HAMK
Hämeen ammatti-
korkeakoulu

Verkkokytkimen laiteohjelmiston päivittämisen automatisointi

Ammattikorkeakoulututkinnon opinnäytetyö

Tieto- ja viestintätekniikka, insinööri (AMK)

Kevät, 2026

Juho-Pekka Hjelt

HAMK
Hämeen ammatti-
korkeakoulu

Tiivistelmä

Koulutus	Tieto- ja viestintätekniikka, insinööri	
Tekijä	Juho-Pekka Hjelt	Vuosi 2026
Työn nimi	Verkkokytkimen laiteohjelmiston päivittämisen automatisointi	
Ohjaaja	Marko Grönfors	

Opinnäytetyön tavoitteena oli rakentaa automaatiotyökalu Aruba CX -mallisien verkkokytkimien ohjelmistoversioiden päivittämiseen Telia Cygate Oy:n asiantuntijoiden käyttöön. Yrityksellä on hallussaan valtava määrä verkkokytkimiä, joiden manuaalinen päivittäminen vie kohtuuttoman paljon aikaa. Automaatiolla laitteet voidaan päivittää massana samanaikaisesti yhden henkilön toimesta.

Automaatiotyökalu ohjelmoitiin Python-ohjelmointikielellä käyttäen Nornir-automaatiokehystä. Kehitystä varten käytössä oli yrityksen verkkokytkimistä koostuva laboratorioympäristö, jossa päästiin kokeilemaan ohjelman suoriutumista vapaasti.

Kehityksen tuloksena valmistui toimiva runko, jonka automaatiotiimi jatkokehitti yrityksen käytäntöihin sopivaksi versioksi. Kehitystyössä jouduttiin turvautumaan hieman epäoptimaalisiin toteutustapoihin, koska syvällistä ohjelmointikokemusta ei ollut. Projektille asetettuihin tavoitteisiin ei kaikkien toiminnallisuuksien puolesta päästy, mutta keskeisimmät toiminnot saatiin implementoitua. Valmistunutta runkoa hyödyntäen automaatiotyökalu saadaan helposti jatkokehitettyä myös muille kytkinmalleille.

Avainsanat	Verkkoautomaatio, Python, Nornir
Sivut	24 sivua

HAMK
Hämeen ammatti-
korkeakoulu

Abstract

DP	Bachelor of engineering, Information and Communication Technology	
Author	Juho-Pekka Hjelt	Year 2026
Subject	Network Switch Firmware Update Automatization	
Supervisor	Marko Grönfors	

The objective of this thesis was to develop an automation tool for updating the firmware versions of Aruba CX-model switches for the use of Telia Cygate Oy's specialists. The company manages a vast number of network switches, and manually updating these devices is unreasonably time consuming. With automation, the devices can be updated in bulk simultaneously by a single person.

The automation tool was programmed in Python using the Nornir automation framework. For development purposes a laboratory environment consisting of the company's network switches was utilised, allowing a free testing of the programme's performance.

The result of the development was a functional framework, which the automation team further refined into a version suitable for the company's practises. Some suboptimal implementation methods had to be used during development, due to a lack of deep programming experience. Not all functionalities targeted for the project were achieved, but the most essential functions were successfully implemented. By utilising the completed framework, the automation tool can also be easily further developed for the other types of switches.

Keywords	Network automation, Python, Nornir
Pages	24 pages

Sisällys

Kuvat

Liitteet

1 Johdanto

Opinnäytetyön tarkoitus on kehittää automaatiotyökalu Aruba CX -verkkokytkimien ohjelmistopäivityksiä varten Telia Cygate Oy:n asiantuntijoiden käyttöön. Yrityksen käytössä on valtava määrä verkkokytkimiä, joiden manuaalinen päivittäminen vie paljon aikaa. Automaation avulla ohjelmistopäivitys voidaan ajaa massana usealle laitteelle yhden henkilön toimesta. Automaation käyttö tekee myös yrityksestä houkuttelevamman asiakkaille ja se ehkäisee ihmisen tekemiä virheitä. Opinnäytetyössä tutkitaan verkkoautomaation hyötyjä sekä yleisimpiä kompastuskiviä ja rakennetaan lopuksi käytännön toteutus.

Verkkoautomaatiolla tarkoitetaan tietoverkkojen käyttöönoton, hallinnan ja valvonnan automatisointia erilaisten ohjelmistojen, skriptien eli komentosarjojen ja työkalujen avulla. Automaation avulla voidaan toteuttaa esimerkiksi verkkolaitteiden valvontaa tai massana ajettavia konfiguraatiomuutoksia. Valvonta-automaatio tarkistaa laitteen tilatietoja jatkuvasti ja ongelmien sattuessa niihin päästään reagoimaan ripeästi. Jos konfiguraatiomuutos täytyy tehdä usealle laitteelle, on se paljon nopeampaa antaa automaation hoidettavaksi.

Verkkokytkimiä on paljon erilaisia ja tässä opinnäytetyössä rakennetaan automaatio ainoastaan Aruba-laitevalmistajan CX-laitemallistolle. Rajaus on ollut pakko tehdä, koska eri laitemallien komennot ovat toisistaan erilaisia. Muille yrityksen käytössä oleville laitemalleille tullaan rakentamaan oma automaationsa muiden toimesta.

Työn lopputuloksena säästyy valtavasti työtunteja ja verkkolaitteiden päivittäminen tulee olemaan jatkossa selkeästi vaivattomampi prosessi. Tuotannossa olevia verkkolaitteita voi harvoin päivittää virka-aikana, joten päivittämisen voi joutua tekemään yöllä ja usein ylityönä. Automaation avulla päivityksen lopussa suoritettava laitteiden uudelleenkäynnistys voidaan tehdä ajastetusti haluttuna ajankohtana.

2 Automaatio

Yritysmaailmassa on paljon asioiden ja palveluiden tuottamista, jotka ovat syvemmin tarkasteltuna monen yksinkertaisen toimenpiteen jatkuvaa toistamista. Automaation tarkoituksena on tarttua näihin manuaalista työtä vaativiin, mutta jatkuvasti täysin samankaltaisiin ja ennalta määritettyihin tehtäviin ja suorittaa ne mahdollisimman vähällä ihmisen syötteellä. Automatisoimalla erilaisia tuotannollisia sekä hallinnollisia tehtäviä saadaan säästettyä merkittäviä määriä aikaa ja rahaa. (IBM, n.d.)

Jokaista yksittäistä prosessia ei kuitenkaan ole kannattavaa automatisoida, vaikka tähän olisi teoriassa valmiudet. Mikäli jokainen suoritettava prosessi on täysin uniikki, voi kaikkien poikkeamien huomioon ottaminen olla täysin mahdotonta. Automaatio tekee päätöksiä lähtökohtaisesti ennalta määritettyihin ohjeisiin perustuen ja tilanteeseen nähden väärän päätöksen tekeminen voi aiheuttaa enemmän kustannuksia, kuin automaatiolla olisi voitu säästää. (TORQ, 2022)

3 Verkkoautomaatio

Verkkoautomaatiolla tarkoitetaan tietoverkkojen käyttöönoton, hallinnan ja valvonnan automatisointia erilaisten ohjelmistojen, skriptien eli komentosarjojen ja työkalujen avulla (China, 2025.-a). Verkkolaitteen käyttöönotto vaatii aina konfiguraation eli asetusten määritystä. Yksinkertaisuudessaan konfiguraatiossa määritetään miten laite kommunikoi muiden tietoliikennelaitteiden kanssa. Jos saman konfiguraatiomuutoksen haluaa tehdä useammalle laitteelle, voi se olla järkevää toteuttaa automaatiolla. (China, 2025.-b)

Skripteillä rakennettu automaatio tarkoittaa sitä, että tyypillisesti Python- tai Bash-ohjelmointikielellä tehtyyn sovellukseen on listattu yksi tai useampi komento, jotka suoritetaan verkkolaitteen komentorivillä. Skriptin avulla näitä komentoja ei tarvitse suorittaa yhdellä laitteella yksi kerrallaan, vaan komennot voidaan ajaa massana usealla laitteella samanaikaisesti tai peräkkäin. (Swanson, 2023)

Automaatiotyökalut ovat skriptausta kehittyneempi vaihtoehto, mutta niiden käyttöönotto vaatii hieman enemmän työtä. Työkaluilla voidaan tulkita laitteen palauttamaa tekstiä helpommin, joten niillä voidaan rakentaa erilaisia työnkulkuja. Työnkuluilla voidaan havaita poikkeamia ja tunnistaa jos komennon suorittaminen epäonnistui laitteella. (Swanson, 2023)

Verkkoalustoja ovat esimerkiksi Intent-Based Networking (IBN), eli lopputuloksen kautta suunnitellun verkon lähestymistapaa hyödyntävät palvelut, kuten Cisco DNA Center. Työkalun tarkoituksena on suunnitella vain haluttu lopputulos ja antaa automaation huolehtia itse toteutuksesta. (Nefkens, 2020) IBN sisältää toteutuksen lisäksi tyypillisesti myös aktiivisen analytiikan seurannan ja sen kautta korjaavien toimenpiteiden automaattisen suorittamisen (Juniper, n.d.).

