jamk.fi

Konesaliverkon konfiguroinnin automatisointi

Jani Arola

Opinnäytetyö
Marraskuu 2020
Tekniikan ja liikenteen ala
Insinööri (AMK), Tieto- ja viestintätekniikan koulutusohjelma
Tietoverkkotekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Arola Jani	Opinnäytetyö, AMK	Marraskuu 2020
	Sivumäärä 46	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Konesaliverkon automatisoinnin suunnittelu ja käyttöönotto

Tutkinto-ohjelma

Tieto- ja viestintätekniikan tutkinto-ohjelma, tietoverkkotekniikka (AMK)

Työn ohjaaja(t)

Sampo Kotikoski, Esa Salmikangas

Toimeksiantaja(t)

Kansaneläkelaitos (Kela)

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi Kansaneläkelaitoksen IT-palvelujen tulosyksikkö. Opinnäytetyön tavoitteena oli suunnitella tapa automatisoida konesaliverkkojen luominen ja sen käyttöönotto, sekä automatisoida verkkojen luominen mahdollisimman monessa verkkoalueen verkkotyypissä.

Opinnäytetyön teoriavaiheessa on kuvattu erityyppisiä konesaleja ja tiedonkulkua konesalien sisällä, automatisointiin käytettävää Ansible-ohjelmiston toimintaa, asentamista ja käyttämistä, sekä Python-ohjelmointikielen perusteita.

Opinnäytetyön suunnitteluvaiheessa todettiin Ansible-ohjelmisto parhaaksi vaihtoehdoksi sen valmiiden moduulien ja käytettävyyden vuoksi. Käyttäjäystävällisyyden säilyttämiseksi, suunnitteluvaiheessa todettiin myös, että automatiikkaa tulisi olla mahdollista ajaa pelkästään yhden pääskriptin tai komennon avulla, että siitä ei tulisi liian monimutkaista. Suunnitellessa havaittiin myös, että Ansible ei pystynyt automatisoimaan suoraan kaikkia osa-alueita, joten päätettiin käyttää niiden automatisointiin Python-ohjelmointikieltä.

Opinnäytetyön toteutusvaiheessa aloitettiin itse koodien kirjoittaminen ja testaaminen. Koodien luomisessa isona apuna toimi vuokaavion piirtäminen, jonka avulla voitiin havainnoida koodin kirjoittamista.

Lopputulokseksi saatiin luotua toimiva automatiikka, jota on mahdollista ajaa yhden pääskriptin avulla käyttäjäystävällisesti ja joka tekee kaikkiin haluttuihin paikkoihin konfiguraatiomuutokset.

Avainsanat (asiasanat)

Automatisointi, Ansible, Konesaliverkot, Python

Muut tiedot (Salassa pidettävät liitteet)

Opinnäytetyön liitteet ovat salassa pidettäviä niiden sisältävien koodien ja topologien pääsyn julkisuuteen välttämiseksi.

jamk.fi

Description

Author(s)	Type of publication	Date
Arola Jani	Bachelor's thesis	November 2020
		Language of publication: Finnish
	Number of pages: 46	Permission for web publication: x

Title of publication

Development and introduction of automatization of data center network

Degree programme

Information and Communications Technology

Supervisor(s)

Sampo Kotikoski, Esa Salmikangas

Assigned by

social insurance institution of Finland (Kela)

Abstract

The thesis was assigned by the IT-department of Social Insurance Institution of Finland. The objective of the thesis was to develop a way to automatize the creation of datacenter networks and introducing it and automatizing creation of networks in as many network areas and network types as possible.

The theory phase of the thesis was to introduce different types of datacenters and flow of information inside datacenters, operation, installation, and usage of Ansible that was used for automatization and fundamentals of Python programming language.

In the design phase of the thesis was thought what the best way is to implement automatization and Ansible was found out to be the best one because of its ready modules and usability. To maintain user-friendliness, it was determined that automatization should be able to be run using single main script or command to keep it from getting too complicated. While designing it was also noticed that Ansible was not capable of automating every step wanted, so it was determined to automate these steps Python was to be used.

The implementation phase started with writing of the automatization codes and testing them. While writing code flowcharts were found as a huge help thanks to their ability to help with perception of writing the code.

As the end result the thesis achieved a working way of automatizing creation of all of the different types of datacenter networks that were wanted to be automated that could be ran with a single main script file.

Keywords/tags (subjects)

Automatization, Ansible, Datacenter networks, Python

Miscellaneous (Confidential information)

Attachments of this thesis are to be kept secret due to them containing code and topologies that should not be released to the public.

Sisältö

Kuviot

Taulukot

Sanaluettelo

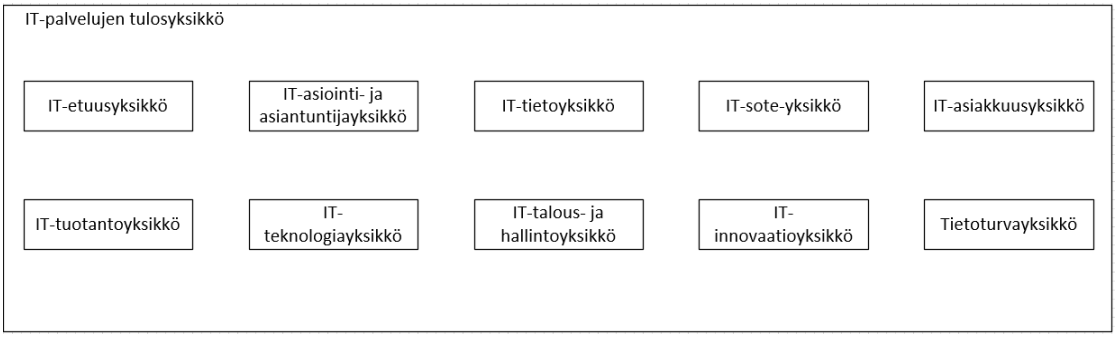
API	Application Programming Interface
APT	Ubuntu-käyttöjärjestelmän ohjelma pakettien hallintaan
CLI	Command line
DevOps	Developement and Operations
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronics Engineers
INI	Epävirallinen tyyppi konfiguraatiotiedostoille.
iSCSI	Internet Small Computer Systems Interface
ISP	Internet Service Provider
Kela	Kansaneläkelaitos
LAN	Local Area Network
NAS	Network Attached Storage
NAT	Network Address Translation
Open source	Avoin lähdekoodi
OSI	Open Systems Interconnection Reference Model
PAT	Port Address Translation
PIP	Python-ohjelma pakettien hallintaa varten
PyPI	Python Package Index
Rest-API	Representational State Transfer API
SAN	Storage Area Network
SDN	Software defined networking
SSH	Secure Shell
VLAN	Virtual Local Area Network
WAN	Wide Area Network
YML/YAML	Yleinen tiedostotyyppi konfiguraatiotiedostoille.
YUM	Centos-käyttöjärjestelmän ohjelma pakettien hallintaan

1 Lähtökohdat

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimii Kela. Kela on itsenäinen sosiaaliturvalaitos, joka toimii eduskunnan valvonnan alaisena. Kela on viiteen tulosyksikköön organisoitu linjaorganisaatio, nämä tulosyksiköt ovat: asiakkuuspalvelut, etuuspalvelut, IT-palvelut, tietopalvelut ja yhteiset palvelut. Kelan henkilöstön määrä oli vuoden 2018 lopussa yhteensä 7732 henkilöä. Suurin osa henkilöstöstä työskentelee etuuspalveluiden parissa 4271 henkilön voimin ja IT-palveluissa työskenteli 913 henkilön voimin. Palvelupisteitä Kelalla oli 160 kappaletta eri puolilla suomea, sekä Kela toimi 138 yhteispalvelua antavissa asiointipisteissä.

IT-Palvelujen tulosyksikkö vastaa Kelan IT:n kehittämis-, ylläpito- ja tuotantopalvelujen saumattomasta toiminnasta. IT-palvelut tuottavat lainsäädäntö- tai sopimuspohjaisesti palveluja myös muutamille ulkoisille toimijoille. Toiminnan punainen lanka on asiakas- ja palvelulähtöisyys. Palvelujen keskeisenä tavoitteena on tuottaa lisäarvoa asiakkaan liiketoiminnalle ja kansalaisille. Näin toteutetaan visiota olla suomalaisen arjen ja hyvinvoinnin mahdollistaja ja varmistaja. IT-palvelut toimivat ketterästi ja joustavasti asiakkaan priorisoinnin mukaisesti. (Henkilöstötilinpäätös 2019.)



Kuvio 1. Kelan IT-Palvelujen tulosyksikkö

1.2 Automatisoinnin tavoitteet ja tutkimuskysymys

Kelan IT-osastolla on ollut jo pitkään aikeena automatisoida työläs konesaliverkkojen luominen. Konfiguraatiot pitää luoda suurelle määrälle eri laitteita ja verkon mukaan eri tavalla. Yleisesti verkkosegmentointi (Ks. luvut 2.6 ja 2.7) pelkästään tuottaa tuplasti eri konfiguraation tarpeita, sekä verkon tyyppi vielä enemmän. Automatisoinnin implementointi verkkoon itsessään vaatii tuntemusta käytetyistä verkkoteknologioista ja manuaalisesta verkkojen luonnista, että automatiikka saadaan tekemään verkot oikealla tavalla.

Vaatimukset toimeksiantajalta oli alkuun suppeat. Toiveena oli automatisoida mahdollisimman monta eri komponenttia verkon luomisesta, säilyttämällä kuitenkin jonkinlainen kontrolli. Kuitenkin verkkojen automatiikan ollessa olematon, pienikin automatisointi olisi suureksi avuksi. Työn alussa kuitenkin todettiin, että jokaisen komponentin automatisointi olisi pienellä työllä mahdollista, joten otettiin tämä tavoitteeksi.

Opinnäytetyön toteutusvaiheen pääkohtina oli ohjelmien kirjoittaminen automatiikkaa varten ja niiden toiminnan testaaminen eristetyssä labraympäristössä käyttäen erillistä palvelinta. Labraympäristössä ohjelmien toiminnan varmistumisesta, niitä testattiin vielä erikseen testiympäristössä toiminnan varmistumiseksi. Toteutusvaiheen aikana myös ohjelmien valmistuttua niitä otettiin tarpeen mukaa käyttöön heti ja konesaliverkkoja tehtiin valmiiksi käytettäviksi.

Opinnäytetyössä pääsääntöiseksi tutkimuskysymykseksi muodostui seuraava:

- Miten Kelan verkkolaitteiden uuden verkon konfiguraatio saadaan luotua automaattisesti mahdollisimman käyttäjäystävällisesti?

1.3 Tutkimusmenetelmät

Opinnäytetyö on rakenteeltaan kehittämistyö, koska tavoitteena on kehitellä tapa tietoverkkojen automaattisen konfigurointien luomiseen konesalilaitteille niiden manuaalisen luomisen sijasta. Tutkimuksessa hyödynnetään koulusta opittuja taitoja, töiden ohessa opittuja tapoja ja käytänteitä, sekä teoriaosuudessa saatuja tietoja. Tutkimuksen tavoitteena on tuoda teknistä osaamista toimeksiantajan verkkojen automatisoinnin sarakkeelle ja vähentää työkuormaa toistuvien konfigurointitehtävien osalta.

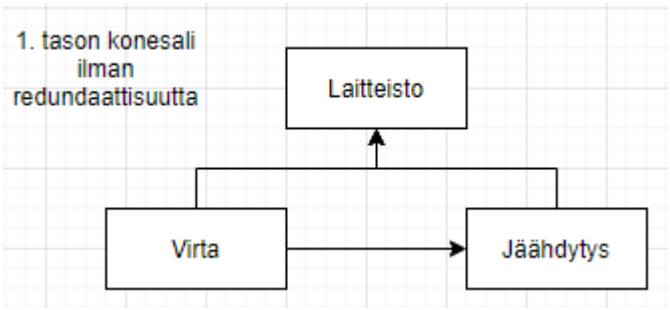
2 Konesalit

2.1 Yleisesti

Konesali on laitos, joka keskittää organisaation IT-toiminnot ja -laitteet, sekä missä se säilöö, hallitsee ja levittää tietojaan. Konesalit sisältävät verkon kriittisimmät järjestelmät ja ovat elintärkeitä päivittäisen toiminnan jatkuvuudelle. Seurauksena on se, että konesalit ja niiden tietojen turvallisuus ja luotettavuus ovat organisaation ensisijainen tavoite. Isommat konesalit ovat normaalisti suuria hallin tyyppisiä tiloja, mutta pienemmissä organisaatioissa voidaan puhua myös kaappidatakeskuksista. Kaappidatakeskukset ovat pieniä huoneita, joskus jopa yksittäinen räkkikaappi, jonne palvelimet ja kytkimet on sijoitettu ja josta toimintaa ajetaan. Tämä ei kuitenkaan välttämättä ole paras mahdollinen ratkaisu, koska kaappidatakeskus rajoittaa huomattavasti redundanttisuusmahdollisuuksia, ilmankiertoa ja laajennettavuutta. Pienissä organisaatioissa tämä kuitenkin toimii ratkaisuna sen pienemmän hinnan takia. (What is A Data Center?)

