



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

Maken als bachelorproef helemaal gedaan is.

Samenvatting

Maken als bachelorproef klaar is.

DIE AFBEELDING bibliografie van die mail van school check up voor huidige afbeeldingen

TEST criteria method op einde aanpassen

Inhoudsopgave

1	Inleiding	15
1.1	Achtergrond	15
1.2	Context	15
1.3	Probleemstelling - Onderzoeksvraag	16
1.4	Opzet van deze bachelorproef	16
2	Inleiding tot Ansible en cloud-init	17
2.1	Ansible	17
2.1.1	Architectuur	17
2.1.2	Playbook - Roles	18
2.1.3	Omgevingen	19
2.2	Cloud-Init	19
2.2.1	Modules	19

2.2.2	User Data - Cloud config	21
2.2.3	Omgevingen	22
2.2.4	Systemen	22
3	Literatuurstudie	23
3.1	An introduction to server provisioning with CloudInit	23
3.1.1	Verschil en samenwerking met Ansible volgens Viktor Petersson	24
3.1.2	Setup cloud config bestand	24
3.2	Using Ansible to Bootstrap My Work Environment Part 4	27
3.2.1	Ansible initial playbook	27
3.2.2	Cloud-init	27
3.2.3	Ansible follow up playbook	30
4	Methodologie	39
4.1	Testomgevingen	39
4.1.1	Lokaal	40
4.1.2	Cloud	41
4.2	Testcriteria	42
5	Opzetten lokale testomgeving	43
5.1	Cloud-init	43
5.1.1	Maken van ISO met cloud-init configuraties	43
5.1.2	Opzetten van testomgeving met cloud-init configuraties	44
5.2	Ansible	45
5.3	Cloud-init & Ansible	46
5.3.1	Ansible playbook koppelen aan cloud-init	46

5.3.2	Uitvoeren playbook	47
6	Opzetten Hetzner Cloud testomgeving	49
6.1	Aanmaken server met Hetzner	49
6.2	Cloud-init	50
6.3	Ansible	50
6.4	Ansible & cloud-init	51
7	Basisconfiguraties op de servers	53
7.1	Aanmaken configuratie bestanden	53
7.1.1	Opstellen cloud-init config bestand	54
7.1.2	Opstellen Ansible playbook	55
7.1.3	Ansible & cloud-init omgevingen	57
7.2	Uitvoering & resultaten	59
7.2.1	Lokaal	59
7.2.2	Cloud	59
8	Server installatie en configuratie	61
8.1	Webserver	62
8.1.1	Cloud-init	62
8.1.2	Ansible	62
8.1.3	Cloud-init & Ansible	62
8.2	Fileserver	62
8.2.1	Cloud-init	62
8.2.2	Ansible	62
8.2.3	Cloud-init & Ansible	62

8.3	Mailserver	62
8.3.1	Cloud-init	62
8.3.2	Ansible	62
8.3.3	Cloud-init & Ansible	62
8.4	Resultaten	62
8.4.1	Lokaal	62
8.4.2	Cloud	62
9	Instellingen aanpassen na het opstarten	63
10	Container & Cluster Configuratie	65
11	Conclusie	67
A	Onderzoeksvoorstel	69
A.1	Introductie	69
A.2	Stand Van Zaken	70
A.3	Methodologie	70
A.4	Verwachte resultaten	71
A.5	Verwachte conclusies	71
B	Verklarende woordenlijst	73
	Bibliografie	77

Lijst van figuren

2.1	Voorbeeld van een Ansible playbook.	18
2.2	Ansible Galaxy site met lijst van roles.	19
2.3	Voorbeeld User-Data Script.	21
2.4	Voorbeeld Cloud Config Data.	22
4.1	Voorbeeld van een vagrantfile.	40
4.2	Foto van de laptop.	41
5.1	Tool voor encoding.	47
7.1	Mappen structuur basis configuraties.	54

Lijst van tabellen

4.1	Specificaties van de laptop.	41
7.1	Resultaten tabel van Basisconfiguraties op de lokale servers.	59
7.2	Resultaten tabel van Basisconfiguraties op de cloud servers.	59

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, bij

Gent.

