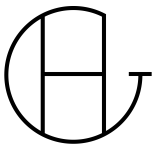


HoGent

Faculteit Bedrijf en Organisatie

Technische voor-en nadelen van Puppet en Ansible. Verloop en redenen van een omschakeling.

Thomas Detemmerman

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

Promotor:
Harm De Weirdt
Co-promotor:
Tom De Wispelaere

Instelling: VRT

Academiejaar: 2016-2017

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Technische voor-en nadelen van Puppet en Ansible. Verloop en redenen van een omschakeling.

Thomas Detemmerman

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

Promotor:
Harm De Weirdt
Co-promotor:
Tom De Wispelaere

Instelling: VRT

Academiejaar: 2016-2017

Tweede examenperiode

Samenvatting

Puppet is een vaak gebruikte tool om servers te configureren. De laatste tijd zijn er echter meer concurrenten op de markt gekomen waaronder Ansible. Ansible zou ten opzichte van Puppet verschillende voordelen bieden maar aan een overschakeling van Puppet naar Ansible gaan vele werkuren vooraf. Er wordt dus beter goed nagedacht of een dergelijke omschakeling ook effectief bestaande problemen zal oplossen.

Dit rapport is van toepassing voor bedrijven die worstelen met de vraag welke configuratietool het meest voor hen geschikt is en of een overschakeling economisch verantwoord is. Om deze vragen te beantwoorden wordt deze vergelijkende studie gevoerd zodat duidelijk wordt welke tool op een welbepaald vlak de betere is.

In eerste instantie zullen tekortkomingen en moeilijkheden beschreven worden die momenteel vookomen in de infrastructuur van Puppet. Gezien het feit dat dit onderziek zich afpeeld op de Vlaamse Radio- en Televisieomroeporganisatie (VRT) zal zullen deze moeilijkheden en tekortkomingen specifiek voor de VRT zijn waar een dergelijke transitperiode van Puppet naar Ansible zich afspeelt. Dit onderzoek is dan ook gericht naar bedrijven die zichzelf hierin herkennen. Vervolgens wordt de technische werking van Ansible en Puppet geanalyseerd en gekeken naar verschillen in hun manier van werken om hetzelfde eindresultaat te bereiken. Later wordt kort aangehaald wat mogelijke gevaren en impact kunnen zijn van slecht geconfigureerde configuration management tools en er wordt afgesloten met een rapport over hoe een dergelijke transitperiode van Puppet naar Ansible precies in zijn werk gaat. In laatste instantie wordt ook het prijskaartje van beide tools aangehaald.

Dit onderzoek loopt binnen het mediabedrijf VRT waar een dergelijke overgang plaatsvindt. De bestaande kennis en ervaringen van mensen uit de praktijk worden in dit rapport

verwerkt. Door deze kennis te combineren met testen uitgevoerd op een proof of concept, die een Puppet en Ansible infrastructuur simuleren, zijn onderstaande resultaten bekomen.

Zo blijkt dat Ansible door verschillende mensen wordt beschouwd als een technologie dat eenvoudig aan te leren en op te stellen is. Bovendien worden de servers die dienen geconfigureerd te worden minder belast vanwege de afwezigheid van een zogenaamde agent. Ansible en Puppet hebben een fundamenteel verschil in hun manier van communiceren. Zo verstuurt Ansible voortdurend kleine bestanden terwijl Puppet deze bundelt in twee grote reeksen. Puppet is complexer om op te stellen en het aanleren van de syntax vraagt meer moeite. Op elke server dient bovendien een Puppetagent geïnstalleerd te worden. Puppet blijkt in de praktijk voor sommige scenario's performanter dan Ansible. Vooral bij zwaar intensieve taken tijdens deploy's scoort Puppet beter qua performantie.

Voor bedrijven die beschikken over de nodige kennis omtrend Puppet wordt een overschakeling afgeraden. Er is geen opmerkelijke winst op welk vlak dan ook. Aan bedrijven daarentegen die niet vertrouwd zijn met geautomatiseerd configuration management wordt geadviseerd om voor Ansible te kiezen. Het is eenvoudig aan te leren en desondanks het feit dat Ansible bij een grotere hoeveelheid servers trager werkt dan Puppet, wordt hetzelfde doel bereikt. Bedrijven die zich herkennen in het profiel van de VRT kunnen de overschakeling overwegen. Aangezien beide tools naast elkaar kunnen bestaan in dezelfde omgeving is een geleidelijke overgang perfect mogelijk. Dit stelt het bedrijf in staat om alles voldoende te testen terwijl Puppetmanifesten geleidelijk naar Ansiblerollen vertaald worden.

Dit onderzoek is gevoerd binnen het team MediaIT binnen de VRT. Dit betekend dan ook dat Ansible hier slechts een beperkt deel van het volledige aantal servers zal beheren. Er wordt in dit rapport dus niet onderzocht hoe deze beide configuration management tool (CMT)'s zich zullen gedragen voor het volledige netwerk. Vermoed wordt dat Puppet beter in staat is om een grote hoeveelheid servers te configureren en dat Ansible in de moeilijkheden zou komen. Zo is Puppet in staat om de Puppetmaster op te splitsen in verschillende servers zoals PE console, Puppet DB, compile master en zo verder. Volgens hun eigen documentatie kunnen ze tot wel 20.000 servers behandelen. Ansible biedt deze mogelijkheid niet. Voorlopig is er weinig onderzoek gevoerd naar hoeveel servers dergelijke tools nu precies aankunnen.

Voorwoord

Drie jaar geleden begon ik aan de hogeschool in Gent met een duidelijke voorkeur voor informatica. Nu, drie jaar later, is diezelfde passie alleen maar groter geworden. Gedurende mijn studententijd vond ik voldoening in zaken zoals programmeren, artificiële intelligentie en netwerk- en systeembeheer. Binnen deze brede interesses, vond ik vooral het domein van systeembeheer zeer interessant.
Het werd mij al snel duidelijk dat goede configuration management tools een absolute meerwaarde konden bieden. Zo herinner ik me nog mijn eerste project waarbij de opdracht was een webserver op te zetten. Ik maakte toen gebruik van Puppet terwijl ik amper de kracht en het potentieel van dit soort technologieën begreep. Inmiddels heb ik de kans gehad om hieromtrent uitgebreide ervaring op te doen dankzij mijn docent dhr. Van Vreckem en mijn stage op de VRT. Dit heeft ertoe geleid ook mijn bachelorproef rond dit fascinerend onderwerp te voeren.

Een goede bachelorproef wordt niet alleen geschreven. Hierbij werd ik ondersteund door heel wat personen die ik zeer dankbaar ben. Bij deze wil ik dan ook de volgende mensen persoonlijk bedanken voor hun bijdrage.

dhr. De Wispelaere Tom

Bedankt voor het voorzien van uitgebreide feedback, het bedenken van oplossingen en de aanbreng van nieuwe ideeën.

dhr. De Weirdt Harm

Bedankt omdat ik bij u terecht kon voor vragen en voor uw mening over de

stand van zaken gedurende de bachelorproef.

mevr. Lambrecht Carine
Bedankt voor het verzorgen van het linguïstisch aspect van de bachelorproef.

dhr. Adams Pieter
Bedankt omdat ik bij u terecht kon voor technische vragen en de introductie van Ansible Tower.

Inhoudsopgave

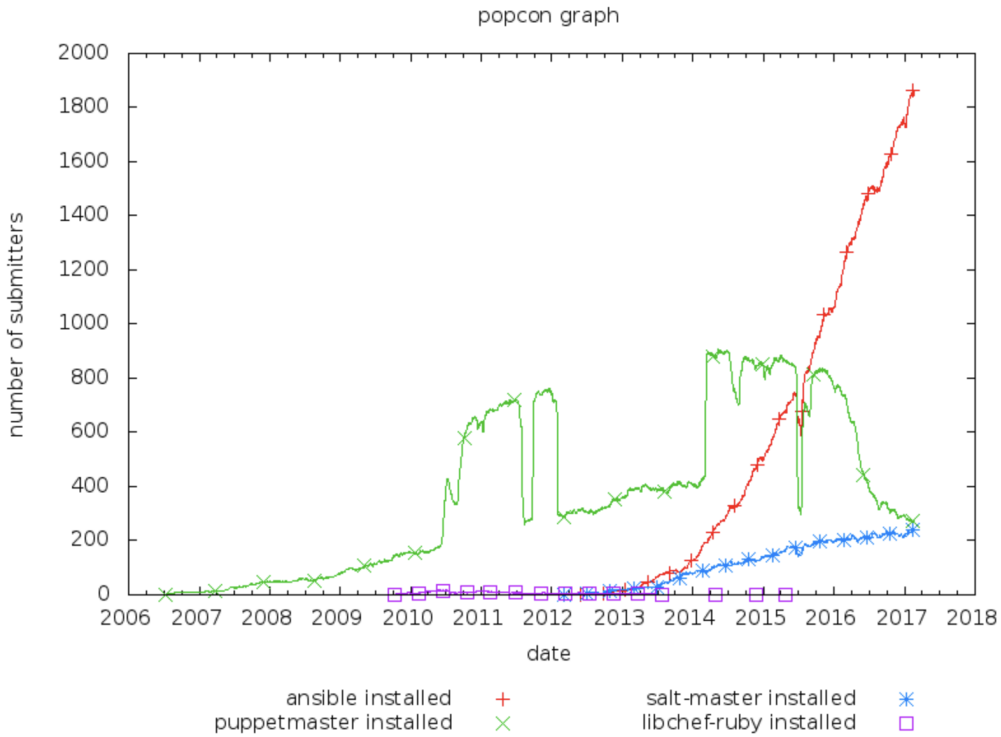
1. Inleiding

1.1 Stand van zaken

Bedrijven kunnen tegenwoordig niet meer zonder IT-infrastructuur. Deze infrastructuur kan uitgebreid en complex zijn. Bovendien moet ze ook nog schalen naarmate het bedrijf groeit. Als systeembeheerder heb je diverse taken zoals incident management, het volgen van de laatste technologische trends of maatregelen treffen tegen cyberdreigingen. Het manueel opzetten en configureren van de zoveelste identieke server is een groot tijd- en geldverlies. Daarom werden configuration management tools in het leven geroepen. De eerst bekende tool was Puppet. Deze technologie stelt ons in staat om configuraties op declaratieve wijze te programmeren (Puppet, g.d.-i). Eens de gewenste configuratie geprogrammeerd is, kunnen extra gelijkaardige servers veel sneller en foutloos opgezet worden.

Puppet is een CMT die sinds lang marktleider is. Dit is ook te zien op grafiek 1.1. Maar daar komt nu verandering in. Er is de laatste jaren meer concurrentie op de markt gekomen waaronder relatief bekenden zoals Salt en Chef.

Echter, één van deze nieuwe CMT's doet het opvallend beter op gebied van populariteit en dat is Ansible inc. Zoals op grafiek 1.1 te zien is, heeft Ansible in 2015 de leiding genomen. Het was bovendien ook in dat jaar dat Ansible werd vernoemd door multinationals waaronder Gartner, die over Ansible schreef in een artikel over 'Cool Vendors in DevOps' (Ronni J. Colville, 2015). Verder was het RedHat (2015) die aankondigde dat er een akkoord was om Ansible over te nemen. Grafiek 1.1 toont hoe vaak Ansible en Puppet gedownload zijn op een Debian distributie en voorlopig laat Ansible zijn concurrenten ver achter zich qua populariteit.



Figuur 1.1: Deze grafiek van Debian (2017) toont het aantal keer dat een bepaald software-pakket geïnstalleerd is op een Debian distributie.

1.1.1 Profiel van Puppet

Puppet is een open source project dat werd ontwikkeld in 2005 door Luke Kanies (Puppet, g.d.-f) met als doel op een betrouwbare manier datacenters te kunnen automatiseren en controleren. Dit zou het hele proces van services installeren moeten versnellen om zo tijd te winnen (Puppet, g.d.-a). Het kan zowel gaan om Linux servers als Windows servers (Puppet, g.d.-d). Om dit te kunnen verwezenlijken maakt Puppet gebruik van het server/client model. De server wordt in dit model de Puppetmaster genoemd. Dit kunnen er één of meerdere zijn. De client wordt de Puppetagent genoemd. Zowel op de master als op de agent dient Puppet geïnstalleerd te zijn om te kunnen functioneren. (Puppet, g.d.-g) (Puppet, g.d.-c)

1.1.2 Profiel van Ansible

Michael DeHaan heeft samen met Saïd Ziouani in 2012 het open source project Ansible gestart (Geerling, 2016). Michael DeHaan ondervond dat mensen moeilijkheden hadden op gebied van eenvoud en automatisatie met de bestaande technologieën. Bovendien waren er bedrijven die verschillende tools combineerden. Daarom wou Michael DeHaan een CMT bouwen dat zorgde voor een duidelijk configuratiebeheer, eenvoudig deployen van nieuwe servers en als het nodig was de mogelijkheid bood tot ad-hoc commando's. Ook Ansible werkt volgens het server/client model. Opvallend is wel dat elke computer waarop

Ansible draait in principe kan fungeren als server. In bedrijven zoals de VRT is er gekozen voor een centraal beheerpunt. Dit wordt Ansible Tower genoemd. In tegenstelling tot Puppet dient er bij Ansible geen additionele software geïnstalleerd te worden op de clients. Dit komt het principe van eenvoudig deployen ten goede. Ondanks het feit dat Ansible voornamelijk gekend is door de Linux community is het ook in staat om Windows servers te configureren. (Ansible, g.d.-d)

1.2 Opzet van deze bachelorproef

Dit onderzoek vindt plaats binnen MediaIT, een team binnen de IT-afdeling van de VRT. Zij zijn verantwoordelijk voor een goede en correcte werking van de servers. MediaIT staat momenteel samen met de rest van de VRT voor twee grote uitdagingen, namelijk de tekortkomingen van Puppet en de digitaliserende wereld.