API-pohjainen ratkaisu toimii käytännössä samoin kuin skripteillä tehty automaatio, mutta komentorivin sijaan käskyt lähetetään laitteen ohjelmistorajapintaan HTTP(S)-protokollan läpi. Ohjelmistorajapinta mahdollistaa vakioidun väylän automaatiotyökalujen ja verkkolaitteiden ohjelmalliselle vuorovaikutukselle (Layer8Packet, 2024).

3.1 Konfiguraation määritys ja muutokset

Automaation tavoitteena on tehdä verkkojen hallinnasta tehokkaampaa ja luotettavampaa. Laitteiden konfiguraation määrittäminen automaatiolla käyttäen valmiiksi laadittuja pohjia takaa sen, että kaikkialla on käytössä yhtenäiset politiikat ja vältytään ihmisen tekemiltä huolimattomuusvirheiltä. (China, 2025.-a)

Jos useaan laitteeseen halutaan sama konfiguraatiomuutos, on se järkevää antaa automaation hoidettavaksi. Kuvitellaan tilanne, jossa sataan verkkokytkimeen tarvitaan muutos, joka vie 30 sekuntia jokaista laitetta kohden. Tehtävän suorittamiseen kuluu yhdeltä henkilöltä 50 minuuttia, kun automaatio hoitaisi sen hetkessä. Esimerkiksi jos automaatio suorittaa tehtävän viidelle laitteelle kerrallaan ja aikaa kuluu seitsemän sekuntia laitetta kohden, on muutos ajettu kaikille laitteille alle kolmessa minuutissa.

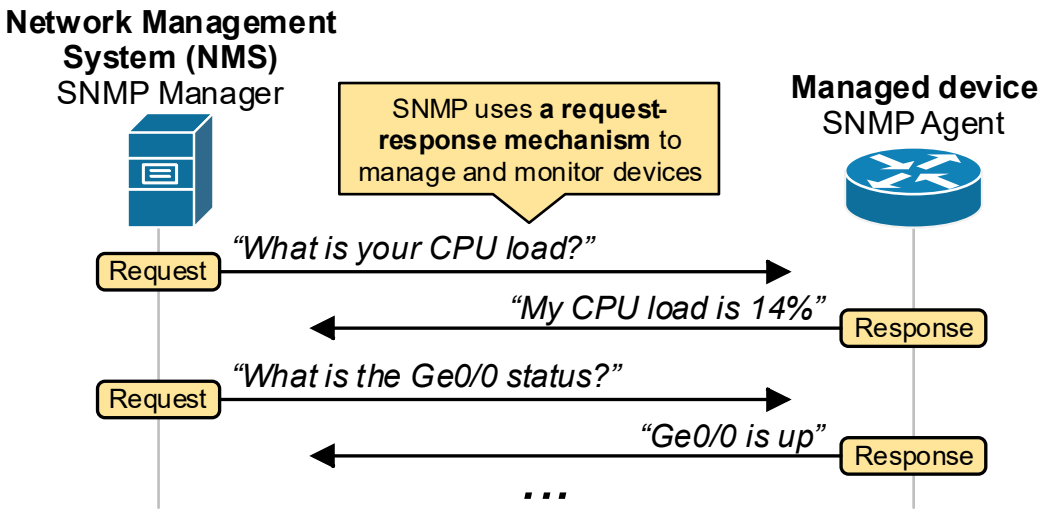
3.2 Valvonta

Myös verkkolaitteiden valvonta voidaan rakentaa automaatiolla. Tietoliikennelaitteilta voidaan kerätä suorituskykytilastoja ja havaita poikkeamia sekä virheitä laitteen

toiminnassa. (China, 2025.-a) Valvonnalla tarkoitetaan tietoverkossa käytettävien laitteiden ja palveluiden jatkuvaa toiminnan tarkkailua ongelmien varalta (VMware, n.d.).

Tietojen kerääminen onnistuu kätevästi esimerkiksi SNMP-kyselyillä. SNMP eli Simple Network Management Protocol on sovellustason tietoliikenneprotokolla, jonka avulla voidaan kysyä laitteen tietoja. SNMP-valvonta on yksinkertaistettuna vuoropuhelua palvelimen SNMP-hallintaohjelman ja valvottavan laitteen SNMP-agentin välillä. Kysymällä proaktiivisesti tietoja laitteen suorituskyvystä ja poikkeamista, päästään ongelmien ilmetessä hyvin nopeasti aiheuttajan jäljille. SNMP-kyselyitä voi käyttää ilmankin automaatiota, mutta parhaan ja reaaliaikaisen kuvan verkon toiminnasta saa kysymällä tietoa jatkuvasti. Automaation virkaa toteuttaa tässä tapauksessa SNMP-hallintaohjelma, jonka avulla kysytään kaikkien valvontaan asetettujen laitteiden tilatietoja tasaisin väliajoin. Esimerkiksi kuvassa 1 kysytään laitteelta prosessorin käyttöastetta sekä verkkoportin Ge0/0 tilatietoa. (Network Academy, n.d.)

Kuva 1 SNMP-kyselyn esimerkki.



3.3 Haasteet

Verkkoautomaatio ei kuitenkaan ole vastaus kaikkeen ja ennen sen käyttöönottoa on monta asiaa, jotka tulisi ottaa huomioon. Kehittämiselle olisi hyvä olla oma nimetty joukko, jotka pääsevät tarpeen vaatiessa analysoimaan ja korjaamaan automaatiossa tapahtuneet virheet.

Vanhoja järjestelmiä voi olla vaikea saattaa automaation piiriin puutteellisien ominaisuuksien takia. Niiden dokumentaatio voi olla olematonta tai niihin ei ole saatavilla ajankohtaisia tietoturvallisia ratkaisuja. Laitteiden käskyttäminen voi koitua haastavaksi, jos ne eivät myöskään tue nykyajan automaatiotyökaluja. Yritys voi joutua rakentamaan itse räätälöidyt skriptit vanhoille laitteille, joiden ohjelmoimiseen ja ylläpitämiseen voi mennä paljon resursseja. (Robinette, 2025)

Automaation käyttöönotto ei ole aivan yksinkertainen toimenpide, vaan sen eteen joutuu näkemään paljon vaivaa. Huolimattomasti tehdyt toimenpiteet voivat avata haavoittuvuuksia, joita rikolliset pääsevät hyödyntämään verkkoon tunkeutumisessa tai laitteisiin asetetaan vääriä konfiguraatioita, jotka pahimmillaan tekevät verkosta käyttökelvottoman. Muutoksien suorittaminen verkossa vaatii yleensä paljon käyttöoikeuksia, joten ne löytyvät luonnollisesti myös automaation hallusta. Laajat oikeudet järjestelmiin ovat houkutteleva saalis rikollisille, joten automaation tekemistä toimenpiteistä on pysyttävä tietoisena samalla tavalla, kuin muidenkin verkon käyttäjien. (Robinette, 2025)

Skaalautuvuus on yksi automaation avaintekijöistä, mutta se voi toimia myös kompastuskivenä. Kehnosti suunnitellut työnkulut ja keskitetyn hallinnan puute voivat ajan kuluessa tuoda esiin samoja ongelmia, joihin automaation kuului olla ratkaisuna. Mitä monimutkaisempi arkkitehtuuri on kyseessä, sitä vaikeammaksi toimivan automaattisen skaalautumisen kehittäminen on. Huomiota pitäisi kiinnittää erityisesti pilvi- ja hybridipohjaisiin sekä useamman toimialueen ratkaisuihin. (Robinette, 2025)

3.4 Automaation kehitysprosessi

Automaation kehittäminen seuraa pitkälti samoja proseduureja, kuin minkä vain sovelluksen kehitys. Oman ympäristön automatisointi voi olla järkevää aloittaa pienistä ja usein toistuvista tehtävistä, jotka tuottavat hyvää lisäarvoa jo olemassa oleviin toiminteisiin. Ennen kuin kehitetty automaatio saadaan valjastettua käyttöön, tulisi sen toimintaa testata perusteellisesti. Verkkoautomaation testausta varten tulisi perustaa oma testiympäristönsä, jonka tulisi pitää sisällään samoja laitteita kuin tuotannollisessa käytössä. Laitteiden hankintakustannus voi tietenkin olla merkittävä riippuen käytössä olevasta laitteistosta, mutta sen tekeminen on usein välttämätön osa automaation rakentamista. (Uneze, 2024)

Automaation tekeminen perustuu aina johonkin tarpeeseen. Monesti voisi ajatella, että kaiken mahdollisen automatisoimalla päästään eniten resursseja säästävään

lopputulokseen. Kuitenkin, jos jonkin toistuvan toimenpiteen tekemiseen kuluu vain pieni hetki, ei se välttämättä vielä riitä perusteeksi automaation rakentamiseen taloudellisesta näkökulmasta tarkasteltuna. Automaation potentiaalinen säästö voi olla niin pieni, että sen vaikutuksen huomaa vasta pitkällä tähtäimellä. Jokaista automaatiota täytyy lähtökohtaisesti kuitenkin ylläpitää, joten mahdolliset säästöt voivat valua täysin ylläpidollisiin kuluihin.

