



## Technisch rapport

### Team 4

Brent Van de Reyd	r0879835	2CCS02
Mathias Wouters	r0879842	2CCS02
Thibaud Wagemans	r0801881	2CCS02
Samir El-Hajjoui	r0897980	2CCS02
Cisse Mertens	r0883623	2CCS02
Pauline Valgaeren	r0781850	2CCS01

**Project Hosting**  
**Bram Verbruggen**  
**Filippo Bagnoli**  
**Laurien Stroobants**

Academiejaar 2022-2023

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel



## **VOORWOORD**

Met trots presenteren wij ons Hostingsproject van het 2e jaar ITF aan Thomas More. Dit project is een resultaat van onze studiekeuze Cloud & Cyber Security. Als IT-studenten zijn we ons bewust van het belang van het veilig opslaan en beheren van data en informatie. Dit project biedt ons de mogelijkheid om onze vaardigheden op dit gebied te ontwikkelen en te versterken.

Wat ons zo interesseert aan dit project is dat het ons in staat stelt om praktijkervaring op te doen in een real-life scenario. We zullen niet alleen leren over verschillende hostingtechnologieën, maar ook over de beveiliging van deze technologieën. Dit project zal ons uitdagen om ons kritisch denken en onze probleemoplossende vaardigheden te gebruiken om de beste oplossing voor onze klanten te vinden.

Ten slotte willen we graag ons dankwoord uitspreken aan iedereen die heeft bijgedragen aan dit project en dit document. We willen onze docenten bedanken voor hun begeleiding en ondersteuning en ons team voor hun toewijding en inzet.

We hopen dat u net zo enthousiast bent over dit project als wij en we kijken uit naar uw feedback.

# INHOUDSTAFEL

<b>VOORWOORD .....</b>	<b>2</b>
<b>INHOUDSTAFEL .....</b>	<b>3</b>
<b>OVERZICHT TABELLEN &amp; FIGUREN .....</b>	<b>5</b>
<b>INLEIDING .....</b>	<b>7</b>
<b>1      TECHNISCHE BESCHRIJVING .....</b>	<b>8</b>
<b>1.1      Beschrijving startsituatie .....</b>	<b>8</b>
<b>1.2      Schematische voorstelling systeem .....</b>	<b>8</b>
<b>1.3      Beschrijving alle onderdelen .....</b>	<b>9</b>
1.3.1      Operation report cards.....	9
<b>1.4      Nodige software &amp; technologieën .....</b>	<b>12</b>
1.4.1      Secure File Transfer Protocol .....	12
1.4.2      vSphere .....	12
1.4.3      Kubernetes .....	12
1.4.4      Application Programming Interface.....	13
1.4.5      Persistent shared storage.....	13
1.4.6      Network File System.....	13
1.4.7      Disaster recovery .....	14
<b>2      HANDLEIDING VOOR DE MEDE DEVELOPERS .....</b>	<b>15</b>
<b>2.1      Virtuele machine .....</b>	<b>15</b>
2.1.1      Playbooks .....	15
2.1.2      Deployment.....	18
<b>2.2      Security.....</b>	<b>27</b>
2.2.1      OpenSCAP.....	27
<b>2.3      API.....</b>	<b>32</b>
2.3.1      Systemd .....	32
2.3.2      API gebruiken .....	33
2.3.3      Main.py .....	33
2.3.4      Crud.py .....	34
2.3.5      Models.py .....	37
2.3.6      Schemas.py .....	38
2.3.7      Database.py .....	39
2.3.8      Auth.py .....	41
2.3.9      Reset_database .....	42
<b>2.4      NFS Shared Map Setup .....</b>	<b>44</b>
2.4.1      Server Configuratie .....	44
2.4.2      Client Configuratie .....	44
2.4.3      Automount van de NFS share: .....	44
<b>2.5      Duplicatie back-ups.....</b>	<b>46</b>
2.5.1      Installatie op VMs .....	46
2.5.2      Back-ups maken in webinterface.....	48
<b>2.6      Monitoring .....</b>	<b>51</b>
2.6.1      Zabbix-monitoring tool .....	51
2.6.2      Zabbix installeren .....	51
<b>3      HANDLEIDING VOOR DE KLANTEN .....</b>	<b>61</b>
<b>3.1      API.....</b>	<b>61</b>
3.1.1      Gebruiker aanmaken .....	61
3.1.2      Gebruiker valideren .....	61

3.1.3	Gebruiker bijwerken .....	61
3.1.4	Gebruiker verwijderen .....	61
3.1.5	Bestanden opladen.....	61
3.1.6	Bestanden bijwerken .....	61
3.1.7	Bestanden verwijderen .....	62
3.1.8	Project deployen .....	62
<b>4</b>	<b>TROUBLESHOOTING .....</b>	<b>63</b>
<b>4.1</b>	<b>Zabbix .....</b>	<b>63</b>
<b>4.2</b>	<b>Disaster recovery plan .....</b>	<b>63</b>
4.2.1	Stap 1: .....	63
4.2.2	Stap 2: .....	63
4.2.3	Stap 3: .....	63
4.2.4	Stap 4: .....	64
4.2.5	Stap 5: .....	64
<b>5</b>	<b>GEBRUIKTE TOOLS .....</b>	<b>65</b>
<b>5.1</b>	<b>Zabbix .....</b>	<b>65</b>
<b>5.2</b>	<b>OpenVPN.....</b>	<b>66</b>
<b>5.3</b>	<b>Uncomplicated Firewall.....</b>	<b>66</b>
<b>5.4</b>	<b>Ansible .....</b>	<b>66</b>
<b>5.5</b>	<b>Poste.io .....</b>	<b>67</b>
<b>5.6</b>	<b>PhpMyAdmin .....</b>	<b>67</b>
<b>5.7</b>	<b>Discord.....</b>	<b>67</b>
<b>5.8</b>	<b>Duplicati.....</b>	<b>67</b>
<b>5.9</b>	<b>SCAP Workbench.....</b>	<b>67</b>
	<b>BESLUIT .....</b>	<b>69</b>
	<b>BIBLIOGRAFIE .....</b>	<b>70</b>
	<b>BIJLAGEN .....</b>	<b>72</b>
	<b>VERKLARENDE WOORDENLIJST .....</b>	<b>73</b>

