Enterprise Linux.

Syllabus.

Bert Van Vreckem, Andy Van Maele.

Professionele bachelor toegepaste informatica

Academiejaar: 2019-2020

Faculteit Bedrijf en Organisatie.

HO GENT

Inhoudsopgave

1

Studiewijzer

De inhoud van deze studiewijzer geldt zowel voor de **reguliere** studenten als de studenten **afstandsleren (TILE)**. Waar nodig wordt duidelijk onderscheid gemaakt tussen zaken die enkel voor één van deze twee groepen gelden.

Wat betreft praktische afspraken, regelingen, verwachtingen, enz. in verband met deze cursus zijn dit de enige geldige bronnen van informatie:

- De studiefichefiche van het opleidingsonderdeel:
  - Regulier: https://bamaflexweb.hogent.be/BMFUIDetailxOLOD.aspx?a=110450&b=5&c=1
  - TILE: https://bamaflexweb.hogent.be/BMFUIDetailxOLOD.aspx?a=110449&b=5&c=1
- Deze studiewijzer
- Documenten op Chamilo
- Aankondigingen op Chamilo; deze worden ook telkens per e-mail naar de betrokken studenten gestuurd

Jullie zijn zelf verantwoordelijk voor het opvolgen en lezen van alle aankondigingen. Studenten worden geacht hun opleidingsgerelateerde e-mails regelmatig op te volgen.

Indien er ergens twijfel over bestaat, of er is iets niet duidelijk, neem dan zo snel mogelijk contact op met je lector. De aangewezen manieren worden opgesomd in

Sectie 1.7. Indien nuttig of nodig wordt het antwoord als aankondiging op Chamilo doorgegeven aan alle studenten.

De ervaring leert dat de onderlinge communicatie tussen studenten via Facebook leidt tot verwarring, foute informatie en overbodige discussie. Gebruik dus a.u.b. de officiële kanalen zodat we tot een open en correcte communicatie rond deze cursus kunnen komen.

1.1. Doel en plaats van de cursus in het curriculum

De bedoeling van deze cursus is om je in staat te stellen om op een betrouwbare, reproduceerbare en schaalbare manier netwerkdiensten in productie te brengen op Linux aan de hand van een Configuration Management System.

Wanneer je als systeembeheerder een serverpark van tientallen, honderden of zelfs duizenden machines (hetzij fysiek, hetzij virtueel) moet beheren, dan is het manueel opzetten, of zelfs scripten van de configuratie niet meer voldoende. In dit soort omgevingen wordt steevast gebruik gemaakt van Configuration Management Systems om de taaklast beheersbaar te houden. In deze cursus maken we gebruik van Ansible, omdat dit een voor beginners toegankelijk systeem is dat qua filosofie en logica aansluit bij klassieke shell scripts. In het werkveld is de kans groot dat je met andere Configuration Management Systems in aanraking komt, bijvoorbeeld Puppet of Chef.

Kennis over Linux wordt meer en meer gevraagd op de arbeidsmarkt en wie een goed resultaat haalt voor deze cursus kan overal aan de slag als Linux system engineer.

1.2. Leerdoelen en competenties

De belangrijkste doelstellingen van deze cursus (die je ook vindt in de studiefiche) zijn:

1. Kan het opzetten van netwerkdiensten automatiseren met een configuratiebeheersysteem (configuration management system).
2. Kan configuratieproblemen bij netwerkdiensten opsporen aan de hand van een systematische en grondige methodologie.
3. Kan de meest geschikte documentatie voor een specifieke situatie (meer bepaald de Linux-distributie, softwareversies, enz.) opzoeken en gebruiken.

Om deze doelstellingen te behalen, dien je de hieronder opgesomde competenties te verwerven.

1.2.1. Configuration Management

- Je moet in staat zijn om Vagrant te gebruiken om nieuwe VMs op te starten en die te configureren.
  - De inhoud een `Vagrantfile` kunnen interpreteren en aanpassen
  - Base boxes beheren (commando `vagrant box`)
  - VMs beheren (`vagrant status`, `vagrant up`, `vagrant halt`, `vagrant reload`, `vagrant destroy`, enz.)
  - Inloggen op een Vagrant VM (`vagrant ssh`)
- Je moet in staat zijn bestaande Ansible-playbooks en rollen te interpreteren en aan te passen voor je eigen doeleinden
- Verder moet je die rollen kunnen toekennen aan hosts:
  - Een directorystructuur volgens de Ansible Best Practices[1] aanhouden
  - `site.yml` aanmaken
  - De inventory-file aanmaken en groepen kunnen definiëren
  - Variabelen voor individuele hosts (`host_vars`) en groepen (`group_vars`) initialiseren

1.2.2. Troubleshooting

Je moet de juiste commando's kennen om RHEL/Centos **7** hosts te beheren en te troubleshooten.

- Basiskennis uit Besturingssystemen (gebruikers, groepen, bestandspermissies, directorystructuur, package management, enz.)
- Services beheren met `systemctl`
- Firewalld beheren met `firewalld-cmd`
- Secure Shell gebruiken (`ssh`, `scp`), inloggen ahv een SSH-sleutel
- SELinux begrijpen en toepassen (Ančincová, 2014), meer bepaald:

[1] http://docs.ansible.com/playbooks_best_practices.html

- Status opvragen en wijzigen (`getenforce`, `setenforce`, `/etc/selinux/config`)
- De verschillende modi begrijpen (`enforcing`, `permissive`, `disabled`)
- SELinux context van bestanden opvragen (`ls -Z`) en aanpassen (`chcon`, `restorecon`)
- Logs kunnen bekijken en problemen opsporen (`/var/log/audit/audit.log`)
- Booleans kunnen opvragen en wijzigen (`getsebool`, `setsebool`)

Te kennen uit de RedHat manuals:

- System Administration Guide (Svistunov, Wadeley, Čapek & Hradílek, 2016)
  - Basic system configuration (hst 1-4)
  - Package Management (hst 5)
  - Infrastructure Services (hst 6-7)
  - Opzetten van netwerkservices: web (hst 9), mail (hst 10) , file/print (hst 12; *NIET* 12.3)
  - Monitoring and automation (hst 16, 18, 19)
- Networking Guide (Jahoda, Heves, Wadeley & Huffman, 2016)
  - Inleiding (hst 1, i.h.b. secties 1.3-4, 1.7-8-9)
  - Command line (sectie 2.4)
  - Hostnamen (hst 3)
  - DHCP servers (hst 10)
  - BIND DNS (hst 11)
- SELinux User's and Administrator's Guide (Jahoda, Ančincová & Čapek, 2016)
  - Inleiding (hst 1)
  - SELinux contexts (hst 2)
  - Working with SELinux (hst 4, secties 1-6, 9)
  - Troubleshooting (hst 10)

1.2.3. Documentatie gebruiken

Je moet in staat zijn de juiste documentatie op te zoeken en te gebruiken, die relevant is voor het systeem waar je mee werkt, i.h.b. RHEL/CentOS 7.

- RHEL/CentOS7 handleidingen, i.h.b. de System Administrator's Guide (Svistunov e.a., 2016), Networking Guide (Jahoda, Heves e.a., 2016) en SELinux User's and Administrator's Guide (Jahoda, Ančincová & Čapek, 2016).
- Ansible documentatie (Ansible, Inc., 2016), in het bijzonder
  - Ansible Best Practices: http://docs.ansible.com/playbooks_best_practices.html
  - Ansible Module Index: http://docs.ansible.com/modules_by_category.html
- Vagrant documentatie (Hashicorp, g.d.).

1.3. Leerinhoud

De cursus bestaat uit drie onderdelen:

1. Realiseren van een labo-opdracht die het opzetten van infrastructuur a.h.v. Ansible inhoudt (wordt verderop in deze studiewijzer de *hoofdopdracht* genoemd). Je kan daarbij als student vrij kiezen uit drie concrete opdrachten (zie Hoofdstuk 2).
2. Volgen van de actualiteit in het vakgebied en dit toepassen op je hoofdopdracht (wordt verder de *actualiteitsopdracht* genoemd)
3. Een aantal *troubleshooting*-labo's.

1.4. Leermateriaal

1.4.1. Handboeken, handleidingen, enz.

Zoals eerder vermeld, gebruiken we als leermateriaal in de eerste plaats de handleidingen van de software die we gebruiken in de cursus. Al deze leermaterialen zijn gratis on-line raadpleegbaar.

- RHEL 7 System Administrator's Guide (Svistunov e.a., 2016)
- RHEL 7 Networking Guide (Jahoda, Heves e.a., 2016)
- RHEL 7 SELinux User's and Administrator's Guide (Jahoda, Ančincová & Čapek, 2016)
- Ansible documentatie (Ansible, Inc., 2016)

- Vagrant documentatie (Hashicorp, g.d.)

Verderop in deze syllabus vind je nog aanvullende informatie die je ook verder op weg kan helpen.

Als herhaling van je Linux-kennis uit Besturingssystemen kan je opnieuw het boek van Cobbaut (2015) raadplegen.

Optioneel is het boek "Ansible for Devops" van Geerling (2016) warm aanbevolen. Je koopt dit best als e-boek, want op die manier krijg je ook toegang krijgt tot nieuwe edities die regelmatig verschijnen (b.v. bij de release van een nieuwe versie van Ansible).

1.4.2. Video

Screencasts en presentaties:

- Crash Course on Vagrant (Weissig, 2014)
- 19 Minutes with Ansible (Weissig, 2015)
- Ansible: Python-Powered Radically Simple IT Automation (DeHaan, 2014)
- SELinux for mere mortals (Cameron, 2012)
- Een fileserver opzetten met Samba (Van Vreckem, 2014)
- Linux Troubleshooting (Van Vreckem, 2015a)

1.4.3. Hardware

De aard van deze cursus maakt dat het essentieel is dat je beschikt over voldoende krachtige hardware. Aanbevolen is minstens:

- Processor: Intel i5 7e generatie of equivalent/beter
- Geheugen: 8 GiB RAM
- Harde schijf: 80 GB vrije schijfruimte

Verder komt het volgende nog van pas:

- Een USB-stick of externe USB-schijf met voldoende ruimte (voor het bijhouden van VMs en broncode)

- Een netwerkkabel (netwerklabo GAARB.0.032 en GSCHB.4.037)
- Oortjes voor het beluisteren/bekijken van videolessen, screencasts of video-presentaties zonder de rest te storen

1.4.4. Software

Voor het uitwerken van de labo-taken maak je gebruik van de hieronder opgesomde software, in principe de laatste stabiele versies bij aanvang van het semester. Als je sommige van de opgesomde applicaties al geïnstalleerd hebt, werk die dan bij tot de laatste versie.

- Een **package manager**, zoals je op een Linux-systeem hebt, kan het installeren van de verschillende nodige softwarepakketen enorm vereenvoudigen. Voor Windows is er Chocolatey[2], voor MacOS X is er Homebrew[3]. Instructies vind je verderop.
- Een **goede teksteditor** met syntaxkleuren: Visual Studio Code, Sublime Text, Notepad++, Atom, Vim, Brackets, enz. **Geen** Notepad, Wordpad of Word!
  - Let op! Stel je editor zo in dat tekst inspringt met **spaties** i.p.v. tabs en zet tabbreedte in op 2 karakters.
  - In Linux hoef je niet specifiek iets extra te installeren. De default teksteditor, Gedit, heeft syntax-colouring. Je kan eventueel ook werken met Vim, installeer dan de package `vim-enhanced`. Vim is niet eenvoudig om mee aan de slag te gaan, maar eens je de vele commando's leert kennen, wordt het een bijzonder krachtige en productieve werkomgeving! Visual Studio Code is ook beschikbaar voor Linux.
- VirtualBox [4] (installeer ook het "Extension Pack").
- Vagrant[5]
- Git[6] client (onder Windows, installeer ook Git Bash)
- Bash shell. Onder Windows, kan je de Bash shell die meegeleverd wordt met Git gebruiken. Mac-gebruikers wordt aangeraden om een recente versie van Bash te installeren via HomeBrew.