2.2 Tier 1

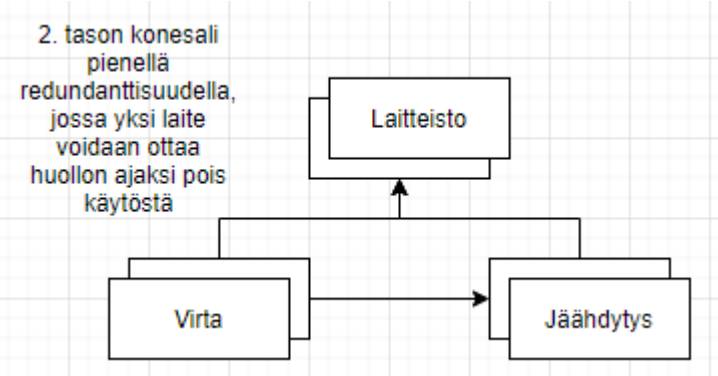
Konesalien kokoja ja käytettävyyttä voidaan verrata eri tasoilla. Ensimmäisen tason konesali on pienin mahdollinen, sekä se ei tarjoa redundanttisuutta palveluille. Konesali on käytettävissä 99.671%, ylläpitäen maksimissaan 28.8 tunnin alhaalla oloajan vuodessa. (Data Center Standards (Tires I-IV); Tier 2 Data Center.)



Kuvio 1. 1. Tason konesalin infrastruktuuri.

2.3 Tier 2

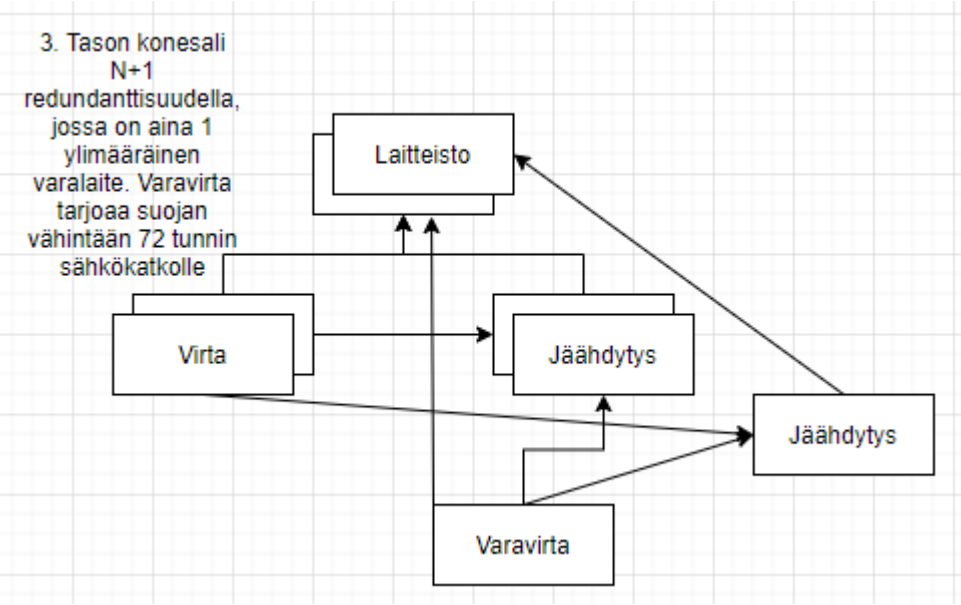
Toisen tason konesalissa on samat- tai parannellut komponentit ja osat 1. tason konesaliin verrattuna, mutta redundanttisuudella tai infrastruktuurisilla komponenteilla. Konesali on siis toiseksi pienin ylhäällä oloajan mukaan. Konesalissa tehokomponentti tai laite voidaan korvata tai poistaa keskeyttämättä tietokoneen ydinkomponenttien virransyöttöä. Konesali takaa 99.741% käytettävyyden, maksimissaan 22 tunnin alhaalla oloajan vuodessa ja osittaisen redundanttisuuden virransyöttöön, sekä jäähdytykseen. (Tier 2 Data Center.)



Kuvio 2. 2. Tason konesalin infrastruktuuri.

2.4 Tier 3

Kolmannen tason konesali on 2. suurin ylhäällä oloajan mukaan. Konesali on sijainti, jossa on redundantit ja kaksitehoiset palvelimet, tallennustila, verkkolinkit ja muut IT-komponentit. Kolmannen tason konesali on yleisimmin käytetty konesalin taso, jossa komponentit saavat virran useista aktiivisista ja riippumattomista virtalähteistä ja jäähdytysresursseista. Konesalin ylhäällä oloaika on 99.982%, maksimissaan 1.6h alhaalla oloajalla vuodessa ja N+1 viansiedon tarjoten vähintään 72-tunnin sähkökatkon suojauksen. (Tier 3 Data Center 2014.)

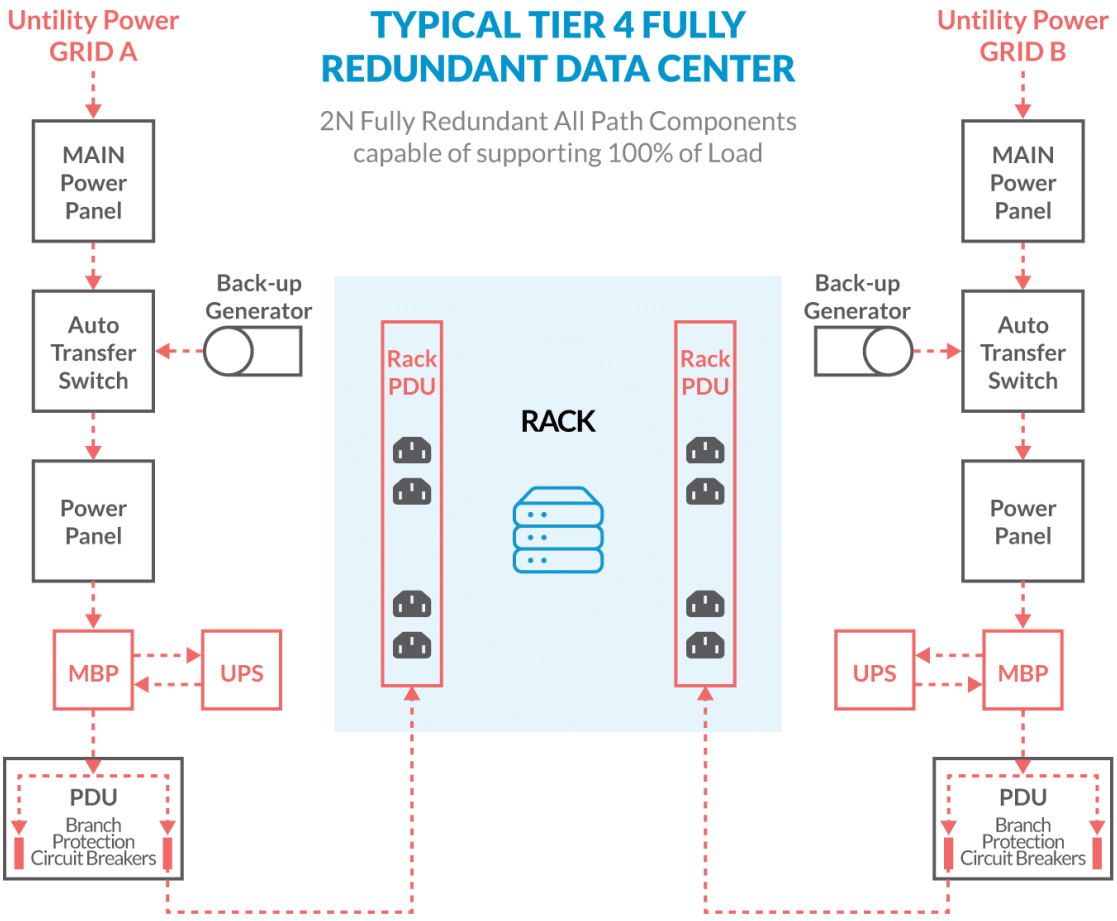


Kuvio 3. 3. Tason konesalin infrastruktuuri.

2.5 Tier 4

Neljännen tason konesali on yritysluokan konesali, jossa on redundantit ja kaksitehoiset instanssit palvelimia, tallennustilaa, verkkolinkkejä ja virran jäähdytyslaitteita. Neljännen tason konesali on edistyneimmän tason konesali, jossa redundanssia sovelletaan koko konesalin tietojenkäsittely- ja muuhun infrastruktuuriin. Konesalin yl-

häällä oloaika on 99.995% vuodessa, 2N+1 täysin redundantti infrastruktuuri 96-tunnin sähkökatkon suojauksella ja maksimissaan 26.3 minuutin alhaalla oloajalla vuodessa. (Data Center Standards (Tiers I-IV).)



Kuvio 4. 4. Tason konesalin infrastruktuuri

2.6 Intranet

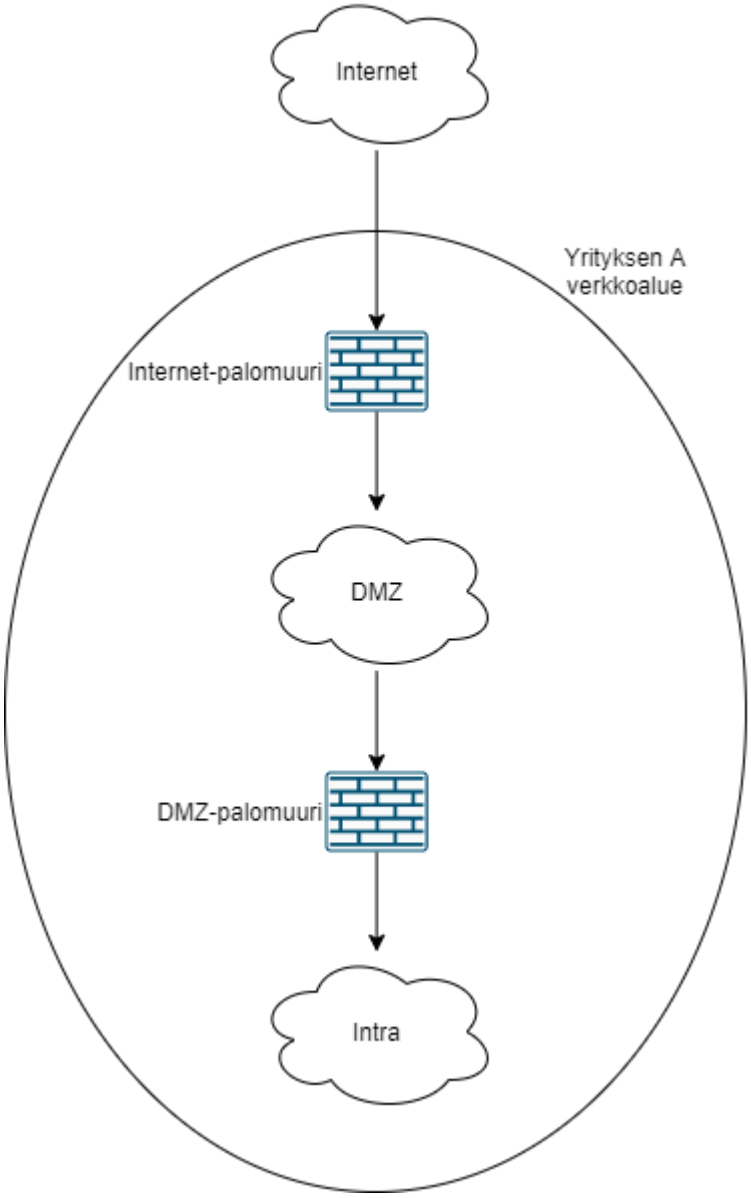
Intranetillä tarkoitetaan tietoliikenne ja tietoturvamielessä sisäverkossa sijaitsevaa verkkosegmenttiä. Intranet on yrityksen sisäinen yksityinen verkko, jota käytetään yrityksen tietojen ja resurssien turvalliseen jakamiseen. Intranet on käytännössä eristetty internetistä, sijaitsee internetin ja DMZ-alueen alla ja sinne ei tule ottaa suoraa yhteyttä internetistä. Intranet helpottaa työntekoa, koska yhteydet on sallittu pelkästään sisäverkon käyttäjiltä. Intranetin ja internetin erona, internet toimii julkisessa

verkossa, kun intranet toimii yksityisessä verkossa ja internetin tiedot ovat saatavilla jokaisella, kun intranetin tiedot on rajattu pelkästään valtuutetuille. (Rouse.)

