HO
GENT

Faculteit Bedrijf en Organisatie

Onderzoek naar samenwerking tussen Cloud-Init en Ansible

Maarten De Smedt

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

Promotor:
Lotte Van Steenberghe
Co-promotor:
Simon Lepla

Instelling: Be-Mobile

Academiejaar: 2018-2019

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Onderzoek naar samenwerking tussen Cloud-Init en Ansible

Maarten De Smedt

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

Promotor:
Lotte Van Steenberghe
Co-promotor:
Simon Lepla

Instelling: Be-Mobile

Academiejaar: 2018-2019

Tweede examenperiode

Woord vooraf

Deze bachelorproef rondt mijn opleiding Toegepaste Informatica aan de HoGent af.

Als onderwerp voor deze bachelorproef heb ik besloten om 2 technologieën, Ansible en cloud-init, met elkaar te vergelijken en na te gaan of een samenwerking mogelijk is. Het bedrijf Be-Mobile heeft me op dit idee gebracht. Zij hadden gehoord van een nieuwe technologie cloud-init en vroegen zich af of dit hun huidige tool, Ansible, kon vervangen en, of dat deze konden samenwerken. Ik vond dit een zeer interessant onderwerp, aangezien ik al bekend was met Ansible. Na een klein onderzoek naar cloud-init heb ik besloten om hierover mijn bachelorproef te maken.

Deze bachelorproef is een werk waar ik persoonlijk wel trots op ben. Maar dit was niet mogelijk geweest zonder de hulp van een aantal mensen. Graag zou ik ze hier even willen bedanken.

Eerst en vooral zou ik graag het bedrijf Be-Mobile en in het bijzondere mijn co-promoter Simon Lepla willen bedanken. Ze gaven mij de nodige middelen om de bachelorproef goed uit te voeren. Bij vragen over technische zaken kon ik ook atlijd bij hen terecht. Graag zou ik ook mijn promoter Lotte Van Steenberghe van de HoGent willen bedanken. Ze heeft mij goed begeleid tijdens deze periode. Ook zou ik graag mijn familie willen bedanken voor hun motivatie en in het bijzonder mijn moeder. Zij heeft tijd vrijgemaakt om mijn bachelorproef na te lezen op eventuele spelfouten.

Ik hoop dat de inhoud van mijn bachelerproef u iets kan kan bijleren!

Maarten De Smedt,
Academiejaar 2018-2019

Samenvatting

Cloud-init en Ansible zijn beide server management configuration tools. Dit zijn tools die servers kunnen configureren. Terwijl Ansible toch een grote speler en bekende naam is in dit terrein, is cloud-init dit toch niet. Hier wordt een onderzoek gevoerd om te bekijken of cloud-init Ansible overbodig kan maken of is het mogelijk dat deze kunnen samenwerken?

Dit werk is gemaakt in samenspraak en in opdracht van het bedrijf Be-Mobile. Zij werkten al een tijd met Ansible om servers te configureren maar botste op een nieuwe technologie: cloud-init. Hun vraag was of de technologie nuttig was in hun setup. Na hier onderzoek over te doen, was er nog geen expliciet antwoord over te vinden. Hierdoor werd dit onderzoek gestart.

Het onderzoek is opgedeeld in 2 delen. Allereest het theoretische gedeelte. Hierin worden Ansible en cloud-init apart bekeken en besproken. Erna werden 2 artikels, *An introduction to server provisioning with CloudInit* en *Using Ansible to Bootstrap My Work Environment Part 4*, besproken. Deze artikels gingen over Ansible en cloud-init. Het tweede deel is het praktisch gedeelte, hier worden verschillende testen uitgevoerd. Bij alle testen worden er servers opgezet met bepaalde configuraties in Hetzner Cloud, een cloud provider. Er wordt op 3 manieren configuraties uitgevoerd: met cloud-init, met Ansible en met Ansible en cloud-init. Er worden 4 soorten configuraties gedaan: de basis configuraties (installeren packages, aanmaken users,...), server configuraties (een LAMP en MySQL server), configuratie na start-up (bekijken hoe er via het script een extra aanpassing kan worden gedaan) en containerization (docker containers op de server). Per hoofdstuk worden de configuratie bestanden opgesteld en de resultaten besproken. Er wordt vooral gekeken naar de complexiteit van de bestanden en de doorlooptijd van de tools.

Het resultaat is niet het geen dat initieel werd verwacht. Het verwachtte resultaat was dat elke server zijn eigen verhaal ging hebben, en dit was toch niet. Uiteindelijk was cloud-init in basisconfiguraties globaal gezien beter. Maar vanaf de iets geavanceerde configuraties was Ansible of “cloud-init en Ansible” beter aangewezen. Voor configuraties na de start-up was Ansible beter. Cloud-init ondersteunde deze functie niet in de manier dat er hier mee gewerkt werd.

De conclusie was dus duidelijk cloud-init kan alleen gebruikt worden voor basis configuraties. Maar vanaf meer geavanceerde configuraties, is het best om over te schakelen naar Ansible of Ansible bij cloud-init te betrekken. Er zijn twee vragen die na dit onderzoek kunnen worden gesteld, waar er in de toekomst onderzoek naar kan worden gedaan. Heeft een andere cloud-provider hetzelfde resultaat als Hetzner Cloud? En heeft een andere aanroeping van de scripts een effect op het resultaat?

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

1. Inleiding

In dit hoofdstuk wordt er een korte inleiding over de bachelorproef gegeven. De oorsprong van het idee en de onderzoeksvraag wordt besproken. Ook wordt er al wat basisinfo gegeven over het onderwerp.

1.1 Achtergrond

De installatie en modificatie van software servers moet voor de gebruiker altijd makkelijker en sneller. Eens de gebruiker weet wat voor server hij wil, wil hij deze liefst zo snel mogelijk opzetten met de nodige specificaties. Of als de gebruiker een kleine aanpassing wil doen aan de server, wil hij dit zo makkelijk mogelijk kunnen aanpassen.

Om dit zo efficiënt mogelijk te doen heb je configuration management tools nodig. Dit zijn tools die gemaakt zijn om software op servers te installeren en te beheren. De meest bekende tools zijn Chef, Puppet, Salt en Ansible. In deze bachelorproef gaat het onder andere over Ansible.

1.2 Context

Ansible is op zich al een zeer goede configuration management tool, maar voor het bedrijf Be-Mobile nog niet efficiënt genoeg.

Be-Mobile is een Big Data verkeersbedrijf. Be-Mobile wil verkeer revolutioneren en de mobiliteits oplossingen voor vandaag en morgen creëren. Hun hoofdzetel ligt in Melle.

Be-Mobile werkt met "Hetzner Cloud". Dit is hun cloud server provider. Met hetzner cloud heb je de optie om te verwijzen naar een cloudconfig file om je server te configureren. Het cloudconfig bestand is het configuratie bestand van cloud-init.

Cloud-init is net zoals Ansible een type van configuration management tool maar speciaal voor cloud servers. Door middel van paramaters in te vullen in dit bestand kan je jouw server configureren.

1.3 Probleemstelling - Onderzoeksvraag

Het probleem is meteen heel duidelijk. Moet er worden overgestapt naar cloud-init in plaats van Ansible. In deze bachelorproef gaat dit worden onderzocht. Waar zijn Ansible en cloud-init verschillend, waar zijn ze hetzelfde en waar vullen ze mekaar aan. De echte onderzoeksvragen waar deze thesis een antwoord op hoopt te vinden is:

- Is Ansible overbodig door het gebruikt van Cloud-init?
- Zijn Ansible en cloud-init compatibel?
- Op welke manier zijn ze compatibel of overbodig?

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een inleiding gegeven tot Ansible en cloud-init

In Hoofdstuk 3 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 4 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 5 wordt getoond hoe de testomgeving is opgezet.

In Hoofdstuk 6, 7, 8 en 9 wordt het effectieve onderzoek uitgevoerd.

In Hoofdstuk 10, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

Er zijn ook 2 bijlagen aanwezig. Het eerste is het Onderzoeksvoorstel voor deze bachelorproef. Het tweede is een Verklarende Woordenlijst. Hier worden andere technische termen die in de bachelorproef voorkomen kort uitgelegd.

2. Inleiding tot Ansible en cloud-init

Dit hoofdstuk is een inleiding over cloud-init en Ansible. Eerst wordt Ansible besproken. Er wordt eerst wat algmene uitleg gegeven, daarna wordt iets dieper op ingegaan op de architectuur, playook, roles en de omgeving. Erna wordt er voor cloud-init hetzelfde gegaan. Eerst wordt er wat algemene uitleg gegeven. Daarna wordt er dieper ingegaan op de modules, Cloud config, omgevingen en systemen.

2.1 Ansible

Ansible is een open source IT configuration management en deployment tool. Ansible hun grote doel is om besturingssysteem configuratie en de implementatie van software te vergemakkelijken. Door dit allemaal onder 1 systeem te steken. De informatie werd gevonden met behulp van het document *Ansible In Depth* (RedHat, 2017).

Ansible staat bekend als een systeem dat makkelijk te leren is als IT administrator, ontwikkelaar of manager. Het probeert er voor te zorgen dat het makkelijk te verstaan is en makkelijk om zelf op te bouwen. Zo kunnen nieuwe gebruikers dit eenvoudig en snel oppikken.Ze proberen uniek te zijn in toegankelijkheid voor gebruikers. Ze willen dat er veel aanpassingsmogelijkheid is voor de expertgebruikers. Maar ook zeer toegankelijk en eenvoudig voor nieuwe gebruikers.

2.1.1 Architectuur

Eén van de belangrijkste verschillen tussen Ansible en andere configuratie management tools, is zijn architectuur. Ansible gaat uit van het "push" model. Ook is er geen additionele software nodig om machines bruikbaar te maken voor Ansible. Het heeft geen extra gebruikers of referenties nodig om te draaien. Het gebruikt gewoon de informatie die de user meegeeft. Daarbij hoort ook dat Ansible geen administrator of sudo toegang nodig heeft. Ansible wordt standaard bestuurt door een remote computer.

Dit zorgt ervoor dat Ansible veiliger wordt. Alleen de informatie die de gebruiker meegeeft wordt gebruikt. Een gebruiker die wel toegang heeft tot de server, maar niet tot de remote computer, kan geen aanpassingen pushen.

2.1.2 Playbook - Roles

Ansible voert de automatisatie en deployment uit via playbooks. Dit zijn yaml bestanden die beschrijven hoe de automatisatie moet verlopen.

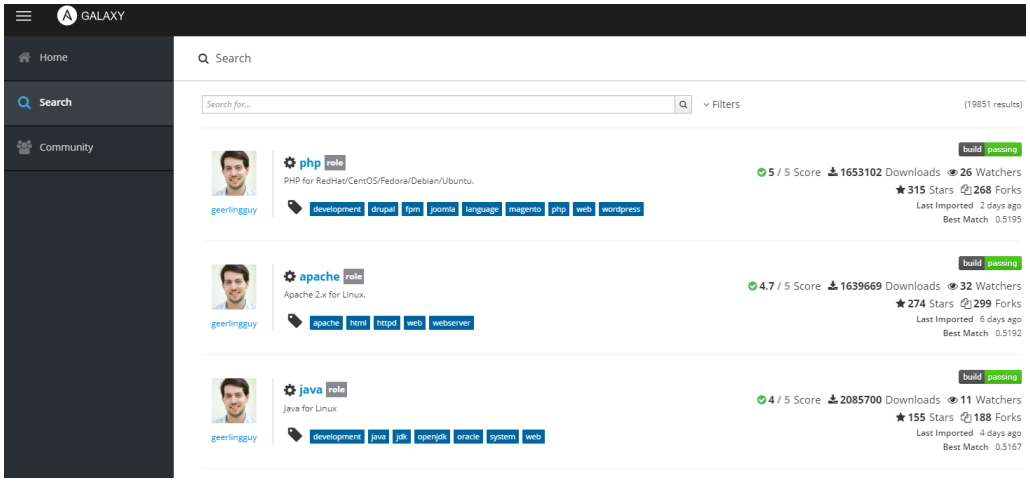
Deze playbooks bevatten verschillende "plays" die de automatisatie definiëren over verschillende hosts. Deze hosts staan bekend als de inventory. Elke "play" bevat verschillende taken die één, enkele of alle hosts moeten uitvoeren. Elke taak roept een Ansible module aan, een klein stukje code dat een specifieke taak uitvoert. Deze kunnen zeer simpel zijn, een bestand op een machine zetten of een specifieke package installeren. Maar ze kunnen ook complex zijn zoals een gehele CloudFormation opstarten in Amazon EC2. Figuur 2.1 is een voorbeeld van een playbook.

```
1  ---
2  - hosts: webservers
3    sudo: yes
4
5    vars:
6      app_name: PleaseDeployMe
7      repo_url: https://github.com/username/repo_name.git
8      repo_remote: origin
9      repo_version: master
10     webapps_dir: /deployed
11     virtualenv_root: /deployed/PleaseDeployMe/mac
12   tasks:
13
14     - name: git pull project
15       git: repo={{repo_url}} dest={{webapps_dir}}/{{app_name}} version=master
16
17       notify:
18         - restart app
19
20     - name: install things
21       pip: name=virtualenv
22
```

Figuur 2.1: Voorbeeld van een Ansible playbook. (Irvin, 2016)

Ansible is geschreven zodat als ze de playbook uitvoeren ook checken of deze task nog moet gedaan worden. Bijvoorbeeld als een Ansible taak is om een webserver op te starten, zal Ansible deze alleen uitvoeren als de webserver nog niet is opgestart. Dit staat bekend als idempotentie. Het zorgt ervoor dat de configuratie altijd snel en efficiënt wordt uitgevoerd.