[2] https://chocolatey.org/
[3] https://brew.sh/
[4] https://www.virtualbox.org/wiki/Downloads
[5] http://vagrantup.com/
[6] https://git-scm.com/downloads

- Ansible[7] (enkel voor wie werkt met een Mac of Linux-laptop. Wie Windows gebruikt kan dit niet installeren. Voor hen is een workaround voorzien)

Installatie op Windows met Chocolatey:

```
C:\> choco install git
C:\> choco install vscode
C:\> choco install virtualbox
C:\> choco install vagrant
```

Installatie op MacOS X met Homebrew:

```
$ brew install bash
$ brew install git
$ brew cask install visual-studio-code
$ brew cask install virtualbox
$ brew cask install virtualbox-extension-pack
$ brew cask install vagrant
$ brew install ansible
```

1.5. Werkvormen

In dit opleidingsonderdeel wordt hoofdzakelijk gewerkt aan de hand van labo-opdrachten. De ondersteunende studiematerialen (zie Sectie 1.4) en deze syllabus moeten je in staat stellen die met succes af te werken.

In de reguliere lessen wordt waar nodig klassikaal uitleg en demo's gegeven over bepaalde onderwerpen.

Studenten **afstandsleren** kunnen vragen stellen tijdens de contactmomenten wanneer de lector aanwezig is, of via e-mail.

1.6. Werk- en leeraanwijzingen

Het werken met labo-opdrachten vergt een zekere mate van zelfstandigheid van jou als student, maar dat is precies ook een attitude die verwacht wordt van een systeembeheerder. Je neemt dus zelf initiatief om de nodige kennis te vergaren

[7] https://docs.ansible.com/ansible/intro_installation.html

en zoekt naar oplossingen voor de problemen die je ongetwijfeld zal tegenkomen. Help elkaar daarin: samenwerken en kennis delen wordt van harte aangemoedigd. De lector is uiteraard ook beschikbaar om je bij te staan als je toch vast komt te zitten en kan je tips geven of verwijzen naar geschikte aanvullende studiematerialen.

Reguliere studenten moeten minstens eens in de drie weken persoonlijk bij de lector langs komen om deelresultaten te tonen. Als je ergens vast zit, en je raakt niet verder, kom dan ook zeker langs zodat de lector je terug op weg kan helpen!

1.7. Studiebegeleiding en planning

Reguliere studenten stellen vragen over de cursus bij voorkeur tijdens de contactmomenten.

Studenten afstandsleren kunnen eveneens vragen stellen tijdens de voorziene contactmomenten waarop de lector aanwezig is, of via e-mail. Gelieve het onderwerp van je bericht te laten voorafgaan door de vermelding “[ELNX]”.

Alle studenten kunnen ook vragen stellen op het Chamilo-forum van de cursus. Als je het antwoord kent op gestelde vragen mag je ook zelf antwoorden. Alvast bedankt in dat geval!

Tabel 1.1 geeft een overzicht van de weekplanning voor de reguliere studenten. Studenten afstandsleren kunnen dit ook gebruiken als een leidraad bij het bewaken van de voortgang van hun studie.

Voor **reguliere studenten** is aanwezigheid op de troubleshooting-labo’s verplicht, aangezien dit evaluatiemomenten zijn. Bij ziekte volg je de normale procedure voor het wettigen van je afwezigheid. Contacteer ook zo snel mogelijk je lector voor een inhaalopdracht.

Bij het begin van een troubleshooting-opdracht krijg je een opstelling voorgeschoteld (typisch een virtuele machine) met configuratiefouten. Het is jouw taak die zo snel mogelijk systematisch op te sporen en op te lossen **volgens de methode die je aangeleerd krijgt**. Over dit proces en je resultaten schrijf je een laboverslag dat je indient op Chamilo.

Studenten afstandsleren kunnen met de lector tijdens die week een moment afspreken waarop je tijd hebt om hier aan te werken. Op het afgesproken tijdstip krijg je een downloadlink naar de opgave en dien je vóór de afgesproken deadline (in principe 4 uur na ontvangst van de opgave) je verslag in op Chamilo.

Lesweek	Onderwerp
1	Inleiding, uitleg opdrachten
	Les + demo basiscommando's EL7, VirtualBox netwerkconfiguratie
2	Finale keuze hoofdopdracht
	Les + demo Ansible
3	Individuele opvolging
4	Presentatie/demo troubleshooting
5	Eerste troubleshooting-opdracht
6	Individuele opvolging
7	Les + Demo BIND, Samba
8	Individuele opvolging
9	Individuele opvolging
10	Individuele opvolging
11	Tweede troubleshooting-opdracht
12	Individuele opvolging
13	Inhaalsessie (Indien ingericht door je lector)

Tabel 1.1: Weekplanning over het semester. Merk op dat er nog verschuivingen kunnen gebeuren naargelang er lessen wegvallen door verlofdagen, enz.

1.8. Evaluatie

De evaluatie van dit opleidingsonderdeel gebeurt voor 100% met niet-periodegebonden evaluatie. Meer bepaald word je beoordeeld op de manier waarop je volgende taken hebt uitgevoerd:

- Hoofdopdracht: opzetten infrastructuur met een configuration management system
- Taak rond actualiteit in het vakgebied
- Troubleshooting van netwerkservices
- Rapportering en documentatie

Troubleshooting wordt geëvalueerd op basis van vaardigheidstests (20% van het totaal), de andere taken op basis van een portfolio dat je samenstelt tijdens de loop van het semester en dat je op een evaluatiemoment tijdens de examenperiode komt verdedigen (80%). Dat portfolio bestaat concreet uit volgende elementen:

- De broncode, door elke student bijgehouden in een toegewezen Git repository
- Gedetailleerde labo-verslagen, eveneens bijgehouden in Git of desgevallend ingediend via Chamilo
- Demonstratie van deelresultaten aan de lector tijdens het semester
- Demonstratie van het eindresultaat aan de lector tijdens de examenperiode

In Hoofdstuk 2 wordt in meer detail uitgelegd wat precies verwacht wordt.

De toekenning van het examencijfer gebeurt op basis van “rubrics” (Andrade, 2000) die beschreven worden in de evaluatiekaart.

In een tabel worden een aantal criteria opgesomd, waar je aan moet voldoen. Je kan “voldoen” op verschillende niveaus, meer bepaald “bekwaam”, “gevorderd”, “deskundig”, of in het slechtste geval “nog niet bekwaam”. Voor elk niveau wordt beschreven wat dat precies inhoudt. De volledige evaluatiekaart is gepubliceerd op Chamilo.

Om te slagen voor dit vak moet je aantonen dat je voor **alle** technische criteria minstens “bekwaam” bent. Met andere woorden, als je voor slechts één criterium “nog niet bekwaam” bevonden wordt, kan je niet slagen. De niet-technische criteria (bv.

rapportering) kunnen het examencijfer afhankelijk van het behaalde niveau nog positief of negatief beïnvloeden.

Merk op dat je zowel een werkend product moet opleveren (= broncode) als de verslagen indienen én demo's geven. Als één van de deliverables ontbreekt, wordt de opdracht beschouwd als niet gerealiseerd.

Reguliere studenten moeten *minstens elke drie weken* deelresulaten opleveren en persoonlijk demonstreren aan de lector. Deze periodes worden beschouwd als tussentijdse deadlines waarvoor art. 5 van het FOER van kracht is. Niet tijdig deelresultaten demonstreren wordt in termen van dat artikel beschouwd als het niet respecteren van tussentijdse deadlines en wordt als dusdanig gesanctioneerd. Een student die meer dan twee maal tussentijdse deadlines niet respecteert, krijgt 0 voor die opdracht ("Nog niet bekwaam") en kan bijgevolg ook niet slagen.

Ook als je weinig tot niets gerealiseerd hebt, kom je langs. Dat is immers een teken dat je ergens vast zit, de lector kan je dan opnieuw op weg helpen.

Studenten afstandsleren kunnen op verschillende manieren deelresultaten opleveren:

- aan de hand van een screencast (bv. publiceren op Youtube, link doorsturen naar de lector),
- via videoconferencing (Google Hangouts of Skype, na afspraak),
- tijdens de contactmomenten voor TILE-studenten, als de lector daar aanwezig is (en enkel na afspraak),

1.8.1. Tweede examenkans

Wie niet slaagt, krijgt een tweede examenkans in de vorm van een individuele opdracht. De precieze opdracht hangt af van je evaluatie. Meer bepaald zal je moet kunnen aantonen dat je voor alle technische criteria minstens "bekwaam" bent. Deze opdracht wordt meteen vastgelegd op het finale evaluatiemoment.

Als je nog geen individuele opdracht gekregen hebt op het evaluatiemoment (bijvoorbeeld omdat je afwezig was), dan kom je naar de feedback. Studenten afstandsleren kunnen dit ook per e-mail regelen.

Wie na afloop van de feedback nog geen afspraak gemaakt heeft voor een individuele opdracht, geeft daarmee te kennen niet te willen deelnemen aan de 2e examenkans.

2

Opdrachten

In dit hoofdstuk worden de verschillende taken en opdrachten toegelicht die je in het kader van deze cursus moet uitvoeren. De bedoeling van deze opdrachten is dat je die gebruikt als aanzet om een dieper inzicht te krijgen in de werking van Linux, Ansible, virtualisatie, enz. Probeer a.u.b. *"voodoo programming"* te vermijden, het intikken van code of uitvoeren van commando's zonder ze te begrijpen, totdat het er op lijkt dat "het werkt". Ook al gebruik je een tool om het configureren van servers te automatiseren, toch blijft het nodig om de onderliggende werking te begrijpen: de structuur en syntax van configuratiebestanden, de belangrijkste commando's voor systeembeheertaken, enz.

2.1. Hoofdopdracht

De hoofdopdracht omvat het opzetten van ict-infrastructuur op basis van Linux. We werken in een gevirtualiseerde omgeving en er is een bijzondere aandacht voor *automatiseren* en *testen*. Je moet dus in staat zijn de systemen "from scratch" op te bouwen met een minimum aan manuele handelingen. Daarna moet je kunnen aantonen dat deze systemen voldoen aan de opgelegde specificaties aan de hand van geautomatiseerde tests en een gedetailleerd testplan en -rapport.

De infrastructuur die je bouwt moet telkens volledig "from scratch" kunnen worden opgezet zonder manuele tussenkomst. Voor het automatiseren van systeembeheertaken maken we altijd gebruik van een configuration management tool, Ansible. Je krijgt hier wat uitleg over, maar het is de bedoeling dat je ook zelfstandig leert werken met Ansible.

Je kiest aan het begin van het semester één van deze drie opdrachten die je de komende maanden verder uitwerkt.

- *Small/Medium Enterprise infrastructure:* het opzetten van een bedrijfsnetwerk voor een KMO.
- *High availability:* het opzetten van een robuust webserverplatform
- *Release Engineering:* het opzetten van een "build pipeline" voor de ontwikkeling van webapplicaties

De eerste opdracht is in detail uitgewerkt met geautomatiseerde functionele tests voor (bija) elke component van het netwerk. Studenten die zich minder zeker voelen in Linux hebben hier een concreet houvast aan. Bovendien leer je op deze manier verschillende typische netwerkservices kennen.

Langs de andere kant zijn de twee andere opdrachten meer representatief voor de manier waarop Linux tegenwoordig in de praktijk wordt toegepast. Studenten die er naar streven (of overwegen) om als Linux system engineer aan de slag te gaan, kunnen met deze opdrachten relevante ervaring op doen.

2.1.1. Small/Medium Enterprise infrastructure

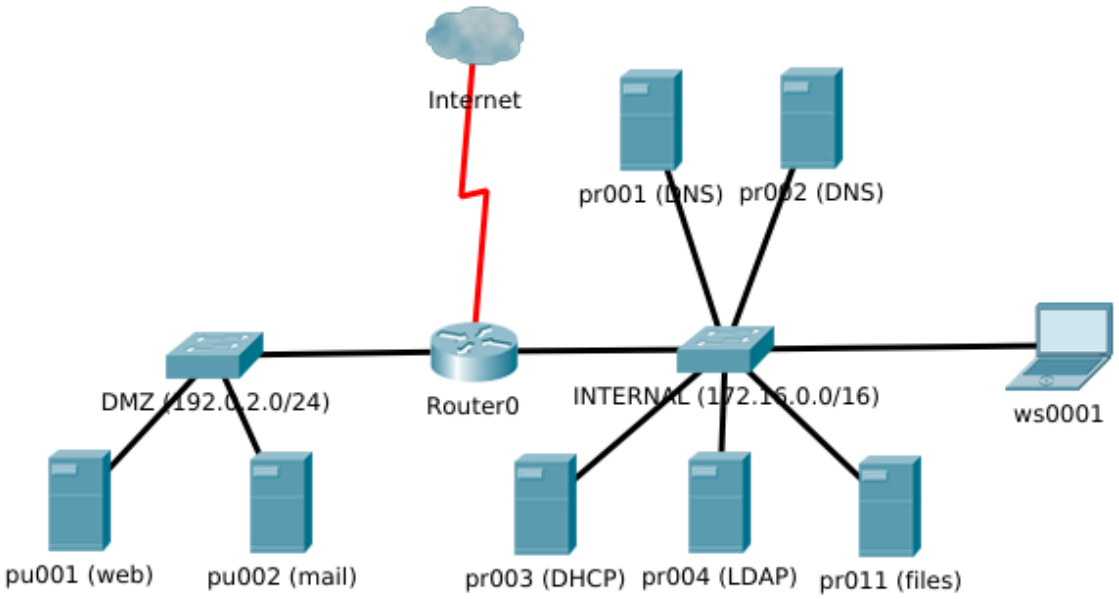
In deze opdracht speel je de rol van de klassieke systeembeheerder of *system engineer* en zet je een volledig (gevirtualiseerd) netwerkdomein op met de enkele typische netwerkdiensten voor een kantooromgeving (zie Figuur 2.1).