Tarveharkinnan jälkeen voidaan siirtyä suunnittelemaan automaation toteutusta. Toteutuksessa käytettävät työkalut voivat vaihdella riippuen siitä, millaista automaatiota ollaan rakentamassa. Esimerkiksi aiemmin mainittua Cisco DNA Center -ratkaisua ei ole järkevää käyttää, jos käytössä on jonkin toisen valmistajan laitteet. Suunnitelmassa olisi hyvä ottaa huomioon mahdollisesti aiemmin luodut automaatiot, jotta samankaltaiset projektit olisivat mahdollisimman yhtenäisiä, sillä niiden ylläpidettävyys helpottuu merkittävästi. Jotkin ratkaisut voivat käydä myös kalliiksi erinäisten lisenssimaksujen myötä.

Toteutusta varten on hyvä olla olemassa testiympäristö, jossa automaation testaus ei ole este normaalille verkon käytölle. Tuotantoon vientiä vaille valmiin projektin voi vielä testauttaa tuotantoympäristön ei kriittisellä alueella. Suoraan tuotannossa testaamista olisi hyvä välttää, sillä riskit käyttökatkoihin voivat olla automatisoidun kohteen mukaan kohtuuttoman suuret.

Tuotannossa olevaa automaatiota voi vielä kehittää uusilla ideoilla ja virheenkorjauksilla. Projektivaiheessa tapahtuneesta perusteellisesta testaamisesta huolimatta, tuotannossa voi tapahtua vielä jotain odottamatonta. Monesti myös joidenkin ominaisuuksien puute huomataan vasta todellisessa käytössä, ja niiden tarpeellisuutta ei osattu huomata projektin aikana.

4 Verkkoautomaatiokehykset

Verkkoautomaatiokehykset ovat työkaluja, jotka sisältävät automaation logiikan. Valmiin kehyksen avulla, käyttäjän ei tarvitse huolehtia kuinka esimerkiksi laitteen antamia tulosteita tulkitaan ja milloin laite on valmiina vastaanottamaan seuraavan komennon. Kehyksen avulla voidaan siirtyä lähes suoraan laitteen konfiguroimiseen. Tarkastellaan seuraavaksi kahta erilaista tämän opinnäytetyön projektiin soveltuvaa vaihtoehtoa.

4.1 Nornir

Nornir on Python-ohjelmointikielellä kehitetty automaatiokehys, joka mahdollistaa rinnakkain suoritettavan verkkoautomaation ilman työkalulle erikseen kehitettyä täsmäkieltä. Nornir pitää huolen verkkolaitteiden inventaariosta, jonka avulla on helppo määrittää automaatiolle tarkoitetut kohteet. Automaation työnkulusta pidetään kirjaa, josta nähdään työn toteutuminen ja mahdolliset virheet laitekohtaisesti. Automaatio suoritetaan kaikille laitteille yksilönä, eli yksittäisen laitteen työn epäonnistuminen ei vaikuta muihin laitteisiin. (Dyks, 2023)

Laitekannan inventaariota voidaan ylläpitää haluamallaan tavalla, mutta yksinkertaisimmillaan se onnistuu SimpleInventory-liitännäisellä. Laitekanta rakentuu kolmella yaml-tiedostolla, joihin listataan laitteiden tietoja. Hosts.yaml-tiedostoon listataan kaikki laitteet, mihin ryhmään ne kuuluvat ja tarvittaessa laitekohtaisia yksilöiviä tietoja, jos sen määrittäminen ryhmälle tai globaalisti ei ole järkevää. Laitteilla käytetään vain kaikista täsmällisintä tietoa, eli yksilötasolla määritetty tieto ylikirjoittaa ryhmälle määritetyn tiedon. Kuvassa 2 on esimerkki hosts-tiedostosta, jossa listataan kolmen laitteen tiedot. (Dyks, 2023)

Kuva 2 hosts.yaml-tiedoston esimerkki.

```
router_1:
  hostname: 192.168.0.83
  port: 22
  groups:
   - site_1
  data:
    type: dmz
router_2:
  hostname:  192.168.0.32
  groups:
   - site_1
  data:
      type: dmz
router_3:
  hostname: 192.168.0.254
  groups:
    - site_2
  data:
      type: wan
```

Laitteille on mahdollista määrittää myös muita tietoja, kuten kirjautumiseen käytettävä tunnus ja käyttöjärjestelmä, mutta ne on parempi määrittää laajemmalla skaalalla ylläpidettävyyden helpottamiseksi. Seuraavaksi tietoja määritetään ryhmäkohtaisesti groups-tiedostolla. Ryhmätasolla on kätevä määrittää asetuksia esimerkiksi toimipisteittäin tai laitevalmistaja kohtaisesti. Automaatiota voidaan tarvittaessa myös suorittaa vain tietylle ryhmälle. Esimerkiksi kuvassa 3 määritetään eri aikapalvelin kullekin toimipisteelle. (Dyks, 2023)

Kuva 3 groups.yaml-tiedoston esimerkki.

```
global:
    groups:
     - site_1
     - site_2
    data:
      ntp: 3.3.3.3
site_1:
    data:
      ntp: 1.1.1.1
site_2:
    data:
      ntp: 2.2.2.2
```

Mikäli laite ei kuulu mihinkään ryhmään, sille asetetaan tässä esimerkissä globaalisti määritetty aikapalvelin. Viimeisenä on defaults-tiedosto eli oletustiedot, jotka ovat käytössä kaikilla laitteilla, ellei niitä korvata täsmällisemmällä tiedolla ryhmä- tai yksilötasolla. Kuvassa 4 on esimerkki defaults-tiedostosta.

Kuva 4 defaults.yaml-tiedoston esimerkki.

```
username: admin
password: admin
platform: ios
port: 22
connection_options:
  netmiko:
    extras:
      secret: admin
```

Kaikilla laitteilla on tässä esimerkissä samat käyttäjätunnukset ja verkkoportti etäyhteyttä varten, joten ne voidaan määrittää oletukseksi. Platform, eli käyttöjärjestelmä on oleellinen tieto, sillä sen avulla Nornir kykenee tulkitsemaan laitteiden antamia tulosteita paremmin. Viimeisenä tarvitaan vielä konfiguraatiotiedosto, jossa osoitetaan aiemmin mainittujen

tiedostojen sijainnit ja määritetään muun muassa käytettävä inventaario. Kuvassa 5 on esimerkki konfiguraatiotiedostosta. (Dyks, 2023)

Kuva 5 config.yaml-tiedoston esimerkki.

```yaml
inventory:
  plugin: SimpleInventory
  options:
    host_file: "inventory/hosts.yaml"
    group_file: "inventory/groups.yaml"
defaults_file: "inventory/defaults.yaml"
runner:
  plugin: threaded
  options:
    num_workers: 10
```

Konfiguraatiotiedostossa määritetään myös runner-toiminteen asetukset. Threaded tarkoittaa sitä, että automaatiota ajetaan laitteille samanaikaisesti ja alempana oleva työläisten määrä kertoo kuinka monelle. (Dyks, 2023)

Nyt esimäärittelyt ovat valmiit ja Nornir voidaan ottaa käyttöön. Käyttö tapahtuu tuomalla Python-ohjelmistokirjastot mukaan ohjelmaan ja alustamalla Nornir-kehyksen konfiguraatiotiedostolla, kuten kuvassa 6 näytetään. (Dyks, 2023)

Kuva 6 Nornir-automaatiokehyksen käyttö koodissa.

```python
from nornir import InitNornir
from nornir.core.task import Result, Task
from nornir_netmiko.tasks import netmiko_send_command, netmiko_send_config
from nornir_utils.plugins.functions import print_result

nr = InitNornir(
     config_file='nornir_config.yaml'
)
```

Nornir-automaatiokehykseen on saatavilla monia eri liitännäisiä, jotka hoitavat itse verkkolaitteille kommunikoinnin. Tähän tarkoitukseen voi käyttää esimerkiksi Netmiko-kirjastoa. Kyseistä kirjastoa voi käyttää myös itsenäisenä, mutta se hyötyy suuresti Nornir-automaatiokehyksen tarjoamista ominaisuuksista, kuten inventaariosta ja monisäikeisyydestä. Itsenäisesti ajettavana kohdelaitteet tulee kertoa aina erikseen tai vaihtoehtoisesti lisätä ne suoraan ohjelmakoodiin ja automaatio suoritetaan laite kerrallaan. (Jin, 2023)

Suoritetaan esimerkiksi kaikille inventaarion laitteille yksinkertainen show-komento, jolla nähdään laitteelle määritetyt bannerit. Kuvassa 7 näkyy ohjelmakoodi, joka hakee laitteilta kyseisen tiedon ja tulostaa tuloksen näkyviin laitekohtaisesti. (Dyks, 2023)

