Ansible Workshop

Inhoudsopgave

Inhoudsopgave **1**

Introductie **3**

- Wat is Ansible 3
- Wat heb je nodig 3
- Indeling workshop 3
- Content USB stick 4
- Intro Vagrant 4
  - Up 4
  - Destroy 4
  - SSH 4
- Intro YAML 4
- Klaarzetten van de workshop 5

Step 0: Inventory **7**

- Doel 7
- Aanmaken hosts 7
  - Opstellen inventory 7
- Resultaat 9

Step 1: User management **10**

- Doel 10
- Aanmaken users 10
  - Module 10
  - Playbook 10
  - Aanmaken gebruikers 11
  - Aanpassing inventory 12
  - Authorized keys 12
- Resultaat 13

Step 2: Security **14**

- Doel 14
- SSHD 14
  - Handler 14
- Firewall 16
  - Installatie: 16

Resultaat 17

Step 3: Database 18

Doel 18

PostgreSQL 18

Installatie: 18

Database en user aanmaken 19

Configuratie changes 19

Resultaat 20

Step 4: Webserver 21

Doel 21

Webserver installatie en configuratie 21

JRE installeren 21

Installatie applicatie 21

Testen applicatie 22

Applicatie als service instellen 22

Resultaat 23

Step 5: Into the clouds! 23

Doel 23

Orchestration 23

Step 6: Untangling the mess 25

Doel 25

Opsplitsen playbook 25

Introductie

Wat is Ansible

Ansible is een open source automation engine waarmee software provisioning, configuration management en application deployment kunnen worden geautomatiseerd.
De automation language wordt geschreven in een leesbaar formaat en kan meerdere keren worden uitgevoerd op de beschikbare hosts, en mits goed geconfigureerd, altijd hetzelfde eindresultaat.

Wat heb je nodig

Op de laptop die gebruikt gaat worden voor de workshop moeten de volgende twee programma's zijn geïnstalleerd:

- VirtualBox (5.1)
- Vagrant 1.9.0 (moet config 2.0 ondersteunen)

Indeling workshop

Het doel van de workshop is om de deelnemers wegwijs te maken met Ansible. Dit wordt bewerkstelligd door een (ietwat versimpeld) real-life voorbeeld waar de gebruiker Ansible moet instrueren om twee machines klaar te zetten met respectievelijk een Java applicatie en een database. Beide machines worden voorzien van basis beveiliging.
Aan het einde van de workshop is het de bedoeling dat de Java applicatie bereikbaar is vanuit de browser en een connectie met de database kan leggen. Op dat punt moeten instructies aan Ansible zodanig opgesteld zijn dat deze als 'professioneel' beschouwd kunnen worden.

We gaan een aantal facetten van Ansible behandelen:

- Inventory (hosts, groepen, instellingen, connectie typen)
- Taken
- Playbooks

Content USB stick

- Vagrant box F8Forum/university_base
- Vagrant box Debian/Jessie64
- F8Forum.jar

Intro Vagrant

Vagrant is een tool om virtual machines mee te ‘managen’. Het automatiseert het gebruik van bijv. Virtualbox door het configureren en provisionen van machines eenvoudiger te maken.

Hieronder volgt een korte lijst van commando's en hun werking:

Up

Met het commando *vagrant up* worden de virtuele machines opgestart, als ze dat nog niet zijn.

Destroy

Met het commando *vagrant destroy* worden de virtuele machines vernietigd. Vagrant vraagt per VM of deze vernietigd moet worden.

SSH

Met het commando *vagrant ssh <naam>* kan er worden ingelogd op een machine. In deze workshop worden de volgende machine

Meer informatie over vagrant:
https://www.vagrantup.com/intro/index.html

Intro YAML

YAML is een leesbare data serialisatie standaard voor alle programmeertalen.
De syntax van YAML bestaat uit simpele key->value pairs, dictionaries en lists. **De indenting gebeurt door middel van spaties, geen tabs!** De indenting (de hoeveelheid spaties, minimaal 2) wordt bepaald door de eerst gevonden indenting in een bestand.

Hieronder een voorbeeld van de ansible site, waarin fruits een lijst is van values:

```
---
# A list of tasty fruits
fruits:
```

```
  - Apple
  - Orange
  - Strawberry
  - Mango
```

Een voorbeeld van een dictionary:

```
# An employee record
martin:
  name: Martin D'vloper
  job: Developer
  skill: Elite
```

En een complexere structuur waarbij beide worden gebruikt:

```
# Employee records
- martin:
    name: Martin D'vloper
    job: Developer
    skills:
      - python
      - perl
      - pascal
- tabitha:
    name: Tabitha Bitumen
    job: Developer
    skills:
      - lisp
      - fortran
      - erlang
```

Meer informatie over YAML:
http://docs.ansible.com/ansible/YAMLSyntax.html
http://www.yamllint.com/

Klaarzetten van de workshop

Voor de start van de workshop dient eerst de volgende github repository uit gecheckt te worden: https://github.com/First8/ansible-workshop/. De directory waarin dit wordt gedaan is de root directory van deze workshop. Daarnaast is er een usb stick verstrekt waarop een aantal bestanden staan. Deze bestanden zijn ook onderdeel van de workshop.
Op de usb stick staan twee virtualboxes, een specifiek gemaakt voor deze workshop met daarop Virtualbox guest additions geïnstalleerd. Dit is de geadviseerde box om te gebruiken. Je kunt deze box toevoegen door het volgende commando op de commandline uit te voeren: **vagrant box add first8/f8forum-base first8_university_base.box**
Daarnaast staat er een jar bestand op de usb stick. Kopieer deze naar een directory genaamd “web” in de workshop root.

