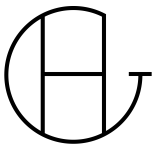


HoGent

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

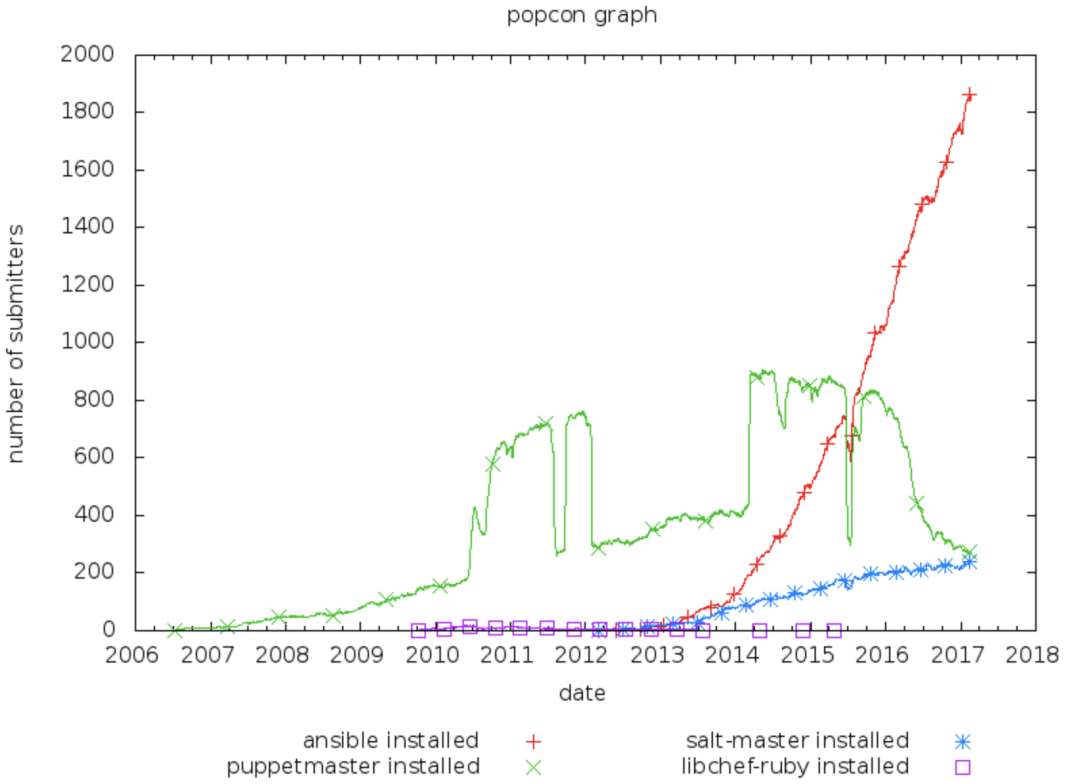
Inhoudsopgave

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 (**PuppetDeclaratief**). 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 CMT 's[1] 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' (**coolvendors**). Verder was het Red Hat die aankondigde dat er een akkoord was om Ansible over te nemen (**redhatovername**). Grafiek 1.1 toont hoevaak 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?

[1] Configuration management tool



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

1.1.1 Profiel van Puppet

Puppet is een open source CMT die werd ontwikkeld in 2005 door Luke Kanies (**PuppetLeaders**) met als doel om op een betrouwbare mannier datacenters te kunnen automatiseren en te controleren. Dit zou het hele proces van services installeren moeten versnellen om zo tijd te winnen(**how-puppet-works**). Dit kan zowel gaan om Linux servers als Windows servers (**PuppetForWindows**). 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. (**Puppetdoc**) (**puppetfaq**)