2.7 DMZ-verkkoalue

DMZ-alue toimii tietoliikenne ja tietoturvamielessä sisäverkossa intranetin ja internetin välialueena. DMZ-aluetta voidaan käyttää tietoturvallisesti sovellusten asentamiseen, joille on tarve päästä sisäverkosta ja ulkoverkosta. DMZ-aluetta voidaan käyttää esimerkiksi sähköpostipalvelimille, nimipalvelimille ja web-applikaatioille, kuten välityspalvelimille, kuormanjakajille, web-palomuureille ja pääsynhallinnalle. DMZ-

alue on yleensä eroteltu intranetistä esimerkiksi palomuurin avulla tietoturvan takaamiseksi.



Kuvio 5. Verkkoalueet.

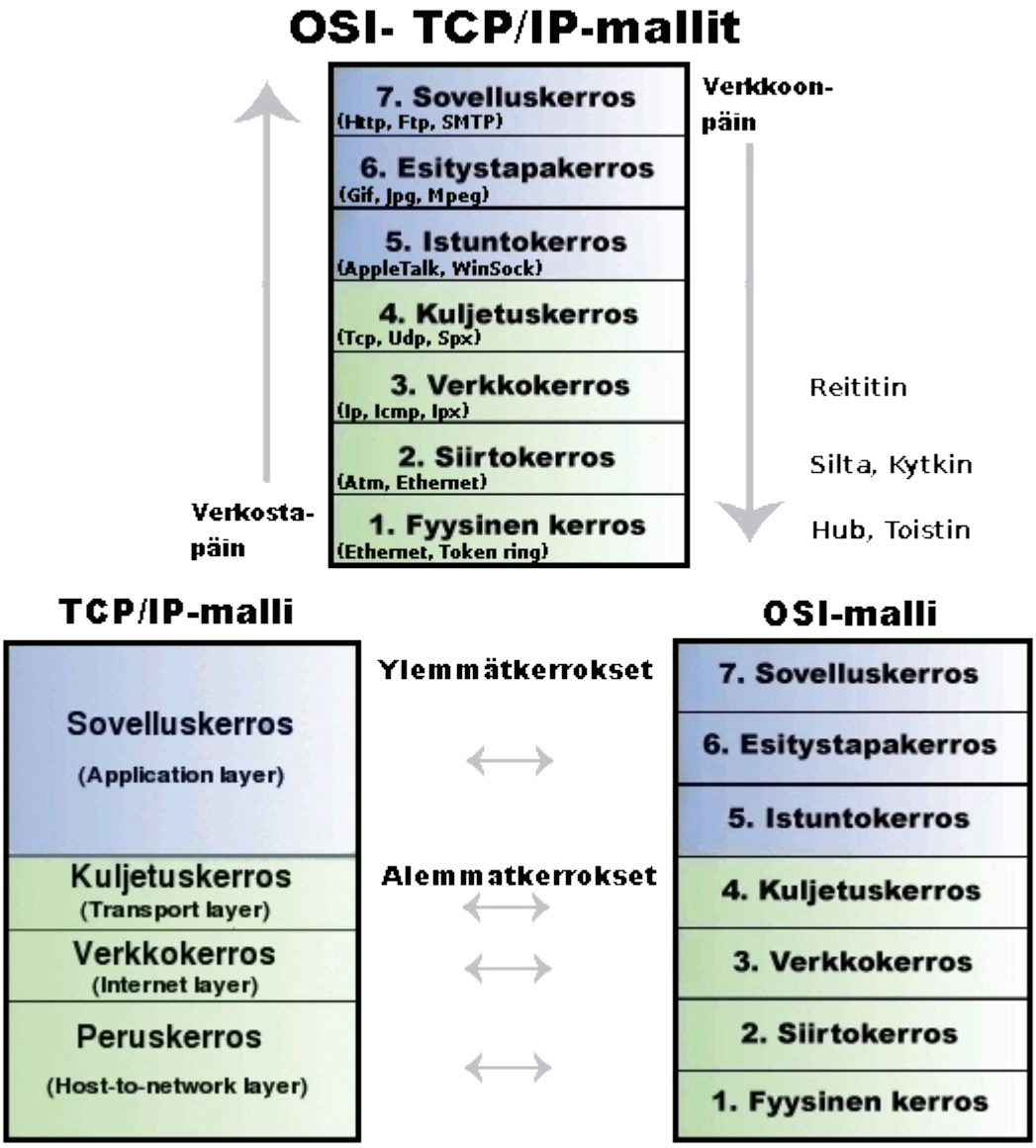
Kuviossa 5. on kuvattu esimerkkinä yrityksen a verkkoalue, jossa ensimmäisenä internetistä tulevat yhteydet menevät internet-alueen palomuurille, joka sijaitsee internetin ja DMZ-alueen välissä. Internet-alueen palomuuri sallii tai estää yhteydet tarvittaessa sisäverkon suuntaan. Internet-alueen palomuurin jälkeen sijaitsee yrityksen DMZ-alue, johon voidaan julkaista WEB-applikaatioita, joille käyttäjän on tarve päästä ulkoverkosta. DMZ-alueen jälkeen yhteydet jatkuvat tarvittaessa DMZ-alueen

palomuurille, joka sijaitsee DMZ- ja Intra-alueen välissä. DMZ-alueen palomuurin jälkeen yhteydet jatkavat alimman tason verkkosegmenttiin intra-alueelle. (Di Palo 2018.)

2.8 OSI-malli

OSI-malli (Open Systems Interconnection Refrence Model) kuvaa tiedonsiirtoprotokollien kerroksia. OSI-mallissa on seitsemän eri kerrosta, joista jokainen käyttää yhtä alemman kerroksen palveluja ja tarjoaa palveluja yhtä kerrosta ylemmäs. OSI-mallin kerrokset ylimmästä alimpaan ovat: Sovelluskerros, jossa sijaitsee käyttäjälle näkyvät sovellukset. Esitystapakerros, joka muuttaa tiedon käyttäjälle sopivaksi, kuten kuvan pikseleiksi ja unicode-tekstin eri maan merkeiksi. Istuntokerros, joka huolehtii useiden yhdessä yhteydessä kulkevista istunnoista. Kuljetuskerros, joka huolehtii, että paketit tulevat perille ja järjestetään oikein. Verkkokerros, joka hoitaa maailmanlaajuisen reitityksen ja kohdekoneen löytämisen. Siirtokerros, joka hoitaa paikallisen lä-

hiverkon laitteiden välisen liikenteen. Fyysinen kerros, joka käsittelee sähköimpulsseja tai muuta fyysistä tekniikkaa. (Mohammed, 2014)



Kuvio 6. OSI-mallin kerrokset.

2.8.1 TCP/IP-malli

OSI-malli on looginen malli, joka on suunniteltu kuvaamaan viestintäjärjestelmän toiminnot jakamalla viestintämenettely pienempiin ja yksinkertaisempiin komponentteihin. Kun puhutaan TCP/IP-mallista, sen on suunnitellut ja kehittänyt puolustusministeriö 1960-luvulla ja se perustuu standardiprotokolliin. TCP/IP on lyhenne sanoista

Transmission Control Protocol/Internet Protocol. TCP/IP-malli on tiivis versio OSI-mallista, sisältäen neljä kerrosta kaikkien seitsemän kerroksen sijaan. Kerrokset ovat Sovelluskerros, Kuljetuskerros, Verkkokerros ja Peruskerros (Ks. Kuvio 6.). (Mohammed, 2014)

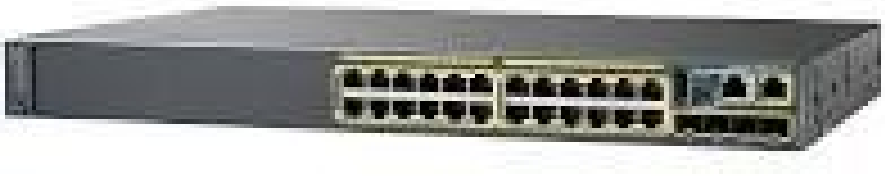
3 Kohdelaitteisto

3.1 Cisco Systems

Cisco on maailmanlaajuinen johtaja tietoverkoissa. Yrityksen perusti kaksi Stanfordin yliopiston tietotekniikkoja vuonna 1984, jotka etsivät helpompaa tapaa yhdistää erityyppisiä tietokonejärjestelmiä. Cisco lähetti ensimmäisen tuotteensa vuonna 1986 ja on nyt monikansallinen yritys, jossa työskentelee yli 35000 työntekijää yli 115 maassa. Nykyään Cisco toimii perustuksena palveluntarjoajien, pienestä keskikoisten yritysten ja yritysasiakkaiden verkoille. Noin 85% maailman internet liikenteestä kulkee Ciscon järjestelmien läpi. (Who is Cisco.)

3.2 Kytkimet

Kytkin on pieni laite, joka suodattaa ja välittää paketteja LAN-segmenttien välillä. Eri kytkinmallit tukevat useita kytkettyjä laitteita. Kuluttajaluokan kytkin tarjoaa joko neljä tai kahdeksan yhteyttä Ethernet-laitteille, kun taas yritysverkon kytkin tukee yleensä 24–128 yhteyttä. Lisäksi kytkimiä voidaan yhdistää toisiinsa, lisäten mahdollisuuden suurempaan kapasiteettimäärään Ethernet-laitteissa. Kytkimiä on tyypillisesti kahdenlaisia, hallittuja ja hallitsemattomia. Hallitsematon kytkin toimii suoraan laatikosta, muttei voida konfiguroida. Hallitsemattomia kytkimiä käytetään tyypillisesti kotiverkkolaitteina. Hallittua kytkintä voidaan konfiguroida. Hallittua kytkintä voidaan tarkkailla ja säätää paikallisesti tai etäyhteydellä, antaen paremman hallinnan verkkoliikenteestä ja -pääsystä. (How Does a Network Switch Work 2018.)



Kuvio 7. Cisco Catalyst 2960-x sarjan 24-porttinen kytkin

On olemassa 2. kerroksen ja 3. kerroksen kytkimiä. 2. kerroksen ja 3. kerroksen termit on otettu OSI-mallin viestintäkerroksista datalinkkikerroksesta ja vastaavasti verkkokerroksesta. Toinen kerros tarjoaa suoran tiedonsiirron kahden laitteen välillä lähiverkon sisällä. Tällainen kytkin toimii pitämällä taulukkoa MAC-osoitteista. Kytkimen MAC-osoitetaulukko tallentaa oppimansa laitteiston MAC-osoitteet ja niihin liittyvän fyysisen portin, jota on viimeksi käytetty. Tietokehykset vaihtavat vain lähiverkon MAC-osoitteet, eikä niitä tunneta sen ulkopuolella. 2. kerroksen kytkin voi määrittää VLAN-verkot tietyille kytkinporteille, jotka puolestaan ovat eri 3. kerroksen aliverkoissa, joten viestintä muiden lähiverkkojen tai VLANien kanssa tarvitsee 3. kerroksen toimiakseen.

Kolmas kerros käsittelee paketin reitityksen loogisella osoittamisella ja aliverkon ohjauksella. Kolmas kerros toimii reitittämällä paketteja niiden IP-osoitteeseen. Kolmannessa kerroksessa tarkistetaan jokaisen paketin lähde- ja kohde-IP reititystaulussa ja määritellään seuraava paras mahdollinen hyppy paketille. Jos reititystaulusta ei löydy kohde-IP:tä, eikä oletusreittiä olla konfiguroitu, paketti pudotetaan. Täten kolmannen kerroksen reititysprosessi aiheuttaa usein jonkin verran viivettä. (Layer 2 vs Layer 3 Switch: Which One Do You Need? 2017.)

