



HoGent

Faculteit Bedrijf en Organisatie

Automatische opzet van Cisco IOS netwerkapparatuur in combinatie met Ansible

Gianni Stubbe

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Jens Buysse
Co-promotor:
Bert Van Vreckem

Instelling: —

Academiejaar: 2015-2016

Derde examenperiode

Faculteit Bedrijf en Organisatie

Automatische opzet van Cisco IOS netwerkapparatuur in combinatie met Ansible

Gianni Stubbe

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Jens Buysse
Co-promotor:
Bert Van Vreckem

Instelling: —

Academiejaar: 2015-2016

Derde examenperiode

Samenvatting

In deze bachelorproef werd onderzoek verricht naar het automatiseren van netwerkapparatuur in combinatie met Ansible en een tegenhanger NAPALM. Er werd specifiek dieper onderzoek gedaan naar apparaten gebaseerd op de Cisco IOS software.

Eerst en vooral werd er gekeken naar de huidige stand van zaken bij de Cisco apparatuur. Nog heel wat IOS apparaten worden gebruikt in de bedrijfswereld maar dat deze qua configuratie nog vrij hard achter lopen. Deze apparaten krijgen nog altijd security updates maar hebben qua hardware en mogelijkheden toch al een kleine achterstand. Met de nieuwere apparatuur die NX-OS draaien zijn er meer mogelijkheden hiervoor. Het nadeel is dan wel dat je vaak met een automatisatietechniek zit die niet echt standaard is en het dus extra moeilijk maakt wil je veel apparaten beheren.

Als eerste grote automatisatietechniek werd gekeken naar de implementatie van Ansible. Dit is een speler die in de wereld van serverprovisioning en -automatisatie al heel wat volgers heeft. Maar nu werd er ook gekeken hoe zij het doen op vlak van netwerkautomatisatie. Na het onderzoek blijkt dat de laatste tijd heel wat vooruitgang is geboekt, bij het publiceren van dit werk zijn de network modules die hier gebruikt zijn pas enkele maanden uitgebracht.

Er werd een kleine opstelling gemaakt met een netwerkswitch van Cisco die IOS als besturingssysteem draaiende had en deze werd aangesloten op een gewone router van het merk D-Link. Hierna werd een configuratie aangemaakt uit een voorbeeld van een CCNA oefening. Deze configuratie werd daarna nagemaakt met Ansible en daarna werd dit ook op de test gezet met NAPALM.

Uit de resultaten blijkt dat het opzetten van deze methodes nog niet zo eenvoudig is als het zou blijken, maar het ziet er wel hoopvol uit. Bij Ansible was het mogelijk om het apparaat in te stellen naar de configuratie die we wensten. Dit in combinatie met variabelen en templates zoals het gewoonlijk ook werkt bij Ansible. Jammer genoeg is dit nog niet op de volledig juiste manier omdat er momenteel nog een probleem is met de module 'ios_config' die aangeraden wordt om configuraties te wijzigen op het IOS besturingssysteem. Er is gebruik gemaakt van 'ios_command' voor dit onderzoek. Dit werkt ook correct maar is niet optimaal omdat deze niet kijkt naar de huidige configuratie.

Hierna werd NAPALM bekijken, dit is een python framework die geschreven is specifiek voor het aansturen van netwerkapparaten waaronder dus ook Cisco IOS apparaten. De setup bleek iets moeilijker te zijn dan bij ansible maar eenmaal in de python shell leek alles vrij logisch. Jammer genoeg zijn er met NAPALM in combinatie met IOS enkele problemen die er voor zorgden dat het verder werken met deze methode dus geen optie was binnen dit onderzoek. Ontwikkeling is wel nog actief dus hopelijk komt er hier snel een oplossing voor. Verder is deze methode iets anders dan de Ansible methode, deze stuurt een configuratie door en geeft de verschillen met de huidige configuratie terug. Hierna kan gekozen worden of je de wijzigingen wilt doorvoeren of niet.

Daarnaast is er nog een Ansible module die inwerkt op NAPALM. Dit zou nuttig kunnen zijn voor het testen na het doorvoeren van configuratie wijzigingen met de Ansible network modules. Jammer genoeg werkte NAPALM op zich al niet en is dit dus niet op de proef gesteld. Het kan wel een interessant iets zijn voor verder onderzoek naar de ideale automatisatie methode.

Als besluit werd er gesteld dat Ansible voor dagelijks gebruik al kan gebruikt worden omdat dit op het moment van schrijven de beste methode voor handen is. De release van Ansible 2.2 is ook volop in ontwikkeling, deze brengt dan nog verbeteringen mee voor deze modules waardoor de mogelijkheden er enkel maar beter op zullen worden.

Voorwoord

Het onderwerp in kwestie is gekozen geweest door mijn interesse in Ansible op zich. Twee jaar geleden heb ik de eerste hands-on gekregen in de les van Mr. Van Vreckem waar het me al snel aansprak. Nu enkele jaren later blijkt Ansible op veel verschillende plaatsen al gebruikt te worden, dit merk ik ook in het dagelijkse leven.

Een van de aspecten waar ik hier mee in aanraking kom is binnen 'Frag-O-Matic', dit is een Lanparty in België die gemiddeld een duizendtal bezoekers trekt per editie. Hier merkte ik dat zo goed als alles die zij deden op servers in playbooks gegoten waren voor Ansible. Een van de aspecten waar wel nog veel werk aan is zijn de netwerkswitches en dergelijke.

Hier viel mij het nieuwe onderwerp in: Ansible met netwerkapparatuur. Na een beetje zoeken merkte ik snel dat dit het geschikte moment was om het onderzoek te verrichten omdat de technologie nog maar sinds kort als eerste stable versie naar buiten werd gebracht. Al snel werd mij terug duidelijk waarom ik Ansible nu ook weer zo tof vond, nu werd dit dan ook nog eens gecombineerd met Cisco apparaten.

Als laatste wil ik Stefanie Geldof en haar broer bedanken voor het gebruik van de Cisco 2960 Series switch, zonder dit was het niet mogelijk geweest dit onderzoek te verrichten. Ook BeSports en Frag-O-Matic wil ik bedanken om mij in aanraking te brengen met het idee voor iets dergelijk. Automatisatie voor een Lanparty lijkt me heel handig maar het is door hen dat ik met het idee kwam.

Ik wens de lezer veel leesplezier toe, welkom in de wereld van automatisatie.

Inhoudsopgave

1	Inleiding	11
1.1	Stand van zaken	12
1.1.1	Wat is Ansible	12
1.1.2	Wat is NAPALM	14
1.1.3	Overige mogelijkheden	15
1.2	Probleemstelling en Onderzoeksvragen	15
1.3	Opzet van deze bachelorproef	16
2	Methodologie	19
2.1	Onderzoek naar mogelijkheden	19
2.2	Uitwerking basis configuratie	20
2.3	Automatiseren van configuratie	20
3	Testopstelling	21

4	Configuratie zonder automatisatie	25
4.1	Voorbeeld configuratie	25
4.1.1	Initiële configuratie	26
4.1.2	Minimale configuratie	26
4.1.3	Complete configuratie	28
4.2	'Automatisatie' of vereenvoudiging via ingebouwde mogelijkheden	29
4.2.1	Commando's in terminal venster kopiëren	29
4.2.2	Volledige running-config in terminal venster kopiëren	30
5	Automatisatie in combinatie met Ansible	33
5.1	Configuratie Ansible	33
5.2	Ansible Playbook	34
5.2.1	Belangrijke playbook onderdelen	35
5.2.2	Uitvoeren playbook	36
5.3	Conclusie	37
6	Automatisatie in combinatie met NAPALM	39
6.1	Installatie NAPALM	40
6.2	Configuratie NAPALM	40
6.3	NAPALM Ansible Module	42
6.4	Conclusie	43
7	Conclusie	45

Bijlagen	48
A Ansible Playbook	48
B Switch Configuration Files	53
Bibliografie	68

1. Inleiding

Cisco netwerkapparatuur is een van de meest gebruikte in zijn soort binnen bedrijfsnetwerken. Grote netwerken die door verschillende bedrijven over heel de wereld aanwezig zijn worden vaak ondersteund met deze apparaten. Hoewel deze vaak rotsvast zijn en nog altijd security updates krijgen, kan een menselijk foutje er wel nog eens voor zorgen dat het even de mist ingaat.(Cisco, 2016) Het instellen van deze apparaten is dan ook nog altijd een heel omslachtige taak en vereist heel wat manuele tussenkomsten. Hoe groter het bedrijf, hoe meer tijd hier dan ook inkruipt als er bijvoorbeeld een kleine wijziging op alle apparaten moet gebeuren. De laatste jaren is er heel wat vooruitgang geboekt bij het automatiseren van servers. Nu is dan ook de nood gekomen om hetzelfde te doen bij netwerkapparaten.

Het configureren van netwerkapparatuur kan dus heel wat tijd in beslag nemen. Hoewel er vaak bij nieuwere apparatuur mogelijkheden zijn om configuratie te automatiseren, zoals bijvoorbeeld met Cisco's eigen NX-OS die draait op zijn Nexus apparatuur, is het toch vaak vervelend dat er geen eenduidige manier is om dit te doen. (Buresh e.a., 2015) Ieder merk heeft een eigen implementatie. Hiervoor zijn er de laatste enkele nieuwe mogelijkheden aan het opduiken. Waar er dus meer onderzoek naar gedaan zal worden.

Vorig jaar heeft Dries Vandooren een gelijkaardig onderzoek uitgevoerd naar automatisering voor netwerkapparatuur met Ansible. Hieruit bleek dat bij hem Ansible nog niet klaar was om gebruikt te worden binnen een productieomgeving. In de tussentijd is er al heel wat extra vooruitgang geboekt en is het dan ook interessant om te kijken in hoeverre dit gebruikt kan worden voor in productie.(Vandooren, 2015)

Er zijn drie methodes die in het onderzoek vergeleken zullen worden. De eerste is hoe

tegenwoordig vaak gewerkt wordt, de andere twee zijn beide geautomatiseerde technieken.

- Volledige Cisco configuratie's importeren
- Ansible's officiële release van Network Tools (beschikbaar vanaf Ansible 2.1)
- NAPALM: Network Automation and Programmability Abstraction Layer with Multivendor support.

