Haaga-Helia
ammattikorkeakoulu Oy

Tuotantopalvelimen käyttöönotto ja palautus

Sami Rita

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2017

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

10.4.2017

Tekijä(t)
Sami Rita

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Raportin/Opinnäytetyön nimi
Tuotantopalvelimen käyttöönotto ja palautus

Sivu- ja liitesivumäärä
88 + 0

Opinnäytetyöprojekti aiheeltaan ”Tuotantopalvelimen käyttöönotto ja palautus” käynnistettiin loppuvuonna 2016 perustuen toimeksiantajan –”Yritys Y”:n omassa projektissa tunnistettuihin tarpeisiin. Opinnäytetyö toteutettiin kokonaisuudessaan valmiiksi vuoden 2017 alkupuolella.

Projektin tavoitteena oli nimensä mukaisesti selvittää ja testata virtuaalisessa labraympäristössä tehokkaita käyttöönotto- ja palautusmenetelmiä toimeksiantajan Linux-palvelimia varten sekä analysoida kokeiltujen menetelmien käyttökelpoisuutta samalla dokumentoiden toteutukset yksilökohtaisesti. Palautettaessa palvelinta olennaisinta oli järjestelmän asetusten ja sovellusasennusten ennalleen saattaminen välittämättä palvelimelle mahdollisesti tallennetusta muusta datasta.

Opinnäytetyö alkaa johdannolla, jossa käydään läpi projektin lähtötilanne, aihe ja toteutusmalli. Johdannon jälkeen tulee tietoperusta tarvittavine ammattikäsitteineen sekä esitellään työssä käytettävät työkalut. Alun teoriaosuuden jälkeen toteutetaan Oracle VirtualBoxilla palautusmenetelmien testausta varten virtuaalinen testiympäristö.

Testiympäristössä suoritetaan käytännön testaukset valituilla menetelmillä, joihin lukeutuvat olemassa olevan palvelimen palautus levykuvasta Clonezilla-työkalulla, uuden palvelimen käyttöönotto Red Hat Linux Enterprisen kickstart-tiedostolla ja palvelinympäristön provisiointi Ansible-automatisointityökalulla.

Testien suorittamisen jälkeen analysoidaan valittujen palvelinten käyttöönotto- ja palautusmenetelmien käyttökelpoisuus sekä käydään läpi jatkokehitysideat. Analyysi sisältää suositukset toimeksiantajalle koskien työkalujen hyödyntämistä omassa kehityshankkeessaan. Viimeisenä on projektin yhteenveto sisältäen opinnäytetyöprosessin ja oman ammatillisen oppimisen käsittelyn.

Projektin tuloksena onnistuttiin selvittämään toimeksiantajan ympäristöön soveltuvia provisiointityökaluja, jotka tehokkaasti konfiguroituna vähentävät manuaalisen palvelintyön määrää merkittävästi. Erityisesti kickstart-tiedoston ja Ansible-työkalun yhteiskäytöllä on mahdollista tehdä palvelimen käyttöönotosta ja palautuksesta aiempaa nopeampi sekä vaivattomampi prosessi palvelinympäristön ylläpitäjälle.

Opinnäytetyön toteutusmalli noudatti löyhästi ohjelmistokehityksestä tuttua perinteistä vesiputousmallia. Vesiputousmallissa tehdään esisuunnittelun perusteella vaatimusmäärittely, jota seuraa suunnittelu, toteutus ja testaus kronologisessa järjestyksessä. Vesiputousmallin yleisperiaatteet soveltui hyvin tämän työn toteutukseen projektin pienuuden ja helpon hallittavuuden vuoksi. Saatujen tulosten myötä tehtävään laajempaan jatkokehitykseen soveltuu oletettavasti ketterämmät toteututusmallit paremmin.

Asiasanat
Palvelimet käyttöönotto palautus Ansible Kickstart Clonezilla

Haaga-Helia
University of Applied Sciences

Abstract

10.4.2017

Author(s)
Sami Rita

Degree programme
Business Information Technology

Report/thesis title	**Number of pages and appendix pages**
Production server deployment	88 + 0

Thesis project called “Production server deployment” based on a client’s requirements started in autumn 2016 and were finished in early 2017. The client – “company Y” needed a reliably deployment and backup solution for their Linux servers.

The aim of the project was to research and test effective deployment and backup methods to handle client’s needs for Linux servers on its own project. Also, analysing the usefulness of the tested tools was included in the thesis project.

The essential parts for the implementation process was to find an effective way for the client’s servers to deploy or return all needed system settings and software installations as required using selected IT-tools. When completing a backup of a server, it was not required to restore any other temporary data that might have been earlier in the server.

The thesis begins with the introduction which includes presentation of the project’s starting point, subject and implementation models. Introduction is followed by theory chapters which includes presenting all used deployment methods and tools. After that follows designing and implementing Oracle VirtualBox based test environment.

In the next chapters selected deployment and backup tools are tested in the earlier created virtual test environment. Selected tests include completing server backup using Clonezilla backup software, deploying new virtual server using Red Hat Enterprise Linux kickstart - configuration file and provisioning multiple Linux servers simultaneously using Ansible IT-automation tool.

In the last chapters tested methods are analysed as a purpose to find out which deployment method or combination of them would suit the best for the needs of client’s own development project. Also, further ideas about development are discussed. Lastly there is a reflection part where whole thesis project including the author’s learning in a project is walk through.

As a result of the project, applicable deployment and backup tools for the client’s Linux server environment were found. By using the seletecd deployment tools system administrator can save time significantly and minimize the need for manual server configuration work. Specially using the combination of kickstart-file and Ansible-tool is a powerful way to perform server deployment process.

Project followed loosely guidelines of waterfall deployment model which is better known from software deployment. Waterfall model’s basic principles suited well for this project, because project was relatively small and easy to maintain. For further development agile methods would probably serve better.

Keywords
Servers deployment restore Ansible Kickstart Clonezilla

Sisällys

1 Johdanto 1
1.1 Projektin tausta 1
1.2 Tekijä ja sidosryhmät 1
1.3 Aihe 1
1.4 Tavoitteet 2
1.5 Toteutusmalli 4
1.6 Vaiheistus 5
2 Tietoperusta 6
2.1 Ammattikäsitteet 6
2.2 Virtualisointi 7
2.3 Levyn kloonaus 8
2.4 Palvelimen provisiointi 8
2.5 Kickstart-tiedosto 9
3 Työkalut 10
3.1 CentOS 7 -käyttöjärjestelmä 10
3.2 Oracle VirtualBox 10
3.3 VMware Workstation Player 12
3.4 MobaXterm 15
3.5 Clonezilla 16
3.6 Ansible 17
3.7 Muut työkalut 18
4 Testiympäristö 20
4.1 Virtualisointityökalun valinta 20
4.2 Arkkitehtuurin suunnittelu 22
4.3 Hypervisorin asennus ja virtuaalikoneiden luominen 25
4.4 Käyttöjärjestelmän asentaminen virtuaalikoneisiin 28
4.5 Verkon konfigurointi 31
5 Palvelimen palautus levykuvasta 35
5.1 Paikallisen levykuvan luonti 35
5.2 Levykuvan palautus paikallisesti 38
5.3 Levykuvan ajo verkon välityksellä 41
6 Palvelimen käyttöönotto Kickstart-tiedostolla 49
6.1 Kickstartin suunnittelu ja toteutus 49
6.2 Käyttöjärjestelmäasennus Kickstartilla 53
7 Palvelimen provisiointi Ansiblella 57
7.1 Ansiblen asennus ja testaus 57
7.2 Ansible-ympäristön suunnittelu 62

7.3 Toteutus ja palvelinten provisiointi 63
8 Käyttöönotto- ja palautusmenetelmät 76
8.1 Yleistä palvelimen käyttöönotosta 76
8.2 Menetelmien analysointi 77
8.3 Menetelmien jatkokehitys 78
9 Projektin yhteenveto 80
Lähteet 81

1 Johdanto

Opinnäytetyön johdanto sisältää työn aiheen ja taustan käsittelyn sekä tekijän esittelyn. Lisäksi käydään läpi työn vaiheistusta ja käytettäviä toimintamalleja. Toimeksiantajaa kutsutaan opinnäytetyössä yrityksen oikean nimen sijaan nimellä ”Yritys Y”.

1.1 Projektin tausta

Opinnäytetyön suunnittelu käynnistettiin perustuen toimeksiantaja Yritys Y:n projektissa tunnistettuihin tarpeisiin. Toimeksianto muotoutui otsikon ”Tuotantopalvelimen käyttöönotto ja palautus” mukaiseen lopulliseen muotoonsa syksyn 2016 aikana, jonka jälkeen itse projekti toteutettiin valmiiksi vuoden 2017 alkupuolella.

Tekijälle opinnäytetyön käynnistämisen motivaattoreina olivat mielenkiintoisen ja tarpeellisen aiheen lisäksi ammatillisen osaamisen laajentaminen ja syventäminen samalla tähdäten tietojenkäsittelyn koulutusohjelmasta valmistumiseen.

1.2 Tekijä ja sidosryhmät

Opinnäytetyön tekijä opiskelee Haaga-Helia Ammattikorkeakoulussa tietojenkäsittelyn koulutusohjelmassa suuntautuen järjestelmäasiantuntijan opintoihin. Opintosuuntaus keskittyy palvelinympäristöjen, tietoverkkojen ja tietokantojen suunnitteluun sekä ylläpitoon. Lisäksi koulutusohjelmassa opetetaan monipuolisten perusopintojen lisäksi liiketoimintaprosessien mallintamista. (Haaga-Helia 2016.)

Opinnäytetyön tekijä oli suorittanut kaikki muut koulutusohjelmassa vaadittavat opinnot, opinnäytetyöprojektin aloitushetkellä syksyn 2016 loppupuolelta, joten opinnäytetyön toteuttaminen oli hyvin ajankohtaista.

Projektin sidosryhmiin kuuluvat Haaga-Helia Ammattikorkeakoulun puolelta opinnäytetyön ohjaamisesta ja arvioimisesta vastaava henkilökunta sekä työn opponoinnista vastaava opiskelija. Yritys Y:n puolelta sidosryhmiin lukeutuvat toimeksiantajan nimeämä ohjaaja sekä erillinen kehitystiimi, jonka tietoon ja käyttöön tämän opinnäytetyöprojektin lopputulokset saatetaan.

1.3 Aihe

Projektissa selvitetään ja testataan palvelimen käyttöönotto- ja palautusmenetelmiä toimeksiantajan projektissa käyttämiä palvelinlaitteistoja varten. Tätä tarkoitusta varten on

kattavien selvityksien jälkeen valittu joukko palvelinten käyttöönottoon ja palautukseen soveltuvia työkaluja. Lisäksi työn alkuosa sisältää teoriapohjaa valituista työkaluista ja käytettävistä menetelmistä.

Testaukset suoritetaan itse rakennetussa virtuaalisessa labraympäristössä ja kaikki toteutettavat testit dokumentoidaan vaihe vaiheelta. Testien suorittamisen jälkeen analysoidaan kunkin työkalun hyödynnettävyys ja soveltuvuus aiheen mukaiseen käyttötarkoitukseen, niin yleisesti kuin itse toimeksiantajan näkökulmasta.

Suoritettavissa testeissä olennaisinta on järjestelmän asetusten ja sovellusasennusten tehokas konfigurointiprosessi. Palvelin tulisi pystyä saattamaan testattavilla työkaluilla käyttövalmiiksi mahdollisimman vähäisin manuaalisin toimenpitein oli kyse sitten palvelimen palauttamisesta käyttökelpoiseksi tai täysin uuden palvelimen käyttöönotosta.

Palvelimen palautukseen soveltuvia työkaluja testatessa ei tarvitse ottaa huomioon palvelimelle mahdollisesti tallennetun muun datan palautusta, joka ei liity järjestelmäasetuksiin tai sovellusasennuksiin.

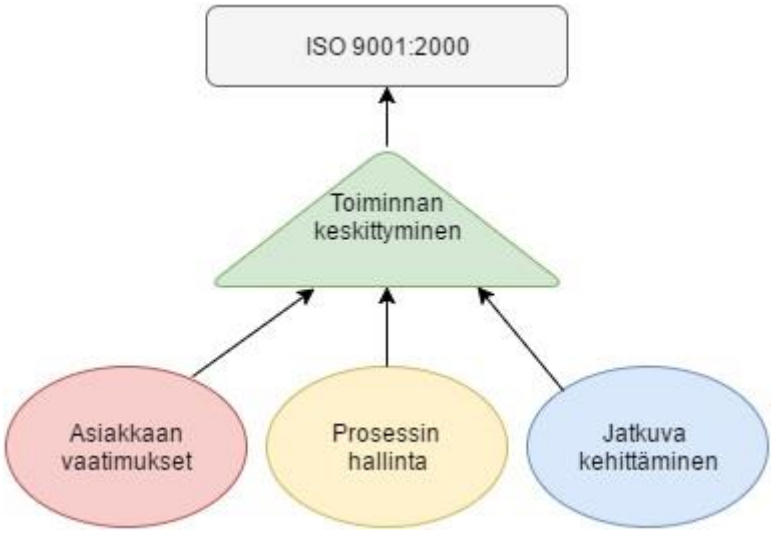
Yritys Y:llä ei ole toistaiseksi täysin valmiita ratkaisuja ja dokumentoituja prosesseja palvelimen käyttöönotolle ja palautukselle. Nykyisellään yksittäisen palvelimen käyttöönotto ja palautus manuaalisine työvaiheineen vie liikaa henkilötyöpäiviä.

1.4 Tavoitteet

Projektin tavoitteena on selvittää toimeksiantajan tarpeita varten tehokkaita palvelimen käyttöönotto- ja palautusmenetelmiä. Tarkoitusta varten opinnäytetyön tekijä selvittää menetelmiin soveltuvimmat työkalut, jotka myös testataan dokumentoidusti vaihe vaiheelta. Työkaluja hyödyntämällä tavoitteena on vähentää toimeksiantajalta palvelimen käyttöönottoon tai palautukseen kuuluvaa aikaa ja manuaalisen konfiguroinnin tarvetta.

Lisäksi testatuista työkaluista on tavoitteena toteuttaa analyysi, jonka sisällön pohjalta toimeksiantajan on mahdollista jatkokehittää omassa projektissaan varsinaisia käyttöönotto- ja palautusprosseja. Analyysi sisältää arvion testattujen työkalujen hyödynnettävyydestä sekä nostaa esille myös havaitut ongelmakohdat ja muut kehittämistarpeet käyttöönottoprosessin tehokkuuden ja eheyden varmistamiseksi.

Opinnäytetyöprojektin tavoitteita käsiteltäessä on lisäksi hyvä nostaa esille tavoiteltavan laadun määritelmä. ISO 9001:2000 -standardin mukaan toiminnan laadullisena kriteerinä on asiakkaiden tarpeiden sekä odotuksien täyttäminen ja ylittäminen mahdollisimman tehokkaasti ja kannattavasti. Lisäksi laadun määritelmä sisältää oleellisten asioiden tekemisen kerralla oikein ja jatkuvan parantamisen ideologian. (Tervonen 2006, 1.) Edellämainitut kriteerit pyritään täyttämään yleisluontoisesti tässä opinnäytetyöprojektissa.



Kuvio 1. ISO 9001:2000 -standardi keskittyy asiakkaiden vaatimusten täyttämiseen, prosessin laadulliseen hallintaan ja menetelmien jatkuvaan kehitykseen. (Persse 2006.)

Kehitettävien prosessien laadullisten kriteerien (toimivuus ja tehokkuus) lisäksi on tärkeää osata listata tavoitteet opinnäytetyön eri sidosryhmien näkökulmasta. Haaga-Helia ammattikorkeakoulun puolelta sidosryhmien tavoitteena on opinnäytetyön täyttävän heidän omat laadulliset kriteerinsä, niin työn sisällön kuin raportoinnin osalta. Tavoitteiden saavuttamisen mittarina käytetään opinnäytetyön arvostelua laadittujen kriteerien perusteella.

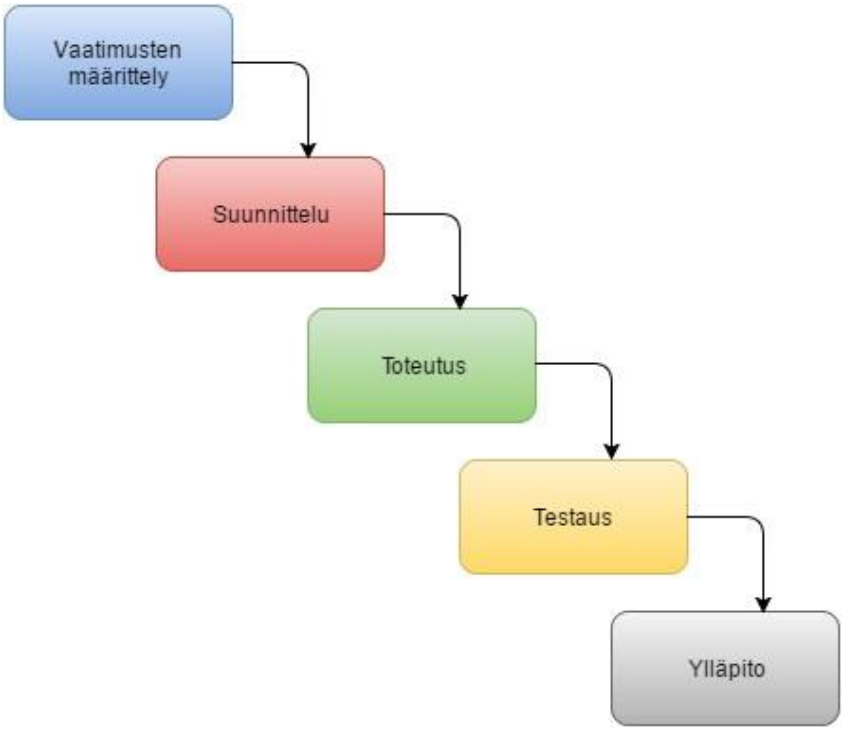
Vastaavasti toimeksiantajayritys Y:n tavoitteena on saada käyttökelpoisia ja tehokkaita palvelimen käyttöönotto- ja palautusmenetelmiä yrityksen omaan kehityshankkeeseen. Opinnäytetyössä on siis perimmäisenä tavoitteena tarjota soveltuvien työkalujen muodossa käyttökelpoisia vaihtoehtoja toimeksiantajalle.

Opinnäytetyön tekijän henkilökohtaisena oppimistavoitteena on saada kattava kuva yleisimmistä palvelimen käyttöönoton- ja palautuksen mahdollista työkaluista. Lisäksi tavoitteena on syventää IT-alan alustatöissä tarvittavaa ammattitaitoa sekä kehittää valmiuksia toteuttaa teknistä dokumentaatiota.

1.5 Toteutusmalli

Työn toteutusmalli mukailee suuripiirteisesti ohjelmistokehityksestä tuttua perinteistä vesiputousmallia. Mallin noudattamisen pystyy havaita erityisesti testiympäristön toteuttamisessa, mutta myös merkittävissä määrin testattavien palautusmenetelmien kohdalla.

Vesiputousmallissa tehdään esisuunnittelun perusteella vaatimusmäärittely, jota seuraa suunnittelu, toteutus ja testaus peräkkäisessä järjestyksessä (SelectBS 2016). Vesiputousmallin yleisperiaatteet soveltuu varsin hyvin tämän kaltaisen työn toteutukseen projektin yksinkertaisuuden, pienuuden ja helpon hallittavuuden ansiosta.



Kuvio 2. Vesiputousmallin vaiheet. (Hughey 2009.)

Ketteriä kehitysmenetelmiä ei tässä pikkuprojektissa suoranaisesti käytetä, mutta jatkokehitykseen sopisi oletettavasti hyvin Scrumin kaltaisen projektinhallinnan viitekehyksen käyttö. Ketteriä menetelmiä hyödyntävä esimerkkitapaus voisi tällöin olla esimerkiksi uuden työkalun implementointi vanhaan ympäristöön, jolloin suunnittelu ja toteutus olisi paloiteltu Scrumin kehitysjonon tehtävälistalle. (Scrumguides 2016.)

Muita työn toteutuksessa hyödynnettäviä menetelmiä voisivat olla BPMN- ja UML-kuvaustekniikoiden käyttö, joka ilmenisi graafisina prosessi- ja käyttötapauskaavioina.

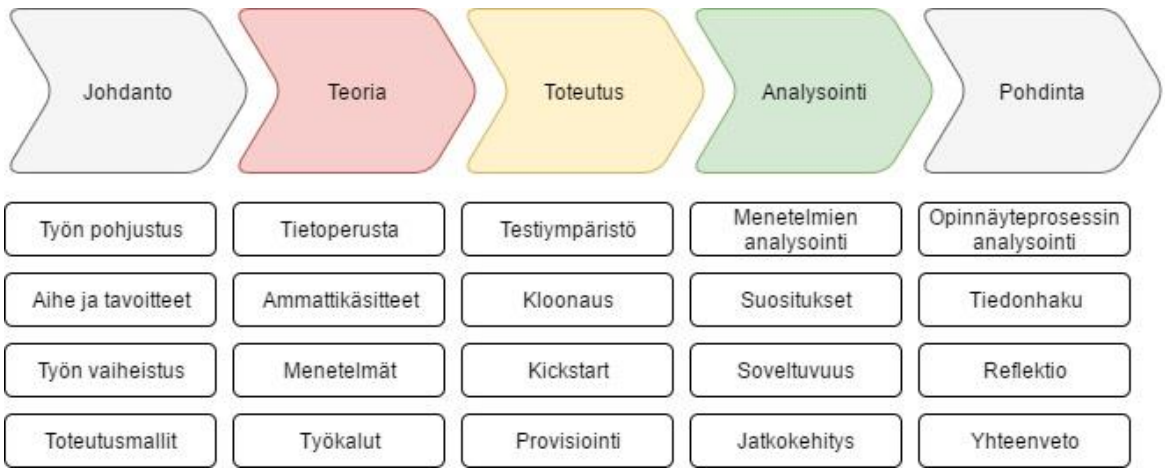
Kaavioiden avulla selkeytettäisiin lukijalle esimerkiksi prosessien suoritusjärjestystä. Mainituista kuvaustekniikoista BPMN (Business Process Model and Notation) on enemmän prosessiorientoitunut menetelmä ja UML (Unified Modelling Language) vastaavasti objektiorientoitunut. (BPMN 2016.)

1.6 Vaiheistus

Työ on vaiheistettu etenevän tarkoituksella suoraviivaisesti. Johdannon jälkeen tulee teoriaosuudet, joissa esitellään pääasiallisesti työssä käytettäviä menetelmiä ja työkaluja. Tietoperustan jälkeen suunnitellaan ja toteutetaan menetelmien testausta varten tarvittava virtuaalinen testiympäristö sekä varmistetaan joukolla testejä ympäristön toimivan vaaditulla tavalla.

Seuraavana vuorossa on opinnäytetyön ydinosuus eli erinäisten käyttöönotto- ja palautusmenetelmien dokumentoitu testaus valituilla työkaluilla. Suoritettaviin testeihin lukeutuvat palvelimen levykuvan palautus Clonezillalla, uuden palvelimen käyttöönotto Red Hat Linux Enterprisen kickstart-konfiguraatiotiedostolla sekä palvelinjoukon asetuksien ja asennuksien provisiointi Ansible-automatisointityökalulla.

Käytännön testauksen valmistuttua analysoidaan työkalujen hyviä ja huonoja puolia sekä arvioidaan näiden soveltuvuutta toimeksiantajan käyttötarpeisiin. Analysoinnin jälkeen käydään lisäksi läpi jatkokehitysideoita pyrkimyksenä lisätä mahdollisuuksia työkalujen hyödyntämiseen tulevaisuudessa. Varsinaisen työn suorituksen jälkeen on vielä projektin yhteenveto, jossa pohditaan opinnäytetyöprosessin sujuvuutta ja omaa ammatillista oppimista.



Kuvio 3. Työn perinteiseen malliin perustuva rakenne kuvattu lineaarisena nuolikuviona.

2 Tietoperusta

Kappaleessa käydään läpi olennaisimmat ammattikäsitteet, esitellään työssä käytettäviä tekniikoita sekä palvelimen käyttöönotto- ja palautusmenetelmiä.

2.1 Ammattikäsitteet

Laitteistot ja komponentit

Palvelin - Muita laitteita LAN:n tai WAN:n kautta palveleva tietokone. (TechTerms 2014.)
CPU - Core Processing Unit, tietokoneen prosessori. (PCMag 2016a.)
RAM - Random Access Memory eli tietokoneen työmuisti. (BBC 2016.)

Käyttöjärjestelmät

Linux - Unix:n perustuvan Open source - käyttöjärjestelmän ydin. (The Balance 2016.)
RHEL - Red Hat Enterprise Linux, kaupallinen Linux-jakelu. (CBR 2016.)
CentOS - RHEL:n lähdekoodiin perustuva Linux-käyttöjärjestelmä. (CentOS 2017a.)
Kickstart - RHEL:n luoma Linux käyttöjärjestelmän asennusmetodi. (Pykickstart 2016.)
GRUB - GRand Unified Bootloader, Linuxin oletus bootloader. (Slashroot 2013.)

Tietoliikenne

LAN - Local Area Network eli lähiverkko. (Indiana University 2013.)
WAN - Wide Area Network, laaja tiedonsiirtoverkko. (Indiana University 2013.)
IP-osoite - Internet Protocol Address, verkkosovittimen looginen osoite. (PCMag 2016b.)
Reititin - Verkkoliikennettä reitittävä laite. (Hitachi 2016.)
Palomuuri - Verkkoliikennettä määritellysti suodattava ohjelma tai laite. (Cisco 2016.)
MAC-osoite - Verkkosovittimen uniikki 48-bittinen fyysinen osoite. (Lifewire 2016.)
SSH - Secure Shell, salausta käyttävä tietoliikenneprotokolla. (Cisco 2009.)
RDP - Remote Desktop Protocol, etäyhteys Windowsiin. (Microsoft 2016.)
FTP - File transfer protocol, tiedonsiirtoprotokolla. (Indiana University 2016.)
Telnet - Salaamaton tiedonsiirtoprotokolla. (Study-CCNA 2016.)
X11 - Sovelluksen ajaminen visuaalisesti Linux-palvelimella. (Uni. of North Carolina 2012.)
MOSH - Mobile Shell, salausta käyttävä tiedonsiirtoprotokolla. (Linoxide 2014.)
VNC - Virtual Network Computing, etäyhteys graafiseen käyttöliittymään. (Lifewire 2015.)

Sovellukset

Clonezilla - Työkalu levyn kloonaamiseen ja palauttamiseen. (Clonezilla 2016.)
Ansible - Automatisointityökalu järjestelmän konfiguroimiseen. (Ansible 2017a.)

VirtualBox - Oraclen kehittämä virtualisointityökalu. (VirtualBox 2016a.)
VMware Workstation Player - Virtualisointityökalu työpöytäkäyttöön. (VMware 2016b.)

Muut
Levykuva - Disk image, kopioidun datan sisältävä tiedosto. (Technopedia 2016.)
UML - The Unified Modeling Language, graafinen mallinnuskieli. (Defit 2016.)
BPMN - Business Process Model and Notation, graafinen mallinnuskieli. (BPMN 2016.)
Repository - Linuxin pakettivarasto, josta ohjelmien haku verkon välityksellä. (IBM 2015.)
Hypervisor - Mahdollistaa resurssien jakamisen virtuaalikoneille. (How-to Geek 2011.)

2.2 Virtualisointi

Virtualisointi on lavea käsite, jolla voidaan tarkoittaa laajaa kirjoa eri asioita informaatioteknologian saralla. Yleinen kuvaus virtualisoinnista on kuitenkin, että suoritetaan jokin asia sovelluspohjaisesti virtuaalisella esitystavalla hyödyntämällä kokonaan tai osittain taustalla olevia ”todellisia” fyysisiä resursseja. (Gartner 2016.)

Lähes mitä tahansa on nykyaikana mahdollista virtualisoida: oli kyse sitten palvelimista, sovelluksista, tallennusratkaisuista tai vaikka tietoliikenneverkoista. Virtualisoinnin hyötyihin lukeutuvat mm. parantunut resurssien hyödyntäminen provisioinnin avulla ja järjestelmien skaalautuvuus. Lisäksi palvelinympäristön ylläpito voi helpottua huomattavasti, kun ei ole sidottu pelkästään kankeaan fyysiseen ratkaisuun. (VMware 2016a.)

Tässä opinnäytetyössä tullaan käyttämään tyypin kaksi virtualisointityökalua (Zdnet 2011.) luomalla isäntätietokoneena (Host OS) toimivan käyttöjärjestelmän päälle virtuaalinen testiympäristö, joka käsittää joukon ohjelmallisesti luotuja virtuaalikoneita (VM). Virtuaalikoneiden välille konfiguroidaan lisäksi pienehkö oma lähiverkko eli LAN. (Indiana University 2013.)

Virtuaalikone eli VM (Virtual Machine) on tarkemmin selitettynä työkalun avulla luotu sovelluspohjainen, virtualisoitu tietokone, joka hyödyntää käytettävän host-tietokoneen fyysisiä resursseja kaikilta osin eli mm. prosessoria (CPU), keskusmuistia (RAM) ja kiintolevyn tallennustilaa. (How-To Geek 2014.)

Virtuaalikone käyttöjärjestelmineen toimii käytännössä täsmälleen samalla tavalla kuin ”oikea fyysinen tietokone”, mutta on täysin riippuvainen host-tietokoneen resursseista, joista haluttu osa allokoidaan VM:n käyttöön. Virtuaalikoneita voi pitää samanaikaisesti

useita päällä ja käytännössä host-tietokoneen resurssit määrittävät lopulta kuinka monta eri virtuaalikonetta pystyy käyttämään vakaasti rinnakkain. (VMware 2016a.)

2.3 Levyn kloonaus

Levyn kloonauksella tarkoitetaan kokonaista tai osittaista kovalevyn kopioimista esimerkiksi suoraan toiselle kovalevylle tai optiselle tallennusmedialle image-formaattiin. (PCMag 2016c.) Levyn kloonauksen voi suorittaa Clonezillan tapaisilla kloonausohjelmilla lokaalisti tai verkon välityksellä. Lisäksi myös perinteisellä dd-komennolla onnistuu levyn kloonaus. (How-To Geek 2010.)

Kokonaisen levyn klooni sisältää kirjaimellisesti kaiken mitä myös alkuperäinen lähde eli tiedot levyosioista, käynnistyssektorin (boot-sector), tiedostojärjestelmän sekä käyttöjärjestelmä- ja sovellusasennukset. Syynä levyn kloonaukselle voi olla esimerkiksi uuden kovalevyn hankinta, varmuuskopion luominen levyrikon varalle tai kopion provisioiminen uusille tietokoneille. (PCWorld 2014.)

2.4 Palvelimen provisiointi

Palvelimen provisioinnilla tarkoitetaan karkeasti sanottuna jonkin toimenpiteen suorittamista automatisoidusti tätä tarkoitusta varten suunnitellun sovelluksen avulla. Kyse voi olla esimerkiksi fyysisen raudan resurssien jakamisesta ohjelmallisesti useammalle virtuaalikoneelle tai käyttöjärjestelmän asetuksien samanaikaisesta provisioinnista useammalle koneelle tarkoitusta varten räätälöidyn sovelluksen avulla.

Provisioinnin voi siis jakaa raudan (VMware, Openstack) ja käyttöjärjestelmän (The Foreman, Kickstart ja Cobbler) provisiointiin sekä lisäksi palvelimen konfiguraationhallintaan (Ansible, Puppet ja Saltstack). Tässä opinnäytetyössä testaan kickstart-tiedoston käyttöä, mutta painopiste on kuitenkin hieman enemmän konfiguraationhallinnan puolella Ansible-sovelluksen testaamisen myötä. (Cyberciti 2015.)

Palvelimen konfiguraationhallintaohjelmalla voidaan ajaa samanaikaisesti yhdelle tai useammalle palvelimelle tarvittavat käyttöjärjestelmän asetukset ja sovellusasennukset käyttötarkoitusta varten suunnitellun ohjelmiston avulla Yksinkertaistettuna siis palvelin saatetaan käyttövalmiiksi haluttua palvelua varten automatisoidusti. (XebiaLabs 2010.)

Kyseiset ohjelmat tehokkaasti optimoituna mahdollistavat palvelimien konfigurointiin tarvittavan ajan ja vaivan säästön järjestelmän ylläpitäjältä. Muihin etuihin kuuluvat mm.

yksinkertaistettu vikatilanteista palautuminen ja palvelimien hallittu asetuksien konfiguroiminen identtiseksi kaikille palvelimille. (DigitalOcean 2016a.)

2.5 Kickstart-tiedosto

Kickstartilla tarkoitetaan Red Hat Enterprise Linuxin kehittämää käyttöjärjestelmän asennusmetodia, jossa OS:n asennus suoritetaan ennalta määritellyn kickstart -konfiguraatiotiedoston avulla. Kickstart-tiedoston avulla käyttöjärjestelmän asennus hoituu siis nopeammin ja vaivattomammin, kun normaalit manuaaliset asennusvaiheet on ennalta automatisoitu. (Red Hat 2017a)

Kickstart on itsessään yksinkertainen .cfg-päätteinen tiedosto, joka tallentuu OS:n asennuksen jälkeen Linuxin *"/root"*-hakemistopolkuun nimellä anaconda-ks.cfg. Kickstart-tiedoston voi luoda joko itse manuaalisesti tavallisella tekstieditorilla tai ohjatusti graafisella kickstart configurator -sovelluksella, jolloin ei tarvitse osata itse varsinaista Kickstartin syntaksia. (CentOS 2017b.)

Vaihtoehtoisesti voi myös kopioida aiemmin asennetun OS:n pohjalta generoidun kickstart-tiedoston ja muokata kyseiseen tiedostoon tarvittavat ominaisuudet, pakettiasennukset ja skriptit. Tämä tapa on erinomainen varsinkin opeteltaessa Kickstart-tiedoston syntaksia ja yksityiskohtaisempaa konfigurointia.

Kickstart-tiedosto jakautuu kahteen pakolliseen osaan: yleiseen komento-osioon, johon sisällytetään käyttöjärjestelmän perusasetukset ja pakettiosioon, johon merkitään asennettavat sovelluspaketit. Tarvittaessa Kickstartiin voi sisällyttää lisäksi vielä pre-script osion, johon voi määritellä eri kielillä (esim. Pythonilla) esiasennettavia skriptejä ja post-script osioon, johon voi lisätä vastaavasti käyttöjärjestelmän asennuksen valmistumisen jälkeen suoritettavia skriptejä. (CentOS 2017b.)

Kickstart-tiedoston asennus on mahdollista tehdä lokaalin CD-levyn tai USB-muistin avulla sekä verkon välityksellä NFS-, FTP- ja HTTP-protokollien avulla. Erikseen tarkoitusta varten räätälöidyn PXE-palvelimen avulla on lisäksi mahdollista ajaa verkon välityksellä kickstart-asennus samanaikaisesti useammalle palvelimelle tai työasemalle.
(Red Hat 2017b.)

3 Työkalut

Kappaleessa esitellään virtualisointiin sekä palvelimen käyttöönottoon- ja palautukseen soveltuvia työkaluja. Esiteltävät työkalut on valittu kattavien esiselvityksien perusteella ja suurinta osaa työkaluista tullaan käyttämään myös toteutusvaiheessa.

Lisäksi kappaleessa esitellään testiympäristöä varten valittu CentOS 7 -käyttöjärjestelmä. Käyttöjärjestelmävalinnan perusteena oli CentOS:n ominaisuuksien erinomainen soveltuvuus tämän projektin vaatimuksiin sekä henkilökohtainen halu syventää CentOS-osaamista.

3.1 CentOS 7 -käyttöjärjestelmä

CentOS on yhteisöpohjaisesti kehitettävä ilmainen Linux-käyttöjärjestelmä, joka perustuu kaupallisen Red Hat Enterprise Linuxin (RHEL:n) avoimeen lähdekoodiin ja CentOS on lisäksi funktionaalisesti täysin yhteensopiva RHEL:n kanssa (CentOS 2017a). CentOS:n ensimmäinen versio julkaisiin toukokuussa 2004 ja tuorein versio on 7.3, joka julkaistiin 12.12.2016. (Softpedia 2016.)

CentOS:n tiedostopaketit ovat RHEL:n tapaisesti päätteiltään .rpm -muotoisia ja paketinhallintaohjelmana käytetään komentorivin kautta toimivaa Yum-ohjelmaa (CentOS 2017a). Oletuspalomuurina käyttöjärjestelmässä on dynaamisesti hallinnoitava Firewalld, joka käyttää verkon ja palomuurin vyöhykkeitä (”zones”) toisin kuin aiemmin CentOS:n oletuspalomuurina ollut perinteinen Iptables. (Unixmen 2015.)

