jamk

Verkko- ja palvelinlaitteiden valvonta-konfiguraation automatisointi Ansiblella

Niko Kallio

Opinnäytetyö, AMK
Elokuu 2025
Tieto- ja viestintätekniikan tutkinto-ohjelma (AMK)

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Kallio, Niko

Verkko- ja palvelinlaitteiden valvontakonfiguraation automatisointi Ansiblella

Jyväskylä: Jyväskylän ammattikorkeakoulu. Elokuu 2025, 53 sivua.

Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

IT-infrastruktuuripalveluja tarjoavilla yrityksillä voi olla monia eri valmistajien verkkolaitteita tai eri Linux-jakeluiden palvelimia käytössä. On tärkeää pystyä valvomaan näitä laitteita muun muassa lokeja keräämällä. Eri valmistajien ja käyttöjärjestelmien laitteista koostuvassa ympäristössä voi kuitenkin olla haastavaa manuaalisesti konfiguroida valvontalokien lähettäminen nopeasti, koska laitteiden syntakseissa voi olla eroavaisuuksia.

Opinnäytetyö toteutettiin tutkimuksellisena kehittämistyönä. Tavoitteena oli kehittää TNNet Oy:lle ratkaisu, jossa Ansiblea käyttämällä voidaan automatisoida valvonnan konfigurointi huomioiden mahdolliset syntaksierot. Lisäksi täytyi rakentaa lokitusympäristö, jossa lokitus tapahtuisi modulaarisesti, eli vain valitut tiedot lokitettaisiin kytkimistä ja palvelimista.

Toteutuksessa määritettiin Ansiblen pelikirjoja käyttämällä kaikki ympäristössä olevat laitteet lähettämään lokiviestit lokipalvelimelle. Mahdollisten syntaksieroavaisuuksien ratkaisuksi laitteista kerättiin eri tavoilla tieto niiden valmistajasta tai käyttöjärjestelmästä, jolloin Ansible pystyi valitsemaan oikean konfiguraatiotiedoston, jotka oli toteutettu Jinja-mallipohjia käyttämällä. Lokitusympäristössä lokien vastaanottamiseen käytettiin Graylogia, jonka avulla lokeja voitiin tarkastella ja analysoida reaaliajassa.

Tulokset osoittivat, että Ansible-rakenne saatiin toteutettua toimeksiantajan haluamalla tavalla. Kaikille testiympäristön laitteille saatiin konfiguroitua lokien lähettäminen lokipalvelimelle yhdellä Ansible-pelikirjalla. Rakenteessa kuitenkin havaittiin kohtia, jotka olisi voitu toteuttaa paremmin tai eri tavalla. Lokitusympäristön tuloksissa havaittiin lokipalvelimen vastaanottavan vain haluttuja viestejä. Tuloksiin perustuvissa keskeisimmissä havainnoissa osoitettiin laitevalvonnan konfiguroinnin onnistuvan Ansiblen automaation avulla

Avainsanat (asiasanat)

Ansible, Automaatio, Konfigurointi, Lokitus, Graylog, Kytkin, Palvelin

Muut tiedot (salassa pidettävät liitteet)

-

jamk

Description

Kallio, Niko

Automation of Monitoring Configuration on Network and Server Devices using Ansible

Jyväskylä: JAMK University of Applied Sciences, September 2020, 53 pages.

Degree Programme in Information and Communications Technology. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

Companies providing IT-infrastructure services may have network devices from different manufacturers or servers with different Linux distros. It is important for these companies to be able to monitor these devices by collecting logs, among other things. However, in an environment composed of devices from different manufacturers and operating systems, it can be challenging to manually configure the sending of monitoring logs quickly, because there might be differences in device syntax.

The thesis was carried out as research-based development work. The objective of the work was to develop a solution for TNNet Oy where Ansible could be used to automate monitoring configuration, considering possible syntax differences. In addition, a logging environment had to be built where logging would be done modularly, meaning that only selected logs would be collected from switches and servers.

In the implementation, Ansible playbooks were used to configure all devices in the environment to send log messages to a log server. To solve potential syntax differences, information about the device manufacturer or OS was collected in different ways, allowing Ansible to select the correct configuration file, which was implemented using Jinja templates. Graylog was used to receive logs in the logging environment, which allowed for real-time viewing and analysis of logs.

The results showed that the Ansible structure was implemented as the client desired. Log forwarding to the log server was successfully configured for every device in the test environment using a single Ansible playbook. However, certain areas of the structure were found that could have been implemented better or differently. The results from the logging environment showed that the log server was receiving only the desired messages. The key findings based on results showed that device monitoring can be successfully configured with Ansible automation, but certain things need to be paid attention to.

Keywords/tags (subjects)

Ansible, Automation, Configuration, Logging, Graylog, Switch, Server

Miscellaneous (Confidential information)

-

Sisältö

Kuviot

Taulukot

Sanasto

Ansible

Ohjelmisto, joka toimii työkaluna IT-järjestelmien automatisointiin.

CLI

Command Line Interface, eli komentorivi. Käytetään antamaan komentoja järjestelmälle.

Kytkin

Laite, johon muut verkon laitteet on kytketty ja välittää tietoliikenteen yhdestä laitteesta toiseen, valittuun laitteeseen.

Infrastructure as Code (IaC)

Infrastruktuuri Koodina. Prosessi, jolla voidaan hallita infrastruktuuria ohjelmoitavalla koodilla.

Inventaario

Ansiblen osa, jonne lisätään hallittavien laitteiden tietoja.

IP-osoite

Internetiin yhdistetyn laitteen tai verkkoliittyvän yksilöivä numeraalinen tunniste.

Palvelin

Tietokone tai ohjelma, joka tekee tiettyjä tehtäviä muille tietokoneille verkon yli.

Pelikirja

Pohjapiirustus Ansiblen automatisoitaville prosesseille, joka koostuu peleistä.

Peli

Vähimmillään määrittely, millä laitteilla suoritetaan mitkäkin tehtävät. Pelejä voi olla yksi tai useampi pelikirjassa.

Portti

Protokollien numeroitu yhtymäkohta, joka välittää tietoliikenteen oikeaan ohjelmaan.

SSH

Secure Shell. Protokolla, jolla voidaan muodostaa salattu yhteys tiettyyn järjestelmään.

Syslog

Lokiviestien lähettämiseen tarkoitettu protokolla.

1 Johdanto

1.1 Tausta ja tavoitteet

Laitevalvonta on tärkeä osa IT-infrastruktuurin ylläpitoa. IT-infrastruktuuripalveluja tarjoavissa yrityksissä voi olla käytössä useita eri verkko- ja tai palvelinlaitteita, joita täytyisi valvoa. Laaja ja hajautettu ympäristö voi kuitenkin koostua useiden eri valmistajien laitteista tai alustoista, jolloin yhtenäisen valvonnan konfigurointi voi olla haastavaa. Yksi olennainen osa IT-infrastruktuurin laitevalvontaa on erilaisten lokien kerääminen, jotka tarjoavat selkeän kuvan järjestelmän käyttäytymisestä, suorituskyvystä ja turvallisuudesta (Stawiarska 2023).

Laitteiden manuaalinen konfigurointi kuluttaa sekä paljon aikaa että yrityksen resursseja, minkä lisäksi laitteiden manuaalinen konfigurointi voi johtaa virhekonfiguraatioisiin (Hyde 2025). Riski virhekonfiguraatioille on todennäköisesti suurempi, kun ympäristössä on erilaisia laitteita, joiden syntaksi voi poiketa toisistaan. Yksi ratkaisu näiden ongelmien välttämiseksi on automatisoida valvonnan konfigurointiprosessi tehokkaasti Ansiblen avulla. Ansiblen kaltaisten automaatiotyökalujen avulla kaikkia laitteita voidaan hallita keskitetysti ja luotettavasti.

Tämän opinnäytetyön toimeksiantajalla TNNet Oy:lla oli haasteena useampi käytössä oleva kytkinmalli ja Linux-jakelu. Tässä kehittämistyössä oli tavoitteena suunnitella ja toteuttaa Ansible-ohjelmistoa käyttäen ratkaisu, jossa valvonnan konfigurointi voidaan automatisoida niin, että järjestelmä tunnistaa eri laitteiden syntaksin. Osana kokonaisuutta rakennetaan myös lokitusympäristö, jossa lokitus tapahtuu modulaarisesti.

Toteutusta varten aihe on rajattu koskemaan Ruckusin ja Huawein kytkimien, sekä AlmaLinuxia ja Ubuntua käyttävien palvelimien valvontakonfiguraatioiden automatisoimista. Opinnäytetyötä varten käytettiin Ansiblea, koska se oli toimeksiantajalla jo entuudestaan käytössä. Opinnäytetyön valmiiden tulosten avulla toimeksiantajan kaltaiset IT-alan toimijat voisivat automatisoida omien laitteidensa valvonnan konfiguroinnin Ansiblella ja näin ollen vähentää tarvetta manuaalisen konfiguroinnille.

1.2 Toimeksiantaja TNNet Oy

Opinnäytetyön toimeksiantaja oli TNNet Oy. TNNet Oy on kotimainen, vuonna 2002 perustettu IT-infraratkaisuja tarjoava yritys. Yritys perustettiin alun perin operaattoriksi, mutta pian se rupesi tarjoamaan sivutuotteina tietoliikenneyhteyspalveluita sekä pelipalvelimia. Myöhemmin TNNet perusti myös oman konesalin. Nykyään sen palvelutarjontaan kuuluu myös tietoturvan ja sisäverkon ratkaisut. TNNet Oy:n liikevaihto vuonna 2024 oli 4,1 miljoonaa euroja ja se työllistää yhteensä 16 henkilöä vuonna 2025. (Tietoja meistä n.d.)

1.3 Tutkimusasetelma

Tämä opinnäytetyö toteutettiin tutkimuksellisena kehittämistyönä, koska työssä pyrittiin keksimään ratkaisu todelliseen toimeksiantajan ongelmaan. Toikko ja Rantanen (2009) kertovat tutkimuksellisessa kehittämistoiminnassa yhdistyvän sekä tutkimukselliset menetelmät että käytännön kehittäminen. Tutkimuksellisessa kehittämistoiminnassa pääpaino on kehittämisessä, jota tutkimus tukee. (Toikko & Rantanen 2009.)

Tutkimusasetelmassa olennaisessa osassa ovat tutkimuskysymykset. Tutkimuskysymykset ohjaavat kehittämisprosessia ja niiden lähtökohtaisesti tulisi olla tarkoin määriteltyjä (Toikko & Rantanen 2009). Tutkimusongelman ja tavoitteiden perusteella tätä työtä varten laadittiin kolme tutkimuskysymystä. Laaditut tutkimuskysymykset ovat valittu niin, että ne vastaavat toimeksiantajan esille nostamiin käytännön haasteisiin sekä kehittämistarpeisiin.

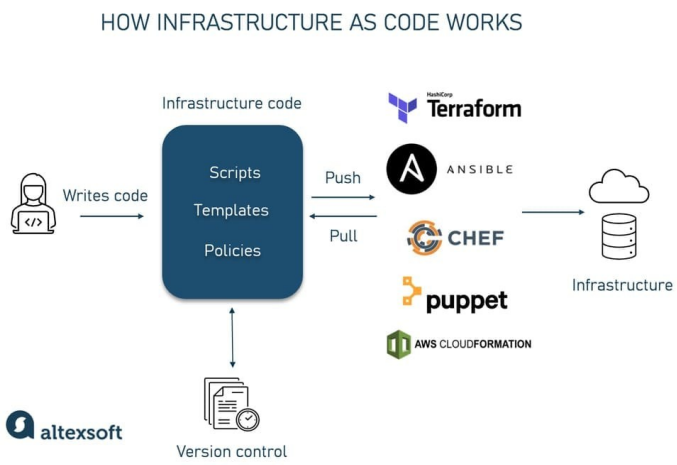
1. Miten verkkolaitteiden ja palvelimien valvonta voidaan automatisoida skaalautuvasti ja modulaarisesti Ansiblea hyödyntäen?
2. Mitä lokitietoja on olennaista kerätä palvelinten ja kytkinten ylläpidon ja tietoturvan kannalta?
3. Miten Ansible voidaan hyödyntää tunnistamaan laitteistokohtaisia erityispiirteitä, kuten syntaksieroja?

2 Ansiblen ja lokituksen perusteet

2.1 Infrastruktuuri koodina

Infrastruktuuri koodina (Infrastructure as Code, IaC), on prosessi, jonka avulla IT-infrastruktuuria voidaan hallita automaattisesti skriptien avulla sen sijaan, että työ tehtäisiin manuaalisesti (ks. kuvio 1). Ilman tätä prosessia, infrastruktuurin hallinnasta vastaavat tiimit joutuisivat konfiguroiman

siihen kuuluvat laitteet tai palvelut erikseen. Ajan myötä tämä voi johtaa siihen, että identtisiksi tarkoitetuista ympäristöistä voi tulla epäjohdonmukaisia, mikä vaikeuttaa niiden konfigurointi sekä hidastaa käyttöönottoprosesseja. IaC-prosessin etuja ovat: alhaisemmat kustannukset, johdonmukainen ympäristö sekä versionhallinta. (Synytsia 2022.)



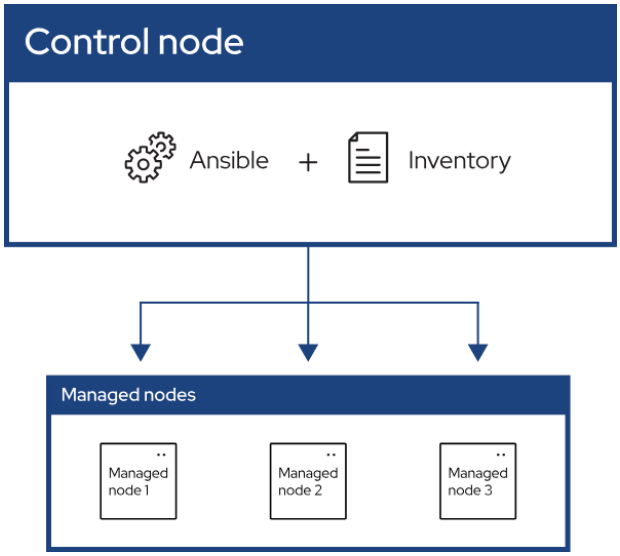
Kuvio 1. Infrastruktuuri koodina havainnollistettuna (Synytsia 2022)

IaC-työkalut voidaan jakaa kahteen eri luokkaan, provisiointiin tarkoitettuihin työkaluihin ja konfiguraation hallintaan tarkoitettuihin työkaluihin. Provisiointi on prosessi, jonka aikana perustetaan IT-infrastruktuuri, eli virtuaalikoneet, tietokannat, resurssien käyttöoikeudet sekä niiden tuominen käyttäjien saataville. Konfiguraation hallinta puolestaan tapahtuu provisioinnin jälkeen, jonka aikana asennetaan ohjelmistoja, konfiguroidaan ja ylläpidetään. Osa IaC-työkaluista sopii molempiin, mutta tyypillisesti ne on suunniteltu vain tiettyä tehtävää varten. Suosituimpia IaC-työkaluja ovat Ansible, Terraform, Puppet, Chef ja AWS CloudFormation. (Synytsia 2022.)

2.2 Ansible automaation välineenä

Ansible on Red Hatin ylläpitämä avoimen lähdekoodin työkalu, jota käytetään IT-järjestelmien automatisointiin, konfiguraatioiden hallinnointiin, sovellusten käyttöönottoon sekä moneen muuhun prosessiin. Ansiblen suurimmat vahvuudet ovat sen yksinkertaisuus ja helppokäyttöisyys. Lisäksi Ansible keskittyy turvallisuuteen ja luotettavuuteen. (How Ansible Works n.d.)

Awati ja Coutemanche (2023) kertovat Ansiblen arkkitehtuurin perustuvan konseptiin ohjaussolmuista ja hallinnoitavista solmusta (ks. kuvio 2). Ohjaussolmuja käytetään komentojen suorittamiseen, kun taas hallinnoitavat ovat laitteita, joita ohjaussolmu automatisoi ja hallinnoi. Esimerkiksi Windows-palvelin on hallinnoitava solmu. (Awati & Coutemanche 2023.)



Kuvio 2. Ansiblen arkkitehtuurin komponentit (Getting started with Ansible 2025)

Ansible käyttää OpenSSH-yhteyttä tiedonsiirtoon. Ansiblen pelikirjat (*playbooks*) ovat YAML-tiedostotyyppisiä pohjia, jotka toimivat pohjapiirustuksina automatisoitaville prosesseille. Pelikirjat mahdollistavat käyttäjän ohjelmoimaan toistuvia tehtäviä, joita voidaan suorittaa automaattisesti ilman koulutusta tai kehittyneiden ohjelmointikielien opiskelua. (Awati & Coutemanche 2023.) Pelikirjat koostuvat peleistä (*plays*), joilla vähimmillään määritetään millä laitteilla suoritetaan mitkäkin tehtävät (Glossary 2025).

Käytännössä Ansible siirtää koodia, ohjelmia tai IT-infrastruktuurin asennus moduulien avulla solmuihin. Nämä solmut voivat olla esimerkiksi fyysisiä palvelimia, virtuaalikoneita tai pilvipalveluinstansseja. Ansible yhdistää hallittuihin solmuihin ja lähettää moduulin kyseiseen solmuun, jossa moduuli suoritetaan SSH-yhteyden avulla. Kun moduuli on suoritettu, se poistetaan. (Awati & Coutemanche 2023.)

