Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Ville-Veikko Putaansuu

Kytkimien ja palomuurien käyttöönottojärjestelmä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

16.12.2020

Tekijä(t) Otsikko Sivumäärä Aika	Ville-Veikko Putaansuu Kytkimien ja palomuurien käyttöönottojärjestelmä 44 sivua + 28 liitettä 16.12.2020
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoverkot
Ohjaaja(t)	Yliopettaja Janne Salonen

Insinöörityön tarkoituksena on toteuttaa Juniper Networksin valmistamien kytkimien ja palomuurien käyttöönottojärjestelmä. Järjestelmän tavoitteena on automaation avulla parantaa käyttöönotettavien laitteiden konfiguraatioiden laatua, nopeuttaa käyttöönottoa ja varmistaa kattava dokumentaatio toimitetuista laitteista ja verkoista.

Järjestelmän kehittämistä ja testausta varten rakensin testiympäristön, joka koostui virtualisoidusta CentOS-jakelupaketin GNU/Linux-palvelimesta ja muutamasta eri mallia olevasta Juniper Networksin EX -sarjan kytkimestä sekä kahdesta SRX300-mallin palomuurista.

Toteutettu järjestelmä koostuu TFTP-, HTTP- ja DHCP-palveluista, Ansible-ohjelmasta sekä Juniperin tälle kehittämästä Juniper.junos-roolista sekä Python-ohjelmointikielellä kehittämistäni skripteistä.

Järjestelmän käyttöliittymäksi Npyscreen-viitekehyksellä luotu Python-skripti on suurin yksittäinen osa kokonaisuudesta.

Avainsanat	Juniper, Ansible, verkkoautomaatio, automaatio

Author(s) Title Number of Pages Date	Ville-Veikko Putaansuu Switch and firewall deployment system 44 pages + 28 appendices 16.12.2020
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Specialisation option	Telecommunications and Data Networks
Instructor(s)	Head teacher Janne Salonen

The purpose of this thesis was to produce automated system to deploy Juniper switches and firewalls. The system aims to guarantee the quality of configuration of the deployed network devices, speed up the deployment process and ensure robust documentation of deployed devices and networks.

To create and test the system I built a test environment consisting of virtualized CentOS distro of GNU/Linux server, couple of different model Juniper Networks EX -series switches and two SRX300 model firewalls.

The implemented system consists of TFTP, HTTP and DHCP services, Ansible software and its Juniper developed Juniper.junos role and scripts I developed with the Python programming language.

The Python script developed with Npyscreen framework to be the user interface for the system became the single largest part of it.

Keywords	Juniper, Ansible, network automation, automation

Sisällys

Liite 21. default.EX2300-24T.j2
Liite 22. template-list.yml, default.yml, default-vpn.yml, globalvariables.yml
Liite 23. VLANs.10.yml
Liite 24. VPNs.10.yml
Liite 25. default-vpn.yml
Liite 26. dailyclean, monthlyclean, start-deploymentscript, deploy.sh, inventory ja inventory.sh
Liite 27. deploymentscript.py
Liite 28. hardcodedvars.txt ja VLANs.txt

Lyhenteet

API	Application Programming Interface. Ohjelmointirajapinta.
OoB	Out of Box. Suoraan pakkauksesta.
YAML	Yaml Ain't Markup Language. Helppolukuiseksi suunniteltu datan serialisointikieli.
WWW	World Wide Web. Internet verkossa toimiva järjestelmä jossa selaimet hakevat verkkosivuja palvelimilta.
CLI	Command Line Interface. Komentorivipohjainen käyttöliittymä.
GUI	Graphical User Interface. Graafinen käyttöliittymä.
VLAN	Virtuaalilähiverkko. Tekniikka verkon loogiseen osiointiin.
DHCP	Dynamic Host Configuration Protocol. Protokolla IP-osoitteiden jakamiseen.
YANG	Yet Another Next Generation. NETCONF protokollaa varten kehitetty tietomalli.
NTP	Network Time Protocol. Protokolla jolla laitteen kello synkronisoidaan aikapalvelimen kanssa.

SNMP	Simple Network Management Protocol. Protokolla laitteiden tilatietojen keräämiseen ja käsittelyyn.
ZTP	Zero Touch Provisioning. Menetelmä provisoida laite koskemattamatta mitään sen hallintarajapinnoista.
IT	Information Technology. Tietotekniikka.
SDN	Software Defined Networking. Ohjelmisto-ohjatut verkot. Verkkosuuntaus, jossa ohjauskerros on eriytetty keskitettyyn pisteeseen eri laitteilta.
MAC	Media Access Control. OSI-mallin toiseksi alimman kerroksen, siirtoyhteyskerroksen, toinen alikerros. Usein puhutaan MAC-osoitteista jotka yksilöivät laitteet verkossa.
REST	Representational State Transfer. Arkkitehtuurimalli sovellusrajapintojen toteuttamiseen http-protokollan avulla.
SD-WAN	Software Defined Wide Area Networks. Ohjelmisto-ohjatut laajaverkot. Alikäsite ohjelmisto-ohjatuille verkoille. Menetelmä tuoda kattokäsitteen käsitteitä laajaverkkoyhteyksiin (ks. SDN).
SaaS	Software as a Service. Sovellus palveluna. Sovelluksen tarjoamista palveluna, usein toimittajan pilvestä.
CSV	Comma Separated Value. Pilkuilla erotellut arvot. Tiedostomuoto johon tallennetaan yksinkertaista taulukkomuotoista tietoa.
XML	Extensible Markup Language. Tiedon merkintä standardi.
JSON	JavaScript Object Notation. Tiedon merkintä standardi.
RFC	Request For Comments. Numeroitu kokoelma dokumentteja ja standardeja koskien Internetin käytännöistä ja teknisistä menetelmistä.
RPC	Remote Procedure Cal. RPC:lla tarkoitetaan mallia jossa järjestelmään syötetty komento lähetetään ajettavaksi toiseen järjestelmään.

HTTP	Hypertext Transfer Protocol. Selainten ja WWW-palvelimien tiedonsiirtoon käyttämä protokolla.
VPN	Virtual Private Network. Virtuaalinen privaatti verkko, tekniikkaa yleensä käytetään kahden erillisen verkon suojatusti toisiinsa yhdistämiseen luomalla ns. VPN-tunneli.
TLS	Transport Layer Security. Tunnetaan myös nimellä SSL (Secure Socket Layer). TLS on tietoliikenteen salausprotokolla.
ICMP	Internet Control Message Protocol. IP:llä toimimiva mutta myös vahvasti sen toimintaa nivoutunut hallintaviestien lähetykseen käytetty protokolla.
EoL	End of Life. EoL viittaa ohjelman kehityksen päättymiseen.
SSH	Secure Shell. SSH on yleensä konsoli etäyhteyksien suojauksessa käytetty tietoliikenteen salausprotokolla.

1 Johdanto

Tietotekniikassa on yleisesti hyödynnetty automaatiota sen moninaisten etujen ansiosta, mutta tietoverkot ovat tavallaan laahanneet hieman perässä. [1.] Juniper Networksin vuonna 2020 suorittamaan kyselyyn osallistuneista organisaatioista 47 % on aloittaneet automatisoimaan verkkojaan vasta alle kolmevuotta sitten. Saman tutkimuksen mukaan organisaatioiden suurin syy verkkoautomaation lisäämiseksi ovat hankalat ja toistuvat tehtävät, joista on automaation avulla mahdollista päästä eroon.

Olen töissä IT-palveluntarjoajalla, eli yrityksessä, joka tuottaa IT-palveluita toisille yrityksille. Toimitamme usein lyhelläkin varoitusajalla verkkolaitteita asiakkaillemme. Uuden verkkolaitteen käyttöönottaminen niin konfiguroinnin, dokumentoinnin, valvonnan ja keskitetyn hallinnan osalta on itseään toistava, erikoisosaamista vaativa ja dokumentaation osalta helposti laiminlyötävissä oleva toimenpide.

Insinöörityöni tavoitteena on päästä automaation avulla eroon mahdollisimman monesta verkkolaitteen käyttöönoton aikana tehtävästä toimenpiteestä tehokkuuden, laadun ja työviihtyvyyden parantamiseksi. Järjestelmän on tarkoitus antaa käyttäjän syöttää dokumentaation ja hänen valitsemansa konfiguraatiopohjan vaatimat tiedot, joiden pohjalta se konfiguroi käyttäjän valitsemat laitteet sekä luo CSV-tiedostot (Comma Separated Values) laitteiden lisäämiseksi työpaikallani käytössä olevien dokumentaatio-, hallinta-, ja valvontajärjestelmiin.

Järjestelmän hyödyt ovat selkeimmillään, kun toimitettavia laitteita on useita ja konfiguraatioihin sisältyy useampia lähiverkkoja. Tällöin valmiit tiedostot, joista jokainen uusi laite ja verkko saadaan lisättyä dokumentaatiojärjestelmään, säästää työntekijän työläältä vaiheelta lisätä jokainen uusi laite ja verkko käsin dokumentaatiojärjestelmän graafisen käyttöliittymän kautta. Tärkeätä kuitenkin on, että järjestelmä on myös nopeampi ja helpompi käyttää yksittäisten laitteiden toimituksessa. Näin konfiguraatioiden ja dokumentaatioiden laatu saadaan varmistettua jokaisessa toimituksessa. Kuvailen järjestelmän pystyttämisen käytettävän palvelimen käyttöjärjestelmän asennuksesta alkaen.

2 Verkkoautomaatio

2.1 Automaatio

Verkkoautomaatio ja automaatio yleisesti pitää sisällään erittäin monta osa-aluetta. Työssäni IT-palveluntarjoajalla voin verkkoautomaation jakaa karkeasti kolmeen osa-alueeseen: käyttöönottoon, ylläpitoon ja hallintaan.

Verkkolaitteiden käyttöönoton automatisointi on uusien kohteiden verkkolaitteiden konfiguroinnin ja dokumentoinnin sekä hallinta- ja valvontajärjestelmiin lisäämisen automatisointia. Työssäni rakentamani järjestelmä vastaa juuri käyttöönoton automatisoinnista.

Ylläpidon automatisointi voi yksinkertaisimmillaan olla valvontajärjestelmän luomien hälytysten pohjalta automaattisesti luodut työpyynnöt työnohjausjärjestelmään, tai pidemmälle vietynä esimerkiksi lokien automaattisen läpikäymisen tunnettujen vikatilanteista kielivien merkintöjen varalta. Näihin reagoivat automaattiset toiminnot tai hälytykset.

Hallinnan automatisoinnilla saadaan selvitettyä mahdollisten muutosten tarve useista laitteista ja/tai toteutettua nämä muutokset ilman laite kerrallaan komentoriviltä tai graafisesta käyttöliittymästä tehtävää työtä. Hallinnan automatisoinnista esimerkkinä on seuraava tilanne: Laitetoimittaja julkaisee heidän laitteilleen uuden tuotantoon suositellun ohjelmistoversion. Laitetoimittajan RSS-tiedotteita valvova skripti lataa automaattisesti uuden ohjelmistoversion, asentaa sen testausympäristöön ja ilmoittaa ylläpitäjille uudesta versiosta sähköpostiin. Automatisoidun tai manuaalisen testauksen jälkeen ylläpitäjä ajaa yhdellä komennolla uuden ohjelmistoversion asennuksen ryhmälle laitteita, joiden toiminta on vähemmän kriittistä, jotta niitä voidaan käyttää “koekaniineina” uudelle ohjelmistolleversiolle. Kun koeryhmän kanssa ohjelmistoversio on todettu käyttökelpoiseksi, ajetaan uusi ohjelmistoversio lopuille laitteille joko ryhmittäin tai kerralla.

Näille kaikille, niin kuin kaikelle automaatiolle, yhteistä on pyrkimys vähentää ihmisten suoritettavien toimenpiteiden määrää. Tämä kasvattaa työtehoa vähentämällä työtehtävien suorittamiseen käytettävää ajantarvetta ja parantaa laatua vähentämällä inhimmillisten virheiden määrää. [1.] Tämän lisäksi automaatio parantaa työviihtyvyyttä

ja joissain tapauksessa vähentää myös erikoisosaamisen tarvetta yrityksessä, esimerkiksi hyvin toteutettu verkkolaitteiden toimituksen automatisointijärjestelmä voi vapauttaa verkkoasiantuntijat yksinkertaisten laitetoimitusten suorittamisesta mahdollistamalla tämän tekemisen pienemmän erikoisosaamisen työntekijöille.

Verkkoautomaatio on nykypäivänä edellä mainituista syistä erittäin tärkeää yrityksille. Länsimaissa 250 IT-asiantuntijalta kysyttäessä 77 % piti verkonautomatisointia heille korkeana prioriteettina, mutta 42 % sanoi verkkoautomaation käyttöönoton olevan heillä vielä arviointi-, testaus- tai kehitysvaiheessa [2]. Eri yrityksien verkkoautomatisaation edistäjät liiketoiminnan kannalta ovat vaihtelevia. Taulukon 1 perusteella yrityksille ja pilvipalveluntarjoajille verkkoautomatisaation tärkeimmät hyödyt tulevat luotettavuuden kasvusta, kun inhimmillisetvirheiden määrää saadan vähennettyä ja ylläpitoon liittyvien toimien automatisoinnin myötä ongelmien havaitseminen ja korjaaminen nopeutuvat. Tietoliikennepalveluntarjoajat operoivat massiivia tietoverkkoja. Mahdollisesti tämän takia heidän on helppo saavuttaa suuria säästöjä verkkotoimintojen automatisoinnilla. Mahdollisesti kysymysasettelun tai vastausvaihtoehtojen takia toisessa tutkimuksessa tietoliikennepalveluntarjoajat raportoivat kahdeksi tärkeimmäksi syyksi toimitusten tehokkuuden ja asiakastyytyväisyyden. [3.]

Taulukko 1. Verkkoautomaation edistäjät liiketoiminnan kannalta [1].

Edistäjä	**Tietoliikennepalveluntarjoajat**	**Pilvipalveluntarjoajat**	**Yritykset**
Luotettavuus	24%	27%	26%
Toimitusten tehookkus	24%	22%	23%
Kulusäästöt	32%	17%	19%
Ketteryys	21%	15%	14%

2.2 Verkkoautomaation työkalut ja järjestelmät

2.2.1 ZTP

ZTP on lyhenne sanoista zero touch provisioning eli nollakosketusprovisiointi, jolla tarkoitetaan laitteen provisiointia OoB (Out of Box) -tilanteessa ilman että laitteen omaan grafiseenhallintaan tai komentoriviin tarvitsee koskea. Lähestulkoon jokaisella laitevalmistajalla on heidän verkkolaitteilleen implementaatio ZTP:stä, ainakin kytkimien osalta. Yleinen ZTP:een toimintaperiaate on, että verkossa laitteille osoitteet lainaava DHCP (Dynamic Host Control Protocol) -palvelu kertoo osoitteen mukana jaettavissa vaihtoehtotiedoissa laittellee, mistä se voi hakea itselleen konfiguraatiotiedoston ja ohjelmistopaketin, jotka laite sitten lataa ja ottaa käyttöön. ZTP-toiminnot lataavat konfiguraatiot ja ohjelmistopaketit yleensä FTP:llä (File Transfer Protocol), TFTP:llä (Trivial File Transfer Protcol) tai HTTP:llä (Hypertext Transfer Protocol). Esimerkiksi Juniper Networks tarjoaa ZTP-toimintoa täysmääräisenä kytkimilleen, mutta palomuureilleen he tarjoavat vain karsitumpaa mallia Autoinstallation-nimellä, jonka toimintaperiaate on sama kuin kytkmien ZTP-toiminnon, mutta se ei osaa hakea ja asentaa ohjelmistopaketteja [4; 5].

ZTP on usein ensimmäinen askel verkkoautomaatiossa provisioinnin ollessa ensimmäinen askel laitteiden käyttöönotossa. Menemättä automatisoinnissa sen syvemmälle saadaan ZTP:llä helposti syötettyä uusille laitteille organisaation sisäiset staattiset muuttujat, esimerkkeinä loki- tai NTP (Network Time Protocol) -palvelimien tiedot. ZTP:llä voi myös helposti alustaa laitteet hallintajärjestelmiin liittämistä varten, esimerkiksi Juniperin EX2300-sarjan kytkintä ei saa suoraan lisättyä Juniperin Junos Space -järjestelmän hallintaan tai Ansible-järjestelmän kohteiksi ilman käsin konfigurointia tai ZTP-toimintoa. Sen tehdasasetuksissa ei ole hallintayhteyksien vaatimia SSH (Secure Shell) -asetuksia. Oletuskonfiguraation oleellinen kohta on nähtävillä liittessä 1.

2.2.2 Keskitetyn hallinnan järjestelmät

Perinteinen tapa hallita verkkolaitteita on joko laitteen CLI:ltä (Command Line Interface) tai GUI:sta (Graphical User Interface). Verkkolaitteiden graafiset käyttöliittymät ovat usein selainpohjaisia tai valmistajan kehittämiä erillisiä ohjelmia. Verkkolaitteiden hallinta

näin yksi kerrallaan näiden CLI:lta tai GUI:lta on paikoittain työlästä. Tätä varten useat verkkolaitetoimittajat tarjoavat keskitetyn hallinnan ratkaisuja tuotteilleen.

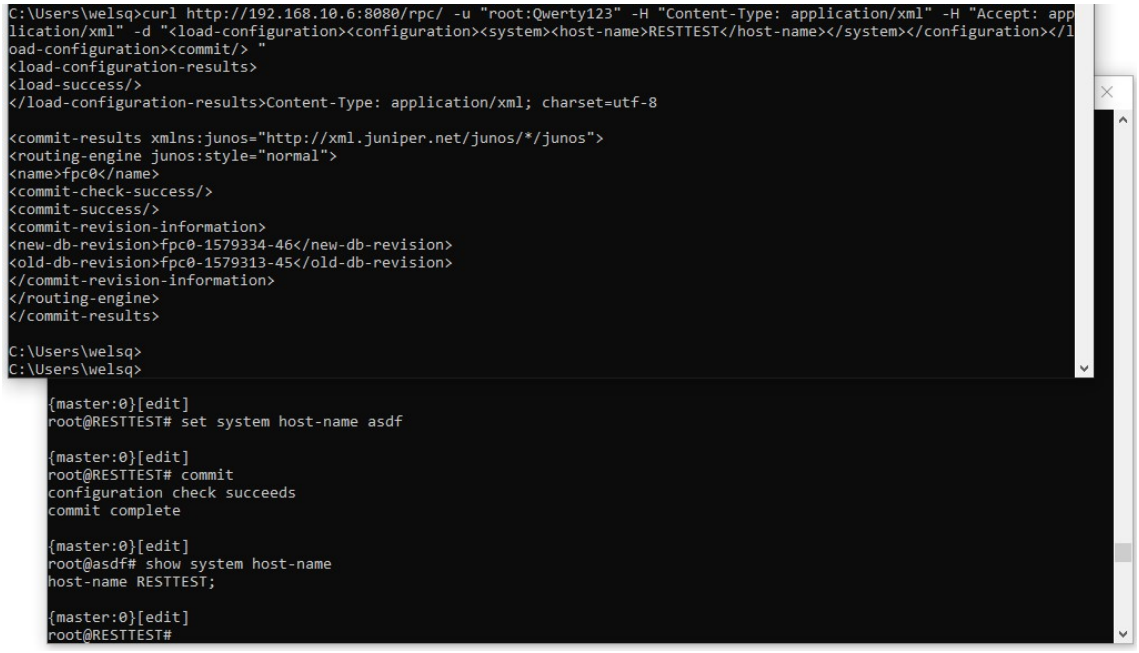
Keskistetyn hallinnan vahvuuksia on tiedon saatavuus yhdestä paikasta ja mahdollisuus suorittaa toimintoja kuten ohjelmistopäivityksi. Lisäksi keskitetyn hallinnan ratkaisut tarjoavat usein vähintään osittaista automatiikkaa. Esimerkiksi WatchGuardin WSM (Watchguard System Manager) -järjestelmässä on mahdollista rakentaa VPN (Virtual Private Network) -tunneleita hallintanäkymässä vain vetämällä ja tiputtamalla hallittavan palomuurin kuvakkeen toisen hallittavan palomuurin kuvakkeen päälle, jolloin WSM konfiguroi VPN-tunnelin näiden välille [6]. Juniper Networksin Junos Space -järjestelmä tarjoaa mahdollisuuden luoda niin sanottuja “Configletteja”, jotka ovat helposti käyttöönotettavia konfiguraatiopohjia, joihin voi sisällyttää pohja- tai laitekohtaisia muuttujia [7].

2.2.3 Ohjelmointirajapinnat

Ohjelmointirajapinnat eli API:t (Application Programming Interface) ovat rajapintoja, joiden avulla ohjelmat tai järjestelmät voivat kommunikoida keskenään. Verkkopalveluissa suosituimmat rajapinnat ovat SOAP (Simple Object Access Protocol) ja REST (Representational State Transfer), joista SOAP on nimensä mukaisesti oma protokollansa, kun REST on enemmänkin suunnittelumalli rajapinnoille.

Tämän hetken suosituin tapa toteuttaa ohjelmointirajapintoja on hyödyntää REST-mallia [8]. REST-mallia noudattavia rajapintoja kutsutaan sanalla RESTful. RESTful-rajapinnat noudattavat tilatonta asiakas-palvelinarkkitehtuuria, ja tiedonsiirto tapahtuu pääosin HTTP:llä. Tilattomalla tarkoitetaan, ettei palvelin tallenna tietoa asiakkaiden pyynnöistä, asiakas-palvelinmalli tarkoittaa. että pyynnöt lähtevät aina asiakkaalta palvelimelle. Palvelin ei siis itsessään ikinä lähetä asiakkaalle pyyntöjä [9]. Tiedon pyytämisen lisäksi pyyntö asiakkaalta palvelimelle voi lähettää tietoa ja vaikuttaa palvelimen konfiguraatioon, HTTP GET -pyynnöllä haetaan tietoa, HTTP PUT- tai POST-pyynnöllä lähetetään tietoa. Tosin POST- tai PUT-pyynnöillä usein myös haetaan tietoa, koska kyseiset pyynnöt pystyvät lähettämään tietoa palvelimelle, missä kerrotaan, mitä tietoa halutaan saada takaisin. Esimerkiksi Juniper Networksin Junos Space -järjestelmää ja JunOS-käyttöjärjestelmällä toimivia verkkolaitteita on mahdollista operoida RESTful-

rajapinnan kautta [10; 11]. Kuvassa 1 nähdään esimerkki POST-pyynnön lähettämisestä curl -d -komennolla Juniper Networksin valmistaman EX2300-sarjan kytkimen REST-rajapintaan, POST-pyynnöllä lähetetään tietona komennot asettaa kytkimen host-name arvoksi RESTTEST. Kuvasta 1 poiketen rajapintoja ei ole tarkoitus käyttää manuaalisesti komentoja ajamalla vaan skriptien tai ohjelmien kautta.



Kuva 1. Esimerkki EX-kytkimen konfiguraation muokkaamisesta REST-rajapinnalla.

2.2.4 NETCONF

2000-luvun alussa oli IETF:llä (Internet Engineering Task Force) huomattu, että verkkolaitteiden ohjelmoitavissa olevan hallintaprotokollan SNMP:een (Simple Network Management Protocol) suosio laitteiden hallintaan käyttämiseen oli olematonta ja sen käyttö rajoittui lähinnä laitteiden tilatietojen seuraamiseen. Kesällä 2002 Internet-arkkitehtuurilautakunta ja IETF:n avainjäsenet kokoontuivat tietoliikennepalveluntarjoajien kanssa selvittämään yhteistä standardia verkkolaitteiden hallinnalle. Samoihin aikoihin Juniper Networks esitteli IETF:lle ja muulle tietoverkkoyhteisölle heidän XML (Extended Markup Language) -merkintäkieleen perustuvan hallintamenetelmänsä. Tämä johti NETCONF-työryhmän perustamiseen ja lopulta NETCONF-protokollan julkaisuun vuoden 2006 joulukuussa. [12; 13.]

Nykypäivänä NETCONF-protokollalla voidaan yhdistää laitteisiin joko SSH-, SOAP-, tai TLS (Transport Layer Security) -tekniikoilla. Avatun yhteyden yli NETCONF lähettää

RPC (Remote Procedure Call) -mallin mukaisesti XML-merkintäkielellä koodattuja komentoja kohdelaitteelle. Kuvassa 1 on nähtävillä curl-komennon -d -vivun jälkeen XML-mallinen komento. RFC (Request For Comments) 7951 toi NETCONF-protokollaan tuen viestien koodaamiseksi JSON (JavaScript Object Notation) -formaattiin vaihtoehtona XML:lle [14]. NETCONF-protokollan julkaisun jälkeen huomattiin pian tarve yhtenäiselle datanmallinnuskielelle verkkolaitteiden konfiguraatio- ja tilatiedoille. Näin syntyi RFC:n 6020 myötä julkaistu YANG (Yet Another Next Generation) [15].

```
module example-sports {

  namespace "http://example.com/example-sports";
  prefix sports;

  import ietf-yang-types { prefix yang; }

  typedef season {
    type string;
    description
      "The name of a sports season, including the type and the year, e.g,
       'Champions League 2014/2015'.";
  }

  container sports {
    config true;

    list person {
      key name;
      leaf name { type string; }
      leaf birthday { type yang:date-and-time; mandatory true; }
    }

    list team {
      key name;
      leaf name { type string; }
      list player {
        key "name season";
        unique number;
        leaf name { type leafref { path "/sports/person/name"; } }
        leaf season { type season; }
        leaf number { type uint16; mandatory true; }
        leaf scores { type uint16; default 0; }
      }
    }
  }
}
```

Kuva 2. YANG-datamallinnuskielellä luotu datamalli urheilujoukkueelle [15].

```
{
  "example-sports:sports": {
    "person": [
      {
        "name": "Lionel Andrés Messi",
        "birthday": "1987-06-24T00:00:00-00:00"
      },
      {
        "name": "Cristiano Ronaldo",
        "birthday": "1985-02-05T00:00:00-00:00"
      }
    ],
    "team": [
      {
        "name": "FC Barcelona",
        "player": [
          {
            "name": "Lionel Andrés Messi",
            "season": "Champions League 2014/2015",
            "number": 10,
            "scores": 43
          }
        ]
      },
      {
        "name": "Real Madrid",
        "player": [
          {
            "name": "Cristiano Ronaldo",
            "season": "Champions League 2014/2015",
            "number": 7,
            "scores": 48
          }
        ]
      }
    ]
  }
}
```

```
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">

  <sports xmlns="http://example.com/example-sports">
    <person>
      <name>Lionel Andrés Messi</name>
      <birthday>1987-06-24T00:00:00-00:00</birthday>
    </person>
    <person>
      <name>Cristiano Ronaldo</name>
      <birthday>1985-02-05T00:00:00-00:00</birthday>
    </person>
    <team>
      <name>FC Barcelona</name>
      <player>
        <name>Lionel Andrés Messi</name>
        <season>Champions League 2014/2015</season>
        <number>10</number>
        <scores>43</scores>
      </player>
    </team>
    <team>
      <name>Real Madrid</name>
      <player>
        <name>Cristiano Ronaldo</name>
        <season>Champions League 2014/2015</season>
        <number>7</number>
        <scores>48</scores>
      </player>
    </team>
  </sports>

</data>
```

Kuva 3. Vasemmalla JSON- ja oikealla XML-esitykset kuvan 2 datamallin ilmentymästä [16].

NETCONF-protokollassa on merkittäviä hyötyjä, mitä vanhemmassa SNMP:ssä ei ollut. NETCONF erottelee tilatiedot ja konfiguraatiotiedot omiin tietosäilöihinsä, mikä helpottaa näiden käsittelyä. Esimerkkinä konfiguraatiotietojen vertailu on helpompaa, kun haetun konfiguraatiotiedon seassa ei tule tilatietoja. Tilatiedot voivat poiketa toisistaan, vaikka konfiguraatiotiedot olisivat identtiset. NETCONF-protokollalla on kyky suorittaa transaktiotyyppisiä muutoksia. Yksi transaktio voi olla esimerkiksi monta konfiguraatiossa eri sijainteihin tehtyä muutosta, jotka ajetaan kaikki kerralla, ja jos yksi niistä ei mene laitteen konfiguraatiotarkistuksesta läpi, koko transaktio perutaan. Transaktiopohjaisuus helpottaa muutosten tekemistä konfiguraatioon. Esimerkiksi yhden transaktion konfiguraatioiden määrityksessä ei tarvitse ottaa huomioon järjestystä toisin kuin vaikka Ciscon IOS-käyttöjärjestelmän (Internetwork Operating System) komentoriviltä tai SNMP:llä konfiguroitaessa, koska tietyt konfiguraatiot ovat toisiinsa sidonnaisia. Ei esimerkiksi voi luoda reittiä kohti verkkorajapintaa ennen kuin kyseinen verkkorajapinta on luotu laitteelle. NETCONF-protokollassa myös määritellään useampi tietosäilö konfiguraatioille, jolloin muutokset voidaan ensin suorittaa ehdokaskonfiguraatioon, josta laite voi tarkistaa sen ajettavuuden ennen kuin se

kopioidaan ajossa olevaksi konfiguraatioksi [17]. NETCONF-protokollasta on tehty myös REST-rajapintaa hyödyntävä RESTCONF, jossa laitteisiin yhdistetään HTTP:tä käyttäen niiden REST-rajanpintaan.

2.2.5 Automaatiojärjestelmät verkoille

Suosituimpia avoimen lähdekoodin verkoille sopivia automaatiojärjestelmiä ovat Saltstack, Ansible, Puppet ja Chef. Automaatiojärjestelmillä on tarkoitus hoitaa suurien laitemäärien käskytystä ja konfiguraation hallintaa. Näistä Ansible on ainoa alun perin agentittomalla arkkitehtuurilla suunniteltu. Chef on näistä ainoa joka ei ole kirjoitushetkeen mennessä lisännyt ainakin osittaista tukea agentittomille ympäristöille. Kaikille verkkolaitteille ei voi asentaa agentteja, mikä johtuu niiden suljetuista käyttöjärjestelmistä. Ansiblen asema suosituimpana perustuu juuri agentittomuuteen ja helppoon oppimiskäyrään, vaikka se saattaa ominaisuuksissaan hävitä esimerkiksi Chefille tai Saltstackille. [18; 19; 20.]

Ansiblessa kohde laitteista muodostetaan inventaario, jossa laitteita voidaan jakaa ryhmiin sekä aliryhmiin. Inventaariossa voidaan myös määrittää ryhmä- tai laitekohtaisia muuttujia. Kohdelaitteille ajetaan tehtävinä erilaisia toimintoja suorittavia moduuleja joko yksittäin tai suositummin osana pelikirjaa. Pelikirja voi sisältää useita tehtäviä ja siihen voi sisällyttää logiikkaa. Ansible moduuleja voi luoda itse, mutta laitevalmistajilla on usein tarjolla hyvin kattavat valikoimat heidän laitteilleen. Lisäksi toisten käyttäjien tekemiä moduuleja tai valmiita pelikirjoja on runsaasti tarjolla esimerkiksi github.com-sivustolla. [21; 22; 23].