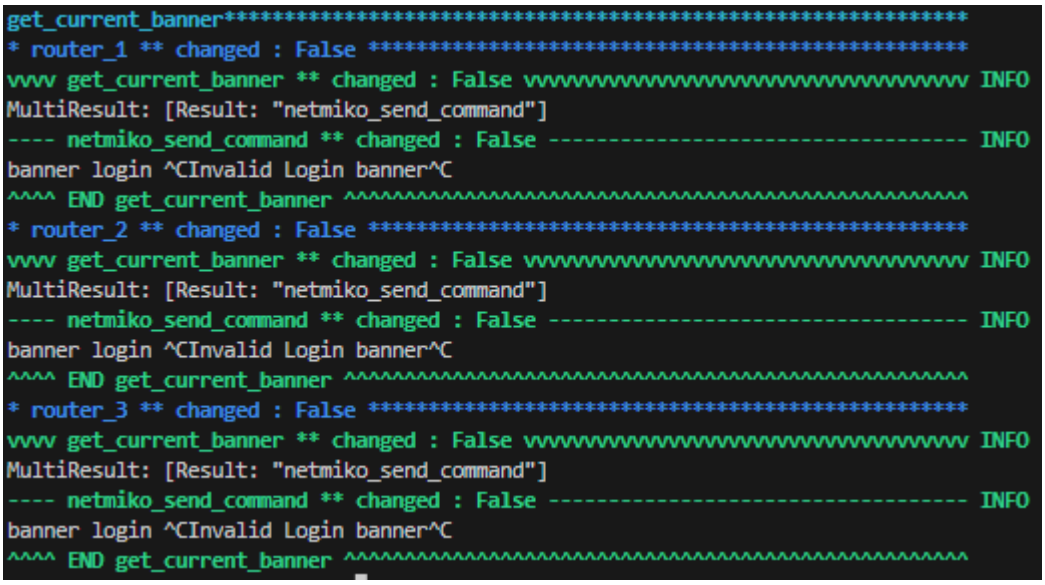
Kuva 7 Nornir ja Netmiko koodiesimerkki.

```
def get_current_banner(task: Task) -> Result:
    output = task.run(netmiko_send_command, command_string='show running-config | i banner' )
    return Result(host=task.host, result=output)

output = nr.run(task=get_current_banner)
print_result(output)
```

Verkkolaitteille suoritetaan Nornir-kehyksen tehtävä, joka pitää sisällään Netmiko-kirjaston komennon lähetys toiminteen. Lähetettävä komento ”show running-config | i banner” näyttää laitteen ajossa olevan konfiguraation ja sisällyttää tulosteeseen vain ne rivit, jotka sisältävät sanan ”banner”. Kuvassa 8 näkyy ohjelman suoritus ja tulokset laitekohtaisesti. (Dyks, 2023)

Kuva 8 Nornir ja Netmiko automaation suorituksen esimerkki.



Tulosteessa nähdään, että jokaisella laitteella on käytössä kirjautumisen yhteydessä näytettävä banneri, jossa lukee ”Invalid login banner” (Dyks, 2023).

4.2 Ansible

Ansible on avoimen lähdekoodin automaatiotyökalu, jonka avulla voidaan automatisoida muun muassa konfiguraatiomuutoksia, sovellusten asennuksia ja IT-orkestrointia. Automaatio rakentuu playbook-skriptien avulla, joihin määritetään haluttu lopputulos. Playbookit ovat yaml-tiedostoja ja ne ovat suunniteltu helposti luettavaksi, joka helpottaa niiden ymmärtämistä ja suunnittelua. Ansiblen arkkitehtuuri rakentuu erilaisista moduuleista, joiden avulla tehtäviä suoritetaan laitteilla. Moduuleja on paljon erilaisia ja jokaiselle toimenpiteelle on omansa, joista jokainen tietää mitä toimenpiteitä kunkin tehtävän suorittaminen vaatii. (Nichols, 2023)

Myös Ansible tukee monia eri vaihtoehtoja inventaarion perustamiselle, mutta yksinkertaisimmillaan se onnistuu yaml- tai ini-tiedostoilla. Esimerkiksi kuvassa 9 on lisätty kaksi reititintä ini-tiedostoon ja määritetty niiden kirjautumistiedot.

Kuva 9 Ansible inventaarion esimerkki.

```
[routers]
R77 ansible_host=192.168.1.77
R88 ansible_host=192.168.1.88

[routers:vars]
ansible_network_os=ios
ansible_user=cisco
ansible_password=cisco
ansible_connection=network_cli
ansible_become_password=cisco123
```

Tiedostoon on ensin määritetty ryhmä ”routers”, johon on lisätty kaksi laitetta ja niiden IP-osoitteet. Alempana oleva ”routers:vars” liittää haluttuja määrityksiä ryhmään, kuten tässä tapauksessa tunnistetiedot sekä yhdistämistavan etäyhteyttä varten. ”Ansible_become_password”-rivi tarkoittaa Cisco-laitevalmistajan reitittimillä korkeamman käyttöoikeuden salasanaa, joka tarvitaan konfiguraatiomuutoksia tehdessä. Tämä yksi tiedosto riittää inventaarioksi ja seuraavaksi voidaan luoda playbook-skripti. Esimerkiksi luodaan laitteille uusi portti kuvan 10 skriptillä. (Network Ninja, 2023)

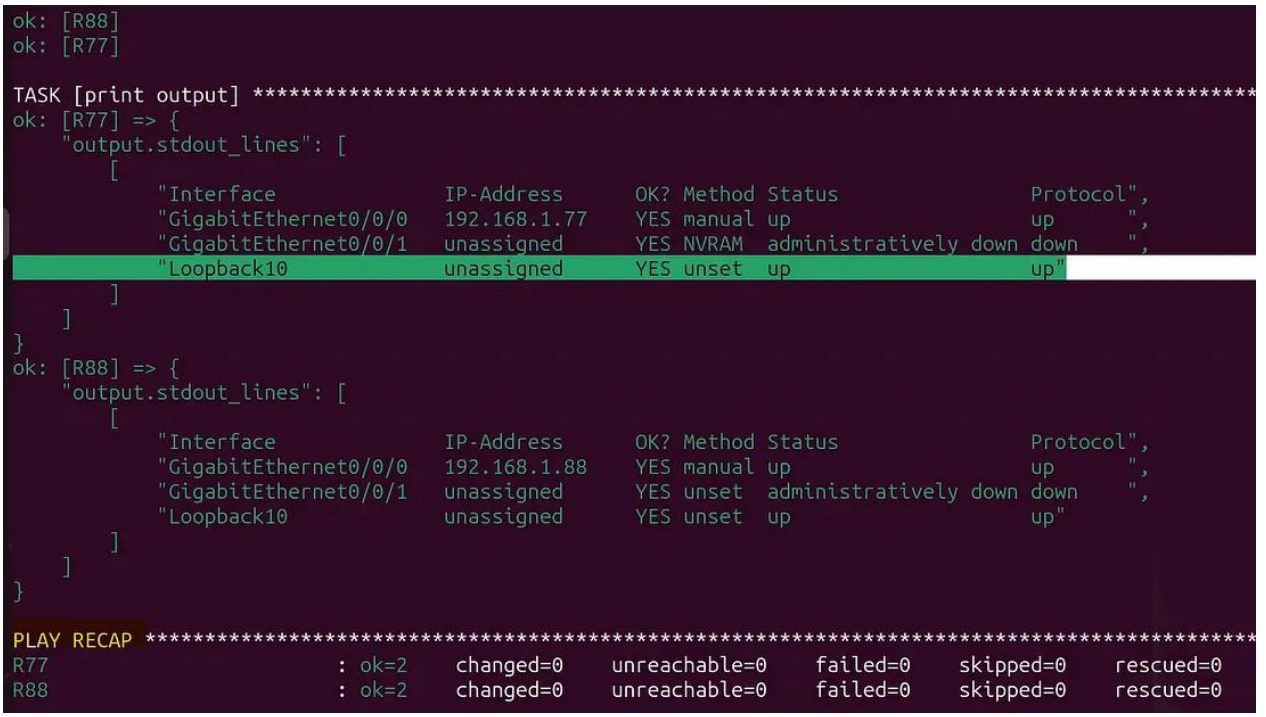
Kuva 10 Ansible playbook-skriptin esimerkki.

```
realninja@Einstein:~$ cat lab.yml
---
- hosts: routers
  become: yes

  tasks:
    - name: Create Lo10
      cisco.ios.ios_interfaces:
        config:
          - name: loopback 10
            description: Created By Ansible
            enabled: true
```

Tiedostoon on määritetty kohteeksi aiemmin luotu ”routers”-ryhmä ja annettu suorittamiselle tarvittavat käyttöoikeudet. Tehtävässä oleva ”cisco.ios.ios_interfaces” on automaation käyttämä moduuli. Moduuli on tehty Cisco-laitevalmistajan laitteiden porttikonfiguraatioita varten, joten se osaa tulkita laitteen antamia tulosteita ja antaa juuri oikeat syötteet halutun lopputuloksen saavuttamiseksi. Laitteille luodaan virtuaalinen loopback-portti, annetaan sille kuvaus ja asetetaan sen tila ylös. Kuvassa 11 nähdään automaation tekemät toimenpiteet ja tieto suoriutumisen onnistumisesta. (Network Ninja, 2023)

Kuva 11 Ansiblen automaation suorittaminen.



