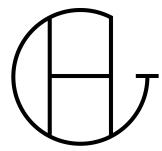


Technische voor- en nadelen van Puppet en Ansible. Verloop en redenen van een omschakeling.

**Build it.
Automate it.**

*use a configuration
management tool*





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

TO DO

Voorwoord

Drie jaar geleden begon ik aan de hogeschool van 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 de meeste aspecten die informatica biedt waaronder zaken zoals programmeren, artificiële intelligentie en netwerk- en systeembeheer. Ondanks het feit dat ik al deze zaken interessant vond, was er één onderdeel die geleidelijkaan mijn voorkeur zou krijgen en dit werd het beheren van servers.

Hier werd 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 om 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 om ook mijn bachelorproef rond dit fasinerende onderwerp te voeren.

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

mevr. Lambrecht Carine

Bedankt voor het verzorgen van het linguïstisch aspect van de bachelorproef en bieden van morele steun.

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 dat ik bij u terecht kon voor vragen en uw mening over de stand van zaken gedurende de bachelorproef.

dhr. Dierick Gerben

Bedankt voor het interessante gesprek betreffende de veiligheid en risico's omtrent deze technologieën.

dhr. Adams Pieter

Bedankt voor dat ik bij u terecht kon voor technische vragen en de introductie van Ansible Tower.

Inhoudsopgave

1	Inleiding	11
1.1	Stand van zaken	11
1.1.1	Profiel van Puppet	13
1.1.2	Profiel van Ansible	13
1.2	Opzet van deze bachelorproef	13
1.3	Probleemstelling en Onderzoeksvragen	14
1.3.1	Wat zijn de redenen van een omschakeling?	14
1.3.2	Wat zijn de technische voor-en nadelen van Puppet en Ansible?	15
1.3.3	Wat is het verloop van een dergelijke transitperiode?	15
2	Methodologie	17
2.1	Redenen van een omschakeling?	17

2.2 Werking van Ansible en Puppet?	18
2.2.1 Overzicht van Puppet en Ansible	18
2.2.2 Werking van Puppet	18
2.2.3 Werking van Ansible	18
2.3 Technische analyse van Ansible en Puppet?	19
2.3.1 Opstelling van de test omgeving	19
2.3.2 Belasting van het netwerk	20
2.3.3 Tijd tot het bekomen van een consistente staat	23
2.3.4 Gebruik van het geheugen	25
2.3.5 Schaalbaarheid	25
2.4 Wat is het verloop van een dergelijke transitperiode?	27
2.5 Veiligheid	28
3 Conclusie	29
4 Bijlage A: Mathematische berekeningen	35
4.1 Hypothese configuratietijd (volledige configuratie)	35
4.2 Hypothese connectietijd	36
5 Bijlage B: datasets	37
5.1 Netwerkverkeer	37
5.1.1 Puppet	37
5.1.2 Ansible	38
5.2 Geheugengebruik	39
5.2.1 Puppet	39
5.2.2 Ansible	39