Kuvio 8. Cisco Nexus 9300 sarjan 48porttinen SFP-kytkin.

3.2.1 Virtual lan (VLAN)

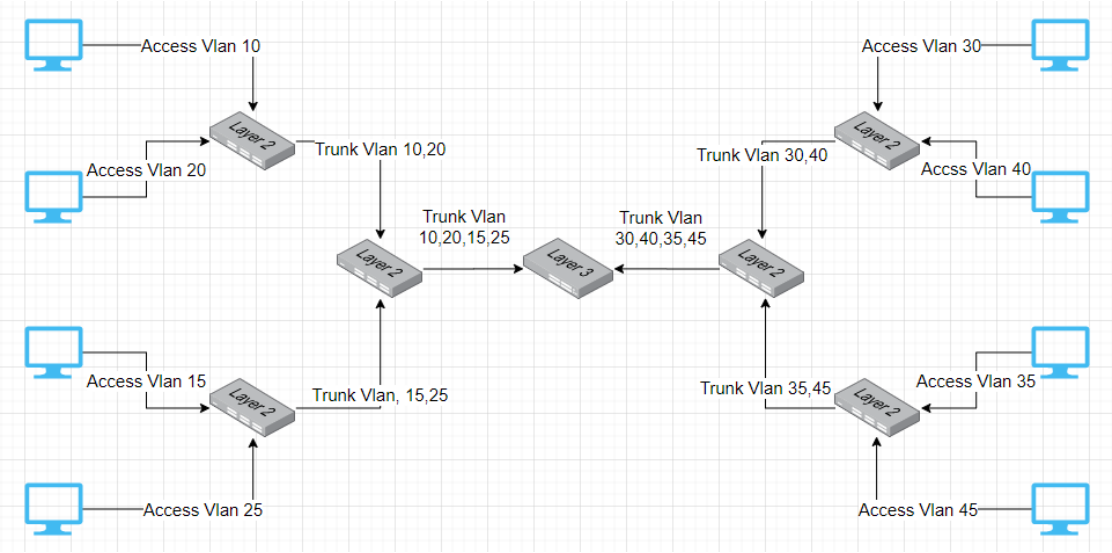
VLAN on loogisesti erillinen IP-aliverkko, joka sallii useiden IP-verkkojen ja aliverkkojen olemassaolon samassa kytketyssä verkossa. VLAN on looginen lähetysalue, joka voi kattaa useita fyysisiä LAN-segmenttejä. Se on tapa, jolla voidaan määrittää kytkimet virtuaalisiin lähiverkkoihin parantamaan verkon suorituskykyä jakamalla suuret 2. kerroksen lähetysalueet pienemmiksi. VLANeja käyttämällä, voidaan ryhmitellä asemat loogisten toimintojen tai sovellusten perusteella käyttäjän fyysisestä sijainnista riippumatta. Jokainen VLAN toimii erillisenä lähiverkkona ja kattaa yhden tai useamman kytkimen. Tämän avulla isäntälaitteet käyttäytyvät, kuin ne olisivat samassa verkkosegmentissä. (VLAN.)

3.2.2 Access ja trunk

VLAN-yhteyslinkkejä on kahden tyyppisiä, pääsy- (access) ja runkoyhteyksiä (trunk). Pääsy-yhteys on linkki, joka on osa vain yhtä VLAN-verkkoa ja on normaalisti käytössä päätelaitteille. Jokainen pääsy-yhteydessä kiinni oleva päätelaite, ei ole tietoinen VLAN-jäsenyydestä. Pääsy-yhteys voi ymmärtää vain vakio Ethernet-kehyksiä. Kytkimet poistavat kaikki VLAN-tiedot kehyksestä ennen niiden lähettämistä päätelaitteelle.

Runkoyhteys voi kuljettaa useita VLAN-liikenteitä ja on yleensä käytössä kytkimien kytkemisessä toisiin kytkimiin tai reitittimiin. Tunnistaakseen VLANin, johon kehys kuuluu, kytkimet tukevat erilaisia tunnistustekniikoita, joista yleisin on IEEE:n 802.1Q

standardin mukainen. Runkoyhteyttä ei ole määritetty tietylle VLANille. Monet VLAN-liikenteet voidaan kuljettaa kytkinten välillä yhdellä fyysisellä runkoyhteydellä. (Understanding VLAN Trunking 2019.)



Kuvio 9. Looginen kuva verkkokytkennöistä.

3.3 Reitittimet

Reititin on laite, joka välittää tietoa kahden tai useamman pakettivälitteisen tietokoneverkon välillä. Reititin tarkastaa tietyn datapaketin kohdeosoitteen, laskee parhaan tavan saavuttaa määränpää ja välittää sen vastaavasti. Reititin on yleinen yhdyskäytävätyyppi. Se on sijoitettu kohtaan, jossa kaksi tai useampia verkkoja kohtaa jokaisessa Internetin läsnäolopisteessä. Sadat reitittimet voivat välittää yhden paketin, kun se siirtyy verkosta toiseen matkalla lopulliseen määränpäähänsä. Reitittimet toimivat OSI-mallissa 3. kerroksessa. (Rouse 2019.)

Kuvio 10. ASR 1002-hx sarjan reititin.

3.3.1 Reitittimen toiminta

Reititin tutkii paketin otsikon kohde-IP osoitteen ja vertaa sitä reititystaulukkoon paketin parhaan seuraavan hypyn määrittämiseksi. Reititystaulukoissa luetellaan ohjeet datan siirtämiseksi tietyille verkkokohteille, joskus muiden muuttujien, kuten kustannusten yhteydessä. Ne muodostavat algoritmisen sääntöjoukon, joka laskee parhaan tavan siirtää liikennettä kohti mitä tahansa tiettyä IP-osoitetta. Jos kutsuttu ja kutsuva laite ovat kuitenkin samassa aliverkossa, voidaan tiedonsiirto suorittaa ilman reitityksen tarvetta kytkemällä (Ks. Osio 3.2.).

Reititystaulukko määrittelee usein oletusreitin, jota reititin käyttää aina, kun se ei löydä parempaa edelleenlähetysvaihtoehtoa paketille. Esimerkiksi tyypillinen kodin reititin ohjaa kaiken lähtevän liikenteen yhdellä oletusreitillä Internet-palveluntarjoajalle. Reititystaulukot voivat olla staattisia eli manuaalisesti konfiguroituja tai dynaamisia. Dynaamiset reitittimet päivittävät reititystaulukkonsa automaattisesti verkon toiminnan perusteella, jakamalla tietoja muiden laitteiden kanssa reititysprotokollien kautta. Monet reitittimet suorittavat myös verkko-osoitteiden muuntamisen, suojaamalla lähiverkon yksityiset IP-osoitteet osoittamalla kaiken lähtevän liikenteen yhdellä jaetulla julkisella IP-osoitteella. Tämä auttaa sekä säilyttämään maailmanlaajuisesti kelvollisia IP-osoitteita, että parantamaan verkon suojausta. (Rouse 2019.)

4 Ansible-ohjelmisto

4.1 Ansiblen käyttö

Ansible on avoimen lähdekoodin IT-automaatio, joka voi parantaa dramaattisesti IT-ympäristön skaalautuvuutta, johdonmukaisuutta ja luotettavuutta. Ansiblen avulla voidaan automatisoida kolmen tyyppisiä tehtäviä. Ansiblen avulla voidaan automatisoida varustamista, asentamalla infrastruktuurin tarvitsemat palvelimet, kokoonpanon hallintaa muuttamalla sovelluksen tai käyttöjärjestelmän konfiguraatiota, käynnistää ja pysäyttää palveluita, toteuttaa turvallisuuspolitiikkaa tai suorittaa monenlaisia muita määritystehtäviä. Ansiblella voidaan myös tehdä DevOpsista helpompaa, automatisoimalla sisäisesti kehitettyjen sovellusten käyttöönotto tuotantojärjestelmiin.

Ansible voi automatisoida IT-ympäristöjä riippumatta siitä, sijaitsevatko palvelimet fyysisen raudan päällä, virtuaalisessa ympäristössä vai pilvessä. Ansiblen avulla voidaan myös automatisoida laaja valikoima järjestelmiä ja laitteita, kuten tietokantoja, tallennuslaitteita, verkkoja ja palomuureja. Tästä vielä helpompaa tekee se, että Ansiblen kanssa ei välttämättä tarvitse tietää komentoja tavoitteiden saavuttamiseen, vaan riittää kun tietää mihin tilaan järjestelmän haluaa. (Hummel 2019.)

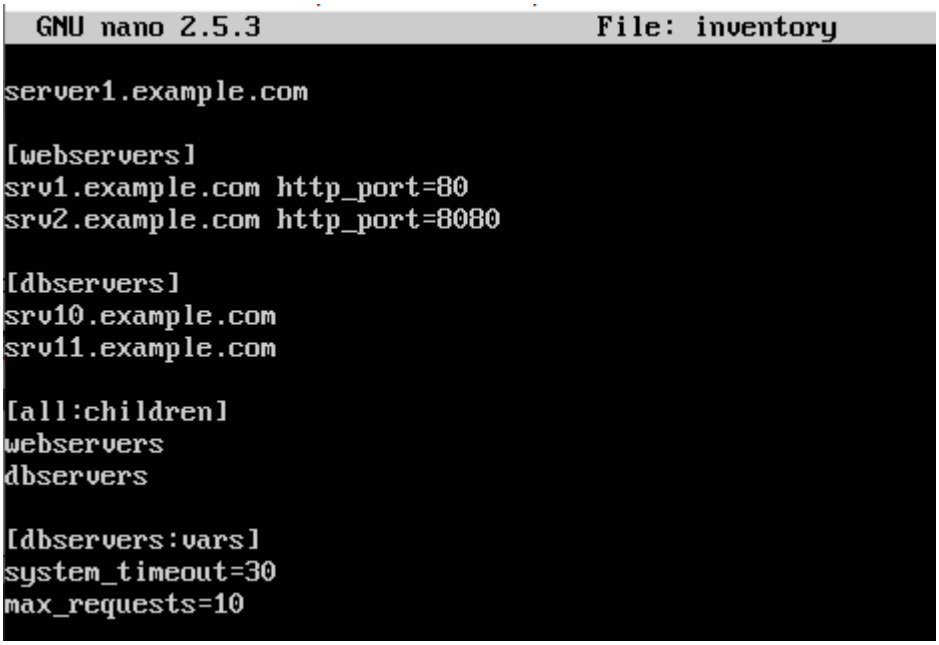
4.2 Ansiblen toiminta

Ansible toimii yhdistämällä noodeihin ja puskemalla niihin pieniä ohjelmia, joita kutsutaan moduuleiksi. Nämä ohjelmat on kirjoitettu resurssimalleiksi järjestelmän halutusta tilasta. Tämän jälkeen Ansible suorittaa nämä moduulit (oletuksena SSH:n kautta) ja poistaa ne valmistumisen jälkeen.

Ansible itse ohjelmana on kirjoitettu python-kielellä ja vaatii pythonin asennuksen toimiakseen. Moduuleja on kuitenkin mahdollista kirjoittaa millä vain halutulla kielellä, jotka sitten ajetaan kohteissa. (How Ansible Works.)

4.2.1 Ansiblen muuttujat

Ansiblessa muuttujat ovat samankaltaisia, kuin missä vain ohjelmointikielessä. Muuttujat helpottavat ohjelmien ajamista antamalla muuttujalle arvo ja käyttämällä sitä tarvitussa paikassa. Ansiblessa on mahdollista asettaa muuttujia moneen eri paikkaan ja niitä käydään etsimässä järjestyksessä eri tasoilta eri tärkeydellä (ks. Taulukko 1). Tämän avulla, jos saman niminen muuttuja on asetettu moneen eri paikkaan, tiedetään mitä muuttujaa Ansible käyttää. (Using Variables 2020.)



Kuvio 11. Esimerkki inventory-muuttujista hostin ja ryhmän tasolla

Taulukko 1. Muuttujien lokaatiot tärkeimmästä vähäisimpään.