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 en 6 wordt getoond hoe de 2 testomgevingen zijn opgezet.

In Hoofdstuk 7, 8, 9 en 10 wordt het effectieve onderzoek uitgevoerd.

In Hoofdstuk 11, 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 algemene uitleg gegeven, daarna wordt iets dieper op ingegaan op de architectuur, playbooks, roles en de omgeving. Daarna wordt er voor cloud-init hetzelfde gedaan. 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 bestuurd 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.

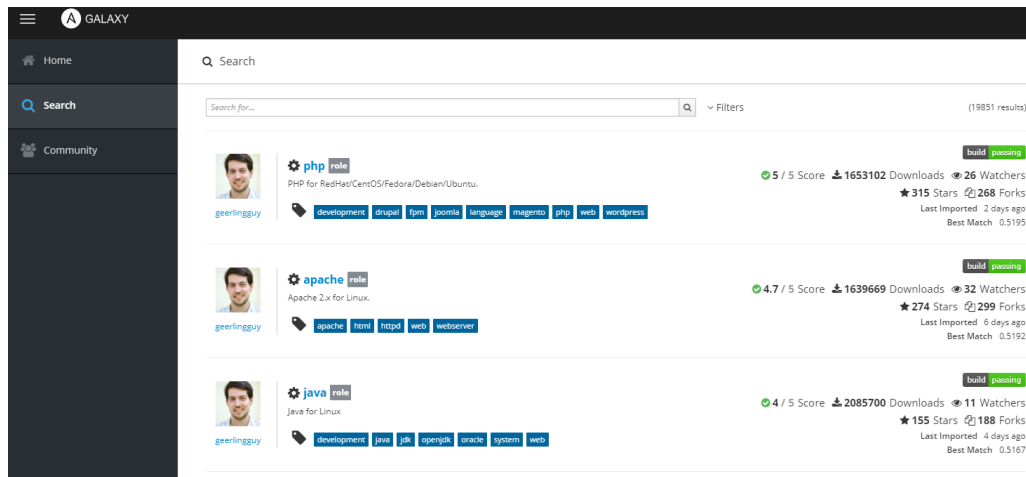
Deze playbooks bevatten verschillende “plays” die de automatisatie definiëren over verschillende hosts. Deze hosts staan bekend als de inventory. Elke “play” bevat verschillende taken die één, enkele of alle hosts moeten uitvoeren. Elke taak roept een Ansible module aan, een klein stukje code dat een 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.

```
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.

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: 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 infrastructures 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 bestandssystemen 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 geven. 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 Pipelining 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.

```
#!/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.

```
#cloud-config

package_upgrade: true

users:
  - default
  - name: maarten
    passwd: $6$Y5N.ote2cyDmb0l0$EizY59wJIIt4hMQwyFXIRpJtk.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_upgrade*. Terwijl er in Hoofdstuk 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 besproken 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 configuration manager

Een laatste voordeel volgens (Petersson, 2017) is dat als meer geavanceerde gebruiker er ook de mogelijkheid is om een extra Server Configuration 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 follow up playbook': in dit gedeelte wordt de EC2 instance geconfigureerd met Ansible.

'Launch and configure': hier wordt de omgeving opgestart en worden sommige variabelen 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, is 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 adres 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 configfiles 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 de

- 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 o

```

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 laatste deel configureert 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/ansible-mystuff
```

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 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 aanzet 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=MASQUER
```

Sectie 3 OpenVPN rol

Volgens (Harney, 2016b) was dit het moeilijkste 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 2 testomgevingen.

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 Testomgevingen

Voor het onderzoek moeten er verschillende testomgevingen worden opgesteld, om alles te kunnen testen. Er is een lokale virtuele omgeving en een omgeving op cloud servers. Er is gekozen voor deze twee omgevingen omdat dit in de praktijk de twee meest voorkomende omgevingen zijn bij server management tools.