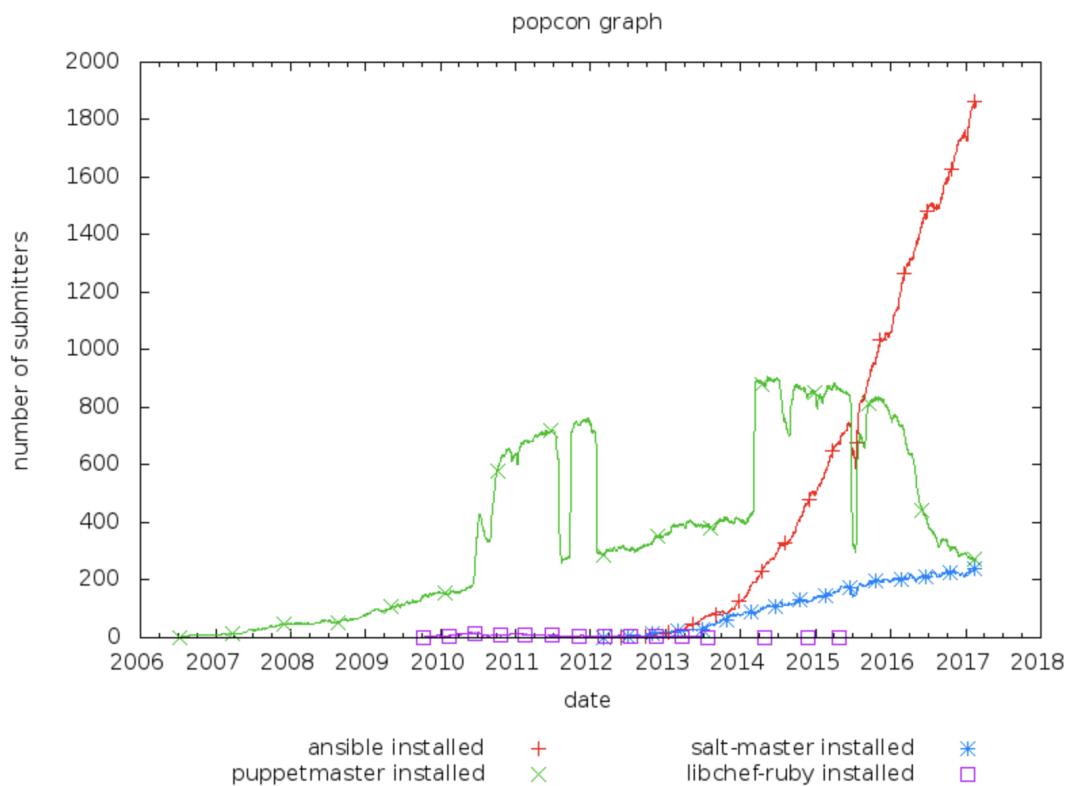
5.3 Deploytijden 40

Bibliografie 44

1. Inleiding

1.1 Stand van zaken

Bedrijven kunnen tegenwoordig niet zonder IT-infrastructuur. Deze infrastructuur kan zeer 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 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.-j). Eens de gewenste configuratie geprogrammeerd is, kunnen extra gelijkaardige servers veel sneller opgezet worden. Puppet is daar altijd al marktleider in geweest. 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 Configuration management tool (CMT) 's doet het opvallend beter op gebied van populariteit en dat is Ansible inc. Zoals op de grafiek 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.



Figuur 1.1: Deze grafiek toont het aantal keer dat een bepaald softwarepakket geïnstalleerd is op een Debian distributie. (Debian, 2017)

1.1.1 Profiel van Puppet

Puppet is een open source project dat werd ontwikkeld in 2005 door Luke Kanies (Puppet, g.d.-g) met als doel om 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.-e). 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.-h) (Puppet, g.d.-d)

1.1.2 Profiel van Ansible

Michael DeHaan ondervond in zijn ervaring dat mensen moeilijkheden ervaarden 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 die zorgde voor een duidelijk configuratiebeheer, eenvoudig deployen van nieuwe servers en als het nodig was de mogelijkheid bood tot ad-hoc commando's. Met dit idee is hij samen met Saïd Ziouani in 2012 het open source project Ansible gestart (Geerling, 2016). Ook Ansible werkt volgens dit server/client model. Opvallend is wel dat elke computer waarop Ansible draait in principe kan fungeren als server. In bedrijven zoals de VRT wordt er wel gekozen voor een centraal punt. Dit wordt dan 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 in de Linux wereld is ook Ansible in staat om Windows servers te configureren. (Ansible, g.d.-c)

1.2 Opzet van deze bachelorproef

Dit onderzoek vindt plaats op MediaIT, een afdeling binnen het mediahuis VRT. Het is één van de afdelingen die verantwoordelijk is voor een goede en correcte werking van de servers. Zij staan momenteel 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 staging en productie. Deze manier van werken is niet met Puppet geïntegreert wat het testen bemoeilijkt. Maar ook andere zaken spelen parten zoals de complexiteit van Puppet en de beperkte functionaliteit tot het monitoren van configuraties.

Ten tweede moet de VRT ook voortdurend vernieuwen. Zo komt er een geheel nieuw gebouw bij die 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 de alleenheerster en steeds meer programma's worden bekeken via sites en app's. 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 met 3,3% t.o.v. het jaar daarvoor wat resulteert in een 22,4%. Dit terwijl de smartphone, computer en tablet gezamenlijk 29,7% halen. Het is vanzelfsprekend dat het mediahuis VRT deze trend moet volgen.

Het is dus belangrijk dat een geschikte CMT gebruikt wordt en dat deze perfect geïntegreerd is met 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,...), zal optimaal moeten opereren in het toekomstige hybride infrastructuur, zal configuraties moeten monitoren, maar boven dit alles zal het eenvoudig in gebruik moeten zijn.

1.3 Probleemstelling en Onderzoeks vragen

Ansible is sinds enige tijd aan een stevige opmars bezig maar er zijn voldoende voorbeelden van opensource (en andere) projecten die na een initiële hype snel in elkaar zakten. Ondertussen heeft Ansible tal van mooie referenties achter zich waarbij deze van NASA nog het meest voor de verbeelding spreekt. Zij gebruikten Ansible om hun datacenter te migreren naar de cloud (RedHat, 2013). Verder heeft Ansible ook heel wat 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 een grote community die het project ondersteunt?

De overschakeling van Puppet naar Ansible is geen kleine stap en kan mogelijk voor veel complicaties zorgen. Daarom wilt 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 belangrijk te weten wat de drijfveren waren voor de beslissing om Puppet te vervangen door Ansible en dat is precies waar deze eerste categorie toe dient. 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 staat 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 namelijk 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 optradën. Al dan niet gevonden oplossingen zullen beschreven en uitgelegd worden zodat andere bedrijven zich goed bewust zijn van wat er 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?