Met Ansible kunnen taken ook ingekapseld worden in een role. Dit wordt gebruikt als er een specifieke configuratie meerdere keren wordt uitgevoerd, bijvoorbeeld het opzetten van een webserver. De Ansible Galaxy site bevat veel roles die kunnen gebruikt en aangepast worden voor het gebruik in een playbook. Figuur 2.2 is een screenshot van de Ansible Galaxy site.



Figuur 2.2: Ansible Galaxy site met lijst van roles.

2.1.3 Omgevingen

Ansible is even makkelijk te deployen in publieke of private cloud omgevingen, als in een lokale omgeving. Voor publieke of private cloud providers kan er gekozen worden voor: Amazon Web Services, Microsoft Azure, Rackspace,... Maar er kan ook op lokale infrastructuren gewerkt worden door middel van virtuele machines. Tools die hiervoor worden gebruikt zijn: VirtualBox, VMWare,...

2.2 Cloud-Init

Net als Ansible is cloud-init ook een type van configuration manager en deployment tool. Op de site van cloud-init (Moser & Harlow, 2019) wordt het ontstaan beschreven. Het is ontwikkeld en uitgebracht als gratis software met de open source licentie en ook de Apache Version 2.0 licentie. De makers, Scott Moser en Joshua Harlow, hebben het in python geschreven.

Cloud-init zorgt voor de customisaties tijdens het opstarten van de cloud of virtuele instanties. Deze service gebeurt heel vroeg in het boot proces. De instantie zoekt naar de *user data* (het cloud-init script) van de gebruiker en voert deze uit.

Veel van de punten die hier verder worden besproken zijn gevonden in de presentatie van (Skinner & Heldebrant, 2017).

2.2.1 Modules

Cloud-init heeft 6 hoofdpijlers waar het modules voor gebruikt, namelijk: schijf configuratie, commando's uitvoeren, gebruikers en groepen creëren, beheren van packages, content bestanden schrijven en bootstrappen van Chef en/of Puppet. Er zijn nog andere modules maar dit zijn de 6 hoofdpijlers van cloud-init en daarbij de meeste gebruikte modules. Er kunnen ook zelf modules toevoegd worden door deze in Python te schrijven. Met behulp van de documentatie van (cloud-init.io, 2019) wordt er per pijler uitleg gegeven.

Schijf configuratie

Er is een module die wordt gebruikt voor de schijf configuratie, namelijk **Disk Setup**. Via deze module kunnen er simpele partities en bestandsystemen worden geconfigureerd. Via de *device_aliases* richtlijnen kunnen er aliassen worden gemaakt voor de block devices. Zodat er makkelijker naar deze kan worden verwezen. Via de *disk_setup* richtlijn wordt de partitie configuratie gedaan. De *table_type* richtlijn wordt gebruikt om de partitie tabel mee te gegeven. Ten laatste is er ook de *fs_setup* richtlijn deze wordt gebruikt om de systeembestand configuratie te doen.

Commando's uitvoeren

Voor het uitvoeren van commando's zijn er 2 modules: **Runcmd** en **Bootcmd**. Beide bevatten maar een richtlijn namelijk *runcmd* en *bootcmd*.

Bij *runcmd* worden de commando's die worden meegegeven elke keer uitgevoerd als het script wordt gedraaid.

Bij *bootcmd* enkel alleen als de instantie wordt opgestart. Ook worden de commando's bij *bootcmd* veel vroeger in het bootproces uitgevoerd.

Gebruikers en groepen

Ook voor Gebruikers en groepen is er slechts één module: **Users and Groups**.

Groepen kunnen worden toegevoegd door deze aan de richtlijn *groups* toe te voegen samen eventuele configuratie per groep.

Gebruikers kunnen dan weer toegevoegd worden door deze aan de richtlijn *users* toe te voegen. Ook bij de gebruikers kan er nog extra configuratie gedaan worden per gebruiker.

Packages

Voor het beheren en configureren van packages zijn er verschillende modules: **Apt Configure**, **Apt Pipelining**, **Package Update Upgrade Install**, **Snap**, **Snappy** en **Yum Add Repo**. Er kunnen packages geïnstalleerd en geconfigureerd worden via Yum en Apt. Meestal ondersteunt een besturingssysteem Yum of Apt. Voor de installatie van een package moeten deze geplaatst worden bij de richtlijn *packages*. Cloud-init weet zelf of het met yum of apt wordt gedaan. De configuratie en het toevoegen van package repo's gebeurt via Apt Configure, Apt Pipelinig en Yum Add Repo. Ook ondersteunt cloud-init de installatie van snap packages. Dit zijn gecontaineriseerde packages. Via Snappy kan deze geconfigureerd worden.

Content bestanden

Voor het aanmaken van bestanden is er de module **Write Files**. In de richtlijn *write_files* word de gecodeerde inhoud van het bestand gezet met het pad.

Chef - Puppet

Ook zijn er 2 modules die Chef en Puppet installeren, configureren en starten. Deze modules zijn logischer wijs **Chef** en **Puppet**. Voor Puppet wordt de richtlijn *puppet* meegegeven. Daaronder worden alle configuraties geplaatst. Chef werkt op dezelfde manier.

2.2.2 User Data - Cloud config

Net zoals Ansible zijn playbook heeft, heeft cloud-init zijn user data. Ook heeft cloud-init de meta-data, maar deze wordt meegegeven door het cloud platform zelf. Dit zijn bijvoorbeeld de server naam en het server id. De 2 bekendste methodes om user data mee te geven zijn User-Data Script en Cloud Config Data.

User-Data Script

Het User-Data script een Linux script dat de server overloopt voor dat hij opstart. Het moet altijd beginnen met *!#* of *Content-Type: text/x-shellscript*. Als er echt wordt gebruikt gemaakt van cloud-init wordt deze methode niet gebruikt. De modules kun je via zo een script bijvoorbeeld niet aanroepen. Dit is eerder voor gebruikers die al een script hebben. In plaats van dit om te zetten naar Cloud Config Data, kunnen ze dat dan script gebruiken. Figuur 2.3 is een voorbeeld van een user-data script.

```
#!/bin/bash
yum update -y
amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2
yum install -y httpd mariadb-server
systemctl start httpd
systemctl enable httpd
usermod -a -G apache ec2-user
chown -R ec2-user:apache /var/www
chmod 2775 /var/www
find /var/www -type d -exec chmod 2775 {} \;
find /var/www -type f -exec chmod 0664 {} \;
echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php
```

Figuur 2.3: Voorbeeld User-Data Script.

Cloud Config Data

De populairste methode is Cloud Config Data. Dit is een yaml bestand met de configuratie van de server in. Een Cloud Config bestand begint altijd met *#cloud-config*. In het bestand worden de modules opgelijst die worden gebruikt en daaronder dan hun configuraties. Dit bestand is veel overzichtelijker dan een script om later aan te passen of om te beheren. Ook heeft het wat gelijkenissen met het playbook van Ansible, doordat dit beide yaml bestanden zijn. Figuur 2.4 is een voorbeeld een cloudconfig bestand.

```
#cloud-config

package_upgrade: true

users:
  - default
  - name: maarten
    passwd: $6$Y5N.ote2cyDmb0l0$EizY59wJIt4hMQwyFXIRpJtk.nqldtp5b4VpABWUGwXU.
    lock_passwd: false

packages:
 - docker.io
 - docker-compose
```

Figuur 2.4: Voorbeeld Cloud Config Data.

2.2.3 Omgevingen

De naam van het programma zegt al genoeg. Cloud-init is een service gemaakt voor cloud instanties en cloud platformen. Bij cloud platformen heb je de optie om user data toe te voegen aan de server(s) die je opstart. Meestal is dat een tekst vak waar je data kan invoeren. Dit kan een User-Data script zijn of een Cloud Config bestand. Voorbeelden van cloud platformen zijn: RackSpace, Amazon Web Services, Hetzner en IBM.

Het is mogelijk om er lokaal mee te werken. Op de server in kwestie moet de package cloud-init worden geïnstalleerd en dan via die package de user data uitvoeren. Je kan deze package installeren en uitvoeren op je host systeem, maar ook in een virtuele omgeving. Programma's om hiervoor te gebruiken zijn: Hyper-V, VMWare en VirtualBox.

2.2.4 Systemen

Volgens de site van cloud-init (Moser & Harlow, 2019) is het oorspronkelijk gemaakt voor Ubuntu systemen. Maar door het succes van de software is het nu beschikbaar op bijna elk Linux systeem: Fedora, Redhat, CentOS,..

3. Literatuurstudie

In dit hoofdstuk wordt een literatuur studie gedaan. Er worden 2 artikels gevonden die worden besproken:

An introduction to server provisioning with CloudInit en **Using Ansible to Bootstrap My Work Environment Part 4**. Deze waren beiden ook een onderdeel van het bachelorproef voorstel.

3.1 An introduction to server provisioning with CloudInit

Het eerste artikel dat gevonden werd is: **An introduction to server provisioning with CloudInit**. Het is geschreven door Viktor Petersson. (Petersson, 2017) beschrijft in zijn artikel de basis van cloud-init en hoe een er mee kan worden gewerkt op CloudSigma.

In deze literatuurstudie wordt er besproken wat de link met Ansible is volgens (Petersson, 2017). Een ander groot onderdeel van het artikel is hoe het Cloud config bestand wordt opgemaakt. En wat dit bestand heeft/geeft qua voordelen. Ook is er een deel waar er wordt getoond hoe er een CloudSigma server wordt opgesteld. Dit wordt niet besproken. Dit is een klein deel van het artikel en niet van nut voor deze bachelorproef/onderzoek.

3.1.1 Verschil en samenwerking met Ansible volgens Viktor Petersson

(Petersson, 2017) wil allereerst de lezer doen inzien dat cloud-init een specifieke plaats heeft in de server provisioning wereld. In (Petersson, 2017) wordt er vermeld dat Cloud-init namelijk een bepaalde eigenschap heeft, die vele provisioning systemen zoals een Ansible, Puppet of Chef niet hebben.

Hoewel cloud-init perfect kan gebruikt worden als een stand-alone provisioning systeem. Is het één van de weinige systemen die het gebruik met andere provisioning systeem ondersteunt, en zelf aanraadt volgens (Petersson, 2017). (Petersson, 2017) vermeldt ook dat hij prefereert cloud-init te gebruiken boven Ansible.

3.1.2 Setup cloud config bestand

Het volgende onderdeel van (Petersson, 2017) dat besproken word is het opstellen van zijn cloud config bestand. Dat is het bestand waar de server configuraties wordt ingevoerd. (Petersson, 2017) legt aan de hand van verschillende functies van het cloud config bestand uit wat de voordelen van cloud-init zijn. In Hoofdstuk 2.2 is dit al meer besproken. Sommige delen gaan minder uitgebreid zijn omdat het al in Hoofdstuk 2.2.1 wordt uitgelegd.

SSH

Het eerste onderdeel van het cloud config bestand van (Petersson, 2017) zijn de SSH keys. Cloud-init is volgens (Petersson, 2017) handig om ssh keys toe te voegen aan de server. Zo kan er verbinding gemaakt worden met de server.

(Petersson, 2017) gebruikt 2 fictieve gebruikers en SSH Keys in zijn voorbeeld. Deze publieke SSH sleutels worden geïnstalleerd op de server voor de gekozen gebruiker. In het fictieve voorbeeld dus een een publieke sleutel voor user1 en user2 op de server host. Op Ubuntu Cloud Images als de gebruikers niet aanwezig zijn, zullen deze worden geïnstalleerd op de standaard gebruiker ubuntu.

(Petersson, 2017) maakt verbinding met de server via het commando *ssh ubuntu@IPADRESS*. Via de toegevoegde public keys gaat dit zonder problemen.

```
#cloud-config
ssh_authorized_keys:
 - ssh-rsa AAA...user1@host
 - ssh-rsa AAA...user2@host
```

Systeem updates

Het volgende voordeel van cloud-init volgens (Petersson, 2017) is het uitvoeren van systeem updates tijdens de eerste boot. Dit is een van de functies die al werd besproken in Hoofdstuk 2.2.1.

In zijn voorbeeld gebruikt (Petersson, 2017) *apt_updgrade*. Terwijl er in Hoofstuk 2.2.1 werd gezien dat updates worden gedaan met *package_upgrade*. *apt_upgrade* is gewoon een alias voor *package_upgrade*.

Ook is *apt_update* zijn standaard waarde al true, als er packages worden geïnstalleerd op de server.

```
#cloud-config
apt_update: true
apt_upgrade: true
```

Installeren packages

Het volgende dat wordt bespreken is het installeren van packages. Ook dit werd al besproken in Hoofdstuk 2.2.1. Extra informatie die nog niet gekend was geeft (Petersson, 2017) niet. Zijn voorbeeld wordt hieronder vermeld.

```
#cloud-config
packages:
 - python-pip
 - fail2ban
 - vim
```

Hostname

Wat (Petersson, 2017) ook handig vindt aan cloud-init is het aanpassen van de hostname. Deze kan ook makkelijk worden aangepast.

```
#cloud-config
hostname: mynode
fqdn: mynode.example.com
manage_etc_hosts: true
```

Commando's

Als er sprake is van een meer geavanceerde gebruiker moeten er ook commando's worden uitgevoerd. Er zijn 2 opties om commando's uit te voeren *runcmd* en *bootcmd*. Hierover kan er ook meer informatie worden gevonden in Hoofdstuk 2.2.1.

```
#cloud-config
runcmd:
 - ls -l /root
```

Server coniguration manager

Een laatste voordeel volgens (Petersson, 2017) is dat als meer geavanceerde gebruiker er ook de mogelijkheid is om een extra Server Configuaration Manager te gebruiken.