Deze opdracht bestaat uit een aantal deeltaken:

- LAMP webserver
- DNS server
- Fileserver
- DHCP server
- Integratie, configuratie router

2.1.2. High availability

In de tweede opdracht verkennen we de taken van een *site reliability engineer (SRE)*. Het is de bedoeling om de infrastructuur op te zetten voor een grote website

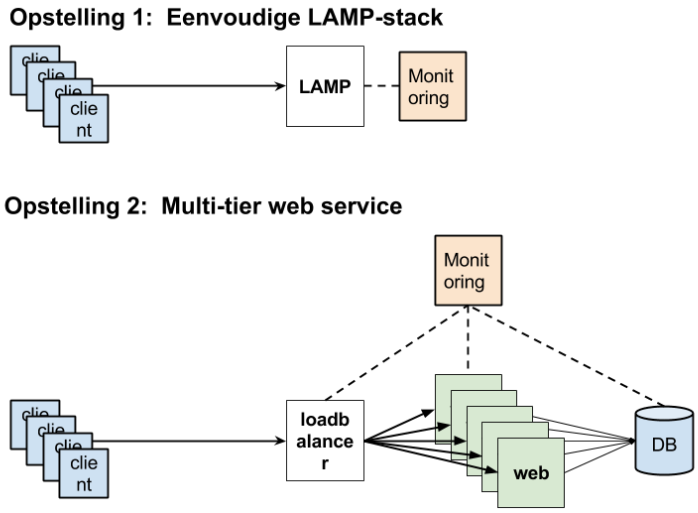


Figuur (2.1)
Diagram van het op te zetten kantoornetwerk in de SME-opdracht. Links van de router bevindt zich het netwerk met publiek toegankelijke services (IP range 192.0.2.0/24), rechts het interne netwerk (172.16.0.0/16) met werkstations en netwerkservices voor intern gebruik. Merk op dat sommige van de vernoemde machines niet tot de opdracht behoren, maar enkel dienen als illustratie.

die veel webtrafiek moet kunnen verwerken. Typisch worden de netwerkservices die samen de klassieke LAMP-stack vormen verdeeld over verschillende machines (zie Figuur 2.2):

- Een *load-balancer* verdeelt netwerktrafiek over *verschillende webservers*;
- Een *cache-systeem* zorgt dat niet alle requests leiden tot het opnieuw genereren van een pagina;
- De *database-backend* komt op een aparte server.

Bij dit soort opstellingen is het ook essentieel dat de beheerder een overzicht heeft van de correcte werking van alle componenten. Om dit te realiseren moet je een monitoring-systeem implementeren met een dashboard dat een overzicht geeft van het platform. Je moet in het bijzonder in staat zijn om de *bottleneck resource* te identificeren, de hardware- of systeembron die het eerst verzadigd wordt door de binnenkomende trafiek. Dat kan de CPU zijn, het geheugen, I/O (netwerk of opslag), enz.



Figuur (2.2)
Netwerkdiagram High Availability-opdracht. In een eerste opstelling draait de webserver in zijn geheel op een enkele VM met een monitoringsysteem voor de opvolging. Het einddoel is een opstelling waar alle services op aparte entiteiten draaien: een loadbalancer die requests verdeelt over een aantal webservers en een aparte database in de backend.

2.1.3. Continuous Integration/Continuous Delivery

Het derde jobprofiel dat je kan verkennen is dat van de *release engineer*. Deze komt vooral voor in omgevingen waar webapplicaties de omzet van het bedrijf genereren, zoals bijvoorbeeld grote websites als Google, Amazon, Facebook, enz. De release engineer ondersteunt een team van software-ontwikkelaars om zo snel en zo vlot mogelijk nieuwe features in productie kunnen brengen. In plaats van grote, ingrijpende releases over langere periodes met veel nieuwe features, streeft men er tegenwoordig naar om kleinere releases te doen aan een hoger tempo (soms zelfs tientallen per dag), met bv. slechts één nieuwe feature of verbetering. Voor deze manier van werken is specifieke infrastructuur nodig en is het nodig een aantal zaken verregaand te automatiseren.

De bedoeling van deze opdracht is om een Continuous Delivery “build pipeline” te bouwen voor een zelf te kiezen casus.

Typisch verloopt dat proces volgens deze stappen:

- De ontwikkelaar werkt op een “development environment” in de vorm van een virtuele machine aangestuurd door Vagrant
- Wanneer de ontwikkelaar code publiceert op de “master” branch van een Git repository wordt het buildproces opgestart, bestaande uit:
  - code-analyse, linting;

    - compilatie (indien van toepassing);
    - uitvoeren van unit tests;
    - packaging (bv. RPM, .deb, tarball);
    - deployment in QA-omgeving (QA: Quality Assurance);
    - uitvoeren van functionele tests;
    - deployment in productie-omgeving.

- De “QA environment” is een VM op een zelf te kiezen virtualisatieplatform (bv. KVM, OpenNebula, Docker)
- Wanneer alle tests slagen, wordt de nieuwe code automatisch gepromoveerd naar de volgende stap, in het beste geval tot en met de “productie-omgeving” op een cloud-platform (bv. Amazon AWS of DigitalOcean).

Je moet de hele pipeline kunnen demonstreren. Na een wijziging in de code te pushen moet je (na verloop van tijd) de wijziging op de productieserver kunnen zien.

De VMs die op de verschillende platformen draaien, worden toch zo identiek mogelijk geconfigureerd. Hier bestaan tools voor, bv. Packer. Je mag ook containers (bv. Docker) gebruiken om je applicatie in te pakken. De build server kan Jenkins zijn, je kan ook kijken naar Travis CI of Gitlab. Voor het laatste kan je gebruik maken van een abonnement op het Gitlab Gold Plan. Indien je hiermee wil werken, neem dan contact op met Bert Van Vreckem.

Enkele concrete suggesties volgen hieronder. Behalve de Drupal-opdracht zijn dit allemaal reële casussen. Neem contact op met Bert Van Vreckem indien je interesse hebt in deze opdrachten.

- ASP.NET-applicatie op Docker[1]. Dit is een reële casus en het resultaat is in het beste geval een opstelling die in productie zal gebruikt worden binnen HOGENT. Recentelijk is er een nieuwe applicatie geschreven voor de administratieve opvolging van stages binnen de opleiding toegepaste informatica. Deze draait echter nog niet in productie. Jouw taak zou dan zijn om er voor te zorgen dat de applicatie kan worden gecompileerd en uitgerold in de productieomgeving. De applicatie is geschreven om in Docker te draaien, dus via deze casus krijg je ook de kans daar ervaring mee op te doen.
- Android application development. Dit is een reële casus en bestaat uit het verbeteren en stroomlijnen van een proof-of-concept van een Jenkins buildserver voor het compileren van Android applicaties die op Google cloud draait.

[1] https://www.docker.com/

- Geautomatiseerd testen van Ansible-rollen met Molecule[2]. Ook dit is een reële casus. De rollen `bertvv.bind`, `bertvv.vsftp`, enz. worden getest via Travis CI[3]. Voor elke ondersteunde Linux-distributie wordt een Docker-container opgestart waarop de Ansible-rol toegepast wordt, waarna er acceptatie-tests op uitgevoerd worden. Het testsysteem is ontwikkeld op een moment dat er nog geen algemeen gebruikte oplossingen bestonden voor het doorgedreven testen van Ansible-rollen. Het is gebaseerd op een shell-script en Docker images die specifiek voor deze toepassing zijn opgezet. Het onderhouden van de testinfrastructuur neemt veel tijd in beslag en regelmatig falen de tests omwille van fouten in het testsysteem. Molecule is een relatief nieuw systeem dat recent als het officiële Ansible-testplatform is verkozen. De bedoeling hier is om een proof-of-concept op te zetten voor het testen van de `bertvv.*`-rollen op basis van Molecule.

- Automatisch genereren van PDF-bestanden uit LaTeX-broncode en publicatie op Github. De bedoeling is hier om vanuit een Github-repo met LaTeX-bestanden bij elke push in Travis-CI een build-omgeving op te zetten, het bestand te compileren (incl. bibliografie, index, enz.), en de resulterende PDF te publiceren op Github, onder Releases. Het systeem moet in staat zijn om bestanden met de HOGENT huisstijlsjablonen te compileren, waarvoor o.a. specifieke fonts nodig zijn (bv. Montserrat). In Travis-CI moet je alle nodige packages downloaden en installeren, wat veel tijd in beslag neemt. Een mogelijke benadering is vooraf een Docker-container te prepareren waar de compilatie in kan uitgevoerd worden[4]. Die kan ook lokaal getest worden, wat tijd bespaart bij het troubleshooten.

- Ontwikkelen van Drupal plugins. De developer krijgt een VM met Drupal die toelaat om code te schrijven op het hostsysteem en die onmiddellijk zichtbaar is (bv. via VirtualBox shared Folders). Bij committen van code in het versiebeheersysteem worden linters/code analyzers uitgevoerd en geautomatiseerde tests. Eventueel kunnen tests op verschillende versies van PHP/MySQL/Drupal in parallel uitgevoerd worden. Als dit lukt, wordt een package gecreëerd dat kan gepubliceerd worden voor gebruikers.

Opmerking. Studenten die deze opdracht combineren met het DevOps-project wordt gevraagd om voor dit vak een andere casus te kiezen. Hetzelfde werk indienen voor twee verschillende vakken is een oneerlijk voordeel dat andere studenten niet hebben. Je kan wel voor beide vakken dezelfde achterliggende technologieën gebruiken zodat je de opgedane kennis wel in beide kan toepassen.

[2] https://github.com/ansible/molecule
[3] Zie bijvoorbeeld https://travis-ci.org/bertvv/ansible-role-bind
[4] Bijvoorbeeld https://github.com/Vincevrp/docker-tex

2.2. Actualiteit

Bij elke rol in de ict-sector is het voortdurend bijwerken van je vakkennis onontbeerlijk om de snelle evolutie in je vakgebied bij te kunnen benen. In Linux systeembeheer is dat niet anders. Wat ons vakgebied kenmerkt is een uitgesproken wil om informatie en kennis te delen. *Sharing is caring!* Er is dan ook een schat van informatie te vinden over de meest recente evoluties via blogs, conferenties waarvan de lezingen op Youtube of Vimeo gepubliceerd zijn, enz.

De bedoeling van deze taak is aan te tonen dat je deze evoluties ook opvolgt en probeert toe te passen in de praktijk. Je kan kiezen uit twee manieren om dit aan te tonen:

- Pas een recentelijk gepubliceerde techniek, tool, ... toe op de opstelling van je hoofdopdracht;
- Doe een bijdrage aan een open source project gerelateerd aan de cursus (incl. tools en projecten die door de lector onderhouden worden[5]).

Samenwerken met één of enkele medestudenten is toegelaten. In dat geval moet elke student individueel kunnen aantonen een significante bijdrage geleverd te hebben via duidelijke rapportering, Git commits, ...

Begin zo snel mogelijk hier aan te werken! Wacht niet tot het einde van het semester om dan snel iets in elkaar te flansen. Dit kan leiden tot een beoordeling "nog niet bekwaam," met als gevolg dat je niet kan slagen.

2.2.1. Nieuwe technieken uitproberen

Zoals eerder aangegeven, vormen Linux-systeembeheerders een "community" waar er veel informatie uitgewisseld wordt. Sommigen gieten zaken die ze bijleren in een blog-artikel, gaan erover spreken op conferenties, enz.