TO DO - hoe eenvoudig ansible is om te leren - slechte monitoring - slecht geautomatiseert
- geen multi environment - expert vertrokken - gui veel werk en complex - oorspronkelijk
geen modules, refactor, in feite nu weer refactor - updates zorgen voor compatibiliteitpro-
blemen (denk ik) - puppet client niet ouder dan puppet master maar omgekeerd denkik wel
(op te zoeken)

2.2 Werking van Ansible en Puppet?

2.2.1 Overzicht van Puppet en Ansible

	Ansible	Puppet
Programmeerparadigma	declaratief	declaratief
Geschreven in	YAML	eigen Domain specific language (DSL)
Gecompileerd naar	Python	Ruby
Communicatieprotocol	SSH	HTTPS
Push / Pull principe (standaard) ^a	push	pull
open poorten ^b	22/tcp (client)	8140/tcp (master) Pup-

^aBeide 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'

^bDit 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)

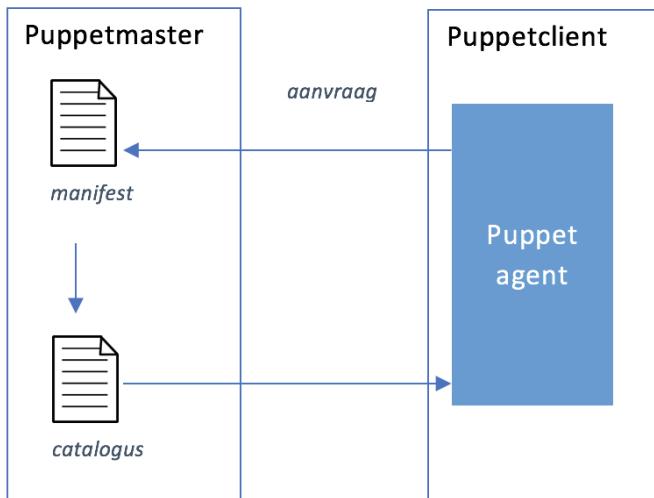
pet (g.d.-k), Weirdt (2014), Ansible (g.d.-b)

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.-f). 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.-h).

2.2.3 Werking van Ansible

In tegenstelling tot Puppet maakt Ansible geen gebruik 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 modules. Eens een verbinding tot stand is gebracht, wordt dit playbook met zijn modules verstuurd naar de te configureren server. Deze worden vervolgens op de Ansible clients uitgevoerd en weer verwijderd. Ook Ansible bezit de functionaliteit om na te gaan of de huidige configuratie



Figuur 2.1: aanvraag van een catalogus bij de Puppetmaster door een Puppetclient. De Puppetagent is een deamon (stukje software) die op de Puppetclient draait.

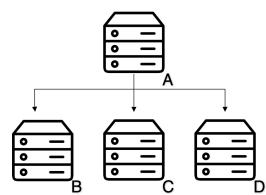
in lijn is met de ontvangen modules. Om servers te configureren met Ansible bestaan er bovendien twee manieren. Ansible playbooks kunnen in principe verstuurd worden naar de servers vanaf elke computer. Voor een grotere hoeveelheid servers is dit echter moeilijk onderhoudbaar en onoverzichtelijk. Hiervoor bestaat er de commerciële versie waarbij de playbooks worden verstuurd vanaf een centraal punt. Dit centraal punt is voorzien van Ansible Tower die een inventaris heeft 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?

2.3.1 Opstelling van de test omgeving

Architectuur van de opstelling

Deze opstelling is gerealiseerd door gebruik te maken van twee Vagrant bestanden. Voor elk Vagrant bestand wordt eerst de master aangemaakt, gevolgd door X aantal client's. Vagrant zorgt ervoor dat elke client verbonden wordt met New Relic, de gekozen tool om de servers te monitoren. In het geval van Puppet wordt hierbij ook nog de Puppet agent geïnstalleerd. Elke client, zowel in de Ansible- als de Puppetinfrastructuur, is gebaseerd op dezelfde basebox en krijgen dezelfde resources toegekend. De masters beschikken over meer resources.



Figuur 2.2: Infrastructuur.

Technische specificaties clients

Basebox: bertvv/centos72

Aantal CPU's: 1

Geheugen: 500 MB

Configuraties van de deploy's

Opmerking: De reden dat MariaDB op dit moment niet gestart wordt is vanwege de gedeeltelijke configuratie. Hierbij wordt gekeken hoelang het duurt om bij een bestaande configuratie enkele aanpassingen door te voeren. Het is dus pas bij de testen van gedeeltelijke configuratie dat MariaDB gestart wordt. 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 gebeurt. Bovendien is er getracht geweest 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-server geïnstalleerd. Vervolgens wordt er een webpagina op de site geplaatst. Deze is geschreven in PHP en controleert of alle zaken werken naar behoren. Om af te sluiten wordt HTTP doorgelaten door de firewall, de service httpd gestart en MariaDB gestopt.