2.2.6 SDN

SDN (Software Defined Networking) terminä viittaa ohjelmisto-ohjattuihin verkkoihin. SDN voidaan jakaa karkeasti kahteen osa-alueeseen, joita voidaan tarjota yhdessä tai erikseen: SD-WAN (Software Defined Wide Area Networks) ja SD-LAN (Software Defined Local Area Networks) eli ohjelmisto-ohjattu laajaverkko ja lähiverkko. SDN:stä on muodostunut hankalasti määriteltävä muotisana. Käytännössä nykyään sillä viitataan kehittyneempiin keskitetyn hallinnan järjestelmiin jotka tyypillisesti pitävät sisällään pitkälle viedyt ZTP-tyyppiset provisiointiominaisuudet, monitorointimahdollisuuden, käskytysjärjestelmän ja konfiguraationhallinnan. Esimerkkejä tälläisistä järjestelmistä on Ciscon Cisco DNA Center (SDN), SD-Access (SD-LAN) ja SD-WAN (SD-WAN) sekä

Juniper Networksin Contrail (SD-WAN) ja MIST (SD-LAN). Alun perin SDN:än ideana oli laitevalmistajariippumaton avoin ohjelmoitavissa oleva verkon hallintajärjestelmä, mutta termi on pitkälti onnistuttu kaappaaman kuvaamaan myös laitevalmistajien omia uuden sukupolven hallintajärjestelmiä [25].

SDN syntyi kun MIT (Massachusetts Institute of Technology) -opiskelija Martin Casadoa halusi kehittää Stanfordin yliopistolle verkon, jossa hallintataso on eriytetty edelleenlähetystasosta, koska hallinta haluttiin pois laitteilta keskistettyyn ohjelmallisesti hallittavissa olevaan järjestelmään. Tästä syntyi Clean Slate -ohjelma, joka taas synnytti OpenFlow-projektin, OpenFlow on verkkolaitteiden edelleenlähetystaulun hallintaan kehitetty protokolla. Siinä missä Netconf-protokollalla voi muokata verkkolaitteen konfiguraatioita ja kysyä tilatietoja, OpenFlow-protokollalla vaikutetaan nimenomaan siihen, miten paketteja ohjataan verkossa. MIT Technology Review -lehden journalisti Kate Green keksi konseptille termin Software Defined Networking. [25.]

3 Järjestelmät ja verkkolaitteiden käyttöönotto

3.1 Järjestelmät

3.1.1 Junos Space

Olemme jo pidempään työpaikallani pyrkineet keskittämään toimittamamme ja ylläpitämämme järjestelmät Juniper Networksin verkkolaitteille, mikä johtueu niiden hyvin toteutetusta CLI-hallinnasta sekä meille suotuisasta hinta-laatusuhteesta. Luonnollisesti tästä seuraa, että olemme ottaneet käyttöön Juniper Networksin Junos Space -keskitetyn hallinntajärjestelmän. Kaikki toimitettavat laitteet on lisättävä Junos Spaceen hallittaviksi. Junos Spacessa on käytettävissä kaksi eri tapaa yhdistää laitteet hallintaan joko suoraan laitteen IP-osoittella tai mallintamalla laite järjestelmään, jolloin Junos Space tarjoaa laitteelle syötettävän konfiguraation, jolla laitteen saa yhdistämään Junos Spaceen käänteisellä SSH-yhteydellä. Ensimmäistä näistä tavoista käytämme laitteille, johon Junos Spacella on suora yhteys. Esimerkiksi kiinteän IP-osoitteen takana olevat palomuurit ja konesalimme kytkimet, ja jälkimmäistä käytämme vaihtuvan IP-osoitteen takana oleviin palomuureihin ja asiakkaiden kytkimiin.

IP-osoitteella lisättäessä käyttäjän täytyy syöttää järjestelmän hallintaan käytettävän käyttäjätunnuksen tiedot (käyttäjänimi ja salasana), IP-osoite, mitä vasten hallintayhteys

avataan, sekä SNMP-tiedot. Käänteisellä hallintayhteydellä lisättäessä käyttäjän tulee syöttää järjestelmään laitemalli, käyttäjätunnus ja SNMP-tiedot. Kummassakin tapauksessa käyttäjä voi syöttää tiedot CSV-tiedostosta.

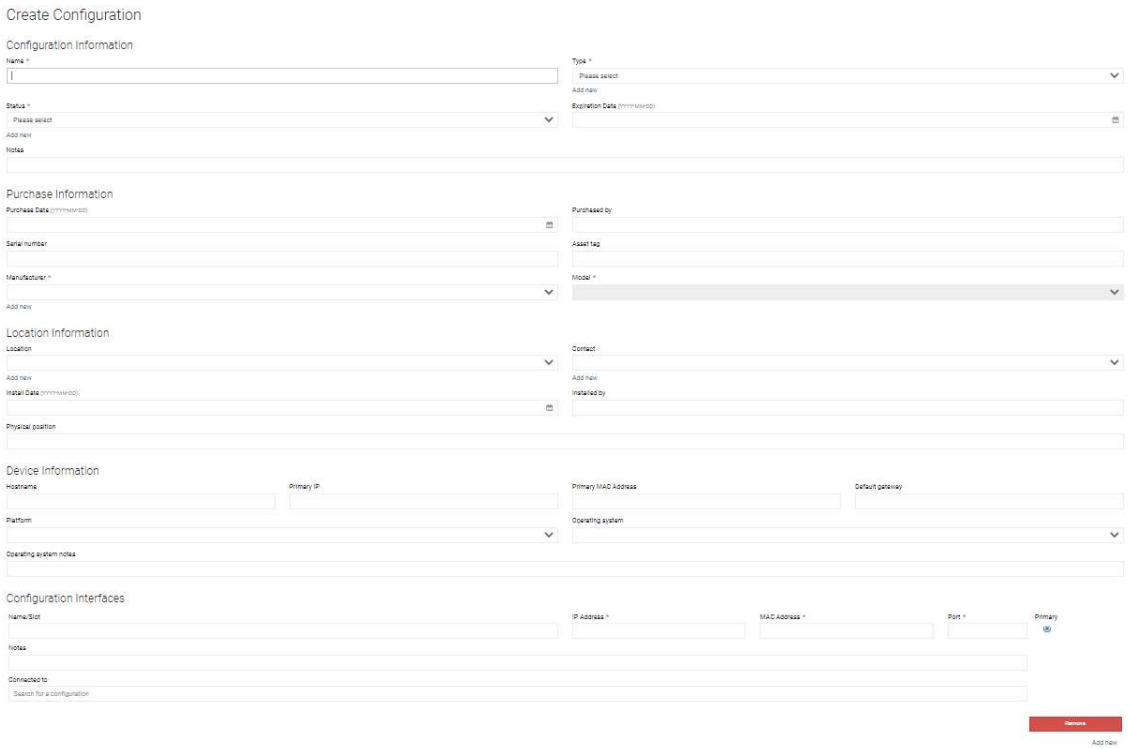
Junos Space -järjestelmässä jaottelemme eri asiakkaiden laitteet omiin hallinta-domaineihin. Jos asiakkuudella ei vielä omaa domainia järjestelmässä ole, täytyy käyttäjän luoda se käsin. Järjestelmää on mahdollista operoida myös REST-rajapinnan kautta. Työni laajuuden rajaamiseksi jätin REST-rajapinnan kautta Junos Spacen operoinnin työni ulkopuolelle.

3.1.2 IT Glue

Dokumentointijärjestelmänä meillä on käytössä ITG Softwaren SaaS (Software as a Service) -tyyppisesti tarjoama IT Glue. Verkkojen osalta meidän on dokumentoitava laitteet, sijainnit, kontaktihenkilöt, lähiverkot ja laajaverkot. Seuraavassa luvussa käyn läpi dokumentaation vähimmäisvaatimukset jokaisen dokumentaation kohdalta.

Kontaktitietoihin sisältyvät etunimi, sukunimi, puhelinnumero ja sähköposti. Sijaintitietoihin sisältyvät maa, kaupunki, katuosoite ja postinumero. Laitetietoihin sisältyvät nimi, hostname, hallintaosoite, malli, sarjanumero, kontakti, sijainti ja toimituksen tyyppi / asset_tag (palvelulaite vai asiakkaan omistama). Lähiverkkotietoihin sisältyvät IP-avaruus, sijainti ja VLAN (Virtual Local Area Network) -tunnus. Laajaverkkotietoihin sisältyvät palveluntarjoaja, liittymätunnus, tyyppi, nopeudet, IP-osoitetiedot, hallinoija ja sijainti.

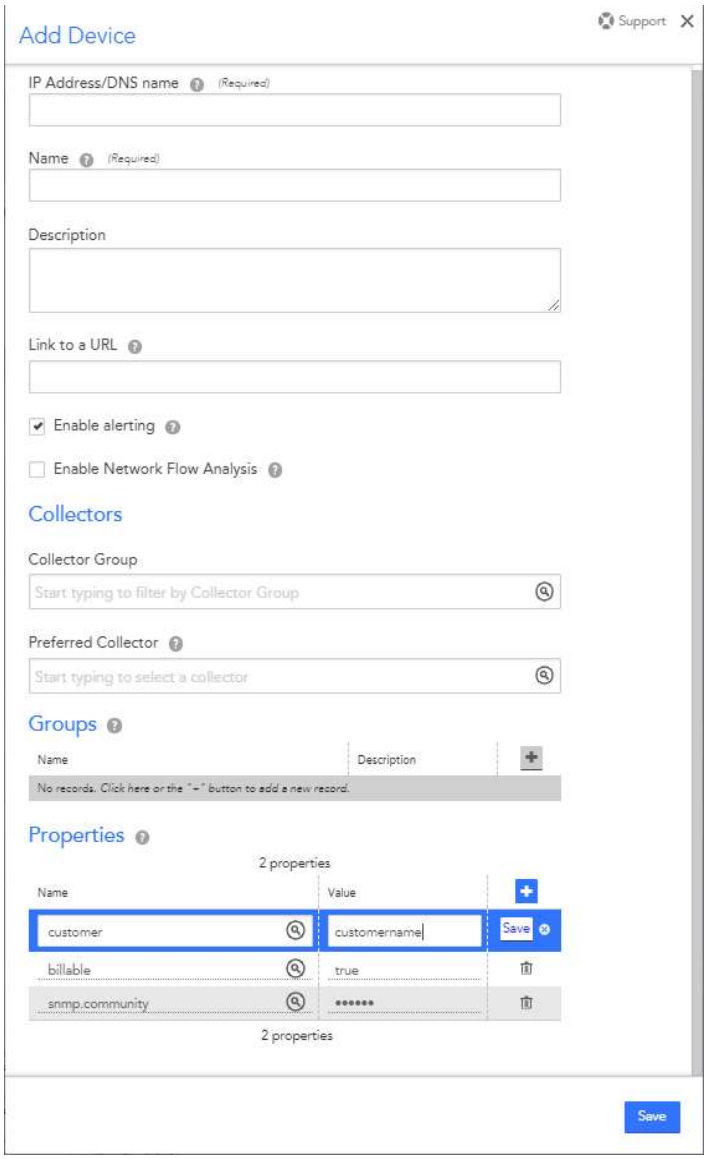
Verkkolaitteiden toimitusten yhteydessä on aina dokumentoitava vähintään laitteet, mutta usein toimitukset menevät uusiin kohteisiin, jolloin kaikki edellä mainitut asiat on dokumentoitava. Dokumentoinnit tehdään usein käsin. Kuvassa 4 esimerkkinäkymästä laitedokumentaation lisäämiseen, mutta dokumentaatiot on mahdollista lisätä myös CSV-tiedostoista. Käsintehtynä tämä työvaihe on varsin puuduttava, varsinkin jos laitteita ja verkkoja on useita. Dokumentaatiot on jaettu asiakaskohtaisten organisaatioiden alle. Jos asiakkuudelle ei ole luotu vielä organisaatiota järjestelmään, täytyy käyttäjän luoda se käsin. Järjestelmää on mahdollista operoida myös REST-rajapinnan kautta, ja tarkoitus on tulevaisuudessa tarkastella mahdollisuutta järjestelmälle luoda dokumentaatiot suoraan REST-rajapinnan avulla.



Kuva 4. IT Glue, laitedokumentaation luomisnäkymä.

3.1.3 LogicMonitor

Valvontajärjestelmänä käytämme LogicMonitor-nimisen yrityksen tarjoamaa LogicMonitor-tuotetta, jota he tarjoavat SaaS-tyyppisesti. Laitteen valvontaan lisäämiseksi käyttäjän on lisäysvaiheessa syötettävä järjestelmään laitteen nimi, IP-osoite, SNMP-tiedot, asiakkaan nimi ja laskutustunniste (laskutetaanko vai ei, joko true tai false). Käyttäjä voi lisätä laitteita valvontana käsin kuvassa 5 nähtävissä olevasta näkymästä, IP-aluetta skannaamalla, jolloin LogicMonitor yrittää ICMP (Internet Control Message Protocol) Echo Request -paketeilla hakea alueelta laitteita ja käyttäjä voi vastanneista laitteista lisätä valitsemansa järjestelmään tai lisäämällä laitteet CSV-tiedoston tiedoista.



Kuva 5. LogicMonitor, uuden laitteen lisäämisnäkymä.

LogicMonitor käyttää kansiorakennetta, jolla erotellaan asiakkuudet, sijainnit ja laitetyypit. Jos asiakkuus tai sijainti on uusi, täytyy näitä varten luoda kansiot käsin. Järjestelmää on mahdollista operoida myös REST-rajapinnan kautta ja tarkoitus on tarkastella tulevaisuudessa mahdollisuutta käyttöönottojärjestelmälle lisätä laitteet, ja luoda tarvittaessa kansiorakenteet REST-rajapinnan kautta.

3.1.4 Salasanat

Työpaikallani on käytössä salasananhallintajärjestelmä, mutta työhöni ei tietoturvasyistä tule mitään liitännäisyyksiä kyseiseen järjestelmään. Käyttäjät operoivat sitä nyt ja jatkossa käsin, joten en siitä sen olemassaoloa enempää työssäni mainitse.

3.2 Verkkolaitteiden käyttöönotto

Verkkolaitteiden toimitus lähtee liikkeelle toimitettavien laitteiden keräämisestä varastosta, paketeista purkamisesta sekä laitteen kytkemisestä asennusverkkoon ja sähköihin. Seuraavaksi käyttäjä käy konsolikaapelin avulla komentoriviltä syöttämässä laitteille asennuksen aikana käytettävät tunnukset, sallivat SSH-yhteydet ja antavat tälle IP-osoitteen asennusverkosta. Seuraavaksi ne käydään lisäämässä hallintaan Junos Space -järjestelmään, jonka kautta laiteille siirretään ja asennetaan kurantti ohjelmistoversio. Laitteiden päivittämisen jälkeen käyttäjä luo ja asentaa konfiguraatiot laitteelle käyttäen Junos Spaceen luotuja konfiguraatiopohjia. Pohjat ovat laitemallikohtaiset eli yhteiset muuttujat täytyy syöttää jokaiselle eri mallia olevalle laitteelle erikseen. Toimituksissa joissa on monta samaa laitemallia olevaa laitetta, kuten kytkimien kohdalla usein on, täytyy vain laitekohtaiset muuttujat syöttää erikseen.

Koska laitteiden asennusverkon IP-osoitteeseen ei laitteiden toimituksen jälkeen voi yhdistää, tulee laitteet poistaa hallinnasta, paitsi jos toimituksessa on mukana palomuuri, joka tulee kiinteän IP-osoitteen omaavan liittymän taakse, jolloin sen IP-osoitteen voi hallinnasta vaihtaa vastaamaan liittymästä sille määrättyyn IP-osoitteeseen. Poistamisen jälkeen tarvittaessa laitteet mallinnetaan järjestelmään laitteiden avaamaan käänteistä hallintayhteyttä varten tarvittavien konfiguraatioiden generoimiseksi, jotka käyttäjä sitten syöttää laitteille, joko nyt tai fyysisten asennusten jälkeen. Kun laitteet on asennettu kohteeseen, täytyy käyttäjän luoda tarvittavat dokumentaatiot IT Glue -järjestelmään ja tarvittaessa lisätä laitteet LogicMonitor-järjestelmään valvontaan.

4 Käyttöönottojärjestelmän suunnittelu

4.1 Tavoitteet järjestelmälle

Järjestelmän tarkoitus on vähentää verkkolaitteiden käyttöönottoon vaadittavien toimenpiteiden määrää ja sitä myöten nopeuttaa laitteiden toimittamista, vähentää virheiden määrää ja tehdä laitteiden käyttöönotosta työntekijälle mielekkäämpää. Järjestelmä ei helpota fyysisiä toimenpiteitä liittyen laitteiden hakemiseen varastosta tai pakkauksista purkamiseen.

Tavoitteena on, että kun laitteet on kytketty asennusverkkoon ja virtoihin, ne päivittyisivät automaattisesti laitemallikohtaisesti määritettyyn ohjelmistoversioon ja olisivat

konfiguraatiot laitteille syöttävän järjestelmän hallittavissa. Käyttäjän tulisi saada päivitetyt laitteet valittavaksi yhteen näkymään. Laitteiden lisäksi käyttäjän tulisi voida valita useammasta määritellystä konfiguraatiopohjasta. Laitteiden ja konfiguraatiopohjan valinnan jälkeen järjestelmä pyytäisi käyttäjältä konfiguraatiopohjan vaatimat tiedot. Mahdollisuuksien mukaan nämä tulisi vaatia vain kerran ja järjestelmän osaisi luoda niiden pohjalta laitekohtaiset muuttujat ja syöttää ne lopuksi laitteille. Laitteiden konfiguroimisen ja päivittämisen lisäksi järjestelmän pitäisi tarjota käyttäjälle CSV-tiedostot dokumentaatioiden luomiseksi sekä valvontaan ja hallintaan liittämiseksi. Käyttäjien tulisi voida lisätä järjestelmään uusia konfiguraatiopohjia ja tuettavia laitemalleja ilman ohjelmointiosaamista.

4.2 Verkkoympäristö

Järjestelmässä käytetään yhtä palvelinta ja kahta lähiverkkoa. Palvelinta kutsuttakoon asennuspalvelimeksi. Toinen lähiverkko on asennusverkko, eli lähiverkko johon käyttöönotettavat laitteet kytketään. Tätä lähiverkkoa ei reititetä muihin verkkoihin, ja toinen voi olla mikä tahansa lähiverkko kunhan työasemilta pääsee yhdistämään siihen SSH:lla. Nykyään on yleistä että yritysten palvelimet sijaitsevat konesaleissa, jos konesaliin menevän yhteyden kaista on alle 1Gb/s, kuten toimistollani, suosittelen, että palvelin pystytään poikkeuksellisesti paikallisesti että asennusverkosta on 1Gb/s kaistaa käytettävissä käyttöönottopalvelimelle verkkolaitteiden ohjelmistojen nopeuttamiseksi. Asennusverkkoon ei tule muuta kuin asennettavat laitteet ja yksi verkkokortti asennuspalvelimelta, tietoturvan kannalta tarkoitus on, ettei asennusverkkoon pääse muualta kuin siihen määritetyiltä kytkinporteilta, joita tulisi olla vain tiloissa, jossa asennuksia suoritetaan, tai asennuspalvelimelta.

4.3 Palvelimen käyttöjärjestelmän valinta

Palvelimeksi järjestelmälle valitsin CentOS 8 GNU/Linux -jakelun käyttöjärjestelmän. Päädyin Linux käyttöjärjestelmään koska järjestelmän vaatimien eri osien konfigurointiin löytyy runsaasti ohjeita ja esimerkkejä Linux-alustalle tehtynä. CentOS valikoitua Linux-distribuutioista sen takia, että se mukailee RHEL (Red Hat Enterprise Linux) distribuutiota. Näin jos Linux-palvelimien käyttö yleistyy työpaikallani, on helpompi siirtyä RHEL-käyttöjärjestelmään, johon on saatavilla kaupallista tukea [26]. Tosin suosittelen käyttämään CentOS Stream -käyttöjärjestelmää, koska CentOS 8 -käyttöjärjestelmän EoL (End of Life) päivämäärä on 31.12.2021 [27].

4.4 Laitteiden provisiointi

4.4.1 ZTP Juniper Networksin verkkolaitteilla

Saadakseni kytkimet provisioitua hallitaviksi ilman manuaalista konfigurointia niiden paikallisen hallinnan kautta täytyy minun hyödyntää niiden ZTP-ominaisuuksia. Palomuureilla vastaava ominaisuus on nimeltään Autoinstallation, mutta toisin kuin kytkimien ZTP se ei tue ohjelmistoversion päivittämistä. Pitääkseni kytkimien ja palomuurien käyttöönottoon liittyvät toiminnot järjestelmässäni mahdollisimman yhteneväisenä, päätin hoitaa kytkimien päivittämisen yhdessä palomuurien kanssa hallintajärjestelmän kautta. ZTP ja Autoinstallation toimintojen tarvitsee vain puskea laitteille konfiguraatiot, joiden avulla niitä voi hallita NETCONF:in ja SSH:n kautta. ZTP ja Autoinstallationin käyttöä varten tarvitsen DHCP-, HTTP- ja TFTP-palveluita. ZTP ja Autoinstallation lataavat konfiguraatiot DHCP-vaihtoehdossa 150 määritellyltä palvelimelta. ZTP:llä voi vielä tarkemmin määritellä konfiguraatio nimen ja latauksessa käytettävän protokollan vaihtoehdolla 43, mutta Autoinstallation-ominaisuus lataa vain vaihtoehdossa 150 määritellystä osoitteesta TFTP:llä network.conf-nimisen tiedoston konfiguraatioksi [28; 29].

4.4.2 DHCP-, HTTP- ja TFTP-palvelut

DHCP-, HTTP- ja TFTP-palveluille ainoa vaatimukseni oli, että nämä ovat yleisesti käytettyjä ohjeiden ja esimerkkien saatavuuden varmistamiseksi. DHCP-palvelun päätin toteuttaa Dhcpd:llä, joka on Internet Systems Consortiumin ylläpitämä avoimen lähdekoodin DHCP-palvelu. HTTP-palveluksi valikoitu Apache Software Foundationin ylläpitämä avoimen lähdekoodin Apache HTTP -palvelu, joka on maailman suosituin HTTP-palvelu. TFTP-palveluksi valikoitui nykyisin Peter Anvin ylläpitämä avoimen lähdekoodin tftp-server.

4.5 Laitteiden hallinta

Käyttöönotettavien laitteiden hallintajärjestelmäksi valikoitui Ansible. Valintaan vaikutti alkuvuodesta 2020 käymäni Juniper Networksin automaatiokurssi, jossa tutustuin järjestelmään. Ansible on agentiton, eli provisiointivaiheessa ei tarvitse selvittää keinoa saada mitään erillistä ohjelmaa asennettua laitteille. Ansiblelle löytyi myös suoraan Juniper Networksin toteuttamat moduulit konfiguraatioiden ja ohjelmistojen

asentamiselle heidän laittelleen. Ansiblessa pystyi myös käyttämään kohde inventaariona YAML (Yet Another Markup Language) -tiedostojen lisäksi skriptejä niin kutsuttuina dynaamisina inventaarioina. Skripteillä toteutetulla dynaamisella inventaariolla saan laitteet Ansiblelle hallittavaksi Dhcpd:n dhcpd.leases-tiedostosta, mistä näkee sen laitteille jakamat IP-osoitteet.

4.6 Konfiguraatiopohjat

Konfiguraatiopohjat päätin toteuttaa Jinja-pohjamoottorilla. Jinja tuli minulle tutuksi samaisella automaatiokurssilla, missä tutustuin Ansibleen. Monet Ansible-moduulit käyttävät Jinja-pohjia, mukaan lukien Juniper Networksin konfiguraatiomoduuleissa on tuki Jinja-pohjien käytölle. Jinja-pohjat osoittautuivat hyvin helppokäyttöisiksi. Muuttujien lisäksi niissä voi käyttää FOR-silmukoita ja IF-ehtolauseita sekä asettaa arvoja pohjan sisällä.

4.7 Käyttöliittymä järjestelmälle

Ansiblessa ei ole itsessään käytettävissä juuri minkäänlaista käyttöliittymää, vaan pelikirjat tai moduulit ajetaan käsin komentoriviltä. Red Hat tarjoaa kyllä Ansible Tower nimistä tuotetta, missä Ansibleen saa selaimella toimivan graafisen käyttöliittymän. Halusin kuitenkin koota toteutukseni maksuttomista ratkaisuista, joten päätin opetella Python-ohjelmointia ja toteuttaa sillä TUI:n (Text User Interface), eli tekstipohjaisen käyttöliittymän. Pohtiessani, miten käyttöliittymän toteuttaisin Pythonilla, kaverini vinkkasi minulle Npyscreen nimisen Python-viitekehyksen yksinkertaisen TUI:n luomiseen.

5 Käyttöönottojärjestelmän osien asentaminen ja konfigurointi

5.1 Demoympäristö

Demoympäristööni kuuluu Dellin palvelimelle asennettu VMwaren ESXi 6,5 hypervisori -palvelin ja 12-porttinen Juniper Networksin EX2300-C -kytkin. Kytkimeltä kymmenen porttia kuuluu asennusverkkoon. Virtualisointialustaan kiinni menevään porttiin on määritelty kotini lähiverkko ilman VLAN-tunnistetta ja asennusverkko VLAN-tunnisteella. Yksi portti on määritelty lähiverkolleni ilman VLAN-tunnistetta, ja se tulee kiinni tietokoneeseeni. Koneeltani ei ole suoraan pääsyä asennusverkkoon. Kotini lähiverkon

IPv4-aliverkko on 192.168.10.0/24, oletusyhdystäkäytävän osoite on 192.168.1.1 jonka kautta liikenne reitittyy internetiin, sisäverkossa ei ole nimipalvelimia, vaan koneellani ja palvelimella tullaan käyttämään julkisia nimipalvelimia.

Kuvailen järjestelmän pystyttämisen CentOS-käyttöjärjestelmän asentamisesta alkaen. Asennuspalvelin-nimisellä virtuaalipalvelimella, johon CentOS asennetaan, on neljä yhden ytimen prosessoria, 6 Gb keskusmuistua, 100 Gb tallennustilaa, kaksi verkkokorttia (toinen kotini lähiverkossa ja toinen asennusverkossa) sekä CD-asema, jonka “sisällä” on centos.org-sivustolta ladattu CentOS 8 -levykuva. Käyttöjärjestelmän asennuksen aikana virtuaalipalvelinta käytetään VMwaren ESXi-tuotteen selainkonsolista.

5.2 CentOS-käyttöjärjestelmän asentaminen ja konfigurointi

5.2.1 CentOS-käyttöjärjestelmän asentaminen

Kun virtuaalipalvelimen käynnistää CentOS 8 -levykuva asemassaan, käynnistyy pienellä viiveellä CentOS-asennusohjelma. Ensimmäiseltä sivulta valitaan asennuksen aikana käytettävä kieli. Tämän voi jättää oletusarvoon eli amerikanenglanti ja siirtyä seuraavalle sivulle. “Installation summary” -sivulta löytyy muutama asetus, jotka täytyy vaihtaa. Ensin lokalisaatioasetusten alta vaihdetaan suomalainen ulkoasu näppäimistolle ja aikavyöhykkeeksi “Europe/Helsinki”. Järjestelmäasetuksista laitetaan asennuskohteeksi virtuaalipalvelimen kovalevy, verkkoasetusten alta kotini lähiverkossa kiinniolevalle ens192-verkkokortille laitetaan manuaalisesti vapaa osoite 192.168.10.10 ja asennusverkossa kiinniolevalle ens224-verkkokortille laitetaan osoitteeksi 192.168.110.10 ja maskiksi 255.255.255.0. Näiden lokalisaatio- ja järjestelmäasetusten asettamisen jälkeen aloitetaan käyttöjärjestelmän asennus “Begin Installation” -painikkeesta. Kun asennus alkaa, on ruudulla vaihtoehdot syöttää root-käyttäjän salasana ja luoda käyttäjä. Demoympäristön pitämiseksi yksinkertaisena syötän root-käyttäjälle salasanaksi Qwerty123 ja luon samalla salasanalla käyttäjän ville, jolle annan järjestelmänvalvojan oikeudet. Kun asennus on valmis, on meidän vielä ennen käytön aloittamista käynnistettävä palvelin uudelleen asennusohjelman oikeassa alakulmassa näkyvästä “Reboot”-painikkeesta. Ennen kuin palvelin lähtee käynnistymään uudelleen, täytyy vielä hyväksyä loppukäyttäjän lisenssisopimus.

Uudelleenkäynnistyksen jälkeen pääsen asennuspalvelimeen käsiksi SSH:lla yhdistämällä sen lähiverkonosoitteeseen 192.168.10.10. Seuraaavaksi käynnistetään komennolla “sudo nmtui” verkkoasetusten TUI root-oikeuksilla ja asetetaan ens224-verkkokortti aktivoitumaan automaattisesti päälle, kun palvelin käynnistetään. Sitten käydään ottamassa SELinux (Security-Enhanced Linux) kernel -moduuli pois päältä, SELinux parantaa järjestelmän käyttöoikeuksien tietoturvaa, mutta kuten monet muut tietoturvaominaisuudet, se osaltaan myös hankaloittaa järjestelemän konfigurointia. SELinux saa pois päältä muokkaamalla polusta /etc/selinux/config sen konfiguraatiotiedostoa root-oikeuksilla ja vaihtamalla kohdasta “SELINUX=encforcing” “enforcing”-sanan tilalle “disabled”, käynnistetään palvelin vielä uudelleen “sudo reboot” -komennolla [30].

Luodaan firewall-cmd palomuurisovellukseen uusi alue ja siirretään asennusverkon verkkokortti tähän zoneen komennoilla “sudo firewall-cmd --new-zone=nwdeployment --permanent” ja “sudo firewall-cmd --zone=nwdeployment --add-interface=ens224 –permanent”. Sallitaan tarvittavavien palveluiden portit uudella alueella komennoilla “sudo firewall-cmd --zone=nwdeployment --add-service=tftp --permanent”, “sudo firewall-cmd --zone=nwdeployment --add-service=http --permanent” ja “sudo firewall-cmd --zone=nwdeployment --add-service=dhcp --permanent”, käynnistetään palomuuri uudelleen komennolla “sudo firewall-cmd –reload”. Vielä ennen uusia asennuksia ladataan ja asennetaan uusimmat päivitykset komennolla “sudo dnf upgrade”.

5.2.2 DHCP-palvelu

Dhcpd asennettaan komennolla “sudo dnf install dhcp-server” ja hyväksytään y-painikkeella paketin koko ja sormenjälki. Liitteessä 2 on luomani /etc/dhcp/dhcpd.conf-tiedosto, josta Dhcpd saa konfiguraationsa. Konfiguraatio on luotu Dhcpd:n ja Juniper Networksin ZTP:n dokumentaatiosivustojen pohjalta. Juniper Networksin Autoinstallation- ja ZTP-ominaisuuksia käyttäville verkkolaitteille kerrotaan DHCPD-vaihtoehdolla 150 palvelimen IP-osoite, mistä konfiguraatiot ladataan. Konfiguraatiossa luodaan vaihtoehtotila nimellä ZTP, jonka sisään luodaan ZTP:n käyttämät asetukset, jotka sitten kapsuloidaan vaihtoehtoon 43. Lisäksi liitteen konfiguraatiossa käytetään hyödyksi Juniper Networksin valmistamien kytkimien DHCP-palvelimelle lähettämää vendor-class-identifier -merkkijonoa, joka sisältää laitteen valmistajan, mallin ja

sarjanumeron. Esimerkkinä ex2300-c -kytkimen lähettämä merkkijono "Juniper-ex2300-c-12t-HW0217520019". Vendor-class-identifier merkkijonon avulla if-lauseita hyödyntämällä saamme määriteltyä kytkimille malli kohtaisesti ZTP:llä provisioitavan konfiguraation, koska jokaisella laitteella on uniikki sarjanumero, täytyy meidän substring-operaattorilla kohdistaa if-lause vendor-class-identifier -merkkijonon ensimmäiseen 14 merkkiin nollakohdan merkistä alkaen. Konfiguraation luomisen jälkeen ajetaan komennot “sudo systemctl enable dhcpd” ja “sudo systemctl start dhcpd”, joista ensimmäinen merkitsee DHCPD:n palveluksi, että se aloitetaan käynnistyksen yhteydessä. [28; 29; 31; 32.]

