TAMPEREEN
AMMATTIKORKEAKOULU

VIRTUALISOINTI OHJELMISTOTESTAUKSESSA

Antti Laine

Opinnäytetyö
Joulukuu 2017
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

LAINE ANTTI:
Virtualisointi ohjelmistotestauksessa

Opinnäytetyö 21 sivua, joista liitteitä 0 sivua
Joulukuu 2017

Tämän opinnäytetyön tavoitteena on kehittää tietopaketti virtualisoinnin käytöstä ohjelmistotestauksessa sekä siihen liittyvistä aiheista, jonka avulla aiheesta saa peruskäsityksen kaikesta siihen liittyvästä perusasiasta. Työssä käydään perusteet virtualisoinnista, ohjelmistotestauksesta sekä virtuaalisen testausympäristön rakentamisesta. Näissä aihetta käsitellään yleisillä malleilla havainnollistaen, käydään läpi eri menetelmien hyötyjä ja haittoja. Ohjelmistotestauksessa käsitellään myös pilviratkaisuja. Teoriaan ja ohjeisiin tutustumalla on kirjoitettu lyhyesti tärkeiden aiheiden pääkohdat niin, että se ohjaa ohjelmiston testauksessa kustannustehokkaaseen laadukkaaseen testaukseen.

Ensin käydään läpi virtualisoinnista perustietoa ja sitten perehdytään sovellus- ja laitteistovirtualisointiin. Virtualisointi ja ohjelmistotestaus ovat pääaiheita ja molemmat käydään läpi. Testauksesta käydään kaikki perusvaiheet läpi sekä automaattiset, että manuaaliset testaustavat. Tarkoituksena on opettaa lukijalle perusmallit ja menetelmät. Lopuksi käydään tiivistettynä perusasiat läpi virtuaalisen testausympäristön rakentamisesta.

Asiasanat: virtualisointi, ohjelmistotestaus, testausympäristö

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software technology

ANTTI LAINE:
Virtualization in software testing

Bachelor's thesis 21 pages, appendices 0 pages
December 2017

The purpose of this thesis is to develop an informative document on the use of virtualization in the software testing and related topics, which will enable the subject to get a basic idea of all the related basic issues. The thesis deals with the basics of virtualization, software testing and the development of a virtual testing environment. These topics are dealt with by common models, illustrating the benefits and disadvantages of different methods. Software testing also deals with cloud solutions. By studying theory and instructions, the key points of important topics have been briefly written to guide the software to cost-effective high-quality testing.

First, basic information on virtualization is introduced and then familiarized with application and hardware virtualization. Virtualization and software testing are the main topics. All the basic steps are tested through both automatic and manual test methods. The aim is to teach the reader basic models and methods. Finally, the basics are summarized through the construction of a viral testing environment.

Key words: virtualization, software testing, testing environment

SISÄLLYS

LYHENTEET JA TERMIT

X86	on Intelin kehittämä ja valmistama suoritinarkkitehtuuri
ARM	on 32-bittinen mikroprosessoriarkkitehtuuri
kernel	määrittelee käyttöjärjestelmän rakenteen, luokituksen ja ominaisuudet
Intel VT	on Virtualisointitekniikka X86-arkkitehtuurille
AMD-V	on Virtualisointitekniikka X86-arkkitehtuurille
Hypervisor	on tekniikka joka eristää ohjelmiston tietokoneen laitteistosta
FTP	on TCP-protokollaa käyttävä tiedostonsiirtomenetelmä kahden tietokoneen välille
SFTP	on salattu versio FTP-tekniikasta
CLI	on tekstipohjainen komentoliittymä
API	tarkoittaa rajapintaa jolla ohjelmistot voivat vaihtaa dataa ja tehdä pyyntöjä toisilleen
ISO	on virtuaalinen levynkuva cd:stä tai dvd:stä
SSH	salattu protokolla tietoliikenteeseen
VNC	on tekniikka graafiseen tietokoneen etäkäyttöön

1 JOHDANTO

Opinnäytetyön aiheena oli luoda tiivis dokumentti, jossa on koottuna virtualisointi, ohjelmistotestaus sekä virtuaalisen testausympäristön rakentaminen. Aiheista oli tarkoitus kirjoittaa niin, että ne ottavat toisensa huomioon. Työ tehtiin itsenäisesti teoriapainotteisesti. Ohjelmistotestaus on tärkeä osa ohjelmistokehitystä ja virtualisointi on näiden tärkeä osa myös.

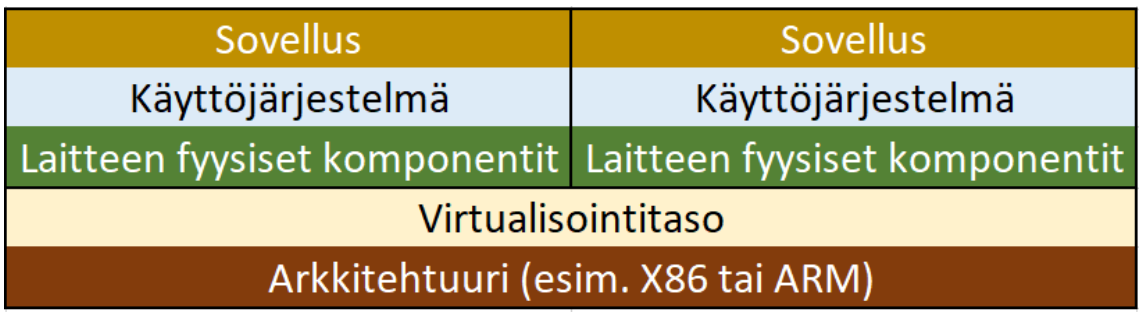
Työn motivaationa toimi kiinnostus virtualisointiin ja aiheen laajuus inspiroi kirjoittamaan yhden dokumentin jossa käydään kaikki perusasiat läpi. Dokumentissa käydään aihealue läpi teorian osalta ja sitten kerrotaan perusteet testiympäristön rakentamisesta. Lopuksi kerrotaan miten tavoitteisiin päästiin ja miten eri aiheista kirjoittaminen onnistui.

2 VIRTUALISOINTI

Virtualisointi tarkoittaa sitä, että tehdään virtuaalinen versio jostakin laitteesta. Näitä voivat olla esimerkiksi palvelimet, verkot ja käyttöjärjestelmät. Myös fyysisen kiintolevyn osioiminen useaksi osioksi voidaan mieltää virtualisoinniksi, koska luodut osiot ovat ikään kuin virtuaalisia kiintolevyjä. Yleensä käyttäjän hyödyntäessä virtuaaliset versiot eivät eroa alkuperäisistä. Pieniä eroja tietenkin on, esimerkiksi suorituskyvyssä. (Vangie Beal, 2017)

Virtualisointitapoja on esimerkiksi täysvirtualisointi, jossa järjestelmän laitteisto virtualisoidaan. Tällöin ohjelmisto toimii lähes alkuperäisellä yhteensopivuudella tässä virtuaalisessa versiossa. Toinen on paravirtualisointi joka on helpompi toteuttaa. Tässä tavassa virtualisoidaan laitteisto osittain ja näille laitteiden osille annetaan yleensä oma uniikki osoite niiden tunnistamiseen. Virtualisointi voidaan jakaa alla listattuihin alueisiin (Colan Infotech, 6.):

- verkon virtualisointi (Network Virtualization)
- tietovarastovirtualisointi (Storage Virtualization)
- palvelinvirtualisointi (Server Virtualization)
- datavirtualisointi (Data Virtualization)
- työpöytävirtualisointi (Desktop Virtualization)
- sovellusvirtualisointi (Application Virtualization)
- laitteistovirtualisointi (Hardware Virtualization).