Ten eerste is de huidige integratie van Puppet niet optimaal. Zo wordt er binnen de VRT gebruik gemaakt van multistage omgevingen zoals test, development, staging en productie. Deze manier van werken is niet met Puppet geïntegreerd wat het testen bemoeilijkt. Maar ook andere zaken spelen parten zoals de complexiteit van Puppet en de beperkte functionaliteit tot het monitorren van configuraties.

Ten tweede moet de VRT ook voortdurend vernieuwen. Zo komt er een geheel nieuw gebouw bij dat onder andere het nieuwe datacenter zal herbergen. Dit terwijl een deel van het oude datatcenter naar de cloud zal verhuizen. Maar niet enkel binnen de VRT veranderen zaken, ook het kijkgedrag van de vlaamse bevolking is veranderd. Televisie is namelijk niet langer alleenheerser en steeds meer programma's worden bekeken via sites en apps. Zo deed Imec (2017) een onderzoek naar het digitale gebruik van de belgische bevolking. Hieruit bleek dat de populariteit van de televisie als favoriet nieuwsmedium in 2016 gezakt is met 3,3% t.o.v. het jaar voordien wat resulteert in 22,4%. Dit terwijl smartphone, computer en tablet gezamelijk 29,7% halen. Het is vanzelfsprekend dat VRT deze trend moet volgen.

Het is dus van van belang dat een geschikte CMT gebruikt wordt en dat deze perfect geïntegreerd is in de bestaande én toekomstige infrastructuur. De CMT moet dus een onderscheid kunnen maken tussen de verschillende omgevingen waarin de servers zich kunnen bevinden (productie, staging,...). Hij zal optimaal moeten opereren in de toekomstige hybride infrastructuur, zal configuraties moeten monitoren, maar boven dit alles zal de CMT ook eenvoudig in gebruik moeten zijn.

1.3 Probleemstelling en Onderzoeksvragen

Ansible is sinds enige tijd aan een stevige opmars bezig maar er zijn voldoende voorbeelden van open source (en andere) projecten die na een initiele hype snel in mekaar zakten. Ondertussen heeft Ansible tal van mooie referenties achter zich waarbij deze van NASA

nog het meest tot de verbeelding spreekt. Zij gebruikten Ansible om hun datacenter te migreren naar de cloud (RedHat, 2013). Verder heeft Ansible ook verscheidene positieve analyses gekregen van belangrijke partijen zoals RedHat en Gartner. Is Ansible echter noemenswaardig beter dan Puppet die reeds een lange bewezen staat van dienst heeft (meer dan 12 jaar) en ook een grote community achter zich heeft die het project ondersteunt?

De overschakeling van Puppet naar Ansible is geen kleine stap en kan mogelijk voor veel complicaties zorgen. Daarom wil dit rapport een hulp bieden aan bedrijven die dezelfde stappen overwegen zodat het op voorhand duidelijk is wat er verwacht kan worden, wat de mogelijkheden zijn en waar een CMT te kort schiet. Dit zal onderzocht worden door middel van de volgende drie grote categorieën.

1.3.1 Wat zijn de redenen van een omschakeling?

Het is van belang te weten wat de drijfveren waren voor de beslissing om Puppet te vervangen door Ansible en dat is precies waar het in deze eerste categorie om draait. Om een profiel van de situatie op te kunnen stellen zal een interview plaatsvinden met de verantwoordelijken binnen de VRT om zo te achterhalen waar Puppet te kort schoot en waarom men denkt dat Ansible hier een oplossing biedt. Als bedrijven hun situatie herkennen in dit profiel, is het geadviseerd om te overwegen of een overstap ook voor hen al dan niet aan te raden is.

1.3.2 Wat zijn de technische voor-en nadelen van Puppet en Ansible?

In deze tweede categorie zal er een vergelijkende studie plaatsvinden waarbij technische aspecten zoals performantie, schaalbaarheid en veiligheid vergeleken worden.

Ten eerste wordt de performantie onderzocht. Hieronder wordt verstaan de tijd die nodig is tot het bekomen van een consistente toestand en deze zal onderzocht worden in twee situaties. Bij de eerste is er namelijk nog geen configuratie aanwezig en dient alles nog geïnstalleerd en geconfigureerd te worden. Bij de tweede situatie is er wel al een configuratie aanwezig en is het de bedoeling dat de CMT enkel de nodige aanpassingen doorvoert en niet alles opnieuw configureert.

Ten tweede is er de schaalbaarheid. Onder schaalbaarheid wordt verstaan: het vermogen om grote vraag te verwerken zonder kwaliteit te verliezen (Informit, 2002). We zullen monitoren hoe Ansible en Puppet hun resources verdelen bij een toenemende drukte, hier onder de vorm van meer servers en uitgebreidere configuraties.

Er wordt afgesloten met een analyse over de veiligheid. Hierbij zal er een literatuurstudie plaatsvinden met onderzoek naar welke veiligheidsproblemen reeds gekend zijn en wat de impact hiervan is op een bedrijfsnetwerk. CMT's hebben administrator rechten tot verschillende servers die ze dienen te configureren. Wanneer de server waarop een CMT draait besmet is, kunnen de gevolgen catastrofaal zijn.

1.3.3 Wat is het verloop van een dergelijke transitperiode?

Problemen die bij de vervanging van Puppet door Ansible optreden, zullen gerapporteerd worden en er zal onderzocht worden waarom deze optraden. Al dan niet gevonden oplossingen zullen beschreven en uitgelegd worden zodat andere bedrijven zich goed bewust zijn van wat er hen te wachten staat en hoe ze eventueel sommige voorvallen best kunnen oplossen. Welke incidenten zich zullen voordoen, valt uiteraard moeilijk te voorspellen.

2. Methodologie

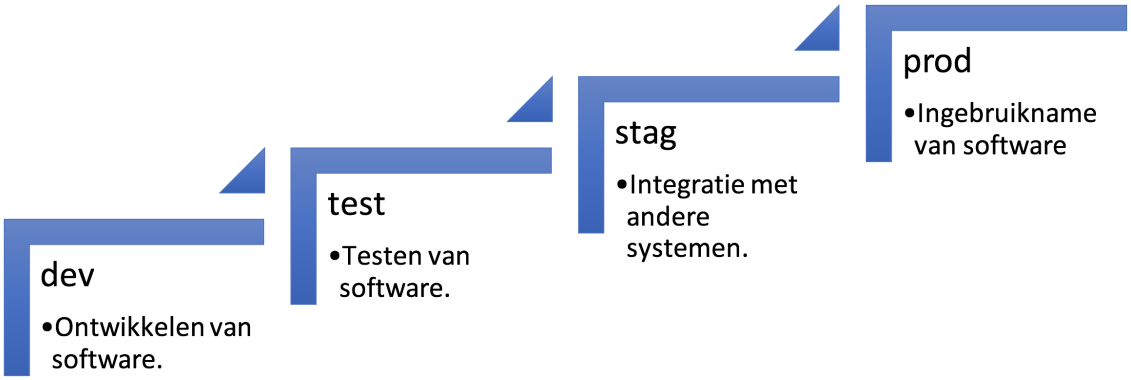
2.1 Redenen van een omschakeling?

Puppet is een CMT met vele mogelijkheden maar heeft als nadeel zijn complexe leercurve. Veel mensen waaronder ook Tehranian (2015) en zijn collega's, Loschwitz (2016) en Raza (2016) delen deze mening. Dit was ook het probleem bij de VRT. Door de complexiteit waren er maar een beperkt aantal mensen die het volledige potentieel van Puppet konden benutten. Binnen VRT was voornamelijk één persoon beslagen in Puppet en bovendien was kennisoverdracht naar andere mensen een pijnpunt vanwege drukke agenda's. Dit maakte hem tot een signle point of failure binnen de organistatie.

Door het gebrek aan een goed ingebouwde monitoringstool kost het veel werk om te controleren welke servers correct geconfigureerd zijn. Idealiter zou er 's ochtends gekeken worden welke servers groen en rood kleuren. Voorlopig gebeurt dit via de command line interface.

Verder maakt de VRT, zoals vele bedrijven, gebruik van meerdere omgevingen. Zo bestaat er van veel servers een versie in development, staging en productie zoals ook te zien is op afbeelding 2.1. Op die manier kunnen nieuwe configuraties eerst in de development omgeving uitgetest worden alvorens deze in productie te brengen. Puppet was op de VRT niet geoptimaliseerd om het verschil te kennen tussen deze omgevingen. Dit heeft tot gevolg dat Puppet nieuwe configuraties pusht naar elke server, ongeacht of deze productie-kritisch is of niet. Dit is vanzelfsprekend niet de bedoeling en om dit te voorkomen, wordt er momenteel handmatig bepaald welke servers mogen updaten en niet.

Puppet biedt voor veel van deze problemen een oplossing (denk maar aan Puppet dash-board). Doch, mochten veel van deze oplossingen geïntegreerd worden, dan zou dit leiden



Figuur 2.1: Ontwikkelingsproces van software.

tot een nieuwe refactor van de infrastructuur. Dit, terwijl een refactor in het verleden al tot twee maal toe gebeurd is. Daarom is er besloten om over te schakelen naar Ansible om een zo groot mogelijk draagvlak te creëren. Hoe meer mensen in staat zijn om met Ansible te werken; hoe beter deze CMT geïntegreerd en gebruikt zal worden. Iets wat bij Puppet niet mogelijk is vanwege de complexiteit.

Ansible biedt met Ansible Tower een geïntegreerde monitoringstool. Bovendien wordt deze CMT door velen geprezen vanwege zijn eenvoud in syntax. Ook de opstelling van de infrastructuur is eenvoudiger. Zo heeft Ansible geen agent nodig om servers te configureren wat een enorm voordeel biedt. Er kan dus zonder iets te moeten installeren onmiddellijk overgegaan worden tot het configureren van clients. Ansible Tower kan met vele externe technologieën overweg wat de mogelijkheden tot continuous deployment vereenvoudigt. Zo is bijvoorbeeld git ingebouwd wat zorgt voor een eenvoudig beheer van de source code.

2.2 Werking van Ansible en Puppet

2.2.1 Overzicht van Puppet en Ansible

In onderstaande tabel kunnen de belangrijkste technische eigenschappen teruggevonden worden. Zo zijn zowel Puppet als Ansible declaratief wat inhoud dat de eindtoestand beschreven wordt en niet hoe deze bereikt moet worden. Een voorbeeld van een declaratief zin is: "Het bestand ‘foo’ moet aanwezig zijn". Merk op dat deze zin niet beschrijft hoe dat bestand aangemaakt moet worden.
Ansible wordt geschreven in YAML wat een eenvoudig leesbare syntax. Ansible zal deze gegevens vertalen naar een uitvoerbaar script dat geschreven is in Python. Puppet is ontwikkeld in Ruby en maakt gebruik van zijn eigen domain specific language (DSL).
Verder ligt de verantwoordelijkheid voor het aanvragen van configuraties bij Puppet bij de clients zelf. Dit wordt het Pull-model genoemd. Je ‘trekt’ als het ware de laatste configuratie los van de master. Hiervoor moet op de master in staat zijn om HTTPS-berichten op poort 8140 te ontvangen. Bij Ansible ligt deze verantwoordelijkheid bij de master zelf en daarom werkt deze volgens het Push-model. De master ‘duwt’ configuraties naar de clients. Dit gebeurt bij Ansible bovendien via SSH naar de standaard poort 22.

	Ansible	**Puppet**
Programmeerparadigma	declaratief	declaratief
Geschreven in	YAML	DSL
Gebasseerd op	Python	Ruby
Communicatieprotocol	SSH	HTTPS
Push / Pull principe [a]	push	pull
open poorten[b]	22/tcp (client)	8140/tcp (master)

[a]Beide technologieën zijn in staat om zowel volgens push als pull methode te functioneren. Zo heeft Ansible 'pull-mode' (Korokithakis, 2013) en Puppet (g.d.-b) 'Puppet kick'

[b]Dit zijn de minimale vereisten van open poorten. Voor sommige features dienen meer poorten open te staan. Bijvoorbeeld 443 voor Ansible Tower of 8140 op elke Puppetclient voor de Puppet kick functionaliteit (Puppet, g.d.-b)

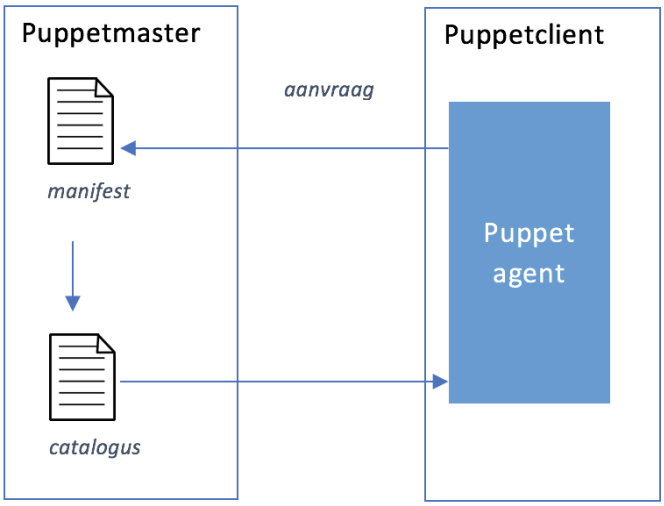
Informatie opgehaald van Puppet (g.d.-j), Weirdt (2014), Ansible (g.d.-b)

2.2.2 Werking van Puppet

Tussen de master en de client bestaat er een vertrouwensrelatie die onderhouden wordt door certificaten. Het is de Puppetmaster die verantwoordelijk is voor het verlenen van deze certificaten. Pas als deze in orde zijn kan Puppet aan de configuraties van de clients beginnen. De verzameling van alle geschreven code wordt een manifest genoemd. Wanneer een Puppetagent wil controleren of hij nog up-to-date is, zal hij een catalogus aanvragen bij de Puppetmaster. Een dergelijke catalogus is in feite een manifest dat de Puppetmaster compileert. Deze catalogus is bovendien uniek voor elke Puppetagent. Dit komt omdat er bij het compileren van het manifest naar de catalogus rekening gehouden wordt met diverse parameters zoals de functie van de server of de distributie van het besturingssysteem dat op die server draait (Puppet, g.d.-e). Eens de Puppetagent zijn persoonlijke catalogus ontvangen heeft, zal deze voor zichzelf controleren of er verschillen zijn tussen zijn huidige configuratie en de staat die beschreven staat in de catalogus. Indien er afwijkingen zijn, worden deze ook automatisch opgelost (Puppet, g.d.-g).