Cloud-init ondersteunt samenwerking met onder andere Chef, Puppet en Salt.

Alles samenbinden

Ten laatste toont (Petersson, 2017) hoe dit allemaal kan worden samengevoegd in een bestand. Dit wordt allemaal samengevoegd in een YAMl bestand en bovenaan wordt er *#cloud-config* gezet. Dit werd ook vermeld in Hoofdstuk 2.2

```
#cloud-config
ssh_authorized_keys:
- ssh-rsa AAA... user1@host
- ssh-rsa AAA... user2@host

hostname: mynode
fqdn: mynode.example.com
manage_etc_hosts: true

apt_update: true
apt_upgrade: true

packages:
- python-pip
- fail2ban
- vim

runcmd:
- ls -l /root
```

3.2 Using Ansible to Bootstrap My Work Environment Part 4

Het tweede artikel is: **Using Ansible to Bootstrap My Work Environment Part 4**. Het is een Blogpost geschreven door Scott Harney. (Harney, 2016b) beschrijft in zijn blogpost hoe hij zijn werk omgeving opstart via Ansible met behulp van cloud-init.

Het is het beste artikel dat werd gevonden waar Ansible en cloud-init beide expliciet werden gebruikt. Voor het eerst werd een omgeving weergegeven die opgezet is door een samenwerking van beide. In (Harney, 2016b) wordt beschreven hoe Scott Harney een EC2 instance op zet op AWS. AWS is één van de cloud providers die cloud-init ondersteunt. Het artikel is opgedeeld in 4 onderdelen.

'*Initial provisioning*', in dit deel wordt de EC2 instance opgestart. Dit gebeurt met een eerste Ansible playbook dat onder andere ook verwijst naar het cloud-init script dat nodig is.

'*Cloud-init*': in het 2de gedeelte wordt het cloud-init script opgemaakt en uitgelegd wat alles doet en betekent.

'*Ansible folow up playbook*': in dit gedeelte wordt de EC2 instance geconfigureerd met Ansible.

'*Launch and configure*': hier wordt de omgeving opgestart en worden sommige variablen verder ingevuld.

Het literatuur onderzoek zal over de eerste 3 onderdelen gaan: initial provisioning, cloud-init en Ansible follow up playbook. Hierin worden Ansible en cloud-init besproken en geconfigureerd.

3.2.1 Ansible initial playbook

Er werd gekozen om de initiële provisioning en post-provisioning in 2 verschillende playbooks te doen. (Harney, 2016b) vond dit het handigste en vindt het onnodig om deze tot 1 playbook te maken.

Dit playbook zorgt voor het aanmaken en opstarten van de instance. Dit wordt gedaan via de Ansible role *ec2*. Via deze role en dit playbook wordt ook het cloud config bestand meegegeven. (Harney, 2016b) gaf dit mee door middel van de optie *user_data*.

3.2.2 Cloud-init

(Harney, 2016b) beschreef hier hoe hij het cloud config bestand, waar hij in het eerste playbook naar verwees, heeft geconfigureerd. Een eerste opmerking van (Harney, 2016b) was dat cloud-init niet veel goede documentatie had. (Harney, 2016b) vond dit opmerkelijk omdat het een populaire tool is. Vervolgens beschreef (Harney, 2016b) hoe zijn cloud config bestand is opgedeeld.

Begin

Allereerst werd de normale gebruiker van (Harney, 2016b) aangemaakt met de nodige configuraties.

```
#cloud-config
users:
- name: {{ ansible_user }}
ssh-authorized-keys:
- ssh-rsa  umm. nope
groups: [ 'admin', 'adm', 'dialout', 'sudo', 'lxd',
                'plugdev', 'netdev' ]
shell: /bin/bash
sudo: ["ALL=(ALL) NOPASSWD:ALL"]
```

Server en host update

In het 2de deel van het bestand nam (Harney, 2016b) zijn Hostname en fqdn en werden de packages geüpdatet. Ook update hij de instances */etc/host* en tijdzone.

```
hostname: "{{ item.hostname }}"
fqdn: "{{ item.fqdn }}"
manage_etc_hosts: true
timezone: US/Central
package_update: true
package_upgrade: true
```

Packages

Om het laatste van zijn script uit te voeren had (Harney, 2016b) 2 packages nodig, namelijk: *python* en *awscli*.

```
packages:
- awscli
- python
```

DNS zone bestand

Het laatste deel was het belangrijkste deel van het cloud config bestand. (Harney, 2016b) moest zijn DNS zone bestand updaten, zodat zijn EC2 instance correct werkte. Dit deed hij door met de optie *write_files* een script aan te maken om deze te updaten. Hij prefereerde deze methode boven het pushen via *systemd*.

```
write_files:
- content: |
#!/bin/sh
FQDN=`hostname -f`
ZONE_ID="{{ zone_id }}"
TTL=300
SELF_META_URL="http://169.254.169.254/latest/meta-data"
PUBLIC_DNS=$(curl ${SELF_META_URL}/public-hostname 2>/dev/null)

cat << EOT > /tmp/aws_r53_batch.json
{
"Comment": "Assign AWS Public DNS as a CNAME of hostname",
"Changes": [
{
"Action": "UPSERT",
"ResourceRecordSet": {
"Name": "${FQDN}.",
"Type": "CNAME",
"TTL": ${TTL},
"ResourceRecords": [
{
"Value": "${PUBLIC_DNS}"
}]}}]}
EOT

aws route53 change-resource-record-sets --hosted-zone-id ${ZONE_ID}
                --change-batch file:///tmp/aws_r53_batch.json
rm -f /tmp/aws_r53_batch.json
path: /var/lib/cloud/scripts/per-boot/set_route53_dns.sh
permissions: 0755e
```

3.2.3 Ansible follow up playbook

Nadat de instance is opgestart en de eerste configuraties zijn uitgevoerd, iss het tijd voor het tweede playbook.

Het eerste deel van het playbook ziet er zo uit.

```
- hosts: tag_Name_candyapplegrey
  gather_facts: True
  roles:
    - role: ec2
    - role: common
    - role: ansible_mystuff
    - role: openvpn_server

  tasks:
  - name: Reboot system if required
    tags: reboot
    become: yes
    command: /sbin/reboot  removes=/var/run/reboot-required
    async: 1
    poll: 0
    ignore_errors: true

- name: waiting for {{ inventory_hostname }} to reboot
  local_action: wait_for host={{ inventory_hostname }} state=
                                started delay=30 timeout=300
  become: no
```

EC2 rol

Dit deel van het playbook voegt de nieuwe publieke sleutel van de opgestarte host toe aan het *known hosts* bestand. Maar Ansible zal ook vragen de sleutel te accepteren bij de eerste ssh. Als deze nog niet in het *known hosts* bestand zit. Dit is het enige dat de EC2 role doet. Het wordt ook alleen aangeroepen als het IP adress gedefinieerd is.

```
name: Add the instance to known hosts
local_action: command sh -c 'ssh-keyscan -t rsa {{ ec2_ip_address }}
                                                >> $HOME/.ssh/known_hosts'
when: ec2_$ip_address is defined
```

Common rol

Voor de common rol wordt verwezen naar een andere artikel namelijk: (Harney, 2016a). Deze rol was zeer uitgebreid, dus heeft het zijn eigen artikel.

Sectie 1 common rol

Git is zeer belangrijk in deze rol dus koos de auteur om dit als allereerste te installeren met de rol. De ssh sleutels worden ook gekopieerd vanuit een private git repository. Erna wordt *with_items* gebruikt. Het is een looping optie in Ansible die de hier gekozen git bestanden kopieerde. Het laatste deel zorgt ervoor dat de auteur zijn private bitbucket repo's beschikbaar zijn ,met zijn vorige sleutels, vanop de instance.

```
- name: install git
become: yes
apt:
    name: git
    state: present

- name: send ssh id
copy:
    src: /home/sharney/.ssh/id_rsa
    dest: /home/sharney/.ssh/id_rsa
    mode: 0600

- name: git configfiles
become: no
copy: src={{ item.src }} dest={{ item.dest }}
with_items:
    - { src: 'dot_gitignore', dest: '/home/sharney/.gitignore' }
    - { src: 'dot_gitignore', dest: '/home/sharney/.gitignore' }
    - { src: 'dot_git', dest: '/home/sharney/.git' }

- name: bitbucket  key to known_hosts
known_hosts:
    key: "{{ lookup('pipe', 'ssh-keyscan -t rsa bitbucket.org')}}"
    name: bitbucket.org
    state: present
```

Sectie 2 common rol

Dit deel van (Harney, 2016a) zijn script, maakte hij om zijn private configuraties te doen. Dit werd ergens online gevonden, meerbepaald dankzij: (Ellingwood, 2014).

```
- name: do configfiles repo
become: no
script: /home/sharney/source/ansible-mystuff/configfiles_setup.sh
creates=/home/sharney/configfiles/.configfiles_done

- name: ssh id permissions fix
become: no
file: path=/home/sharney/.ssh/id_rsa mode=0600
```

```
#!/bin/sh

# do the bits to setup configflies repo
if [ ! -d $HOME/configfiles ]; then
mkdir $HOME/configfiles
cd $HOME
git clone --no-checkout git@bitbucket.org:scott_harney/configfiles.git
git reset --hard origin/master
fi


if [ ! -e $HOME/Dropbox/.tmux.conf ]; then
ln -s $HOME/Dropbox/.tmux.conf
fi
if [ ! -e $HOME/Dropbox/.dir_colors ]; then
ln -s $HOME/Dropbox/.dir_colors
fi

touch $HOME/configfiles/.configfiles_done
```

Sectie 3 common rol

Het volgende deel van de rol voegt een ssh github sleutel toe. Voor het clonen van publieke repos later in het in de rol. Ook worden de andere noodzakelijke packages geïnstalleerd. Sommige packages zijn evenwel niet nodig als er geen GUI aanwezig is op de instance.

```
- name: github key to known_hosts
known_hosts:
    key: "{{ lookup('pipe', 'ssh-keyscan -t rsa github.com') }}"
    name: github.com
    state: present

- name: install packages for emacs and more
action: apt pkg={{ item }} state=installed install_recommends=yes
become: yes
with_items:
    - emacs24
    - emacs24-el
    - pandoc
    - tmux
    - zsh
    - ispell
    - vpnc
    - fonts-hack-ttf
    - ruby
    - ruby-aws-sdk
    - python-pip
    - python-pip-whl
    - virtualenv
    - curl
    - openjdk-8-jre
    - fonts-crosextra-caladea
    - fonts-crosextra-carlito
```

Sectie 4 common rol

Dit waren gewoon python, pip en ruby onderdelen die helpen bij het beheren van AWS.

```
- name: python pip install items
become: yes
pip: name={{ item }} state=present
with_items:
    - powerline-status
    - awscli
    - saws
- name: aws sdk v1 for ruby for awscli
become: yes
gem: name=aws-sdk-v1 state=present
```

Sectie 5 common rol

Dit deel zorgt voor de installatie van *spacemacs*. *spacemacs* is een linux tekst editor met de power en uitbreidbaarheid van *Emacs* en de werking van *vi/vim*.

```
 name: check for spacemacs already imported
stat: path=/home/sharney/.emacs.d/spacemacs.mk
register: spacemacs_import

- name: mv .emacs.d out of the way for spacemacs
command: mv /home/sharney/.emacs.d /tmp
when: spacemacs_import.stat.exists == False

- name: spacemacs
git: repo=https://github.com/syl20bnr/spacemacs
                                              dest=/home/sharney/.emacs.d
when: spacemacs_import.stat.exists == False

- name: for spacemacs org-protocol-capture-html
git: repo=https://github.com/alphapapa/org-protocol-capture-html.git dest=/hom

- name: fix .emacs.d/private via source copy
copy: src=/home/sharney/.emacs.d/private/private
dest=/home/sharney/.emacs.d/private
when: spacemacs_import.stat.exists == False
# note: relying on the fact that the host running ansible has checked out priv
```

Sectie 6 common rol

Hier installeert (Harney, 2016a) zijn tijdzone. Ook worden hier bestanden gesynchroniseerd die in zijn Dropbox staan.

```
- name: gen en_US.UTF-8 locale
become: yes
locale_gen: name=en_US.UTF-8 state=present

- name: set default locale to en_US.UTF-8
become: yes
command: /usr/sbin/update-locale LANG=en_US.UTF-8

- name: symlink for dir_colors
file: src=/home/sharney/Dropbox/.dir_colors
path=/home/sharney/.dir_colors state=link

- name: symlink for .tmux.conf
file: src=/home/sharney/Dropbox/.tmux.conf path=/home/sharney/.tmux.conf
          state=link
```

Sectie 7 common rol

Het laaste deel conifureert de vpn. Ook worden de files van zijn website gepulled. Ten laatste is de customisatie van *zsh*.

```
- name: copy vpnc config
become: yes
copy:
    src: vpnc
    dest: /etc
    owner: root
    group: yes
    mode: 0700

- name: git clone scottharney.com
git: repo=git@bitbucket.org:scott_harney/scottharney.com.git
dest=/home/sharney/scottharney.com

- name: install ohmyzsh via git
git: repo=https://github.com/robbyrussell/oh-my-zsh.git
dest=/home/sharney/.oh-my-zsh depth=1

- name: fix .zshrc
copy: src=/home/sharney/.zshrc dest=/home/sharney/.zshrc
```

Ansible bestanden rol

Deze rol pulled alle playbooks van (Harney, 2016b) zijn git repo *ansible_mystuff* en zet het op de server. Zo kan hij vanop deze instance aan zijn playbooks werken.

```
- name: install ansible
become: yes
apt:
    name: ansible
    state: present

- name: ansible-mystuff repo for deployment of util hosts
git: repo=git@bitbucket.org:scott_harney/ansible-mystuff.git  dest=~/source/an
```

OpenVPN rol

Via de OpenVPN rol kan (Harney, 2016b) zijn instance gebruiken op plaatsen die geen veilige verbinding hebben (bijvoorbeeld een café). Ook heeft hij toegang hij tot bestanden van EC2 instances die geen internet gateway hebben. Deze instance wordt door deze rol een ssh host en een gateway naar (Harney, 2016b) zijn VPC. Omdat het een aparte rol is kunnen deze functies later hergebruikt worden.