De bedoeling hier is om zo'n artikel of lezing toe te passen op de labo-opdracht. Een paar voorbeelden als inspiratie:

- Hayden (2015) en Davila (2015) beschrijven een manier om RHEL of CentOS-systemen te testen op vlak van beveiliging, gebaseerd op Ansible. Is het mogelijk dat toe te passen op onze systemen? In het artikel gaat het over versie 6, terwijl wij op versie 7 zitten. In hoeverre kan dit aangepast worden?

[5] https://github.com/bertvv/

- Gerelateerd aan het vorige voorstel: voer een security-audit uit met Lynis[6] op de VMs in je opstelling. Tracht de belangrijkste problemen op te lossen en verwerk dit in de Ansible-rollen die je gebruikt hebt. Dien eventueel een Pull Request in bij het Github-project van die rol.
- Fail2ban is een Intrusion Prevention System dat een server kan beschermen tegen brute-force of denial of service-aanvallen (Sawiyati, 2014). Kan je dit toepassen op onze servers? Het is uiteraard wel de bedoeling dit via Ansible te doen. Dat kan hetzij via een bestaande rol (zie Ansible Galaxy), die je zo nodig aanpast, hetzij één die je zelf schrijft.
- Secure Shell is de standaard manier om Linux-servers op een veilige manier van over het netwerk te beheren. Maar volgens stribika (2015) is het mogelijk om `sshd` nog beter te beveiligen. Kan je dit toepassen op onze servers? Kan je eventueel een Ansible-rol voor het beher van `sshd` gebruiken en verbeteren op basis van het artikel?
- Zoals onze opstelling nu is, zullen wachtwoorden in de `host_vars` of `group_vars` bestanden opgeslagen worden. Dit is niet ideaal: we steken onze code in een versiebeheersysteem, maar wachtwoorden horen daar met het oog op beveiliging helemaal niet in thuis. Ansible heeft hiervoor een oplossing: Ansible Vault[7]. Blanc (2015) beschrijft een methode om het gebruik van Ansible Vault zo transparant mogelijk te maken. Kan je het toepassen in onze opstelling?
- Johnson (2015) schreef een artikel over het versnellen van Ansible. Kloppen zijn aanbevelingen? Kan je dat aantonen, m.a.w. het tijdverschil meten tussen de standaardinstellingen en zijn aanpassingen?

Je kan de blogs waar naar gerefereerd werd in deze voorbeelden opvolgen (bv. via een RSS reader), hieronder volgen er nog enkele:

- AT Blog: http://www.atcomputing.nl/blog/
- Cron Weekly newsletter: https://www.cronweekly.com/
- Erika Heidi: http://erikaheidi.com/blog/
- Everything Sysadmin (Tom Limoncelli): http://everythingsysadmin.com/
- Everything is a Freaking DNS Problem (Kris Buytaert): http://www.krisbuytaert.be/blog/
- Fedora Magazine: http://fedoramagazine.org/

[6] https://cisofy.com/lynis/
[7] https://docs.ansible.com/ansible/playbooks_vault.html

- Linux Action News: https://linuxactionnews.com/
- Linux Journal: http://www.linuxjournal.com/
- Major.io (Major Hayden): https://major.io/
- ma.ttias.be (Mattias Geniar): https://ma.ttias.be/
- Planet CentOS: http://planet.centos.org/
- Runaway Sequence (Aaron Hunter): http://sharknet.us/
- Standalone Sysadmin (Matt Simmons): https://www.standalone-sysadmin.com/blog/
- SysadminCasts (Justin Weissig): https://sysadmincasts.com/
- The Geek Stuff: http://www.thegeekstuff.com/

Vond je andere interessante blogs? Geef maar door aan de lectoren! Andere bronnen van informatie zijn Youtube of Vimeo (voor presentaties van conferenties of screencasts), Twitter, enz.

Ben je zelf buiten de opleiding om bezig met Linux? Bespreek met de lector of je je ervaringen, experimenten, ... eventueel kan inbrengen voor deze opdracht.

2.2.2. Bijdrage aan een open source project

Alle tools waar we in de cursus gebruiken zijn open source. Sommige, zoals Ansible, werden door een softwarebedrijf ontwikkeld die daar een businessmodel rond gebouwd hebben. Andere werden door enthousiastelingen in hun vrije tijd ontwikkeld. In elk geval kunnen we gratis gebruik maken van software van hoge kwaliteit, dankzij de inspanningen van velen.

Het is passend daar iets voor terug te doen, dus de bedoeling is om een significante bijdrage te leveren aan een open source project dat gerelateerd is aan de cursus. Dit kan een kleine bijdrage zijn, maar voorwaarde is wel dat ze aanvaard is door de auteur(s) van het project.

Je mag hiervoor samenwerken met één of meerdere medestudenten, maar de individuele bijdrage van elk teamlid moet aantoonbaar zijn (bv. aan de hand van Git commits). Zet hiervoor een aparte, publieke Github repository op, en verwijs er naar vanuit je laboverslag voor deze opdracht.

Enkele mogelijkheden:

- Op Ansible Galaxy zijn veel rollen te vinden die beter kunnen, of waar de auteur niet genoeg tijd heeft om zaken toe te voegen of fouten op te lossen. Implementeer een nieuwe feature, zorg er voor dat ze op CentOS 7 draaien, verbeter fouten, ... Ook de lector appreciëert hulp bij het verder ontwikkelen van zijn Ansible-rollen https://galaxy.ansible.com/bertvv/ en https://github.com/search?q=user%3Abertvv+ansible.

  Het resultaat wordt ingediend als een Pull Request, de code volgt de richtlijnen voor stijl[8], en wordt getest. Opties zijn niet *hard-coded* maar blijven instelbaar via rolvariabelen. Waar mogelijk wordt een standaardwaarde gekozen die in de meeste gevallen het beste resultaat geeft, of wordt aan de hand van de eigenschappen van het systeem een waarde "berekend" of bepaald die voor dat systeem optimaal is. Alle nieuwe variabelen/features worden goed gedocumenteerd.

  Voor concrete ideeën kan je de Issues bekijken op het Github-project en één ervan op je te nemen.

- Schrijf een Ansible rol (waar nog geen alternatief voor CentOS voor bestaat) en publiceer die op Ansible Galaxy. Dit doe je best in samenwerking met één of meerdere medestudent(en)! Wanneer je voor de hoofdopdracht een nieuwe Ansible rol schrijft, kan je die algemeen bruikbaar maken en publiceren.

- Pas een techniek die beschreven is voor een andere distributie (CentOS 6, Debian, Ubuntu, ...) toe op CentOS 7. In de vorige sectie vind je een paar concrete voorbeelden.

Andere ideeën zijn ook welkom, bespreek die met de lector. Ook op Chamilo kan je nog enkele voorstellen vinden.

2.3. Rapportering en documentatie

In de `assignment/` directory van je repository komt alle documentatie terecht in verband met de labo-taken. Het gebruikte formaat is Markdown (Github, Inc., 2016; Gruber, 2004), wat de standaard geworden is voor documentatie op Github (en ook andere Git hosting-oplossingen zoals Gitlab of Bitbucket). Markdown is een eenvoudig tekstformaat dat kan gerenderd worden als HTML, of makkelijk naar andere formaten kan omgezet worden (PDF, LaTeX presentatie met reveal.js, enz.). Leer het formaat goed kennen en controleer of je verslagen duidelijk leesbaar zijn (op Github of reeds lokaal met een editor met Markdown-ondersteuning). Gebruik bv. codeblokken met syntax colouring, links naar relevante code elders in de repository, enz.

[8] https://github.com/bertvv/ansible-role-skeleton/wiki

2.3.1. Laboverslagen

Er is een sjabloon voorzien voor de laboverslagen. Pas dit sjabloon aan voor jezelf, vul bovenaan je naam in. Voor elke deeltaak maak je een apart verslag, waar je in detail uitlegt wat je precies gedaan hebt. Hoe heb je de taak aangepakt? Welke bronnen heb je gebruikt? Reflecteer ook over je vooruitgang: wat ging goed/niet goed? Wat heb je geleerd? Waar zit je vast, waar heb je nog problemen mee? Let wel: het is niet de bedoeling om code te gaan knippen en plakken in je laboverslag!

Bij elk laboverslag hoort een testplan en -rapport. 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.

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, en een beschrijving van het verwachte resultaat.

Door het lezen van het testrapport moet het duidelijk zijn in hoeverre de opdracht is uitgevoerd, wat het effectieve resultaat van de tests was. Je kan transcripties toevoegen van de terminal (gebruik hiervoor Markdown codeblokken, geen screenshots!). Het testrapport zou evenveel informatie moeten bevatten als de demo.

Ook bij de **troubleshooting-oefeningen** hoort een verslag. Deel je verslag op in secties voor de verschillende fasen in het troubleshooting-proces. Je kan je verslag al voorbereiden door de commando's die je zeker moet uitvoeren er alvast in op te sommen, samen met verwachte resultaten (voor zover dit al op voorhand kan). Sla dit op als een apart sjabloon specifiek voor troubleshooting. Bij het uitvoeren van de oefening kan je dit dan aanvullen met de effectieve resultaten die je krijgt, en de commando's/configuratiewijzigingen die je hebt uitgevoerd om problemen op te lossen.

2.3.2. Cheat-sheets en checklists

Wanneer je niet gewend bent om met Linux te werken, dan is het niet evident om op te zoeken en te onthouden welke commando's je nodig hebt voor welke taak. Via Google vind je wel vaak een oplossing, maar dat ligt niet altijd voor de hand. Er zijn bijvoorbeeld recentelijk substantiële wijzigingen doorgevoerd in de architectuur van de belangrijkste Linux-distributies waardoor bepaalde commando's (die je erg vaak tegenkomt bij Googlen) niet meer werken.

Om jezelf te helpen bij het onthouden van de belangrijkste commando's, is het bijhouden van een cheat-sheet of "spiekbriefje" een nuttig hulpmiddel. Als je een

bepaald commando een paar keer bent moeten gaan opzoeken, dan is het best dat eens te noteren zodat je het in de toekomst sneller terugvindt en op de duur ook beter onthoudt.

Hetzelfde geldt voor procedures van handelingen die steeds terugkomen. Bijvoorbeeld, als je wil nagaan of de IP-instellingen van een host kloppen, gebruik je altijd dezelfde commando's. Wanneer je die telkens opnieuw moet gaan opzoeken verspil je tijd en het is best mogelijk dat je zo zaken over het hoofd ziet. Door checklists bij te houden, verminder je het opzoekwerk en kan je ook vlotter werken (Simmons, 2009).

Je kan inspiratie opdoen op deze Github-repository waar een aantal cheat-sheets en checklists gepubliceerd zijn: https://github.com/bertvv/cheat-sheets.

Bij de evaluatie wordt er rekening mee gehouden hoe je deze documenten hebt bijgehouden in de loop van het jaar (aan de hand van de commit log).

2.3.3. Bloggen

Technische blogs zoals deze die eerder aangehaald werden zijn een meer en meer voorkomende manier om ervaringen en informatie uit te wisselen binnen ons vakgebied. Studenten die gaan solliciteren (en ook wie al in het vak staat) kunnen met een eigen blog aan potentiële werkgevers aantonen dat ze gepassioneerd zijn door hun vak, bezig zijn met de laatste technologieën en open staan voor het delen van kennis. Je kan een eigen blog opstarten en schrijven over wat je in deze cursus (en ook andere!) leert, zaken die je zelf opgezocht en ondervonden hebt, ...

Voorbeeldjes:

- Jürgen Van Meerhaeghe: https://jurgenvm.blogspot.be/
- Stijn Spanhove: http://www.spanhove.com/blog.htm
- Thomas Clauwaert: https://ciberth.blogspot.be/
- Toon Lamberigts en Tomas Vercautter: https://t0t0.github.io/

Dit is vrijblijvend, maar heeft wel een positieve invloed op je examencijfer.

3

Aan de slag

In dit hoofdstuk vind je wat technische achtergrond over een aantal onderwerpen die in de labo-opdrachten aan bod kunnen komen. Alles wat hier meegegeven wordt is gebaseerd op jarenlange ervaring, en typische problemen die studenten tegenkomen.

De bedoeling is om een minimum aan informatie mee te geven om je in staat te stellen snel aan de slag te gaan en een aantal typische beginnersfouten te vermijden. Voordat je aan de opdracht begint, lees je dus best eerst de informatie hier na en ga je ook best de **gerefereerde bronnen opzoeken en bestuderen**!