Deze configuraties kunnen teruggevonden worden op gitbub op de volgende links¹:

Ansible: <https://github.com/ThomasDetemmerman/AnsibleTower>

Puppet: <https://github.com/ThomasDetemmerman/PuppetMaster>

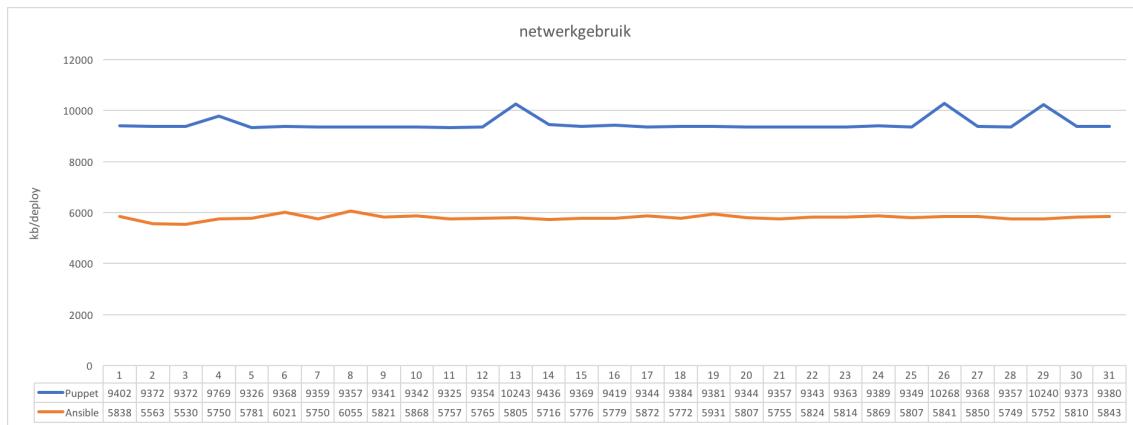
2.3.2 Belasting van het netwerk

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 2.3 en 2.4 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 Ansible Tower hier onmiddellijk van op de hoogte gebracht. Zoals te zien is op grafiek 2.4 is er geregelde 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 t6 de communicatie nul is. Ansible had voor die periode niets te melden².

¹Mogelijk komt beschreven configuratie in dit artikel niet volledig overeen met de code in de Github repositories. Deze repositories worden namelijk ook gebruik voor andere testen binnen dit onderzoek.

²Hier betreft het de service MariaDB die werd geïnstalleerd



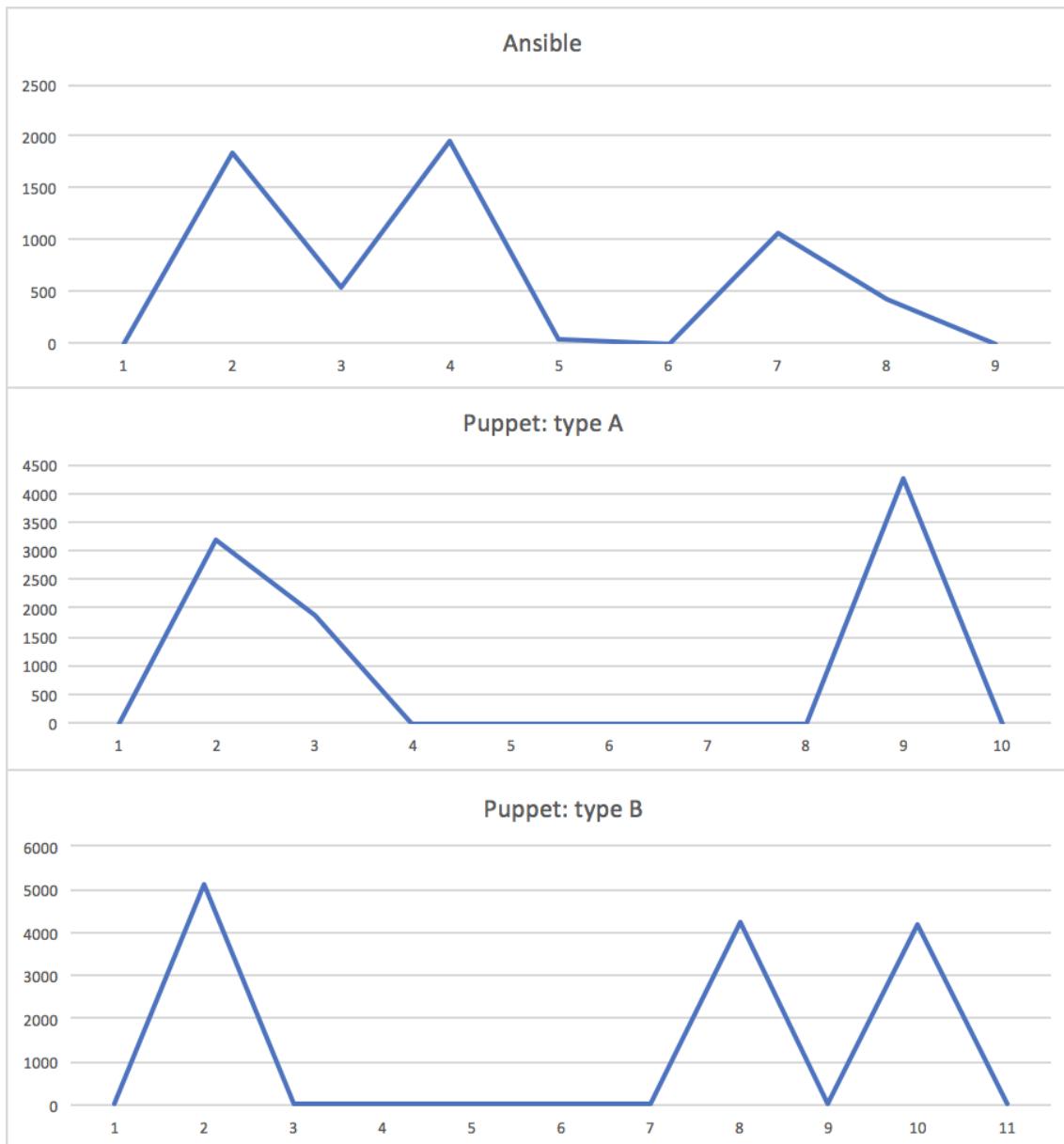
Figuur 2.3: Totaal verbruikte netwerkcapaciteit per client gedurende het deployen. Dit bevat enkel communicatie tussen master en client.