Sectie 1 OpenVPN rol

Dit installeerde de openvpn package en kopieerde de al aanwezige configuraties naar de instance.

```
- name: install openvpn
become: yes
apt:
    name: openvpn
    state: present

- name: copy openvpn configuration data
become: yes
copy: src=openvpn dest=/etc
```

Sectie 2 OpenVPN rol

Hier wordt ip forwarding aangezet en een *iptables* is nu aanwezig voor het VPN verkeer.

```
- name: set up ip forwarding
become: yes
sysctl: name="net.ipv4.ip_forward" value=1 sysctl_set=yes state=present
                reload=yes

- name: update /etc/rc.local
become: yes
copy: src=rc.local dest=/etc/rc.local mode=0755

- name: do iptables
become: yes
iptables: state=present table=nat chain=POSTROUTING
                  source=192.168.3.0/24 out_interface=eth0 jump=MASQUERADE
```

Sectie 3 OpenVPN rol

Volgens (Harney, 2016b) was dit het moeilijste deel van zijn playbook. Voor dit deel heeft hij lang moeten troubleshooten. Het grootste probleem was een bug die openvpn reports niet goed bij hield in Xenial. (Harney, 2016b) heeft zo zijn bug opgelost.

```
- name: fix debian bug in openvpn service
https://bugs.launchpad.net/ubuntu/+source/openvpn/+bug/1580356?
become: yes
copy: src=openvpn@.service dest=/lib/systemd/system/
```

Sectie 4 OpenVPN rol

Er is een *systemd* module in Ansible 2.2 maar (Harney, 2016b) koos er voor om dit toch via de command line te doen. Dit is iets dat hij later zou kunnen aanpassen naar de module van Ansible. Maar momenteel werkt dit al.

```
- name: restart systemd daemon after updating unit config
become: yes
command: systemctl daemon-reload

- name: enable openvpn services
become: yes
command: systemctl enable openvpn.service

- name: restart openvpn services
become: yes
command: systemctl restart openvpn.service/
```

4. Methodologie

In dit hoofdstuk wordt besproken welke methodes er gehanteerd zijn om de resultaten te bekomen. Dit hoofdstuk is onderverdeeld in 2 delen.

Ten eerste wordt er info gegeven over de testomgeving.

Ten tweede wordt er besproken welke testcriteria er gekozen zijn en waarom. In deze bachelorproef wordt er vooral focus gelegd op de praktijk, vermits het een praktische onderzoek is.

4.1 Testomgeving

Voor het onderzoek moeten er verschillende testomgevingen worden opgesteld, om alles te kunnen testen. Er is een omgeving op cloud servers. Voor deze omgeving wordt er Hetzner Cloud gebruikt. In de inleiding werd het bedrijf Be-Mobile al vernoemd en werd ook al vermeld dat zij één van de drijfveren van het onderzoek zijn. Via hen is toegang verkregen op een Hetzner Cloud omgeving om in te testen.

Hetzner is een Duits bedrijf dat gespecialiseerd is in het hosten van servers. Hun datacenters liggen in Nuremberg (Duitsland), Falksenstein (Duitsland) en Helsinki (Finland).

Via de commandline tool werd de server aangemaakt. Ook wordt er een SSH sleutel voorzien zodat er toegang is tot de aangemaakte servers. Info van hetzner werd gevonden dankzij de site van (Hetzner, g.d.).

4.2 Testcriteria

Het volgende dat wordt besproken is de keuze van de testcriteria die worden gekozen in hoofdstukken 6, 7, 8 en 9.

4.2.1 Segmenten

In Hoofdstuk 6 worden basis configuraties op de servers uitgevoerd. Dit om na te gaan wat de beste optie is als er gewoon wat kleine basis configuraties worden veranderd. Dit zal het aanmaken van gebruikers en groepen, aanmaken van mappenstructuur, installeren van packages en het toevoegen van een SSH sleutel zijn. Hier wordt gekeken naar de complexititeit van de configuratie bestanden en snelheid van uitvoeren.

In Hoofdstuk 7 worden verschillende servers, LAMP en MySQL, geïnstalleerd. Zo wordt bestudeerd of het installeren en configureren van servers een ander resultaat heeft dan basis configuraties. Ook om na te gaan of er verschillen zijn per server. Hier wordt ook gekeken naar de complexititeit van de configuratie bestanden en snelheid van uitvoeren.

In Hoofdstuk 8 wordt gekeken met welke optie je de server best aanpast na het opstarten. Eens is de server is opgestart kan er iets moeten worden aangepast via het script. Welke optie is dan het best om snel deze verandering door te voeren. Hier wordt gekeken naar de complexiteit van het opnieuw uitvoeren van het script. Kan dit zeer makkelijk of moet dit via en grote omweg?

Ten laatste wordt in Hoofdstuk 9 de configuratie van containers onderzocht. Bijna elk bedrijf gebruikt containers voor hun netwerk. Het is dus logisch om te bekijken wat hier voor de resultaten zijn. Misschien zijn deze wel helemaal anders dan de resultaten van hiervoor? Ook hier wordt wel weer gekeken naar de complexititeit van de configuratie bestanden en snelheid van uitvoeren.

4.2.2 Complexiteit

Met de complexiteit van bestanden wordt bedoelt, hoe overzichtelijk een bestand juist is na het invullen van alle configuraties. Dit kan misschien ervaren worden als iets subjectiever maar doordat de bestanden worden weergegeven zullen de resultaten worden bewezen.

4.2.3 Snelheid van uitvoeren

De snelheid van uitvoer, moet wel met een korreltje zou worden bekeken. Doordat cloud-init standaard bepaalde installatie uitvoert bij het opstarten van een server. Deze duren in deze setup tussen de 20 en 45 seconden. Heeft dit altijd al een langere uitvoeringssnelheid dan Ansible. Het is wel nog steeds interessant om te bekijken. Bij sommige kan het verschil misschien kleiner zijn dan verwacht

5. Opzetten Hetzner Cloud testomgeving

In dit hoofdstuk wordt besproken hoe de testomgevingen op Hetzner zijn opgezet. Er zijn 3 omgevingen: één met cloud-init, één met Ansible en één met cloud-init en Ansible.

5.1 Aanmaken server met Hetzner

Het aanmaken van servers kan op verschillende manieren. Hier wordt er gekozen om dit te doen via de command line. Via de site van Hetzner kan je hun command tool downloaden. Via het commando **hcloud** kunnen er nu server worden aangemaakt en beheert. Voor toegang te krijgen tot Hetzner Cloud moet je een bepaald bedrag betalen. Hierna kan je netwerken opzetten en via een Hetzner Token kan je op dit netwerk servers toevoegen. Via het bedrijf Be-Mobile is er Hetzner Token vekregen.

Voor het aanmaken van de server zijn er 2 variabelen die worden toegevoegd. **Name**, zo kan de naam van de server worden meegegeven. Zo kunnen de servers uit elkaar houden. **Ssh-key**, via Hetzner kan je een ssh sleutel toevoegen om toegang te verkrijgen op de server. Via het bedrijf Be-Mobile is er een ssh sleutel verkegen. Het commando voor het aanmaken van een server is dan:

```
hcloud server create --name <naam van de server>
    --ssh-key <naam van de sleutel in hetzner>
```

Er zijn nog twee andere belangerijke commando's die worden gebruikt. Via het commando hieronder is er een lijst met alle servers die beschikbaar zijn.

```
hcloud server list
```

Het commando hieronder verwijdert een server.

```
hcloud server delete <naam of id van de server>
```

5.2 Cloud-init

Het opzetten van een omgeving met cloud-init in Hetzner Cloud is eigenlijk zeer simpel. Op (W. Hetzner, g.d.) staat er dat bij het maken van de server er *user data* kan worden toegevoegd. Zoals werd gezien in hoofdstuk 2.2 is dit de data die cloud-init gebruikt voor zijn configuraties.

De user data is een vairbale die moet worden toegevoegd zoals de naam een ssh sleutel. Via **user-data-from-file** kan er een bestand worden toegevoegd tijdens het aanmaken van de server met de user data. Het commando dat wordt gebruikt om een server aan te maken ziet er dan zo uit:

```
hcloud server create --name <naam van de server>
    --ssh-key <naam van de sleutel in hetzner>
    --user-data-from-file <pad naar bestand>
```

5.3 Ansible

Het opzetten van een omgeving in Hetzner Cloud met Ansible is niet zo makkelijk. Er is geen variabale *playbook* beschikbaar waar er een Ansible playbook kan aan worden toegevoegd.

Er zal gewerkt worden met behulp van GitHub. Het playbook dat wordt uitgevoerd, zal worden verkregen met een *git clone* of *git pull*. Daarna zal het worden uitgevoerd aan de hand van het commando:

```
ansible-playbook -i , <playbook bestand>
```

Via dit commando wordt het playbook op de localhost uitgevoerd.

Als het op meerdere hosts moet worden uitgevoerd zal er een inventory bestand worden toegevoegd. Het commando ziet er dan zo uit:

```
ansible-playbook -i <inventory bestand> <playbook bestand>
```

5.4 Ansible & cloud-init

Het opzetten van een omgeving met Ansible en cloud-init wordt gedaan door een combinatie van de vorige 2 manieren. De server wordt opgezet met het commando:

```
hcloud server create --name <naam van de server>
--ssh-key <naam van de sleutel in hetzner>
--user-data-from-file <pad naar bestand>
```

De commando's die worden uitgevoerd om een Ansible playbook uit te voeren, worden in cloud-init verwerkt. In de module *runcmd* wordt eerst het commando *git clone* gezet, met verwijzing naar de repo met het playbook. Erna wordt het commando *ansible-playbook* er gezet, deze voert het playbook uit.

Het enige extra dat wordt toegevoegd is een private ssh sleutel. Dit is de ssh-sleutel die toelaat dat je de git repo mag binnenhalen. Deze voegen we toe doormiddel van de module *ssh_keys* en daar de submodule *rsa_private*. Er worden ook nog 2 commando's boven de andere 2 geplaatst zodat dit werkt. Allereerst het commando *ssh-keyscan github.com » /etc/ssh/ssh_known_hosts*. Hiermee kent het systeem github en gaat dit geen error geven. Het tweede commando is *ssh-agent bash -c 'ssh-add /etc/ssh/ssh_host_rsa_key; git clone <repo>'*. Met dit commando zal cloud-init de ssh sleutel gebruiken die is toegevoegd.

6. Basisconfiguraties op de servers

In dit hoofdstuk wordt voor het eerst een onderzoek uitgevoerd.

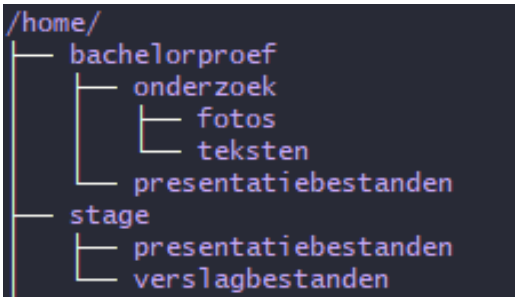
Er wordt voor elke server een configuratie bestand gemaakt dat 4 basisconfiguraties zal uitvoeren: gebruikers en groepen toevoegen, packages installeren en updates, mappen structuur aanmaken en ssh configuratie.

6.1 Aanmaken configuratie bestanden

In dit hoofdstuk zal uitglegd worden hoe de playbooks en cloudconfig bestanden worden aangemaakt en opgesteld.

Eerst wordt uitgelegd hoe de cloud-init configuraties worden gedaan, erna het Ansible playbook. Ten laatste zullen de set-ups worden gemaakt waar een combinatie wordt gebruikt.

De gebruiker die zal worden aangemaakt is *bachelor* met wachtwoord *proef*. De gebruiker zal behoren tot de groep *test* die ook zal worden aangemaakt. De packages die zullen worden geïnstalleerd zijn: *pwgen*, *tree* en *git*. De mappenstructuur die zal worden aangemaakt ziet er uit zoals die foto hieronder. En ten laatste zal er op de server een publieke ssh sleutel worden toegevoegd zodat er toegang is vanaf een test server. Figuur 6.1 is een visueel voorbeeld van de mappenstructuur.



Figuur 6.1: Mappen structuur basis configuraties.

6.1.1 Opstellen cloud-init config bestand

Als eerste zal het cloudconfig bestanden worden opgesteld. Dit bestand werd opgesteld met behulp van (cloud-init.io, 2019).

Eerst werd er bekeken welke modules er nodig zullen zijn voor het opstellen van dit bestand. De modules *packages* en *package_upgrade* zullen worden gebruikt voor de packages. De modules *users* en *groups* zullen worden gebruikt voor het toevoegen van gebruikers en groepen.

Voor het toevoegen van de ssh sleutel, wordt de module *ssh_authorized_keys* gebruikt. Hier worden de publieke sleutels die zijn toegelaten toegevoegd.

Voor het aanmaken van bestanden bestaat er de module *write_files*, spijtig genoeg is er geen module voor het aanmaken van mappen. Voor het aanmaken van de mappenstructuur werd dus de module *runcmd* gebruikt.