1.1 Stand van zaken

Cisco IOS is een NOS (of Network Operating System). Dit is een van de oudste besturingssystemen die nog dagelijks gebruikt wordt op netwerkapparatuur. Het is gekend om heel robuust te zijn en is dus vaak aan te raden wanneer er nood is aan een iets geavanceerdere netwerkopstelling indien de nodige kennis aanwezig is. IOS staat voor Internetworking Operating Systems en gaat al een hele periode mee, de eerste versies verschenen in 1990. Hoewel dit voor informaticastandaarden al een afgeschreven iets zou zijn krijgen apparaten die bijvoorbeeld in 2006 geproduceerd zijn nog altijd security updates en zijn ze voor een bedrijfsnetwerk dan ook nog uiterst geschikt. (Slattery, 2007)

Het enige waarbij deze apparatuur wel duidelijk toont dat deze toch wel wat verouderd begint te worden is dus automatisatie en provisioning. Bij nieuwere apparatuur worden er vaak ook al verschillende besturingssystemen meegeleverd die vaak heel wat meer automatisatie mogelijkheden ingebouwd hebben. Een voorbeeld hiervan is het laatste nieuwe besturingssysteem van Cisco genaamd NX-OS die op de productlijn met naam 'Nexus' draait. Deze wordt door hen beschreven als 'Open, Programmable, Agile'. Dit lost natuurlijk nog altijd het probleem niet op dat we hebben met onze oudere maar wel nog functionele apparatuur. (Cisco, 2008)

De huidige uitwerking om een apparaat te configureren bestaat vaak uit twee methodes. De eerste is het handmatig uitvoeren van ieder commando. De tweede is een copy te nemen van de running config of startup config en deze te plakken in het terminal venster van een gelijk apparaat. Dit kan je ook doen door de config op te slaan op een TFTP server en deze dan van daaruit laten ophalen door een nog te configureren apparaat. Deze spreekt dan de TFTP server aan en neemt de configuratie op.

1.1.1 Wat is Ansible

Dit is waar Ansible een noodzakelijk iets is. Ansible is iets die de laatste jaren een heel snelle opmars aan het maken is. Maar wat is het nu precies? Ansible is een project die het automatiseren van servers over ssh mogelijk maakt. Dit klinkt niet heel spectaculair maar dat is ook niet de hoofdreden waarvoor je als netwerkbeheerder er gebruik van moet maken. Ansible maakt het mogelijk om op een heel gestructureerde manier te bepalen wat op een

server gedaan wordt, geïnstalleerd wordt, naar gekopieerd wordt etc. Het belangrijkste aspect moet dan toch wel zijn dat alles in een heel leesbare en eenvoudig verstaanbare vorm opgeslagen wordt in documentjes die in yaml geschreven zijn. Indien je niet veel verstaat van Ansible is het toch nog mogelijk om een playbook te bekijken en te zien wat er precies zal gebeuren als deze uitgevoerd wordt.

Nu zullen we even dieper ingaan van waar Ansible nu precies uit ontstaan is. De geschiedenis van Ansible zoals beschreven door oprichter DeHaan (2013).

Ansible is dus nog iets vrij recent, het is iets dat sinds 2012 in ontwikkeling is, startende bij een Red Hat Emerging Technologies groep in 2006. Hier werkten werknemers vrij aan wat zelf interessant leek voor de klant. Hieruit kwam 'Cobbler' voort, dit had als doel het beheren van datacenters eenvoudiger te maken. Dit werd vrij snel groot en kreeg mogelijkheden om bijvoorbeeld DHCP en DNS te beheren in combinatie met dit pakket. Dit werd zo groot dat het gebruikt werd om de grootste DNS netwerken te beheren, heel wat grotere gameservers zoals Modern Warfare van Call of Duty en dan nog enkele kritische zaken zoals financiële onderdelen op Wall Street.

Toen kwamen configuratie managementtools zoals Puppet en Chef op de proppen. Deze waren in combinatie met Cobbler een sterke combinatie. Cobbler stelde het basissysteem in en dan werd Puppet gebruikt om de virtuele machines op het apparaat te configureren. Hiermee bleken er nog wel wat problemen te zijn waardoor deze oplossing voor verschillende situaties niet ideaal was. Met Puppet was het toen bijvoorbeeld niet mogelijk om een server te herstarten.

Om dit probleem op te lossen werd Func in het leven geroepen. Het concept was vrij eenvoudig: een centrale server die andere servers aanspreekt om bepaalde taken uit te voeren. Dit is vrij gelijkaardig aan een 'Master/slave' configuratie waarbij de 'slaves' luisteren naar de 'master'. Hoewel het niet een heel groot succes werd is het toch gebruikt geweest door Tumblr en zijn er 2 spin-offs uit voortgekomen die elk ook in hun resultaten niet volledig waren. Dit duidt er wel op dat er in de correcte richting gewerkt werd.

Deze periode is het moment dat de populariteit van de term 'DevOps' heel erg toenam. Dit werd toen nog vooral geassocieerd met Cobbler, Puppet en Func. Daarna kreeg de betekenis een bredere zin die meer duidde op de cultuur rond het automatiseren van infrastructuur. Zelf is de oprichter van Ansible nog een lange tijd na deze periode een Puppet voorstander geweest maar het bleek al snel dat er een eenvoudigere taal nodig was om een groter publiek te kunnen bereiken. Zo wou hij ook de enthousiasteling bereiken naast de server expert.

Door alles samen te zetten kwam de nood voor Ansible. Zoals de oprichter DeHaan (2013)

het zelf zegt ' it was mostly to build the tool that you could not use for six months, come back to, and still remember'. Dus rond februari 2012 werd de fundatie gelegd voor het volledige project, omdat de ontwikkelaar al heel wat kennis had binnen de DevOps wereld sloeg het vrij snel aan. Samen met de community werd en wordt alles onderhouden en aangevuld. (DeHaan, 2013)

Nu een viertal jaar later is Ansible uitgegroeid tot een vaste waarde voor het automatiseren van infrastructuur. In de tussentijd is er nog de toevoeging van onder andere Ansible Tower (de betaalde versie van Ansible voor grotere bedrijven). Dit biedt de mogelijkheid om een overzicht te houden van je hele serverinfrastructuur en maakt het bijhouden van Ansible playbooks eenvoudig en overzichtelijk. Dit wordt vaak aangekaart op de website, maar hiernaast blijft het gewone Ansible ook bestaan voor de community en enthousiastelingen. (DeHaan, 2014)

Dan sinds kort de laatste belangrijke toevoeging, vooral ook voor dit onderzoek. De toevoeging van de network-modules aan Ansible. Vanaf versie 2.0 is de eerste mogelijkheid beschikbaar voor het automatiseren, maar het is pas vanaf versie 2.1 die op 25 mei 2016 uitgebracht werd dat deze ook in de core-modules zijn ingesloten. Deze bevat de mogelijkheid om veel netwerkapparatuur van verschillende merken aan te spreken via een Ansible playbook. Enkele voorbeelden hiervan zijn Junos VyOS en nuttig voor ons onderzoek: IOS. (Cammarata, 2016)(Ansible, 2016)

1.1.2 Wat is NAPALM

NAPALM staat voor 'Network Automation and Programmability Abstraction Layer with Multivendor support'. In andere woorden iets met een gelijkaardig doel als wat Ansible probeert te bereiken met hun network modules, het configureren van netwerkapparatuur vereenvoudigen en hier meer bepaald deze van verschillende merken. Net als Ansible is alles ook open source terug te vinden op een Github pagina waardoor ook hier community kan aan meewerken. Het nadeel aan deze optie is dat deze minder aanhangers zal hebben omdat het iets volledig nieuw op zich is speciaal voor netwerkapparatuur.(Barroso, 2016c)

De eerste commits op Github dateren van begin 2015 waardoor het nog niet zo lang bestaat op zich maar qua netwerkautomatisatie op zich dus wel al een stuk langer in ontwikkeling dan de modules van Ansible. Deze wordt nog altijd actief ondersteund waardoor het wel een goede oplossing kan zijn voor een huidig netwerk te automatiseren. Support voor verschillende apparaten is goed en duidelijk gedocumenteerd met welk netwerk besturingssysteem wat precies mogelijk is. Hieruit is vrij snel duidelijk dat IOS het grootste deel van de mogelijkheden ondersteund.(Barroso, 2016d)

1.1.3 Overige mogelijkheden

Als laatste bestaan er nog enkele kleinere opties om het opzetten van een netwerk met Cisco IOS apparatuur te vereenvoudigen. Dit is dan niet de automatisatie zoals gewoonlijk maar het is wel eenvoudiger dan het gewoon configureren van een apparaat. Omdat deze een manier bieden die vandaag nog vaak gebruikt worden zijn deze dus ook meegenomen in het onderzoek.

Een van de oudere methodes om het opzetten van een switch eenvoudiger te maken is het kopiëren van de huidige configuratie van een netwerk apparaat naar een gelijke die gelijkaardig geconfigureerd moet zijn. Op die manier moet je toch al heel wat minder werk doen. Het nadeel hiermee is dat je welk nog elk apparaat afzonderlijk moet configureren al is het dan wel niet de gehele configuratie. Het voordeel van deze methode (tegenover het gewoon handmatig configureren) kan zijn dat als je bijvoorbeeld 10 switches hebt die elk dezelfde configuratie moeten hebben. Het enige verschil zal dan de ip adressering zijn wat dan uiteindelijk een kleinere wijziging is en een stuk minder werk is dan handmatig alle commando's opnieuw invoeren. Een ander nadeel kan zijn dat dit nog vrij handmatig is waardoor een foutje toch nog af en toe kan voorkomen bij het te snel willen zijn en je dus iets over het hoofd zou kunnen zien.

Een laatste die nog vermeld moet worden is de 'ansible-cisco-snmp' module van Networklore. Deze is vorig jaar uitgebreid besproken geweest in het onderzoek van Dries Vandooren en wordt daarom hier niet verder besproken. Voor dieper onderzoek kan ik verwijzen naar zijn werk. Bij de conclusie zal er ook rekening gehouden worden met zijn bevindingen. Een interessant iets wat wel op te merken valt aan de module is dat ongeveer sinds de network modules van Ansible zelf uitgekomen zijn er geen commits meer toegevoegd zijn aan de repository op Github.(Vandooren, 2015)

1.2 Probleemstelling en Onderzoeks vragen

Tegenwoordig wordt nog heel veel netwerkbeheer gedaan aan de hand van manueel wijzigingen doorvoeren en configuraties uitschrijven. Dit kan heel tijdrovend zijn en zorgt dus voor veel overhead. Tijd is iets die veel IT'ers te kort komen en door het huidig werk te automatiseren zou er dus weer meer tijd vrijkomen voor andere taken.

Een voorbeeld uit het echte leven zou het volgende kunnen zijn. In België heb je een van de grootste Lanparties genaamd 'Frag-O-Matic'. Deze hebben jaarlijks 2 edities waar ongeveer een 1000 tal gamers op afkomen, het opzetten van dergelijke structuur is dan ook een heel werk. In het netwerkteam alleen al zitten een 15 tal enthousiastelingen die naast hun vaak full time job of school nog heel wat avonden tijd vrij maken om het netwerk voor de volgende editie op punt te stellen. Alles wordt uitgetekend, uitgewerkt en uiteindelijk getest. Zelf al hebben zij hun grootste deel van de configuratie op voorhand klaar zou het

handiger zijn moest er een automatisch manier zijn om alles in 1 keer te provisionen en zo het werk van dit team te verlichten.