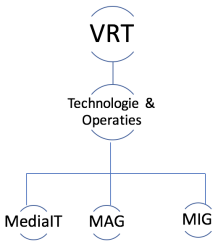
1.1.2 Profiel van Ansible

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 configuratie beheer, eenvoudig deployen van nieuwe servers en als het nodig was de mogelijkheid boodt tot ad-hoc commando's. Met dit idee is hij samen met met Saïd Ziouani in 2012 het open source project Ansible gestart (**ansiblefordevops**). 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 addionele 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 opwaarste trend in de digitaliserening van de wereld. Imec deed een onderzoek naar hoe mensen deze digitale bronnen consumeren. Hieruit bleek dat televie niet langer de alleenheerser is en dat steeds meer programma's worden bekeken via site's en app's. Zo is de populariteit van de televiesie als favoriete 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 (**digimeter**). Het is vanzelfsprekend dat het mediahuis VRT deze trend moet volgen. Bovendien wordt er een geheel nieuw gebouw verwacht dewelke 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 mediabedrijf. Ze stockeren petabytes aan data en zijn verantwoordelijk voor een correcte



Figuur 1.2: Organigram waarbinnen dit onderzoek zich afspeelt.

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 de VRT zoals weerspiegeld is op het organigram in figuur 1.2. Zij zijn één van de afdelingen verantwoordelijk voor een goede en correcte werking van de servers. Zij gebruiken momenteel Puppet maar deze voldoet niet aan de verwachtingen van de bussiness. 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 detaïl 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 beste 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 diens heeft (meer dan 12 jaar) en een grote community die het project ondersteunt?

1.3 Probleemstelling en Onderzoeksvragen

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 compatiblieteistproblemen (denk ik)

2.2 Technische werking van Ansible en Puppet?

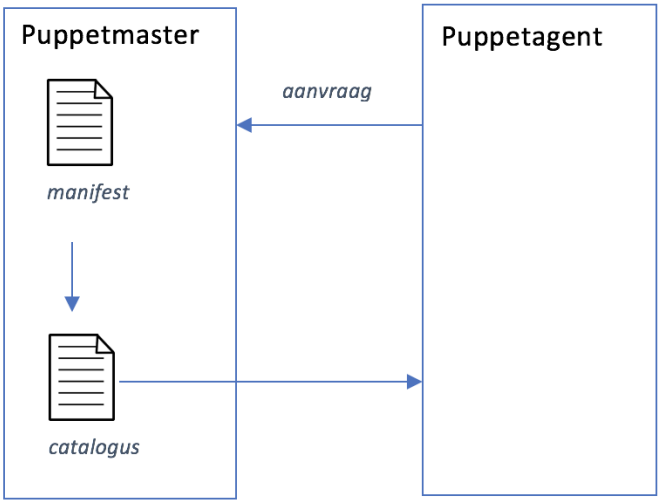
2.2.1 Overzicht van Puppet en Ansible

	Ansible	**Puppet**
Programmeerparadigma[1]	declaratief	declaratief
Programmeertaal	YAML	eigen DSL
Communicatieprotocol	SSH	HTTPS
open poorten[2]	22/tcp (client)	8140/tcp (master)

(**languagePuppet**)(**masterproef**) (**ansibledoc**)

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



Figuur 2.1: aanvraag van een catalogus bij de Puppetmaster door een Puppetclient.

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 (**Puppetlanguagecatalog**). 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 (**Puppetdoc**).

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 met 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 (**ansibledoc**).

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 zo een betrouwbare mannier te kunnen verwezelijken zijn de configuraties van Ansible en Puppet zo analoog mogelijk gehouden en worden dezelfde services geïnstalleerd en geconfigureerd. Vervolgens word elke configuratie 30 keer uitgevoerd. De tijd wordt onderverdeelt in twee tijden.
De eerste tijd is de tijd die het kost om een verbinding aan te gaan tussen de master en de client en het versturen van de configuratie. Bij Ansible kon deze tijd gewoon berekend worden op basis van de resultaten[3], bij Puppet was dit 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 gebasseerd op de feedback van de CMT.

Tijd tot het bekomen van een verbinding (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

Hypothese

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

Significantieniveau en waarden
$\alpha = 0.05 => -1.96 en + 1.96$

	Puppet	Ansible
x	60.7	49.4
σ	11.5	11.6
n	30	30

[3] total elapsed time - elapsed time from role(s)