Muuttujan paikka
Extra muuttujat
Include parametrit
Rooli parametrit
Set_facts ja reksisteröidyt muuttujat

Include_vars
Taskin muuttujat
Blokin muuttujat
Roolin muutujat (role/vars/main.yml)
Taskin vars_files suoritus
Taskin vars_prompt suoritus
Taskin vars suoritus
Hostin faktat / välimuistin set_facts
Pelikirjan host_vars
Inventory host_vars
Inventory tiedosto tai skripti host_vars
Pelikirjan group_vars/ryhmä
Inventoryn group_vars/ryhmä
Pelikirjan group_vars/kaikki
Inventoryn group_vars/kaikki
Inventory-tiedoston tai skriptin group_vars
Roolin defaultit
CLI:n muuttujat esim. ”-u käyttäjä”

4.2.2 Ansiblen inventory

Ansiblen inventory-tiedostossa määritellään hostit ja host-ryhmiä, joille komennot, moduulit ja pelikirjan taskit ajetaan. Tiedosto voi olla monta eri formaattia, ympäristöstä ja plugineista riippuen. Yleisimmät formaatit ovat kuitenkin ini ja yml. Tiedoston oletuspaikka on /etc/ansible/hosts. Tarvittaessa on mahdollista luoda myös projektikohtaisia luettelotiedostoja vaihtoehtoisissa paikoissa.

Inventory-tiedostoon voidaan luetella yksittäisiä hosteja tai host-ryhmiä. Tämän avulla voidaan määrittää muuttujia yksittäisille hosteille tai hostien ryhmille, esimerkiksi linux-ryhmälle muuttujaksi linux-käyttöjärjestelmä. (Understanding the Ansible Inventory File When Managing Devices Running Junos OS 2018.)

```
 GNU nano 2.5.3                          File: inventory

server1.example.com

[webservers]
srv1.example.com
srv2.example.com

[dbservers]
srv10.example.com
srv11.example.com

[all:children]
webservers
dbservers_
```

Kuvio 12. Esimerkki Ansiblen inventoryn syntaxista

4.2.3 Pelikirjat

Ansiblen pelit ovat joustavia moduulien takia, jotka koskevat kohteen hallittujen palvelimien erilaisia näkökohtia. Moduuleja on olemassa monille järjestelmän kokoonpanon osille, mukaan lukien ohjelmistojen asennus ja käyttäjän hallinta. Ansible tarjoaa monia moduuleja, kuten myös avoimen lähdekoodin yhteisö.

Pelikirjat koostuvat siis peleistä, jotka koostuvat moduuleista. Pelikirja suoritetaan, kun ylläpitäjä ajaa *ansible-playbook*-komennon kohdekoneille. Pelikirja käyttää inventory-tiedostoa määrittääkseen koneet, jotka kuuluvat pelikirjan hallintaan. (Rouse 2017.)

```
 GNU nano 2.5.3                      File: helloworld.yml

---
- name: This is a hello-world example
  hosts: localhost
  tasks:
    - name: Create a file called '/tmp/testfile.txt' with the content 'hello world'.
      copy:
        content: hello world
        dest: /tmp/testfile.txt
```

Kuvio 13. Esimerkki hello-world pelikirjasta.

4.2.4 Moduulit

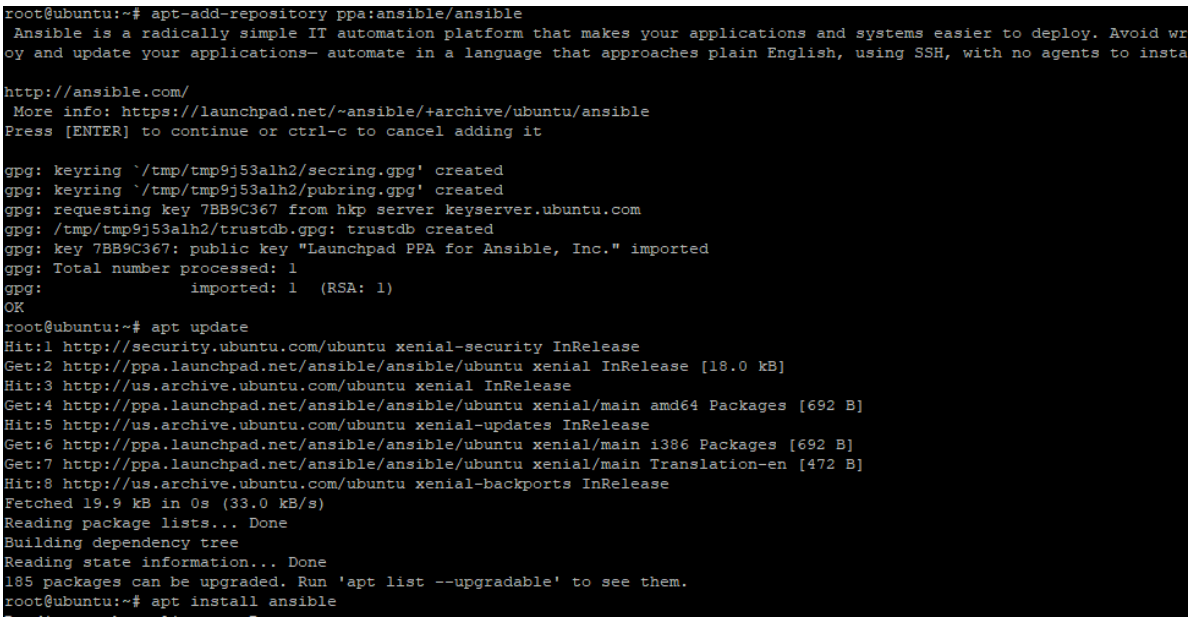
Kuten aiemmin on mainittu, Ansible toimii yhdistämällä noodeihin ja lähettämällä niihin pieniä ohjelmia, joita kutsutaan moduuleiksi etäsuoritusta varten. Tämä eroaa agenttipohjaisen konfiguraatiohallinnan yleisestä hakumallista, jossa konfiguraatio haetaan järjestelmästä.

Moduulit on kartoitettu resursseihin ja niiden vastaaviin tiloihin, jotka on esitetty YAML-tiedostoissa. Moduulien avulla voidaan käytännössä hallita kaikkea sellaista, jossa on API-, CLI- tai konfiguraatiotiedosto, jonka kanssa voidaan kommunikoida. Tavanomaisia näihin kuuluvia laitteita ovat esimerkiksi verkkolaitteet kuten kuormanjakajat, kytkimet, palomuurit, konttien orkestraattorit, itse kontteja ja virtuaalikoneiden instansseja.

Moduulien rakennusmalli on yksinkertainen: JSON stdoutissa. YAML-tiedostoissa ilmoitetut kokoonpanot viedään verkon kautta esimerkiksi SSH:n tai muun yhteyslaajennuksen avulla pieninä komentosarjoina, jotka suoritetaan kohdepalvelimissa. Moduulit voidaan kirjoittaa millä tahansa kielellä, jolla JSON voidaan palauttaa. Suurin osa Ansiblen moduuleista on kuitenkin kirjoitettu Pythonissa käyttäen Ansiblen APIa. (Da Silva 2019.)

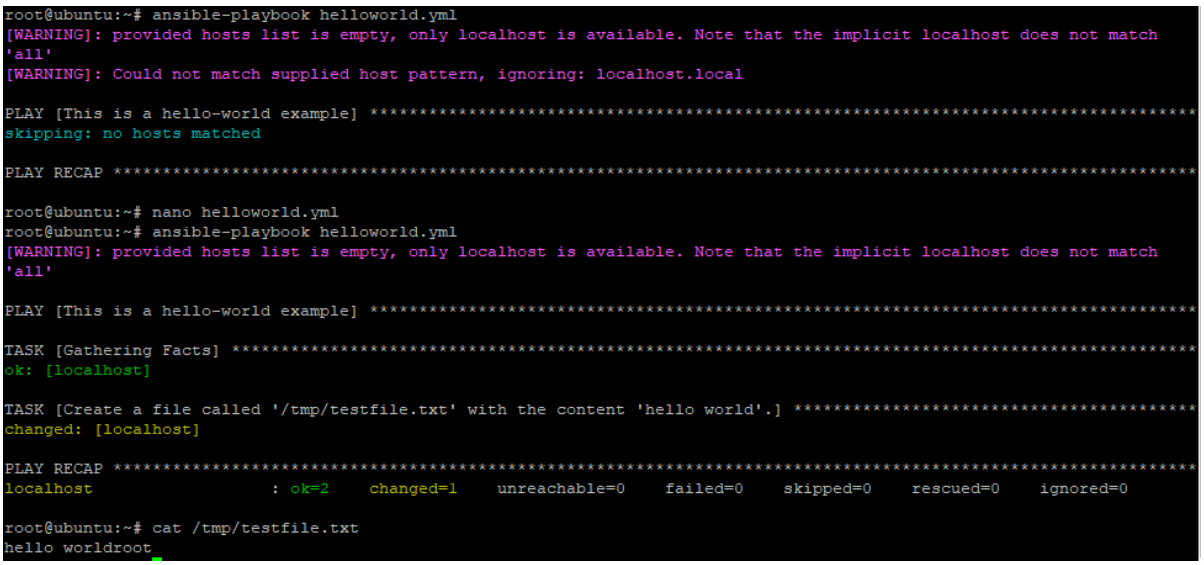
4.3 Asennus

Asennuksen kohdejärjestelmänä käytetään Linux-pohjaista Ubuntu 16.04 palvelinta. Ansible tarjoaa virallisen asennuspakettinsa Ubuntu-pohjaisille järjestelmille pakettivarastosta. Asennuspaketti voidaan hakea sieltä komennolla *apt-add-repository ppa:ansible/ansible* ja asentaa käyttämällä APT-pakettienhallintaan tarkoitettua ohjelmaa ensin päivittämällä paketit komennolla *apt update* ja tämän jälkeen asentamalla itse paketin komennolla *apt install ansible* (ks. Kuvio 14).



Kuvio 14. Ansiblen asennus.

Toinen yleisesti käytetty Linux-pohjainen käyttöjärjestelmä varsinkin yritysympäristöissä on Centos-käyttöjärjestelmä. Centos käyttää YUM-ohjelmaa pakettiensa hallintaan ja kyseisellä käyttöjärjestelmällä Ansible voidaan asentaa ensin hakemalla EPEL (Extra Packages for Enterprise Linux) YUM-ohjelman komennolla *yum install epel-release*. Lisäpakettien hakemisen jälkeen voidaan ansible taas asentaa YUM-ohjelman komennolla *yum install ansible*.



Kuvio 15. Hello-world ohjelman ajaminen Ansiblella.

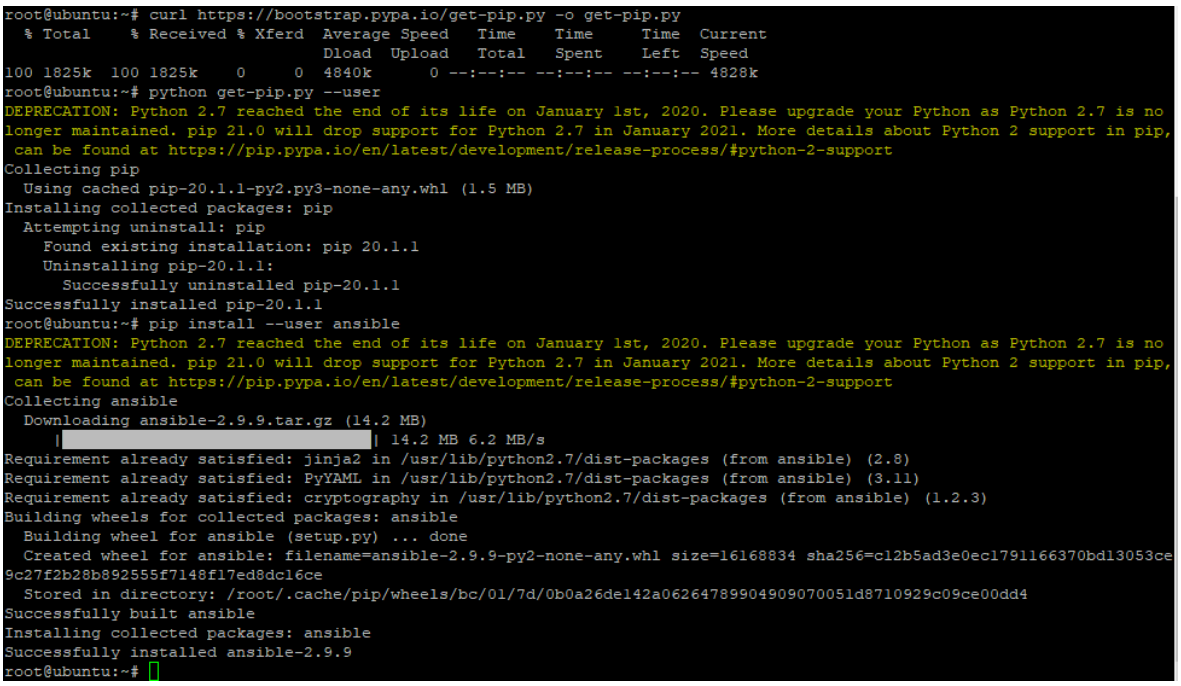
Ansible voidaan myös halutessa asentaa käyttämällä Pythonin PIP-ohjelmaa asentamalla ensin itse PIP-ohjelman komennoilla

curl https://bootstrap.pypa.io/get-pip-py -o get-pip.py

joka tekee tiedoston get-pip.py, jossa sijaitsee linkistä löytyvä data. Tämä tiedosto voidaan sen jälkeen asentaa Pythonin avulla komennolla

python get-pip.py –user

joka asentaa PIP-ohjelman ja PIP-ohjelman avulla komennolla *pip install –user ansible* voidaan asentaa ansible (Ks. Kuvio 16.).