## OVERZICHT TABELLEN & FIGUREN

Schematische voorstelling van het systeem .....	8
WRM Antimalware .....	11
WRM Transfer Methode .....	12
WRM Containerbeheer .....	12
Test-make-cluster.yaml .....	15
Usermod 16 .....	
Test-join-cluster.yaml .....	17
Worker nodes .....	17
Storageclass.yaml .....	18
Namespace.yaml .....	18
Deployment-laravel-PV.yaml .....	19
Deployment-mysql-PV.yaml .....	21
Deployment-pv.yaml .....	22
Deployment-mysql-PV.yaml .....	24
Deploy.sh 25 .....	
Status check .....	25
Status check 2 .....	26
Status check 3 .....	26
Website klant .....	26
PhpMyAdmin login .....	26
Logo OpenSCAP .....	27
Installatie SCAP .....	27
Uitvoeren van de scan SCAP .....	28
Kies bestand SCAP .....	28
Guide to the secure configuration of ubuntu 22.04 .....	29
Scan knop SCAP .....	29
Wachtwoord ingeven SCAP .....	29
Askpass SCAP .....	29
Askpass wachtwoord SCAP .....	29
SCAP scan bezig .....	30
SCAP scan klaar .....	30
SCAP scan resultaat .....	30
Back-up toevoegen Duplicati .....	48
Nieuwe back-up Duplicati .....	48
Algemene back-upinstellingen Duplicati .....	48
Back-updoel Duplicati .....	49
Connectie testen Duplicati .....	49
Back-up uitvoeren Duplicati .....	50
Back-ups bekijken in onedrive Duplicati .....	50
Zabbix logo .....	51
Connectie zabbix .....	53
Tijdzone selectie zabbix .....	53
Inloggen zabbix .....	54
Dashboard properties zabbix .....	54
Dashboard zabbix .....	54
Pagina toevoegen zabbix .....	55
Widgets zabbix .....	55
Diagrammen kiezen zabbix .....	56
Selecteren zabbix .....	56
Host zabbix .....	56
Gekozen host zabbix .....	57
Aanpassen grafiek zabbix .....	57
Monitoren zabbix .....	57

Volgende grafiek zabbix .....	58
Legenda aanpassen zabbix .....	58
Kleur aanpassen zabbix .....	59
Add grafiek zabbix.....	59
Kijken door pagina's zabbix .....	60
WRM Monitoring Tools .....	65
WRM Firewalls .....	66
Schematische voorstelling van het systeem uitvergroot .....	72

# **INLEIDING**

In het kader van dit project was er een uitdaging om een hostingplatform te ontwikkelen waarbij in het begin nog niet helemaal duidelijk was welke functies er nodig waren. Om tot een duidelijker beeld te komen, is er gebruikgemaakt van feedback van medestudenten.

Het doel van het hostingplatform is om veiligheid te garanderen bij het transporteren van bestanden en het hebben van een database die backups kan maken. Er worden verschillende tools gebruikt, waaronder ansible, kubernetes, virtuele machines, apache, mailserver, php myadmin en mysql. Het platform maakt gebruik van een API en heeft geen frontpagina.

De technische aspecten van het hostingplatform, inclusief de gebruikte tools en veiligheidsmaatregelen, worden beschreven in hoofdstuk 1 van de documentatie. In hoofdstuk 2 wordt een handleiding aangeboden voor het gebruik van het platform, zodat gebruikers kunnen leren hoe ze hun website of applicatie kunnen hosten. Hoofdstuk 3 biedt een troubleshooting gedeelte aan waarin veelvoorkomende problemen worden besproken en hoe deze op te lossen zijn. In hoofdstuk 4 worden de verschillende tools besproken en hoe deze bijdragen aan het platform.

De documentatie biedt een goed beeld van het hostingplatform en de mogelijkheden die het biedt aan gebruikers. Het is relevant gezien de groeiende vraag naar veilige en betrouwbare hostingplatform.

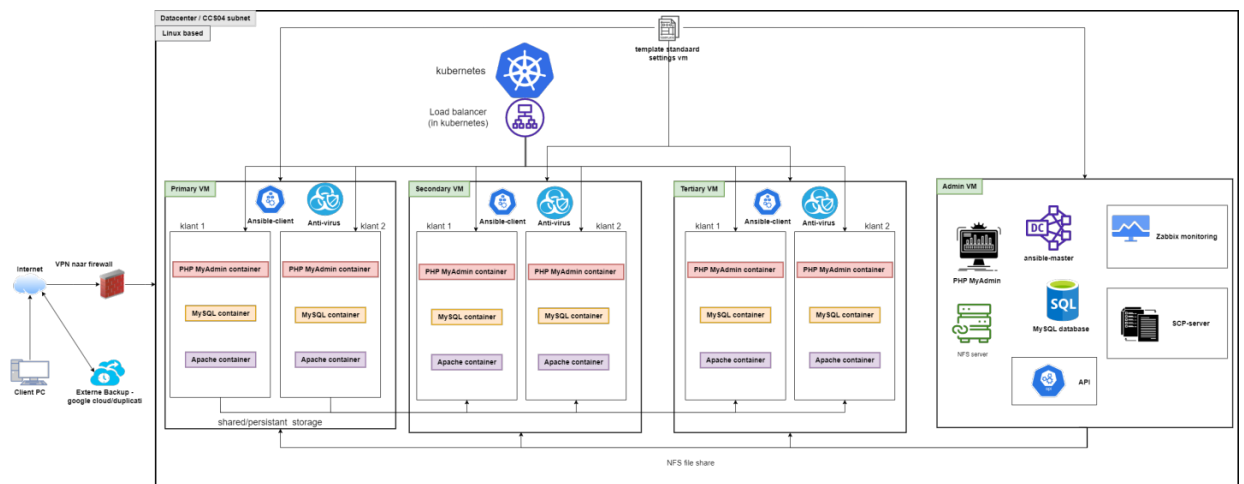


# 1 TECHNISCHE BESCHRIJVING

## 1.1 Beschrijving startsituatie

Er is gestart door met de gegeven informatie op zoek te gaan naar verschillende oplossingen voor hetzelfde probleem. Dit is gedaan door informatie te verkrijgen van derdejaars studenten en door onderzoek te doen op het internet. Elke oplossing is afgewogen en beoordeeld om te zien welke optie het beste gebruikt kon worden. Als groep waren we het hierover meteen eens, waardoor de rest van het startproces soepel verliep. Bij sommige onderdelen was er geen andere optie dan degene die in onze documentatie verder wordt beschreven.