De workshop root dient er als volgt uit te zien voor de start van de workshop:

- ansible.cfg: *de specifieke Ansible configuratie voor tijdens de workshop*
- ansible_key/
    - id_rsa: *private key, gaat gebruikt worden voor authenticatie*
    - id_rsa.pub: *public key, gaat gebruikt worden voor authenticatie*
- bootstrap_database.yml: *de initiële acties voor de database VM*
- bootstrap_web.yml: *de initiële acties voor de webserver VM*
- bootstrap.yml: *de initiële acties voor de controller VM*
- f8forum.yml: *playbook voor de volledige f8forum applicatie*
- inventory.ini: *bevat de IP addresssen van de VMs*
- Vagrantfile: *de instellingen om de VMs aan te maken*
- web/: bevat de webapplicatie en *bestanden om de webapplicatie te starten*
    - f8forum.jar
    - f8forum_startscript
    - start.sh

Voor deze workshop worden voornamelijk commandline acties uitgevoerd. Om te starten ga je naar de workshop root (waar de git repo is uitgecheckt). Hier voer je het commando: **vagrant up** uit. Dit zorgt ervoor dat alle virtuele machines worden aangemaakt.
Als alles succesvol is aangemaakt door vagrant zouden er drie machines moeten draaien:

1. controller (192.168.33.10)
2. web (192.168.33.20)
3. database (192.168.33.30)

De 'controller' machine is uitgerust met Ansible 2.2.2.0, en bevat een gebruiker 'ansible' (wachtwoord: 'ansible'). Deze machine wordt gebruikt om de anderen te provisionen.

Het aanpassen van de bestanden kan gedaan worden vanaf het host systeem, de ansible commando's dienen te worden uitgevoerd vanaf de controller machine.
Log daarom vanaf de terminal in op de controller machine met **vagrant ssh controller**.
De commando's die uitgevoerd moeten worden op het host systeem zullen in de workshop in het **blauw** gemarkeerd zijn, de commando's die uitgevoerd moeten worden op de controller in **rood**.
Op de controller machine kun je alle bestanden uit de workshop root vinden in de directory **/vagrant.** Deze map is gedeeld met het host systeem. Alle wijzigingen worden hierin dus direct gereflecteerd, vandaar dat voor het bewerken gewoon de favoriete texteditor van de gebruiker gebruikt kan worden op het host systeem.

Aan het einde van de workshop zijn de machines 'web' en 'database' voorzien van respectievelijk een Java webapplicatie en een PostgreSQL database.

Step 0: Inventory

Zorg ervoor dat je de voorgaande stappen hebt uitgevoerd en de drie vagrant machines draaien.
SSH naar de controller machine toe d.m.v **vagrant ssh controller** en ga naar de gedeelde directory **/vagrant**

Doel

Het doel van deze opdracht is het toevoegen van de hosts aan de ansible inventory. Vervolgens kan getest worden of de hosts bereikt kunnen worden zodat er in een later stadium ansible modules op kunnen worden uitgevoerd.

Aanmaken hosts

Opstellen inventory

De 'inventory' van Ansible is een centraal component voor het werken met Ansible. Een inventory bevat alle hosts die beheerd kunnen worden, eventueel voorzien van extra parameters.

Binnen Ansible bestaan er twee typen inventories, een *statische* en een *dynamische* inventory. Een *statische* inventory wordt gebruik om op voorhand de bekende hosts te definiëren. De inventory wordt genoteerd in een INI formaat. Deze variant is handig op het moment dat de servers in eigen beheer zijn. Een *dynamische* inventory wordt gebruik om middels een script (bijv. Bash of Python) de hosts te verzamelen. Deze worden in-memory gedurende de Ansible run opgeslagen. Deze variant van een inventory wordt voornamelijk gebruikt bij cloud providers zoals Amazon Web Services of Digital Ocean.

In deze workshop wordt met name gefocust op de statische variant.
Voor de workshop is er al een inventory aangemaakt in **/vagrant.** Deze bevat op dit punt slechts één entry, die gebruikt is voor het opzetten van de virtuele machine. Om te zorgen dat de web en database server ook te bereiken zijn moeten deze worden opgenomen in de inventory.

Naast het definiëren van hosts in de inventory is het ook een goed idee om *te testen* of de entries in de inventory daadwerkelijk bereikbaar zijn. Hiervoor kan handig gebruik worden gemaakt van de ping module. Dit soort acties zijn in regel slechts éénmalig en hoeven daarom niet te worden ingeregeld in de versiecontrole. Hiervoor biedt Ansible het gebruik van *ad-hoc* commando's. Hieronder volgt een voorbeeld aanroep hoe dit op de command-line kan worden aangeroepen:

ansible -i inventory.ini all -m ping

ansible: aanroepen van Ansible ad-hoc commando's
-i geeft aan welk inventory gebruikt moet worden
all: de host of hosts waar het commando voor aangeroepen moet worden
-m ping: de Ansible module die aangeroepen moet worden

I.p.v. 'all' *kan ook gebruik gemaakt worden van bijvoorbeeld* 'web' *om slechts een subset van de beschikbare servers in de inventory aan te spreken.*

1. Open het bestand **inventory.ini**
2. Voeg onderstaande regels toe onder de entry 'controller' in de inventory:
   web ansible_host=192.168.33.20
   database ansible_host=192.168.33.30

3. Probeer (!) de machines te bereiken met het onderstaande commando:
   ansible -i inventory.ini all -m ping

Dit gaat fout; De output van het ad-hoc commando zou aan moeten tonen dat er geprobeerd is drie hosts te bereiken, maar slechts één host daadwerkelijk bereikbaar is (controller). Dit komt omdat Ansible niet kan inloggen op web en database, wat noodzakelijk is voor de ping module om te slagen.