Kuvio 16. Ansiblen asennus Pythonin PIP-ohjelmaa käyttäen.

5 Python ohjelmointikieli

5.1 Yleisesti

Python on tulkittu, oliokeskeinen, korkean tason ohjelmointikieli, jolla on dynaaminen semantiikka. Sen korkean tason sisäänrakennetut tietorakenteet yhdistettynä dynaamiseen kirjoittamiseen ja dynaamiseen sitomiseen tekevät siitä erittäin houkuttelevan nopeaa sovelluskehitystä varten, sekä skriptaus tai liimakielenä olemassa olevien komponenttien yhdistämiseksi toisiinsa. Pythonin yksinkertainen, helposti opittava syntaksi korostaa luettavuutta ja vähentää tarvetta ohjelman ylläpidolle. Python tukee moduuleja ja paketteja, jotka rohkaisevat ohjelman modulaarisuutta ja koodin uudelleenkäyttöä. Pythonin tulkki ja laaja standardikirjasto ovat saatavana lähde- tai binäärimuodossa ilmaiseksi kaikille tärkeimmille alustoille ja niitä voidaan levittää vapaasti.

Python on laajalti suosittu sen tuottavuuden lisäämisen vuoksi. Koska pythonissa ei ole kokoamisvaihetta, muokkaa-testaa-virheenkorjaus sykli on nopeaa. Python-ohjelmien vianmäärityksestä tekee helppoa, kun vika tai huono syöte ei koskaan aiheuta segmentointivirhettä. Sen sijaan, tulkin havaitessa virheen, se aiheuttaa poikkeuksen. Kun ohjelma ei löydä poikkeusta, tulkki tulostaa pinojäljen. Lähdetason virheenkorjain mahdollistaa paikallisten ja yleisten muuttujien tarkastamisen, mielivaltaisten lausekkeiden arvioinnin, raja-arvojen asettamisen, sekä koodirivin läpi käynnin rivi kerrallaan. Itse virheenkorjain on kirjoitettu pythonissa. Toisaalta usein nopein tapa korjata ohjelma on lisätä muutama tulostuslausunto lähteeseen, jonka avulla voidaan suorittaa nopeaa muokkaa-testaa-virheenkorjaus sykliä. (What is Python?)

5.2 PIP ja PyPI

PIP on pythonin pakettien asennukseen käytetty työkalu. PIP:n avulla voidaan asentaa paketteja pääsääntöisesti PyPIstä. PyPI indeksoi suuren määrän kirjastoja ja sovelluksia, jotka kattavat kaikki pythonin käyttökohteet. Paketteja asentaessa PIP ratkaisee ensin pakettien riippuvuudet, tarkistaa onko ne jo asennettu järjestelmään ja jos ei ole, asentaa ne. Kun kaikki paketin riippuvuudet on asennettu järjestelmään PIP asentaa pyydetyt paketit. Pakettien asennus tapahtuu oletuksena yleisesti, asentaen kaikki paketit koneeseen yhteen käyttöjärjestelmästä riippuvaan sijaintiin. (Pip – The Python Package Installer; Kollár 2019.)

6 Automatisoinnin suunnittelu

6.1 Vaatimusmäärittely

Automatisoinnin vaatimusmäärittelyn pohjakivenä toimii repetitiivisten tehtävien eliminointi. Verkossa ollessa monessa eri verkkosegmentissä useampi eri laite, jotka on

kahdennettu redundanttisuuden vuoksi, tarkoittaa suurta määrää laitteita ja näin ollen tarvetta konfiguraation muutokseen monelle eri laitteelle. Tavoitteena on siis automatisoinnin avulla saada ajettua uutta verkkoa luodessa usealle verkkolaitteelle tarvittava konfiguraatio. Tarvittaviin laitteisiin kuuluu mm. kytkimet, IP-osoitteen hallintapalvelu (IPAM), sekä palvelinalusta. Automatisoinnissa kuitenkin jätetään reitityspiste lisäämättä, koska siinä halutaan pitää kontrolli verkkojen suhteen. Tämän ansiosta, jos tehdään useampi kappale verkkoja automatisoinnin avulla, tarvitsee käydä erikseen lisäämässä verkon reitityspiste. Manuaalinen reitityspisteen lisäys auttaa myös tietoturvamielessä, koska sen seurauksena ei-toivotun laitteen liittyessä verkkoon se ei pysty keskustelemaan ulos muualle.

6.2 Käytetyt ohjelmointikielet

Automatisoinnin hallinnassa todettiin BASH-skriptauskieli olevan paras käytettävä, koska sillä mahdollistettiin helppo ohjelmien kutsuminen, automatisoinnissa käytettyjen ohjelmien hallinta, sekä yleisesti komentojen ajaminen palvelimella. Itse konfiguraatioiden automatisointiin valittiin Ansible-ohjelma, sen valmiiksi löytyvien laajojen moduulien ja hyvän hallinnan ja käytettävyyden vuoksi. Python-ohjelmointikieli havaittiin taas muiden komponenttien automatisointiin, joihin ei löytynyt valmiita moduuleja. Osana pythonin valintaan toimi myös Ansiblen toimivuus pythonin perustalla, jotta voitiin pitää palvelimelle asennusta vaativien ohjelmien määrä pienenä.

6.3 Automatisoinnin hyödyt

Automatisoinnin suurimpana hyötynä saadaan eliminoitua verkon luomisen prosessista inhimilliset virheet. Ajamalla samoja skriptejä joka kerta, pystytään varmistamaan, että prosessi suoritetaan joka kerta samalla tavalla. Kyseessä ollessa suurempi ympäristö nähdään automatisoinnin hyötyä myös työvoiman säästämisessä. Normaalisti yhtä kohdeverkkoa luodessa aikaa kuluu n. 30 minuuttia, mutta automatisointi

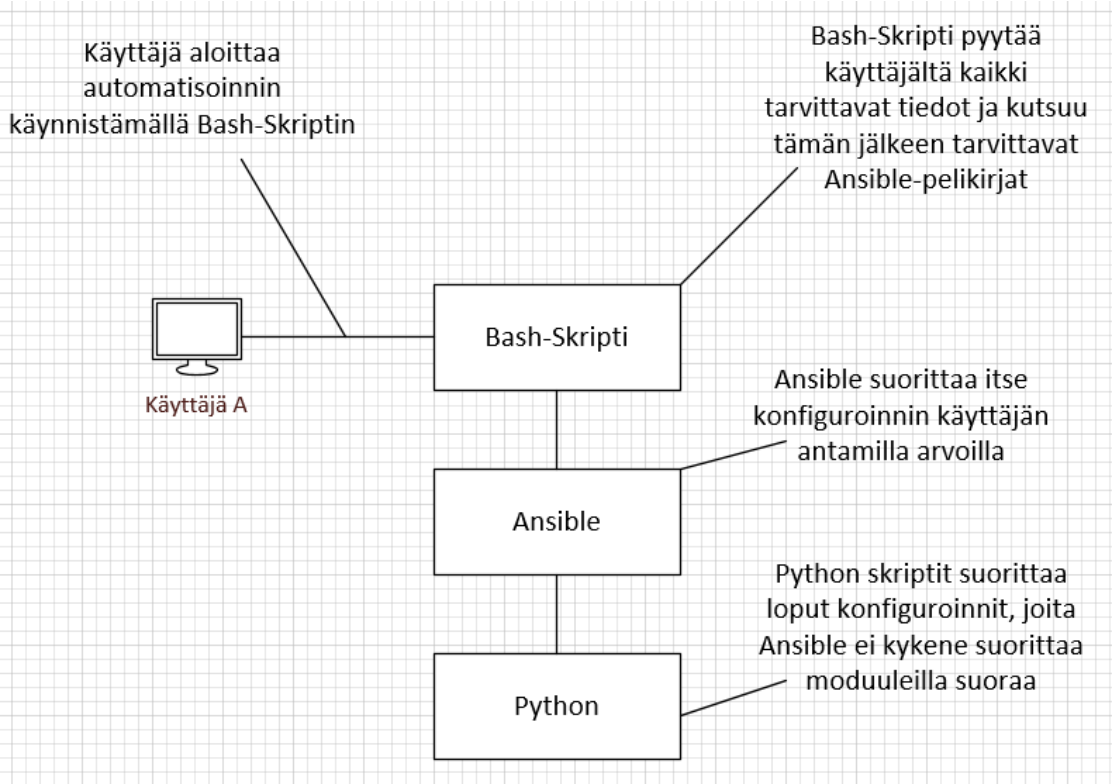
hoitaa saman tehtävän muutamassa minuutissa ja se ei vaadi ihmisvoimaa muuten kuin skriptin aloittamiseen.

6.4 Ympäristö

Automatisoinnin kohdeympäristönä toimii Kelan konesalit. Konesalien laitteisto koostuu useammasta eri mallin kytkimestä, palvelinalustasta ja IPAM-palvelusta, jonne verkko on luotava. Konesalit on kahdennettu ja jaettu kahdelle eri verkkosegmentille, joka lisää automatisoinnin haastavuutta ja kohdelaitteiden määrää (Ks. Liite 10.).

6.5 Tavoite

Automatisoinnin toiminnan tavoitteena on luoda skripti, jota käyttäjän on helppo käyttää ja tarvittaessa itse muokata. Tämä saavutetaan luomalla pääskripti, jota voidaan kutsua ja pääskripti kutsuu tarvittaessa muita skriptejä halutun konfiguraation luomiseksi. Pääskripti luodaan BASH-skriptauksella, koska sillä saadaan helposti ja selkeästi pyydettyä käyttäjältä kaikki tarvittavat tiedot verkon luomista varten ja sen avulla voidaan myös kutsua tarvittaessa pelikirjoja ja muita automatisointiin liittyviä ohjelmia. Itse laitteiden konfiguroinnin automatisointiin valittiin pääsääntöisesti Ansible sen kattavan moduulitarjonnan vuoksi, joka tukee suurinta osaa käytössä olevaa laitteistoa. Kaikkia tarvittavia komponentteja ei ole mahdollista automatisoida Ansiblen avulla, joten muiden automatisointiin kirjoitetaan itse python-skriptit ja käytetään niitä hyödyksi.