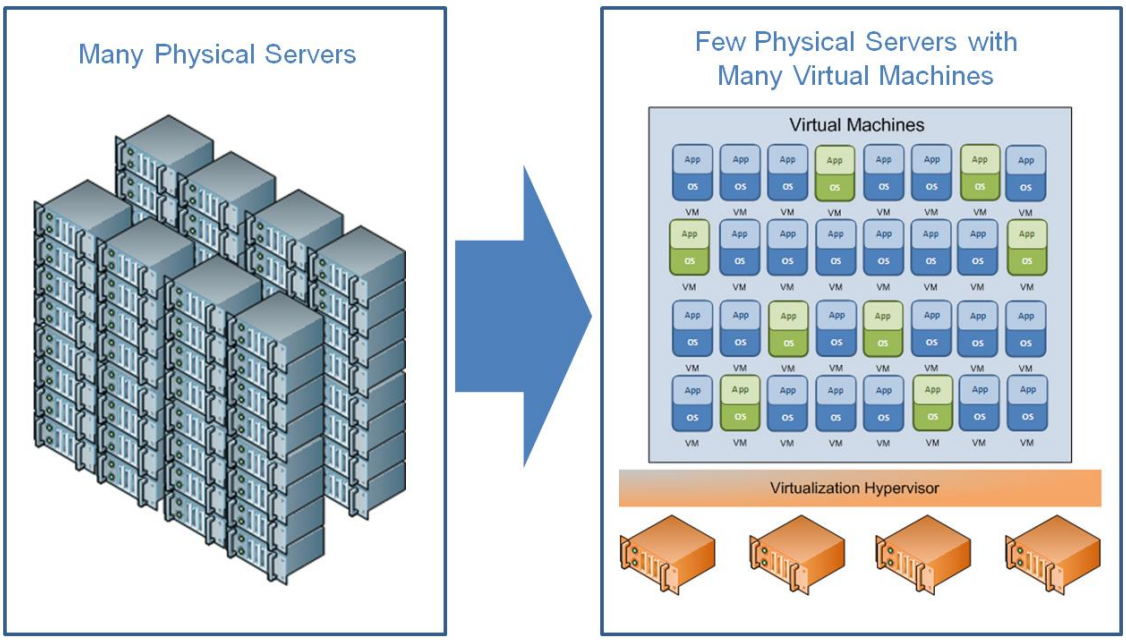


KUVA 1. Esimerkki kahdesta virtualisoidusta laitteesta yhdellä fyysisellä laitteella (Antti Laine, 2017)

2.1 Virtualisoinnin hyödyt

Suurin hyöty virtualisoinnissa on taloudellinen. Ohjelmistotestaajat ja -suunnittelijat voivat testata ohjelmiaan ja ohjelmistopäivityksiään turvallisesti ja laajallakin skaalalla ennen asiakkaille jakamista. Eri asioiden virtualisoinnista on erityistä hyötyä ohjelmistotestauksessa, koska laitteiden komponentit sekä ohjelmisto ovat virtualisoituja joten niitä voidaan muokata ja tutkia tarpeiden ja tilanteiden mukaan. Laitteista voidaan tallentaa usein tilannekuvia joita voidaan käyttää useasti eri testauksen vaiheissa. Fyysisiin verrattuna virtuaaliset on mahdollista eristää muusta järjestelmästä.

Tiedontallennuslaitteiden virtualisoinnissa voidaan esimerkiksi yhdistää useita verkkotallennuslaitteita yhdeksi virtuaaliseksi tallennuslaitteeksi. Yksi fyysinen serveri voidaan jakaa virtualisoinnilla useisiin pienempiin virtuaalisiin servereihin joissa virtuaalisia komponentteja voidaan rajoitetusti muokata tarpeiden mukaan ja käyttöjärjestelmä voi olla jokaisessa näissä eri. Myös käyttöjärjestelmätason virtualisointia voidaan tehdä jossa toimitaan kernel-tasolla. Verkon resursseja voidaan käyttää virtuaalilaitteilla fyysisessä verkossa. Ohjelmia voidaan myös virtualisoida juurikin ohjelmistotestausta varten. (Vangie Beal, 2017)



KUVA 2. Virtualisoinnin hyöty näkyy sähkönkulutuksessa ja näin ollen kustannuksissa (Boost IT, 2016)

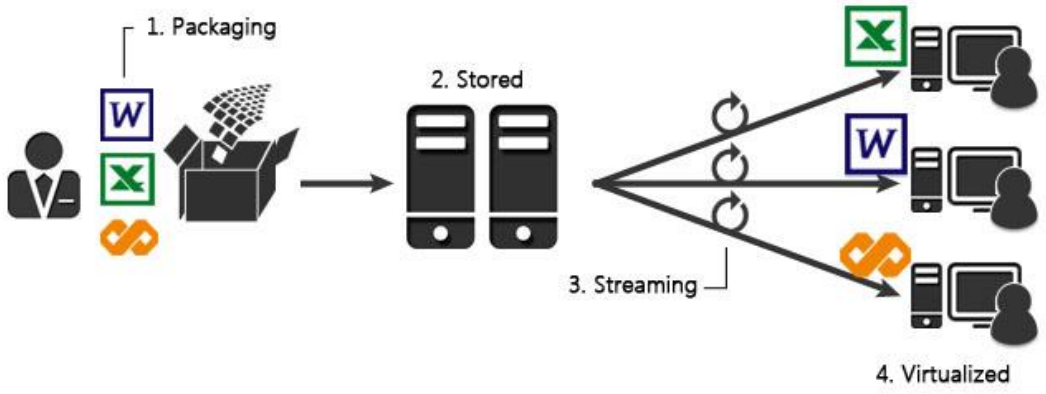
Virtualisointitekniikka on yksi tärkeimmistä asioista it-maailmassa. Monissa ohjelmointiyrityksissä on tarve servereille ohjelmointitestausta varten kun oman työaseman tehot eivät skaalaudu tarpeeksi isoksi. Tällöin serverien virtualisointi on järkevää, koska se säästää tilaa, vähentää sähkönkulutusta sekä nopeuttaa ja helpottaa testausta kun virtuaalisia laitteita ja niiden komponentteja voidaan luoda aina testauksesta riippuen tarpeen mukaan. Kymmenen fyysisen serverin sijasta voidaan virtualisoida nämä kymmenen yhteen ja asentaa tarpeen mukaan eri käyttöjärjestelmät jokaiseen laitteeseen sekä virtualisoida myös oheislaitteita. Myös tilanteet joissa samalla fyysisellä serverillä asennetut ohjelmat menevät esimerkiksi kirjastojen päällekkäisyyksien takia sekaisin tai aiheuttavat muita ristiriitoja, voidaan välttää kun ohjelmat ovat omilla virtuaaliservereillään juuri niille asennetuilla kirjastoillaan. Kun serverit ovat virtuaalisia, voidaan raudan resursseja dynaamisesti jakaa niille servereille, jotka sitä eniten tietyllä hetkellä tarvitsevat. Virhetilanteissa voidaan välittömästi palauttaa levynkuva uuteen virtuaaliserveriin ja ratkaista ongelmaa sillä välin kun toinen serveri toimii alkuperäisen tilalla väliaikaisesti. (Thomas Burger, 2017, 3.)