Testiympäristössä tullaan käyttämään CentOS 7:n minimal-versiota, joka on tarkoitettu palvelinkäyttöön ja siitä on karsittu lukuisia täysversiosta löytyviä ominaisuuksia. Ominaisuuksien karsimisen syynä on pyrkimys tehdä käyttöjärjestelmästä vakaampi ja tietoturvallisempi. Merkittävin karsittu ominaisuus lienee graafinen käyttöliittymä, sillä minimal-versiota hallitaan ainoastaan komentorivin välityksellä. (Superuser 2015.)

3.2 Oracle VirtualBox

VirtualBox on Oraclen ilmainen open-source pohjainen virtualisointityökalu, jonka uusin versio on 5.1.10 (VirtualBox 2016b). VirtualBox soveltuu erityisesti labratestaukseen ja sen käyttöönotto on helppoa myös aloittelijalle. VirtualBoxin kehitti vuonna 2007 alkujaan Innotek, jonka Sun Microsystems osti vuonna 2008 (E-Commerce times 2008). VirtualBox siirtyi lopulta Oraclelle sen ostaessa Sunin vuonna 2010 (Cnet 2010.) ja tuotteen

viralliseksi nimeksi tuli ”Oracle VM VirtualBox”. (Oracle 2010.)

Oraclen VirtualBoxilla ajetaan isäntäkoneen käyttöjärjestelmän (”host OS”) päällä virtuaalikoneita (VM), joiden sisällä ajetaan haluttua käyttöjärjestelmää (”Guest OS”). Kyseessä on siis työpöytäkäyttöön soveltuva tyypin kaksi hypervisor, joka toimii Host OS:n päällä pelkän fyysisen raudan sijaan. Sovelluksella voi ajaa montaa virtuaalikonetta samanaikaisesti ja VM:t käyttävät suoraan isäntäkoneen fyysisiä resursseja toimiakseen. (VirtualBox 2016a.)

VirtualBoxin minimivaatimuksiin lukeutuu 512MB keskusmuistia, mutta varsinaisen Guest OS:n käyttöön on syytä varata sen verran keskusmuistia, kun käytettävän käyttöjärjestelmän oma todellinen suositus on. Kiintolevytilan osalta VirtualBoxilla ei ole merkittäviä vaatimuksia. (VirtualBox 2016c.)

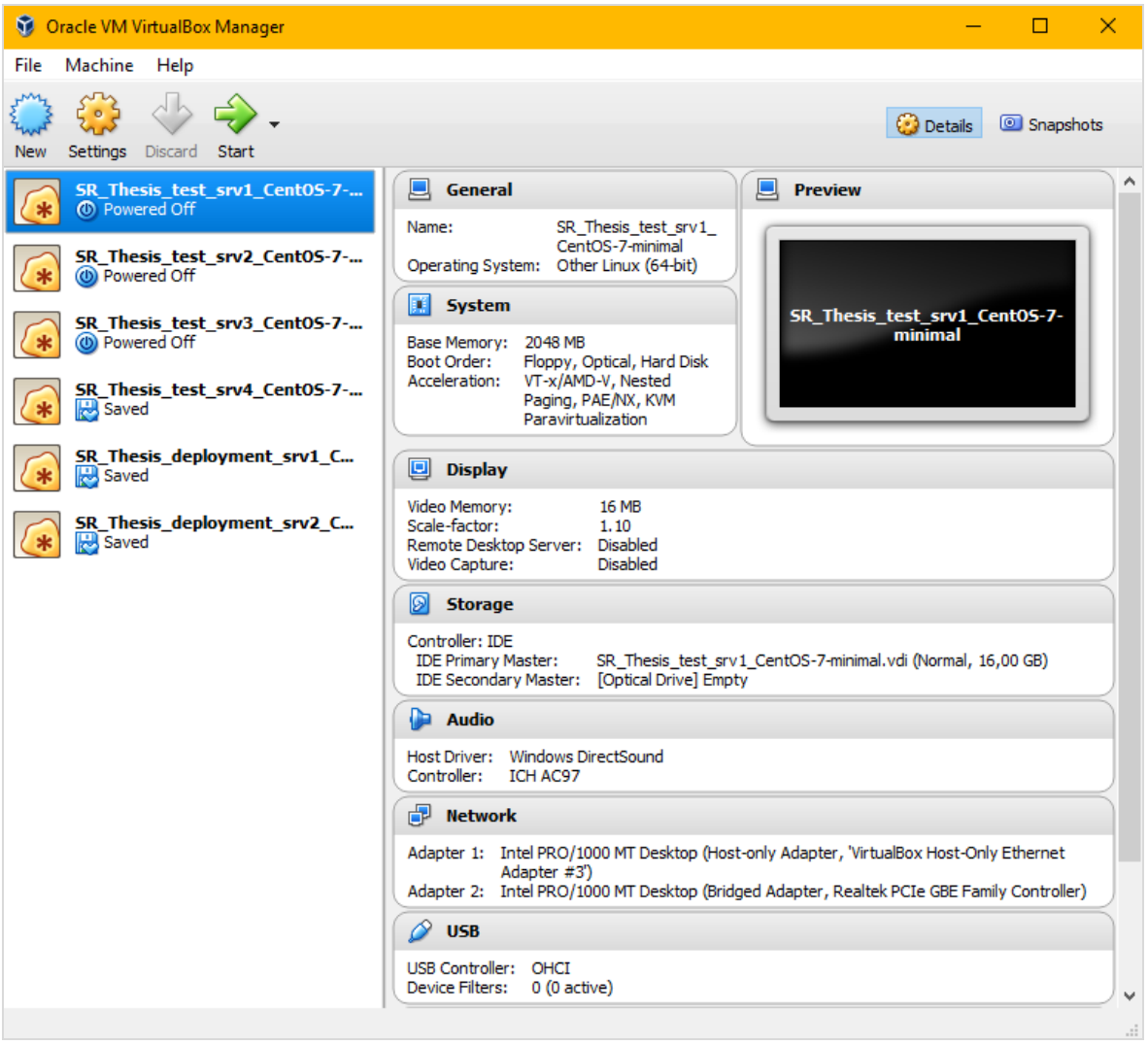
VirtualBoxin voi asentaa asentaa kaikkiin x86-pohjaisiin Intel- ja AMD-tietokoneisiin, joissa on käytössä tuettu Windows-, Mac-, Linux- tai Solaris-käyttöjärjestelmä ja virtuaalikoneet itsessään voivat käyttää vierailijakäyttöjärjestelmänä (Guest OS) käytännössä mitä tahansa edellä mainittua käyttöjärjestelmää paitsi Applen OS X -käyttöjärjestelmää, jonka ajaminen virtualisoidusti muiden käyttöjärjestelmien päällä on kielletty. (VirtualBox 2016d.)

Olisi siis esimerkiksi mahdollista ajaa samanaikaisesti Windows 10 -isäntäkoneella neljää eri virtuaalikonetta, joista kahdessa olisi Linux Ubuntu 16 -käyttöjärjestelmä ja kahdessa Windows 7 -käyttöjärjestelmä. Samalla myös host-käyttöjärjestelmä Windows 10 olisi edelleen käytettävissä normaaliin tapaan.

Virtuaalikonetta luodessa määritetään VM:lle varattava keskusmuistin ja kiintolevytilan määrä. Luodut virtuaalikoneet tallennetaan host-koneen kiintolevylle useimmiten VDI-levykuvana (Virtual Disk Image), jolle varataan tarvittavan verran tilaa isäntäkoneen kiintolevyltä. Virtualbox tukee VDI-formaatin lisäksi VMwaren VMDK-, Microsoftin VHD- ja Parallelsin HDD-formaatteja. Image-tiedostoon tallennettua virtuaalikonetta voi käyttää myös toisen host-tietokoneen VirtualBoxilla. (VirtualBox 2016a.)

Virtuaalikoneesta on mahdollisuus ottaa ”snapshot”, joka luo virtuaalikoneen sen hetkisestä tilasta palautuspisteen erilliseen tiedostoon. Snapshot mahdollistaa siis virtuaalikoneen palauttamiseen täsmälleen sen hetkiseen tilaan (sovellusasennukset ja järjestelmäasetukset), kuin VM oli snapshotin ottohetkellä. Ominaisuus on erittäin hyödyllinen tilanteissa, jossa virtuaalikone hajoaa tai korruptoituu esimerkiksi päivityksen

asentamisen vuoksi ja on tarpeen palauttaa käyttöjärjestelmä toimivaksi. (VirtualBox 2016a.)



Kuva 1. Oracle VirtualBoxin yleisnäkymä. Vasemmassa reunassa näkyvät luodut virtuaalipalvelimet ja oikeassa reunassa valittuna olevan palvelimen tekniset tiedot. Uuden virtuaalipalvelimen luonti onnistuu ylhäältä ”New”-painikeesta ja olemassa olevan palvelimen käynnistäminen tehdään valitsemalla palvelin vasemmanpuoleisesta listasta ja painamalla yläreunan ”Start”-painiketta.

3.3 VMware Workstation Player

VMware Workstation Player on VMwaren kehittämä kakkostyypin hypervisor eli sovellus työpöytävirtualisointiin. Ohjelman uusin versio on 12.5.2 ja sen käyttö on ilmaista yksityiskäyttöön, mutta yrityskäyttö vaatii maksullisen lisenssin. Vaativampaan käyttöön on lisäksi olemassa aina maksullinen ja enemmän ominaisuuksia sisältävä VMware Workstation Pro -versio. (VMware 2016b.)

Sovellus tukee host-käyttöjärjestelmäksi yleisempiä 64-bittisiä Windows- ja Linux-käyttöjärjestelmiä. Minimivaatimuksena isäntäkoneelle on 64-bittinen x86 Intel Core 2 Duo -prosessori tai AMD Athlon 64 FX -tuplaydinprosessori. Keskusmuistin eli RAM:n minimivaatimus on 2GB, mutta VMwaren suositus on 4GB. (VMware 2016c, 13-14.)

Workstation Player mahdollistaa 32- ja 64-bittisten Windows- sekä Linux-virtuaalikoneiden (Guest OS) luomisen yksinkertaisten ”step-by-step” vaiheiden avulla. Tuettujen Guest-käyttöjärjestelmien joukkoon lukeutuvat yleisempien Linux-distrojen lisäksi myös mm. Fedora ja Mint sekä Unix-pohjainen Solaris. (VMware 2016b).

Workstation Playerin perusversio ei tue Oraclen VirtualBoxin tapaisesti useamman VM:n käyttämistä samanaikaisesti eikä myöskään snapshottien ottamista, mutta sen sijaan maksullisesta Workstation Player Pro:sta löytyy molemmat ominaisuudet. Playerin Pro -versiota ei kuitenkaan käsitellä tarkemmin tässä opinnäytetyössä. (VMware 2016c, 96.)

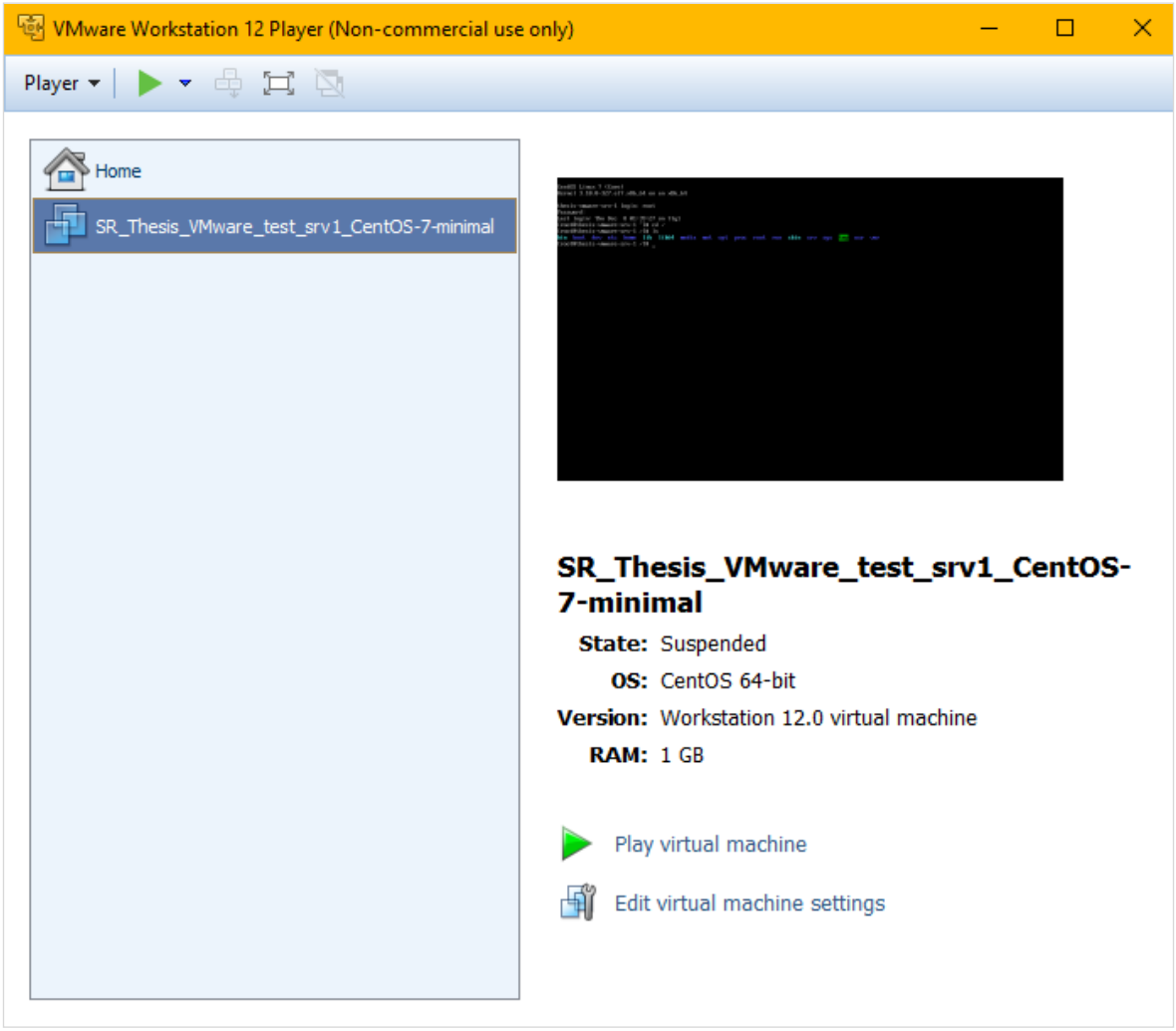
Playerillä virtuaalikonetta luodessa käyttöjärjestelmä on mahdollista asentaa fyysiseltä levyltä tai lokaalilla .iso -päätteisellä image-tiedostolla. Virtuaalikonetta luotaessa on lisäksi valittava imagen koko ja määritettävä haluaako VM:n tiedostojen jakautuvan useaan erilliseen osaan vai kaiken tallentuvan yhteen isoon image-tiedostoon. Luodut virtuaalikoneet tallennetaan VMwaren omaan VMDK-formaattiin.

Virtuaalikoneen asentamisen jälkeen on vielä mahdollista asentaa VM:n käytettävyyttä optimoivat työkalut eli ”VMware toolsit” juuri luodun koneen käyttöön. Testaamisen perusteella Linux-käyttöjärjestelmille tarkoitettujen VMware toolsien viimeisin versio on 10.0.10.

Testimielessä Workstation Playerillä suoritettu virtuaalikoneen asennus sujui yksinkertaisesti käyttäen yllä mainitun VM:n luontiprosessin mukaisesti ”.iso”-päätteistä image-tiedostoa CentOS 7 -minimal käyttöjärjestelmän asennukseen. Workstation Player osottautui pitkälti Oracle VirtualBoxin kaltaiseksi varsin minimalistisen käyttöliittymän omaavaksi ohjelmaksi, jonka perustoiminnot oppi hetkessä.

Ainoa Playerin käytössä alkuun hivenen ärsyttänyt ominaisuus oli vaatimus painaa CTRL+ALT saadakseen hiiren kursorin takaisin host-tietokoneen käyttöön siirtyessä VM:n käyttämisestä takaisin host-tietokoneen puolelle eikä ”ominaisuus” korjaantunut VMware työkalujen asentamisella tai ”%APPDATA%\VMware” -kansion asetuksia muuttamalla.

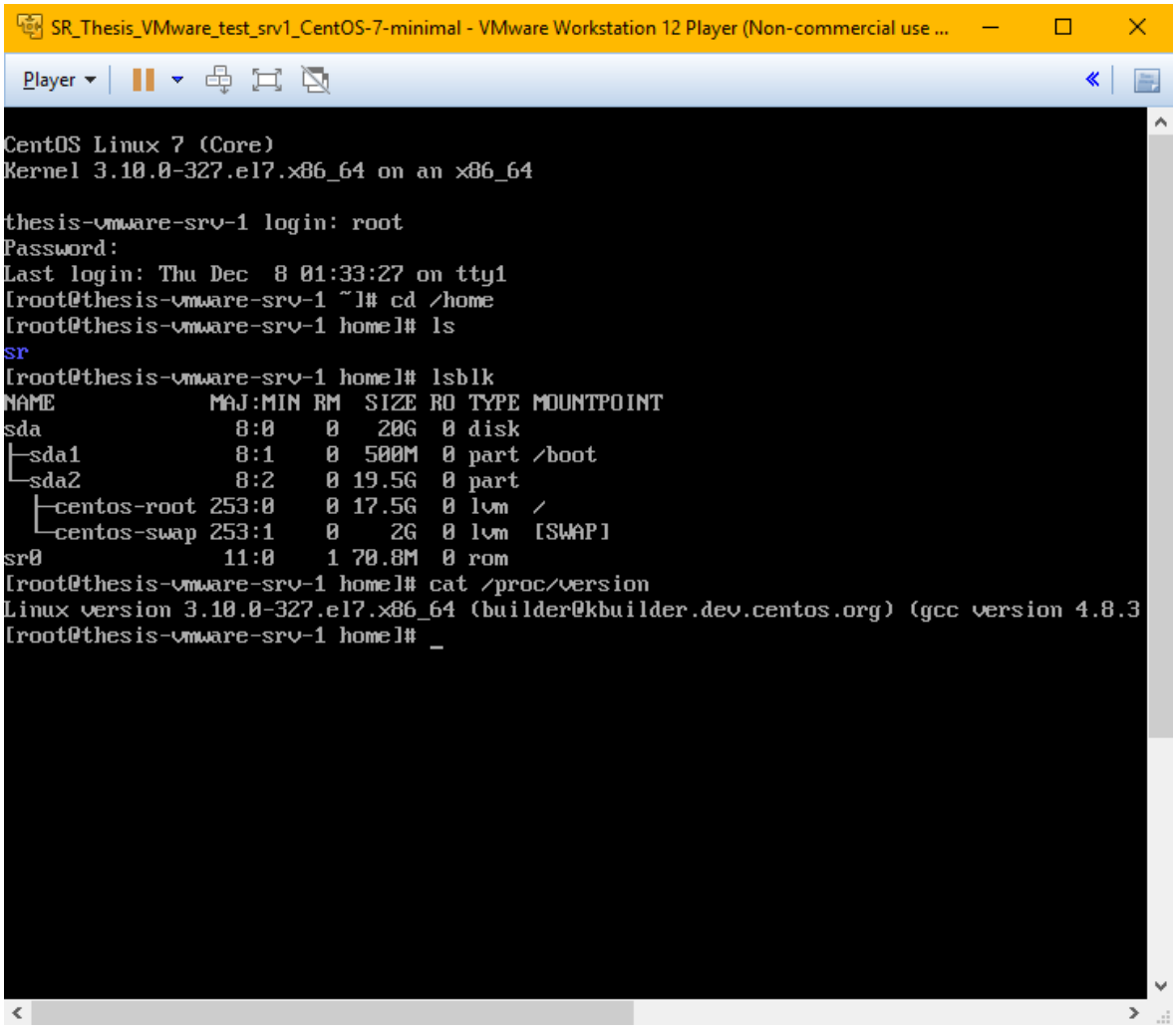
Muutoin Workstation Playerin kautta luodun virtuaalikoneen testaaminen sujui kaikinpuolin mainiosti VM:n toimiessa luotettavasti koko ajan.



Kuva 2. VMware Workstation Playerin käyttöliittymä. Vasemmalla puolella listataan virtuaalipalvelimet ja oikealla puolella on listalta valitun, käynnissä olevan testipalvelimen livenäkymä ja tekniset tiedot.

Mikäli vasemmalta on valittuna palvelimen sijaan ”Home” - painike, niin oikeaan reunaan ilmestyy pikanäppäimet, joista voi vastaavasti luoda täysin uuden VM:n tai avata tietokoneen lokaalilta kiintolevyltä olemassa olevan virtuaalikoneen. Lisäksi oikeasta reunasta löytyy tällöin myös päivitysmahdollisuus maksulliseen Pro-versioon ja linkki kattavaan käyttöoppaaseen.

Yläreunan painikkeista ”Player”-vetovalikossa on itse sovelluksen ja luotujen virtuaalikoneiden asetuksia, vihreästä painikkeesta voi käynnistää listalta valitun virtuaalikoneen ja neljän nuolen ruutuikonista saa virtuaalikoneen koko ruudulle.

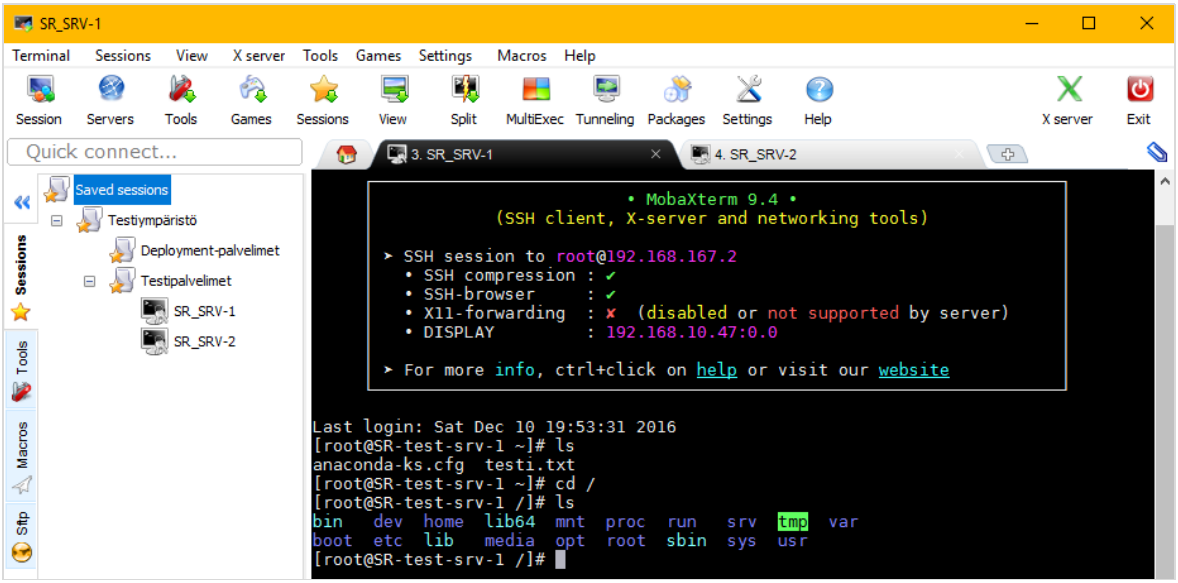


Kuva 3. VMware Workstation Playerillä luotu virtuaalikone 64-bittisellä CentOS 7 -käyttöjärjestelmällä. Koneelle määritettiin 1GB keskusmuistia ja 20GB kiintolevytilaa.

3.4 MobaXterm

MobaXterm on Mobatekin kehittämä palvelin- ja SSH-asiakasohjelma, jonka uusin versio on 9.4. Sovellus tukee useita eri yhteysprotokollia kuten SSH:ta, RDP:tä, FTP:tä, X11:a, MOSH:a, Telnetiä ja VNC:tä. Sovelluksesta löytyy kahta eri versiota: ilmainen ”home edition”-versio, jossa ominaisuuksien käyttöä on hieman rajoitettu ja maksullinen ”professional edition”, josta löytyy perusversion ominaisuudet ilman käyttörajoituksia ja lisäksi täysin uniikkeja ominaisuuksia. (Mobatek 2016a.)

Sovelluksen käyttöliittymä sisältää samanaikaisten etäyhteyksien hallintaa helpottavat välilehdet, jotka muistuttavat ulkoasultaan pitkälti Googlen Chromen vastaavia. Ohjelmalla on myös mahdollista tallentaa käytettävien terminaaliyhteyksien tiedot suosikkeihin, jotka voi halutessaan järjestellä myös kansioihin. Kokemuksieni perusteella kyseinen ominaisuus selkeyttää huomattavasti palvelinympäristön etähallintaa.



Kuva 4. MobaXtermin perusnäkymä, kun palvelimeen otettu SSH-yhteys.

”Multi-exec mode” -ominaisuuden avulla sen sijaan voi käskyttää samanaikaisesti useita palvelimia. Multi-exec modea käyttäessä ruudukoissa näkyvät auki olevat etäyhteydet ja ominaisuus mahdollistaa kaikkien ruudukon palvelimien hallitsemisen samanaikaisesti. Käytännön esimerkki ominaisuudesta on komennon kirjoittaminen terminaaliin kerran, joka suoritetaan välittömästi kaikille ruudukon palvelimille. Tämä nopeuttaa huomattavasti identtisten komentojen syöttämistä laajalle palvelinjoukolle. (Mobatek 2016b.)

Muita mainittavia ominaisuuksia sovelluksessa ovat SSH-tunnelin luominen, täysi X-server tuki, lisäosat eli pluginit ja mahdollisuus kannettavaan ”portable”-versioon, jota ei tarvitse asentaa perinteisellä tavalla pysyvästi tietokoneelle (Mobatek 2016b). Tulen käyttämään MobaXtermiä testiympäristön palvelimien hallintaan, kun palvelimet on ensin luotu virtuaaliympäristöön ja lähiverkko konfiguroitu näiden palvelimien välille.

3.5 Clonezilla

Clonezilla on NCHC Free Software Labsin kehittämä avoimen lähdekoodin työkalu, joka soveltuu kovalevyn kloonaamiseen ja palauttamiseen. Clonezilla perustuu DRBL:n, Partimageen, Ntfscloneen ja Udpcastiin sekä tukee kaikkia yleisempiä käyttöjärjestelmiä ja suurta määrää eri tiedostotyyppejä. Expert-moden lisäksi valittavana on beginner-mode, jonka käytttö on yksinkertaisempaa sisältäen samalla vähemmän ominaisuuksia.

Clonezilla mahdollistaa esimerkiksi kokonaisen levyn tai levyosion varmuuskopioinnin imageksi (.img) tai muuhun haluttuun tiedostomuotoon. Lisäksi myös levyn blokkien

kopioiminen suoraan kovalevyltä toiselle on mahdollista. Tuettujen tiedostomuotojen kohdalla sovellus kopioi vain käytössä olevat blokit lähdelevyltä.

Palvelimelle suoritettavan sovellusasennuksen lisäksi myös live-käyttö on Clonezillassa tuettu, niin paikallisesti kuin Linuxille tarkoitetussa ”server editionissa” esim. CD:n tai USB-muistin kautta. Paikallisen version käyttö onnistuu luonnollisesti vain yhdellä palvelimella kerrallaan siinä missä server editionilla on mahdollista ajaa halutut Clonezillan taskit hallintapalvelimelta yli 40 palvelimelle samanaikaisesti. (Linuxlinks 2016.)

Clonezilla on monipuolisuudestaan huolimatta käytännössä puhdasverinen kloonityökalu, minkä vuoksi se ei itsessään osaa pienentää tai suurentaa varmuuskopion kokoa. Tämän seikan vuoksi kohdelevyn tai muun tallennusmedian tulee olla samansuuruinen tai suurempi kuin kopioitavan levyn koko. (Lifehacker 2012.)

3.6 Ansible

Ansible on palvelinten konfiguraationhallintaan kehitetty työkalu, jolla voidaan provisioida palvelinjoukolle samanaikaisesti esimerkiksi käyttöjärjestelmän konfiguraatiot ja sovellusasennukset. Komennot hallintapalvelimelta kohdepalvelimille ajetaan SSH-yhteydellä ”push”-menetelmällä eikä konfiguroitaviin palvelimiin tarvitse asentaa erillisiä agentteja, Ansible on siis ”agentiton” työkalu. (Ansible 2017a.)

Ansiblea käytetään tavallisesti komentorivin kautta, mutta on olemassa myös Ansible Tower, joka tuo käyttöön graafisen käyttöliittymän. Ansiblen komennot kohdepalvelimille voidaan ajaa ad-hoc tyyliin, mutta suositeltavampaa on käyttää playbookkeja, joiden avulla palvelinjoukon asetusten kuntoon saattaminen on hallitumpaa. Playbookit perustuvat Ansiblen omaan YAML-kieleen (”YAML Ain’t Markup Language”).
(Ansible 2017b.)

Yksittäisessä playbookissa suoritetaan aina yksi tai useampi ”play”, jotka erotetaan toisistaan kolmella väliviivalla ja yksittäinen play voi sisältää useita eri rooleja, jotka kohdennetaan playn ”hosts”-määrityksellä halutulle palvelinryhmälle. Roolit sisältävät yhden tai useamman käytännön tehtävän eli taskin, joten tarvittaessa yhdelläkin roolilla on mahdollista suorittaa useita peräkkäisiä toimenpiteitä palvelimille.

Normaalin taskin yhteyteen voi halutessaan määrittää ”handler taskin”, joka ajetaan kertaalleen kohdepalvelimille mikäli handlerin kutsuva tavallinen tehtävä suoritetaan onnistuneesti. Mikäli juuri suoritettu tavallinen taski olisi esimerkiksi web-palvelimen

asennus, niin onnistuneen sovellusasennuksen jälkeen handler taskin avulla voitaisiin suorittaa vielä tarvittava web-palvelun eli servicen uudelleenkäynnistys. (DigitalOcean 2016b.)

Palvelinten ryhmitteleminen eri rooleja varten tehdään Ansiblessa inventaarion (inventory) avulla. Inventaarion palvelinryhmiä käyttämällä suositellaan ajamaan esimerkiksi tietyn playbookin sisältö samanaikaisesti useammalle palvelimelle. Inventaarion INI-konfiguraatiotiedosto sijaitsee oletuksena polussa ”/etc/ansible/hosts”.

Ansiblen tehtävien eli taskien suorittamiseen käytetään valmiiksi ohjelmoituja moduuleita, joista kukin toimii oman syntaksin mukaisesti. Moduulit ovat siis tekninen keino, joka käytännössä poistaa Ansiblessa tavallisten OS-komentojen syöttämisen tarpeen käyttäjältä. Moduuleja on tarjolla valmiiksi satamäärin erinäisiin käyttötarkoituksiin ja niitä on myös mahdollista kehittää itse.

Ansiblen historiasta vielä sen verran, että Ansiblen kehitti alkujaan Pythonilla Michael DeHaan, joka on myös palvelinsovellus Cobblerin takana. (Ansible 2013.) Nykyisellään Ansiblen omistaa Red Hat, joka osti Ansiblen vuonna 2015. (Red Hat 2015.)

3.7 Muut työkalut

Tein todella laajasti selvityksiä valitessani soveltuvia työkaluja tähän opinnäytetyöhön ja selvitystöiden aikana löytyi useita mielenkiintoisia sovelluksia palvelimen käyttöönoton, palautuksen ja varmuuskopioinnin suorittamiseen jo esiteltyjen vaihtoehtojen lisäksi. Alla mainittaviin sovelluksiin en valitettavasti ehdi tarkemmin pureutumaan vielä tässä opinnäytetyössä, mutta esittelen silti tiivistetysti löytämäni työkalut.

Kloonausohjelmista täytyy mainita Redo backup, joka toimii niin Linuxilla kuin Windowsilla. Redo toimii ilman asennusta suoraan CD:ltä tai USB-tikulta ja se sisältää graafisen käyttöliittymän, jonka ansiosta Redo soveltuu hyvin myös vähemmän kokeneille käyttäjille. PING sen sijaan käyttää Partimagea ja tarjoaa käytännössä Linux OS:lle samankaltaiset ominaisuudet kuin Windows-puolelta tuttu Symantec Ghost. Muita maininnan arvoisia kloonausohjelmia ovat Mondo Rescue, G4L, Partclone ja perinteinen Partimage sekä DCFLDD, joka perustuu Linuxista tuttuun DD-komentorivisovellukseen. (Tecmint 2014.)

Konfiguraationhallintaohjelmista Ansiblen realistisina vaihtoehtoina olivat mm. Puppet, Chef, CFEngine ja Saltstack. Ruby-kieleen pohjautuvasta Puppetista löytyy, niin ilmainen open-source vaihtoehto kuin maksullinen enterprise-versio, jota voi kokeilla ilmaiseksi 10

nodeen asti. Merkittävin ero versioiden välillä on enterprisesta löytyvä graafinen käyttöliittymä, täysi tuki virtuaalipalvelimille ja maksullinen 24/7 tukipalvelu. Puppetin Forge repositorysta löytyy jopa yli 2000 moduulia käytettäväksi molemmille Puppetin versioille. (Puppet 2017.)

Toinen ruby-kieleen pohjautuva sovellus on Chef, josta löytyy sekä open-source, että enterprise-versio. Cheffin avulla käytetään ”cookbookkeja”, joita löytyy yli 800 ja lisäksi Chef on täysin integroitavissa pilvipalveluiden kanssa. C-kieleen pohjautuva CFEngine on sen sijaan kaikista mainituista työkaluista vanhin joskin sovelluksen ydintoiminnot kohdistuvat jo osittain perinteisten palvelinympäristöjen sijaan pilvipalveluihin. CFEnginellä on yksinkertaista luoda konfigurointiprofiili, jonka avulla suoritettavat toimenpiteet voi vielä varmistaa ennen lopullista komentojen ajoa. (Tomsitpro 2014.)

Saltstack on vastaavasti Ansiblen tavoin Pythonilla kehitetty ja vastikään viime vuosina julkaistu sovellus. Saltstack käyttää helppokäyttöistä YAML-syntaksia minkä vuoksi sen käyttäminen ei varsinaisesti vaadi ohjelmointitaitoja. Muiden sovelluksien tapaan myös Saltstackista löytyy enterprise-versio sekä open-source projekti nimeltään Salt Open (Upguard 2017). Spacewalk on sen sijaan järjestelmän hallintaan tarkoitettu työkalu, joka on yksinkertaisesti sanottuna RHEL Satelliten ilmainen vastine. Spacewalkilla voi esimerkiksi hallita palvelinten asetuksia, asentaa sovelluspäivitykset keskitetysti sekä provisioida kickstart-tiedoston samanaikaisesti joukolle palvelimia (Spacewalk 2015).

Lopuksi vielä palvelimen provisiointisovelluksista maininnan ansaitsee avoimen lähdekoodin The Foreman, Stacki, Cobbler ja FAI (DevOps 2016). Nelikon lisäksi myös eri tason RAID-konfiguroinneista on suuri apu palvelimia varmistaessa ja Rsync sovellukseen pohjautuvalla Rsnapshotilla voi määritellä säännöllisten snapshottien oton Linux-käyttöjärjestelmällä (Rsnapshot 2017).

4 Testiympäristö

Tämä kappale kattaa testiympäristön suunnittelun tarvittavine työkalujen valintoineen. Testiympäristö tulee koostumaan käytännössä joukosta virtuaalipalvelimia ja pienehköstä lähiverkosta. Oleellinen asia testiympäristön suunnittelussa on virtualisointityökalun valinta, jonka avulla toteutetaan CentOS 7 -käyttöjärjestelmällä toimivat testipalvelimet.

Virtualisointityökalun vaatimuksena on siis mahdollisuus ajaa Windows-hostilla Linux Guest OS -käyttöjärjestelmää. Lisäksi kaikkien palvelimien tulee kyetä kommunikoimaan keskenään sisäverkon (LAN) välityksellä ja päästä internetiin toisen virtuaalisen verkko-interfacen kautta.

4.1 Virtualisointityökalun valinta

Esiselvityksien ja aiempien kokemuksien perusteella oli jo entuudestaan tiedossa Oracle VirtualBox ja VMware Workstation Player -virtualisointityökalut, joiden tiesin täyttävän merkittävän osan asettamistani vaatimuksista kevyeen palvelinvirtualisointiin. Tämän vuoksi myös esittelin kyseiset sovellukset tarkemmin heti alkuun.