## 1.2 Schematische voorstelling systeem



*Schematische voorstelling van het systeem*

## **1.3 Beschrijving alle onderdelen**

### **1.3.1 Operation report cards**

Een operation report card is een evaluatie-instrument om het succes van operationele processen binnen een organisatie te meten. Het is een nuttige tool om de prestaties van een organisatie te verbeteren door inzicht te krijgen in welke gebieden goed presteren en welke gebieden verbetering nodig hebben. Met een operation report card kunnen organisaties belangrijke KPI's identificeren en meten om hun processen te verbeteren, risico's te beperken en operationele efficiëntie te verbeteren. In de volgende hoofdstukken zullen we de verschillende aspecten van het operation report card-proces bespreken en laten zien hoe het kan worden geïmplementeerd om het succes van een organisatie te verbeteren. Hieronder worden de operation report cards vermeld die wij als groep hebben gekozen.

#### **1.3.1.1 Are "the 3 empowering policies" defined and published ?**

De drie empowerment beleidslijnen worden gedefinieerd en gepubliceerd in de mail die de klanten aankrijgen bij het aanmaken van een account.

#### **1.3.1.2 Do you have a password safe ?**

Het team heeft besloten om KeePass te gebruiken als wachtwoordbeheerprogramma. Voor deze toepassing wordt er een sterk hoofdwachtwoord gebruikt, dat alleen bekend is bij de ontwikkelaars en niet opgeslagen wordt op de systemen. Bovendien wordt KeePass lokaal gehost, waardoor het niet toegankelijk is van buitenaf. Dit verhoogt de veiligheid van de wachtwoordbeheerpraktijken en zorgt ervoor dat gevoelige informatie op een betrouwbare manier wordt opgeslagen en beschermd.

#### **1.3.1.3 Does each service have appropriate monitoring ?**

Het monitoren gaat gebeuren met de tool Zabbix.

#### **1.3.1.4 Do you use configuration management tools like cfengine/puppet/chef/ansible**

Er is besloten om Ansible te gebruiken als configuratie management tool voor het project. Deze keuze is gebaseerd op eerdere ervaringen met de software tijdens lessen in Linux Network Services, waardoor het makkelijker is om ermee aan de slag te gaan in vergelijking met andere alternatieven zoals Chef of Cfengine. Daarnaast is er veel online informatie beschikbaar over het gebruik van Ansible. DSC is ook overwogen als optie, maar bleek niet compatibel te zijn met de Linux-based VM's. Het gebruik van een configuratie management tool automatiseert het beheer en de configuratie van de VM's, wat het aanpassen van serverconfiguraties bijvoorbeeld eenvoudiger maakt. Dit zorgt voor een efficiënter gebruik van tijd en streeft naar een optimaal resultaat.

#### **1.3.1.5 Is OS installation automated ?**

Ja, de afhandeling van de images op de webserver verloopt automatisch via kubernetes.

#### **1.3.1.6 Can you automatically patch software across your entire fleet ?**

Om dit te bereiken, zal de configuratie management tool Ansible opnieuw worden ingezet. Deze keuze is gebaseerd op de mogelijkheid om verschillende taken uit te voeren met behulp van één tool. Hoewel er alternatieven beschikbaar zijn die gelijkwaardig zijn, is Ansible gekozen vanwege de reeds bestaande ervaring en vertrouwde met de software.

#### **1.3.1.7 Can your servers keep operating even if 1 disk dies ?**

Om te zorgen dat de server blijft draaien in geval van een schijfstoring, wordt de server geconfigureerd met een redundante opslagoplossing, zoals RAID. RAID combineert meerdere harde schijven om betrouwbaarheid, prestaties en/of capaciteit te verbeteren. Verschillende RAID-niveaus zijn beschikbaar, waarbij RAID 5 en RAID 6 vaak worden gebruikt voor servers die een hoge mate van redundantie vereisen. Naast het implementeren van RAID is het belangrijk om regelmatig back-ups te maken van alle gegevens op de server. Om een RAID-implementatie uit te voeren, zijn enkele belangrijke stappen vereist, zoals het kiezen van een RAID-niveau, het selecteren van de harde schijven, het installeren van de harde schijven, het configureren van de RAID-controller en het testen en bewaken van de RAID-implementatie. De RAID zal worden opgezet door het datacenter waarop het hostingsplatform draait (Gillis A., 2021).

#### **1.3.1.8 Are your back-ups automated ?**

Ja, de back-up scripts zorgen voor de automatische lokale en externe back-ups en kunnen indien nodig handmatig worden geactiveerd (Fisher T. , 2023).

#### **1.3.1.9 Do machines in your data center have remote power / console access**

Alle virtuele machines worden geconfigureerd aan de hand van een "remote access" SSH connectie. Deze kan geopend worden door het gebruik van VMware software, er is ook de mogelijkheid om via een VPN tunnel connectie te maken met de web browser. Eens ingelogd kan er connectie met de machines gemaakt worden.

### 1.3.1.10 Do desktops, laptops, and servers run self-updating, silent, anti-malware software ?

WRM Antimalware

Antimalware Cloud gebaseerde Systemen	Bescherming (40%)	Prestaties (25%)	Gebruiksgemak (25%)	Prijs (10%)	Totaal Score
McAfee Endpoint Security	8	7	7	6	7.4
ClamAv	9	8	8	9	8.6
Sophos Intercept X	9	7	8	8	8.1
Trend Micro Cloud One	8	9	7	7	7.9

ClamAV is gekozen vanwege de beste prestaties in de WRM. Er zijn verschillende redenen waarom ClamAV kan worden gebruikt. Allereerst is het een gratis en opensource programma, waardoor het voor iedereen beschikbaar is. Dit is met name handig voor kleine bedrijven of individuen die geen budget hebben voor dure antivirussoftware. Ten tweede biedt ClamAV real-time scanning, waardoor bestanden automatisch worden gescand zodra ze worden gedownload of geopend. Dit verlaagt het risico op infecties en beschermt tegen potentiële schade (ClamAVNet, n.d.).

### 1.3.1.11 Do you have a written security policy ?

Het is belangrijk om een beveiligingsbeleid te hebben om de website en de gegevens ervan te beschermen. Hier zullen verschillende punten in opgesomd worden, denk aan data encryptie, monitoring, incident response, access controls.