Tulosteessa nähdään lista laitteen porteista uuden portin luonnin jälkeen ja onnistumisen tilatieto, joka tässä tapauksessa on molemmilla ”ok”. Kun kaikki laitteet on käyty läpi, nähdään vielä yhtenäisesti suorituksen onnistuminen laitekohtaisesti. (Network ninja, 2023)

5 Tietoturva

Automaation tietoturvan olisi hyvä nojata luotettuihin ohjeisiin ja standardeihin, kuten least privilege -periaatteeseen tai NIST SP 800-53 -standardiin. Least privilege tarkoittaa käytäntöä, jossa käyttäjien pääsy eri järjestelmiin on oletuksena estetty, ellei työtehtävä sitä erikseen vaadi (CyberArk, n.d.). NIST SP 800-53 -standardi on Yhdysvaltain standardisointi- ja teknologiainstituutin julkaisema luettelo tietojärjestelmien tietosuoja- ja suojausohjaimista, jotka auttavat turvaamaan salassa pidettäviä tietoja joutumasta vääriin käsiin (NIST, 2025).

Automatisoitujen järjestelmien tulisi käyttää vain turvallisia kirjautumismenetelmiä ja salattuja protokollia. Esimerkiksi pelkällä salasanalla kirjautuminen olisi hyvä estää ja sallia pääsy käyttäen SSH-avaimia. SSH, eli Secure Shell on suojattuun etäyhteyteen käytetty protokolla, jossa kaikki laitteiden välillä lähetetyt tiedot ovat salattuja. Protokolla mahdollistaa käyttäjän kirjautumaan etänä laitteelle käyttäen etukäteen luotuja

kryptografisia avaimia. Avaimien suojausta on äärimmäisen vaikea murtaa verrattuna perinteiseen salasanaan, joka tekee etäyhteyksien käytöstä turvallisempaa. (Ruostemaa, 2024)

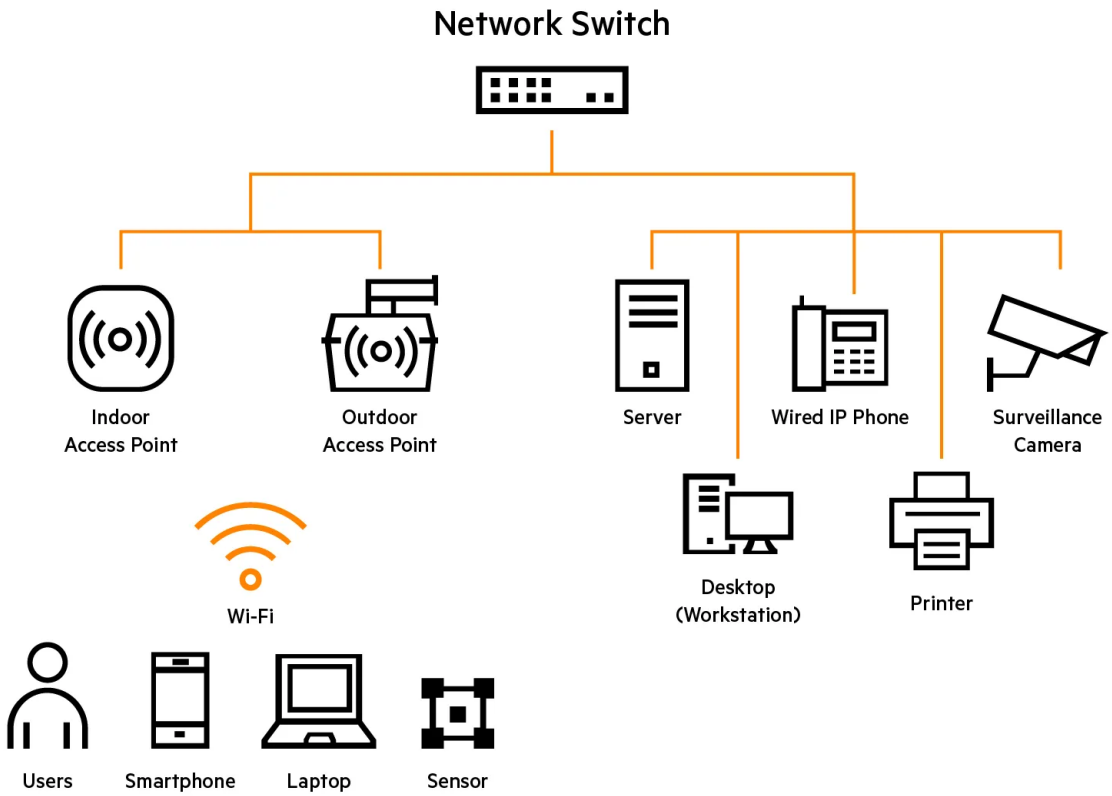
Automaatiolla tehtyjen toimenpiteiden alullepanijan henkilöllisyys pitäisi pystyä aukottomasti todistamaan. Automaation käytöstä pitäisi siis syntyä lokitietoja, josta käy ilmi kuka teki, mitä tehtiin ja milloin tehtiin. Lokitiedon mahdollistamiseksi, automaatiotyökaluja tulisi voida suorittaa vain todennetut käyttäjät. (Farin & Roy, 2021, s. 11)

Verkkoautomaation mahdollistamiseksi on työkalun käytössä usein oltava korkeilla käyttöoikeuksilla varustettu käyttäjä. Tämä tekee siitä kiinnostavan kohteen kyberrikollisille, sillä käyttäjän turvin pääsee tunkeutumaan laajalti verkon eri puolille. Tietoturvallisen automaation elinehto on tietoturvallinen palvelinympäristö. Mikäli automaatiota ylläpitävälle palvelimelle pääsee käsiksi, on myös automaation tietoturva vaarantunut. Riskin pienentämiseksi automaation tallentamia tietoja voi suojata salausalgoritmilla, jotta niihin ei pääse käsiksi pelkällä palvelinympäristön murtamisella. (Farin & Roy, 2021, s. 9-10)

6 Verkkokytkimet

Verkkokytkimet, eli kytkimet ovat tietoteknisiä laitteita, jotka yhdistävät eri verkkolaitteet osaksi verkkoa. Niiden pääasiallinen tehtävä on mahdollistaa verkon laitteiden välinen liikennöinti ja hallinnoida verkkoliikenteen kulkua. Kytkimet pitävät listaa verkon mac-osoitteista eli päätelaitteen yksilöivistä tunnisteista ja ohjaavat liikennettä oikeaa osoitetta kohti. Kytkimen tärkeimpiä ominaisuuksia on verkon jakaminen virtuaalilähiverkkoihin, joiden avulla yhden verkon laitteet voidaan sijoitella omiin sisäisiin verkkoihin. Näiden avulla voidaan eristää esimerkiksi kaikille avoin vierasverkko toimiston sisäisestä verkkoyhteydestä ja täten parantaa tietoturvaa. Kuvassa 12 havainnollistetaan tyypillinen lähiverkko, joka voisi olla käytössä esimerkiksi toimistossa. (HPE, 2025.)

Kuva 12 Lähiverkon esimerkki.



Kytkimiä on valtavasti erilaisia ja erilaisiin käyttötarkoituksiin. Verkkoa rakentaessa tulisikin miettiä, mitä ominaisuuksia ja minkälaista suorituskykyä tilanne vaatii. Oleellisia harkinnan kohteita ovat esimerkiksi verkkoporttien nopeus, verkkoliikenteen kapasiteetti, vikasietoisuus sekä Power over Ethernet -ominaisuus, joka mahdollistaa virransyötön verkkoportin kautta. (HPE, 2025.)

7 Aruba CX -kytkimen päivittämisen automatisointi

Opinnäytetyön tarkoituksena on rakentaa automaatio verkkokytkimien ohjelmistoversioiden päivittämiseen yrityksen asiantuntijoiden käyttöön. Eri kytkinmalleilla ja laitevalmistajilla on toisistaan poikkeava komentoliittymän syntaksi eli komennot rakentuvat eri tavalla. Tämän takia automaatioon ohjelmoidut käskyt eivät toimi kaikissa kytkinmalleissa. Tässä projektissa käsitellään vain Aruba CX -mallisien kytkimien päivittämistä. HPE Aruba Networking on Yhdysvaltalaisen Hewlett Packard Enterprise -organisaation alaisuuteen kuuluva tytäryhtiö, joka tarjoaa kehittyneitä yritystason verkkoratkaisuja (HP, 2015).