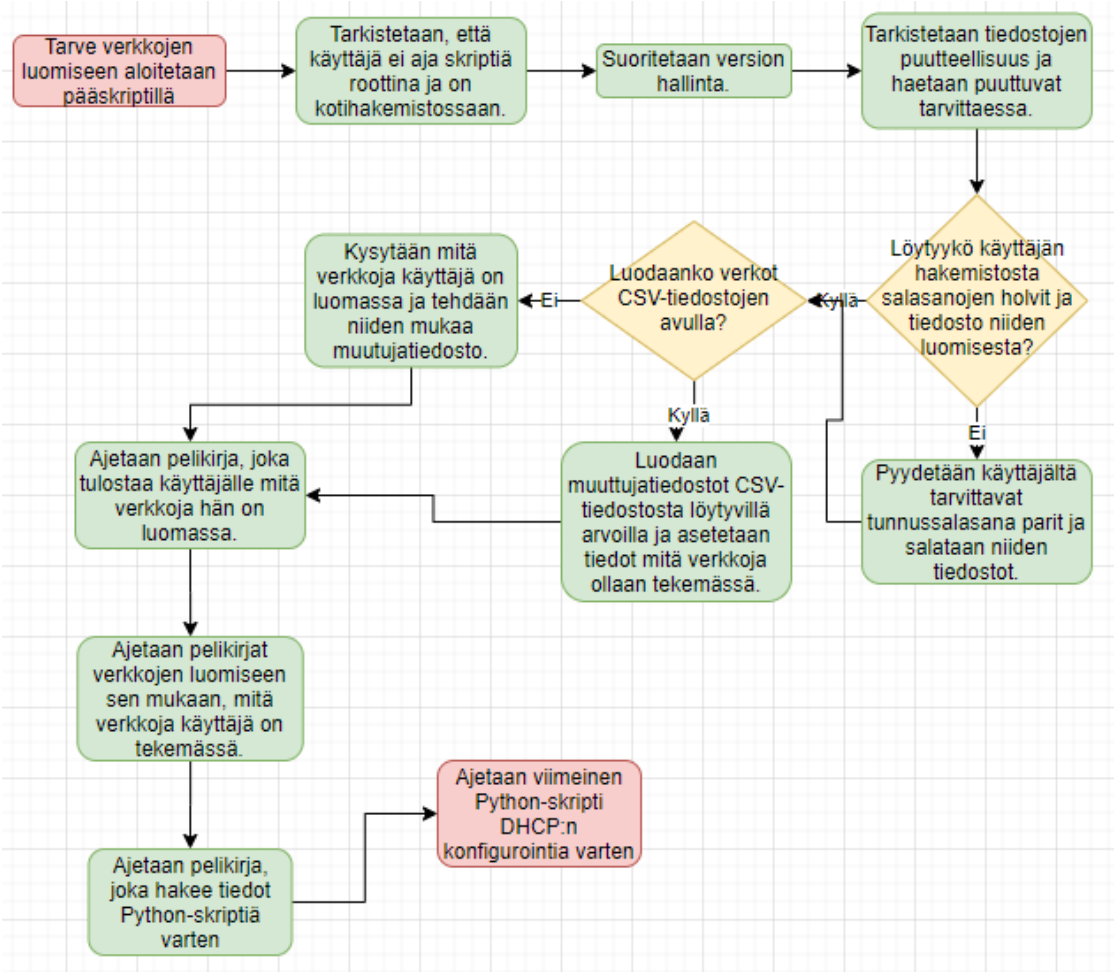


Kuvio 17. Automatisoinnin kulku

7 Automatisoinnin toteutus

7.1 Johdanto

Automatisoinnin tavoitteena on ajaa yhtä skriptiä, joka suorittaa tarvittava tiedostojen-, version- ja käyttäjähallinnan (Ks. Osio 7.2). Pääskripti kutsuu tarvittaessa pelikirjoja ja muita skriptejä tarvittavan automatiikan luomiseksi. Tämä voidaan kuvata helposti vuokaaviota käyttämällä (ks. Kuvio 18), jossa on käyty läpi automatiikan kaikki eri osat niiden suunnitellussa järjestyksessä.



Kuvio 18. Verkkojen luomisen vuokaavio.

7.2 BASH-kielen rooli automatisoinnissa

Pääskripti, joka kutsuu aliohjelmia ja pelikirjoja rakennettiin käyttäen BASH-skriptauskieltä, jota käytetään enimmäkseen Linux-pohjaisissa käyttöjärjestelmissä. Pääskriptin tavoitteena on toimia niin, että pelkästään sitä kutsumalla voidaan ajaa kaikki tarvittava automatiikka ilman muuta turhaa tarvetta. BASH-skriptin rakennus aloitettiin tiedostojen hallinnalla. Automatiikka tarvitsee toimiakseen useampaa eri skriptiä ja tiedostoa, joten tarvittavat tiedostot sijoitettiin palvelimella yhteen juuri-kansioon, josta skripti käy kopioimassa ne käyttäjän kotihakemistoon helpompaa käyttöä ja tarvittaessa omaa testailua ja muokkausta varten. Tässä vaiheessa päätet-

tiin, että skriptiä tulisi ajaa omalla käyttäjätunnuksella omasta kotihakemistosta automatisoinnin ja käyttäjätunnusten hallinnan helpottamiseksi. Skripti luo aluksi muuttujat käyttäjän omalle tunnukselle ja hakemistolle, josta käyttäjä ajaa skriptiä. Tämän jälkeen voitiin verrata käyttäjätunnuksen muuttujaa, ettei käyttäjä aja skriptiä juurena ja hakemiston osuvan käyttäjän kotihakemiston mukaiseen polkuun.

Käyttäjänhallinnan jälkeen skriptillä suoritettiin versiointi. Tämän avulla, käyttäjät voivat muokata ohjelmia halutessaan kotihakemistossa ja niitä ajaessa tiedostoja ei yli kirjoiteta juurihakemistosta löytyvillä tiedostoilla. Versioinnin avulla suoritettiin myös tarvittaessa mahdollisuus puhtaalle tiedostojen asennukselle poistamalla versiotiedosto, jonka jälkeen skripti kysyy käyttäjältä, haluaako hän hakea puhtaat tiedostot juurihakemistosta ja kopioida ne omaan kotihakemistoon. Versiointitiedostona toimii yksinkertainen tekstitiedosto, jonka ylimmässä rivissä sijaitsee automatiikan versionumero ja sen alle voi halutessaan kirjoittaa tehdyt muutokset. Versioinnin ajettaessa, skripti tarkistaa käyttäjän kotihakemistosta löytyvän versiotiedoston versionumeron ja vertaa sitä juuressa olevaan tiedostoon ja versioiden erotessa, kertoo käyttäjälle molempien versioiden numerot ja kysyy, haluaako hän hakea uudet tiedostot juurihakemistosta.

Automatiikan tarvittaessa useampaa määrää ohjelmia toimiakseen, skriptiin luotiin tarkistus, jonka avulla testataan, että ajettava pääskripti ei poikkea juuresta löytyvästä skriptistä ja kaikkien tarvittavien tiedostojen löytyvän. Puuttuvien tiedostojen hakeminen haluttiin suorittaa vain tarvittaessa, ilman ylikirjoittamista. Tämä suoritettiin kertomalla skriptille kaikki tarvittavat tiedostot automatiikan ajamiseen ja tiedoston puuttuessa, se kopioidaan erikseen kotihakemistoon. Samalla suoritetaan pääskriptien oikeellisuuden tarkastaminen luomalla ajetusta skriptistä hash-arvo ja vertaamalla sitä juurihakemistosta löytyvän pääskriptin hash-arvoon. Arvojen poiketessa toisistaan käyttäjää varoitetaan, mutta skriptin ajoa ei estetä käyttäjän tehdessä omia testauksiaan.

Laitteiden suuren määrän ja eri tyyppien takia usealle laitteelle käytetään eri kirjautumistapoja. Tämän takia piti kehitellä tapa, jolla käyttäjä pystyy ajamaan skriptejä eri laitteille eri tunnuksilla mahdollisimman vaivattomasti. Tässä otettiin apuun Ansiblen Vaultit. Ansible Vaulttien avulla pystyttiin kirjoittamaan käyttäjän eri laitteiden tunnukset ja salasanat tiedostoihin ja salaamaan tiedostot niin, että kukaan muu ei pääse niihin käsiksi ilman pääsalasanaa. Tämä suoritettiin skriptillä pyytämällä muuttujiin käyttäjän tunnussalasana -parit ja kirjoittamalla ne tiedostoihin. Tämän jälkeen suoritettiin Ansible Vaultin avulla tiedostojen salaaminen ja samalla Ansible Vault pyytää käyttäjältä omien tiedostojen pääsalasanan. Skripti tarkistaa, että käyttäjän kotihakemistosta löytyy kaikki tarvittavat holvit ja jos niissä havaitaan puutteita, pyydetään käyttäjää syöttämään tunnussalasana -parit uudelleen. Tämän avulla saatiin myös suoritettua salasanojen vaihtaminen. Käyttäjän salasanan vanhetessa voi hän poistaa holvitiedoston ja skripti pyytää häneltä tunnussalasana -parin uudelleen.

Verkkojen luontia varten skriptiin tehtiin aliohjelma, joka kysyy minkä alueen ja minkä tyyppistä verkkoa käyttäjä haluaa tehdä. Tästä haluttiin tehdä mahdollisimman käytännöllinen, joten verkkojen luontia ei rajoitettu vain yhteen, vaan käyttäjä pystyy tekemään useamman määrän verkkoja tarvittaessa kerralla. Selvempää käyttöä varten rajoitettiin kuitenkin, että käyttäjä pystyy tekemään vain yhden tyyppisiä verkkoja kerralla esimerkiksi testiin. Verkkoja kysyttäessä skripti tallentaa tiedot taulukkoihin, joiden avulla voidaan kirjoittaa Ansiblea varten muuttujatiedosto, josta Ansiblen pelikirjat voivat hakea tietoja.

Suurta verkkomäärää ajettaessa luotiin käyttäjille mahdollisuus ajaa skripti CSV-tiedoston avulla. Käyttäjän tehdessä oikealla muotoilulla CSV-tiedoston skripti kysyy, haluaako hän ajaa verkot sen avulla. CSV-tiedostoa käyttäessä havaittiin parhaaksi tavaksi luoda aliohjelma, joka hakee CSV-tiedostosta tarvittavat arvot ja niiden avulla kirjoittaa Ansiblen muuttujat automatiikkaa varten samalla tavalla kuin käyttäjän itse syöttäessä tietoja skriptille.

Viimeisenä osana pääskriptiä, kun tiedostot on haettu, käyttäjätiedot ja verkkojen tiedot on paikoillaan, skripti ajaa tarvittaessa Ansiblen pelikirjoja, jotka suorittavat itse automatiikan (ks. Osio 7.3) ja ajaa tarvittavat python-skriptit (ks. Osio 7.4).

7.3 Ansiblen rooli automatisoinnissa

Ansiblen tavoitteena oli koittaa automatisoida mahdollisimman paljon osa-alueita ja pitää Ansiblen pelikirjojen sisällä mahdollisimman moni osa-alue. Tavoitteena oli luoda mahdollisimman helposti luettava ja muokattava automatiikka, jotta se voidaan jatkossa implementoida muualle tarvittaessa. Ansiblen automatisointi suoritettiin luomalla jokaiselle verkkotyypille ja verkkosegmentille omat pelikirjat, joita pääskripti kutsuu. Pelikirjojen sisällä suoritettiin itse automatisointi.

Automatisointi aloitettiin kytkimistä, koska ne ovat verkon keskeisin asia ja yhdistää kaikki eri komponentit. Kytkinten automatisointia varten löytyy verkosta valmiit Ansiblen moduulit, joita käytettiin hyödyksi. Kytkimien automatisointi aloitettiin ottamalla kaikista kytkimistä, joihin automatiikka on tekemässä muutoksia, varmuuskopiot. Tällä voidaan varmistaa, että jos jokin sattuu menemään pieleen, käyttäjä voi hakea kotihakemistostaan varmuuskopion ja ajaa sen välittömästi kytkimiin, ettei virheistä satu haittaa. Varmuuskopioiden jälkeen automatiikka luo VLANit kytkimille ja lisää ne kaikkiin tarvittaviin rajapintoihin, että VLANit pystyvät keskustelemaan tarvittavien komponenttien kanssa.

Kytkinten automatisoinnin jälkeen siirryttiin palvelinalustoille, joissa palvelimet pääsääntöisesti sijaitsevat. Palvelinalustojen automatisointi saatiin myös tehtyä helposti, sillä käytetyt alustat ovat saman merkkisiä ja nämäkin yleisiä, joten Ansiblesta löytyi välittömästi valmiit moduulit niitä varten. Palvelinalustojen moduulit vaativat kuitenkin pythonilla asennettavan moduulin toimiakseen. Pythonista löytyvän moduulin

asennuksen jälkeen, saatiin palvelinalustojen automatiikka luotua myös vaivattomasti.

Verkon viimeisenä komponenttina, jonne verkot piti luoda, on IPAM-palvelu. IPAM-palvelulle löytyi myös suurin osa moduuleista valmiiksi, yhtä lukuun ottamatta (ks. Osio 7.4). Tässä vaiheessa olisi ollut mahdollista suorittaa IPAM-palvelun kaikki tarvittava automatisointi Pythonin avulla, mutta havaittiin järkevämmäksi tavaksi käyttää Ansiblea siellä missä mahdollista. Näin ollen kirjoitettiin Ansiblesta löytyvien moduulien avulla automatiikka, jonka avulla saatiin luotua verkkoalue IPAM-palvelulle. Tämän moduulin suorittamiseen oli myös tarve asentaa pythonilla erillinen moduuli.