De entries in een inventory zijn in het patroon:
<hostnaam> <opties key=value>

Verderop in de opdrachten zullen we de volgende opties gebruiken:
ansible_host - Hiermee wordt de host expliciet opgegeven. In dit geval een ip adres maar dit mag ook bijvoorbeeld een hostname zijn.
ansible_user - Geeft de gebruiker aan waarmee er moet worden ingelogd door ansible
ansible_ssh_pass - Wachtwoord van de gebruiker waarmee wordt ingelogd
ansible_become_pass - Het wachtwoord gebruikt voor privilege escalation

Privilege escalation wordt gebruikt om bijvoorbeeld een actie als root uit te voeren terwijl ansible is ingelogd met gebruiker ansible. Dit is noodzakelijk voor bijvoorbeeld het installeren van packages.

Standaard probeert Ansible dit te doen met de huidige gebruikersnaam. In dit geval moeten we echter ook een wachtwoord gebruiken. Hiervoor moeten we een aanpassing maken in de inventory:

4. Pas de entries in de inventory aan:
   web ansible_host=192.168.33.20 ansible_user=vagrant ansible_ssh_pass=vagrant
   database ansible_host=192.168.33.30 ansible_user=vagrant ansible_ssh_pass=vagrant

5. Probeer de machines te bereiken met het onderstaande commando:
   ansible -i inventory.ini all -m ping

Alle hosts zouden nu success moeten retourneren.

Resultaat

Nu de hosts zijn toegevoegd aan de inventory van ansible en deze bereikbaar zijn, kunnen we de hosts gaan provisionen.

Step 1: User management

Problemen met de vorige opdracht?
Check de volgende git branch uit om de vorige opdracht over te slaan:
git checkout step1

Doel

Nu we machines in de inventory hebben staan gaan we een user aanmaken op deze machines om mee in te loggen. Vervolgens gaat ansible deze user gebruiken om zijn acties mee uit te voeren.

Aanmaken users

Over het algemeen wordt het als good practice ervaren om een speciale gebruiker voor Ansible aan te maken. Enkele voordelen hiervan zijn:

- De Ansible gebruiker hoeft niet expliciet per host opgenomen te worden in de inventory
- Er kan in logs beter onderscheid worden gemaakt tussen handmatige acties en geautomatiseerde acties

Voor de workshop is er tijdens het aanmaken van de 'controller' VM een gebruiker 'ansible' met wachtwoord 'ansible' aangemaakt. Een dergelijke gebruiker moet nu ook op de andere machines worden aangemaakt.

Module

- user
- authorized_key

Playbook

Playbooks zijn een verzameling van acties die door ansible uitgevoerd moeten worden. Hiermee kunnen we een reproduceerbare lijst van acties vastleggen en uitvoeren.

Voor de onderstaande acties gaan we gebruikmaken van de volgende twee playbooks in de workshop root:

- *bootstrap_database.yml*
- *bootstrap_web.yml*

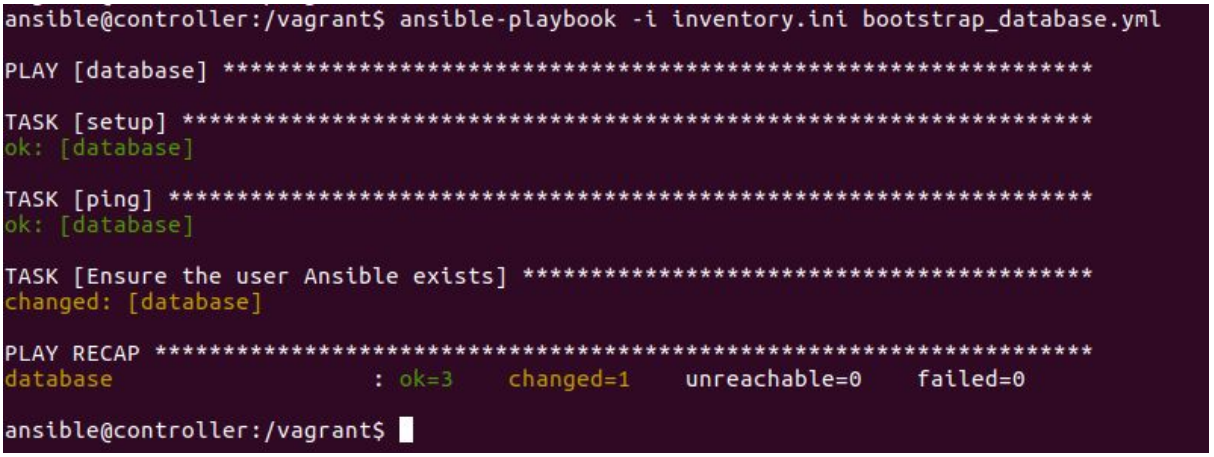
Aanmaken gebruikers

> **Idempotentie** is de eigenschap van een object (of systeem) en/of een operatie daarop dat het object niet meer verandert als de operatie nogmaals wordt uitgevoerd.
>
> Bron: Wikipedia

In tegenstelling tot de 'ping' tests bij het aanmaken van de inventory is het een goed idee om dit soort acties idempotent, dus, herhaalbaar te maken. In plaats van het gebruik van de *ad-hoc* functionaliteit van Ansible is het hier waarschijnlijk verstandiger om gebruik te maken van een *playbook*. Voor de drie machines zijn al playbooks aangemaakt.