Virtualisointi on servereiden lisäksi erittäin hyödyllistä normaaleilla työasemillakin, koska voidaan virtualisoida testauksessa tarvittava käyttöjärjestelmä sen sijaan, että asennettaisiin se fyysiseen työasemaan käytössä olevan käyttöjärjestelmän rinnalle. Voidaan siis testata jotain sovellusta esimerkiksi Windows XP -käyttöjärjestelmällä kun fyysisessä työasemassa on asennettuna Mac OS. Tämä myös vähentää testauksen keskeyttäviä virheitä kuten muistivikoja tai korruptoituneita ohjelmistoajureita. (Thomas Burger, 2017, 3.)

2.2 Sovellusvirtualisointi

Sovellusvirtualisointi tarkoittaa sitä, että ohjelma virtualisoidaan eristetysti muista ohjelmista ja se on mahdollista ladata erilliseltä serveriltä. Itse virtualisointialusta on asiakaslaitteessa. Sovelluksen tarvitsemia resursseja voidaan joko ladata reaaliajassa, tallentaa asiakaskoneeseen tai ladata se pysyvästi asiakaskoneeseen. Käyttäjällä on siis mahdollisuus käyttää perustoimintoja kuten hiiri ja näppäimistö, mutta ohjelma toimii omassa ikkunassaan ja suoritetaan oikeasti eri paikassa. Nykyään tämä toteutetaan erittäin sulavasti ja monesti käyttäjä ei tätä edes huomaa. Nykyään yrityksissä on monien ohjelmien koh-

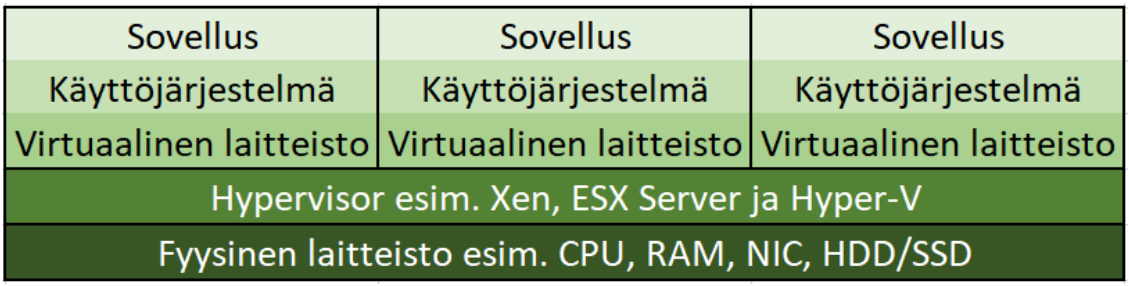
dalla helpompi ylläpitää palvelua serverillä ja työntekijät sitten käyttävät tätä omilla koneillansa etänä. Ohjelmatestausmielessä serveriltä voidaan jakaa tietty resurssi käyttäjätunnuksen ja salasanan avulla jolla ohjelmoimaansa tai ylipäätään testaamaansa ohjelmaa voidaan testata laajemmalla skaalalla. (Margaret Rouse, 2017, 9.)



KUVA 3. Sovelluksen virtualisointi serveriltä (SOFTonNET, 2017)

2.3 Laitteistovirtualisointi

Laitteistovirtualisoinnilla tarkoitetaan koodin integroimista laitteistoon. Esimerkiksi Intel:llä on tekniikka nimeltään Intel VT ja AMD:llä tekniikka AMD-V. Näillä tekniikoilla voidaan ajaa muokkaamattomia virtuaalikoneita ilman CPU:n emulointia. Tätä suositellaan erityisesti tuotannossa. Virtuaalikoneissa ei ole ollenkaan isäntäkäyttöjärjestelmää välissä, vaan koodi näyttää isäntäpalvelimen resurssit suoraan virtuaalikoneelle. Näin säästytään tiheältä päivitysten asentamiselta ja ylläpito on helpompaa. Näitä nimitetään nimellä Hypervisor. Tähän perustuvia järjestelmiä ovat esimerkiksi Xen, VMware ESX Server ja Microsoft:n Hyper-V. (Virtuatopia, 2016, 5.)



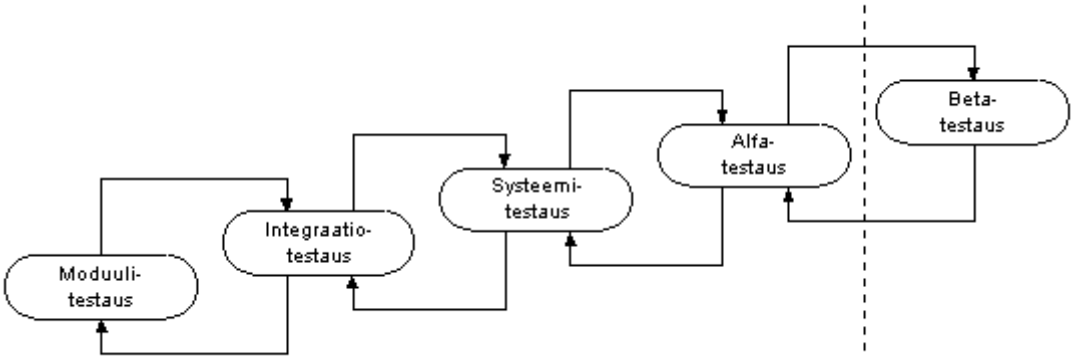
KUVA 4. Laitteistovirtualisoinnissa virtuaalikoneiden ja fyysisen tason välissä on Hypervisor (Antti Laine, 2017)

3 OHJELMISTOTESTAUS

Ohjelmistotestauksessa on tarkoituksena saada ohjelmiston virheet ilmenemään sen käytössä jotta ne voidaan korjata. Testauksessa yritetään todistaa se, että tämä täyttää asiakkaan vaatimukset. Testausta voidaan tehdä ohjelmiston kehityksen eri vaiheissa ja näin ennaltaehkäistä virheitä. Lopuksi tietenkin tehdään suurempia testejä, jotta ohjelmiston laadullisuudesta päästään varmuuteen. Tavoitteena on selvittää ja nähdä ohjelmiston viat, häiriöt ja mahdollisesti myös puutteet sekä varmistaa sen yhteensopivuus toisten käytettävien ohjelmistojen kanssa, jos niitä on. Ohjelmiston ollessa kesken voidaan tehdä dynaamista testausta koodille joka tarkoittaa sitä, että sitä ajetaan testitapauksilla läpi ja näin selviää paljon tärkeitä asioita sen toimivuudesta. (Cem Kaner, 2006)

3.1 Testauksen eri vaiheet

Ohjelmakoodin testauksen kulku etenee yleensä monilla erilaisilla menetelmillä ja tavoilla. Testaaminen aloitetaan vaatimuusanalyysistä jossa kesken suunnitteluvaiheen testaajat selvittävät mitä on mahdollista testata ja millä parametreillä ne toimivat. Testaajat tekevät tätä kehittäjien kanssa yhteistyössä jotta lopputulos on mahdollisimman hyvä. Testauksesta tehdään testaussuunnitelma. Tähän kirjataan eri testausmenetelmät joita yleensä ovat testiskenaariot, -tapaukset, -data ja -skriptit. Testit voidaan myös jakaa suuremman aikavälin laajempiin vaiheisiin joita voivat olla moduuli-, integraatio-, systeemi-, alfa- sekä betatestaus (KUVA 5.). Näistä kaikista ajallisesta koosta huolimatta raportoidaan testaamisen lopuksi, tehdään raportit siitä miten se meni ja päätetään onko ohjelmisto julkaisukelpoinen.