Vaikka verkkoalueita ja tyyppejä oli useampi määrä, saatiin Ansiblen avulla helposti luotua automatiikka niistä jokaiselle. Kun oli kirjoitettu pohja yhden verkon automatiikalle, voitiin se kopioida ja korvata yhden verkontyypin vaativat tiedot, kuten rajapinnat, toisella. Automatiikkaa luodessa käytettiin myös osiossa 7.2 mainittuja muuttujia, joten tietojen haku ei tuottanut ongelmia eri verkkoalueilla ja tyypeillä. Piti muuttaa pelkästään Ansiblen käyttämä muuttujalista oikealle verkkotyypille, jonka jälkeen Ansible haki tarvittavat tiedot oikeasta paikasta.

7.4 Python-kielen rooli automatisoinnissa

Automatisointia aloittaessa, huomattiin että kaikkia tarvittavia komponentteja ja niiden osia ei pystytty automatisoimaan valmiilla moduuleilla, joten näiden komponenttien automatisointiin käytettiin kustomoimalla tehtyjä omia ohjelmia ja niitä kutsutaan tarvittaessa rekursiivisesti muilla skripteillä. Tässä tapauksessa käytetyt laitteet ovat hyvin suosittuja, joten ainut osa mille ei löytynyt suoraa automatisoitua Ansiblen moduulia, on verkko-osoitteiden varaamista varten IP-osoitteiden hallintapalvelusta.

Kohdeympäristössä käytössä olevassa IP-osoitteiden hallintapalvelussa on käytettävissä REST-api (Representational state transfer), joka toimii kommunikointikanavana palvelimen ja tietokoneen välillä. REST-apia voitiin hyödyntää verkko-osoitteiden varaamiseen puskemalla tarvittavat konfiguraatiot kohdepalveluun sen avulla. Koska automatisoinnilla on tarve pystyä tehdä useampi verkkoalue kerralla, piti ohjelmaan implementoida mahdollisuus luoda myös useampi varaus kerralla. Tämä suoritettiin pythonilla tekemällä ensin ohjelma, joka hakee verkkoalueista varaukseen tarvittavan osoiteavaruuden ja tekee näistä CSV-tiedoston kohdepalvelimen luettavassa muodossa. Kohdepalvelimen API-rajapinnassa on mahdollisuus tuoda CSV-tiedoston avulla dataa ja kirjoittaa ne riveiksi palvelimen tietokantaan. Siispä Pythonin avulla tehtiin ohjelma, joka haki CSV-tiedoston, jossa osoiteavaruudet ovat, tallensi sen kohdepalvelimelle, käynnisti tietojen viemisen tietokantaan ja tietojen viemisen jälkeen kuittaa palvelimelle ohjelman loppumisesta.

8 Automatisoinnin testaus

8.1 Testiympäristö

Testiympäristössä voitiin käyttää Kelalla valmiiksi löytyviä tuotantoympäristöä vastaavia vanhempia fyysisiä laitteita, virtuaalisesti emuloituja laitteita sekä erillisiä laboratoriokäyttöön suunniteltuja laitteita. Testiympäristön kattavan laitemäärän ansiosta saatiin rakennettua tuotantoa vastaava ympäristö automatisoinnille, jonka seurauksena automatisoinnin toimiessa se voitaisiin ottaa käyttöön tuotannossa siirtämällä ohjelmat oikeille palvelimille ja vaihtamalla kohdelaitteiden IP-osoitteet ja nimet oikeiksi. Testiympäristönä toimi tier 3 vastaava laitteiston infrastruktuuri.

8.2 Testitulokset

Vaatimuksena testiympäristössä piti saada luotua verkkokonfiguraatiot kaikille tarvittaville laitteille tietoturvallisessa ja helposti käytettävässä tavalla, samoin kuten vaatimusmäärittelyssä todettiin. Tässä vaiheessa saatiin hyvin myös testattua ohjelmien eri koodien osien toimivuutta, sillä konfigurointivirheen sattuessa se ei vaikuttaisi tuotannossa oleviin oikeisiin konfiguraatioihin mitenkään. Testivaiheessa meni jonkin aikaa, koska samalla havaittiin haasteita skriptien käyttäjäystävällisyyttä, tietoturvallisuutta ja automatisoinnin hallintaa luodessa. Haasteista päästiin kuitenkin yli käyttämällä aliohjelmia avuksi pääskriptissä, jotka auttoivat jokaisen osa-alueen hallinnassa.

9 Yhteenveto

9.1 Pohdinta

Opinnäytetyön tavoitteena oli automatisoida mahdollisimman monta eri komponenttia verkon luomisessa. Automatisoinnista oli käyty jo pitkään keskustelua, mutta ajan puutteen vuoksi siihen ei keretty perehtymään. Automatisointi oli helppo aloittaa koulusta opittujen pohjatietojen avulla, sekä itseopiskelemalla ja verkosta löytyvien avuliaitten materiaalien avulla. Määrällisesti automatisoinnin vaade oli suuri, joten automatisoinnin luomisessa ja kehittämisessä kesti odotettua enemmän aikaa. Aikavaatimuksella ei kuitenkaan ollut erillistä määrettä, joka oli hyödyksi ja voitiin rauhassa suorittaa automatisointia eri komponentteihin. Automatisoinnin suorittamista pitkitti myös uuden järjestelmän saapuminen, joka piti myös ottaa huomioon ohjelmissa.

Automatisoinnissa käytetystä ohjelmistosta koitettiin saavuttaa mahdollisimman helposti luettavaa ja -käytettävää. Skripteissä käytettiin kommentteja, kertomaan miksi

mahdollisesti muiden silmiin epäselviä koodeja oli kirjoitettu, sekä rivinvaihtoja ja välitystä paremman helposti luettavuuden saavuttamiseksi. Aliskriptejä käytettiin myös mahdollisuuksien mukaan, koska niitä on helpompi muokata, etsiä kullekin osalle kuuluvat koodit ja niitä voidaan kutsua helpommin.

Automatisoinnin yksi pääpiirteistä oli helpottaa työnmäärää ajamalla skripti, joka luo tarvittavat verkkoalueet. Normaalisti käsin tehdessä yhden verkon luomisessa kesti n. 20 – 30minuuttia ja automatisoinnin valmistuessa, voitiin suuri määrä verkkoja luoda kerralla ja tehdä muita töitä samalla. Ajan säästö huomattiin jo heti automatisoinnin valmistumisen jälkeen ensimmäisiä verkkoja luodessa. Ensimmäisiä verkkoja luodessa havaittiin hieman tarvittavaa säätöä, koska verkoissa havaittiin toimimattomuutta. Toimimattomuuksien paikallistamisen jälkeen saatiin kuitenkin ohjelmiin tehtyä tarvittavat muutokset, jonka jälkeen saatiin myös eliminoitua inhimilliset virheet. Kaikki verkot tehdään automatiikan avulla samalla tavalla, jonka ansiosta yhden verkon toimessa kaikki muutkin verkot toimivat.

9.2 Jatkokehitys

Vaikka automatisointi saatiin suoritettua kaikille tarvittaville komponenteille, löytyy sille koko ajan lisää kehitettävää. Automatisoinnilla saa luotua normaaleja konesaliverkkoja, mutta verkkoja on niin monta erityyppistä, että niille saa kehitellä lisää pelikirjoja ja aliohjelmia. Myös uusien ja erityyppisten laitteiden tullessa konesaleihin tai konesalien infrastruktuurin muuttuessa, tarvitsee ohjelmia ja pelikirjoja muuttaa sen mukaan.

Lähteet

Da Silva, J. Jr. 2019. What you need to know about Ansible modules. Opensource verkkosivuilla. https://opensource.com/article/19/3/developing-ansible-modules

Data Center Standards (Tiers I-IV). N.D. Colocation America verkkosivuilla. Viitattu 10.5.2020. https://www.colocationamerica.com/data-center/tier-standards-overview.htm

Di Palo, R. 2018. DMZ in a nutshell: An introduction. Acrosec verkkosivuilla. Viitattu 10.4.2020. https://www.acrosec.jp/dmz-intro/?lang=en

Henkilöstötilinpäätös 2018, 2019. Kansaneläkelaitoksen verkkosivuilla. Viitattu 5.6.2020. https://www.kela.fi/documents/10180/17802081/Henkilostotilinpaatos_2018_valmis.pdf/98b4a980-e330-41c2-a103-427e80c09e3e?version=1.0

How Ansible Works. N.d. Red Hat verkkosivuilla. Viitattu 20.3.2020. https://www.ansible.com/overview/how-ansible-works

How Does a Network Switch Work. 2018. Fiber Optic Network Products verkkosivuilla. Viitattu 2.4.2020. http://www.fiberopticshare.com/network-switch-work.html

Hummel, G. 2019. What is Ansible? Cloudacademy verkkosivuilla. Viitattu 25.3.2020. https://cloudacademy.com/blog/what-is-ansible/

Kollár, L. K. 2019. Managing Python packages the right way. Opensource verkkosivuilla. Viitattu 21.10.2019. https://opensource.com/article/19/4/managing-python-packages

Layer 2 vs Layer 3 Switch: Which One Do You Need? 2017. FS verkkosivuilla. Viitattu 18.4.2020. https://community.fs.com/blog/layer-2-switch-vs-layer-3-switch-which-one-do-you-need.html

Mohammed, M., 2014. Guide to OSI and TCP/IP models. Viitattu 11.4.2020. https://janet.finna.fi, Books24x7

Organisaatio. N.d. Kansaneläkelaitoksen verkkosivuilla. Viitattu 14.5.2020. https://www.kela.fi/organisaatio

OSI ja TCP/IP-malli. N.d. Krimaka verkkosivuilla. Viitattu 11.4.2020. http://www.krimaka.net/tietotekniikka/verkko-ja-ethernet/osi-ja-tcp-ip-mallit.html

Pip – The Python Package Installer. N.d. Pypa verkkosivuilla. Viitattu 21.10.2019. https://pip.pypa.io/en/stable/

Rouse, M. 2017. Ansible playbook. TechTarget verkkosivuilla. Viitattu 12.4.2020. https://searchitoperations.techtarget.com/definition/Ansible-playbook

Rouse, M. 2019. Router. TechTarget verkkosivuilla. Viitattu 23.5.2020. https://searchnetworking.techtarget.com/definition/router

Rouse, M. N.d. Intranet. TechTarget verkkosivuilla. Viitattu 20.4.2020. https://whatis.techtarget.com/definition/intranet

TCP/IP Model. N.d. GeeksforGeeks verkkosivuilla. Viitattu 11.4.2020. https://www.geeksforgeeks.org/tcp-ip-model/

Tier 2 Data Center. N.d. Techopedia verkkosivuilla. Viitattu 1.5.2020. https://www.techopedia.com/definition/29858/tier-2-data-center

Tier 3 Data Center. 2014. Techopedia verkkosivuilla. Viitattu 23.4.2020. https://www.techopedia.com/definition/29859/tier-3-data-center

Understanding the Ansible Inventory File When Managing Devices Running Junos OS. 2018. Juniper Networks verkkosivuilla. Viitattu 25.3.2020. https://www.juniper.net/documentation/en_US/junos-ansible/topics/concept/junos-ansible-inventory-file-overview.html

Understanding VLAN Trunking. 2019. Solarwinds msp verkkosivuilla. Viitattu 10.3.2020. Understanding the Ansible Inventory File When Managing Devices Running Junos OS. 2018

Using Variables. 2020. Ansible verkkosivuilla. Viitattu 22.3.2020. https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html

VLAN. N.d. Orbit Computer Solutions verkkosivuilla. Viitattu 25.4.2020. https://www.orbit-computer-solutions.com/vlan/

What is A Data Center?. N.d. Palo Alto Networks verkkosivuilla. Viitattu 10.6.2020. https://www.paloaltonetworks.com/cyberpedia/what-is-a-data-center.

What Is Python? N.d. Python verkkosivuilla. Viitattu 21.10.2019. https://www.python.org/doc/essays/blurb/

Who is Cisco. N.d. Cisco verkkosivuilla. Viitattu 9.4.2020. https://www.cisco.com/c/en_au/about/who-is-head.html