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' niet gebruikt voor rollen in productie. Deze lopen doorgaans tijdens de nacht en is het voldoende om de dag daarna een algemeen overzicht te krijgen.

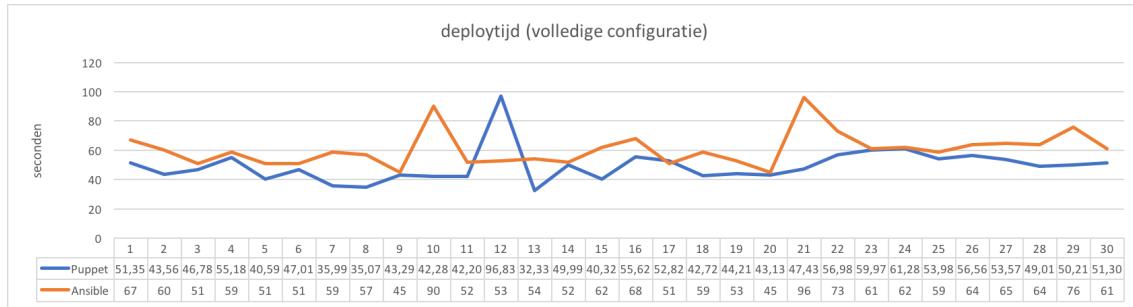
Bij Puppet is dit anders. Hierbij is er enkel communicatie tussen de server en de client bij het begin en op einde van de deploy. Opmerkelijk hierbij is dat er twee types te herkennen zijn. Op figuur 2.4 zijn deze 'Puppet type A' en 'Puppet type B' genoemd. Bij Puppet type A is te zien hoe de deploy bestaat uit twee reeksen. Hierbij wordt in het begin de catalogus opgehaald en op het einde wordt een rapport terug gekoppeld naar de master (Puppet, g.d.-c). Dit terwijl bij type B uit drie reeksen bestaat. De oorzaak van deze derde reeks is niet bekend. Het opnieuw versturen van de tweede reeks was tijdelijk een piste maar dit is vermoedelijk niet het geval. Moest er een tcp-pakketje verloren gegaan zijn, zou uitsluitend dat pakketje opnieuw verstuurd worden en niet de hele reeks. Verder heeft de poging om de inhoud van de pakketjes te lezen tot niet veel geleid. De verbinding is namelijk geëncrypteerd door het HTTPS-protocol met als gevolg dat tools zoals Wireshark of tcpdump geen oplossing kunnen bieden over de inhoud hiervan.

Een nadeel aan het feit dat er enkel in het begin en ophet einde communicatie is, is dat er op de master geen live feedback voor testen gevuld kan worden. Dit kan echter wel opgelost worden door in te loggen op de client en hier de live feedback volgen met 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 2.3. Hier heeft Ansible een gemiddelde van 5802,29 kilobytes/deploy. Puppet heeft bij deploy's van type A (bestaande uit twee reeksen) een gemiddelde van 9392,5 kilobytes/deploy en bij type B (bestaande uit drie reeksen) 13742,35 kilobytes/deploy. Gezamelijk heeft Puppet een gemiddelde van 11777,90 kilobytes/deploy.



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



Figuur 2.5: Tijd tot het bekomen van een consistente staat, vertrekende van een 'lege' server.

2.3.3 Tijd tot het bekomen van een consistente staat

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. Vervolgens wordt elke configuratie 30 keer uitgevoerd. 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, 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 resultaten³. 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. Beide waarden zijn gebaseerd op de feedback van de corresponderende CMT.

configuratietijd

Aangezien grafiek 2.5 geen eenduidig verschil toont tussen beide CMT's is er met behulp van de Z-toets aangetoond of er al dan niet een statistisch verschil is. Deze berekeningen kunnen teruggevonden worden in bijlage A: hypothese configuratietijd.

Hierbij blijkt dat Z buiten het kritisch gebied valt waardoor de nulhypothese verworpen kan worden. Bijgevolg wordt aangenomen dat beide gemiddelden niet tot dezelfde verzameling behoren. 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 is vertrokken van servers die reeds geconfigureerd zijn en slechts enkele aanpassingen moeten doorgevoerd worden. Deze aanpassingen zijn een service starten en de inhoud van de webpagina veranderen. De

³connectietijd = totaal verstreken tijd - \sum (tijd playbooks)

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 is geconfigureerd en dat geen aanpassingen 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 waarbij er minder dan een seconde nodig heeft om vast te stellen dat 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.