2.2.1 Ansiblen etuja

Manuaalinen konfigurointi voi olla aikaa kuluttavaa ja altis konfiguraatiovirheille. Ansiblea käyttämällä järjestelmäylläpitäjät voivat luoda infrastruktuuriin sopivia Ansible pelikirjoja, joiden avulla he voivat automatisoida nämä konfigurointiprosessit. Lisäksi Ansiblea käyttämällä järjestelmäylläpitäjät voivat ottaa sovelluksia käyttöön nopeammin ja luotettavammin. Käyttöönottoprosessit voidaan automatisoida kokonaisuudessa Ansiblen avulla käytännössä yhden komennon avulla (Uglov 2023.)

Ansiblen automaation käytön etuja ovat myös sen kyky sovittaa monimutkaisia työnkulkuja useisiin järjestelmiin yli alustojen. Se voidaan integroida erilaisiin työkaluihin tai palveluihin, joita voivat olla pilvipalvelualustat, konttien ajoympäristöt tai versionhallintajärjestelmät. Ansiblella toteuttavan automaation avulla organisaatioiden on helpompaa skaalata infrastruktuuria kysynnän mukaan, jolloin voidaan luoda infrastruktuuriin käyttöönottomalleja. Käyttöönottomallien avulla sovellusten tai palveluiden instansseja voidaan ottaa käyttöön ilman, että niitä tarvitsee tehdä erikseen manuaalisesti. (Uglov 2023.)

2.2.2 Ansiblen roolit

Vaikka Ansiblen pelikirjat ovat tehokas tapa luoda automatisoitavia tehtäviä, ne eivät välttämättä ole parhain lähestymistapa (What is an Ansible Role – and how is it used? 2024), koska Fahim (2025) toteaa pelikirjojen ongelman olevan yleisen rakenteen sekavuus, johon puolestaan Ansiblen roolit ovat ratkaisu. Roolit jakavat pelikirjat pienempiin osiin, jotka keskittyvät yhteen tehtävään kerrallaan. Roolien avulla automatisoitaviin tehtäviin saadaan järjestelmällinen rakenne. (Fahim 2025.)

Ansible Roles (2024) korostaa roolien olevan mainio ominaisuus Ansiblessa ajettavien skriptien parantamiseksi. Kun pelikirjoista luodaan järjestelmällisiä ja uudelleenkäytettäviä osia, monimutkaisista määrityksistä tulee roolien avulla yksinkertaisempia ja automaatiotehtävät ovat tehokkaampia ja helpommin hallittavia. Rooleja tulisi käyttää mm. projekteissa, joissa on suuria ja monimutkaisia pelikirjoja sekä kun tarvitsee automatisoida toistuvia tehtäviä. (Ansible Roles 2024.)

2.2.3 Jinja2 osana Ansiblea

Ansiblen ominaisuuksiin kuuluu sen integrointi Jinja2-ohjelmistoon. Jinja2 on Python-pohjainen templating engine, eli mallipohjamoottori. Jinjan avulla voidaan luoda dynaamisia malleja, joihin voidaan lisätä muuttujia ja muita kontekstuaalisia tietoja upotettavaksi suorituksen ajaksi. Dynaamiset mallit ovat erittäin käteviä, kun luodaan konfiguraatiotiedostoja tai skriptejä, joiden täytyy mukautua ympäristöön tai ehtoihin. Mallipohjien avulla Ansiblen pelikirjoista voidaan tehdä joustavampia ja uudelleenkäytettävämpiä, mikä vähentää mahdollisia päällekkäisyyksiä. (Ansible and Jinja2: Creating Dynamic Templates 2025.)

2.3 Perehdytys lokitukseen

Lokituksella tarkoitetaan lokitietojen tallennusta ja hyödyntämistä, joita kerätään selvittämään mitä, miksi ja milloin jotakin tapahtui. Lokitieto on tiettyyn kellonaikaan kirjattu tieto tapahtumasta ja sen aiheuttajasta. Lokitieto voi esimerkiksi olla muutos tietojärjestelmässä, sovelluksessa tai tietoverkossa. Hyvästä lokista löytyisi aikaleima, tapahtuma sekä tekijä, käyttöoikeus, tapahtuman lähde ja tapahtuman tila. (Näin kerää ja käytät lokitietoja 2023.)

Tietoturvan näkökulmasta lokitus on hyvin tärkeää. Lokit pitävät sisällä tietoa siitä mitä tapahtui mihinkin aikaan ja kenen toimesta, jonka takia ne ovat kriittinen lähde tunnistamaan ja tutkimaan tietoturvauhkia. Kaikkia lokeja voi pitää tietoturvalokeina, vaikka niitä ei olisi luotu ensisijaisesti tietoturvaa varten. (Why Logs Are the Foundation of Security n.d.)

Esimerkiksi Linux-palvelimien syslog-lokitiedostot tallentavat ensisijaisesti tietoa järjestelmän sisällä tapahtuvista prosesseista ja sovelluksista, mutta ne voivat silti olla tärkeä lähde tietoturvan kannalta. Useimmista käyttöjärjestelmistä löytyy todennuslokit, joiden avulla pystytään seuraamaan palvelimen kirjautumisia ja yhteydenmuodostamisia. Nämä voivat olla tarpeellisia mahdollisissa brute force -hyökkäyksissä, eli väsytyshyökkäyksissä, jolloin järjestelmään yritetään toistuvasti kirjautua eri salasanayhdistelmillä. (Why Logs Are the Foundation of Security n.d.)

2.3.1 Lokituksen lainsäädäntö

Lainsäädännössä on asetettu vaatimuksia liittyen lokien sisältöön, säilytysaikaan, lokeissa olevan tiedon eheyden varmistamiselle sekä lokin käyttötarkoitukselle. Lokitietojen keräämisen ja käsittelyn tulee perustua lainsäädäntöön. (Näin kerää ja käytät lokitietoja 2023.) Mikäli loki sisältää minkäänlaisia henkilötietoja, se luokitellaan henkilörekisteriksi. EU:n yleisen tietosuoja-asetuksen (2016) mukaisesti henkilötiedoiksi katsotaan kaikki tiedot, jotka liittyvät tunnistettuun tai tunnistettavissa olevaan luonnolliseen henkilöön. Tällaisia tietoja voivat olla nimi ja sähköpostiosoite sekä tunnistetiedot kuten käyttäjätunnus, sijaintitieto tai IP-osoite. (EU:n yleinen tietosuoja-asetus 2016/679.)

2.4 Verkkolaitteiden lokitus

Verkkolaitteiden lokitus on prosessi, jonka aikana verkkolaitteessa tapahtuneet tapahtumat dokumentoidaan. Näitä tapahtumia voivat esimerkiksi olla virheilmoitukset, varoitukset tai normaalia informaatiota. (What is network device logging and why is it necessary? n.d.) Informatiivisia tapahtumia voivat olla esimerkiksi ylläpidon tekemät konfiguraatiomuutokset.

Verkkolaitteissa, kuten kytkimissä on paljon lokitettavaa tietoa, joka on tarpeellista verkon tietoturvan valvonnassa. Verkkolaitteiden lokitiedoista ilmenee esimerkiksi verkkolaitteen kokoonpano ja toiminta. Näiden lokitietojen avulla voidaan tutkia verkon käytettävyyttä, luotettavuutta, suorituskykyä sekä havaitsemaan mahdollisia ongelmia. Näitä ongelmia voivat esimerkiksi olla virheelliset konfiguraatiot, toimintahäiriöt tai ruuhkat. Lisäksi verkkolaitteiden lokitus voi auttaa topologian, arkkitehtuurin ja suunnittelun tarkastamisessa. (Gupta n.d.)

2.5 Palvelimien lokitus

Myös palvelimet sisältävät lokitietoja. Palvelinlokit tyypillisesti sisältävät yksityiskohtaisia tietoja palvelimella tapahtuneista toiminnoista ja tapahtumista. Näitä tietoja ovat esimerkiksi ihan arkipäiväiset verkkosivulla vierailut, tiedoston lataukset tai tietokantaan pääseminen. Palvelimien lokitiedoista ilmenee päivämäärä ja kellonaika, käytetty resurssi sekä tapahtumaan liittyvät mahdolliset virheet. (What are Server Logs? n.d.)

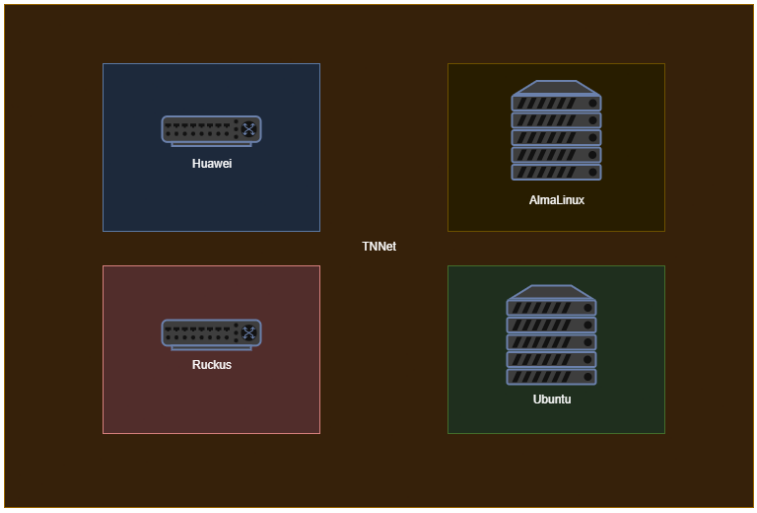
Palvelimien lokitus on tärkeää, koska niistä on apua erilaisissa tilanteissa. Jos palvelimella suoritettava palvelu tai prosessi ei toimi, voidaan palvelinlokeja käyttää vianmäärityksessä selvittämään, mitä tapahtui ja miksi. Palvelinlokeista myös selviää palvelimella tapahtuva epätavallinen toiminta, kuten epäonnistuneet sisäänkirjautumisyritykset tai tuntemattomat IP-osoitteet, jotka voivat viitata mahdollisiin tietoturvaloukkauksiin. (What are Server Logs? n.d).

3 Toteutusympäristö ja vaatimukset

3.1 Käytössä olevat kytkinmallit ja Linux-jakelut

Aluksi oli tärkeää kartoittaa millaisia kytkimiä ja palvelimia toimeksiantajalla oli käytössä. Käytössä olleet kytkimet olivat Ruckus ICX 8-sarjan sekä Huawein S5735-sarjan kytkinmalleja. Palvelimissa oli käytössä eri Linux-jakelut, jotka olivat AlmaLinux sekä Ubuntu. AlmaLinux on ilmainen ja avoimen lähdekoodin Linux-jakelu, joka on suunniteltu vaihtoehdoksi CentOS Linuxille. Se perustuu Red Hat Enterprise Linuxin (RHEL) lähdekoodiin. (What is AlmaLinux 2025.) Ubuntu sen sijaan on Sheldonin (2023) mukaan Debianiin perustuva avoimen lähdekoodin käyttöjärjestelmä, joka julkaistiin ensimmäisen kerran vuonna 2004. Ubuntu Serveriä, joka on Ubuntun palvelinversio, käytetään datakeskuksissa ja pilviympäristöissä. (Sheldon 2023.)

Ympäristössä oli siis käytössä kahden eri valmistajan kytkimiä ja kahta eri Linux-jakelua (ks. kuvio 3). Kytkinten kohdalla huomioitavaa on, että laitteiden käyttöjärjestelmät eroavat toisistaan, sillä Ruckusin kytkimissä on käytössä FastIron ja Huawein kytkimissä VRP. Myös palvelinten Linux-jakelut perustuvat eri jakeluihin, AlmaLinux RHEL-pohjaisiin ja Ubuntu Debian-pohjaisiin, joten myös näiden käyttöjärjestelmien kohdalla on pieniä eroavaisuuksia.

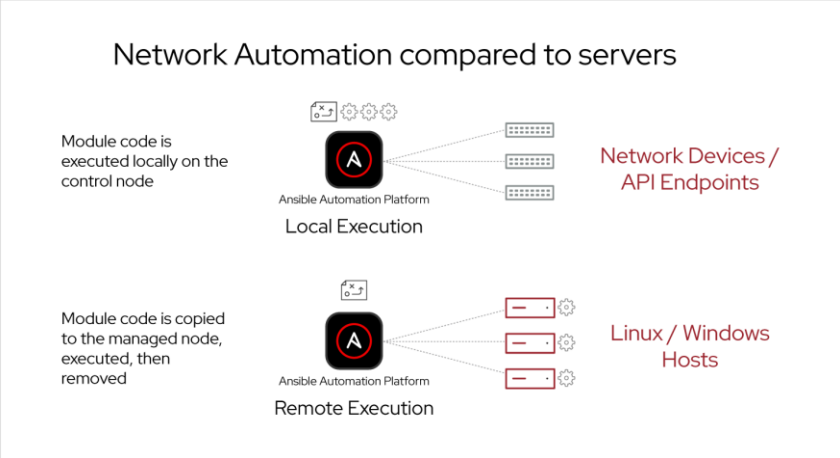


Kuvio 3. Ympäristö visuaalisesti havainnoituna

Miten tällaista ympäristöä pystytään hallitsemaan Ansiblen avulla? Ympäristössä olevat Linux-pohjaiset palvelimet eli AlmaLinux ja Ubuntu ovat hallittuja solmuja. Jotta Linuxeja voidaan automatisoida, Ansible muodostaa yhteyden näihin hallittuihin solmuihin ja lähettää niille ohjelmia, joita kutsutaan Ansible-moduuleiksi. Nämä moduulit suoritetaan oletuksena SSH:n kautta ja ne poistetaan niiden valmistuessa. (How Ansible Works n.d.)

Myös verkkolaitteita on mahdollista hallita, mutta niiden hallinta ja automatisointi eroaa Linux (ks. kuvio 4). Useimmat Ansiblen moduuleista toimivat hallituissa solmuissa, mutta verkkolaitteita käsittelevät moduulit eivät. Ansible on kirjoitettu Python-ohjelmointikieltä käyttäen, jonka takia se suoritetaan myös Pythonia käyttäen. Verkkolaitteista suurin osa ei pysty suorittamaan Pythonia, jonka takia Ansiblea käytettäessä verkkomoduulit suoritetaan ohjaussolmuissa. (How Network Automation is Different 2025.)

Verkkomoduulien suorittaminen ohjaussolmuissa hallittujen solmujen sijaan mahdollistaa tuen useille yhteysprotokollille. Näitä protokollia ovat XML over SSH, CLI over SSH, API over HTTPS. Valittava yhteysprotokolla riippuu alustasta ja moduulin tarkoituksesta. Riippuen verkkomoduulista, se voi tukea vain yhtä protokollaa tai useampaa eri vaihtoehtoa. Yhteysprotokolla voidaan määritellä ansible_connection muuttujaan. (How Network Automation is Different 2025.)



Kuvio 4. Eroavaisuudet verkkolaitteiden hallinnassa verrattuna Linux/Windows laitteisiin (How Ansible Works 2025)

Ansiblella on alustatuki monelle eri verkkolaitteille, mukaan lukien Huawein CloudEngine- ja Ruckusin ICX-kytkimille. Huawein CE- ja Ruckusin ICX-kytkimille tarkoitetut moduulit ovat osa community.network-kokoelmaa. Näiden moduulien avulla Huawein ja Ruckusin kytkimille voidaan suorittaa valittuja komentoja sekä tehdä muutoksia laitteiden konfiguraatioihin.

3.2 Laitesyntaksin tunnistaminen

Yksi toimeksiantajan toivomuksista olikin, että automaatiossa järjestelmä tunnistaisi oikean syntaksin, jolloin ylläpidon ei tarvitsisi erikseen määritellä ajettavia komentoja. Ympäristössä olevien laitteiden kartoittamisen jälkeen ilmeni, että laitteiden syntakseissa on pieniä eroja. Kytkimien kohdalla syntaksin eroavaisuudet liittyivät siihen, miten laitteille suoritetaan erinäköisiä konfiguraatiomuutoksia ja niiden tallentamiseen. Palvelimien välillä suurimmat eroavaisuudet liittyivät pakettienhallintaan.

Ansiblessa on muuttuja, nimeltään facts, jonka avulla pystytään keräämään erilaisia tietoja inventaarioon lisätystä laitteesta. Faktoja käyttämällä voidaan saada selville monenlaista tietoa laitteesta tai järjestelmästä, kuten sen IP-osoite, päivämäärä, tietoja sen arkkitehtuurista tai yksityiskohtaisia laitetietoja, kuten malli. (Discovering variables: facts and magic variables 2025.)

Kuinka tätä facts moduulia voidaan soveltaa laitesyntaksin tunnistamisessa? Ansiblen faktoja käyttämällä voidaan palvelimien kohdalla kerätä tieto käyttöjärjestelmästä, jonka avulla palvelimet voidaan kategorisoida käyttöjärjestelmän mukaisesti, jolloin Ansible osaa suorittaa pelikirjan, jossa on juuri tietylle Linux-tyypille sopivat komennot.

3.3 Lokitettavien tietojen määrittely

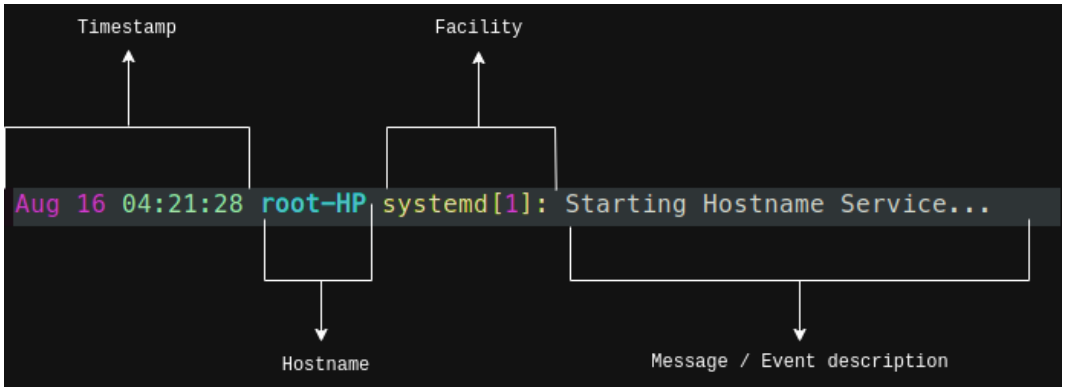
Sekä kytkimissä että palvelimissa on saatavilla paljon yksityiskohtaisia lokitietoja. Toimeksiantajan toivomus oli, että vain haluttuja tietoja lokitettaisiin. Jotta ei tarvitsisi lokittaa kaikkialta kaikkea, on perehdyttävä, että millaiset lokitiedot ovat miltäkin laitteilta tarpeellisia. Priority logs for SIEM ingestion: Practitioner guidance (2025) on useiden eri maiden kyberturvallisuusviranomaisten laatima ohjeistus. Nämä viranomaiset ovat laatineet kyberturvallisuuden toimijoille suosituksia lokitiedoista, jotka tulisi priorisoida tietoturvan hallintajärjestelmiä varten. Lokitietojen tallennuspäätöksien tulisi perustua organisaation toimintaympäristöön ja riskiprofiiliin.