Ansible is een tool die gebruikt wordt voor virtuele omgevingen door gebruikers thuis, maar ook in grote cloud omgevingen door bedrijven. Het zou dus niet correct zijn om het alleen lokaal te testen of alleen op de cloud.

Voor het onderzoek is het eveneens interessant omdat er veel meer kans is op verschillende resultaten. Die kunnen dan met elkaar worden vergeleken en kan er worden bekeken wat de oorzaak ervan is. Dit geeft ook de mogelijkheid om het onderzoek in de toekomst verder uit te breiden. Waarom bijvoorbeeld Ansible beter is lokaal zonder cloud-init dan met (dit is een hypothetische stelling).

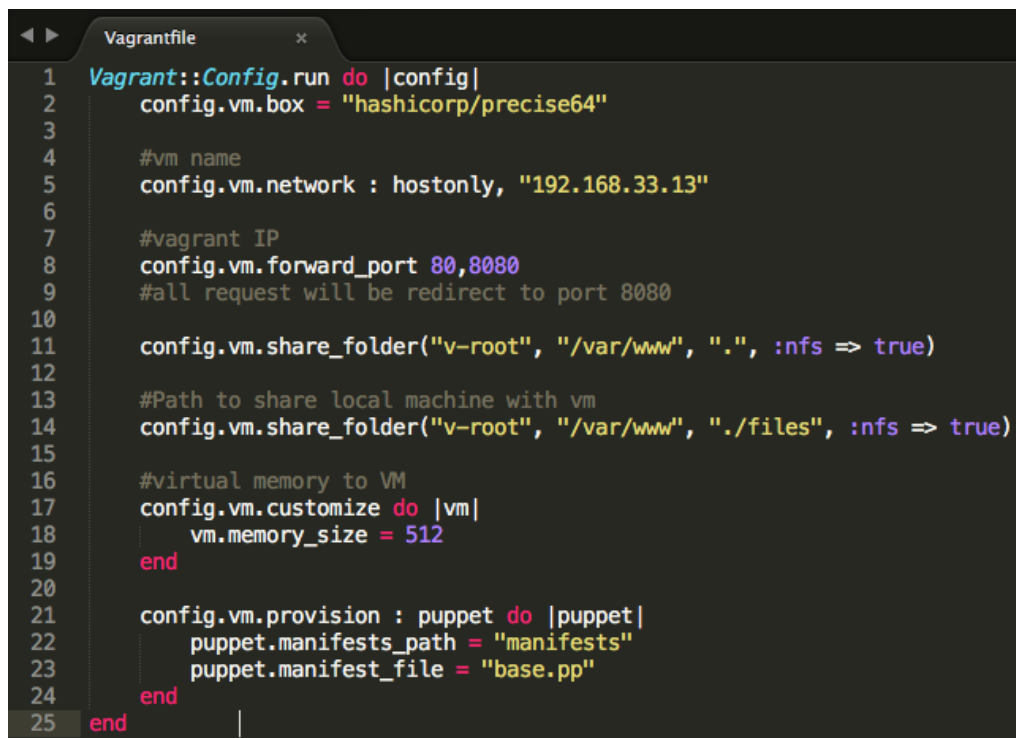
Per omgeving is er hieronder wat meer informatie te vinden. Over het opzetten van de omgevingen zijn voor beide aparte hoofdstukken aan toegewijd, namelijk: Hoofdstuk 5 en Hoofdstuk 6.

4.1.1 Lokaal

Voor de lokale omgeving zal er worden gewerkt met 2 technologieën, namelijk: VirtualBox en vagrant.

VirtualBox is een programma waarmee je virtuele machines kunt aanmaken en beheren. Hiermee worden de servers lokaal aangemaakt.

Het tweede programma dat wordt gebruikt is vagrant. Vagrant is een command tool die die servers configureerd. In de actieve directory wordt het commando *vagrant init* gedaan, dan staat er een *vagrantfile*. Dit is het bestand dat kan worden geconfigureerd naargelang de wens van de gebruiker. Daarna wordt het commando *vagrant up* gedaan. Dit start de server op. De server wordt opgestart met behulp van een tool die virtuele machines beheert en aanmaakt. In dit geval dus VirtualBox.