5.2.3 Apache HTTP -palvelu

Apache HTTP Server -palvelu asennetaan komennolla “sudo dnf install httpd” ja painetaan y-näppäintä ladattavan paketin koon hyväksymiseksi. Käydään muokkaamassa konfiguraatiotiedostoa /etc/httpd/conf/httpd.conf, korvataan rivi “Listen 80” rivillä “Listen 192.168.110.10:80”. Tekijä määrittelee HTTP-palvelun kuuntelemaan vain osoitteen 192.168.110.10 porttia 80. Seuraavaksi otetaan oletuksena index.html-sivun puuttuessa näytettävä Welcome-sivu pois päältä kommentoimalla #-merkkiä käyttäen konfiguraatiotiedoston “/etc/httpd/conf.d/welcome.conf” neljä ensimmäistä kommentoimatonta riviä alkaen rivistä “<LocationMatch "^/+$">” loppuen riviin “</LocationMatch>”. Tämän jälkeen asetetaan HTTP-palvelin käynnistymään palvelimen käynnistymisen yhteydessä komennolla “sudo systemctl enable httpd” ja käynnistetään palvelu komennolla “sudo systemctl start httpd”. Nyt Apatch HTTP Server tarjoaa osoitteeseen 192.168.110.10:80 HTTP-protokollalla yhdistäville listauksen kansion “/var/www/html/” sisällöstä.

5.2.4 TFTP-palvelu

Koska onnistuin provisioimaan Autoinstallationilla SRX300-mallin palomuureihin konfiguraation vain TFTP-palvelun kautta, asennetaan Tftpd-ohjelma komennolla “sudo dnf install tftp-server”. Vaihdetaan konfiguraatiotiedostosta /usr/lib/systemd/system/tftp.service käytettävä kansio oletuskansiosta /var/lib/tftpboot kansioon /var/www/html. Sitten käsketään palveluiden ladata muutokset konfiguraatiotiedostoista komennolla “sudo dnf daemon-reload”, asetetaan Tftpd

palveluksi komennolla “sudo dnf enable tftp” ja käynnistetään se komennolla “sudo dnf start tftp”.

5.3 Python

5.3.1 Python 3.8.3:n asentaminen

Aloitetaan Python-ympäristön asentaminen asentamalla Pythonin vaatimat sidonnaisuudet. CentOS pakettiryhmä “Development tools” asennetaan komennolla “sudo dnf groupinstall ‘Development Tools’” ja sitten loput sidonnaisuudet asennetaan komennolla “sudo dnf install openssl-devel bzip2-devel libffi-devel ncurses-devel”. Ladataan Python 3.8.3 Python Software Foundationin kotisvuilta kotikansioon komennolla “wget https://www.python.org/ftp/python/3.8.3/Python-3.8.3.tgz ~” ja puretaan paketti komennolla “tar xvf ~/Python-3.8.3.tgz”. Siirrytään paketista purettuun Python-3.8.3-kansioon, ajetaan asennuksen optimointiskripti configure ja asennetaan Python 3.8.3 komennolla “sudo make altinstall”. [33.]

5.3.2 Python Virtualenv -ympäristön luominen

Seuraavaksi luodaan asennetaan Virtualenv Python -paketti, jolla voidaan luoda eristettyjä Python-ympäristöjä. Ideana on, että myöhemmin voi tehdä esimerkiksi testausta varten toisen Virtualenv-ympäristön samalle palvelimelle, jossa voi testata käytettävien Ansible-moduulien ja Python-kirjastojen päivitettyjä versioita ennen kuin päivittää nämä käytössä olevasta ympäristöstä. Asennetaan Pythoniin pakettienhallintajärjestelmää PIP:iä käyttäen Virtualenv-paketti komennolla “pip3.8 install virtualenv --user”. Luodaan Virtualenv-ympäristöä varten kansio ja annetaan kansion omistajuus käyttäjälle ville käyttäen komentoja “sudo mkdir /venv” ja “sudo chown ville /venv”, siirrytään luotuun kansioon ja luodaan komennolla “virtualenv deployment” deployment-niminen Virtualenv-ympäristö. Ympäristön saa aktivoitua komennolla “source /venv/deployment/bin/activate”.

5.3.3 Python-kirjastojen ja Ansiblen asentaminen

Liittessä 3 on listattu kaikki skripteissä ja Ansible-moduuleissa käytetyt Python-kirjastot. Huomaa että listassa on mukana Ansible. Kirjastot asennetaan pip:llä deployment Python-ympäristön sisällä käyttäen komentoa “pip install”. Perään tulee välilyönneillä

eroteltu lista, missä kaikki liitteen 3 paketit ovat ilman versionumeroita. Sudo-oikeuksilla asennettu Ansible asentuu normaalisti kansioon /etc/ansible/, mutta pip komennolla asennettaessa se asentuu Python-ympäristön kansioon, tässä tapauksessa kansioon /venv/deployment/bin/.

5.4 Ansible

5.4.1 Ansiblen konfigurointi

Luodaan ville-käyttäjälle ympäristömuuttuja ANSIBLE_CONFIG, joka kertoo Ansiblelle sen konfiguraatiotiedoston polun. Ympäristömuuttuja luodaan lisäämällä rivi “export ANSIBLE_CONFIG=/venv/deployment/bin/ansible.cfg” /home/ville/.bash_profile-tiedostoon. Muutos astuu voimaan seuraavan sisäänkirjautumisen yhteydessä. Ympäristömuuttujan saa heti käyttöön kirjoittamalla .bash_profile-tiedostoon lisätty rivi suoraan komentoriville. Liittessä 4 on konfiguraationi Ansiblelle. Karsin siitä pois rivit, jotka ei ovät ole käytössä, mutta jätin konfiguroitavat osa-alueet näkyville hakasulkuihin. Uusimpaan GitHub-sivustolla jaettuun esimerkkikonfiguraatioon löytyy linkki Ansiblen dokumentaatiosivuilta. Konfiguraatiossani asetetaan polut Python-tulkille, inventaario- sekä roolikansioille ja logitiedostolle. Lisäksi nostetaan aikakatkaisuarvoja yhteksille ja RPC:eille sekä otetaan SSH-avainten tarkastus pois päältä. Luodaan Ansible.cfg-tiedostoon määritellyt polut, jotka ovat /venv/deployment/app/inventory, /venv/deployment/bin/roles, /venv/deployment/app/log/ ja /venv/deployment/app/tmp.

5.4.2 Ansible inventaario ja ryhmämuuttujat

Luodaan inventaariolle runko. Emme syötä sinne käsin kohteita, kohteet muodostetaan myöhemmin esiteltävässä Python-skriptissä. Luodaan hosts.yaml-tiedosto Ansible.cfg-tiedostossa määriteltyyn polkuun, /venv/deployment/app/inventory. Tiedoston sisältö on nähtävissä liitteessä 5. Tiedostossa määritellään ryhmän all alaryhmä deployment, jolla on kaksi alaryhmää: rdyforsoft ja rdyforconf. Ryhmämuuttujia varten luodaan inventaariokansion alle kansio group_vars. Group_vars-kansioon luodaan deployment-ryhmän muuttujille YAML-tiedosto, ja muuttujat kohdennetaan haluttuun ryhmään YAML-tiedoston nimellä, joka on deployment.yaml. Tiedoston sisältö on nähtävissä liitteessä 6. Deployment.yaml-tiedostossa määritellään SSH-yhteyksissä käytetyn argumentit, joiden avulla varmistetaan, ettei SSH-yhteyksien avaintentarkistus estä yhteyksiä. Jos järjestelmä tallentaisi käyttöönotettavien laitteiden SSH-avaimet niiden sen hetkisille IP-

osoitteille, yhteys seuraavaan laitteeseen, joka saisi DHCP-palvelimelta saman osoitteen, ei toimisi. Tiedostossa määritellään myös käyttöönoton aikana laitteilla käytettävät kirjautumistiedot, käyttäjänimenä root ja salasanana Qwerty123.

5.4.3 Juniper.junos-rooli

Seuraavaksi asennetaan Juniper.junos-rooli Ansible Galaxy -repositoriosta. Ansible-roolit ovat tapa koota ja eristää toisistaan tehtäviä, muuttujia ja tiedostoja. Valmiita rooleja löytyy Ansiblen Galaxy -repositoriosta, seuraavaksi ladattava Juniper Networksin kehittämä Juniper.junos-rooli on lähinnä kokoelma valmiita moduuleita, joihin sisältyy kaikki järjestelmäni tarvitsemat toimenpiteet. Asennetaan juniper.junos-rooli seuraavalla komennolla haluttuun polkuun “ansible-galaxy install --roles-path /venv/deployment/bin/roles/Juniper.junos”.

6 Skriptit, pelikirjat, laitekonfiguraatiot ja muuttujatiedostot

6.1 Ansible-inventaarioskripti

Ansiblessa on mahdollisuus staattisesti luotujen inventaariotiedostojen lisäksi käyttää niin sanottuja dynaamisiainventaarioita. Toimiakseen inventaarioskriptin pitää ulostulossa antaa inventaario JSON-muodossa ja sen tulee pystyä ottamaan vastaan argumentit “--list”, “-c” ja “--count”. Noista ensimmäisen ei käytännössä tarvitse tehdä mitään, mutta “-c” tai “--count” argumentit annettaessa tulee skriptin osata tulostaa ruudulle JSON-muotoisen inventaarion sijasta inventaarion sisältämien kohteiden määrä. Luodaan liitteessä 7 nähtävillä oleva hosts-from-leases.py python-skripti inventaariokansioon, josta Ansible ottaa sen automaattisesti käyttöön. Käyttäjälle ville täytyy antaa oikeus suorittaa skripti. Tämä onnistuu komennolla “chmod 0774 hosts-from-leases.py”. Skriptin alussa täytyy Ansiblea varten määritellä käytettävä Python-tulkki. Skriptissä Python Subprocess -moduulin avulla suoritetaan käyttöönottojärjestelmässä kaksi komentoa, joilla haetaan komentoriviohjelmia grep, sort ja awk komentoriviohjelmia hyödyntäen dhcpd.leases-tiedostosta DHCP-palvelun aktiiviset DHCP-lainat, joiden vendor-class-identifier -merkkijonot ovat “ready-for-conf” tai “ready-for-soft”. Komentojen antamat ulostulo dekoodataan Pythonille käytettävään muotoon ja niistä tehdään listat, joista ne for-silmukalla syötetään targets-sanakirjaan niille kuuluviin avaimiin. Targets-sanakirja on se, joka annetaan skriptistä ulostulona

JSON-muodossa, paitsi jos argumentit -c tai --count on annettu, jolloin ulostulossa annetaan kohteiden yhteenlaskettu määrä.

6.2 ZTP:llä asetettavat konfiguraatiot ja päivityksen jälkeiset konfiguraatiot

ZTP:llä jaettavat konfiguraatiot tulee sijoittaa HTTP- ja TFTP-palvelun juureen kansioon /var/www/html/. Konfiguraatiot on listattuna liitteissä 8, 9 ja 10. Konfiguraatiot ovat hyvin yksinkertaiset. Niissä asetetaan käyttäjälle root-salasanaksi Qwerty123, sallitaan SSH- sekä NETCONF-yhteydet ja asetetaan laitteet hakemaan IPv4-osoitteet DHCP:llä vendor-class-indentifier arvona ready-for-software. Huomioitavana on, että EX2200-sarjan kytkimille käytetään ge-0/0/0 -verkkoliitännän sijaan liitäntää me0, koska tämä laitteen takana sijaitseva hallintaan käytettävä verkkoliitäntä on ainoa, mistä EX2200-sarjan kytkimillä toimii ZTP. SRX300-sarjan palomuureissa käytössä oleva Autoinstallation-ominaisuus ei korvaa olemassa olevaa konfiguraatioita palvelimelta ladattavalla vaan liittää ne yhteen. Tämän takia liitteessä olevassa SRX300-sarjan konfiguraatiossa on delete:-käskyjä. Niillä siivotaan oletuskonfiguraatiossa olevat turhat asetukset pois.

ZTP:llä asetettavissa konfiguraatiossa laitteiden vendor-class-identifier arvoksi asetettiin ready-for-software. Laitteiden päivittämisen jälkeen sen tulisi olla ready-for-conf. Tätä varten luodaan laitteiden päivittämisessä käytettäviä tiedostoja varten kansio /venv/deployment/app/update ja luodaan sinne tiedostot ex2200rdyforconfig.conf, ex2300rdyforconfig.conf ja srxrdyforconfig.conf. Näiden sisälle syötetään vendor-class-identifier arvon muuttamiseen tarvittavat konfiguraation osat. Nämä konfiguraatiot löytyvät liitteestä 11.

6.3 Ansible-pelikirjat

6.3.1 Päivityspelikirja

Luodaan ensin laitteiden päivittämiseen käytettävälle pelikirjalle muuttujatiedosto, ettei pelikirjaan tarvitse koskea, kun halutaan vaihtaa ohjelmistoversioita, joihin laitteet päivitetään. Luodaan /venv/deployment/app/update-kansioon muuttujatiedosto nimeltä updatevars.yml ja syötetään sen sisään YAML-muodossa muuttujat halutuille ohjelmistoversiolle, asennuspakettien nimet päivityksen jälkeiselle konfiguraatiolle ja käytettävälle polulle. Tiedoston sisältö on liitteessä 12.

Luodaan pelikirjoja varten kansio /venv/deployment/app/playbooks ja sinne pelikirja nimellä update-devices.yml. Liittessä 13 on pelikirjan sisältö. Pelikirjassa ensin asetetaan kohteet määrittelevälle hosts-muuttujalle arvoksi ryhmä rdyforsoft, otetaan käyttöön Juniper.junos-rooli ja otetaan käyttöön muuttujatiedostoksi aiemmin luotu updatevars.yml-tiedosto.

Ensimmäisessä toimenpiteessä kerätään kohteilta tietoja ja tallennetaan ne junos-muuttujaan. Tämä muuttuja sisältää mm. laitteen mallin ja ohjelmistoversion. Seuraavaksi tulevat mallikohtaiset toimenpiteet, joissa ajetaan ohjelmistopäivitys laitteille muuttujatiedostoon määritellyllä paketilla, jos laitteen junos-muuttujaan tallennettu ohjelmistoversio ei täsmää muuttujatiedostossa määriteltyyn. Nämä toimenpiteet ajetaan laitteille vain, jos niiden junos-muuttujaan tallennettu malli vastaa when-lausekkeessa määriteltyä mallia. Toimenpiteet myös käynnistävät ohjelmistoasennuksen jälkeen laitteet uudelleen asennuksen viimeistelemiseksi. Seuraavassa toimenpiteessä odotetaan, että laitteet alkavat taas vastaamaan niiden SSH-palvelun portista 22. Viimeisissä toimenpiteissä syötetään laitteille, taas when-lauseella kohdennettuna oikeille malleille, muuttujatiedostoon määritellyt konfiguraatiotiedostot. Pelikirja siis päivittää laitteiden ohjelmistoversiot ja vaihtaa niiden vendor-class-indentifier arvoiksi ready-for-conf. Jos laite on jo halutussa versiossa, vain sen konfiguraatiota muutetaan.

6.3.2 Päivitysskripti

Tarkoitus ei ole, että käyttäjät ajaisivat Ansiblella update-devices.yml-pelikirjaa käsin komentoriviltä. Pelikirja ajastetaan Unix-pohjaisten käyttöjärjestelmien ajastuspalvelulla cronilla viiden minuutin välein ajettavaksi. Luodaan ensin bash-skriptejä varten kansio /venv/deployment/app/bashscripts ja sinne bashs-kripti update-devices, joka aktivoi Python-ympäristön ja sitten ajaa Ansiblella päivityspelikirjan. Samoin kuin inventaarioskriptin tapauksessa meidän täytyy antaa käyttäjlle ville oikeus suorittaa skripti. Käytettävä bash-skripti on liittessä 14. Siinä ensin if-lausekkeessa tarkistetaan, ettei pelikirja ole jo ajossa. Näin varmistetaan, ettei laitteille yritetä ajaa kahta päällekkäistä asennusta, jos ei, niin aktivoidaan Python-ympäristö, ajetaan pelikirja ja käynnistetään DHCP-palvelu uudelleen. DHCP-palvelun uudelleenkäynnistyksellä varmistetaan etteivät vanhat ready-for-software vendor-class-indentifier -arvolla varustetut IP-lainat näy aktiivisina ja inventaarioskriptin mukaan eivät tartu kuin näiden laitteiden tuoreet merkinnän ready-for-conf omaavat lainat. Jotta bash-skriptissä oleva

DHCP-palvelun uudelleenkäynnistyskomento toimii, täytyy meidän sallia sudoers-tiedostossa käyttäjälle ville oikeus ajaa komento “sudo systemctl restart dhcpd.service” ilman salasanakyselyä. Tämä onnistuu, kun /etc/sudoers-tiedostoon lisätään rivi “%ville ALL = NOPASSWD: /bin/systemctl restart dhcpd.service”. Sudoers tiedostoa pitää muokata komennolla visudo, joka käyttää VI-tekstieditoria.

Cronissa ajastetaan tehtäviä kirjaamalla niitä ylös crontab-tiedostoihin. Käyttäjän omaa crontab-tiedostoa pääsee muokkaamaan VI-tekstieditorilla komennolla “crontab -e”, tiedostoon lisätään rivi “*/5 * * * * /venv/deployment/app/bashscripts/update-devices”. Crontabissa rivin alussa ensimmäinen tähti viittaa minuutteihin, toinen tunteihin, kolmas päiviin, neljäs kuukausiin ja viides viikonpäiviin. Lisätty rivi määrittelee, että komento ajetaan joka viides minuutti, joka tunti, joka päivä, joka kuukausi ja joka viikonpäivä.

6.3.3 Tietojenkeräyspelikirja

Luodaan pelikirja print-dev-inf.yml playbooks-kansioon, joka kerää ryhmän rdyforconf-laitteiden IP-osoitteet, mallit ja sarjanumerot käyttöönottoskriptiä varten. Lisäksi luodaan pelikirjaa varten playbooks-kansioon kansio tmp, jonka sisään se tallettaa nuo tiedot. Pelikirjan sisältö liittesssä 15. Pelikirjassa ryhmän rdyforconf-laitteita vasten ajetaan ensin Juniper.junos-roolin tiedonkeruutoimenpide ja seuraava toimenpide kirjoittaa tarvittavat tiedot tiedostoina muodossa “’IP-osoite’,’malli’,’sarjanumero’”. Pelikirjaa ei tulla ajamaan käsin vaan siitä tulee osa käyttöönottoskriptiä.

6.3.4 Testaus- ja konfigurointipelikirjat

Luodaan playbooks-kansioon pelikirjat käyttöönottoskriptille konfigurointia varten, testauspelikirja nimillä commit_check.yml ja konfigurointipelikirja nimellä commit.yml. Näiden sisällöt ovat liitteessä 16. Pelikirjoja tullaan käyttämään käyttöönottoskriptin kautta. Kummassakin pelikirjassa käytetään roolin Juniper.junos juniper_junos_config -toimepidettä. Testauspelikirjassa check-muuttujalle on annettu arvo true ja commit-muuttujalle false, kun taas konfigurointipelikirjassa nämä ovat käänteisesti. Käyttöönottoskripti antaa pelikirjoille niitä ajettaessa muuttujat targets ja configdir. Targets tulee olemaan lista konfiguroitavista kohteista kaksoispisteellä erotettuna ja configdir tulee olemaan kansion polku, missä laitteille joko asetettavat tai laitteilla testattavat konfiguraatiot sijaitsevat. Kummatkin pelikirjat valitsevat määritetystä polusta niiden IP-osoitteiden mukaan nimetyt konfiguraatiotiedostot ja korvaavat niillä kokonaan

laitteilla jo olevan konfiguraation. Testauspelikirja varmistaa, että konfiguraation pystyy ottamaan ajoon ja palauttamaa joko hyväksyvän viestin tai listan virheistä, mutta ei ota konfiguraatiota ajoon. Konfiguraatiopelikirja vain ottaa konfiguraation ajoon. Konfiguraatiopelikirja ei käytännössä pääse palauttamaan skriptille mitään tietoja. Konfiguraation käyttöönoton jälkeen laitteille olevat yhteydet päättyvät aikakatkaisuun niiden IP-osoitteiden ja käyttäjätunnusten vaihtuessa.

6.3.5 Klusterointipelikirja

Luodaan SRX-palomuurit klusteriasetuksille konfiguroiva pelikirja playbooks-kansioon nimellä cluster.yml, jonka sisältö on liitteessä 17. Klusterointipelikirjaa ei tulla ajamaan käsin vaan se on osa käyttöönottoskriptiä, mistä se saa kohteen muuttujaan target ja palomuurille asetettavan klusterin id-arvon. Pelikirja käyttää Juniper.junos-roolin toimenpidettä juniper_junos_srx_cluster.

6.4 Jinja2-konfiguraatiopohjat ja muuttujatiedostot

6.4.1 Jinja2-konfiguraatiopohjat

Luodaan /venv/deployment/app/templates-kansio Jinja2-konfiguraatiopohjia varten ja sinne Jinja2-tiedostot nimillä default.SRX300.j2, default-vpn.SRX300.j2, default.EX2200-24T-4G.j2 ja default.EX2300-24T.j2. Tiedostojen sisällöt ovat liitteissä 18, 19, 20 ja 21. SRX300-muurien konfiguraatiopohjat ovat pitkälti samanlaiset. Erona on, että default-vpn -konfiguraatiopohjassa on VPN-tunnelikonfiguraatiot mukana. Pohjissa on tavallisten muuttujien lisäksi käytetty IF- ja SET-lausekkeita, FOR-silmukoita sekä ipaddr-suodatinta. Esimerkkinä IF- ja SET-lausekkeista voi katsoa SRX300 -palomuurien konfiguraatipohjien alkua. SRX-palomuurien verkkoliitäntöjen nimet muuttuvat konfiguraatiossa, jos ne asetetaan klusteritilaan. Tämä toteutetaan IF-lausekkeella tarkistamalla, onko iscluster-muuttuja arvoltaan True vai False ja SET-lausekkeella vaihdetaan verkkoliitäntöjen muuttujat vastaamaan tarvittavia nimiä. FOR-silmukoilla pohjiin saadaan lisättyä asiat joiden määrät vaihtelevat. Näitä ovat pohjissa lähinnä VPN-tunnelit ja sisäverkot. Yksinkertaisimmat esimerkit näistä on nähtävillä EX2200- ja EX2300-kytkimien pohjissa konfiguraatioiden vlans-osiossa, jossa luodaan VLAN-konfiguraatiot jokaiselle VLANs-sanakirjan sisällä olevalle VLAN:ille. Ipaddr-suodatinta käytetään IP-osoite tietojen manipuloimiseen. Sillä saa esimerkiksi suodatettua ip-osoite/maski tyylillä kirjoitetusta IP-osoitteesta pelkkä IP-osoite ilman

maskia. Esimerkki tämän kätöstä palomuurien pohjien konfiguraatioiden security-osion log-alaosion source-address kohdasssa, johon tuleva wan1ip-muuttuja tulee pitämään sisällään kauttaviivan jälkeen IP-verkon maskin, joka suodatetaan pois lopullisesta konfiguraatiosta. Pohjista on työpaikkani järjestelmien IP-osoitteet korvattu osoitteella 123.123.123.123.

6.4.2 Muuttujatiedostot

Luodaan alkuun /venv/deployment/app-polkuun tiedosto template-list.yml. Seuraavaksi luodaan kansio /venv/deployment/app/variablefiles ja tänne tiedostot default.yml, default-vpn.yml, globalvariables.yml ja hardcodedvars.yml. Kaikkien näiden tiedostojen sisältö löytyy liitteestä 22. Template-list.yml on yksinkertainen lista käytettävissä olevista konfiguraatiopohjista. Näin käyttöönottoskriptin ei tarvitse parsia templates-kansion sisältöä ja sinne voi luoda uusia konfiguraatiopohjia, joita voi testata ennen kuin ottaa ne käyttöönottoskriptille käyttöön. Default.yml ja default-vpn.yml sisältävät samannimisten konfiguraatiopohjien muuttujat. Globalvariables.yml sisältää muuttujat, jotka ovat kaikkien konfiguraatiopohjien käytettävissä, ja hardcodedvars.yml sisältää muuttujat, jotka on koodattu käyttöönottoskriptin sisään. Hardcodedvars.yml-tiedostoa ei suoraan käytetä mihinkään. Tarkoituksena on, että sieltä voi tarkistaa, mitä muuttujia on skriptin luotuna, etteivät käyttäjät uusia konfiguraatiopohjia tehdessään vahingossa nimeä näiden kanssa päällekkäisiä muuttujia.

Luodaan kansio /venv/deployment/app/floatingvars. Tämän kansion sisään luodaan YAML-tiedostot konfiguraatiossa käytettäville lukumäärältään vaihteleville asioille. Jos konfiguraatiopohja käyttää muita lukumäärältään vaihtelevia asioita kuin sisäverkkoja, niin konfiguraatiopohjalle luodaan sen nimellä YAML-tiedosto, josta käyttöönottoskripti voi lukea, mitä näistä määrältään vaihtelevista asioista konfiguraatiopohja käyttää. Luodaan tänne siis tiedostot yhdestä kymmeneen tyylillä VLANs.1.yml päättyen VLANs.10.yml:ään ja samalla tavalla VPNs.1.yml päättyen VPNs.10.yml:ään. VPNs.10.yml- ja VLANs.10.yml-tiedostojen sisällöt liitteissä 23 ja 24. Muiden tiedostojen sisältö voidaan johtaa näistä. Esimerkiksi VLANs.5.yml-tiedoston sisältö on muuten sama kuin tiedoston VLANs.10.yml, mutta se päättyy VLAN:iin numero viisi. Luodaan vielä tiedosto default-vpn.yml, jonka sisältö on liitteessä 25. Tiedostossa listattu default-vpn -pohjalle käyttöön määrävaihtuvista asioista vain VPNs, sillä VLANs on käytössä jokaisessa pohjassa oletuksena.

6.5 Loput bash-skriptit

6.5.1 Siivousskripti

Luodaan kansioon /venv/deployment/app/bashscripts skriptit dailyclean ja monthlyclean. Dailyclean poistaa kansioiden /venv/deployment/app/tmp/ ja /venv/deployment/app/playbooks/tmp/ sisällöt. Ne ajastetaan crontabiin suoritettavaksi joka yö. Monthlyclean-skripti poistaa kansion /venv/deployment/app/log/ sisällön, ja se ajastetaan kuukausittain ajettavaksi. Skriptien sisällöt ovat liitteessä 25.

6.5.2 Käynnistysskripti

Luodaan bashscripts-kansioon skripti nimellä start-deploymentscript, joka aktivoi käyttäjän ville oikeuksilla Python-ympäristön ja suorittaa käyttöönottoskriptin. Luodaan sitten /etc/profile.d/-kansioon deploy.sh-niminen skripti, jossa asetetaan skriptille start-deploymentscript alias deploy. Näin käyttäjien ei tarvitse kuin kirjoittaa deploy terminaaliin ja syöttää käyttäjän ville salasana ajaaksen käyttöönottoskriptin. Näiden skriptien sisällöt ovat liittessä 26.

6.5.3 Inventaarioskripti

Luodaan bashcripts-kansioon skripti nimellä inventory, joka käyttäjän ville oikeuksilla aktivoi Python-ympäristön ja ajaa komennon “ansible-inventory –list”. Luodaan /etc/profile.d/-kansioon inventory.sh-niminen tiedosto, jossa asetetaan juuri luodulle inventaarioskriptille alias inventory. Kummankin tiedoston sisältö on liittessä 25. Näin käyttäjä voi tarkistaa käytössä olevat laitteet helposti komennolla inventory.

6.6 Käyttöönottoskripti

6.6.1 NPSAppManaged-luokka

Luodaan /venv/deployment/app-polkuun tiedosto deploymentscript.py, jonka sisältö on liitteessä 27. Skriptissä on Npyscreen-viitekehyksellä toteutettu tesktipohjainen käyttöliittymä, jolla Ansible-pelikirjoja ja inventaarioskriptiä hyödyntämällä konfiguroidaan verkkolaitteita Jinja2-pohjien ja käyttäjän antamien muuttujien pohjalta luoduista konfiguraatioista ja luodaan CSV-tiedostot liitännäisille järjestelmille.

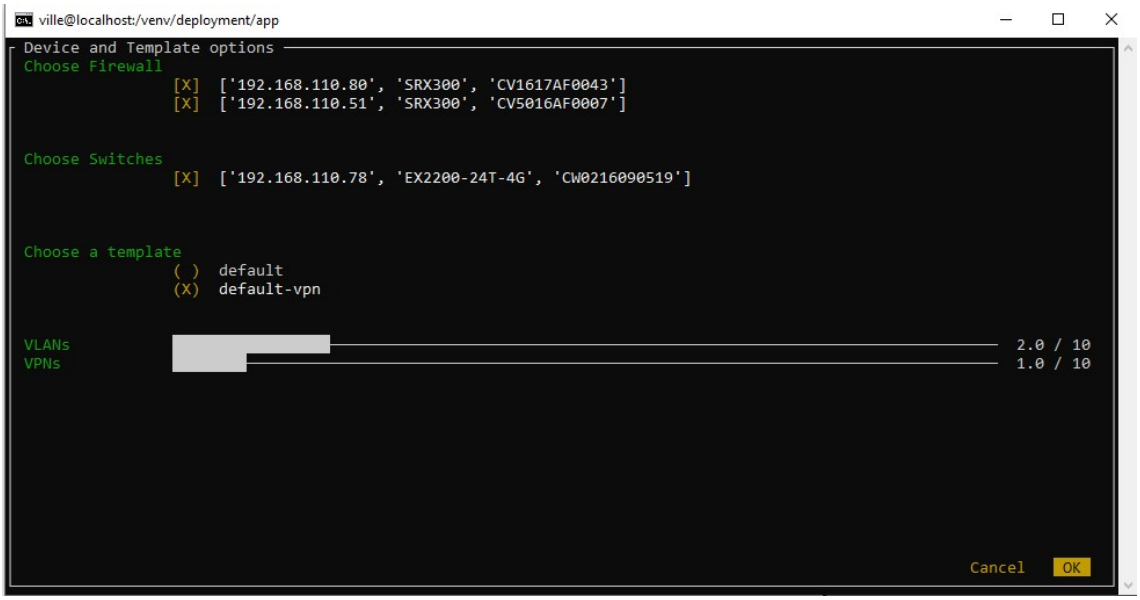
Käyttöönottoskriptissä luodaan Npyscreen-viitekehyksen NPSAppManaged-luokan olio, joka hoitaa sen sisälle luotavien lomakkeiksi kutsuttujen olioiden esittämistä. Yksi lomake vastaa yhtä käyttäjälle näytettyä näkymää. Lomakkeiden sisälle luodaan widgeteiksi kutsuttuja olioita, jotka luokan mukaan antavat käyttäjän esimerkiksi valita asioita listasta tai syöttää tekstiä. Skriptissä käytetään seitsemän eri tyypin lomaketta, jotka käymme seuraavaksi läpi siinä järjestyksessä kuin ne käyttäjälle näytetään.

