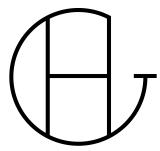


Technische voor- en nadelen van Puppet en Ansible. Verloop en redenen van een omschakeling.

**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 boodt waaronder zaken zoals programmeren, artificiële intelligentie, netwerkbeheer 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 de technische vragen.

Inhoudsopgave

1	Inleiding	9
1.1	Stand van zaken	9
1.1.1	Profiel van Puppet	11
1.1.2	Profiel van Ansible	11
1.2	Opzet van deze bachelorproef	11
1.3	Probleemstelling en Onderzoeksvragen	13
1.3.1	Wat zijn de redenen van een omschakeling?	13
1.3.2	Wat zijn de technische voor-en nadelen van Puppet en Ansible?	13
1.3.3	Wat is het verloop van een dergelijke transitperiode?	13
2	Methodologie	15
2.1	Wat zijn de redenen van een omschakeling?	15

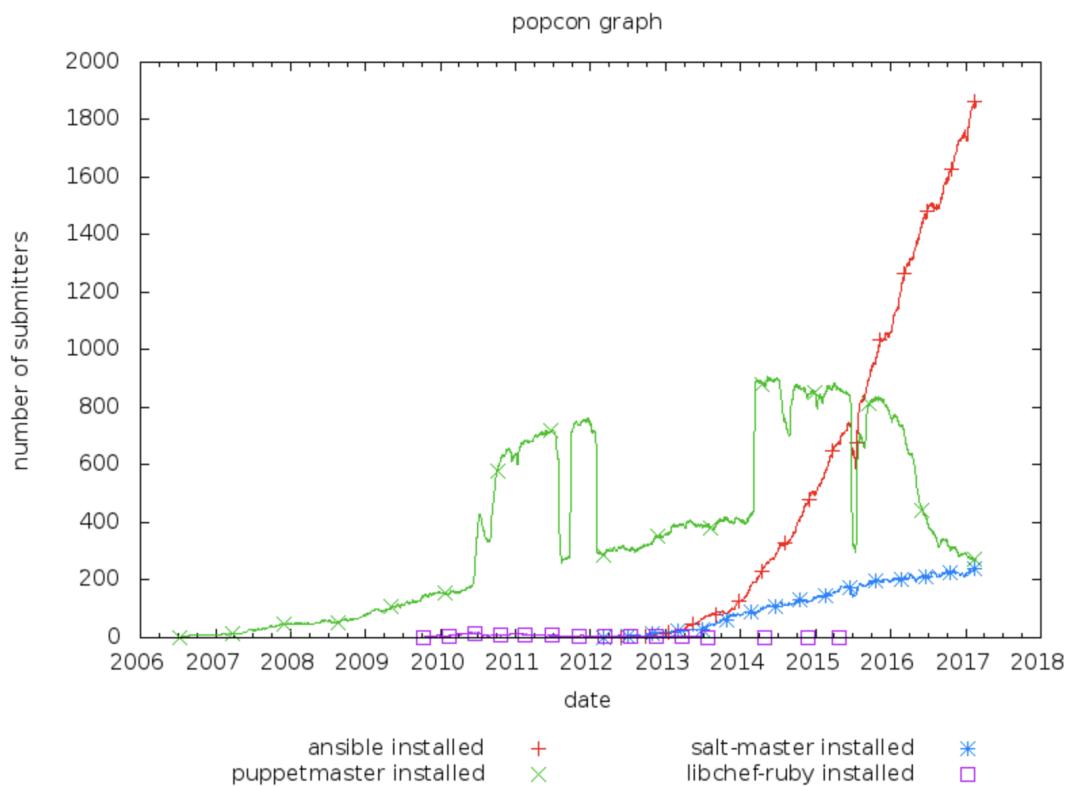
2.2	Technische analyse van Ansible en Puppet?	15
2.2.1	Overzicht van Puppet en Ansible	15
2.2.2	Werking van Puppet	16
2.2.3	Werking van Ansible	16
2.2.4	Performantie	17
2.2.5	Belasting van het netwerk	20
2.2.6	Gebruik van het geheugen	21
2.3	Schaalbaarheid	22
2.3.1	inleiding	22
2.3.2	Tips om de performantie van Ansible te verbeteren	22
2.3.3	Tips om de performantie van Puppet te verbeteren	25
2.4	Wat is het verloop van een dergelijke transitperiode?	25
3	Conclusie	27
4	Bijlagen: ruwe data	33
4.1	Netwerkverkeer	33
4.1.1	Puppet	33
4.1.2	Ansible	34
4.2	Geheugengebruik	35
4.2.1	Puppet	35
4.2.2	Ansible	35
4.3	Deploytijden	36
	Bibliografie	40

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 systeembeheerde 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.-i). 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) 's¹ 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. Maar wat zijn Puppet en Ansible nu eigenlijk?