Näistä kahdesta työkalusta Oracle VirtualBoxista oli ihan käytännön kokemusta, kun eräällä koulun Linux-kurssilla tuli käytettyä kyseistä sovellusta jonkin verran mm. muutaman Ubuntu-virtuaalikoneen luomiseen. VMwaren Workstation Playeristä sen sijaan olin kyllä kuullut paljon, mutta varsinaisia käyttökokemuksia ei löytynyt entuudestaan.

Työpaikan kautta tosin löytyi hieman kokemusta VMwaren ESXi-ympäristön hallinnasta kuten virtuaalikoneiden luomisesta ja resurssien kasvattamisesta sekä verkkoadapterien konfiguroimisesta palvelimelle ja muista vastaavista ”perustehtävistä”. Täten VMwaren tuotteet ei sentään aivan täysin vieraita olleet lähtövaiheessa.

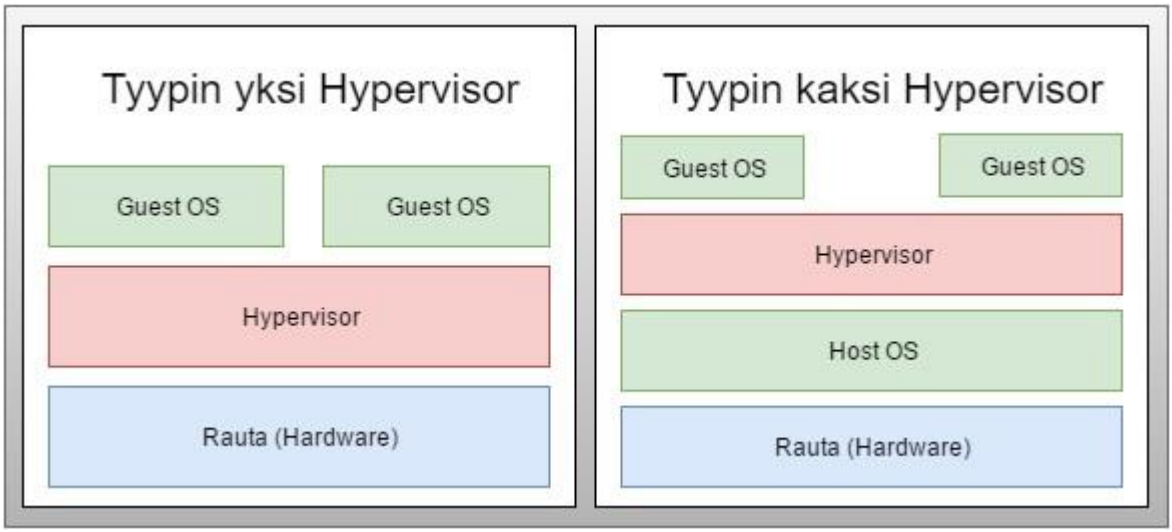
Näiden kahden edellämainitun työkalun lisäksi suoritin paljon selvityksiä tavoitteenani löytää mahdollinen kolmas virtualisointityökalu vertailtavaksi. Soveltuvaa työkalua etsiessä löytyi useita muitakin virtualisoinnin mahdollistavia työkaluja, mutta valitettavasti VirtualBoxin ja Workstation Playerin kanssa selvästi samaan kategoriaan asettuvaa, Windows-hostikonetta tukevaa ja nykyaikaan päivitettyä vaihtoehtoa ei löytynyt.

Suurilta valmistajilta kuten VMwarelta (ESXi), Citrixiltä (XenServer) ja Microsoftilta (Hyper-V) sekä open-source pohjaiselta KVM-projektilta löytyi kyllä hyvinkin laajasti vaihtoehtoisia ratkaisuja virtualisointiin, mutta nämä vaihtoehdot olivat pääasiassa

tarkoitettu suurempiin, puhtaasti oikean palvelinraudan päällä ajettaviin ympäristöihin – pienen, yhdellä kotikoneella toteutettavan labraympäristön sijaan.

Muut löytämäni vaihtoehdot olivat vastaavasti joko suoraan vanhentuneita, kuten pitkään aikaan päivittämättömät ohjelmat ”Colinux” ja ”Virtual PC 2007” tai sitten eivät vain yksinkertaisesti soveltuneet käytettäväksi Windows-hostilla. Tällaisia vaihtoehtoja olivat esimerkiksi ”Qemu”, ”Gnome-boxes” ja ”Virt Manager”, joita käytetään Linux-hostilla ja ”VMware Fusion” sekä ”Parallels Desktop”, jotka ovat tarkoitettu Windows virtuaalikoneen käyttämiseen MAC-tietokoneella.

Näin ollen vaihtoehdoksi jäi virtualisointityökalun valinnan suorittaminen Oracle VirtualBoxin ja VMware Workstation Playerin välillä, jotka ovat selvästi markkinoiden suosituimmat tyypin kaksi hypervisorit.



Kuvio 4. Tyypin yksi ja kaksi hypervisorien arkkitehtuurierot. Ykköstyypin hypervisorilla virtualisointi toteutetaan suoraan palvelinalustan päälle, kun kakkostyypin hypervisor-sovellukset ajavat sen sijaan virtuaalikoneita isäntäkoneen päällä. (Microsoft 2011.)
Ykköstyypin hypervisorit ovat yleisesti tarkoitettu ammattimaiseen ja raskaaseen palvelinkäyttöön siinä missä tässä työssä käytettävä, niin sanottu kakkostyypin hypervisor on tarkoitettu kevyeen työpöytävirtualisointiin.

Taulukko 1. Ominaisuusvertailu Oracle VirtualBoxin ja VMware Workstation Playerin välillä. (VMware 2016d ja VirtualBox 2016e.)

Virtualisointityökalun valinta - ominaisuusvertailu.		
Sovelluksen nimi ja versio	**VirtualBox 5.1.10**	**Workstation Player 12.5.1**
Tuki Windows-hostille	**Kyllä**	**Kyllä**
Usean VM:n luominen	**Kyllä**	**Kyllä**
Tuki 3D grafiikoille	**Kyllä**	**Kyllä**
Usean Guest OS:n samanaikainen ajo	**Kyllä**	**Pro - versiossa**
Snapshottien otto	**Kyllä**	**Pro - versiossa**
Kaupallinen käyttö sallittu	**Kyllä**	**Maksullisella versiolla**

Valinta oli lopulta hyvinkin helppo tehdä Oracle VirtualBoxin hyväksi. Esitestauksen perusteella valinta olisi osunut jo muutenkin astetta miellyttävämmän käyttökokemuksen tarjonneelle VirtualBoxille, mutta reilun ominaisuusvertailun jälkeen valinta sai lopullisen vahvistuksen.

Workstation Playerin perusversiolla ei siis edes olisi ollut mahdollisuutta toteuttaa yksinään testiympäristöä funktionaalisesti, koska se tukee vain yksittäisen virtuaalikoneen samanaikaista käyttöä.

Pro-versiossa olisi usean Guest OS:n samanaikaisen ajon mahdollisuus, mutta yksi kriteeri työkalun valinnalle oli nimenomaan edullisuus: Workstation Player Pro maksaa 251,95€ (VMware 2016e) ja VirtualBox on vastaavasti täysin ilmainen työkalu.

VirtualBoxin valinnan jälkeen oli lopulta tiedossa kaikki testiympäristön rakentamisessa ja hallinnoinnissa käytettävät työkalut (Oracle VirtualBox ja SSH-client MobaXterm) sekä palvelimiin asennettava käyttöjärjestelmä (CentOS 7 -minimal).

4.2 Arkkitehtuurin suunnittelu

Seuraavaksi oli vuorossa varsinainen testiympäristön arkkitehtuurin suunnittelu ennen toteutusvaihetta. Olin jo esisuunnittelun aikana testannut Oracle VirtualBoxia sen verran perusteellisesti, että osasin hahmottaa pääpiirteittäin minkälainen ympäristö työkalujen testausta varten olisi mahdollista toteuttaa VirtualBoxilla.

Opinnäytetyön myöhemmissä kappaleissa testataan yhteensä kolmea eri työkalua, jonka perusteella päättelin kuuden testipalvelimen riittävän testejä varten. Tämän päätelmän mukaisesti päätin toteuttaa ”4+2 palvelinmallin” mikä tarkottaa, että kuudesta palvelimesta neljä on tavallisia testipalvelimia ja kaksi deployment-palvelimia.

Deployment-palvelimien rooliin kuuluu siis palvella tavallisia testipalvelimia esimerkiksi tilanteissa missä tarvitsee ajaa komentoja testipalvelimien käyttöönottoa tai palautusta varten. Esimerkiksi Ansiblen asennan nimenomaisesti yhdelle deployment-palvelimista.

Testiympäristön vaatimuksiin kuuluvat lisäksi, että kaikkia palvelimia pystyy käyttämään tarvittaessa samanaikaisesti ja kaikkien palvelimien välillä tulee pingin kulkea sekä yhteydenoton onnistua keskenään esimerkiksi SSH-protokollan avulla. Tulen siis konfiguroimaan pienen lähiverkon kaikkien palvelimien kesken käyttäen host-only adapteria ja staattisia IP-osoitteita.

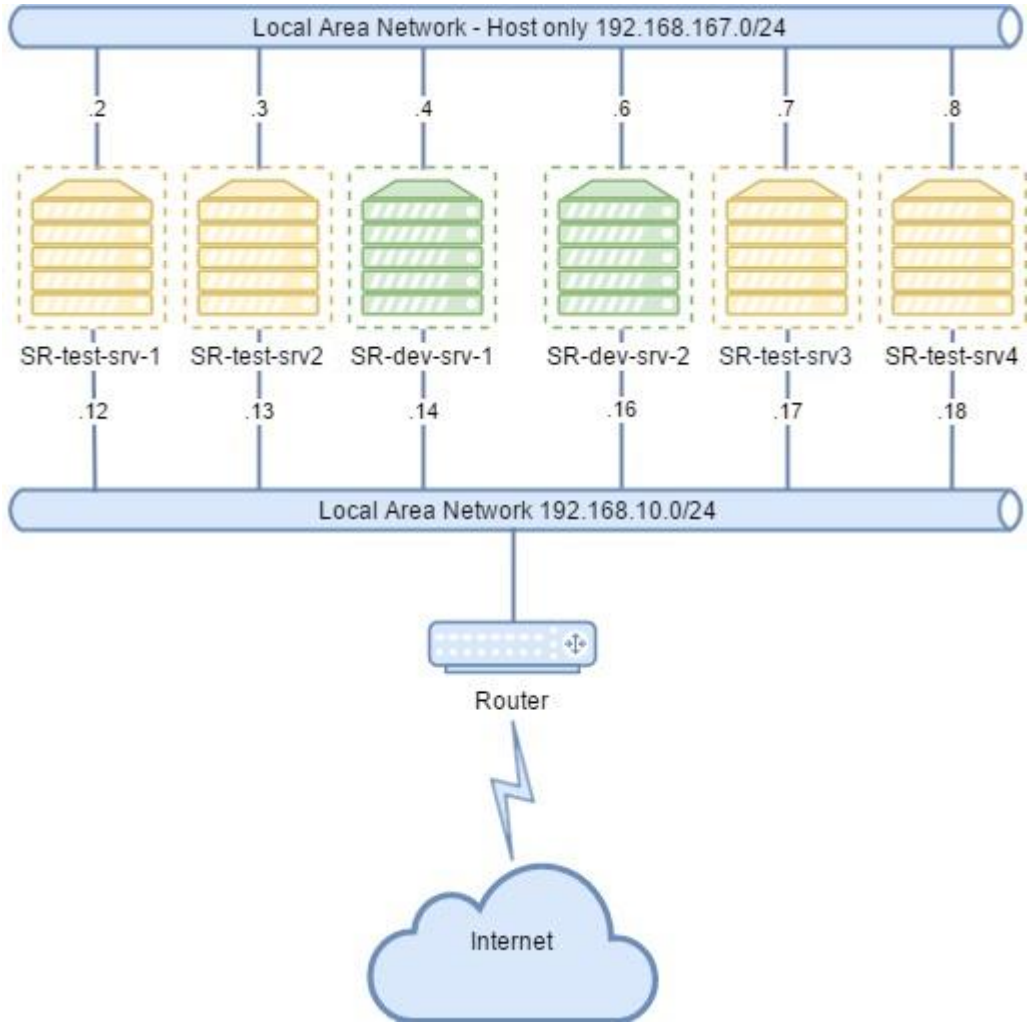
Lisäksi jokaisella palvelimella pitää olla pääsy internetiin, jotta on mahdollista hakea tarpeelliset työkalut Linuxin pakettivarastoista. Tämän mahdollistamiseksi tulen konfiguroimaan jokaiselle virtuaalipalvelimelle toisen verkkointerfacen käyttäen siltaavaa tilaa, joka mahdollistaa VirtualBoxissa staattisten IP-osoitteiden käytön.

Muita erityisiä vaatimuksia testiympäristölle ei ole alkuun, koska tarkoituksena on nimenomaan, että palvelimet ovat mahdollisimman raa'assa tilassa työkaluja varten. Testattavien työkalujen avulla siis konfiguroidaan tarkemmin käyttöjärjestelmän asetuksia ja asennetaan sovelluksia testipalvelimille.

Testiympäristön pystytyksen jälkeen tulen käyttämään MobaXterm SSH-sovellusta palvelimien hallintaan, koska sen avulla palvelimien käyttö on vaivattomampaa ja ainoa vaatimus clientin käytölle on verkon konfigurointi ja palvelimien päällä olo VirtualBoxissa.

Taulukko 2. Testiympäristön vaatimusmäärittely.

Vaatimusmäärittely	
Nro	Vaatimus
1	Kuuden virtuaalipalvelimen luonti - 2GB RAM-muistia ja 20gt kovalevytilaa per palvelin.
2	Luotujen palvelimien tulee voida olla päällä ja käytettävissä samanaikaisesti.
3	Pingauksen tulee toimia kaikkien palvelimien kesken.
4	SSH-yhteyden ottaminen palvelinten kesken tulee toimia.
5	Palvelimissa tulee käyttää ainoastaan staattisia IP-osoitteita.
6	Jokaisella palvelimella tulee olla pääsy internetiin 2.verkkointerfacen kautta.



Kuvio 5. Toteutettavan testiympäristön lähiverkon topologia, jossa on kuvattu kaikki kuusi virtuaalipalvelinta sekä näille konfiguroitavat verkot. Vihreällä merkityt palvelimet ovet deployment-palvelimia ja keltaisella merkityt tavallisia testipalvelimia.

Taulukko 3. Oraclen VirtualBoxilla luotavia virtuaalipalvelimia käytetään host-laitteistolla, jonka tärkeimmät ominaisuudet ovat lueteltu oikeanpuolisessa sarakkeessa.

Käytettävä laitteisto	
Käyttöjärjestelmä	Windows 10 Pro 64-bit
Prosessori	Intel Core i5 2500K @ 3.30GHz
RAM-muisti	24GB DDR3 1866MHz
Kiintolevy	256GB SSD 2,5”

Seuraavaksi on vuorossa testiympäristön toteutus – alkuun asennetaan Oracle VirtualBox host-koneelle, jonka jälkeen luodaan kyseisellä ohjelmalla kuusi virtuaalipalvelinta

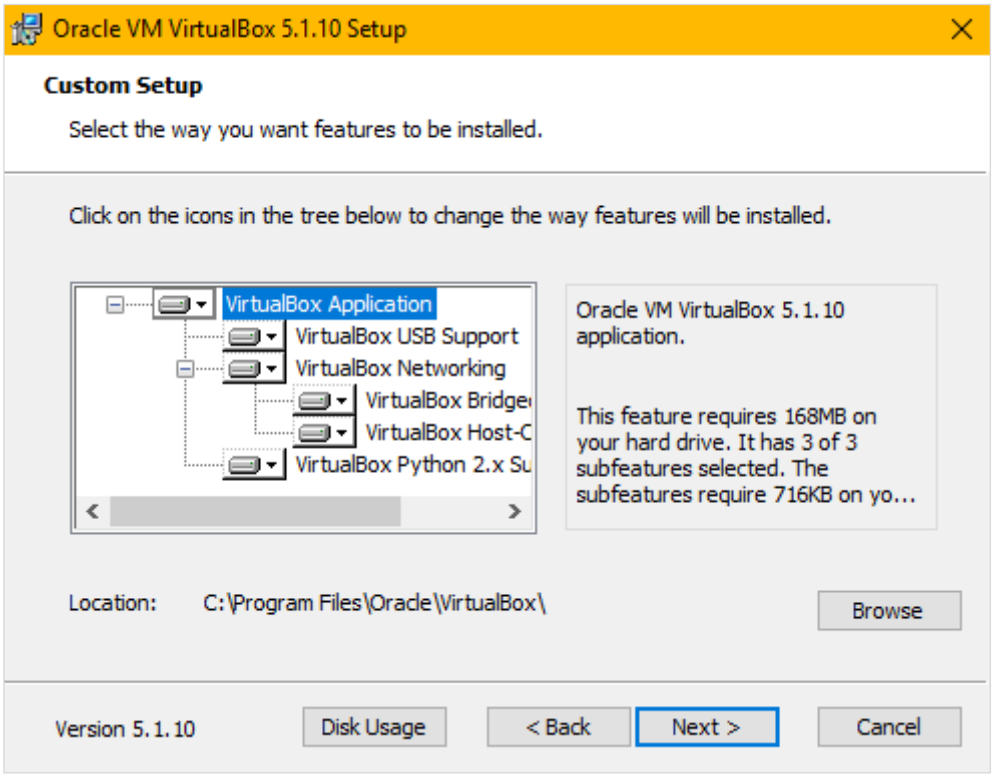
suunnittelukappaleen vaatimusmäärittelyn mukaisesti. Jokaiselle luotavalle palvelimelle asennetaan ja konfiguroidaan CentOS 7-käyttöjärjestelmän minimal-versio, joka on pelkän komentorivin avulla hallittava palvelinkäyttöjärjestelmä.

Käyttöjärjestelmän perusasennuksen lisäksi tulee konfiguroida VirtualBoxilla jokaiselle palvelimelle virtuaalinen host-only verkko palvelinten välisiin yhteyksiin ja toinen siltaava verkkointerface, jolla mahdollistetaan pääsy internetiin sovellustyökalujen lataamista varten.

Testiympäristön toteutusvaiheesta eli virtuaalikoneiden luomisesta VirtualBoxilla, CentOS 7 -asennuksista virtuaalikoneisiin ja tarvittavien verkkojen konfiguroimisesta esitellään pääsääntöisesti vain yhden palvelimen asennusprosessi. Toteutettavat toimenpiteet, kun ovat tässä vaiheessa täsmälleen samat jokaiselle kuudelle palvelimelle.

4.3 Hypervisorin asennus ja virtuaalikoneiden luominen

Aloitin toteutuksen lataamalla CentOS:n ja Oracle VirtualBoxin ohjelmistopaketit valmistajien kotisivuilta. Oracle VirtualBoxista latasin 5.1.10 version ja CentOS 7 -käyttöjärjestelmästä minimal-version 1511. CentOS-paketin koko oli vain 603MB, joten kyseessä oli hyvinkin pieni käyttöjärjestelmäpaketti. Vuorossa oli alkuun VirtualBoxin asennus – alkunäkymä sisälsi vain tervetulotoivotukset ja sovelluksen versiotiedot.



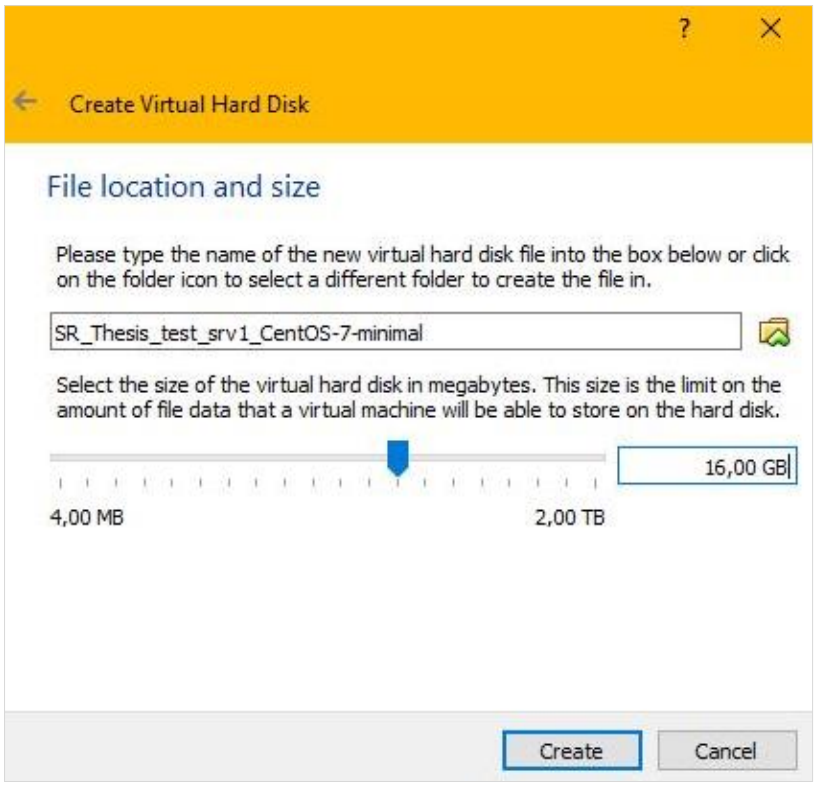
Kuva 5. Toisessa vaiheessa pyydettiin valitsemaan asennukseen sisällytettävät ominaisuudet joihin kuuluivat mm. USB-tuki ja verkkoadapterit. Valitsin kaikki ominaisuudet ja etenin asennuksessa.

Kolmannessa vaiheessa oli ainoastaan varoitus, että Oracle VirtualBox katkaisee ja uudelleenkäynnistää käytettävän verkkoyhteyden asennushetkellä ja etenemisen jälkeen ohjelma asentui tietokoneelle. Viimeisessä, neljännessä asennusikkunassa ilmoitettiin asennuksen onnistuneen.

Neljännen ikkunan jälkeen painoin lopuksi ”finish” ja vasta-asennettu Oracle VirtualBox aukesi nopeasti. Asennusprosessi oli siis hyvinkin yksinkertainen mikä oli vain hyvä asia, koska asennusvaiheet sisälsivät vain olennaisia asetuksia.

Seuraavaksi loin tarvittavat virtuaalikoneet VirtualBoxilla. Virtuaalikoneet voi luoda joko käyttäen aloittelijoille tarkoitettua opastettua tapaa (”guided-mode”) tai kokeneemmille käyttäjille suunnattua ”expert-modea”. Lopputulos on sama valitusta tavasta huolimatta.

Guided-moden ero expert-modeen verrattuna on ainoastaan virtuaalikoneen asetuksien valinta yksitellen ”wizard”-tyylisesti ja lisäksi jokaisen asetuksen vieressä on lyhyt selitys sen merkityksestä. Guided-modessa on yhteensä kuusi asennusvaihetta, kun expert-modessa täysin samat asetukset on sisällytetty kahteen asennusvaiheeseen ilman ylimääräisiä selityksiä.



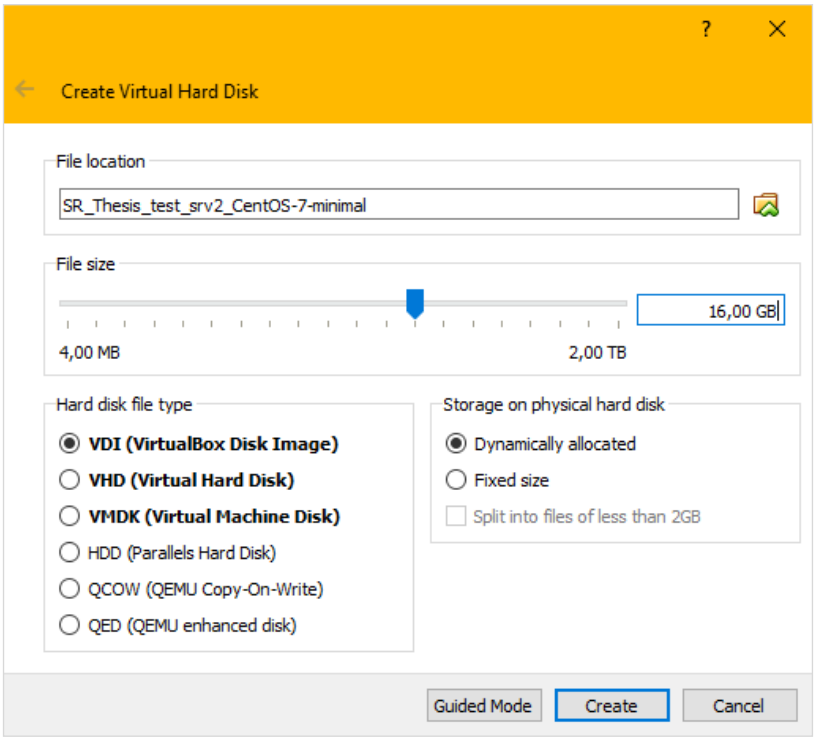
Kuva 6. Ensimmäisen VM:n loin guided-modella ja loput palvelimet käyttäen hivenen nopeampaa expert-modea. Kuvassa Guided-moden viimeinen valintaikkuna ennen virtuaalikoneen luomista.

Taulukko 4. Luotujen virtuaalikoneiden nimet VirtualBoxissa.

Virtuaalikoneiden nimet Oraclen VirtualBoxissa
SR_Thesis_test_srv1_CentOS-7-minimal
SR_Thesis_test_srv2_CentOS-7-minimal
SR_Thesis_test_srv3_CentOS-7-minimal
SR_Thesis_test_srv4_CentOS-7-minimal
SR_Thesis_deployment_srv1_CentOS-7-minimal
SR_Thesis_deployment_srv2_CentOS-7-minimal

Alla on vielä selitetty kuinka virtuaalikone luodaan VirtualBoxilla käyttäen Expert-modea. Expert-modella virtuaalikonetta luodessa painetaan aluksi VirtualBoxin aloitusnäkymän vasemmasta yläreunasta ”new”-painiketta, jonka jälkeen esiin tulleesta pop-up ikkunasta painetaan ”Expert-mode”-nappia. Tämän jälkeen expert-mode tulee välittömästi esiin.

Expert modessa on kaksi asennusikkunaa: näistä ensimmäisessä nimetään luotava virtuaalikone ja valitaan VM:lle myöhemmin asennettavan käyttöjärjestelmän tyyppi ja nimi VirtualBoxin omaa luokittelua varten. Lisäksi valitaan keskusmuistin määrä ja tarvittaessa virtuaalisen kiintolevyn asennus.



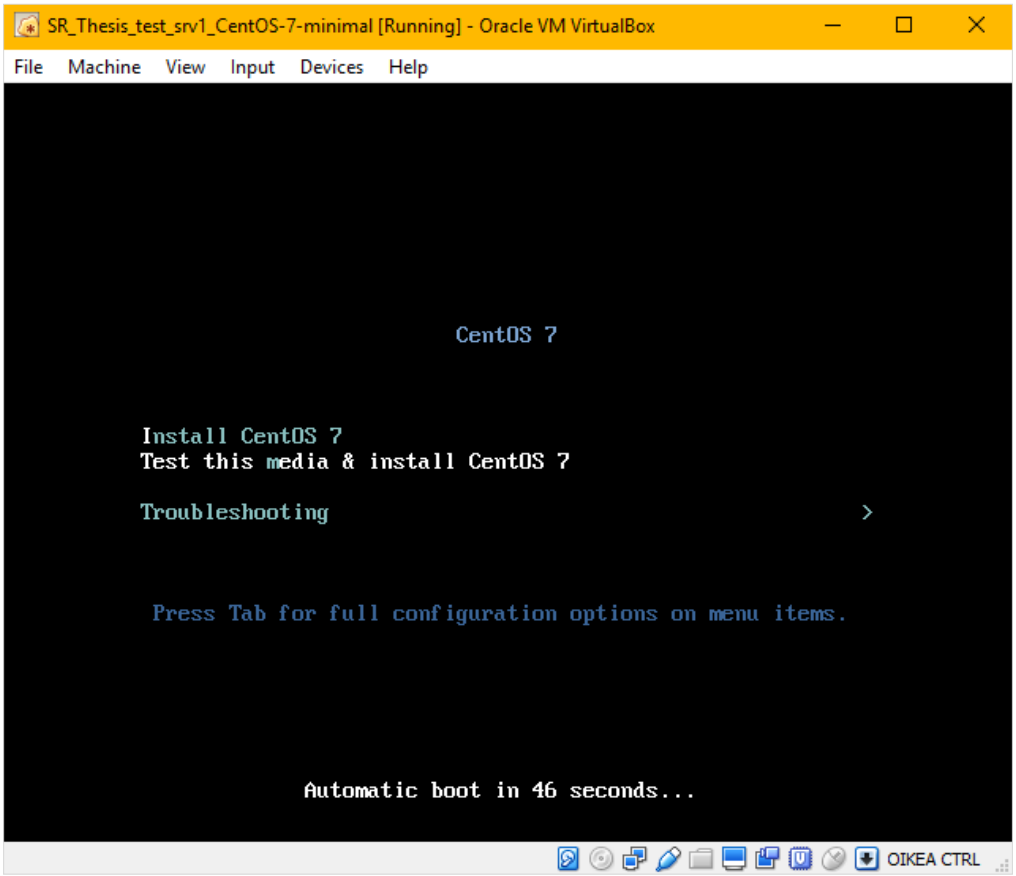
Kuva 7. Toisessa Expert-moden asennusikkunassa valitaan yllä olevan kuvan mukaisesti luotavan virtuaalikoneen tallennuspaikka host-koneen fyysiseltä kiintolevyltä ja valitaan käytettävä virtuaalikoneen tiedostoformaatti tarvittavalla tallennustilalla.

Lisäksi valitaan vielä kasvatetaanko virtuaalikoneen käyttämää levytilan määrää dynaamisesti tarpeen mukaan vai määritetäänkö virtuaalikoneelle suoraan kiinteä maksimikoko. Lopuksi painetaan ”create”, jolloin virtuaalikone luodaan välittömästi.

4.4 Käyttöjärjestelmän asentaminen virtuaalikoneisiin

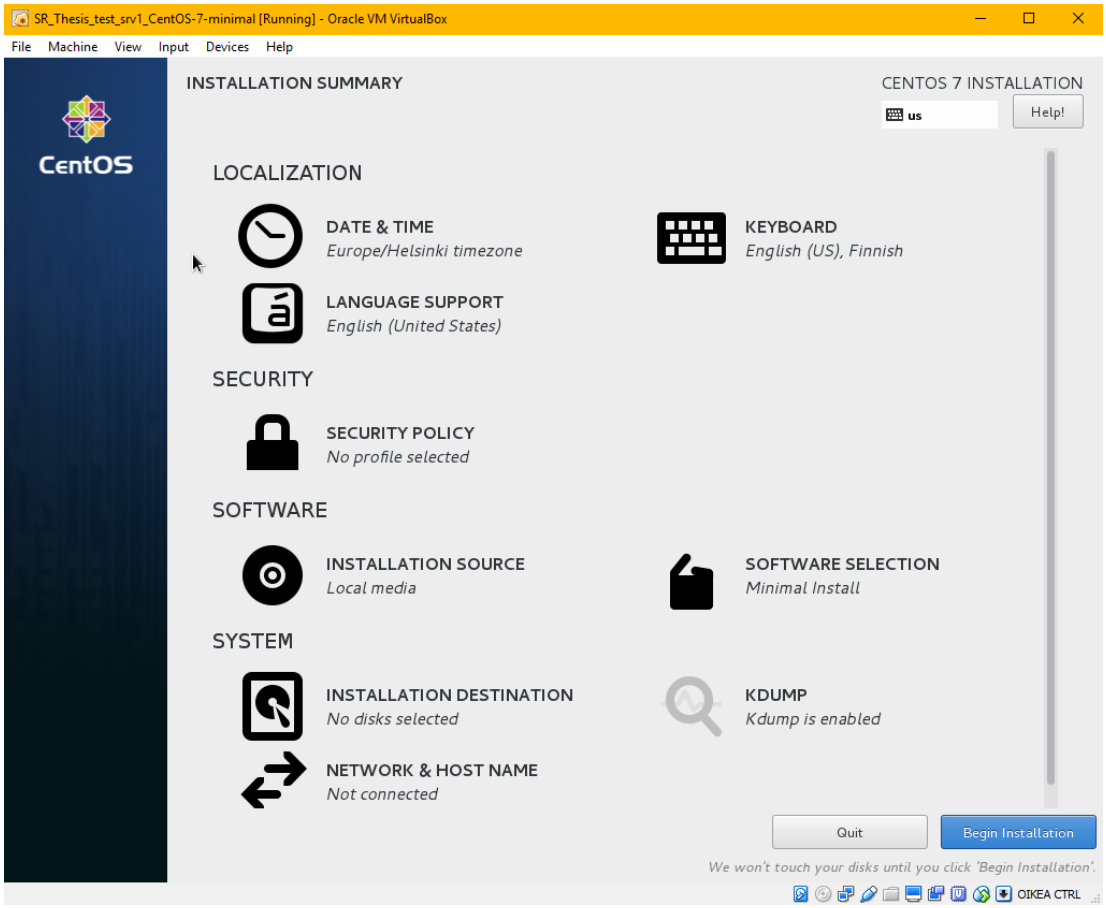
CentOS:n asentamiseen virtuaalikoneeseen ei tarvitse muuta kuin CentOS-imagen, jonka olin jo ladannut aiemmin CentOS:n virallisilta sivuilta. Olin siis täysin valmis aloittamaan käyttöjärjestelmien asennukset luoduille virtuaalikoneille.

Aluksi tuplaklikkasin juuri luotua virtuaalipalvelinta VirtualBoxin avausnäkymässä, jolloin se käynnistyi omaan ikkunaansa. Seuraavaksi valittiin avautuneessa pop-up ikkunassa CentOS:n image hostikoneen tiedostosijainnista. Tämän jälkeen CentOS:n asennusikkuna tuli näkyville ja varsinainen asennusprosessi alkoi.



Kuva 8. VirtualBoxin testipalvelimille asennetaan CentOS 7:n minimal-versio.

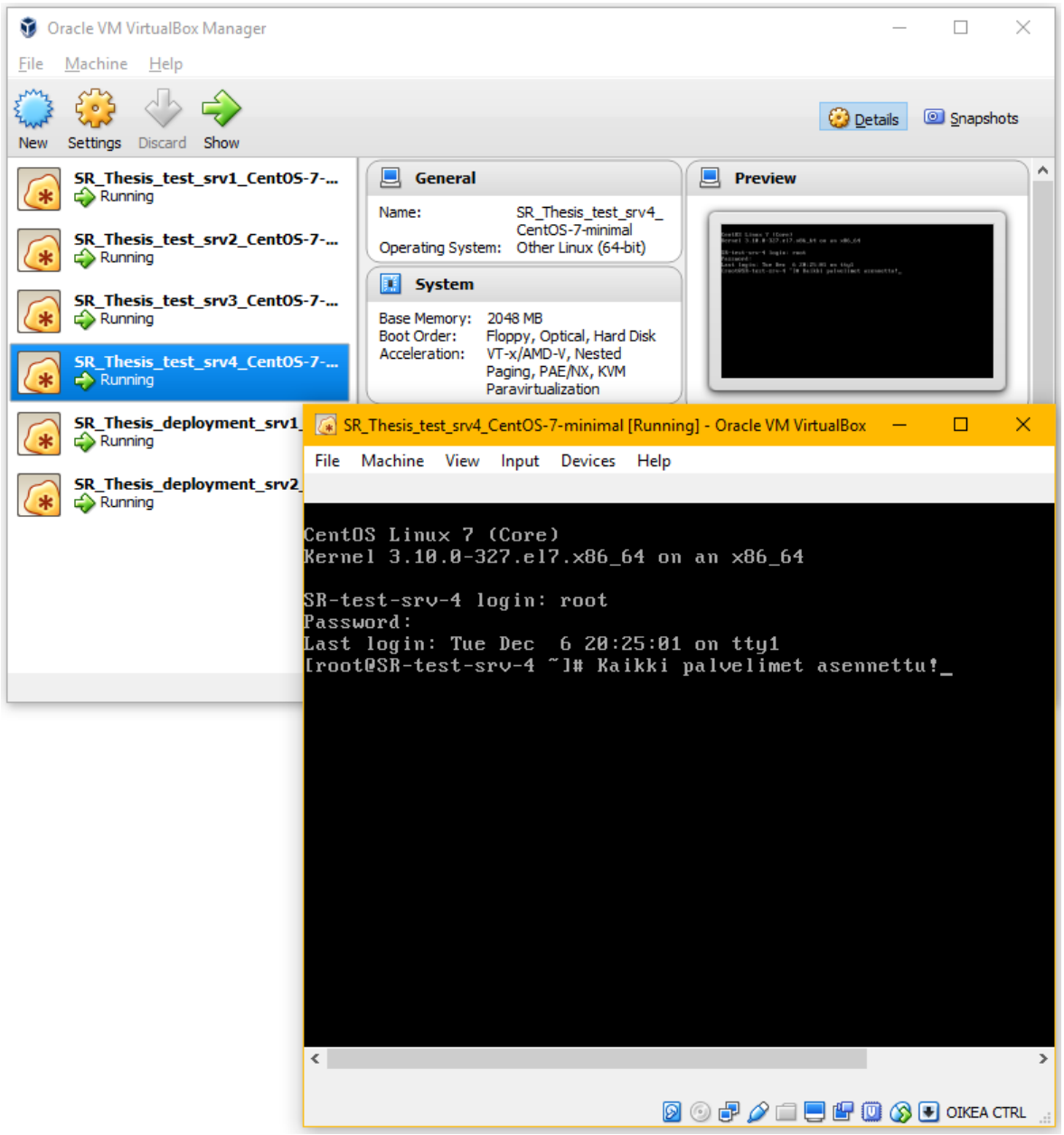
Valitsin CentOS:n asennusikkunasta suoraan ”install CentOS 7” -kohdan sillä tiesin asennettavan imagen olevan eheä. Tämän jälkeen valittiin asennusprosessissa käytettävä kieli – oletuksena ollut englanti (US) ajoi asiansa. Seuraavaksi latautuikin CentOS:n asennusprosessin pääikkuna eli ”installation summary” alla olevan kuvan mukaisesti.