Packages

Hierna werd het effectieve bestand gevormd. Onder de *packages* werden de 3 packages opgelijst: *git*, *pwgen* en *tree*. Bij *package_upgrade* werd *true* gezet.

```
packages:
- pwgen
- git
- tree

package_upgrade: true
```

Users & Groups

Bij *groups* moest niet veel gedaan worden. Er werd gewoon de groepsnaam van de groep gezet.

Bij *users* werden verschillende dingen ingevuld. Eerst en vooral de name, *bachelor*. Bij groups de naam van de aangemaakte groep. Het wachtwoord onder passwd, proef, werd

geëncrypteerd met de tool: („Tool makepasswd”, g.d.). Als shell werd ook voor */bin/bash* gekozen.

```
groups:
- test

users:
- default
- name: bachelor
- passwd: 985b56433efe9898290b88d4dab853a2f09d7eb7a7b1b8d2cdd431
- groups: test
- shell: /bin/bash
```

Runcdm

Bij de module *runcmd* werden alle commando's gezet voor het aanmaken van de mappen structuur. Dit zijn gewoon 5 *mdkir* commando's. Dit is het commando voor het aanmaken van mappen. Met de paramater *-f* wordt heel de mappen structuur meegegeven aangemaakt als die er nog niet is.

```
runcmd:
- mkdir /home/bachelorproef/onderzoek/fotos -f
- mkdir /home/bachelorproef/onderzoek/teksten -f
- mkdir /home/bachelorproef/presentatiebestanden -f
- mkdir /home/stage/presentatiebestanden -f
- mkdir /home/stage/veslagbestanden -f
```

Ssh

Als laatste wordt de ssh sleutel toegevoegd aan de server. Bij de waarde <*insert key value*> wordt de publieke sleutel van de host die gaat connecteren gezet.

```
ssh_authorized_keys:
- ssh-rsa <INSERT KEY VALUE>
```

Helemaal onderaan werd ook iets gezet onder de module *final_message*, namelijk: *"The system is finally up, after $UPTIME seconds"*. Door de waarde *$UPTIME* kan er gekeken worden hoelang de server nodig had om alles uit te voeren.

6.1.2 Opstellen Ansible playbook

Als tweede werd het Ansible playbook opgesteld. Als host wordt gekozen voor localhost. Aan het *ansible.cfg* bestand werd ook nog de regel *callback_whitelist = profile_tasks*. Zo kan er bekeken worden hoelang het draaien van het playbook duurde. Alle configuraties die worden gedaan werden in verschillende *tasks* gezet.

Packages

Voor het upgraden van de packages en installeren van de nodige packages werd de task *apt* gebruikt.

```
- name: Upgrade all packages to the latest version
apt:
name: "*"
state: latest
- name: Install git
apt:
name: git
state: present
- name: Install tree
apt:
name: tree
state: present
- name: Install pwgen
apt:
name: pwgen
state: present
```

Mappen structuur

Voor het aanmaken van de mappen structuur werd de task *file* gebruikt. De state werd dan op directory gezet.

```
- name: Creates direcory fotos
file:
path: /home/bachelorproef/onderzoek/fotos
state: directory
- name: Creates direcory teksten
file:
path: /home/bachelorproef/onderzoek/teksten
state: directory
- name: Creates direcory bachelor presentatie
file:
path: /home/bachelorproef/presentatiebestanden
state: directory
- name: Creates direcory stage presentatie
file:
path: /home/stage/presentatiebestanden
state: directory
- name: Creates direcory stage verslag
file:
path: /home/stage/verslagbestanden
state: directory
```

Users & groups

Bij het aanmaken van de gebruikers en groepen werden de tasks *group* en *user* gebruikt. Bij de *user* werden het shell en password meegeven. Ook de groups werden meegegeven.

```
- name:  add group test
group:
name: test
state: present
- name:  add user bachelor
user:
name: bachelor
groups: test
shell: /bin/bash
password: 985b56433efe9898290b88d4dab853a2f09d7eb7a7b1b8d2cdd431e2920c35
```

Ssh

Voor het toevoegen van de ssh sleutel aan de server, zodat er een extra connectie kan worden gelegd, wordt de task *authorized_key* gebruikt. De user die wordt meegegeven is de bachelor user.

```
- name: Set authorized key taken from file
authorized_key:
user: bachelor
state: present
key: ssh-rsa <INSERT KEY VALUE>
```

6.1.3 Ansible & cloud-init omgevingen

Beide bestanden die apart gemaakt zijn, zijn vrij complexloos. Beide bestanden zijn vrij duidelijk. Wat als nadeel is voor het gebruik van een combinatie van Ansible en cloud-init. De enige uitvoering die complex is de bestanden is het aanmaken van de mappen via cloud-init. Dit is een beetje 'dirty' gedaan. Voor het werken met een combinatie gaat eerst alles via cloud-init worden gedaan. Het aanmaken van de mappen dan via Ansible.

Verwijzing Ansible

Voor de cloud setup wordt de verwijzing via github gedaan. Er wordt een git repository aangemaakt met het playbook in. Die wordt dan gecloned in cloud-init en uitgevoerd.

In cloud-init wordt drie modules toegevoegd: *runcmd* en *ssh_keys*. In de module *ssh_keys* wordt de private sleutel toegevoegd connectie legt met de git repo. In de module *runcmd* worden 3 commando's gezet: het toevoegen van github aan het known host bestand, de git clone en het uitvoeren van het playbook. Bij de packages wordt ook de package *ansible* gezet.

```
ssh_keys:
rsa_private: | <key>
runcmd:
- ssh-keyscan github.com >> /etc/ssh/ssh_known_hosts
- ssh-agent bash -c 'ssh-add /etc/ssh/ssh\_host\_rsa\_key;
git clone <repo>'
- ansible-playbook -i, /home/ansible/playbook.yaml
```

6.2 Uitvoering & resultaten

6.2.1 Complexiteit

Overzichtelijkheid

Eerst en vooral wordt er gekeken naar het de overzichtelijkheid van het playbook bestand en het cloudconfig bestand. Terwijl beide bestanden yaml bestanden zijn, zien ze er toch compleet anders uit. In het cloudconfig bestand is alles mooi onderverdeeld in verschillende modules, die de taken uitvoert. Terwijl bij het Ansible playbook alles opgesomd bij de tasks staat. Cloud-init is hierdoor veel overzichtelijker. Het is hierdoor ook makkelijker om het bestand te wijzigen. Bij het gebruik van de combinatie het playbook wel overzichtelijker doordat dit maar een paar taken heeft. In figuur 6.2 staan de layouts van het cloud-init en Ansibles bestand.

```
#cloud-config
packages:
  - pwgen
  - git
  - tree
package_upgrade: true
runcmd:
  - mkdir /home/bachelorproef/onderzoek/fotos -p
  - mkdir /home/bachelorproef/onderzoek/teksten -p
  - mkdir /home/bachelorproef/presentatiebestanden -p
  - mkdir /home/stage/presentatiebestanden -p
  - mkdir /home/stage/veslagbestanden -p
groups:
  - test
users:
  - default
  - name: bachelor
  - passwd: 985b56433efe9898290b88d4dab853a2f09d7eb7a7b1b8d2cdd431
  - sudo: true
  - groups: test
  - shell: /bin/bash
ssh_authorized_keys:
  - ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDmaqbeev+BS4vgppVTV5qBpp
final_message: "The system is finally up, after $UPTIME seconds"
```

```
- hosts: localhost
  tasks:
    - name: Upgrade all packages to the latest version
      apt:
        name: "*"
        state: latest
    - name: Install git
      apt:
        name: git
        state: present
    - name: Install tree
      apt:
        name: tree
        state: present
    - name: Install pwgen
      apt:
        name: pwgen
        state: present
    - name: Set authorized key taken from file
      authorized_key:
        user: charlie
        state: present
        key: "{{ lookup('file', '/vagrant/key') }}"
    - name: Creates direcory fotos
      file:
        path: /home/bachelorproef/onderzoek/fotos
        state: directory
    - name: Creates direcory teksten
      file:
        path: /home/bachelorproef/onderzoek/teksten
        state: directory
    - name: Creates direcory stage presentatie
      file:
        path: /home/bachelorproef/presentatiebestanden
        state: directory
    - name: Creates direcory bachelor presentatie
      file:
        path: /home/stage/presentatiebestanden
        state: directory
    - name: Creates direcory stage verslag
      file:
        path: /home/bachelorproef/stage/verslagbestanden
        state: directory
    - name:  add group test
      group:
        name: test
        state: present
    - name:  add user bachelor
      user:
        name: bachelor
        groups: test,admin
        shell: /bin/bash
        password: 985b56433efe9898290b88d4dab853a2f09d7eb7a7b1b8d2cdd431e2920c3586abb7f15af7ef538
```

Figuur 6.2: Cloud-init en Ansible basis configuraties layout.

Aanroepen uitvoeringen

Ten tweede wordt er gekeken hoe alles werd aangeroepen. Voor het toevoegen van users, groepen en packages en het aanmaken van een nieuwe ssh sleutel zijn er voor Ansible en cloud-init aparte modules om dit te doen. Hier is weinig verschil. Maar voor het aanmaken van de mappen was er een verschil. Bij het toevoegen van de mappen is er bij Ansible een aparte module hiervoor. Terwijl bij cloud-init dit moest worden gedaan via runcmd en het *mkdir* commando.

Het aanroepen van de Ansible configuraties via cloud-init gebeurde via de commando module (*runcmd*) in cloud-init. Dit gebeurde op een vrij efficiënte manier. Al moet er wel elke keer ene private sleutel van de repo worden meegegeven.

6.2.2 Snelheid

Als er puur naar de snelheid wordt gekeken is Ansible duidelijk sneller. Maar cloud-init heeft configuraties die het altijd uitvoert. Hierdoor kan deze statistiek soms misleidend. De gemiddelde uitvoersnelheid voor Ansible is 137.53 seconden en voor cloud-init is dit 177.77 seconden. Als er bij cloud-init het verschil van de startup configuraties wordt afgetrokken, komt dit eerder op 140-145 seconden. Dit ligt veel dichter bij Ansible, dit verschil is eigenlijk verwerpbaar. Al is Ansible een beetje sneller is dit niet zo een groot verschil dat dit een voordeel is voor Ansible. Er kan worden vastgesteld dat ze min of meer dezelfde uitvoeringssnelheid hebben.

De cloud-init en Ansible configuraties hebben bijna dezelfde uitvoeringssnelheid als cloud-init, namelijk 177.60 seconden. Net zoals bij cloud-init kan er hier worden afgerond naar 140-145 seconden door de eerste basisconfiguraties van cloud-init. Ook hier is dus weer geen groot verschil merkbaar.

In tabel 6.1 staan de resultaten van de snelheden.

Uitvoeringstijd	Resultaat 1	Resultaat 2	Resultaat 3
cloud-init	179.75 sec	187.74 sec	165.51 sec
Anisble	168.69 sec	132.59 sec	111.31 sec
cloud-init & Ansible	149.10 sec	190.13 sec	193.59 sec

Tabel 6.1: Snelheid van Basisconfiguraties op de servers.

6.2.3 Resultaat

Voor basisconfiguraties lijkt het meteen wel beter om dit niet met Ansible en cloud-init te doen. Dit lijkt iets te omslachtig voor het maar basis werk dat moet gebeuren.

Voor Ansible en cloud-init apart is er niet zoveel verschil. Al is het cloudconfig bestand wel veel overzichtelijker, heeft Ansible iets meer variatie qua modules. Toch lijkt cloud-init in deze setup de beste optie. Het meegeven van het script gebeurt op een veer

makkelijker manier. En doordat het maar basisconfiguraties zijn die hier worden getest, is het voordeel van de meer variatie van modules in Ansible nog niet zo doorslaggevend. De overzichtelijkheid van cloud-init is dat dan wel. Het bestand is veel compacter en veel duidelijker. Ook doordat het bestand op een makkelijkere manier kan worden uitgevoerd lijkt cloud-init hier de betere optie. Ansible geeft wel een betere output weer. Maar dit is hier nog niet doorslaggevend doordat er nog geen geavanceerde acties gebeuren.

Cloud-init lijkt, zeker als het in een gelijkaardige setup als deze gebeurt, de beste optie om eerste basisconfiguraties aan de server toe te voegen.

7. Server installatie en configuratie

Dit hoofdstuk voert een tweede onderzoek. Er wordt gekeken hoe makkelijk een volledig type server wordt geïnstalleerd en geïmplementeerd.

De twee type servers die worden geïnstalleerd zijn: MySQL en LAMP.

7.1 LAMP server

Een LAMP server is een van de populairste variaties van een webserver op linux. Via de documentatie van (Potter, 2019) wordt een kleine uitleg gegeven.

De 'L' staat voor Linux het besturingssysteem dat op de server draait.

De 'A' staat voor apache dit is de web server software.

De 'M' kan voor 2 dingen staan: MariaDB of MySQL. Dit zijn de database server softwares die worden gebruikt. In deze setup wordt gekozen voor MariaDB aangezien er al een aparte MySQL server wordt aangemaakt.

De 'P' staat voor PHP. Dit is de programmeer taal die de websites zullen gebruiken.

Een variatie op LAMP is LEMP. Dit is bijna hetzelfde alleen gebruikt het nginx als web server software in plaats van apache.

7.1.1 Cloud-init

Het bestand werd opgemaakt met behulp van (Butcher, 2013). Hierdoor krijgen we een basis lamp stack.

Allereerst werd het cloud-init script opgemaakt. Het script bevat niet zoveel functies. Het bevat 3 modules: *package_upgrade*(voor het updaten van de server packages), *packages* (voor de packages die nodig zijn) en *runcmd* (voor de taken die moeten worden uitgevoerd).