Linux-palvelimista ohjeistetaan keräämään muun muassa kaikki todennustapahtumat, eli onnistuneet kirjautumisyritykset sekä -istunnot, mukaan lukien SSH:lla tapahtuvat, sekä oikeuksien korottamiseen liittyvät toiminnot, kuten sudo ja su. Myös muutokset todennusmekanismeissa- ja konfiguraatioissa, järjestelmäpalveluiden konfiguraatiomuutokset sekä käyttäjien, ryhmien ja salasanojen muutokset olisi syytä lokittaa. Näiden asioiden lisäksi ohjeistuksessa mainitaan esimerkiksi pakettienhallintaan liittyvien tapahtumien (asennus, poisto ja uudelleenkonfigurointi) ja verkkotapahtumien, kuten isäntänimen muuttuminen lokittaminen. Kytkimien kohdalla puolestaan ohjeistetaan keräämään lokitietoja seuraavanlaisista tapahtumista: kaikki järjestelmän muutostapahtumat sekä kirjautumiset. Lisäksi muutokset konfiguraatioissa sekä reititystauluissa tulisi lokittaa. (Priority logs for SIEM ingestion: Practitioner guidance 2025.)

Ennen kuin voidaan määritellä lokitettavat tiedot, on perehdyttävä näiden laitteiden lokitusjärjestelmiin. Ruckusin kytkimissä on lokitusta varten käytössä syslog (Show logging n.d.), mutta mikä oikeastaan on syslog ja miten se toimii? Kinaro (2023) kertoo syslogin olevan lokiviestien lähettämiseen tarkoitettu protokolla, jota käyttämällä lokiviestit voidaan lähettää eri järjestelmien, kuten palvelimien, reitittimien, kytkinten sekä palomuurien välillä tai vaihtoehtoisesti saman isäntälaitteen sisällä.

Syslogin arkkitehtuuri perustuu asiakas-palvelin arkkitehtuuriin. Arkkitehtuurissa asiakkaita voivat olla sovellukset, palvelut tai laitteet, kuten verkossa olevat reitittimet tai kytkimet. Arkkitehtuurin toinen osapuoli puolestaan on palvelin, jonka avulla lokeja varsinaisesti hallitaan. Syslogin ominaisuuksien, kuten monipuolisuuden ja yhteensopivuuden erilaisten laitteiden kanssa takia se on tärkeä työkalu turvallisuuden ylläpidossa ja poikkeamien hallinnassa. (Kinaro 2023.)

Miten syslogin viesti rakentuu ja millaiselta se näyttää? Kuviosta 5 huomataan, että syslogin viestien rakenne voidaan jakaa eri osiin. Syslogin viestit ovat jäsenneltyjä ja niistä ilmenee tietoja, kuten aikaleima, isäntänimi, tila ja sen taso sekä viestin sisältö. Tila viittaa ohjelmaan, prosessiin tai komponenttiin, joka viestin on luonut ja taso puolestaan ilmaisee vakavuuden. (Kinaro 2023.)



Kuvio 5. Syslog lokiviestin tyypillinen rakenne (Kinaro 2023)

Kuinka tarkalleen ottaen nämä tilat ja vakavuustasot toimivat? Syslogin tilat luokittelevat lokiviestit niin, että viestin luonut komponentti tai sovellus voidaan tunnistaa. Syslog määrittelee 24 vakituista tilaa (ks. taulukko 1), joita ilmaistaan numeraalisilla koodeilla 0–23. Jokainen tila koskee tiettyä aluetta järjestelmässä tai sovelluksessa. Tilojen lisäksi syslogille on määritelty kahdeksan eri tasoa, joita käytetään viestien luokitteluun vakavuuden perusteella (ks. taulukko 2). Tasot 5–7 ovat tyypillisesti sovelluksien käytössä, kun taas puolestaan tasot 0–4 ovat suureksi osin käyttöjärjestelmän käytössä. (Kinaro 2023.)

Taulukko 1. Syslog tilat (Kinaro 2023, muokattu)

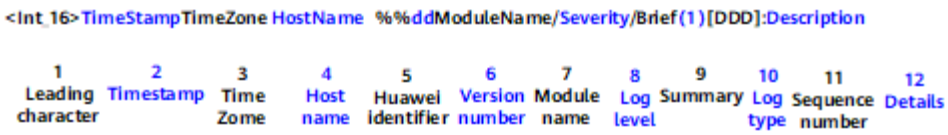
Koodi	Tila	Kuvaus
0	Kernel	Kernelin, eli ytimen viestit
1	User-level	Käyttäjätasossa toimivien prosessien tai sovelluksien viestit
2	Mail	Sähköpostipalvelimia ja prosessia koskevat viestit
3	Daemon	Daemon, eli taustapalveluiden viestit
4	Auth	Todennustapahtumat, kuten kirjautumiset
5	Syslog	Syslogin omaa toiminnan viestit
6	Lpr	Tulostuspalveluita koskevat viestit
7	News	Usenet-uutisien viestit
8	uucp	UUCP (Unix-to-Unix Copy) viestit
9	cron/clock	Ajastettujen tehtävien viestit
10	authpriv	Yksityisten auth-lokien viestit
11	FTP	FTP (File Transfer Protocol) viestit
12	NTP	NTP (Network Time Protocol) viestit
13	security	Turvallisuuteen liittyvien tapahtumien viestit
14	console	Järjestelmän konsolissa näkyvät viestit
15	solaris-cron	Solaris cron -ohjelmaa koskevat viestit
16–23	local0-local7	Käyttäjän itse määrittämiä tiloja sovelluksille tai palveluille

Taulukko 2. Syslog tasot (Kinaro 2023, muokattu)

Taso	Vakavuus	Kuvaus
0	Emergency	Järjestelmä käyttökelvoton
1	Alert	Tarvitaan välittömiä toimia
2	Critical	Kriittiset olosuhteet
3	Error	Virhetilanne
4	Warning	Varoittavat olosuhteet
5	Notice	Normaalit mutta merkittävät olosuhteet
6	Informational	Informatiivinen viesti
7	Debug	Vianetsintätason viesti

Huawei puolestaan käyttää kytkimissään Information Centeriä, mikä Overview of Information Center (2021) mukaan toimii eräänlaisena tietokeskuksena laitteelle. Huawein laitteiden lokit lähetetään yhdessä ansojen ja vianetsintäviestien kanssa Information Centeriin, minkä kautta niitä voidaan hallita yhteisesti ja tulostaa joustavasti. Tiedot tallennetaan reaaliajassa, mikä auttaa ylläpitäjiä seuraamaan verkon toimintaa sekä analysoimaan mahdollisia vikoja. (Overview of Information Center 2021.)

Keskus luokittelee ja suodattaa siihen saapuneet tiedot tyypin ja vakavuuden perusteella (Overview of Information Center 2021). Huawein Information Centerin voi siis nähdä pitkälti samanlaisena kuin syslogin, koska laitteissa tapahtuvat tapahtumat luokitellaan vakavuustasojen mukaan, jotka ovat käytännössä täysin samat kuin syslogissa. Information Centerin lokiviestin rakenteessa on kuitenkin havaittavissa enemmän yksityiskohtia (ks. kuvio 6). Myös näitä Information Centerin keräämiä tietoja voidaan kuitenkin lähettää erilliselle palvelimelle.



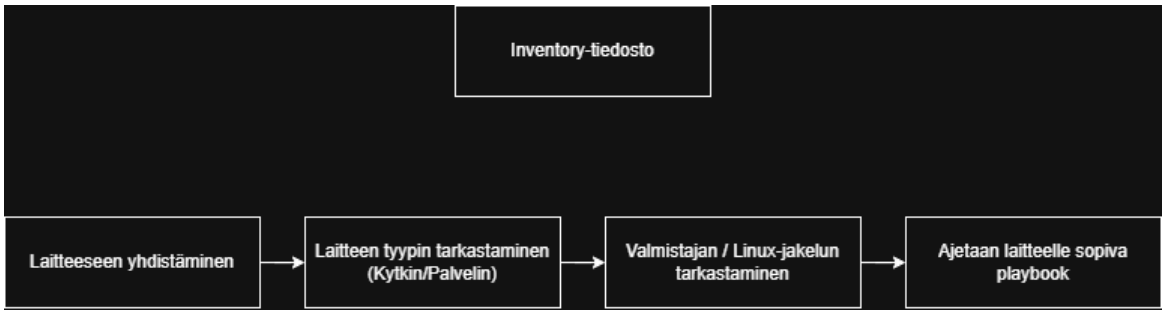
Kuvio 6. Information Centerin lokiviestin rakenne (Understanding the Information Center 2021)

Entä miten palvelimien lokitus toimii? Ubuntussa on versiosta 20.04 LTS eteenpäin ollut oletusarvoisena lokitusjärjestelmänä rsyslog. Rsyslog on paranneltu versio syslogista, jonka ominaisuuksia ovat monisäikeinen lokien käsittely, kriteereihin perustuva lokien suodattaminen, tuki erilaisille lokimuodoille, kuten JSON ja CSV, turvallinen lokien siirtäminen TCP-, UDP- ja TLS-protokollien kautta, lokien välitys etäpalvelimelle sekä lokien kirjoittaminen tietokantaan. (Whittaker 2025.) Myös AlmaLinuxissa on versiosta 9 eteenpäin ollut käytössä rsyslog oletusarvoisena log daemonina (Codex 2024).

4 Ratkaisun suunnittelu

4.1 Ansible-rakenteen suunnittelu

Seuraavaksi täytyi suunnitella, millainen Ansiblen rakenteen kannattaisi olla. Rakennetta suunnitellessa täytyi ottaa huomioon toimeksiantajan tavoite järjestelmästä, joka tunnistaisi laitekohtaisen syntaksin. Tähän päästäisiin ehtolauseiden kautta vaiheittain etenevää (ks. kuvio 7) rakennetta käyttämällä. Ehtolausekkeita käyttämällä pelikirjat voivat suorittaa tietyn tehtävän perustuen muuttujan tai edellisen tehtävän palautusarvoon. Muuttujalle voidaan myös määrittää tietty arvo, mikäli tietyt kriteerit täyttyvät. (Conditionals 2025.) Lisäksi arkkitehtuurissa hyödynnettäisiin Ansiblen rooleja sekä Jinjan mallipohjia.



Kuvio 7. Ansiblen vaiheittain etenevä rakenne

Ensimmäisenä ennen kuin haluttuisiin laitteisiin voidaan yhdistää, Ansiblen inventaarioon tarvitsee määritellä kaikkien konfiguroitavien laitteiden IP-osoitteet. Koska ympäristössä voi olla kymmeniä kytkimiä, jotka kuuluvat samaan IP-osoiteavaruuteen, inventaarioon voitaisiin tarvittaessa määritellä laitteiden IP-osoitteiden alue sen sijaan, että laitteiden IP-osoitteet kirjoitettaisiin yksitellen. Jos on tiedossa, millaista laitetta tietty IP-osoite koskee, tällöin inventaarioon määriteltäisiin halutessaan myös laitteiden nimet tai mahdolliset ryhmitykset, kuten *switches* tai *servers*.

Seuraavaksi selvitettäisiin inventaarioon lisättyjen laitteiden käyttöjärjestelmät/valmistajat. Palvelimien kohdalla tarkastuksessa kerättäisiin tieto palvelimen käyttöjärjestelmästä Ansiblen faktojen avulla. Kerättyjen faktojen pohjalta tarkastetaan ehtolausekkeen avulla, että onko yhdistettävässä laitteessa käytössä AlmaLinux vai Ubuntu. Kytkimiltä puolestaan ei kerätä faktatietoja, mutta myös

niiden kohdalla ehtolausekkeiden avulla pyritään selvittämään, onko kyseessä Ruckusin vai Huawein kytkin.

Kun kaikki laitekohtaiset tarkastukset on tehty ja Ansiblella on tiedossa, että millaiseen laitteeseen se on tekemässä mahdollisia muutoksia, viimeisenä vaiheena ajettaisiin varsinainen konfiguraatiomuutokset suorittava pelikirjan tehtävä. Tässä vaiheessa käytettäisiin apuna Ansiblen rooleja, mutta varsinaiset konfiguraatioon liittyvät komennot olisi määritelty Jinja-mallipohjiin, jotka Ansible sitten suorittaisi.

4.2 Lokitusympäristön suunnittelu

Lokitusympäristön suunnittelussa täytyi ottaa huomioon toimeksiantajan toive modulaarisesti tapahtuvasta lokituksesta. Käytännössä modulaarisuus tarkoittaisi, että vain halutut tiedot lokitetaan verkko- ja palvelinlaitteista. Tämä lokitusympäristö voidaan jakaa kahteen osaan, eli lokitusvalintoihin Ansiblessa sekä käsittelyyn niitä vastaanottavassa päässä. Ensimmäinen toteutuu pelikirjoissa, jossa voidaan valita muuttujan arvoon perustuen, että minkä tason lokeja lähetetään.

Yksi olennainen osa lokitusympäristössä on työkalu, jonka avulla näitä lokeja voidaan hallita ja tarkastella. Tätä varten vertailtiin erilaisia työkaluja. Vertailussa keskityttiin työkaluihin, jotka olivat avoimen lähdekoodin työkaluja ja niistä oli tarjolla versio, joka oli lisensoitu vapaaseen käyttöön. Erityisesti kiinnitettiin huomiota työkalujen ominaisuuksiin, mutta myös mahdolliseen resurssienkulutukseen. Eri lähteiden pohjalta vertailukohteiksi nostettiin Elastic Stack, Fluentd, Grafana Loki, Graylog ja OpenSearch. Näistä etenkin Elastic Stack, Fluentd ja Graylog korostuivat eri lähteissä.

Elastic Stack. Koostuu kolmesta osasta: Elasticsearchista, Logstashista sekä Kibanasta. Elasticsearch käyttää hajautettua ja reaaliaikaista haku- ja analysointimoottoria säilöntään ja lokien indeksointiin. Logstash puolestaan vastaa lokien keräämisestä, jäsentämisestä ja siirtämisestä Elasticsearchiin. Kibana tarjoaa mahdollisuuden visuaalisointiin sekä raportointiin. (Rajender 2024.) Tietyt ominaisuudet, kuten hälytykset ja roolipohjainen pääsynhallinta puuttuvat ilmaiseksi saatavilla olevasta versiosta (Top Log Management Tools 2024).

Fluentd. Bocetta (2019) kertoo Fluentdin olevan vahva vaihtoehto tietojen keräämiseen, mutta ei sisällä täydellistä frontend-käyttöliittymää. Mainitsee työkalun suurimman edun olevan sen yhteensopivuus muiden yleisien työkalujen kanssa. (Bocetta 2019.) Lisäksi Top Log Management Tools (2024) listaa Fluentdin vahvuuksiin mm. resurssien tehokkaan käyttämisen sekä laajan ympäristön laajennuksille.

Grafana Loki. Rajender (2024) kertoo Grafana Lokin olevan lokien keräysjärjestelmä, joka on suunniteltu tallentamaan ja hakemaan lokitietoja kaikista sovelluksista ja infrastruktuureista. Tietoja voi suodattaa aikaleiman ja tunnisteen mukaisesti. Visuaalinen käyttöliittymä Grafanassa. Tiedot tallennetaan pitkäaikaiseen tallennustilaan vain kerran, eli ei tee ns. raskasta indeksointia, joten kuormittaa vähemmän resursseja. (Top 6 Log Management Tools and How to Choose n.d.)

Graylog. Suunniteltu keskitetyksi lokienhallintajärjestelmäksi, jonka avulla voidaan selata ja analysoida tietoja nopeasti. Bocettan (2019) mukaan Graylog on helposti skaalattavissa ja pystyy tasapainottamaan kuormitusta backend-palvelimella sekä käsittelemään päivässä teratavun verran lokitietoja. Top Log Management Tools (2024) kuitenkin mainitsee Graylogin heikkouksina rajalliset mahdollisuudet visualisointiin.

OpenSearch. Perustuu Elasticsearchiin. Bandurchin (2025) kertoo OpenSearchin keskeisimpiin ominaisuuksiin kuuluvan edistyneen hakemisen, roolipohjaisen pääsyyn, kustannuksien optimoinnin lokitustallennustilan porrastuksella sekä hälytyslaajennukset, jotka mahdollistavat ilmoitukset tiimille esim. Slackin kautta.

Graylog ja Grafana Loki otettiin tarkempaan vertailuun ja vertailussa huomioitiin, millaiseen ympäristöön työkalujen käyttäminen sopii parhaiten. Sinhan (2025) mukaan Graylog on ihanteellinen vaihtoehto, kun organisaatiolla on tarvetta syvällisimpiin hakuihin eri lähteistä, kuten syslogeista, sovelluslokeista tai turvallisuuslokeista. Grafana Loki puolestaan sopii paremmin pilvipohjaiseen ympäristöön, missä lokeja syntyy suuria määriä, mutta niiden elinkaari on lyhyt. (Sinha 2025.)