Kuva 9. Yllä olevissa valikoissa on muutamia asetuksia, jotka on hyvä konfiguroida ennen kuin käynnistää lopullisesti varsinaisen asennuksen. Samalla säästää aikaa, kun ei tarvitse myöhemmin palata kyseisten asetuksien pariin.

Aluksi lisäsin näppäimistön kieleksi suomen ja vaihdoin ajaksi, niin ikään Suomen käyttämän aikavyöhykkeen. Konffasin myös palvelimen hostnamen ja valitsin asennuksen tallennuspaikaksi virtuaalikoneen oman virtuaalilevyn.

Seuraavaksi aloitin varsinaisen asennuksen ”begin installation” -painikkeella. Asennuksen ollessa käynnissä oli vielä mahdollista konfiguroida tunnuksia ja asetinkin root-tunnukselle tässä vaiheessa vahvan salasanan sekä loin vielä toisen käytettävän tunnuksen. Asennuksen valmistustuttua palvelin oli heti käytettävissä bootin jälkeen.



Kuva 10. Kuuden CentOS-palvelimen asennus sujui melko nopeaan tahtiin.

Pystytin testiympäristön palvelimet tällä kertaa manuaalisesti käsityönä sen sijaan, että olisin kloonannut VirtualBoxilla ensimmäiseksi luodusta palvelimesta muutaman kopion nopeaan tahtiin ja vaihtanut näille joukon tarvittavia asetuksia.

Mikäli manuaalista työtä olisi ollut testiympäristön rakentamisessa merkittävästi enemmän, niin tällöin olisi jo kannattanut suoraan vain kloonata palvelimet. VirtualBox osaa kätevästi generoida verkkoadapterien MAC-osoitteetkin yksilölliseksi kloonattaville koneille välttääkseen verkko-ongelmat. Käyttäessä staattisia IP-osoitteita pitää toki tällöinkin konfiguroida IP-osoitteet itse yksilölliseksi.

4.5 Verkon konfigurointi

Testiympäristön toteutusvaiheesta oli vielä jäljellä verkkoasetusten konfigurointi juuri luoduille palvelimille. Tehtävänä oli alkuun konfiguroida VirtualBoxissa kaikille palvelimille host-only sisäverkko. Host-only adapterin käyttö soveltuu erinomaisesti testiympäristöön, koska kaikki virtuaalikoneet ajetaan samalta hostilta.

Host-only adapterin erikoisuus on siis, että se mahdollistaa kaikkien samalla hostilla olevien virtuaalipalvelinten ja lisäksi itse host-koneen välisen kommunikoinnin loopback-periaatteella, mutta liikennöinti ei onnistu host-koneen ulkopuolelle – Ei edes toiselle samassa lähiverkossa olevalle tietokoneelle.

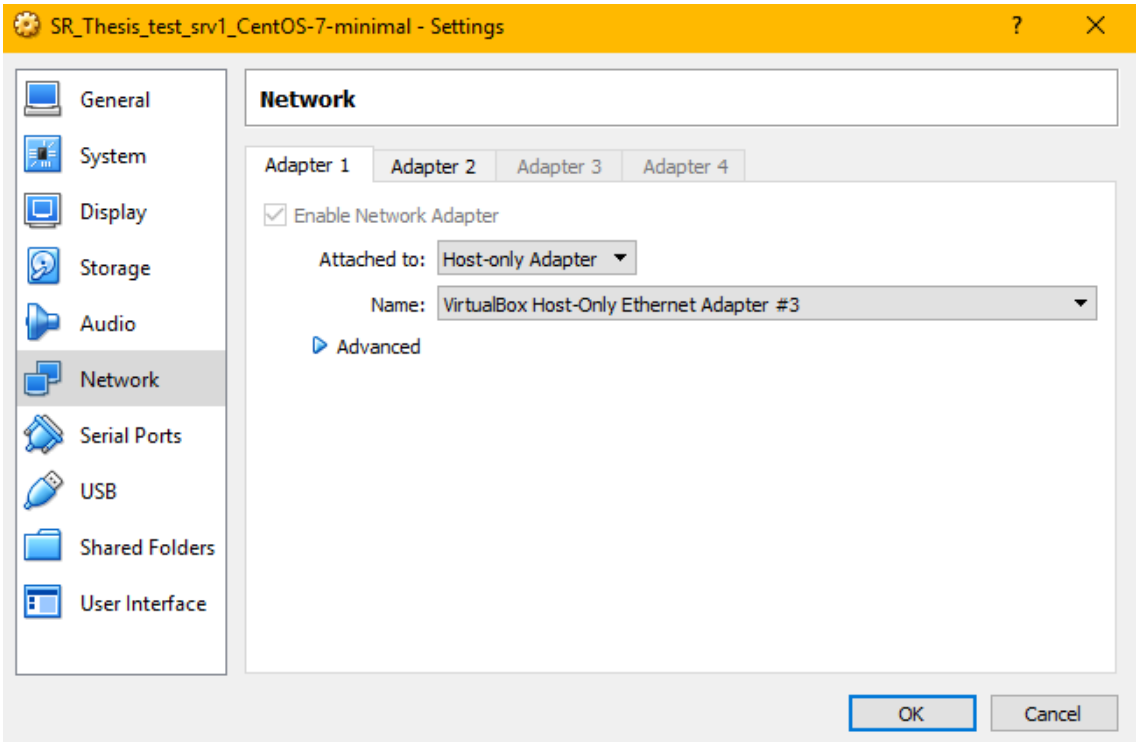
Host-only adapterin lisäksi joka palvelimelle tuli kuitenkin mahdollistaa pääsy internetiin konfiguroimalla siltaavaa tilaa käyttävä toinen verkko-interface. VirtualBoxin virtuaalinen verkkoadapteri käyttää bridged-tilassa suoraan host-tietokoneen verkkokorttia mahdollistaakseen internet-yhteyden. (VirtualBox 2016f.)

Kaikki palvelimille asetettavat IP-osoitteet suunniteltiin staattiseksi eli manuaalisesti itse määriteltäviksi. Palvelimille suunnitellut IP-osoitteet löytyvät sivun 24 verkkotopologia-kuvasta. Muita erikoisempia vaatimuksia verkolle ei tässä vaiheessa ollut, joten pystyin samantien aloittamaan verkkoasetusten konfiguroimisen.

Aluksi loin VirtualBoxin asetuksista host-only ethernet adapterin palvelimia varten. Adapterin luominen onnistui polusta: *File -> Preferences -> Host-only Networks*

Luotavaan adapteriin tarvitsi lisätä ainoastaan host-only verkon määrittävän IP-osoitteen (192.168.167.1) ja verkkomaskin (255.255.255.0). Host-only adapteria käyttäessä ei tarvitse kuitenkaan konfiguroida oletusyhdyskäytävää itse palvelimissa, vaikka VirtualBoxissa adapterin asetuksista löytyvä ”.1”-päätteinen osoite siihen viittaisikin.

Adapterin konfiguroinnin jälkeen Windows host-kone tunnisti jo kyseisen virtuaaliadapterin minkä sai selville suoraan ”ipconfig”-komennolla. Seuraavaksi konfiguroin verkkoadapterit kuntoon palvelimille. Ensimmäiseksi adapteriksi asetin jokaiselle palvelimelle hetki aiemmin luodun host-only adapterin ja toiseksi adapteriksi host-koneen varsinaista verkkokorttia sillattuna (”bridged”) käyttävän virtuaalisen adapterin.



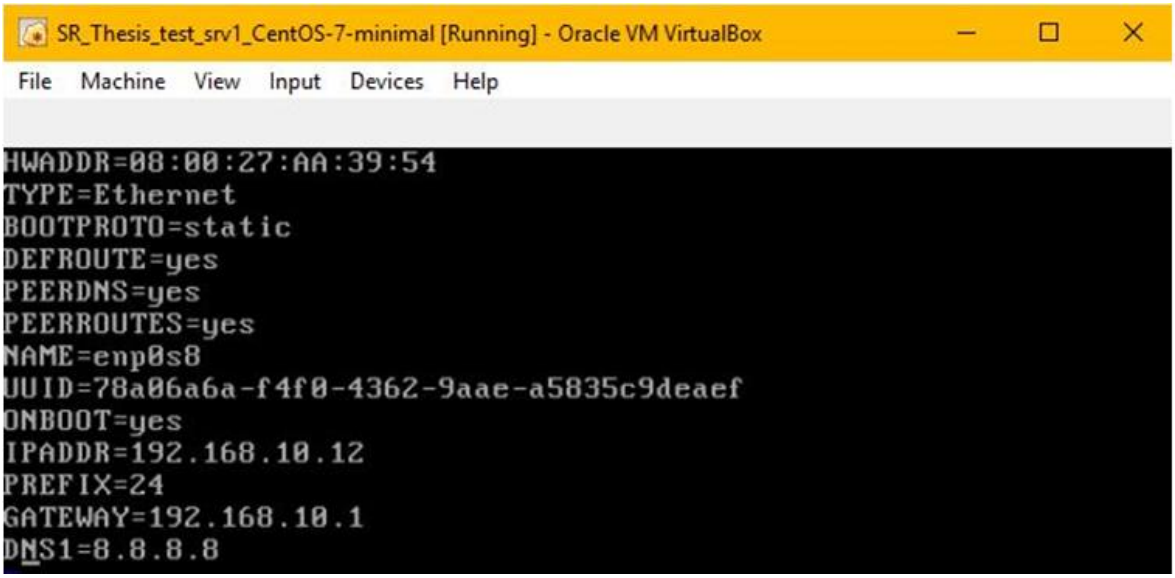
Kuva 11. VirtualBoxin verkkoadapterien konfigurointi onnistuu network-välilehdeltä.

Seuraavaksi oli aika konfiguroida verkko-interfacet kuntoon itse testipalvelimien käyttöjärjestelmissä. Yksi helpoimmista keinoista asettaa verkkoasetukset kuntoon pysyvästi on muokata polusta *”/etc/sysconfig/network-script”* löytyviä verkkoadapterien konfigurointitiedostoja ja lopuksi uudelleenkäynnistää verkkoadapteri.

Testiympäristön palvelimille on konfiguroitu kaksi virtuaaliadapteria per palvelin, joten jokaiselle kuudelle palvelimelle tuli asettaa verkkoasetukset kuntoon kahteen erilliseen konfigurointitiedostoon. Konfigurointitiedostot on aina nimetty verkkoadaperin nimen perusteella.

Verkkoadapterien konfiguraatiotiedostojen muokkaus onnistui kohtuullisen nopeasti jokaiselle kuudelle palvelimelle. Ainoastaan IP-osoitteet, UUID:t ja MAC-osoitteet olivat yksilöllisiä konfigurointitiedostoissa – kaikki muut tiedot olivat identtisiä jokaisella palvelimella.

Esimerkiksi host-only verkkoadapterin ”enp0s3” konfigurointitiedostossa tarvitsi muokata neljää eri kohtaa (BOOTPROTO, ONBOOT, IPADDR ja PREFIX). Edellä mainittujen asetuksien avulla varmistin, että palvelimella on valmiina uudelleenkäynnistyksenkin jälkeen staattista IP-osoitetta käyttävät asetukset oikealla verkkomaskilla.



Kuva 12. Vastaavasti palvelimien toisella verkkoadapterilla eli siltausta käyttävällä enp0s8:lla tarvitsi muokata yllä olevan kuvan mukaisesti attribuuttien seuraavat kohdat:

- BOOTPROTO (staattisen IP:n käyttö.)
- ONBOOT (adapterin automaattinen käynnistyminen.)
- IPADDR (interfacelle määritettävä IP-osoite.)
- PREFIX (verkkomaskin pituus.)
- GATEWAY (oletusyhdyskäytävä, jonka kautta liikenne kulkee.)
- DNS1 (nimipalvelimen avulla saadaan tietoon verkko-osoitetta vastaava IP-osoite.)

Adapterien konfigurointien lisäksi asensin vielä palvelimille net-tools nimisen työkalun, jonka jälkeen oli enää jäljellä verkon toimivuuden varmistus. Kaikkien alla olevassa taulukossa lueteltujen palvelimien pitäisi nyt pystyä pingaamaan toisiaan ja SSH-yhteyden toimia. Lisäksi internet-yhteyden kuuluisi toimia toisen verkko-interfacen kautta.

Taulukko 5. Testipalvelimien IP-osoitteet verkkoadapterin mukaan.

Testiympäristö		
Palvelin	Host-only interface (enp0s3)	Bridged virtual interface (enp0s8)
SR-test-srv-1	192.168.167.2	192.168.10.12
SR-test-srv-2	192.168.167.3	192.168.10.13
SR-dev-srv-1	192.168.167.4	192.168.10.14
SR-dev-srv-2	192.168.167.6	192.168.10.16
SR-test-srv-3	192.168.167.7	192.168.10.17
SR-test-srv-4	192.168.167.8	192.168.10.18

Lopuksi vielä varmistetaaan testiympäristön verkon toimivan suunnitellusti tekemällä muutamia yksinkertaisia testejä. Käytännössä suoritettaviin testeihin lukeutuvat:

- Järjestelmän virtuaalikoneiden keskinäinen pingaus
- Yhteydenotto virtuaalikoneen SSH (Secure Shell) -clientillä toiseen virtuaalikoneeseen.
- Verkkoyhteyden toimivuuden varmistus virtuaalikoneilla.
- Pakettivaraston (eli repositoryn) toimivuuden varmistus virtuaalikoneilla.

Taulukot 6-9. Testitapaukset 1-4 taulukoituna testitapauksen, toimenpiteen ja tuloksen mukaan. Vihreä väri indikoi testitapauksen lopputuloksen olevan vaatimusten mukainen.

Testitapaus 1: Järjestelmän virtuaalikoneiden keskinäinen pingaus.
Toimenpide: Testata ping-komentoa vuorotellen eri testipalvelimilla jokaisen muun testipalvelimen host-only interfacen IP-osoitteeseen.
Tulos: Pingi kulki keskenään kaikkien palvelimien välillä.

Testitapaus 2: Dev-palvelimella SSH-yhteys toiseen virtuaalikoneeseen.
Toimenpide1: Ottaa SR-dev-srv-1 palvelimella SSH-yhteys SR-test-srv-1 (192.168.167.2) ja SR-test-srv-2 (192.168.167.3) palvelimille.
Toimenpide 2: Ottaa SR-dev-srv-2 palvelimella SSH-yhteys SR-test-srv-3 (192.168.167.7) ja SR-test-srv-4 (192.168.167.8) palvelimille.
Tulos: SSH-yhteys toimi vaatimusten mukaisesti.

Testitapaus 3: Verkkoyhteyden toimivuuden varmistus virtuaalikoneilla.
Toimenpide: Pingaus jokaisella testipalvelimella yle.fi ja google.fi - sivustoihin, joissa pingin käyttöä ei ole estetty sekä Googlen nimipalvelimeen 8.8.8.8.
Tulos: Kaikilla testipalvelimilla toimi pingaus testikohteisiin.

Testitapaus 4: Pakettivaraston (eli repositoryn) toimivuuden varmistus virtuaalikoneilla.
Toimenpide: Asentaa levyliikenteen monitorointiin Linuxissa tarkoitettu Iotop-sovellus pakettivarastosta jokaiselle testipalvelimelle. Käytettävä komento: ”yum install iotop”.
Tulos: Sovellus asentui jokaiselle testipalvelimelle.

Kaikki testit sujuivat vaatimusten mukaisesti ja kaikilla palvelimilla oli käytössä määritetyt staattiset IP-osoitteet. Testiympäristö siis valmistui suunnitellusti ja seuraavaksi oli aika siirtyä palvelimien käyttöönoton ja palautuksen mahdollistavien työkalujen testaukseen.

5 Palvelimen palautus levykuvasta

Ensimmäisessä vaiheessa (kappaleet 5.1-5.2) otan Clonezillan tavallisella live-versiolla yksittäisestä virtuaalipalvelimesta (Test_srv1) levykuvan. Onnistuneen kloonaamisen jälkeen palautan juuri luodun levykuvan samaiselle palvelimelle.

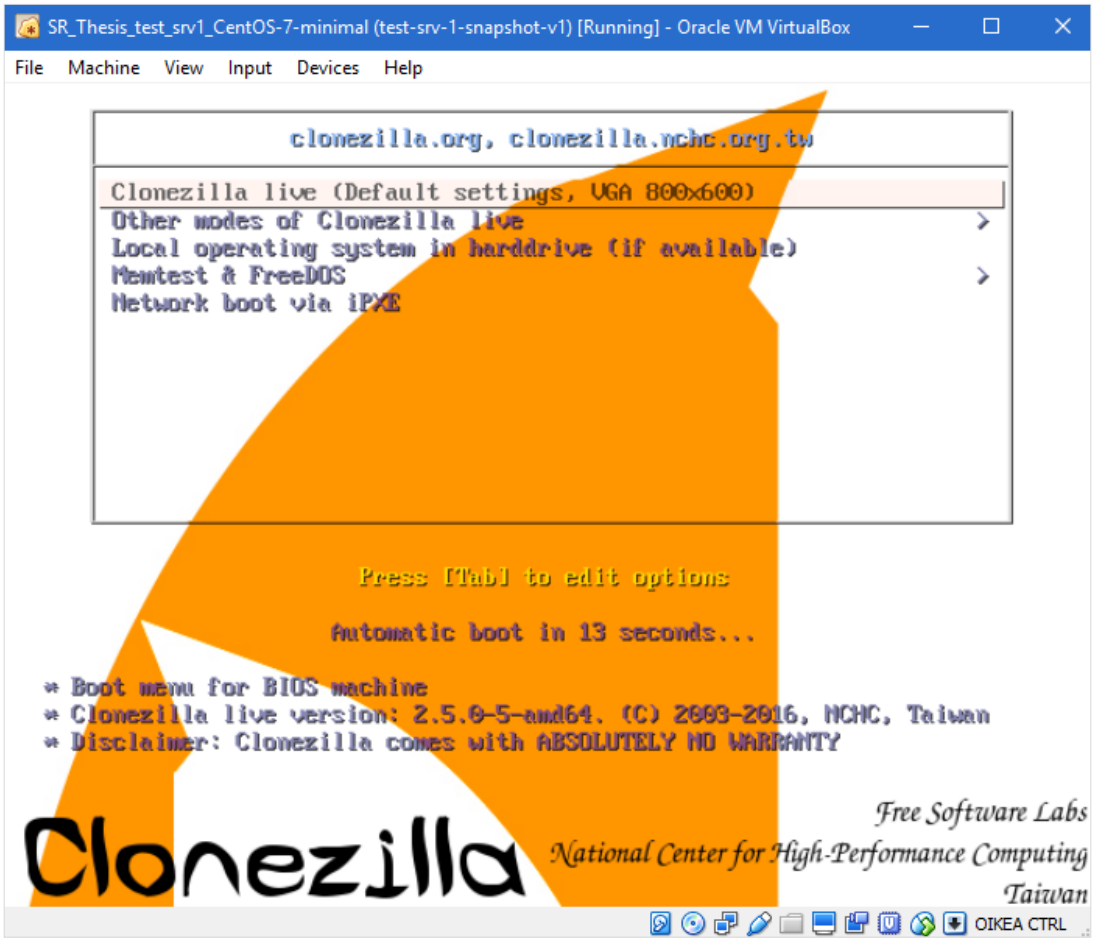
Toisessa vaiheessa (kappale 5.3) otan käyttöön Clonezilla server editionin sisältävän DRBL-live ympäristön Deployment_srv1 -palvelimella. Tarvittavien konfigurointien jälkeen kloonaan Test_srv1:n levykuvan Test_srv2:lle verkon välityksellä ja lopuksi varmistan kloonauksen onnistumisen.

5.1 Paikallisen levykuvan luonti

Alkuun latasin Clonezillan sivuilta sovelluksen live-version 2.5.0-5. Live-version käyttö eroaa normaalin version käytöstä siinä mielessä, että sovellusta ei tarvitse asentaa erikseen palvelimelle, vaan sovellusta käytetään palvelimella mountin välityksellä (CD:n tai USB-muistin kautta) vain tarvittavan ajanjakson.

Ennen Clonezillan testaamista loin vielä Test_srv1 -palvelimen juureen ”oppari-testi.txt” nimisen testitiedoston, johon kirjoitin sisällöksi pari riviä tekstiä. Tiedoston oli tarkoitus todistaa levykuvan ottamisen ja palautuksen onnistumisen eli tämä kyseinen tiedosto tulisi edelleen löytyä palvelimelta, kun levykuva on otettu ja palautettu. Levykuvan ottamisen jälkeen luon vielä myöhemmin toisen tekstitiedoston, jonka vastaavasti pitäisi hävitä, kun palautan aiemmin otetun levykuvan palvelimelle.

Seuraavaksi oli aika aloittaa itse Clonezilla-työkalun live-version testaus. Mounttasin ladatun clonezilla.iso -tiedoston testipalvelimen virtuaaliselle levyasemalle ja käynnistin palvelimen uudelleen, jolloin valitsin koneen boottavan virtuaaliselta CD-asemalta. Palvelimen käynnistyessä uudelleen tuli alla näkyvä Clonezillan aloitusruutu näkyviin ja aloitinkin välittömästi toimenpiteet levykuvan ottamiseksi ”Test_srv1”-palvelimesta.



Kuva 13. Clonezillan aloitusnäkymä. Navigoin suoraan näkymän ylimmästä ”Clonezilla live” -kohdasta eteenpäin.

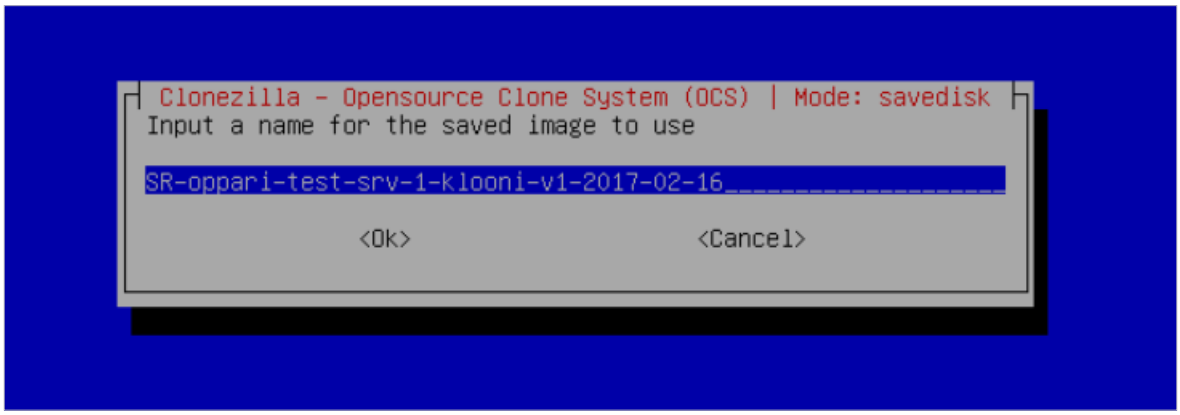
Aluksi tuli useita siniharmaataustaisia ikkunoita, joissa pyydettiin valitsemaan perusasetuksia – valittiin esimerkiksi käytettävä kieli (en_US.UTF-8 English), näppäimistölle tehtävät toimenpiteet (”don’t touch keymap”) ja kysyttiin haluaako käynnistää normaalisti Clonezillan vai mahdollisesti edetä shellin tai komentorivinäkymän kautta (valitsin ”Start Clonezilla”).

Clonezillan varsinaisen käynnistyksen jälkeen sama väriteema edelleen pysyi, mutta kysymykset muuttuivat enemmän sovelluksen ydintoimintoja koskevaksi. Seuraavassa ikkunassa kysyttiin haluaako kopioida kovalevyn image-tiedostoon vai kloonata suoraan toiselle kovalevylle – valitsin ”device-image” eli kopioinnin levykuvaan.

Lisäksi pyydettiin valitsemaan myöhemmin luotavan imagen tallennuskohde, vaihtoehtoja oli lähes kymmenen (esimerkiksi paikallinen tallennusväline ja NFS- sekä SSH-palvelin). Valitsin käytettävän paikallista tallennusvälinettä eli USB-muistia tässä tapauksessa ja lopuksi vielä mounttasin käytettävän USB-muistin Test_srv1:lle.

Seuraavaksi tuli näkyviin virtuaalipalvelimella käytettävissä olevat levyt sekä valittiin luotavan imagen tallentamista varten kohdelevy ja -polku. Näiden vaiheiden jälkeen valisin vielä käytettävän beginner modea eli aloittelijalle suunnattua levykuvan luomistapaa, joka käyttää osittain oletusasetuksia imagen luomiseen.

Muissa ikkunassa pyydettiin vielä valitsemaan tehdäänkö koko levystä vai ainoastaan tietystä osiosta image – valitsin koko levyn eli vaihtoehdon ”save local disk as an image”. Lisäksi asetin luotavalle imagelle nimeksi ”SR-oppari-test-srv-1-klooni-v1-2017-02-16” ja määritin kopioitavan levyn lähteeksi testipalvelimen virtuaalisen kovalevyn.

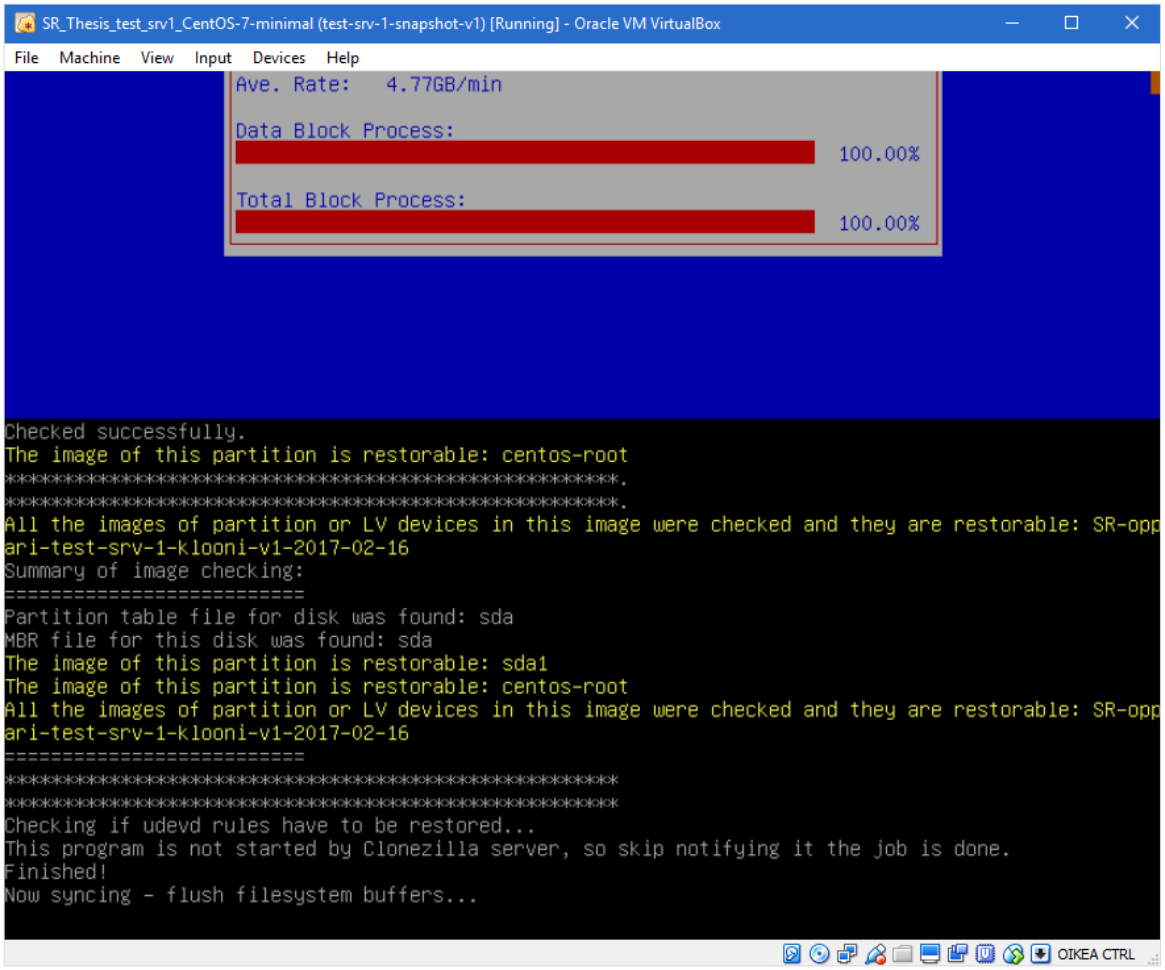


Kuva 14. Nimen valitseminen luotavalle levyimagelle.

Clonezillassa on valintaikkunoita aika reipas määrä, joten vielä kysyttiin haluaako tarkistuttaa lähdelevyn tiedostojärjestelmän tai imagen luonnin jälkeen itse imagen sekä haluaako suojata imagen salauksella. Valitsin kolmesta kysytystä toimenpiteestä ainoastaan imagen eheyden tarkistusten suoritettavaksi kopioinnin jälkeen.

Nyt oltiin jo lähellä päästä toteuttamaan levykuvan luonti. Enää piti valita haluaako palvelimen boottaavan automaattisesti imagen luomisen jälkeen ja varmistaa levykuvan luomisen aloitus. Annoin luvan suorittaa kopioinnin ja itse levykuvan luonti sujui onnistuneesti käytännössä muutamassa sekunnissa – sisältöä kun testipalvelimella on olematon määrä ja lisäksi kyseessä on kevyt CentOS minimal -asennus.

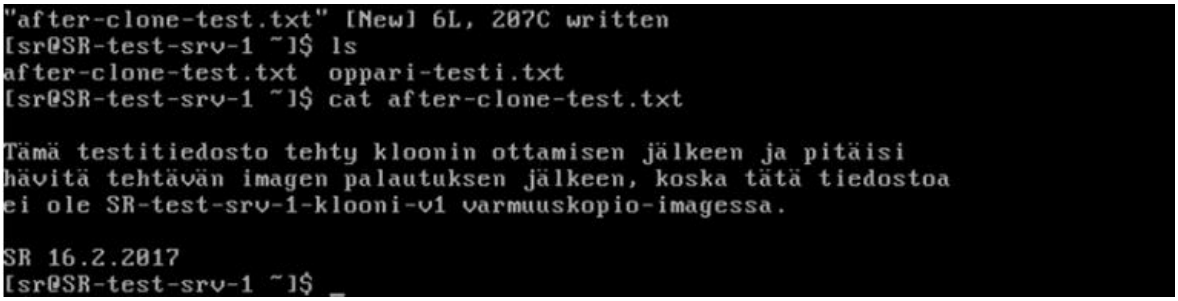
Kuva 15. Kloonaus valmistunut ja levykuva tallennettu onnistuneesti USB-muistille. Lopuksi Clonezilla suoritti vielä valitsemani imagen tarkistuksen – levykuva oli täysin eheä ja palauttamiskelpoinen Clonezillan mukaan. Itse levykuva löytyi määritysten mukaisesti USB-muistilta.



Kuva 16. Levykuvan eheys tarkistettu ja täten käyttökelpoisuus varmistettu. Seuraavaksi vuorossa palautuksen testaus kyseisellä levyimagella käyttäen samaa testipalvelinta.

5.2 Levykuvan palautus paikallisesti

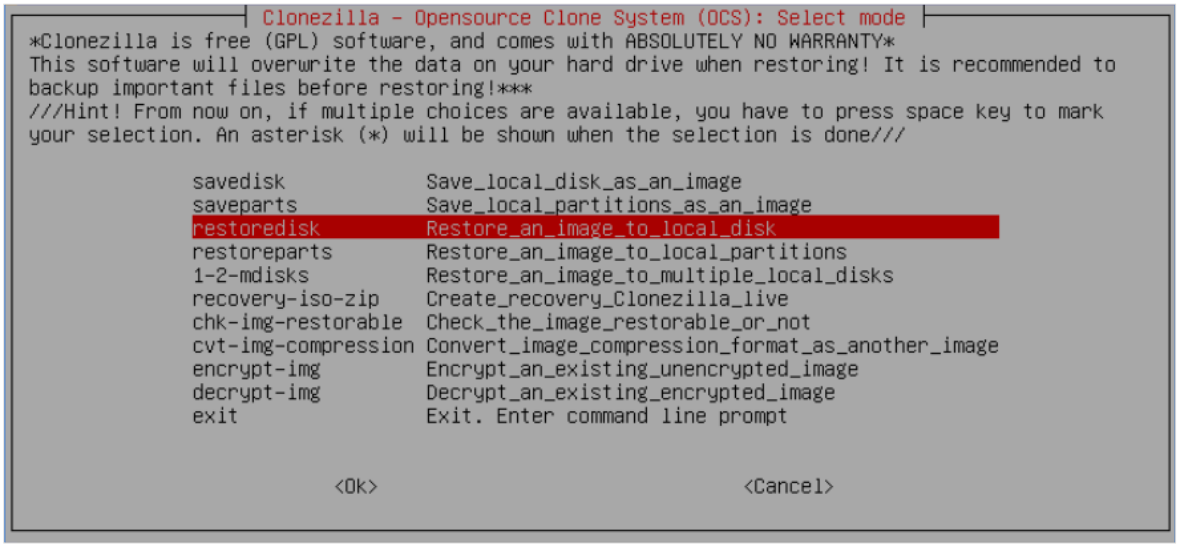
Heti perään oli vuorossa juuri otetun levyimagen palautus samalle ”Test_srv1”-palvelimelle. Ennen levykuvan ottamista loin ”oppari-testi.txt” -nimisen tiedoston, jonka pitäisi siis olla tallessa myös juuri otetussa levyimagessa. Ennen levykuvan palauttamista loin kuitenkin vielä Test_srv1 -palvelimelle uuden ”after-clone-test.txt” -nimisen tiedoston, jonka vastaavasti pitäisi hävitä levykuvan palauttamisen jälkeen koska kyseistä tiedostoa ei ole ollut olemassa aiemmin suoritetun kloonaamisen aikoihin.



Kuva 17. Levykuvan ottamisen jälkeen luodun ”after-clone-test.txt” -tiedoston sisältö cat-komennolla. ls-komennolla näkyy myös ennen levykuvaa luotu ”oppari-testi.txt”-tiedosto.

Toisen testitiedoston luomisen jälkeen aloitin välittömästi toimenpiteet levykuvan palauttamiseksi Clonezillalla. Vaiheet olivat levykuvan luomisen kanssa aikalailla samankaltaiset: valittiin esimerkiksi miltä ja minkä tyyppiseltä laitteelta tai palvelimelta levykuva luetaan ja tietty itse levykuvan polku sekä beginner moden käyttäminen.