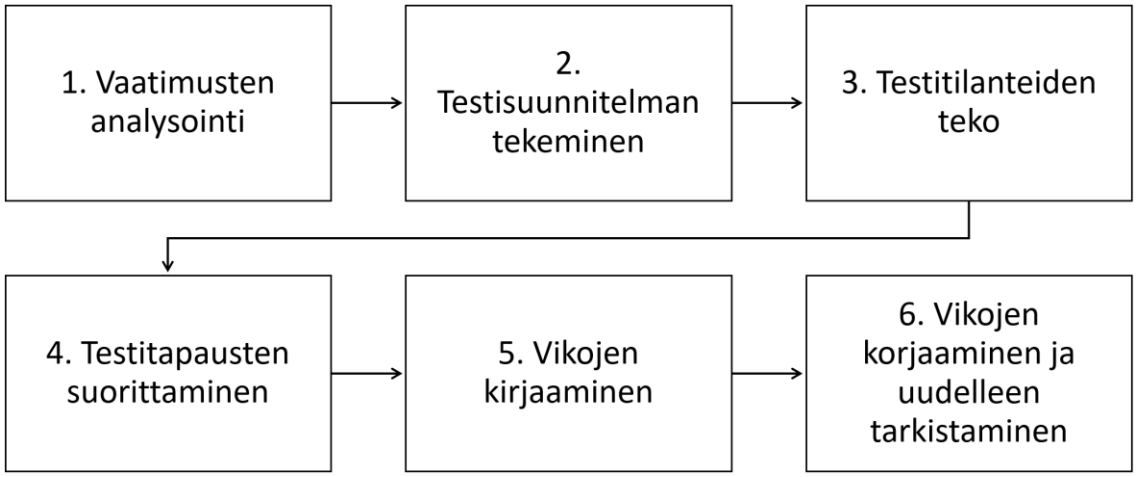


KUVA 5. Testauksen tasot (Tuomas Kautto, 1996)

Tämän jälkeen on mahdollista tehdä virheanalyysi kehitysryhmän kanssa asiakkaan osallistuessa tähän myös. Näin saadaan selville se miten ja mitä korjataan tai hylätään koodissa. Sitten on mahdollista testata havaittujen vikojen ilmeneminen korjausten tekemisen jälkeen ja tehdä regressiotestaus korjatulle koodille. Tämä tarkoittaa sitä, että ohjelmistoa testataan erilaisilla uusilla testiskenaarioilla ja uudella laitteistolla. Näin selviää uusienkin vikojen ilmeneminen ja ne voidaan vielä korjata. Testauksen suoritettua loppuun arkistoidaan kaikki mahdollinen tuloksia, lokeja, kaapattua dataa ja dokumentteja myöden jotta myöhemmissä projekteissa voidaan hyötyä jo tehdystä ohjelmistotestauksesta. (Jiantao Pan, 1999)

3.2 Manuaalinen

Manuaalinen testaaminen tarkoittaa prosessia minkä aikana on tarkoituksena löytää ohjelmaviat ja -bugit. Testaaja toimii tässä ohjelmiston käyttäjän roolissa, ja näin saadaan selville, toimiiko ohjelma, kuten on tarkoituksena. Ennen testiä tehdään dokumentti jossa määritellään eri tavat joilla käyttäjä mahdollisesti tulee käyttämään ohjelmistoa. Nämä tavat kattavat suurimman osan ohjelmiston käyttötilanteista. Kaikki eri tilanteet testaaja tekee manuaalisesti ilman ohjelmien automaattista apua. (Lakshay Sharma, 2016)



KUVA 6. Manuaalisen testaamisen prosessi (Antti Laine, 2017)

Kun ohjelmistoa testataan, täytyy ensin analysoida mitä vaaditaan. Sen jälkeen tehdään kunnollinen suunnitelma mitä ja miten testaaminen tehdään. Manuaalisessa testauksessa kaikki tehdään itse joten testitilanteet joita tullaan tekemään, täytyy tehdä etukäteen. Sitten voidaan suorittaa eri käyttäjätapaukset ja samalla kirjataan mitä vikoja on tullut ilmi

testauksen aikana. Nämä ilmenevät viat kirjataan ylös jotta ne voidaan korjata. Lopuksi korjatuissa tapauksissa testataan niitä uudelleen niin kauan, että ne todetaan korjatuiksi.

3.3 Automaattinen

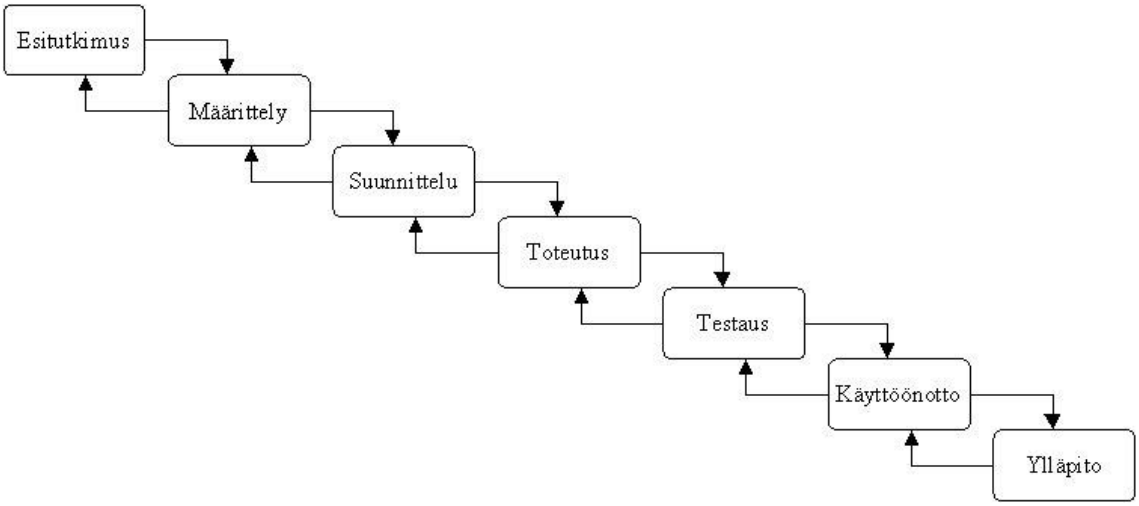
Automaattinen ohjelmistotestaus on yksinkertaisesti prosessi jossa automatisoidaan ohjelmiston testauksen vaiheet. Tämä vaatii jonkin ohjelmiston jolla voidaan tehdä skriptejä joilla ohjelmiston eri testauksen vaiheet voidaan automatisoida. Automatisoidun testauksen hyötyjä ovat esimerkiksi se, että eri testitapauksia voidaan toistaa nopeasti ja helposti useita kertoja peräkkäin. Testejä voidaan suorittaa rinnakkain useita ja testaamisessa tapahtuu vähemmän vahinkoja kun ihminen ei ole suorittamassa testejä. Tietenkin edellämainittujen asioiden vuoksi myös säästetään rahaa ja aikaa.