Toetsingsgrootheiden

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

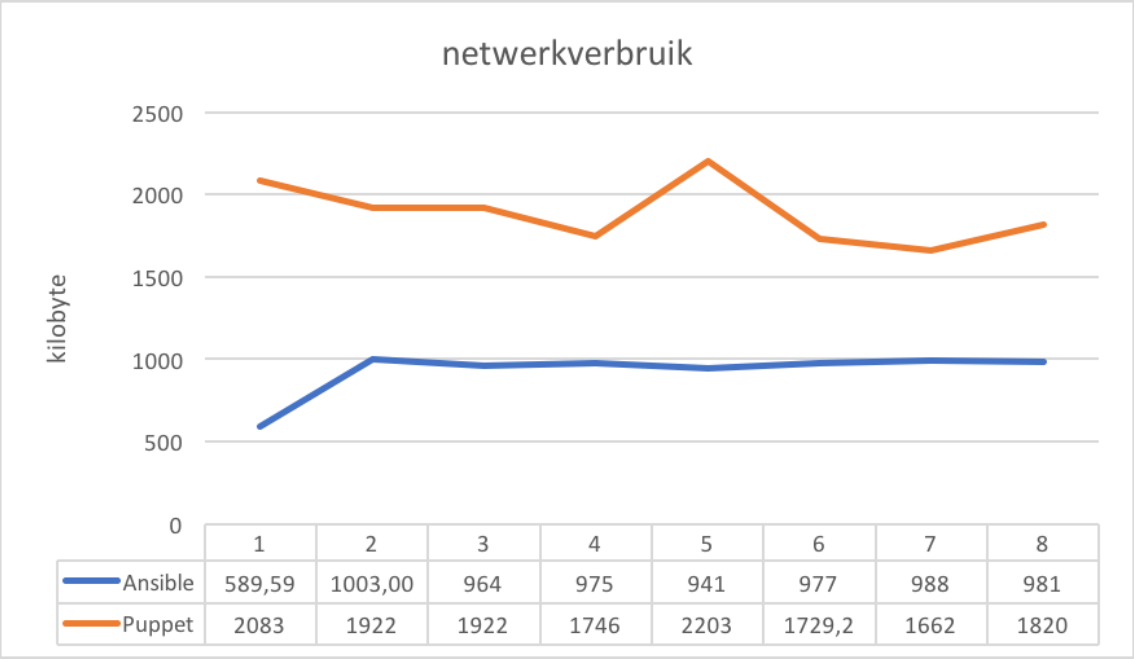
Conclusie
Z valt buiten het kritisch gebied waardoor de nulhypotese verworpen kan worden. Bijgevolg wordt aangenomen dat beide gemiddelde niet tot dezelfde verzameling behoren en dat Ansible gemiddeld trager is dan Puppet.

2.2.5 Belasting van het netwerk

Ansible en Puppet hebben een groot verschil in de manier van communiceren en dit weerspiegeld zicht in het gedrag van de CMT. Op afbeelding 2.3 bevinden zich links alle Ansible clients, te herkennen aan hun naam die begint met een A. Rechts staan alle puppet clients, te herkennen aan de PP. De grafieken 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 word 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 word verstuurd.

De mannier van communicatie is te herkennen in de grafieken. Zo onderhoud Ansible de communicatie met de client gedurende de deploy. Hiermee wordt bedoelt dat Ansible op de hoogte is van de laatste stand van zaken op de client. Wanneer een bepaalde taak voltooid is wordt Ansibel Tower hier onmiddelijk van op de hoogte gebracht. Op afbeelding 2.3 komt deze mannier van werken duidelijk terug op grafiek Anode1 en Anode2. Hier is te zien hoe het netwerk voordurend belast wordt. Ook grafiek Anode3 - Anode5 zijn volgens deze logica te verklaren. Ansible verstuurd namelijk enkel data als de verandering voltooid is. Er worden dus geen onnodige berichten verstuurd. De 'val' in de grafiek Anode3 - Anode5 is dus te verklaren een trage download van een bepaalde service[4]. De download duurde langer dan een minuut waardoor er een minuut lang niets te rapporteren viel. Deze mannier van werken is handig tijdens het schrijven van nieuwe Ansible rollen. Je krijgt namelijk live feedback tijdens het uittesten. Een nadeel hieraan is dat het netwerk geen