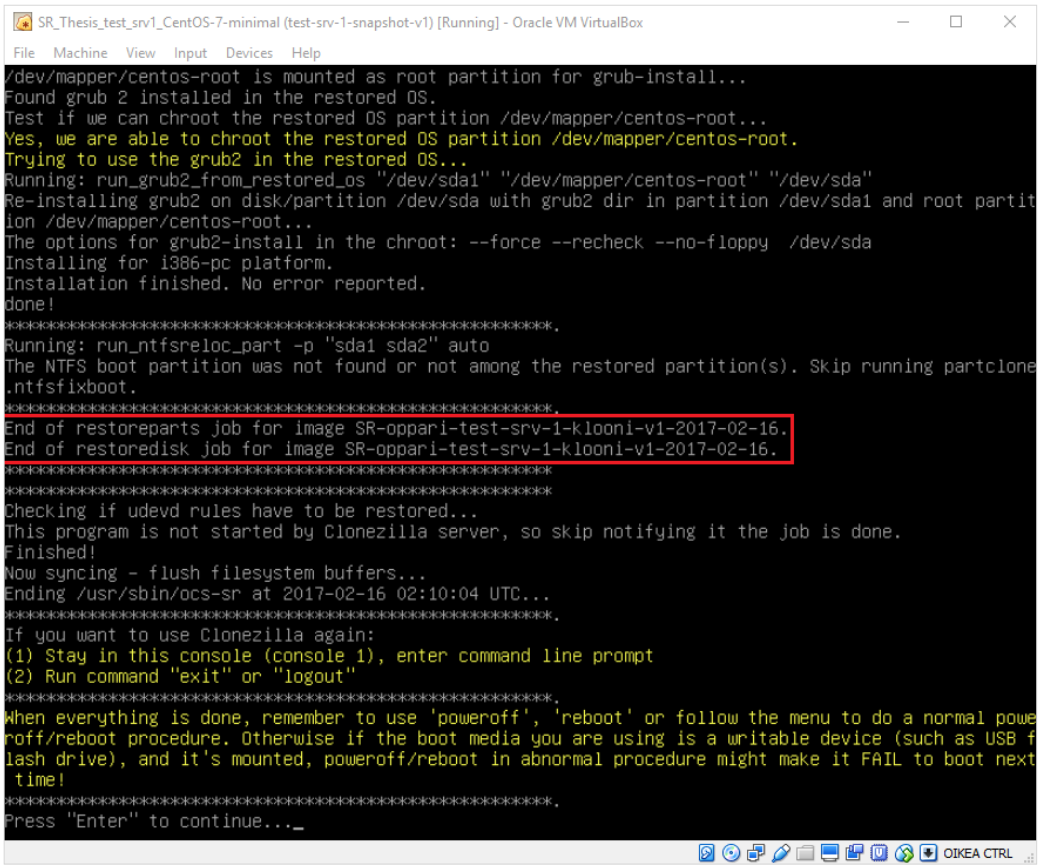
Tämän jälkeen erona valinnassa oli palautuksen suorittamisen valitseminen aiemman ”savedisk” -valinnan sijaan eli nyt valittiin ”restoredisk - restore an image to local disk”.



Kuva 18. Clonezillan ”beginner mode” tarjoaa kuvanmukaiset vaihtoehdot levykuvan ottamiseen tai palauttamiseen käyttäessä ”device-image” -kloonaustapaa.

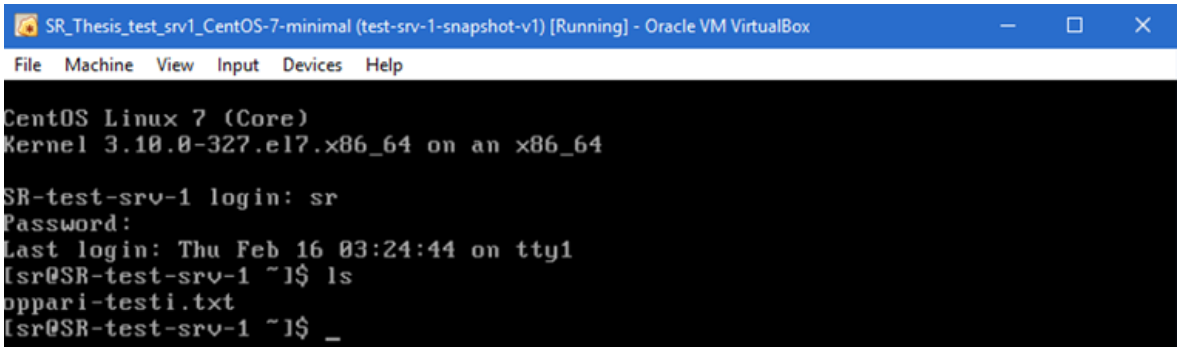
Seuraavaksi kysyttiin palautettavaa imagea ja valitsin aiemmin luodun levykuvan, joka oli myös ainoa tarjolla ollut vaihtoehto. Lisäksi kysyttiin kohdelevyä, jolle imagen sisältö ajetaan. Tässä vaiheessa on elintärkeää valita oikea vaihtoehto mikäli kovalevyjä tai levyn osioita on listattuna useampi, koska kaikki data valitulta kohdelevyltä poistetaan ja palautettavan imagen data korvaa täysin aiemman sisällön.

Ennen levykuvan palauttamista valitsin vielä tehtävän imagen eheyden tarkistuksen ja tarkistuksen tulos oli tismalleen sama kuin levykuvan ottamisen jälkeen. Seuraavaksi olikin jo vuorossa varmennus palautuksen suorittamisesta ja Clonezilla toisti ”oletko varma?” kysymyksen käyttäjälle kahteen kertaan. Vastasin myöntävästi ja levyn palautus käynnistettiin.



Kuva 19. Levyn onnistuneessa palauttamisessa ei kestänyt pariakymmentä sekuntia enempää – Clonezilla myös pystyi käyttämään onnistuneesti palvelimen bootloaderia. Kuvaan merkitty punaisella onnistuneesta levyn palautuksesta kertovat kohdat.

Lopuksi oli jäljellä enää palvelimen boottaus ja OS:n toimivuuden varmistaminen käytännössä. Palvelin käynnistyi entiseen tapaan ripeästi ja oli vielä aika tarkastaa onko juuressa vielä tallessa ennen levykuvan ottamista luotu ”oppari-testi.txt” -tiedosto ja onko levykuvan ottamisen jälkeen samaan polkuun luotu ”after-clone-test.txt” -tiedosto vastaavasti hävinnyt.



Kuva 20. Ennen levykuvan ottamista luotu ”oppari-testi.txt” oli tallessa, mutta levykuvan ottamisen jälkeen luotu ”after-clone-test.txt” oli odotetusti hävinnyt palautuksen suorittamisen jälkeen.

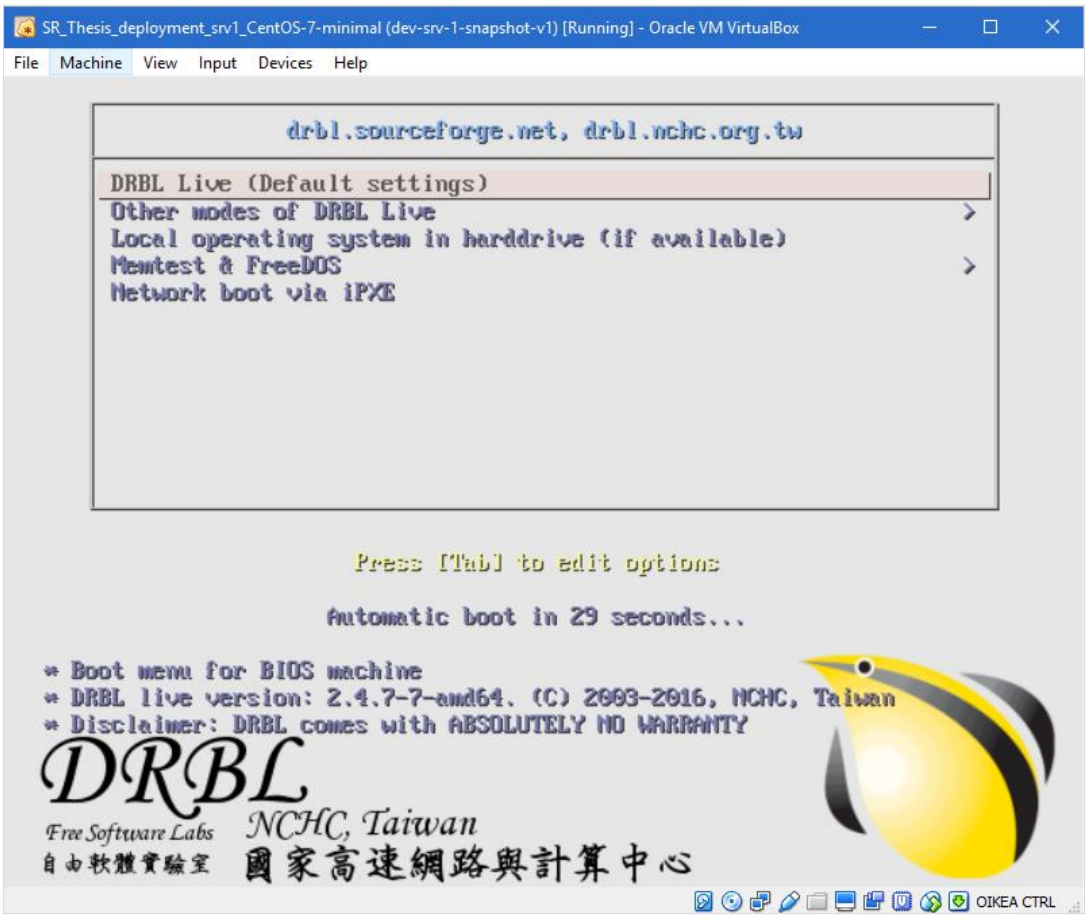
Lopuksi testasin vielä verkkoyhteyden toimivuuden – pingi kulki entiseen tapaan palvelimella, jonka myötä varmistui palvelimen palautuksen onnistuneen ilman ongelmia. Näin oli lokaalisti tehty palvelimen palautus käyttäen Clonezillan perusversiota livenä eli asentamatta kyseistä ohjelmaa pysyvästi itse palvelimelle.

5.3 Levykuvan ajo verkon välityksellä

Seuraavaksi oli vuorossa vielä Clonezillan server editionin live-testaus. Käytännössä siis tämä tarkoittaa DRBL eli ”Diskless Remote Boot in Linux” -ympäristön (joka sisältää Clonezilla SE:n) live-version ajamista Deployment_srv1 -palvelimella. Clonezillan server edition versiolla olisi teoriassa mahdollista ajaa samanaikaisesti jopa yli 40:lle verkon palvelimelle Clonezillan kloonaus. (DRBL 2017.)

Tässä testitapauksessa käytän Deployment_srv1 -hallintapalvelimella Clonezilla server editionia kloonaten 5.1 kappaleessa luodun Test_srv1:n levykuvan lähiverkon välityksellä Test_srv2 palvelimelle. Aiempaa kokemusta Clonezilla SE:n käytöstä ei ole, joten on mielenkiintoista nähdä kuinka kloonaus onnistuu. Kloonauksen jälkeen Test_srv:2:lla pitäisi vielä vaihtaa ainakin hostname, generoida palvelimen kloonatut UUID:t yksilölliseksi sekä konfiguroida verkkoasetukset takaisin alkuperäiseksi.

Alkuun latasin DRBL-liven .iso-tiedoston koneelle ja mounttasin kyseisen imagen Deployment_srv1 -palvelimen virtuaaliseen levyasemaan. Mounttauksen jälkeen käynnistin Deployment_srv1:n uudestaan ja valitsin boottauksen kyseiseltä levyasemalta.

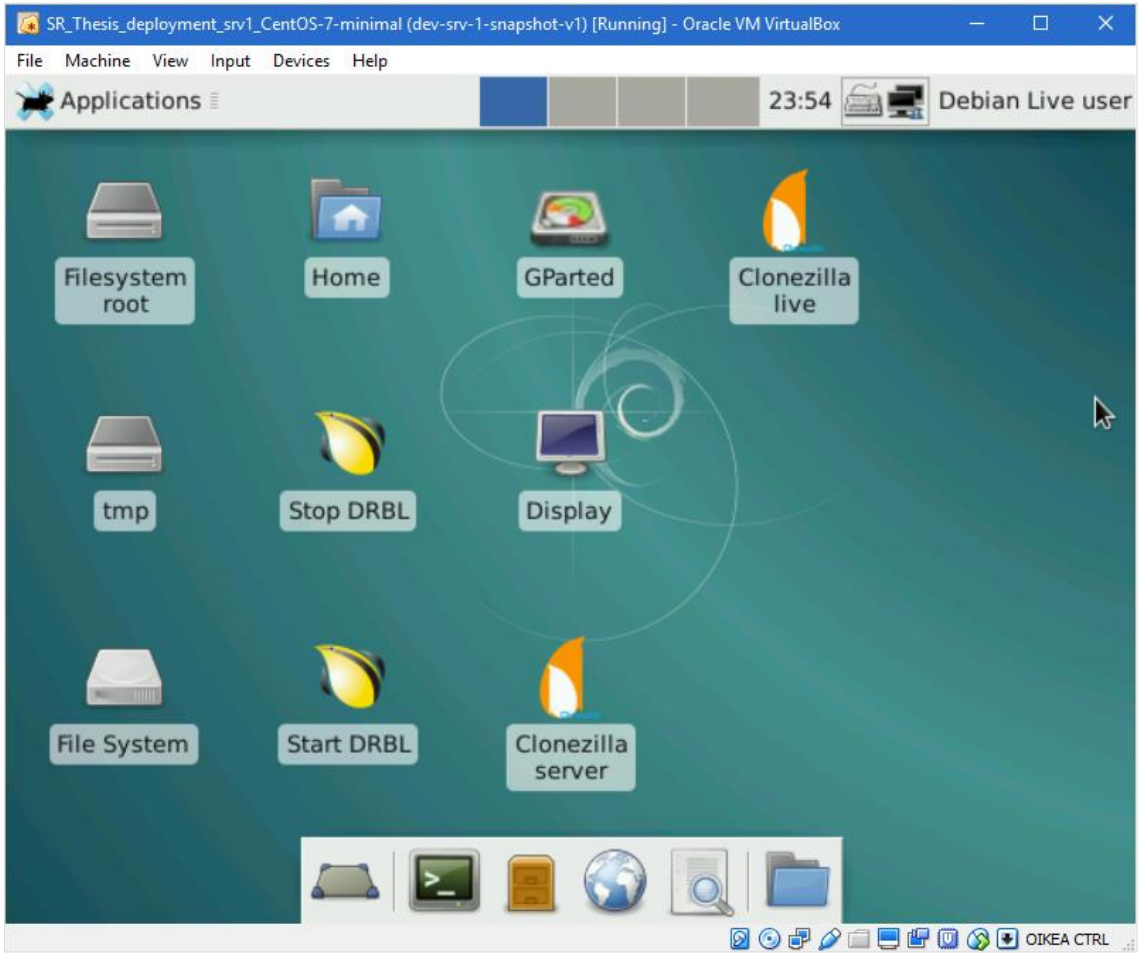


Kuva 21. DRBL-ympäristön avausnäkymä. Etenin käyttämään DRBL-liveä.

DRBL-liven asetukset piti määrittää aluksi. Valisin samoja peruasetuksia kielestä lähtien kuin aiemmin testatun tavallisen Clonezilla-liven kanssa. Valitsin myös käytettävän eth0-interfacen kautta staattista IP:tä, jonka lisäksi määritin verkon maskin, nimipalvelimen ja oletusyhdyskäytävän.

Itse kloonausasetuksissa valitsin käytettävän paikallista tallennusvälinettä (”local dev”) levykuvien tallennukseen. Lisäksi valitsin Deployment_srv1:lle mountatun USB-muistin olevan levykuvien varsinainen tallennuspaikka ja tarkan tallennuspolun olevan kyseisen USB-muistin juuri.

Seuraavaksi pyydettiin vielä varmistamaan pariin otteeseen enteriä painamalla valitut asetukset ja DRBL-ympäristön varsinainen konfigurointi alkoi. Tässä konfigurointivaiheessa esimerkiksi DRBL:n palveluita (services) käynnistettiin, luotiin käyttäjätunnuksia sekä valittiin käytettäviä Clonezilla clienttejä.

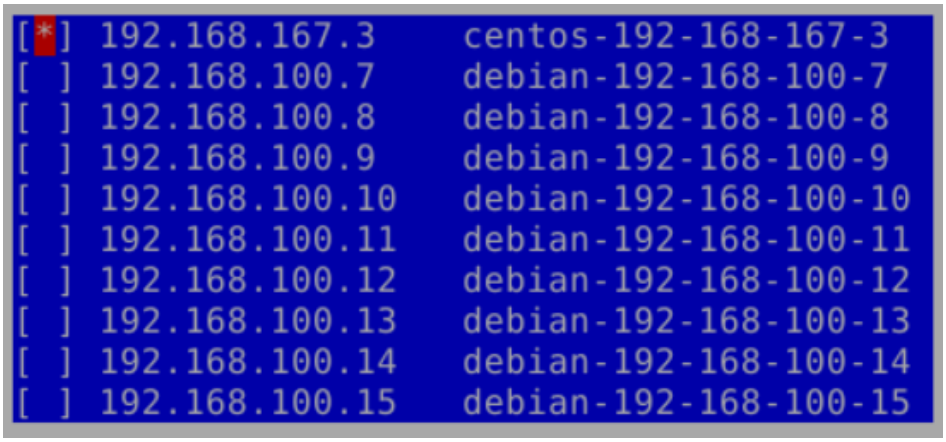


Kuva 22. DRBL:n konfiguroinnin jälkeen avautui Debian-pohjainen käyttöliittymä.

Tässä välissä siirryin käyttämään Test_srv2 -palvelinta, jolle oli kohtapuolin tarkoitus ajaa aiemmin luotu Test_srv1:n levykuva verkon välityksellä. Loin Test_srv2:lla testitiedoston ”test-srv-2-filu.txt”, jonka kuuluisi hävitä Test_srv1:n levykuvan kloonauksen valmistuessa Test_srv2:lle. Testitiedoston luomisen jälkeen sammutin Test_srv2:n hetkeksi odottamaan tulevaa levykuvan ajoa nykyisen asennuksen päälle.

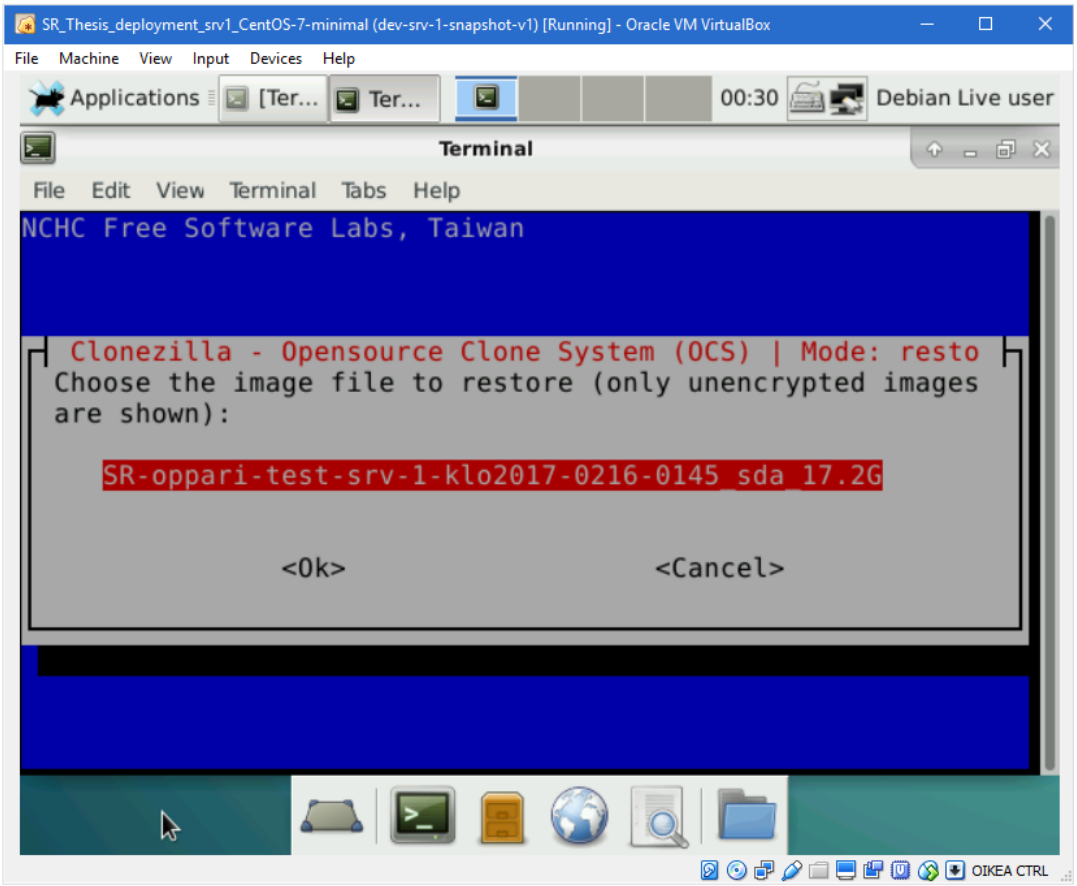
Seuraavaksi oli vuorossa toteuttaa levykuvan ajo Clonezilla SE -hallintapalvelimelta Test_srv2:lle. Siispä palasin takaisin Deployment_srv1 -palvelimelle ja lisäsin DRBL:n client-listaan Test_srv2:n IP:n 192.168.167.3. Tämä hoitui yksinkertaisesti konfiguroimalla polusta *”etc/drbl”* löytyvää ”IP_HOST_TABLE” -tiedostoa, jonne lisäsin Test_srv2:n IP:n ja hostin tunnisteeksi vielä ”centos-192-167-3”.

Tämän jälkeen klikkasin Clonezillan server -kuvaketta ja näin alkoi varsinaisen levykuvan palautuksen asetuksien valinta. Alkuun valitsin ”part select client(s) by IP or MAC address” ja “set mode by clients IP address list” -asetukset, joista jälkimmäisessä valitsin vielä erikseen tähdellä Test_srv2 -hostin IP:n alle olevan kuvan mukaisesti.



Kuva 23. Valitsin listalta Test_srv2:n IP:n. Muut listalla näkyvät palvelimet ovat DRBL:n automaattisesti luomia pseudoclienttejä, joita en poistanut erikseen, kun tarkoitus oli vain ainoastaan testata Clonezilla Server Editionin liveversiota.

Seuraavissa ikkunoissa valitsin tuttujen perusasetuksien lisäksi palautuksen tehtävän valitulle clientille kokonaisesta levystä, imageksi aiemmin luodun ”Test_srv1”:n kopion ja käytettäväksi multicasting-lähetystapaa. Lisäksi asetin clienttien määräksi yhden – enempään ei tässä testissä ollut tarvetta. Varmistuksien jälkeen tarvittavat kloonauksen esitoimenpiteet tehtiin Clonezilla-hallintapalvelimella ja oli aika siirtyä Test_srv2:n puolelle.

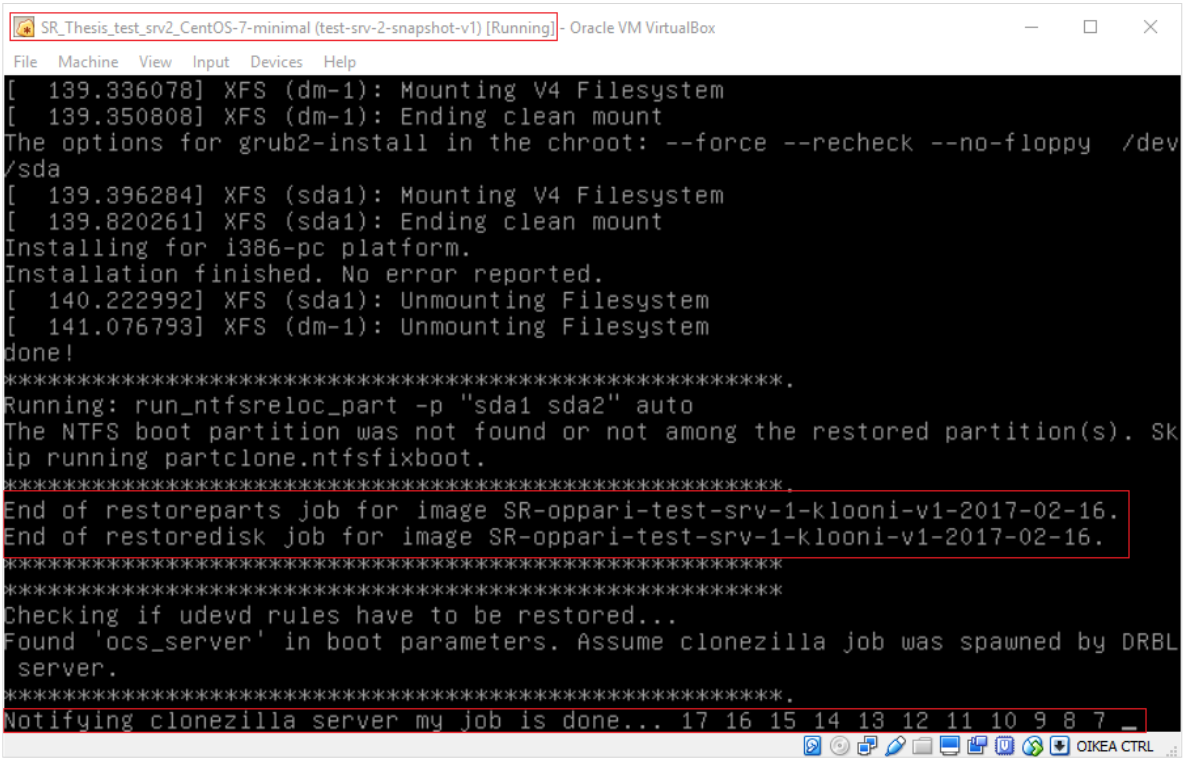


Kuva 24. Ajettavan levykuvan valitseminen Deployment_srv1 -palvelimella. Valittu levykuva on sama kuin kappaleessa 5.1 luotu Test_srv1 -palvelimen levykuva.

Vielä oli jäljellä viimeinen vaihe eli itse Test_srv1:n levykuvan ajo Test_srv2:lle lähiverkon välityksellä. Siispä käynnistin Test_srv2:n painaen heti perään F12:sta, jotta pääsin valitsemaan palvelimen käynnistyksen tapahtuvan lähiverkon (LAN) kautta. Tämän jälkeen parin sekunnin ajan latautui mustalla ruudulla tietoja havaitusta Clonezillan verkkobootista, jonka jälkeen avautui DRBL-ympäristön näkymä.

DRBL:n ikkunasta löytyikin odotetusti aiemmin ”Deployment_srv1”:llä konfiguroitu ”Test_srv1”:n levykuvan palautus käyttäen multicastingia. Valitsin kyseisen vaihtoehdon, jonka jälkeen ruudulla näkyi asetuksien lataaminen Clonezillan hallintapalvelimelta noin puolen minuutin ajan. Tämän jälkeen alkoi varsinainen ”Test-srv1”-levykuvan ajo Deployment_srv1 -palvelimelta Test_srv2:n kovalevylle.

Kuva 25. Test_srv2 löysi verkkoboottia käyttämällä Deployment_srv1 -palvelimella konfiguroidun Test_srv1:n levykuvan ajamisen multicastingia käyttäen.



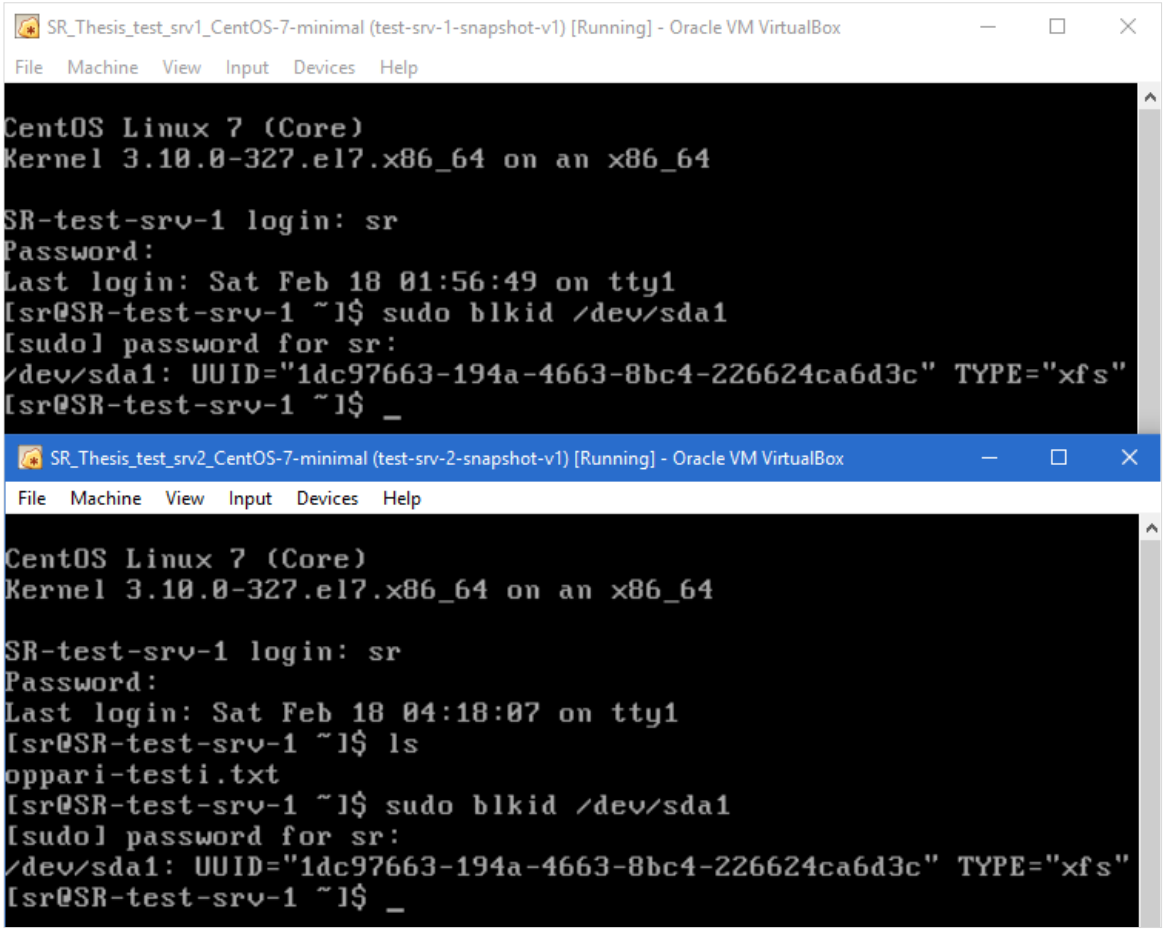
Kuva 26. Tilanne Test_srv2:lla levykuvan ajon valmistuttua.

Kaikkien vaiheiden suorittamisessa kesti kokonaisuudessaan muutamia minuutteja, jonka jälkeen tuli ilmoitus palautuksen onnistumisesta. Lopulta Test_srv2 -palvelimelta lähti ilmoitus Clonezilla-hallintapalvelimelle (Deployment_srv1:lle) työn eli jobin valmistumisesta.

”*Cloned successfully.”*
”*Finished restoring image SR-oppari-test-srv-1-klooni-v1-2017-02-16 to /dev/centos/root*.”
“*End of restoreparts job for image SR-oppari-test-srv-1-klooni-v1-2017-02-16*.”
“*End of restoredisk job for image SR-oppari-test-srv-1-klooni-v1-2017-02-16*.”
“*Notifying Clonezilla server my job is done…*”

Test_srv2 käynnistyi lopulta uudestaan normaalisti ja ennen levykuvan ajoa palvelimelle luotu ”test-srv-2-filu.txt” -tekstitiedosto oli hävinnyt ja tilalle oli tullut ennen Test_srv1:n imagen ottamista luotu ”oppari-testi.txt” mikä osaltaan todisti Test_srv1:n levykuvan kopioinnin onnistuneen Test_srv2:lle.

Lisäksi Test_srv2:lla oli vaihtunut hostnameksi SR-test-srv-1.oppari, verkkoasetukset olivat identtiset Test_srv1:n kanssa ja ennen kaikkea myös tiedostojärjestelmien UUID:t olivat nyt identtisiä Test_srv1:llä ja Test_srv2:lla.



Kuva 27. Test_srv1:n ja Test_srv2:n UUID:t vertailtavina.

Test_srv2 -palvelimen asetukset sai takaisin kuntoon muutamalla edellä tehtävällä toimenpiteellä. Aluksi muokkasin staattisen hostnamen alkuperäiseksi komennolla *”sudo hostnamectl set-hostname SR-test-srv-2”*. Tämän jälkeen generoin molempia verkkointerfaceja varten uudet identtiset UUID:t (Universally Unique Identifiers) komennoilla *”uuidgen enp0s3”* ja *”uuidgen enp0s8”* sekä otin generoidut UUID:t talteen. (PCMag 2016d.)

Seuraavaksi tuli muokata molempien verkkointerfacien asetukset ennalleen aiemman verkkosuunnitelman mukaisesti – tämä tehtiin polun ”*cd/etc/network-scripts*” tiedostoissa ”ifcfg-enp0s3” ja ”ifcfg-enp0s8”. Tarvittavat muutokset interface-tiedostoissa olivat ainoastaan aiemmin generoidun UUID:n, staattisen IP:n ja verkkokortin MAC-osoitteen vaihtaminen. Muutoksien jälkeen käynnistin network-servicen uudestaan sekä pysäytin ja disabloin konflikteja aiheuttaneen NetworkManagerin.

Lopuksi sain vielä tiedostojärjestelmän UUID:n muokattua unmounttaamalla aluksi sda1-osion komennolla *”umount /dev/sda1”* ja syöttämällä perään komennon *”xfs_admin -U*

generate /dev/sda1". Tämän jälkeen mounttasin takaisin kyseisen osion ja uuden generoidun UUID:n muokkasin vielä ”/etc/fstab” ja ”/etc/grub2.cfg” -tiedostoihin. Jälkimmäisen tiedoston muokkaus, jotta palvelimen käynnistyksen aikana ei tulisi *"no such device"* -virheilmoitusta.

Tämä tiedostojärjestelmän UUID:n korjaustapani ei ollut aivan täysin parhaan käytännön mukainen, mutta käypä korjaustapa tässä tilanteessa, kun tarkoitukseni oli vain demota kuinka saada kloonattu palvelin muokatulla UUID:llä ripeästi käyttökuntoiseksi enkä tule enempää näitä virtuaalisia testipalvelimia käyttämään opinnäytetyöni jälkeen. Oikeassa tuotantoympäristössä tulisi luonnollisesti GRUB2-bootloaderin korjaustoimenpiteet suorittaa suositeltujen käytäntöjen mukaan alusta loppuun.

Nyt oli siis Clonezillan molempien live-versioiden (tavallinen ja server edition) testaukset viety onnistuneesti maaliin asti. Clonezilla sisältää sen verran laajasti ominaisuuksia, että silti jäi vielä opeteltavaa tulevaisuuteen, mutta perustoiminnot tuli omaksuttua kohtuullisen hyvin. Clonezilla on tehtyjen testien perusteella oikein käytettynä varsin tehokas työkalu, joka soveltuu useissa käyttötapauksissa Linux-palvelimien ja -työasemien kloonaamiseen ainakin pienemmissä ympäristöissä.

6 Palvelimen käyttöönotto Kickstart-tiedostolla

Tämä kappale sisältää kickstart-tiedoston suunnittelun, luomisen ja testiasennuksen. Kickstart-tiedoston määritelmä on esitelty kappaleessa 2.5.

6.1 Kickstartin suunnittelu ja toteutus

Alkuun oli vuorossa teorian opettelua kuinka luoda ja toteuttaa vaatimukset täyttävä kickstart-tiedosto sekä vaatimusmäärittelyn luominen kickstartille. Verkosta hakemalla löytyikin runsaasti hyödyllistä tietoa kickstartin syntaksin mahdollistamista ominaisuuksista, erinäisistä konfigurointitavoista ja myös Red Hatin virallinen dokumentaatio oli jo itsessään varsin kattava ja yksilökohtainen. (Red Hat 2017b.)

Taulukko 10. Kickstart-tiedoston vaatimusmäärittely.

Vaatimukset kickstart-tiedostolle
CentOS 7 -käyttöjärjestelmän minimal-asennus verkon tai levyn kautta.
Levyn osiointi /boot (ext4), / (ext4) ja swap -osioihin.
Bootloaderin konfigurointi.
Konfiguroidut verkkoasetukset staattisella IP-osoitteella.
Palomuurin konfigurointi sisältäen porttien 20, 22, 23, 80 ja 443 avaamisen.
Chronyd ja postfix -palveluiden disablointi.
Sovelluspaketit alla olevan taulukon mukaisesti.
Base-, minimal- ja EPEL-repositoryjen määrittäminen.
”sr”-nimisen tunnuksen luonti sudo-oikeuksilla.
englanninkielinen OS, suomenkielinen näppäimistö ja Suomen aikavyöhyke.
Selinuxin disablointi.
Salasanakäytäntöjen määrittäminen.