Op die manier zou het per editie in plaats van alles opnieuw te kopiëren zijn eerder de playbooks eens doorlopen en aanpassen of aanvullen waar nodig. Een testopstelling kan dan ook op een korte tijd opgezet worden om de werking van bepaalde onderdelen van het netwerk te testen.

Dit is een van de vele mogelijkheden die mogelijk gemaakt worden via het automatiseren van Cisco switches en routers. Er zullen naast dit team nog veel andere teams en netwerkbeheerders zijn die een geautomatiseerde omgeving op prijs zouden stellen.

Mijn onderzoeks vragen sluiten dan ook vooral aan op de mogelijkheid van het gebruik van Ansible voor het automatiseren binnen de scope van netwerkbeheer.

- Is het mogelijk om met Ansible een netwerk te provisionen op Cisco IOS apparatuur.
- In hoeverre is de implementatie volwassen om dit voor het dagelijks leven te gebruiken en dit te laten integreren bij bedrijven.
- Zijn er dingen die niet mogelijk zijn met Ansible.
- Wat is de minimumvereiste voor de apparatuur om aan de slag te kunnen gaan en playbooks te pushen vanaf Ansible.

1.3 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeks vragen.

Hoofdstuk 3 is waar we een kijkje nemen naar hoe de testopstelling er precies uitziet en wat we precies nodig hebben om alles werkende te krijgen.

In Hoofdstuk 4 wordt de basisinstelling van het Cisco apparaat vastgelegd en wat het te bereiken resultaat moet zijn.

In Hoofdstuk 5 gaan we verder in op de bereiken van de doelstelling met het Ansible framework.

In Hoofdstuk 6 kijken we naar de mogelijkheden die beschikbaar zijn met het nog vrij onbekende NAPALM.

In Hoofdstuk 7, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeks vragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek

binnen dit domein.

2. Methodologie

2.1 Onderzoek naar mogelijkheden

Eerst en vooral werd er gezocht naar wat de mogelijkheden zijn de dag van vandaag in vergelijking met een jaar geleden. Omdat dit vooral rond Ansible draait is er dus eerst naar dat onderdeel gekeken. Het is belangrijk dat het onderzoek toekomstgericht is en dus ook met de laatste mogelijkheden werkt. Omdat er heel veel vooruitgang geboekt wordt binnen deze sector is het dus cruciaal dat dit documentatie gebruikt die zo up-to-date mogelijk is.

Hiervoor werd er gezocht op verschillende fora, er werd gekeken naar de blogposts van Ansible zelf en heel vaak wordt er ook gekeken op Github repositories. Deze zijn vaak een heel goede indicatie van in hoeverre een project nog actief ondersteund wordt en deze nog verder ontwikkeld wordt. Iets die meer dan een jaar geen commit meer heeft ontvangen kan je zo goed als niet nuttig beschouwen in vergelijking met de projecten die de laatste jaar op gang zijn gekomen.

Heel vaak werd er ook door documentatie gegaan om te kijken wat mogelijk was met welke tools om te zien of dit voldoet aan wat we willen bereiken. Goede documentatie is ook een belangrijke factor om het project goed te kunnen gebruiken in het verdere proces. Ook maakt dit ook heel veel uit om de keuze voor een bepaald pakket eenvoudiger te maken. Wanneer iets niet goed gedocumenteerd is, dan is het een heel stuk moeilijker om iets in productie te nemen en lijkt het ook geen goede oplossing voor grotere bedrijfsnetwerken bijvoorbeeld.

Het is na dit zoekproces dat Ansible en NAPALM vastgelegd werden als de 2 methodes die dieper onderzocht zouden worden.

2.2 Uitwerking basis configuratie

Om het configureren op de test te stellen moet er eerst een basis configuratie voor handen zijn. Dit is hoe het eindresultaat zou moeten zijn na het uitvoeren van gelijk welke automatisatie methode. Als dit niet zo is dan kunnen we beslissen dat het gebruik van een bepaalde techniek niet geschikt is voor dagelijks gebruik of implementatie in een bedrijfsinfrastructuur.

De basisconfiguratie zal zich richten op Cisco IOS apparatuur meer bepaald wat voor handen is. Hoofdzakelijk zal er gewerkt worden met een 2960 series switch die voor dit onderzoek ten allen tijde beschikbaar was. De configuratie zal vooral inwerken op enkele basiscommando's die in het dagelijks leven het meest gebruikt worden. Hieronder valt bijvoorbeeld VLAN tagging en bijvoorbeeld ip adressering.

2.3 Automatiseren van configuratie

Als laatste hebben we dan nog het belangrijkste aspect van dit onderzoek, dit zijnde het automatiseren van de configuratie die bereikt is met de bovenstaande methode. Er zal gestart worden van een bepaalde basisconfiguratie, dit het liefst zo dicht mogelijk bij een factory reset. Dus er wordt ook gekeken naar hoeveel werk er op voorhand nodig is om alles werkende te kunnen krijgen via een automatisch platform zoals Ansible. Ook moet er rekening gehouden worden met de configuratie in combinatie met Ansible. Hoeverre kan het automatiseren voor problemen zorgen bijvoorbeeld als Ansible een ip adres verandert en zichzelf zo buitensluit bijvoorbeeld. Dit probleem heeft dan ook te maken met dat Ansible voor netwerkapparatuur over SSH loopt die een geldige ip adressering nodig heeft om te kunnen werken.

Net zoals het configureren van een server zal er ook eerst gekeken worden naar de stappen die manueel nodig zijn om het resultaat te behalen. Daarna zullen deze stappen overgezet worden in 'Tasks' zoals dit binnen Ansible noemt. Het voordeel vergeleken met een server role is dat de stappen bij een switch en router veel eenduidiger zullen zijn. Het installeren van een DNS server op Linux is net iets ingewikkelder dan de commando's laten uitvoeren binnen Ansible die nodig zijn om een Cisco apparaat te configureren.

3. Testopstelling

Om dit alles te testen werd er gebruik gemaakt van een Cisco 2960 Series switch die geproduceerd is in 2006. Het model van deze is meer bepaald de C2960-24LT die een 24 100Mbit poorten heeft waarvan er acht voorzien zijn van POE. Als laatste zijn er nog twee Gigabit interfaces aanwezig. De switch had versie 12.2(44R)SE1 geïnstalleerd bij het uitvoeren van de tests. Deze kwam nieuw uit de doos dus stond deze bij het ontvangen volledig op standaardinstellingen. Hierna werd er gekeken hoe we de werkelijkheid zoveel mogelijk konden nabootsen. Daarom is de switch tijdens het configureren over SSH achter een router geplaatst. Op deze manier was het bijvoorbeeld ook heel eenvoudig om via een laptop op afstand en over een WiFi-netwerk het apparaat te beheren, ongeveer zoals dit in werkelijkheid zou verlopen.

Omdat er voor het instellen van de basisconfiguratie nog nood is aan een console connectie werd een seriële kabel met usb aansluiting aangekocht, dit is goedkoop terug te vinden op Ebay en werk met nieuwe computers prettig en hiervoor was ook geen installatie nodig. Dit was noodzakelijk omdat veel computers tegenwoordig geen DB9 aansluiting meer hebben, hoewel dit de enige kabel is die meegeleverd wordt met een Cisco switch.



Figuur 3.1: De aangekochte seriële consolekabel, deze is te vinden op Ebay als een 'FTDI USB to RJ45'. Er wordt ook duidelijk vermeld dat dit specifiek voor Cisco apparaten is.

Om deze in gebruik te nemen bij Windows is het even kijken onder apparaatbeheer welke poort deze precies in beslag neemt. Dit kan je terug vinden onder de sectie 'poorten'. Op mijn apparaat was dit COM4. Hierna kan je in Putty bij seriële connectie gewoon COM1 aanpassen naar COM4 en de juiste snelheid meegeven, dit staat standaard correct maar indien dit niet zo is moet het 9600 zijn. Daarna kan je aan de slag met het apparaat zoals gewoonlijk.

Op een Apple computer is de procedure een klein beetje anders. Het is een beetje zoeken hoe het apparaat precies noemt maar via tabcompletion is het vrij eenvoudig terug te vinden.

```
$ screen /dev/tty.usb
      % invoer van tab
$ screen /dev/tty.usbserial-AK05D1K3
switch1>
```

Verder is het noodzakelijk om op de switch een ip adres in te stellen en een default gateway. Enkel op deze manier is het mogelijk om een connectie vanaf afstand te maken met de switch. Verder om een SSH verbinding tot stand te kunnen brengen is er een username en een wachtwoord nodig, ook dit moet ingesteld worden. Als laatste wordt een SSH key aangemaakt. Vanaf dan is het mogelijk om via SSH connectie te maken, zoals je met een server zou doen. Dit kan via Putty op Windows of via het SSH commando op bijna ieder unix systeem. Dit is hoe Ansible en NAPALM connectie zullen maken met het apparaat dus deze stap is cruciaal voor het verdere proces.



Figuur 3.2: Links op de afbeelding de router met ip adres: 172.17.1.1 deze is aangesloten op de switch poort fa0/24. Deze behoort tot vlan1 die het ip adres 172.17.1.2 heeft. Daarboven een laptop die via usb console kabel aan de switch verbonden is en via wifi aan de router. Op deze manier was het eenvoudig de switch resetten via console kabel en eenvoudig Ansible uit te voeren over WiFi dan.

Na het opstellen van deze testopstelling werd er over gegaan op het configureren van de switch met de basis instellingen die benodigd zijn voor SSH connecties toe te staan.

4. Configuratie zonder automatisatie

Bij de configuratie van voorgemelde switch zijn er drie grote lijnen die we kunnen zien: eerst en vooral het instellen van de basisinstellingen. Dit bestaat uit het configureren van een wachtwoord, hostname en het instellen van standaardwaarden zoals een 'no ip domain-lookup'. Daarna wordt er een configuratie aangemaakt met enkele vlans. Deze krijgen een ip adres en een beschrijving. Als laatste onderdeel hebben we dan nog het instellen van een VTP (VLAN Trunk Protocol) server op het apparaat. Dit allemaal samen zou al goed moeten aantonen in hoeverre het mogelijk is om deze zaken te automatiseren.