6.6.2 DeploymentApp-luokka

Käyttöönottoskriptin riviltä 790, josta suoritus alkaa, näkee että skriptissä luodaan luokan DeploymentApp olio nimellä deploymentApp ja suoritetaan tämän run-metodi. Tämä suorittaa koko ohjelman. DeploymentApp-luokka on luomani aliluokka NPSAppManaged-luokasta. Luodun olion luokan koodi alkaa riviltä 736. DeploymentApp -luokkaan määritellään eri lomakkeissa käytettävät muuttujat. Tässä vaiheessa mielenkiintoisia näistä ovat muuttujat readyvardic, templates, fvarlist, switches ja firewalls. Readyvardic on sanakirja, johon tallennetaan liitännäisten järjestelmien CSV-tiedostojen tulostamiseen käytettävät muuttujat ja niistä Jinja2-pohjiin yhdistettynä muodostetaan konfiguraatiotiedostot. Heti alussa readyvardic-sanakirjaan lisätään globalvariables.yml-tiedoston muuttujat. Muuttujaan templates haetaan funktiolla getTemplates tiedoston template-list.yml sisältämä lista käytössä olevista eri Jinja2-pohjista. Fvarlist muuttujaan haetaan getAllfvar-funktiolla lista kaikista konfiguraatiopohjissa käytettävistä määrävalintaisista asioista. Tällä hetkellä näitä on VPNs ja VLANs. Switches- ja firewalls-muuttujat sisältävät listat käytettävissä olevista kytkimistä ja palomuureista. Sisältö näihin listoihin saadaan getDevices-funktiosta, joka ajaa tietojenkeräyspelikirjan saadakseen kaikkien Ansiblen rdyforconf-ryhmään kuuluvien laitteiden IP-osoitteet, mallit ja sarjanumerot pilkuilla eroteltuina, ja palauttaa nämä mallin nimien alun mukaan switches- tai firewalls -listohin. Lisäksi määritellään tapahtumankäsittelijät, jotka ohjelman kaatuessa tai käyttäjän keskeyttäessä suorituksen painamalla ctrl- ja c-näppäintä samanaikaisesti poistetaan ohjelman käytönaikana luomat väliaikaiset tiedot. Metodi onStart suoritetaan- kun ohjelma aloitetaan run-metodilla, onStart-metodissa lisätään addForm-metodilla lomakkeita. Lomakkeita on seitsemää eri tyyppiä, jotka ovat Npyscreen viitekehyksen FormMultiPageAction-luokasta tekemiäni alaluokkia. Ensimmäinen käyttäjälle näytettävä määritellään tunnisteella MAIN. Se on luokaltaan DevicesAndTemplates.

6.6.3 DevicesAndTemplates-lomake

DevicesAndTemplates-lomakkeen koodi alkaa riviltä 426. Lomakkeelle lisätään monivalinta-widgetit palomuureille ja kytkimille. Valitse yksi -tyyppinen widget konfiguraatiopohjille ja liukuvalitsin-widgetit määrävalintaisille asioille. Palomuurien ja kytkimien monivalinta widgeteille vaihtoehdot tulevat firewalls- ja switches-listoista, konfiguraatiopohja-widgetin vaihtoehdot tulevat templates-listasta ja liukuvalitsimia luodaan jokaiselle fvar-listalta löytyvälle asialle. Widgetit tulevat ruudulle näkyviin siinä järjestyksessä, kun ne on luotu. Liukuvalitsimet piilotetaan, kunnes käyttäjä valitsee konfiguraatiopohjan, jolloin tarkistetaan, mitä määrävalintaisia asioita konfiguraatiopohja tarvitsee. Tarvittavat muutetaan näkyviksi.



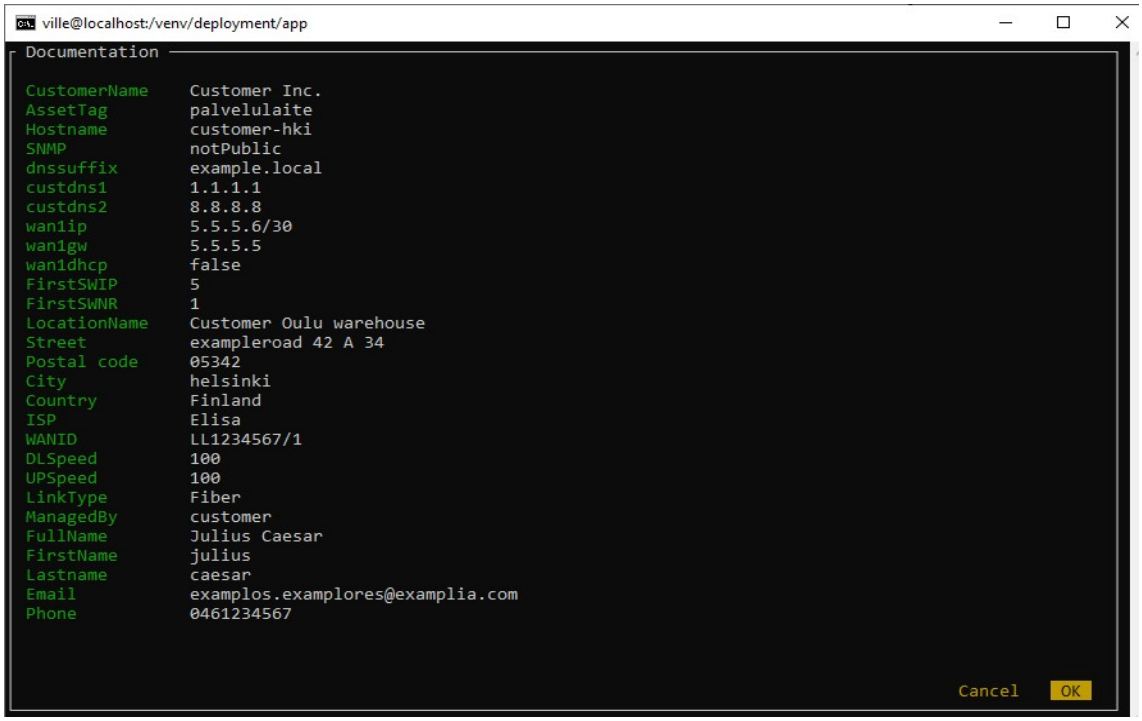
Kuva 6. DevicesAndTemplates-lomake.

Ok-painikeeseen on määritelty muutamia tarkistuksia. Varmistetaan esimerkiksi, että on valittu vähintään yksi konfiguroitava laite ja konfiguraatiopohja. Jos ei ole niin käyttäjälle näytetään ilmoitus ongelmasta. Mielenkiintoisin näistä tarkistuksista on tiedoston /venv/deployment/tmp/lockfile.yml tarkistus, jossa varmistetaan, ettei valittuja laitteita ole joku toinen käyttäjä jo valinnut. Jos laitteet ovat vapaat, ne kirjataan tiedostoon, ettei toiset käyttäjät voi valita niitä. Ok-painike myös tallentaa readyvardic-sanakirjaan avaimelle iscluster joko True tai False riippuen, onko käyttäjä valinnut kaksi palomuuria vai ei. Kun tarkistuksista päästään läpi, tallennetaan käyttäjän valinnat omiin muuttujiinsa ja käyttäjälle näytetään seuraava lomake, joka on tunnisteeltaan ja luokaltaan

Documentation. Cancel-painikkeesta seuraavaksi lomakkeeksi asetetaan None, jolloin ohjelman suoritus lopetetaan.

6.6.4 Documentation-lomake

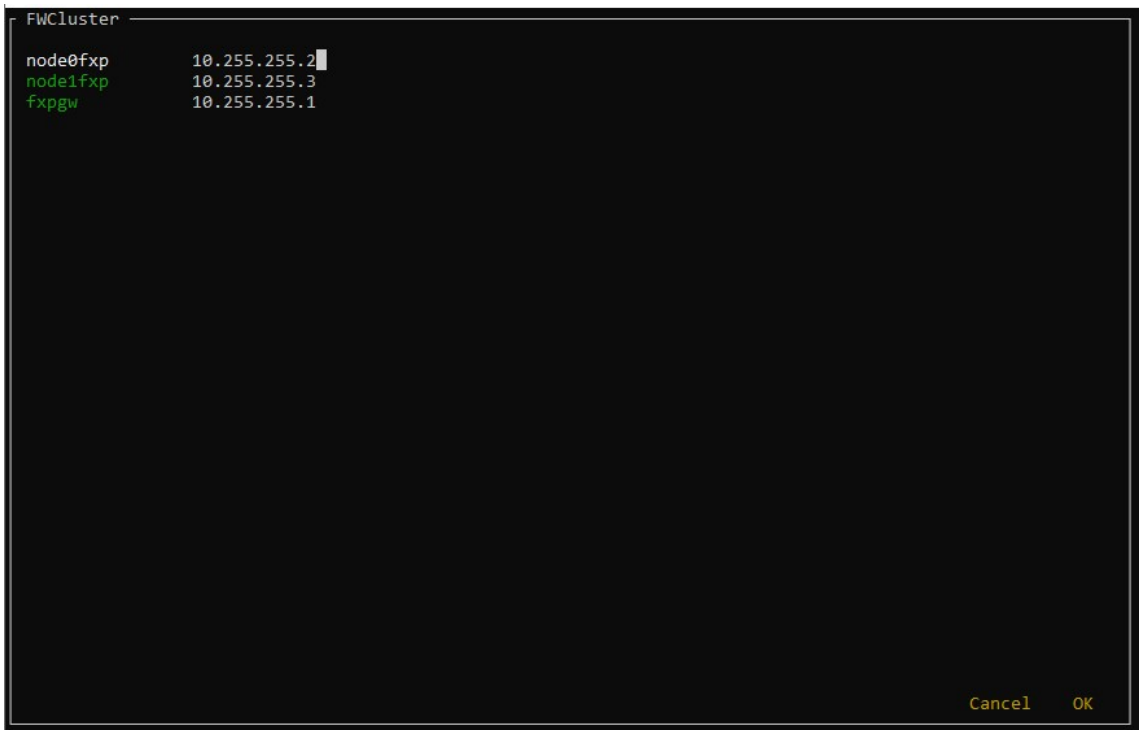
Documentation-lomakkeessa on liuta widgettejä, joissa on otsikko ja tekstinsyöttökenttä, johon käyttäjä antaa arvon otsikon mukaiselle muuttujalle. Lomakkeen koodi alkaa riviltä 526. Syötettävät asiat on listattu hardcodedvars.yml-tiedostoon, mutta niitä ei ladata sieltä ohjelmaan vaan ne on koodattu sinne käsin. Nämä muuttujat on koodattu käsin, koska ne ovat oleellisia tuotettaville CSV-tiedostoille. Cancel-painikkeesta poistetaan lockfile-tiedostosta sinne listatut laitteet, jotka käyttäjän valinnat olivat sinne listanneet lukittaviksi ja siirrytään takaisin edelliseen lomakkeeseen. Ok-panikkeesta tallennetaan annetut tiedot readyvardic-sanakirjaan sekä tarkistetaan, oliko iscluster-muuttujan arvo True vai False. Jos se on True, niin seuraavaksi lomakkeeksi asetetaan FWCluster. Jos arvo on False, siirrytään käyttäjän valitseman konfiguraatiopohjan mukaan nimettyyn TemplateVariables-luokan lomakkeeseen.



Kuva 7. Documentation-lomake.

6.6.5 FWCluster-lomake

FWCluster-lomake on hyvin yksinkertainen ja pitää sisällään vain muutaman otsikoidun tekstinsyöttö-widgetin, joihin käyttäjä syöttää SRX-palomuurien klusterikonfiguraatioiden vaatimat tiedot. Lomakkeen koodi alkaa riviltä 574. Cancel-painike siirtää käyttäjän takaisin Documentation-lomakkeeseen ja Ok-painikkeesta siirrytään tarvittavaan TemplateVariables-luokan lomakkeeseen.

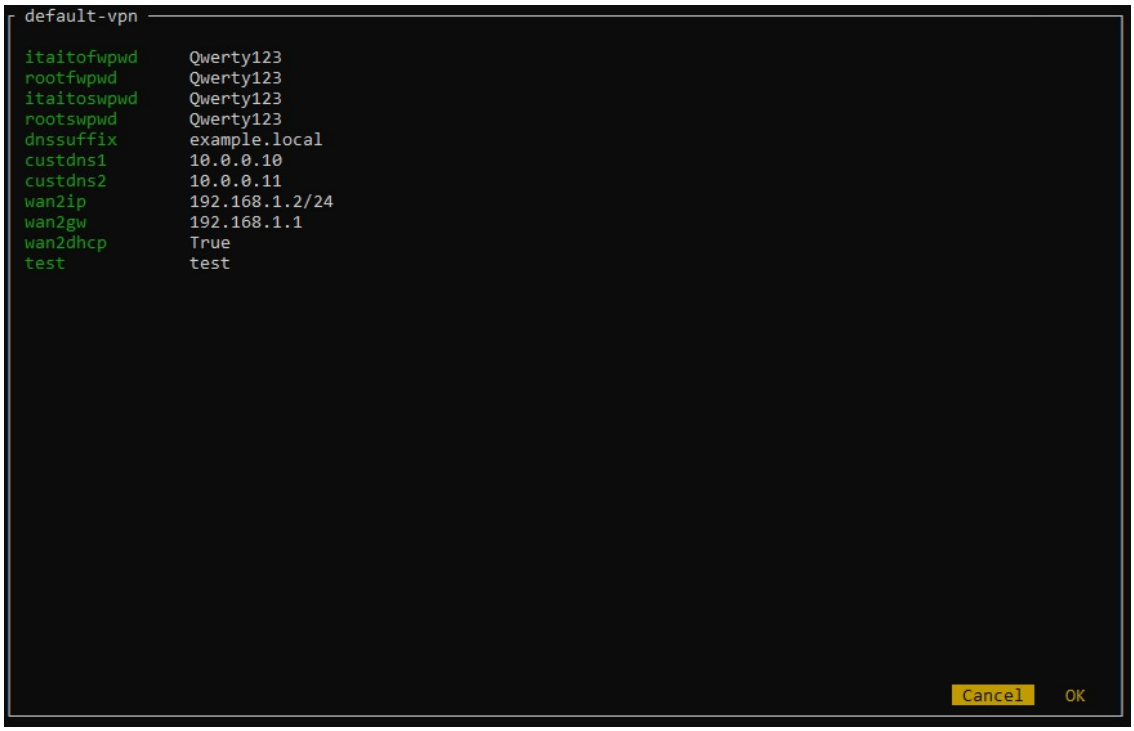


Kuva 8. FWCluster-lomake.

6.6.6 TemplateVariables-luokan lomakkeet

TemplateVariables-luokan lomakkeet ovat ensimmäinen lomaketyyppi, jota luodaan enemmän kuin yksi. Näiden koodi alkaa riviltä 593. Lomakkeita luodaan yksi jokaista templates.yml-tiedostossa listattua konfiguraatiopohjaa kohden tunnisteena konfiguraatiopohjan nimi. Näiden sisältämät otsikoidut tekstinsyöttöwidgetit saavat sisältönsä /venv/deployment/app/variablefiles/-polusta konfiguraatiopohjien nimillä olevista YAML-tiedostoista, joihin on listattu konfiguraatiopohjan vaatimat muuttujat. Käyttäjälle näytetään näistä oikea lomake hänen ensimmäisellä sivulla valitsemansa konfiguraatiopohjan mukaan. Cancel-painikkeesta siirrytään joko FWCluster- tai Documentation-lomakkeeseen riippuen iscluster-muuttujan arvosta. Ok-painikkeesta

syötetyt muuttujat tallennetaan readyvardic-sanakirjaan ja seuraavaksi lomakkeeksi asetetaan FloatingVariables-luokan lomake, se mikä näistä lomakkeista esitetään käyttäjälle riippuu käyttäjän valitseman konfiguraatiopohjan käyttämistä määrältään vaihtelevista asioista ja käyttäjän niille ensimmäisessä lomakkeessa liukuvalitsimella asettamasta määrästä.



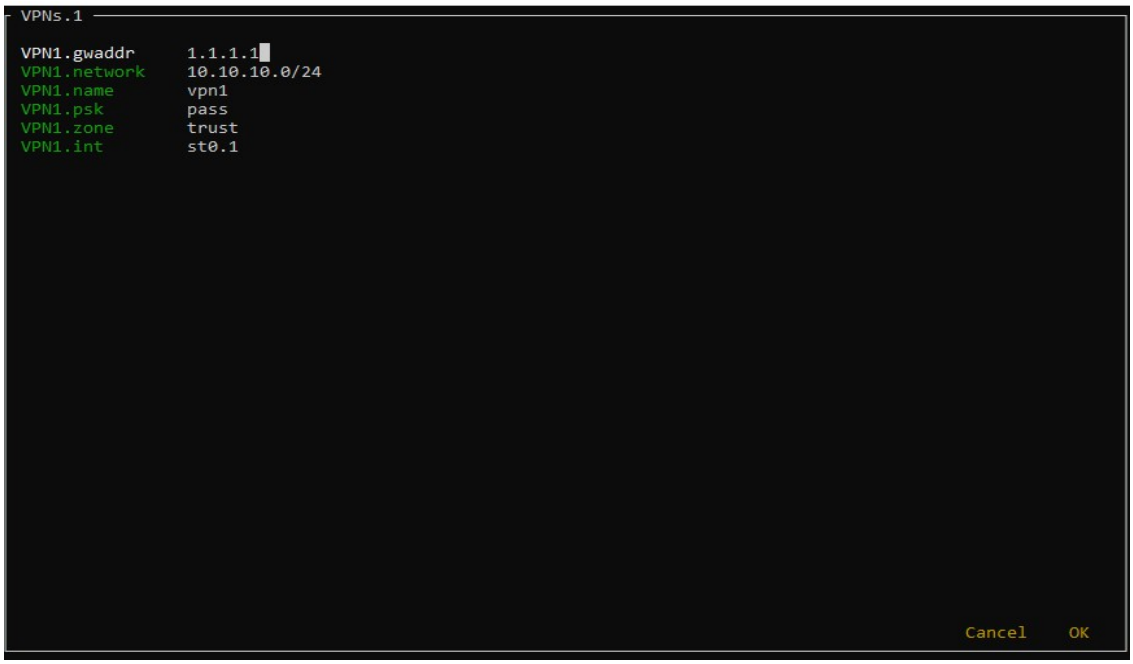
Kuva 9. default-vpn konfiguraatiopohjan lomake TemplatesVariables-luokasta.

6.6.7 FloatingVariables-luokan lomakkeet

FloatingVariables-luokan lomakkeita luodaan kymmenen jokaista määrältään vaihtelevaa asiaa kohden, eli tällä hetkellä näitä on kymmenen lomaketta VPN-konfiguraatioille ja kymmenen lomaketta VLAN-konfiguraatioille. Lomakkeiden tunnisteet ovat asian nimi ja määrä pisteellä eroteltuna. Luokan koodi alkaa riviltä 617. Jokainen lomake saa sisältönsä /venv/deployment/app/floatingvariables-kansiossa sijaitsevista YAML-tiedostoista, jotka on nimetty asian ja lukumäärän mukaan. Esimerkiksi lomake VLANs.5.yml sisältää sanakirjat viiden eri VLAN:in konfiguraatioille. Koska määrältään vaihtelevia asioita voi olla useampia, tämän luokan lomakkeita voidaan näyttää käyttäjälle useampia. Cancel-painikkeesta siirrytään joko edelliseen FloatingVariables-luokan lomakkeeseen, jos edellinen lomake oli tätä luokkaa, tai käytettyyn TemplateVariables-luokan lomakkeeseen. Ok-painike tallentaa annetut

muuttujat readyvardic-sanakirjaan siirtää käyttäjän seuraavaan FloatingVariables-luokan lomakkeeseen tai jos kyseessä oli näistä viimeinen lomake, niin käyttäjä siirretään Verify-lomakkeeseen. Jos kyseessä oli viimeinen FloatingVariables-luokan lomake, Ok-painike luo myös readyvardic-sanakirjan muuttujien ja Jinja2-pohjien perusteella konfiguraatiotiedostot jokaiselle valitulle laitteelle. Jos valitulle laitteelle ei valitulla konfiguraatiopohjalla löydy omaa konfiguraatiopohjaa, käytetään laitteen mallin mukaista default-nimistä konfiguraatiopohjaa. Konfiguraatioista luodaan myös tiedostot /var/www/html/-kansion alle skriptin suorituskellonajan mukaan luotuun alikansioon. Tänne luodaan myös tiedosto, johon on listattu readyvardic-sanakirjan sisältö. Lisäksi CSV-tiedostot luodaan tässä vaiheessa samaan polkuun. Konfiguraatio, CSV-tiedostot ym. luodaan generateConfig()-funktiossa, jonka koodi alkaa riviltä 217.

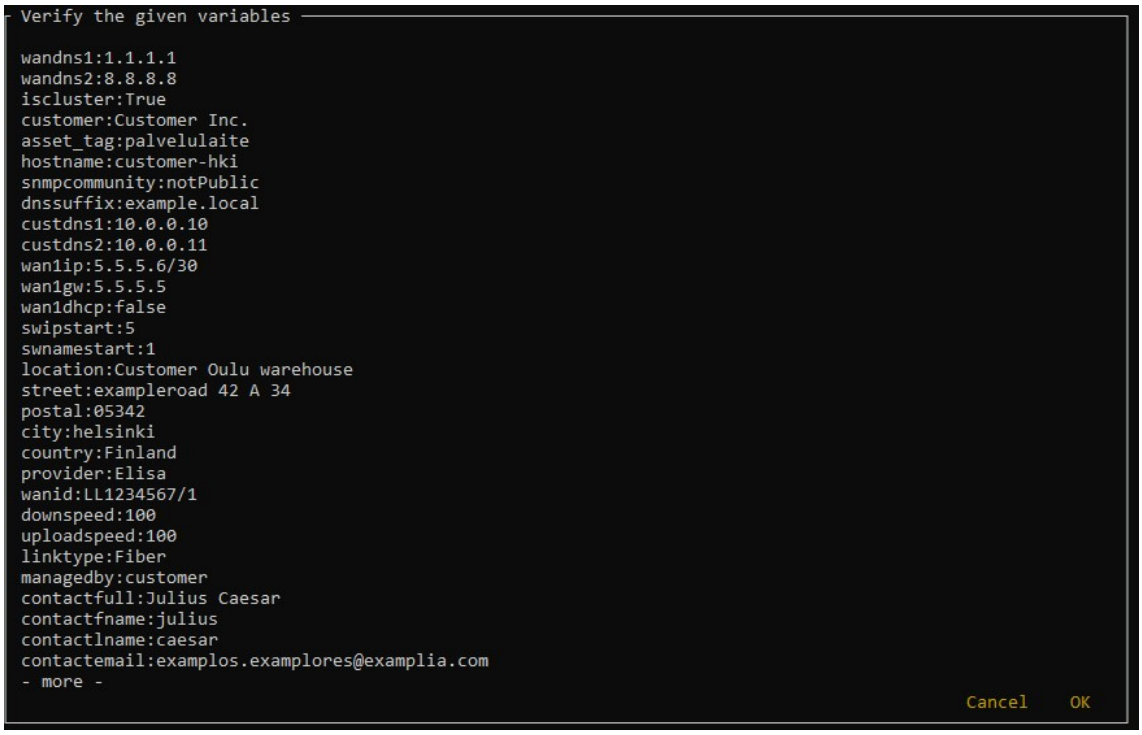
Kuva 10. FloatingVariables-luokan VLANs.2-lomake.



Kuva 11. FloatingVariables-luokan VPNs.1-lomake.

6.6.8 Verify-lomake

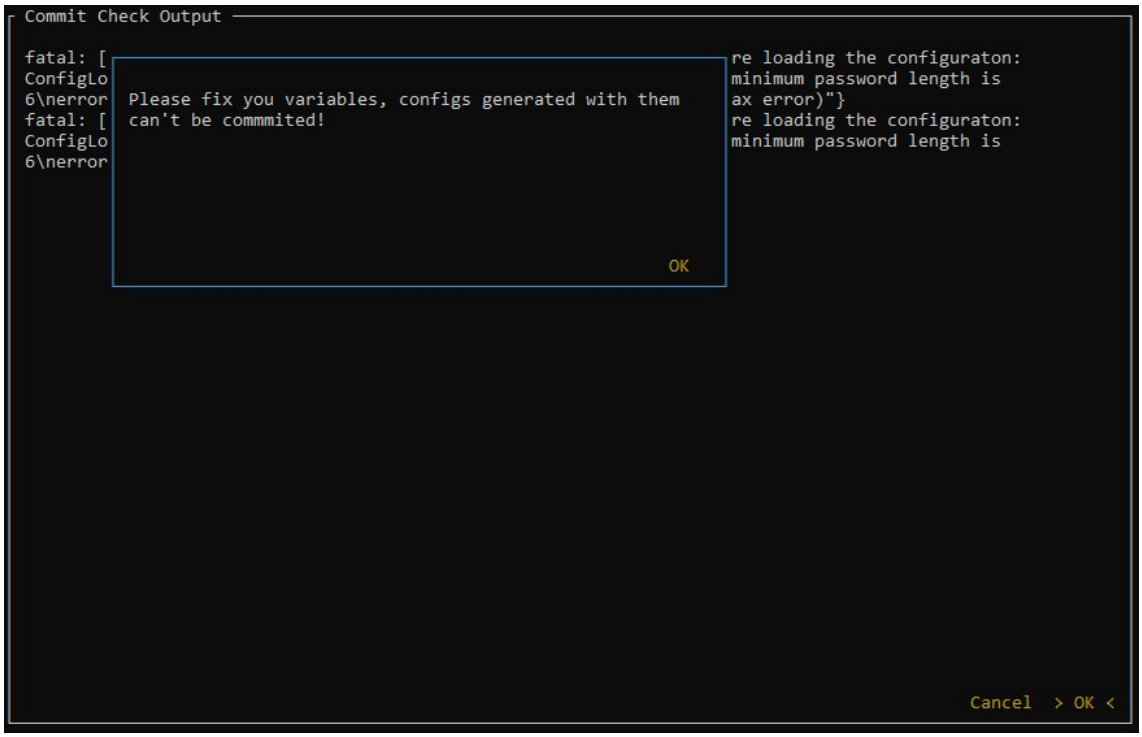
Verify-lomakkeessa käyttäjälle näytetään readyvardic-sanakirjan sisältö, jotta hän voi tarkistaa antamansa muuttujat. Nämä tosin on helpompi tarkistaa käyttöönottopalvelimen käyttöönottoverkon osoitteessa olevalta HTTP-sivulta. Verify-lomakkeen koodi alkaa riviltä 675. Cancel-painike siirtää käyttäjän takaisin viimeiselle FloatingVariables-luokan lomakkeelle ja Ok-painike siirtää käyttäjän CommitCheckOutput-lomakkeeseen.



Kuva 12. Verify-lomake.

6.6.9 CommitCheckOutput-lomake

CommitCheckOutput-lomake ajaa ennen lomakkeen esittämistä käyttäjälle testauspelikirjan. Pelikirjan palauttamat tiedot näytetään lomakkeessa käyttäjälle. Lomakkeen koodi alkaa riviltä 694. Cancel-painikkeesta siirrytään takaisin Verify-lomakkeeseen. Ok-painikkeesta tarkistetaan, ettei testauspelikirja palauttamassa tekstissä ole FAILED!-merkkijonoa. Jos on, käyttäjälle näytetään viesti, ettei konfiguraatiota voi asettaa laitteelle. Jos ei ole, niin valituille laitteille asetetaan niiden konfiguraatiot. Poikkeuksena on tilanne, jossa iscluster-muuttujan arvo on True. Tällöin palomuurit asetetaan klusteritilaan, ja ne käynnistyvät uudelleen. Käyttäjä joutuu tässä tapauksessa syöttämään luodun konfiguraation palomuureille käsin. Koska kyseessä on viimeinen lomake lockfile-tiedostosta, poistetaan sinne käyttäjän valitsemista laitteista luodut rivit ja seuraavaksi lomakkeeksi asetetaan None, mikä päättää ohjelman.



Kuva 13. CommitCheckOutput-lomake ja virheilmoitus konfiguraatiosta.

6.7 Ohjeistukset

6.7.1 Muuttujien selvennökset

Luodaan /var/www/html/help/ ja sinne tiedostot VLANs.txt ja hardcodedvars.txt. Tiedostot sisältävät selitykset käyttöönottoskriptiin syötettäville muuttujille niissä tapauksissa, kun esimerkki ei ole ilmiselvä. Näiden sisältö on liitteessä 27. VPN-tunneleiden ja tämänhetkisten konfiguraatiopohjien muuttujat ovat niin selvät, ettei niistä ole tarvetta kirjoittaa erikseen ohjeistusta.

6.7.2 Järjestelmän käyttäminen

Käyttäjän purettua toimitettavat laitteet laatikoista tulee hänen kytkeä ne ge-0/0/0 -verkkoliitännästä käyttöönottoverkkoon. Poikkeuksena ovat EX2200-sarjan kytkimet, jotka tulee kytkeä takana olevasta me0-portista, ja tämän jälkeen kytkeä virtajohto. Tässä vaiheessa kuluu tovi aikaa ennen kuin laitteet ovat käynnistyneet ja päivitysskripti saa ne päivitettyä, mikä voidaan käyttää tarvittavan tiedon keräämiseen. Jos käyttäjä tarvitsee muistutusta tarvittavista tiedoista, hän voi tarkistaa YAML-tiedostot käyttöönottopalvelimen HTTP-sivuilta.

Käyttäjä kirjautuu hänelle luoduilla tunnuksilla käyttöönottopalvelimelle SSH-yhteydellä. Hänen olisi hyvä ohjata SSH-yhteyden läpi portti omalta koneeltaan käyttöönottopalvelimen käyttöönottoverkon osoitteen HTTP-porttiin, jotta hän voi yhdistää oman koneensa eteenpäinpäinohjattuun porttiin nähdäkseen käyttöönottopalvelimen HTML-kansion sisällä olevat kansiot ja tiedostot. Tuo yhteys kannattaa tallentaa käytettävään SSH-klienttiin, jotta porttiohjausta ei tarvitse aina tehdä uudelleen. Käyttäjä voi tarkistaa, onko laitteet valmiita konfiguroitavaksi komennolla inventory. Jos laitteita näkyy readyforconf-ryhmässä odotettu määrä, ne ovat todennäköisesti valmiit, ellei joku muu ole käyttämässä järjestelmää samaan aikaan. Käyttöönottoskriptin voi suorittaa komennolla deploy. Ensimmäiseltä sivulta käyttäjä valitsee toimitettavat laitteet ja käytettävän konfiguraatiopohjan sekä määrävalinnaisten asioiden lukumäärät. Tämän jälkeen hän syöttää pyydetyt muuttujat aina seuraavaan lomakkeeseen siirtyen, kunnes tullaan Verify-lomakkeeseen, jolloin hänen olisi hyvä pikaisesti tarkistaa antaneensa muuttujat, ettei niissä ole silmäänpistäviä virheitä. Syötetyt muuttujat on helpompi tarkistaa käyttöönottopalvelimen HTTP-sivulta käyttöönottoskriptin suoritusajankohdan mukaan nimetystä alikansiosta, johon on tulostettu lista muuttujista ja käyttäjän niille antamista arvoista.