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

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

Hypothese A: Ansible stuurt taak per taak.

Hypothese P: Puppet stuurt een gehele configuratie op het moment van initialisatie.

Test: De verbinding wordt verbroken tijdens het configureren.

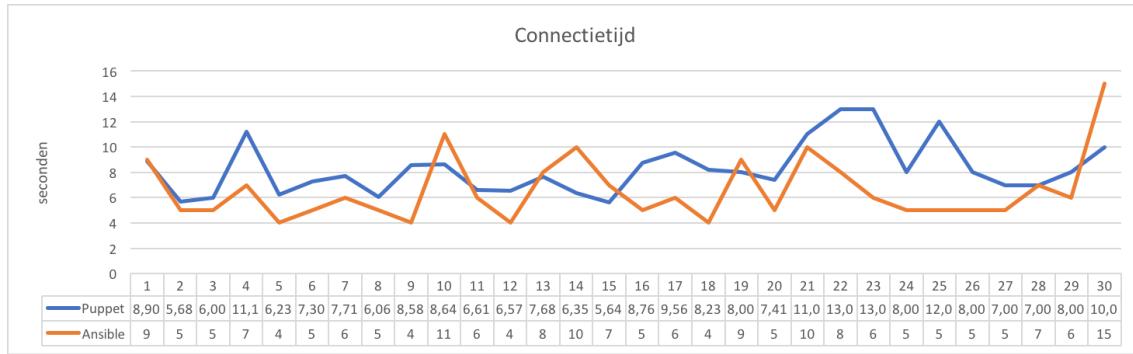
Verwachting A: Nadat de verbinding verbroken is wordt geen nieuw taak gestart.

Verwachting P: De configuratie gaat zonder problemen verder.

Resultaat A: De configuratie valt stil. Opmerkelijk hierbij is dat ondanks de configuratie stilvalt, deze niet stopt. De master heeft namelijk niet door dat de verbinding verbroken is en verondersteld dat de client eenvoudigweg nog bezig is met configureren. Als later de verbinding hersteld wordt, gaat de configuratie opnieuw verder.

Resultaat P: De configuratie gaat ongestoord verder.

Er wordt 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) hekend dit probleem en biedt hiervoor een alternatief aan, genaamd 'pipelinging'. Hierdoor zouder er minder SSH-verbindingen nodig zijn die modules moeten overzetten. Een nadeel hieraan is wel dat `requiretty` uitgezet moet worden wat mogelijk tot een beveiligingsprobleem kan leiden.



Figuur 2.6: Tijd tot het initialiseren van een deploy

Connectietijd

Ook hier lopen de resultaten door elkaar zoals te zien is op grafiek 2.6, bovendien liggen de gemiddelden hier veel dichter bij elkaar. Om vast te stellen of er een significant verschil is, is er opnieuw gebruik gemaakt geweest van de Z toets. Deze berekeningen kunnen teruggevonden worden in bijlage A, hypothese connectietijd. Z ligt hier echter in het aanvaardingsgebied waardoor we de nulhypothese, die stelt dat beide gemiddelden gelijk zijn, kunnen aanvaarden. Er is bijgevolg geen statistisch verschil tussen beide waarden.

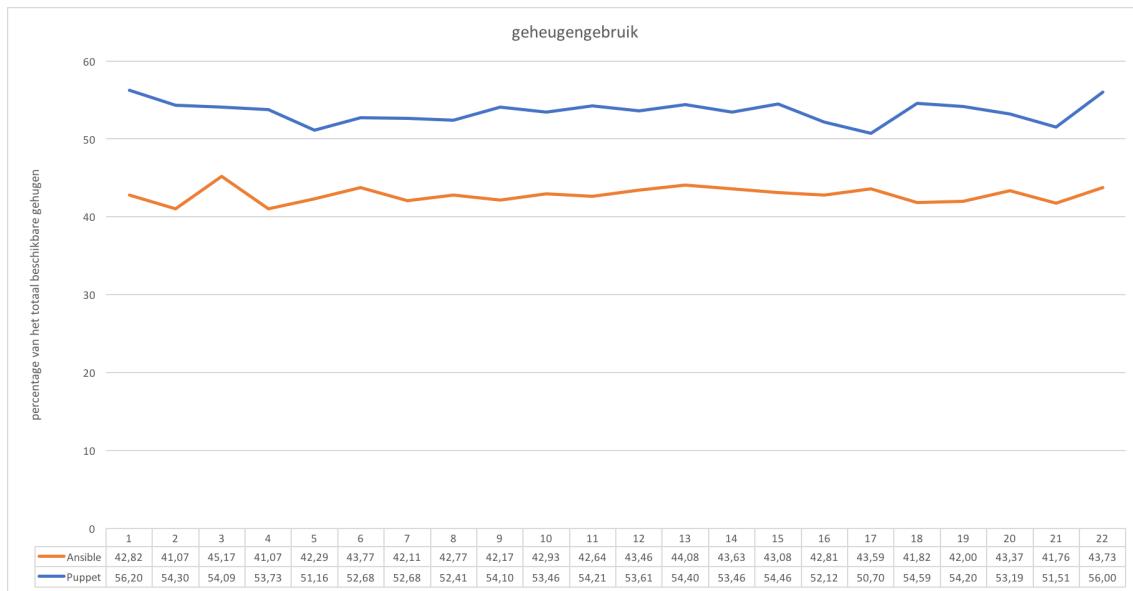
2.3.4 Gebruik van het geheugen

Op grafiek 2.7 is per tijdseenheid het gemiddeld gebruikt ramgeheugen 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 ramgeheugen daalde onmiddellijk naar gelijkwaardige waarden als deze van de Ansibleclient. Zonder configuratie gebruiken Puppetclients gemiddeld 58% geheugen van de 500MB ram. 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.