Ohjelmiston testauksen automatisoinnin voi aloittaa miettimällä, minkälaista testausta ohjelmistolle aikoo tehdä ja selvittää mikä työkalu tähän sopii parhaiten. Sen jälkeen suunnittelee rungon (framework) ja aloittaa kirjoittamaan skriptejä joilla lopullista testausta tehdään. Myös vikojen raportointi voidaan suunnitella automaattiseksi. Ohjelmistokehityksessä täytyy myös ottaa se huomioon, että testauksessa käytettäviä skriptejä täytyy myös ylläpitää.

Framework tarkoittaa testauksen automatisointiin tarkoitetun ohjelmiston runkoa. Käytännössä sitä, että miten käsitellään saatua dataa, ohjelmointistandardeja, tietojen raportointia ja lokitiedostojen kirjaamista.

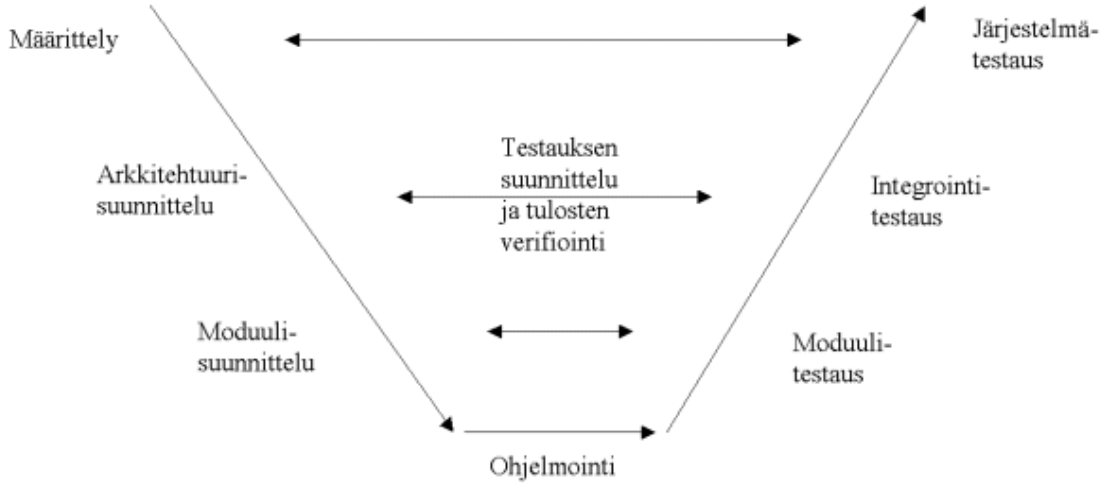
3.4 Vesiputous- ja V-malli

Vesiputouksen ideana on havainnollistaa selkeä listaus ohjelmiston valmistuksen eri vaiheista. Tästä on useita erilaisia versioita, mutta yleisimpänä on ns. vaihejakomalli. Vaiheita ei ole tarkoitus kuitenkaan välttämättä seurata järjestyksessä. Vesiputousmallia ovat seuranneet yleensä suunnitelmaa lähtökohtaisena pidettävät menetelmät.



KUVA 7. Yleinen vesiputousmalli (okol.org, 2004)

Vesiputousmallia käytetään nykyäänkin pienissä projekteissa, mutta on siitä huono, että testaus tehdään liian myöhään. Jos vikoja ilmenee paljon, ei niille enää tässä vaiheessa ehditä juuri tehdä mitään. Ikään kuin tämän takia kehitettiin ns. V-malli jossa testauksen suunnittelu tehdään jo aikaisemmassa vaiheessa sekä itse testaus heti kun dataa on saatavilla (KUVA. 6.). Näin aikaa käytetään hyvin hyödyksi ja välitarkastusten ansiosta virheiden korjausta tehdään pienemmissä erissä.



KUVA 8. V-malli (Petri Vakevainen, 2001)

3.5 Ketterät menetelmät

Ohjelmistokehitys on jatkuvaa uuden opettelua. Tavat ja käsitykset muuttuvat jatkuvasti. Ketteryys ohjelmistokehityksessä tarkoittaa sitä, että pysytään ajantasalla uusista tekniikoista, mukaudutaan erilaisiin tilanteisiin ja ajatellaan muutos positiivisena asiana. Tätä kutsutaan ketteräksi testaamiseksi (agile) joka voi olla esimerkiksi tutkivaa testausta. Siinä on tarkoitus jatkaa testejä niissä saatavien tulosten perusteella.

Tutkivassa testauksessa ohjelmistoa käytetään normaalisti, mutta yritetään kokoajan oppia siitä lisää ja kirjataan havaintoja myöhempää kehitystä varten talteen. Tavoitteena voi olla joko uuden oppiminen ohjelmistosta tai sitten sen rikkominen virhetilanteiden synnyttämiseksi. Selvitetään aluksi ohjelmiston tarkoitus, käyttö ja myös päivitetään testaajat uusien ominaisuuksien tasalle. Selvitetään mitä asiakas arvostaa ohjelmiston toiminnassa eniten ja mahdollisesti panostetaan siihen testausvaiheessa. Myös ohjelmiston osiin joissa on suuri riski asioiden mennä pieleen, täytyy panostaa. Käyttöskenaariot, yleistieto toiminnasta, mahdolliset poikkeukset toiminnassa sekä erityisesti tahallinen väärinkäyttö ovat hyviä asioita selvittää ennen testausta.

Tarkoitus on tehdä mahdollisimman paljon testejä jotta tuloksia saadaan myös paljon. Jätetään suunnittelu vähemmälle ja panostetaan siihen, että saadaan realistisia tuloksia eikä luoteta vain vanhoihin dokumentteihin. Ohjelmisto on tarkoitus ymmärtää ja tehdä havaintoja jotta tunnistetaan mahdolliset ongelmat ja virheet. Yritetään saada ohjelma rikki keinolla millä hyvänsä, koska ohjelmistoissa on aina virheitä. (Matti Vuori, 2010)

3.6 Testitilanteet

Test case tarkoittaa tilannetta jossa testataan ohjelmiston toimivuus ja tarkastellaan sen käyttäytymistä muutenkin. Testitilanteet on tarkoitus luoda niin, että ne ovat mahdollisimman yksinkertaisia, tehokkaita ja jokaisen helppo ymmärtää. Kun testitilanteita suunnitellaan useita, on niiden toimittava itsenäisesti ja näin ollen raportoitava omista toiminnoistaan. Jotta tilanteet erotellaan selkeästi toisistaan, tarvitaan uniikki tunnistenumero eli id. Alkutietoina vaaditaan myös testattavat moduulit, oletettavat tapahtumat, testidata, testin vaiheet, oletettavat tulokset, saatu lopputulos sekä mahdollisesti vielä lisäinformaatiota testistä, esimerkiksi kommentteja.