Udasin (2025) mukaan Graylogin valinta on perusteltua, kun tarvitaan tehokkaita ja joustavia hakutoimintoja. Myös tarve vaatimustenmukaisuudelle ja auditoinnille on syy valita Graylog, jonka

tarjoamat raportointiominaisuudet auttavat täyttämään sääntelyvaatimukset. (Udasi 2025.) Grafana Loki kannattaa puolestaan valita, mikäli resurssienkulutukseen liittyvien kustannuksien säästäminen on tärkeää, jolloin Grafana Lokin tehokas arkkitehtuuri voi vähentää merkittävästi kustannuksia. Myös Sinha (2025) painottaa Lokin olevan optimoitu pilvipohjaisille ja konttiteknologiaan perustuville ympäristöille, jolloin se on kustannustehokas ratkaisu.

Vertailun perusteella tähän opinnäytetyön ympäristöön sopivaksi lokienhallintatyökaluksi valikoitui Graylog. Vaikka Elastic Stack sekä OpenSearch tarjoavat paremmat toiminnot lokitetojen analytiikka varten, niiden käyttö vaatii useita raskaita komponentteja, mikä nostaa resurssienkulutusta. Resurssienkulutuksen näkökulmasta Grafana Loki tai Fluentd olisivat olleet parhaimmat vaihtoehdot, mutta Fluentd ei tarjoa kaikkia tarpeellisia ominaisuuksia ja Grafana Loki sopii paremmin toisenlaiseen ympäristöön.

Graylogin käyttäminen puolestaan tarjoaa ratkaisun, joka vastaa parhaiten työn tavoitteisiin. Se tarvitsee vain Elastic Stackin kolmeen komponenttiin verrattuna vain yhden (Elasticsearch tai OpenSearch), eli resurssienkulutus on oletettavasti matalampi kuin Elastic Stackissa. Lisäksi Graylog sisältää visuaalisen käyttöliittymän, on tarvittaessa helposti skaalattavissa ja se voi vastaanottaa ympäristön laitteiden syslogeja suoraan ilman erillisiä agentteja. On kuitenkin huomioitava, että suuremmassa ympäristössä olisi perusteltua valita Elastic Stack sen tarjoamien ominaisuuksien vuoksi tai vaihtoehtoisesti valita OpenSearch sen avoimen Apache 2.0 lisenssin takia.

Toteutus

4.3 Ansible-rakenteen toteutus

Tässä työssä Ansiblen hallintakoneena toimi Ubuntu, johon pääsi yhdistämään SSH-yhteyden avulla. Koska toteutusvaihe sisälsi Ansiblen pelikirjojen ohjelmointia, työssä hyödynnettiin ohjelmointiin tarkoitettua tekstieditoria Visual Studio Codea sekä siinä käytettävää Remote-SSH-lisäosaa. Remote-SSH-lisäosan avulla on mahdollista avata etäkansio etätietokoneella, virtuaalikoneella tai kontilla, kunhan SSH-palvelin on käynnissä (Remote Development using SSH 2025).

4.3.1 Ansiblen asentaminen

Kun Visual Studio Codella saatiin yhteys Ansiblen hallintakoneeseen, seuraavaksi asennettiin Ansible. Ansiblen asennuksessa on eroavaisuuksia riippuen hallintakoneen käyttöjärjestelmästä. Opinnäytetyön tapauksessa hallintakoneessa oli käyttöjärjestelmänä Ubuntu. Ubuntulle Ansible-paketit asennetaan Personal Package Archiven (PPA) kautta (Installing Ansible on specific operating systems 2025). Ubuntun Ansible PPA-asennuksen komennot (ks. kuvio 8) ajettiin hallintakoneella ja Ansible saatiin asennettua onnistuneesti.

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible
```

Kuvio 8. Ansiblen asentaminen Ubuntulle (Installing Ansible on specific operating systems 2025)

4.3.2 Ansible-inventaarion määrittäminen

Varsinainen rakenteen toteutus aloitettiin suunnitelman mukaisesti luomalla inventaario, johon määriteltiin yhdistettävien laitteiden IP-osoitteet sekä muita mahdollisia yhdistämiseen vaikuttavia muuttujia, kuten SSH-yhteyden portti. Ansible käyttää oletusarvoisesti inventaariona /etc/ansible/hosts-tiedostoa, mutta inventaario voidaan kuitenkin luoda myös muuhun sijaintiin. Inventaario voidaan toteuttaa monessa eri formaatissa, mutta yleisemmin käytetään sisäänrakennettuja INI- tai YAML-formaatteja. (How to build your inventory 2025.) Tätä työtä varten inventaario luotiin .ini-tiedostona hakemistoon **/home/ubuntu/oppari-ansible/**.

Palvelimiin liittyvät tiedot määriteltiin ensimmäisenä inventaarioon. Tässä työssä oli vain kaksi aliverkkoon kuuluvaa laitetta, jonka vuoksi laitteiden IP-osoitteet määriteltiin yksitellen omiin ryhmiinsä (ks. kuvio 9). Molemmat palvelimet olisi voinut vaihtoehtoisesti määritellä yhteen yksittäiseen ryhmään (ks. kuvio 10). Ensimmäiseen ratkaisuun päädyttiin, koska molemmilla palvelimilla oli eroavaisuuksia yhdistämiseen vaikuttavissa muuttujissa, jotka on helpompi määritellä ryhmämuuttujien koskiessa tiettyä käyttöjärjestelmää tai laitemallia.

```
2
3   [ubuntu] # Ubuntu 20.04
4   62.204.28.141
5
6   [almalinux] #AlmaLinux 9
7   62.204.28.186
8
```

Kuvio 9. Palvelimien IP-osoitteiden määrittely inventaariossa

```
[Palvelimet]
# Ubuntu 20.04
62.204.28.141

# AlmaLinux 9
62.204.28.186
```

Kuvio 10. Vaihtoehtoinen tapa ryhmittää palvelimet

Palvelimien jälkeen inventaarioon lisättiin kytkimet (ks. kuvio 11). Kytkimien kohdalla oli käytössä sama IP-osoite SSH-yhteyttä varten, mutta portit erottivat laitteet toisistaan. Huawein kytkin käytti porttia 9022 ja Ruckusin porttia 9023. SSH-yhteys käyttää oletuksena porttia 22 yhdistämiseen, jonka vuoksi kytkimien portit täytyi erikseen määritellä inventaarioon. Tähän hyödynnettiin Ansiblen **ansible_port**-muuttujaa. Muuttujan avulla saadaan määriteltyä portti SSH-yhteyttä varten, mikäli se on muu kuin oletusarvoinen portti 22 (How to build your inventory 2025). Sekä palvelimiin että kytkimiin kuuluvat laiteryhmät määriteltiin vielä ns. aliryhmiksi osaksi pääryhmiä *palvelimet* sekä *kytkimet* (ks. kuvio 12).

```
17  [huawei] # Huawei S5375
18  huawei1 ansible_host=89.236.118.86 ansible_port=9022
19
20  [ruckus] # Ruckus ICX
21  ruckus1 ansible_host=89.236.118.86 ansible_port=9023
22
```

Kuvio 11. Kytkimien tiedot inventaariossa

```
inventory.ini
1  #Palvelimiin kuuluvat ryhmät
2  [palvelimet:children]
3  almalinux
4  ubuntu
5
6  #Kytkimiin kuuluvat ryhmät
7  [kytkimet:children]
8  huawei
9  ruckus
```

Kuvio 12. Palvelimet ja kytkimet ryhmitykset

Kytkimien kohdalla on myös muita muuttujia, jotka täytyy määritellä laitteiden automatisointia varten. Kun Ansiblea käytetään verkkolaitteiden automatisoinnissa, yhdistämiseen käytettävä protokolla täytyy erikseen määritellä **ansible_connection**-muuttujalla. Yhteysprotokollan lisäksi täytyy määritellä **ansible_network_os**, joka määrittelee tarkasti verkkolaitteessa käytössä olevan käyttöjärjestelmän. (How Network Automation is Different 2025.) Kyseinen muuttuja voidaan määritellä heti inventaariossa ryhmä- tai laitekohtaisiin muuttujiin, mutta myös erikseen pelikirjoissa. Nämä muuttujat määriteltiin kytkimien ryhmäkohtaisiin muuttujiin (ks. kuvio 13).

```
! huawei.yml ×
group_vars > ! huawei.yml
1  ansible_connection: ansible.netcommon.network_cli
2  ansible_network_os: community.network.ce
```

Kuvio 13. Huawein yhteysmuuttujat määriteltynä ryhmäkohtaisiin muuttujiin

Kaikkia muuttujia ei kuitenkaan kannata kirjoittaa suoraan inventaarioon. Salasanat ja mahdollisesti myös käyttäjätunnukset ovat tietoja, joita ei haluta tiedostoihin sellaisinaan näkyviin. Tästä syystä kytkimiin ja palvelimeen yhdistämiseksi tarvittavat käyttäjätunnukset ja salasanat suojattiin Ansible Vaultin avulla. Ellingwood (2022) kertoo Ansible Vaultin olevan ominaisuus, jonka avulla salattuja tietoja voidaan sisällyttää. Ansible Vaultin avulla voidaan salata suojeltavaa tietoa. Salauksessa käytetään AES256 salausalgoritmia, jonka avain on käyttäjän antama salasana. (Ellingwood 2022.)

Sekä palvelimien ja kytkimien käyttäjätunnukset sekä salasanat määriteltiin erillisiin vars-tiedostoihin group_vars-hakemistoon. Jokaiselle laiteryhmälle luotiin oma tiedosto ryhmämuuttujia varten,

kuten esimerkiksi huawei.yml. Kun kirjautumistietojen toimivuus oli testattu, tiedot salattiin Ansible Vaultin avulla (ks. kuvio 14). Onnistuneen salauksen jälkeen muuttujat olivat näkyvissä vain salatussa muodossa (ks. kuvio 15).

```
ubuntu@ansible:~/oppari-ansible$ ansible-vault encrypt_string 'administrator' --name 'ansible_user'
New Vault password:
Confirm New Vault password:
Encryption successful
ansible_user: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          63313333326639396361313532306432326666356362336537613466623262373135363732333335
          63633766356330323931633262356533333133326134333310a373262343030636562636663386538
          38633035623131653938616239383032316433613566636338386532383336336263613562616239
          38363435363864376 40a663838316263643866333963666566383833303934626532633531313139
          6465
```

Kuvio 14. Käyttäjätunnuksen salaaminen ansible_user-muuttujaa varten

```
group_vars > ! huawei.yml
1   ansible_user: !vault |
2           $ANSIBLE_VAULT;1.1;AES256
3           63313333326639396361313532306432326666356362336537613466623262373135363732333335
4           63633766356330323931633262356533333133326134333310a373262343030636562636663386538
5           38633035623131653938616239383032316433613566636338386532383336336263613562616239
6           38363435363864376 40a663838316263643866333963666566383833303934626532633531313139
7           6465
8
9   ansible_password: !vault |
10          $ANSIBLE_VAULT;1.1;AES256
11          66646130356430623836626132653038626236333335373534623730353333616663643362633039
12          39303065653937313133663438613436366162306462393 10a3665643132383136646235633034 37
13          34646435613133373330323664653563356362303261633561323437613339353466336437376637
14          34376330326639393 60a623866306266643036643834383066323064376563346338363565383330
15          33643838316665363964633836303930643664353262336231646266623361383736
16
```

Kuvio 15. Huawein käyttäjätunnus ja salasana salatussa muodossa

4.3.3 Kytkimien ja palvelimien käyttöjärjestelmien tarkastaminen

Inventaarion oltua valmis, seuraavana määriteltiin tarkka laitetyyppi, jotta Ansible tietää minkälaisen konfiguraatiotiedoston sen täytyy myöhäisemmässä vaiheessa suorittaa. Ansiblelle luotiin pelikirja nimeltä **infraKonfiguraatio.yml**. Pelikirjaan luotiin ensimmäinen peli sekä tehtävä, jossa tarkastettiin palvelimen käyttöjärjestelmä (ks. kuvio 16). Tätä vaihetta varten palvelimista faktat kerättiin **gathering_facts**-moduulin avulla, jonka arvo määriteltiin pelissä heti hosts-muuttujan jäl-

keen. Varsinainen tunnistus tapahtui tehtävässä silmukkaa ja ehtolauseketta käyttämällä, jossa device_type-muuttujan arvoksi määriteltiin ”almalinux” tai ”ubuntu” riippuen siitä, mitä arvoa faktojen avulla palvelimilta kerätty ansible_distribution vastasi.

```yaml
- name: Laitetyypin tarkistaminen
  hosts: palvelimet
  gather_facts: yes

  tasks:

    - name: Tunnista palvelimen Linux-jakelu
      set_fact:
        device_type: "{{ item }}"
      loop:
        - "{{ 'almalinux' if ansible_distribution == 'AlmaLinux' else '' }}"
        - "{{ 'ubuntu' if ansible_distribution == 'Ubuntu' else '' }}"
      when: item != ''
```

Kuvio 16. Palvelimien Linux-jakelun tunnistaminen

Palvelimien jälkeen selvitettiin kytkimien valmistaja. Ansiblen gather_facts-moduuli ei suoraan toimi kytkimien kanssa, koska verkkolaitteet eivät voi suorittaa Pythonia, joten kytkimistä ei voinut kerätä aivan samalla tavalla tietoa laitteen käyttöjärjestelmästä. Aiemmin inventaarioon oli määritelty molemmille kytkimille niiden yhteysprotokollat ansible_connection-muuttujalla sekä käyttöjärjestelmä ansible_network_os-muuttujalla.

Tässä opinnäytetyön toteutuksessa kuitenkin sovellettiin lisäksi tilannetta, missä yhteysmuuttujia ei ollut välttämättä erikseen määritelty. Kytkimille luotiin samaan pelikirjaan uusi peli, jossa sillä ensimmäisenä tehtävänä oli tallentaa kytkimen käyttöjärjestelmä (ks. kuvio 17). Tässä tehtävässä käytettiin lähestulkoon samanlaista rakennetta, jossa tarkastettiin silmukan ja ehtolausekkeen avulla, että vastasiko ansible_network_os-muuttujan arvo joko Huawein community.network.ce- tai Ruckusin community.network.icx-moduuleja, arvon perusteella device_type-muuttujaan tallennettiin arvoksi ”huawei” tai ”ruckus”. Mikäli ansible_network_os ei ollut määritelty, tällöin muuttujan arvo oli tyhjä ja mitään ei tallenneta.

```yaml
- name: Tarkastetaan kytkimen valmistaja
  hosts: kytkimet
  gather_facts: no
  connection: local

  tasks:

    # Tallennetaan kytkimen valmistaja device_type muuttujaan, mikäli ansible_network_os on ennalta määritelty
    - name: Tallenna kytkimen valmistaja
      set_fact:
        device_type: "{{item}}"
      loop:
        - "{{ 'huawei' if (ansible_network_os | default('')) == 'community.network.ce' else '' }}"
        - "{{ 'ruckus' if (ansible_network_os | default('')) == 'community.network.icx' else '' }}"
      when: item != ''
```

Kuvio 17. Tallennetaan kytkinvalmistaja, mikäli se on ennalta määritetty

Jos ansible_network_os ei ollut määritelty, se täytyi selvittää toisella tavalla. Jotta kytkinvalmistaja voitiin saada selville, se selvittiin Netmikon avulla. Netmiko on avoimen lähdekoodin Python kirjasto, jota voidaan käyttää verkkolaitteiden hallinnointiin (Byers 2021). Tätä tilannetta varten Ansiblen projektihakemistoon luotiin Python-skripti, nimeltä **kytkinTunnistus.py** (ks. kuvio 18). Kyseinen skripti lisättiin osaksi aiemmin luotua kytkimien peliä infraKonfiguraatio.yml-pelikirjassa, jossa se suoritettaisiin osassa ”Tunnista kytkimen valmistaja” -tehtävää (ks. kuvio 19).

Tehtävässä skriptille annettiin yhdistämiseen vaadittavat kytkimen IP-osoite, käyttäjänimi, salasana ja portti. Tämän jälkeen skripti otti Netmikon avulla SSH-yhteyden kytkimelle, jossa laitteella suoritettiin yksinkertainen komento, minkä avulla saatiin näytettyä laitteen versiotiedot. Seuraavaksi ehtolausekkeen avulla tarkastettiin versiokomennon tuloste ja löytyykö siitä käyttöjärjestelmään tai valmistajaan liittyvä tieto. Jos ehtolauseke täyttyi, silloin skripti palautti merkkijonon "ruckus" tai "huawei".