2.3.5 Schaalbaarheid

Het zou niet bedrijfs-realisch zijn om het gedrag van de resources - gemeten in 2.2 - op dezelfde wijze te meten door meer clients toe te voegen. Bij Ansible is het namelijk Ansible Tower die bepaalt wanneer de clients geupdate moeten worden. Maar Puppet werkt niet volgens dit principe, hier valt deze verantwoordelijkheid onder de clients zelf. De kans is dus klein dat in het geval van Puppet alle clients tegelijk een catalogus aan zouden vragen en hierdoor de Puppetmaster (te) zwaar zouden belasten.

Toch zijn er hier en daar zaken die de performantie ten goede komen. Zo blijft het



Figuur 2.7: Verbruikt percentage van het RAM geheugen. Gemeten bij servers met elk 500 MB.

downloaden van software de grootste bottleneck van een goede en snelle deploy. Heel wat verkeer moet het lokale netwerk verlaten om bestanden van een externe bron op te halen. Een oplossing hiervoor is om een lokale server op te stellen die een kopie bewaart van de (meest) gebruikte services. (Ansible, g.d.-a).

Tips om de performantie van Ansible te verbeteren

Standaard is Ansible geconfigureerd om 5 servers tegelijk te kunnen configureren. Dit aantal kan verhoogd worden door de parameter `forks` naar 25 tot zelfs 100 te brengen.

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 transactie blok wat de performantie ten goede komt. Zo zou het dus beter zijn om listing 2.1 te structureren zoals listing 2.2.

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

```

1  - name: installeer PHP
2    package:
3      name: php
4      state: present
5
6  - name: installeer php-mysql
7    package:
8      name: php-mysqli
9      state: present
10

```

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) .

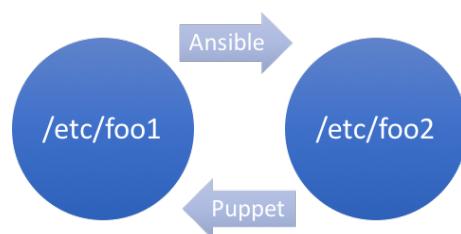
Tips om de performantie van Puppet te verbeteren

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 varieert van server tot server. Dit is standaard '(aantal CPU's) - 1' met een minimum van één en een maximum van vier. Het aantal gebruikte CPU's staat gelijk aan het aantal aanvragen dat Puppet tegelijk kan behandelen. Wanneer er meer aanvragen zijn dan beschikbare CPU's zal het overschot aan aanvragen geblokkeerd worden tot er een slot vrijkomt. Dit aantal kan handmatig verhoogd worden met behulp van de variabele `max-active-instances`. Wanneer deze naar bijvoorbeeld twee gebracht wordt, zullen twee core's van de processor gebruikt worden, wat tot gevolg heeft dat er ook twee aanvragen tegelijk behandeld kunnen worden.

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

2.4 Wat is het verloop van een dergelijke transitperiode?

Het is vanzelfsprekend dat de manier om Ansible te integreren afhankelijk is 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 overgang. 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.



Figuur 2.8: Vicieuze cirkel van twee CMT's hetzelfde bestand proberen te configureren.

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 2.8. In Ansible staat een bestand met een extra spatie, hier noemen we deze voor het gemak 'foo1'. 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 gerestart 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 niet voor 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.

2.5 Veiligheid

To Do, gesprek met mr. Gerben

3. Conclusie

TO DO

Lijst van acroniemen

CMT Configuration management tool. 11, 13–15, 20–22, 27, 28, 45

DSL Domain specific language. 17

Verklarende Woordenlijst

ad-hoc commando . 13

Catalogus Eng: Catalog. Een catalogus beschrijft de gewenste configuratie voor een specifieke computer puppetdoc. 23

Configuratietijd De tijd die de configuration management tool nodig heeft tot het bekomen van een volledig geconfigureerde server.. 20, 21

Connectietijd Dit is 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.. 20, 21

Fork Het aanmaken van een child process door zichzelf te dupliceren (Linux, 2017). 26

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.. 20

Package manager Een mechanisme die het mogelijk maakt om software te installeren op UNIX gebaseerde systemen. (*voorbeelden: yum, apt, dpkg,....* 28

Programmeerparadigma Synoniemen zijn ook programmeerstijl of programmeermodel.
17

Pull Een manier van communiceren waarbij de actie gestart wordt vanuit de clients, de ontvangers (Techopedia, g.d.). 17