1. In het bestand **bootstrap_web.yml** verwijder de 'ping' taak, en haal het commentaar voor de taak 'user' weg
2. Doe hetzelfde in **bootstrap_database.yml**
3. Wanneer je nog niet bent ingelogd op de controller host, doe dit dan met het commando: **vagrant ssh controller,** en navigeer naar de gedeelde map d.m.v. **cd /vagrant**
4. Login als de 'ansible' gebruiker: **su ansible**
5. Voer de playbooks nu uit door middel van de onderstaande commando's:
   ansible-playbook -i inventory.ini bootstrap_database.yml
   ansible-playbook -i inventory.ini bootstrap_web.yml

   De output zou ongeveer gelijk moeten zijn aan de onderstaande afbeelding:



Elke taak die Ansible uitvoert, wordt beschreven. Als dit voor een host "**changed**" is, dan betekent dat dat Ansible een wijziging heeft moeten maken. Als deze "**ok**" is, was de betreffende host al in de gewenste toestand.

6. Als een playbook nog een keer wordt uitgevoerd, zal de uitkomst er nu zo uitzien:

```
ansible@controller:/vagrant$ ansible-playbook -i inventory.ini bootstrap_web.yml

PLAY [web] *********************************************************************

TASK [setup] *******************************************************************
ok: [web]

TASK [Ensure the user Ansible exists] ******************************************
ok: [web]

PLAY RECAP *********************************************************************
web                        : ok=2    changed=0    unreachable=0    failed=0

ansible@controller:/vagrant$
```

Zoals hierboven te zien is, heeft Ansible de tweede keer geen wijzigingen hoeven te doen om het systeem in de gewenste toestand te krijgen.

Aanpassing inventory

Op dit punt bevatten alle machines die in de inventory gedefinieerd zijn een gebruiker ‘ansible'' met als wachtwoord 'ansible'. Als er echter een playbook wordt uitgevoerd zal nog steeds gebruik worden gemaakt van de gebruiker 'vagrant'. Omdat vanaf dit punt in de workshop de bedoeling is dat op alle machines wordt ingelogd met de gebruiker 'ansible' is het *ansible_user* attribuut overbodig.

5. Open de 'inventory.ini' en verwijder de *ansible_user* attributen.
6. Pas de wachtwoorden 'vagrant' aan naar 'ansible'.
7. Login (op de controller!) als de gebruiker ansible, indien dit nog niet het geval is: **su ansible**
8. Probeer de machines te bereiken met het onderstaande commando: **ansible -i inventory.ini all -m ping**

Authorized keys

Het gebruik van wachtwoorden om op andere machines in te loggen wordt in de praktijk over het algemeen afgeraden. Specifiek voor Ansible zijn er een tweetal problemen:

- Naast Ansible moet ook het programma 'sshpass' geïnstalleerd zijn. (Voor de workshop wordt dit gedaan bij het opstarten van de VM, zie *bootstrap.yml*)
- Het wachtwoord wordt nu plaintext opgeslagen in de inventory. Hier is omheen te werken, door deze encrypted op te slaan.

Om dit probleem tegemoet te komen is het beter om gebruik te maken van *authorized keys*. Met deze methode van inloggen wordt gebruik gemaakt van een key-pair, waarbij de server over de public key beschikt. Dit kan eenvoudig met Ansible worden bewerkstelligd door de authorized_key module te gebruiken.

7. In het bestand **bootstrap_web.yml** haal het commentaar voor de taak 'authorized_key' weg en vul de ontbrekende attributen in (zie de Ansible documentatie, gebruik hiervoor de bovenstaande link)
8. Doe hetzelfde in **bootstrap_database.yml**
9. Voer de playbooks opnieuw uit.

Dit gaat fout; voor het aanmaken van de gebruiker zijn verhoogde rechten nodig. Voorheen werd hiervoor gebruik gemaakt van de waarde in 'ansible_ssh_pass'. In plaats daarvan kan gebruik gemaakt worden van het attribuut 'ansible_become_pass'.

10. Open de 'inventory.ini'
11. Voeg voor alle entries het attribuut ansible_become_pass=ansible toe
12. Voer de playbooks opnieuw uit.

Let op:
Voer eerst de playbooks uit, voordat je verder gaat met de volgende stappen!
De keys moeten eerst aan de machines worden toegevoegd voordat de inventory wordt aangepast.

De inventory.ini hoort er nu zo uit te zien:

```
controller ansible_connection=local ansible_become_pass=ansible

web ansible_host=192.168.33.20 ansible_ssh_pass=ansible ansible_become_pass=ansible
database ansible_host=192.168.33.30 ansible_ssh_pass=ansible ansible_become_pass=ansible
```

Op dit punt zijn de wachtwoorden om in te loggen niet meer nodig.

13. Open de 'inventory.ini' en verwijder de 'ansible_ssh_pass' attributen.
14. Probeer de machines te bereiken met het onderstaande commando:
    ansible -i inventory.ini all -m ping

Om de idempotentie van de authorized_key te demonstreren is het op dit punt een goed idee om het playbook opnieuw uit te voeren.

15. Voer de playbooks opnieuw uit.

Resultaat

Op dit punt zijn alle hosts voorzien van een gebruiker 'ansible', die verhoogde rechten heeft. Daarnaast kan er nu vanaf de *controller* worden ingelogd op de hosts zonder het gebruik van een wachtwoord.

Step 2: Security

Problemen met de vorige opdracht?
Check de volgende git branch uit om de vorige opdracht over te slaan:
git checkout step2
Voer vervolgens het volgende commando voor vagrant uit:
vagrant up --provision

Doel

In het vorige hoofdstuk zijn aanpassingen gemaakt aan de configuratie waardoor Ansible d.m.v. een keypair kan inloggen. Om deze reden kan de SSH server worden geconfigureerd om login pogingen door middel van gebruikersnaam/wachtwoord af te wijzen.

SSHD

Vanaf dit punt zullen alle acties worden uitgevoerd als de gebruiker **ansible** op de **controller** machine. Daarnaast gaat er gebruik gemaakt worden van een nieuw playbook genaamd **f8forum.yml**. Hier gaan alle toekomstige taken in geplaatst worden.