### 1.3.1.12 Can a user's account be disabled on all systems in 1 hour ?

De opslag van alle gebruikers in één database maakt snelle deactivering mogelijk wanneer dat nodig is.

### 1.3.1.13 Can you change all privileged (root) passwords in 1 hour ?

Om alle wachtwoorden binnen een uur te kunnen veranderen, zal er een soort bestand/database worden bijgehouden waar het root wachtwoord is gebruikt.

## 1.4 Nodige software & technologieën

### 1.4.1 Secure File Transfer Protocol

WRM Transfer Methode

Transfer Methode	Snelheid (40%)	Betrouwbaarheid (25%)	Gebruiksgemak (25%)	Beveiliging (10%)	Totaal Score
SCP	8	9	7	10	8.25
SFTP	7	9	8	9	8.5
FTP	5	6	6	7	6
HTTP	4	7	9	5	6.25

"Secure File Transfer Protocol", is een beveiligd overdrachtsprotocol voor bestanden. Het is een uitbreiding van het standaard File Transfer Protocol (FTP) en biedt verbeterde beveiliging tijdens het overzetten van bestanden.

In tegenstelling tot FTP, dat werkt met onversleutelde communicatie, maakt SFTP gebruik van een beveiligde SSH-verbinding om bestanden veilig over te dragen tussen een lokale machine en een externe server. Dit betekent dat de gegevens die worden verzonden en ontvangen via SFTP versleuteld zijn, waardoor ze niet kunnen worden onderschept of gelezen door onbevoegden. (Yusuf, 2023)

### 1.4.2 vSphere

VMware vSphere is een virtualisatieplatform voor het creëren en beheren van gevirtualiseerde omgevingen. Het platform maakt gebruik van hypervisor-technologie om fysieke servers in meerdere virtuele machines om te zetten. Hierdoor kunnen meerdere besturingssystemen en applicaties op één fysieke server worden uitgevoerd, wat de efficiëntie van het servergebruik vergroot en de kosten verlaagt. VMware vSphere biedt ook functies voor resourcebeheer, automatisering, beveiliging en back-up van virtuele machines. Het wordt veel gebruikt in datacenters en cloudomgevingen om infrastructuur te consolideren en te beheren (VMware vSphere | Enterprise Workload Platform, 2023).

### 1.4.3 Kubernetes

WRM Containerbeheer

Containerbeheer	Schaalbaarheid (40%)	Veiligheid (25%)	Gebruiksgemak (25%)	Beschikbaarheid (10%)	Totaal Score
Kubernetes	9	7	6	8	7,5
Docker	5	7	9	7	7

In de WRM is te zien dat kubernetes een hogere score behaalt dan docker.

Kubernetes (K8s) is een open-source platform voor het automatiseren van containerbeheer en het beheren van gedistribueerde toepassingen op schaal. Het stelt gebruikers in staat om containergebaseerde toepassingen te implementeren, schalen en beheren, ongeacht de infrastructuur waarop ze

worden uitgevoerd. Met Kubernetes kunnen gebruikers gemakkelijk containers orkestreren en automatisch schalen, terwijl ze tegelijkertijd hoge beschikbaarheid, failover en service discovery mogelijk maken. Hierdoor kunnen ontwikkelaars en systeembeheerders complexe toepassingen in een cloudomgeving beheren met minder complexiteit en handmatig werk (Production-Grade Container Orchestration, n.d.-b).

#### **1.4.4 Application Programming Interface**

Een Application Programming Interface (API) is een set van definities, protocollen en hulpmiddelen voor het bouwen van softwaretoepassingen. Het biedt een gestandaardiseerde manier voor verschillende softwaretoepassingen om met elkaar te communiceren en informatie uit te wisselen. Een API definieert de toegangspunten voor de interactie tussen verschillende softwarecomponenten en bepaalt hoe deze interactie plaatsvindt, inclusief de syntaxis en structuur van gegevensuitwisseling. Het gebruik van API's maakt het mogelijk om verschillende softwaretoepassingen te integreren, waardoor een meer geavanceerde functionaliteit ontstaat die anders niet mogelijk zou zijn. API's worden veel gebruikt in web ontwikkeling, mobiele applicaties en Cloud computing. Deze API gebruiken wij voor het inloggen en aansturen van ons web hostings platform (Wat is een API? (n.d.).).

#### **1.4.5 Persistent shared storage**

Een persistent shared storage (PSS) is een type opslagoplossing waarbij meerdere servers toegang hebben tot dezelfde opslagruimte, waardoor gegevens en bestanden kunnen worden gedeeld tussen verschillende applicaties en systemen. Dit wordt gedaan via een netwerkverbinding die toegang biedt tot een centrale opslaglocatie, zoals een netwerkgebonden opslag (NAS) of een storage area network (SAN).

In hostingprojecten wordt PSS vaak gebruikt om een schaalbare en redundante opslagoplossing te bieden voor websites en applicaties die op meerdere servers draaien. Hierdoor kunnen gegevens en bestanden centraal worden opgeslagen en kunnen meerdere servers deze opslagruimte delen, wat de prestaties en betrouwbaarheid van de hostingomgeving verbetert.

Bij het gebruik van PSS is het belangrijk om te zorgen voor een goede configuratie van het netwerk en de opslagapparatuur, om ervoor te zorgen dat de prestaties en betrouwbaarheid van de hostingomgeving optimaal zijn. Ook moet er rekening worden gehouden met de beveiliging van de opgeslagen gegevens en moet er een backup- en herstelplan worden opgesteld om ervoor te zorgen dat de gegevens te allen tijde veilig zijn NetApp. (2019).

#### **1.4.6 Network File System**

Network File System (NFS) is een netwerkprotocol voor het gedistribueerd delen van bestanden. Een bestandssysteem definieert hoe gegevens worden opgeslagen in bestanden en worden opgehaald van opslagapparaten zoals harde schijven, solid-state drives en tapedrives. NFS is een netwerkprotocol voor het delen van bestanden dat definieert hoe bestanden worden opgeslagen en opgehaald van opslagapparaten op het netwerk. (Loshin, 2022)

#### 1.4.7 Disaster recovery

Disaster recovery is het proces van het herstellen van systemen en gegevens na een grote storing, zoals een natuurramp, een hardwarestoring of een cyberaanval. Voor een hostingsplatform is het van cruciaal belang om een disaster recovery-methodologie te hebben om ervoor te zorgen dat klanten snel en efficiënt kunnen worden geholpen in geval van een storing.