Tarkastamisen jälkeen käyttäjä voi siirtyä seuraavaan lomakkeeseen, joka käy testaamassa konfiguraatiot laitteilla ja näyttää joko laitteiden ilmoittamat virheet tai hyväksyvät kuittaukset. Viimeinen Ok-painallus saa järjestelmän asettamaan konfiguraatiot laitteille ja tulostaa liitännäisten järjestelmien CSV-tiedostot HTTP-sivulle. Poikkeuksena ovat tilanteet, joissa käyttäjä on valinnut enemmän kuin kaksi muuria. Tällöin muurit asetetaan klusteritilaan ja käynnistetään uudelleen. Käyttäjän tulee hakea konfiguraatiot /venv/deployment/app/tmp/-kansiosta ja syöttää laitteille käsin, vaihtoehtoisesti hän voi kopioida konfiguraatiot HTTP-sivulta, mutta tällöin hänen tulee korjata konfiguraatioon sieltä pois siivotut VPN-tunneleiden ja käyttäjien salasanat.

6.7.3 Tuettavien mallien lisääminen

Tällä hetkellä on mahdollista lisätä vain Juniper Networksin verkkolaitteita EX- ja SRX-sarjoista. Uutta mallia lisättäessä on otettava huomioon DHCP-palvelun konfiguraatiot ja ZTP:llä provisioitava konfiguraatio. Kannattaa DHCP-palvelun konfiguraatioissa ohjata ne laitteet käyttämään samaa konfiguraatiota, minkä osalta tämä on mahdollista, esimerkiksi kaikkien EX2300-sarjan kytkimien, porttimäärästä ja muista eroista riippumatta, on mahdollista käyttää nykyistä ex-skeleton.config -konfiguraatiota.

Seuraavaksi malli on lisättävä päivityspelikirjaan. Tätä varten on suositeltu versio kirjattava updatevars.yml-tiedostoon ja suositellun ohjelmistoversion asennuspaketti on siirrettävä palvelimelle. Lopuksi mallille on luotava jokaista eri tyyppistä konfiguraatiopohjaa kohden Jinja2-tiedostot. Voi kopioida olemassa olevan pohjan, jos laitteelle sopii sama konfiguraatio jonkun järjestelmässä jo olevan mallin kanssa.

6.7.4 Konfiguraatiopohjan lisääminen

Uuden konfiguraatiopohjan luomiseksi täytyy jokaiselle laitemallille luoda oma Jinja2-tiedosto templates-kansioon. On mahdollista käyttää samaa tiedostoa eri malleille jos mallit tukevat samaa konfiguraatiota. Juniper Networksin tapauksessa mallien välisissä konfiguraatioissa on erona lähinnä verkkoliitäntöjen nimet ja määrä, jolloin eri malleille on helppo tehdä Jinja2-pohjat, kun niiden konfiguraatiot ovat niin samanlaiset. Lisäksi konfiguraatiopohjan nimi on lisättävä template-list.yml-tiedostoon. Jos konfiguraatiopohjaan lisätään käyttöön muita määrävalintaisia asioita kuin VLANs, niin nämä asiat on lisättävä /venv/deployment/app/floatingvars-kansioon konfiguraatiopohjan mukaan nimettyyn YAML-tiedostoon sanakirjan avaimina, ks. mallia default-vpn.yml. Jos konfiguraatiopohja vaatii muuttujia, jotka eivät ole niiden nimen perusteella ilmiselviä, kannattaa luoda HTTP-sivun help-kansioon ohjetiedosto selitteiden kanssa

6.7.5 Määrävalintaisten asioiden lisääminen

Konfiguraatiopohjille uuden määrävalintaisen asian luomiseksi käytettäväksi täytyy lisätä /venv/deployment/app/floatingvars-kansioon kymmenen YAML-tiedostoa, jotka on nimetty asian ja sen lukumäärän mukaan. Tiedostojen tulee sisältää nimenmukainen määrä sanakirjoja YAML-muodossa, jotka sisältävät konfiguroitavan asian tarvitsemat muuttujat. Katsoa mallia VLANs- ja VPNs-tiedostoista. Jos asia sisältää muuttujia, jotka eivät ole nimenperusteella selkeitä, on hyvä luoda HTTP-sivun help-kansioon selite tiedosto.

7 Yhteenveto

Työ on mielestäni onnistunut, mutta ei kuitenkaan täysin valmis. Uskon, että parannettavaa vielä löytyy, varsinkin käyttöönottoskriptistä. Käyttöönottoskripti on ensimmäinen ohjelmointiprojektini, ja se näkyy tuloksessa. Työ kuitenkin on

käyttöönottovalmis. Järjestelmä varmasti sujuvoittaa verkkolaitteiden käyttöönottoa ja dokumentointia. Erityisesti, kun toimitukseen tulee useampi kuin yksi verkkolaite tai lähiverkko. Jatkokehitystä kaivataan, mutta sitä on helpompi tehdä, kun kokemusta järjestelmän käytöstä on enemmän. Tietoturvan puolesta selkein asia saada poistettua käyttöönottoskriptin suorittavalta käyttäjältä järjestelmänvalvojan oikeudet. Toiminnallisuuteen liittyen haluaisin lisätä skriptissä syötettävien muuttujien nimiin/sanakirja-avaimiin luokitukset, joiden perusteella käyttäjän antamat muuttujat tarkistettaisiin ja hänelle ilmoitettaisiin, jos syötetyssä muuttujassa olisi vikaa ja vian syyn. Iso askel olisi päästä eroon CSV-tiedostoista ja siirtyä käyttämään liitännäisten järjestelmien REST-rajapintoja.

Henkilökohtaisesti tärkeimpänä asian koin Python-ohjemoinnista ja Ansiblen käytöstä saadun kokemuksen. Lisäksi automaatioon perehtyminen aiheena on innostanut minua miettimään päivittäisiä työtapojan yleisesti tuottavuuden ja viihtyvyyden parantamiseksi. Näiden eväiden avulla on hyvä lähteä suunnittelemaan automaatiotason lisäämistä työpaikallani. Palveluntarjoajana on usein hankala toteuttaa automaatiota johtuen asiakkaiden pirstaloituneista IT-ympäristöistä, mutta oman konesaliympäristömme verkkolaitteiden konfiguraationhallinta on otollinen kohde automatisoinnille. Kun kokemusta ja sitä myöten rohkeutta kertyy lisää, löytyy työpaikaltani varmasti isompiakin asioita, mitä on mahdollista automatisoida. Seuraava askel kuitenkin on ottaa järjestelmä käyttöön työpaikallani.

Lähteet

1 O Duibhir, Dona. 2020. The State of Network Automation Report. Juniper Networks Inc. Verkkodokumentti. <https://www.juniper.net/us/en/forms/sonar/> Luettu 15.12.2020.

2 McGillicuddy, Shamus. 2019. Research Summary: Enterprise Network Automation for 2020 and Beyond. Red Hat Inc. Verkkodokumentti. <https://www.redhat.com/en/resources/enterprise-network-automation-ema-analyst-paper> Luettu 15.12.2020.

3 Mortensen, Mark. 2020. Economic Benefits of Network Automation. Cisco Inc. Verkkodokumentti. <https://www.cisco.com/c/dam/en/us/products/collateral/cloud-systems-management/network-services-orchestrator/acg-economic-benefits-of-network-automation.pdf> Luettu 15.12.2020.

4 Zero Touch Provisioning. 2020. Verkkodokumentti. Juniper Networks Inc. <https://www.juniper.net/documentation/en_US/junos/topics/topic-map/zero-touch-provision.html> Luettu 17.12.2020.

5 Understanding Autoinstallation of Configuration Files. 2020. Verkkodokumentti. Juniper Networks Inc. <https://www.juniper.net/documentation/en_US/junos/topics/topic-map/understanding-autoinstallation-config-files.html> Luettu 17.12.2020.

6 Managed Branch Office VPN Tunnels (WSM). 2020. Verkkodokumentti. WatchGuard. <https://www.watchguard.com/help/docs/help-center/en-US/Content/en-US/Fireware/bovpn/managed/bovpn_managed_tunnels_about_wsm.html> Luettu 17.12.2020.

7 CLI Configlets Overview. Verkkodokumentti. 2017. Juniper Networks Inc. <https://www.juniper.net/documentation/en_US/junos-space18.1/platform/topics/concept/cli-configlets-overview.html> Luettu 17.12.2020.

8 Kelsey. RapidAPI Developer Survey Insights. 2020. Verkkodokumentti. RapidAPI. <https://rapidapi.com/blog/rapidapi-developer-survey-insights/> Luettu 18.12.2020.

9 Avraham, Shif Ben. What is REST — A Simple Explanation for Beginners, Part 2: REST Constraints. Verkkodokumentti. 2017. Medium. <https://medium.com/extend/what-is-rest-a-simple-explanation-for-beginners-part-2-rest-constraints-129a4b69a582> Luettu 18.12.2020.

10 Junos Space Platform API Reference. Verkkodokumentti. Juniper Networks Inc. <https://www.juniper.net/documentation/en_US/junos-space-sdk/16.1/apiref/index.html> Luettu 18.12.2020.

11 Understanding the REST API. 2003. Verkkodokumentti. 2020. Juniper Networks Inc. <https://www.juniper.net/documentation/en_US/junos/topics/concept/rest-api-overview.html> Luettu 18.12.2020.

12 Schoenwaelder, J. Overview of the 2002 IAB Network Management Workshop. 2003. Verkkodokumentti. Internet Research Task Force. <https://tools.ietf.org/html/rfc3535> Luettu 18.12.2020.

13 Enns, Rob. NETCONF Configuration Protocol. 2006. Verkkodokumentti. Internet Research Task Force (IRTF). https://tools.ietf.org/html/rfc4741 Luettu 18.12.2020.

14 Lhotka, L. JSON Encoding of Data Modeled with YANG. 2016. Verkkodokumentti. Internet Research Task Force (IRTF). <https://tools.ietf.org/html/rfc7951> Luettu 19.12.2020.

15 Bjorklund, M. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). 2010. Verkkodokumentti. Internet Research Task Force (IRTF). <https://tools.ietf.org/html/rfc6020> Luettu 19.12.2020.

16 YANG. Verkkodokumentti. Wikimedia Foundation. <https://en.wikipedia.org/wiki/YANG> Luettu 19.12.2020.

17 Hannadige, Sahan. NETCONF/YANG for network automation over SNMP. 2019. Verkkodokumentti. Linkedin. <https://www.linkedin.com/pulse/netconfyang-network-automation-over-snmp-sahan-hannadige> Luettu 19.12.2020.

18 Froehlich, Andrew. The 8 leading options in network automation tools. 2020. Verkkodokumentti. Techtarget. <https://searchnetworking.techtarget.com/feature/The-8-leading-options-in-network-automation-tools> Luettu 19.12.2020.

19 Perkin, Roger. Network Automation Tools. 2020. Verkkodokumentti. <https://www.rogerperkin.co.uk/network-automation/tools/> Luettu 19.12.2020.

20 DeNisco Rayome, Alison. Ansible overtakes Chef and Puppet as the top cloud configuration management tool. 2019. Verkkodokumentti. TechRepublic. <https://www.techrepublic.com/article/ansible-overtakes-chef-and-puppet-as-the-top-cloud-configuration-management-tool/> Luettu 19.12.2020.

21 Basic Concepts. 2020. Verkkodokumentti. Red Hat Inc. <https://docs.ansible.com/ansible/latest/network/getting_started/basic_concepts.html> Luettu 19.12.2020.

22 Wang, Derek. aruba-ansible-modules. 2020. Verkkodokumentti. GitHub Inc. <https://github.com/aruba/aruba-ansible-modules> Luettu 19.12.2020.

23 Understanding the Ansible for Junos OS Modules. 2019. Juniper Networks Inc. Verkkodokumentti. <https://www.juniper.net/documentation/en_US/junos-ansible/topics/reference/general/junos-ansible-modules-overview.html> Luettu 19.12.2020.

24 DeNisco Rayome, Alison. Ansible overtakes Chef and Puppet as the top cloud configuration management tool. 2019. Verkkodokumentti. TechRepublic. <https://www.techrepublic.com/article/ansible-overtakes-chef-and-puppet-as-the-top-cloud-configuration-management-tool/> Luettu 19.12.2020.

25 Shein, Esther. The First 10 Years of Software Defined Networking. 2018. Verkkodokumentti. The Linux Foundation. <https://www.linuxfoundation.org/blog/2018/05/the-first-10-years-of-software-defined-networking/> Luettu 19.12.2020.

26 CentOS Linux. Verkkodokumentti. The CentOS Project. <https://www.centos.org/about/> Luettu 20.12.2020.

27 The CentOS Project. Download. Verkkodokumentti. The CentOS Project. <https://www.centos.org/download/>. Luettu 30.12.2020.

28 Zero Touch Provisioning. 2020. Verkkodokumentti. Juniper Networks Inc. <https://www.juniper.net/documentation/en_US/junos/topics/topic-map/zero-touch-provision.html> Luettu 02.12.2021.

29 Understanding Autoinstallation of Configuration Files. 2020. Verkkodokumentti Juniper Networks Inc. <https://www.juniper.net/documentation/en_US/junos/topics/topic-map/understanding-autoinstallation-config-files.html> Luettu 02.12.2021.

30 What is SELinux. Verkkodokumentti. Red Hat Inc. <https://www.redhat.com/en/topics/linux/what-is-selinux> Luettu 02.12.2021.

31 dhcp-options(5) - Linux man page. Verkkodokumentti. die.net. <https://linux.die.net/man/5/dhcp-options> Luettu 02.12.2021.

32 dhcpd-eval(5) - Linux man page. Verkkodokumentti. die.net. <https://linux.die.net/man/5/dhcp-options> Luettu 02.12.2021.

33 Mutal, Josphat. Install Python 3.8 on CentOS 7 / CentOS 8. 2020. Verkkodokumentti. Computingforgeeks. <https://computingforgeeks.com/how-to-install-python-on-3-on-centos/> Luettu 02.12.2021.

Juniper EX2300 kytkimen tehdasasetusten “System” -haara

```
system {
auto-snapshot;
phone-home {
server https://redirect.juniper.net;
rfc-compliant;
}
services {
ssh;
netconf {
ssh;
rfc-compliant;
yang-compliant;
}
}
}
```

dhcpd.conf

```
#according to man this writes vendor-class-indentifier to under address lease in lease
file
set vendor-string = option vendor-class-identifier;
#create ZTP option space, all codes are set to type text
option space ZTP;
#option ZTP.image-file-name code 0 = text;
option ZTP.config-file-name code 1 = text;
#option ZTP.image-file-type code 2 = text;
option ZTP.transfer-mode code 3 = text;
#package the ZTP option space to option 43
option ZTP-encapsulation code 43 = encapsulate ZTP;
#sets the option-150, juniper uses this option for configuration server, to be the ip-
address of the server
option option-150 code 150 = ip-address;

subnet 192.168.110.0 netmask 255.255.255.0 {
	interface ens224;
	option ZTP.transfer-mode "http";
	option broadcast-address 192.168.110.255;
#note the short lease times so clients are refreshed quickly
	max-lease-time 60;
	default-lease-time 60;
	pool {
		range dynamic-bootp 192.168.110.50 192.168.110.100;
		option option-150 192.168.110.10;
#the vendor strings had some extra characters after the device so here we match the
first 14 letters against
#Juniper-ex2200
		if (substring(option vendor-class-identifier, 0,14)="Juniper-ex2200") {
			option ZTP.config-file-name "ex2200-skeleton.config";
#option ZTP.image-file-name "jinstall-ex-2200-12.3R12.4-domestic-signed.tgz";
		}
#Juniper-ex2300
		elsif (substring(option vendor-class-identifier, 0,14)="Juniper-ex2300") {
			option ZTP.config-file-name "ex-skeleton.config";
#option ZTP.image-file-name "junos-arm-32-18.1R3.3.tgz";
		}
	}
}
```

Python kirjastot ja paketit

pip list

Package	Version
ansible	2.9.10
ansible-runner	1.4.6
bcrypt	3.1.7
cffi	1.14.0
cryptography	2.9.2
docutils	0.16
future	0.18.2
importlib-resources	3.0.0
Jinja2	2.11.2
junos-eznc	2.5.4+8.gfc5debc
jxmlease	1.0.3
lockfile	0.12.2
lxml	4.5.1
MarkupSafe	1.1.1
ncclient	0.6.7
netaddr	0.8.0
npyscreen	4.10.5
ntc-templates	1.5.0
paramiko	2.7.1
pathspec	0.8.0
pexpect	4.8.0
pip	19.2.3
psutil	5.7.0
ptyprocess	0.6.0
pycparser	2.20
pyez	0.1.1
PyNaCl	1.4.0
pyparsing	2.4.7
pyserial	3.4
python-daemon	2.2.4
PyYAML	5.3.1
scp	0.13.2
setuptools	41.2.0
six	1.15.0
textfsm	1.1.0
transitions	0.8.2
yamllint	1.23.0
yamlordereddictloader	0.4.0
zipp	3.1.

ansible.cfg

```
[defaults]
inventory      = ../app/inventory/
interpreter_python = /venv/deployment/bin/python
roles_path = ./roles:~/.ansible/roles:/usr/share/ansible/roles
log_path = ../app/log/ansible.log
local_tmp = ../app/tmp
# Host key checking is enabled by default
host_key_checking = False
# Default timeout for connection plugins
timeout = 30
[inventory]
# List of enabled inventory plugins and the order in which they are used.
enable_plugins = host_list, script, yaml
[privilege_escalation]
[paramiko_connection]
[ssh_connection]
[persistent_connection]
# Configures the persistent connection timeout value in seconds. This value is
# how long the persistent connection will remain idle before it is destroyed.
# If the connection doesn't receive a request before the timeout value
# expires, the connection is shutdown. The default value is 30 seconds.
connect_timeout = 45
# The command timeout value defines the amount of time to wait for a command
# or RPC call before timing out. The value for the command timeout must
# be less than the value of the persistent connection idle timeout (connect_timeout)
# The default value is 30 second.
command_timeout = 40
[sudo_become_plugin]
[selinux]
[colors]
[diff]
[galaxy]
```

hosts.yaml

```
---
all:
  children:
    deployment:
      children:
        rdyforsoft:
        rdyforconf:
```

deployment.yaml

```
---
ansible_ssh_common_args: "-o StrictHostKeyChecking=no -o CheckHostIp=False -o
UserKnownHostsFile=/dev/null"
ansible_user: "root"
ansible_password: "Qwerty123"
```

hosts-from-leases.py

```
#!/venv/deployment/bin/python
import subprocess
import argparse
import json

parser = argparse.ArgumentParser()
parser.add_argument('-c', '--count', action='store_true', required=False, dest='count')
parser.add_argument('--list', action='store_true', required=False, dest='list')
args = parser.parse_args()

leases_rdy_for_soft = subprocess.Popen(
    "grep -B 9 'ready-for-soft' /var/lib/dhcpd/dhcpd.leases | grep -B 5 'binding state
active' | grep 'lease' | sort -u | awk '{print $2}'",
    shell=True, stdout=subprocess.PIPE).stdout.read()

leases_rdy_for_conf = subprocess.Popen(
    "grep -B 9 'ready-for-conf' /var/lib/dhcpd/dhcpd.leases | grep -B 5 'binding state
active' | grep 'lease' | sort -u | awk '{print $2}'",
    shell=True, stdout=subprocess.PIPE).stdout.read()

leases_rdy_for_soft = leases_rdy_for_soft.decode()

leases_rdy_for_soft = leases_rdy_for_soft.split('\n')

leases_rdy_for_conf = leases_rdy_for_conf.decode()

leases_rdy_for_conf = leases_rdy_for_conf.split('\n')

try:
    leases_rdy_for_soft.remove('')
    leases_rdy_for_conf.remove('')
except ValueError:
    pass

targets = {
    "_meta": {
        "hostvars": {}
    },
    "rdyforsoft": {
        "hosts": []
    },
    "rdyforconf": {
        "hosts": []
    },
    "local": {
        "hosts": []
    }
}
for rdyforsoft in leases_rdy_for_soft:
    targets["rdyforsoft"]["hosts"].append(rdyforsoft)
```

```
for rdyforconf in leases_rdy_for_conf:
    targets["rdyforconf"]["hosts"].append(rdyforconf)

if args.count:
    print(len(targets["rdyforsoft"]["hosts"]) + len(targets["rdyforconf"]["hosts"]))
else:
    print(json.dumps(targets))
```

ex-skeleton.config

```
system {
root-authentication {
    encrypted-password
"$6$BXRlcQl$S.zO3ybugy.u5APUefSW4oat/WI/IR9JDB3Ryi1sZaQqycowYU.QN2jRS
YCh94ipP.klHoka/m4mWxTdM2SpS/"; ## SECRET-DATA
}
login {
    password {
        format sha1;
    }
}
    services {
        ssh {
            root-login allow;
        }
        netconf {

            ssh;
        }
    }
}
interfaces {

    ge-0/0/0 {

        unit 0 {

            family ethernet-switching;
        }
    }
    irb {

        unit 0 {

            family inet {

                dhcp {

                    vendor-id ready-for-software;
                }
            }
        }
    }
}
vlans {
    default {
        vlan-id 1;
        l3-interface irb.0;
    }
}
```

ex2200-skeleton.config

```
system {
    root-authentication {
        encrypted-password "$sha1$19066$.U1p0PPf$TkIT/7JrDdxKOr.XsAjuK/Xuxhxw";
## SECRET-DATA
    }
    login {
                password {
            format sha1;
        }
    }
    services {
                ssh {
            root-login allow;
        }
                netconf {
            ssh;
        }
    }
}
interfaces {
   me0 {
       unit 0 {
            family inet {
                dhcp {
                    vendor-id ready-for-software;
                    }
                }
            }
}
}
```

network.conf

```
delete: system;
delete: security;
delete: protocols;
delete: access;
delete: vlans;
delete: interfaces;
system {
    root-authentication {
        encrypted-password
"$5$ng781ovT$1kNTWjWH3USNjNxm/tB4I.anghRWmGmNK5/nmxtCBd1"; ## SECRET-
DATA
    }
    login {
        password {
            format sha1;
        }
    }
    services {
        ssh {
            root-login allow;
        }
        netconf {
            ssh;
        }
    }
}
security {
    zones {
        security-zone trust {
            interfaces {
                ge-0/0/0.0 {
                    host-inbound-traffic {
                        system-services {
                            dhcp;
                            bootp;
                            ssh;
                            netconf;
                        }
                    }
                }
            }
        }
    }
}
interfaces {
    ge-0/0/0 {
        unit 0 {
            family inet {
                dhcp-client {
                    vendor-id ready-for-software;
                }
            }
        }
    }
}
```

ex2200rdyforconfig.conf

```
interfaces {
    me0 {
        unit 0 {
            family inet {
                dhcp {
                    vendor-id ready-for-config;
                }
            }
        }
    }

}
```

ex2300rdyforconfig.conf

```
interfaces {
    irb {
        unit 0 {
            family inet {
                dhcp {
                    vendor-id ready-for-config;
                }
            }
        }
    }
}
```

srxrdyforconfig.conf

```
interfaces {
    ge-0/0/0 {
        unit 0 {
            family inet {
                dhcp {
                    vendor-id ready-for-config;
                }
            }
        }
    }
}
```

updatevars.yml

```
---
ex2200soft: "12.3R12-S17"
ex2300soft: "18.2R3-S6.5"
srx300soft: "18.4R3-S4.2"
ex2200pkg: "jinstall-ex-2200-12.3R12-S17-domestic-signed.tgz"
ex2300pkg: "junos-arm-32-18.2R3-S6.5.tgz"
srx300pkg: "junos-srxsme-18.4R3-S4.2.tgz"
ex2200conf: “ex2200rdyforconfig.conf”
ex2300conf: “ex2300rdyforconfig.conf”
srx300conf: “srxrdyforconfig.conf”
path: "/venv/deployment/app/update"
```

update-devices.yml

```
---
- name: update-devices-to-desired-version
  hosts: rdyforsoft
  vars_files:
    - /venv/deployment/app/update/updatevars.yml
  roles:
    - Juniper.junos
  connection: local
  gather_facts: no
  strategy: host_pinned

  tasks:
    - name: Get Junos device information
      juniper_junos_facts:
      register: junos

    - name: Upgrade Junos OS
      juniper_junos_software:
        version:"{{ srx300soft }}"
        local_package: "{{ path }}/{{ srx300pkg }}"
        no_copy: false
        validate: false
        logdir: "{{ path }}/"
        reboot: true
      when: junos.facts.model is match("SRX300*")

    - name: update EX2200
      juniper_junos_software:
        version: "{{ ex2200soft }}"
        local_package: ""{{ path }}/{{ ex2200pkg }}"
        reboot: true
        validate: false
        logdir: "{{ path }}/"
      when: junos.facts.model is match("EX2200*")

    - name: update EX2300
      juniper_junos_software:
        version: "{{ ex2300soft }}"
        local_package: ""{{ path }}/{{ ex2300pkg }}"
        reboot: true
        validate: false
        logdir: "{{ path }}/"
      when: junos.facts.model is match("EX2300*")

    - name: wait_reboot
      wait_for:
        host: "{{ inventory_hostname }}"
        port: 22
        timeout: 600
```

```
- name: Change id to ready-for-config SRX300
  juniper_junos_config:
    src: "{{ path }}/{{ srx300conf }}"
    timeout: 300
    load: "replace"
  when: junos.facts.model is match("SRX300*")

- name: Change id to ready-for-config EX2300
  juniper_junos_config:
    src: ”{{ path }}/{{ ex2300conf }} "
    timeout: 120
    load: "replace"
  when: junos.facts.model is match("EX2300*")

- name: Change id to ready-for-config EX2200
  juniper_junos_config:
    src: ”{{ path }}/{{ ex2200conf }}"
    timeout: 120
    load: "replace"
```

update-devices

```
#!/bin/bash
if ps -ef | grep -v grep | grep update-devices.yml ; then

        exit 0

else
        source "/venv/deployment/deployment/bin/activate"
        "ansible-playbook" “/venv/deployment/playbooks/update-devices.yml"
        deactivate
        sudo /bin/systemctl restart dhcpd.service
        exit 0
fi
```

print-dev-inf.yml

```
---
- name: get-model-version-sn-print
  hosts: rdyforconf
  roles:
    - Juniper.junos
  connection: local
  gather_facts: no

  tasks:
    - name: Get Junos device information
      juniper_junos_facts:
      register: junos_facts

    - name: write info to file
      vars:
        model: "{{junos_facts.facts.model}}"
        serial: "{{junos_facts.facts.serialnumber}}"
        version: "{{junos_facts.facts.version}}"
      file:
        state: touch
        path: "/venv/deployment/app/playbooks/tmp/{{ inventory_hostname }},{{ model }},{{
serial }}"
```

commit_check.yml

```
---
- name: commit_check
  hosts: "{{ targets }}"
  roles:
    - Juniper.junos
  connection: local
  gather_facts: no

  tasks:
    - name: Load and Commit
      juniper_junos_config:
        load: override
        src: "{{ configdir }}/{{ inventory_hostname }}.conf"
        check: true
        diff: false
        commit: false
        timeout: 60
```

commit.yml

```
---
- name: commit_config
  hosts: "{{ targets }}"
  roles:
    - Juniper.junos
  connection: local
  gather_facts: no

  tasks:
    - name: Load and Commit
      juniper_junos_config:
        load: override
        src: "{{ configdir }}/{{ inventory_hostname }}.conf"
        check: false
        diff: false
        commit: true
```

cluster.yml

```
---
- name: Enable chassis cluster
  hosts: "{{ target }}"
  connection: local
  gather_facts: no
  roles:
    - Juniper.junos
  tasks:
    - name: Enable an SRX cluster
      juniper_junos_srx_cluster:
        enable: true
        cluster_id: 1
        node_id: "{{ id }}"
```