¹Configuration management tool



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 CMT die werd ontwikkeld in 2005 door Luke Kanies (Puppet, g.d.-f) 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.-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)

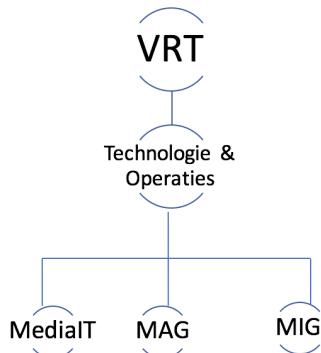
1.1.2 Profiel van Ansible

Michael DeHaan is iemand die zeer vertrouwd was met Puppet. In zijn ervaring vond hij dat mensen moeilijkheden ondervonden op gebied van eenvoud en automatisatie. 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.

1.2 Opzet van deze bachelorproef

De laatste jaren is er een opwaartse trend in de digitalisering van de wereld. Imec deed een onderzoek naar hoe mensen deze digitale bronnen consumeren. Hieruit bleek dat televisie niet langer de alleenheerster is en dat 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 gezamelijk 29,7% halen. Het is vanzelfsprekend dat het mediahuis VRT deze trend moet volgen. Bovendien wordt er een geheel nieuw gebouw verwacht dat ook het datacenter zal herbergen. Zo komt de VRT voor complexe vraagstukken te staan zoals: "hoe groot moet dit datacenter worden en komen er extra locaties bij met back-up servers?". Deze vragen moeten al een antwoord hebben voor de aanvang van het nieuwe gebouw.

Al deze servers zijn van vitaal belang en zorgen voor een correcte werking van het me-



Figuur 1.2: Organigram waarbinnen dit onderzoek zich afspeelt.

diabedrijf. Ze stockeren petabytes aan data en zijn verantwoordelijk voor een correcte uitzending van programma's. Veel afdelingen binnen de VRT maken bovendien gebruik van multi-stage omgevingen zoals testing, staging, productie... Het is dus belangrijk dat een geschikte CMT gebruikt wordt en dat deze perfect geïntegreerd is met de bestaande en toekomstige infrastructuur. In dit onderzoek vallen kleinere CMT's zoals Chef en Salt buiten de scope en zal de focus liggen op Puppet en Ansible.

Dit onderzoek vindt plaats op MediaIT, een afdeling binnen het mediahuis VRT zoals weerspiegeld is op het organigram in figuur 1.2. Het is één van de afdelingen die verantwoordelijk is voor een goede en correcte werking van de servers. Zij gebruiken momenteel Puppet maar deze voldoet niet aan de verwachtingen van de business. Zo is Puppet onder andere niet geïntegreerd met de multi-stage omgevingen wat het testen bemoeilijkt. Verder is er een beperkte functionaliteit voor het monitoren van deploy's en daarom is er dan ook besloten om de huidige Puppet-infrastructuur te vervangen door Ansible.

Deze bachelorproef zal in detail beschrijven en uitleggen wat er precies misgelopen is. Vervolgens zal er gekeken worden of Ansible deze problemen überhaupt kan oplossen en hoe dit dan het best gedaan wordt. Ook zal er een analyse gebeuren die de technische verschillen blootlegt. Dit rapport wil 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.

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 mekaar zakten. Ondertussen heeft Ansible tal van mooie referenties achter zich en heeft het positieve analyses gekregen van belangrijke partijen zoals RedHat en Gartner. Is Ansible echter noemenswaardig beter dan bijvoorbeeld Puppet die reeds een lange bewezen staat van dienst heeft (meer dan 12 jaar) en een grote community die het project ondersteunt?

1.3 Probleemstelling en Onderzoeks vragen

De overschakeling van Puppet naar Ansible is geen kleine stap en kan mogelijk voor veel complicaties zorgen. Daarom weet men best op voorhand wat er te wachten staat en zullen er in dit onderzoek verschillende relevante zaken onderzocht worden die kunnen worden opgedeeld in 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**). 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 optraden. 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 Wat zijn de redenen van een omschakeling?