De disaster recovery-methodologie voor het hostingsplatform zal als volgt worden geschreven:

*Evaluatie:* Identificeer potentiële risico's voor ons hostingsplatform en beoordeel de impact van elke risicofactor op ons systeem en de klanten. Prioriteer de herstelactiviteiten op basis van de impact en de urgentie.

*Planning:* Ontwikkel een gedetailleerd plan voor disaster recovery met een focus op het minimaliseren van de downtime en het herstel van gegevens en systemen. Documenteer procedures en protocollen voor het herstelproces, inclusief communicatie met klanten.

*Back-up:* Maak regelmatig back-ups van alle belangrijke gegevens en zorg ervoor dat deze op een externe locatie worden opgeslagen om toegankelijkheid te garanderen. Test de back-ups regelmatig om de integriteit van de gegevens te waarborgen.

*Testen:* Test regelmatig de disaster recovery-procedures om te zorgen voor betrouwbare en tijdige reacties in geval van een storing. Train personeel in de procedures en protocollen om te zorgen voor een snelle en efficiënte reactie.

*Implementatie:* Implementeer de disaster recovery-procedures in de organisatie en zorg ervoor dat de betrokken medewerkers getraind en bekend zijn met de procedures. Blijf het proces regelmatig evalueren en bijwerken om te zorgen voor optimale prestaties.

## 2 HANDLEIDING VOOR DE MEDE DEVELOPERS

### 2.1 Virtuele machine

#### 2.1.1 Playbooks

Voor het aanmaken van onze microk8s cluster gebruiken wij Ansible playbooks die van op de admin vm worden gestart. Deze worden dan op al onze nodes (die in de cluster moeten staan) uitgevoerd. Deze playbooks installeren en configureren de cluster zo zodat de API onze deployments kan uitvoeren en alles opzetten. De playbooks installeren dus docker en microk8s en zorgen dat deze opstarten bij het booten van de servers/nodes. Hierna zorgen de playbook ervoor dat de worker nodes de cluster joinen en zo er dus voor zorgen dat onze cluster (momenteel) 3 nodes bevat. Dit is schaalbaar omdat er in de playbooks dynamisch word gekeken hoeveel hosts er zijn en op elke host dit uitvoerd.

##### 2.1.1.1 Playbook make-cluster

In deze playbook gaan we op alle hosts de installatie van docker en microk8s doen. Eerst zorg ik dat alle dependencies en packages geïnstalleerd. Daarna voeg ik de docker repository toe zodat we deze erna kunnen installeren. Meteen hierna installeren we microk8s en zorgen we dat docker en microk8s opstarten zodra dat de servers opstarten. En als laatste zorgen we dat docker en microk8s opgestart worden en draaien op alle hosts.

```
test-make-cluster.yml
1 ---
2 - name: Install Docker and MicroK8s
3   hosts: servers
4   remote_user: admin-ccs04
5   become: true
6
7   tasks:
8     - name: Update package lists
9       apt:
10         update_cache: yes
11
12     - name: Configure dpkg
13       become: true
14       shell: sudo dpkg --configure -a
15
16     - name: Install dependencies
17       apt:
18         name: ['apt-transport-https', 'ca-certificates', 'curl', 'gnupg-agent', 'software-properties-common']
19         state: present
20
21     - name: Add Docker GPG key
22       apt_key:
23         url: https://download.docker.com/linux/ubuntu/gpg
24         state: present
25
26     - name: Add Docker repository
27       apt_repository:
28         repo: "deb [arch=amd64] https://download.docker.com/linux/ubuntu {{ ansible_lsb.codename }} stable"
29         state: present
30
31     - name: Install Docker
32       apt:
33         name: docker-ce
34         state: present
35
```

*Test-make-cluster.yml*



Hier ziet u ook dat we voor microk8s een alias maken zodat we deze alias kunnen gebruiken om alles met microk8s uit te voeren. Verder ziet u ook dat we ervoor zorgen dat de juiste usermod word gedaan zodat de user de juiste rechten heeft om alles uit te voeren zonder extra permissies.

```
- name: Install MicroK8s
  shell: |
    curl -sSL https://get.docker.com/ | sh
    snap install microk8s --classic
    usermod -a -G microk8s {{ ansible_user }}
    echo "alias microk8s.kubectl='microk8s kubectl'" >> ~/.bashrc
    source ~/.bashrc
  args:
    executable: /bin/bash

- name: Enable Docker service
  systemd:
    name: docker
    enabled: true
    state: started

- name: Enable and start MicroK8s
  command: microk8s.start
```

*Usermod*

#### 2.1.1.2 Playbook join-cluster

In deze playbook zorgen we ervoor dat de worker nodes de master node joinen en zo een cluster vormen. Dit zorgt ervoor dat we alles van de admin server kunnen doen en niet op elke server moeten inloggen en het joint commando moeten uitvoeren. Verder enablen we ook nog enkele services zoals dns (binnen de cluster) en metallb (de loadbalancer die wij gebruiken binnen onze cluster). Bij het enablen van deze loadbalancer geven we ook meteen een IP-range mee die hij mag gebruiken, zodat hij publieke IP kan geven en zodat wij hier dan aan kunnen om het project te bekijken/te hosten.

In deze playbook ga ik in de ansible hosts alle beschikbare nodes ophalen en zet deze in een variabele. In de eerste variabele zetten we de master node (dit is de eerste node in de hosts file), in de 2<sup>de</sup> variabele zetten we alle worker nodes (dus vanaf host nummer 2). Dit is zodat we dingen enkel op de master kunnen uitvoeren zoals bv. Het aanmaken van een join token voor de worker nodes.