2.2.3 Werking van Ansible

In tegenstelling tot Puppet maakt Ansible geen gebruik van zogenaamde agents. Dit betekent dat de Ansibleserver enkel de naam en het wachtwoord dient te kennen van de servers die hij moet configureren. Het authenticeren kan op verschillende manieren. Er wordt aangeraden om gebruik te maken van een SSH-key, wat het eenvoudigst is, maar ook andere middelen zoals een eenvoudig wachtwoord of het Kerberos-protocol worden ondersteund.
De gewenste configuraties worden geschreven in playbooks met bijhorende Ansiblerollen. Eens een verbinding tot stand is gebracht, wordt het gecompileerde playbook naar de te configureren server gestruurd. Deze worden vervolgens op de clients uitgevoerd en weer verwijderd. Ook Ansible bezit de functionaliteit om na te gaan of de huidige configuratie in lijn is met de ontvangen modules.
Om servers te configureren met Ansible bestaan er bovendien twee manieren. Ansible



Figuur 2.2: Aanvraag van een catalogus bij de Puppetmaster door een Puppetclient. De Puppetagent is een daemon (stukje software) die op de Puppetclient draait.

playbooks kunnen in principe verstuurd worden naar de servers vanaf elke computer. Voor een grotere hoeveelheid servers is dit echter moeilijk onderhoudbaar en onoverzichtelijk. Hiervoor bestaat er de commerciële versie waarbij de playbooks verstuurd worden vanaf een centraal punt. Dit centraal punt wordt Ansible Tower genoemd; die heeft een inventaris van alle servers en playbooks die onder zijn verantwoordelijkheid vallen. Verschillende gebruikers kunnen vervolgens verschillende toegangsrechten krijgen zodat personen enkel servers kunnen configureren die onder hun bevoegdheid vallen (Ansible, g.d.-b).

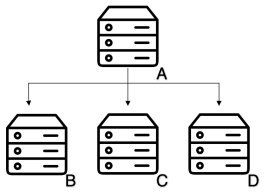
2.3 Technische analyse van Ansible en Puppet

Het luik technische analyse kan opgedeelt worden in een inleiding gevolgd door drie essentiële onderdelen. De inleiding bespreekt de opstelling van de testomgeving zodat deze eenvoudig na te bouwen is voor derden.
Hierna komt de eerste categorie aan bod die de belasting van het netwerk bespreekt en hoe beide CMT's zich gedragen op het netwerk.
Vervolgens komt de tweede categorie ter sprake betreffende de tijd dat het kost tot het bekomen van een consistente toestand. Dit wordt getest voor de volgende drie situaties: een lege server, een gedeeltelijk geconfigureerde server en een volledig geconfigureerde server.
In laatste instantie wordt het geheugengebruik behandeld waarbij gekeken wordt welke CMT het meeste werkgeheugen gebruikt en waarom.

2.3.1 Opstelling van de testomgeving

Architectuur van de opstelling

Deze opstelling is gerealiseerd door gebruik te maken van Vagrant (g.d.). Dit is een tool die het mogelijk maakt om op eenvoudige wijze meerdere virtuele omgevingen op te zetten, te configureren en te beheren. Voor elke omgeving bestaat er een Vagrantbestand. Er is dus een Vagrantbestand voor de Ansibleinfrastructuur en één voor de Puppetinfrastructuur. Voor elk van deze omgevingen wordt eerst de master aangemaakt, gevolgd door een X-aantal clients. Vagrant zorgt ervoor dat elke client verbonden wordt met New Relic. Dit is de gekozen tool om de servers te monitoren. In het geval van Puppet wordt hierbij ook nog de Puppetagent geïnstalleerd. Elke client, zowel in de Ansible- als de Puppetinfrastructuur, is gebaseerd op dezelfde basebox en krijgt dezelfde resources toegekend. De masters beschikten over meer resources.



Figuur 2.3: Server A is de master en configureert clients B tot D.

Technische specificaties clients
Basebox: centos/7
Aantal CPU's: 1
Geheugen: 500 MB

Modus operandi

Elke server wordt opgezet door middel van Vagrant. De situatie vlak nadat Vagrant de servers opgezet heeft, wordt hierna de 'beginstand' genoemd. Vanuit deze situatie nemen de CMT's het over. Afhankelijk van de test worden de resultaten afgelezen van ofwel de CMT ofwel de monitoringstool. Elke test wordt 30 keer uitgevoerd. Telkens de test opnieuw wordt uitgevoerd, wordt deze eerst terug naar de beginstand gebracht[1]. Deze gegevens worden vervolgens afgelezen en in een tabel genoteerd. Deze ruwe data wordt achteraf opgeschoont[2] en kan teruggevonden worden in bijlage C, datasets. Achteraf is het op deze data dat de analyses en mathematische berekingen zijn gebeurd. De conclusies kunnen teruggevonden worden in onderstaande secties.

Configuratie van servers door CMT's

Voor de testen in sectie 2.3 is er gekozen om gebruik te maken van een LAMP-stack. Dit is zowel voor Ansible als Puppet gebeurd. Bovendien werd er getracht om in de mate van het mogelijke beide configuraties zo analoog mogelijk te houden. Om dit te realiseren wordt er httpd, php, php-mysql en mariaDB geïnstalleerd op de servers. Vervolgens wordt er een webpagina op de site geplaatst. Deze is geschreven in PHP en controleert of alle zaken naar behoren werken. Om af te sluiten wordt HTTP doorgelaten door de firewall, de service httpd gestart en MariaDB gestopt.

[1] Weliswaar is deze beginstand voor de testen van 'gedeeltelijke configuratie' en 'geen configuratie' verschillend.

[2] Zo worden gegevens die geen onderdeel meer uitmaakten van de test verwijderd of werd er gezorgd dat elke dataset even lang is. Belangrijk is te weten dat deze aanpassingen enkel gebeurd zijn waar mogelijk en dat deze geen invloed hebben op berekeningen en conclusies.

Opmerking: De reden dat MariaDB op dit moment nog niet gestart wordt is vanwege de testen bij 'gedeeltelijke configuratie'. Hierbij wordt gemeten hoelang het duurt om bij een bestaande configuratie enkele aanpassingen door te voeren. Het is dus pas bij deze testen dat MariaDB gestart wordt.

Deze configuraties kunnen teruggevonden worden op Github op de volgende links[3]:
Ansible: https://github.com/ThomasDetemmerman/AnsibleTower
Puppet: https://github.com/ThomasDetemmerman/PuppetMaster

2.3.2 Belasting van het netwerk

Ansible en Puppet hebben een groot verschil in de manier van communiceren en dit weerspiegelt zich in het gedrag van de CMT. De grafieken op afbeeldingen 2.4 en 2.5 weerspiegelen uitsluitend het dataverkeer tussen de server (Ansible Tower of Puppetmaster) en de desbetreffende client. Andere data zoals bijvoorbeeld het downloaden van services of uploaden van logbestanden naar de monitoringstool zijn hierin niet opgenomen. Dit wordt verwezenlijkt door gebruik te maken van verschillende netwerkkaarten. Wanneer er geen deploy gebeurt, is de kilobyte/ minuut op deze netwerkkaart gelijk aan nul; een bewijs dat hier geen andere data dan deze van de CMT over wordt verstuurd.

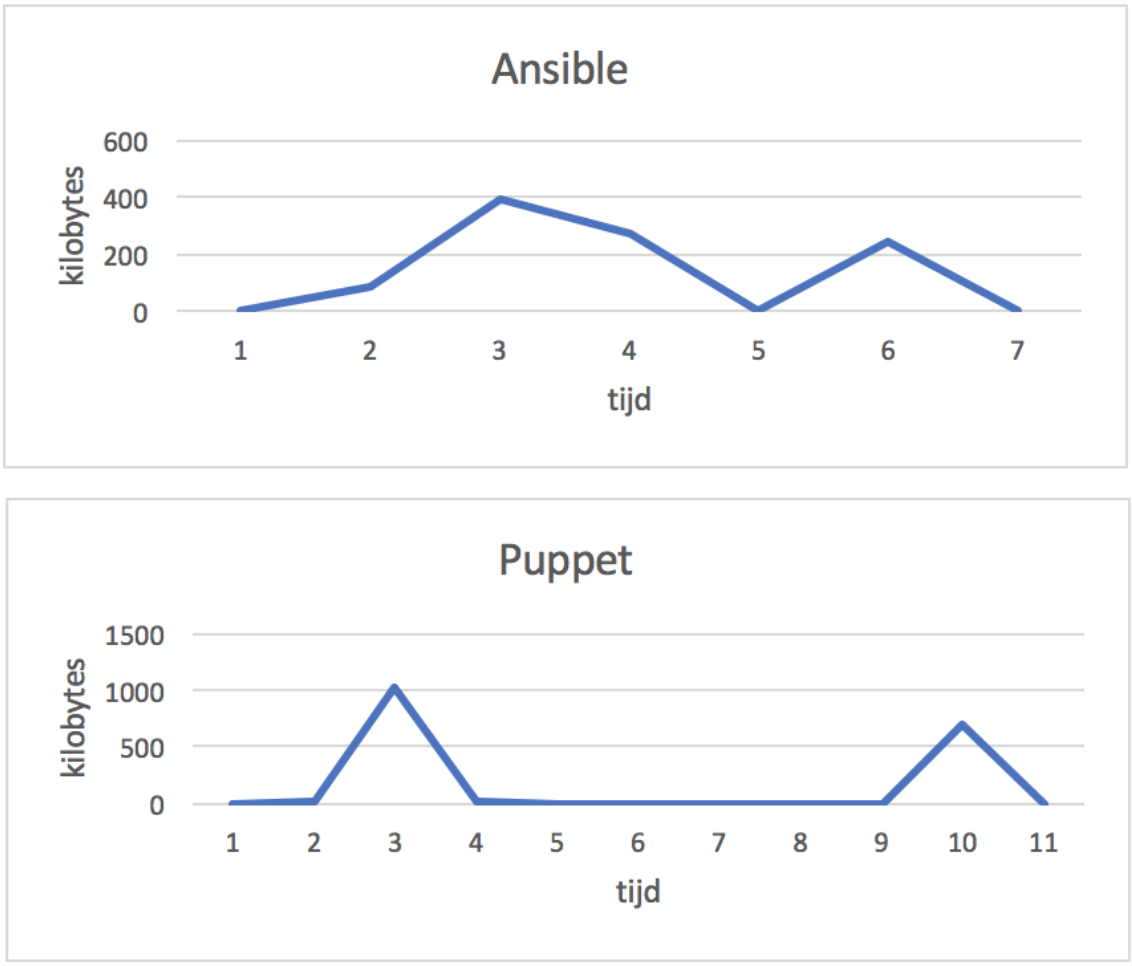
De manier van communicatie is te herkennen in de grafieken. Zo onderhoudt Ansible de communicatie met de client gedurende de deploy. Hiermee wordt bedoeld dat Ansible op de hoogte is van de laatste stand van zaken op de client. Wanneer een bepaalde taak voltooid is, wordt Ansibel Tower hier onmiddellijk van op de hoogte gebracht. Zoals te zien is op grafiek 2.4 is er geregeld communicatie tussen beide servers. Weliswaar is er enkel communicatie wanneer iets voltooid is; er is dus geen onnodige communicatie. Zo is ook te zien hoe op T5 de communicatie nul is. Ansible had voor die periode niets te melden[4].

Deze manier van werken is handig tijdens het testen van nieuwe Ansible rollen. Je krijgt namelijk live feedback tijdens het uittesten. Een nadeel is wel dat het netwerk geen rust krijgt. Bovendien wordt deze functionaliteit van 'live feedback' enkel gebruikt voor testen. Eens een rol operationeel is, loopt deze doorgaans tijdens de nacht en is het voldoende om de dag daarna een algemeen overzicht te krijgen.