default.SRX300.j2

```
{% if iscluster == 'False' %}{% set laninterface = 'irb' %}{% set wan1interface = 'ge-
0/0/0' %}{% set wan2interface = 'ge-0/0/1' %}{% endif %}
{% if iscluster == 'True' %}{% set laninterface = 'reth5' %}{% set wan1interface = 'reth3'
%}{% set wan2interface = 'reth4' %}{% endif %}
system {
    login {
        user mgmtuser {
            uid 2000;
            class super-user;
            authentication {
                plain-text-password-value "{{ mgmtuserfwpwd }}";
            }
        }
    }
    ntp {
    server 194.100.49.139;
    server 194.100.49.151;
}
    root-authentication {
        plain-text-password-value "{{ rootfwpwd }}";
    }
    host-name {{ hostname }}-fw;
    time-zone Europe/Helsinki;
    name-server {
        {{ wandns1 }};
        {{ wandns2 }};
    }
    syslog {
                host {{ hostname }}-fw {
          any warning;
       }
        user * {
        any emergency;
    }
        file messages {
            any notice;
            authorization info;
        }
        file kmd-logs {
            daemon info;
            match KMD;
        }
        file policy_session {
            any any;
            user info;
            match RT_FLOW;
            archive size 1000k world-readable;
            structured-data;
        }
        file idp_log {
```

```
        any any;
        match RT_IDP;
    }
    file interactive-commands {
      interactive-commands info;
      match UI_CMDLINE_READ_LINE;
}
}
services {
outbound-ssh {
                        client cluster_123.123.123.123 {
                                device-id DEVIDA;
                                secret $9$EkjcSlLxNds2W8XNdb2g36/CO1;
                                services netconf;
                                123.123.123.123 {
                                        port 7804;
                                }
                        }
          }
    ssh {
        root-login deny;
        protocol-version v2;
        max-sessions-per-connection 32;
        connection-limit 3;
        rate-limit 3;
    }
    netconf {
        ssh;
    }
    dns {
        max-ncache-ttl 10;
        dns-proxy {
            interface {
            {%- for vlan, value in VLANs.items() %}
            {%- if value.zone == 'trust' %}
            {{ laninterface }}.{{ value.id }};
            {%- endif %}
            {%- endfor %}
            }
            default-domain {{ dnssuffix }} {
                forwarders {
                    {{ wandns1 }};
                    {{ wandns2 }};
                }
            }
            default-domain * {
                forwarders {
                    1.1.1.1;
                    8.8.8.8;
                }
            }
        }
    }
```

```
        dhcp-local-server {
            group local {
            {%- for vlan, value in VLANs.items() %}
            {%- if value.dhcp == 'True' %}
                interface {{ laninterface }}.{{ value.id }};
            {%- endif %}
            {%- endfor %}
            }
        }
    }

}
{%- if iscluster == 'True' %}
groups {
    node0 {
        system {
            host-name {{ hostname }}-fw-node0;
            backup-router {{ fxpgw }} destination 123.123.123.123;
        }
        interfaces {
            fxp0 {
                unit 0 {
                    family inet {
                        address {{ node0fxp }};
                    }
                }
            }
        }
    }
    node1 {
        system {
            host-name {{ hostname }}-fw-node0;
            backup-router {{ fxpgw }} destination  123.123.123.123;
        interfaces {
            fxp0 {
                unit 0 {
                    family inet {
                        address {{ node1fxp }};
                    }
                }
            }
        }
    }
}
apply-groups "${node}";
chassis {
    cluster {
        reth-count 3;
        redundancy-group 0 {
            node 0 priority 100;
            node 1 priority 1;
        }
        redundancy-group 1 {
```

```
            node 0 priority 100;
            node 1 priority 1;
            interface-monitor {
                ge–0/0/5 weight 255;
                ge–0/0/4 weight 255;
                ge–7/0/5 weight 255;
                ge–7/0/4 weight 255;
            }
        }
    }
}
{%- endif %}



security {
  log {
    mode stream;
    format sd-syslog;
    source-address {{ wan1ip | ipaddr('address') }};
    stream space-log-stream {
      severity debug;
      format sd-syslog;
      category all;
      host {
        123.123.123.123;
        port 514;
      }
    }
  }
 screen {
trap interval 300;
ids-option internal-screen {
    icmp {
        ip-sweep threshold 10000;
        flood threshold 1000;
        ping-death;
    }
    ip {
        bad-option;
        record-route-option;
        timestamp-option;
        stream-option;
        source-route-option;
        loose-source-route-option;
        strict-source-route-option;
        unknown-protocol;
        tear-drop;
    }
    tcp {
        syn-fin;
        fin-no-ack;
        tcp-no-flag;
```

```
            syn-frag;
            port-scan threshold 10000;
            syn-flood {
                alarm-threshold 1024;
                attack-threshold 200;
                source-threshold 4000;
                destination-threshold 4000;
                timeout 20;
            }
            land;
            winnuke;
            tcp-sweep threshold 10000;
        }
        udp {
            flood threshold 100000;
            udp-sweep threshold 10000;
            port-scan threshold 10000;
        }
        limit-session {
            source-ip-based 10000;
            destination-ip-based 10000;
        }
    }
    ids-option external-screen {
        icmp {
            ping-death;
        }
        ip {
            source-route-option;
            tear-drop;
        }
        tcp {
            port-scan threshold 10000;
            syn-flood {
                alarm-threshold 1024;
                attack-threshold 200;
                source-threshold 4000;
                destination-threshold 4000;
                timeout 20;
            }
            land;
        }
        udp {
            udp-sweep threshold 15000;
            port-scan threshold 10000;
        }
    }
    }
    alarms {
        potential-violation {
            authentication 2;
            policy {
                source-ip {
```

```
                threshold 500;
                duration 20;
            }
            destination-ip {
                threshold 500;
                duration 20;
            }
        }
        replay-attacks {
            threshold 1500;
        }
        idp;
    }
}

    address-book {
        global {
        {%- for vlan, value in VLANs.items() %}
            address {{ value.name }}-{{ value.network }} {
                description "{{ value.name }}";
                {{ value.network }};
             }
        {%- endfor %}
        }
    }
    nat {
        source {
          {%- for vlan, value in VLANs.items() %}
            rule-set {{ value.zone }}-untrust {
              from zone {{ value.zone }};
              to zone [ untrust untrust2 ];

              rule {{ value.zone }}-default-untrust-snat {
                match {
                  source-address-name {{ value.name }}-{{ value.network }};
                  destination-address 0.0.0.0/0;
                }
                then {
                  source-nat {
                    interface;
                  }
                }
              }
            }
        {%- endfor %}
            rule-set junoshost {
                from zone junos-host;
                to zone [ untrust untrust2 {%- for vlan, value in VLANs.items() %} {{
value.zone }} {%- endfor %} ];

                rule junoshost-source-nat-def {
                    match {
                        source-address 0.0.0.0/0;
```

```
                destination-address 0.0.0.0/0;
            }
            then {
                source-nat {
                    interface;
                }
            }
        }
    }
}
policies {
    {%- for vlan, value in VLANs.items() %}
    from-zone {{ value.zone }} to-zone untrust {
        policy {{ value.zone }}-default-untrust-allow {
            match {
                source-address {{ value.name }}-{{ value.network }};
                destination-address any;
                application any;
                }
            then {
                permit;
            }
        }
    }
    {%- endfor %}
    {%- for vlan, value in VLANs.items() %}
    from-zone {{ value.zone }} to-zone untrust2 {
        policy {{ value.zone }}-default-untrust2-allow {
            match {
                source-address {{ value.name }}-{{ value.network }};
                destination-address any;
                application any;
                }
            then {
                permit;
            }
        }
    }
    {%- endfor %}
    {%- for vlan, value in VLANs.items() %}
    from-zone {{ value.zone }} to-zone {{ value.zone }} {
        policy {{ value.zone }}-default-internal-allow {
            match {
                source-address {{ value.name }}-{{ value.network }};
                destination-address {{ value.name }}-{{ value.network }};
                application any;
                }
            then {
                permit;
            }
        }
    }
```

```
    {%- endfor %}
    global {
        policy Global-implicit-deny {
            match {
                source-address any;
                destination-address any;
                application any;
            }
            then {
                deny;
            }
        }
    }
}
zones {
    {%- for vlan, value in VLANs.items() %}
    security-zone {{ value.zone }} {
        screen internal-screen;
        interfaces {
          {{ laninterface }}.{{ value.id }} {
            host-inbound-traffic {
              system-services {
                ping;
                traceroute;
                dhcp;
                dns;
              }
            }
          }
        }
    }
    {%- endfor %}
    security-zone untrust {
        screen external-screen;
        interfaces {
            {{ wan1interface }}.0 {
                host-inbound-traffic {
                    system-services {
                        ping;
                        traceroute;
                        ssh;
                        snmp;
                        ike;
                    }
                }
            }
        }
    }
    security-zone untrust2 {
        screen external-screen;
        interfaces {
            {{ wan2interface }}.0 {
                host-inbound-traffic {
```

```
                    system-services {
                        ike;
                        ping;
                        traceroute;
                        ssh;
                        dhcp;
                    }
                }
            }
        }
    }
}
interfaces {
    {%- if iscluster == 'True' %}
    ge–0/0/5 {
        gigether–options {
            redundant–parent reth5;
        }
    }
    ge–0/0/4 {
        gigether–options {
            redundant–parent reth4;
        }
    }
    ge–0/0/3 {
        gigether–options {
            redundant–parent reth3;
        }
    }
    ge–7/0/5 {
        gigether–options {
            redundant–parent reth5;
        }
    }
    ge–7/0/4 {
        gigether–options {
            redundant–parent reth4;
        }
    }
    ge–7/0/3 {
        gigether–options {
            redundant–parent reth3;
        }
    }
    fab0 {
        fabric–options {
            member–interfaces {
                ge–0/0/2;
            }
        }
    }
    fab1 {
```

```
        fabric–options {
            member–interfaces {
                ge–1/0/2;
            }
        }
    }
    {%- endif %}
    {%- if iscluster == 'False' %}
    interface-range trunk-wlan {
        member "ge-0/0/[2-3]";
        {%- for vlan, value in VLANs.items() %}
        {%- if value.nwmgmt == 'True' %}
        native-vlan-id {{ value.id }};
        {%- endif %}
        {%- endfor %}
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan {
                    members [ {% for vlan, value in VLANs.items() %}{{ value.name }} {%
endfor %}];
                }
            }
        }
    }
    interface-range trunk-uplink {
        member "ge-0/0/[4-7]";
        {%- for vlan, value in VLANs.items() %}
        {%- if value.wks == 'True' %}
        native-vlan-id {{ value.id }};
        {%- endif %}
        {%- endfor %}
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan {
                    members [ {% for vlan, value in VLANs.items() %}{{ value.name }} {%
endfor %}];
                }
            }
        }
    }
    {%- endif %}
    {{ wan1interface }} {
        description wan1;
        {%- if iscluster == 'True' %}
        redundant–ether–options {
            redundancy–group 1;
        }
        {%- endif %}
        unit 0 {
            family inet {
                address {{ wan1ip }};
```

```
            }
        }
    }
    {{ wan2interface }} {
        description wan2;
        {%- if iscluster == 'True' %}
        redundant–ether–options {
            redundancy–group 1;
        }
        {%- endif %}
        unit 0 {
            family inet {
            {% if wan2dhcp == 'False' %}address {{ wan2ip }};{% endif %}
            {%- if wan2dhcp == 'True' %}
            dhcp {
                retransmission-attempt 50000;
                retransmission-interval 6;
                force-discover;
            }
            {%- endif %}
            }
        }
    }
    {{ laninterface }} {
        {%- for vlan, value in VLANs.items() %}
        unit {{ value.id }} {
            family inet {
                address {{ value.network | ipaddr('net') | ipaddr(value.gw) }};
            }
        }
        {%- endfor %}
    }
      lo0 {
    unit 0 {
        family inet {
            filter {
                input re-filter;
            }
        }
    }
  }
}
routing-options {
static {
    route 0.0.0.0/0 {
        next-hop {{ wan1gw }};
        qualified-next-hop {{ wan2gw }} {
            preference 10;
        }
    }
}
  instance-import import-wan;
}
```

```
routing-instances {
     wan1 {
        interface {{ wan1interface }}.0;
        instance-type virtual-router;
     }
     wan2 {
        interface {{ wan2interface }}.0;
        instance-type virtual-router;
     }
 }
policy-options {
   policy-statement import-wan {
      from {
         instance-list [ wan1 wan2 ];
         interface [ {{ wan1interface }}.0 {{ wan2interface }}.0 ];
      }
      then accept;
   }
}
snmp {
   location "{{ street }} {{ city }}  {{ country }}";
   community {{ snmpcommunity }};
}
{%- if iscluster == 'False' %}
vlans {
     {%- for vlan, value in VLANs.items() %}
    {{ value.name }} {
        vlan-id {{ value.id }};
        l3-interface irb.{{ value.id }};
   }
    {%- endfor %}
}
{%- endif %}
firewall {
   filter re-filter {
      term allow-ssh-management {
         from {
            source-prefix-list {
               mgmt-addresses;
            }
            protocol tcp;
            destination-port 22;
         }
         then accept;
      }
      term allow-snmp-monitoring {
         from {
            source-prefix-list {
               mgmt-addresses;
               monitoring-addresses;
            }
            protocol udp;
            destination-port 161;
```

```
    }
    then accept;
}
term allow-ping-monitoring {
    from {
        source-prefix-list {
            mgmt-addresses;
            monitoring-addresses;
            private-addresses;
        }
        protocol icmp;
        icmp-type [ echo-request unreachable echo-reply ];
    }
    then accept;
}
term allow-ike {
    from {
        protocol udp;
        port [ 500 4500 ];
    }
    then accept;
}
term allow-esp {
    from {
        protocol esp;
    }
    then accept;
}
term allow-established {
    from {
        tcp-established;
    }
    then accept;
}
term allow-ntp-replies {
    from {
        source-address {
            194.100.49.151/32;
            194.100.49.152/32;
        }
        protocol udp;
        source-port 123;
    }
    then accept;
}
term allow-dns-replies {
    from {
        source-address {
            {{ wandns1 }};
            {{ wandns2 }};
        }
        source-prefix-list {
            private-addresses;
```

```
            }
            protocol udp;
            source-port 53;
            destination-port 53;
        }
        then accept;
    }
    term allow-icmp-replies {
        from {
            protocol icmp;
            icmp-type echo-reply;
        }
        then accept;
    }
    term allow-dhcp-client-accept {
        from {
            source-address {
                0.0.0.0/32;
            }
            destination-address {
                255.255.255.255/32;
            }
            protocol udp;
            source-port 68;
        }
        then accept;
    }
    term allow-dhcp-server-accept {
        from {
            protocol udp;
            source-port [ 67 68 ];
            destination-port [ 67 68 ];
        }
        then accept;
    }
    term deny-rest {
        then {
            log;
            discard;
        }
    }
  }
}
policy-options {
  prefix-list mgmt-addresses {
  {%- for address in managementaddresses %}
    {{ address }};
  {%- endfor %}
  }
  prefix-list monitoring-addresses {
  {%- for address in monitoringaddresses %}
    {{ address }};
  {%- endfor %}
```

```
  }
  prefix-list private-addresses {
        192.168.0.0/16;
        172.16.0.0/12;
        10.0.0.0/8;
  }
}
```

default-vpn.SRX300.j2

```
{% if iscluster == 'False' %}{% set laninterface = 'irb' %}{% set wan1interface = 'ge-
0/0/0' %}{% set wan2interface = 'ge-0/0/1' %}{% endif %}
{% if iscluster == 'True' %}{% set laninterface = 'reth5' %}{% set wan1interface = 'reth3'
%}{% set wan2interface = 'reth4' %}{% endif %}
system {
    login {
        user mgmtuser {
            uid 2000;
            class super-user;
            authentication {
                plain-text-password-value "{{ mgmtfwuser }}";
            }
        }
    }
    ntp {
    server 194.100.49.139;
    server 194.100.49.151;
}
    root-authentication {
        plain-text-password-value "{{ rootfwpwd }}";
    }
    host-name {{ hostname }}-fw;
    time-zone Europe/Helsinki;
    name-server {
        {{ wandns1 }};
        {{ wandns2 }};
    }
    syslog {
                host {{ hostname }}-fw {
          any warning;
       }
        user * {
        any emergency;
    }
        file messages {
            any notice;
            authorization info;
        }
        file kmd-logs {
            daemon info;
            match KMD;
        }
        file policy_session {
            any any;
            user info;
            match RT_FLOW;
            archive size 1000k world-readable;
            structured-data;
        }
        file idp_log {
```

```
        any any;
        match RT_IDP;
    }
    file interactive-commands {
      interactive-commands info;
      match UI_CMDLINE_READ_LINE;
}
}
services {
outbound-ssh {
                        client cluster_123.123.123.123 {
                                device-id DEVIDA;
                                secret $9$EkjcSlLxNds2W8XNdb2g36/CO1;
                                services netconf;
                                123.123.123.123 {
                                        port 7804;
                                }
                        }
            }
    ssh {
        root-login deny;
        protocol-version v2;
        max-sessions-per-connection 32;
        connection-limit 3;
        rate-limit 3;
    }
    netconf {
        ssh;
    }
    dns {
        max-ncache-ttl 10;
        dns-proxy {
            interface {
            {%- for vlan, value in VLANs.items() %}
            {%- if value.zone == 'trust' %}
            {{ laninterface }}.{{ value.id }};
            {%- endif %}
            {%- endfor %}
            }
            default-domain {{ dnssuffix }} {
                forwarders {
                    {{ wandns1 }};
                    {{ wandns2 }};
                }
            }
            default-domain * {
                forwarders {
                    1.1.1.1;
                    8.8.8.8;
                }
            }
        }
    }
    dhcp-local-server {
```

```
            group local {
            {%- for vlan, value in VLANs.items() %}
            {%- if value.dhcp == 'True' %}
               interface {{ laninterface }}.{{ value.id }};
            {%- endif %}
            {%- endfor %}
            }
         }
      }

}
{%- if iscluster == 'True' %}
groups {
   node0 {
      system {
         host-name {{ hostname }}-fw-node0;
         backup-router {{ fxpgw }} destination 123.123.123.123;
      }
      interfaces {
         fxp0 {
            unit 0 {
               family inet {
                  address {{ node0fxp }};
               }
            }
         }
      }
   }
   node1 {
      system {
         host-name {{ hostname }}-fw-node0;
         backup-router {{ fxpgw }} destination  123.123.123.123;
      interfaces {
         fxp0 {
            unit 0 {
               family inet {
                  address {{ node1fxp }};
               }
            }
         }
      }
   }
}
apply-groups "${node}";
chassis {
   cluster {
      reth-count 3;
      redundancy-group 0 {
         node 0 priority 100;
         node 1 priority 1;
      }
      redundancy-group 1 {
         node 0 priority 100;
```

```
            node 1 priority 1;
            interface-monitor {
                ge–0/0/5 weight 255;
                ge–0/0/4 weight 255;
                ge–7/0/5 weight 255;
                ge–7/0/4 weight 255;
            }
        }
    }
}
{%- endif %}



security {
  log {
    mode stream;
    format sd-syslog;
    source-address {{ wan1ip | ipaddr('address') }};
    stream space-log-stream {
      severity debug;
      format sd-syslog;
      category all;
      host {
        109.234.244.57;
        port 514;
      }
    }
  }
 screen {
trap interval 300;
ids-option internal-screen {
    icmp {
        ip-sweep threshold 10000;
        flood threshold 1000;
        ping-death;
    }
    ip {
        bad-option;
        record-route-option;
        timestamp-option;
        stream-option;
        source-route-option;
        loose-source-route-option;
        strict-source-route-option;
        unknown-protocol;
        tear-drop;
    }
    tcp {
        syn-fin;
        fin-no-ack;
        tcp-no-flag;
        syn-frag;
        port-scan threshold 10000;
```

```
        syn-flood {
            alarm-threshold 1024;
            attack-threshold 200;
            source-threshold 4000;
            destination-threshold 4000;
            timeout 20;
        }
        land;
        winnuke;
        tcp-sweep threshold 10000;
    }
    udp {
        flood threshold 100000;
        udp-sweep threshold 10000;
        port-scan threshold 10000;
    }
    limit-session {
        source-ip-based 10000;
        destination-ip-based 10000;
    }
}
ids-option external-screen {
    icmp {
        ping-death;
    }
    ip {
        source-route-option;
        tear-drop;
    }
    tcp {
        port-scan threshold 10000;
        syn-flood {
            alarm-threshold 1024;
            attack-threshold 200;
            source-threshold 4000;
            destination-threshold 4000;
            timeout 20;
        }
        land;
    }
    udp {
        udp-sweep threshold 15000;
        port-scan threshold 10000;
    }
}
}
alarms {
    potential-violation {
        authentication 2;
        policy {
            source-ip {
                threshold 500;
                duration 20;
```

```
            }
            destination-ip {
                threshold 500;
                duration 20;
            }
        }
        replay-attacks {
            threshold 1500;
        }
        idp;
    }
}

    address-book {
        global {
        {%- for vlan, value in VLANs.items() %}
            address {{ value.name }}-{{ value.network }} {
                description "{{ value.name }}";
                {{ value.network }};
             }
        {%- endfor %}
        }
    }
    nat {
        source {
          {%- for vlan, value in VLANs.items() %}
            rule-set {{ value.zone }}-untrust {
              from zone {{ value.zone }};
              to zone [ untrust untrust2 ];

              rule {{ value.zone }}-default-untrust-snat {
                match {
                  source-address-name {{ value.name }}-{{ value.network }};
                  destination-address 0.0.0.0/0;
                }
                then {
                  source-nat {
                    interface;
                  }
                }
              }
            }
        {%- endfor %}
            rule-set junoshost {
                from zone junos-host;
                to zone [ untrust untrust2 {%- for vlan, value in VLANs.items() %} {{
value.zone }} {%- endfor %} ];

                rule junoshost-source-nat-def {
                    match {
                        source-address 0.0.0.0/0;
                        destination-address 0.0.0.0/0;
                    }
```

```
                then {
                    source-nat {
                        interface;
                    }
                }
            }
        }
    }
}
policies {
    {%- for vlan, value in VLANs.items() %}
    from-zone {{ value.zone }} to-zone untrust {
        policy {{ value.zone }}-default-untrust-allow {
            match {
                source-address {{ value.name }}-{{ value.network }};
                destination-address any;
                application any;
                }
            then {
                permit;
            }
        }
    }
    {%- endfor %}
    {%- for vlan, value in VLANs.items() %}
    from-zone {{ value.zone }} to-zone untrust2 {
        policy {{ value.zone }}-default-untrust2-allow {
            match {
                source-address {{ value.name }}-{{ value.network }};
                destination-address any;
                application any;
                }
            then {
                permit;
            }
        }
    }
    {%- endfor %}
    {%- for vlan, value in VLANs.items() %}
    from-zone {{ value.zone }} to-zone {{ value.zone }} {
        policy {{ value.zone }}-default-internal-allow {
            match {
                source-address {{ value.name }}-{{ value.network }};
                destination-address {{ value.name }}-{{ value.network }};
                application any;
                }
            then {
                permit;
            }
        }
    }
    {%- endfor %}
    global {
        policy Global-implicit-deny {
```

```
            match {
                source-address any;
                destination-address any;
                application any;
            }
            then {
                deny;
            }
        }
    }
}
zones {
    {%- for vlan, value in VLANs.items() %}
    security-zone {{ value.zone }} {
        screen internal-screen;
        interfaces {
          {{ laninterface }}.{{ value.id }} {
            host-inbound-traffic {
              system-services {
                ping;
                traceroute;
                dhcp;
                dns;
              }
            }
          }
        }
    }
    {%- endfor %}
    security-zone untrust {
        screen external-screen;
        interfaces {
            {{ wan1interface }}.0 {
                host-inbound-traffic {
                    system-services {
                        ping;
                        traceroute;
                        ssh;
                        snmp;
                        ike;
                    }
                }
            }
        }
    }
    security-zone untrust2 {
        screen external-screen;
        interfaces {
            {{ wan2interface }}.0 {
                host-inbound-traffic {
                    system-services {
                        ike;
                        ping;
```

```
                    traceroute;
                    ssh;
                    dhcp;
                }
            }
        }
    }
  }
}
interfaces {
  {%- if iscluster == 'True' %}
  ge–0/0/5 {
    gigether–options {
      redundant–parent reth5;
    }
  }
  ge–0/0/4 {
    gigether–options {
      redundant–parent reth4;
    }
  }
  ge–0/0/3 {
    gigether–options {
      redundant–parent reth3;
    }
  }
  ge–7/0/5 {
    gigether–options {
      redundant–parent reth5;
    }
  }
  ge–7/0/4 {
    gigether–options {
      redundant–parent reth4;
    }
  }
  ge–7/0/3 {
    gigether–options {
      redundant–parent reth3;
    }
  }
  fab0 {
    fabric–options {
      member–interfaces {
        ge–0/0/2;
      }
    }
  }
  fab1 {
    fabric–options {
      member–interfaces {
        ge–1/0/2;
      }
```

```
        }
    }
    {%- endif %}
    {%- if iscluster == 'False' %}
    interface-range trunk-wlan {
        member "ge-0/0/[2-3]";
        {%- for vlan, value in VLANs.items() %}
        {%- if value.nwmgmt == 'True' %}
        native-vlan-id {{ value.id }};
        {%- endif %}
        {%- endfor %}
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan {
                    members [ {% for vlan, value in VLANs.items() %}{{ value.name }} {%
endfor %}];
                }
            }
        }
    }
    interface-range trunk-uplink {
        member "ge-0/0/[4-7]";
        {%- for vlan, value in VLANs.items() %}
        {%- if value.wks == 'True' %}
        native-vlan-id {{ value.id }};
        {%- endif %}
        {%- endfor %}
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan {
                    members [ {% for vlan, value in VLANs.items() %}{{ value.name }} {%
endfor %}];
                }
            }
        }
    }
    {%- endif %}
    {{ wan1interface }} {
        description wan1;
        {%- if iscluster == 'True' %}
        redundant–ether–options {
            redundancy–group 1;
        }
        {%- endif %}
        unit 0 {
            family inet {
                address {{ wan1ip }};
            }
        }
    }
    {{ wan2interface }} {
```

```
        description wan2;
        {%- if iscluster == 'True' %}
        redundant–ether–options {
           redundancy–group 1;
        }
        {%- endif %}
        unit 0 {
           family inet {
           {% if wan2dhcp == 'False' %}address {{ wan2ip }};{% endif %}
           {%- if wan2dhcp == 'True' %}
           dhcp {
              retransmission-attempt 50000;
              retransmission-interval 6;
              force-discover;
           }
           {%- endif %}
           }
        }
     }
     {{ laninterface }} {
        {%- for vlan, value in VLANs.items() %}
        unit {{ value.id }} {
           family inet {
              address {{ value.network | ipaddr('net') | ipaddr(value.gw) }};
           }
        }
        {%- endfor %}
     }
       lo0 {
     unit 0 {
        family inet {
           filter {
              input re-filter;
           }
        }
     }
   }
   {%- for vpn, value in VPNs.items() %}
   {{ value.int }} {
   description {{ value.name }}-vpn;
   family inet;
   }
   {{ value.int }}00 {
   description {{ value.name }}-vpn-bu;
   family inet;
   }
   {%- endfor %}
}
routing-options {
static {
   route 0.0.0.0/0 {
      next-hop {{ wan1gw }};
      qualified-next-hop {{ wan2gw }} {
```

```
            preference 10;
        }
    }
{%- for vpn, value in VPNs.items() %}
    route {{ value.network }} {
        next-hop {{ value.int }};
        qualified-next-hop {{ value.int }}00 {
            preference 10;
        }
    }
{%- endfor %}
}
  instance-import import-wan;
}
routing-instances {
      wan1 {
          interface {{ wan1interface }}.0;
          instance-type virtual-router;
      }
      wan2 {
          interface {{ wan2interface }}.0;
          instance-type virtual-router;
      }
  }
policy-options {
    policy-statement import-wan {
        from {
            instance-list [ wan1 wan2 ];
            interface [ {{ wan1interface }}.0 {{ wan2interface }}.0 ];
        }
        then accept;
    }
}
snmp {
    location "{{ street }} {{ city }}  {{ country }}";
    community {{ snmpcommunity }};
}
{%- if iscluster == 'False' %}
vlans {
     {%- for vlan, value in VLANs.items() %}
     {{ value.name }} {
         vlan-id {{ value.id }};
         l3-interface irb.{{ value.id }};
    }
     {%- endfor %}
}
{%- endif %}
firewall {
    filter re-filter {
        term allow-ssh-management {
            from {
                source-prefix-list {
                    mgmt-addresses;
```

```
        }
        protocol tcp;
        destination-port 22;
    }
    then accept;
}
term allow-snmp-monitoring {
    from {
        source-prefix-list {
            mgmt-addresses;
            monitoring-addresses;
        }
        protocol udp;
        destination-port 161;
    }
    then accept;
}
term allow-ping-monitoring {
    from {
        source-prefix-list {
            mgmt-addresses;
            monitoring-addresses;
            private-addresses;
        }
        protocol icmp;
        icmp-type [ echo-request unreachable echo-reply ];
    }
    then accept;
}
term allow-ike {
    from {
        protocol udp;
        port [ 500 4500 ];
    }
    then accept;
}
term allow-esp {
    from {
        protocol esp;
    }
    then accept;
}
term allow-established {
    from {
        tcp-established;
    }
    then accept;
}
term allow-ntp-replies {
    from {
        source-address {
            194.100.49.151/32;
            194.100.49.152/32;
```

```
        }
        protocol udp;
        source-port 123;
    }
    then accept;
}
term allow-dns-replies {
    from {
        source-address {
            {{ wandns1 }};
            {{ wandns2 }};
        }
        source-prefix-list {
            private-addresses;
        }
        protocol udp;
        source-port 53;
        destination-port 53;
    }
    then accept;
}
term allow-icmp-replies {
    from {
        protocol icmp;
        icmp-type echo-reply;
    }
    then accept;
}
term allow-dhcp-client-accept {
    from {
        source-address {
            0.0.0.0/32;
        }
        destination-address {
            255.255.255.255/32;
        }
        protocol udp;
        source-port 68;
    }
    then accept;
}
term allow-dhcp-server-accept {
    from {
        protocol udp;
        source-port [ 67 68 ];
        destination-port [ 67 68 ];
    }
    then accept;
}
term deny-rest {
    then {
        log;
        discard;
```

```
            }
        }
    }
}
policy-options {
  prefix-list mgmt-addresses {
  {%- for address in managementaddresses %}
    {{ address }};
  {%- endfor %}
  }
  prefix-list monitoring-addresses {
  {%- for address in monitoringaddresses %}
    {{ address }};
  {%- endfor %}
  }
  prefix-list private-addresses {
        192.168.0.0/16;
        172.16.0.0/12;
        10.0.0.0/8;
  }
}
security {
    ike {
        proposal sha2aes2dh5lt28800 {
            authentication-method pre-shared-keys;
            dh-group group5;
            authentication-algorithm sha-256;
            encryption-algorithm aes-256-cbc;
            lifetime-seconds 28800;
        }
        {%- for vpn, value in VPNs.items() %}
        policy {{ value.name }}-main-p1-pol {
            mode main;
            proposals sha2aes2dh5lt28800;
            pre-shared-key ascii-text {{ value.psk }};
        }
        policy {{ value.name }}-aggr-p1-pol {
            mode aggressive;
            proposals sha2aes2dh5lt28800;
            pre-shared-key ascii-text {{ value.psk }};
        }
        gateway {{ value.name }}-gw {
            ike-policy {{ value.name }}-main-p1-pol;
            address {{ value.gwaddr }};
            dead-peer-detection;
            external-interface {{ wan1interface }}.0;
        }
        gateway {{ value.name }}-gw-bu {
            ike-policy {{ value.name }}-aggr-p1-pol;
                    local-identity hostname {{ hostname }}-fw;
            address {{ value.gwaddr }};
            dead-peer-detection;
            external-interface {{ wan2interface }}.0;
```

```
        }
        {%- endfor %}
    }
    ipsec {
        proposal sha2aes2lt3600 {
            protocol esp;
            authentication-algorithm hmac-sha-256-128;
            encryption-algorithm aes-256-cbc;
            lifetime-seconds 3600;
        }
        policy sha2aes2lt3600-dh5 {
            perfect-forward-secrecy {
                keys group5;
            }
            proposals sha2aes2lt3600;
        }
        {%- for vpn, value in VPNs.items() %}
        vpn {{ value.name }}-vpn {
            bind-interface {{ value.int }};
            ike {
                gateway {{ value.name }}-gw;
                ipsec-policy sha2aes2lt3600-dh5;
            }
            establish-tunnels immediately;
        }
        vpn {{ value.name }}-vpn-bu {
            bind-interface {{ value.int }}00;
            ike {
                gateway {{ value.name }}-gw-bu;
                ipsec-policy sha2aes2lt3600-dh5;
            }
            establish-tunnels immediately;
        }
        {%- endfor %}
    }
{%- for vpn, value in VPNs.items() %}
    zones {
          security-zone {{ value.zone }}{
              interfaces {
              {{ value.int }};
              {{ value.int }}00;
              }
          }
    }
{%- endfor %}
```