Projektin tavoitteena on luoda runko kytkinpäivitysten automaatiolle, jonka pohjalta automaatiotiimi voi jatkokehittää yrityksen standardeihin sopivan tuotantoversion ja tehdä työkalun toimivaksi myös muille käytössä oleville kytkinmalleille. Edistymistä tarkastellaan palaverissa kerran viikossa.

7.1 Vaatimukset

Automaation suoriutumisen kannalta tärkeää on se, ettei laitteelle voida asettaa sille vääränlaista ohjelmistoa. Ohjelmiston sopivuus tulisi jollain tavalla varmistaa oikeaksi virhetilanteiden välttämiseksi.

Uusi ohjelmisto otetaan verkkolaitteilla käyttöön uudelleenkäynnistyksen yhteydessä. Käyttäjän tulisi voida määrittää kytkimet uudelleenkäynnistymään heti tai haluttuna ajankohtana.

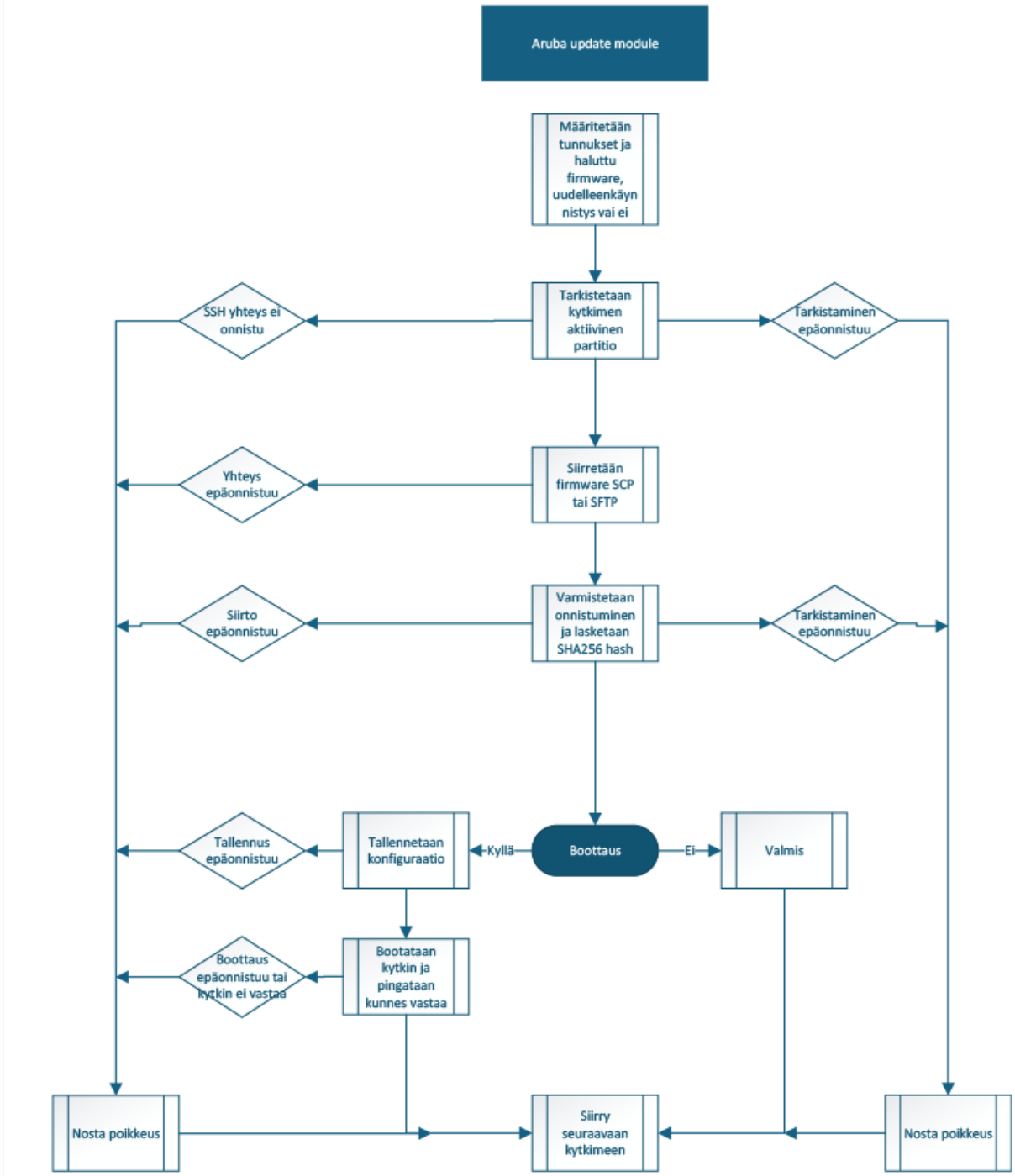
Työkalun ei pidä ottaa kantaa, onko asennettava ohjelmisto uudempi tai vanhempi kuin laitteella jo valmiiksi oleva. Tämä voidaan välttää helposti jättämällä sen ohjelmointi pois toteutuksesta.

7.2 Toteutus

Työkalu on rakennettu Python-ohjelmointikielellä käyttäen Nornir ja Netmiko -kirjastoja. Ohjelma muodostaa yhteyden verkkolaitteelle SSH-protokollaa käyttäen ja olettaa komentoliittymän syötteeseen tulevan tietynlaisia ennalta määritettyjä tulosteita. Tulosteiden avulla ohjelma kykenee siirtymään toimenpiteissä eteenpäin täysin itsenäisesti. Automaation toimintaperiaate on esitetty kuvassa 13.

Automaation testausta varten käytössä oli useasta Aruba CX -kytkimestä koostuva laboratorioympäristö, jossa suoriutumista sai kokeilla täysin vapaasti kellonajasta riippumatta. Vapaa käyttö mahdollisti projektin nopean etenemisen, kun laitteiden uudelleenkäynnistämiselle ei tarvinnut pyytää erikseen lupaa.

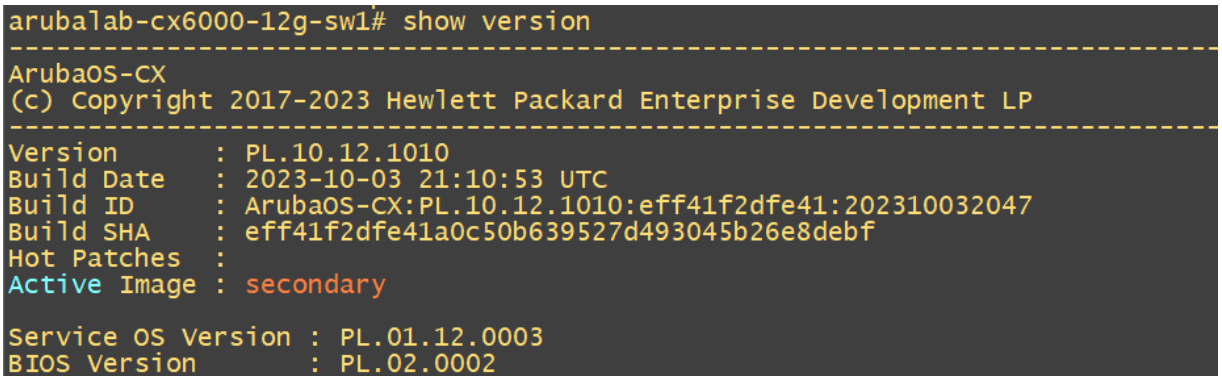
Kuva 13 Vuokaavio.



7.2.1 Aktiivisen laiteohjelmiston selvitys

Aruba CX -kytkimissä on kaksi eri lohkoa laiteohjelmistoille. Ensimmäinen vaihe on selvittää, kumpi näistä on tällä hetkellä käytössä ja valita kohteeksi päinvastainen lohko. Laitteelle lähetetään komento ”show version”, joka tulostaa kytkimen käytössä olevan laiteohjelmiston version sekä tiedon, kumpi lohko on tällä hetkellä käytössä. Kuvassa 14 on esimerkki ”show version” tulosteesta.

Kuva 14 show version -tuloste.



Tekstistä suodatetaan halutut merkkijonot käyttäen säännöllisiä lausekkeita. Ohjelman suorittamisen kannalta oleellista on vain aktiivisen laiteohjelmiston tila, joten versiotiedot haetaan ainoastaan tiedon rikastamiseksi. Kytkimet jaetaan kahteen eri listaan, riippuen kumpi laiteohjelmistoista on aktiivisena.

7.2.2 Ohjelmiston lataus

Laiteohjelmisto ladataan kytkimelle lähettämällä sille komento, jolla muodostetaan yhteys palvelimeen ja kopioidaan siellä oleva tiedosto epäaktiivisen laiteohjelmiston tilalle. Komento voisi olla esimerkiksi ”copy scp://admin@192.168.1.10/ArubaOS-CX_6100-6000_10_12_1010.swi secondary”, jossa kirjaudutaan admin-tunnuksella 192.168.1.10 IP-

osoitteeseen, ladataan sieltä ”ArubaOS-CX_6100-6000_10_12_1010.swi” niminen tiedosto ja asetetaan se kytkimen secondary-lohkoon.