```python
import sys
from netmiko import ConnectHandler

def kytkinTunnistus():

    # Määritellään kokoelmaan mahdolliset kytkimen laitetyypit
    # https://github.com/ktbyers/netmiko/blob/develop/PLATFORMS.md#supported-ssh-device_type-values
    device_types = [
        ("ruckus_fastiron", "show version"),
        ("huawei", "display version")
    ]

    for device_type, version_cmd in device_types:
        # Määritellään yhdistämiseen tarvittavat parametrit
        device = {
            "device_type": device_type,
            "host": sys.argv[1],
            "username": sys.argv[2],
            "password": sys.argv[3],
            "port": int(sys.argv[4]) if len(sys.argv) == 5 else 22,
        }

        # Yritetään yhdistää kytkimeen ja suorittaa versiokomento, jonka jälkeen suljetaan yhteys
        try:
            conn = ConnectHandler(**device)
            output = conn.send_command(version_cmd)
            conn.disconnect()

            # Tunnistetaan onko versiokomennon tulosteessa kytkimen valmistajaan tai käyttöjärjestelmään viittauksia
            if "Ruckus" in output or "FastIron" in output:
                return "ruckus"
            elif "Huawei " in output or "VRP" in output:
                return "huawei"
        except Exception:
            pass

    return "Kytkintyyppiä ei tunnistettu"

if len(sys.argv) not in [4, 5]:
    sys.exit(1)

kytkinTyyppi = kytkinTunnistus()
print(kytkinTyyppi)
```

Kuvio 18. Netmiko Python-skripti kytkimen valmistajan tunnistamiseen

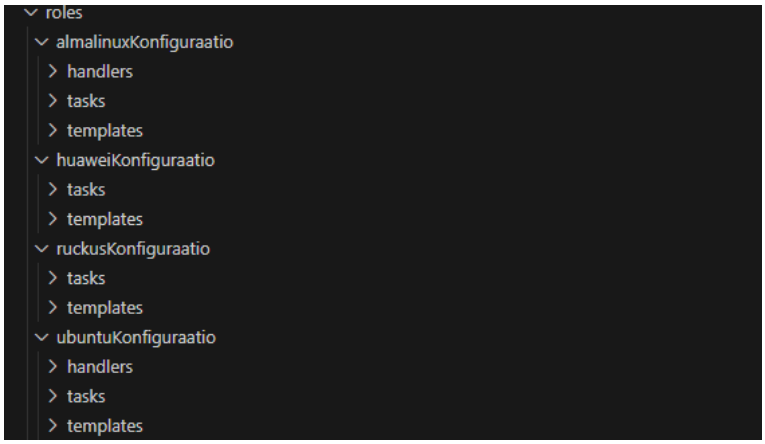
Skriptin suorituksen jälkeen Ansible jatkoi suoraan seuraavaan tehtävään, joka oli skriptin palauttaman arvon tallentaminen device_type-muuttujaan. Laitetyypin lisäksi tehtävässä asetettiin yhdistämismuuttujat. Myös tässä tehtävässä oli ehtona, että ansible_network_os ei ollut alun perinkään määritelty. Mikäli kyseinen muuttuja oli määritetty, tällöin Ansible ohitti sekä tunnistus- että määrittelytehtävän.

```yaml
    # Tunnistetaan kytkimen valmistaja Netmiko skriptillä, mikäli ansible_network_os ei ole määritelty
    - name: Tunnista kytkimen valmistaja
      command: >
        ~/netmiko-env/bin/python kytkinTunnistus.py {{ ansible_host }} {{ ansible_user }} {{ ansible_password }} {{ ansible_port }}
      register: kytkinvalmistaja
      delegate_to: localhost
      when: ansible_network_os is undefined

    # Tallennetaan Netmiko-skriptistä saatu kytkinvalmistaja device_type muuttujaan sekä määritellään verkkolaitteen yhteysmuuttujat
    - name: Määrittele kytkinvalmistaja laitetyyppiin  ja aseta network_os
      set_fact:
        device_type: "{{ kytkinvalmistaja.stdout }}"
        ansible_connection: "{{ 'network_cli' if kytkinvalmistaja.stdout == 'ruckus' or 'huawei' else 'ssh' }}"
        ansible_network_os: "{{ 'community.network.icx' if kytkinvalmistaja.stdout == 'ruckus' else 'community.network.ce' }}"
      when: ansible_network_os is undefined
```

Kuvio 19. Kytkimen tunnistaminen ja laitetyypin sekä yhteysmuuttujien määrittäminen

Kun palvelimen Linux-jakelu tai kytkimen valmistaja oli selvillä, seuraava vaihe rakenteessa oli, että Ansible ajaa laitteelle oikeanlaisen Jinja-mallipohjan Ansiblen rooleja hyödyntäen. Ansiblen projektihakemistoon luotiin jokaiselle laitetyypille oma alihakemisto roles-hakemistoon (ks. kuvio 20), jossa hakemisto nimi oli muotoiltu siten, että nimessä yhdistyvät laitetyyppi ja sana ”Konfiguraatio”, eli esimerkiksi ruckusKonfiguraatio. Lisäksi jokaiselle roolille luotiin hakemistot tehtäviä sekä Jinja-mallipohjia varten. Palvelimien kohdalla luotiin käsittelijöitä varten handlers-hakemisto.



Kuvio 20. Roolien hakemistorakenne

Kun jokaisen laitetyypin roolin hakemistorakenne oli oikeanlainen, infraKonfiguraatio.yml-pelikirjaan lisättiin uusi peli. Tässä pelissä Ansible konfiguroi laitteiden syslogien lähettämisen laitekohtaisia rooleja käyttäen (ks. kuvio 21). Tässä osiossa hosts-muuttujan arvoksi määriteltiin kaikki laitteet, koska suoritettavat tehtävät olivat kaikille laitteille samanlainen. Peliin luotiin vain yksi tehtävä laitekohtaisen roolin valitsemiseen. Tehtävässä otettiin käyttöön include_role-moduuli, jonka avulla pelikirjan voi ohjata käyttämään Ansiblen rooleja. Moduulille annettiin muuttuja name ja sen arvoksi {device_type}Konfiguraatio, jossa {{device_type}} arvo määräytyi pelikirjan aiemmissa tehtävissä tallennetun faktan perusteella. Tehtävän suorittamisen ehtona oli, että device_type on määritelty.

```yaml
- name: Konfiguroi syslog laitekohtaisia rooleja käyttäen
  hosts: all
  gather_facts: no

  tasks:

    # Käytetään rooleja konfiguraatiota varten, jossa rooli määriytyy tarkastusvaiheiden device_typen perusteella
    - name: "Laitekohtaisen roolin valitseminen"
      include_role:
        name: "{{device_type}}Konfiguraatio"
      when: device_type is defined
```

Kuvio 21. Tehtävä laitekohtaisen roolin käyttämiseksi

Ennen varsinaisten Jinja-mallipohjien toteutusta, lisättiin uusia muuttujia. Koska kaikille laitteille oli tarkoitus konfiguroida sama lokipalvelimen osoite, osoite määriteltiin erillisenä muuttujana nimeltä lokipalvelin_osoite. Tällä tavalla osoite voidaan halutessa vaihtaa helposti toiseen, kun sitä ei ole suoraan kovakoodattu jokaiseen pelikirjaan. Lokipalvelimen osoitteen lisäksi luotiin muuttujat lokipalvelimen porttia, lokitusvalintaa ja syslog-tilaa varten. Lokitusvalinnalla voitiin määritellä, että halutaanko laitteilta lähettää minimaaliset lokit vai suositellut. Oletusarvoisesti kaikki lokitusta koskevat muuttujat määriteltiin all.yml-tiedostoon, jossa määritellään kaikkia laitteita koskevat muuttujat (ks. kuvio 22).

Ansiblen muuttujat toimivat hierarkisesti, jonka takia palvelimille voidaan käyttää eri lokitusvalintaa kuin kytkimille, mikäli niin halutaan. Muuttuja voidaan määritellä ryhmäkohtaisesti palvelimet.yml-tiedostossa. Tässä työssä palvelimet oli vielä määritelty omiin ryhmiin Linux-jakeluiden perusteella, joten lokitusvalinta voidaan määritellä laitetyypin mukaisesti, kuten AlmaLinuxille tehtiin (ks. kuvio 23). Lisäksi lokitusvalinta voidaan määritellä myös laitekohtaisesti, jolloin käytetään host_vars-hakemistossa.

group_vars > ! all.yml

```yaml
1  lokipalvelin_osoite: 62.204.28.108 # Määritellään lokeja vastaanottavan palvelimen osoite
2  lokipalvelin_portti: 514 # Määritellään lokipalvelimen portti
3  lokitusvalinta: suositeltu # Määritellään halutaanko minimaalinen lokitus vai suositeltu lokitus
4  syslog_tila: local1 # Määritellään mitä syslog-facilitya halutaan käyttää (local0 - local07)
```

Kuvio 22. Syslogia koskevat muuttujat kaikkia laitteita koskevissa muuttujissa

group_vars > ! almalinux.yml

```
1  lokitusvalinta: minimaalinen
```

Kuvio 23. Poikkeava lokitusvalinta määriteltynä AlmaLinuxille

4.3.4 Mallipohjien luominen

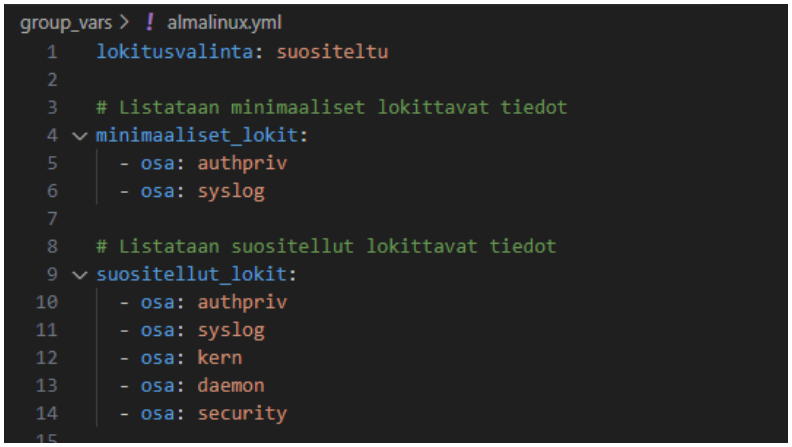
Suunnitteluvaiheessa päätettiin, että jokaisen laitteen konfiguraatiota koskevia komentoja varten käytettiin Jinjan mallipohjia. Nämä mallipohjat luotiin jokaisen laitteen laitekohtaisen roolin templates-hakemistoon. Ensimmäisenä mallipohja toteutettiin palvelimille. Rsyslogin asetukset määritellään omassa konfiguraatiotiedostossa, joten mallipohjaan toteutettiin samantyyppinen rakenne. Koska sekä AlmaLinux että Ubuntu käyttivät rsyslogia, molemmille palvelimelle toteutettiin täsmälleen samanlainen mallipohja (ks. kuvio 24).

Mallipohjassa käytettiin myös ehtolausekkeita avuksi. Ensimmäisenä ehtolausekkeella tarkastettiin mikä lokitusvalinta oli käytössä. Lokitusvalinnan perusteella muuttujalle lokitettavat_osat annettiin arvoksi ryhmäkohtaisiin muuttujiin määritetty sopiva lista (ks. kuvio 25). Seuraavana silmukka käy läpi yksittäin lokitettavat_osat-muuttujan arvoksi annetun listan. Lista antaa aina yhden osan ja sille määritetään samalla lokipalvelimen IP-osoite sekä portti.

roles > almalinuxKonfiguraatio > templates > rsyslog.conf.j2

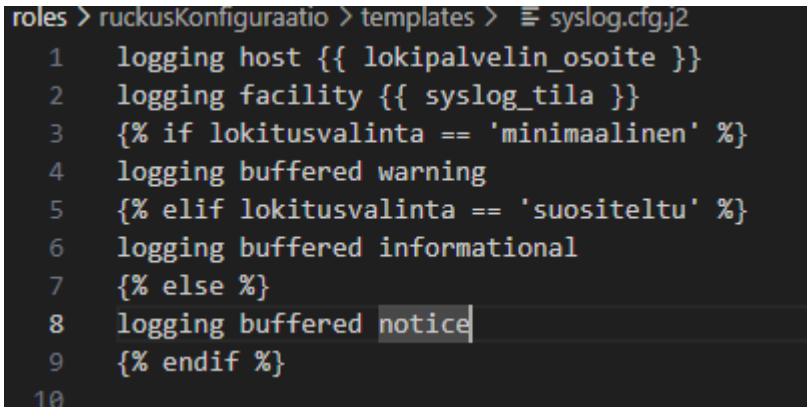
```
1  {% if lokitusvalinta == "minimaalinen" %}
2    {% set lokitettavat_osat = minimaaliset_lokit %}
3  {% elif lokitusvalinta == "suositeltu" %}
4    {% set lokitettavat_osat = suositellut_lokit %}
5  {% else %}
6    # Minimaaliset mikäli lokitusvalintaa ei ole määritelty erikseen
7    {% set lokitettavat_osat = minimaaliset_lokit %}
8  {% endif %}
9
10 {% for loki in lokitettavat_osat %}
11 {{ loki.osa }}.* @{{ lokipalvelin_osoite }}:{{ lokipalvelin_portti }}
12 {% endfor %}
```

Kuvio 24. Palvelimien Jinja mallipohja



Kuvio 25. Ubuntun lokiosat listattuna muuttujiin

Molemmille kytkimille määriteltiin omat Jinja-mallipohjat. Myös kytkimien mallipohjissa tarkastettiin ehtolausekkeen avulla, mikä lokitusvalinta oli valittu ja valinnan perusteella määriteltiin minkä syslog-tason lokeja lähetettiin lokipalvelimelle. Ensimmäisenä mallipohja luotiin Ruckusille (ks. kuvio 26). Mallipohjassa määriteltiin komentojen avulla, minne Ruckusin lokit lähetetään, mitä syslog-tilaa käytettäisiin ja minkä syslog-tason lokeja, riippuen aiemmin määritellystä lokitusvalinnasta. Huawein mallipohja oli rakenteeltaan hyvin samantyyppinen, mutta lokien lähettämiseen tarvittavat komennot olivat erilaisia (ks. kuvio 27).



Kuvio 26. Ruckusin kytkimen Jinja-mallipohja

roles > huaweiKonfiguraatio > templates > info-center.cfg.j2

```
info-center enable
{% if lokitusvalinta == 'minimaalinen' %}
info-center loghost {{ lokipalvelin_osoite }} channel loghost level warning
{% elif lokitusvalinta == 'suositeltu' %}
info-center loghost {{ lokipalvelin_osoite }} channel loghost level informational
{% else %}
info-center loghost {{ lokipalvelin_osoite }} channel loghost level notification
{% endif %}

```

Kuvio 27. Huawein kytkimen Jinja-mallipohja

4.3.5 Laitekohtaisten roolien luominen

Kun mallipohjat olivat valmiita, seuraavana nämä mallipohjat otettiin käyttöön jokaisen laitteen roolitiedostossa. Sekä Ubuntulle että AlmaLinuxille konfiguraatio otettiin käyttöön samalla tavalla (ks. kuvio 28). Mallipohja haetaan omasta tiedostosijainnistaan src-muuttujalla. Lisäksi täytyi määritellä mihin sijaintiin mallipohja suoritetaan. Rsyslogin konfiguraatio on /etc/rsyslog.d/-hakemiston tiedostossa 50-remote.conf, joten se asetettiin kohdesijainniksi dest-muuttujaa käyttämällä.

Koska rsyslogin konfiguraatiot muuttuisivat Ansiblen suorittamien Jinja-mallipohjien takia, rsyslog olisi käynnistettävä palvelimilla uusiksi. Tätä varten luotiin käsittelijä handlers-hakemiston tiedostoon main.yml, johon määriteltiin tehtävä rsyslogin uudelleenkäynnistämiseksi. Tälle tehtävälle käytettiin ansiblen service-moduulia (ks. kuvio 29).

roles > almalinuxKonfiguraatio > tasks > main.yml

```
- name: Toteuta rsyslog-konfiguraatio templaten perusteella
  ansible.builtin.template:
    src: rsyslog.conf.j2
    dest: /etc/rsyslog.d/50-remote.conf
    backup: yes
    owner: root
    group: root
    mode: '0644'
  notify: Käynnistä rsyslog uudelleen
  become: true

```

roles > ubuntuKonfiguraatio > tasks > main.yml

```
- name: Toteuta rsyslog-konfiguraatio templaten perusteella
  ansible.builtin.template:
    src: rsyslog.conf.j2
    dest: /etc/rsyslog.d/50-remote.conf
    backup: yes
    owner: root
    group: root
    mode: '0644'
  notify: Käynnistä rsyslog uudelleen
  become: true

```

Kuvio 28. AlmaLinuxin ja Ubuntun samanlaiset roolit

roles > ubuntuKonfiguraatio > handlers > ! main.yml

```yaml
- name: Käynnistä rsyslog uudelleen
  ansible.builtin.service:
    name: rsyslog
    state: restarted
  become: true
```

Kuvio 29. Käsittelijä rsyslogin uudelleenkäynnistämiseksi palvelimilla

Kytkimien konfiguraatiomuutokset toteutettiin eri tavalla kuin palvelimien. Kytkimien kohdalla käytettiin molempien laitteiden omia community.network-kirjaston moduuleja, jotka olivat ce_config sekä icx_config. Näitä moduuleja käytetään, kun Huawein CE ja Ruckusin ICX laitteille halutaan ajaa konfiguraatiomuutoksia.

Ruckusille luotiin roolikohtaisen hakemistoon tasks-alihakemistoon uusi tehtävä (ks. kuvio 30). Tehtävässä otetaan käyttöön icx_config-moduuli, jonka jälkeen lookup-komento hakee ja suorittaa rivi riviltä mallipohjassa määritetyt konfiguraatiokomennot. Ruckusin kytkimillä konfiguraatiokomentojen ajaminen vaatii tiettyjä käyttöoikeuksia. Ansiblessa nämä oikeudet saadaan käyttöön ansible_become- sekä ansible_become_method-muuttujilla. Kyseiset muuttujat määriteltiin samoihin ryhmäkohtaisiin muuttujiin, jonne oli aiemmin määritelty kirjautumisen käyttäjätunnus ja salasana (ks. kuvio 31).