4.1 Voorbeeld configuratie

Voor de configuratie van de Switch is er het voorbeeld gevuld van een oefening uit de CCNA cursus in combinatie met een blogpost die beschrijft hoe je SSH openzet op cisco apparatuur. De SSH connectie is nodig om vanaf Ansible scripts te kunnen laten uitvoeren. (Natarajan, 2013)

De opzet van een switch wordt door het benodigde onderdeel dan ook verdeeld in twee onderdelen. Het eerste zijn de commando's om van de initiële configuratie naar de minimale configuratie te raken. Dan daarna hebben we het tweede onderdeel met commando's die nodig zijn om de complete configuratie van de apparatuur af te maken. Hierdoor hebben we als resultaat drie verschillende configuraties in bijlage: Initial Config, Minimal Config en Complete Config. (zie Bijlage B)

4.1.1 Initiële configuratie

De initiële configuratie is deze die beschikbaar is zonder enige configuratie op de switch. Op de huidige switch stond er al wat configuratie na wat prutsen om de commando's terug op te frissen, dus werd deze teruggezet naar 'Factory Defaults'. Hiervoor zijn twee commando's nodig. Het ene commando verwijdert de huidige configuratie en de tweede verwijdert de vlan configuratie (indien deze aanwezig is) die op het moment aanwezig is op de switch. (Cisco, 2014)

```
switch1#write erase
Erasing the nvram filesystem will remove all configuration files!
  Continue? [confirm]
[OK]
Erase of nvram: complete
switch1#delete flash:vlan.dat
Delete filename [vlan.dat]?
Delete flash:vlan.dat? [confirm]
switch1#reload
System configuration has been modified. Save? [yes/no]: no
Proceed with reload? [confirm]
```

Na het uitvoeren van de commando's wordt nog even gevraagd om de reload van de switch te bevestigen (wat we ook doen). Als er gevraagd zou worden om configuratie wijzigingen op te slaan dan kiezen we voor 'no'. Hierna komt de switch in de staat terecht waarvan we uitgaan hoe deze in gebruik wordt genomen door gebruikers. Na het heropstarten zal deze dan ook vragen of we de initiële setup willen doorlopen.

4.1.2 Minimale configuratie

Omdat het nodig is om vanaf de switch op afstand te kunnen aanspreken is er nood aan SSH in combinatie met een management ip adres. Omdat we verder werken op de CCNA oefening is er gekozen om een ip adres met ip '172.17.1.2' in te stellen op vlan1. Verder wordt het crypto commando gebruikt om een RSA key aan te maken met een sterke van 2048 bits omdat dit tegenwoordig grotendeels de standaard is en er zelf al vaak geopteerd wordt om keys van 4096 bits te nemen. Verder is het ook nodig om een gebruikersnaam en wachtwoord in te stellen om gebruik te kunnen maken van remote SSH logins. Dan pas zal het mogelijk zijn om een connectie te maken over SSH. Als laatste wordt hier ook nog de 'password-encryption' service geactiveerd om de beveiliging te optimaliseren en de wachtwoorden niet in clear text in de configuratie te laten staan.

```
Would you like to enter the initial configuration dialog? [yes/no]: no
Switch>en
Switch#conf t
```

```
Enter configuration commands, one per line. End with CNTL/Z.  
Switch(config)#ip default-gateway 172.17.1.1  
Switch(config)#int vlan1  
Switch(config-if)#ip address 172.17.1.2 255.255.255.0  
Switch(config-if)#no shut  
Switch(config-if)#exit  
Switch(config)#ip domain-name website.be  
Switch(config)#crypto key generate rsa  
The name for the keys will be: Switch.website.be  
Choose the size of the key modulus in the range of 360 to 2048 for your  
General Purpose Keys. Choosing a key modulus greater than 512 may take  
a few minutes.
```

```
How many bits in the modulus [512]: 2048  
% Generating 2048 bit RSA keys, keys will be non-exportable...[OK]
```

```
Switch(config)#username switchadmin password cisco  
Switch(config)#enable secret class  
Switch(config)#no ip domain-lookup  
Switch(config)#line console 0  
Switch(config-line)#logging synchronous  
Switch(config-line)#login local  
Switch(config-line)#line vty 0 4  
Switch(config-line)#transport input ssh  
Switch(config-line)#login local  
Switch(config-line)#password cisco  
Switch(config-line)#exit  
Switch(config)#service password-encryption  
Switch(config)#end  
Switch#copy running-config startup-config  
Destination filename [startup-config]?  
Building configuration...  
[OK]  
Switch#reload  
Proceed with reload? [confirm]
```

Na dat we deze configuratie hebben ingeladen is het mogelijk om via ssh connectie te maken met de switch en vanaf daaruit de configuratie af te werken. In ons voorbeeld weten we dat we poort 1-23 zullen veranderen van configuratie dus daarom connecteer je voor ssh best via poort 24 die op vlan1 zit, dit omdat er anders disconnects kunnen voorkomen tijdens het wijzigen van de configuratie op de switch. Als we poort 24 op het management vlan houden is er geen probleem. Daarom dat we ook hier al een ip adres instellen voor de default gateway en vlan1.

4.1.3 Complete configuratie

Als laatste hebben we dan nog de volledige configuratie. Dit zijn de commando's die uitgevoerd moeten worden om de volledige correcte configuratie te verkrijgen. Het is de bedoeling dat ieder automatisatie methode hetzelfde eindresultaat behaalt, gelijk aan wat door deze commando's bereikt wordt. De onderstaande commando's moeten uitgevoerd worden om alles werkende te hebben zoals het hoort. De uiteindelijke volledige configuratie is onderaan beschikbaar in de bijlage B.

```
Username: switchadmin
Password:
Switch>en
Password:
Switch#
Switch#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#hostname switch1
switch1(config)#vtp mode server
Device mode already VTP SERVER.
switch1(config)#vtp domain Lab5
Changing VTP domain name from NULL to Lab5
switch1(config)#vtp password cisco
Setting device VLAN database password to cisco
switch1(config)#int range fa0/1-5
switch1(config-if-range)#switchport mode trunk
switch1(config-if-range)#switchport trunk native vlan 99
switch1(config-if-range)#no shutdown
switch1(config-if-range)#exit
switch1(config)#vlan 99
switch1(config-vlan)#name management
switch1(config-vlan)#vlan 10
switch1(config-vlan)#name faculty-staff
switch1(config-vlan)#vlan 20
switch1(config-vlan)#name students
switch1(config-vlan)#vlan 30
switch1(config-vlan)#name guest
switch1(config-vlan)#exit
switch1(config)#int vlan99
switch1(config-if)#ip address 172.17.99.11 255.255.255.0
switch1(config-if)#exit
switch1(config)#int range fa0/6-10
switch1(config-if-range)#switchport access vlan 30
switch1(config-if-range)#int range fa0/11-17
switch1(config-if-range)#switchport access vlan 10
switch1(config-if-range)#int range fa0/18-23
switch1(config-if-range)#switchport access vlan 20
```

```
switch1(config-if-range)#exit
switch1(config)#spanning-tree vlan 99 priority 4096
switch1(config)#end
*Mar 1 00:02:09.679: %SW_VLAN-6-VTP_DOMAIN_NAME_CHG:
    VTP domain name changed to Lab5.
switch1(config)#end
*Mar 1 00:02:11.642: %LINEPROTO-5-UPDOWN: Line protocol on
    Interface Vlan99, changed state to down
switch1(config)#end
switch1#
*Mar 1 00:02:16.985: %SYS-5-CONFIG_I: Configured from
    console by switchadmin on console
switch1#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
switch1#reload
Proceed with reload? [confirm]
```

Na het uitvoeren van bovenstaande commando's zijn er enkele vlans ingesteld op de switch en heeft deze de vtp server functionaliteit gekregen. Ook wordt er een spanning-tree prioriteit ingesteld.

4.2 'Automatisatie' of vereenvoudiging via ingebouwde mogelijkheden

Voor de configuratie van een Cisco apparaat kan vaak wel nog een basis methode gebruikt worden om alles te automatiseren. Wel moeten we hier bij zeggen dat deze methode veel grondigere moet manueel getest worden op kleine foutjes. Ook dit kunnen we nog opdelen in twee methodes.

4.2.1 Commando's in terminal venster kopiëren

Deze methode is vrij eenvoudig en is bij voorgaande voorbeelden al gebruikt geweest omdat het gelijk staat aan het letterlijk uitvoeren van bepaalde commando's. Ieder commando die je normaal zou uitvoeren staat in de configuratie in de correcte volgorde. Let op, hierbij moet je ook rekening houden dat je een enter of dergelijke ook tussen je lijnen moet plaatsen om bepaalde zaken te bevestigen. Als je een reload commando wil bevestigen moet je dus een lege lijn na het commando plaatsen.

Een voorbeeld hiervan vind je onderaan in de bijlagen. Dit is het template die hier gebruikt is geweest om de basis configuratie op te zetten waarmee Ansible zal vergeleken worden. Voor een apparaat is dit nog vrij haalbaar maar als je meerdere apparaten hebt in een netwerk waarbij je een voor een moet connecteren om dan een configuratie te plakken die

elke keer een klein beetje anders is. Dan is het nog hopen dat je een bepaald item niet vergeet te veranderen of fout verandert. Hier is dus heel veel plaats voor 'human error'. Voor de gebruikte configuratie op punt stond is de switch toch wel een paar keer gewist geweest om terug van vooraf te beginnen.

4.2.2 Volledige running-config in terminal venster kopiëren

De tweede mogelijkheid is het volledig kopiëren van een configuratie in de terminal. Wanneer men het commando 'show running-config' uitvoert dan krijg je de volledige huidige configuratie terug. Deze is zo opgezet dat deze in de configuratie modus kan geplakt worden in de terminal. Hiervoor moet je dus wel in de configuratie modus zitten die je bereikt met het commando 'configure terminal' of in het kort 'conf t'.

Ook deze methode lijkt helemaal niet feilloos en bij het uitgevoerde onderzoek werkt dit in de praktijk nog minder dan bovengenoemde methode. Bij het plakken van de complete configuratie in de configuratie modus loopt het voorbij de helft mis. Het lijkt alsof de commando's door elkaar beginnen te lopen. Ook staat niet de volledige vlan configuratie in de running-config. Deze wordt wel aangemaakt maar bijvoorbeeld de benaming mist wel wat het voor toekomstige foutopsporing iets moeilijker zou kunnen maken.

```
...
switch1(config-if)#interface FastEthernet0/11
switch1(config-if)#switchport access vlan 10
% Access VLAN does not exist. Creating vlan 10
switch1(config-if)#
switch1(config-if)#interface FastEthernet0/12
switch1(config-if)#switchport access vlan 10
switch1(config-if)#
switch1(config-if)#interface FastEthernet0/13
switch1(config-if)#switchp vlan 20
^
% Invalid input detected at '^' marker.

switch1(config-if)#
switch1(config-if)#interface FastEthernet0/20
switch1(config-if)#switchport access vlan 20
% Access VLAN does not exist. Creating vlan 20
switch1(config-if)#
switch1(config-if)#interface FastEthernet0/21
switch1(config-if)#switchport access vlan 20
switch1(config-if)#
...
```

Het lijkt er op dat het automatisch aanmaken van de vlans voor problemen zorgt bij het

4.2 'Automatisatie' of vereenvoudiging via ingebouwde mogelijkheden

verder configureren. Bij de eerste 'invalid input detected' merken we dat het commando een samenvoeging is van 'vlan 10' en 'switchport access vlan 20'. Dus voor productie lijkt dit dan toch misschien zelf nog iets minder geschikt dan eerste methode. Het voordeel aan deze wel is dat je de volledige configuratie zou meenemen. Als je een switch hebt die al gedeeltelijk geconfigureerd zou zijn dan kan dit een veiligere methode zijn.