In het vorige hoofdstuk zijn aanpassingen gemaakt aan de configuratie waardoor Ansible d.m.v. een keypair kan inloggen. Om deze reden kan de SSH server worden geconfigureerd om loginpogingen door middel van gebruikersnaam/wachtwoord af te wijzen.

Met de module lineinfile kunnen regels in een configuratiebestand worden bewerkt. Om te zorgen dat het niet meer mogelijk is om met wachtwoorden in te loggen moet de regel 'PasswordAuthentication' worden aangepast in 'PasswordAuthentication no' in de SSHD configuratie. Dit bestand bevindt zich in **/etc/ssh/sshd_config**. Om de gemaakte wijziging echter door te voeren moet de service worden herstart. Dit kan geverifieerd worden door vanaf de 'controller' machine in te loggen als gebruiker vagrant op bijvoorbeeld 'web' (192.168.33.20).

De SSHD service kan via de service module worden herstart. Bij voorkeur moet deze echter alleen herstart worden als de *lineinfile* module daadwerkelijk een aanpassing maakt.

Handler

Ansible maakt gebruik van het concept *handler*. Een handler kan aan een taak worden toegevoegd door middel van een 'notify' attribuut wat verwijst naar de naam van de handler. Als de taak die de handler geregistreerd heeft het systeem verandert ('changed') dan wordt de handler uitgevoerd. De handler wordt maximaal één keer per playbook uitgevoerd ongeacht hoe vaak hiernaar verwezen wordt.

Handlers zijn ideaal voor modules die initieel moeten worden afgespeeld na de installatie van een package of bij het doorvoeren van een configuratiewijziging; wordt de software geïnstalleerd of de configuratie aangepast? Herstart dan, en alleen dan, de service.

1. Open het playbook genaamd **f8forum.yml** (maak deze aan indien nodig). De host kan worden geconfigureerd op **all**, gebruik eventueel bootstrap.yml als blauwdruk.
2. Om wachtwoord authenticatie uit te schakelen, moet het configuratiebestand van ssh aangepast worden met behulp van de lineinfile module. Zorg ervoor dat in het bestand **/etc/ssh/sshd_config** de regel die begint met **PasswordAuthentication** vervangen wordt door de regel **PasswordAuthentication no**
3. Gebruik een handler en de service module om na het aanpassen van de configuratie de service te herstarten.

Het playbook zou er ongeveer zo uit moeten zien:

```
    - name: Disallow password authentication
      lineinfile:
        dest: /etc/ssh/sshd_config
        regexp: "^PasswordAuthentication"
        line: "PasswordAuthentication no"
        state: present
      notify: Restart SSH
      become: true

  handlers:
    - name: Restart SSH
      service:
        name: ssh
        state: restarted
      become: true
```

3. Voer het playbook opnieuw uit. Om de idempotentie te testen kan je dit meerdere keren doen. Alleen de eerste keer zouden er taken moeten zijn die **changed** zijn.
4. Om de wijziging te testen kan de inventory tijdelijk worden aangepast zodat er weer gebruik wordt gemaakt van de user **vagrant** met password **vagrant.**

```
controller ansible_connection=local

web ansible_host=192.168.33.20 ansible_ssh_user=vagrant ansible_ssh_pass=vagrant
database ansible_host=192.168.33.30 ansible_ssh_user=vagrant ansible_ssh_pass=vagrant
```

Firewall

Installatie:

Naast het strakker inregelen van de toegang via SSH, is het ook verstandig om een firewall te installeren. Standaard ondersteunt Ansible UFW en firewalld. Vanwege de simpliciteit wordt er tijdens de workshop gebruik gemaakt van UFW.

Allereerst moet deze geïnstalleerd worden, dit kan met apt module. Als UFW geïnstalleerd is, staan de poorten echter nog open en de firewall uit.

Wanneer een taak met root rechten dient te worden uitgevoerd, voeg dan **become: true** toe aan de taak.

1. Gebruik de apt module om ufw te installeren. Geef als name **ufw** op en state **present.** De actie dient uitgevoerd te worden met root rechten. Voeg hiervoor **become: true** toe aan de module aanroep.

Allereerst is het belangrijk dat de SSH poort (22) open blijft staan, anders kan Ansible niet verder met het playbook uitvoeren.

Door de UFW module aan te roepen met de state 'enabled' en de policy op 'deny' wordt de firewall ingeschakeld en de rest van de poorten standaard gesloten. Met 'deny' worden, i.t.t 'reject' de pakketten stilletjes gedropt.

2. Zet met de module ufw de ssh poort open. Ook hiervoor zijn root rechten nodig. Voer een allow rule uit voor de ssh poort. Gebruik de volgende parameters:
    - port: ssh
    - policy: allow
    - rule: limit
    - proto: tcp
3. Vervolgens zetten we alle overige poorten dicht. Door de state van ufw te veranderen kunnen we de service laten starten. Gebruik de volgende parameters voor de ufw module:
    - policy: deny
    - state: enabled

Een probleem waar we nu tegenaan lopen is dat het dichtzetten van alle poorten steeds opnieuw wordt uitgevoerd. Dit kan voorkomen worden door gebruik te maken van de register functionaliteit.
Met register kan een variabele worden geregistreerd bij de ene taak en uitgelezen in een daarop volgende taak.