Automaation ohjelmiston lataamisen funktio tarkistaa ensin kumpi laiteohjelmistoista on aktiivinen ja rakentaa aiemmin mainitun komennon. Tämän jälkeen yritetään aloittaa tiedoston lataaminen SCP-protokollaa käyttäen ja sen epäonnistuessa yritetään latausta myös SFTP-protokollalla. Mikäli kumpikaan ei onnistu, siirrytään seuraavaan laitteeseen.

Kun yhteys on muodostettu onnistuneesti, kytkin pyytää palvelimen SSH-avaimen hyväksyntää, mikäli sitä ei löydy kytkimen tiedossa olevista avaimista. Automaatio hyväksyy avaimen ja antaa sitten salasanan. Mikäli avainta ei kysytä, ohjelma aikakatkaistaan ja siirrytään suoraan salasanan syöttämiseen. Jos SSH-avaimen hyväksynnässä tai salasanan syöttämisessä tulee ongelmia, siirrytään seuraavaan laitteeseen.

Kun oikea salasana on syötetty, alkaa laiteohjelmiston lataaminen palvelimelta. Kytkin tulostaa samalle riville latauksen etenemisen prosenttina ilmaistuna. Automaatio on laitettu tarkistamaan latauksen eteneminen kymmenen sekunnin välein. Ohjelma ei pysty lukemaan samaa riviä montaa kertaa, joten kytkimelle lähetetään tyhjä rivi, jonka avulla prosentti tulostuu uudelle riville ja sitä voidaan lukea. Kuvassa 15 näkyy kytkimen tulostama latauksen aloittaminen ja sen edistyminen.

Kuva 15 laiteohjelmiston lataus

Latauksen jälkeen kytkin tarkistaa laiteohjelmiston eheyden, jonka edistymisen näkee ”show images” komennon tulosteessa. Automaatiota kehittäessä oli haastavaa tunnistaa missä vaiheessa kytkin siirtyy ohjelmiston lataamisesta sen tarkistamiseen, joten rakensin siihen hieman epäoptimaalisen ohituksen. Jos prosenttia ei enää löydy tulosteesta niin automaatio asettaa muuttujalle arvon manuaalisesti, jonka ohjelma tunnistaa ja tietää siirtyä eteenpäin. Kun eheyden tarkistaminen on valmistunut onnistuneesti, lasketaan laiteohjelmistolle vielä varmuuden vuoksi tarkistussumma. Tämä on sinänsä turha toimenpide, sillä kytkin tekee sen itsekin, mutta toiminteen lisääminen oli osana

alkuperäistä suunnitelmaa. Kuvassa 16 on ”show images” komennon tuloste eheyden tarkistamisen aikana.

Kuva 16 show images -tuloste.

```
arubalab-cx6000-12g-sw1# show images
--------------------------------------------------------------------------------
ArubaOS-CX Primary Image
--------------------------------------------------------------------------------
Version : PL.10.12.1010
Size    : 317 MB
Date    : 2023-10-03 21:10:53 UTC
SHA-256 : Verifying 10%

--------------------------------------------------------------------------------
ArubaOS-CX Secondary Image
--------------------------------------------------------------------------------
Version : PL.10.12.1010
Size    : 317 MB
Date    : 2023-10-03 21:10:53 UTC
SHA-256 : 3da4a34cc7c95ccf2e55746912bf60d5f1744d9aa8b11d4191f65e77755d3a1e

Default Image : primary
Boot Profile Timeout : 2 seconds

------------------------------------------------------
Management Module 1/1 (Active)
------------------------------------------------------
Active Image       : secondary
Service OS Version : PL.01.12.0003
BIOS Version       : PL.02.0002
```

7.2.3 Uudelleenkäynnistys

Mikäli ohjelman suorituksen alussa määritettiin uudelleenkäynnistys suoritettavaksi, tehdään se onnistuneen laiteohjelmiston latauksen jälkeen. Kytkimen konfiguraatio tallennetaan ja laite käynnistetään uudelleen käyttäen uutta laiteohjelmistoa. Jos konfiguraation tallennus epäonnistuu, ei uudelleenkäynnistystä tehdä. Kytkimelle lähetetään ping-paketteja, kunnes se nousee takaisin verkkoon, jonka jälkeen siirrytään seuraavaan laitteeseen. Automaatio ei ota kantaa siihen missä järjestyksessä kytkimet ovat kytkettynä verkkoon, joten ne käynnistetään uudelleen yksi kerrallaan. Tämä tehdään sen takia, että kytkimiä on usein monta peräkkäin ja yhteydet menetetään kaikkiin kytkimessä kiinni oleviin laitteisiin uudelleenkäynnistämisen aikana.

On huomattu, että kyseisen kytkinmalliston uudelleenkäynnistys saattaa kestää jopa kymmenen minuuttia, joten on tärkeää asettaa automaatiolle riittävän pitkä odotusaika ennen aikakatkaisua. Kytkin saattaa käynnistyä odotettua hitaammin ja automaatio

merkitsee päivityksen epäonnistuneeksi, jonka vuoksi laitteen tila tulee tarkastaa manuaalisesti jälkikäteen.

7.3 Automaation suorittaminen

Automaation suorittaminen yhdelle laitteelle on esitetty kokonaisuudessaan liitteessä 1. Aluksi kerrotaan ohjelmalle etäyhteydessä käytettävä tunnus sekä osoitetaan miltä palvelimelta haetaan määritellyn niminen laiteohjelmistotiedosto. Valitaan uudelleenkäynnistys toteutettavaksi, jonka jälkeen automaatio aloittaa itsenäisen suorittamisen. Ohjelma tulostaa komentoriville tilatietoja päivityksen etenemisestä ja sulkeutuu itsestään kytkimen vastatessa ping-pakettiin.

Automaation etenemistä on kohtuullisen vähän mitään konkreettista esitettävää, mutta komentorivin tulosteet tuovat hyvin esille sen, mitä käytännössä tapahtuu. Vaikka päivitys on mahdollista suorittaa usealle laitteelle samanaikaisesti, tehdään se selkeyden kannalta tässä esimerkissä vain yhdelle laitteelle.

7.4 Jatkokehitys

Automaatio on rakennettu kohtuu lyhyellä perehtymisellä Nornir-automaatiokehykseen, joten siinä on varmasti parantamisen varaa. Kytkimeltä saatavia tietoja suodatetaan monella jos-lausekkeella, vaikka Nornir kykenisi jo itsessään parempaan tulosteen tulkintaan. Ohjelmaan voisi implementoida tietokannan, jonka avulla onnistuneesti päivitetyt laitteet voitaisiin käynnistää uudelleen omana ajonaan. Tällä hetkellä kyseinen tieto katoaa ohjelman sulkeutuessa. Ohjelma tulostaa palvelimen komentoriville jokaisen laitteen edistymisen samaan aikaan, joka ei selkeyden kannalta ole ollenkaan järkevää. Parempi ratkaisu olisi tulostaa automaation onnistuminen jokaisen laitteen kohdalla yhdelle riville ja tallentaa edistyminen omaan lokitiedostoonsa, jota voi jälkikäteen tarkastella esimerkiksi ongelmatilanteiden selvityksessä.

Automaation toimintaa on testattu ekstensiivisesti laboratorioympäristössä, jossa ulkoiset muuttujat eivät ole päässeet vaikuttamaan lopputulokseen. Ohjelmaan on rakennettu suuri määrä virheentarkistuksia, jotka pysäyttävät suorittamisen välittömästi ongelmien ilmetessä, mutta vielä ei ole tiedossa miten esimerkiksi epävakaa yhteys vaikuttaa automaation suoriutumiseen.

8 Yhteenveto

Verkkoautomaatio on äärimmäisen tärkeä työkalu tämän päivän yritysmaailmassa. Erilaisten toimenpiteiden automatisointi on erittäin kustannustehokasta ja se vapauttaa asiantuntijoiden aikaa muihin tehtäviin. Kehitysprosessi olisi hyvä aloittaa jostain tarpeesta ja suunnitella huolella, sillä huolimattomasti tehdyt toteutukset voivat aiheuttaa paljon harmia esimerkiksi sotkemalla laitteiden konfiguraatiot tai mahdollistamalla kyberrikollisille pääsyn verkkoon. Automaation rakentamiselle on olemassa laaja kirjo erilaisia työkaluja sekä palveluita. Ylläpidettävyyden kannalta selvästi vaivattomin ratkaisu olisi käyttää laitevalmistajien tarjoamia pilvihallintaratkaisuja, joiden avulla verkkolaitteiden konfiguraatioita ja ohjelmistoversioita voidaan hallita kootusti erilaisten sapluunojen avulla. Niiden käyttö on kuitenkin maksullista ja laitemäärän kasvaessa myös lisenssimaksut nousevat. Lisenssimaksut eivät kuitenkaan ole välttämättömiä automaation käytölle ja ne voidaan välttää kehittämällä automaatiot itse käyttäen esimerkiksi tarjolla olevia avoimen lähdekoodin tuotteita, kuten Nornir-automaatiokehystä.