5. Automatisatie in combinatie met Ansible

Waar het nu wel allemaal om draait is natuurlijk het automatiseren van alles binnen de Ansible structuur. Er werd een playbook opgemaakt met de basisstructuur en daar werd geleidelijk aan de benodigde configuratie aan toegevoegd.

5.1 Configuratie Ansible

Voor het gebruik van Ansible in combinatie met de 'network-modules' heb je de laatste release nodig van Ansible. Bij het schrijven is dit 2.1.1.0. Deze heeft enkele belangrijke onderdelen voor netwerk apparatuur en is dan ook de eerste versie die volledige ondersteuning biedt hiervoor. Dus voor we aan het verder onderzoek kunnen beginnen moet eerst gekeken worden of we voldoen aan deze minimum vereiste. (Ansible, 2016)

Verder is er een moeilijke keuze geweest qua besturingssysteem om de Ansible rol op uit te voeren. Er werd eerst gekozen om gebruik te maken van een MacBook met de laatste versie 'El Capitan'. Na heel wat zoeken en na het testen van de development release vanaf source bleef er een probleem aanhouden met de 'paramiko' package die nodig is om te kunnen communiceren met netwerkapparaten. Deze was zogezegd niet beschikbaar hoewel deze meerdere malen opnieuw geïnstalleerd werd. Daarna werd er gebruik gemaakt van een laptop waar Ubuntu 16.04 LTS op staat. Ook hier werd de laatste release en development versie op getest. Alles leek te werken maar er bleek een probleem met de ios_config module. Dus om het toch nog even te testen werd er naar Windows opgestart en werd er gebruik gemaakt van het nieuwe 'Bash on Ubuntu on Windows'. Ook hier werd de laatste release versie getest en de laatste versie die in ontwikkeling is. Ook hier kwam het

probleem terug met de `ios_config` module. Dit bleek dus een bug te zijn met de module op zich.¹

Verdere configuratie verliep dan verder via de nieuwe implementatie 'Bash on Ubuntu on Windows' (die beschikbaar is geworden met de Windows 10 Anniversary Update) wat verassend genoeg volledig werkte op ieder ander vlak. Hou er wel rekening mee dat deze moet opgestart worden als administrator omdat anders bepaalde netwerk mogelijkheden niet beschikbaar zijn voor bash. Het is een goede vooruitgang om ook apparaten te kunnen configureren met Ansible vanaf een Windows apparaat. Dit staat natuurlijk los van het onderzoek maar is wel interessant om even op te merken daar het verder onderzoek vanaf een Windows apparaat verliep.



Figuur 5.1: Bash in Windows of via de correcte term 'Bash on Ubuntu on Windows'

Verder zal er niet ingegaan worden op vlak van installatie van Ansible. Er wordt verwacht dat de basis van Ansible beschikbaar is op het controllerend apparaat.

5.2 Ansible Playbook

Bij het schrijven van de playbook voor Ansible werd er zoveel mogelijk gebruik gemaakt van de verschillende mogelijkheden om te tonen wat er momenteel ongeveer al allemaal mogelijk is bij het configureren van netwerksystemen. Verder werden de rollen voor verschillende apparaten opgedeeld in een logische structuur per functie.

Voor de werkelijke uitvoering van de playbook is er maar gebruik gemaakt van een apparaat namelijk 'switch1'. Er is wel nog ruimte voorzien om dit gemakkelijk uit te breiden naar een tweede apparaat. De configuratie hiervan is wel aangemaakt maar staat in commentaar. De volledige opstelling kan je vinden onderaan in de bijlagen onder Bijlage A. Verder werd er hier geen gebruik gemaakt van de module `ios_config` omdat deze in de huidige staat voor problemen zorgt. Dit is wel de aan te raden manier om configuratie wijzigingen te doen omdat deze zal kijken wat er in de huidige configuratie staat. De `ios_command` module die in deze playbook is gebruikt is dus niet de ideale manier maar kan wel werken met variabelen en templating wat je niet hebt bij het gewoon aanmaken van een configuratie.

Om een idee te geven van hoe de `ios_config` module precies werkt is er wel een 'common-config' rol toegevoegd die hetzelfde doet als de 'common-command' rol. Dit om aan

¹Github issue die dit weergeeft: <https://github.com/ansible/ansible-modules-core/issues/4419>

te tonen hoe het wel correct zou kunnen werken indien het probleem met deze module opgelost wordt.

5.2.1 Belangrijke playbook onderdelen

Vervolgens zal er wat dieper ingegaan worden op enkele specifieke onderdelen uit de playbook die kenmerkend zijn voor de network-modules.

Een van de belangrijkste onderdelen zijn de login gegevens voor toegang te krijgen tot de switch. Je kan deze afzonderlijk meegeven bij iedere task maar het is eenvoudiger om deze als een host of group variabele mee te geven. Dit is mogelijk door de parameter 'provider' die beschikbaar is bij iedere IOS module. Deze ondersteund de parameters die onder de module terecht horen in een object, speciaal voor het eenvoudig maken van templating. Verder moet ook de transport parameter meegegeven worden om de correcte connectie te kunnen maken, dit is 'cli' voor deze apparaten.

Verder is er gebruik gemaakt van hashes als variabelen. Dit maakt het heel eenvoudig om commando's die heel hard op elkaar lijken te herhalen. Dit kan bijvoorbeeld het instellen van interfaces zijn of zoals hier bijvoorbeeld het toevoegen van vlans.

```
# roles/vlan/tasks/main.yaml
- name: set vlans
  ios_command:
    commands:
      - conf t
      - vlan {{ item.key }}
      - name {{ item.value.name }}
    provider: "{{ cli }}"
  with_dict: "{{ vlans }}"

# host_vars/switch1.yaml
vlans:
  99:
    name: management
  10:
    name: faculty-staff
  20:
    name: students
  30:
    name: guest
```

Verder is er ook nog een inventory file gebruikt met geneste groepen. Als hoofdgroepen hebben we namelijk 'routers' en 'switches'. Maar onder switches zit er een subcategorie

'server-switches' en 'client-switches' die het zo mogelijk maakt om heel eenvoudig verschillende configuraties te sturen naar de verschillende groepen switches. Een client heeft bijvoorbeeld geen vlan rol, in de toekomst zouden hier nog andere verschillen kunnen bijkomen waardoor een verdeling in subcategorieën extra gemakkelijk is.

5.2.2 Uitvoeren playbook

Na dat de volledige configuratie aangemaakt was werd deze opnieuw uitgevoerd tegen een switch die terug werd gezet naar de minimale configuratie die nodig is om er van op afstand mee te connecteren via SSH. Nadien werd de configuratie naast elkaar geplaatst om te kijken wat het resultaat was. Onderstaande uitvoer van Ansible toont dat de playbook volledig correct werd uitgevoerd.

```
ubuntu@LAPTOP-GIANNI:/mnt/c/Users/Gianni/Documents/GitHub/BP2016NEW/
ansible$ ansible-playbook ios.yml -i inventory

PLAY [server-switches] ****
TASK [common-command : set hostname] ****
ok: [switch1]

TASK [common-command : disable service 'config'] ****
ok: [switch1]

TASK [vtp : set vlans] ****
ok: [switch1]

TASK [trunking : set interfaces 1-5 as trunk ports] ****
ok: [switch1]

TASK [trunking : set ip trunk vlan] ****
ok: [switch1]

TASK [vlan : set vlans] ****
ok: [switch1] => (item={'value': {u'name': u'faculty-staff'}, 'key': 10})
ok: [switch1] => (item={'value': {u'name': u'management'}, 'key': 99})
ok: [switch1] => (item={'value': {u'name': u'students'}, 'key': 20})
ok: [switch1] => (item={'value': {u'name': u'guest'}, 'key': 30})

TASK [vlan : assign interfaces to vlans] ****
ok: [switch1] => (item={'value': {u'vlan': 30}, 'key': u'6-10'})
ok: [switch1] => (item={'value': {u'vlan': 10}, 'key': u'11-17'})
ok: [switch1] => (item={'value': {u'vlan': 20}, 'key': u'18-23'})
```

```
TASK [vlan : set spanning-tree] *****
ok: [switch1]

TASK [saveconfig : save running-config to startup-config] *****
ok: [switch1]

PLAY RECAP *****
switch1 : ok=9    changed=0    unreachable=0    failed=0
```

Na het uitvoeren van de Ansible rol bleek dan ook dat de configuratie identiek gelijk was aan elkaar.

5.3 Conclusie

We kunnen hier uit besluiten dat Ansible op zich correct werkt met netwerkapparatuur. Het biedt extra mogelijkheid in vergelijking met het gewoon configureren wat het een heel stuk overzichtelijker maakt en eenvoudiger om een groot netwerkpark te beheren. Het nadeel is wel dat de module `ios_config` nog niet werkt. Deze heeft duidelijker weer wat de precieze wijzigingen zijn die toegepast zijn op de configuratie. Eenmaal deze module terug volledig werkende zou zijn lijkt dit een goede optie om het configureren van Cisco IOS apparatuur te automatiseren in iedere situatie.

Voor dit aan te kaarten is als de optimale manier van automatisatie voor Cisco apparatuur zal de tegenhanger NAPALM op de test gezet worden.

6. Automatisatie in combinatie met NAPALM

Naast Ansible hebben we nog een tweede optie die een heel stuk minder bekend is dan de eerstgenoemde. Dit is NAPALM of ook wel 'Network Automation and Programmability Abstraction Layer with Multivendor support'. Kortgezegd is dit een project geschreven in python om netwerkapparatuur te kunnen aanspreken. Het verschil met ansible zit hem hier in het feit dat je geen tasks gaat definiëren maar dat je een bepaalde configuratie opstelt voor je apparaat en deze daarna test tegenover de huidige configuratie. Voor dat deze permanent naar het apparaat doorgestuurd wordt krijg je een overzicht vergelijkbaar met dat van een git commit waar de wijzigingen zitten in de configuratie. Indien je akkoord bent met de wijzigingen kun je deze 'committen' naar het apparaat of afwijzen om de configuratie te herwerken.