4. Registreer een variabele tijdens de installatie taak en gebruik deze om alleen poorten open en dicht te zetten wanneer de installatie taak in ‘changed’ state is. Gebruik hiervoor **register: <variabelenaam>** en **when: <variabelenaam>.changed**
5. Voer de playbooks opnieuw uit. Als alles goed is gegaan zou je onderstaande output moeten zien. De configuratie taken worden alleen uitgevoerd wanneer de installatie taak **changed** is

```
ansible@controller:/vagrant$ ansible-playbook -i inventory.ini f8forum.yml

PLAY [all] *********************************************************************

TASK [setup] *******************************************************************
ok: [controller]
ok: [web]
ok: [database]

TASK [Disallow password authentication] ****************************************
ok: [controller]
ok: [web]
ok: [database]

TASK [Verify UFW is present on the system] *************************************
changed: [database]
changed: [controller]
changed: [web]

TASK [Deny access on all ports] ************************************************
changed: [controller]
changed: [database]
changed: [web]

TASK [Allow access on the SSH port] ********************************************
changed: [controller]
changed: [web]
changed: [database]

PLAY RECAP *********************************************************************
controller                 : ok=5    changed=3    unreachable=0    failed=0
database                   : ok=5    changed=3    unreachable=0    failed=0
web                        : ok=5    changed=3    unreachable=0    failed=0
```

Resultaat

In de voorgaande stappen hebben we onze machines geconfigureerd en aanvullende packages geïnstalleerd. Nu de security van de machines is aangescherpt, kunnen we naar de installatie van de applicatie gaan toewerken.

Step 3: Database

Problemen met de vorige opdracht?
Check de volgende git branch uit om de vorige opdracht over te slaan:
git checkout step3

Doel

Voor de data persistentie gaan we gebruik maken van een Postgresql database.
Graag willen we een recente versie van deze database installeren op de database host in onze inventory.
Vervolgens is er een gebruiker nodig om in te loggen en een schema waar de Java applicatie gebruik van kan maken die we in de volgende stap gaan installeren.

PostgreSQL

Installatie:

Om ervoor te zorgen dat de volgende acties alleen worden uitgevoerd op de database host(s), specificeren we een nieuw blok met als eerste een hosts declaratie met als waarde *database* (de naam gebruikt in de inventory file).

Daarna moet er een apt repository toegevoegd worden die de meest recente versies van postgresql beschikbaar stelt.

1. Voeg een nieuw 'host' blok met een ' tasks' list toe. De host moet ingesteld worden op 'database'
2. Voeg de volgende apt key met apt_key module aan de database host toe : *https://www.postgresql.org/media/keys/ACCC4CF8.asc*
3. Voeg de volgende apt repository toe met de apt_repository module: **deb http://apt.postgresql.org/pub/repos/apt/ jessie-pgdg main**

Wanneer dit gedaan is, kunnen we de Postgresql database gaan installeren. Om Postgresql succesvol te installeren en om de postgresql module van Ansible te kunnen gebruiken moeten er twee packages worden geïnstalleerd.

- python-psycopg2
- postgresql-9.5

Denk er ook aan om de cache van apt te updaten voordat de installatie van de packages wordt uitgevoerd. Er is immers een repository toegevoegd.

3. Gebruik de apt module om de packages **python-psycopg2** en **postgresql-9.5** te installeren. Gebruik de state om de packages te installeren en te starten.

Om deze packages te installeren kan meerdere malen de apt module achter elkaar worden aangeroepen, maar er is ook een manier om dit in één aanroep te doen.
Hiervoor kan **with_items** worden gebruikt.
Hierbij wordt er een aanroep gedefinieerd die voor alle items in een lijst wordt gebruikt.

Dit ziet er als volgt uit:

```
- name: Verify installation of PostgreSQL 9.5 and Ansible adapter for Python
  apt:
    name: "{{ item }}"
    state: present
    update_cache: yes
  with_items:
    - python-psycopg2
    - postgresql-9.5
  become: true
```

4. Gebruik de **with_items** aanroep om de benodigde packages te installeren op de database host.

Database en user aanmaken

Ansible heeft een aantal modules voor het beheren van een Postgresql database.
Om de acties van deze modules te gebruiken, moeten deze worden uitgevoerd als gebruiker postgresql. Hiervoor kan become_user worden gebruikt. Gebruik **become_user** in combinatie met become: true om root rechten te verkrijgen.

Gebruikte modules:

- postgresql_db
- postgresql_user

5. Gebruik eerst de module **postgresql_db** om een database aan te maken genaamd **f8forum_db**.
6. Maak daarna de user **f8forum_user** met password **f8forum_user** aan met de module **postgresql_user**. Geef hierbij ook de zojuist gemaakte database naam mee en zet de privileges op **ALL** en geef de volgende rollen mee: **NOSUPERUSER** en **NOCREATEDB**.

Configuratie changes

Om Postgresql beschikbaar te maken voor de applicatie die we hierna gaan installeren moet de default configuratie worden aangepast. Hiervoor kan gebruik gemaakt worden van de module **lineinfile**.

Met deze module kan een regel in een bestand worden toegevoegd of aangepast.

7. Voeg aan het bestand: **/etc/postgresql/9.5/main/pg_hba.conf** de volgende regel toe: **"host \t all \t all \t{{ web_ip }}/0 \tmd5"**
8. Voeg in het bestand: **/etc/postgresql/9.5/main/postgresql.conf** de regel: **listen_addresses = '*'** toe na de regel: **#listen_addresses = 'localhost'** ...
9. Reload de postgresql service met behulp van de service module. Dit kan het beste worden gedaan door middel van het gebruik van een handler. Beide **lineinfile** acties zouden deze handler moeten aanroepen.
10. In de firewall ufw moet de poort 5432 opengezet worden voor tcp connecties.

Resultaat

Er is nu een database geïnstalleerd en geconfigureerd op de database host. Nu kunnen we de webapplicatie connectie laten maken deze database.

Step 4: Webserver

Problemen met de vorige opdracht?
Check de volgende git branch uit om de vorige opdracht over te slaan:
git checkout step4

Doel

Voor onze webapplicatie moet Java 8 worden geïnstalleerd. Daarna moeten de applicatie bestanden naar de web host worden gekopieerd en wordt er een init script geïnstalleerd voor de applicatie.