3.1. Opzetten werkomgeving

Installeer eerst de nodige software, meer bepaald de laatste stabiele versie van de applicaties opgesomd in Sectie 1.4.4. De volledige neerslag van al wat je voor deze cursus doet wordt bijgehouden in het versiebeheersysteem Git. Via Chamilo vind je een link die, als je er op doorklikt, een nieuwe repository creëert waar je in kan werken. Deze is zichtbaar voor jou en de lector. Naast de configuratie van de opgezette systemen zal je er ook je documentatie bijhouden, zoals testrapporten, procedures, cheat sheets en checklists.

Richtlijnen voor het opstarten van je Git project:

1. In principe moet je al een Github account hebben. Als je dit nog niet gedaan hebt, koppel dan zeker je @student.hogent.be adres aan het account. Je kan

dan het Github Student Developer Pack[1] aanvragen met allerlei interessante aanbiedingen.

2. Maak een SSH-sleutelpaar aan om het pushen naar Github te vereenvoudigen. Het commando is `ssh-keygen`, volg de richtlijnen die het geeft. Geef voor je gemak een lege *passphrase* op (zoniet moet je telkens je de sleutel gebruikt je passphrase intikken). Normaal zou er in de directory `~/.ssh`[2] twee bestanden aangemaakt moeten zijn: `id_rsa` en `id_rsa.pub`. Het eerste is je private sleutel (die je geheim moet houden), het tweede de publieke. Die laatste kan je op Github registreren via je profielinstellingen (klik op je avatar rechtsboven, volg *Settings* en dan *SSH and GPG keys*).

3. Basisconfiguratie Git, indien je dit nog niet gedaan hebt (kijk na in het configuratiebestand `~/.gitconfig`):

```
$ git config --global user.name "VOORNAAM NAAM"
$ git config --global user.email "VOORNAAM.NAAM@student.hogent.be"
$ git config --global push.default simple
$ git config --global core.autocrlf input
$ git config --global pull.rebase true
```

4. Maak lokaal een directory aan die je voorbehoudt voor al wat met deze cursus te maken heeft. Binnen deze directory kan je je Github-repository klonen. Klik op de Github-pagina van je repository op de groene knop rechts (*Clone or download*), kies voor "Clone with SSH" kopieer de link van de vorm `git@github.com:HoGentTIN/REPO_NAAM-GEBRUIKERSNAAM.git`.

```
$ cd Documents/Courses/EnterpriseLinux/
$ git clone git@github.com:HoGentTIN/REPONAAM-GEBRUIKERSNAAM.git repo
```

   Dit maakt een lokale kopie van de repository in een subdirectory `repo`. Je kan zelf de naam van deze directory kiezen en die achteraf verplaatsen, alles blijft gewoon werken.

5. Creëer een nieuwe *branch* met de naam `solution` om je eigen code en documentatie bij te houden. Verderop wordt duidelijk waarom dit belangrijk is.

[1] https://education.github.com/pack

[2] Het symbool ~ is natuurlijk de *home directory* van de gebruiker. In Linux komt dit overeen met `/home/USER`, onder Windows met `C:\Users\USER` of `C:\Gebruikers\USER`, onder MacOS met `/Users/USER`

```
$ git checkout -b solution
```

6. Bekijk de bestanden in de `assignment/` directory. Hier vind je de opgave van deelopdrachten en sjablonen voor verslagen en cheat sheets in Markdown-formaat (Gruber, 2004). Pas alvast het sjabloon aan, vul er je eigen naam in.

3.1.1. Errata

Wanneer er errata in de opgave gepubliceerd worden, kan je die relatief eenvoudig binnen halen, maar enkel als je een eigen branch aangemaakt hebt.

1. Eerst moet je zorgen dat je updates kan binnenhalen van de repository met de opdracht. Het volgende commando zorgt dat je kan synchroniseren met die repository. Dit moet slechts één keer gebeuren.

```
$ git remote add upstream https://github.com/HoGentTIN/elnx-sme.git
```

2. Wanneer er nieuwe commits gebeurd zijn in de opgave, kan je de wijzigingen telkens zo ophalen:

```
$ git checkout master
$ git pull upstream master
$ git checkout solution
$ git rebase master
```

   - De eerste twee regels zorgen er voor dat jouw versie van de master-branch up-to-date gebracht wordt met de nieuwe commits.
   - In de derde regel ga je opnieuw naar je eigen branch. Voorlopig is er daar nog niets gewijzigd
   - In de vierde regel, tenslotte, ga je de wijzigingen in de opgave op jouw eigen versie toepassen. Mogelijks komen er hier conflicten naar boven tussen bepaalde bestanden in de opgave en jouw wijzigingen. Lees goed de instructies die Git hier geeft om deze conflicten op te lossen.

   Deze procedure werkt niet als je geen branch voor je eigen oplossing gemaakt hebt. In dat geval haal je jezelf een hoop ellende op de hals...

   Als er conflicten optreden in bestanden, blijft Git in “rebase-modus” steken en kan je voorlopig niet verder werken. Voer eerst `git status` uit om een lijst te krijgen met alle betrokken bestanden.

```
$ git status
rebase in progress; onto e5bd2df
You are currently rebasing branch 'master' on 'e5bd2df'.
  (fix conflicts and then run "git rebase --continue")
  (use "git rebase --skip" to skip this patch)
  (use "git rebase --abort" to check out the original branch)
Unmerged paths:
  (use "git reset HEAD <file>..." to unstage)
  (use "git add <file>..." to mark resolution)

    both modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

Open deze één voor één met je teksteditor en zoek naar de markeringen voor conflicten, bv.

```
If you have questions, please
<<<<<<< HEAD
open an issue
=======
ask your question in IRC.
>>>>>>> branch-a
```

Bewerk alle bestanden met conflicten en verbeter de code. Vervolgens voeg je alle aangepaste bestanden toe met `git add .` en kan je het rebase-proces verder zetten met `git rebase --continue`. Gebruik tussendoor regelmatig `git status` om te zien hoe ver je staat en welke commando's je nodig hebt om een stap verder te gaan.

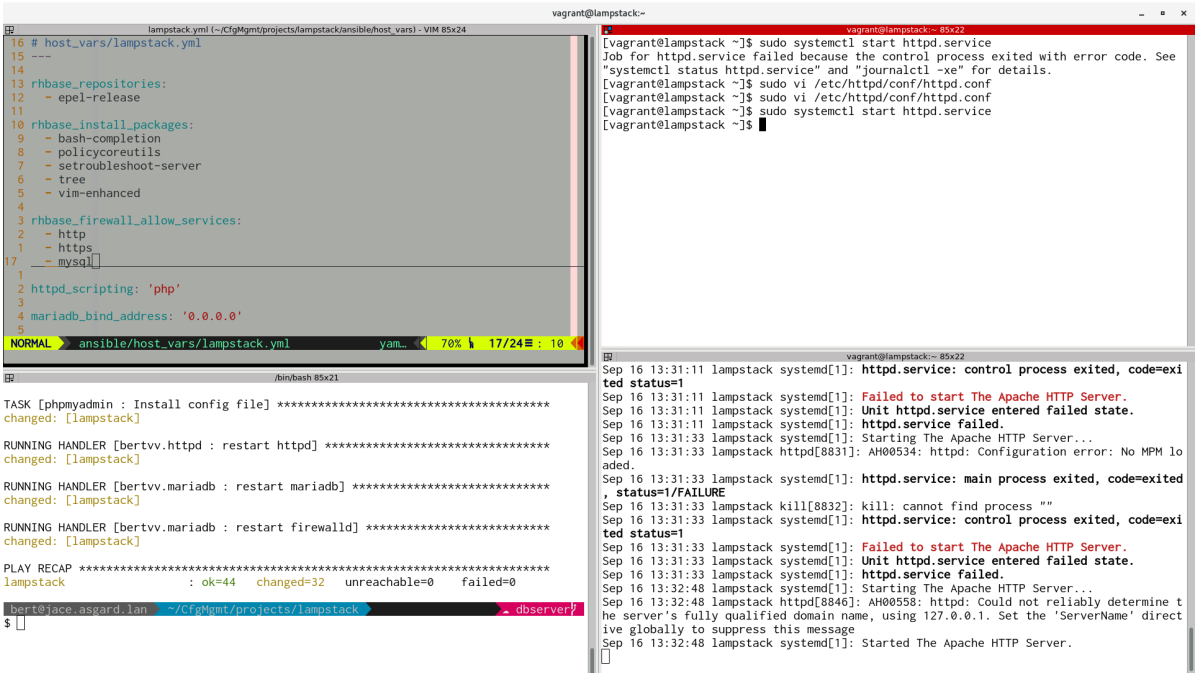
3.2. Algemene richtlijnen

In deze sectie vind je enkele algemene richtlijnen die je helpen vlotter en efficiënter te werken.

- Voor je aan een (deel-)opdracht begint, bereid je eerst voor door alle aangereikte studiematerialen te bestuderen: handleidingen, screencasts, ...Die "van buiten blokken" is helemaal niet nodig, maar zorg er in elk geval voor dat je er in die mate vertrouwd bent, dat je snel gericht kan zoeken naar juiste, rele-

vante informatie. Je vindt die via de bronvermeldingen in deze syllabus, of via de opgave

- Open **verschillende terminalvensters/consoles naast elkaar** (zie Figuur 3.1). Elke terminal krijgt zijn eigen functie, bijvoorbeeld:
  - Vim editor (of VS Code/Sublime/Notepad/...in een apart venster);
  - doorvoeren van wijzigingen aan de configuratie;
  - ingelogd op VM, voor commando's;
  - ingelogd op VM, voor tonen logbestanden.
- **Werk stap voor stap.** Schrijf niet teveel code ineens. Probeer eerst een minimaal werkende opstelling te verkrijgen en registreer meteen in Git. Maak minimale wijzigingen en **test elke wijziging uit**. Hoe groter en ingrijpender de wijzigingen, hoe meer kans op fouten en hoe moeilijker die te debuggen zijn. Zodra iets werkt, en je bent een stap verder, registreer je dit meteen in Git en geef je een duidelijke, beschrijvende commit-boodschap.
- **gebruik Git op de command-line.** Bij de meeste Git commando's krijg je gedetailleerde uitleg over hoe je een stap verder moet gaan en ook hoe je de laatste stap kan ongedaan maken. Dit geeft op de duur een beter inzicht in hoe Git precies werkt.
- Probeer **elke commit te beperken tot één enkele "reden"** om wijzigingen aan te brengen aan de bestaande code. Dit maakt de "geschiedenis" van je project transparanter en maakt ook dat je makkelijker kan terugkeren naar een bepaalde stap wanneer je de mist in gaat.
- **Maak backups** van de originele, ongewijzigde configuratiebestanden zodat je er op kan terugvallen als er iets misloopt. Soms heb je zodanig zitten "prutsen" dat je er niet meer in slaagt de service te laten werken.
- **Gebruik `vagrant destroy`** (zie Sectie 3.4). Wanneer je veel manuele wijzigingen hebt aangebracht in een VM, ben je op de duur niet meer zeker dat die zich in de gewenste toestand bevindt. Of je kan door experimenteren de VM onbruikbaar gemaakt hebben. Door de VM te verwijderen en opnieuw op te bouwen (met `vagrant up`) kan je opnieuw beginnen van een werkende versie (als je de vorige richtlijnen opvolgt, tenminste!).
- Ook wanneer je denkt klaar te zijn met een deelopdracht, genereer de VM nog eens helemaal opnieuw en voer alle acceptatietests uit.



Figuur (3.1)

Gebruik van verschillende terminals. In deze schermafbeelding is Terminator (een terminal-applicatie voor Linux) opgedeeld in vier aparte shells. Linksboven is Vim geopend met een broncodebestand. De terminal linksonder wordt gebruikt om wijzigingen in de broncode door te voeren op de virtuele machines. Rechtsboven is er ingelogd op de virtuele machine waar momenteel wijzigingen op aangebracht worden en kunnen commando's uitgevoerd worden om zaken uit te proberen of problemen op te sporen. Rechtsonder, tenslotte, worden relevante logbestanden van de VM getoond.

3.3. Bash tips

In deze sectie vind je een aantal tips voor het gebruik van de Bash-shell. In deze cursus maak je intensief gebruik van de shell, en het loont de moeite om Bash wat beter te leren kennen. Het zit immers vol met interessante features die je toelaten productiever te werken. Online is hier nog veel meer over te vinden (Rowe, 2009).

De tips hier gelden ook voor Windows- en MacOS-gebruikers! Windows-gebruikers hebben beschikking over een recente versie van Bash via Git (Git Bash). In MacOS X zit ook een Bash-shell, weliswaar een zeer oude versie[3]. Je kan een recente versie installeren via HomeBrew[4].

- **Gebruik `TAB`-*completion*.** Je kan een deel van een commando of pad intikken en dan de `TAB`-toets indrukken. Indien mogelijk zal Bash het woord vervolledigen, of mogelijke alternatieven tonen. Als je de package `bash-completion` installeert zijn er nog veel meer mogelijkheden.
- **Gebruik de *command history*.** Bash houdt de commando's die je eerder gebruikt hebt bij. Met de pijltjestoetsen kan je eerdere commando's terug halen. Gebruik `Ctrl-R` om te zoeken in de command history. Je kan dan een fragment van het commando intikken, Bash toont dan het laatste commando waar dat tekstfragment in voorkomt.
- **Bang bang!** In Bash zijn er enkele shortcuts voorzien voor (delen van) het vorige commando. `!!` staat voor het gehele vorige commando, `!$` voor het laatste argument en `!*` voor alle argumenten. Een voorbeeldje van het gebruik:

```
$ yum update
You need to be root to perform this command.
$ sudo !!
sudo yum update
[...]
$ mkdir -p some/long/path/i/dont/want/to/repeat
$ cd !$
cd some/long/path/i/dont/want/to/repeat
$
```

- **Toetsenbordcombinaties.** Naast `Ctrl-R` van hierboven zijn er nog een aantal nuttige toetsenbordcombinaties.

[3] Dit is omwille van de softwarelicentie. Recente versies van Bash vallen onder de open source GPL-licentie, waar Apple niets mee te maken wil hebben.
[4] http://brew.sh

- `Ctrl-C` onderbreekt het huidige proces
- `Ctrl-Z` pauzeert het huidige proces (haal het terug met `fg`)
- `Ctrl-D` sluit de shell af (enkel op lege regel)
- `Ctrl-K` verwijdert de tekst rechts van de cursor
- `Ctrl-U` verwijdert de tekst links van de cursor
- `Ctrl-T` verwisselt letterteken onder de cursor met dat voor de cursor
- `Alt-T` verwisselt woord voor met woord na de cursor

Er bestaan nog *veel* meer combinaties. Deze kan je vinden in de man-pagina `bash(1)` in de sectie READLINE (subsectie *Readline Command Names* en volgende).

- **Personaliseer de shell.** Je kan het gedrag van de shell in hoge mate aan je eigen wensen aanpassen door het bestand `~/.bashrc` aan te passen. Je kan het gedrag van tab-completion of de command history aanpassen, een eigen commandoprompt instellen, aliassen (zie verder) of functies definiëren, enz. Het zou ons te ver leiden om alle mogelijkheden uit te diepen, maar je kan een relatief eenvoudig voorbeeld vinden in https://github.com/bertvv/server-dotfiles/ (installatie-instructies te vinden in de README), en een meer uitgebreid voorbeeld in https://github.com/bertvv/dotfiles/ (of gelijknamige repositories van andere Github-gebruikers).

- **Aliassen** zijn een soort shortcuts voor commando's die je vaak gebruikt. Dit kan enorm veel typwerk besparen. Je kan ze toevoegen in je `~/.bashrc`. Enkele voorbeelden ter inspiratie:

```
1 alias l='ls -l --si --time-style=long-iso --color'
2 alias a='git add'
3 alias c='git commit -m'
4 alias h='git log --pretty="format:%C(yellow)%h %C(blue)%ad
      %C(reset)%sC(red)%d %C(green)%an%C(reset), %C(cyan)%ar"
     --date=short --graph --all'
5 alias s='git status'
6 alias vu='vagrant up'
7 alias vD='vagrant destroy'
8 alias elnx='cd /home/bert/Documents/Courses/EnterpriseLinux/16-17'
```

Nog meer voorbeelden vind je in https://github.com/bertvv/dotfiles/blob/master/.bash.d/aliases.sh.

3.4. Vagrant

Vagrant is een command line tool die het aanmaken en configureren van virtuele machines automatiseert. Het ondersteunt een aantal virtualisatieplatforms, o.a. VirtualBox, Hyper-V, libvirt, enz. Wij zullen het gebruiken in combinatie met VirtualBox. Voor een demo van de werking van Vagrant, bekijk eerst de screencast van Weissig (2014).

Je Git repository bevat een Vagrant-omgeving voor het opzetten van de VMs voor je labo-opdracht. Er zijn alvast twee VMs gedefinieerd. Je kan een overzicht van de VMs opvragen met (voorbeeld uit de SME-opdracht van Sectie 2.1.1):

```
$ vagrant status
Current machine states:

router                    not created (virtualbox)
pu001                     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 initiële setup bevat twee virtuele machines met namen `router` en `pu001`, respectievelijk. We beginnen met `pu001`, de router wordt later geconfigureerd.

Start de VM met `vagrant up pu001`. De eerste keer dat je dit doet wordt er een basis-VM gedownload met een minimale installatie van de laatste stabiele versie van CentOS. Doe dit best op een performant netwerk: het bekabelde netwerk op de campus of thuis/op kot. Deze “base box” wordt bijgehouden en zal telkens dienst doen als basis voor het opzetten van alle hosts in ons netwerk. Het downloaden gebeurt dus slechts één keer-.

Na opstarten kan je inloggen met `vagrant ssh pu001`. 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. Het commando `vagrant` moet altijd uitgevoerd worden vanuit de directory waar het bestand `Vagrantfile` zich bevindt.

De belangrijkste Vagrant commando's worden opgesomd in Tabel 3.1. 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`

Tabel 3.1: De belangrijkste `vagrant`-commando's

Alle VMs in de opstelling krijgen twee netwerkinterfaces:

- Adapter 1: NAT-interface (de "management interface" voor Vagrant, Internetverbinding voor de VM)
- Adapter 2: Host-only interface (gebruikt om vanop het hostsysteem de netwerkservices op de VM aan te spreken)

Zorg dat je goed begrijpt hoe deze interfaces werken, zoniet wordt troubleshooten erg moeilijk (Van Vreckem, 2015c).

3.5. Ansible

Ansible 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 brengt het systeem naar die toestand. Bekijk de screencast van Weissig (2015) voor een inleiding op Ansible. In een vervolg-screencast wordt ook de combinatie van Ansible en Vagrant belicht. In deze gids geven we enkel wat uitleg om aan de slag te gaan met de startopstelling

voor de labo-opdracht, maar als je op zoek bent naar een goed naslagwerk over Ansible, overweeg dan om het e-boek van Geerling (2016) te kopen. Het boek is al enkele jaren oud, maar de auteur werkt nog verder aan het boek, updates zijn gratis.

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 variant van JSON), 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*[5] genoemd. Als je deze playbooks op een specifieke manier structureert en herbruikbaar maakt, spreekt men van een *rol*[6]. Je kan rollen toekennen aan een host, en specifieke instellingen toepassen door het invullen van *variabelen*[7]. Op Ansible Galaxy[8] kan je tientallen rollen vinden die door hun auteurs gepubliceerd zijn.

De directorystructuur van een Ansible-project ligt vast en wordt uitvoerig beschreven in de documentatie[9].

3.5.1. Rollen toekennen aan hosts

Om een rol toe te kennen aan een host, bewerk je de *master playbook* `ansible/site.yml`. Dit bestand bevat een overzicht van alle hosts onder het beheer van Ansible, met de rollen van elke host (spaties worden getoond):

```
1 # site.yml
2 ---
3 - hosts: pu001
4   become: true
5   roles: []
```

Host `pu001` heeft nog geen rollen toegekend, dat gaan we nu veranderen:

```
1 # site.yml
2 ---
3 - hosts: pu004
```

[5] https://docs.ansible.com/ansible/playbooks.html
[6] https://docs.ansible.com/ansible/playbooks_roles.html#roles
[7] https://docs.ansible.com/ansible/playbooks_variables.html
[8] https://galaxy.ansible.com/
[9] https://docs.ansible.com/ansible/playbooks_best_practices.html

```
  become: true
  roles:
    - bertvv.rh-base
```

Onder `roles:` kan je zoveel rollen toevoegen als nodig. Let goed op de indentatie. Je **moet** telkens inspringen met 2 spaties, en alle data op hetzelfde niveau moet links mooi uitgelijnd zijn.

3.5.2. Rollen installeren

Alle rollen die gebruikt worden in `site.yml` moeten beschikbaar zijn in de directory `ansible/roles/` in een subdirectory met dezelfde naam als deze gebruikt in `site.yml`.

De naam van de rol `bertvv.rh-base` is geschreven in de vorm `AUTEUR.ROLNAAM`. Dit wijst er op dat deze rol op Ansible Galaxy gepubliceerd is. Je kan die daar terugvinden onder de url "http://galaxy.ansible.com/AUTEUR/ROLNAAM/." Je kan daar de rol downloaden en uitpakken op de juiste plaats in de Ansible-directorystructuur.

Dit is het eenvoudigste als je **MacOS of Linux** op je hostsysteem draait. Je kan dan Ansible installeren en gebruik maken van het commando `ansible-galaxy`:

```
$ ansible-galaxy -p ansible/roles install bertvv.httpd
- downloading role 'httpd', owned by bertvv
- downloading role from
    https://github.com/bertvv/ansible-role-httpd/archive/v1.2.1.tar.gz
- extracting bertvv.httpd to ansible/roles/bertvv.httpd
- bertvv.httpd was installed successfully
$
```

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

Onder **Windows** wordt Ansible niet ondersteund, en is dit commando dus ook niet beschikbaar. Je kan wel manueel een rol downloaden van de Github-repository (bijvoorbeeld voor `bertvv.rh-base` van https://github.com/bertvv/ansible-role-rh-base/releases) en dan op de juiste plaats uitpakken.

In je repository is ook een scriptje voorzien dat het installatieproces automatiseert: `scripts/role-deps.sh`. Als je het uitvoert in een Bash shell vanuit de hoofddirectory

van je repository, zal het in `ansible/site.yml` alle rollen die er vernoemd worden van Ansible Galaxy (of indien de rol daar niet gepubliceerd is van Github) downloaden en installeren. Dit script werkt trouwens ook op MacOS en Linux.

```
$ ./scripts/role-deps.sh
- downloading role 'rh-base', owned by bertvv
- downloading role from
    https://github.com/bertvv/ansible-role-rh-base/archive/v2.3.0.tar.gz
- extracting bertvv.rh-base to
    /home/bert/Documents/Vakken/enterprise-linux/opgaven/elnx-sme-bertvv/ansible/roles/be
- bertvv.rh-base (v2.3.0) was installed successfully
$
```

Merk op dat rollen van Ansible Galaxy genegeerd worden door Git. Het heeft geen zin je eigen repository te "vervuilen" met code die al elders onderhouden wordt. Het is dus ook de bedoeling dat je *geen wijzigingen* aanbrengt in bestaande rollen. Als dit toch nodig is (omdat er bv. features ontbreken die je zelf geïmplementeerd hebt), is het beter de oorspronkelijke rol te "forken" op Github en daar te onderhouden.

Na installatie van de rollen, kan je met `vagrant provision` de *master playbook* uitvoeren.

3.5.3. Hosts configureren

Als je de documentatie voor een rol naleest (typisch het bestand README.md van de Github repository), vind je in principe 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 voorbeeld:

```
---
rhbase_repositories:
  - epel-release
rhbase_install_packages:
  - bash-completion
  - vim-enhanced
```

```
7 rhbase_firewall_allow_services
8   - http
9   - https
```

Telkens je een wijziging aanbrengt in de Ansible-configuratie, kan je die doorvoeren met `vagrant provision`. Het is dan normaal niet nodig om de VM te rebooten of te verwijderen en opnieuw op te zetten.

3.6. Basiskennis CentOS

CentOS is een Linux-distributie die afgeleid is van RedHat, maar initieel volledig door de *community* onderhouden werd. RedHat is een commerciële Linux-distributie van het gelijknamige bedrijf. Je kan RedHat Enterprise Linux (RHEL) enkel gebruiken als je er een support contract voor tekent. Nu is RHEL wel volledig open source en is het dus toegelaten om de code te hergebruiken en te wijzigen. Dat is precies wat in het CentOS-project gebeurd is. Alles wat onder het merkenrecht valt (de naam RHEL, logo's, enz.) is verwijderd en vervangen. CentOS is dus volledig compatibel met RHEL, maar gratis, zij het zonder support vanuit RedHat. RedHat heeft echter wel goede relaties met het CentOS-project, zelfs in die mate dat ze sinds 2014 het project beginnen sponsoren zijn en de belangrijkste medewerkers van het project in dienst genomen hebben. In deze cursus gebruiken we CentOS omdat de RedHat-familie in het Vlaamse bedrijfsleven de meest gebruikte Linux-distributie is.