A screenshot of a code editor window titled 'Vagrantfile'. The code is written in Ruby and defines a Vagrant virtual machine configuration. It includes settings for the VM box (hashicorp/precise64), network (hostonly, 192.168.33.13), forward port (80, 8080), shared folders (v-root to /var/www and ./files), virtual memory (512 MB), and puppet provisioning (manifests_path, manifest_file).

```
1 Vagrant::Config.run do |config|
2   config.vm.box = "hashicorp/precise64"
3
4   #vm name
5   config.vm.network : hostonly, "192.168.33.13"
6
7   #vagrant IP
8   config.vm.forward_port 80,8080
9   #all request will be redirect to port 8080
10
11   config.vm.share_folder("v-root", "/var/www", ".", :nfs => true)
12
13   #Path to share local machine with vm
14   config.vm.share_folder("v-root", "/var/www", "./files", :nfs => true)
15
16   #virtual memory to VM
17   config.vm.customize do |vm|
18     vm.memory_size = 512
19   end
20
21   config.vm.provision : puppet do |puppet|
22     puppet.manifests_path = "manifests"
23     puppet.manifest_file = "base.pp"
24   end
25 end
```

Figuur 4.1: Voorbeeld van een vagrantfile.

Laptop

Deze testomgeving wordt opgezet op de laptop: **Asus X750L**. Asus bracht deze laptop eind 2013 op de markt. De laptop is aangekocht in augustus 2014, en is bij het uitvoeren van het onderzoek bijna 5 jaar oud. Bijna alle specificaties zijn hetzelfde toen hij werd aangekocht. Alleen is de harde schijf vervangen van een HDD van 500 gigabyte naar een SSD van 500 gigabyte. Deze vervanging werd begin 2019 gedaan en bij het uitvoeren van het onderzoek is dat 2-3 maand geleden. Hieronder is een uitgebreide tabel met specificaties van de laptop. De data is verkregen door: (Hardware.info, 2013).

Specificaties	
Fabrikant	Asus
Model	ASUS x750L
Besturingssysteem	Windows 10
CPU	Intel Core i7 4500U @ 2.4 GHz
Geheugen	8GB DDR3 @ 1600MHz
GPU	Nvidia GeForce GT 740M
Interne schijven	Crucial MX500 (500 GB)

Tabel 4.1: Specificaties van de laptop.



Figuur 4.2: Foto van de laptop.

4.1.2 Cloud

Voor de cloud 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), Falkenstein (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.-a).

4.2 Testcriteria

Het volgende dat wordt besproken is de keuze van de testcriteria die worden gekozen in hoofdstukken 7, 8, 9 en 10.

In Hoofdstuk 7 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 kan bijvoorbeeld gebruikers aanmaken zijn, maar ook: commando's uitvoeren, mappen aanmaken.

In Hoofdstuk 8 worden verschillende servers 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.

In Hoofdstuk 9 wordt gekeken met welke optie je de server best aanpast na het opstarten. Eens de server is opgestart maar er wordt een kleinigheid gewijzigd in het script. Welke optie is dan het best om snel deze verandering door te voeren.

Ten laatste wordt in Hoofdstuk 10 de configuratie van containers en clusters onderzocht. Bijna elk bedrijf gebruikt containers en clusters 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?

5. Opzetten lokale testomgeving

In dit hoofdstuk wordt besproken hoe de lokale testomgevingen zijn opgezet. Er zijn 3 lokale testomgevingen. De eerste is één die cloud-init gebruikt, de tweede gebruikt Ansible en de derde gebruikt cloud-init en Ansible. Voor het opzetten wordt er VirtualBox en Vagrant gebruikt.

5.1 Cloud-init

De eerste testomgeving die wordt besproken is de omgeving met cloud-init. Deze wordt opgezet met behulp van: (Tomkins, 2016). Er wordt hiervoor in 2 stappen gewerkt. Eerst wordt een ISO met de cloud-init configuraties aangemaakt met behulp van de setup van (Tomkins, 2016). Daarna wordt deze ISO toegevoegd aan de testomgeving.