Webserver installatie en configuratie

JRE installeren

Voor de installatie van de webapplicatie dient Java 8 te worden geïnstalleerd. Om de openjdk-8-jre package beschikbaar te maken op de Debian Jessie machines die wij tot onze beschikking hebben moeten we een backport repository toevoegen.

1. Voeg een nieuw 'host' blok met een ' tasks' list toe. De host moet ingesteld worden op 'web'
2. Voeg via de **apt_repository** module de volgende repository toe: **deb http://ftp.de.debian.org/debian jessie-backports main**
3. Vervolgens kan de package **openjdk-8-jre** worden geïnstalleerd met de apt module. Geef bij de installatie aan dat de default release **jessie-backports** is, anders weet de packagemanager niet welke openjre package hij dient te installeren.

Installatie applicatie

De applicatie die geïnstalleerd dient te worden is een jar genaamd **f8forum.jar** en een **start.sh** script. Deze moeten naar de web host toe worden gekopieerd.

De applicatie is geconfigureerd om te verbinden met een database op de host 'database'. De *hosts* file moet hiervoor een extra entry krijgen.

Daarna kan de applicatie gestart worden met de command module.

4. Plaats beide bestanden op de host in de directory **/opt/f8forum**. Deze directory kan aangemaakt worden met de file module. Vervolgens kunnen de bestanden worden verplaatst met copy. Zet de modus op **755**.
5. Open in de firewall de poort 8080 voor tcp connecties

6. Gebruik de lineinfile module om de entry *192.168.33.30 database* aan *etc/hosts* toe te voegen. De betreffende regel kan onderaan worden toegevoegd.

7. Voer het volgende met de command module uit: **“/opt/f8forum/start.sh &”**.

(Optioneel) Om idempotentie te garanderen moet er een check worden uitgevoerd of het commando **“/opt/f8forum/start.sh &”** al eerder is uitgevoerd. Door via de shell module een commando uit te voeren en deze in een variable te registreren, kan de aanroep van **“/opt/f8forum/start.sh &”** conditioneel gemaakt worden.

Testen applicatie

Als alles goed is gegaan zou de applicatie nu bereikbaar moeten zijn op
http://192.168.33.20:8080/
Wanneer de applicatie bereikbaar is, mag deze weer worden afgesloten.

Is er geen webpagina zichtbaar op het adres?
Als dit het geval is, dan is de applicatie waarschijnlijk gecrashed. Voer de volgende stappen uit om het probleem te achterhalen:

1. Verbindt vanaf het host-systeem met de 'web' VM: **vagrant ssh web**
2. Navigeer naar locatie van de applicatie (**cd /opt/f8forum**)
3. Start de applicatie: **java -jar f8forum.jar**
4. De output wordt nu naar de console gelogd. Hiermee kan de oorzaak van de crash worden onderzocht.

Log in op de machine met **vagrant ssh web**, achterhaal de pid door het commando **ps aux | grep java**. Voer vervolgens het commando **sudo kill <PID>**, waarbij <PID> moet worden vervangen door het process id.

Applicatie als service instellen

Problemen met de vorige opdracht?
Check de volgende git branch uit om de vorige opdracht over te slaan:
git checkout step4_1

Om de applicatie ook een reboot te laten overleven, kunnen we de applicatie ook installeren als service.

5. Kopieer het bestand **f8forum_startscript** naar **/etc/init.d/f8forum** op de web host. **Let op:** de naam van het bestand wordt aangepast naar **f8forum** op de host ipv **f8forum_startscript**.
6. Voer het volgende commando uit: **"update-rc.d f8forum defaults"**. Dit installeert het startup script naar de standaard runlevels van linux.

Nu kan de applicatie ook worden gestart en gestopt met het commando: **service f8forum (start|stop)**.
Wanneer je het playbook nu nogmaals uitvoert, wordt de command task die bovenstaand commando uitvoert steeds als changed aangemerkt. Dit kan voorkomen worden door de **creates** optie te gebruiken. Hiermee kun je aangeven dat wanneer een bepaald bestand bestaat het commando niet hoeft te worden uitgevoerd.

7. Voeg een creates optie toe die checkt of het volgende bestand bestaat: **/etc/rc5.d/*f8forum**.

Dit zou er voor moeten zorgen dat het commando maar een keer wordt uitgevoerd.

8. Het command dat gebruikt wordt om de applicatie te starten in een van de vorige stappen kan nu worden vervangen door de service module.

Resultaat

We hebben nu jre en de java web applicatie geïnstalleerd als een service op de web host. Hiermee kan de applicatie worden gestart en gestopt en deze connect met de database server.

Step 5: Into the clouds!

Problemen met de vorige opdracht?
Check de volgende git branch uit om de vorige opdracht over te slaan:
git checkout step5

Doel

Op dit punt hebben we een werkend playbook die een volledige Java webapplicatie en database kan uitrollen. Dit doen we echter op lokale VMs, dus nuttig is het nog niet. In dit hoofdstuk gaan we de gemaakte playbooks uitvoeren op machines die worden aangevraagd bij een cloud provider.

Orchestration

Het aanvragen van servers (dus niet het inrichten daarvan) heet in Ansible jargon *orchestration*. Voor deze workshop is er gekozen om deze servers af te nemen bij DigitalOcean.

Tot nu toe is er gebruik gemaakt van de user *ansible*. Vanwege instellingen in Vagrant moet dit hoofdstuk als de gebruiker *vagrant* worden uitgevoerd. Dit heeft te maken met missende executierechten voor de gebruiker *ansible* in de */vagrant* map.