TO DO

- 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 compatibiliteitproblemen (denk ik) - puppet client niet ouder dan puppet master maar omgekeerd denk ik wel (op te zoeken)

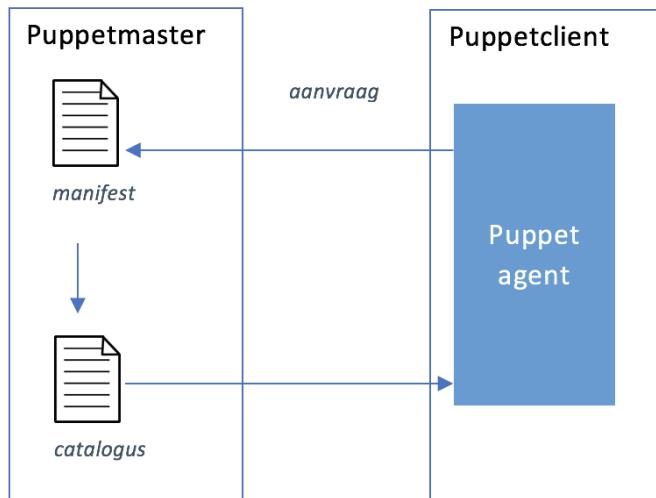
2.2 Technische analyse van Ansible en Puppet?

2.2.1 Overzicht van Puppet en Ansible

	Ansible	Puppet
Programmeerparadigma ^a	declaratief	declaratief
Programmeertaal	YAML	eigen DSL
Communicatieprotocol	SSH	HTTPS
open poorten ^b	22/tcp (client)	8140/tcp (master)

^asynoniemen zijn ook programmeerstijl of programmeermodel, voorbeelden zijn object-georienteerd, procedureel, imperatief..., (Stanchev, Green Jr, & Dimitrov, 2003)

^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)



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.

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 code die je schrijft 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

Ansible maakt geen gebruik van agenten. 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 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 niet aangeraden en 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 (Ansible, g.d.-b).

2.2.4 Performantie

Het is interessant om te weten wat de verhoudingen zijn tussen de deploy-tijd tussen Ansible en Puppet. Om dit op een betrouwbaar mogelijke manier te kunnen verwezenlijken, zijn de configuraties van Ansible en Puppet zo analoog mogelijk gehouden en worden dus dezelfde services geïnstalleerd en geconfigureerd. 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 tot er effectief overgegaan kan worden tot configureren. Hieronder zitten zaken zoals het opstellen van een verbinding, het verzamelen van nodige gegevens en het versturen van een gepersonaliseerde configuratie. Bij Ansible kon deze tijd gewoon berekend worden op basis van de resultaten¹. Bij Puppet is dit echter niet mogelijk en bijgevolg zijn deze resultaten met de hand gemeten.

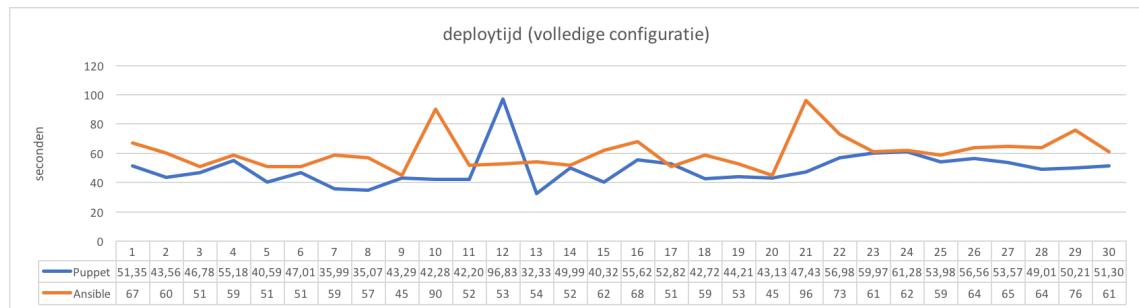
De tweede tijd is de tijd die nodig is om de configuratie effectief uit te voeren. Beide waarden zijn gebaseerd op de feedback van de corresponderende CMT.

Tijd tot het bekomen van een verbinding (connectietijd) (in seconden)

Puppet	9	6	6	11	6	7	8	6	9	9	7	7	8	6	6
	9	10	8	8	7	10	8	6	5	5	5	5	7	6	15
Ansible	9	5	5	7	4	5	6	5	4	11	6	4	8	10	7
	5	6	4	9	5	11	13	13	8	12	8	7	7	8	10

Tijd tot het bekomen van een consistente staat (deploytijd) (in seconden)

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 2.2: Tijd tot het bekomen van een consistente staat, vertrekkende van een 'lege' server.

Deploytijd

Aangezien grafiek 2.2 geen duidelijk verschil toont tussen beide CMT's zal met behulp van de Z-toets aangetoond worden of er al dan niet een statistisch verschil is.