We gaan in deze playbook het join commando uitvoeren om een join token te krijgen van de master node om de worker nodes te joinen in de cluster. Daarna gaan we de output van dit commando in een variabele steken zodat we dit kunnen aanroepen op de worker nodes en ook kunnen uitvoeren hierop.

```

tes-join-cluster.yml
1 - name: Join MicroK8s nodes to the cluster
2   hosts: servers
3   remote_user: admin-ccs04
4   become: true
5   vars:
6     master_node: "{{ groups['all'][0] }}" # First server in the inventory is the master node
7     worker_nodes: "{{ groups['all'][2:] }}" # Remaining servers are worker nodes
8
9   tasks:
10    - name: "Create join key and read output"
11      command: microk8s add-node --token-ttl 999
12      register: add_output
13      when: inventory_hostname == master_node
14
15    - name: "Define the 'join_command' variable"
16      set_fact:
17        join_command: "{{ (add_output.stdout_lines | select('match', '.*172.26.104.1.*'))[0] }}"
18      when: inventory_hostname == master_node
19
20    - name: "Debug: Printing the join command"
21      debug:
22        var: join_command
23      when: inventory_hostname == master_node

```

### Test-join-cluster.yml

In dit deel van de playbook staat het deel waar het join commando word uitgevoerd op de worker nodes. Je zien ook dat onder elke uit te voeren taak word meegegeven welke node/nodes dit moeten uitvoeren. Dit word gedaan aan de hand van de parameter "when : " en dan de variabele met de node/nodes in om dit op uit te voeren.

```

- name: "Debug: Printing the join command"
  debug:
    var: join_command
  when: inventory_hostname == master_node

- name: "Add K8S join command to a dummy host"
  add_host:
    name: "K8S_JOIN HOLDER"
    command: "{{ join_command + ' --worker' }}"
  when: inventory_hostname == master_node

- name: "Execute the join command in a worker node, Debug: Printing the join command"
  debug:
    var: hostvars['K8S_JOIN HOLDER']['command']
  when: inventory_hostname != master_node and inventory_hostname in worker_nodes

- name: "Join the cluster from the generated join command"
  command: "{{ hostvars['K8S_JOIN HOLDER']['command'] }}"
  when: inventory_hostname != master_node and inventory_hostname in worker_nodes

- name: "enable metallb, dns and dashboard"
  add_host:
    name: "metallb_and_dashboard"
    command: microk8s enable dashboard dns metallb:172.26.104.20-172.26.104.25
  when: inventory_hostname == master_node

```

### Worker nodes

Als laatste staat er nog om metallb (de loadbalancer die wij gebruiken) en dns (dns binnen onze cluster) te eneblen. Dit ook weer enkel op de master node door middel van de variabelen en parameter die we eerder hebben besproken.

## 2.1.2 Deployment

Voor de deployment van de Laravel applicatie hebben we gebruik gemaakt van kubernetes. Kubernetes is een open-source platform voor container organisatie/orkestratie, dat betekent dat kubernetes de implementatie van gecontaineriseerde applicaties vereenvoudigt. Kubernetes zorgt ook voor het automatiseren van een deployment net zoals het ook zorgt voor het scalen en managen van gecontaineriseerde applicaties.

We hebben verschillende opties zorgvuldig overwogen om te gebruiken voor de deployment, waaronder Docker en Docker Compose. Na een grondige analyse kozen we echter voor Kubernetes als ons voorkeursplatform, omdat kubernetes heel schaalbaar is en ook een hoge beschikbaarheid geeft aan de klant. Ook zijn er veel opties om kubernetes te automatiseren zoals Ansible en scripts.

### 2.1.2.1 Deployment files

```
1  apiVersion: storage.k8s.io/v1
2  kind: StorageClass
3  metadata:
4    name: standard
5  provisioner: kubernetes.io/no-provisioner
```

*Storageclass.yaml*

Eerst ga ik de API Versie specificeren van de storage resource. Daarna geef ik aan dat deze deployment een StorageClass gaat aanmaken genaamd standard. Als laatste zeg ik nog dat ik geen provisioner meegeef, een provisioner voorziet opslag resources voor de persistent volumes. Dit doen we omdat we de opslag resources manueel voorzien.

```
1  apiVersion: v1
2  kind: Namespace
3  metadata:
4    name: project-1
```

*Namespace.yaml*

Hier begin ik ook eerst met de API versie te specificeren, dit doe ik in elke deployment dus het gaat gewoon overal v1 zijn. In deze deployment maak ik een Namespace genaamd project-1. Je kan een Namespace eigenlijk wat vergelijken met een virtuele cluster binnen in de kubernetes cluster.

Aan de namespace worden alle andere deployments gelinkt zodat all nodes bij het correcte project runnen. Dit maakt het ook makkelijk om een project te verwijderen en beter te filteren per project. Dus we gaan voor elk nieuw project dat wordt aangemaakt een nieuwe namespace moeten aanmaken.

```

3  apiVersion: apps/v1
4  kind: Deployment
5  metadata:
6    name: laravel-app
7    namespace: project-1
8  spec:
9    replicas: 2
10   selector:
11     matchLabels:
12       app: laravel-app
13   template:
14     metadata:
15       labels:
16         app: laravel-app
17     spec:
18       containers:
19         - name: laravel
20           image: bitnami/laravel
21           workingDir: /app
22           ports:
23             - containerPort: 8000
24           env:
25             - name: DB_HOST
26               value: mysql-ser
27             - name: DB_PORT
28               value: "3306"
29             - name: DB_DATABASE
30               value: testing
31             - name: DB_USERNAME
32               value: laravel
33             - name: DB_PASSWORD
34               value: password
35           volumeMounts:
36             - name: laravel-data
37               mountPath: /app
38       volumes:
39         - name: laravel-data
40           persistentVolumeClaim:
41             claimName: laravel-pvc
42
43   ---
44
45   apiVersion: v1
46   kind: Service
47   metadata:
48     name: laravel-ser
49     namespace: project-1
50   spec:
51     selector:
52       app: laravel-app
53     ports:
54       - protocol: TCP
55         port: 8000
56         targetPort: 8000
57     type: LoadBalancer

```

*Deployment-laravel-PV.yaml*

In deze deployment ga ik de laravel container opstarten zodat daarin de laravel applicatie in kan draaien. In het metadata gedeelte geef ik de deployment een naam en voeg ik hem toe aan de namespace project-1 die dat ik hiervoor heb aangemaakt. Ik run hiervan 2 replicas zodat we ook het loadbalancen kunnen laten zien omdat hij anders alles op één pod draait, vervolgens geef ik de pods de label laravel-app. Hierna ga ik een template maken die dat voor elke laravel pod dan gebruikt wordt. Hierin geef ik terug de labe laravel-app en geef ik vervolgens de container een naam en geef ik hem ook de docker image die dat hij moet gebruiken. We gebruiken de bitnami laravel image omdat deze image al meteen een apache webserver bevat waarop php en laravel is geïnstalleerd. Daarna specificeer ik de folder die dat de container moet deployen, dat is de /app folder. Om het project op een website te kunnen raadplegen moeten we de interne poort (30230) forwarden naar poort 8000 omdat dit de default poort is van laravel. Daarna maak ik al de environment variabelen aan die de container moet gebruiken. Deze heb ik allemaal moeten overnemen van de .env file en van de database.php file die dat in het laravel project staan. Omdat het project anders niet met de laravel en mysql container gaat communiceren.

