Taak 0: Opzetten werkomgeving

Enterprise Linux 15-16

Bachelor toegepaste informatica, HoGent Bedrijf en Organisatie

Inhoudsopgave

1 Leerdoelen

- Git kunnen gebruiken
  - Accounts en repositories beheren met Git en BitBucket
  - Toegang tot een repository met SSH keys beheren
  - Basistaken in Git: wijzigingen in broncode bijhouden (`add`, `commit`), synchroniseren met een online repository (`push`/`pull`)
- Vagrant kunnen gebruiken
  - VMs kunnen beheren met Vagrant: `vagrant up`, `halt`, `reload`, `destroy`, `provision`, `status`, `ssh`
- Ansible kunnen gebruiken
  - Een bestaande Ansible-rol kunnen toepassen op een VM
  - Specifieke instellingen voor een host (`host_vars`) of groep van hosts (`group_vars`) kunnen aanpassen.

2 Aanvullend studiemateriaal

In aanvulling op de werken opgegeven in de studiewijzer volgen hier nog enkele bronnen die specifiek voor dit labo relevant zijn.

- Ansible, Inc. (2015) *Ansible Documentation*. Opgehaald op 2015-09-16 van https://docs.ansible.com/ansible/
- Weissig, J. (2015a) *Crash Course on Vagrant (revised).* Sysadmin casts, episode #42. Opgehaald op 2015-09-16 van https://sysadmincasts.com/episodes/42-crash-course-on-vagrant-revised.
- Weissig, J. (2015b) *19 Minutes With Ansible.* Sysadmin casts, episode #43. Opgehaald op 2015-09-16 van https://sysadmincasts.com/episodes/43-19-minutes-with-ansible-part-1-4.

3 Installatie software

Installeer de nodige software, zoals aangegeven in de studiewijzer.

4 Versiebeheer

1. Maak een account aan op Bitbucket. Gebruik je `student.hogent.be` emailadres bij registratie. Studenten krijgen dan immers een gratis onbeperkt abonnement. Dat houdt in dat je ook gratis private repositories kan gebruiken.
2. Als je dit nog niet gedaan hebt, configureer Git op je eigen pc. **Let op:** doe dit niet op de klaspc's waar ook anderen op werken!

   ```
   $ git config --global user.name "BITBUCKET_USERNAME"
   $ git config --global user.email "EMAILADRES@student.hogent.be"
   $ git config --global push.default simple
   ```

3. Zorg voor toegang tot je repositories met een ssh-sleutelpaar. Het effect is dat je bij een `git push` geen wachtwoord meer moet ingeven.
4. Creëer een fork van de repository https://bitbucket.org/bertvanvreckem/enterprise-linux-labo als *private* repository onder je eigen account. Zorg er voor dat gebruiker `bertvanvreckem` toegang heeft. Voeg de URL toe in dit rekenblad.
5. Maak een lokale kopie van je eigen repository. Het voorbeeld hier gaat er vanuit dat je in Bash werkt (op Windows: Git Bash):

   ```
   $ git clone --config core.autocrlf=input git@bitbucket.org:BITBUCKET_USERNAME/REPO.git
   ```

   waar je uiteraard je eigen gebruikersnaam en repository aanvult.
6. Vul in de `README.md`, `verslag/laboverslag-sjabloon.md` en `doc/cheat-sheet.md` je naam in en de URL naar je Bitbucket repository. Je kan ook de uitleg in deze bestanden verwijderen, die kan je altijd terugvinden in de oorspronkelijke repository. Registreer al deze wijzigingen alvast in Git (`git add`, `git commit`, `git push`) met een passende "commit message".
7. Kopieer het sjabloon voor het laboverslag naar `verslag/laboverslag-00.md`. Vul dit aan met wat je tot nu toe gedaan hebt.

5 Vagrant en Ansible

5.1 Vagrant

Je Git repository bevat een Vagrant-omgeving voor het opzetten van de VMs waaruit het netwerk bestaat. Er is alvast één VM gedefinieerd, die zal in een volgend labo dienst doen als LAMP-server. Je kan een overzicht van de VMs opvragen met:

```
$ vagrant status
Current machine states:

pu004                     not created (virtualbox)

The environment has not yet been created. Run 'vagrant up' to
create the environment. If a machine is not created, only the
default provider will be shown. So if a provider is not listed,
then the machine is not created for that environment.
```

De hostnaam `pu004` slaat op het feit dat deze host in het *pu*blieke netwerk zit. Het getal is een volgnummer dat verder geen betekenis heeft. Hosts in het private netwerk beginnen met `pr` gevolgd door een nummer (bv. `pr002`).