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

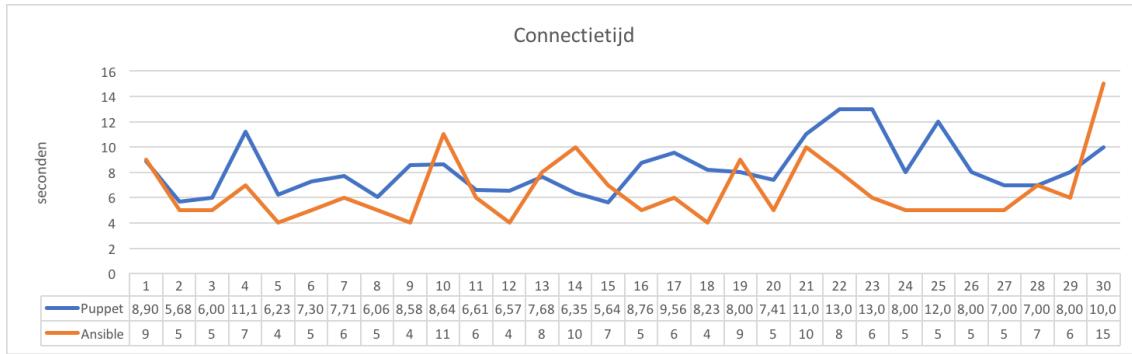
Toetsingsgrootheden

$$\begin{aligned}
 Z &= \frac{\bar{x}_p - \bar{x}_a}{\sqrt{\frac{\sigma_p^2}{n_p} + \frac{\sigma_a^2}{n_a}}} \\
 &= \frac{49,4 - 60,7}{\sqrt{\frac{11,6^2}{30} + \frac{11,5^2}{30}}} \\
 &= -3,789
 \end{aligned} \tag{2.1}$$

Conclusie

Z valt buiten het kritisch gebied waardoor de nulhypotese verworpen kan worden. Bijgevolg wordt aangenomen dat beide gemiddelden niet tot dezelfde verzameling behoren.

¹connectietijd = totaal verstreken tijd - \sum (tijd playbooks)



Figuur 2.3: Tijd tot het initialiseren van een deploy

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 11,3 seconden.

De verschillen worden echter nog groter wanneer de test gedaan wordt met een gedeelte configuuratie. 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 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 deploystijd.

Connectietijd

Ook hier lopen de resultaten door elkaar zoals te zien is op grafiek 2.3, bovendien liggen de gemiddelden hier veel dichter bij elkaar. Om vast te stellen of er een significant verschil is, zal er opnieuw gebruik gemaakt worden van de Z toets.

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

$$\begin{aligned}
 Z &= \frac{\bar{x}_p - \bar{x}_a}{\sqrt{\frac{\sigma_p^2}{n_p} + \frac{\sigma_a^2}{n_a}}} \\
 &= \frac{6,27 - 6,57}{\sqrt{\frac{4,10^2}{30} + \frac{6,11^2}{30}}} \\
 &= 1,27
 \end{aligned} \tag{2.2}$$

Conclusie

Z ligt 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.

2.2.5 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 Ansible Tower hier onmiddellijk van op de hoogte gebracht. Zoals te zien is op grafiek 2.5 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 t6 de communicatie nul is. Ansible had voor die periode niets te melden².

Deze manier van werken is handig tijdens het schrijven 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' in productie niet vaak gebruikt. In realiteit lopen deze jobs tijdens de nacht en is het voldoende om de dag daarna een algemeen overzicht te krijgen.