default.EX2200-24T-4G.j2

```
system {
    host-name {{ hostname }}-sw{{ swnr }};
    time-zone Europe/Helsinki;
    root-authentication {
        plain-text-password-value "{{ rootswpwd }}";
    }
    ntp {
    server 194.100.49.139;
    server 194.100.49.151;
}
time-zone Europe/Helsinki;
    name-server {
        {{ wandns1 }};
        {{ wandns2 }};
    }
    login {
        user mgmtuser {
            uid 2000;
            class super-user;
            authentication {
                plain-text-password-value "{{ mgmtuserdwpwd }}";
            }
        }
    }
    services {
        ssh {
            root-login deny;
            protocol-version v2;
            connection-limit 5;
            rate-limit 5;
        }
                netconf {
            ssh;
        }
       outbound-ssh {
                            client cluster_123.123.123.123 {
                                        device-id DEVIDA;
                                        secret $9$EkjcSlLxNds2W8XNdb2g36/CO1;
                                        services netconf;
                                        123.123.123.123 {
                                                    port 7804;
                                        }
                            }
                }

    }
        syslog {
             host {{ hostname }}-sw{{ swnr }} {
          any warning;
      }
```

```
    user * {
        any emergency;
    }
    file messages {
        any notice;
        authorization info;
    }
    file interactive-commands {
       interactive-commands info;
       match UI_CMDLINE_READ_LINE;
    }
    }
 }
interfaces {
    interface-range access-wks {
        member "ge-0/0/[0-44]";
        unit 0 {
            family ethernet-switching {
                port-mode access;
                vlan {
                    members {% for vlan, value in VLANs.items() %}{% if value.wks == 'True'
%}{{ value.name }};{% endif %}{% endfor %}
                }
            }
        }
    }
    interface-range trunk-wlan {
        member "ge-0/0/[45-46]";
        unit 0 {
            family ethernet-switching {
                {%- for vlan, value in VLANs.items() %}
                {%- if value.nwmgmt == 'True' %}
                native-vlan-id {{ value.id }};
                {%- endif %}
                {%- endfor %}
                port-mode trunk;
                vlan {
                    members [ {% for vlan, value in VLANs.items() %}{% if value.nwmgmt !=
'True' %}{{ value.name }} {% endif %}{% endfor %}];
                }
            }
        }
    }
    interface-range trunk-uplink {
        member "ge-0/0/[47-48]";
        member "ge-0/1/[0-3]";
        unit 0 {
            family ethernet-switching {
                {%- for vlan, value in VLANs.items() %}
                {%- if value.wks == 'True' %}
                native-vlan-id {{ value.id }};
                {%- endif %}
                {%- endfor %}
                port-mode trunk;
```

```
                vlan {
                    members [ {% for vlan, value in VLANs.items() %}{% if value.wks != 'True'
%}{{ value.name }} {% endif %}{% endfor %}];
                }
            }
        }
    }
    vlan {
        {%- for vlan, value in VLANs.items() %}
        {%- if value.nwmgmt == 'True' %}
        unit {{ value.id }} {
            family inet {
                address {{ value.network | ipaddr('net') | ipaddr(swip) }};
                 filter {
                input re-filter;
            }
            }
        }
        {%- endif %}
        {%- endfor %}
    }
me0 {
    unit 0 {
        family inet {
            address 10.255.255.254/30;
        }
    }
}
}
ethernet-switching-options {
    port-error-disable {
        disable-timeout 120;
    }
    storm-control {
        action-shutdown;
        interface access-wks;
    }
}

firewall {
  family inet {
    filter re-filter {
        term ssh-management {
            from {
                source-prefix-list {
                    mgmt-addresses;
                }
                protocol tcp;
                destination-port 22;
            }
            then accept;
        }
        term snmp-management {
```

```
            from {
                source-prefix-list {
                    mgmt-addresses;
                }
                protocol udp;
                destination-port 161;
            }
            then accept;
        }
        term icmp-ping {
            from {
                protocol icmp;
                icmp-type echo-request;
            }
            then accept;
        }
        term allow-icmp-replies {
            from {
                protocol icmp;
                icmp-type [ echo-reply unreachable ];
            }
            then accept;
        }
        term ntp-replies {
            from {
                protocol udp;
                source-port 123;
            }
            then accept;
        }
        term tcp-established {
            from {
                protocol tcp;
                tcp-established;
            }
            then accept;
        }
        term implicit-deny {
            then {
                log;
                discard;
            }
        }
    }
  }
}
snmp {
    community {{ snmpcommunity }};
    location "{{ street }} {{ city }}  {{ country }}";
}
routing-options {
    static {
```

```
        route 0.0.0.0/0 next-hop {% for vlan, value in VLANs.items() %}{% if value.wks ==
'True' %}{{ value.network | ipaddr('net') | ipaddr(value.gw) | ipaddr('address') }};{% endif
%}{% endfor %};
    }
}
protocols {
    lldp {
        interface all;
    }
    lldp-med {
        interface all;
    }
    rstp;
}
vlans {
     {%- for vlan, value in VLANs.items() %}
     {{ value.name }} {
         vlan-id {{ value.id }};
     {%- if value.nwmgmt == 'True' %}
         l3-interface vlan.{{ value.id }};
     {%- endif %}
    }
     {%- endfor %}
}
policy-options {
  prefix-list mgmt-addresses {
  {%- for address in managementaddresses %}
    {{ address }};
  {%- endfor %}
   {% for vlan, value in VLANs.items() %}{% if value.wks == 'True' %}{{ value.network |
ipaddr('net') | ipaddr(value.gw) | ipaddr('address') }};{% endif %}{% endfor %};
  }
}
```

default.EX2300-24T.j2

```
system {
    host-name {{ hostname }}-sw{{ swnr }};
    time-zone Europe/Helsinki;
    root-authentication {
        plain-text-password-value "{{ rootswpwd }}";
    }
    ntp {
    server 194.100.49.139;
    server 194.100.49.151;
}
time-zone Europe/Helsinki;
    name-server {
        {{ wandns1 }};
        {{ wandns2 }};
    }
    login {
        user mgmtuser {
            uid 2000;
            class super-user;
            authentication {
                plain-text-password-value "{{ mgmtuserswpwd }}";
            }
        }
    }
    services {
    outbound-ssh {
                            client cluster_123.123.123.123 {
                                    device-id DEVIDA;
                                    secret $9$EkjcSlLxNds2W8XNdb2g36/CO1;
                                    services netconf;
                                    123.123.123.123 {
                                            port 7804;
                                    }
                            }
                }
        ssh {
            root-login deny;
            protocol-version v2;
            connection-limit 5;
            rate-limit 5;
        }
                netconf {
            ssh;
        }

    }
        syslog {
             host {{ hostname }}-sw{{ swnr }} {
          any warning;
       }
```

```
    user * {
        any emergency;
    }
    file messages {
        any notice;
        authorization info;
    }
    file interactive-commands {
       interactive-commands info;
       match UI_CMDLINE_READ_LINE;
    }
    }
 }
interfaces {
    interface-range access-wks {
        member "ge-0/0/[0-44]";
        unit 0 {
            family ethernet-switching {
                interface-mode access;
                vlan {
                    members {% for vlan, value in VLANs.items() %}{% if value.wks == 'True'
%}{{ value.name }};{% endif %}{% endfor %}
                }
                storm-control default;
                recovery-timeout 120;
            }
        }
    }
    interface-range trunk-wlan {
        member "ge-0/0/[45-46]";
        {%- for vlan, value in VLANs.items() %}
        {%- if value.nwmgmt == 'True' %}
        native-vlan-id {{ value.id }};
        {%- endif %}
        {%- endfor %}
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan {
                    members [ {% for vlan, value in VLANs.items() %}{{ value.name }} {%
endfor %}];
                }
            }
        }
    }
    interface-range trunk-uplink {
        member "ge-0/0/[47-48]";
        member "ge-0/1/[0-3]";
        {%- for vlan, value in VLANs.items() %}
        {%- if value.wks == 'True' %}
        native-vlan-id {{ value.id }};
        {%- endif %}
        {%- endfor %}
        unit 0 {
```

```
            family ethernet-switching {
                interface-mode trunk;
                vlan {
                    members [ {% for vlan, value in VLANs.items() %}{{ value.name }} {%
endfor %}];
                }
            }
        }
    }
    irb {
        {%- for vlan, value in VLANs.items() %}
        {%- if value.nwmgmt == 'True' %}
        unit {{ value.id }} {
            family inet {
                address {{ value.network | ipaddr('net') | ipaddr(swip) }};
                 filter {
                input re-filter;
            }
            }
        }
        {%- endif %}
        {%- endfor %}
    }
me0 {
    unit 0 {
        family inet {
            address 10.255.255.254/30;
        }
    }
}
}
forwarding-options {
    storm-control-profiles default {
        all;
        action-shutdown;
    }
}


firewall {
  family inet {
    filter re-filter {
        term ssh-management {
            from {
                source-prefix-list {
                    mgmt-addresses;
                }
                protocol tcp;
                destination-port 22;
            }
            then accept;
        }
        term snmp-management {
```

```
            from {
                source-prefix-list {
                    mgmt-addresses;
                }
                protocol udp;
                destination-port 161;
            }
            then accept;
        }
        term icmp-ping {
            from {
                protocol icmp;
                icmp-type echo-request;
            }
            then accept;
        }
        term allow-icmp-replies {
            from {
                protocol icmp;
                icmp-type [ echo-reply unreachable ];
            }
            then accept;
        }
        term ntp-replies {
            from {
                protocol udp;
                source-port 123;
            }
            then accept;
        }
        term tcp-established {
            from {
                protocol tcp;
                tcp-established;
            }
            then accept;
        }
        term implicit-deny {
            then {
                log;
                discard;
            }
        }
    }
  }
}
snmp {
    community {{ snmpcommunity }};
    location "{{ street }} {{ city }}  {{ country }}";
}
routing-options {
    static {
```

```
        route 0.0.0.0/0 next-hop {% for vlan, value in VLANs.items() %}{% if value.wks ==
'True' %}{{ value.network | ipaddr('net') | ipaddr(value.gw) | ipaddr('address') }};{% endif
%}{% endfor %};
    }
}
protocols {
    lldp {
        interface all;
    }
    lldp-med {
        interface all;
    }
rstp {
    interface all;
}

}
vlans {
      {%- for vlan, value in VLANs.items() %}
     {{ value.name }} {
         vlan-id {{ value.id }};
     {%- if value.nwmgmt == 'True' %}
         l3-interface irb.{{ value.id }};
     {%- endif %}
    }
     {%- endfor %}
}
policy-options {
  prefix-list mgmt-addresses {
  {%- for address in managementaddresses %}
    {{ address }};
  {%- endfor %}
   {% for vlan, value in VLANs.items() %}{% if value.wks == 'True' %}{{ value.network |
ipaddr('net') | ipaddr(value.gw) | ipaddr('address') }};{% endif %}{% endfor %};
  }
}
```

template-list.yml

```
#list all templates here.
---
- default
- default-vpn
```

default.yml

```
---
mgmtuserfwpwd: Qwerty123
rootfwpwd: Qwerty123
mgmtuserswpwd: Qwerty123
rootswpwd: Qwerty123
wan2ip: 192.168.1.2/24
wan2gw: 192.168.1.1
wan2dhcp: True
```

default-vpn.yml

```
---
mgmtuserfwpwd: Qwerty123
rootfwpwd: Qwerty123
mgmtuserswpwd: Qwerty123
rootswpwd: Qwerty123
wan2ip: 192.168.1.2/24
wan2gw: 192.168.1.1
wan2dhcp: True
```

globalvariables.yml

```
---
monitoringaddresses:
  - 123.123.123.4
  - 123.123.123.5
  - 123.123.123.6
managementaddresses:
  - 123.123.123.4
  - 123.123.123.5
  - 123.123.123.6
wandns1: 1.1.1.1
wandns2: 8.8.8.8
```

hardcodedvars.yml

```
---
wan1ip:
wan1gw:
wan1dhcp:
```

dnssuffix:
custdns1:
custdns2:
customer:
asset_tag:
hostname:
snmpcommunity:
swipstart:
swnamestart:
location:
street:
postal:
city:
country:
provider:
wanid:
downspeed:
uploadspeed:
linktype:
wanipaddress:
managedby:
contactfull:
contactfname:
contactlname:
contactemail:
contactphone:
iscluster:
node0fxp:
node1fxp:
fxpgw:

VLANs.10.yml

```
VLANs:
 VLAN1:
  dhcp: True
  dhcpL: 100
  dhcpH: 200
  id: 10
  name: vlan10
  network: 172.16.10.0/24
  gw: 1
  nwmgmt: True
  dhcpsrv: cust-city-fw
  wks: True
  zone: trust
 VLAN2:
  dhcp: True
  dhcpL: 100
  dhcpH: 200
  id: 20
  name: vlan20
  network: 192.168.10.0/24
  gw: 1
  nwmgmt: False
  dhcpsrv: cust-city-fw
  wks: False
  zone: guest
 VLAN3:
  dhcp: True
  dhcpL: 100
  dhcpH: 200
  id: 30
  name: vlan30
  network: 10.0.30.0/24
  gw: 1
  nwmgmt: False
  wks: False
  dhcpsrv: cust-city-fw
  zone: trust
 VLAN4:
  dhcp: True
  dhcpL: 100
  dhcpH: 200
  id: 40
  name: vlan40
  network: 172.16.40.0/24
  gw: 1
  nwmgmt: False
  wks: False
  dhcpsrv: cust-city-fw
  zone: trust
 VLAN5:
```

dhcp: True
dhcpL: 100
dhcpH: 200
id: 50
name: vlan50
network: 172.16.50.0/24
nwmgmt: False
wks: False
dhcpsrv: cust-city-fw
zone: trust
VLAN6:
dhcp: True
dhcpL: 100
dhcpH: 200
id: 60
name: vlan60
network: 172.16.60.0/24
nwmgmt: False
dhcpsrv: cust-city-fw
wks: False
zone: trust
VLAN7:
dhcp: True
dhcpL: 100
dhcpH: 200
id: 70
name: vlan70
network: 172.16.70.0/24
nwmgmt: False
dhcpsrv: cust-city-fw
wks: False
zone: trust
VLAN8:
dhcp: True
dhcpL: 100
dhcpH: 200
id: 80
name: vlan80
network: 172.16.80.0/24
nwmgmt: False
dhcpsrv: cust-city-fw
wks: False
zone: trust
VLAN9:
dhcp: True
dhcpL: 100
dhcpH: 200
id: 90
name: vlan90
network: 172.16.90.0/24
nwmgmt: False
dhcpsrv: cust-city-fw
wks: False
zone: trust

VLAN10:
 dhcp: True
 dhcpL: 100
 dhcpH: 200
 id: 100
 name: vlan10
 network: 172.16.100.0/24
 nwmgmt: False
 dhcpsrv: cust-city-fw
 wks: False
 zone: trust

VPNs.10.yml

```
‘VPNs:
 VPN1:
  gwaddr: 1.1.1.1
  network: 10.10.10.0/24
  name: vpn1
  psk: pass
  zone: trust
  int: st0.1
 VPN2:
  gwaddr: 1.1.1.2
  network: 10.20.10.0/24
  name: vpn2
  psk: pass
  zone: trust
  int: st0.2
 VPN3:
  gwaddr: 1.1.1.3
  network: 10.30.10.0/24
  name: vpn3
  psk: pass
  zone: trust
  int: st0.3
 VPN4:
  gwaddr: 1.1.1.4
  network: 10.40.10.0/24
  name: vpn4
  psk: pass
  zone: trust
  int: st0.4
 VPN5:
  gwaddr: 1.1.1.1.5
  network: 10.50.10.0/24
  name: vpn5
  psk: pass
  zone: trust
  int: st0.5
 VPN6:
  gwaddr: 1.1.1.6
  network: 10.60.10.0/24
  name: vpn6
  psk: pass
  zone: trust
  int: st0.6
 VPN7:
  gwaddr: 1.1.1.7
  network: 10.70.10.0/24
  name: vpn7
  psk: pass
  zone: trust
  int: st0.7
```

VPN8:
 gwaddr: 1.1.1.8
 network: 10.80.10.0/24
 name: vpn8
 psk: pass
 zone: trust
 int: st0.8
VPN9:
 gwaddr: 1.1.1.9
 network: 10.90.10.0/24
 name: vpn9
 psk: pass
 zone: trust
 int: st0.9
VPN10:
 gwaddr: 1.1.1.10
 network: 10.100.0.0/24
 name: vpn10
 psk: pass
 zone: trust
 int: st0.10

default-vpn.yml

VPNs:

dailyclean

```
#!/bin/bash
rm /venv/deployment/app/tmp/*
rm /venv/deployment/app/playbooks/tmp/*
```

monthlyclean

```
#!/bin/bash
rm /venv/deployment/app/log/
```

start-deploymentscript

```
#!/bin/bash
cd "/venv/deployment/app"
source "/venv/deployment/deployment/bin/activate"
su welsq --session-command "'/venv/deployment/deployment/bin/python3.8'
'/venv/deployment/app/deploymentscript.py'"
```

deploy.sh

```
alias deploy='/venv/deployment/app/bashscripts/start-deploymentscript’
```

inventory

```
#!/bin/bash
cd "/venv/deployment/app"
source "/venv/deployment/deployment/bin/activate"
su welsq --session-command "ansible-inventory --list"
```

inventory.sh

```
alias deploy='/venv/deployment/app/bashscripts/start-deploymentscript’
```