Een voorbeeld van hoe de wijzigingen worden weergegeven kan je hier onder zien.

```
+ hostname pyeos-unit-test-changed
- hostname pyeos-unit-test
router bgp 65000
vrf test
+ neighbor 1.1.1.2 maximum-routes 12000
+ neighbor 1.1.1.2 remote-as 1
- neighbor 1.1.1.1 remote-as 1
- neighbor 1.1.1.1 maximum-routes 12000
vrf test2
+ neighbor 2.2.2.3 remote-as 2
+ neighbor 2.2.2.3 maximum-routes 12000
- neighbor 2.2.2.2 remote-as 2
```

```
- neighbor 2.2.2.2 maximum-routes 12000
interface Ethernet2
+ description ble
- description bla
```

Hoewel dit enkele mogelijkheden biedt die een terminal niet heeft, mist het in vergelijking met Ansible wel nog andere punten die daar wel een meerwaarde bieden. Een van de grootste maar ook enigste voordelen blijft wel het op voorhand zien van de wijzigingen die zullen gebeuren aan je apparaat. Hierdoor is iets over het hoofd zien iets dat minder vaak zal voorkomen. Moest er dan toch nog een foutje in geslopen zijn kan je nog je wijzigingen ongedaan maken met de rollback functionaliteit.

6.1 Installatie NAPALM

De installatie van NAPALM is vrij eenvoudig in het eerste opzicht. Als eerste voorwaarde wordt er verwacht van het hostsysteem dat Python geïnstalleerd is.

Na dat deze voorwaarde voldoet wordt er eerst en vooral een virtualenv ingesteld om alles in te runnen, dit zorgt ervoor dat alle dependencies correct zijn specifiek voor deze package en dat de permissies correct zijn. (Reitz, 2015)

```
$ sudo pip install virtualenv
$ virtualenv venv
$ source venv/bin/activate
```

Daarna wordt NAPALM geïnstalleerd binnen deze virtuele environment via het pip commando.

```
(venv)$ pip install napalm
```

6.2 Configuratie NAPALM

Eenmaal de installatie correct doorlopen is kunnen we aan de slag met de bibliotheek. Hiervoor werden de eerste stappen gevuld die beschreven staan in de documentatie van NAPALM. Dit beschrijft het uitvoeren van de commando's via de Python shell. Het is natuurlijk ook mogelijk om dit in een python script om te zetten om dit te vereenvoudigen. Dit zou dan ook mogelijkheid kunnen bieden tot het toevoegen van enkele commandline variabelen. Hierdoor zou je een configuratie al eenvoudiger naar een bepaald apparaat kunnen sturen maar de configuraties zelf zullen toch nog altijd op voorhand gemaakt moeten worden tot in de details. (Barroso, 2016a)

```
(venv)$ python
>>> from napalm import get_network_driver
>>> driver = get_network_driver('ios')
>>> device = driver('172.17.1.2', 'switchadmin', 'admin')
>>> device.open()
ValueError: An error occurred in dynamically determining
    remote file system: dir Translating "dir"
% Unknown command or computer name, or unable to find computer address
```

Dit is echter waar het misloopt. Bij het proberen connecteren met een Cisco IOS apparaat krijg je de bovenstaande error. Hoewel je zou denken dat het ip adres dan fout zou zijn geeft dit een andere error terug die weergegeven wordt in volgende statement. Hier werd een willekeurig ip adres ingegeven om aan te tonen dat er verschil is met het bovenstaande.

```
>>> device = driver('213.213.1.1', 'randomuser', 'randompassword')
>>> device.open()
netmiko.ssh_exception.NetMikoTimeoutException: Connection to
device timed-out: cisco_ios 213.213.1.1:22
```

Na wat zoeken op het internet blijkt dat dit een bug is bij de NAPALM bibliotheek. Het probleem is dat deze probeert het 'dir' commando uit te voeren wanneer die connecteert met het apparaat om te kijken waar de huidige configuratie zich bevindt. Het Cisco apparaat moet daarvoor in de enable mode staan zijn. Maar het 'enable' commando wordt niet uitgevoerd, noch wordt er ondersteuning geboden voor een enable password. In dezelfde Github issue staat wel uitgelegd hoe we hier omheen kunnen werken.¹ We geven bij het aanmaken van het apparaat een optionele parameter mee die aangeeft waar de directory zich bevindt.(Barroso, 2016d)

```
>>> optional_args = {'dest_file_system': 'flash:'}
>>> device = driver('172.17.1.2', 'switchadmin',
'cisco', optional_args=optional_args)
>>> device.open()
>>>
```

Nu wordt er met het commando 'load_replace_candidate' een configuratie vergeleken met de huidige running config. Ook hier loopt het jammer genoeg mis.

```
>>> device.load_replace_candidate
    (filename='../../annexes/switch_config/complete_config')
ValueError: Unexpected output from check_file_exists
```

¹<https://github.com/napalm-automation/napalm-ios/issues/14>

Hier werd gekeken of het niet zou kunnen dat de directory structuur niet zou kloppen door een willekeurige bestandsnaam in te geven. Dit bleek een andere error te geven en dus heeft het bovenstaande daar niet mee te maken.

```
>>> device.load_replace_candidate
      (filename='../../annexes/switch_config/unexisting_config')
IOError: [Errno 2] No such file or directory:
'../../annexes/switch_config/unexisting_config'
```

Dit heeft wederom te maken met de enable functionaliteit die niet aanwezig is in de package. Dit betekent dat momenteel deze bibliotheek niet gebruikt kan worden in combinatie met Cisco IOS apparatuur en het dus voor dagelijks gebruik ook afgeschreven is. Gelukkig is de development wel nog vrij actief dus kan het goed zijn dat deze binnenkort er toch nog de mogelijkheid komt om dit te gebruiken met de IOS apparaten.

6.3 NAPALM Ansible Module

Naast de gewone python implementatie heeft NAPALM ook een Ansible module die je kan gebruiken in je project. Deze moet je dan toevoegen in je library folder waarna je van de module gebruik kan maken. Hiervoor moet de NAPALM python module wel afzonderlijk geïnstalleerd zijn op de hostcomputer.

Wat een toffe combinatie zou kunnen zijn is dit combineren met de eerder besproken Ansible network modules. Zo zou je eerst het apparaat kunnen configureren met de `ios_command` of `ios_config` module. Eenmaal deze alles geconfigureerd heeft op het netwerkapparaat kan je nadien nogmaals de volledig gewenste configuratie met NAPALM doorsturen naar het apparaat. Zo kan je kijken of er verschillen zijn tussen wat je device nu draait en wat je zou verwachten. Voor productie lijkt dit dan ook een mooie extra omdat je nog veel duidelijker een beeld terug krijgt als er iets niet zou kloppen.

Een voorbeeld van het gebruik van de Ansible module zou er er als volgt kunnen uitzien. Dit uit de Github Documentatie van Barroso (2016b).

```
- name: assemble configuration
  assemble:
    src=../compiled/{{ inventory_hostname }}/
    dest=../compiled/{{ inventory_hostname }}/running.conf

- name: install napalm config
  napalm_install_config:
    hostname={{ inventory_hostname }}
```

```
username={{ user }}  
dev_os={{ os }}  
password={{ passwd }}  
config_file=../compiled/{{ inventory_hostname }}/running.conf  
commit_changes={{ commit_changes }}  
get_diffs=True  
diff_file=../compiled/{{ inventory_hostname }}/diff
```

6.4 Conclusie

NAPALM kan een grote vooruitgang zijn op de huidige werking bij het configureren van een netwerkapparaat. Hoewel er nog vaak aan gewerkt lijkt te worden mist er toch nog de volledige ondersteuning voor Cisco IOS apparaten. Daarom is dit voor het gebruik in combinatie met die apparatuur nog niet mogelijk om dit correct te gebruiken.

Verder kunnen we wel besluiten dat de beloofde mogelijkheden een goede manier zijn om configuratiewijzigingen overzichtelijk te kunnen weergeven. Dit maakt het wijzigen van configuraties veiliger om te doen en beter maar vooral automatisch gedocumenteerd.

7. Conclusie

Nu we de belangrijkste mogelijkheden hebben gezien is het tijd voor het besluit. De automatisatie van servers is al even een standaard aan het worden in combinatie met Ansible. Maar is Ansible ook capabel genoeg om netwerkapparaten en meer bepaald Cisco IOS apparaten aan te spreken? Kort gezegd, ja. Maar wel met enkele opmerkingen.

Er is nog nood aan verdere actieve ontwikkeling van het project om het volledig klaar te noemen voor dagelijks gebruik. Toch lijkt het al goed genoeg om dit aan te raden voor mensen met grotere netwerken die zij te beheren hebben. Met de module die tijdens dit project gebruikt is kan je al heel ver raken en heb je ook alle voordelen die Ansible heeft. Dus het gebruiken van variabelen, meerdere apparaten in een playbook aanspreken, enz.

De tegenhanger NAPALM daarentegen is nog niet volledig genoeg om gebruik te maken in het dagelijks leven, toch zeker niet bij IOS. Hoewel dit niet wil zeggen dat andere besturingssystemen niet werken kan het toch een teken zijn dat er minder ondersteuning voorzien wordt.

In vergelijking met het onderzoek die vorig jaar verricht is door Dries Vandooren is er al heel wat verbetering. Naast zoals die van Networklore die vaak door een paar enkele personen onderhouden wordt hebben we nu ook de officiële Ansible modules nu die door een veel groter publiek gevolgd worden maar ook beter onderhouden worden. Dit is dan ook een veel betere en een meer volledige implementatie dan deze van Networklore die met SNMP werkte.

Om even terug te komen op de onderzoeks vragen. Is het mogelijk om apparaten met het IOS te provisionen via Ansible? Het kan bevestigd worden dat dit werkt en het is dan ook een goede ontwikkeling dat dit mogelijk is. Is het klaar voor dagelijks gebruik? Dit hangt af van de situatie maar in de meeste gevallen kunnen we dit aanraden. De reden hiervoor is het antwoord op de vraag of Ansible iets nog niet kan. En dit is vooral tonen wat er gewijzigd is en wat niet. Dit zit in het nu niet werkende `ios_config`, maar het bestaat wel. Los van dit werkt het wel zoals we gewoon zijn van Ansible.

Als laatste, voor je zelf aan de slag gaat moet je wel rekening houden met de minimum vereisten voor deze onderdelen in gebruik te nemen. Voor Ansible is er nood aan een versie die minimum 2.1.0 is. Op het apparaat die Cisco draait moet SSH ingeschakeld worden samen met een gebruikersnaam en een wachtwoord.

Verdere ontwikkelen zullen uitwijzen hoe ver de mogelijkheden zullen gaan met deze frameworks. De vooruitgang die nu al is geboekt, is onweerlegbaar al een hele stap en het is dan ook uitkijken naar welke grote projecten we hier rond zullen zien.

Verder onderzoek naar hoe we met Ansible nu een volledig netwerk inclusief servers kunnen beheren lijkt dan ook een interessante opvolging.