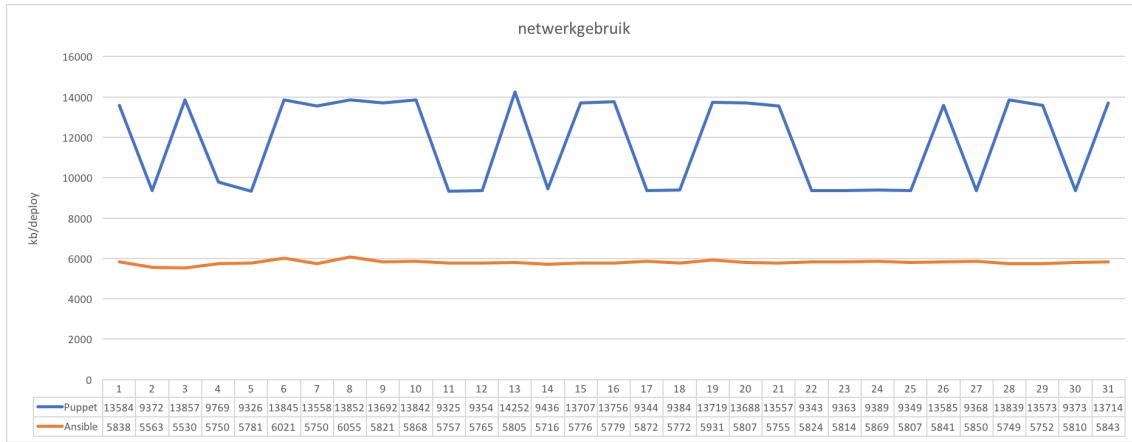
Bij Puppet is dit anders. Hierbij is er enkel communicatie tussen de server en de client bij het begin en op einde van de deploy. Opmerkelijk hierbij is dat er twee types te herkennen zijn. Op figuur 2.5 zijn deze Puppet type A en B genoemd. Bij Puppet type A is te zien hoe de deploy bestaat uit twee reeksen terwijl dit bij type B uit drie reeksen bestaat. De oorzaak van deze derde reeks is niet gekend. 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 konden 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 2.4. 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. Gezamenlijk heeft Puppet een gemiddelde van 11777,90 kilobytes/deploy.

2.2.6 Gebruik van het geheugen

Op grafiek 2.6 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 de Vagrantfile, 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

²Hier betreft het de service MariaDB die werd geïnstalleerd



Figuur 2.4: Totaal verbruikte netwerkcapaciteit per client gedurende het deployen. Dit bevat enkel communicatie tussen master en client.

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.