Taulukko 11. Kickstart -tiedoston avulla asennettavat paketit.

Asennettavat paketit
epel-release, wget, iotop, htop, lsof, nmap, net-tools, openssh-clients, telnet, rkhunter.

Kattavan tiedonhaun ja itsensä perehdyttämisen jälkeen oli vuorossa varsinaisen kickstart-tiedoston kehittäminen. Aloitin oman kickstart-tiedoston luomisen tarkastelemalla jo olemassa olevaa, testipalvelimelle automaattisesti generoitua kickstart-tiedostoa. Konfiguroidut asetukset olivat tässä kickstartissa varsin yksinkertaiset ja useimpien rivien komentojen tarkoituksen ymmärsi välittömästi tiedostoa lukemalla.

Tätä kickstart-pohjaa tarkastelemalla oli tarkoitus ensinnäkin oppia käytännön tasolla kickstartin rakenne ja jatkojalostaa tiedosto omien vaatimuksien mukaiseksi suoritettavien konfigurointien ja testauksien avulla. Lisäksi lopputuloksena syntyvää kickstart-tiedostoa tulisi olla mahdollista hyödyntää soveltaen muissa tulevaisuuden käyttötapauksissa.

Alkuun törmäsin pariin ongelmaan kickstart-tiedoston testauksissa, jotka johtuivat yksinkertaisesti puutteellisesta konfiguroinnista. Lopulta löydettyäni ensimmäiset ongelmakohdat ja lisättyäni puuttuneet rivit (esim. levyn alustuksen) sain lopulta ensimmäisen kerran kickstart-tiedoston asentumaan onnistuneesti uudelleen testitarkoituksessa luodulle Test_srv2 -virtuaalipalvelimelle.

Tämän jälkeen aloin tarkemmin käymään läpi mitä asetuksia tulee vielä lisätä kickstart-tiedostoon, jotta kaikki vaatimusmäärittelyssä listatut ominaisuudet saadaan sisällytettyä OS-asennukseen. Testasin vaihe vaiheelta valittujen ominaisuuksien toimivuutta jalostaen kickstart-tiedostoa valmiimmaksi ”trial and error” -menetelmän tapaan.
(Cambridge Dictionary 2017.)

Lopulta runsaan selvitystyön ja testauksen myötä sain valmiiksi testatusti toimivan Kickstart-tiedoston versioltaan 1.0. Jäljellä oli siis vielä kickstartin vieminen Deployment_srv2 -webpalvelimelle ja test-srv2 -palvelimella suoritettava käyttöjärjestelmän asennus lataamalla luotu kickstart-tiedosto lähiverkon välityksellä Deployment_srv2:lta.

Alla kokonaisuudessaan opinnäytetyötä varten luotu kickstart-tiedosto, jossa jokaiselta riviltä löytyy selitykset asennuksessa tapahtuvasta toiminnasta englanniksi ja suomeksi. Oletuksena kickstartin konfiguraatiossa on tekstipohjainen – verkon välityksellä suoritettava käyttöjärjestelmäasennus. Seuraavassa kappaleessa demoan vielä mainitun kickstartin testiasennuksen Test_srv2 -palvelimella.

Taulukko 12. Kickstart-tiedoston versio 1.0 kokonaisuudessaan.

Kickstart v.1.0

```
## Kickstart installation file created by Sami Rita
## Kickstart asennustiedosto, tekijä Sami Rita
## Created for testing purposes | Tehty testitarkoitusta varten
## v.1.0

# Install or Upgrade OS | Asenna vai päivitä käyttöjärjestelmä
install
#upgrade

# Install OS via CD-rom or internet | Asenna OS CD-levyn tai verkon välityksellä
#cdrom
url --url="http://mirror.centos.org/centos/7/os/x86_64/”

# Accept RHEL EULA | Hyväksy Red Hatin käyttäjäehdot
eula --agreed

# Don't configure the X Window System | Älä konfigruoi X Window -järjestelmää.
skipx

# Use graphical or text install | Käytä graafista tai tekstipohjaista asennusta
#graphical
text

# Run the Setup Agent on first boot | Käynnistä asennus bootin jälkeen
firstboot --enable

# Define used disks | Määritä käytettävät levyt
ignoredisk --only-use=sda

# Keyboard layouts | Näppäimistön kieli
keyboard fi

# System language | Käyttöjärjestelmän kieli
lang en_US.UTF-8

# Network information | Verkkokonfiguraatiot
network --bootproto=static --device=enp0s3 --onboot=on --ip=192.168.10.13 --
netmask=255.255.255.0 --gateway=192.168.10.1 --nameserver=8.8.8.8 --hostname=SR-kickstart-
testi.oppari

# Firewall configuration | Palomuurin asetukset
firewall --enabled --port=20:tcp,22:tcp,23:tcp,80:tcp,443:tcp

# Root-user password | Juurikäyttäjän salasana
rootpw --plaintext centos

# System services | järjestelmän palvelut
services --disabled=chronyd, postfix

# System timezone | Järjestelmän aika
timezone Europe/Helsinki --isUtc --nontp

# Create administrator account "sr" with password "centos"
# Luo tunnus "sr" salasanalla "centos" ja ylläpitäjän oikeuksilla.
user --name=sr --groups=wheel  --password=centos --plaintext

# System bootloader configuration | Järjestelmän bootloader konfiguraatio
bootloader --append="crashkernel=auto" --location=mbr --boot-drive=sda

# Partitions | Levyn osiointi
```

```
part /boot --fstype=ext4 --size=500
part / --fstype=ext4 --size=6500
part swap --fstype=swap --size=1000

# Clear and initialize old or invalid Linux-partitions/disks. | Poista ja alusta vanhat sekä virheelliset
Linux-levyosiot.
clearpart --linux --initlabel
zerombr

# SELinux configuration | SELinuxin asetukset
selinux --disabled

# Other repositorys | Muut repot
repo --name="EPEL" --baseurl="http://download.fedoraproject.org/pub/epel/7/$basearch"

# Reboot system after installing OS is ready | Uudelleenkäynnistä palvelin asennuksen
valmistuttua
reboot

# Packages | pakettiosio
%packages
@^minimal
@core
epel-release
net-tools
wget
iotop
htop
telnet
nmap
rkhunter
openssh-clients
lsof
%end

# Enable Kdump feature | Kdumpin käyttö
%addon com_redhat_kdump --enable --reserve-mb='auto'
%end

# Password policy | Salasanakäytännöt
%anaconda
pwpolicy root --minlen=6 --minquality=50 --notstrict --nochanges --notempty
pwpolicy user --minlen=6 --minquality=50 --notstrict --nochanges --notempty
pwpolicy luks --minlen=6 --minquality=50 --notstrict --nochanges --notempty
%end

# Run Post-Configuration Scripts | asennuksen jälkeiset skriptit
%post
yum -y update
echo "Server installed. Palvelin asennettu. t.SR"
%end
```

6.2 Käyttöjärjestelmäasennus Kickstartilla

Tässä kappaleessa asennan alkuun Deployment_srv2:lle Apache web-palvelimen, jonka jälkeen tallennan samalle palvelimelle aiemmin luodun kickstart-tiedoston ja konfiguroin kyseisen tiedoston käyttöoikeudet kuntoon, jotta sitä voi hyödyntää käyttöjärjestelmäasennuksessa uudelleen luotavalla Test_srv2 -palvelimella.

Lopuksi suoritan käyttöjärjestelmäasennuksen Test_srv2:lla CentOS-imagen ja Deployment_srv2:lla sijaitsevan kickstartin avulla sekä varmennan asennuksen valmistumisen jälkeen vielä Kickstartin sisältämien asetuksien tulleen voimaan.

Siispä ei muuta kuin käytännön palvelintöihin. Alkuun asennan Deployment_srv2:lle Apache web-palvelimen, jotta pystyn ladata Test_srv2 -palvelimen OS-asennuksessa kickstartin tältä web-palvelimelta. Asennus suoritetaan komennolla ”*yum install httpd*”, jonka jälkeen luon polkuun *”/var/www/html/”* tiedoston ks.cfg, johon liitän aiemmin tehdyn kickstart-tiedoston sisällön ja tallennan kyseisen tiedoston.

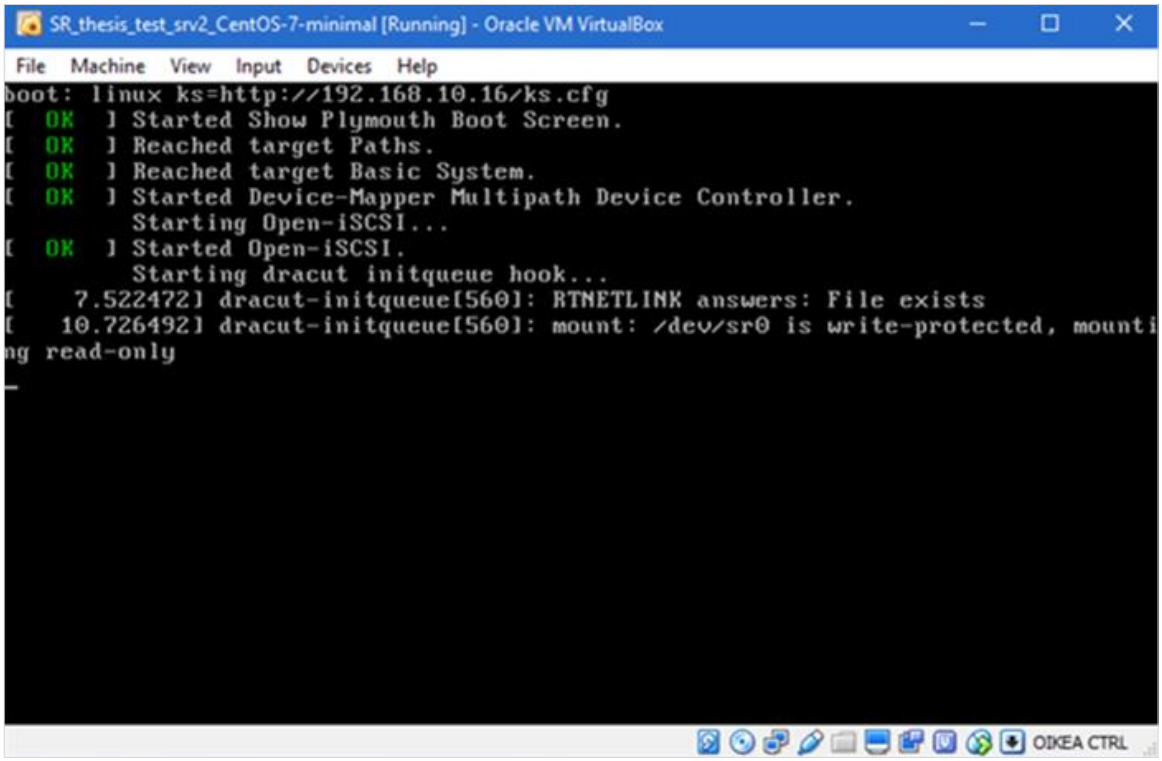
Lopuksi säädän chmod 755 -käyttöoikeudet ks.cfg kickstart-tiedostolle, jotta tiedostoa voi hyödyntää myöhemmin Test_srv2 -palvelimella ja käynnistän aiemmin asennetun apache web-palvelimen servicen uudestaan komennolla *”service httpd restart*” varmistaakseni asetusten ajantasaisuuden. Nyt olikin kaikki tarvittavat toimenpiteet tehty itse Deployment_srv2 -palvelimella.

```
[root@SR-dev-srv-2 html]# service httpd restart
Redirecting to /bin/systemctl restart  httpd.service
[root@SR-dev-srv-2 html]# service httpd status
Redirecting to /bin/systemctl status  httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
   Active: active (running) since Sat 2017-02-25 02:04:30 EET; 2s ago
     Docs: man:httpd(8)
           man:apachectl(8)
 Main PID: 2218 (httpd)
   Status: "Processing requests..."
   CGroup: /system.slice/httpd.service
           ├─2218 /usr/sbin/httpd -DFOREGROUND
           ├─2219 /usr/sbin/httpd -DFOREGROUND
           ├─2220 /usr/sbin/httpd -DFOREGROUND
           ├─2221 /usr/sbin/httpd -DFOREGROUND
           ├─2222 /usr/sbin/httpd -DFOREGROUND
           └─2223 /usr/sbin/httpd -DFOREGROUND

Feb 25 02:04:30 SR-dev-srv-2.oppari systemd[1]: Starting The Apache HTTP Server...
Feb 25 02:04:30 SR-dev-srv-2.oppari systemd[1]: Started The Apache HTTP Server.
[root@SR-dev-srv-2 html]#
```

Kuva 28. Apachen web-palvelun status Deployment_srv2:lla.

Seuraavaksi oli vuorossa varsinainen kickstartin asennus uudelleen luodulle Test_srv2 -palvelimelle. Aluksi lisäsin Test_srv2:n virtuaaliseen levyasemaan CentOS-imagen ja

käynnistin saman palvelimen uudestaan. Bootin jälkeen navigoin F12:lla boottivalikkoon ja valitsin boottauksen CD-aseman kautta, johon oli liitetty tämä CentOS-image. Tämän jälkeen CentOS:n normaali asennusikkuna tuli näkyviin ja painoin vielä esc-painiketta, jonka jälkeen pääsin ajamaan kickstartin käynnistyksen komentorivin kautta.

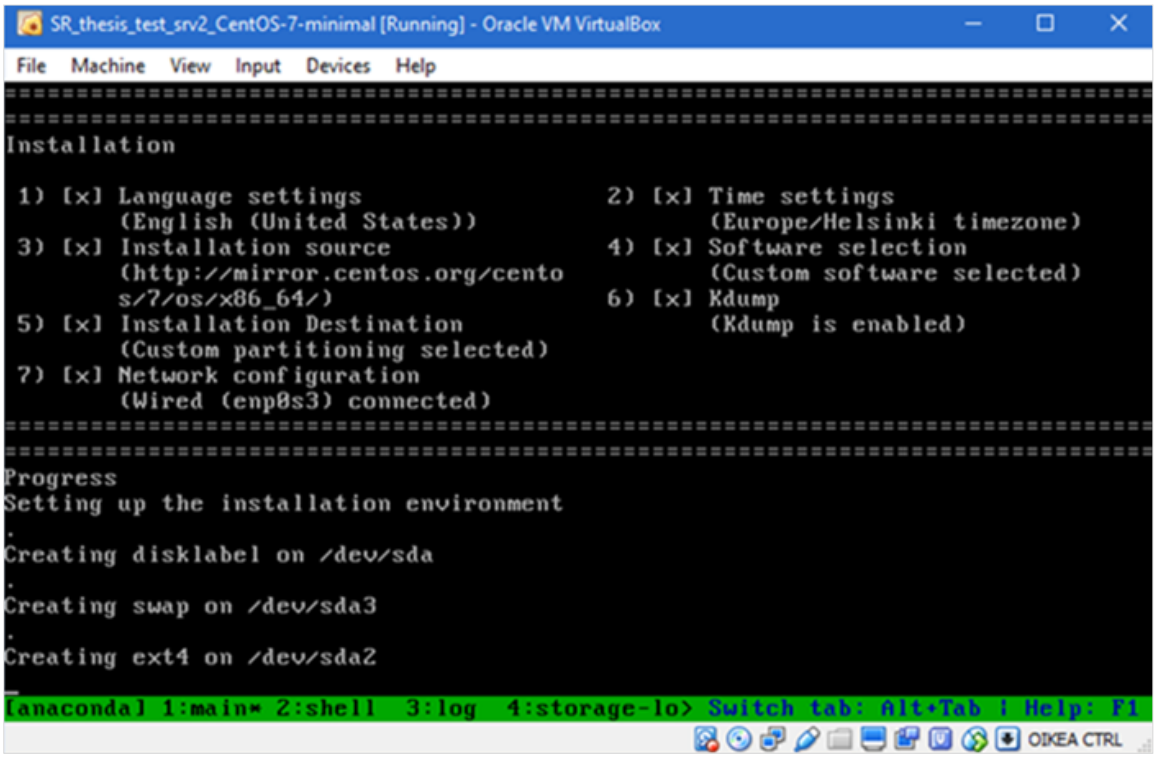


Kuva 29. Kickstartin ajo käynnistettiin komennolla ”linux ks=http://192.168.10.16/ks.cfg”.

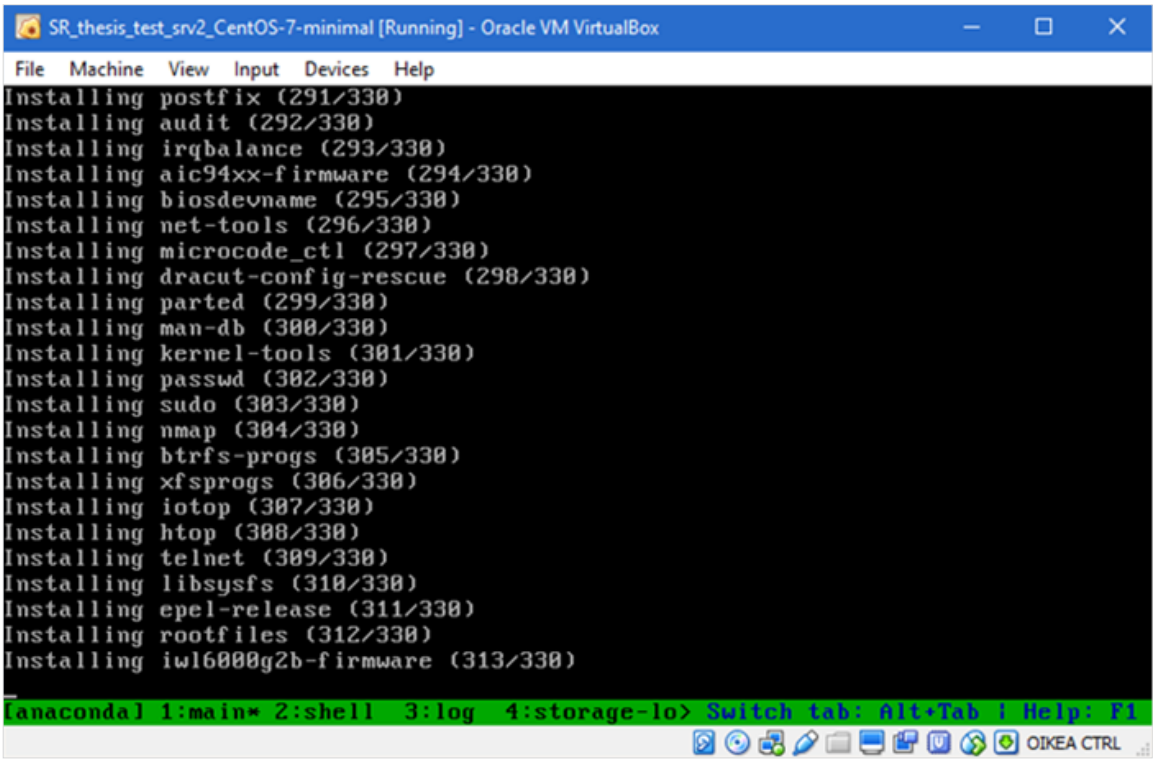
IP-osoite 192.167.10.16 on siis Deployment_srv2 -palvelimen, johon äsken asennettiin Apachen web-palvelin toiminnallisuudet. Itse kickstart-tiedosto ks.cfg tallennettiin nimenomaisesti web-palvelimen (www-kansion) juureen, jonka ansiosta kickstart-tiedoston sisällön pystyy ladata Test_srv2 -palvelimella kyseisestä polusta.

Kickstartin käynnistämisen jälkeen CentOS:n asennus sujuu täysin automaattisesti ilman tarvetta tehdä mitään manuaalisia toimenpiteitä. Tämä nimenomaisesti on kickstart-tiedoston käytön tarkoitus ja vahvuus: kaikki asetukset on määritelty ennakkoon kickstartissa.

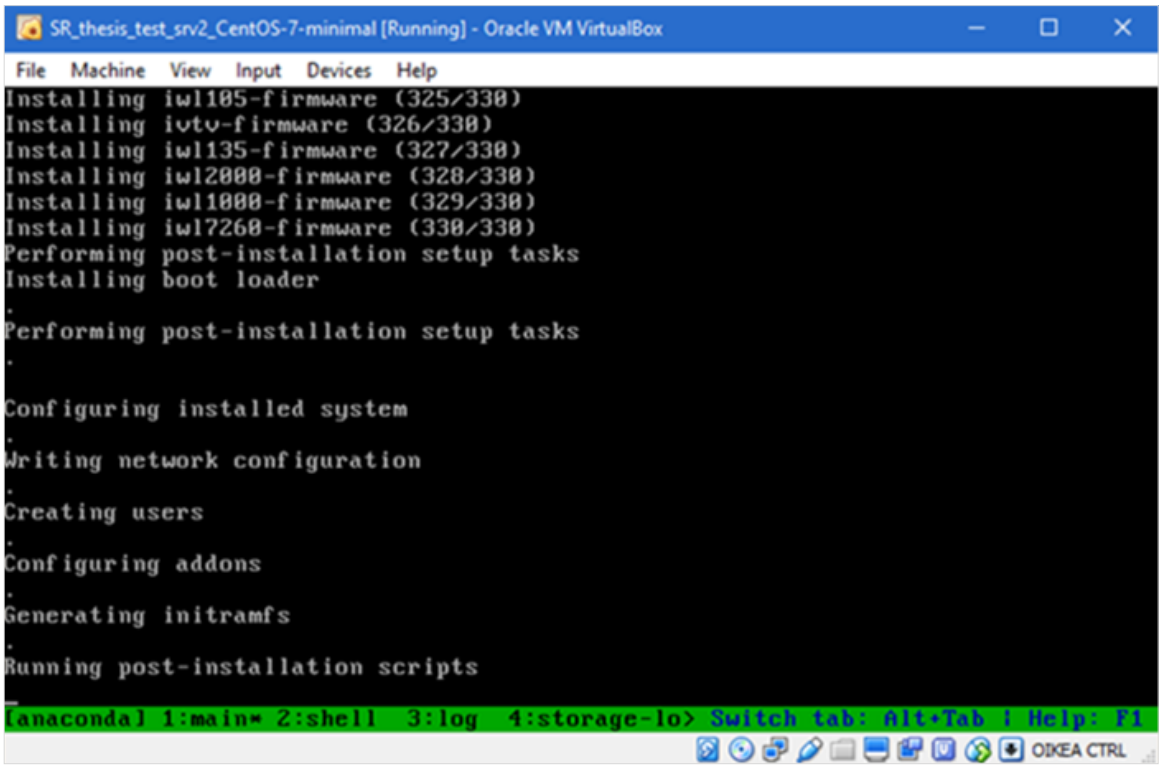
Omassa kickstart-tiedostossa oli määritelty varsinaisena asennuslähteenä käytettävän CentOS:n virallista latausikkunaa, jonka osoite on näkyvissä alla olevassa kuvankaappauksessa. Alla siis näkymiä tekstipohjaisesti ajetun asennuksen eri vaiheista.



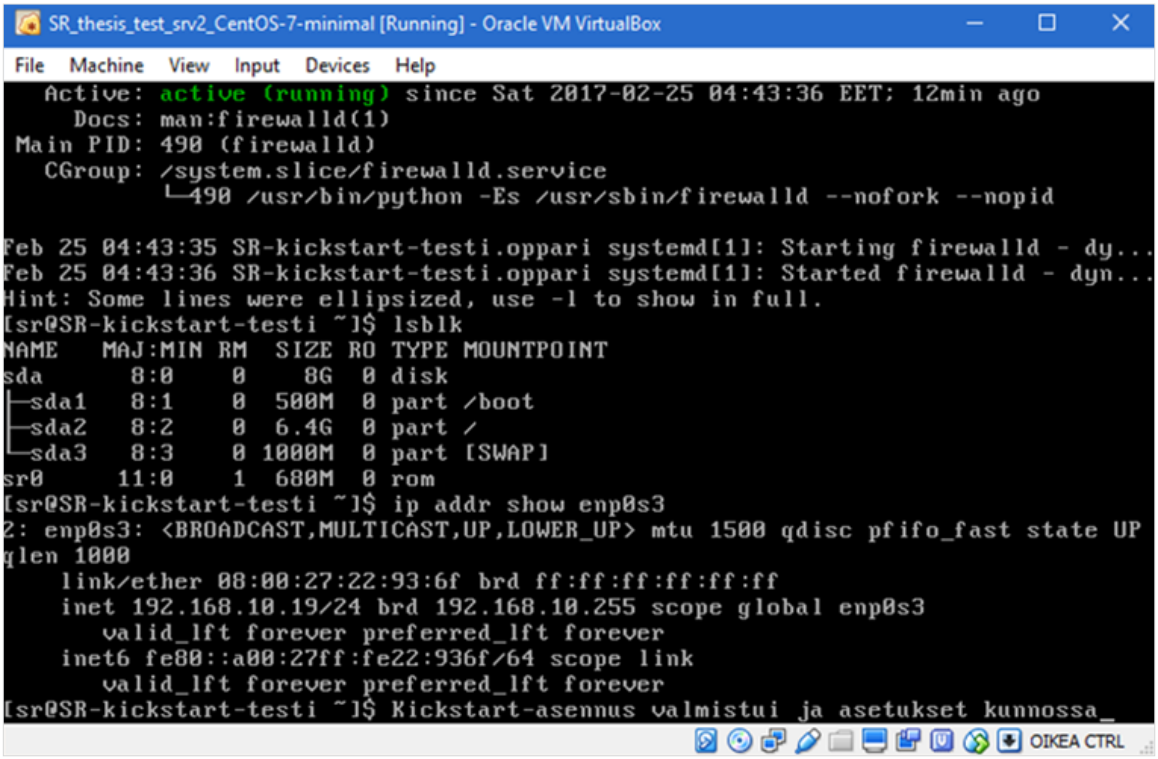
Kuva 30. Asennuksen alussa latautui kickstart-tiedostossa määritetyt käyttöjärjestelmän perusasetukset.



Kuva 31. Seuraavaksi asentui kaikki sovelluspaketit. Mukana myös kaikki erikseen kickstartissa määritellyt paketit kuten nmap, iotop ja epel-release.



Kuva 32. Automaatio hoiti bootloaderin asennuksen, käyttäjien luonnin ja OS:n asetuksien viimeistelyn. Lopuksi ajettiin vielä asennuksen jälkeiset (”post-install”) skriptit. Näiden vaiheiden jälkeen palvelin käynnistyi uudestaan ja OS oli heti käyttövalmis.



Kuva 33. Test_srv2 -palvelimen asetukset olivat täsmälleen kickstartin määritysten mukaiset, joten asennus onnistui suunnitellusti. Kicktart-testaaminen oli siis tältä erää ohi ja vielä oli jäljellä palvelinten konfiguraationhallintaan kehitetyn Ansible-työkalun testaus.

7 Palvelimen provisiointi Ansiblella

Palvelinten konfiguraationhallintatesteihin valikoitui työkaluksi esiselvityksien ja aiempien kokemuksien perusteella toimeksiantajan tarpeisiin soveltuva Ansible. Sovellus tarjoaa suoraviivaiset käyttömahdollisuudet palvelinten konfiguraationhallintaan ja sovelluksen YAML-kielen syntaksi on kohtuullisen helposti omaksuttavissa.

Ensimmäiseksi tulen konfiguroimaan Deployment_srv1:n Ansiblen hallintapalvelimeksi, lisään Ansiblen hosts-tiedostoon käytettävät testipalvelimet (Test_srv 2-4) ja kokeilen muutamia ad-hoc komentoja sekä playbookin käyttöä kyseisille palvelimille. Toisessa kappaleessa suunnittelen asettujen tavoitteiden mukaisesti Ansible-ympäristön rakenteen.

Suunnitelman pohjalta toteutan Ansible-ympäristön kokonaisuudessaan (playbookit, roolit ja taskit) sekä ajan kehitettyjen playbookkien sisällön kohdepalvelimille. Lopuksi varmistan palvelinten provisioinnin onnistuneen tarkastamalla kohdepalvelinten tilan. Itse Ansiblen yleiskuvaus ja toimintaperiaatteet on selitetty tarkemmin tietoperustan kappaleessa 3.6.

7.1 Ansiblen asennus ja testaus

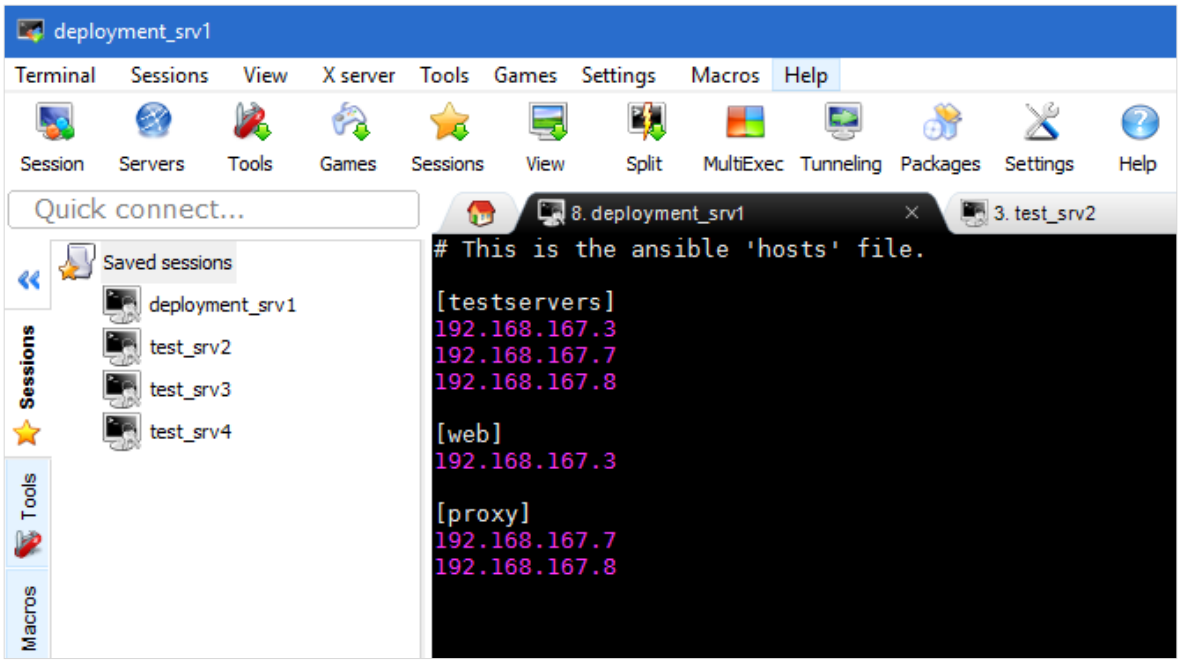
Aloitin suoraviivaisesti Ansiblen asentamisella Deployment_srv1:lle. Ansiblen RPM-asennuspaketti on saatavilla EPEL-repositorystä, joten asensin kyseisen pakettivaraston komennolla *”yum install epel-release”*. Itse Ansiblen asennus onnistui yksinkertaisesti komennolla *”yum install ansible”.* Ansiblen version 2.2.1.0 asennuksessa kesti joitakin sekunteja ja mukana asentui toistakymmentä Python-liitännäispakettia sekä lisäksi PyYAML- ja sshpass-paketit. (Ansible 2017c.)

Ansible-kansiosta löytyi asentamisen jälkeen vain konfiguraatiotiedosto ansible.cfg, oletus hosts-tiedosto sekä tyhjä roolit-kansio. Kansion tarkastelun jälkeen navigoin Ansible-testauksen kohdepalvelimille (Test_srv 2-4) ja varmistin kyseisillä palvelimilla olevan asennettuna Python. Lisäksi asensin vielä ”Tree”-sovelluksen komennolla *”yum install -y tree”* kaikille palvelimille. Tree-sovellus mahdollistaa hakemistorakenteen selkeämmän esityksen (puumuodossa) komentorivillä.

Seuraavaksi konfiguroin Ansiblen hosts-tiedoston, jossa palvelimet järjestellään ryhmiin IP-osoitetta tai hostnamea käyttäen. Tietyn ryhmän palvelimille voidaan provisioida samanaikaisesti vaikka kokonaisen playbookin sisältö ja yksittäinen palvelin voi kuulua moneen eri hosts-ryhmään. Käyttötarkoitukseltaan samantyyppiset palvelimet kannattaakin sijoittaa lähtökohtaisesti samoihin ryhmiin.

Suunnittelin Ansible-ympäristöä varten host-ryhmien rakenteet seuraavalla tavalla: ”testservers”-ryhmään lisäsin kaikki kolme testipalvelinta (Test_srv 2-4), ”web”-ryhmään lisäsin vain Test_srv2:n (192.168.167.3) ja kolmanteen ”proxy”-ryhmään lisäsin Test_srv3:n (192.168.167.7) sekä Test_srv4:n (192.168.167.8).

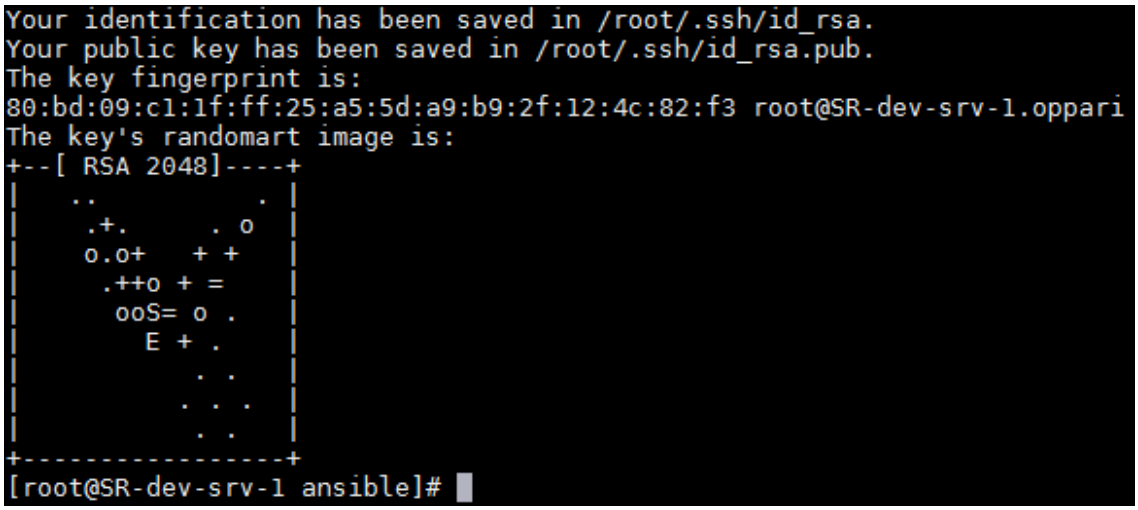
Mikäli siis haluaisin ajaa playbookin sisällön samanaikaisesti jokaiselle testipalvelimelle, niin se onnistuisi esimerkiksi käyttämällä host-tunnistetta ”testservers”, joka sisältää kaikkien palvelinten IP-osoitteet. Muut ryhmät (web ja proxy) loin varautuen tulevissa kappaleissa tehtävään Ansible-ympäristön rakentamiseen.



Kuva 34. Hallintapalvelimen inventaarioon määritettiin kolme eri palvelinryhmää.

Inventaarion konfiguroimisen jälkeen piti generoida SSH-avaimet hallintapalvelimella ja lisätä näiden julkiset avaimet kohdepalvelimille, jotta Ansiblen hallintapalvelin pystyy käyttämään kohdepalvelimia SSH-yhteyden välityksellä ilman salasanan syöttämistä.

Siispä seuraavaksi generoin SSH-avaimet komennolla: *”ssh-keygen -t rsa.”* ja parin enter-painalluksen jälkeen salattu avain tallentui aina automaattisesti polkuun *”/root/.ssh/id_rsa.”* sekä julkinen avain polkuun *”/root/.ssh/id_rsa.pub.”*.



Kuva 35. Generoidun SSH-avaimen julkinen ”sormenjälki”.

Generoin SSH-avaimia yhteensä kolme kappaletta eli yhden kutakin testipalvelinta kohti. Salatut avaimet säilytetään tietysti vain Deployment_srv1:llä, mutta julkiset SSH-avaimet kopioin asiaankuuluvasti testipalvelimille 2-4. Julkisen avaimen kopioinnin syntaksi oli *”ssh-copy-id root@kohdekoneen IP”* eli kopiointi onnistui seuraavilla komennoilla:

“ssh-copy-id root@192.168.167.3”
“ssh-copy-id root@192.168.167.7”
“ssh-copy-id root@192.168.167.8”

Alla vielä esimerkki kuinka julkisen avaimen kopiointi eteni komentorivillä.

[root@SR-dev-srv-1 ansible]# ssh-copy-id root@192.168.167.8
The authenticity of host '192.168.167.8 (192.168.167.8)' can't be established.