Opmaken bestand

Package_upgrade wordt allereerst op true gezet. Bij de *packages* wordt: *apache2, mysql-server, libapache2-mod-php5* en *php5-mysql* gezet. Dit zijn de packages dit nodig zijn voor de lamp stack.

```
package_upgrade: true

packages:
- apache2
- mysql-server
- libapache2-mod-php5
- php5-mysql
```

Bij *runcmd* werden 2 commando's geplaats. Het eerste maakt het php bestand aan dat de server gaat gebruiken. Het tweede verwijdert het standaard html bestand.

```
runcmd:
- echo '<?php phpinfo(); ' > /var/www/html/index.php
- rm /var/www/html/index.html
```

7.1.2 Ansible

Dit bestand werd opgemaakt met behulp van (Bekker, 2018). Dit maakt een basis lamp stack aan. Het playbook is onderverdeeld in 3 delen: installeren van de packages, starten van de services en het toevoegen van het php bestand.

Installeren packages

Eerst en vooral worden de packages geïnstalleerd: *apache2, my-sqlserver, php* en *php-mysql*.

```
- name: install lamp stack
become: yes
become_user: root
apt:
pkg:
- apache2
```

```
- mysql-server
- php
- php-mysql
state: present
update_cache: yes
```

Starten services

Als tweede werden de *apache2* en *mysql* server gestart.

```
- name: start apache service
become: yes
become_user: root
service:
name: apache2
state: started
enabled: yes

- name: start mysql service
become: yes
become_user: root
service:
name: mysql
state: started
enabled: yes
```

Php bestand

Ten laatste werd het php bestand aangemaakt en de map waar het moet komen. Het bestand kon via Ansible wel niet worden aangemaakt. Dus werd het via github meegeven. Het stond ook in de repo van het playbook.

```
- name: create target directory
file: path=/var/www/html state=directory mode=0755

- name: deploy index.php
become: yes
become_user: root
copy:
src: /repo/index.hphp
dest: /var/www/html/index.php
```

7.1.3 Cloud-init & Ansible

Voor de configuratie van Ansible en cloud-init worden de taken in 2 verdeeld.

De packages worden geinstalleerd via cloud-init. Ook het aanmaken van het php bestand gebeurt via cloud-init. Het starten van de services wordt gedaan door Ansible ook het aanmaken van de directory wordt gedaan door Ansible.

7.2 MySQL server

De tweede server die werd gekozen is een MySQL server. MySQL is een database management systeem (DBMS). Een MySQL server is dus simpel weg een database server.

7.2.1 Cloud-init

Dit bestand werd opgemakt met behulp van (NMDias, 2017).

Voorr het aanmaken van een MySQL server werden 3(4) modules gebruikt. Allereest de *packages* (en *package_upgrade*) voor de packages, *users* voor de database user en *runcmd* voor de taken/commando's.

Packages

De packages die werden geïnstalleerd zijn: *mysql-client, libmysqclient-dev* en *mysql-server*.

```
packages:
- mysql-client
- libmysqlclient-dev
- mysql-server

package_upgrade: true
```

User

Er werd een gebruiker toegevoegd dev. Met een ssh sleutel zodat er toegang is via deze gebruiker.

```
users:
- name: dev
ssh-authorized-keys:
- ssh-rsa <key>
sudo: ['ALL=(ALL) NOPASSWD:ALL']
groups: sudo
shell: /bin/bash
```

Commando's

Ten laatste al de commando's die worden uitgevoerd. Hier wordt de mysql server geconfigureerd.

```
runcmd:
- sed -i -e '/^Port/s/^.*$/Port 22/' /etc/ssh/sshd_config
- sed -i -e '/^PermitRootLogin/s/^.*$/PermitRootLogin no/' /etc/ssh/sshd_confi
- sed -i -e '/^PasswordAuthentication/s/^.*$/PasswordAuthentication no/' /etc/
- sed -i -e '$aAllowUsers dev' /etc/ssh/sshd_config
- sudo service ssh restart
- sudo ufw default deny incoming
- sudo ufw default allow outgoing
- sudo ufw allow ssh
- sudo ufw allow http
- sudo ufw allow https
- sed -i -e '/^ENABLED/s/^.*$/ENABLED=yes/' /etc/ufw/ufw.conf
- sudo ufw enable
```

7.2.2 Ansible

Dit bestand werd gemaakt met behulp van (Hassin, 2013).

Allereest werden weer de my-sql packages geïnstalleerd.

```
- name: install my-sql
become: yes
become_user: root
apt:
pkg:
- mysql-server-core-5.7
- mysql-client-core-5.7
- libmysqlclient-dev
- python-mysqldb
- mysql-server
- mysql-client
```

Hierna werden de my-sql configuraties gedaan: het starten van de service, verwijderen van de test database, een my-sql gebruiker aanmaken, anonieme gebruikers verwijderen en het wachtwoord aanpassen voor de root gebruiker.

```
- name: Start the MySQL service
action: service name=mysql state=started

- name: Remove the test database
mysql_db: name=test state=absent

- name: Create deploy user for mysql
mysql_user: user="deploy" host="%" password=maarten priv=*.*:ALL,GRANT
```

```
- name: Ensure anonymous users are not in the database
mysql_user: user='' host=localhost state=absent
with_items:
- 127.0.0.1
- ::1
- localhost

- name: Update mysql root password for all root accounts
mysql_user: name=root host=localhost password=maarten
with_items:
- 127.0.0.1
- ::1
- localhost
```

7.2.3 Cloud-init & Ansible

Voor de configuratie van cloud-init en Ansible werden gelijkaardige dingen gedaan als bij LAMP. In principe werden bijna alle configuraties weer gedaan door Ansible. Alleen werden de packages geïnstalleerd door cloud-init.

7.3 Uitvoering & resultaten

7.3.1 Complexiteit

Allereerst wordt de complexiteit weer bekeken. Daarin is er een verschil tussen MySQL en LAMP.

Bij de LAMP server is door de weinige configuraties, het cloud-init bestand heel overzichtelijk. Maar bij de MySQL server is dit dan weer niet. Doordat er verschillende configuraties worden gedaan en deze allemaal in de *runcmd* module worden gedaan, is het bestand niet zo duidelijk.

Bij Ansible daarentegen zijn er door de verschillende modules allemaal verschillende taken hiervoor apart. In deze setup zijn die verschillende taken wel overzichtelijk.

Ook kunnen er met Ansible meer geavanceerde configuraties worden uitgevoerd bij beide servers. Cloud-init is beperkt tot de configuraties die via een commando kunnen worden gedaan. Ansible kan veel meer uitvoeren.

7.3.2 Snelheid

Qua snelheden is het vrij duidelijk. Ansible is veel efficiënter en sneller in het uitvoeren van de taken dan “Ansible en cloud-init” en cloud-init. Het verschil zonder de initiële configuraties doet er zelf niet toe.

Dit is, t.o.v het vorige hoofdstuk, wel een verschil dat merkbaar is. Waardoor Ansible in dit onderdeel duidelijk de beter is ten opzichte van de andere 2 opties.

Ook is t.o.v. het vorige hoofdstuk het verschil tussen cloud-init en “Ansible en cloud-init” veel groter. In de onderstaande tabellen 7.1 en7.2 staan de resultaten.

Uitvoeringstijd	Resultaat 1	Resultaat 2	Resultaat 3
cloud-init	230.57 sec	229.85 sec	234.71 sec
Anisble	62.71 sec	76.10 sec	65.48 sec
cloud-init & Ansible	150.97 sec	127.93 sec	129.80 sec

Tabel 7.1: Snelheid tabel van LAMP configuraties op de servers.

Uitvoeringstijd	Resultaat 1	Resultaat 2	Resultaat 3
cloud-init	204.03 sec	214.15 sec	281.82 sec
Anisble	29.26 sec	35.86 sec	111.31 sec
cloud-init & Ansible	127.71 sec	85.43 sec	66.81 sec

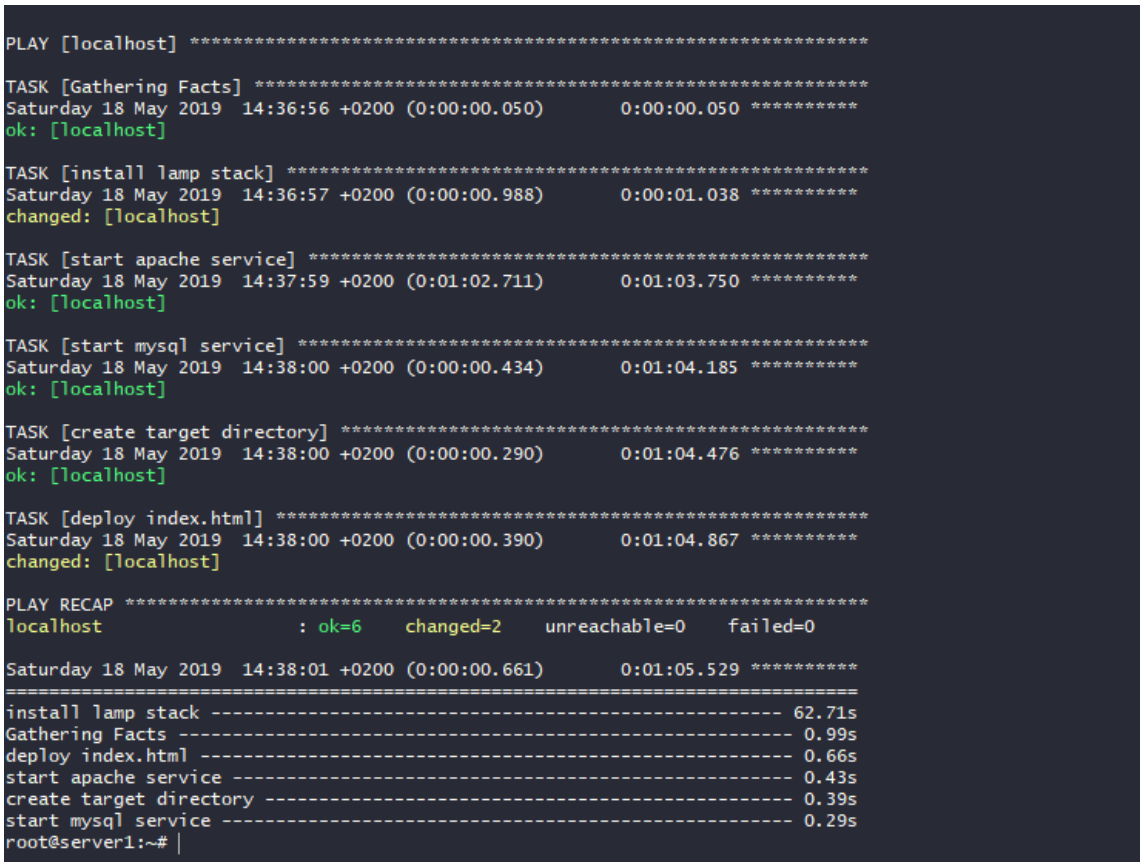
Tabel 7.2: Snelheid tabel van MySQL configuraties op de servers.

7.3.3 Resultaat

Voor server configuratie is het duidelijk dat Ansible de beste optie is. Ansible is veel sneller, overzichtelijker en heeft veel meer opties om te configureren.

Ook door de output die Ansible geeft (Figuur 7.1) is het veel makkelijker om fouten uit een server configuratie te halen. Door de vele configuratie die deze scripts soms hebben, is het overzichtelijk dat in de output alles is opgelijst als 1 taak.

Al is cloud-init niet helemaal nutteloos hier. Als er bij de opstart van de server deze configuraties moeten worden gedaan. Is het de beste optie om cloud-init en Ansible te gebruiken. Doordat het cloud-init veel makkelijker kan worden meegegeven als een variabele. In dit script wordt er dan verwezen naar het Ansible playbook.



Figuur 7.1: Ansible output voorbeeld.

8. Script aanpassen en 2de keer uitvoeren

In dit hoofdstuk wordt bekeken hoe makkelijk of moeilijk een script kan worden aangepast en dan nog eens uitgevoerd. Zodat er via het script nog aanpassingen kunnen worden gedaan als er een configuratie niet juist was. De eventuele testen worden uitgevoerd op de hetzner cloud servers.

8.1 Cloud-init aanroepen

Eerst gaat er worden bekeken of een cloud-init makkelijk nog eens kan worden uitgevoerd, met een andere configuratie.

In de setup voor cloud-init die hier wordt gebruikt. Wordt cloud-init niet uitgevoerd via een commando. Het script wordt meegegeven als een variabele als de server wordt aangemaakt.

Nadat de server is aangemaakt en geconfigureerd kan het script wel worden gevonden. Het script staat in het bestand */var/lib/cloud/instances/<instance-naam>/user-data.txt*.

Nu het script gevonden is, wordt gekeken hoe er met een commando cloud-init kan worden geladen. Op een server die opgezet is met cloud-init staat automatisch ook meteen het cloud-init.

Via documentatie van (Scaleway, g.d.) werd gekeken hoe dit config bestand weer kan worden doorlopen (na het ook aan te passen). Dit wordt gedaan door via een andere instance deze instance weer op te zetten. Alleen is dit niet hoe de setup in deze bachelorproef werkt. Voor deze setup en deze situatie blijkt het dus onmogelijk om een cloudconfig script aan te passen en een tweede keer uit te voeren.

8.2 Ansible aanroepen

Na cloud-init te bekijken, wordt er nu gekeken of Ansible dit wel goed ondersteunt.

In de setup die hier wordt gebruikt, wordt het playbook aangeroepen via een commando. In theorie lijkt het dus makkelijk om dit script weer aan te roepen en een kleine verandering door te voeren. Maar is dit ook zo in de praktijk?