Start de VM met `vagrant up`. De eerste keer dat je dit doet wordt er een basis-VM gedownload met een minimale installatie van CentOS 7.1. Die zal telkens dienst doen als basis voor het opzetten van alle hosts in ons netwerk.

Na opstarten kan je inloggen met `vagrant ssh pu004`. Je bent ingelogd als gebruiker `vagrant` en kan commando's uitvoeren met `root`-rechten door er `sudo` voor te plaatsen (geen wachtwoord vereist). Als het nodig mocht zijn: het wachtwoord van de gebruikers `vagrant` en `root` is telkens `vagrant`.

Als je `ls /` uitvoert, zal je merken dat er een directory `/vagrant` bestaat. Dit is je lokale repository die gemount is binnen de VM. Dit is een eenvoudige manier om bestanden te delen tussen VM en host-systeem.

Let er op dat je VMs niet meer vanuit je VirtualBox GUI opstart of bewerkt. Doe dit nu enkel met Vagrant en vanuit een terminal.

De belangrijkste Vagrant commando's volgen hieronder. Daar waar `[VM]` tussen rechte haken staat, is dat een optioneel argument. Als je het weglaat, wordt de actie op *alle* VMs tegelijk uitgevoerd.

Commando	Functie
`vagrant status`	Geef een overzicht van de Vagrant-omgeving
`vagrant up [VM]`	Start `VM` op
`vagrant provision [VM]`	Draai het configuratiescript op `VM`
`vagrant ssh VM`	Log in op `VM` als gebruiker `vagrant`
`vagrant halt [VM]`	Stop `VM`
`vagrant reload [VM]`	Herstart `VM`
`vagrant destroy [VM]`	Vernietig `VM`

5.2 Ansible

Ansible (2015) is een *configuration management system,* d.w.z. het staat in voor het configureren van een host vanaf een minimale installatie tot een volledig operationeel systeem. Het concept van een configuration management tool is dat je beschrijft wat de gewenste toestand van het systeem is, de tool zorgt er voor dat het systeem in die toestand terecht komt.

Sommige config management systemen hebben een eigen taal ontwikkeld om die configuratie in te beschrijven (bv. Puppet), andere gebruiken een bestaande taal (bv. Chef, configuratie in Ruby). Bij Ansible beschrijf je de configuratie van een systeem met Yaml, een eenvoudig tekstformaat om data te structureren op een manier die makkelijk te interpreteren is zowel door mensen als computers. De bestanden die deze configuratiecode bevatten worden *playbooks* genoemd. Als je deze playbooks op een bepaalde manier structureert en herbruikbaar maakt, spreekt men van een *rol*. Je kan rollen toepassen op een host, en specifieke instellingen toepassen door het invullen van *variabelen*. Op Ansible Galaxy kan je tientallen rollen vinden die door hun auteurs gepubliceerd zijn.

5.2.1 Rollen toekennen aan hosts

Om een rol toe te kennen aan een host, bewerk je `ansible/site.yml`. Dit bestand bevat een overzicht van alle hosts onder het beheer van Ansible, met de rollen van elke host.

```
# site.yml
---
- hosts: pu004
  sudo: true
  roles: []
```

Host `pu004` heeft nog geen rollen toegekend, dat gaan we nu veranderen:

```
# site.yml
---
- hosts: pu004
  sudo: true
  roles:
    - bertvv.el7
```

Onder `roles:` kan je zoveel rollen toevoegen als nodig. Let goed op de indentatie. Je **moet** telkens inspringen met 2 spaties.

5.2.2 Rollen installeren

De naam van deze rol is geschreven in de vorm `AUTEUR.ROLNAAM`. Dit wijst er op dat deze rol op Ansible Galaxy gepubliceerd is. Je kan die dan gaan opzoeken, downloaden en installeren onder `ansible/roles`. Let er op dat de directory die de rol bevat `bertvv.el7` moet zijn. Als je hostsysteem Linux of MacOS is, kan je Ansible installeren en gebruik maken van het commando `ansible-galaxy`:

```
$ ansible-galaxy -p ansible/roles bertvv.el7
```

Dit commando zal de rol `bertvv.el7` downloaden en uitpakken onder directory `ansible/roles`.