Bij Puppet is dit anders. Hierbij is er enkel communicatie tussen de master en de client bij het begin en op einde van de deploy. Tijdens de eerste reeks vraagt de client een catalogus op bij de master. Vervolgens worden eventuele plugin's gesynchroniseerd en vraagt de master facts op bij de client. Op basis van deze gegevens wordt uiteindelijk de verwachte catalogus verstuurd. Tijdens het configureren is er geen communicatie tussen beide servers. Op het einde wordt een rapport in JSON-formaat terug naar de master gekoppeld (DarylW, 2017).

[3]Mogelijk komt de beschreven configuratie in dit rapport niet volledig overeen met de code op Github. Deze repositories worden namelijk ook gebruikt voor andere testen binnen dit onderzoek.

[4]Hier betreft het de service MariaDB die wordt gedownload.

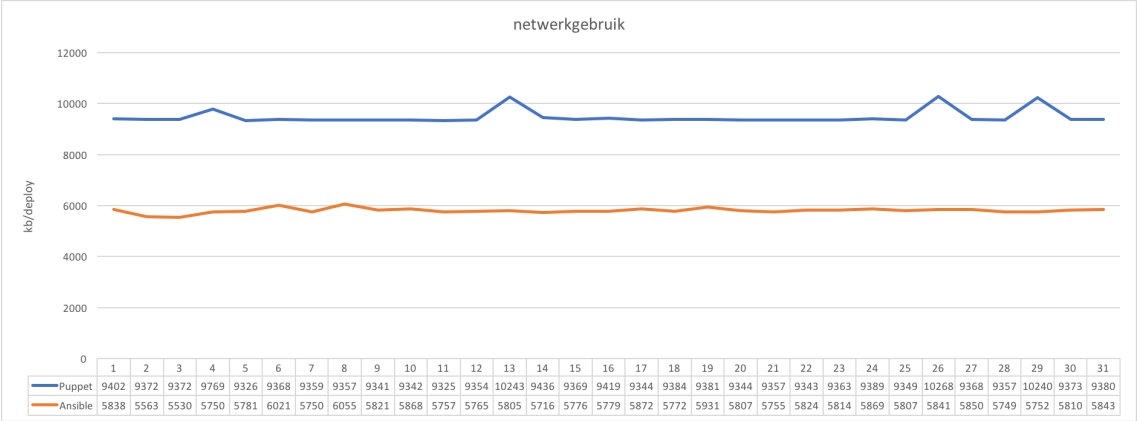


Figuur 2.4: Twee manieren van communiceren. Aantal kilobytes per tijdseenheid op een netwerkkaart die uitsluitend bedoeld is voor communicatie met Ansible Tower / Puppetmaster.

Een nadeel aan het feit dat er enkel in het begin en op het einde communicatie is, is dat er op de master geen live feedback gevolgd kan worden. Dit kan echter wel opgelost worden door in te loggen op de desbetreffende client en hier de live feedback te volgen met behulp van het commando "`puppet agent -t`". Het wordt wel nog steeds pas op het einde van de deploy terug naar de master gestuurd.

Vervolgens is er ook gekeken naar de totale netwerkbelasting. Hiervoor is er per client een cumulatie genomen van de kilobytes/minuut gedurende de gehele deploy. Deze waarden zijn terug te vinden in grafiek 2.5. Hier heeft Ansible een gemiddelde van 5802,29 kilobytes/deploy. Dit terwijl Puppet gemiddeld 9464,32 kilobytes/deploy haalt.

We kunnen dus concluderen dat Ansible minder grote bestanden verstuurt over het netwerk door deze op te splitsen in meerdere kleine bestanden verspreid over de hele deploy. Hierbij is wel een voortdurende verbinding vereist tussen beide servers. Puppet aan de andere kant weet de communicatie te beperken tot twee reeksen. Op het einde van de rit heeft Puppet wel meer bestanden uitgewisseld dan Ansible.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Puppet	9402	9372	9372	9769	9326	9368	9359	9357	9341	9342	9325	9354	10243	9436	9369	9419	9344	9384	9381	9344	9357	9343	9363	9389	9349	10268	9368	9357	10240	9373	9380
Ansible	5838	5563	5530	5750	5781	6021	5750	6055	5821	5868	5757	5765	5805	5716	5776	5779	5872	5772	5931	5807	5755	5824	5814	5869	5807	5841	5850	5749	5752	5810	5843

Figuur 2.5: Totaal verbruikte netwerkcapaciteit per client gedurende het deployen. Dit bevat enkel communicatie tussen master en client.

2.3.3 Tijd tot het bekomen van een consistente toestand

Het is interessant om te weten wat de verhouding is tussen de gemiddelde configuratietijd van Ansible en Puppet. Om dit op een zo betrouwbaar mogelijke manier te kunnen verwezenlijken, zijn de configuraties van Ansible en Puppet zo analoog mogelijk gehouden zoals te lezen is in sectie 2.3.1. De tijd kan worden onderverdeeld in twee delen.
Het eerste deel van de tijd zal de connectietijd genoemd worden. Dit is de tijd die het kost alvorens er effectief overgegaan kan worden tot configureren. Hieronder vallen zaken zoals het verzamelen van de nodige configuraties en het verzamelen van server-specifieke waarden (zoals bijvoorbeeld distributie) en het compileren van een catalogus of module. Bij Ansible kon deze tijd gewoon berekend worden op basis van de resultaten[5] dewelke af te lezen zijn van het dashboard dat geïntegreerd is met Ansible Tower. Bij Puppet is dit echter niet mogelijk en bijgevolg zijn deze resultaten met de hand gemeten.
De tweede tijd is de configuratietijd. Dit is de tijd die nodig is om de configuratie effectief uit te voeren. Deze tijden zijn eenvoudig af te lezen op de uitvoer van de CMT's en het volstaat om deze te noteren.

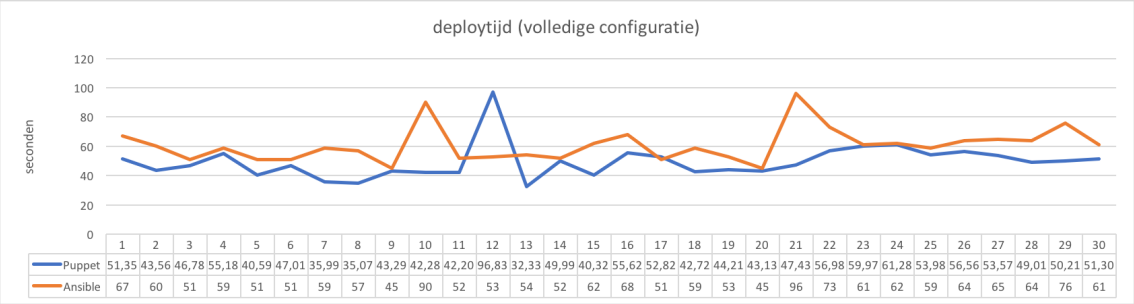
configuratietijd

Op afbeelding 2.6 is geen éénduidig verschil aan te tonen tussen beide CMT's. Bijgevolg is er met behulp van de Z-toets aangetoond of er al dan niet een statistisch verschil is. Deze berekingen kunnen teruggevonden worden in bijlage A: hypothese configuratietijd.

Hierbij blijkt dat Z buiten het kritisch gebied valt waardoor de nulhypotese verworpen kan worden. Er kan dus worden geconcludeerd dat wanneer er wordt vertrokken van een lege server, Ansible er gemiddeld langer over doet dan Puppet. Dit is in dit geval een verschil van gemiddeld 11,3 seconden.

De verschillen worden echter nog groter wanneer de test gedaan wordt met een gedeeltelijke configuratie. Hiermee wordt bedoeld dat er vertrokken is van servers die reeds

[5] connectietijd = totaal verstreken tijd - $\sum$ (tijd playbooks)



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Puppet	51,35	43,56	46,78	55,18	40,59	47,01	35,99	35,07	43,29	42,28	42,20	96,83	32,33	49,99	40,32	55,62	52,82	42,72	44,21	43,13	47,43	56,98	59,97	61,28	53,98	56,56	53,57	49,01	50,21	51,30
Ansible	67	60	51	59	51	51	59	57	45	90	52	53	54	52	62	68	51	59	53	45	96	73	61	62	59	64	65	64	76	61

Figuur 2.6: Tijd tot het bekomen van een consistente toestand, vertrekkende van een ‘lege’ server.

geconfigureerd zijn en er slechts enkele aanpassingen doorgevoerd moeten worden. Deze aanpassingen zijn een service starten en de inhoud van de webpagina veranderen. De resultaten lopen niet door elkaar waardoor een Z-toets niet echt nodig is. Ansible deed gemiddeld 19,10 seconden voor de deploy met een vrij grote variatie van 33,40 seconden. Puppet haalde maar een vrij consistente 3,10 seconden met een variatie van 0,35.

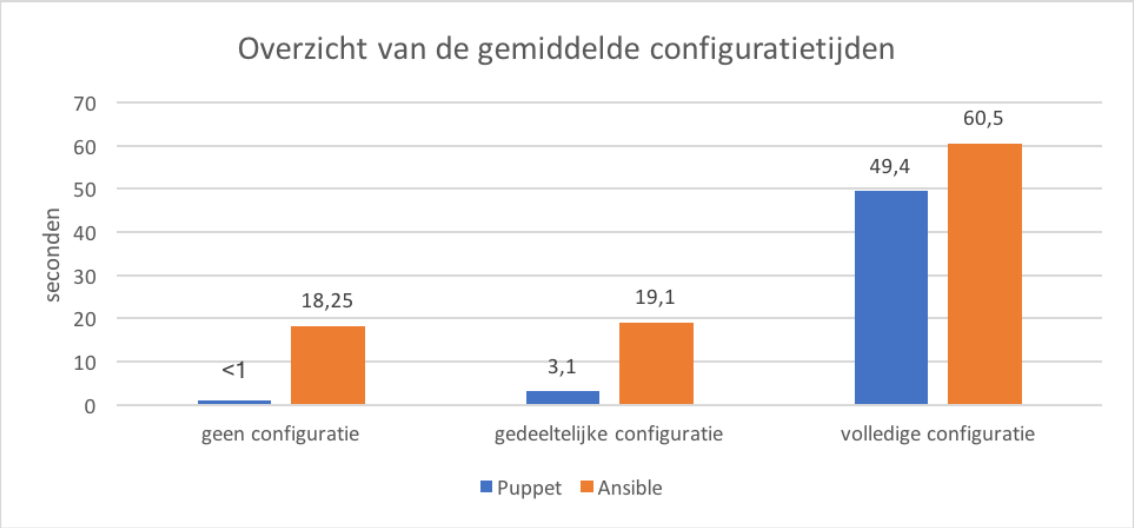
In laatste instantie is er gekeken naar de tijd die het zou kosten tot de CMT vaststelt dat de server reeds volledig geconfigureerd is en dat er geen aanpassingen meer nodig zijn. Hierbij resulteert Ansible op een gemiddelde van 18 seconden met opnieuw een vrij grote variatie van 18,25 seconden. Ook hier doet Puppet het opnieuw beter aangezien deze minder dan een seconde nodig heeft om vast te stellen dat er geen aanpassingen nodig zijn. Uiteraard zijn al deze waarden afhankelijk van de configuratie maar ze geven wel een duidelijke indicatie van de verschillen tussen beide CMT’s.

Ansible heeft vrij inconsistente deploytijden. Vermoedelijk is dit te wijten aan het netwerk tussen de master en de client. Ansible moet voor elk werkpakket een nieuwe verbinding opzetten. Puppet hiertegen verstuurd alle bestanden op hetzelfde moment en vervolgens kan de client alle nodige informatie van zijn eigen harde schijf lezen, ongeacht een al dan niet betrouwbare verbinding tussen de master en de client. Het inlezen van bestanden op een harde schrijf is namelijk nog steeds betrouwbaarder dan communiceren over het netwerk.

Opmerking: De connectietijd is niet meegerekend in deze metingen. Het omvat hier uitsluitend de configuratietijd.

De reden dat Ansible trager is dan Puppet is te wijten aan de manier van communiceren. Puppet verstuurt namelijk een volledige configuratie bij de start. Ansible doet dit niet en verstuurt taak per taak. Hierdoor wordt de verbinding meerdere malen geinitialiseerd en opnieuw verbroken. Deze veronderstelling werd gestaafd door de volgende test:

> Hypothese A: Ansible verstuurt taak per taak.
> Hypothese P: Puppet verstuurt een gehele configuratie op het moment van initialisatie.
>
> Test: de verbinding wordt verbroken tijdens het configureren.
>
> Verwachting A: nadat de verbinding verbroken is, wordt geen nieuw taak gestart.



Figuur 2.7: Gemiddelde configuratietijd in seconden.

Verwachting P: de configuratie gaat zonder problemen verder.

Resultaat A: de configuratie valt stil. Opmerkelijk hierbij is dat niettemin de configuratie stilvalt, deze niet stopt. De master heeft namelijk niet door dat de verbinding verbroken is en veronderstelt dat de client eenvoudigweg nog bezig is met configureren. Als later de verbinding hersteld wordt, gaat de configuratie opnieuw verder. Met andere woorden, als de connectie onderbroken wordt zal het huidige werkpakket voltooid worden maar er wordt geen nieuwe meer gestart. Wanneer de verbinding hersteld is gaat de configuratie verder waar deze zojuist geëindigd was en wordt de eindtoestand alsnog bereikt.
Resultaat P: de configuratie gaat ongestoord verder.