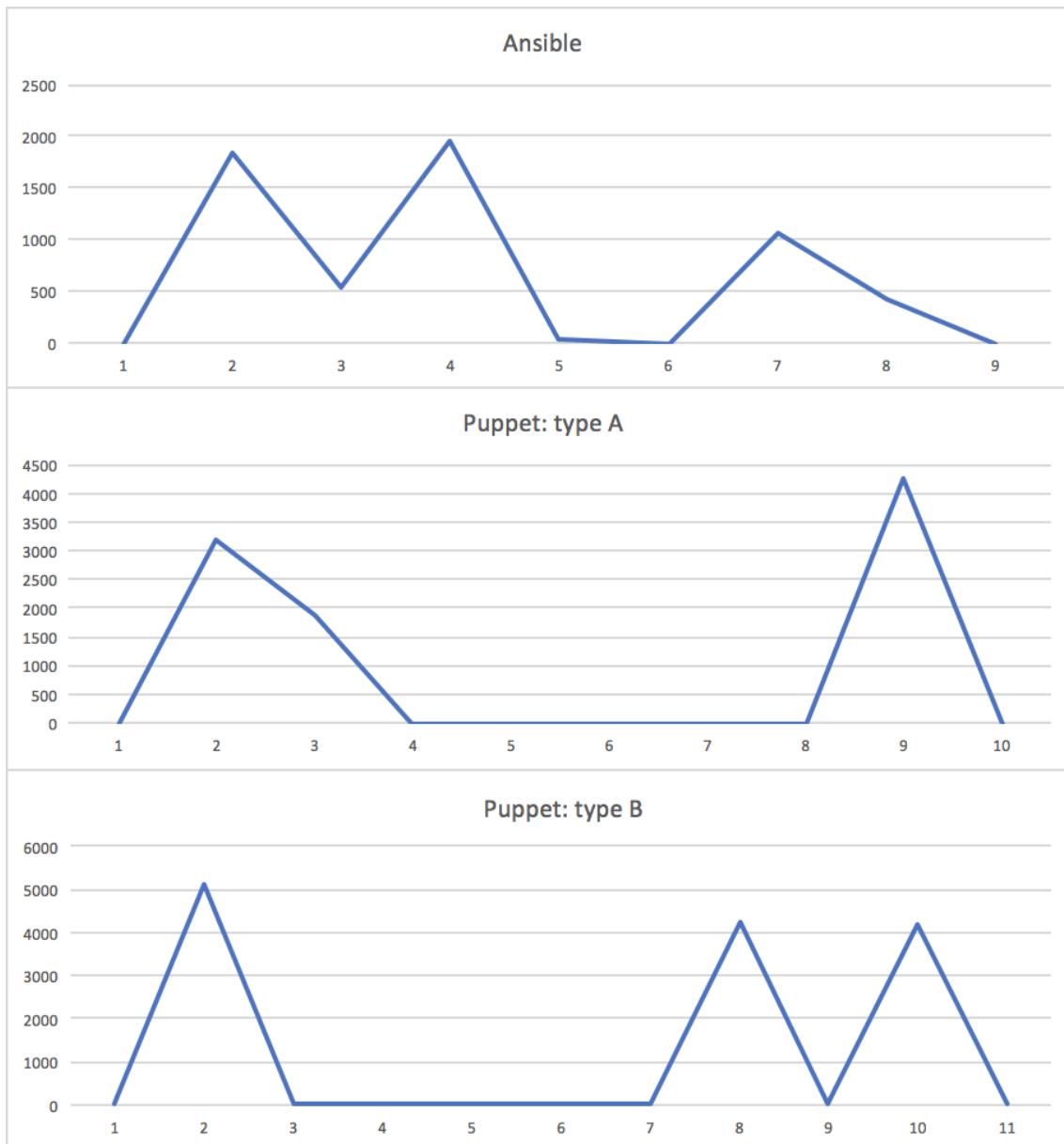
2.3 Schaalbaarheid

2.3.1 Inleiding

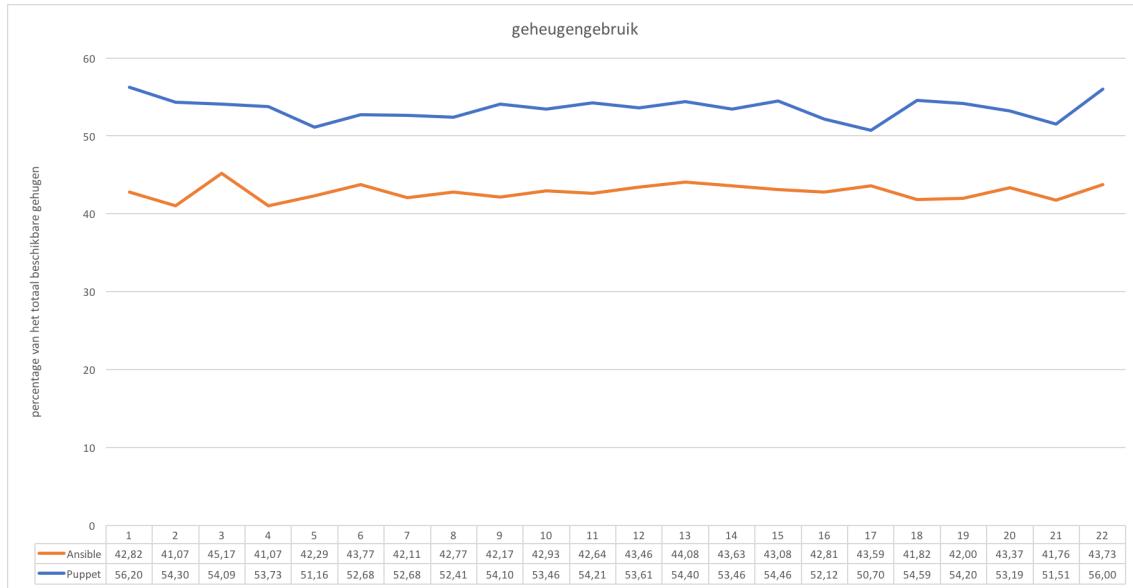
Initieel was het de bedoeling dat het gedrag van de resources gemeten in 2.2 gemeten zouden worden bij toenemende drukte door meer clients toe te voegen. Tijdens het opstellen van deze proef bleek dat dit geen bedrijfs-realistische situatie. Bij Ansible is het namelijk Ansible Tower die bepaald wanneer de clients geüpdatet moeten worden. Bij Puppet valt dit onder de verantwoordelijkheid van de clients zelf. Bijgevolg is de kans klein dat in het geval van Puppet alle clients tegelijk een catalogus zouden aanvragen en hierdoor de Master zwaar belasten. Toch zijn er hier en daar zaken die de performantie ten goede komen. Zo blijft het downloaden van software de grootst 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).

2.3.2 Tips om de performantie van Ansible te verbeteren

Standaard is Ansible geconfigureerd om 5 servers tegelijk te kunnen configureren. Dit aantal kan verhoogt worden naar 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. Dit komt omdat bij het gebruik van `with_items`, Ansible deze packages zal combineren in één transactie block dewelke de performantie ten goede komt.



Figuur 2.5: Drie types van communicatie. Aantal kilobytes per tijdseenheid op een netwerkkaart die uitsluitend bedoeld is voor communicatie met Ansible Tower / Puppetmaster.



Figuur 2.6: Verbruikt percentage van het RAM geheugen. Gemeten bij servers met elk 500 MB.