5.1.1 Maken van ISO met cloud-init configuraties

Voor het aanmaken van de ISO er een Linux omgeving nodig. Allereerst wordt er een omgeving opgezet om de ISO's in te maken. Deze wordt ook opgezet met Vagrant en VirtualBox. De box *ubuntu/xenial64* wordt gebruikt (dit is dezelfde als degene die later in de testomgeving wordt gebruikt). Met het commando *vagrant up* wordt de server aangemaakt. De vagrantfile ziet er zo uit:

```
Vagrant.configure('2') do |config|  
  config.vm.box = 'ubuntu/xenial64'  
end
```

Via het commando *vagrant ssh* is er toegang tot de server. Zodra deze toegang tot de machine er is, wordt de package *genisoimage* geïnstalleerd. Dit gebeurt via het commando *sudo apt-get install genisoimage*.

Nadat die package geïnstalleerd is, wordt een user-data bestand aangemaakt. In dit bestand komt de cloudconfig data. Hierna wordt een tweede bestand aangemaakt, het meta-data bestand. In dit bestand komt nog extra minimale configuratie. De local-hostname en instance-id moeten hierin worden gezet.

Als deze bestanden zijn aangemaakt wordt een Makefile gecreëerd. Via dit bestand wordt de iso aangemaakt. De Makefile ziet er als volgt uit:

```
nocloud.iso: meta-data user-data
mkisofs \
-joliet -rock \
-volid "cidata" \
-output nocloud.iso meta-data user-data
```

Ten laatste wordt het *make* package geïnstalleerd. Via het commando *make* wordt het iso bestand aangemaakt.

5.1.2 Opzetten van testomgeving met cloud-init configuraties

Een tweede omgeving wordt aangemaakt (de effectieve testomgeving) opnieuw doormiddel van Vagrant en VirtualBox. Net zoals in het vorige gedeelte wordt ook hier de box *ubuntu/xenial64* gebruikt.

In het vorige gedeelte is het iso bestand aangemaakt met de cloud-init configuraties. Allereerst moet dit iso bestand worden gekopieerd naar de map van de omgeving. In de *vagrantfile* die wordt gebruikt moet worden doorverwezen naar deze iso. Dit wordt gedaan met deze configuraties in de *vagrantfile*.

```
IMAGE_PATH = File.join(File.dirname(__FILE__), "no-cloud.iso")
```

```
Vagrant.configure(2) do |config|
```

```
  config.vm.box = "ubuntu/xenial64"
```

```
  config.ssh.password = nil
```

```
  config.vm.synced_folder ".", "/vagrant", disabled: true
```

```
  config.vm.provider :virtualbox do |vb|
```

```
    vb.customize [
```

```
      "storageattach", :id,
```

```
      "--storagectl", "SCSI",
```

```
      "--port", "1",
```

```
      "--type", "dvddrive",
```

```

"--medium" , IMAGE_PATH
]
vb.linked_clone = true
end
end

```

Als hierna *vagrant up* wordt gedaan, wordt de server aangemaakt met de configuraties van cloud-init.

5.2 Ansible

De tweede lokale testomgeving die wordt opgezet is die met Ansible. Deze is minder complex om op te zetten dan degene met cloud-init. Dat er maar één machine nodig is. Deze wordt opgezet met behulp van (Young, 2019). Ook deze omgeving gebruikt Vagrant en VirtualBox. Als box wordt ook opnieuw dezelfde gebruikt als de vorige keren.

Het opzetten van de testomgeving gebeurt in 4 stappen.