Omdat we ook persistent volumes maken moet ik nog zeggen dat ik het aangemaakte volume, dat verwijst naar de laravel persistentvolumeclaim, moet mounten naar de /app folder zodat hier de bestanden van het project inkomen.

Om de pods te kunnen raadplegen buiten de cluster moet ik ook een service aanmaken. Hier moet ik weeral net zoals bij de deployment een naam geven en hem toevoegen aan de namespace. Vervolgens moet ik opgeven naar welke pods de service moet routeren en moet ik ook poort 8000 openzetten. Als laatste moet ik nog meegeven dat de service een load balancer moet gebruiken om buiten de cluster beschikbaar te zijn. Hiervoor gebruiken we MetalLB zoals al eerder verteld.

```

3  apiVersion: apps/v1
4  kind: Deployment
5  metadata:
6    name: mysql-db
7    namespace: project-1
8  spec:
9    selector:
10     matchLabels:
11       app: mysql-db
12    replicas: 1
13    template:
14     metadata:
15       labels:
16         app: mysql-db
17     spec:
18       containers:
19         - name: mysql
20           image: mysql:8.0
21           ports:
22             - containerPort: 3306
23           env:
24             - name: MYSQL_ROOT_PASSWORD
25               value: password
26           volumeMounts:
27             - name: mysql-data
28               mountPath: /var/lib/mysql
29       volumes:
30         - name: mysql-data
31           persistentVolumeClaim:
32             claimName: mysql-pvc
33
34  ---
35
36  apiVersion: v1
37  kind: Service
38  metadata:
39    name: mysql-ser
40    namespace: project-1
41  spec:
42    selector:
43      app: mysql-db
44    ports:
45      - protocol: TCP
46        port: 3306
47        targetPort: 3306

```

*Deployment-mysql-PV.yaml*

Voor de MySQL pods te deployen doe ik in het begin hetzelfde als bij laravel alleen geef ik toepasselijker namen mee. Ook gebruik ik maar 1 replica omdat het anders voor problemen kan zorgen dat ik meerde databases draai die eigenlijk naar dezelfde db file moeten schrijven. Daarna ga ik ook weer een template maken waarin ik de pods de mysql label geef. Daarna ga ik de specificaties van de MySQL containers geven, ik zeg in het kort gewoon dat hij de default docker image van MySQL moet gebruiken (versie 8.0) en ook dat hij op de default poort van MYSQL moet draaien, poort 3306. Daarna geef ik nog een wachtwoord mee dat gezet is in een variabele. Met dat wachtwoord moet je in de database inloggen, je kan als username gewoon root gebruiken. Ten slotte ga ik hier ook weer een persistent volume mounten naar de /var/lib/mysql folder.

Om de deployment weer buiten de cluster te kunnen raadplegen moet er weer een service aangemaakt worden, dit gebeurt op dezelfde manier als de laravel service. Alleen zijn de namen veranderd naar de juiste MySQL namen die dat we hebben gebruikt en ook moet default poort 3306 geforward worden. Zoals je kan zien gebruiken we hier geen loadbalancer omdat dit totaal niet nodig is want we hebben maar één replica draaien.

```

3  apiVersion: v1
4  kind: PersistentVolume
5  metadata:
6    name: laravel-pv
7    namespace: project-1
8  spec:
9    capacity:
10     storage: 2Gi
11    accessModes:
12     - ReadWriteMany
13    persistentVolumeReclaimPolicy: Retain
14    storageClassName: standard
15    hostPath:
16     path: /home/admin-ccs04/data/laravel
17
18  ---
19
20  apiVersion: v1
21  kind: PersistentVolume
22  metadata:
23    name: mysql-pv
24    namespace: project-1
25  spec:
26    capacity:
27     storage: 2Gi
28    accessModes:
29     - ReadWriteMany
30    persistentVolumeReclaimPolicy: Retain
31    storageClassName: mysql-volume
32    hostPath:
33     path: /home/admin-ccs04/data/mysql

```

```

37  apiVersion: v1
38  kind: PersistentVolumeClaim
39  metadata:
40    name: laravel-pvc
41    namespace: project-1
42  spec:
43    accessModes:
44     - ReadWriteMany
45    resources:
46     requests:
47      storage: 2Gi
48    volumeName: laravel-pv
49    storageClassName: standard
50
51  ---
52
53  apiVersion: v1
54  kind: PersistentVolumeClaim
55  metadata:
56    name: mysql-pvc
57    namespace: project-1
58  spec:
59    accessModes:
60     - ReadWriteMany
61    resources:
62     requests:
63      storage: 2Gi
64    volumeName: mysql-pv
65    storageClassName: mysql-volume

```

*Deployment-pv.yaml*

We zijn begonnen met eerst de persistent volumes aan te maken, we geven de persistent volumes voor zowel de laravel deployment als de MySQL deployment 2 GB. Dit doen we ook voor de persistent volume claims. Vervolgens heb ik de access mode moeten instellen voor de persistent volumes, hiervoor gebruiken we ReadWriteMany. Dit gebruiken we zodat de pods juist kunnen interacteren met de persistent volumes. De reclaim policy staat op retain, dat wil zeggen dat de data opgeslagen wordt zodat de laravel en MySQL pods met de correcte data terug kunnen worden gedeployd.

Vervolgens heb ik de persistent volume claims aangemaakt. Deze zijn nodig om een stuk opslag aan te vragen. Hier doe ik in grote delen hetzelfde als in het aanmaken van de persistent volumes, alleen moet ik ze nog linken aan de juiste persistent volume. Dit doen we door aan het attribuut volumeName de naam van de persistent volume te geven.