Bij Ansible wordt er dus per taak, en dus ook per connectie, een module overgebracht van de master naar de client. Deze bestanden zijn terug te vinden in `~/.ansible/tmp`. Hierin is duidelijk te zien hoe, tijdens een configuratie, er per taak een nieuwe module (geschreven in Python) verschijnt en vervolgens weer verdwijnt. Dit wordt in dit onderzoek gezien als één van de grootste oorzaken waarom Ansible trager is. Ansible (g.d.-a) herkent dit probleem en biedt hiervoor een alternatief aan, genaamd 'pipelinging'. Hierdoor zouden er minder SSH-verbindingen nodig zijn die modules moeten overzetten. Om dit mogelijk te maken moet `requiretty` uitgezet worden. Meer tips om de performantie te van Ansible te verbeteren kunnen gevonden worden in sectie 2.3.5.

Connectietijd

De connectietijd is de periode tussen het moment waarop de opdracht om te configureren gegeven wordt en het moment waarbij er effectief overgegaan kan worden tot configureren. Hieronder vallen zaken zoals het verzamelen van bepaalde gegevens, compileren en het overbrengen van de nodige bestanden en zo verder. Dit fenomeen wordt in dit rapport de connectietijd genoemd. Bij de resultaten van de connectietijd is geen duidelijk verschil op

te merken. Bovendien liggen de gemiddelden zeer dicht bij elkaar. Om vast te stellen of er een significant verschil is, is er opnieuw gebruik gemaakt van de Z-toets. Deze berekingen kunnen teruggevonden worden in bijlage A, hypothese connectietijd. Z ligt hier echter in het aanvaardingsgebied waardoor de nulhypothese, die stelt dat beide gemiddelden gelijk zijn, aanvaard kunnen worden. Er is bijgevolg geen statistisch verschil tussen beide reeksen.

Er kan hier dus geconcludeerd worden dat Ansible nood heeft aan een voortdurend goede verbinding tussen de master en de client om een snelle configuratie te kunnen verzekeren. Bij Puppet is dit niet nodig aangezien hierbij alle nodige bestanden voor een goede configuratie van meet af aan al aanwezig zijn. Vermoed wordt dat dit één van de redenen is waarom Puppet sneller is in configureren.

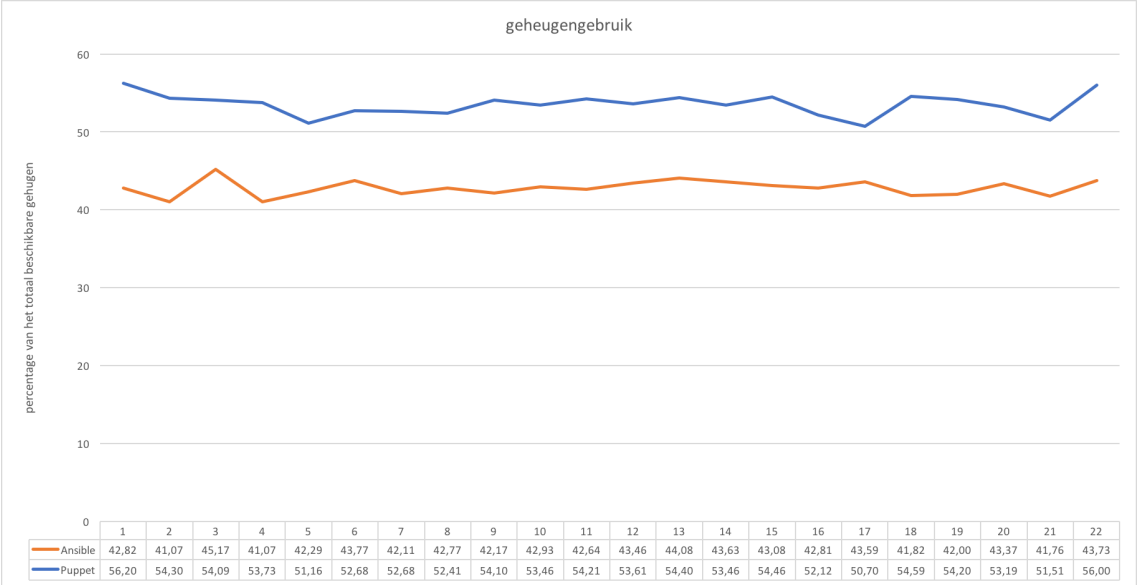
2.3.4 Gebruik van het geheugen

Op grafiek 2.8 is per tijdseenheid het gemiddeld gebruikt werkgeheugen weergegeven. Hierop is te zien hoe Puppet opvallend meer geheugen gebruikt. Niet alleen tijdens een deploy maar ook ervoor en erna. Zelfs wanneer een Ansible client en een Puppet client juist opgezet worden met behulp van Vagrant, is er al een verschil in het gebruikte geheugen. Gezien het feit dat er al een verschil waar te nemen is in deze vroege levensfase van de server en het enige verschil in configuratie op dit moment de Puppetagent is, werd vermoed dat het verschil hier aan te wijten is. Dit vermoeden werd gestaafd toen de Puppetagent tijdelijk uitgezet werd. Het werkgeheugen daalde onmiddellijk naar gelijkwaardige waarden als deze van de Ansibleclient. Zonder configuratie gebruiken Puppetclients gemiddeld 58% geheugen van de 500MB werkgeheugen. Bij Ansible is dit 47%. Dit betekent dat met een verschil van 11%, er bij 500 MB 55MB meer RAM-geheugen gebruikt wordt bij Puppet.

Er is dus een duidelijk verschil tussen het geheugengebruik van Puppet clients ten opzichte van Ansible clients. Hierbij gebruikt Puppet dankzij de Puppetagent beduidend meer resources. Ondanks het feit dat een service meer of minder doorgaans niet zorgt voor radicale gevolgen, is de grootste bottleneck nog steeds dat deze service in de eerste plaats geïnstalleerd en geconfigureerd dient te worden. Iets dat niet door Puppet verzorgd kan worden. Bij virtuele servers kan dit opgelost worden door gebruik te maken van een template.

2.3.5 Schaalbaarheid

Puppet kan minder servers in parallel behandelen dan Ansible. Dit is ook niet nodig bij Puppet. Eens de Puppetmaster de catalogus gecompileerd heeft, neemt de Puppetagent die draait op de client het over. Dit is een beduidend kortere doorlooptijd dan Ansible. Aangezien Ansible geen gebruik maakt van deze agents valt niet enkel het compileren onder de verantwoordelijkheid van de master maar ook de uitvoering hiervan. Dit betekent dat Ansible zich over zijn clients moet ontfermen zolang deze hun gewenste consistente toestand niet bereikt hebben.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Ansible	42,82	41,07	45,17	41,07	42,29	43,77	42,11	42,77	42,17	42,93	42,64	43,46	44,08	43,63	43,08	42,81	43,59	41,82	42,00	43,37	41,76	43,73
Puppet	56,20	54,30	54,09	53,73	51,16	52,68	52,68	52,41	54,10	53,46	54,21	53,61	54,40	53,46	54,46	52,12	50,70	54,59	54,20	53,19	51,51	56,00

Figuur 2.8: Verbruikt percentage van het RAM geheugen. Gemeten bij servers met elk 500 MB.

Puppet heeft buiten de kortere doorlooptijd ook nog het principe van pull-mode dat in zijn voordeel werkt. Aanvragen voor catalogussen gebeuren meer verspreid aangezien elke Puppetclient zelf verantwoordelijk is voor zijn updates. De kans dat veel servers tegelijk een aanvraag doen is vrij klein. Bij Ansible worden vanwege de push-mode wel meerdere servers tegelijk behandeld. Het is dus vanzelfsprekend dat Ansible in staat moet zijn om meerdere configuraties in parallel te behandelen.

Tips om de performantie van Ansible te verbeteren

Standaard is Ansible geconfigureerd om 5 servers tegelijk te kunnen configureren. Dit aantal kan toenemen door de parameter `forks` te verhogen naar 100 of zelfs meer. Dit zal inhouden dat Ansible 100 servers op hetzelfde moment kan configureren. Om dit te realiseren is Ansible afhankelijk van het werkgeheugen. Hoe meer werkgeheugen er beschikbaar is, hoe hoger de parameter `forks` geïnitialiseerd kunnen worden.

Ansible adviseert verder ook om gebruik te maken van `with_items` bij het installeren van meerdere packages. Door het gebruik van `with_items` zal Ansible deze packages combineren in één transactieblok wat de performantie ten goede komt. Zo zal listing 2.1 twee werkpakketten versturen terwijl dit eenvoudig tot één werkpakket had beperkt kunnen blijven door deze te structuren zoals listing 2.2.

Listing 2.1: Installeer de vereiste onderdelen van PHP in twee delen

```
1
2 - name: installeer PHP
3   package:
4     name: php
5     state: present
6
```

```
7  - name: installeer php-mysqli
8    package:
9      name: php-mysqli
10     state: present
```

Listing 2.2: Installeer de vereiste onderdelen van PHP in één keer

```
1  - name: installeer PHP en bijhorende extensies
2    package:
3      name: {{ item }}
4      state: present
5    with_items:
6      - php
7      - php-mysqli
```

Verder beschikt Ansible over een 'Pull-mode' waarbij elke server zelf instaat voor zijn configuratie. Elke server haalt de code op van een centraal punt en configureert vervolgens zichzelf. Deze manier van werken vereist wel een scriptje op elke server en doet denken aan de werking van Puppet. Een centraal management punt is in deze opstelling echter niet aanwezig en is dus af te raden voor grotere infrastructuren (Ansible, g.d.-a) .

Tips om de performantie van Puppet te verbeteren

Puppet is door zijn manier van werken 'van nature' meer geloadbalanced dan Ansible. Toch zijn ook hier een paar zaken die de performantie ten goede komen. Het aantal aanvragen dat Puppet tegelijk kan behandelen is afhankelijk van het aantal cores in de processor dat Puppet ter beschikking krijgt. Wanneer er meer aanvragen zijn dan beschikbare cores zal het overschot aan aanvragen geblokkeerd worden tot er een core weer vrijkomt. Het aantal cores dat Puppet mag gebruiken kan handmatig ingesteld worden met behulp van de parameter `max active instances`. Wanneer deze naar bijvoorbeeld twee gebracht wordt, zullen twee cores van de processor gebruikt worden, wat tot gevolg heeft dat er ook twee aanvragen tegelijk behandeld kunnen worden.

Een tweede punt is het verhogen van de `max heap size` van Java Virtual Machine (JVM). Door dit te verhogen kan het proces meer geheugen opvragen bij het besturingssysteem (Puppet, g.d.-h).

Ondanks het feit dat beide tools mogelijkheden bieden om de performantie te verhogen, blijft Puppet beter geloadbalanced door zijn manier van werken. Ansible zal dus sneller nood hebben aan een upgrade van de hardware dan Puppet. Een voordeel bij Ansible is wel dat het hoofdzakelijk het werkgeheugen is dat een upgrade nodig zal hebben. Dit komt goedkoper uit dan in het geval van Puppet waarbij een CPU met meerdere cores vereist is.

2.4 Veiligheid

Servers kunnen op allerlei manieren corrupt worden. Malafide personen kunnen binnendringen en bestanden aanpassen met als doel bijvoorbeeld een firewall uitschakelen. Weliswaar hoeft het niet altijd zo drastisch te verlopen. Soms worden servers in een inconsistente toestant gebracht door intern personeel dat goedbedoeld updates wilt doorvoeren met onvoorziene nadelige consequenties tot gevolg. Een CMT kan hier een oplossing bieden. Aangezien een goed geconfigureerde CMT op regelmatige basis de nodige servers controleert, kunnen snel mogelijke gevaren vastgesteld en opgelost worden. Stel bijvoorbeeld dat een verkeerde gebruikersgroep toegang heeft gekregen tot een belangrijke productieserver door een menselijke fout. Wanneer eventjes later de CMT de client zijn configuratie nakijkt, zal deze vaststellen dat de gebruikersgroepen niet meer in orde zijn en zal deze aanpassen [6].

Normaal komen veiligheidsproblemen pas heel laat aan het licht en meestal zelfs pas als het al te laat is. Dankzij een CMT kan dit in belangrijke mate gereduceerd worden. Belangrijk hierbij is dat de rapporten die de CMT genereert op regelmatige basis gecontroleerd worden. Wanneer een belangrijke configuratie om een verkeerde manier aangepast is geweest, moet misschien de oorzaak van deze wijziging onderzocht worden om dit in de toekomst te kunnen voorkomen.

Een goede beveiliging van de Puppetmaster of Ansible Tower is van het allergrootst belang. Dit omdat deze server toegang heeft tot verschillende andere servers. Wanneer een ‘gewone’ server in verkeerde handen valt, kan dit vanzelfsprekend tot verregaande gevolgen leiden maar de impact op andere servers blijft beperkt. Wanneer dit echter de Ansible- of Puppetmaster betreft, is de impact veel groter. Deze master heeft niet enkel toegang tot vele andere servers maar doorgaans ook belangrijke rechten om de nodige zaken te kunnen configureren.

Bij grotere bedrijven worden Ansible rollen en Puppet modules doorgaans bewaard op daarvoor bestemde servers bedoeld voor source control. Ook voor deze server in het van belang dat personen enkel toegang krijgen tot specifieke rollen en modules. Zo moet de databankbeheerder enkel toegang krijgen tot code die zijn databanken configureren terwijl dat voor andere personen weer andere rollen en modules zullen zijn. Het is dus van belang dat de correcte rechten worden toegepast op deze source control servers.