Om je te helpen bij het leren kennen van deze distributie vind je hieronder enkele suggesties.

- **Lees eerst de RHEL 7-documentatie** (Jahoda, Ančincová & Čapek, 2016; Jahoda, Heves e.a., 2016; Svistunov e.a., 2016), ga niet in het wilde weg Googlen naar commando's. Recent zijn er redelijk wat ingrijpende wijzigingen gebeurd in het Linux-landschap waardoor vele commando's die nog altijd in blog-artikels en HOWTO's staan niet meer werken of verouderd zijn. De delen van de handleidingen die bij uitstek relevant zijn, worden opgesomd in Sectie 1.2.
- Maak voor jezelf een **cheat sheet** met de belangrijkste commando's, in het bijzonder voor:
  - netwerkinstellingen opvragen met `ip`;
  - de werking van `sudo`;

- installatie van software beheren met `yum`;
- services beheren met `systemctl`;
- de firewall beheren met `firewall-cmd`;
- de juiste commando's voor het valideren van configuratiebestanden, bv.
  - Samba: `testparm`
  - Apache: `apachectl configtest`
  - BIND: `named-checkconf`
  - ISC-dhcpd: `dhcpd -t`
  - ...

Je kan je laten inspireren door deze cheat sheet: https://github.com/bertvv/cheat-sheets/blob/master/src/EL7.md Kopieer deze niet blind! Je moet je eigen cheat sheet zelf opstellen, zo zal je de commando's beter onthouden.

3.7. DNS en BIND

DNS is essentieel voor de correcte werking van een domein, en redelijk wat (volgens sommigen *alle*[10]) netwerkproblemen zijn terug te leiden tot fouten in DNS. Er zijn verschillende implementaties van DNS, maar veruit de meest gebruikte (en dus essentieel voor de werking van het Internet als geheel) is BIND[11].

In deze syllabus geven we een heel summiere inleiding. Voor een uitgebreid overzicht van de werking en configuratie van BIND, zie Aitchison (2015).

3.7.1. Zonebestanden

DNS is op zich geen complexe netwerkservice. Het komt neer op een databank met enkele tabellen (= zonebestanden) die in (een strak) tekstformaat zijn opgemaakt. Er zijn verschillende types van records, o.a.

- `A` bevat het IPv4-adres voor een gegeven hostnaam;
- `AAAA` idem, maar voor IPv6;
- `CNAME` bevat de "originele" hostnaam voor een gegeven alias (kan bijvoorbeeld worden gebruikt voor "www.linuxlab.lan");
- `MX` bevat verwijzingen naar de mailservers voor dit domein;

[10] http://www.krisbuytaert.be/blog/
[11] https://www.isc.org/downloads/bind/

- NS bevat verwijzingen naar de DNS-servers voor dit domein;
- PTR bevat de hostnaam voor een gegeven IP-adres (bevindt zich normaal in een zgn.*reverse lookup* zonebestand);
- enz. (lees de documentatie!)

Een zonebestand begint met een zgn. SOA-record, wat staat voor Start Of Authority. Hieronder vind je een voorbeeld van een *forward zone file* voor een domein met de naam "linuxlab.lan" voor het IPv4-netwerk 192.168.15.0/24. Op de precieze betekenis gaan we hier niet in, dit wordt elders voldoende uitgediept (Aitchison, 2015).

```
; /var/named/linuxlab.lan
; Forward lookup zone file for `linuxlab.lan.'
$ORIGIN linuxlab.lan.
$TTL 1W
;        primary DNS   email address admin
@ IN SOA srv001        hostmaster (
   2015101216   ; serial
   1D           ; refresh
   1H           ; retry
   1W           ; expire
   1D           ; minimum TTL
)
```

De overeenkomstige *reverse zone file* begint als volgt:

```
; /var/named/15.168.192.in-addr.arpa.
; Reverse lookup zone file for `linuxlab.lan.'
$ORIGIN 15.168.192.in-addr.arpa.
$TTL 1W
@ IN SOA srv001.linuxlab.lan. hostmaster.linuxlab.lan. (
   2015101216   ; serial
   1D           ; refresh
   1H           ; retry
   1W           ; expire
   1D           ; minimum TTL
)
```

De syntax van deze *zone files* is heel strak en het is makkelijk om fouten te maken.

De meest voorkomende fouten zijn de volgende:

- Hostnamen die volledig uitgeschreven zijn (*fully qualified domain name* of FQDN) moeten in een zonebestand altijd afgesloten worden met een punt, bv. "`pu001.linuxlab.lan.`". Namen die niet op een punt eindigen, worden aangevuld met de waarde van `$ORIGIN` die aan het begin van een zonebestand gegeven wordt (d.i. de domeinnaam, in ons geval "`linuxlab.lan.`"). Bv. `pu002` wordt dan "`pu002.linuxlab.lan.`". Als je een hostnaam volledig uitschrijft en je vergeet de punt, dan zal de domeinnaam dus verkeerd geïnterpreteerd worden. "`pu002.linuxlab.lan`" wordt immers omgezet naar "`pu002.linuxlab.lan.linuxlab.lan.`".
- IP-adressen worden op een eigenaardige manier genoteerd. Ten eerste wordt het host-deel van het netwerkadres niet geschreven, de getallen in de "dotted quad"-notatie worden omgekeerd en je moet er "`in-addr.arpa.`" achter schrijven. Met andere woorden, `192.0.2.0/24` wordt als "`2.0.192.in-addr-arpa.`" geschreven.

3.7.2. DNS troubleshooting

Het is niet altijd evident om fouten op te sporen in de configuratie van een BIND DNS-server. Daarom deze tips:

- Test de syntax van configuratiebestanden voordat je de service opstart. Voor het hoofd-configuratiebestand `/etc/named.conf` is het commando:

```
$ sudo named-checkconf /etc/named.conf
```

- De syntax testen van zonebestanden gebeurt zo (voorbeeld voor forward en reverse zone, respectievelijk):

```
$ sudo named-checkzone linuxlab.lan /var/named/linuxlab.lan
$ sudo named-checkzone 15.168.192.in-addr.arpa \
    /var/named/15.168.192.in-addr.arpa
```

- Je kan foutboodschappen van de service bekijken met `journalctl`. Het handigste is om een aparte console te openen, in te loggen op je server en dan het volgende commando uit te voeren:

```
$ sudo rndc querylog on
$ sudo journalctl -l -f -u named.service
```

Het eerste commando zorgt dat DNS-queries van clients in de logs verschijnen (wat standaard uit staat). Het tweede commando toont de logs van BIND.

Open een andere console om commando's uit te voeren (bv. de service herstarten). Je ziet dan meteen relevante info- en foutboodschappen verschijnen.

- DNS werkt zowel over UDP als TCP, afhankelijk van welk protocol de client kiest. Controleer dus je firewall-instellingen! Het beste is om de dns-service toe te voegen aan de regels in plaats van de poortnummers te specifiëren:

```
  $ sudo firewall-cmd --add-port=53/udp --permanent # --> FOUT
  $ sudo firewall-cmd --add-service=dns --permanent # --> GOED
```

Naast het valideren van de configuratiebestanden, is het ook belangrijk om te testen of de DNS-server ook correct antwoordt op aanvragen van clients. Installeer de package `bind-tools` om een aantal nuttige tools ter beschikking te krijgen. Het commando `nslookup` zal je misschien herkennen vanuit Windows. Het werkt hier op dezelfde manier:

```
$ nslookup www.hogent.be
Server:   195.130.131.1
Address:  195.130.131.1\#53

Non-authoritative answer:
Name:  www.hogent.be
Address: 178.62.144.90
```

Hier wordt gevraagd wat het IP-adres is voor hostnaam "www.hogent.be". Als antwoord krijgen we 178.62.144.90. Het antwoord werd gegeven door een DNS-server met IP-adres 195.130.131.1. Er wordt aangegeven dat het antwoord *non-authoritative* is, wat wil zeggen dat de DNS-server die het antwoord gaf niet de DNS-server is die ook de eindverantwoordelijke is voor het hogent.be-domein.

Je kan een query met `nslookup` ook richten naar een specifieke DNS-server door die ook mee te geven op de command line:

```
$ nslookup www.hogent.be 8.8.8.8
Server:^^I^^I8.8.8.8
Address:^^I8.8.8.8#53

Non-authoritative answer:
Name:^^Iwww.hogent.be
Address: 178.62.144.90
```

Merk op dat als je de DNS-server niet expliciet vermeldt, de DNS-server ondervraagd wordt die door DHCP werd toegewezen en die vermeld wordt in het bestand `/etc/resolv.conf`.

Naast `nslookup`, bevat de `bind-utils` een nog veelzijdiger commando om DNS-servers te ondervragen: `dig`.

```
$ dig www.hogent.be

; <<>> DiG 9.10.5-P2-RedHat-9.10.5-2.P2.fc25 <<>> www.hogent.be
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23001
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.hogent.be.^^I^^I^^IIN^^IA

;; ANSWER SECTION:
www.hogent.be.^^I^^I2796^^IIN^^IA^^I178.62.144.90

;; Query time: 11 msec
;; SERVER: 195.130.131.1#53(195.130.131.1)
;; WHEN: Tue Sep 26 00:45:51 CEST 2017
;; MSG SIZE  rcvd: 58
```

De syntax van de uitvoer is compatibel met die van zonebestanden. We zien hier dus dat er voor www.hogent.be een A-record bestaat dat verwijst naar IP-adres 178.62.144.90. Alle regels die beginnen met een kommapunt zijn commentaar.

Je kan vragen enkel de relevante info af te drukken, zonder de commentaren:

```
$ dig +short www.hogent.be
178.62.144.90
```

Een specifieke DNS-server ondervragen gebeurt door die op de command-line mee te geven, voorafgegaan door een "@":

```
$ dig +short @8.8.8.8 www.hogent.be
178.62.144.90
```

Tenslotte kan je ook verschillende soorten records opvragen. Bijvoorbeeld, wie is de "authoritative name server" voor het domein hogent.be?

```
$ dig +short NS hogent.be
ns2.belnet.be.
ens2.hogent.be.
ns1.belnet.be.
ens1.hogent.be.
```

Wat is het IPv6 adres voor download.fedoraproject.org?

```
$ dig +short AAAA download.fedoraproject.org
wildcard.fedoraproject.org.
2001:4178:2:1269::fed2
2610:28:3090:3001:dead:beef:cafe:fed3
2605:bc80:3010:600:dead:beef:cafe:fed9
```

Zo doe je een "reverse lookup":

```
$ dig +short -x 195.130.131.1
asse.dnscache02.telenet-ops.be.
```

Hier krijg je enkel antwoord voor als de ondervraagde DNS-server ook effectief een PTR-record heeft voor het opgegeven IP-adres.

3.8. Fileservers en Samba

De meest gebruikte manier om met Linux een fileserver op te zetten die voor alle desktop-operating systems beschikbaar is, is met Samba. Het Samba-project heeft al een lange geschiedenis achter de rug en is eigenlijk een onafhankelijke implementatie van het SMB-protocol dat je misschien beter kent als Windows Network Neighbourhood.

De laatste versie van Samba laat zelfs toe als een Active Directory Domain Controller op te treden, maar dat valt buiten het bestek van deze cursus.

Het opzetten van een Samba fileserver (Van Vreckem, 2014; Vernooij, Terpstra & Carter, 2010) is soms een uitdaging, vooral wat betreft het juist instellen van de toegangsrechten. Als je bepaalde gebruikers wil lees- of schrijftoegang geven tot een share, dan moet dit op drie verschillende niveaus correct ingesteld zijn:

1. **Bestandspermissies:** De gewone bestandspermissies moeten de gebruiker lees- of schrijftoegang geven.
2. **Samba configuratie:** De share moet via het Samba-configuratiebestand `/etc/samba/smb.conf` de juiste toegangsrechten toekennen aan de gebruiker
3. **SELinux:** De directory moet de juiste SELinux context hebben (zie RedHat manual)

Als ook maar één van deze drie elementen te streng is ingesteld, hebben de gebruikers niet de gewenste toegang. Door dit proces te automatiseren, kan je vervelende fouten (en de tijd die nodig is die op te lossen) vermijden.