ECDSA key fingerprint is 98:ef:44:9f:1b:36:fa:6d:13:13:f4:09:be:de:ba:6c.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed

/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@192.168.167.8's password:

Number of key(s) added: 1
Now try logging into the machine, with: "ssh 'root@192.168.167.8'"
and check to make sure that only the key(s) you wanted were added.

Onnistuneiden avainten lisäyksien jälkeen kaikkien testipalvelimien tiedot ilmestyivät Deployment_srv1:llä ”known_hosts”-tiedostoon polussa ”/root/.ssh/ ”. Testasin lopuksi vielä SSH-yhteyden muodostamista Deployment_srv1:ltä jokaiseen testipalvelimeen ja yhteydet toimivat ilman ongelmia.

Nyt oli vihdoin aika testata Ansiblen ”ad-hoc”-komentojen syöttämistä hallintapalvelimella. Ensimmäiseksi testasin kaikille testipalvelimille pingausta käyttäen ping-moduulia ja hostnamen kysymistä käyttäen argument-moduulia, käytetyt komennot:

“ansible -m ping testservers”
“ansible -m command -a ”hostname” testservers”

Olin lisännyt aiemmin Hosts-tiedostoon ”testservers”-ryhmän, josta löytyy jokaisen palvelimen (Test_srv 2-4) IP-osoite. Tämän ansiosta jokaista IP-osoitetta ei tarvinnut syöttää erikseen komentoriville, vaan pelkkä hosts-ryhmän nimi riitti Ansiblelle tunnistamaan kohdepalvelimet.

Tietysti ”all”-komento olisi myös tässä tapauksessa ollut toimiva ratkaisu, kun testi kohdistui kirjaimellisesti kaikkiin palvelimiin. Jokatapauksessa ad-hoc testi onnistui erinomaisesti – pingi vastasi ja hostien nimistä saatiin oikeat vastaukset kuten alla olevassa kuvassa näkyy.

```
[root@SR-dev-srv-1 ~]# ansible -m ping testservers
192.168.167.7 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.167.8 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.167.3 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[root@SR-dev-srv-1 ~]# ansible -m command -a "hostname" testservers
192.168.167.7 | SUCCESS | rc=0 >>
SR-test-srv-3.oppari

192.168.167.8 | SUCCESS | rc=0 >>
SR-test-srv-4.oppari

192.168.167.3 | SUCCESS | rc=0 >>
SR-test-srv-2.oppari

[root@SR-dev-srv-1 ~]#
```

Kuva 36. Ad-hoc komentojen testaamista Ansiblella – ”SUCCESS”!

Loin vielä alta löytyvän yksinkertaisen playbookin testaamista varten – tarkoituksena oli vain nähdä kuinka playbookin ajo tapahtuu ruudulla. Playbook sisältää yhden playn, jossa on kaksi yksinkertaista tehtävää eli taskia peräkkäin. Yleensä playbookit eivät itsessään kuitenkaan sisällä taskeja, vaan niihin listataan ainoastaan suoritettavat roolit, joiden omiin konfiguraatiotiedostoihin määritetään itse roolissa suoritettavat taskit.

```
---
- hosts: web
 become_user: sudo
 tasks:
     - name: Create test file and add text | Luo testitiedosto ja lisää teksti
       copy: content="Testitiedoston sisältö" dest="/home/sr/testitiedosto.txt"

     - name: Install Ruby | Asenna Ruby
       yum: name=ruby state=latest
```

Playbookin ajaminen tehdään komennolla *"ansible-playbook testbook.yml*", jossa "testbook.yml" on luomani playbookin nimi.

Playbookkia ajaessa oletuksena suoritetaan aluksi "setup"-moduuli, joka tarkastaa kohdepalvelimesta tietoja ("facts"). Tämän jälkeen vasta ajetaan playbookista löytyvät tehtävät, joista ensimmäisessä luodaan testitiedosto määritettyyn polkuun ja toisessa asennetaan Ruby. Playbook ajettiin "web"-ryhmälle, joka sisälsi tässä tapauksessa ainoastaan Test_srv2-palvelimen. (Ansible 2017e.)

```
[root@SR-dev-srv-1 ansible]# ansible-playbook testbook.yml

PLAY [web] *********************************************************************

TASK [setup] *******************************************************************
ok: [192.168.167.3]

TASK [Create test file and add text | Luo testitiedosto ja lisää teksti] ********
changed: [192.168.167.3]

TASK [Install Ruby | Asenna Ruby] **********************************************
changed: [192.168.167.3]

PLAY RECAP *********************************************************************
192.168.167.3              : ok=3    changed=2    unreachable=0    failed=0
```

Kuva 37. Testbook.yml -playbook ajettiin Test_srv2:lle (192.168.167.3) onnistuneesti.

Kuvassa näkyvä "changed"-teksti playbookin ajon yhteydessä kertoo kohdepalvelimelle tehdyn kyseisen rivin osalta määritetyt muutokset. "Ok"-teksti vastaavasti tarkoittaa yleensä, ettei mitään tarvinnut muuttaa tilanteen ollessa valmiiksi kunnossa. Vasta "unreachable" tai "failed" viestit kielivät suoranaisesti jostain ongelmasta esimerkiksi playbookin syntaksissa tai palvelimen tavoitettavuudessa.

Tarkistin vielä varmuuden vuoksi kohdepalvelimen (Test_srv2) tilanteen ja rubyn paketit olivat asentuneet ja testitiedosto.txt ilmestynyt *"/home/sr/"*-polkuun sisällöllä "testitiedoston sisältö". Playbook-testi siis onnistui ja Ansiblen asennus osoittautui teknisesti kaikinpuolin toimintakykyiseksi.

7.2 Ansible-ympäristön suunnittelu

Kappaleessa käydään alkuun karkeasti läpi Ansible-testaamisen tavoitteet, joiden pohjalta suunnitellaan hostien, playbookkien, roolien ja taskien yhdessä muodostama Ansible-testiympäristön rakenne.

Ansible-testiympäristön avulla pyritään saamaan vahva kuva työkalun tarjoamista mahdollisuuksista provisioida palvelinympäristö ketterällä tavalla käyttövalmiiksi. Tämän tueksi kehitetään em. ympäristö, joka suunnitellaan simuloimaan yleisiä palvelimien provisiointiin liittyviä käyttötapauksia.

Yleisiin tavoitteisiin kuuluu, että Ansiblella saataisiin provisioitua jokaiselle palvelimelle testitunnus ja tarpeelliset sovellusasennukset sisältäen pakettivaraston päivityksen. Lisäksi palomuuurin sekä NTP:n asennus ja konfigurointi olisi mainio lisä toteutukseen.

Spesifisempiin tavoitteisiin lukeutuvat web-palvelimen asennus, jonka avulla ylläpidettäisiin HTML-pohjaista verkkosivustoa. Sivuston template tulisi kehittää itse ja provisioida em. palvelimelle. Ylläpidettävän web-sivuston käyttöä sujuvoittamaan ja tietoturvaa lisäämään tulisi konfiguroida vielä edustapalvelimet (käänteisproxyt).

Yllä mainittujen tavoitteiden pohjalta suunnittelin kolmen testipalvelimen provisioinnin toteutettavan kahdella playbookilla, jotka molemmat sisältävät kaksi Ansible-roolia. Molempien playbookkien rooleista toinen on sama, joten yhteensä tarvetta on luoda kolme erilaista roolia.

Molemmissa playbookeissa käytetään ”yleisroolia”, jonka avulla provisioidaan kaikille palvelimille sovellusasennukset, tunnuksen luonti ja NTP:n sekä palomuurin konfigurointi. Web-ryhmälle eli yksittäiselle Test_srv2-palvelimelle provisioidaan yleisen roolin lisäksi Apache web-palvelimen asennus sisältäen HTML-sivuston pystytyksen.

Vastaavasti Proxy-ryhmän palvelimille eli Test_srv3:lle ja Test_srv4:lle provisioidaan saman yleisen roolin lisäksi Nginx-sovelluksen asennus ja konfigurointi. Nginx asetetaan toimimaan käänteisproxyna Apache-webpalvelimelle mikä mahdollistaa luotavan HTML-sivuston käytön onnistuvan myös proxyina toimivien edustapalvelimien verkko-osoitteilla.

Kuvaan vielä alla olevissa taulukoissa jokaisen luotavan playbookin, roolin, taskin sekä hostin ja näiden väliset keskinäiset suhteet.

Taulukot 13-14. Ansible-ympäristön rakenne kuvattuna kahdessa taulukossa.

Playbook 1	
Playbook	apache-yleinen.yml
Hosts	web (Test_srv2)
Rooli 1 ↓	apache
Taskit	apache.yml, main.yml
Handler taskit	main.yml, restart-apache.yml
Templatet	index.html.j2
Tiedostot	cat.jpg
Rooli 2 ↓	yleinen
Taskit	createuser.yml, firewalld.yml, install-software.yml, main.yml, ntp.yml
Handler taskit	main.yml, restart-firewalld.yml, restart-ntp.yml

Playbook 2	
Playbook	nginx-yleinen.yml
Hosts	proxy (Test_srv3 ja Test_srv4)
Rooli 1 ↓	nginx
Taskit	main.yml, nginx.yml
Handler taskit	main.yml, nginx yml
Templatet	nginx.conf.j2
Rooli 2 ↓	yleinen
Taskit	createuser.yml, firewalld.yml, install-software.yml, main.yml, ntp.yml
Handler taskit	main.yml, restart-firewalld.yml, restart-ntp.yml

Yhteensä Ansible ympäristön tekninen toteutus vaatii siis 20 eri tiedoston konfiguroimista, joten työtä on runsaasti, kun tiedostojen sisällön kehitys ja toimivuuden varmistus vie oman aikansa. Yksinkertaistetusti sanoen siis Ansible-osuudessa on vielä jäljellä ympäristön tekninen toteutus kokonaisuudessaan ja playbookkien ajo kohdepalvelimille.

7.3 Toteutus ja palvelinten provisiointi

Toteutan aluksi suunnitelmassa mainitut kolme eri roolia (yleinen, apache ja nginx), jonka jälkeen luon tarvittavat playbookit ja ajan ne kohdepalvelimille. Lopuksi todennan provisioinnin onnistuneen tarkastamalla testipalvelinten asetukset.

Varsinaisen käytännön toteutuksen aloitin luomalla jokaiselle tarvittavalle roolille hakemistorakenteen. Ansible Galaxyn *”init”*-komennon avulla kokonaisen roolirakenteen luominen onnistui helposti yksittäisellä komennolla, käytetyt komennot:

”ansible-galaxy init yleinen -p roles”
”ansible-galaxy init apache -p roles”
“ansible-galaxy init nginx -p roles”

Galaxyn luoma kansiorakenne sisälsi seuraavat kansiot sekä yksittäisen README.md-tiedoston. Osassa kansioita oli lisäksi vielä main.yml-tiedosto sisällä.

- defaults (+main.yml)
- files
- handlers (+main.yml)
- meta (+main.yml)
- tasks (+main.yml)
- templates
- tests
- vars (+main.yml)
- README.md -tiedosto

Tässä opinnäytetyössä tulen käyttämään tasks-, handlers-, templates- ja files-kansioita.

Kansiorakenteiden luomisen jälkeen ensimmäisenä vuorossa oli yleisen roolin toteutus sisältäen kaikkien tarvittavien taskien kehityksen. Siispä ryhdyin tekemään kaikki kyseiset .yml-konfigurointitiedostot lisäten vielä tasks- ja handlers-kansioiden main.yml-tiedostoihin ”include”-komennot kaikista tehdyistä taskeista. Kyseisen toimenpiteen avulla Ansible löytää playbookkia ajaessa taskit ja handlerit sekä rakenne selkeytyy. (Ansible 2017f.)

Testasin aina roolin kehityksen edetessä sen hetkisten asetusten toimivuutta ja yleisen roolin kehitys sujuikin pääosin ilman suurempia ongelmia. Palomuurin porttikonfigurointien voimaan saamisessa kohdepalvelimille oli alkuun kuitenkin hieman ongelmia, mutta lopulta sain kuitenkin roolin täysin toimivaksi eikä mitään ongelmia jäänyt ratkaisematta. Yleisen roolin tiedostoissa on tarkemmat selitykset mitä kukin tehtävä tekee.

Taulukko 15. ”Yleinen”-rooli kokonaisuudessaan.

Rooli: ”yleinen”

Taskit

main.yml

```
# Asentaa muutaman sovelluksen
- include: install-software.yml

# Asentaa ja käynnistää NTP-palvelun
- include: ntp.yml

# Luo Sami-käyttäjätunnuksen
- include: createuser.yml

# Asentaa ja konfiguroi Firewalld-palomuurin
- include: firewalld.yml
```

createuser.yml

```
# Luo Sami-käyttäjätunnuksen palvelimelle

- user:

  name=Sami

  shell=/bin/bash

  groups=wheel append=yes
```

firewalld.yml

```
# Asentaa ja konfiguroi Firewalld-palomuurin

- name: Install Firewalld
  yum: name=firewalld state=latest

- name: Start Firewalld service
  service: name=firewalld state=started enabled=yes

- name: Set port rules
  firewalld: port={{item}}/tcp permanent=true immediate=yes state=enabled
  with_items:
  - 23
  - 80
  - 443

  notify: Restart Firewalld
```

install-software.yml

```
# Asentaa alla luetellut sovellukset

- name: Install software
  yum: name={{item}} state=latest update_cache=yes
  with_items:
    - vim
    - wget
    - iotop
    - tar
    - curl
    - libselinux-python
```

ntp.yml

```
# Asentaa ja käynnistää NTP-palvelun

- name: Install NTP
  yum: name=ntp state=latest

- name: Start NTP
  service: name=ntpd state=started enabled=yes
  notify: Restart NTP
```

Handler taskit

main.yml

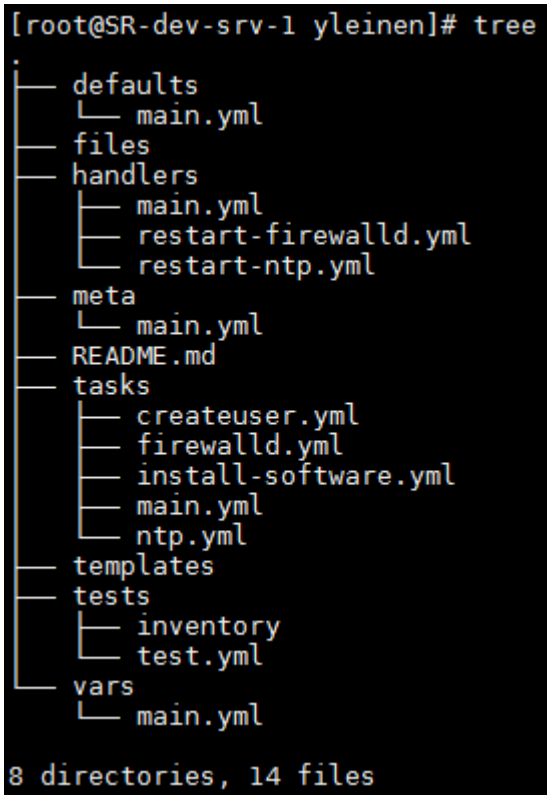
```
# Uudelleenkäynnistää Firewalld-palomuurin
- include: restart-firewalld.yml

# Uudelleenkäynnistää NTP-palvelun
- include: restart-ntp.yml
```

restart-firewalld.yml

```
- name: Restart Firewalld
  service: name=firewalld state=restarted
```

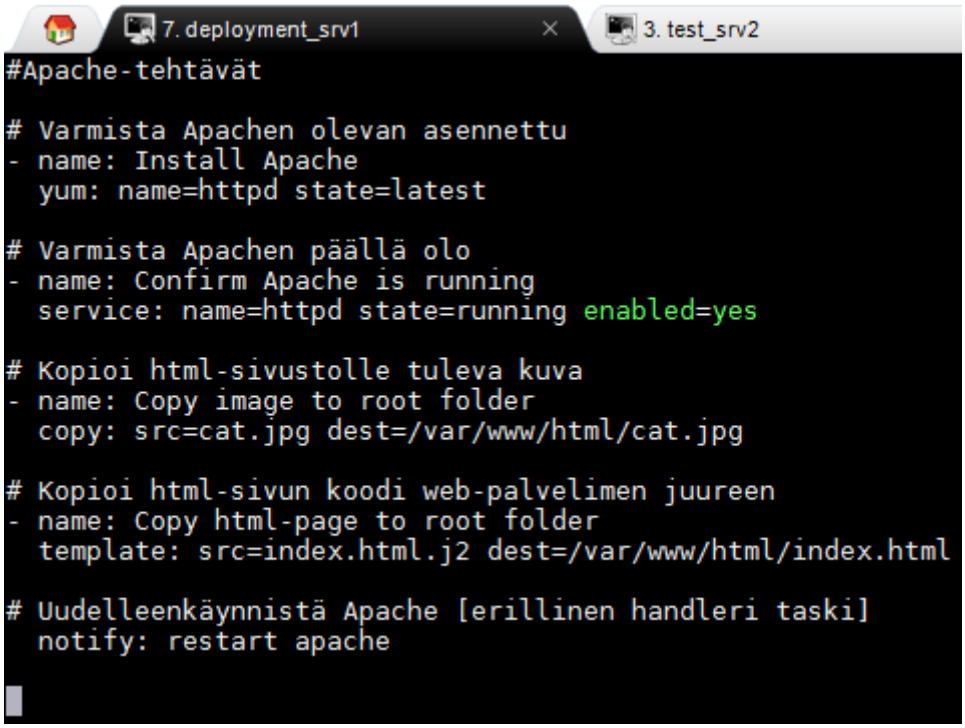
restart-ntp.yml

```
- name: Restart NTP
  service: name=ntpd state=restarted
```



Kuva 38. Yleisen roolin hakemistorakenne sisältäen kaikki luodut tiedostot.

Toisena roolina toteutin ”apachen”, joka nimensä mukaisesti asentaa palvelimelle Apache web-palvelun ja konfiguroi yksinkertaisen HTML-sivuston. Seurasin toteutuksessa osittain Cloudacademyn Apache-tutoriaalia. (Cloudacademy 2015.)



Kuva 39. Apache-taskissa käytettiin yum, copy, service ja template moduuleita.

Apache-roolin toteuttamisessa edelliseen nähden oli erilaista lähinnä, että latasin wget-sovelluksella Ansiblen files-kansioon yksittäisen kuvan ja loin suhteellisen yksinkertaisen index.html-tiedoston verkkosivustoa varten. Lisäsin kyseisen tiedoston Ansiblen templates-kansioon .j2-päätteisenä, josta se tullaan provisioimaan Test_srv2-palvelimelle ja playbookin ajon jälkeen sivuston kuuluisi toimia normaalisti.

Kaikki muut taskit ja handler taskit loin pitkälti samalla kaavalla kuin edellisessä roolissa. Joitakin pieniä eroavaisuuksia oli tietysti, mutta lähinnä jo mainittu template-tiedoston luominen ensimmäistä kertaa oli tässä roolissa uutta verrattuna edelliseen. Seuraavalla sivulla on tarkemmat tiedot ”apache”-roolin tehtävistä.

Taulukko 16. ”Apache”-rooli kokonaisuudessaan.

Rooli: ”apache”

Taskit

main.yml

```
- include: apache.yml
```

apache.yml

```
#Apache-tehtävät

# Varmista Apachen olevan asennettu
- name: Install Apache
  yum: name=httpd state=latest

# Varmista Apachen päällä olo
- name: Confirm Apache is running
  service: name=httpd state=running enabled=yes

# Kopioi html-sivustolle tuleva kuva
- name: Copy image to root folder
  copy: src=cat.jpg dest=/var/www/html/cat.jpg

# Kopioi html-sivun koodi web-palvelimen juureen
- name: Copy html-page to root folder
  template: src=index.html.j2 dest=/var/www/html/index.html

# Uudelleenkäynnistä Apache [erillinen handleri taski]
  notify: restart apache
```

Handler taskit

main.yml

```
- include: restart-apache.yml
```

restart-apache.yml

```
---
- name: restart apache
  service: name=httpd state=restarted
```

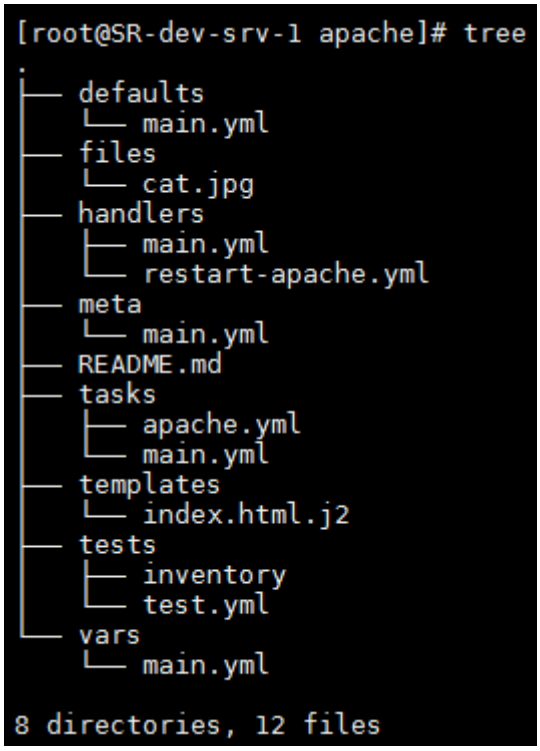
Template

index.html.j2

```
<html>
<head>
<meta charset="UTF-8">
<title>Thesis test-site</title>
</head>
<body>
<h2>
<p style="text-align:center;">Samin testisivu opinnäytetyöhön</p>
</h2>
<p style="text-align:center;"><img src=cat.jpg alt=”cat”/></p>
<center><p id="clock"></p></center>
<script>
document.getElementById("clock").innerHTML =Date();
</script>
<p style="text-align:center;">Sivusto hostattu Test_srv2-palvelimella (192.168.167.3)</p>
</body>
</html>
```

Tiedostot

cat.jpg



Kuva 40. Apache-roolin hakemistorakenne sisältäen kaikki luodut tiedostot.

Viimeinen toteutettava rooli oli ”nginx”, jolla kohdepalvelimet konfiguroidaan Nginx-käänteisproxyksi. Palvelimet toimivat tällöin Apache-palvelimella ylläpidettävän HTML-sivuston edustapalvelimina. Käytännössä tämä tarkoittaa, että samaan HTML-sivustoon saa yhteyden, niin suoraan Apache-palvelimen oman verkko-osoitteen kautta kuin myös Nginx:llä toimivien edustapalvelimien välityksellä.

Taulukko 17. Kehitetty ”nginx”-rooli kokonaisuudessaan.

Rooli: ”nginx”
Taskit
main.yml
- include: nginx.yml
nginx.yml

#Nginx-tehtävät
 	
# Asenna EPEL-repository 	
- name: Install EPEL-repository
  yum: name=epel-release state=installed
 	
# Asenna Nginx 	
- name: Install Nginx
  yum: name=nginx state=installed
 	
# Korvaa alkuperäinen nginx.conf-tiedosto uudella 	
- name: Replace Nginx configuration file
  template: src=nginx.conf.j2 dest=/etc/nginx/nginx.conf

```
# Käynnistä Nginx-palvelu [erillinen handleri taski]
  notify: Start Nginx
```

Handler taskit

main.yml

```
- include: start-nginx.yml
```

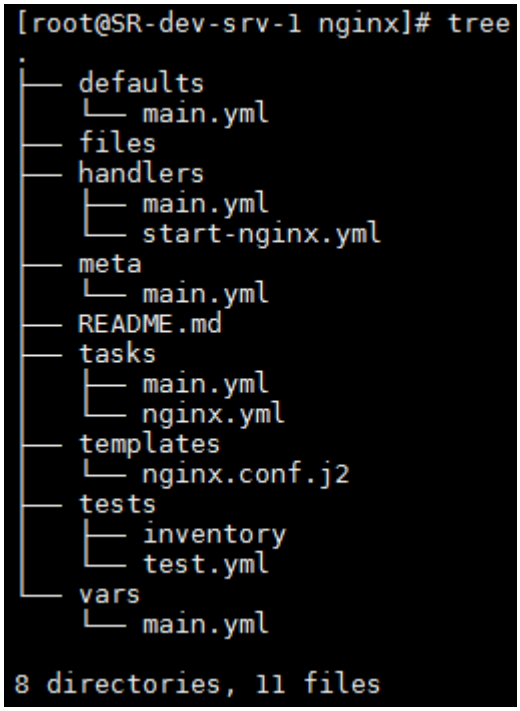
start-nginx.yml

```
- name: Start Nginx
  service: name=nginx state=started
```

Templatet

nginx.conf.j2

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;


server {
    listen 80;
    location / {
    proxy_pass http://192.168.167.3;
    proxy_set_header X-Real-IP  $remote_addr;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_set_header Host $host;
    }
 }
}
```



Kuva 41. Nginx-roolin hakemistorakenne sisältäen kaikki luodut tiedostot.

Nginx-roolin toteuttamisessa vaativin asia oli saada Nginx toimimaan käänteisproxyna Apache-palvelimelle – kyseisen ominaisuuden sai sisällytettyä rooliin nginx.conf-tiedoston konfiguroinnilla. Muutoin nginx-roolin luonnissa ei ollut muihin rooleihin verrattuna suuria eroavaisuuksia. Nginx-roolin toteuttamisen jälkeen kaikki kolme roolia oli vihdoinkin kehitetty ja testattu kehityksen aikana toimiviksi.

Seuraavaksi oli vuorossa toteuttaa tarvittavat kaksi playbookkia, joiden luonti on hyvin pieni toimenpide verrattuna kokonaisten roolien kehittämiseen. Kauaa ei playbookkien toteuttamisessa mennytkään – konfiguroin playbookit seuraavalla tavalla:

Taulukot 18-19. Suunnitelman mukaisesti luodut playbookit.

Playbook 1	Playbook 2
Playbook: apache-yleinen.yml	Playbook: nginx-yleinen.yml

--- 	
- hosts: web
  become_user: sudo
  roles:
  - yleinen
  - apache
	--- 	
	- hosts: proxy
	  become_user: sudo
	  roles:
	  - yleinen
	  - nginx

Ensimmäinen playbook ”apache-yleinen.yml” sisältää nimensä mukaisesti kaksi eri roolia (apache ja yleinen) ja se ajetaan web-ryhmän Test_srv2-palvelimelle (192.168.167.3).

Toinen playbook ”nginx-yleinen.yml” sisältää myös kaksi roolia (nginx ja yleinen) ja se ajetaan molemmille proxy-palvelimille eli Test_srv3:lle (192.168.167.7) ja Test_srv4:lle (192.168.167.8).

Lopuksi oli enää jäljellä playbookkien ajaminen kohdepalvelimille ja lopputulosten katselmointi. Ensimmäiseksi ajoin ”apache-yleinen” -playbookin Test_srv2:lle komennolla:

“ansible-playbook apache-yleinen.yml”

Playbookin ajo sujui ongelmitta kuten alla olevan kuvan “changed”-teksteistä voi päätellä eli muutokset tehtiin onnistuneesti Test_srv2:lle.

```
[root@SR-dev-srv-1 ansible]# ansible-playbook apache-yleinen.yml

PLAY [web] *********************************************************************

TASK [setup] *******************************************************************
ok: [192.168.167.3]

TASK [yleinen : Install software] **********************************************
changed: [192.168.167.3] => (item=[u'vim', u'wget', u'iotop', u'tar', u'curl', u'libselinux-python'])

TASK [yleinen : Install NTP] ***************************************************
changed: [192.168.167.3]

TASK [yleinen : Start NTP] *****************************************************
changed: [192.168.167.3]

TASK [yleinen : Create user] ***************************************************
changed: [192.168.167.3]

TASK [yleinen : Install Firewalld] *********************************************
changed: [192.168.167.3]

TASK [yleinen : Start Firewalld service] ***************************************
changed: [192.168.167.3]

TASK [yleinen : Set port rules] ************************************************
changed: [192.168.167.3] => (item=23)
changed: [192.168.167.3] => (item=80)
changed: [192.168.167.3] => (item=443)

TASK [apache : Install Apache] *************************************************
changed: [192.168.167.3]

TASK [apache : Confirm Apache is running] **************************************
changed: [192.168.167.3]

TASK [apache : Copy image to root folder] **************************************
changed: [192.168.167.3]

TASK [apache : Copy html-page to root folder] **********************************
changed: [192.168.167.3]

RUNNING HANDLER [yleinen : Restart Firewalld] **********************************
changed: [192.168.167.3]

RUNNING HANDLER [yleinen : Restart NTP] ****************************************
changed: [192.168.167.3]

RUNNING HANDLER [apache : restart apache] **************************************
changed: [192.168.167.3]

PLAY RECAP *********************************************************************
192.168.167.3              : ok=15   changed=14   unreachable=0    failed=0

[root@SR-dev-srv-1 ansible]#
```

Kuva 42. ”Apache-yleinen.yml” -playbook ajettuna loppuun asti.

Heti perään ajoin ”nginx-yleinen” -playbookin Test_srv3:lle ja Test_srv4:lle komennolla:

”ansible-playbook nginx-yleinen.yml”

Myös ”nginx-yleinen” -playbookin suoritus meni alusta loppuun ilman virheitä.

```
PLAY RECAP *********************************************************************
192.168.167.3              : ok=15   changed=14   unreachable=0    failed=0

[root@SR-dev-srv-1 ansible]# ansible-playbook nginx-yleinen.yml

PLAY [proxy] *******************************************************************

TASK [setup] *******************************************************************
ok: [192.168.167.7]
ok: [192.168.167.8]

TASK [yleinen : Install software] **********************************************
changed: [192.168.167.7] => (item=[u'vim', u'wget', u'iotop', u'tar', u'curl', u'libselinux-python'])
changed: [192.168.167.8] => (item=[u'vim', u'wget', u'iotop', u'tar', u'curl', u'libselinux-python'])

TASK [yleinen : Install NTP] ***************************************************
changed: [192.168.167.8]
changed: [192.168.167.7]

TASK [yleinen : Start NTP] *****************************************************
changed: [192.168.167.7]
changed: [192.168.167.8]

TASK [yleinen : Create user] ***************************************************
changed: [192.168.167.8]
changed: [192.168.167.7]

TASK [yleinen : Install Firewalld] *********************************************
changed: [192.168.167.7]
changed: [192.168.167.8]

TASK [yleinen : Start Firewalld service] ***************************************
changed: [192.168.167.8]
changed: [192.168.167.7]

TASK [yleinen : Set port rules] ************************************************
changed: [192.168.167.8] => (item=23)
changed: [192.168.167.7] => (item=23)
changed: [192.168.167.7] => (item=80)
changed: [192.168.167.8] => (item=80)
changed: [192.168.167.8] => (item=443)
changed: [192.168.167.7] => (item=443)

TASK [nginx : Install EPEL-repository] *****************************************
changed: [192.168.167.7]
changed: [192.168.167.8]

TASK [nginx : Install Nginx] ***************************************************
changed: [192.168.167.7]
changed: [192.168.167.8]

TASK [nginx : Replace Nginx configuration file] ********************************
changed: [192.168.167.8]
changed: [192.168.167.7]

RUNNING HANDLER [yleinen : Restart Firewalld] **********************************
changed: [192.168.167.8]
changed: [192.168.167.7]

RUNNING HANDLER [yleinen : Restart NTP] ****************************************
changed: [192.168.167.8]
changed: [192.168.167.7]

RUNNING HANDLER [nginx : Start Nginx] ******************************************
changed: [192.168.167.8]
changed: [192.168.167.7]

PLAY RECAP *********************************************************************
192.168.167.7              : ok=14   changed=13   unreachable=0    failed=0
192.168.167.8              : ok=14   changed=13   unreachable=0    failed=0

[root@SR-dev-srv-1 ansible]#
```

Kuva 43. ”Nginx-yleinen.yml” -playbookin ajonaikainen näkymä.

Molempien playbookkien suoritus onnistui alusta loppuun asti eli kaikki konfiguroidut muutokset tehtiin Ansiblen mukaan kaikille kohdepalvelimille ilman virheitä. Seuraavaksi olikin jäljellä enää kohdepalvelimille suoritettava muutoksien todentaminen sekä provisioidun HTML-verkkosivuston tarkastelu ja siihen liittyvän käänteisproxyn testaus.

Alkuun tarkastin Test_srv4-palvelimelta, että ajettujen roolien (nginx ja yleinen) tekemät muutokset tulivat oikeasti voimaan palvelimella. Selvitin asian seuraavilla komennoilla:

1) *”firewall-cmd –list ports”*
   ✓ Portit 23,80 ja 443 olivat asiaankuuluvasti auki.

2) *”cat /etc/passwd | grep Sami”*
   ✓ Palvelimelta löytyi “Sami”-tunnus.

3) *”service nginx status”*
   ✓ Nginx oli asennettu ja aktiivisena.

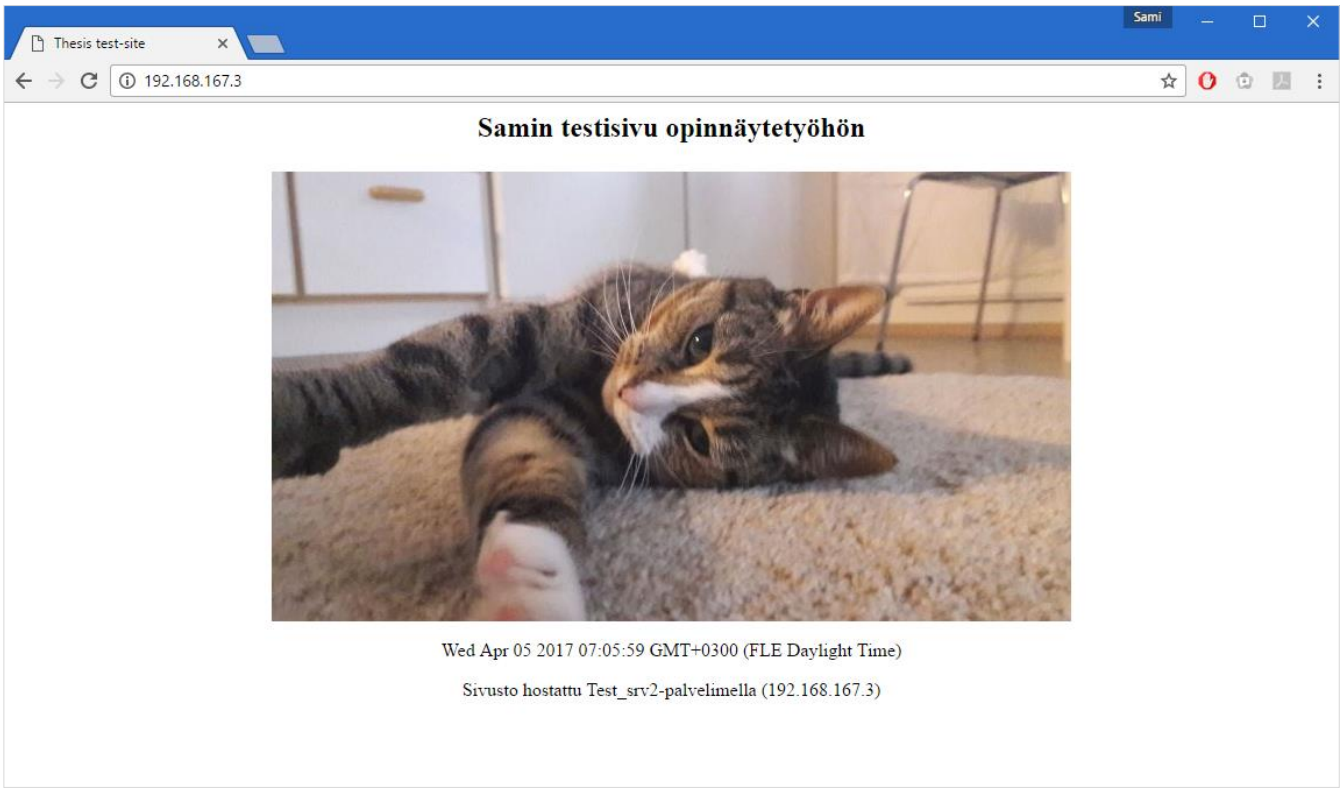
4) *”service ntpd status”*
   ✓ NTP oli asennettu ja aktiivisena.

5) *”curl google.fi”*
   ✓ Curl-sovellus oli asennettuna.

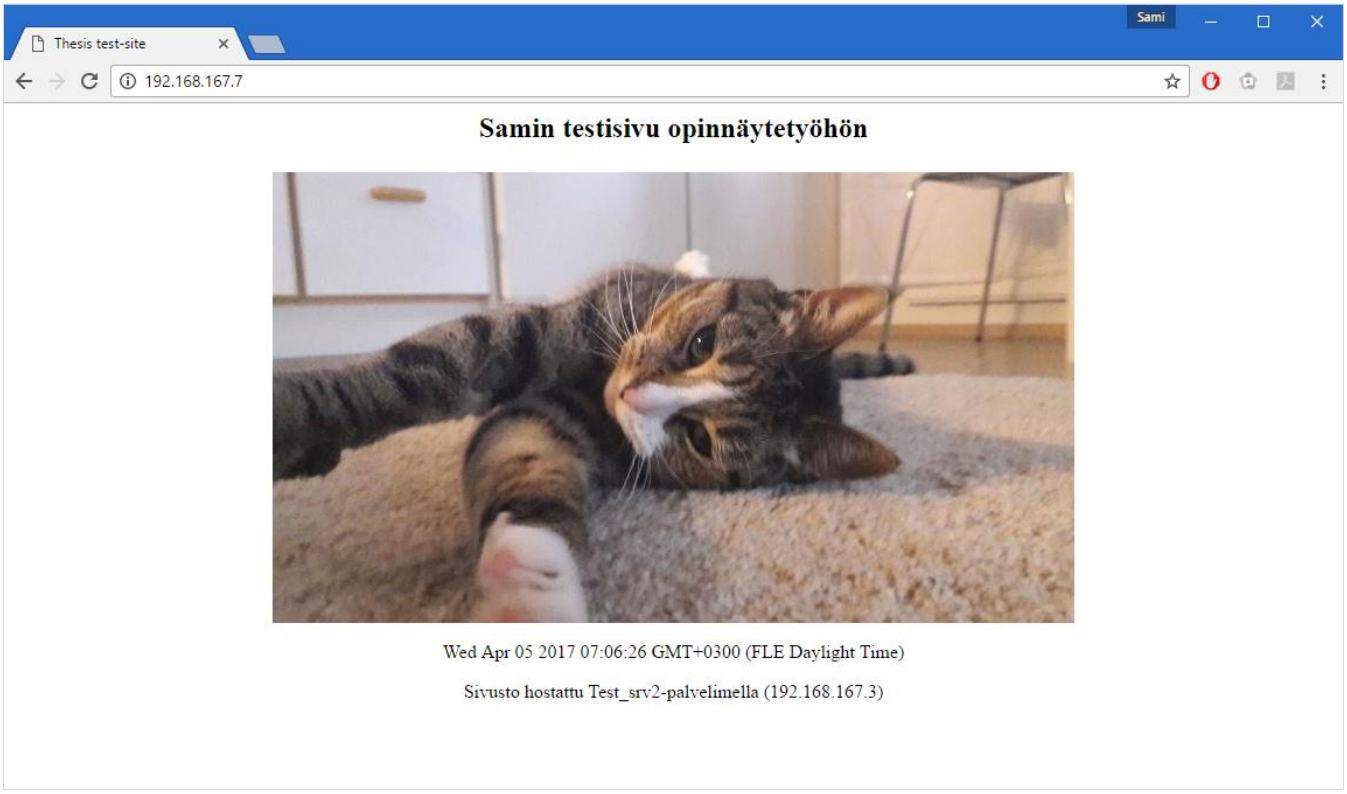
```
[root@SR-test-srv-4 html]# firewall-cmd --list-ports
443/tcp 23/tcp 80/tcp
[root@SR-test-srv-4 html]# cat /etc/passwd | grep Sami
Sami:x:1001:1001::/home/Sami:/bin/bash
[root@SR-test-srv-4 html]# service nginx status
Redirecting to /bin/systemctl status  nginx.service
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2017-04-05 06:51:54 EEST; 28min ago
  Process: 10786 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
  Process: 10783 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
  Process: 10782 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
 Main PID: 10789 (nginx)
   CGroup: /system.slice/nginx.service
           ├─10789 nginx: master process /usr/sbin/nginx
           └─10790 nginx: worker process