Zowel Ansible als Puppet schenken de nodige aandacht om zo veilig mogelijk te werk te gaan. Beiden gebruiken encryptie in hun vorm van communiceren. Puppet maakt hierbij gebruik van HTTPS terwijl dit bij Ansible via SSH geregeld wordt. Verder moet bij Puppet elke server handmatig geautoriseerd worden alvorens deze configuraties bij de master kan aanvragen. Bij Ansible is dit niet nodig. Hij bepaalt zelf welke servers hij wenst te configureren. Weliswaar dient hij toegang te krijgen tot elk van deze servers. Het is dus niet mogelijk om met een ongeauthoriseerde server configuraties te ontfutselen bij de master en de client op deze wijze door te laten gaan als legitiem.

[6] Het kan weliswaar gebeuren dat deze aanpassingen terecht zijn en dat de configuratie van de CMT verouderd is. Daarom zal de VRT Ansible playbooks laten draaien in check-modus. Hiebij controleert hij de server maar voert geen aanpassingen door.

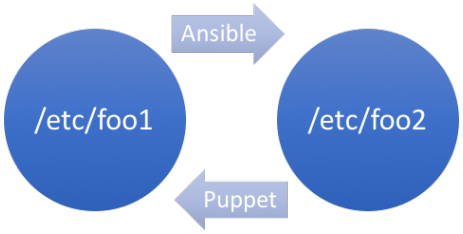
Of Ansible veiliger is dan Puppet is moeilijk te bewijzen. Er zijn wel enkele aanwijzingen die stellen dat Ansible een voordeel heeft op gebied van veiligheid. Zo dient bij Puppet poort 8140 voortdurend open te staan om aanvragen te behandelen. Bij Ansible worden deze configuraties gestart door de master zelf. Hierbij moet wel poort 22 (SSH) open staan op elke client, maar dit is doorgaans sowieso al van toepassing. Bovendien heeft Ansible geen daemon zoals Puppet die op rootlevel geïnstalleerd moet worden.

2.5 Wat is het verloop van een transitperiode?

In deze sectie wordt beschreven hoe de VRT deze transitperiode ervaren heeft. Allereerst wordt beschreven hoe een geleidelijke overgang perfect mogelijk is maar op welke punten gelet moet worden. Vervolgens wordt beschreven hoe de VRT ansible geïntegreerd heeft in zijn bestaande Puppet omgeving.

2.5.1 Een geleidelijke overgang

De manier waarop Ansible het best geïntegreerd kan worden verschilt van bedrijf tot bedrijf. In het geval van de VRT verloopt dit vlot. **Ansible en Puppet kunnen namelijk perfect naast elkaar in dezelfde infrastructuur bestaan.** Dit is ideaal voor een geleidelijke en gefaseerde overgang die bij de VRT de voorkeur geniet boven een ‘big bang’ scenario. Hierdoor worden risico’s ingeperkt en past het beter bij een cultuur gebasseerd op ‘agile’ principes waarbij getracht wordt veranderingen te spreiden in kleinere interatieve stappen.



Figuur 2.9: Vicieuze cirkel van twee CMT’s die hetzelfde bestand proberen te configureren.

Eén en dezelfde server kan bovendien geconfigureerd worden door zowel Puppet als Ansible. Dit heeft als voordeel dat niet alle modules geschreven in Puppet onmiddellijk vertaald hoeven te worden naar Ansible. Enkele rollen kunnen geschreven zijn voor Ansible terwijl een ander deel nog onder de bevoegdheid van Puppet valt.

Belangrijk hierbij is wel dat Puppet en Ansible verschillende configuraties dienen te behandelen. Wanneer beide CMT’s eenzelfde configuratie uitvoeren, kan er door subtiele verschillen een vicieuze cirkel ontstaan. Veronderstel een situatie zoals in figuur 2.9. In Ansible staat een bestand met een extra spatie, hier wordt deze voor de gelegenheid ’foo1’ genoemd. Bij Puppet staat deze extra spatie er niet, deze wordt ’foo2’ genoemd.

1. Puppet configureert de service met bestand foo1 en start deze op.
2. Eventjes later stelt Ansible vast dat de configuratie niet meer overeenkomt met deze beschreven in het playbook. Hij wijzigt foo1 naar foo2 en herstart de service zodat aanpassingen doorgevoerd zouden worden.

3. Op een later ogenblik zal Puppet dit waarnemen en het bestand terugdraaien naar foo1, opnieuw gevolgd door een gestart van de gerelateerde service.

Het is vanzelfsprekend dat dit voor onnodige aanpassingen zorgt met eventueel ongewenste of onverwachte gevolgen zoals het voortdurend herstarten van een service.

In de testopstelling is gekeken geweest naar het gedrag van beide CMT's wanneer deze op hetzelfde moment dezelfde server trachten te configureren. Dit veroorzaakte geen problemen. Het enigste wat dit als nadelig gevolg kan hebben, is dat de configuratie langer duurt dan normaal. Wanneer CMT A een lock heeft op de package manager, dan zal CMT B moeten wachten tot deze vrij komt.

2.6 Prijskaartje

Ansible heeft een ander revenue model dan Puppet. Voor standaard enterprise support rekent Ansible (2017) 9.092,15 euro voor 100 nodes. Puppet (2017) berekent de prijst op het aantal gebruikte nodes. Dit is dan 109,12 euro per node. Bij 100 nodes komt Puppet dus duurder uit. Zijn er echter minder als 100 nodes, dan is Ansible duurder want er wordt voor exact 100 nodes betaald. Bij meerdere nodes dient een offerte aangevraagd te worden. Ook voor de premium editie dient voor beide technologieën een offerte aangevraagd te worden. [7]

[7] De prijzen omgerekend van USD naar EUR tegen de wisselkoers van 5/5/2017

3. Conclusie

De syntax van Puppet wordt ten opzichte van die van Ansible door velen aanzien als complex. Dit is bovendien niet enkel het geval voor het schrijven van modules, ook het opstellen van een infrastructuur vergt kennis en ervaring. Elke server die met Puppet dient te communiceren vereist een Puppetagent geïnstalleerd te hebben dewelke bovendien zodanig geconfigureerd is dat deze de Puppetmaster kan bereiken. Vervolgens moeten certificaten uitgewisseld worden. De master verstuurt namelijk alle bestanden in één keer en vervolgens neemt de agent het van de master over. Een voordeel van deze werkwijze is dat de Puppetmaster minder zwaar belast wordt doordat de Puppetagents zelf een deel van de verantwoordelijkheid dragen. Hierdoor kan Puppet beter overweg bij een groeiende infrastructuur. Puppet is duidelijk ook sneller. Het verschil in overhead is vooral opmerkelijk wanneer er relatief weinig aangepast dient te worden.

Ansible daarentegen wordt geprezen voor de eenvoudige syntax en infrastructuur. Ook het feit dat Ansible geen agent gebruikt is één van zijn paradepaardjes. Hierdoor valt de volledige verantwoordelijkheid onder de bevoegdheid van de master. De resources van de clients worden hierdoor gespaard maar dit kan wel nadelige gevolgen hebben voor de performantie van de master bij grotere infrastructuren. Gezien het feit dat de master (bij Ansible) zich voortdurend moet ontfermen over de clients, worden meerdere kleinere bestanden verstuurd naar de clients in tegenstelling tot Puppet die alles in één keer doorstuurt. Ondanks een voortdurende belasting van het netwerk, heeft Ansible op het einde van de rit het netwerk minder belast. Bovendien is er door deze manier van werken een live feedback mogelijk die te volgen is vanop de master. Bij Puppet moet hiervoor ingelogd worden op de desbetreffende client.

Aan bedrijven die beschikken over de nodige kennis omtrent Puppet, wordt geadviseerd om bij deze infrastructuur te blijven. Een overschakeling kost geld en werk en bovendien

is er geen winst op gebied van performantie en schaalbaarheid.

Bedrijven die geen ervaring hebben met dit soort technologieën wordt aangeraden om voor Ansible te kiezen. Ansible is eenvoudig op te stellen en aan te leren. De essentie van dit soort tools is nog steeds het configureren van servers. De eenvoud in syntax moet dan ook gezien worden als belangrijkste factor. Andere aspecten zoals configuratietijden zijn interressant maar blijven bijzaak.

Ook voor bedrijven die zichzelf herkenen in de situatie van de VRT, zoals beschreven is in sectie 2.1, wordt deze stap zeker aangeraden. Dit soort overschakelingen kan geleidelijk verlopen aangezien beide tools kunnen bestaan in dezelfde omgeving (zie 2.5). Bovendien is de overstap relatief eenvoudig zonder veel complexiteit. Het rendement van de overschakeling weegt dus zeker op ten opzichte van het werk dat vereist is voor een overschakelingsproject.

	Puppet	Ansible
Infrastructuur opstellen		✓
Leercurve		✓
Configuratietijd	✓	
Schaalbaarheid	✓	
Belasting netwerk		✓
Onafhankelijk van een goede verbinding	✓	
Geheugengebruik		✓

Naar de toekomst toe wordt voorspeld dat Ansible verder zal toenemen in populariteit en zal bijgevolg deze van Puppet geleidelijk aan afnemen. Dankzij de eenvoud van Ansible wordt een groot draagvlak gecreëerd en wordt het project ondersteund door een grote gemeenschap. Eenvoud primeert namelijk in de meeste gevallen boven performantie. Denk maar aan Assembler. Dankzij deze programmeertaal kunnen programma's performant geschreven worden maar toch moest deze inboeten voor nieuwere eenvoudigere programmeertalen zoals bijvoorbeeld C. '

Lijst van acroniemen

CMT configuration management tool. 4, 11–14, 18, 20–22, 24, 25, 29–32, 41

DSL domain specific language. 18

JVM Java Virtual Machine. 29

VRT Vlaamse Radio- en Televisieomroeporganisatie. 3–5, 13, 14, 17, 30, 31, 34

Verklarende Woordenlijst

Ad-hoc commando Een ad-hoc commando is een taak die snel uitgevoerd moet worden maar die niet opgeslagen wordt voor later gebruik (Ansible, g.d.-c). Het is een eenmalig commando dat geen deel uitmaakt van een groter geheel zoals een playbook. 12

Catalogus Eng: Catalog. Een catalogus is een term uit de Puppetwereld en is in feite een gecompileerde module. Deze bevat de gewenste configuratie voor een specifieke computer. (Puppet, g.d.-g). 22, 28

Configuratietijd De tijd die de configuration management tool nodig heeft tot het bekomen van een volledig geconfigureerde server. 23–25

Connectietijd De tijd die het kost alvorens er effectief overgegaan kan worden tot configureren. Hieronder zitten zaken zoals het opstellen van een verbinding, het verzamelen van de nodige gegevens en het versturen van een gepersonaliseerde configuratie. 23, 25, 26

Deploy Een configuratie uitvoeren. In dit rapport is een vaak gebruikt synoniem configureren. Bijvoorbeeld: Puppet voert een deploy uit op server A. Dit betekend dat Puppet server A configureerd. 22, 23, 25, 27

Fact Ansible en Puppet maken gebruik van facts. Dit zijn gegevens die de master nodig heeft van zijn clients. Dit zijn zaken zoals hostname, besturingssysteem, IP adres etc. 22

Fork Het aanmaken van een child process door zichzelf te dupliceren (Linux, 2017). 28

Gedeeltelijke configuratie Er is reeds een configuratie aanwezig op de server maar deze is niet meer up-to-date. Bijgevolg moet er een deel opnieuw geconfigureerd worden. 21

Package manager Een mechanisme dat het mogelijk maakt om software te installeren op UNIX gebaseerde systemen *(voorbeelden: yum, apt, dpkg,..)*. 32

Plugin Extra functionaliteiten worden toegevoegd door middel van een plugin. In het geval van Puppet zijn dit functionaliteiten die geschreven worden in Ruby. 22

Programmeerparadigma Synoniemen zijn ook programmeerstijl of programmeermodel. Voorbeelden zijn object-georienteerd, procedureel, imperatief... (Stanchev, Green Jr, & Dimitrov, 2003). 18

Pull Een manier van communiceren waarbij de actie gestart wordt vanuit de clients, de ontvangers (Techopedia, g.d.). 18

Push Een manier van communiceren waarbij de actie gestart wordt vanuit een centraal punt, de zender (Techopedia, g.d.). 18

Werkpakket Een term binnen dit onderzoek. Ansible vertaalt een configuratie geschreven in YAML naar meerdere Pythonscriptjes. Elk van deze scriptjes wordt pas verstuurt wanneer zijn voorhanger voltooid is. Zo één scriptje wordt hier een werkpakket genoemd. 25, 26, 28

Bibliografie

Ansible. (g.d.-a). *Ansible Performance Tuning*. Verkregen van https://www.ansible.com/blog/ansible-performance-tuning

Ansible. (g.d.-b). *How Ansible works*. Verkregen van https://www.ansible.com/how-ansible-works

Ansible. (g.d.-c). *Introduction To Ad-Hoc Commands*. Verkregen van http://docs.ansible.com/ansible/intro_adhoc.html

Ansible. (g.d.-d). *Windows Support*. Verkregen van http://docs.ansible.com/ansible/intro_windows.html

Ansible. (2017, mei). *Pricing*. Verkregen van https://www.ansible.com/pricing

DarylW. (2017, april). *Network usage*. Verkregen van https://ask.puppet.com/question/30445/network-usage/

Debian. (2017). *Debian Popularity Contest*. Verkregen van http://popcon.debian.org

Geerling, J. (2016). *Ansible For DevOps*. Leanpub.

Imec. (2017). digimeter 2016.