Käydään seuraavaksi läpi miten laadukas testitilanne (test case) luodaan. Ensimmäiseksi pitää olla tiedossa käyttäjän vaatimukset, syy testata ohjelmistoa ja selvittää kaikki ohjelmiston ominaisuudet vähintään perustasolla. Kun testaustoimintoja suoritetaan, on oltava tiedossa miten skenaariot määritellään. Kun vaatimukset sovelluksen toiminnallisista osista on selvillä, on todennäköisempää saada aikaiseksi laadukkaampi tehokas testausskenaario. Toiminnallisen puolen lisäksi käyttöjärjestelmä, ohjelmiston turvallisuus ja laitteisto jolla ohjelmistoa ajetaan, ovat tärkeitä. Näitä kaikkia tarvitaan datan keräämisessä. (Software Testing Class, 2017)

3.7 Pilviratkaisut

Pilvipalvelulla tarkoitetaan kapasiteetti- tai ohjelmistopalveluita tarjoavaa internetin kautta toimivaa palvelua jolla käyttäjä pääsee hyödyntämään etänä toimivan palvelimen resursseja. Pilvellä (cloud) on kolme erilaista palvelumallia jotka tarjoavat ominaisuuksia käyttäjän etänä käytettäväksi.

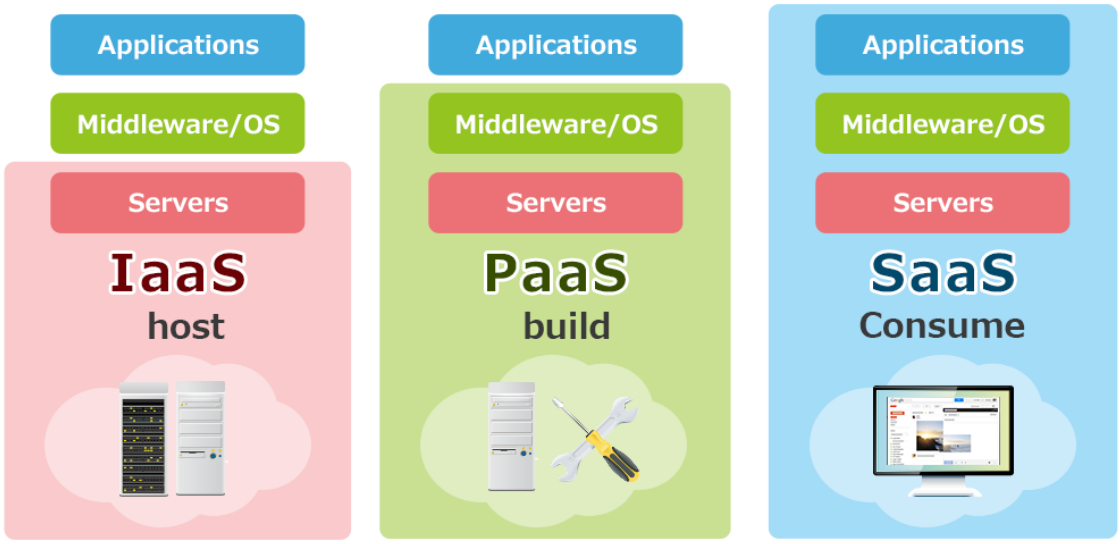
- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

IaaS tarjoaa fyysisiä resursseja kuten tallennustilaa, verkkolaitteita, laskentaservereitä jne. Se kuinka paljon käyttäjälle tästä fyysisestä raudasta tarjotaan, riippuu yleensä siitä kuinka paljon käyttäjä siitä on valmis maksamaan. Palvelu on yleensä jonkinlainen web-käyttöliittymä jonka kautta on mahdollista muuttaa palvelun käytössä olevia resursseja, asetuksia sekä verkkoyhteyksiä. Palomuuria on myös mahdollista muuttaa käyttäjän tarpeiden mukaisesti. Yleensä koko palvelin ja konfigurointi jää asiakkaan vastuulle, joten käyttö ei ole helpoimmasta päästä. (Sneha Nadig, 2017)

PaaS on käytännössä valmis alusta sovelluksia tai palveluja joita voidaan käyttää kehitykseen ja testaukseen. Lyhyesti se on siis palvelun muodossa oleva sovellusalusta. Tätä tarjotaan ohjelmistokehityksen tarpeisiin. Esimerkiksi virtuaalinen palvelin käyttöjärjestelmineen ja sovelluksineen on valmiina käyttöä varten. Kuten IaaS-palvelumallissa, niin myös PaaS-palvelumallissa tarjotaan yleensä web-käyttöliittymä, mutta tässä on lisäksi myös SSH-tuki sekä FTP-/SFTP-palvelu. Mahdollisesti myös komentorivityökalut eli ns.

CLI ja jonkinlainen rajapinta eli ns. API jolla on mahdollista käyttää PaaS-palvelumallia omissa sovelluksissaan. Tietoturva jää käyttäjän vastuulle.

SaaS on yksinkertaistettuna etäpalvelu joka on käyttäjälle saatavilla internetin välityksellä. Esimerkiksi verkkosivupohjaiset sähköpostipalvelut ovat SaaS-palvelumallin tyyppisiä. Käyttäjällä ei ole käytännössä ollenkaan vastuuta palvelusta vaan vastuu on palvelun tuottajalla. Yleensä nämä ovat selainpohjaisia. Tarjottavia palveluita voi olla useita jolloin palvelun tuottaja yleensä tukee ns. SSO-tapaa (Single sign-on) jolloin yhdellä kirjautumisella pääsee käsiksi kaikkiin palveluihin. Näin ollen ei käyttäjän tarvitse kirjautua useilla eri tunnuksilla palveluihin. (Heidi Eronen 2016)



KUVA 9. Kolme palvelumallia yksinkertaisesti esitettynä (Mohmmed Albihany, 2016)