[4]Hier betreft het de service MariaDB



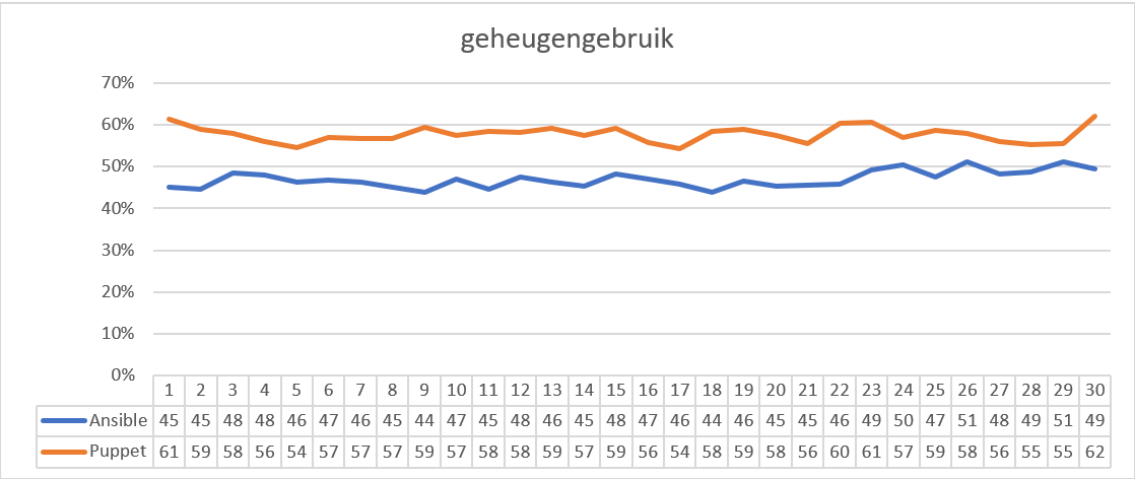
Figuur 2.2: Totaal verbruikte netwerkcapaciteit per client gedurende het deployen. Dit bevat enkel communicatie tussen master en client.

rust krijgt. Bovendien word 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 erop een algemeen overzicht te krijgen.

Bij puppet is dit anders. Hierbij is er enkel communicatie tussen de server en de client op het begin en het einde van de deploy. Dit komt duidelijk terug in PPnode1, PPnode4 - PPnode6. Ook grafiek PPnode3 is op deze mannier te verklaren. Hier ging het downloaden van de services namelijk veel vlugger, waardoor na 1 minuut het bericht van voltooid al verstuurd kon worden. Een nadeel hieraan is dat er op de master geen live feedback voor testen gevolgd 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?
TO DO: wat is dit zogenaamde bericht van voltooid? Worden er bij Puppet wel log files terug naar de master gestuurd? te onderzoek en te vragen aan Pieter

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 2.2. Hier heeft Ansible een gemiddelde van 927,32 kilobytes/deploy en Puppet 1885,9 kilobytes/deploy. Belangrijk om te weten is dat er getracht is geweest het playbook van Ansible en het manifest van Puppet zo gelijkaardig mogelijk te houden.

Figuur 2.3: Aantal kilobytes per minuut op een netwerkkaart die uitsluitend bedoelt voor communicatie met Ansible Tower / Puppetmaster.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Ansible	45	45	48	48	46	47	46	45	44	47	45	48	46	45	48	47	46	44	46	45	45	46	49	50	47	51	48	49	51	49
Puppet	61	59	58	56	54	57	57	57	59	57	58	58	59	57	59	56	54	58	59	58	56	60	61	57	59	58	56	55	55	62

Figuur 2.4: Verbruikt percentage van het RAM geheugen. Gemeten bij servers met elk 500 MB.

2.2.6 Gebruik van het geheugen

Op grafiek 2.4 is per tijdseenheid het gemiddelde gebruikte 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 Puppet agent is werd vermoed dat het verschil hier aan te wijten is. Dit vermoeden werd gestaafd toen de puppet agent tijdelijk uitgezet werd. Het ramgeheugen daalde onmiddelijk naar gelijkwaardige waarden als deze van de Ansible client. Zonder configuratie hebben Puppet clients een gemiddeld verbruik van 58% gebruikt geheugen. Bij Ansible is dit 47%. Dit betekend dat met een verschil van 11% er bij 500 MB, 55MB gebruikt wordt aan Puppet.

2.3 Wat is het verloop van een dergelijke transitperiode?

3. Conclusie

TO DO

Lijst van figuren

Lijst van tabellen