

VMware. 2012. VMware vSphere® Data Protection. Luettu 4.7.2013. http://www.vmware.com/files/pdf/techpaper/Introduction-to-Data-Protection.pdf

LIITTEET

Liite 1. KVM:n käyttämä virtuaalikoneen XML – tiedosto.


```
<domain type='kvm'>
  <name>ubuntu3</name>
  <uuid>bc47eb69-7fba-0e72-aca3-92dfb01c5bee</uuid>
  <memory>4194304</memory>
  <currentMemory>4194304</currentMemory>
  <memtune>
    <hard_limit>4000000</hard_limit>
    <soft_limit>4000000</soft_limit>
  </memtune>
  <vcpu>1</vcpu>
  <os>
    <type arch='x86_64' machine='pc-1.0'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <devices>
    <emulator>/usr/bin/kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'/>
      <source file='/vm/ubuntu3/ubuntu-
kvm/tmpYq_baH.qcow2'/>
        <target dev='hda' bus='ide'/>
      <address type='drive' controller='0' bus='0'
unit='0'/>
    </disk>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'/>
      <source file='/vm/ubuntu3/ubuntu-
kvm/tmpAVQXQI.qcow2'/>
      <target dev='hdb' bus='ide'/>
      <address type='drive' controller='0' bus='0'
unit='1'/>
    </disk>
    <controller type='ide' index='0'>
      <address type='pci' domain='0x0000' bus='0x00'
slot='0x01' function='0x1'/>
    </controller>
    <interface type='network'>
      <mac address='52:54:00:4f:76:91'/>
```

```
      <source network='default'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00'
slot='0x08' function='0x0'/>
    </interface>
    <interface type='bridge'>
      <mac address='52:54:00:25:1f:c2'/>
      <source bridge='br0'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00'
slot='0x09' function='0x0'/>
    </interface>
    <serial type='pty'>
      <target port='0'/>
    </serial>
    <console type='pty'>
      <target type='serial' port='0'/>
    </console>
    <input type='mouse' bus='ps2'/>
    <graphics type='vnc' port='-1' autoport='yes' lis-
ten='127.0.0.1'>
      <listen type='address' address='127.0.0.1'/>
    </graphics>
    <video>
      <model type='cirrus' vram='9216' heads='1'/>
      <address type='pci' domain='0x0000' bus='0x00'
slot='0x02' function='0x0'/>
    </video>
    <memballoon model='virtio'>
      <address type='pci' domain='0x0000' bus='0x00'
slot='0x04' function='0x0'/>
    </memballoon>
  </devices>
</domain>
```