Opinnäytetyön tavoitteena oli rakentaa Aruba CX -verkkokytkimen ohjelmistopäivityksen automaatio, joka onnistui kokonaisuudessaan ihan hyvin. Projektin aloituksessa asetettuihin tavoitteisiin ei täysin päästy, sillä laitteiden uudelleenkäynnistämistä ei ole mahdollista ajastaa haluttuun ajankohtaan. Kyseisen ominaisuuden olisi saanut lisättyä esimerkiksi tietokannan avulla, johon voitaisiin tallentaa uusi laitekanta onnistuneesti päivitetyistä laitteista. Uutta laitekantaa vasten voitaisiin suorittaa oma ajonsa pelkästään laitteiden uudelleenkäynnistykselle. Syvemmän ohjelmoinnin kokemuksen puutteen vuoksi, projektin jotkin ratkaisut voivat olla hieman epäoptimaalisia, mutta kokonaisuus toimii testausten perusteella luotettavasti.

Alkuperäinen tarkoitus oli olla myös itse osana yrityksen standardeihin soveltuvan version kehityksessä, mutta ajan puutteen vuoksi en siihen osallistunut. Yrityksen automaatiotiimi on saanut jatkokehitettyä rakentamastani rungosta varsin hyvän yrityksen standardeihin sopivan version, johon eräskin asiakas on koenäytösten perusteella ollut erittäin tyytyväinen. Automaation ansiosta tullaan säästämään valtavasti asiantuntijoiden työtunteja, kun tärkeät päivitykset voidaan ajaa massana yhden henkilön voimin.

Projektissa aikaansaadut tulokset hyödyttävät yritystä nopeuttamalla verkkokytkimien laiteohjelmistojen päivityksen prosessia ja tekemällä siitä selvästi vaivattomamman asiantuntijoille. Automaatiotiimin vetäjä on ollut tyytyväinen tekemääni projektiin ja olen saanut hyvää palautetta myös muilta yrityksen tahoilta.

Lähteet

China, C. (14.3.2025.-a). *What is network automation?*

https://www.ibm.com/think/topics/network-automation

China, C. (7.1.2025.-b). *What is network configuration?*

https://www.ibm.com/think/topics/network-configuration

Cisco, (19.1.2023). *Industrial Automation Security Design Guide 2.0.*

https://www.cisco.com/c/en/us/td/docs/Technology/industrial-automation-security-design-guide/m-introduction.html

CyberArk, (n.d.). *What is Least Privilege?*

https://www.cyberark.com/what-is/least-privilege/

Dyks,M. (17.8.2023). *Python Nornir for simplifying network automation: code examples.*

https://codilime.com/blog/python-nornir-for-network-automation-code-examples/

Dyks,M. (17.8.2023). *Python Nornir for simplifying network automation: code examples [kuva].*

https://codilime.com/blog/python-nornir-for-network-automation-code-examples/

Farin, J. & Roy, D. (17.11.2021). *Network Automation and Security.*

https://ceur-ws.org/Vol-3056/paper-06.pdf

HP. (2.3.2015). *HP to Acquire Aruba Networks to Create an Industry Leader in Enterprise Mobility.*

https://investor.hp.com/news-events/news/news-details/2015/HP-to-Acquire-Aruba-Networks-to-Create-an-Industry-Leader-in-Enterprise-Mobility/default.aspx

HPE. (16.10.2025). *What are network switches?*

https://www.hpe.com/fi/en/what-is/network-switch.html

IBM. (n.d.). What is automation?

https://www.ibm.com/think/topics/automation

Jin, K. (13.2.2023). *Automation tools: Paramiko, Netmiko, NAPALM, Ansible, Nornir or …?*

https://blog.apnic.net/2023/02/13/automation-tools-paramiko-netmiko-napalm-ansible-nornir-or/

Juniper (n.d.). *What is Intent-Based Networking?*

https://www.juniper.net/us/en/research-topics/what-is-intent-based-networking.html

Layer8Packet, (27.9.2024). *The Role of APIs in Network Automation: Transforming How Networks Operate.*

https://www.layer8packet.io/home/the-role-of-apis-in-network-automation-transforming-how-networks-operate

Network Academy. (n.d.). *Simple Network Management Protocol (SNMP*).

https://www.networkacademy.io/ccna/network-services/simple-network-management-protocol-snmp

Network Academy. (n.d.). *Simple Network Management Protocol (SNMP)* [kuva].

https://www.networkacademy.io/ccna/network-services/simple-network-management-protocol-snmp

Network Ninja, (23.7.2023). *Ansible: Complete Network Automation Beginner's Guide.*

https://medium.com/@n3tworkn1nja/ansible-complete-network-automation-beginners-guide-b25986a2c5f1

Nefkens, P. (12.2.2020). *Intent-Based Networking.*

https://www.ciscopress.com/articles/article.asp?p=2995353

Nichols, K. (19.10.2023). *Nornir vs. Ansible: How to Choose?*

https://netboxlabs.com/blog/nornir-vs-ansible-how-to-choose/

NIST, (20.3.2025). *Security and Privacy Controls for Information Systems and Organizations*.

https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final

Robinette, D. (2.7.2025). *Network Automation in 2025: Technologies, Challenges, and Solutions.*

https://www.selector.ai/blog/network-automation-in-2025-technologies-challenges-and-solutions/

Ruostemaa, J. (13.11.2024). *How to generate and use SSH keys for secure authentication on Linux, macOS, and Windows.*

https://upcloud.com/docs/guides/use-ssh-keys-authentication/

Swanson, S. (26.9.2023). *Understanding Network Automation: Tools, Scripting & IaC.*

https://ivision.com/blog/understanding-network-automation-tools-scripting-iac/

TORQ. (18.10.2022). *When to Automate and When Not to Automate.*

https://torq.io/blog/when-to-automate-and-when-not-to-automate/

Uneze, C. (27.9.2024). *How to tackle network automation challenges and risks.*

https://www.techtarget.com/searchnetworking/tip/How-to-tackle-network-automation-risks-and-tasks

VMware, (n.d.). *What is Network Monitoring?*

https://www.vmware.com/topics/network-monitoring

Liite 1. Automaation suorittaminen

(cx_patch) sh-5.1$ python patch.py

Username: juhohj

Enter password:

File name: ArubaOS-CX_6100-6000_10_12_1010.swi

Watcherin IP-osoite: 10.204.92.6

Käynnistetäänkö kytkimet uudelleen?(y/n): y

=== Step 1: Gathering Active Image Info ====

arubalab-cx6000-12g-sw1 | Active: secondary | Version: PL.10.12.1010

==== Devices with Primary as Active Image ===

=== Devices with Secondary as Active Image ===

- arubalab-cx6000-12g-sw1 (10.205.244.133)

=== Step 2: Uploading Firmware to Inactive Partition ===

arubalab-cx6000-12g-sw1 The primary image will be deleted.

arubalab-cx6000-12g-sw1 SSH-avain ei vaadi hyväksyntää.

arubalab-cx6000-12g-sw1 lataa... 13%

arubalab-cx6000-12g-sw1 lataa... 22%

arubalab-cx6000-12g-sw1 lataa... 31%

arubalab-cx6000-12g-sw1 lataa... 40%

arubalab-cx6000-12g-sw1 lataa... 49%

arubalab-cx6000-12g-sw1 lataa... 59%

arubalab-cx6000-12g-sw1 lataa... 67%

arubalab-cx6000-12g-sw1 lataa... 77%

arubalab-cx6000-12g-sw1 lataa... 86%

arubalab-cx6000-12g-sw1 lataa... 95%

arubalab-cx6000-12g-sw1 lataa... 100%

arubalab-cx6000-12g-sw1 Verifying...

arubalab-cx6000-12g-sw1 - Firmware copied successfully. Waiting for validation...

arubalab-cx6000-12g-sw1 Verifying 0%

arubalab-cx6000-12g-sw1 Verifying 25%

arubalab-cx6000-12g-sw1 Verifying 50%

arubalab-cx6000-12g-sw1 Verifying 75%

arubalab-cx6000-12g-sw1 Verifying 100%

arubalab-cx6000-12g-sw1 - Firmware verification complete.

arubalab-cx6000-12g-sw1 Katsotaan checksum...

arubalab-cx6000-12g-sw1 Checksum OK
3da4a34cc7c95ccf2e55746912bf60d5f1744d9aa8b11d4191f65e77755d3a1e

=== Step 3: Rebooting switches ===

arubalab-cx6000-12g-sw1 Tallennetaan konfiguraatio...

arubalab-cx6000-12g-sw1 bootataan...

arubalab-cx6000-12g-sw1 The system is going down for reboot.

Pinging arubalab-cx6000-12g-sw1 until it comes back online...

Pinging 10.205.244.133 until it is reachable or 600 seconds have passed...

10.205.244.133 is now reachable!