```yaml
# Suoritetaan konfiguraatiomuutokset Ruckusin kytkimiin templatea käyttäen
- name: Suorita Ruckusin kytkimen lokien lähettäminen
  community.network.icx_config:
    lines: "{{ lookup('ansible.builtin.template', 'syslog.cfg.j2').splitlines() }}"
```

Kuvio 30. RuckusKonfiguraatio roolin tehtävä

```yaml
ansible_become: true
ansible_become_method: enable
```

Kuvio 31. Ruckusin ryhmämuuttujiin lisätyt muuttujat

Seuraavaksi luotiin samanlainen tehtävä Huaweille. Huawein komennot ajettiin pitkälti samalla tavalla (ks. kuvio 32). Tehtävässä ce_config-moduuliin suoritetaan lookup-komennolla Jinja-mallipohjasta löytyvät komennot. Erona Ruckusin tehtävään oli, että tehtävän lopussa täytyisi suorittaa erikseen save-komento, jonka avulla info-centeriin liittyvät muutokset tallennetaan.

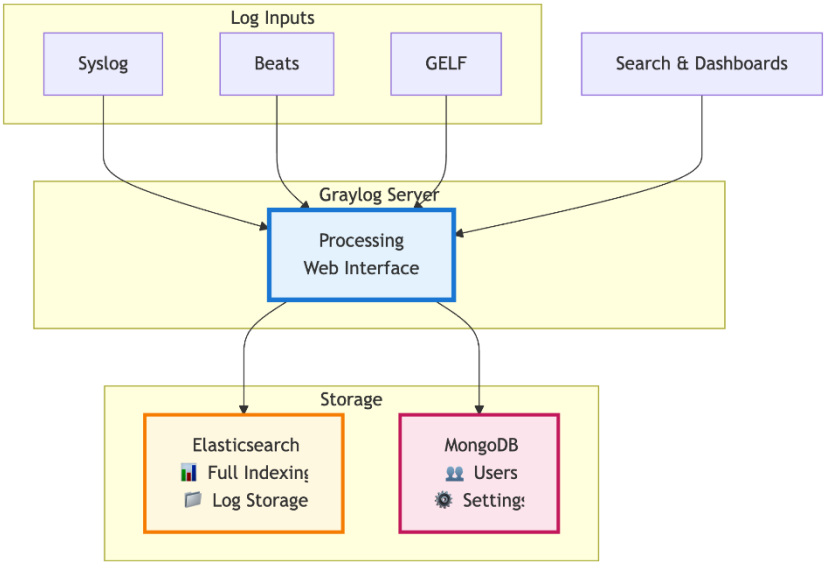
```
# Suoritetaan konfiguraatiomuutokset Huawein kytkimiin templatea käyttäen
- name: Konfiguroi Huawein syslog viestien lähettäminen
  community.network.ce_config:
    lines: "{{ lookup('ansible.builtin.template', 'info-center.cfg.j2') }}"
  save: true
```

Kuvio 32. HuaweiKonfiguraatio roolin tehtävä

4.4 Lokitusympäristön rakentaminen

Toinen rakennettava kokonaisuus oli lokitusympäristö. Suunnitteluvaiheessa päädyttiin ratkaisuun, jossa lokitusympäristön työkaluna toimii Graylog, joka vastaanottaa ympäristön laitteiden lokit suoraan syslogin avulla. Lokitusympäristö toteutettiin samalle palvelimelle, mikä toimii Ansible hallintakoneena. Näin toimittiin, koska lokitusjärjestelmää hyödynnettiin kevyesti Ansible-rakenteen toimivuuden testaukseen. On syytä huomioida, että oikeassa tuotantoympäristössä Ansiblen hallintakone sekä lokipalvelin ovat todennäköisemmin erillisiä laitteita. Yksi syy tähän on, että lokipalvelin tarvitsee yleensä paljon resursseja lokien säilöntään. Ansiblen pelikirjoja voidaan suorittaa kevyemmällä laitteella.

Graylogin kokonaisuus koostuu kolmesta osasta, MongoDB-tietokannasta, OpenSearchista sekä Graylog-palvelimesta (ks. kuvio 33). Graylog on mahdollista ottaa käyttöön niin sanotulla suoralla käyttöönotolla, jossa sen osat asennetaan ja määritellään yksitellen tai vaihtoehtoisesti Docker-kontteja käyttäen. Tässä opinnäytetyössä Graylogin käyttöönotto suoritettiin Docker-kontteja käyttämällä. Tähän ratkaisuun päädyttiin, koska Dockeria käyttämällä kaikki Graylogin osat voidaan kerralla ottaa käyttöön ja tarvittaessa sulkea ja konfiguroida uudelleen.



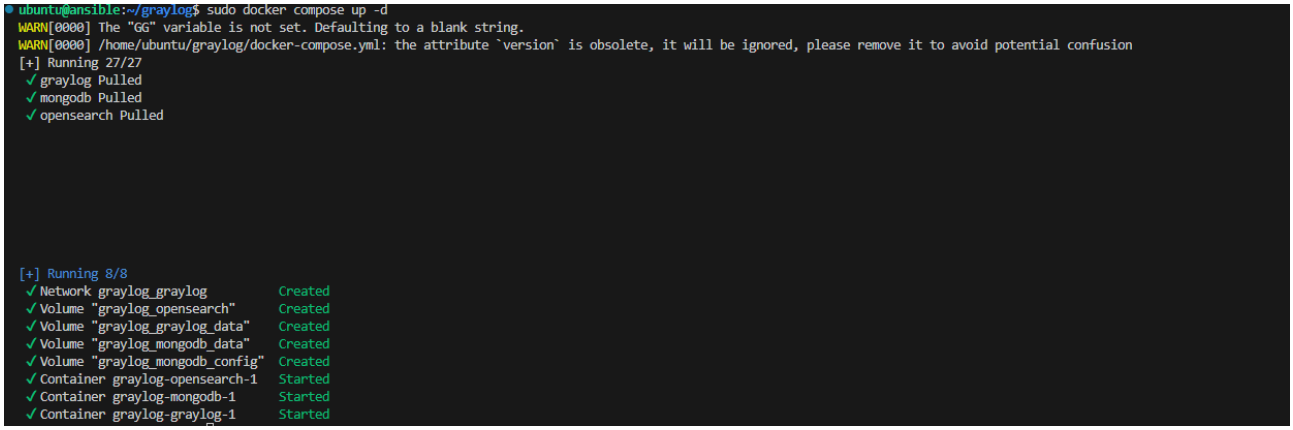
Kuvio 33. Graylogin arkkitehtuuri (Udasi 2025)

Vaihe alkoi Dockerin asentamisella. Kun Docker oli asennettu, määriteltiin docker-compose.yml. Tähän tiedostoon määriteltiin tarvittavat osat, eli MongoDB, OpenSearch sekä Graylog palvelin. Graylog vaatii kaksi salasanaa, joista ensimmäinen oli ns. salaisuus (*secret*). Lisäksi käyttöliittymän kirjautumiseen tarvittiin eri salasana, joka täytyi olla salattuna SHA256-muodossa. Salasanojen luomisen jälkeen määriteltiin portit. Oletusarvoisesti syslog käyttää porttia 514 UDP-protokollalla (Kinaro 2023). Linuxissa ensimmäiset 1024 porttia vaativat root-tason oikeuksia, jonka takia Graylogin oli käytettävä isompaa porttia. Portiksi otettiin käyttöön 1514, jonne ohjattiin portin 514 liikenne (ks. kuvio 34).

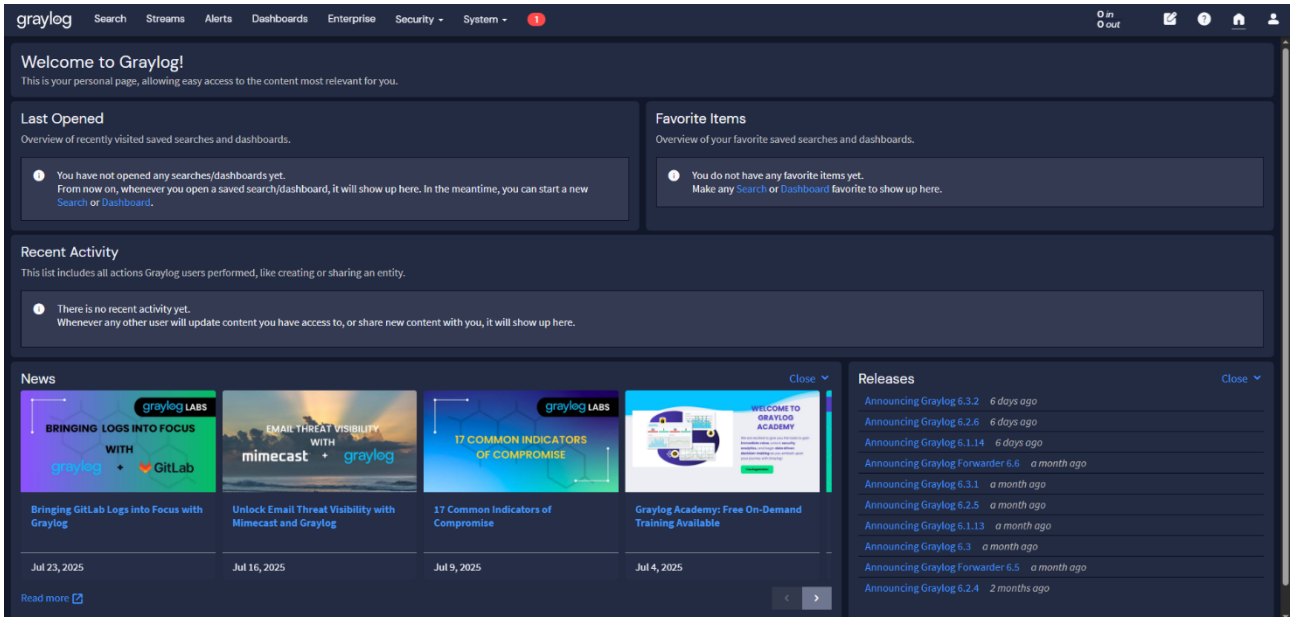
```
# Syslog TCP
- "514:1514/tcp"
# Syslog UDP
- "514:1514/udp"
```

Kuvio 34. Portin 514 liikenne ohjataan porttiin 1514

Kontit saatiin käynnistettyä komennolla **docker compose up -d** (ks. kuvio 35). Kun kontit oli käynnistynyt, konttien tilat voitiin vielä erikseen tarkistaa komennolla **docker ps**. Tämän jälkeen Graylogin käyttöliittymään pääsi selaimen kautta (ks. kuvio 36).



Kuvio 35. Docker konttien rakentaminen



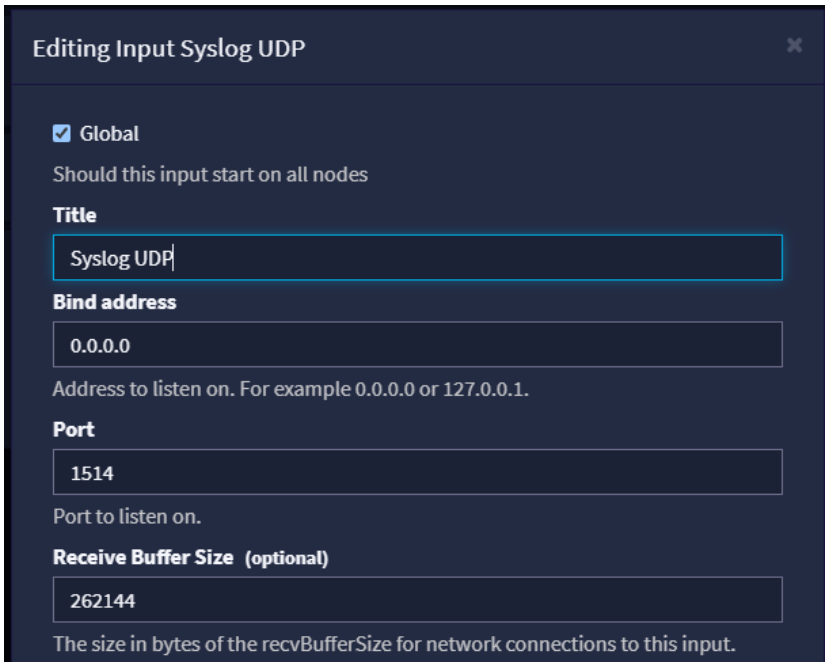
Kuvio 36. Graylogin käyttöliittymä selaimessa

Pian Graylogin käynnistämisen jälkeen selvisi, että käyttöliittymään sekä MongoDB-tietokantaan oli kaikilla pääsy, mikä oli huomattava tietoturvariski. Hallintakoneelle oli aikaisemmin toteutettu palomuurisääntöjä UFW-palomuurilla, mutta Docker ohitti kyseiset säännöt konttien käynnistyksessä. Tämän takia docker-composeen täytyi tehdä muutoksia. Graylogin käyttöliittymän, OpenSearchin sekä MongoDB:n portit sidottiin käyttämään IP-osoitetta 127.0.0.1 (ks. kuvio 37). IP-osoitteella 127.0.0.1 kyseiset palvelut olisivat saavutettavissa vain lokaalilla laitteella, eli hallintakoneella. Muutoksien jälkeen käynnissä ollut Docker-kontti suljettiin komennolla **docker-compose down** ja käynnistettiin uudelleen samalla komennolla kuin se oli aiemmin käynnistetty.

```
mongodb:
  image: "mongo:6.0.18"
  ports:
    - "127.0.0.1:27017:27017"
```

Kuvio 37. MongoDB:n portin sitominen IP-osoitteeseen 127.0.0.1

Onnistuneiden muutoksien jälkeen kirjauduttiin takaisin Graylogin hallintapaneeliin selaimen kautta. Jotta Graylog voi vastaanottaa lokeja, sille täytyi määritellä input. Koska kytkimiltä ja palvelimilta kerättiin lokit syslogin kautta, luotiin Graylogille Syslog UDP input (ks. kuvio 38). Input määriteltiin kuuntelemaan IP-osoitetta 0.0.0.0. Kyseistä IP-osoitetta käytettiin, jotta Graylog hyväksyisi saapuvan liikenteen kaikilta lokipalvelimen, tässä tapauksessa Ansible-hallintakoneen, verkkoliitännöiltä. Tällä tavoin Graylog pystyy vastaanottamaan lokeja mistä tahansa IP-osoitteesta. Lisäksi inputin portiksi merkattiin 1514, johon portin 514 liikenne aikaisemmin ohjattiin.

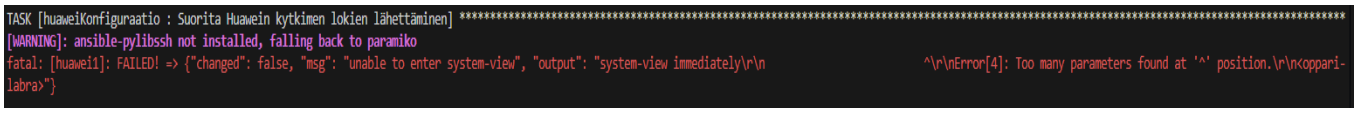


Kuvio 38. Graylogille luotu syslog UDP input

4.5 Testaus

Ansible-rakenteen sekä lokitusjärjestelmän oltua valmis, oli vuorossa niiden testaaminen yhtenä kokonaisuutena. Testaus aloitettiin suorittamalla infraKonfiguraatio.yml-pelikirja, jossa tapahtui

kaikki laitteisiin liittyvät tarkastukset, joiden jälkeen viimeisessä tehtävässä valittiin oikea Ansiblen rooli. Pelikirja ajettiin komennolla **ansible-playbook infraKonfiguraatio.yml -i inventory.ini --ask-vault-password**. Tämän jälkeen Ansible kysyi salasanaa, jota käytettiin aiemmin arkaluontoisten muuttujien salamiseen Ansible Vaultilla. Ansible suoritti pelikirjassa määritetyt tarkastustehtävät palvelimelle ja kytkimille, mutta roolien kohdalla aiheutui virheilmoitus Huawein rooliin liittyen (ks. kuvio 39). Virheilmoituksen mukaisesti Huawei ei voinut siirtyä ns. konfiguraatiotilaan, koska laite havaitsi parametrejä, joita se ei ymmärtänyt.

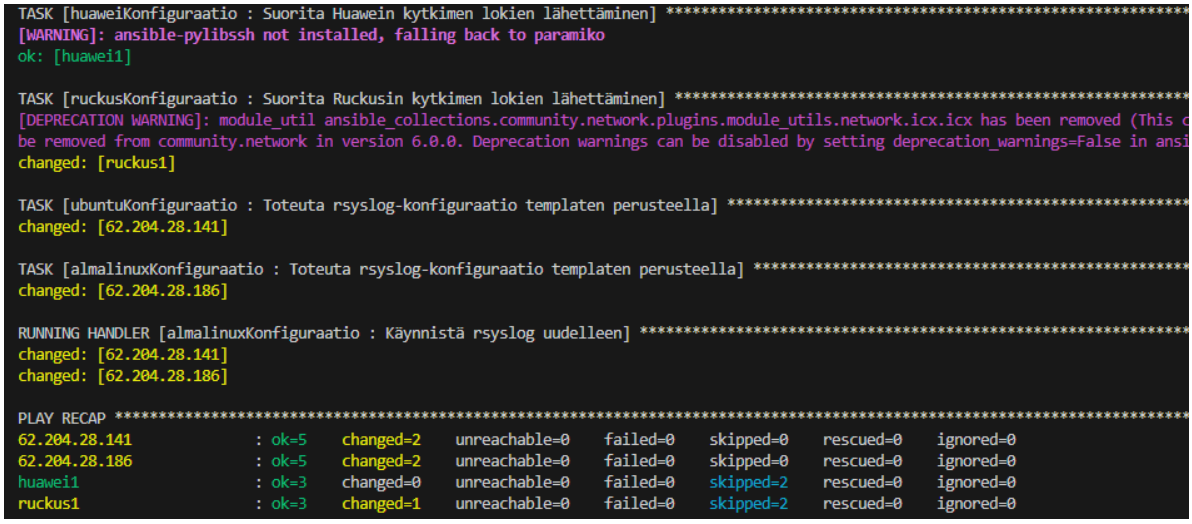


Kuvio 39. Huawein roolia koskeva virheilmoitus

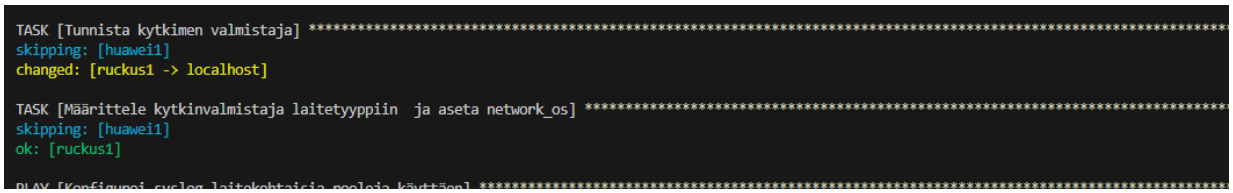
Huawein roolin kohdalla tehtiin muutoksia (ks. kuvio 40). Käytettäväksi moduuliksi vaihdettiin ce_command-moduuli, jolla voidaan ajaa yksittäisiä komentoja. Ennen mallipohjan muuttamista komennoiksi, annettiin komento system-view ja lopussa komento quit. Näiden muutoksien jälkeen pelikirja suoritettiin kerran, tällä kertaa onnistuneesti (ks. kuvio 41). Myös Netmikolla toteutetun kytkinTunnistus.py-skriptin toimivuutta testattiin Ruckusin kohdalla (ks. kuvio 42).