deploymentscript.py

```
import os
import npyscreen
import copy
import time
import ansible_runner
import yaml
import re
import jinja2
from ansible.plugins.filter import ipaddr
from datetime import datetime
import csv
import ipaddress
import os.path
import signal
import sys
from functools import partial
import getpass
#Checks the lock file against provided list of devices, boolean is returned True is there is overlap and overlapping devices are listed.
def check_lockfile(devices):
    locklist = []
    islocked = False

    if os.path.isfile('./tmp/lockfile.yml') is False:
        pass
    else:
        with open('./tmp/lockfile.yml') as file:
            for line in file:
                locklist.append(line.strip('\n'))
            if None in locklist:
                pass
            else:
                for dev in devices:
                    if str(dev) in locklist:
                        islocked = True

    return islocked, locklist

#Removes devices selected in the current instance from the lockfile.
def clear_lockfile(devices):
    lines = []
    locked = []
    for device in devices:
        locked.append(str(device))
    if os.path.isfile('./tmp/lockfile.yml') is False:
        pass
    else:
        with open('./tmp/lockfile.yml', 'r') as file:
            for line in file:
                lines.append(line)
```

```
        with open('./tmp/lockfile.yml', 'w') as file:
            for line in lines:
                if not any(lock in line for lock in locked):
                    file.write(line)

#Adds devices selected in the current instance to lockfile to stop multiple users from selecting overlapping devices.
def append_lockfile(devices):
    locklist = []
    for dev in devices:
        locklist.append(str(dev))

    if os.path.isfile('./tmp/lockfile.yml') is False:
        with open('./tmp/lockfile.yml', 'w') as file:
            for device in devices:
                file.write(f'{device}\n')
    else:
        with open('./tmp/lockfile.yml', 'r') as file:
            for line in file:
                locklist.append(line.strip('\n'))
        locklist = list(dict.fromkeys(locklist))
        with open('./tmp/lockfile.yml', 'w') as file:
            for device in locklist:
                file.write(f'{str(device)}\n')
#handler for ctrl+c to clear the lockfile
def signal_handler(devices, sig, frame):
    clear_lockfile(devices)
    sys.exit(0)
#removes the lockfile is instance crashes, this is to avoid leaving devices locked. This should be improved later so that it only removes devices selected current instance.
def crashclean(type, value, traceback):
    os.remove('./tmp/lockfile.yml')
    print(f'Application Crashed sorry, lockfile has been deleted {type} {value} {traceback}')
    time.sleep(3)



#runs playbook to that gathers information about the available devices and prints to file from which the info is imported to the script.
def getDevices():
    ansible_runner.run(private_data_dir="/venv/deployment/app/log",
playbook="/venv/deployment/app/playbooks/print-dev-inf.yml",
                       quiet=1)
    firewalls = []
    switches = []
    devdir = "/venv/deployment/app/playbooks/tmp/"

    for file in os.listdir(devdir):
        values = file.split(',')
        if values[1].startswith('EX'):
            switches.append(values)
        if values[1].startswith('SRX'):
            firewalls.append(values)
```

```
    for file in os.listdir(devdir):
        fullpath = os.path.join(devdir, file)
        os.unlink(fullpath)
    if len(switches) < 1:
        switches.append('No Switches ready')
    if len(firewalls) < 1:
        firewalls.append('No Firewall Ready')

    return [switches, firewalls];

#reformats dic (vardic) to a list with keys included for the variable check form.
def dicValuesList(dic):
    lst = []
    for key in dic:
        if type(dic[key]) is dict:
            lst.append(f'{key}')
            for subkey in dic[key]:
                lst.append(f'{subkey}')
                for subsubkey in dic[key][subkey]:
                    lst.append(f'{subsubkey}:{dic[key][subkey][subsubkey]}')
        else:
            lst.append(f'{key}:{dic[key]}')
    return lst
#get the list of available templates
def getTemplates():
    templates = []

    with open(f'./template-list.yml') as file:
        templates = yaml.full_load(file)

    return templates

#get the variable dic of supplied template
def getRequiredfvar(template):
    fvars = {}
    if template == None:
        template = 'default'
    if os.path.isfile(f'./floatingvars/{template}.yml') is False:
        pass
    else:
        with open(f'./floatingvars/{template}.yml') as file:
            fvars = yaml.full_load(file)
    fvars['VLANs'] = ''
    return fvars
#get list of "floating variables", if VLANs not included it's added, deduplicates the list to
avoid duplicates.
def getAllfvar():
    templates = []
    fvarlist = []

    for template in os.listdir('./templates/'):
        templates.append(template.split('.')[0])

    templates = list(set(templates))
```

```
    for template in templates:
        if os.path.isfile(f'./floatingvars/{template}.yml') is False:
            pass
        else:
            with open(f'./floatingvars/{template}.yml') as file:
                for fvar in yaml.full_load(file):
                    fvarlist.append(fvar)
    fvarlist.append('VLANs')
    fvarlist = list(set(fvarlist))

    return fvarlist
#runs ansible playbook which runs Juniper commit check command on provided targets, configuration files (labeled {hostipaddress}.conf) are in the configdir. Returns the output.
def commitCheck(targets, configdir):
    r = ansible_runner.run(private_data_dir="/venv/deployment/app/log/",
                           playbook="/venv/deployment/app/playbooks/commit_check.yml", quiet=1,
                           extravars={"targets": f"{targets}", "configdir": f"{configdir}"})
    output = []
    fulloutput = ''

    fulloutput = str(r.stdout.read())
    for line in fulloutput.splitlines():
        if "fatal:" in line:
            output.append(escape_ansi(line))
        if "ok:" in line:
            output.append(escape_ansi(line))
        if "changed:" in line:
            output.append(escape_ansi(line))

    return output
#runs ansible playbook which runs Juniper commit command on provided targets, configuration files (labeled {hostipaddress}.conf) are in the configdir.
def commitCommit(targets, configdir):
    r = ansible_runner.run(private_data_dir="/venv/deployment/app/log/",
                           playbook="/venv/deployment/app/playbooks/commit.yml", quiet=1,
                           extravars={"targets": f"{targets}", "configdir": f"{configdir}"})
    output = []
    fulloutput = ''

    fulloutput = str(r.stdout.read())
    for line in fulloutput.splitlines():
        if "fatal:" in line:
            output.append(escape_ansi(line))
        if "ok:" in line:
            output.append(escape_ansi(line))
        if "changed:" in line:
            output.append(escape_ansi(line))

    return output

def extractSelecteddevip(switches, firewalls):
    ipstring = ''
    if None in switches:
```

```
        pass
    else:
        for i in switches[0]:
            ipstring = ipstring + i[0] + ':'
    if None in firewalls:
        pass
    else:
        ipstring = ipstring + str(firewalls[0][0][0])

    return ipstring


#generates config files based on chosen template and the given variables. Also outputs CSV files for documentation systems, configuration files and variable files to apache wwww dir for ease of access for users.
def generateConfig(template, firewalls, switches, vardic, templatedir, configdir, curwwwdir):
    fwsn = ''
    swip = ''
    vardic = copy.deepcopy(vardic)
    f = ipaddr.FilterModule()
    templatefile = ''
    loader = jinja2.FileSystemLoader(f'{templatedir}/')
    env = jinja2.Environment(loader=loader)
    env.filters.update(f.filters())
    dirty_words = ['rootswpwd', 'rootfwpwd', 'rootswpwd', 'rootswpwd', 'mgmtuserswpwd', 'mgmtuserfwpwd', 'vpnpsk', 'plain-text-password-value', 'pre-shared-key ascii-test']
    if os.path.exists(curwwwdir):
        pass
    else:
        os.mkdir(f'{curwwwdir}')


    #Easier part of the documentaions, mostly based on user input on Documentation form.
    with open(f'{curwwwdir}/contact.csv', 'w') as contactcsv:
        fieldnames = ['first_name', 'last_name', 'primary_email', 'primary_phone']
        writer = csv.DictWriter(contactcsv, fieldnames=fieldnames)
        writer.writeheader()
        writer.writerow({'first_name': vardic['contactfname'], 'last_name': vardic['contactlname'], 'primary_email': vardic['contactemail'], 'primary_phone': vardic['contactphone']})

    with open(f'{curwwwdir}/location.csv', 'w') as locationcsv:
        fieldnames = ['name', 'address_1', 'city', 'country', 'postal']
        writer = csv.DictWriter(locationcsv, fieldnames=fieldnames)
        writer.writeheader()
        writer.writerow({'name': vardic['location'], 'address_1': vardic['street'], 'city': vardic['city'], 'country': vardic['country'], 'postal': vardic['postal']})

    with open(f'{curwwwdir}/wancsv.csv', 'w') as wancsv:
        fieldnames = ['Location', 'Provider', 'account_number', 'upload_speed', 'download_speed', 'router_firewall', 'ip_addresses', 'managed_by', 'link_type']
        writer = csv.DictWriter(wancsv, fieldnames=fieldnames)
```

```
        writer.writeheader()
        writer.writerow({'Location': vardic['location'], 'Provider': vardic['provider'],
'account_number': vardic['wanid'], 'upload_speed': vardic['uploadspeed'],
'download_speed': vardic['downspeed'],
                         'router_firewall': vardic['hostname']+'-fw', 'ip_addresses':
f"{vardic['wan1ip']} gw: {vardic['wan1gw']}", 'managed_by': vardic['managedby'],
'link_type': vardic['linktype']})
    #First tricky documentation to print, gets the device list for network documentation.
    with open(f'{curwwwdir}/networks.csv', 'w') as networkscsv:
        fieldnames = ['location', 'name', 'subnet', 'gateway', 'vlan', 'switches', 'dhcp_scope',
'dhcp_server', 'dns_servers', 'domain_name']
        writer = csv.DictWriter(networkscsv, fieldnames=fieldnames)
        writer.writeheader()
        sw = ''
        counter = 0
        if None in switches:
            sw = 'none'
        else:
            while counter < len(switches[0]):
                if counter == 0:
                    sw = vardic['hostname'] + '-sw' + str(int(vardic['swnamestart']) + counter)
                    counter = counter + 1
                else:
                    sw = sw + ' ' + vardic['hostname'] + '-sw' + str(int(vardic['swnamestart']) +
counter)
                    counter = counter + 1
        if None in firewalls:
            pass
        else:
            if sw == 'none':
                sw = vardic['hostname'] + '-fw'
            else:
                sw = sw + ' ' + vardic['hostname'] + '-fw'
        for network in vardic['VLANs']:
            if vardic['VLANs'][network]['zone'] == 'trust' or 'dmz' or 'server':
                dns_servers = f"{vardic['custdns1']};{vardic['custdns2']}"
                domain_name = vardic['dnssuffix']
            else:
                dns_servers = '1.1.1.1;8.8.8.8'
                domain_name = 'dummy.domain'
            scope = str(vardic['VLANs'][network]['dhcpL']) + '-' +
str(vardic['VLANs'][network]['dhcpH'])
            writer.writerow({'location': vardic['location'], 'name':
vardic['VLANs'][network]['name'], 'subnet': vardic['VLANs'][network]['network'], 'gateway':
vardic['VLANs'][network]['gw'],
                             'vlan': vardic['VLANs'][network]['id'], 'switches': sw, 'dhcp_scope':
scope, 'dhcp_server': vardic['VLANs'][network]['dhcpsrv'], 'dns_servers': dns_servers,
'domain_name': domain_name})
    #Generates device documentation and configurations. Note that if no jinja2 template
for the device model exists for the chosen configuration template, default template for
the model is used.
    with open(f'{curwwwdir}/devices.csv', 'w') as devicescsv:
        fieldnames = ['name', 'hostname', 'configuration_status', 'configuration_type',
'primary_ip', 'location', 'contact', 'serial_number', 'asset_tag', 'manufacturer', 'model']
```

```
        writer = csv.DictWriter(devicescsv, fieldnames=fieldnames)
        writer.writeheader()
        if None in firewalls:
            pass
        else:
            with open(f'{curwwwdir}/lmfw.csv', 'w') as lmfwcsv:
                lmfieldnames = ['IP', 'displayname', 'properties']
                lmwriter = csv.DictWriter(lmfwcsv, fieldnames=lmfieldnames)
                lmwriter.writeheader()
                templatefile = 'default'
                for templatefiles in os.listdir(templatedir):
                    file = templatefiles.split('.')
                    if file[0] == template:
                        if file[1] == firewalls[0][0][1]:
                            templatefile = templatefiles
                if templatefile == 'default':
                    for templatefiles in os.listdir(templatedir):
                        file = templatefiles.split('.')
                        if file[0] == 'default':
                            if file[1] == firewalls[0][0][1]:
                                templatefile = templatefiles
                jinja2template = env.get_template(templatefile)
                config = jinja2template.render(vardic)
                if len(firewalls[0]) == 1:
                    fwsn = firewalls[0][0][2]
                else:
                    fwsn = firewalls[0][0][2] + ';' + firewalls[0][1][2]
                with
open(f'{curwwwdir}/{firewalls[0][0][1]}.{firewalls[0][0][2]}.{firewalls[0][0][0]}.tmp1', 'w') as
conffile:
                    conffile.write(config)
                with
open(f'{curwwwdir}/{firewalls[0][0][1]}.{firewalls[0][0][2]}.{firewalls[0][0][0]}.tmp2', 'w') as
conffile:
                    yaml.dump(vardic, conffile, default_flow_style=False)
                vardic['iscluster'] = 'False'
                config = jinja2template.render(vardic)
                with open(f'{configdir}/{firewalls[0][0][0]}.conf', 'w') as conffile:
                    conffile.write(config)
                with
open(f'{curwwwdir}/{firewalls[0][0][1]}.{firewalls[0][0][2]}.{firewalls[0][0][0]}.tmp1', 'r') as
dirtyfile, open(f'{curwwwdir}/{firewalls[0][0][1]}.{firewalls[0][0][2]}.{firewalls[0][0][0]}.conf',
'w') as cleanfile:
                    for line in dirtyfile:
                        if not any(dirty_word in line for dirty_word in dirty_words):
                            cleanfile.write(line)
                        for dirty_word in dirty_words:
                            if dirty_word in line:
                                cleanfile.write(f"{dirty_word} PASSWORDHERE; \n")
                with
open(f'{curwwwdir}/{firewalls[0][0][1]}.{firewalls[0][0][2]}.{firewalls[0][0][0]}.tmp2', 'r') as
dirtyfile, open(f'{curwwwdir}/{firewalls[0][0][1]}.{firewalls[0][0][2]}.{firewalls[0][0][0]}.yml',
'w') as cleanfile:
                    for line in dirtyfile:
```

```
                    if not any(dirty_word in line for dirty_word in dirty_words):
                        cleanfile.write(line)
                    for dirty_word in dirty_words:
                        if dirty_word in line:
                            cleanfile.write(f"{dirty_word} PASSWORDHERE; \n")

os.remove(f'{curwwwdir}/{firewalls[0][0][1]}.{firewalls[0][0][2]}.{firewalls[0][0][0]}.tmp1')

os.remove(f'{curwwwdir}/{firewalls[0][0][1]}.{firewalls[0][0][2]}.{firewalls[0][0][0]}.tmp2')

            wanip = vardic['wan1ip'].split('/')
            wanip = wanip[0]
            writer.writerow({'name': vardic['customer'] + ' ' + vardic['city'] + ' ' + 'firewall',
'hostname': vardic['hostname'] + '-fw',
                             'configuration_status': 'active', 'configuration_type': 'Firewall',
'primary_ip': wanip,
                             'location': vardic['location'], 'contact': vardic['contactfull'],
'serial_number': fwsn, 'asset_tag': vardic['asset_tag'], 'manufacturer': 'Juniper', 'model':
firewalls[0][0][1]})
            lmwriter.writerow({'IP': wanip, 'displayname': vardic['hostname'] + '-fw',
'properties': f"Billable=False\nLocation={vardic['street']}
{vardic['city']}\nsnmp.community={vardic['snmpcommunity']}\nCustomer={vardic['custo
mer']}"})

        if None in switches:
            pass
        else:
            with open(f'{curwwwdir}/spacesw.csv', 'w') as spaceswcsv:
                with open(f'{curwwwdir}/lmsw.csv', 'w') as lmswcsv:
                    lmfieldnames = ['IP', 'displayname', 'properties']
                    lmwriter = csv.DictWriter(lmswcsv, fieldnames=lmfieldnames)
                    lmwriter.writeheader()
                    spacefieldnames = ['hostname', 'sn', 'model']
                    spacewriter = csv.DictWriter(spaceswcsv, fieldnames=spacefieldnames)
                    for vlan in vardic['VLANs']:
                        if vardic['VLANs'][vlan]['nwmgmt'] == 'True' or 'true':
                            swip = ipaddress.ip_network(vardic['VLANs'][vlan]['network'])
                    i = -1
                    for device in switches[0]:
                        i = i + 1
                        templatefile = 'default'
                        for templatefiles in os.listdir(templatedir):
                            file = templatefiles.split('.')
                            if file[0] == template:
                                if file[1] == device[1]:
                                    templatefile = templatefiles
                        if templatefile == 'default':
                            for templatefiles in os.listdir(templatedir):
                                file = templatefiles.split('.')
                                if file[0] == 'default':
                                    if file[1] == device[1]:
                                        templatefile = templatefiles
                        vardic['swnr'] = int(vardic['swnamestart']) + i
                        vardic['swip'] = int(vardic['swipstart']) + i
```

```
                jinja2template = env.get_template(templatefile)
                config = jinja2template.render(vardic)
                writer.writerow({'name': vardic['customer'] + ' ' + vardic['city'] + ' ' + 'switch'
+ str(vardic['swnr']), 'hostname': vardic['hostname'] + '-sw' + str(vardic['swnr']),
                                 'configuration_status': 'active', 'configuration_type': 'Switch',
'primary_ip': swip[int(vardic['swip'])],
                                 'location': vardic['location'], 'contact': vardic['contactfull'],
'serial_number': device[2], 'asset_tag': vardic['asset_tag'], 'manufacturer': 'Juniper',
'model': device[1]})
                lmwriter.writerow({'IP': swip[int(vardic['swip'])], 'displayname':
vardic['hostname'] + '-fw', 'properties': f"Billable=False\nLocation={vardic['street']}
{vardic['city']}\nsnmp.community={vardic['snmpcommunity']}\nCustomer={vardic['custo
mer']}"})
                spacewriter.writerow({'hostname': vardic['hostname'] + '-sw' +
str(vardic['swnr']), 'sn': device[2], 'model': device[1]})
                with open(f'{configdir}/{device[0]}.conf', 'w') as conffile:
                    conffile.write(config)
                with open(f'{curwwwdir}/{device[1]}.{device[2]}.{device[0]}.tmp1', 'w') as
conffile:
                    conffile.write(config)
                with open(f'{curwwwdir}/{device[1]}.{device[2]}.{device[0]}.tmp2', 'w') as
conffile:
                    yaml.dump(vardic, conffile, default_flow_style=False)
                with open(f'{curwwwdir}/{device[1]}.{device[2]}.{device[0]}.tmp1', 'r') as
dirtyfile, open(f'{curwwwdir}/{device[1]}.{device[2]}.{device[0]}.conf', 'w') as cleanfile:
                    for line in dirtyfile:
                        if not any(dirty_word in line for dirty_word in dirty_words):
                            cleanfile.write(line)
                        for dirty_word in dirty_words:
                            if dirty_word in line:
                                cleanfile.write(f"{dirty_word} PASSWORDHERE; \n")
                with open(f'{curwwwdir}/{device[1]}.{device[2]}.{device[0]}.tmp2', 'r') as
dirtyfile, open(f'{curwwwdir}/{device[1]}.{device[2]}.{device[0]}.yml', 'w') as cleanfile:
                    for line in dirtyfile:
                        if not any(dirty_word in line for dirty_word in dirty_words):
                            cleanfile.write(line)
                        for dirty_word in dirty_words:
                            if dirty_word in line:
                                cleanfile.write(f"{dirty_word} PASSWORDHERE; \n")
                os.remove(f'{curwwwdir}/{device[1]}.{device[2]}.{device[0]}.tmp1')
                os.remove(f'{curwwwdir}/{device[1]}.{device[2]}.{device[0]}.tmp2')
    #set the permissions to the newly created directory in apache www dir and the files
inside, some iterations of the script had issues with having read permissions for any user,
might be redundant now.
    os.chmod(f'{curwwwdir}', 0o0775)
    for file in os.listdir(curwwwdir):
        os.chmod(f'{curwwwdir}/{file}', 0o0775)
#Function to remove ansi (colors etc.) from string.
def escape_ansi(line):
    ansi_escape =re.compile(r'(\x9B|\x1B\[)[0-?]*[ -\/]*[@-~]')
    string = ansi_escape.sub('', line)
    return string
```

```
#Modified class of a npyscreen class that triggers event_value_edited function when value is edited.
class SelectOneEvent(npyscreen.TitleSelectOne):
    def when_value_edited(self):
        self.parent.event_value_edited()


#Class of the first form, objects available in fwSelect and swSelect are generated with the getDevices function.
#When template is chosen event is triggered to show or hide "floating variables" required for the tempalte.
class DevicesAndTemplates(npyscreen.FormMultiPageAction):

    fvarSliders = []

    def create(self):
        self.nextrely = 0
        y, x = self.useable_space()
        self.fwSelect       = self.add(npyscreen.TitleMultiSelect, max_height=y//4-2, rely=1, name="Choose Firewall",
                                       values=self.parentApp.firewalls, scroll_exit=True)
        self.swSelect               = self.add(npyscreen.TitleMultiSelect, max_height=y//4-2, name="Choose Switches",
                                       values=self.parentApp.switches, scroll_exit=True)
        self.templateSelect         = self.add(SelectOneEvent, max_height=y//4-2, name="Choose a template",
                                       values=self.parentApp.templates, scroll_exit=True)
#Generate slider for each "floating variable" in the fvarlist, fvarlist is generated with the getAllfvar function.
        for fvar in self.parentApp.fvarlist:
            self.fvar = self.add_widget_intelligent(npyscreen.TitleSlider, max_height=1, name=fvar, scroll_exit=True, hidden=True, out_of=10, lowest=1)
            self.fvarSliders.append(self.fvar)

#Check that user selections/input meet certain requirements, if they are the chosen values are saved to variables in the "parent app" and next form is set.
#If requirements aren't met, user gets a notification informing them of the issue and the form doesn't change.
    def on_ok(self):
        if len(self.templateSelect.get_selected_objects()) > 0 and (self.swSelect.get_selected_objects() is not None or (self.fwSelect.get_selected_objects() is not None and len(self.fwSelect.get_selected_objects()) < 3)):
            self.parentApp.selectedTemplate = self.templateSelect.get_selected_objects()[0]
            self.parentApp.templatefvarlist.clear()
            self.parentApp.fvarcounter = 0
            self.parentApp.selectedSwitches.clear()
            self.parentApp.selectedSwitches.append(self.swSelect.get_selected_objects())
            self.parentApp.selectedFirewalls.clear()
            self.parentApp.selectedFirewalls.append(self.fwSelect.get_selected_objects())
            self.parentApp.selectedDevices.clear()
            if self.swSelect.get_selected_objects() is not None:
                for device in self.swSelect.get_selected_objects():
```

```
                self.parentApp.selectedDevices.append(device)
            if self.fwSelect.get_selected_objects() is not None:
                for device in self.fwSelect.get_selected_objects():
                    self.parentApp.selectedDevices.append(device)
                if len(self.fwSelect.get_selected_objects()) > 1:
                    self.parentApp.readyvardic['iscluster'] = 'True'
                else:
                    self.parentApp.readyvardic['iscluster'] = 'False'
            else:
                self.parentApp.readyvardic['iscluster'] = 'False'
            if self.parentApp.readyvardic['iscluster'] == 'True':
                if self.fwSelect.get_selected_objects()[0][1] == self.fwSelect.get_selected_objects()[1][1]:
                    i = 1
                    for slider in self.fvarSliders:
                        if slider.hidden == False:
                            if slider.value == 0:
                                i = i - 1
                                message_to_display = "Number of variables can't be zero"
                                npyscreen.notify_confirm(message_to_display, title='')
                            slidervalueint = int(slider.value)

self.parentApp.templatefvarlist.append(f'{slider.name}.{slidervalueint}')
                    if i == 1:
                        islocked, lockeddevices = check_lockfile(self.parentApp.selectedDevices)
                        if islocked is True:
                            npyscreen.notify_confirm(f'The following devices are locked {str(lockeddevices)}', title='')
                        else:
                            append_lockfile(self.parentApp.selectedDevices)
                            self.parentApp.setNextForm('Documentation')
                else:
                    message_to_display = 'If two firewalls selected, models must match'
                    npyscreen.notify_confirm(message_to_display, title='')
            else:
                i = 1
                for slider in self.fvarSliders:
                    if slider.hidden == False:
                        if slider.value == 0:
                            i = i - 1
                            message_to_display = "Number of variables can't be zero"
                            npyscreen.notify_confirm(message_to_display, title='')
                        slidervalueint = int(slider.value)
                        self.parentApp.templatefvarlist.append(f'{slider.name}.{slidervalueint}')
                if i == 1:
                    islocked, lockeddevices = check_lockfile(self.parentApp.selectedDevices)
                    if islocked is True:
                        npyscreen.notify_confirm(f'The following devices are locked {str(lockeddevices)}', title='')
                    else:
                        append_lockfile(self.parentApp.selectedDevices)
                        self.parentApp.setNextForm('Documentation')
```

```
        else:
            message_to_display = 'You must choose atleast one template and device and
no more than two firewalls'
            npyscreen.notify_confirm(message_to_display, title='')
#Cancel quits the script/app and clears the lockfile.
    def on_cancel(self):
        clear_lockfile(self.parentApp.selectedDevices)
        self.parentApp.setNextForm(None)

#When template is chosen floating variable sliders are hidden or revealed depending if
they are required by the template.
    def event_value_edited(self):
        if len(self.templateSelect.get_selected_objects()) > 0:
            self.fvarList = getRequiredfvar(self.templateSelect.get_selected_objects()[0])
            for fvarSlider in self.fvarSliders:
                if fvarSlider.name in self.fvarList:
                    fvarSlider.hidden = False
                else:
                    fvarSlider.hidden = True
                fvarSlider.update()

#Documentation form which contains most of the hardcoded variables, these are mostly
necessary for the documentation CSVs, but some are used by most firewall
configurations (e.g wan1ip).
class Documentation(npyscreen.FormMultiPageAction):
    documentationDic = {}

    def create(self):
        self.documentationDic['customer'] =
self.add_widget_intelligent(npyscreen.TitleText, name="CustomerName",
value='Customer Inc.', max_height=4, begin_entry=20)
        self.documentationDic['asset_tag'] =
self.add_widget_intelligent(npyscreen.TitleText, name="AssetTag",
value="palvelulaite", max_height=4, begin_entry=20)
        self.documentationDic['hostname'] =
self.add_widget_intelligent(npyscreen.TitleText, name="Hostname", value="customer-
hki", max_height=4, begin_entry=20)
        self.documentationDic['snmpcommunity'] =
self.add_widget_intelligent(npyscreen.TitleText, name="SNMP", value="notPublic",
max_height=4, begin_entry=20)
        self.documentationDic['dnssuffix'] =
self.add_widget_intelligent(npyscreen.TitleText, name="dnssuffix",
value="example.local", max_height=4, begin_entry=20)
        self.documentationDic['custdns1'] =
self.add_widget_intelligent(npyscreen.TitleText, name="custdns1", value="1.1.1.1",
max_height=4, begin_entry=20)
        self.documentationDic['custdns2'] =
self.add_widget_intelligent(npyscreen.TitleText, name="custdns2", value="8.8.8.8",
max_height=4, begin_entry=20)
        self.documentationDic['wan1ip'] = self.add_widget_intelligent(npyscreen.TitleText,
name="wan1ip", value="5.5.5.6/30", max_height=4, begin_entry=10)
        self.documentationDic['wan1gw'] =
self.add_widget_intelligent(npyscreen.TitleText, name="wan1gw", value="5.5.5.5",
max_height=4, begin_entry=20)
```

```
        self.documentationDic['wan1dhcp'] =
self.add_widget_intelligent(npyscreen.TitleText, name="wan1dhcp", value="false",
max_height=4, begin_entry=20)
        self.documentationDic['swipstart'] =
self.add_widget_intelligent(npyscreen.TitleText, name="FirstSWIP", value="5",
max_height=4, begin_entry=20)
        self.documentationDic['swnamestart'] =
self.add_widget_intelligent(npyscreen.TitleText, name="FirstSWNR", value="1",
max_height=4, begin_entry=20)
        self.documentationDic['location'] = self.add_widget_intelligent(npyscreen.TitleText,
name="LocationName", value="Customer Oulu warehouse", max_height=4,
begin_entry=20)
        self.documentationDic['street'] = self.add_widget_intelligent(npyscreen.TitleText,
name="Street", value="exampleroad 42 A 34", max_height=4, begin_entry=20)
        self.documentationDic['postal'] = self.add_widget_intelligent(npyscreen.TitleText,
name="Postal code", value="05342", max_height=4, begin_entry=20)
        self.documentationDic['city'] = self.add_widget_intelligent(npyscreen.TitleText,
name="City", value="helsinki", max_height=4, begin_entry=20)
        self.documentationDic['country'] = self.add_widget_intelligent(npyscreen.TitleText,
name="Country", value="Finland", max_height=4, begin_entry=20)
        self.documentationDic['provider'] =
self.add_widget_intelligent(npyscreen.TitleText, name="ISP", value="Elisa",
max_height=4, begin_entry=20)
        self.documentationDic['wanid'] = self.add_widget_intelligent(npyscreen.TitleText,
name="WANID", value="LL1234567/1", max_height=4, begin_entry=20)
        self.documentationDic['downspeed'] =
self.add_widget_intelligent(npyscreen.TitleText, name="DLSpeed", value="100",
max_height=4, begin_entry=20)
        self.documentationDic['uploadspeed'] =
self.add_widget_intelligent(npyscreen.TitleText, name="UPSpeed", value="100",
max_height=4, begin_entry=20)
        self.documentationDic['linktype'] = self.add_widget_intelligent(npyscreen.TitleText,
name="LinkType", value="Fiber", max_height=4, begin_entry=20)
        self.documentationDic['managedby'] =
self.add_widget_intelligent(npyscreen.TitleText, name="ManagedBy",
value="customer", max_height=4, begin_entry=20)
        self.documentationDic['contactfull'] =
self.add_widget_intelligent(npyscreen.TitleText, name="FullName", value="Julius
Caesar", max_height=4, begin_entry=20)
        self.documentationDic['contactfname'] =
self.add_widget_intelligent(npyscreen.TitleText, name="FirstName", value="julius",
max_height=4, begin_entry=20)
        self.documentationDic['contactlname'] =
self.add_widget_intelligent(npyscreen.TitleText, name="Lastname", value="caesar",
max_height=4, use_two_lines=False, begin_entry=20)
        self.documentationDic['contactemail'] =
self.add_widget_intelligent(npyscreen.TitleText, name="Email",
value="examplos.examplores@examplia.com", max_height=4, begin_entry=20)
        self.documentationDic['contactphone'] =
self.add_widget_intelligent(npyscreen.TitleText, name="Phone", value="0461234567",
max_height=4, begin_entry=20)
```

```
#Saves the variable values given by the user to the readyvardic dictionary in the parent app, if two firewalls were chosen in the first form, next form is set to FWCluster, otherwise next is template variables form.
    def on_ok(self):
        for variable in self.documentationDic:
            self.parentApp.readyvardic[variable] = self.documentationDic[variable].value
        if self.parentApp.readyvardic['iscluster'] == 'True':
            self.parentApp.setNextForm('FWCluster')
        else:
            self.parentApp.setNextForm(self.parentApp.selectedTemplate)
#On cancel, clear the lockfile and go back to the first form.
    def on_cancel(self):
        clear_lockfile(self.parentApp.selectedDevices)
        self.parentApp.setNextForm('MAIN')

#short from to prompt the user for variables required in the SRX chassis cluster configuration.
class FWCluster(npyscreen.FormMultiPageAction):
    documentationDic = {}

    def create(self):
        self.documentationDic['node0fxp'] = self.add_widget_intelligent(npyscreen.TitleText, name="node0fxp", value='10.255.255.2', max_height=4, begin_entry=20)
        self.documentationDic['node1fxp'] = self.add_widget_intelligent(npyscreen.TitleText, name="node1fxp", value="10.255.255.3", max_height=4, begin_entry=20)
        self.documentationDic['fxpgw'] = self.add_widget_intelligent(npyscreen.TitleText, name="fxpgw", value="10.255.255.1", max_height=4, begin_entry=20)



    def on_ok(self):
        for variable in self.documentationDic:
            self.parentApp.readyvardic[variable] = self.documentationDic[variable].value
        self.parentApp.setNextForm(self.parentApp.selectedTemplate)

    def on_cancel(self):
        self.parentApp.setNextForm('Documentation')
#Form to prompt the user for variables required by the chosen template, currentvar dictionary is the dictionary of the template for which the form is created.
#One of these forms is created for each template regardless of the users choise, the chosen template only affects which of these forms is shown to user.
class TemplateVariables(npyscreen.FormMultiPageAction):
    vardic = {}
    FIX_MINIMUM_SIZE_WHEN_CREATED = True
    storedic = {}
#deepcopy must be used to avoid inheriting changes when currentvar changes.
    def create(self):
        self.vardic = copy.deepcopy(self.parentApp.currentvar)
        self.storedic = copy.deepcopy(self.vardic)
        for var in self.vardic:
            self.var = self.add_widget_intelligent(npyscreen.TitleText, name=var, value=str(self.vardic[var]), max_height=2)
```

```
            self.storedic[var] = self.var
#Next form is set to the first item of the list of "floating variables" required by the chosen template. User inputted values are naturally saved to dictionary in the parent app.
    def on_ok(self):
        for var in self.vardic:
            self.vardic[var] = self.storedic[var].value
        self.parentApp.readyvardic.update(self.vardic)
        self.parentApp.setNextForm(self.parentApp.templatefvarlist[0])
#Cancel moves the users back to previous form, previous form is either Documentation of FWCluser depending on whether or not iscluster key in readyvardic is False or True.
    def on_cancel(self):
        if self.parentApp.readyvardic['iscluster'] == 'False':
            self.parentApp.setNextForm('Documentation')
        else:
            self.parentApp.setNextForm('FWCluster')
#For each floating variable ten of these forms are created. Those which are shown to user depend on the requirements of the template and the count of floating variable set by the user with the sliders on the first form.
class FloatingVariables(npyscreen.FormMultiPageAction):

    returndic = {}
    storedic = {}
    i = 0
    y = 0
    def create(self):
        self.returndic = copy.deepcopy(self.parentApp.currentfvar)
        self.storedic = copy.deepcopy(self.returndic)
        for fvartype in self.returndic:
            for fvar in self.returndic[fvartype]:
                for var in self.returndic[fvartype][fvar]:
                    self.var = self.add_widget_intelligent(npyscreen.TitleText, name=f'{fvar}.{var}', value=str(self.returndic[fvartype][fvar][var]), max_height=2)
                    self.storedic[fvartype][fvar][var] = self.var
#Verifies that some special requirements of the mandatory VLANs variables are met, appends the variable values given by user to readyvardic.
#The next form is set to be the FloatingVariables form of the next fvar item in the templates fvarlist and the user set count of the said fvar.
#If this was the last form for floating variables, configurations and documentation CSVs are generated to configdir and www dir and next form is set to Verify.
    def on_ok(self):
        self.i = 0
        self.y = 0
        for fvartype in self.storedic:
            if fvartype == 'VLANs':
                for fvar in self.storedic[fvartype]:
                    if self.storedic[fvartype][fvar]['nwmgmt'].value == 'true' or self.storedic[fvartype][fvar]['nwmgmt'].value == 'True':
                        self.i = self.i + 1
                    if self.storedic[fvartype][fvar]['wks'].value == 'true' or self.storedic[fvartype][fvar]['wks'].value == 'True':
                        self.y = self.y + 1
            else:
                self.i = 1
                self.y = 1
```

```
        if self.i == 1 and self.y == 1:
            for fvartype in self.returndic:
                for fvar in self.returndic[fvartype]:
                    for var in self.returndic[fvartype][fvar]:
                        self.returndic[fvartype][fvar][var] = self.storedic[fvartype][fvar][var].value

            self.parentApp.readyvardic.update(self.returndic)
            self.parentApp.fvarcounter = self.parentApp.fvarcounter + 1
            if self.parentApp.fvarcounter == len(self.parentApp.templatefvarlist):
                generateConfig(self.parentApp.selectedTemplate,
self.parentApp.selectedFirewalls, self.parentApp.selectedSwitches,
self.parentApp.readyvardic, self.parentApp.templatedir, self.parentApp.configdir,
self.parentApp.curwwwdir)
                self.parentApp.setNextForm('Verifyvariables')
            else:

self.parentApp.setNextForm(self.parentApp.templatefvarlist[self.parentApp.fvarcounter]
)
        else:
            npyscreen.notify_confirm('You must set one and only one WKS and NWMGMT
network', title='')

#sets the next form to either previous floating variable form or the template form of the
selected template.
    def on_cancel(self):
        self.parentApp.fvarcounter = self.parentApp.fvarcounter - 1
        if self.parentApp.fvarcounter < 0:
            self.parentApp.fvarcounter = 0
            self.parentApp.setNextForm(self.parentApp.selectedTemplate)
        else:

self.parentApp.setNextForm(self.parentApp.templatefvarlist[self.parentApp.fvarcounter]
)



#this form just lists all the variables and their user given values for verification by the
user, on cancel you go to the last floating variable page and on ok you go to the
CommitCheckOutput form.
class Verify(npyscreen.FormMultiPageAction):

    def create(self):
        self.verifyvars = self.add(npyscreen.Pager, values='')

    def beforeEditing(self):
        self.verifyvars.values = dicValuesList(self.parentApp.readyvardic)


    def on_ok(self):
        self.parentApp.setNextForm('CommitCheckOutput')

    def on_cancel(self):
        self.parentApp.fvarcounter = self.parentApp.fvarcounter - 1
```

```
self.parentApp.setNextForm(self.parentApp.templatefvarlist[self.parentApp.fvarcounter]
)

#This form displays runs the commit check on the devices with config generated in the
last floating variable form and displays the output given by the ansible playbook.
#If the output of the commit check playbook doesn't contain 'FAILED!' string, the user
can press OK to commit the configurations to selected devices, if iscluster is True, SRX
firewalls are set to cluster mode and rebooted.
#For chassis cluster firewalls, the user has to manually input the configurations after they
have rebooted. The commit playbook will timeout, as the devices lose connectivity when
the configuration is commited, I might fix this later.
class CommitCheckOutput(npyscreen.FormMultiPageAction):
    dummylist = []
    dummylist.append(None)
    failmsg = "Please fix you variables, configs generated with them can't be commmited!"
    def create(self):
        self.commitcheck = self.add(npyscreen.Pager, values='', autowrap=True)
#commit check and commit playbooks are run against string that contains all the target
ip addresses separated by a colon (e.g 1.1.1.1:1.1.1.2:1.1.1.3), extractselecteddevip
creates this string.
    def beforeEditing(self):
        self.commitcheck.values                                                    =
commitCheck(extractSelecteddevip(self.parentApp.selectedSwitches,
self.parentApp.selectedFirewalls), self.parentApp.configdir)

    def on_ok(self):
        if 'FAILED!' in str(self.commitcheck.values):
            npyscreen.notify_confirm(self.failmsg, title='')
        else:
            if self.parentApp.readyvardic['iscluster'] == 'True':
                message_to_display  =  commitCommit(extractSelecteddevip(self.dummylist,
self.parentApp.selectedFirewalls), self.parentApp.configdir)
                if 'FAILED!' in str(message_to_display):
                    npyscreen.notify_confirm(self.failmsg, title='')
                else:
                    ansible_runner.run(private_data_dir="/venv/deployment/app/log/",
                                       playbook="/venv/deployment/app/playbooks/cluster.yml",
quiet=1,
                                       extravars={"target":    self.parentApp.selectedFirewalls[0][0][0],
"id": "0"})
                    ansible_runner.run(private_data_dir="/venv/deployment/app/log/",
                                       playbook="/venv/deployment/app/playbooks/cluster.yml",
quiet=1,
                                       extravars={"target":    self.parentApp.selectedFirewalls[0][1][0],
"id": "1"})
                    clear_lockfile(self.parentApp.selectedDevices)
                    self.parentApp.setNextForm(None)

            else:
                message_to_display                                                 =
commitCommit(extractSelecteddevip(self.parentApp.selectedSwitches,
self.parentApp.selectedFirewalls), self.parentApp.configdir)
                if 'ConnectTimeoutError' in str(message_to_display):
```

```
                npyscreen.notify_confirm('Devices have been configured', title='')
                clear_lockfile(self.parentApp.selectedDevices)
                self.parentApp.setNextForm(None)
            else:
                npyscreen.notify_confirm('Something went wrong', title='')

    def on_cancel(self):
        self.parentApp.setNextForm('Verifyvariables')

#The actual application is created here, most of these variables are here so that values set in forms can be saved to variable accesible to other forms.
class DeploymentApp(npyscreen.NPSAppManaged):

    templates = getTemplates()
    fvarlist = getAllfvar()
    switches, firewalls = getDevices()
    selectedTemplate = ''
    fvarcounter = 0
    fvarlistlength = 0
    templatefvarlist = []
    currentfvar = {}
    currentvar = ''
    readyvardic = {} #this is the main variable dictionary against which, configurations are generated with jinja2 templates and variables in this dictionary.
    selectedSwitches = []
    selectedFirewalls = []
    documentationDic = {}
    selectedDevices = []
    templatedir = "./templates"
    templatefile = ''
    wwwdir = '/var/www/html'
    configdir = '/venv/deployment/app/tmp'
    curwwwdir = f'{wwwdir}/{datetime.now().strftime("%Y-%m-%d-%H-%M")}'
    signal.signal(signal.SIGINT, partial(signal_handler, selectedDevices)) #this creates the handler for ctrl+c, which in turn clears the lockfile
    sys.excepthook = crashclean #This creates "crash" handler so delete the lockfile is the script crashes
#Next few lines import global variables to readyvardic
    if os.path.isfile('./variablefiles/globalvariables.yml') is False:
        pass
    else:
        with open('./variablefiles/globalvariables.yml') as file:
            globalvars = yaml.full_load(file)
            readyvardic.update(globalvars)
#This creates all the form, note that fvar forms and template variable forms are "dynamically" created based on yml files in floatingvars and variablefiles directories.
    def onStart(self):
        self.addForm('MAIN', DevicesAndTemplates, name='Device and Template options')
        self.addForm('Documentation', Documentation, name='Documentation')
        self.addForm('FWCluster', FWCluster, name='FWCluster')
        for fvar in self.fvarlist:
            for i in range(10):
                i = i+1
```

```
                with open(f'./floatingvars/{fvar}.{i}.yml') as file:
                    self.currentfvar = yaml.full_load(file)
                    self.addForm(f'{fvar}.{i}', FloatingVariables, name=f'{fvar}.{i}')
        for template in self.templates:
            with open(f'./variablefiles/{template}.yml') as file:
                self.currentvar = yaml.full_load(file)
            self.addForm(template, TemplateVariables, name=template)
        self.addForm('Verifyvariables', Verify, name='Verify the given variables')
        self.addForm('CommitCheckOutput', CommitCheckOutput, name='Commit Check
Output')


    def onCleanExit(self):
        npyscreen.notify_wait("Goodbye!")



if __name__ == '__main__':
    deploymentApp = DeploymentApp()
    deploymentApp.run()
```

hardcodedvars.txt

customer: Asiakkaan nimi, käytetään LMään lisäämis CSV:ssä, täytyy täsmätä asiakkaan nimeen laskutusjärjestelmässä.
asset_tag: palvelu- vai ostolaite
hostname: Laitteiden hostnamen alkuosa, esim. asiakas-hki, loppu osa täytetään automaattisesti konfiguraatio pohjissa
swipstart: ensimmäisen kytkimen osoite verkkolaitteidenhallintaverkosta (nwmgmt), pelkkä viimeinen oktetti (esim. 5)
swnamestart: ensimmisen kytkimen hostname osuuden numero (yleensä 1)
location: Sijainnin nimi jos jo olemassa IT Gluessa, täytyy täsmätä tarkalleen.
street: katuosoite, ei tarvita jos sijainti jo dokumentoitu
postal: postinumero, ei tarvita jos sijainti jo dokumentoitu
city: kaupunki, ei tarvita jos sijainti jo dokumentoitu
country: maa, ei tarvita jos sijainti jo dokumentoitu
provider: palveluntarjoaja, ei tarvita jos liittymä jo dokumentoitu
wanid: liittymätunnus, ei tarvita jos liittymä jo dokumentoitu
downspeed: latausnopeus, ei tarvita jos liittymä jo dokumentoitu
uploadspeed: lähetysnopeus, ei tarvita jos liittymä jo dokumentoitu
linktype: linkkityyppi, ei tarvita jos liittymä jo dokumentoitu
wanipaddress: liittymän ip-osoitteet, ei tarvita jos liittymä jo dokumentoitu
managedby: liittymän omistaja, ei tarvita jos liittymä jo dokumentoitu
contactfull: Kontaktin kokonimi, jos kontakti on jo dokumentoitu, täytyy täsmätä siihen.
contactfname: etunimi, ei tarvita jos kontakti on jo dokumentoitu
contactlname: sukunimi, ei tarvita jos kontakti on jo dokumentoitu
contactemail: sposti, ei tarvita jos kontakti on jo dokumentoitu
contactphone: puhelinnumero, ei tarvita jos kontakti on jo dokumentoitu

VLANs.txt

VLAN1:
dhcp: Konffitaanko DHCP, True tai False
nwmgmt: Onko verkkoalaitteiden hallintaverkko, True tai False, vain yksi voi ja pitää olla.
dhcpsrv: cust-city-fw, dokumentaatioon, palomuurin tai DHCP palvelimen nimi.
wks: Onko työasemaverkko, True tai False, vain yksi voi ja pitää olla.