8.2.1 Praktisch

Het script van hoofdstuk 6 wordt hier hergebruikt. Normaalgezien zal dit worden gedaan bij complexere playbooks. Maar in deze bachelorproef wordt gewoon getest of het mogelijk is.

Eerst wordt er een server aangemaakt en wordt het playbook doorlopen. Hierna wordt het playbook aangepast. De user *bachelor* wordt aangepast naar *tweederun*.

```
- name:  add user bachelor
  user:
    name: tweederun
    groups: test
    shell: /bin/bash
    password: 985b56433efe9898290b88d4dab853a2f09d7eb7a7b1b8d2cdd431e
```

Na de aanpassing door te voeren in het playbook. Wordt het playbook nog is doorlopen. Bij de output van het playbook wordt de message changed weergegeven bij het toevoegen van de user. Als er nu wordt geprobeerd te veranderen naar deze user gaat dit.

Via Ansible is het dus mogelijk om een verandering door te voeren aan het script en dit dan een tweede keer uit te voeren.

8.3 Resultaat

Het resultaat is vrij duidelijk. Als er een script wordt gemaakt, met variabele die kunnen veranderen, is het beter dat Ansible hier wordt voor gebruikt. Ansible ondersteunt veel beter de mogelijkheid om een script een tweede keer te doorlopen en hierbij te checken of er dingen moeten worden aangepast. Cloud-init ondersteunt, toch in deze setup, de functie niet.

9. Container & Cluster Configuratie

Dit hoofdstuk voert een laatste onderzoek uit, Containerization.

Hier wordt bekeken hoe Docker het best kan worden geïmplementeerd. Docker is de grootste en populairste technologie in containerization.

9.1 Docker

Docker is een tool dat gemaakt is om het maken, deployen en uitvoeren van applicaties makkelijker te maken door deze in containers te stoppen.

Containers zijn een package van een applicatie met al zijn dependencies en libraries. Heel de applicatie is nu één enkele package. (opensource, 2019)

9.1.1 Cloud-init

De configuratie van docker met cloud-init gebeurt via docker-compose. Docker-compose is een manier om docker containers aan te maken te configureren. Er wordt een yaml bestand aangemaakt met de docker-compose modules en verwijzingen. Hierna wordt dit bestand via een commando uitgevoerd.

Het eerste dat wordt gedaan in het cloud-init bestand is installeren van de packages. De package *docker-compose* wordt ook geïnstalleerd. Via deze technologie worden de containers aangemaakt.

```
packages:
- docker.io
- docker-compose
```

Als tweede wordt het docker-compose bestand aangemaakt. Dit wordt gedaan via de module *write_files*.

```
write_files:
- path: "home/docker-compose.yml"
content: |

    version: '3.3'
    services:
    db:
    image: mysql:5.7
    restart: always
    environment:
    MYSQL_ROOT_PASSWORD: somewordpress
    MYSQL_DATABASE: wordpress
    MYSQL_USER: wordpress
    MYSQL_PASSWORD: wordpress
```

Ten laatste wordt via een commando de container aangemaakt.

```
runcmd:
-  docker-compose -f home/docker-compose.yml up
```

9.1.2 Ansible

Via Ansible kan de docker-container worden aangemaakt met een module in Ansible. Hier wordt niet gewerkt met docker-compose.

Eerst wordt gewoon docker geïnstalleerd.

```
- name: install docker
become: yes
become_user: root
apt:
pkg:
- docker.io=18.06.*
state: present
update_cache: yes
```

Er zijn wel 3 pip packages die moeten worden geïnstalleerd. De eerste is *docker-py* als deze niet werd geïnstalleerd werkte de module *docker_container* niet. De andere zijn *setuptools* en *wheel*. Dit zijn 2 pip packages die *docker-py* nodig had om te functioneren.

```
- name: install pip
apt: name=python-pip state=present

- name: pip install setuptools
pip: name=setuptools

- name: pip install wheel
pip: name=wheel

- name: pip install docker-py
pip: name=docker-py
```

Als laatste werd de container aangemaakt via de module van Ansible.

```
- name: Create a mysql container
docker_container:
    name: mysqlcontainer
    image: mysql:5.7
    restart_policy: always
    env:
    MYSQL_ROOT_PASSWORD: somewordpress
    MYSQL_DATABASE: wordpress
    MYSQL_USER: wordpress
MYSQL_PASSWORD: wordpress
```

9.1.3 Ansible & cloud-init

In de omgeving met Ansible en cloud-init werden de packages *docker.io* en *python-pip* via cloud-init geïnstalleerd. De rest werd gedaan via Ansible. Er werd dus weer gewerkt via de Ansible module en niet docker-compose. Ook de pip packages werden dus weer geïnstalleerd.

9.2 Uitvoering & resultaten

9.2.1 Complexiteit

De complexiteit van beide bestanden is wat hetzelfde. Bij cloud-init is het misschien wat omslachtig om te werken via de docker-compose package, en dit bestand gewoon aan te maken. Al moeten er met Ansible meer taken worden uitgevoerd om een container aan te maken. Alle pip packages moeten worden geïnstalleerd en de python-pip package zelf.

Ansible kan wel makkelijker worden uitgebreid en aangepast. Er moet gewoon een extra taak worden toegevoegd en er kan een extra docker container worden aangemaakt. Bij cloud-init kan dit ook worden gedaan via het docker-compose bestand. Maar als er meerdere containers blijven toegevoegd worden, gaat dit niet meer overzichtelijk zijn. Bij

Ansible kunnen alle aparte containers nog goed onderscheiden worden. Ook is de ouput bij Ansible hiervoor veel handiger. Er kan perfect worden meegevolg welke container al aangemaakt is, en welke nog niet.

9.2.2 Snelheid

Als we naar snelheden (Tabel 9.1) kijken is Ansible weer de beste. Zelfs met de aftrekking van de initïele startconfiguraties komen de andere 2 niet in de buurt. Dit is weer een duidelijk pluspunt voor Ansible.

Uitvoeringstijd	Resultaat 1	Resultaat 2	Resultaat 3
cloud-init	101 sec	138 sec	116 sec
Anisble	63.91 sec	30.16 sec	33.76 sec
cloud-init & Ansible	116.22 sec	105.49 sec	110.47 sec

Tabel 9.1: Snelheid tabel van container configuraties op de servers.

9.2.3 Resultaat

Voor containerization lijkt het in eerste instantie beter om voor Ansible te gaan. Het uitbreiden naar meerdere containers is het makkelijkst met Ansible, wat toch de doorslag geeft. De snelheid is ook een pluspunt.

Als er een docker server wordt opgezet met maar 1 container (of een klein aantal) kan er wel overwogen worden om voor cloud-init te gaan. Zeker als dit moeten worden opgezet bij de startup van de server. Want het meegeven van een cloud-init script, is nog steeds veel makkelijker dan het meegeven van een Ansible script.

Als er docker server moet worden opgezet met meerdere containers bij de start up, is het best om cloud-init en Ansible te gebruiken. Weer door het makkelijk meegeven van het cloud-init script. Hierin kan er dan ook makkelijker worden verwezen naar het Ansible playbook.

10. Conclusie

Doorheen de bachelorproef werd er gezocht naar een antwoord op de vragen. Kunnen Ansible en cloud-init samen functioneren, zo ja hoe ? Maakt cloud-init Ansible overbodig? Na een onderzoek kan er antwoord gegeven worden op die vragen.

Het resultaat wordt hierna vergeleken met de verwachte resultaten.

Ten laatste wordt er besproken hoe dit onderzoek kan worden uitgebreid in de toekomst. Zijn er bijkomende vragen ontstaan?

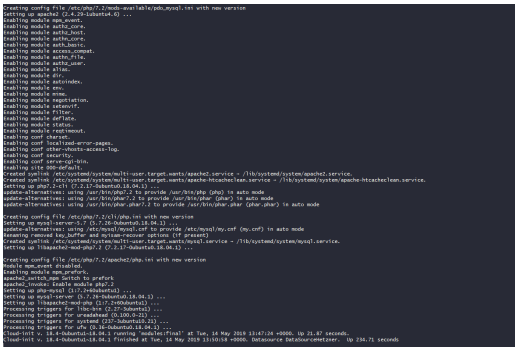
10.1 Antwoord op onderzoeksvragen

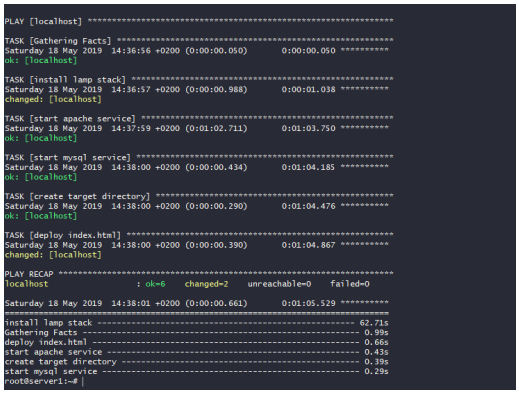
Maakt cloud-init Ansible overbodig? Nee, eigenlijk niet. In de hoofdstukken 7 en 9 werd gezien dat cloud-init, voor de meer geavanceerde configuratie, toch niet de juiste oplossing is. Als er servers moesten geconfigureerd worden met meerdere rules, schoot cloud-init nog te kort. De enige module waarin in kon gewerkt hiervoor was *runcmd*. Ook werd in hoofdstuk 8 gezien dat cloud-init geen optie heeft om wijzigingen door te voeren en het script een tweede keer te draaien. Alleszins niet in de setup die hier wordt gebruikt. Al kan cloud-init wel voor bepaalde dingen gebruikt worden. In hoofdstuk 6 werd gezien dat voor basisconfiguraties cloud-init misschien wel beter is dan Ansible. Cloud-init maakt Ansible dus zeker niet overbodig. Hiervoor zijn de modules te gelimiteerd. Maar als er een omgeving moet worden opgezet met juist basis configuraties, is het misschien aangeraden om cloud-init hiervoor te gebruiken.

Kunnen Ansible en cloud-init samen functioneren? Ja, dit kunnen ze zeker, maar of dit moet worden gedaan hangt af van de situatie. Het samenwerken van cloud-init zou eerder nuttig zijn voor als de server wordt aangemaakt, er meteen configuratie willen meegegeven worden. Als dit niet het geval is, is een samenwerking niet aan te raden. Ook is een samenwerking alleen van nut als de configuraties die worden gedaan meer zijn dan de basis. In hoofdstuk 6 werd gezien dat cloud-init de basis configuraties nog alleen kan doen zonder de hulp van Ansible.

Hoe kunnen ze samen functioneren? In praktijk lijkt het best om deze samen te gebruiken met elk zijn bepaalde functies. Cloud-init is in deze setup zeer goed als initieel script, doordat dit zo makkelijk kan worden meegegeven. In dit script kunnen eerste basis configuraties worden gedaan. Hierna wordt het Ansible playbook aangeroepen met de meer geavanceerde functies.

Als er een pure vergelijking wordt gedaan tussen de beide tools, is Ansible toch de winnaar. Ansible heeft, zoals hier al meerdere keren is aangehaald, veel meer functionaliteiten dan cloud-init. Ook heeft Ansible een veel duidelijkere output (Figuur 10.1). Bij cloud-init wordt veel info getoond als het script loopt, dit kan wat overweldigend zijn. Terwijl bij Ansible alles mooi is opgelijst in de verschillende taken. Ook is het hierdoor veel makkelijker om eventuele fouten uit een script te halen.





Figuur 10.1: Cloud-init en Ansible outputs.

10.2 Vergelijking met verwachte resultaten

De verwachtingen zijn toch gedeeltelijke anders met de uiteindelijke resultaten/conclusies. In het voorstel werd er verwacht dat elke omgeving een ander oplossing ging hebben. Ook werd er verwacht dat de taken van Ansible en cloud-init gingen variëren van server tot server.

Dit is niet helemaal lijn met het effectieve resultaat. Er werd verwacht dat elke omgeving zijn eigen verhaal ging hebben en dat is toch niet gebeurt. Er kwam altijd één constante boven water. Cloud-init is goed voor de basis maar als er geavanceerder wilt gegaan worden is Ansible duidelijk beter.

Uiteindelijk werd er verwacht dat cloud-init meer functionaliteiten ging hebben dat dat het uiteindelijk heeft. Er werd wel verwacht da Ansible ruimer ging zijn van mogelijkheden, aangezien dit veel populairder is. Maar er werd niet verwacht dat cloud-init er zoveel minder ging hebben.

10.3 Verdere uitbreiding?

Het onderzoek kan worden uitgebreid. Een restrictie die dit onderzoek had, is dat er maar één cloud omgeving kon worden gewerkt. Het zou interessant om te bekijken of er in andere omgevingen hetzelfde resultaat zou zijn of misschien iets helemaal anders. Ook de aanroeping van het cloud-init script en Ansible playbook kan op een andere manier. Door dit aan te aanroepen via een andere server, en deze scripts daardoor dus niet uit te voeren op de localhost.

De uiteindelijke vragen die na dit onderzoek kunnen worden gesteld zijn.:

- Hebben andere cloud-providers hetzelfde resultaat?
- Heeft een andere aanroeping van de scripts een effect op het resultaat?

A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1 Introductie

Voor de installatie van servers wordt al jaren Ansible gebruikt. Ansible is een universele 'taal' die taken voor servers automatiseert door middel van hun playbook. Zo een Ansible playbook is een georganiseerde unie van scripts dat het werk voor de server configuratie definieert.

Cloud-Init is momenteel een van de industrie standaarden voor het opbouwen van cloud servers, het maakt gebruik van cloud images. Dat zijn besturingssysteem sjablonen en elke instantie begint als een identieke kloon van elke andere instantie. De gebruikersgegevens geven elke cloud instantie haar persoonlijkheid. Doormiddel van cloud-int worden deze gegevens op de instantie toegepast.