```
6  # Suoritetaan konfiguraatiomuutokset Huawein kytkimiin templatea käyttäen
7  - name: Suorita Huawein kytkimen lokien lähettäminen
8    community.network.ce_command:
9      commands: "{{ ['system-view'] + lookup('ansible.builtin.template', 'info-center.cfg.j2').splitlines() + ['quit'] }}"
```

Kuvio 40. Huawein roolin tehtävään tehdyt muutokset



Kuvio 41. Onnistunut testi



Kuvio 42. Kytkintunnistus onnistuu Netmikolla

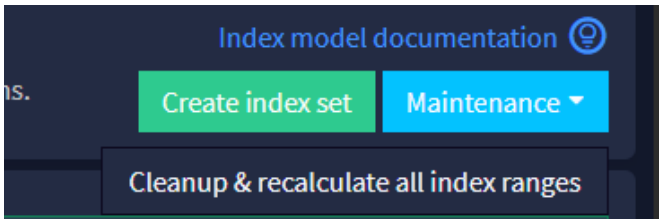
Pelikirjojen tekemät muutokset tarkastettiin vielä erikseen manuaalisesti jokaisella laitteella. Kaikkien laitteiden kohdalla oli tehty juuri ne muutokset, mitkä oli haluttu. Palvelimien kohdalla muutokset näkyivät rsyslogin konfiguraatiotiedossa (ks. kuvio 43). Kytkimien lokituksen muutokset näkyivät laitteen konfiguraatiossa (ks. kuvio 44).

Kuvio 43. Ubuntulle suoritetut rsyslogin konfiguraatiot käytännössä

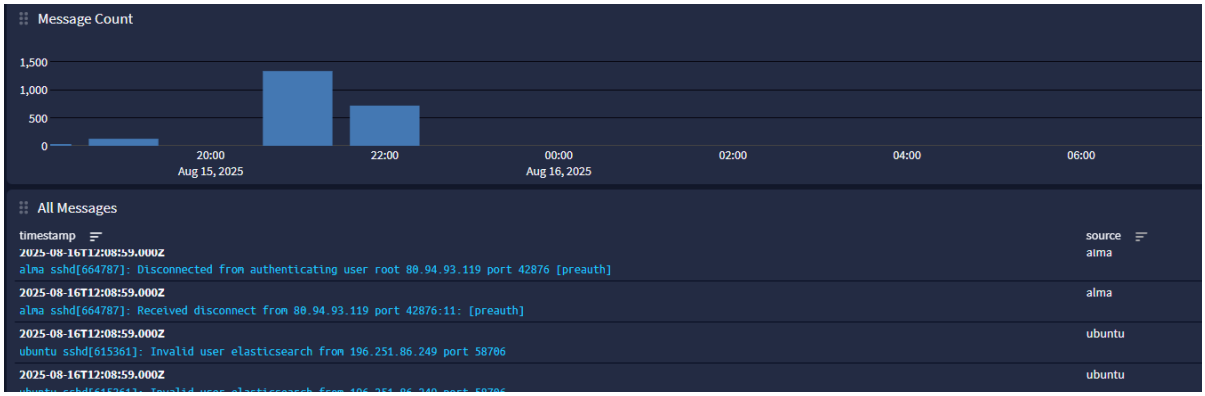
```
<oppari-labra>display current-configuration | include info-center
info-center loghost 62.204.28.108 level informational
```

Kuvio 44. Huawein info-centerin konfiguraatio käytännössä

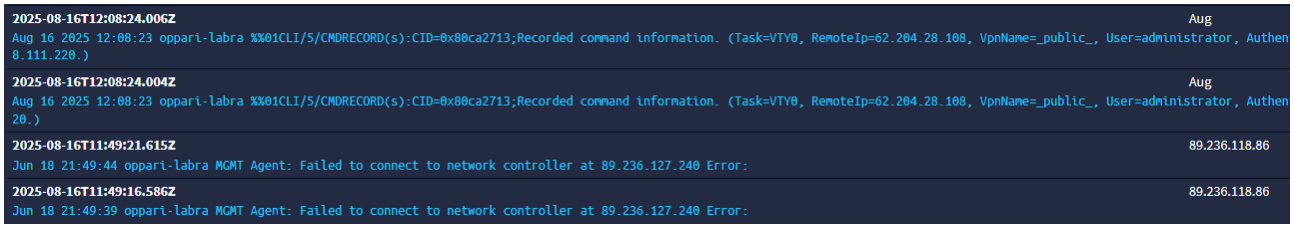
Kun Ansiblen pelikirja oli suoritettu onnistuneesti kaikkien laitteiden osalta, Graylogista tarkastettiin lokien vastaanotto. Syslog input osoitti vastaanottavansa lokeja, mutta nämä lokit eivät näkyneet niitä hakiessa. Hakuvalikko antoi virheilmoituksen *OpenSearch exception [type=index_not_found_exception, reason=no such index []]*. Tämän virheen korjaamiseksi Graylogin oletusindeksille suoritettiin puhdistus ja indeksien uudelleenlaskenta (ks. kuvio 45). Indeksin korjaamisen jälkeen lokit olivat näkyvissä ja niitä pystyi tarkastelemaan (ks. kuvio 46). Suurin osa vastaanotetuista lokeista olivat peräisin palvelimilta, mutta myös kytkimien lokien vastaanottaminen oli todennettavissa (ks. kuvio 47).



Kuvio 45. Indeksien puhdistus ja korjaaminen

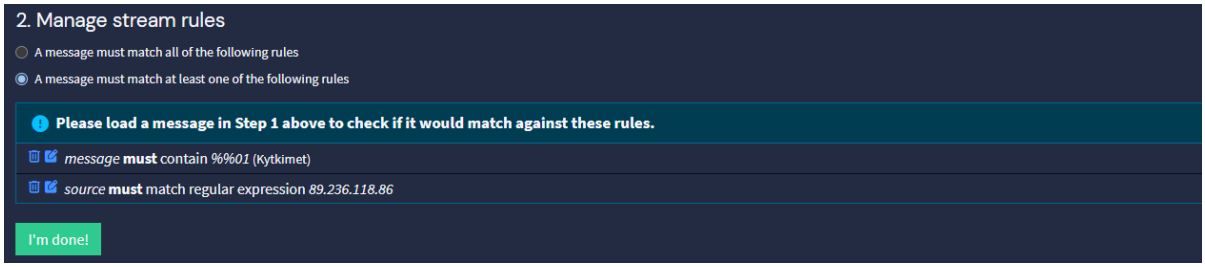


Kuvio 46. Graylogin vastaanottamat lokit



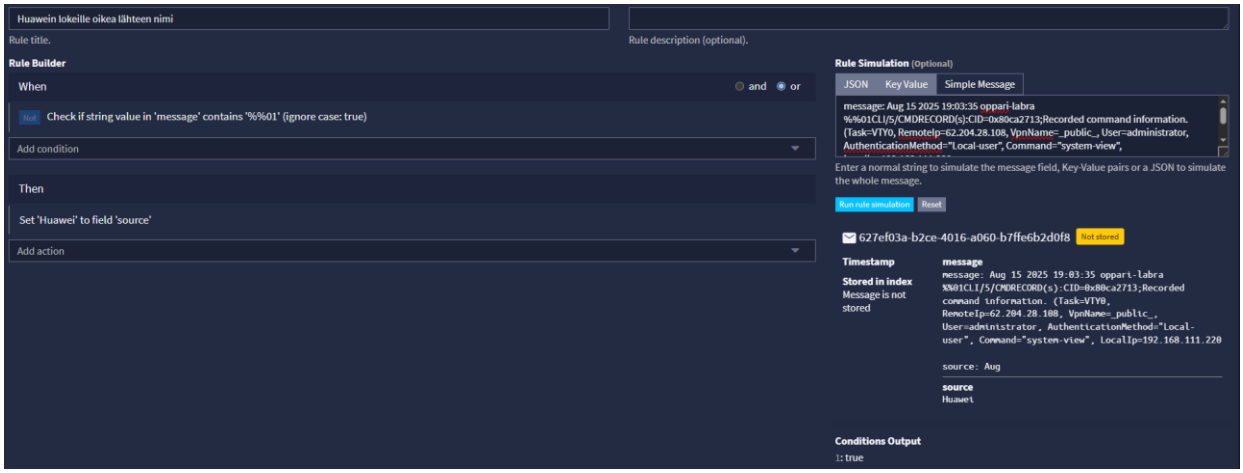
Kuvio 47. Kytkimien lokit Graylogissa

Kytkimien lokien tarkastellussa ilmeni, että Huawein kytkimiä koskevien lokien lähteeksi oli merkitty Aug, vaikka lähde on tyypillisesti laitteen nimi tai IP-osoite. Lokin lähteeksi oli valikoitunut viestissä ilmenevä aikaleiman kuukausi. Tämä ongelma todennäköisesti johtui Huawein info-centerin lokien muotoilusta, jonka takia Graylog ei pysty jäsentämään viestistä oikeaa lähdettä samalla tavalla kuin normaaleista syslog-viesteistä. Lähteen nimi oli kuitenkin mahdollista korjata oikeaksi Graylogin tarjoamien toimintojen avulla. Ensimmäisenä luotiin striimi (*stream*) nimeltä kytkimet (ks. kuvio 48). Striimin säännöksi määriteltiin, että lokiviestistä tulisi löytyä Huawein tunniste %%01 tai sen lähteen täytyy vastata kytkimien IP-osoitetta.



Kuvio 48. Striimin säännöt

Seuraavaksi luotiin putki (*pipeline*) Graylogiin nimellä Huawei-CE. Putkelle luotiin uusi sääntö, jonka tarkoitus oli muuttaa Huawein laitteilta saapuvien lokien lähde oikeaksi (ks. kuvio 49). Myös tässä säännölle annettiin ehdoksi, että mikäli saapunut lokiviesti sisältää tunnisteen %%01, asetetaan lähteeksi Huawei. Kyseinen tunniste valittiin, koska alkuperäisesti lähteeksi merkattu aikaleimasta ilmenevä kuukausi vaihtuu pakollisesti, jolloin siihen vertaaminen ei ole toimiva ratkaisu pitkällä aikajänteellä. Säännön toimivuutta simuloitiin ennen sen kokeilemista käytännössä.



Kuvio 49. Putken sääntö Huawein lähteen nimen korjaamiseksi

Luotu sääntö otettiin käyttöön putkelle. Säännön käyttöönoton jälkeen aloitettiin lokien vastaanotto. Juuri luotujen striimin ja putken toimivuutta testattiin kirjautumalla Huawein kytkimelle, joka lähetti lokiviestin Graylogiin. Lokiviestistä oli näkyvissä, että sen lähde oli muuttunut putken säännön mukaisesti Huaweiksi (ks. kuvio 50).

Kuvio 50. Huawein lokit putken sääntöjen käyttöönoton jälkeen

5 Tulokset

5.1 Ansiblen rakenteen arviointi

Ansiblen rakenteen prosessi erosi siitä, mitä suunnitteluvaiheessa alun perin oli mietitty. Tähän vaikutti suuresti se, miten Ansible toimii ohjelmistona itsessään esimerkiksi laitteisiin yhdistämisessä ja tietojen keräämisessä. Suunnitelman toisessa vaiheessa oli alun perin tarkoituksena tarkastetaan, että onko inventoryyn lisätty laite kytkin vai palvelin. Tämä vaihe jäi kuitenkin pois toteutuksessa, koska Ansible käsittelee verkkolaitteita eri tavalla kuin palvelimia.

Pelikirjojen rakenne on pyritty pitämään yksinkertaisena ja Ansible pystyi suorittamaan tehtävät nopeasti testausvaiheessa. Jokainen tehtävä on yleisesti kuvailtu kommenteilla, mutta toimintojen logiikkaa olisi voinut myös kommentoida paremmin. Roolien käyttämistä voi pitää onnistuneena. Niiden avulla saatiin selkeästi erotettua kytkin- sekä palvelinkohtaiset konfiguraatiot. Molempien palvelimien kohdalla kuitenkin käytetään täysin samanlaista roolirakennetta sekä Jinja-mallipohjaa, joka on käytännössä turhaa toistuvuutta.

Skaalautuvuutta tarkasteltiin sen pohjalta, kuinka helposti ympäristöön on mahdollista lisätä uusia laitteita, jos esimerkiksi yrityksellä on käytössä muita kuin Huawein tai Ruckusin kytkimiä. Tämä tilanne edellyttäisi uusien ryhmämuuttujien ja roolien lisäämistä, sekä muutoksia ehtolausekkeisiin. Palvelimia koskevat ehtolausekkeet sijaitsevat vain kuitenkin yhdessä pelikirjassa, jolloin niitä on mahdollista lisätä nopeasti. Mikäli ympäristöön lisätään samanlaisia kytkimiä ja palvelimia kuin mitä opinnäytetyössä on käytetty, tällöin tarvitsee ainoastaan lisätä laitteiden tiedot inventaarioon oikeisiin ryhmiin.

Muuttujien kohdalla laitteiden kirjautumistiedot määriteltiin ryhmämuuttujiin, esimerkiksi Huawein kohdalla group_vars-hakemiston tiedostoon huawei.yml. Tämä ei ollut ehkä skaalautuvuuden näkökulmasta parhain ratkaisu, mikäli ympäristöön otettaisiin käyttöön lisää Huawein kytkimiä. Vaikka laitteille käytettäisiin samaa käyttäjätunnusta kirjautumiseen, salasanat ovat lähtökohtaisesti aina uniikkeja, eli jokaisella kytkimellä on käytössä eri salasana. Kirjautumistiedot olisi ollut siis parempi määritellä isäntäkohtaisiin muuttujiin ryhmäkohtaisten muuttujien sijaan.

5.2 Lokitusympäristön arviointi

Lokitusympäristössä lokitus oli tarkoitus toteuttaa modulaarisesti. Osa tästä modulaarisesta lokituksesta toteutui jo Ansible-rakenteen kautta. Palvelimien osalta Graylog vastaanotti vain niiden syslog-tilojen viestejä, jotka olivat määriteltynä rsyslogin konfiguraatioon. Kytkimien osalta modulaarisuus jää vajaaksi, koska kyseenomaisissa laitteissa lokien lähettäminen etäpalvelimelle perustuu usein vain tietyn syslog tason ja sitä alempien tasojen lokeihin, mikä rajaa mahdollisuutta toteuttaa lokitettavien osien valitseminen samalla tavalla kuin palvelimien kohdalla.

Kaikkien laitteiden lokitietoja vastaanotettiin. Tuloksia varten viestejä kerättiin lyhyissä muutaman kymmenen minuutin aikajaksoissa kolmen päivän aikana, jolloin Graylog vastaanotti yhteensä 4475 lokiviestiä. Suurin osa kerätyistä lokeista oli palvelimiin liittyviä epäonnistuneita kirjautumisyrityksiä. Kerätyissä lokeissa oli lisäksi myös lokeja, jotka koskivat kytkimien konfiguraatiomuutoksia. Myös palvelimien korotettujen oikeuksien käytöstä oli tallennettu erinäisiä lokeja. Nämä edellä mainitut lokit ovat laitevalvonnan näkökulmasta välttämättömiä, joten lokitusta voi pitää niiden osalta onnistuneena. Kytkimistä kerättyjen lokien määrä jäi hyvin pieneksi, mutta myös niiden kohdalla kerätyt lokit koskivat kirjautumistapahtumia sekä konfiguraatiomuutoksia.

Graylogin asentamiseen valittiin Docker-kontit. Valinta oli onnistunut ja Graylog oli hyvin nopeasti otettavissa käyttöön. Tässä vaihtoehdossa kuitenkin oli huomioitava Dockerin tapa sivuttaa UFW-palomuurin säännöt, jonka takia konttien liikenne oli julkista. Tämän takia Graylogia koskevien porttien kanssa täytyi olla tarkkana, että kriittisien osien portit olivat vain lokaalisti saatavilla.

5.3 Tulosten yhteenveto

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa Ansible-ratkaisu, jonka avulla infrastruktuuriin kuuluvien laitteiden valvontakonfiguraatiot voidaan automatisoida tunnistaen syntaksierot laitteiden välillä. Osana tätä tavoitetta oli myös lokitusympäristö, jossa lokitus tapahtuisi modulaarisesti. Kun Ansiblen rakennetta sekä lokitusjärjestelmää tarkastellaan kokonaisuutena, sen voidaan nähdä vastaavan pitkälti toimeksiantajan antamiin tavoitteisiin. Toteutettujen Ansiblen pelikirjojen avulla voidaan samanaikaisesti automatisoida eri kytkinvalmistajien sekä Linux-jakeluiden

valvontakonfiguraatiot ottaen huomioon mahdolliset syntaksierot. Lokitusympäristössä lokitus tapahtuu palvelimien kohdalla modulaarisesti, mutta kytkimien kohdalla järjestelmä vaatii jatkokehitystä.

Opinnäytetyön alussa oli kolme tutkimuskysymystä, joihin pyrittiin tämän työn avulla keksimään vastaukset. Ensimmäinen kysymys oli, että miten verkkolaitteiden ja palvelimien valvonta voidaan automatisoida skaalautuvasti ja modulaarisesti Ansiblea hyödyntäen? Palvelimille luotiin omat ryhmät inventaarioon, jotka nimettiin palvelimen käyttöjärjestelmien mukaisesti. Myös kytkinvalmistajille luotiin omat ryhmät. Molemmat palvelinryhmät määriteltiin vielä pääryhmään nimeltä *palvelimet*. Samanlailla toimittiin myös kytkimien kohdalla. Toteutuksessa käytettiin vain yhtä pelikirjaa, joka suoritti konfiguraatiomuutokset kerralla kaikkiin inventaarioon lisättyihin laitteisiin. Ympäristöön on helppo lisätä lisää Huawein ja Ruckusin kytkimiä, vaikka kytkimien mallit eivät olisikaan täysin sama kuin opinnäytetyössä käytettyjen.

Toisena oli, että mitä lokitietoja on olennaista kerätä palvelinten ja kytkinten ylläpidon ja tietoturvan kannalta? Opinnäytetyön perusteella molempien laitteiden kohdalla on olennaista kerätä onnistuneet ja epäonnistuneet kirjautumiset, järjestelmämuutokset sekä konfiguraatioiden muutokset. Tämän lisäksi palvelimien kohdalla myös korotettujen oikeuksien (sudo, su) käyttämisien lokittaminen. Viimeisenä kysymyksenä oli, miten Ansible voidaan hyödyntää tunnistamaan laitteistokohtaisia erityispiirteitä, kuten syntaksieroja? Laitteiden käyttöjärjestelmät selvitettiin ehtolausekkeiden avulla, jonka pohjalta tieto tallennettiin muuttujaan. Muuttujaan tallennetun tiedon perusteella Ansible osasi suorittaa oikean pelikirjan, jossa oli juuri kyseenomaisen laitteen syntaksiin sopivat komennot.

6 Pohdinta

6.1 Keskeisimmät havainnot

Opinnäytetyön tulokset osoittavat, että toteutuksen avulla on mahdollista automatisoida valvonnan konfiguraatio eri valmistajan kytkimille sekä Linux-jakeluille. Palvelimien kohdalla Alma-Linuxissa sekä Ubuntussa oli käytössä rsyslog, mikä helpotti konfiguraatiomuutoksien tekemistä,

kun lokipalvelimelle lähetettävät lokit pystyivät erikseen määrittelemään. Kytkimien kohdalla lokien lähettäminen lokipalvelimelle saatiin toteutettua vain muutaman komennon avulla, jotka olivat helppo lisätä Jinja-mallipohjiin.

Vaikka opinnäytetyön havainnot osoittavat automatisoinnin olevan onnistuessaan tehokas tapa toteuttaa yhtenäiset valvontakonfiguraatiot IT-infrastruktuurin laitteille, on siinä edelleen kiinnitettävä huomiota tiettyihin asioihin. Automatisoinnin voidaan todeta vähentävän virheellisien konfiguraatioiden, mutta se ei poista niitä kokonaan. Myös automatisoinnissa pelikirjojen luonnissa inhimilliset virheet ovat edelleen mahdollisia. Opinnäytetyössä näitä pelikirjoja koskevia virheitä oli havaittavissa esimerkiksi Huawein kytkimen konfiguraation automatisoinnissa. Tämä takia infrastruktuurin automatisoinnin kanssa kannattaa toimia kuten ohjelmistoprojektien kanssa, eli testata automatisoivien koodien toimivuutta erillisessä testiympäristössä ennen sen tuomista tuotantoympäristöön.