Allereerst wordt een vagrantfile aangemaakt. In dit vagrantfile wordt verwezen naar het playbook, de inventory en de map waar deze zich bevinden. Via dit vagrantfile is het ook mogelijk om ansible uit te voeren met vagrant via een Windows systeem. Het vagrantfile ziet er als volgt uit.:

```

# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/xenial64"

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
  end

  config.vbguest.auto_update = false
  config.vbguest.no_install = false
  config.vbguest.no_remote = true

  config.vm.hostname = "ansible-teslocal"

  config.vm.provision "ansible_local" do |ansible|
    ansible.provisioning_path = "/vagrant/provisioning"
    ansible.inventory_path = "inventory"
    ansible.playbook = "playbook.yml"
    ansible.limit = "all"
  end
end

```

end

Hierna wordt de map *provisioning* aangemaakt in de map van de omgeving. In die map worden 3 bestanden aangemaakt. Allereerst een *ansible.cfg* bestand. Hierin worden configuraties gezet zodat Vagrant en Ansible niet "paniker" over ssh sleutels.

```
[ defaults ]
host_key_checking = no
[ ssh_connection ]
ssh_args = -o ControlMaster=auto -o ControlPersist=60s -o UserKnownHos
```

Het tweede bestand is een inventory bestand. Hierin wordt gezet welke host aanwezig is en welke connectie deze heeft.

```
local-test ansible_connection=local
```

Het derde bestand dat wordt aangemaakt is het effectieve playbook.

Het laatste dat wordt gedaan is het installeren van de vagrant-plugin *vbguest*. Dit wordt gedaan met het commando *vagrant plugin install vagrant-vbguest*.

Als nu *vagrant up* wordt gedaan, wordt de server opgestart met de configuraties van Ansible.

5.3 Cloud-init & Ansible

De derde omgeving die wordt opgezet is één met Ansible en cloud-init. Zoals wordt beschreven zijn er voor Puppet en Chef aparte modules om configuraties uit te voeren. Voor Ansible is dit echter niet het geval. Dus moet een creatieve oplossing gezocht worden. Voor de combinatie van Ansible en cloud-init wordt met de *write_files* module in cloud-init gewerkt.

5.3.1 Ansible playbook koppelen aan cloud-init

Een server wordt lokaal aangemaakt met de manier van cloud-init. Er wordt wel een bestand toegevoegd met module *write_files*. Dit bestand is het playbook van de server. De content van het bestand moet wel worden geëncodeerd via de Base64 standaard. Dit wordt gedaan via de tool: (**toolbas64**). In de tekst wordt de te encodere tekst gezet. Daarna wordt in het andere tekstvak de geëncodeerde versie gegeven. Met deze tool wordt het playbook geëncodeerd.

Base64 Encode☆

The screenshot shows a web interface for a Base64 encoding tool. At the top, the title 'Base64 Encode☆' is displayed. Below it, a text input area is labeled 'Enter the text to Base64 Encode' and contains the text: `- hosts: pr010
become: true
roles:`. To the right of the input area is a 'get sample' button. Below the input area are three buttons: 'Encode', 'Load', and 'Browse'. Below these buttons, the output section is labeled 'The Base64 Encoded:' and displays the encoded string: `LSBob3N0czogcHlwMTAKICBIZWNvbWU6IHRydWUKICByb2xlcz0=`. A copy icon is visible to the right of the output text.

Figuur 5.1: Tool voor encoding.

5.3.2 Uitvoeren playbook

Als het playbook in de *write_files* module staat, moeten nog 2 dingen gebeuren.

Allereest moet de package *ansible* worden geïnstalleerd via cloud-init. Dit werd gedaan door *ansible* toe te voegen aan de module *packages*.

Ook moet het playbook worden aangeroepen om te draaien. Dit gebeurt door bij de module *runcmd* het commando *ansible-playbook -i, <aangemaakt playbook>* te zetten.

Via deze omgeving kunnen cloud-init en Ansible configuraties worden gedaan.

6. Opzetten Hetzner Cloud testomgeving

In dit hoofdstuk wordt besproken hoe de testomgevingen op Hetzner zijn opgezet. Het zijn weer de 3 zelfde omgevingen als in het vorige hoofdstuk: namelijk één met cloud-init, één met Ansible en één met cloud-init en Ansible.

6.1 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 verkregen.