Informit. (2002, mei). *Scalable and High- Performance Web Applications.* Verkregen van http://www.informit.com/articles/article.aspx?p=26942seqNum=18

Korokithakis, S. (2013). *Automated, large-scale deployments with Ansible's pull-mode*. Verkregen van https://www.stavros.io/posts/automated-large-scale-deployments-ansibles-pull-mo/

Linux. (2017, Maart). *Linux Programmer's Manual*. Verkregen van http://man7.org/linux/man-pages/man2/fork.2.html

Loschwitz, M. (2016). *Ansible as an alternative to the Puppet configuration tool*. Verkregen van http://www.admin-magazine.com/Archive/2016/31/Ansible-as-an-alternative-to-the-Puppet-configuration-tool

Puppet. (g.d.-a). Verkregen van https://puppet.com/product/how-puppet-works

Puppet. (g.d.-b). Verkregen van https://docs.puppet.com/puppet/3.6/man/kick.html

Puppet. (g.d.-c). *FAQ*. Verkregen van https://puppet.com/product/faq

Puppet. (g.d.-d). *Installing Puppet Enterprise for a Windows environment*. Verkregen van https://docs.puppet.com/pe/latest/windows_installing.html

Puppet. (g.d.-e). *Language: Basics*. Verkregen van https://docs.puppet.com/puppet/4.9/lang_summary.html

Puppet. (g.d.-f). *Leadership*. Verkregen van https://puppet.com/company/leadership

Puppet. (g.d.-g). *Puppet Documentation*. Verkregen van docs.puppet.com

Puppet. (g.d.-h). *Puppet Server: Tuning Guide*. Verkregen van https://docs.puppet.com/puppetserver/latest/tuning_guide.html

Puppet. (g.d.-i). *Puppet's Declarative Language: Modeling Instead of Scripting*. Verkregen van https://puppet.com/blog/puppet?s-declarative-language-modeling-instead-of-scripting

Puppet. (g.d.-j). *Why Puppet has its own configuration language*. Verkregen van https://puppet.com/blog/why-puppet-has-its-own-configuration-language

Puppet. (2017, mei). *Pricing*. Verkregen van https://puppet.com/product/pricing

Raza, A. (2016, september). *Puppet vs. Chef vs. Ansible vs. SaltStack*. Verkregen van http://www.intigua.com/blog/puppet-vs.-chef-vs.-ansible-vs.-saltstack

RedHat. (2013). *NASA: Increasing Cloud Efficiency With Ansible And Ansible Tower*.

RedHat. (2015, oktober). Verkregen van https://www.redhat.com/en/about/press-releases/red-hat-acquire-it-automation-and-devops-leader-ansible

Ronni J. Colville, L. L. (2015, april). Verkregen van https://www.gartner.com/doc/3034319/cool-vendors-devops

Stanchev, P., Green Jr, D., & Dimitrov, B. (2003). High level color similarity retrieval.

Techopedia. (g.d.). *Push Technology*. Verkregen van https://www.techopedia.com/definition/5732/push-technology

Tehranian, D. (2015, januari). *Ansible vs Puppet – An Overview of the Solutions*. Verkregen van https://dantehranian.wordpress.com/2015/01/20/ansible-vs-puppet-overview/

Vagrant. (g.d.). *Introduction to Vagrant*. Verkregen van https://www.vagrantup.com/intro/index.html

Weirdt, H. D. (2014). *Configuratieafhankelijkheden gebruiken om gedistribueerde applicaties efficiënt te beheren* (masterscriptie, KU Leuven).

Lijst van figuren

Bijlage A: Mathematische berekeningen

Hypothese configuratietijd (volledige configuratie)

Gebruikte dataset in bijlage C: deploytijden (volledige configuratie)

Hypothese

$$H_0 : \mu_p = \mu_a$$
$$H_a : \mu_p \neq \mu_a$$

Significantieniveau en waarden

α = 0.05 => -1.96 en +1.96

	Ansible	Puppet
$\bar{x}$	60.7	49.4
σ	11.5	11.6
n	30	30

Toetsingsgrootheden

$$\begin{aligned} Z &= \frac{\bar{x}_p - \bar{x}_a}{\sqrt{\frac{\sigma_p^2}{n_p} + \frac{\sigma_a^2}{n_a}}} \\ &= \frac{49{,}4 - 60{,}7}{\sqrt{\frac{11{,}6^2}{30} + \frac{11{,}5^2}{30}}} \\ &= -3{,}789 \end{aligned} \tag{3.1}$$

Hypothese connectietijd

Gebruikte dataset in bijlage C: deploytijden (connectietijd)

Hypothese

$$H_0 : \mu_p = \mu_a$$
$$H_a : \mu_p \neq \mu_a$$

Significantieniveau en waarden

α = 0.05 => -1.96 en +1.96

	Puppet	Ansible
$\bar{x}$	6,27	6,57
σ	4,10	6,11
n	30	30

Toetsingsgrootheden

$$\begin{aligned} Z &= \frac{\bar{x}_p - \bar{x}_a}{\sqrt{\frac{\sigma_p^2}{n_p} + \frac{\sigma_a^2}{n_a}}} \\ &= \frac{6{,}27 - 6{,}57}{\sqrt{\frac{4{,}10^2}{30} + \frac{6{,}11^2}{30}}} \\ &= 1{,}27 \end{aligned} \tag{3.2}$$

Bijlage B: Bachelorproefvoorstel

Technische voor-en nadelen van Puppet en Ansible. Verloop en redenen van een omschakeling

Onderzoeksvoorstel Bachelorproef

Thomas Detemmerman[1], Tom De Wispelaere[2]

Samenvatting
Om servers eenvoudig te beheren worden configuration management tools gebruikt. Dit zijn technologieën die het mogelijk maken om servers te configureren tot een specifiek gewenste consistente staat. Een vaste speler hierin was Puppet maar de laatste vier jaar evolueerde een nieuwe technologie, Ansible genaamd, tot een ware concurrent. Bedrijven onderzoeken of de overstap van Puppet naar Ansible een meerwaarde zou kunnen bieden voor hun firma en velen wagen die omschakeling dan ook. Dit onderzoek zal nagaan waar deze trend vandaan komt. Hiervoor zullen de drijfveren voor een omschakeling achterhaald worden en zal er bericht worden over problemen die zich kunnen voordoen tijdens de overstap. Verder worden technische verschillen zoals performantie, veiligheid, schaalbaarheid, tijd en efficiëntie van resourcegebruik onderzocht. Dit artikel probeert een hulp te bieden voor ondernemingen die de overstap in overweging nemen.

Sleutelwoorden
Systeem- en netwerkbeheer. Ansible — Puppet — configuration management — automatiseren

Contact: [1] thomas.detemmerman.v3945@student.hogent.be; [2] tom.dewispelaere@vrt.be;

Inhoudsopgave

1. Introductie

Ansible is een relatief jong configuration management tool [1] opgericht door Michael DeHaan, iemand die zeer vertrouwd was met Puppet (Geerling, 2016). Hij vond dat bedrijven die Puppet gebruikten moeilijkheden ondervonden op gebied van eenvoud en automatisatie. Daarom is hij samen met Saïd Ziouani Ansible Inc. gestart. Het bedrijf kon zich de laatste vier jaar een weg banen naar de top. In 2015 werd Ansible vernoemd in een artikel van Gartner getiteld 'Cool Vendors in DevOps'. Het was ook dat jaar dat Red Hat aankondigde dat er een akkoord was om Ansible over te nemen. Ansible doet het dus zeker niet slecht maar Puppet bestaat inmiddels al 11 jaar (Puppet) en veel DevOps hebben een goede ervaring met deze configuration tool. Toch verschijnen er steeds meer artikels over systeembeheerders die de overstap wagen van Puppet naar Ansible. Dit is ook het geval voor MediaIT, de informatica afdeling van de VRT. Bedrijven vervangen bestaande werkende configuraties niet zomaar door nieuwe tenzij daar een goede reden voor is. Is Ansible dan zoveel beter en zouden toekomstige bedrijven ook deze overstap moeten wagen?

[1] Een framework voor automatische configuratie van servers

2. State-of-the-art

Dan Tehranian is één van de DevOps die de overstap van Puppet naar Ansible gewaagd heeft. Hij publiceerde in januari 2015 twee artikels. Het ene over zijn literatuurstudie waarin hij de verschillen beschreef. Het tweede artikel bevatte zijn bevindingen na een week met Ansible gewerkt te hebben. Ook Ryan D Lane (2015) is iemand die een dergelijk onderzoek voerde. Niettemin het bij hem al vaststond dat hij geen Puppet meer ging gebruiken en eigenlijk onderzocht of hij voor Salt of Ansible moest gaan, had hij ook zijn bevindingen geconcludeerd over Puppet. Beide besluiten dat Ansible een lagere leercurve heeft door haar simpliciteit in syntax. Ook de agent-less architectuur van Ansible is een vaak vernoemd voordeel. Dit zijn inderdaad gegevens waar rekening mee gehouden moet worden. Maar dat is echter ook niet alles. Voor bedrijven hebben waarden zoals performantie, resource gebruik, schaalbaarheid en veiligheid prioriteit. Het zijn dan ook deze aspecten die door dit onderzoek behandeld zullen worden.

Dit onderzoek kan opgedeeld worden in drie delen die op de best passende manier gecontroleerd zullen worden. In eerste instantie zal er onderzocht worden welk configuration

management tool performanter, schaalbaarder en veiliger is. Bij het tweede deel zal er rekening gehouden worden met de businessrequirements. In dit geval deze van de VRT. Hierbij zal worden nagekeken wat de drijfveren van MediaIT waren om deze overstap te maken. In het laatste deel zullen de problemen die tijdens de omschakeling plaatsvinden geanalyseerd en beschreven worden. Op basis van deze zaken zal dit onderzoek trachten te concluderen voor toekomstige bedrijven of deze overstap aan te raden is.

3. Methodologie

3.1 Technische aspecten

In dit deel worden zaken onderzocht zoals performantie, resource gebruik, schaalbaarheid en veiligheid. Deze technische aspecten zullen gemeten worden in twee situaties. De eerste is tijdens de initiële deployment. Met andere woorden, alle services worden voor de eerste keer geïnstalleerd en geconfigureerd. Bij de tweede situatie zal gekeken worden hoe beide tools omgaan met aanpassingen bij bestaande configuraties.

Allereerst zal er een vergelijkende proef plaatsvinden. Om deze zo correct en betrouwbaar mogelijk uit te voeren zal er gefocust worden op twee grote deelaspecten, namelijk performantie en resource gebruik. Onder performantie wordt verstaan de tijd die nodig is om een server op te zetten tot een bepaald gewenste consistente staat. Verder wordt er bij resouce gebruik vooral gekeken naar de belasting van het netwerk en geheugengebruik.

Voor deze vergelijkende proef zullen Puppet en Ansible elk een server moeten configureren. Om te garanderen dat geen van beide tools bevoordeeld zou zijn, zullen de servers voorbereid worden met Vagrant [2]. Deze zal moeten verzekeren dat alle servers identiek zijn zodat zowel Ansible als Puppet vanuit éénzelfde situatie kunnen starten.

Tijdens de 'provisioning'[3] van de servers zullen nog nader te bepalen tools meten hoeveel geheugen gebruikt is geweest, hoelang dit alles geduurd heeft en hoe zwaar het netwerk hieronder geleden heeft. Op deze mannier zal er een oordeel geveld worden over de performantie en resource gebruik van beide technologieën. Ansible biedt ook de mogelijkheid om in accelerated mode te opereren (Ansible, 2016). Ook deze resultaten zullen gerapporteerd worden.

Voor de schaalbaarheid zal er vergelijkbaar te werk gegaan worden. Onder schaalbaarheid wordt verstaan: het vermogen om grote vraag te verwerken zonder kwaliteit te verliezen (informit, 2002). We zullen monitoren hoe Ansible en Puppet hun resources verdelen bij een toenemende drukte, hier onder de vorm van meer servers en uitgebreidere configuraties.
Voor het gedeelte veiligheid zal er in de eerste plaats een literatuurstudie plaatsvinden met onderzoek naar welke veiligheidsproblemen reeds gekend zijn en wat de best practises zijn om deze te voorkomen. Afhankelijk van de uitkomst van de literatuurstudie zal getracht worden een dergelijk veiligheidslek te reproduceren.

3.2 Redenen voor omschakeling

h

Dit gedeelte van het onderzoek zal zich meer op sociaal vlak afspelen. De verantwoordelijken binnen VRT voor de overstap van Puppet naar Ansible zullen een paar vragen voorgelegd krijgen, al dan niet tijdens een informeel gesprek. Op deze manier zal getracht worden de drijfveren achter hun beslissing tot overstap bloot te leggen.

3.3 Analyse van obstructies tijdens transitperiode

De omschakeling bij VRT is reeds begonnen op het moment van het schrijven van dit onderzoeksvoorstel en zal nog steeds bezig zijn tijdens het onderzoek zelf. Problemen die bij de vervanging van Puppet door Ansible optreden zullen gerapporteerd worden en er zal onderzocht worden waarom deze optraden. Al dan niet gevonden oplossingen zullen beschreven en uitgelegd worden. Welke incidenten zich zullen voordoen, valt uiteraard moeilijk te voorspellen. Bijgevolg is de grootte van deze sectie moeilijk in te schatten.