Zo zou het dus beter zijn om listing 2.1 beter te structureren en te schrijven zoals in listing 2.2.

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).

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-mysql
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

```

2.3.3 Tips om de performantie van Puppet te verbeteren

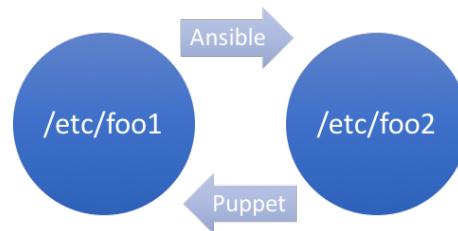
Puppet is door zijn manier van werken van nature meer geloadbalanced dan Ansible, zie 2.3.1, toch zijn ook hier een paar zaken die de performantie ten goede komen. Het aantal aanvragen die Puppet tegelijk kan behandelen varieert van server tot server. Dit is standaard 'num-cpu's - 1' met een minimum van 1 en een maximum van vier. Het aantal gebruikte cpu's staat gelijk aan het aantal aanvragen die Puppet tegelijk kan behandelen. Wanneer er meer aanvragen zijn dan beschikbare CPU's zal het overschot aan aanvragen geblokkeerd worden totdat 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, dit heeft tot gevolg 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.-h).

2.4 Wat is het verloop van een dergelijke transitperiode?

Het is vanzelfsprekend dat de manier van Ansible integreren afhankelijk is van bedrijf tot bedrijf. Toch is dit in het geval van de VRT vlot verlopen. 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. 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 behandelen. Wanneer beide CMT's eenzelfde configuratie uitvoeren kan er door subtiele verschillen een vicieuze cirkel ontstaan.

Beeld u zich de situatie in zoals op figuur 2.7. 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.



Figuur 2.7: Vicieuze cirkel van twee CMT's hetzelfde bestand proberen te configureren.

1. Puppet configureert de service met bestand foo2 en start deze op.
2. Eventjes later stelt Ansible vast dat de configuratie niet meer overeenkomt met deze beschreven in het playbook. Hij wijzigt foo2 naar foo1 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 zorgt voor onnodige aanpassingen met eventueel ongewenste of onverwachte gevolgen zoals het voortdurend herstarten van een service.

Twee CMT's die dezelfde server op hetzelfde moment configureren zouden elkaar niet mogen hinderen. Ten hoogste duurt de configuratie iets langer omdat er een lock bestaat op de Package managers waardoor de ene CMT op de andere dient te wachten.

3. Conclusie

TO DO

Acroniemen

CMT Configuration management tool. 9, 11–13, 17–20, 25, 26, 41

Verklarende Woordenlijst

Package manager Een mechanisme die het mogelijk maakt om software te installeren op UNIX gebaseerde systemen. (*voorbeelden: yum, apt, dpkg.....* 26

4. Bijlagen: ruwe data

4.1 Netwerkverkeer

4.1.1 Puppet

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	gemiddelde
0	3270	1873	0	0	0	0	4259	0	4182	0	0	0	13584
0	5115	0	0	0	0	0	0	4257	0	0	0	0	13584
0	5079	0	0	0	13	0	0	0	8765	0	0	0	9372
0	3248	2228	0	0	13	0	4280	0	0	0	0	0	13857
0	3203	1870	0	0	0	0	0	4253	0	0	0	0	9769
0	3202	1875	0	0	13	4278	0	4477	0	0	0	0	9326
0	5102	0	0	0	0	4257	0	4199	0	0	0	0	13845
0	5092	0	0	0	0	0	8760	0	0	0	0	0	13558
0	3222	1858	0	0	0	0	8612	0	0	0	0	0	13852
0	5063	0	0	0	0	0	8779	0	0	0	0	0	13692
0	5085	0	0	0	13	0	0	4227	0	0	0	0	13842
0	5080	0	0	0	13	0	4261	0	0	0	0	0	9325
0	5074	0	0	13	0	0	5156	4009	0	0	0	0	9354
0	1641	3512	0	0	13	0	4270	0	0	0	0	0	14252
0	5084	0	0	0	13	0	4272	875	3463	0	0	0	9436
0	5158	0	0	0	13	0	4248	4337	0	0	0	0	13707
0	3210	1864	0	0	13	0	4257	0	0	0	0	0	13756