Bijlagen

A Ansible Playbook

```
|-- group_vars
|   '-- switches.yaml
|-- host_vars
|   |-- switch1.yaml
|   '-- switch2.yaml
|-- roles
|   |-- common-command
|   |   '-- tasks
|   |       '-- main.yaml
|   |-- common-config
|   |   '-- tasks
|   |       '-- main.yaml
|   |-- saveconfig
|   |   '-- tasks
|   |       '-- main.yaml
|   |-- trunking
|   |   '-- tasks
|   |       '-- main.yaml
|   |-- vlan
|   |   '-- tasks
|   |       '-- main.yaml
`-- vtp
    '-- tasks
    '-- main.yaml
|-- inventory
`-- ios.yaml
```

group_vars/switches.yaml

```
cli:  
  host: "{{ management_ip }}"  
  username: "switchadmin"  
  password: "cisco"  
  authorize: yes  
  auth_pass: "class"  
  transport: cli
```

host_vars/switch1.yaml

```
management_ip: 172.17.1.2  
  
vlan99_ip: "172.17.99.11 255.255.255.0"
```

```
vlans:  
  99:  
    name: management  
  10:  
    name: faculty-staff  
  20:  
    name: students  
  30:  
    name: guest
```

interfaces:

```
  6-10:  
    vlan: 30  
  11-17:  
    vlan: 10  
  18-23:  
    vlan: 20
```

```
vtp: server
```

host_vars/switch2.yaml

```
management_ip: 172.17.1.3  
  
vlan99_ip: "172.17.99.12 255.255.255.0"
```

```
vtp: client
```

roles/common-command/tasks/main.yaml

```
- name: set hostname  
  ios_command:
```

```

  commands:
    - conf t
    - hostname "{{ inventory_hostname }}"
  provider: "{{ cli }}"

- name: disable service 'config'
  ios_command:
    commands:
      - conf t
      - no service config
  provider: "{{ cli }}"

```

roles/common-config/tasks/main.yaml

```

- name: set hostname
  ios_config:
    lines:
      - ['hostname {{inventory_hostname}}']
  provider: "{{ cli }}"

- name: disable service 'config'
  ios_config:
    lines:
      - ['no service config']
  provider: "{{ cli }}"

```

roles/saveconfig/tasks/main.yaml

```

- name: save running-config to startup-config
  ios_command:
    commands:
      - write memory
  provider: "{{ cli }}"

```

roles/trunking/tasks/main.yaml

```

- name: set interfaces 1-5 as trunk ports
  ios_command:
    commands:
      - conf t
      - int range fa0/1-5
      - switchport mode trunk
      - switchport trunk native vlan 99
      - no shutdown
  provider: "{{ cli }}"

```

```
- name: set ip trunk vlan
  ios_command:
    commands:
      - conf t
      - int vlan99
      - ip address {{ vlan99_ip }}
  provider: "{{ cli }}"
```

roles/vlan/tasks/main.yaml

```
- name: set vlans
  ios_command:
    commands:
      - conf t
      - vlan {{ item.key }}
      - name {{ item.value.name }}
  provider: "{{ cli }}"
  with_dict: "{{ vlans }}"

- name: assign interfaces to vlans
  ios_command:
    commands:
      - conf t
      - int range fa0/{{ item.key }}
      - switchport access vlan {{ item.value.vlan }}
  provider: "{{ cli }}"
  with_dict: "{{ interfaces }}"

- name: set spanning-tree
  ios_command:
    commands:
      - conf t
      - spanning-tree vlan 99 priority 4096
  provider: "{{ cli }}"
```

roles/vtp/tasks/main.yaml

```
- name: set vlans
  ios_command:
    commands:
      - conf t
      - vtp mode {{ vtp }}
      - vtp domain Lab5
      - vtp password cisco
  provider: "{{ cli }}"
```

```
_____  
[routers]  
router1  
  
[server-switches]  
switch1  
  
[client-switches]  
switch2  
  
[switches:children]  
server-switches  
client-switches
```

./inventory

```
_____  
group_vars/switches.yaml  
_____  
- hosts: server-switches  
  gather_facts: no  
  connection: local  
  
  vars:  
    limit_to: "*"  
  
  roles:  
    - common-command  
#    - common-config  
    - vtp  
    - trunking  
    - vlan  
    - saveconfig  
  
#- hosts: client-switches  
#  gather_facts: no  
#  connection: local  
#  
#  vars:  
#    limit_to: "*"  
#  
#  roles:  
#    - common-command  
##    - common-config  
#    - vtp  
#    - trunking  
#    - saveconfig
```

group_vars/switches.yaml

B Switch Configuration Files

Switch Commands

```
# all switches (except for a unique ip address)

en
conf t
ip default-gateway 172.17.1.1
int vlan1
ip address 172.17.1.2 255.255.255.0
no shut
exit
hostname switchname
ip domain-name website.be
crypto key generate rsa
2048
username switchadmin password cisco
enable secret class
no ip domain-lookup
line console 0
logging synchronous
login local
line vty 0 4
transport input ssh
login local
password cisco
exit
service password-encryption
end
write memory
reload

.

# Switch 1
en
class
conf t
vtp mode server
vtp domain Lab5
vtp password cisco
int range fa0/1-5
switchport mode trunk
switchport trunk native vlan 99
no shutdown
```

```
exit
vlan 99
name management
vlan 10
name faculty-staff
vlan 20
name students
vlan 30
name guest
exit
int vlan99
ip address 172.17.99.11 255.255.255.0
exit
int range fa0/6-10
switchport access vlan 30
int range fa0/11-17
switchport access vlan 10
int range fa0/18-23
switchport access vlan 20
exit
spanning-tree vlan 99 priority 4096
end
write memory
reload
```

```
.
```

```
# Optional Switch 2

vtp mode client
vtp domain Lab5
vtp password cisco
end
int range fa0/1-5
switchport mode trunk
switchport trunk native vlan 99
no shut
int vlan99
ip address 172.17.99.12 255.255.255.0
end
write memory
```

Initial Switchconfig

Current configuration : 1274 bytes


```
interface FastEthernet0/8
!
interface FastEthernet0/9
!
interface FastEthernet0/10
!
interface FastEthernet0/11
!
interface FastEthernet0/12
!
interface FastEthernet0/13
!
interface FastEthernet0/14
!
interface FastEthernet0/15
!
interface FastEthernet0/16
!
interface FastEthernet0/17
!
interface FastEthernet0/18
!
interface FastEthernet0/19
!
interface FastEthernet0/20
!
interface FastEthernet0/21
!
interface FastEthernet0/22
!
interface FastEthernet0/23
!
interface FastEthernet0/24
!
interface GigabitEthernet0/1
!
interface GigabitEthernet0/2
!
interface Vlan1
  no ip address
  no ip route-cache
  shutdown
!
ip http server
ip http secure-server
!
```

```
control-plane
!
!
line con 0
line vty 5 15
!
end
```

VLAN	Name	Status	Ports
1	default	active	Fa0/1, Fa0/2, Fa0/3, Fa0/4 Fa0/5, Fa0/6, Fa0/7, Fa0/8 Fa0/9, Fa0/10, Fa0/11, Fa0/12 Fa0/13, Fa0/14, Fa0/15, Fa0/16 Fa0/17, Fa0/18, Fa0/19, Fa0/20 Fa0/21, Fa0/22, Fa0/23, Fa0/24 Gi0/1, Gi0/2
1002	fdmi-default	act/unsup	
1003	token-ring-default	act/unsup	
1004	fdmnet-default	act/unsup	
1005	trnet-default	act/unsup	

Minimal Switchconfig

```
Current configuration : 3226 bytes
!
version 12.2
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname Switch
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$ad1E$eENpnWdsRwvYDTMRF1bDU1
!
username switchadmin password 7 070C285F4D06
no aaa new-model
system mtu routing 1500
ip subnet-zero
!
no ip domain-lookup
ip domain-name website.be
```

```
!
!
crypto pki trustpoint TP-self-signed-3395035520
    enrollment selfsigned
    subject-name cn=IOS-Self-Signed-Certificate-3395035520
    revocation-check none
    rsakeypair TP-self-signed-3395035520
!
!
crypto pki certificate chain TP-self-signed-3395035520
    certificate self-signed 01
        30820249 308201B2 A0030201 02020101 300D0609 2A864886 F70D0101 04050030
        31312F30 2D060355 04031326 494F532D 53656C66 2D536967 6E65642D 43657274
        69666963 6174652D 33333935 30333535 3230301E 170D3933 30333031 30303030
        35385A17 0D323030 31303130 30303030 305A3031 312F302D 06035504 03132649
        4F532D53 656C662D 5369676E 65642D43 65727469 66696361 74652D33 33393530
        33353532 3030819F 300D0609 2A864886 F70D0101 01050003 818D0030 81890281
        8100A673 DFD43E3D 3BE774FD 0CB6712E A10DE672 23FEAB25 17958BA9 ED53EF31
        16E89D4B D12CC2BF C3F01FA9 F59BB01D C357FF77 326CA8B6 EDE96337 8F9FE592
        A130BE8A 855DFE7F B6244C18 599619CC 332DCDAB 47820375 C3178B35 716936BA
        66C276C6 E64E93C3 0742C86B 5392508F 479A4BEF 77D02D10 2E651B8B 890B5AAB
        4C6F0203 010001A3 71306F30 0F060355 1D130101 FF040530 030101FF 301C0603
        551D1104 15301382 11537769 7463682E 77656273 6974652E 6265301F 0603551D
        23041830 16801477 D48AFA9A FABC2C8F ADA95327 05318386 80905C30 1D060355
        1D0E0416 041477D4 8AFA9AFA BC2C8FAD A9532705 31838680 905C300D 06092A86
        4886F70D 01010405 00038181 004087FF 7549472F 7A4F0F86 5679CCF6 E7181F92
        3ADD4015 4E9FA2E7 B01FDF33 A21201B7 9EFBBA7A 84517C3D 8344650C 5059EB07
        C3A063FD 24A56FDC BEC1ADA8 0FD8C3DE 1F06EBB2 2F102D9E 16191DFB D56291D2
        072E7607 FF83C621 1281B7D0 68B8F6DC ED15CE01 7ED31165 ACE71D37 AA5E2538
        476F348E 65D8C7CE EF0F32A6 F3
    quit
!
!
!
!
!
spanning-tree mode pvst
spanning-tree extend system-id
!
vlan internal allocation policy ascending
!
!
!
interface FastEthernet0/1
!
interface FastEthernet0/2
```

```
!
interface FastEthernet0/3
!
interface FastEthernet0/4
!
interface FastEthernet0/5
!
interface FastEthernet0/6
!
interface FastEthernet0/7
!
interface FastEthernet0/8
!
interface FastEthernet0/9
!
interface FastEthernet0/10
!
interface FastEthernet0/11
!
interface FastEthernet0/12
!
interface FastEthernet0/13
!
interface FastEthernet0/14
!
interface FastEthernet0/15
!
interface FastEthernet0/16
!
interface FastEthernet0/17
!
interface FastEthernet0/18
!
interface FastEthernet0/19
!
interface FastEthernet0/20
!
interface FastEthernet0/21
!
interface FastEthernet0/22
!
interface FastEthernet0/23
!
interface FastEthernet0/24
!
interface GigabitEthernet0/1
```

```
!
interface GigabitEthernet0/2
!
interface Vlan1
  ip address 172.17.1.2 255.255.255.0
  no ip route-cache
!
ip default-gateway 172.17.1.1
ip http server
ip http secure-server
!
control-plane
!
!
line con 0
  logging synchronous
  login local
line vty 0 4
  password 7 14141B180F0B
  login local
  transport input ssh
line vty 5 15
  login
  !
end
```