Jammer genoeg wordt Ansible onder Windows niet ondersteund, en is het commando dus ook niet beschikbaar. Je kan wel manueel een rol downloaden van de Github-repository (bijvoorbeeld voor `bertvv.el7`: https://github.com/bertvv/ansible-role-el7/releases) en dan op de juiste plaats uitpakken. Om een en ander te vereenvoudigen kan je onder `scripts/` een scriptje vinden dat dit proces automatiseert: `scripts/role-deps.sh`. Als je het uitvoert (onder Windows in Git Bash, vanuit de directory met je lokale repository), zal het in `ansible/site.yml` alle rollen die er vernoemd worden van Ansible Galaxy downloaden en installeren.

Als je dit gedaan hebt, kan je met `vagrant provision` de rol toepassen.

5.2.3 Hosts configureren

Als je de documentatie voor een rol naleest (typisch het README-bestand van de Github repository dat je ziet op de hoofdpagina van de repository), vind je meestal een lijst met *rolvariabelen* die je kan invullen voor het toepassen van concrete instellingen voor hetzij specifieke hosts, hetzij groepen van hosts. Variabelen die voor *alle* hosts gelden, moeten gedefinieerd worden in het bestand `ansible/group_vars/all.yml`. Variabelen voor specifieke hosts in `ansible/host_vars/HOSTNAAM.yml`. Het formaat is opnieuw Yaml, en de structuur van zo'n bestand is zeer eenvoudig. Je geeft een opsomming van variabelen met de waarde die je er aan wil geven. Een abstract voorbeeld:

```
---
var1: val1
var2:
  - val21
```

```
  - val22
var3:
  - key311: val311
    key312: val312
  - key321: val321
    key322: val322
var4: val4
```

Zorg er nu voor dat op *alle* hosts van het netwerk aan volgende requirements voldaan is:

- De EPEL-repository moet geïnstalleerd zijn (EPEL = Extra Packages for Enterprise Linux, een package repository met software die geen onderdeel is van de distributie, maar wel er voor gebouwd).
- De volgende packages moeten geïnstalleerd zijn:
  - bash-completion
  - bind-utils
  - git
  - nano
  - tree
  - vim-enhanced
  - wget
- Maak een gebruiker aan voor jezelf. Neem je voornaam (in kleine letters!) als gebruikersnaam. Deze gebruiker wordt "administrator" van het systeem.
- Genereer op je hostsysteem een ssh-sleutelpaar en zorg er voor dat de publieke sleutel op elke VM geïnstalleerd wordt zodat je kan inloggen als "administrator" zonder een wachtwoord in te voeren.
- Maak gebruik van de optie die een aangepast `/etc/motd`-bestand installeert dat bij inloggen informatie geeft over netwerkinstellingen.

Na wijziging van de bestanden met variabelen, kan je de wijzigingen doorvoeren met `vagrant provision`.

6 Testen

Voordat je een nieuwe host in productie kan brengen, is het belangrijk te valideren dat die volledig voldoet aan de opgelegde requirements. Voor een deel heeft Ansible zelf al tests ingebouwd, wat de nood aan "Unit tests" grotendeels overbodig maakt. Als je in een playbook beschrijft dat "de Apache webserver moet draaien", dan heb je de garantie dat dat ook zal gebeuren. Het is dus niet nodig om achteraf te testen of de service effectief draait. Als Ansible er om de ene of de andere reden niet in slaagt om Apache op te starten, zal de uitvoering van het playbook falen en word je meteen op de hoogte gebracht.

Testen blijft wel zin hebben op een hoger niveau, bv. integratie- of acceptatietests.

6.1 Automatisch testen van hosts

Er zijn verschillende tools beschikbaar om het testen van infrastructuur te automatiseren. Wij gaan gebruik maken van BATS, een eenvoudig testframework gebaseerd op Bash. Normaliter zal je bij elke labo-taak een testsuite krijgen waarmee je kan verifiëren in hoeverre je de requirements hebt gerealiseerd.

Binnen je repository vind je een directory `test/` met daarin een script `runbats.sh`. Dit script automatiseert het uitvoeren van de juiste tests op elke host. Als je ingelogd bent op een host, kan je alle tests voor die host uitvoeren met hetzelfde commando:

```
$ sudo /vagrant/test/runbats.sh
```

Dit script zal zo nodig BATS downloaden en de tests uitvoeren. Tests die op elke host uitgevoerd moeten worden, plaats je in de `test/` directory. Tests die specifiek zijn voor een bepaalde host, kan je plaatsen in een subdirectory met dezelfde naam als de host. Bijvoorbeeld, stel dat dit de structuur is van de test-directory:

```
$ tree test/
test/
├── common.bats
├── pu004
│   └── lamp.bats
└── runbats.sh

1 directory, 3 files
```

Wanneer je ingelogd bent op `pu004`, dan zal de test `common.bats` uitgevoerd worden, en ook `lamp.bats`. Als je ingelogd zou zijn op een host met de naam `pr001`, dan wordt enkel `common.bats` uitgevoerd, want er zijn geen tests specifiek voor die host voorzien.