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 belangrijke 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 verwijdt een server.

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

6.2 Cloud-init

Het opzetten van een omgeving met cloud-init in Hetzner Cloud is eigenlijk zeer simpel. Op (Hetzner, g.d.-b) 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 vaarbale 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>
```

6.3 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>
```

6.4 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. Daarna 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 zal github de ssh sleutel gebruiken die is toegevoegd.

7. Basisconfiguraties op de servers

In dit hoofdstuk wordt voor het eerst een onderzoek uitgevoerd.

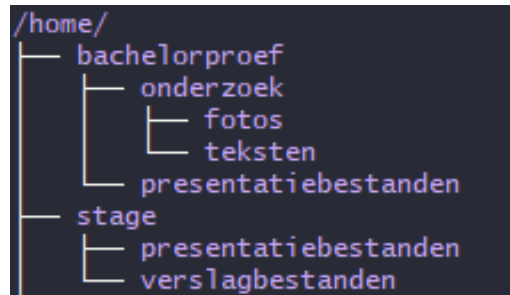
Er wordt voor elke server een configuratie bestand gemaakt dat 4 basisconfiguraties zal uitvoeren: gebruikers en groepen toevoegen, packages installeren en updates, mappen structuur aanmaken en ssh configuratie.

7.1 Aanmaken configuratie bestanden

In dit hoofdstuk zal uitgelegd 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 admin zijn en 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 7.1: Mappen structuur basis configuraties.

7.1.1 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*. Sudo werd op *true* gezet zodat de gebruiker sudo rechten heeft. Bij *groups* de naam van de

aangemaakte groep. Het wachtwoord onder `passwd`, proef, werd geëncrypteerd met de tool: (**toolmkpass**). Als shell werd ook voor `/bin/bash` gekozen.

```
groups :
- test

users :
- default
- name: bachelor
- passwd: 985b56433efe9898290b88d4dab853a2f09d7eb7a7b1b8d2cdd431
- sudo: true
- 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 *mkdir* commando's. Dit is het commando voor het aanmaken van mappen.

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

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.

7.1.2 Opstellen Ansible playbook

Als tweede werd het Ansible playbook opgesteld. Als host wordt gekozen voor *local-test* en *localhost*, afhankelijk of het de lokale of cloud setup is. 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 direcorey fotos
file:
path: /home/bachelorproef/onderzoek/fotos
state: directory
– name: Creates direcorey teksten
file:
path: /home/bachelorproef/onderzoek/teksten
state: directory
– name: Creates direcorey bachelor presentatie
file:
path: /home/bachelorproef/presentatiebestanden
state: directory
– name: Creates direcorey stage presentatie
file:
path: /home/stage/presentatiebestanden
state: directory
– name: Creates direcorey 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 meegegeven. Ook de groups werden meegegeven waaronder de admin group zodat de user sudo rechten heeft.

```
- name: add group test
group:
name: test
state: present
- name: add user bachelor
user:
name: bachelor
groups: test , admin
shell: /bin/bash
password: 985b56433efe9898290b88d4dab853a2f09d7eb7a7b1b8d2cdd431e2920c3
```

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. Voor de lokale setup wordt dit via een bestand gedaan in de gedeelde map *vagrant*. Voor de cloud setup wordt dit opgehaald via git.

```
- name: Set authorized key taken from file
authorized_key:
user: charlie
state: present
key: "{{ lookup('file', '/vagrant/key') }}"

- name: Set authorized key taken from file
authorized_key:
user: charlie
state: present
key: http://www.github.com/path-to-key
```

7.1.3 Ansible & cloud-init omgevingen

Beide bestanden die apart gemaakt zijn, zijn vrij complex loos. 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.

Verwijzig Ansible lokaal

De verwijzing naar Ansible in de lokale setup zal gebeuren via de module *write_files*. Met de tool (**toolbas64**) werd het playbook gecodeerd. Hieronder is het playbook dat wordt gecodeerd.