Samba voorziet een commando voor het controleren of het configuratiebestand `/etc/samba/smb.conf` correct is: `testparm`. Het drukt ook de inhoud van het configuratiebestand af in de meest eenvoudige en compacte vorm. Dit kan ook van pas komen om de configuratie die je zelf hebt opgebouwd te “optimaliseren”. Samba voorziet namelijk standaardinstellingen die je niet moet expliciet schrijven (bijvoorbeeld `guest ok = no`) en er zijn ook vaak verschillende manieren om hetzelfde te schrijven (bijvoorbeeld `guest ok = no` is het zelfde als `public = yes`). Sommige opties worden zelfs genegeerd afhankelijk van de waarde van andere (bijvoorbeeld `guest only` heeft geen effect als `guest ok` niet is ingesteld). Geef de optie `-s` of `--suppress-prompt` mee, om te vermijden dat er gevraagd wordt om “ENTER” in te drukken voordat je het overzicht van de configuratie te zien krijgt.

Om te testen of de share toegankelijk is van buitenaf, kan je vanop het hostsysteem in de file explorer werken, maar dit is niet zo interessant. De foutboodschappen

geven weinig of geen informatie die helpt bij het vinden van de oorzaak van het probleem. Gebruik liever `smbclient`, daarmee krijg je alvast iets duidelijker foutboodschappen. Enkele voorbeelden van het gebruik:

- Geef een overzicht van de shares op server `FILES`

```
smbclient -L //files/
```

- Log in op een share als gebruiker `lizae` met wachtwoord `letmein`

```
smbclient //files/public/ -Ulizae%letmein
```

- Log in op een share als "gast"

```
smbclient //files/public -U%
```

Voor gedetailleerde richtlijnen naar het troubleshooten van Samba, zie Carter, Vernooij, Bannon en Shearer (2010), Trigdell, Vernooij en Shearer (2010)

3.9. Troubleshooting

Het gebruik maken van een configuration management system laat een systeembeheerder toe om op een gecontroleerde en betrouwbare manier snel netwerkservices in productie te brengen. Het configuration management system vereenvoudigt het opzetten van een service en door de doorgedreven automatisering worden fouten tot een minimum beperkt.

Jammer genoeg ontslaat dit ons niet van de verantwoordelijkheid om de systemen die we beheren van binnen en van buiten te kennen en de interne werking te begrijpen. Wanneer er toch fouten de kop op steken en onze systemen zijn niet beschikbaar voor onze gebruikers, brengt het configuration managementsysteem niet altijd soelaas. Integendeel, op dat moment is het niet meer dan een extra laag complexiteit.

Een systematische en grondige aanpak kan uren werk uitsparen. De impact van de onbeschikbaarheid van netwerkservices is meestal bijzonder zwaar. In het beste geval heb je boze gebruikers, maar in het slechtste geval kan je bedrijf tienduizenden euro's per uur verliezen aan verloren productiviteit en potentiële inkomsten. Op zo'n moment gaat met googlen naar een oplossing kostbare tijd verloren (als

je al *kan* googlen), en als je op dat moment nog moet beginnen handleidingen lezen, wordt het zeker nachtwerk.

De beste manier om het troubleshooten van netwerkservices systematiseren is om de TCP/IP-stack als model te nemen. Test eerst de onderste laag, en pas als daar alle mogelijke problemen zijn opgespoord ga je naar de laag er boven: *bottom-up troubleshooting*.

1. Datalinklaag: kabels, netwerkpoorten op de switch/router, netwerkkaart, enz.;
2. Internetlaag: IP adresconfiguratie, default gateway, DNS-server en connectiviteit binnen het LAN;
3. Transportlaag: toestand netwerkservice, open netwerkpoorten, firewall-instellingen;
4. Applicatielaag: fouten in configuratiebestanden, logbestanden, bereikbaarheid service vanop het netwerk, enz.

Deze werkwijze wordt verder uitgediept in Van Vreckem (2015b). **Bestudeer die grondig**, verwerk wat je er leert in je eigen cheat sheet. Tip: je kan een gelijkaardige checklist maken voor bv. Windows Server. De werkwijze is net hetzelfde, je moet enkel opzoeken hoe je op Windows deze zaken kan controleren.

Nog enkele tips bij het troubleshooten:

- Werk je checklist bij wanneer je nieuwe dingen leert.
- Wees **grondig**. Sla nooit stappen over (een typische: kabels vergeten nakijken, of IP instellingen onvoldoende controleren).
- Lees de **foutboodschappen**. Die geven aan wat er precies misloopt. Zoek ze op met Google.
- Werk in **kleine stappen** en verifieer elke stap.
- Geen veronderstellingen! **Testen!**
- Maak een **backup van configuratiebestanden**, in elk geval de *originele* en de laatst gekende werkende versie.
- **Valideer configuratiebestanden** voordat je de service opstart.
- Werk met verschillende terminals, gebruik er minstens één voor het tonen van de logbestanden (`journalctl -f`)

3.10. Vaak voorkomende problemen

In deze sectie worden enkele vaak voorkomende problemen met hun oplossing opgesomd.

3.10.1. VM opstarten vanuit VirtualBox GUI

```
Vagrant was unable to mount VirtualBox shared folders. This is usually
because the filesystem "vboxsf" is not available. This filesystem is
made available via the VirtualBox Guest Additions and kernel module.
Please verify that these guest additions are properly installed in the
guest. This is not a bug in Vagrant and is usually caused by a faulty
Vagrant box. For context, the command attempted was:

mount -t vboxsf -o uid=1000,gid=1000 vagrant /vagrant

The error output from the command was:

/sbin/mount.vboxsf: mounting failed with the error: No such device
```

Deze foutboodschap geeft aan dat de directory `/vagrant`, die je projectdirectory op je fysieke systeem voorstelt (de kloon van je Github-repo), niet beschikbaar is binnen de VM. Meestal is de oorzaak dat je de VM hebt opgestart vanuit de VirtualBox GUI in plaats van via het commando `vagrant up`. Het commando zal niet enkel de VM aanzetten, maar ook een aantal basisinstellingen toepassen zoals port forwarding voor SSH, netwerkinterfaces een IP adres toekennen, en de `/vagrant` directory beschikbaar maken binnen de VM.

Start VMs altijd op via de command line. Je hebt de VirtualBox GUI *niet* meer nodig, tenzij voor troubleshooting.

Bibliografie

Aitchison, R. (2015). *DNS for Rocket Scientists*. Verkregen 16 september 2016, van http://www.zytrax.com/books/dns/

Andrade, H. G. (2000). Using Rubrics to Promote Thinking and Learning: i. *Educational Leadership*, *57*(5), 13–18. Verkregen 15 september 2016, van http://www-tc.pbs.org/teacherline/courses/rdla230/docs/session_2_andrade.pdf

Ansible, Inc. (2016). *Ansible Documentation*. Verkregen 15 september 2016, van https://docs.ansible.com/ansible/

Aun. (2016, maart 30). 10 Tips To Improve MariaDB Performance. Verkregen 15 september 2016, van http://linuxpitstop.com/tips-to-improve-mariadb-performance/

Blanc, M. (2015). Transparent encryption with ansible vault revisited. Verkregen 16 september 2015, van https://leucos.github.io/articles/transparent-vault-revisited/

Cameron, T. (2012). SELinux for mere mortals (RedHat Summit 2012, Boston). Verkregen 15 september 2016, van https://www.youtube.com/watch?v=MxjenQ31b70

Carter, G., Vernooij, J., Bannon, D. & Shearer, D. (2010). Analyzing and Solving Samba Problems. In J. R. Vernooij, J. H. Terpstra & G. Carter (Red.), *The Official Samba 3.5.x HOWTO and Reference Guide*. Verkregen van https://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/problems.html

Cobbaut, P. (2015, mei 24). Linux Fundamentals. Verkregen 25 september 2017, van http://linux-training.be/linuxfun.pdf

Davila, J. (2015). Automatically testing and validating the Ansible STIG Role for Red Hat 6. Verkregen 16 september 2015, van http://blog.davila.io/posts/automatically-testing-and-validation-the-ansible-stig-role-for-red-hat-6.html

DeHaan, M. (2014). Ansible: Python-Powered Radically Simple IT Automation (PyCon 2014). Verkregen 15 september 2016, van http://youtu.be/Qi0AhK7PMCI

Geerling, J. (2016, september 6). *Ansible for Devops: Server and configuration management for humans*. Leanpub. Verkregen 16 september 2016, van https://leanpub.com/ansible-for-devops

Github, Inc. (2016). *Getting started with writing and formatting on GitHub*. Verkregen 16 september 2016, van https://help.github.com/articles/getting-started-with-writing-and-formatting-on-github/

Gruber, J. (2004). Markdown. Verkregen 16 september 2016, van https://daringfireball.net/projects/markdown/

Hashicorp. (g.d.). *Vagrant Documentation*. Verkregen 15 september 2016, van https://www.vagrantup.com/docs/

Hayden, M. (2015). Automated Testing for Ansible CIS Playbook on RHEL/CentOS 6. Verkregen 16 september 2015, van https://major.io/2015/08/05/automated-testing-for-ansible-cis-playbook-on-rhelcentos-6/

Jahoda, M., Ančincová, B. & Čapek, T. (2016). *Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide*. Red Hat, Inc. Verkregen 15 september 2016, van https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/SELinux_Users_and_Administrators_Guide/

Jahoda, M., Heves, J., Wadeley, S. & Huffman, C. (2016). *Red Hat Enterprise Linux 7 Networking Guide*. Red Hat, Inc. Verkregen 15 september 2016, van https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/

Johnson, A. (2015). Making Ansible a bit faster. Verkregen 16 september 2015, van http://adamj.eu/tech/2015/05/18/making-ansible-a-bit-faster/

mnx.io. (g.d.). A proper server naming scheme. Verkregen van https://mnx.io/blog/a-proper-server-naming-scheme/

Moundalexis, A. (2016, september 6). Locking Down an SSH Server. Verkregen 15 september 2016, van http://www.moundalexis.com/v2/2016/09/06/securing-sshd.html

Mytton, D. (2015). Server Naming Conventions and Best Practices (Server Density blog). Verkregen 16 september 2016, van https://blog.serverdensity.com/server-naming-conventions-and-best-practices/

Rowe, S. (2009). Advancing in the Bash Shell. Verkregen 16 september 2016, van http://samrowe.com/wordpress/advancing-in-the-bash-shell/

Sawiyati. (2014). How to install Fail2ban on CentOS. Verkregen 19 september 2015, van http://www.servermom.org/install-fail2ban-centos/1809/

Simmons. (2009). If you can't script it, use a checklist. Verkregen 16 september 2015, van http://www.standalone-sysadmin.com/blog/2009/07/if-you-cant-script-it-use-a-checklist/

stribika. (2015). Secure Secure Shell. Verkregen 16 september 2015, van https://stribika.github.io/2015/01/04/secure-secure-shell.html

Svistunov, M., Wadeley, S., Čapek, T. & Hradílek, J. (2016). *Red Hat Enterprise Linux 7 System Administrator's Guide*. Red Hat, Inc. Verkregen 15 september 2016, van https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/

Trigdell, A., Vernooij, J. & Shearer, D. (2010). The Samba Checklist. In J. R. Vernooij, J. H. Terpstra & G. Carter (Red.), *The Official Samba 3.5.x HOWTO and Reference Guide*. Verkregen van https://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/diagnosis.html

Van Vreckem, B. (2014). Een fileserver opzetten met Samba. Verkregen 23 september 2016, van https://youtu.be/w2RxBkqQ3ZQ

Van Vreckem, B. (2015a, oktober 18). Linux Troubleshooting (deel 1). Verkregen 26 september 2017, van https://youtu.be/ciXpmDwJKOM

Van Vreckem, B. (2015b). Troubleshooting a network service. Verkregen 15 september 2016, van https://github.com/bertvv/cheat-sheets/blob/master/print/NetworkTroubleshooting.pdf

Van Vreckem, B. (2015c, september 29). VirtualBox Networking: an overview (Notes to Self Blog). Verkregen 17 september 2016, van https://bertvv.github.io/notes-to-self/2015/09/29/virtualbox-networking-an-overview/

The Official Samba 3.5.x HOWTO and Reference Guide. (2010). Verkregen 23 september 2016, van https://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/

Weissig, J. (2014, december 16). Crash Course on Vagrant (revised): Sysadmin Cast, episode 42. Verkregen 15 september 2016, van https://sysadmincasts.com/episodes/42-crash-course-on-vagrant-revised

Weissig, J. (2015, januari 13). 19 Minutes With Ansible: Sysadmin Cast, episode 43. Verkregen 15 september 2016, van https://sysadmincasts.com/episodes/43-19-minutes-with-ansible-part-1-4