Het testbestand `common.bats` is al aanwezig. Je moet nog wel een kleine aanpassing doen en dat is in de volgende lijn je eigen gebruikersnaam invullen:

```
admin_user=bert
```

Wanneer je een demo geeft aan de lector voor de evaluatie vaan de taak, dan is het uitvoeren van de acceptatietests een vast onderdeel!

6.2 Testplan en -rapport

Bij elk laboverslag hoort een testplan. Het testplan is een opsomming van alle handelingen die nodig zijn om aan te tonen dat het op te zetten systeem zich gedraagt volgens de specificaties. Deze worden al getest aan de hand van het testscript, is een testplan dan nog nodig? Jazeker, want niet *alle* specificaties kunnen ook effectief worden geverifieerd. Het inloggen via ssh vanop het hostsysteem, bijvoorbeeld kan je niet aantonen vanuit de BATS tests.

Het testplan is eigenlijk het scenario van de demo die je geeft aan de lector om aan te tonen dat je alle aspecten van de opdracht correct hebt uitgevoerd. Dit bestaat uit concrete handelingen of commando's die je moet uitvoeren met het verwachte resultaat.

Om je op weg te helpen, krijg je bij dit labo het tesplan, maar in volgende labo's moet je dit zelfstandig uitwerken. In je eigen laboverslag pas je dit testplan waar nodig aan en geef je de resultaten die jij zelf verkrijgt.

0. Ga op het hostsysteem naar de directory met de lokale kopie van de repository.
1. Voer `vagrant status` uit
    - je zou één VM moeten zien met naam `pu004` en status `not created`. Als deze toch bestaat, doe dan eerst `vagrant destroy -f pu004`.
2. Voer `vagrant up pu004` uit.
    - Het commando moet slagen zonder fouten (exitstatus 0)
3. Log in op de server met `vagrant ssh srv010` en voer de acceptatietests uit:

    ```
    [vagrant@pu004 test]$ sudo /vagrant/test/runbats.sh
    Running test /vagrant/test/common.bats
    ✓ EPEL repository should be available
    ✓ Bash-completion should have been installed
    ✓ bind-utils should have been installed
    ```

```
✓ Git should have been installed
✓ Nano should have been installed
✓ Tree should have been installed
✓ Vim-enhanced should have been installed
✓ Wget should have been installed
✓ Admin user bert should exist
✓ Custom /etc/motd should be installed

10 tests, 0 failures
```

4. Log uit en log vanop het hostsysteem opnieuw in, maar nu met ssh. Er mag geen wachtwoord gevraagd worden.

```
$ ssh bert@192.0.2.50
Welcome to pu004.localdomain.
enp0s3     : 10.0.2.15         fe80::a00:27ff:fe5c:6428/64
enp0s8     : 192.0.2.50        fe80::a00:27ff:fecd:aeed/64
[bert@pu004 ~]$
```

7 Evaluatie

Deliverables:

- Bitbucket repository (met eigen naam ingevuld op de voorziene plaatsen)
- Labo-verslag, tijdig ingediend op Chamilo
- Demo met acceptatietest

Om de score in de rechterkolom te halen, moet je **alle** taken tot en met de overeenkomstige lijn realiseren.

Taak	Score
Software installeren (VirtualBox, Vagrant, Git, editor)	
Bitbucket account aanmaken, repo opzetten	
Labo-verslag is tijdig ingediend op Chamilo en volledig	
Demo gegeven aan de hand van testplan	
VM met Vagrant aanmaken en opstarten (`vagrant up`)	voldoende
Acceptatietests kunnen uitvoeren	goed
Repo- en package-installatie geslaagd *	
Installatie `/etc/motd` geslaagd *	zeer goed
Aanmaken admin-gebruiker geslaagd *	uitstekend
Pushen naar Bitbucket kan met ssh-key	
Inloggen vanop hostsysteem met ssh-key geslaagd *	uitmuntend

(*) Er wordt verondersteld dat deze acceptatietest slaagt onmiddellijk na het opnieuw creëren van de VM. M.a.w. na een `vagrant destroy -f` gevolgd door `vagrant up`.