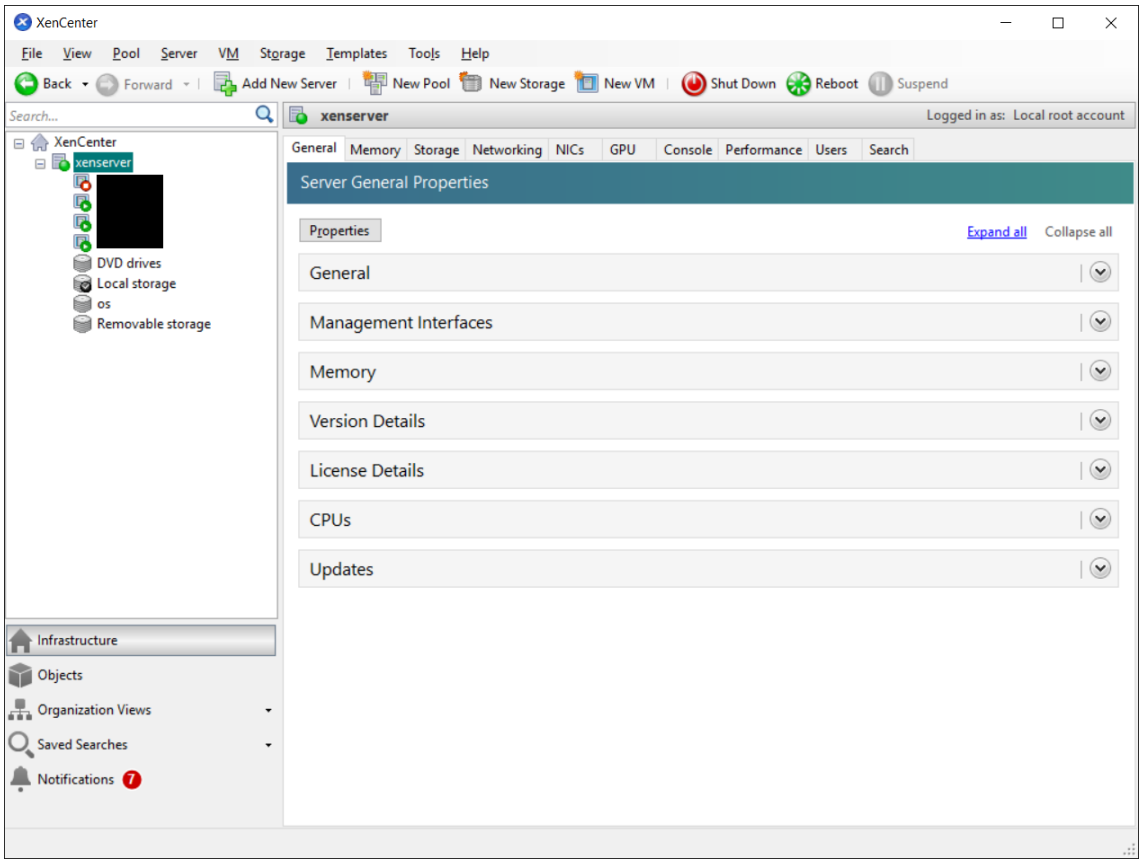
Pilvipalveluita on kolmea eri tyyppiä; julkisia, yksityisiä ja ns. hybridi-versioita. Julkisissa palvelut ovat saatavilla kaikille, yksityisissä palvelut ovat saatavilla vain esimerkiksi yrityksen intranetissä ja hybridi-versiossa palveluiden saatavuus on näistä edellämainituista välimalli jossa palvelusta riippuen voidaan saatavuus laittaa ulkoverkkoon tai vain yrityksen omille työntekijöille.

Pilvipalveluiden hyötyjä ovat mm. saatavuus silloin kun tarvii, matalat kustannukset, helposti muokattava ja skaalattavuus käyttötarpeen mukaan. Organisaatioissa on tarve investoida tarvittavaan laitteistoon ja ohjelmistoinfrastruktuuriin normaaleissa testaustapauksissa. Yleensä yrityksellä oleva testausympäristö ei vastaa kunnolla käyttäjän ympäristöä ja asiakasympäristö muuttuu niin nopeasti lyhyessä ajassa, että yrityksen on haastavaa pysyä tässä ajantasalla. Pilviratkaisuissa on helpompi kloonata käyttäjiä vastaava

testausympäristö ja myös päivittää sitä ajantasalle helpommin. Näin ollen kustannuksissakin säästetään, koska ympäristöä voi muokata tarpeen mukaan samaan hintaan. Resursseja ei jää käyttämättä, koska yritys voi maksaa vain siitä mitä se tarvitsee. Testiympäristöä on helppo muokata. Resurssien suuruutta voidaan skaalata käyttötarpeiden mukaan ja ajoittaa raskaimmat testit tilanteen mukaisesti. (Sneha Nadig, 2017)

4 VIRTUAALISEN TESTAUSYMPÄRISTÖN RAKENTAMINEN

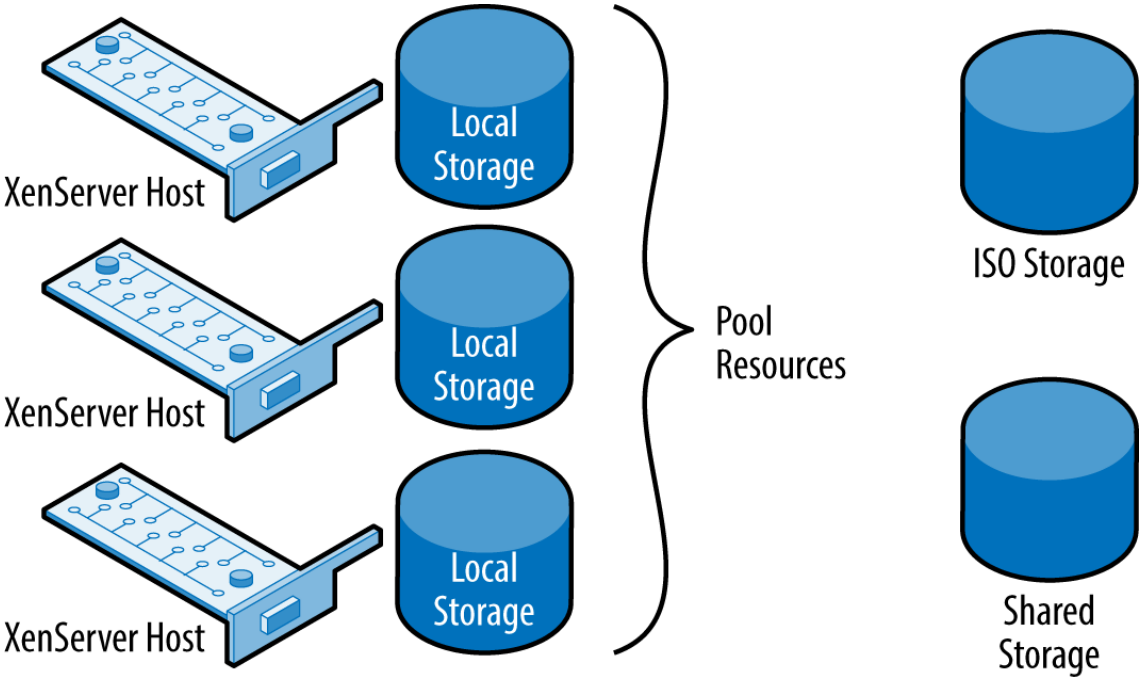
Testausympäristön rakentaminen aloitetaan yleensä valitsemalla virtualisointityökalu ja käyttöjärjestelmä. Lisäksi palvelin täytyy saada toimimaan yleensä fyysisen verkon kanssa, jotta sitä voidaan käyttää myös internetin välityksellä. Esimerkkinä tässä käytetään Citrix XenServer -virtualisointialustaa. Citrix Systems on virtualisointiratkaisuja tarjoava yritys. Xenserver on hypervisor. Se on avointa lähdekoodia ja kilpailee mm. Vmwaren sekä ESXi tuotteiden kanssa. Sen hallinnointi tapahtuu XenCenter-nimisellä ohjelmistolla. Se on graafinen Windows-pohjainen käyttöliittymä. XenCenter mahdollistaa XenServer palvelimien sekä virtuaalikoneiden hallinnan. Virtuaalikoneisiin asennettuihin käyttöjärjestelmiin suositellaan asentamaan XenServer Tools -ajurit.