Daarnaast moet er in tegenstelling tot de vorige hoofdstukken een *dynamic inventory* worden gebruikt. Dit is een uitvoerbaar Python script, wat in plaats van de *inventory.ini* gebruikt gaat worden.

API Token
Om dit hoofdstuk succesvol te doorlopen is een API key nodig. Deze kan je aanvragen bij een begeleider. Natuurlijk kan je ook zelf registreren bij DigitalOcean en je eigen API token genereren!

1. Verwijder de *inventory.ini* uit de repository. Deze is niet meer nodig.
2. Haal de configuratie en het script op om dynamisch inventories te genereren voor DigitalOcean. Maak het script uitvoerbaar (*chmod +X <filename>)*.
3. Om het playbook en de provisioning te draaien is het DO token nodig als omgevingsvariabele. Voer het volgende uit in de controller VM: **export DO_API_TOKEN=<token>**
4. Maak in de repository een nieuw bestand *digitalocean.yml* aan, met een hostblok en takenlijst voor localhost (controller). *Tip: kijk dit af of kopieer dit uit f8forum.yml*

Om met DigitalOcean te communiceren gebruikt Ansible de DOPy wrapper. Voor deze workshop hebben we versie 0.3.5 nodig. Deze kan, op dezelfde wijze als de apt module, geïnstalleerd worden met de Python package manager pip.

5. Voeg een taak toe in *digitalocean.yml* om DOPy te installeren
6. Log in op de controller: **vagrant ssh controller**
7. Ter test kan nu gekeken worden of alles werkt:
    a. Navigeer naar /vagrant: **cd /vagrant**
    b. Voer het volgende commando uit: **ansible -i digital_ocean.py all -m ping**

    Ansible zou een melding moeten geven dat er geen hosts gevonden zijn.
8. Voeg een taak toe in *digitalocean.yml* om door middel van de user model een SSH keypair voor de *vagrant* gebruiker te genereren.
9. De public key die in stap 8 gegenereerd is moet nu worden geupload worden naar DigitalOcean. Hiervoor kan je de onderstaande taak kopieren (let op de indentatie). Verander *first8* in het *name* attribuut met je naam:

```
- name: Ensure the SSH pubkey of the Vagrant user is available on DO
digital_ocean:
  name: first8_ssh_key
  command: ssh
  ssh_pub_key: "{{ lookup('file', '/home/vagrant/.ssh/id_rsa.pub') }}"
  state: present
register: uploaded_ssh_key
```

10. Nu de public key geregistreerd, kunnen we op DigitalOcean een droplet aanvragen. Gebruik hiervoor de onderstaande aanroep. Vergeet niet het *name* attribuut van je eigen naam te voorzien om de droplet uniek te houden!

```
- name: Verify that a droplet for 'web' exists.
  digital_ocean:
    command: droplet
    name: first8-web-droplet
    unique_name: true
    state: present
    size_id: 1gb
    region_id: ams2
    image_id: debian-8-x64
    ssh_key_ids: "{{ uploaded_ssh_key.ssh_key.id }}"
    wait_timeout: 500
  register: web_droplet
```

11. De droplet is aangemaakt, echter is deze nog niet in de inventory toegevoegd. Dit komt omdat het de dynamic inventory *voor* de orchestration liep. Om deze reden moeten we deze toevoegen in de *in-memory inventory*:

```
- name: Add 'web' to the in-memory inventory
  add_host:
    name: web
    ansible_host: "{{ web_droplet.droplet.ip_address }}"
    ansible_user: root
```

Nu is de droplet door Ansible geïdentificeerd als host *web*, net als de eerdere variant via Vagrant. Deze is bereikbaar via het IP adres van de eerder aangemaakte droplet.

12. Herhaal stap 10 en 11 voor een database droplet.
13. Op dit punt zijn beide droplets aangemaakt en opgenomen in de (in-memory) inventory. Nu kunnen we het originele *f8forum.yml* op te nemen middels het *include* statement, en worden de droplets op exact dezelfde wijze geconfigureerd als de eerder gebruikte virtuele machines!
14. Navigeer naar het IP van de web droplet op poort 8080. het F8forum zou ook hier te zien moeten zijn. *(Tip: deze is te vinden als debug statement wanneer het playbook wordt uitgerold.)*

Step 6: Untangling the mess

Problemen met de vorige opdracht?
Check de volgende git branch uit om de vorige opdracht over te slaan:
git checkout step6

Doel

Het playbook dat tot zover is gemaakt begint inmiddels behoorlijk groot en log te worden. Door de playbooks op te splitsen en includes te gebruiken wordt het playbook minder log en kunnen delen van de infrastructuur apart worden geupdate.

Opsplitsen playbook

Kijkend naar het playbook dat we tot zover hebben gemaakt, kunnen er een aantal onderwerpen worden onderscheiden.
Het gedeelte waarin het usermanagement en security wordt geconfigureerd, het opzetten van de database en het installeren van de applicatie.
Splits deze onderwerpen in aparte files zodat er deze 3 playbooks overblijven:

- security.yml
- database.yml
- webapp.yml

Dit kan gedaan worden door de verschillende blokken in deze files te knippen en plakken.

1. Splits het playbook op in drie aparte playbooks.

Om ervoor te zorgen dat de security steps ook worden uitgevoerd wanneer alleen het database of webapp playbook worden uitgevoerd, kan er in deze beide files een include worden gebruikt.

2. Voeg een include toe aan het database en webapp playbook zodat deze beide gebruik maken van het bootstrap playbook. Hiervoor moet je wel de hostdeclaratie uit het security playbook halen.

Je kunt nu de database en de webapp los provisionen. Om de mogelijkheid te behouden om alle hosts in een keer te provisionen, kan er een all.yml playbook aangemaakt worden die een include doet van zowel database als het webapp playbook.

3. Maak het all playbook die alle hosts provisioned.