VLAN	Name	Status	Ports
1	default	active	Fa0/1, Fa0/2, Fa0/3, Fa0/4 Fa0/5, Fa0/6, Fa0/7, Fa0/8 Fa0/9, Fa0/10, Fa0/11, Fa0/12 Fa0/13, Fa0/14, Fa0/15, Fa0/16 Fa0/17, Fa0/18, Fa0/19, Fa0/20 Fa0/21, Fa0/22, Fa0/23, Fa0/24 Gi0/1, Gi0/2
1002	fdci-default	act/unsup	
1003	token-ring-default	act/unsup	
1004	fd dinet-default	act/unsup	
1005	tr net-default	act/unsup	

Complete Switchconfig

```
Current configuration : 4110 bytes
!
version 12.2
```

```
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname switch1
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$ad1E$eENpnWdsRvvYDTMRF1bDU1
!
username switchadmin password 7 070C285F4D06
no aaa new-model
system mtu routing 1500
ip subnet-zero
!
no ip domain-lookup
ip domain-name website.be
!
!
crypto pki trustpoint TP-self-signed-3395035520
enrollment selfsigned
subject-name cn=IOS-Self-Signed-Certificate-3395035520
revocation-check none
rsakeypair TP-self-signed-3395035520
!
!
crypto pki certificate chain TP-self-signed-3395035520
certificate self-signed 01
3082024A 308201B3 A0030201 02020101 300D0609 2A864886 F70D0101 04050030
31312F30 2D060355 04031326 494F532D 53656C66 2D536967 6E65642D 43657274
69666963 6174652D 33333935 30333535 3230301E 170D3933 30333031 30303030
35355A17 0D323030 31303130 30303030 305A3031 312F302D 06035504 03132649
4F532D53 656C662D 5369676E 65642D43 65727469 66696361 74652D33 33393530
33353532 3030819F 300D0609 2A864886 F70D0101 01050003 818D0030 81890281
8100A673 DFD43E3D 3BE774FD 0CB6712E A10DE672 23FEAB25 17958BA9 ED53EF31
16E89D4B D12CC2BF C3F01FA9 F59BB01D C357FF77 326CA8B6 EDE96337 8F9FE592
A130BE8A 855DFE7F B6244C18 599619CC 332DCDAB 47820375 C3178B35 716936BA
66C276C6 E64E93C3 0742C86B 5392508F 479A4BEF 77D02D10 2E651B8B 890B5AAB
4C6F0203 010001A3 72307030 0F060355 1D130101 FF040530 030101FF 301D0603
551D1104 16301482 12737769 74636831 2E776562 73697465 2E626530 1F060355
1D230418 30168014 77D48AFA 9AFABC2C 8FADA953 27053183 8680905C 301D0603
551D0E04 16041477 D48AFA9A FABC2C8F ADA95327 05318386 80905C30 0D06092A
864886F7 0D010104 05000381 81006AA2 B75C7693 7562D9CC DC1F2EA7 1C686C77
F8536E96 FB7EF881 F8CEB1C7 1E2BDFDD 4248806D 87424281 24B4940C 8FF74A21
```

```
059C9F53 AF3C49C8 C2DF7262 FC20015C 31AA9A8A B9B850CA 40700B80 323E1CCE
D1397329 14FB561C AD0445DC DD666155 4AEC3C75 6B48B2E6 85ACA015 C40D6F26
7455E576 FDC95466 CFAD97D9 F5C2
quit
!
!
!
!
!
!
spanning-tree mode pvst
spanning-tree extend system-id
spanning-tree vlan 99 priority 4096
!
vlan internal allocation policy ascending
!
!
!
interface FastEthernet0/1
switchport trunk native vlan 99
switchport mode trunk
!
interface FastEthernet0/2
switchport trunk native vlan 99
switchport mode trunk
!
interface FastEthernet0/3
switchport trunk native vlan 99
switchport mode trunk
!
interface FastEthernet0/4
switchport trunk native vlan 99
switchport mode trunk
!
interface FastEthernet0/5
switchport trunk native vlan 99
switchport mode trunk
!
interface FastEthernet0/6
switchport access vlan 30
!
interface FastEthernet0/7
switchport access vlan 30
!
interface FastEthernet0/8
switchport access vlan 30
```

```
!
interface FastEthernet0/9
switchport access vlan 30
!
interface FastEthernet0/10
switchport access vlan 30
!
interface FastEthernet0/11
switchport access vlan 10
!
interface FastEthernet0/12
switchport access vlan 10
!
interface FastEthernet0/13
switchport access vlan 10
!
interface FastEthernet0/14
switchport access vlan 10
!
interface FastEthernet0/15
switchport access vlan 10
!
interface FastEthernet0/16
switchport access vlan 10
!
interface FastEthernet0/17
switchport access vlan 10
!
interface FastEthernet0/18
switchport access vlan 20
!
interface FastEthernet0/19
switchport access vlan 20
!
interface FastEthernet0/20
switchport access vlan 20
!
interface FastEthernet0/21
switchport access vlan 20
!
interface FastEthernet0/22
switchport access vlan 20
!
interface FastEthernet0/23
switchport access vlan 20
!
```

```

interface FastEthernet0/24
!
interface GigabitEthernet0/1
!
interface GigabitEthernet0/2
!
interface Vlan1
  ip address 172.17.1.2 255.255.255.0
  no ip route-cache
!
interface Vlan99
  ip address 172.17.99.11 255.255.255.0
  no ip route-cache
!
ip default-gateway 172.17.1.1
ip http server
ip http secure-server
!
control-plane
!
!
line con 0
  logging synchronous
  login local
line vty 0 4
  password 7 14141B180F0B
  login local
  transport input ssh
line vty 5 15
  login
!
end

```

VLAN	Name	Status	Ports
1	default	active	Fa0/1, Fa0/2, Fa0/3, Fa0/4 Fa0/5, Fa0/24, Gi0/1, Gi0/2
10	faculty-staff	active	Fa0/11, Fa0/12, Fa0/13, Fa0/14 Fa0/15, Fa0/16, Fa0/17
20	students	active	Fa0/18, Fa0/19, Fa0/20, Fa0/21 Fa0/22, Fa0/23
30	guest	active	Fa0/6, Fa0/7, Fa0/8, Fa0/9 Fa0/10
99	management	active	
1002	fdci-default	act/unsup	

1003 token-ring-default	act/unsup
1004 fddinet-default	act/unsup
1005 trnet-default	act/unsup

Bibliografie

- Ansible. (2016). Ansible Changes by Release. Verkregen 21 augustus 2016, van <https://github.com/ansible/ansible/blob/stable-2.1/CHANGELOG.md>
- Barroso, D. (2016a). First Steps Manipulating Config. Verkregen 22 augustus 2016, van https://napalm.readthedocs.io/en/latest/tutorials/first_steps_config.html
- Barroso, D. (2016b). Github: napalm-ansible. Verkregen 22 augustus 2016, van <https://github.com/napalm-automation/napalm-ansible>
- Barroso, D. (2016c). Network Automation and Programmability Abstraction Layer with Multivendor support. Verkregen 19 augustus 2016, van <https://github.com/napalm-automation/napalm>
- Barroso, D. (2016d). Supported Devices. Verkregen 19 augustus 2016, van <https://napalm.readthedocs.io/en/latest/support/index.html>
- Buresh, B., Daugherty, B., Obediente, C., Roberts, E., Pfeifer, J., Garreau, K., ... Escalona, T. (2015, oktober 16). Programmability and Automation with Cisco Open NX-OS. Verkregen 19 augustus 2016, van http://www.cisco.com/c/dam/en/us/td/docs/switches/datacenter/nexus9000/sw/open_nxos/programmability/guide/Programmability_Open_NX-OS.pdf
- Cammarata, J. (2016, januari). Ansible 2.0 has arrived. Verkregen 19 augustus 2016, van <https://www.ansible.com/blog/ansible-2.0-launch>
- Cisco. (2008). Cisco NX-OS. Verkregen 19 augustus 2016, van <http://www.cisco.com/c/en/us/products/ios-nx-os-software/nx-os/index.html>
- Cisco. (2014). Resetting Catalyst Switches to Factory Defaults. Verkregen 19 augustus 2016, van <http://www.cisco.com/c/en/us/support/docs/switches/catalyst-2900-xl-series-switches/24328-156.html>
- Cisco. (2016, juli 21). Download Software. Verkregen 23 augustus 2016, van <https://software.cisco.com/download/release.html?mdfid=281231709&softwareid=>

- 280805680&os=&release=12.2.58-SE2&relind=AVAILABLE&rellifecycle=&reltype=latest&i=!pp
- DeHaan, M. (2013, december). The Origins Of Ansible. Verkregen 19 augustus 2016, van <https://www.ansible.com/blog/2013/12/08/the-origins-of-ansible>
- DeHaan, M. (2014, februari 10). TOWER 1.4.5 (FORMERLY AWX) RELEASED. Verkregen 23 augustus 2016, van <https://www.ansible.com/blog/2014/02/10/tower-1-4-5-formerly-awx-released>
- Natarajan, R. (2013, augustus 22). How to Enable SSH on Cisco Switch, Router and ASA. Verkregen 23 augustus 2016, van <http://www.thegeekstuff.com/2013/08/enable-ssh-cisco/>
- Reitz, K. (2015, november 15). Virtual Environments. Verkregen 22 augustus 2016, van <http://docs.python-guide.org/en/latest/dev/virtualenvs/>
- Slattery, T. (2007, oktober). The history of the cisco cli. Verkregen 19 augustus 2016, van <http://www.netcraftsmen.com/the-history-of-the-cisco-cli/>
- Vandooren, D. (2015). *Configuratie van Cisco apparatuur door gebruik te maken van Ansible* (Bachelorproef, Hogeschool Gent).

Lijst van figuren

3.1 De aangekochte seriële consolekabel, deze is te vinden op Ebay als een 'FTDI USB to RJ45'. Er wordt ook duidelijk vermeld dat dit specifiek voor Cisco apparaten is.	22
3.2 Links op de afbeelding de router met ip adres: 172.17.1.1 deze is aangesloten op de switch poort fa0/24. Deze behoort tot vlan1 die het ip adres 172.17.1.2 heeft. Daarboven een laptop die via usb console kabel aan de switch verbonden is en via wifi aan de router. Op deze manier was het eenvoudig de switch resetten via console kabel en eenvoudig Ansible uit te voeren over WiFi dan.	23
5.1 Bash in Windows of via de correcte term 'Bash on Ubuntu on Windows'	34