KUVA 10. XenCenter graafinen käyttöliittymä (Antti Laine, 2017)

XenServer palvelimen asennus aloitetaan siitä, että luodaan asennusmedia joko usb- tai cd-medialle. Sen jälkeen palvelin käynnistetään sitä käyttäen, asetetaan näppäimistöasettelu ja hyväksytään EULA. Valitaan puhdas asennus, asetetaan root-käyttäjän eli yl-

läpitäjän salasana ja sitten asettaa pysyvä ip-osoite tai käyttää DHCP:tä. Valitaan host-name ja sitten DNS. Aikavyöhyke asetetaan ja seuraavaksi päätetään, että halutaanko käyttää NTP-asetusta. Lopuksi aloitetaan itse XenServerin asennusprosessi. Kun tämä on valmis, asennusmedia poistetaan ja serveri käynnistetään uudelleen. XenCenter asennetaan joko asennusmedialta tai erikseen lataamalla se valmistajan sivustolta. Kun ohjelma on avattu, lisätään serveri ”Add New Server”-kohdasta antamalla palvelimen ip-osoite, käyttäjä ”root” ja salasana joka asetettiin XenServer:n asennusvaiheessa. (Citrix Systems, 2017)



KUVA 11. XenServer toimintaperiaate (J.K. Benedict & Tim Mackey, 2017)

Ympäristö toteutetaan yhdellä fyysisellä palvelimella, luoden kaksi virtuaalista palvelinta. Virtuaalikoneita varten luodaan ns. ISO Storage eli yhteinen verkkojako, jossa asennusmedioita säilytetään ja josta niitä käytetään uusia virtuaalikoneita luodessa ja asentaessa. Virtuaalikonetta hallitaan Console-välilehdeltä käsin graafisesti. Asennusmedia liitetään iso-tiedostona virtuaalikoneelle, käyttäen XenCenter-hallintapaneelia. Virtuaalikone käynnistetään asennusmedialla ja käyttöjärjestelmän asennus suoritetaan samalla tavalla kuten fyysiselläkin tietokoneella. Kun käyttöjärjestelmä on asennettu, asennetaan XenServer Tools -ajurit. Tämän jälkeen on mahdollista ottaa käyttöön jokin etähallintaohjelmisto kuten SSH tai VNC ja jatkaa virtuaalikoneen hallintaa etänä. Lopuksi voidaan asentaa tarvittava ohjelmistotestausympäristö virtuaalikoneeseen, esimerkiksi kuten se

oli alun perin asennettuna työasemassa. Nyt testausympäristö on erillinen työasemasta ja siihen pääsee käsiksi muutkin etänä helposti.

5 POHDINTA

Opinnäytetyön tavoitteena oli tehdä teoriapainoitteinen tiivis dokumentti virtualisoinnista ohjelmistotestauksessa. Henkilökohtainen tavoite oli oppia virtualisoinnista sekä ohjelmistotestauksesta enemmän. Vaikka virtualisoinnista ja ohjelmistotestauksesta löytyy paljon tietoa erikseen, niin yhtä tiivistä dokumenttia ei tarpeeksi nykyaikaisena löytynyt. Alkuperäinen tavoite oli liian laaja joten kirjotusvaiheessa aikataulua joutui muokkaamaan realistisemmaksi. Opinnäytetyössäni halusin kirjoittaa virtualisoinnista aluksi perusasioita ja sitä miksi se on tärkeä osa ohjelmistokehitystä. Kirjoitin sen hyödyistä ja sen jälkeen lyhyesti sovellus- ja laitteistovirtualisoinnista. Ohjelmistotestauksesta kerroin yleistietoa ja sitten testauksen vaiheista. Testausta voidaan tehdä manuaalisesti ja automaattisesti niin näin ne tärkeinä myös. Kun testauksesta puhutaan niin moni on kuullut puhuttavan vesiputousmallista ja V-mallista joten näin niistä kertomisen myös tärkeänä. Ketterät menetelmät ja erilaiset testitilanteet ovat myös yleinen puheenaihe kun ohjelmistokehityksestä puhutaan. Halusin ottaa mukaan uuttakin tekniikkaa, joten valitsin pilviratkaisuista kirjoittamisen. Ne ovat nykyään ohjelmistotestauksen tärkeä osa. Lopuksi kerron yksinkertaistettuna virtuaalisen testausympäristön rakentamisesta yksittäisen työntekijän näkökulmasta. Tavoitteena on kertoa yksinkertaisia tapoja perustaa virtuaalinen testausympäristö kun fyysisenä työkaluna on vain oma työasema ja maksimissaan lisäksi jokin palvelin.

Tämän aikana tuli saavutettua oleellisimmat asiat virtualisoinnin ja ohjelmistotestauksen osalta. Aiheen laajuudesta huolimatta asiat saatiin tiivistettyä ja kuvien sekä esimerkkien käyttö auttaa asioiden hahmottamisessa ja ymmärtämisessä. Opin projektista paljon uutta niin aiheesta kuin opettamisesta. Tiedon hakeminen oli myös opettavaista näin laajassa mittakaavassa. Lähdemateriaali on elektronista. Syy siihen, että en käyttänyt kirjoja on tiedon nopea muuttuminen it-alalla.

LÄHTEET

Vangie Beal, Virtualization 2017. Luettu 8.10.2017
http://www.webopedia.com/TERM/V/virtualization.html

Thomas Burger, The Advantages of Using Virtualization Technology in the Enterprise 5.3.2017. Luettu 8.10.2017
https://software.intel.com/en-us/articles/the-advantages-of-using-virtualization-technology-in-the-enterprise

Margaret Rouse, app virtualization (application virtualization) 9.2017. Luettu 8.10.2017
http://searchvirtualdesktop.techtarget.com/definition/app-virtualization

Colan Infotech, WHAT ARE THE TYPES AND BENEFITS OF VIRTUALIZATION 1.6.2017. Luettu 8.10.2017
https://colaninfotech.com/blog/types-benefits-virtualization/

Virtuatopia, An Overview of Virtualization Techniques 29.5.2016. Luettu 11.10.2017
http://www.virtuatopia.com/index.php/An_Overview_of_Virtualization_Techniques#Hardware_Virtualization

Cem Kaner, Exploratory Testing 2006. Luettu 11.10.2017
http://www.kaner.com/pdfs/ETatQAI.pdf

Jiantao Pan, Software Testing - 18-849b Dependable Embedded Systems 1999. Luettu 11.10.2017
https://users.ece.cmu.edu/~koopman/des_s99/sw_testing/

Lakshay Sharma, Manual Testing 9.4.2016. Luettu 10.11.2017
http://toolsqa.com/software-testing/manual-testing/

Matti Vuori, Ketterä testaus ja testaus ketterässä ohjelmistokehityksessä 6.10.2010. Luettu 11.11.2017
https://www.mattivuori.net/julkaisuluettelo/liitteet/kettera_testaus.pdf

Software Testing Class, How to Write Good Test Cases? 2017. Luettu 12.11.2017
http://www.softwaretestingclass.com/how-to-write-good-test-cases/

Sneha Nadig, Getting Started with Cloud Testing 28.10.2017. Luettu 17.11.2017
http://www.softwaretestinghelp.com/getting-started-with-cloud-testing/

Heidi Eronen, IaaS, PaaS, SaaS? Mikä pilvipalvelu sopii yrityksellesi 15.3.2016. Luettu 17.11.2017
https://blog.planeetta.net/iaas-paas-saas

Citrix Systems, Inc. Citrix XenServer ® 7.2 Quick Start Guide 2017. Luettu 4.12.2017
https://docs.citrix.com/content/dam/docs/en-us/xenserver/current-release/downloads/xenserver-quick-start-guide.pdf