Dit zijn allebei provisioning systemen. Op een verschillende manier doen ze in theorie hetzelfde. Ze brengen de server allebei naar de gewenste toestand van de gebruiker. Doordat deze op verschillende manieren werken, wordt er in de praktijk in combinatie met beide gewerkt. In dat geval gebruik je eerste cloud-init om de server naar een gewenste toestand te brengen waar na Ansible het kan overnemen.

Maar is dit de perfecte samenwerking? Het bedrijf Be-mobile is opzoek naar het antwoord. In dit onderzoek bestuderen we waar deze mekaar aanvullen en hoe ze dan op een perfect performante manier werken. Natuurlijk is het goed mogelijk dat deze mekaar niet aanvullen

en dan is het onderzoek waar en wanneer je voor wat moet kiezen en waarom dit mekaar overbodig maakt.

In dit onderzoek zullen we dit trachten te ontdekken. De vragen waar het, de antwoorden op wil vinden zijn:

- Vullen Ansible en Cloud-Init mekaar aan, of maken ze elkaar overbodig?
- Ook hoe ze mekaar aanvullen en/of hoe ze mekaar overbodig maken?

A.2 Stand Van Zaken

Ansible en Cloud-Init zijn geen onbekende voor mekaar dus er is wel een basis gevonden waarop er verder kan gewerkt worden. Maar zijn niet zo bekend dat de ultieme samenwerking al gevonden is. In het onderzoeken van de 2 systemen zijn er 5 artikels gevonden die een samenwerking beschrijven of afraden.

In de artikels **"An introduction to server provisioning with CloudInit"(Petersson, g.d.)**, **“Using Ansible to Bootstrap My Work Environment Part 4” (Harney, 2016b)** en **“Customizing Cloud Assembly Deployments with Cloud-Init” (Arkland, 2018)** is er een vaak voorkomend fenomeen gevonden. Meestal gebruiken de gebruikers/auteurs eerst cloud-init om de machine op te starten, daarna ansible voor de verdere specifieke eigenschappen.

Het artikel geschreven door Petersson (g.d.) is wel interessant doordat hij ook andere systemen buiten cloud-init heeft getest, waaronder puppet en chef. Maar toch verkiest hij om Ansible te gebruiken. Dit betekent dat een samenwerking met ansible optimaler is.

De artikels **“Zero Touch Provisioning of Infoblox Grid on OpenStack using Ansible”(Aditya, 2018)** en **"Automated Ansible AWX Installation"(„Automated Ansible AWX Installation”, g.d.)** beschrijven dan weer hoe beide elke als een apart systeem kunnen worden gebruikt zonder de hulp van de ander. Maar toch zijn er een paar gelijkenissen zo gebruikt „Automated Ansible AWX Installation” (g.d.) in "Automated Ansible AWX Installation"notities van een Ansible installatie om het met cloud-init te installeren.

Er is duidelijk al bekend dat er samengewerkt kan worden, maar ook soms weer niet. Maar criteria om te bepalen wanneer welke het best is, is nog niet bekend.

A.3 Methodologie

Dit onderzoek zal gevoerd worden door virtuele server omgevingen op te zetten met Cloud-init en/of Ansible. De programma’s die hierbij zullen worden gebruikt zijn VirtualBox, Vagrant en Atom.

Virtualbox is een programma om virtuele machines op te draaien. Met Vagrant kan je van

in je shell met een simpel commando virtuele machines op starten. Atom is dan weer een teksteditor om de configuratie goed in neer te pennen. Normaal is deze het vermelden niet waar maar door de overzichtelijke manier van werken die het aanbiedt is deze toch veel beter dan een gewone Notepad.

Met Ansible zal er voor verschillende servers een testomgeving worden opgezet. Dit zal dan ook gebeuren met cloud-init. Deze resultaten en omgevingen kunnen dan worden vergeleken met elkaar. De technische eigenschappen/resultaten worden dan onder de loep genomen: de performantie, de snelheid van het opstarten,... Ook zal er worden gekeken naar de config files van beiden en kan er worden gekeken welke daar per omgeving "beter" is. M.a.w. welke er op een kortere overzichtelijkere manier hetgeen kan bekomen. Be-Mobile kan ook nog verdere criteria bepalen waardoor we een keuze gaan maken. Daarna worden ook combinaties van de twee aangemaakt en dan kunnen we deze vergelijken met de originele servers.

Ook kan het zijn dat er vragen worden gesteld aan medewerkers van het bedrijf Be-Mobile om te bekijken wat hun mening over beide is.

A.4 Verwachte resultaten

De verwachtingen zijn dat een combinatie van cloud-init en Ansible meestal het beste zal zijn. Dat er voor elke server tot een bepaald moment cloud-init of ansible zal worden gebruikt waarna ansible of cloud-init het zal overnemen. Ook zijn de verwachtingen dat er voor elke omgeving een andere oplossing zal zijn. Er zal veel afhangen van het type besturingssysteem dat draait en van de servers om te kunnen bepalen welke het best wordt gekozen.

De verwachtingen zijn ook dat niet bij alles een combinatie zal worden gebruikt. Ook kan het misschien zijn dat het beter is dat je één van beide kiest voor een bepaalde server omgeving.

A.5 Verwachte conclusies

De verwachte conclusie is dat elke server omgeving een andere uitkomst zal hebben. Er veel zal afhangen van welke server je precies wilt draaien op welk systeem. Dan kan er worden gekozen voor een bepaalde combinatie van beide of misschien één van beide. De conclusie zal niet zijn welke van de twee beter is. Maar eerder hoe ze het best gebruikt worden per serveromgeving.

B. Verklarende woordenlijst

Server Configuration Management Tool: Dit zijn tools die zorgen de voor automatisatie en configuratie van servers.

Chef: Chef is een server configuration management tool. Het zorgt voor de automatische configuratie op servers. Bij Ansible wordt er via playbooks gewerkt. Chef werkt met 'recepten', vandaar de naam.

Puppet: Puppet is ook een server configuration management tool. Samen met Ansible is dit de populairste.

Salt: Ook Salt is een server configuration management tool.

Cloud Server Provider: Dit is een bedrijf dat cloud computing aanbiedt. Dit kan zijn het aanmaken van servers op de cloud.

Hetzner Cloud: Hetzner Cloud is een cloud server provider.

Vagrant: Vagrant is een tool die virtuele machines aanmaakt en beheert. Vagrant werkt wel altijd samen met een ander programma dat deze dan draait.

VirtualBox: VirtualBox is een virtualisatie programma. Hiermee worden virtuele machines gedraaid op een computer.

Pull: Pull betekent aanvragen van een programma of computer.

Push: Push is het tegenovergestelde van Pull. Push stuurt data naar een programma zonder eerst een aanvraag te doen.

Deployment: Software deployment zijn alle activiteiten ervoor zorgen dat een software systeem klaar is voor gebruik.

Yaml bestand: (YAML Ain't Markup Language) Dit is een soort bestand dat vooral gebruikt wordt als configuratie bestand.

Package: In Linux worden alle applicaties in packages gestoken.

CloudFormation: Dit is een service dat helpt om Amazon Web Services op te zetten en te vormen.

Amazon EC2: Dit is een webservice van Amazon voor de Cloud Servers te beheren en configureren.

AWS: (Amazon Web Services) Dit is een Server Cloud Provider van Amazon.

Idempotentie: Een programma heeft idempotentie als, het programma niet meer verandert wanneer de operatie nog is wordt uitgevoerd.

Bootstrappen: Het laden van een besturingssysteem.

VPC: (Virtual Private Cloud) Een private cloud oplossing in een publieke cloud infrastructuur.

Block devices: Dit is een opslag apparaat dat bestanden leest en schrijft in 'fixed-size' blocks.

Ubuntu: Een Linux besturingssysteem.

CloudSigma: Dit is een Zwitserse server cloud provider.

SSH Keys: Manier om toegang te verkrijgen tot een server. Via een publieke sleutel, heb je toegang tot een systeem die bijpassende private sleutel heeft.

FQDN: (Fully Qualified Domain Name) Dit het volledige domein adres.

Git: Git is een gratis opensource distributiesysteem. Het is een makkelijk tool waar projecten op kunnen worden bewaard of worden gedeeld met anderen.

Repository (repo): Dit is een map of bestandslocatie waar alle projectbestanden staan.

BitBucket: Bitbucket is een repository hosting systeem dat git ondersteunt. Het wordt vooral gebruikt door bedrijven.

Vagrantfile: Het configuratie bestand voor het aanmaken van een virtuele machine met Vagrant.

Vagrant box:Box's zijn het package formaat voor vagrant omgevingen.

Makefile: het configuratie bestand om met het commando *make* een programma te bouwen.

VPN: (Virtual Private Network) Dit creëert een veilige geëncrypteerde connectie over een minder veilige netwerk, meestal een publiek netwerk.

Python: Python it is een programmeertaal.

Pip: Pip is een package beheerder, die packages geschreven in python installeert.

Ruby: Ruby is een programmeertaal

Base64: Base64 is een tool om binaire data te coderen of decoderen naar een tekst formaat.

Bibliografie

Aditya. (2018). Zero Touch Provisioning of Infoblox Grid on OpenStack using Ansible. *Info Blox*. Verkregen van https://community.infoblox.com/t5/Community-Blog/Zero-Touch-Provisioning-of-Infoblox-Grid-on-OpenStack-using/ba-p/14910

Arkland, C. D. (2018). Customizing Cloud Assembly Deployments with Cloud-Init. *Blog VMware*. Verkregen van https://blogs.vmware.com/management/2018/11/customizing-cloud-assembly-deployments-with-cloud-init.html

Automated Ansible AWX Installation. (g.d.). *devendor tech*. Verkregen van https://www.devendortech.com/articles/AWX_Tower.html

Bekker, R. (2018). Setup a LAMP Stack With Ansible Using Ubuntu. Verkregen van https://blog.ruanbekker.com/blog/2018/07/08/setup-a-lamp-stack-with-ansible-using-ubuntu/

Butcher, M. (2013). Instant LAMP Server: Using Cloud-Init to Pre-configure Cloud Compute Instances. Verkregen van http://technosophos.com/2013/04/25/instant-lamp-server-using-cloud-init-pre-configure-cloud-compute-instances.html

cloud-init.io. (2019). *cloud-init-docs-modules*. cloud-init. Verkregen van https://cloudinit.readthedocs.io/en/latest/topics/modules.html

Ellingwood, J. (2014). How To Use Git to Manage your User Configuration Files on a Linux VPS. Verkregen van https://www.digitalocean.com/community/tutorials/how-to-use-git-to-manage-your-user-configuration-files-on-a-linux-vps

Harney, S. (2016a). Using Ansible to Bootstrap My Work Environment Part 3. Verkregen van https://www.scottharney.com/using-ansible-to-bootstap-my-work-environment-part-3/

Harney, S. (2016b). Using Ansible to Bootstrap My Work Environment Part 4. Verkregen van https://www.scottharney.com/using-ansible-to-bootstap-my-work-environment-part-4/

Hassin, I. (2013). Ansible MySQL. Verkregen van https://gist.github.com/ihassin/8106956

Hetzner. (g.d.). *Site Hetzner*. Verkregen van https://www.hetzner.com/

Hetzner, W. (g.d.). *Wiki Hetzner - Can I use Cloud-Init when creating servers?* Verkregen van https://wiki.hetzner.de/index.php/CloudServer/en#Can_I_use_Cloud-Init_when_creating_servers.3F

Irvin, D. (2016). Example Ansible Playbook. Verkregen van https://blog.rackspace.com/how-to-deploy-ansible-accessible-explanation

Moser, S. & Harlow, J. (2019). cloud-init.io. Verkregen van https://cloud-init.io/

NMDias. (2017). How to setup MySQL with cloud-init? Verkregen van https://www.digitalocean.com/community/questions/how-to-setup-mysql-with-cloud-init

What is Docker? (2019). Verkregen van https://opensource.com/resources/what-docker

Petersson, V. (g.d.). An introduction to server provisioning with CloudInit. *Cloudsigma*. Verkregen van https://www.cloudsigma.com/an-introduction-to-server-provisioning-with-cloudinit/

Petersson, V. (2017). An introduction to server provisioning with CloudInit. Verkregen van https://www.cloudsigma.com/an-introduction-to-server-provisioning-with-cloudinit/

Potter, J. (2019). What is a LAMP stack? Verkregen van https://www.liquidweb.com/kb/what-is-a-lamp-stack/

RedHat. (2017). *ANSIBLE IN DEPTH*. Red Hat Inc. Verkregen van https://www.ansible.com/hubfs/pdfs/Ansible-InDepth-WhitePaper.pdf

Scaleway. (g.d.). How to use Cloud-Init to configure your server at first boot. Verkregen van https://www.scaleway.com/en/docs/how-to-use-cloud-init-to-configure-your-server-at-first-boot/

Skinner, M. & Heldebrant, M. (2017). *Cloud-Init*. RedHat. Verkregen van https://people.redhat.com/mskinner/rhug/q3.2014/cloud-init.pdf

Tomkins, A. (2016). Testing cloud-init with Vagrant. Verkregen van https://www.alextomkins.com/2016/09/testing-cloud-init-with-vagrant/

Tool Base 64 encoding. (g.d.). Verkregen van https://codebeautify.org/base64-encode

Tool makepasswd. (g.d.). Verkregen van https://www.mkpasswd.net/index.php

Young, J. (2019). Ansible with Vagrant on Windows. Verkregen van https://blog.zencoffee.org/2016/08/ansible-vagrant-windows/