Apr 05 06:51:54 SR-test-srv-4.oppari systemd[1]: Starting The nginx HTTP and reverse proxy server...
Apr 05 06:51:54 SR-test-srv-4.oppari nginx[10783]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Apr 05 06:51:54 SR-test-srv-4.oppari nginx[10783]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Apr 05 06:51:54 SR-test-srv-4.oppari systemd[1]: Failed to read PID from file /run/nginx.pid: Invalid argument
Apr 05 06:51:54 SR-test-srv-4.oppari systemd[1]: Started The nginx HTTP and reverse proxy server.
[root@SR-test-srv-4 html]# service ntpd status
Redirecting to /bin/systemctl status  ntpd.service
● ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2017-04-05 06:51:54 EEST; 28min ago
  Process: 10747 ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 10748 (ntpd)
   CGroup: /system.slice/ntpd.service
           └─10748 /usr/sbin/ntpd -u ntp:ntp -g

Apr 05 06:51:54 SR-test-srv-4.oppari ntpd[10748]: Listen normally on 5 lo ::1 UDP 123
Apr 05 06:51:54 SR-test-srv-4.oppari ntpd[10748]: Listen normally on 6 enp0s3 fe80::a00:27ff:febd:ea21 UDP 123
Apr 05 06:51:54 SR-test-srv-4.oppari ntpd[10748]: Listen normally on 7 enp0s8 fe80::a00:27ff:fee6:ad24 UDP 123
Apr 05 06:51:54 SR-test-srv-4.oppari ntpd[10748]: Listening on routing socket on fd #24 for interface updates
Apr 05 06:51:54 SR-test-srv-4.oppari ntpd[10748]: 0.0.0.0 c016 06 restart
Apr 05 06:51:54 SR-test-srv-4.oppari ntpd[10748]: 0.0.0.0 c012 02 freq_set kernel 0.000 PPM
Apr 05 06:51:54 SR-test-srv-4.oppari ntpd[10748]: 0.0.0.0 c011 01 freq_not_set
Apr 05 06:52:01 SR-test-srv-4.oppari ntpd[10748]: 0.0.0.0 c614 04 freq_mode
Apr 05 07:08:47 SR-test-srv-4.oppari ntpd[10748]: 0.0.0.0 0612 02 freq_set kernel 19.833 PPM
Apr 05 07:08:47 SR-test-srv-4.oppari ntpd[10748]: 0.0.0.0 0615 05 clock_sync
[root@SR-test-srv-4 html]# curl google.fi
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.fi/">here</A>.
</BODY></HTML>
[root@SR-test-srv-4 html]#
```

Kuva 44. Testikomentojen perusteella voi todeta provisioinnin onnistuneen.

Viimeisenä tehtävänä oli varmistaa Apache-palvelimella (Test_srv2) ylläpidettävän HTML-sivun toimivan, niin Test_srv2:n IP-osoitteella kuin myös Nginx-käänteisproxyilla.



Kuva 45. Sivusto toimi Apache web-palvelimen (Test_srv2) IP-osoitteella 192.168.167.3.



Kuva 46. Sivusto toimi myös Test_srv3:n (192.168.167.7) Nginx-käänteisproxyn kautta.

8 Käyttöönotto- ja palautusmenetelmät

Kappaleessa käydään yleisesti läpi testattujen käyttöönotto- ja palautusmenetelmien hyviä ja huonoja puolia sekä analysoidaan menetelmien käyttökelpoisuutta toimeksiantaja Y:n kehityshanketta silmällä pitäen. Lopuksi vielä alustetaan menetelmien jatkokehitystä tulevaisuutta varten.

8.1 Yleistä palvelimen käyttöönotosta

Toiminnallisen osuuden valmistuttua oli vielä jäljellä palvelimen käyttöönotto- ja palautusmenetelmien konkreettinen käsittely pohjautuen testatuista työkaluista saatuihin kokemuksiin. Palvelimen käyttöönotto tai palautus on mahdollista toteuttaa lukuisilla eri tavoilla ja universaalia totuutta absoluuttisesti parhaasta tavasta ei sinällään ole.

Eri yritysten tarpeisiin soveltuu erilaiset menetelmät ja prosessit: ehkäpä merkittävin sopivan menetelmän valintaan vaikuttava seikka on palvelinympäristön koko. Siinä missä kymmenentä palvelinta voi järjestelmän ylläpitäjä hallita huomattavasti kevyemmilläkin prosesseilla, niin määrän kasvaessa satoihin tai tuhansiin – palvelimet muuttuvat täysin ”kasvottomaksi” ja mahdollisia yksilöllisiä erityispiirteitä yksittäisestä palvelimesta on huomattavasti vaikeampi muistaa tai tietää.

Kyle Rankin on kuvannut artikkelissaan ”How to deploy a server” erinomaisesti palvelinten ylläpidon eri sukupolvia. Alkujaan ylläpitäjät suorittivat palvelinten käyttöönoton alusta loppuun käsin jota seurasi kustomoitujen levykuvien tarjoamat mahdollisuudet kloonaamisineen. Image-sukupolven jälkeen yleistyi palvelinten käyttöönotto konfiguraatitiedostoilla kuten RHEL:n kickstartilla sekä siihen sisällytettävillä skripteillä.

Kickstartin tekstipohjainen syntaksi helpotti käyttöjärjestelmän konfiguraation hallintaa sen sijaan, että aina suurien muutoksien jälkeen olisi joutunut luomaan uuden levykuvan, joka sisältäisi esimerkiksi kaikki viimeisimmät päivitykset. Silti asennuksessa käytettävät konfiguraatiotiedostot ja skriptit eivät yksinään vielä pystyneet takaamaan eri aikakausina yritykseen asennettujen palvelinten olevan varmasti täysin identtisiä.

Kyseisen ongelman ratkaisemiseksi on kehitetty keskitetyn palvelinhallinnan mahdollistavia työkaluja, joiden välityksellä komennot ajetaan samanaikaisesti useampaan kohdepalvelimeen. Lisäksi samaisten työkalujen avulla pystytään havaitsemaan onko kaikille palvelimille ajettu täysin identtiset asennukset ja asetukset.

Kyseisiä työkaluja hyödyntämällä voidaan siis varmistua konfiguraation pysyvän kaikilla palvelimilla samanlaisena. (Rankin 2013.)

Tässä opinnäytetyössä testaamani työkalut kävivät tavallaan läpi palvelimen käyttöönoton kaikki eri sukupolvet alkaen virtuaaliympäristön käsin pystyttämisestä, jota seurasi palvelinten varmuuskopiointi ja palautus levykuvia käyttäen. Tämän jälkeen toteutettiin palvelimen käyttöönotto kickstart-tiedoston avulla ja viimeisimpänä oli palvelinjoukon provisiointi keskitetyn konfiguraationhallinnan mahdollistavalla Ansiblella.

8.2 Menetelmien analysointi

Testatuista menetelmistä palvelimen kloonaus Clonezilla-työkalulla sopii nykyaikana palvelimen ensisijaiseksi käyttöönottotavaksi vain tilanteissa, jossa palvelimia on vain muutamia tai mikäli palvelimiin tehdään harvoin muutoksia. Usein päivityksiä ja konfiguraatiomuutoksia tarvitsevien palvelimien käyttöönottoon Clonezilla ei ole varsinkaan yksinään ihanteellinen vaihtoehto. Tarvittavien levykuvien ajantasaisena pitäminen manuaalisesti veisi useissa tapauksissa kohtuuttomasti aikaa ja resursseja.

Säännöllisesti tapahtuva varmuuskopionti Clonezillalla sen sijaan voi olla edelleen ensi- tai toissijaisena palautustapana toimiva varmistamaan tietojen tallessaolo palvelinrikon sattuessa. Tällöinkin prosessin automatisointi on taikasana, jonka avulla kyseisen menetelmän käytön voi perustella. Säännöllisellä syklillä ihmisvoimin suoritettuna en näe menetelmää enää järkevänä kuin poikkeustapauksissa hyvin pienissä ympäristöissä.

Kickstart-konfigurointitiedoston hyödyntäminen jälkiasennusskripteineen sopii vastaavasti varsinaiseen palvelimen käyttöönottoon jo paremmin verrattuna pelkän levykuvan käyttöön ja itse asetuksienkin muuttaminen on tarvittaessa helpompaa. En suosittele kuitenkaan palvelimen saattamista tuotantokuntoon yksinomaan kickstartin ja post-install skriptien avulla kuin tapauksissa, jossa muutoksia palvelimelle tehdään jälkikäteen harvemmin ja palvelinten kokonaislukumäärä on suhteellisen alhainen.

Kickstart ei myöskään yleensä riitä yksinään palvelimen palautukseen, mutta sillä on kuitenkin edelleen useissa eri käyttötapauksissa perusteltu rooli osana nykyaikaisia palvelimen käyttöönotto- ja palautusprosesseja toimimalla palvelinasennuksen ensiaskeleena. Kickstartilla on siis suositeltavaa asentaa palvelimille OS konfiguroiduilla perusasetuksilla, jonka jälkeen sovelluspuoli sekä muut monimutkaisemmat konfiguroinnit on pitkällä tähtäimellä kannattavampaa hoitaa keskitetysti palvelinten provisioinnin mahdollistavalla sovelluksella.

Viimeinen testatuista työkaluista oli palvelinten konfiguraationhallintaan soveltuva Ansible, jota suosittelen hyödynnettävän toimeksiantajan kehityshankkeessa ensisijaisena työkaluna palvelinten käyttöönottamisessa ja päivittämisessä. Ansiblen roolit, playbookit ja host-ryhmät huolellisesti suunnittelemalla ja toteuttamalla on mahdollista vähentää merkittävästi palvelinten ylläpitäjän työmäärää erityisesti palvelinmäärien kasvaessa.

Lisäksi opinnäytetyössä on ollut lähtökohtana, ettei palautettavan palvelimen varsinaisella datalla kuten mediatiedostoilla ole välitöntä palauttamistarvetta, vaan ainoastaan käyttöjärjestelmän ja sovellusten asennukset määritettyine asetuksineen tulee pystyä palauttamaan ennalleen. Kyseisen harkitun rajauksen vuoksi Ansible soveltuu tarvittaessa myös merkittäväksi osaksi palautusprosesseja.

Selvitysten perusteella siis toimeksiantajan käyttötapauksessa tehokkain vaihtoehto palvelimen käyttöönotto- ja palautusprosesseja varten on Kickstartin ja Ansiblen yhdistelmäkäyttö. Kickstartin avulla tulisi asentaa alkuun varsinainen käyttöjärjestelmä konfiguroiden samalla OS:n tarpeelliset perusasetukset kuten levyosiot ja verkot. Tarkasti rajatun kickstart-asennuksen jälkeen loput toimenpiteet kuten sovellusten asennukset ja konfiguroinnit suosittelen suorittamaan Ansiblen playbookeilla.

8.3 Menetelmien jatkokehitys

Jo käytettävien tai tulevaisuudessa käyttöönotettavien menetelmien jatkokehitysideoista yksi tärkeimmistä on järjestelmän ylläpitäjien käyttöosaamisen syventäminen valittujen työkalujen osalta suuntaan, joka palvelee suoraan kehityshankkeen tarpeita.

Toinen ydinasia on hioa käyttöönotto- ja palautusprosessit siten, että jokaisella käytettävällä työkalulla on selkeästi määritelty rooli prosessissa ilman, että työkaluja käytetään päällekkäin identtisiin käyttötarkoituksiin. Mitä vähemmillä työkaluilla selviää – sitä parempi, mutta menetelmissä käytettävien työkalujen pitää myös täyttää kaikki vaadittavat ominaisuudet eheän käyttöönottoprosessn takaamiseksi.

Huolellisesti tehty suunnittelu on siis avainasemassa palvelinten käyttöönotto- ja palautusmenetelmiä valitessa sekä välttämättömiä prosesseja kehittäessä. Testattujen työkalujen osalta listaan alle vielä mainitsemisen arvoisia ideoita sovelluskäytöstä sekä muita kehitysideoita, jotka liittyvät tavalla tai toisella kyseiseen työkaluun.

Ansiblella suosittelen suunnittelemaan roolit siten, että kaikkia rooleja pystyy käyttämään eri playbookeilla ilman tarvetta muuttaa roolin tehtävien syntaksia aina uudestaan eri

käyttötapauksessa. Ideana on, että kerran luotu rooli olisi aina käyttökelpoinen uudestaan ja uudestaan. Tätä ajatustapaa tukemaan on olemassa myös oletusmuuttujat (”default variables”), joilla on mahdollista määrittää muuttujalle yleinen arvo, jota voi käyttää toisessakin roolissa. (Ansible 2017d.)

Lisäksi on olemassa web-palvelimelle asennettava ja selaimella hallittava – graafiseen käyttöliittymään perustuva Ansible Tower. Sovellus mahdollistaa playbookkien ajastamisen ja töiden (”jobs”) reaaliaikaisen seurannan dashboardin välityksellä. Towerissa on muutenkin kattavasti ominaisuuksia kuten tarkkaan yksilöitävät mallipohjat (”job templatet”), joissa määritetään käytettävä inventaario, käyttöoikeudet ja playbookit. Myös Towerin monipuolinen tunnustenhallinta tukee AD-integraatiota.

Ansible Towerin hinta tukipaketteineen on melko korkea, joten todellinen tarve ja sopivuus omiin käyttötarkoituksiin kannattaa varmistaa huolellisella testaamisella. Towerista on olemassa myös ilmainen open-source vaihtoehto nimeltään Semaphore, mutta kyseessä on luonnollisesti ominaisuuksiltaan karsittu versio, jolle ei ole saatavilla virallista tukea. Mainitut rajoitteet kannattaa siis huomioida mikäli harkitsee kyseisen vaihtoehdon käyttöönottoa vakavammin. (Semaphore 2017.)

Myös tavalliselle Ansiblelle löytyy tarvittaessa korvaavia vaihtoehtoja markkinoilta, mutta Ansiblen selkeinä etuina ovat keveys, joka ilmenee agentittomuutena kohdepalvelimilla ja suhteellisen matala oppimiskynnys, jonka myötä sovelluksen käyttö toimeksiantajan ympäristössä ei vaadi erityisen suuria panostuksia. En suosittele siis kartoittamaan Ansiblen kokonaan korvaavia vaihtoehtoja ilman erityisen painavia perusteita.

Kickstartin osalta kannattaa vastaavasti perehtyä tarkemmin syntaksin tukemiin asetuksiin, jotta itse käyttöjärjestelmän saa optimoitua kaikkien vaatimuksien mukaisesti heti kickstartin ajovaiheessa. Kickstart-tiedoston käyttökelpoisuus on niin ikään hyvä varmistaa tapauksissa, jossa palvelimia löytyy runsaasti lokaatioista, joihin ei ole fyysistä pääsyä. Lisäksi kickstartin integrointimahdollisuus suoraan palvelimen provisiointityökaluun kuten The Foremaniin on selvittämisen arvoinen asia.

Lopuksi olisi vielä hyvä selvittää palvelinten varmuuskopiointiprosesseihin varakeinojen tarpeellisuutta poikkeustilanteisiin, jolloin ensisijaiset menetelmät (Ansible+kickstart yhdistelmä tai vastaava) eivät olisikaan käytettävissä syystä tai toisesta. Samalla varmistuisi löytyykö testatulle Clonezilla-työkalulle hyödyllistä roolia esimerkiksi toissijaisena varmistuskeinona käyttämällä automatisoitua palvelinten levykuvan ottoa.

9 Projektin yhteenveto

Projektin aikana muovautuneet olennaisimmat tavoitteet tuli mielestäni saavutettua, niin testattujen menetelmien kuin henkilökohtaisen oppimisen osalta. Työssä tehtävä kokonaisuus hieman eli projektin aikana johtuen alkuperäisestä – liian laajasta ja kunnianhimoisesta aiheesta, mutta onneksi ”punainen lanka” löytyi riittävän ajoissa.

Opin opinnäytetyöprojektin aikana valtavasti, niin teoreettisia kuin käytännön asioita käsittelemistäni teknologioista – osa näistä oli jo aiemmin osattujen taitojen syventämistä runsaasti lähtötasoa pidemmälle ja osa vastaavasti täysin uuden asian omaksumista. Oletan myös tiedonhakutaitojen parantuneen työn valmistamisen aikana.

Itse työssä tuli käytettyä aika maksimaalista määrää eri työkaluja mitä ylipäätään on mahdollista yksittäisen opinnäytetyön kohdalla. VirtualBoxilla rakennettu testiympäristö sopi erinomaisesti käyttöönotto- ja palautusmenetelmien testaukseen ja voin vahvasti suositella kyseistä sovellusta kevyeen työpöytävirtualisointiin.

Käyttämäni lähteet nojasivat vahvasti elektroniseen materiaaliin ja perusteluni tälle on lähteiden ajantasaisuuden takaaminen nopeasti muuttuvalla IT-alalla. Projektinhallintaan sen sijaan jäi osittain parantamisen varaa – näin valtavasti vaivaa työn eteen, mutta toisaalta laatimani aikataulut olivat ylioptimistisia ja rehellisesti sanoen opinnäytetyön valmistuminen myöhästyi suunnitellusta.

Toivon ja uskon toimeksiantajan hyötyvän vaivannäöstäni erityisesti välillisin vaikutuksin sekä näen merkittävässä osassa opinnäytetyötä potentiaalia jatkojalostaa tehtyjä kokeiluja ja syntyneitä ideoita tuotantoon tähtäävän aidon palvelinympäristön hyväksi.

Lopuksi todettakoon helpotuksen olevan suuri, kun henkilökohtaisesti suuren panostuksen työn ohessa vaatinut opinnäytetyöprojekti on saatu valmiiksi, mutta toisaalta olo on samanaikaisesti myös hieman hämmentynyt projektin päätyessä.

Lähteet

Ansible 2013. The Origins of Ansible. Luettavissa: https://www.ansible.com/blog/2013/12/08/the-origins-of-ansible. Luettu: 14.2.2017.

Ansible 2017a. About Ansible. Luettavissa: http://docs.ansible.com/ansible/index.html. Luettu: 13.2.2017.

Ansible 2017b. How Ansible works. Luettavissa: https://www.ansible.com/how-ansible-works. Luettu: 14.2.2017.

Ansible 2017c. Installation. Luettavissa: http://docs.ansible.com/ansible/intro_installation.html#installation. Luettu: 26.2.2017.

Ansible 2017d. Ansible Tower user guide. Luettavissa: http://docs.ansible.com/ansible-tower/latest/html/userguide/overview.html. Luettu: 21.3.2017.

Ansible 2017e. Ansible setup module. Luettavissa: http://docs.ansible.com/ansible/setup_module.html. Luettu: 2.4.2017.

Ansible 2017f. Include a play or task list. Luettavissa: https://docs.ansible.com/ansible/include_module.html. Luettu: 5.4.2017.

The Balance 2016. What is Linux. Luettavissa: https://www.thebalance.com/what-is-linux-2533683. Luettu: 4.12.2016.

BBC 2016. ROM and RAM. Luettavissa: http://www.bbc.co.uk/schools/gcsebitesize/ict/hardware/1datastoragerev1.shtml. Luettu: 4.12.2016.

BPMN 2016. What is BPMN. Luettavissa: http://www.bpmn.org/. Luettu: 5.12.2016.

Cambridge Dictionary 2017. Trial and error. Luettavissa: http://dictionary.cambridge.org/dictionary/english/trial-and-error. Luettu: 25.2.2017.

CBR 2016. What is RHEL. Luettavissa: http://www.cbronline.com/what-is/what-is-rhel/. Luettu: 4.12.2016.

CentOS 2017a. CentOS Linux. Luettavissa:
https://www.centos.org/about/. Luettu: 22.2.2017.

CentOS 2017b. Kickstart konfigurator. Luettavissa:
https://www.centos.org/docs/5/html/Installation_Guide-en-US/ch-redhat-config-kickstart.html. Luettu: 22.2.2017.

Cisco 2009. Protocol basics: Secure Shell protocol. Luettavissa:
http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-46/124-ssh.html. Luettu: 4.12.2016.

Cisco 2016. What is a firewall. Luettavissa:
http://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html.
Luettu: 4.12.2016.

Clonezilla 2016. What is Clonezilla? Luettavissa:
http://clonezilla.org/. Luettu: 4.12.2016.

Cloudacademy 2015. Lean to build Ansible playbooks. Luettavissa:
http://cloudacademy.com/blog/building-ansible-playbooks-step-by-step/. Luettu: 4.4.2017.

Cnet 2010. Oracle buys Sun, becomes hardware company. Luettavissa:
https://www.cnet.com/news/oracle-buys-sun-becomes-hardware-company/.
Luettu: 7.12.2016.

Cyberciti 2015. Difference between OS provisioning, hardware provisioning and configuration management. Luettavissa:
https://www.cyberciti.biz/tips/server-provisioning-software.html#comment-610506.
Luettu: 12.2.2017.

Defit 2016. Unified Modeling Language. Luettavissa:
https://www.defit.org/uml/. Luettu: 4.12.2016.

E-Commerce times 2008. Sun gets desktop virtualization chops with Innotek buy.
Luettavissa: http://www.ecommercetimes.com/story/61661.html. Luettu: 7.12.2016.

DevOps 2016. FLAP, Part 1: Server provisioning. Luettavissa:
https://devops.com/flap-part-1-server-provisioning/. Luettu: 15.2.2017.

DigitalOcean 2016a. An introduction to configuration management. Luettavissa: https://www.digitalocean.com/community/tutorials/an-introduction-to-configuration-management. Luettu: 12.2.2017.

DigitalOcean 2016b. Configuration Management 101: Writing Ansible playbooks. Luettavissa: https://www.digitalocean.com/community/tutorials/configuration-management-101-writing-ansible-playbooks. Luettu: 14.2.2017.

DRBL 2017. About DRBL. Luettavissa: http://drbl.org/about/. Luettu: 18.2.2017.

Gartner 2016. Virtualization. Luettavissa: http://www.gartner.com/it-glossary/virtualization/. Luettu: 4.12.2016.

Haaga-Helia 2016. Järjestelmäasiantuntija. Luettavissa: http://www.haaga-helia.fi/fi/opinto-opas/koulutusohjelmat/tietojenkasittelyn-koulutusohjelma-helsinki-nuoret/opintojaksokuvauks-2?userLang=fi. Luettu: 28.11.2016.

Hitachi 2016. Definition of router. Luettavissa: http://hitachi-id.com/concepts/router.html. Luettu: 4.12.2016.

How-To Geek 2010. Clone a hard drive using an ubuntu live CD. Luettavissa: http://www.howtogeek.com/howto/19141/clone-a-hard-drive-using-an-ubuntu-live-cd/. Luettu: 13.2.2017.

How-To Geek 2011. What is a virtual machine hypervisor. Luettavissa: http://www.howtogeek.com/66734/htg-explains-what-is-a-hypervisor/. Luettu: 8.12.2016.

How-To Geek 2014. How to create and use virtual machines. Luettavissa: http://www.howtogeek.com/196060/beginner-geek-how-to-create-and-use-virtual-machines/. Luettu: 7.12.2016.

Hughey, T. 2009. The Traditional waterfall approach. Luettavissa: http://www.umsl.edu/~hugheyd/is6840/waterfall.html. Luettu: 4.12.2016.

IBM 2015. RPM and YUM package management. Luettavissa: http://www.ibm.com/developerworks/linux/library/l-lpic1-102-5/index.html. Luettu: 7.12.2016.

Indiana University 2013. What is the difference between a LAN, a MAN, and a WAN. Luettavissa: https://kb.iu.edu/d/agki. Luettu: 4.12.2016.

Indiana University 2016. What is FTP, and how do I use it to transfer files. Luettavissa: https://kb.iu.edu/d/aerg. Luettu: 10.12.2016.

Koudata 2010. Tietojärjestelmän suunnittelumallit. Luettavissa: http://koudata.fi/node/195. Luettu: 30.11.2016.

Lifehacker 2012. The best disk cloning app for Linux. Luettavissa: http://lifehacker.com/5891933/the-best-disk-cloning-app-for-linux. Luettu: 13.2.2017.

Lifewire 2015. VNC - Virtual Network Computing. Luettavissa: https://www.lifewire.com/vnc-virtual-network-computing-818104. Luettu: 10.12.2016.

Lifewire 2016. Introduction to MAC Addresses. Luettavissa: https://www.lifewire.com/introduction-to-mac-addresses-817937. Luettu: 4.12.2016.

Linoxide 2014. What is Mosh. Luettavissa: http://linoxide.com/how-tos/mosh-alternative-ssh/. Luettu: 10.12.2016.

Linuxlinks 2016. Clonezilla. Luettavissa: http://www.linuxlinks.com/article/20080412135215257/Clonezilla.html. Luettu: 13.2.2017.

Mobatek 2016a. MobaXterm front page. Luettavissa: http://mobaxterm.mobatek.net/. Luettu: 10.12.2016.

Mobatek 2016b. MobaXterm features. Luettavissa: http://mobaxterm.mobatek.net/features.html. Luettu: 10.12.2016.

Microsoft 2011. Hypervisors. Luettavissa: https://blogs.technet.microsoft.com/chenley/2011/02/09/hypervisors/. Luettu: 10.12.2016.

Microsoft 2016. Remote Desktop Protocol. Luettavissa: https://msdn.microsoft.com/en-us/library/aa383015.aspx. Luettu: 10.12.2016.

Oracle 2010. The Oracle VM Product Line Welcomes Sun. Luettavissa: https://web.archive.org/web/20100407074836/http://blogs.oracle.com/virtualization/2010/02/the_oracle_vm_product_line_wel.html. Luettu: 7.12.2016.

PCMag 2016a. Defifition of CPU. Luettavissa: http://www.pcmag.com/encyclopedia/term/40436/cpu. Luettu: 4.12.2016.

PCMag 2016b. Definition of IP address. Luettavissa: http://www.pcmag.com/encyclopedia/term/45349/ip-address. Luettu: 4.12.2016.

PCMag 2016c. Definition of cloning software. Luettavissa: http://www.pcmag.com/encyclopedia/term/39838/cloning-software. Luettu: 12.2.2017.

PCMag 2016d. Definition of UUID. Luettavissa: http://www.pcmag.com/encyclopedia/term/53596/uuid. Luettu: 18.2.2017.

PCWorld 2014. When to image a hard drive, and when to clone it. Luettavissa: http://www.pcworld.com/article/2847308/when-to-image-a-drive-and-when-to-clone-it.html. Luettu: 12.2.2017.

Persse, James R. 2006. Process improvement essentials. Luettavissa: http://flylib.com/books/en/4.259.1.53/1/. Luettu: 5.12.2016.

Puppet 2017. Puppet Enterprise and open source Puppet. Luettavissa: https://puppet.com/product/puppet-enterprise-and-open-source-puppet. Luettu: 14.2.2017.

Pykickstart 2016. What are kickstart installations? Luettavissa: http://pykickstart.readthedocs.io/en/latest/kickstart-docs.html#chapter-1-introduction. Luettu: 12.2.2017.

Rankin, K. 2013. How to deploy a server. Luettavissa: http://www.linuxjournal.com/content/how-deploy-server. Luettu: 16.3.2017.

Red Hat 2015. Red Hat to acquire IT automation and DevOps leader Ansible. Luettavissa: https://www.redhat.com/en/about/press-releases/red-hat-acquire-it-automation-and-devops-leader-ansible. Luettu: 14.2.2017.

Red Hat 2017a. What are kickstart installations? Luettavissa: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Installation_Guide/ch-kickstart2.html#s1-kickstart2-whatis. Luettu: 5.4.2017.

Red Hat 2017b. Kickstart syntax reference. Luettavissa: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Installation_Guide/sect-kickstart-syntax.html. Luettu: 23.2.2017.

Rsnapshot 2017. Filesystem snapshot utility based on rsync. Luettavissa: http://rsnapshot.org/. Luettu: 15.2.2017.

Scrumguides 2016. Definition of Scrum. Luettavissa: http://www.scrumguides.org/scrum-guide.html#definition. Luettu: 4.12.2016.

Select Business Solutions 2016. What is the waterfall model. Luettavissa: http://www.selectbs.com/analysis-and-design/what-is-the-waterfall-model. Luettu: 5.12.2016.

Semaphore 2017. Ansible-Semaphore. Luettavissa: https://github.com/ansible-semaphore/semaphore. Luettu: 20.3.2017.

Slashroot 2013. Linux grub (Grand Unified Bootloader) tutorial. Luettavissa: http://www.slashroot.in/linux-grubgrand-unified-bootloader-tutorial. Luettu: 13.2.2017.

Softpedia 2016. CentOS Linux 7 (1611) Released, It’s derived from Red Hat Enterprise Linux 7.3. Luettavissa: http://news.softpedia.com/news/centos-linux-7-1611-released-it-s-derived-from-red-hat-enterprise-linux-7-3-510911.shtml. Luettu: 13.2.2017.

Spacewalk 2015. What can Spacewalk do? Luettavissa: http://spacewalk.redhat.com/. Luettu: 14.2.2017.

Study CCNA 2016. Telnet & SSH. Luettavissa: http://study-ccna.com/telnet-ssh/. Luettu: 10.12.2016.

Superuser 2015. Difference between (CentOS) distribution types. Luettavissa: http://superuser.com/a/968894. Luettu: 7.12.2016.

Technopedia 2016. Disk image. Luettavissa: https://www.techopedia.com/definition/12705/disk-image. Luettu: 4.12.2016.

Tech Terms 2014. Server. Luettavissa: http://techterms.com/definition/server. Luettu: 4.12.2016.

Tecmint 2014. 8 Best open-source ”Disk cloning/backup” software for Linux servers. Luettavissa: http://www.tecmint.com/linux-disk-cloning-tools/. Luettu: 14.2.2017.

Tervonen, S. 2006. Laadunhallinta osana organisaation toimintaa. TieVie - asiantuntijakoulutus. Organisatorinen muutos. Oulu. Luettavissa: http://tievie.oulu.fi/koulutusresurssit/kalvot/2006/Oulu_15op/tervonen_laadunhallinta6.pdf. Luettu: 5.12.2016.

Tomsitpro 2014. Open source configuration management tools. Luettavissa: http://www.tomsitpro.com/articles/open-source-cloud-computing-software,2-754-10.html. Luettu: 14.2.2017.

University of North Carolina 2012. Research computering – what is X11 forwarding. Luettavissa: http://help.unc.edu/help/research-computing-what-is-x11-forwarding/. Luettu: 10.12.2016.

Unixmen 2015. Iptables vs. Firewalld. Luettavissa: https://www.unixmen.com/iptables-vs-firewalld/. Luettu: 7.12.2016.

Upguard 2017. Salt vs. Chef. Luettavissa: https://www.upguard.com/articles/salt-vs-chef. Luettu: 14.2.2017.

VirtualBox 2016a. VirtualBox manual. Luettavissa: https://www.virtualbox.org/manual/ch01.html#features-overview. Luettu: 7.12.2016.

VirtualBox 2016b. Changelog for VirtualBox 5.1. Luettavissa: https://www.virtualbox.org/wiki/Changelog. Luettu: 13.2.2017.

VirtualBox 2016c. End-user documentation. Luettavissa: https://www.virtualbox.org/wiki/End-user_documentation. Luettu: 8.12.2016.

VirtualBox 2016d. VirtualBox front-page. Luettavissa: https://www.virtualbox.org/. Luettu: 4.12.2016.

VirtualBox 2016e. Features overview. Luettavissa: https://www.virtualbox.org/manual/ch01.html#features-overview. Luettu: 10.12.2016.

VirtualBox 2016f. Introduction to networking modes. Luettavissa: https://www.virtualbox.org/manual/ch06.html. Luettu: 11.12.2016.

VMware 2016a. What is virtualization. Luettavissa: http://www.vmware.com/solutions/virtualization.html. Luettu: 4.12.2016.

VMware 2016b. Workstation Player FAQ. Luettavissa: http://www.vmware.com/products/player/faqs.html. Luettu: 7.12.2016.

VMware 2016c. Using VMware Workstation Player for Windows. Luettavissa: http://pubs.vmware.com/player-12-windows/topic/com.vmware.ICbase/PDF/workstation-player-12-windows-user-guide.pdf. Luettu: 8.12.2016.

VMware 2016d. Compare. Workstation Player or Workstation pro. Luettavissa: http://www.vmware.com/products/workstation.html. Luettu: 10.12.2016.

VMware 2016e. VMware store. Workstation 12.5 Pro. Luettavissa: http://store.vmware.com/store/vmwde/en_IE/DisplayProductDetailsPage/ThemeID.29219600/productID.323427300?src=eBIZ_StoreHome_Featured_WorkstationPro_EU. Luettu: 10.12.2016.

XebiaLabs 2010. Deployment automation vs. Server provisioning. Luettavissa: https://blog.xebialabs.com/2010/12/20/deployment-automation-vs-server-provisioning/. Luettu: 12.2.2017.

Zdnet 2011. What sucks worse than Oracle’s Virtualbox. Luettavissa: http://www.zdnet.com/article/what-sucks-worse-than-oracles-virtualbox/. Luettu: 7.12.2016.