```

- hosts: localhost
tasks:
- name: Creates direcorey fotos
  file:
  path: /home/bachelorproef/onderzoek/fotos
  state: directory
- name: Creates direcorey teksten
  file:
  path: /home/bachelorproef/onderzoek/teksten
  state: directory
- name: Creates direcorey stage presentatie
  file:
  path: /home/bachelorproef/presentatiebestanden
  state: directory
- name: Creates direcorey bachelor presentatie
  file:
  path: /home/stage/presentatiebestanden
  state: directory
- name: Creates direcorey stage verslag
  file:
  path: /home/bachelorproef/stage/verslagbestanden
  state: directory

```

In cloud-init wordt in de module *write_files* het bestand aangemaakt. In de module *runcmd* wordt dan het playbook aangeroepen met een commando.

```

write_files:
- path: "/home/ansible/playbook.yaml"
  encoding: "b64"
  content: Ci0gaG9zdHM6IGxvY2FsaG9zdAogIHRhc2tzOgogICAgLSBuYW1lOiBDcmVhdG
runcmd:
- ansible-playbook -i , /home/ansible/playbook.yaml

```

Verwijzing Ansible cloud

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*, *no_ssh_fingerprints* en *ssh_keys*. In de module *ssh_keys* wordt de private sleutel toegevoegd connectie legt met de git repo. In de module *runcmd* worden 2 commando's gezet: de git clone en het uitvoeren van het

playbook. *no_ssh_fingerprints* wordt op true gezet. Zo geeft het geen error dat de server de host niet kent.

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

7.2 Uitvoering & resultaten

7.2.1 Lokaal

	Uitvoeringstijd	Aantal lijnen code
cloud-init	0 sec	24
Anisble	0 sec	53
cloud-init & Ansible	0 sec	0

Tabel 7.1: Resultaten tabel van Basisconfiguraties op de lokale servers.

7.2.2 Cloud

	Uitvoeringstijd	Aantal lijnen code
cloud-init	0 sec	0
Anisble	0 sec	53
cloud-init & Ansible	0 sec	0

Tabel 7.2: Resulaten tabel van Basisconfiguraties op de cloud servers.

Meerdere server bekijken bij cloud

8. Server installatie en configuratie

Dit hoofdstuk voert een tweede onderzoek.

Er worden 3 type servers gemaakt: een webserver, een fileserver en een mailserver.

8.1 Webserver

8.1.1 Cloud-init

8.1.2 Ansible

8.1.3 Cloud-init & Ansible

8.2 Fileserver

8.2.1 Cloud-init

8.2.2 Ansible

8.2.3 Cloud-init & Ansible

8.3 Mailserver

8.3.1 Cloud-init

8.3.2 Ansible

8.3.3 Cloud-init & Ansible

8.4 Resultaten

8.4.1 Lokaal

8.4.2 Cloud

9. Instellingen aanpassen na het opstarten

10. Container & Cluster Configuratie

HOOFDSTUK PAS MEERDERE SERVERS

11. Conclusie

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer blandit

lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetur libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem.

Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

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-init 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 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
- 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>
- Hardware.info. (2013). *Specificaties Asus X750L*. Verkregen van <https://be.hardware.info/product/204862/asus-x750l/specifications>
- Harney, S. (2016a). Using Ansible to Bootstrap My Work Environment Part 3. Verkregen van <https://www.scottharney.com/using-ansible-to-bootstrap-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-bootstrap-my-work-environment-part-4/>
- Hetzner. (g.d.-a). *Site Hetzner*. Verkregen van <https://www.hetzner.com/>
- Hetzner. (g.d.-b). *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
- Moser, S. & Harlow, J. (2019). cloud-init.io. Verkregen van <https://cloud-init.io/>

- 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/>
- RedHat. (2017). *ANSIBLE IN DEPTH*. Red Hat Inc. Verkregen van <https://www.ansible.com/hubfs/pdfs/Ansible-InDepth-WhitePaper.pdf>
- 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/>
- Young, J. (2019). Ansible with Vagrant on Windows. Verkregen van <https://blog.zencoffee.org/2016/08/ansible-vagrant-windows/>