Kun opinnäytetyön aihetta sekä tuloksia tarkastellaan kestävän kehityksen näkökulmasta, automatisoinnin onnistuessa infrastruktuurin konfigurointiin kuluu vähemmän resursseja. Digitalisaation jatkuvan kasvamisen myötä esimerkiksi konesalien määrän noustessa myös valvottavan infrastruktuurin määrä kasvaa. Valvontakonfiguraatioiden automatisointi vähentää manuaalisen työn määrää, joka vapauttaa ylläpitotyöhön tarkoitettuja resursseja muuhun käyttöön.

6.2 Muutos- ja jatkokehitysideat

Opinnäytetyön tuloksien pohjalta suositellaan tiettyjä muutoksia nykyisiin pelikirjoihin sekä erilaisia jatkokehitysideoita. Jatkokehityksessä on kuitenkin syytä huomioida, että tässä työssä hyödynnetty community.network-kokoelma poistuu Ansiblesta versiossa 2.19. Opinnäytetyötä tehdessä Ansiblesta oli käytössä versio 2.18.

Muutosehdotus liittyy palvelimien roolien käyttämiseen. Sekä AlmaLinux että Ubuntu käyttivät molemmat rsyslogia, mutta molempia käyttöjärjestelmiä varten oli oma rooli. Turhan toistuvuuden välttämiseksi koodissa, voitaisiin näille käyttöjärjestelmille käyttää yhtä ja samaa roolia sekä Jinja-mallipohjaa, vaikka lokitettavat osat olisikin määritelty molempien omiin ryhmäkohtaisiin muuttujiin.

Toteutetussa versiossa ei ollut minkäänlaista tarkistusta onnistuneesta konfiguraatiosta. Ansiblen pelien yhteenveto osoittaa, että onnistuivatko kaikki pelikirjan tehtävät, muokattiinko jotakin isäntälaitteessa vai tapahtuiko virhe. Varmistuakseen konfiguraatioon tehdyistä muutoksista, tieto täytyy selvittää erikseen kirjautumalla näille laitteille. Ratkaisu tähän olisi, että roolien kohdalla luotaisiin vielä toinen tehtävä, joka näyttäisi lokitusta koskeviin konfiguraatioihin tehdyt muutokset. Kytkimien kohdalla tämän saisi selville tulostavassa toisessa roolin tehtävässä kytkimen konfiguraation.

Lokitusympäristössä kytkimien modulaarinen lokitus vaatisi vielä jatkokehitystä. Yksi ratkaisu kehittää modulaarisuutta kytkimien kohdalla voisi olla saapuvien lokien erillinen suodatus niitä vastaanottavassa päässä Graylogin putkien sääntöjen avulla. Tämä vaatisi ensin selvityksen, millaisia viestejä kytkimet lähettävät milläkin syslog-tasolla ja onko tasolla sellaisia lokiviestejä, jotka voidaan suodattaa pois ilman, että viestin poissuodatuksella on haitallisia vaikutuksia laitevalvontaan ja tietoturvaan.

6.3 Opinnäytetyön eettisyys

Opinnäytetyö toteutettiin noudattaen Tutkimuseettisen Neuvottelukunnan (TENK) hyvää tieteellistä käytäntöä sekä Jyväskylän Ammattikorkeakoulun eettisiä periaatteita. Työssä käytetyt materiaalit ovat saatavilla ja luettavissa julkisissa lähteissä. Työn tietoperustassa käytetty materiaali on pyritty ensisijaisesti keräämään mahdollisimman uusista, riippumattomista ja ei-kaupallisista lähteistä. Toteutusvaiheessa on kuitenkin käytetty Ansiblen dokumentaatiota, mahdollisimman ajankohtaisimman ja luotettavan tiedon varmistamiseksi. Lähdemerkinnät sekä viittaukset työssä käytettyihin lähteisiin on tehty Jyväskylän Ammattikorkeakoulun raportointiohjeen mukaisesti. Plagioinnin välttämiseksi työ on tarkistettu opiskelijalle saatavilla olleiden plagioinnintarkastustoimintojen avulla. Työssä käytettiin Googlen Gemini-tekoälyä avuksi opinnäytetyön raportin rakenteen tarkistamisessa ja jäsentämisessä.

Työn tuloksien luotettavuus perustuu rajallisessa ympäristössä toteutettuihin testeihin. Ympäristö sisälsi toimeksiantajan tarjoamia aitoja verkkolaitteita, joten tulokset heijastavat todellisia käyttötapauksia. On kuitenkin huomioitava, että rajallisessa ympäristössä on ollut käytössä vain yksi kaksi eri kytkintä sekä kaksi eri Linux-jakelua, joten sen toimivuudesta suuremmassa, useamman

saman valmistajan tai käyttöjärjestelmän sisältävässä ympäristössä ei ole luotettavaa tietoa. Toteutukseen perustuvia tuloksia on pyritty analysoimaan objektiivisesti. Lokitusjärjestelmän arvioinnissa hyödynnetyt lokitiedot ovat kerätty lyhyen ja rajallisen aikavälin sisällä, joten kaikkien toteutusvaiheessa laitteille määritettyjen lokien keräämistä ei ole välttämättä pystytty todentamaan. Opinnäytetyössä esiintyvät mahdolliset virheet ovat opinnäytetyön kirjoittajan vastuulla.

Lähteet

Ansible and Jinja2: Creating Dynamic Templates. 2025. GeeksForGeeks artikkeli. Viitattu 7.8.2025. https://www.geeksforgeeks.org/devops/ansible-and-jinja2-creating-dynamic-templates/.

Ansible Roles. 2024. Geeksforgeeks artikkeli. Viitattu 7.5.2025. https://www.geeksforgeeks.org/ansible-roles/.

Awati, R. & Coutemanche, M. 2023. What is the Ansible automation platform. TechTarget artikkeli. Viitattu 14.4.2025. https://www.techtarget.com/searchitoperations/definition/Ansible.

A 2016/679. 2016. EU:n yleinen tietosuoja-asetus. Viitattu 15.4.2025 https://eur-lex.europa.eu/legal-content/FI/TXT/?uri=celex:32016R0679.

Bandurchin, A. 2025. 6 Open-Source Log Management Tools In 2025. Viitattu 17.6.2025. https://uptrace.dev/blog/open-source-log-management.

Bocetta, S. 2019. 5 useful open source log analysis tools. Viitattu 17.6.2025. https://opensource.com/article/19/4/log-analysis-tools.

Byers, K. 2021. Netmiko Library. Viitattu 13.7.2025. https://pynet.twb-tech.com/blog/netmiko-python-library.html.

Codex, A. 2024. Configuring Syslog for Centralized Logging on AlmaLinux 9. Reintech blogipostaus. Viitattu 12.06.2025. https://reintech.io/blog/configure-syslog-centralized-logging-almalinux-9.

Conditionals. 2025. Ansible dokumentaatio. Viitattu 1.8.2025. https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_conditionals.html.

Discovering variables: facts and magic variables. 2025. Ansible dokumentaatio. Viitattu 5.5.2025. https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_vars_facts.html.

Ellingwood, J. 2022. How To Use Ansible Vault to Protect Sensitive Playbook Data. DigitalOcean ohje. Viitattu 5.7.2025. https://www.digitalocean.com/community/tutorials/how-to-use-vault-to-protect-sensitive-ansible-data#what-is-ansible-vault.

Fahim, M. 2025. What Are Ansible Roles & Why Are They Important? Cyberpanel blogipostaus. Viitattu 5.5.2025. https://cyberpanel.net/blog/ansible-roles.

Getting started with Ansible. 2025. Ansible dokumentaatio. Viitattu 28.5.2025. https://docs.ansible.com/ansible/latest/getting_started/index.html.

Glossary. 2025. Ansiblen dokumentaatio. Viitattu 20.8.2025. https://docs.ansible.com/ansible/latest/reference_appendices/glossary.html.

Gupta, M. N.d. What network security logs are essential for effective monitoring? LinkedIn yhteisöartikkeli. Viitattu 15.4.2025. https://www.linkedin.com/advice/3/what-network-security-logs-essential-effective-u2o1c.

How Ansible Works. N.d. Red Hat Ansible. Viitattu 14.4.2025. https://www.redhat.com/en/ansible-collaborative/how-ansible-works.

How Network Automation is Different. 2025. Ansible dokumentaatio. Viitattu 4.7.2025. https://docs.ansible.com/ansible/latest/network/getting_started/network_differences.html.

How to build your inventory. 2025. Ansible dokumentaatio. Viitattu 17.6.2025. https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html.

Hyde, T. 2025. The Hidden Costs of Manual Infrastructure Management. Viitattu 10.7.2025. https://www.blackdown.org/hidden-costs-of-manual-infrastructure-management/.

Installing Ansible on specific operating systems. 2025. Ansible dokumentaatio. Viitattu 17.6.2025. https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html.

Kinaro, F. 2023. A Comprehensive Introduction to Syslog. Earthly blogipostaus. Viitattu 18.5.2025. https://earthly.dev/blog/what-is-syslog/.

Näin keräät ja käytät lokitietoja. 2023. Traficom Kyberturvallisuuskeskus. Viitattu 14.4.2025. https://www.kyberturvallisuuskeskus.fi/fi/ajankohtaista/ohjeet-ja-oppaat/nain-keraat-ja-kaytat-lokitietoja.

Overview of Information Center. 2021. Huawei konfigurointiopas. Viitattu 4.6.2025. https://support.huawei.com/enterprise/en/doc/EDOC1000178167/7186ee75/overview-of-the-information-center#dc_fd_info_0002.

Priority logs for SIEM ingestion: Practitioner guidance. 2025. Australian Signals Directorate. Viitattu 9.6.2025. https://www.cyber.gov.au/resources-business-and-government/maintaining-devices-and-systems/system-hardening-and-administration/system-monitoring/implementing-siem-and-soar-platforms/priority-logs-for-siem-ingestion-practitioner-guidance.

Rajender, D. 2024. What Is Log Monitoring? It's Importance, Key Components and Open-Source Options. Opsverse. Viitattu 17.6.2025. https://opsverse.io/2024/05/03/what-is-log-monitoring-its-importance-key-components-and-open-source-options/.

Remote Development using SSH. 2025. Visual Studio Code dokumentaatio. Viitattu 12.7.2025. https://code.visualstudio.com/docs/remote/ssh.

Sheldon, R. 2023. What is Ubuntu? Techtarget. Viitattu 8.8.2025. https://www.techtarget.com/searchdatacenter/definition/Ubuntu.

Show logging. N.d. CommScope Ruckus FastIron viiteopas. Viitattu 16.06.2025. https://docs.commscope.com/bundle/fastiron-10020-commandref/page/GUID-D1307AB0-38DE-49B7-9C4D-A40E47174F71.html.

Sinha, N. 2025. Graylog vs Loki – Choosing the Right Log Management Tool. Viitattu 7.8.2025. https://signoz.io/comparisons/graylog-vs-loki/.

Stawiarska, J. IT infrastructure monitoring – everything you need to know. 2023. Codilime blogi. Viitattu 9.6.2025 https://codilime.com/blog/it-infrastructure-monitoring-all-you-need-to-know/.

Synytsia, M. 2022. Infrastructure as Code Explained: Benefits, Types, and Tools. Altexsoft blogi. Viitattu 8.8.2025. https://www.altexsoft.com/blog/infrastructure-as-code/.

Tietoja meistä. N.d. TNNet Oy verkkosivut. Viitattu 14.4.2025. https://tnnet.fi/tietoja-meista/.

Toikko, T, & Rantanen, T. 2009. Tutkimuksellinen kehittämistoiminta. Viitattu 20.6.2025. https://urn.fi/URN:ISBN:978-951-44-7732-4.

Top 6 Log Management Tools and How to Choose. N.d. Exabeam. Viitattu 17.6.2025. https://www.exabeam.com/explainers/log-management/top-6-log-management-tools-and-how-to-choose/.

Top Log Management Tools. 2024. Zenarmor. Viitattu 17.6.2025. https://www.zenarmor.com/docs/network-security-tutorials/top-log-management-tools.

Understanding the Information Center. 2021. Huawei dokumentaatio. Viitattu 4.6.2025. https://support.huawei.com/enterprise/en/doc/EDOC1000178167/681dee3/understanding-the-information-center.

Udasi, A. 2025. Graylog vs Loki: Key Differences and Use Cases. Last9.io blogi. Viitattu 7.8.2025. https://last9.io/blog/graylog-vs-loki/#when-should-you-pick-graylog-or-loki.

Uglov, D. 2023. Benefits of Ansible. LinkedIn artikkeli. Viitattu 15.4.2025. https://www.linkedin.com/pulse/benefits-ansible-denis-uglov/.

What are Server Logs?. N.d. Simpletiger asiasanasto. Viitattu 15.4.2025. https://www.simpletiger.com/resources/glossary.

What is AlmaLinux. 2025. GeeksForGeeks artikkeli. Viitattu 16.6.2025. https://www.geeksforgeeks.org/linux-unix/what-is-almalinux/.

What is an Ansible Role-and how is it used? 2024. Red Hat Ansible. Viitattu 4.5.2025. https://www.redhat.com/en/topics/automation/what-is-an-ansible-role.

What is network device logging and why is it necessary? N.d. ManageEngine Eventlog Analyzer. Viitattu 14.4.2025. https://www.manageengine.com/products/eventlog/logging-guide/network-device-logging.html.

Whittaker, G. 2025. Streamline Your Logs: Exploring Rsyslog for Effective System Log Management on Ubuntu. Linux Journal. Viitattu 16.6.2025. https://www.linuxjournal.com/content/streamline-your-logs-exploring-rsyslog-effective-system-log-management-ubuntu.

Why Logs Are the Foundation of Security. N.d. Mezmo oppimisartikkeli. Viitattu 15.4.2025. https://www.mezmo.com/learn-observability/why-logs-are-the-foundation-of-security.