0	5086	0	0	13	0	0	4285	0	0	0	0	0	9344
0	5087	0	0	13	0	4281	875	3463	0	0	9384	0	13719
0	5059	0	0	13	0	4272	0	4344	0	0	0	0	13719

0	5097	0	0	13	0	4247	875	3325	0	0	0	0	0	13688
0	3011	2054	0	0	13	0	0	4265	0	0	0	0	0	13557
0	5077	0	0	13	0	0	0	4273	0	0	0	0	0	9343
0	5091	0	0	13	0	4285	0	0	0	0	0	0	0	9363
0	3210	1870	0	0	13	4256	0	0	0	0	0	0	0	9389
0	5092	0	0	13	0	0	5163	3317	0	0	0	0	0	9349
0	5084	0	0	13	0	0	4271	0	0	0	0	0	0	13585
0	5076	0	0	13	0	4268	4482	0	0	0	0	0	0	9368
0	5093	0	0	13	0	0	5134	3333	0	0	0	0	0	13839
0	5072	0	0	13	0	0	0	4288	0	0	0	0	0	13573
0	5082	0	0	13	0	4285	4334	0	0	0	0	0	0	9373

4.1.2 Ansible

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	gemiddelde
0	1811	0	1133	1401	0	0	0	0	1071	422	0	0	5838
0	1838	0	2302	0	0	1371	52	0	0	0	0	0	5563
0	1832	0	2248	0	0	0	663	787	0	0	0	0	5530
0	1852	2153	316	0	0	0	1429	0	0	0	0	0	5750
0	1830	619	1877	0	0	1455	0	0	0	0	0	0	5781
0	1858	2713	0	0	762	688	0	0	0	0	0	0	6021
0	1830	490	1955	0	0	1428	47	0	0	0	0	0	5750
0	1824	0	2748	0	0	1431	52	0	0	0	0	0	6055
0	1855	168	2315	0	0	360	1123	0	0	0	0	0	5821
0	1826	168	2098	342	0	0	1434	0	0	0	0	0	5868
0	1859	1792	664	0	0	1442	0	0	0	0	0	0	5757
0	1865	0	2466	0	0	1375	59	0	0	0	0	0	5765
0	1846	168	2355	0	0	0	1436	0	0	0	0	0	5805
0	1819	498	1971	0	0	1428	0	0	0	0	0	0	5716
0	1852	2474	0	0	0	1450	0	0	0	0	0	0	5776
0	1832	168	2312	0	0	358	1109	0	0	0	0	0	5779
0	1852	0	930	1629	0	0	0	1461	0	0	0	0	5872
0	419	1452	909	1566	0	0	1426	0	0	0	0	0	5772
0	1865	0	883	1692	0	0	0	1109	382	0	0	0	5931
0	1879	0	2458	0	0	0	1048	422	0	0	0	0	5807
0	1826	2234	252	0	0	0	462	981	0	0	0	0	5755
0	1844	0	2530	0	0	0	1391	59	0	0	0	0	5824
0	1865	168	2317	0	0	0	1464	0	0	0	0	0	5814
0	1831	413	2096	0	0	0	1482	47	0	0	0	0	5869
0	1839	0	2484	0	0	0	1095	389	0	0	0	0	5807
0	1872	1444	1091	0	0	1434	0	0	0	0	0	0	5841
0	1889	505	2031	0	0	0	0	1425	0	0	0	0	5850
0	1859	168	2293	0	0	1015	414	0	0	0	0	0	5749
0	1845	0	858	1599	0	0	796	654	0	0	0	0	5752

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

4.2 Geheugengebruik

4.2.1 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				

4.2.2 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	

4.3 Deploytijden

Deploytijd		Full config		Partial config		No config	
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>
- Debian. (2017). *Debian Popularity Contest*. Verkregen van <http://popcon.debian.org>
- Geerling, J. (2016). *Ansible For DevOps*. Leanpub.
- Imec. (2017). digimeter 2016.
- 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). *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?language=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>

- 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.
- 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)	10
1.2 Organigram waarbinnen dit onderzoek zich afspeelt.	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.	16
2.2 Tijd tot het bekomen van een consistente staat, vertrekende van een 'lege' server.	18
2.3 Tijd tot het initialiseren van een deploy	19
2.4 Totaal verbruikte netwerkcapaciteit per client gedurende het deployen. Dit bevat enkel communicatie tussen master en client.	22
2.5 Drie types van communicatie. Aantal kilobytes per tijdseenheid op een netwerkkaart die uitsluitend bedoeld is voor communicatie met Ansible Tower / Puppetmaster.	23
2.6 Verbruikt percentage van het RAM geheugen. Gemeten bij servers met elk 500 MB.	24
2.7 Vicieuze cirkel van twee CMT's hetzelfde bestand proberen te configureren.	25