Push Een manier van communiceren waarbij de actie gestart wordt vanuit een centraal punt, de zender. (Techopedia, g.d.). 17

4. Bijlage A: Mathematische berekeningen

4.1 Hypothese configuratietijd (volledige configuratie)

Gebruikte dataset 5.3

Hypothese

$$H_0 : \mu_p = \mu_a$$

$$H_a : \mu_p \neq \mu_a$$

Significantieniveau en waarden

$$\alpha = 0.05 \Rightarrow -1.96 \text{ en } +1.96$$

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

Toetsingsgrootheden

$$\begin{aligned} Z &= \frac{\bar{x}_p - \bar{x}_a}{\sqrt{\frac{\sigma_p^2}{n_p} + \frac{\sigma_a^2}{n_a}}} \\ &= \frac{49.4 - 60.7}{\sqrt{\frac{11.6^2}{30} + \frac{11.5^2}{30}}} \\ &= -3,789 \end{aligned} \tag{4.1}$$

4.2 Hypothese connectietijd

Gebruikte dataset 5.3

Hypothese

$$H_0 : \mu_p = \mu_a$$

$$H_a : \mu_p \neq \mu_a$$

Significantieniveau en waarden

$$\alpha = 0.05 \Rightarrow -1.96 \text{ en } +1.96$$

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

Toetsingsgrootheden

$$\begin{aligned}
 Z &= \frac{\bar{x}_p - \bar{x}_a}{\sqrt{\frac{\sigma_p^2}{n_p} + \frac{\sigma_a^2}{n_a}}} \\
 &= \frac{6,27 - 6,57}{\sqrt{\frac{4,10^2}{30} + \frac{6,11^2}{30}}} \\
 &= 1,27
 \end{aligned} \tag{4.2}$$

5. Bijlage B: datasets

5.1 Netwerkverkeer

5.1.1 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

5.1.2 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

5.2 Geheugengebruik

5.2.1 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	

5.2.2 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	

5.3 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		

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). *Windows Support*. Verkregen van http://docs.ansible.com/ansible/intro_windows.html
- Debian. (2017). *Debian Popularity Contest*. Verkregen van <http://popcon.debian.org>
- Geerling, J. (2016). *Ansible For DevOps*. Leanpub.
- Imec. (2017). digimeter 2016.
- informati. (2002, mei). *Scalable and High- Performance Web Applications*. Verkregen van <http://www.informati.com/articles/article.aspx?p=26942&seqNum=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>
- 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). Verkregen van <https://docs.puppet.com/puppet/4.9/architecture.html>
- Puppet. (g.d.-d). *FAQ*. Verkregen van <https://puppet.com/product/faq>
- Puppet. (g.d.-e). *Installing Puppet Enterprise for a Windows environment*. Verkregen van https://docs.puppet.com/pe/latest/windows_installing.html
- Puppet. (g.d.-f). *Language: Basics*. Verkregen van https://docs.puppet.com/puppet/4.9/lang_summary.html
- Puppet. (g.d.-g). *Leadershipt*. Verkregen van <https://puppet.com/company/leadership>
- Puppet. (g.d.-h). *Puppet Documentation*. Verkregen van docs.puppet.com

- Puppet. (g.d.-i). *Puppet Server: Tuning Guide*. Verkregen van https://docs.puppet.com/puppetserver/latest/tuning_guide.html
- Puppet. (g.d.-j). *Puppet's Declarative Language: Modeling Instead of Scripting*. Verkregen van <https://puppet.com/blog/puppet?i=s-declarative-language-modeling-instead-of-scripting>
- Puppet. (g.d.-k). *Why Puppet has its own configuration language*. Verkregen van <https://puppet.com/blog/why-puppet-has-its-own-configuration-language>
- 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>
- Techopedia. (g.d.). *Push Technology*. Verkregen van <https://www.techopedia.com/definition/5732/push-technology>
- Weirdt, H. D. (2014). *Configuratieafhankelijkheden gebruiken om gedistribueerde applicaties efficiënt te beheren* (masterscriptie, KU Leuven).

Lijst van figuren

1.1 Deze grafiek toont het aantal keer dat een bepaald softwarepakket geïnstalleerd is op een Debian distributie. (Debian, 2017)	12
2.1 aanvraag van een catalogus bij de Puppetmaster door een Puppetclient. De Puppetagent is een deamon (stukje software) die op de Puppetclient draait.	19
2.2 Infrastructuur.	19
2.3 Totaal verbruikte netwerkcapaciteit per client gedurende het deployen. Dit bevat enkel communicatie tussen master en client.	21
2.4 Drie types van communicatie. Aantal kilobytes per tijdseenheid op een netwerkkaart die uitsluitend bedoeld is voor communicatie met Ansible Tower / Puppetmaster.	22
2.5 Tijd tot het bekomen van een consistente staat, vertrekende van een 'lege' server.	23
2.6 Tijd tot het initialiseren van een deploy	25
2.7 Verbruikt percentage van het RAM geheugen. Gemeten bij servers met elk 500 MB.	26
2.8 Vicieuze cirkel van twee CMT's hetzelfde bestand proberen te configureren.	27