4. Verwachte resultaten

Gezien het grote enthousiasme voor Ansible door mensen binnen de professionele wereld zoals MediaIT van de VRT, en gespecialiseerde bedrijven zoals Gartner (2015) en Red Hat (2015) wordt verwacht dat Ansible goede, en wellicht betere, resultaten zal opleveren dan Puppet tijdens de vergelijkende studie op gebied van technische aspecten zoals performantie, schaalbaarheid en resource gebruik. Op vlak van veiligheid is het momenteel nog in het duister tasten. Dit is ook het geval bij het onderzoek naar de reden van de omschakeling. Vermoed wordt dat de prominenste reden de eenvoud in syntax is. Gezien het verschil in architectuur tussen Puppet en Ansible worden wel een aantal problemen verwacht tijdens de transitperiode. Mogelijkheden zijn bijvoorbeeld de zogenaamde 'Puppet master servers' en 'Puppet agents' die niet meer nodig zullen zijn. De grootste obstructie is uiteraard nog steeds de volledige vertaling van deze Puppet manifests naar Ansible playbooks.

5. Verwachte conclusies

Ondanks het feit dat Puppet momenteel nog steeds tot de marktleiders behoort (UpGuard, sd), wordt verwacht dat Ansible een volwaardige concurrent zal zijn. Of het ook wel degelijk beter is dan Puppet en of het een heel transitieproject waard is, valt echter moeilijk te voorspellen. Dat zal het onderzoek zelf moeten uitwijzen.

Referenties

[1] Geerling, J. (2016). Ansible for DevOps. Leanpub.

[2] Tool voor het autmatisch opzetten van virtuele machines, al dan niet in combinatie met een configuration management tool

[3] De techniek van het automatisch installeren of aanpassen van configuraties (Vagrant)

[2] HashiCorp. (2016). provisioning. Opgehaald van vagrantup: https://www.vagrantup.com/docs/provisioning/

[3] Gartner. (2015, April 21). Cool Vendors in DevOps, 2015. Opgehaald van Gartner: https://www.gartner.com/doc/3034319/cool-vendors-devops-

[4] UpGuard. (sd). Ansible vs Puppet. Opgehaald van UpGuard: https://www.upguard.com/articles/ansible-puppet

[5] Lane, R. D. (2014, Augustus 4). Moving away from Puppet: SaltStack or Ansible? Opgehaald van Ryan D Lane: http://ryandlane.com/blog/2014/08/04/moving-away-from-puppet-saltstack-or-ansible/

[6] RedHat. (2015, Oktober 16). Red Hat to Acquire IT Automation and DevOps Leader Ansible. Opgehaald van RedHat: https://www.redhat.com/en/about/press-releases/red-hat-acquire-it-automation-and-devops-leader-ansible

[7] Tehranian, D. (2015, Januari 20). Ansible vs Puppet – An Overview of the Solutions. Opgehaald van Dan Tehranian's Blog: https://dantehranian.wordpress.com/2015/01/20/ansible-vs-puppet-overview/

[8] Tehranian, D. (2015, Januari 20). Ansible vs Puppet – Hands-On with Ansible. Opgehaald van Dan Tehranian's Blog: https://dantehranian.wordpress.com/2015/01/20/ansible-vs-puppet-hands-on-with-ansible/

[9] Puppet. (2016, November 16). Puppet FAQ. Opgehaald van puppet: https://puppet.com/product/faq

[10] Ansible. (2016, December 6). Accelerated Mode. Opgehaald van Ansible: http://docs.ansible.com/ansible/playbooks_acceleration.html

[11] informit. (2002, Mei 24). Scalable and High-Performance Web Applications. Opgehaald van informit: http://www.informit.com/articles/article.aspx?p=26942seqNum=18

Bijlage C: Datasets

Netwerkverkeer

Netwerkverkeer: Puppet

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	gemiddelde
0	3270	1873	0	0	0	0	4259	0	0	0	9402
0	5115	0	0	0	0	0	0	4257	0	0	9372
0	5079	0	0	0	13	0	0	0	4280	0	9372
0	3248	2228	0	0	13	0	4280	0	0	0	9769
0	3203	1870	0	0	0	0	0	4253	0	0	9326
0	3202	1875	0	0	13	4278	0	0	0	0	9368
0	5102	0	0	0	0	4257	0	0	0	0	9359
0	5092	0	0	0	0	0	4265	0	0	0	9357
0	3222	1858	0	0	0	0	4261	0	0	0	9341
0	5063	0	0	0	0	0	4279	0	0	0	9342
0	5085	0	0	0	13	0	0	4227	0	0	9325
0	5080	0	0	0	13	0	4261	0	0	0	9354
0	5074	0	0	13	0	0	5156	0	0	0	10243
0	1641	3512	0	0	13	0	4270	0	0	0	9436
0	5084	0	0	0	13	0	4272	0	0	0	9369
0	5158	0	0	0	13	0	4248	0	0	0	9419
0	3210	1864	0	0	13	0	4257	0	0	0	9344
0	5086	0	0	13	0	0	4285	0	0	0	9384
0	5087	0	0	13	0	4281	0	0	0	0	9381
0	5059	0	0	13	0	4272	0	0	0	0	9344

0	5097	0	0	13	0	4247	0	0	0	0	9357
0	3011	2054	0	0	13	0	0	4265	0	0	9343
0	5077	0	0	13	0	0	0	4273	0	0	9363
0	5091	0	0	13	0	4285	0	0	0	0	9389
0	3210	1870	0	0	13	4256	0	0	0	0	9349
0	5092	0	0	13	0	0	5163	0	0	0	10268
0	5084	0	0	13	0	0	4271	0	0	0	9368
0	5076	0	0	13	0	4268	0	0	0	0	9357
0	5093	0	0	13	0	0	5134	0	0	0	10240
0	5072	0	0	13	0	0	0	4288	0	0	9373
0	5082	0	0	13	0	4285	0	0	0	0	9380

Netwerkverkeer: Ansible

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	gemiddelde
0	1811	0	1133	1401	0	0	0	0	1071	422	5838
0	1838	0	2302	0	0	1371	52	0	0	0	5563
0	1832	0	2248	0	0	0	663	787	0	0	5530
0	1852	2153	316	0	0	0	1429	0	0	0	5750
0	1830	619	1877	0	0	1455	0	0	0	0	5781
0	1858	2713	0	0	762	688	0	0	0	0	6021
0	1830	490	1955	0	0	1428	47	0	0	0	5750
0	1824	0	2748	0	0	1431	52	0	0	0	6055
0	1855	168	2315	0	0	360	1123	0	0	0	5821
0	1826	168	2098	342	0	0	1434	0	0	0	5868
0	1859	1792	664	0	0	1442	0	0	0	0	5757
0	1865	0	2466	0	0	1375	59	0	0	0	5765
0	1846	168	2355	0	0	0	1436	0	0	0	5805
0	1819	498	1971	0	0	1428	0	0	0	0	5716
0	1852	2474	0	0	0	1450	0	0	0	0	5776
0	1832	168	2312	0	0	358	1109	0	0	0	5779
0	1852	0	930	1629	0	0	0	1461	0	0	5872
0	419	1452	909	1566	0	0	1426	0	0	0	5772
0	1865	0	883	1692	0	0	0	1109	382	0	5931
0	1879	0	2458	0	0	0	1048	422	0	0	5807
0	1826	2234	252	0	0	0	462	981	0	0	5755
0	1844	0	2530	0	0	0	1391	59	0	0	5824
0	1865	168	2317	0	0	0	1464	0	0	0	5814
0	1831	413	2096	0	0	0	1482	47	0	0	5869
0	1839	0	2484	0	0	0	1095	389	0	0	5807
0	1872	1444	1091	0	0	1434	0	0	0	0	5841
0	1889	505	2031	0	0	0	0	1425	0	0	5850
0	1859	168	2293	0	0	1015	414	0	0	0	5749
0	1845	0	858	1599	0	0	796	654	0	0	5752

0	1862	0	2115	361	0	0	1070	402	0	0	5810
0	1845	534	1949	27	0	1070	418	0	0	0	5843

Geheugengebruik

Geheugengebruik: Puppet

38.02	48.85	64.39	58.46	67.23	68.15	63.54	57.98	39.22		
37.42	46.06	57.93	57.77	65.97	68.31	63.2	53.36	38.66		
37.70	42.26	57.95	55.13	66	67.53	67.56	62.64	45.18	38.93	
36.67	41.61	58.03	56.97	65.13	66.72	62.52	53.49	42.39		
36.99	41.3	48.8	57.57	65.01	67.31	63	51.31	41.55	38.71	
36.80	41.61	57.52	54.85	61.38	66.61	62.32	53.7	39.34		
38	48.7	61.46	60.96	66.70	62.36	54.25	42.47	39.19		
36.69	41.75	57.98	51.38	66.52	68.44	62.63	47.55	38.74		
37.33	45.9	64.3	57.66	67.31	67.22	53.91	39.16			
37.03	41.58	48.73	57.59	57.75	67.14	67.95	62.56	55.44	38.83	
37	42.37	59.58	55.64	58.48	67.23	67.08	62.18	53.87	38.65	
37	41.61	57.97	54.09	66.49	68.04	66.36	51.82	39.09		
37.03	48.05	64.32	57.52	67.27	68.38	62.5	45.87	38.66		
36.69	41.63	57.97	55.94	65.91	67.98	68.05	58.06	43.89	38.49	
37.69	48.63	63.75	58.6	65.06	66.85	62.89	47.65	38.99		
37.31	45.07	61.34	56.72	67.33	68.22	57.82	46.82	41.8	38.76	
36.99	42.2	57.28	55.5	65.74	67.32	49.68	42.23	39.37		
36.69	41.3	48.8	64.3	57.73	64.5	65.60	66.38	62.41	53.99	38.83
37.67	48.66	60.56	58.47	65.7	66.31	62.3	49.6	38.51		
37.33	48.02	64.34	56.08	65.73	66.51	57.94	43.94	38.84		
36.89	44.85	57.86	48.88	66.55	68.41	57.49	44.51	38.18		
37.77	48.71	64.35	57.59	66.52	68.03	67.90	57.86	52.6	38.71	

Geheugengebruik: Ansible

28.45	46.62	38.17	50.78	49.33	40.54	31.5		
28.48	33.23	40.31	41.26	50.89	49.7	40.63	31.49	
29.17	46.64	44.1	50.83	52.12	32.17			
28.41	31.28	46.75	44.14	50.86	49.93	32.9	31.6	
28.44	41.05	39.67	50.74	50.82	48.81	33.43	31.51	
28.69	46.71	41.26	50.8	48.44	31.66			
28.42	31.46	38.67	49.11	50.86	52.09	41.17	31.43	
28.43	34.86	39.87	50.74	50.85	48.82	31.49		
28.42	38.58	37.19	34.72	50.79	53.09	49.02	31.82	
28.41	33.97	38.63	49.6	50.86	55.29	40.42	31.72	
28.14	42.31	36.57	41.56	50.33	54.77	41.76	31.21	

27.77	30.66	45.93	40.55	50.22	50.24	50.27	48.3	31.49
27.8	40.44	37.76	50.21	50.25	51.82	46.83	31.22	
27.84	40.54	46.02	35.36	50.27	50.29	54.06	41.29	31.21
27.84	29.36	46.03	43.23	50.35	52.6	48.49	31.52	
27.82	33.44	47.45	38.56	50.27	50.3	48.29	31.38	
27.82	40.76	34.26	50.26	52.57	51.02	32.66		
28.15	41.08	34.30	50.27	52.62	41.39	31.26		
27.84	30.76	46.02	43.56	50.32	51.91	40.24	31.21	
27.83	37.97	40.13	45.08	50.32	50.28	48.38	31.46	
28.09	42.83	35.52	50.24	50.32	48.39	32.78	32.22	
27.9	40.79	40.65	50.3	50.35	50.89	41.84	31.27	

Deploytijden

Connectietijd		Volledige configuratie		Gedeeltelijke configuratie		Geen configuratie vereist	
Puppet	Ansible	Puppet	Ansible	Ansible	Puppet	Puppet	Ansible
8.9	9	51.35	67	14	2.92	0.41	19
5.68	5	43.56	60	20	3	0.38	11
6	5	46.78	51	23	2.96	0.34	12
11.18	7	40.59	59	23	2.98	0.37	18
6.23	4	55.18	51	14	2.91	0.39	22
7.3	5	47.01	51	19	5.86	0.38	20
7.71	6	35.99	59	30	3.11	0.4	9
6.06	5	35.07	57	15	2.93	0.43	11
8.58	4	43.29	45	21	2.88	0.42	16
8.64	11	42.28	90	22	3.24	0.38	10
6.61	6	42.2	52	14	3.19	0.62	11
6.57	4	96.83	53	14	4.36	0.36	10
7.68	8	32.33	54	20	2.95	0.38	15
6.35	10	49.99	52	25	2.93	0.36	17
5.64	7	40.32	62	13	2.97	0.36	19
8.76	5	55.62	68	14	2.9	0.4	15
9.56	6	52.82	51	13	2.96	0.35	22
8.23	4	42.72	59	18	2.85	0.42	16
8	9	44.21	53	16	2.95	0.59	16
7.41	5	43.13	45	24	2.87	0.39	19
11	10	47.43	96	19	2.87	0.4	19
13	8	56.98	73	15	2.87	0.38	19
13	6	59.97	61	23	3.09	0.42	10
8	5	61.28	62	13	2.95	0.44	9
12	5	53.98	59	14	2.93	0.44	10
8	5	56.56	64	32	2.96	0.4	11

7	5	53.57	65	14	2.93	0.9	10
7	7	49.01	64	14	2.88	0.42	11
8	6	50.21	76	25	2.8	0.42	11
10	15	51.3	61	32	3.1		