De laatste deployment is een phpMyAdmin container. Deze runnen we zodat de gebruiker de database makkelijk kan raadplegen. Hiervoor doen we ongeveer hetzelfde als bij de andere deployments, als image hebben we de default docker image van phpmyadmin gebruikt en deze runt op poort 80. Hier heb ik ook nog environment variabelen gemaakt waarmee phpmyadmin gaat connecteren met de database.



```

3  apiVersion: apps/v1
4  kind: Deployment
5  metadata:
6    name: phpmyadmin-deployment
7    namespace: project-1
8    labels:
9      app: phpmyadmin
10 spec:
11   replicas: 1
12   selector:
13     matchLabels:
14       app: phpmyadmin
15   template:
16     metadata:
17       labels:
18         app: phpmyadmin
19     spec:
20       containers:
21         - name: phpmyadmin
22           image: phpmyadmin/phpmyadmin:latest
23           ports:
24             - containerPort: 80
25           env:
26             - name: PMA_HOST
27               value: mysql-ser
28             - name: PMA_PORT
29               value: "3306"
30             - name: MYSQL_ROOT_PASSWORD
31               value: password
32           volumeMounts:
33             - name: sessions-volume
34               mountPath: /sessions
35       volumes:
36         - name: sessions-volume
37           emptyDir: {}
38
39 ---
40
41 apiVersion: v1
42 kind: Service
43 metadata:
44   name: phpmyadmin-service
45   namespace: project-1
46 spec:
47   selector:
48     app: phpmyadmin
49   ports:
50     - protocol: TCP
51       port: 80
52       targetPort: 80
53   type: LoadBalancer

```

*Deployment-mysql-PV.yaml*

```
#!/bin/bash

# Apply the namespace
microk8s kubectl apply -f /home/admin-ccs04/kubernetes/namespace.yaml
sleep 10

# Apply the persistent volume
microk8s kubectl apply -f /home/admin-ccs04/kubernetes/deployment-pv.yaml
sleep 10

# Apply the MySQL deployment
microk8s kubectl apply -f /home/admin-ccs04/kubernetes/deployment-mysql-PV.yaml
sleep 10

# Apply the Laravel deployment
microk8s kubectl apply -f /home/admin-ccs04/kubernetes/deployment-laravel-PV.yaml
sleep 10

# Apply the PHPMyAdmin deployment
microk8s kubectl apply -f /home/admin-ccs04/kubernetes/deployment-phpmyadmin.yaml

echo "-----"
echo "Deployment completed successfully!"
echo "-----"
```

*Deploy.sh*

Voor alles te deployen heb ik een scriptje gemaakt dat dit automatisch doet. Buiten de commando's in het script moet ik alleen nog de storageclass deployen, maar dat moet alleen maar gebeuren als de hele cluster weg is en deze terug wordt opgesteld. Dat is omdat dit niet aan een namespace hangt dus verwijder ik het ook niet. Ik heb na het deployen van elke pod een sleep van 10 seconden gezet omdat het anders kan gebeuren dat er een pod in de pending status blijft vast zitten. Het scriptje kan gewoon worden uitgevoerd door `./deploy.sh` uit te voeren.

### 2.1.2.2 Deployment status check

```
admin-ccs04@primary-vm:~/kubernetes$ microk8s kubectl get pods -o wide -n project-1
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE	READINESS	GATES
mysql-db-6dfff5d7c9-mh7vr	1/1	Running	0	8h	10.1.174.20	secondary-vm	<none>		<none>	
laravel-app-5c465f9b65-ltmbp	1/1	Running	0	8h	10.1.174.21	secondary-vm	<none>		<none>	
laravel-app-5c465f9b65-cm22f	1/1	Running	0	8h	10.1.67.207	primary-vm	<none>		<none>	
phpmyadmin-deployment-677bc56798-zdhgg	1/1	Running	0	8h	10.1.67.208	primary-vm	<none>		<none>	

```
admin-ccs04@primary-vm:~/kubernetes$
admin-ccs04@primary-vm:~/kubernetes$
```

*Status check*

Het eerste commando dat uitgevoerd moet worden is het bovenstaande. Dit laat zien of dat alle pods runnen en niet op pending blijven staan. Je kan ook zien op welke node elke pod draait, dat laat ook zien dat de loadbalancer juist werkt omdat hij de laravel pods bijvoorbeeld verspreid over twee nodes. Mocht er een node toch in pending blijven staan kan je eerst in de describe kijken om te zien welke informatie je hier over de pod te zien krijgt. Mocht daar niet echt een duidelijke error in staan kan je nog in de logs kijken, dit zegt meestal heel duidelijk wat de pod juist allemaal aan het doen is en of dat het ook daadwerkelijk succesvol is uitgevoerd.

```
admin-ccs04@primary-vm:~/kubernetes$ microk8s kubectl get svc -o wide -n project-1
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	SELECTOR
mysql-ser	ClusterIP	10.152.183.102	<none>	3306/TCP	8h	app=mysql-db
laravel-ser	LoadBalancer	10.152.183.147	172.26.104.20	8000:30230/TCP	8h	app=laravel-app
phpmyadmin-service	LoadBalancer	10.152.183.193	172.26.104.21	80:31075/TCP	7h59m	app=phpmyadmin

```
admin-ccs04@primary-vm:~/kubernetes$
```

### Status check 2

Vervolgens ga ik de services nakijken of dat alles goed is gedeployed en dat de pods beschikbaar zijn buiten de cluster. Doormiddel van de loadbalancer hebben we hiervoor ook externe IP's gekregen die dat we in de webbrowser met de

```
admin-ccs04@primary-vm:~/kubernetes$ microk8s kubectl get persistentvolume
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
laravel-pv	2Gi	RWX	Retain	Bound	project-1/laravel-pvc	standard		8h
mysql-pv	2Gi	RWX	Retain	Bound	project-1/mysql-pvc	mysql-volume		8h

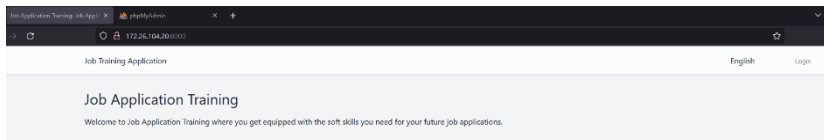
```
admin-ccs04@primary-vm:~/kubernetes$
```

### Status check 3

correcte poort kunnen bekijken.

Je kan ook de persistent volumes bekijken door bovenstaand commando. Dit zou je doen om na te kijken of dat de persistent volumes juist zijn gedeployed.

Als alles goed is gelukt kan je op de IP die dat de loadbalancer je heeft gegeven het project raadplegen. Hieronder staat een voorbeeld project dat we runnen.



### Website klant

Als laatste hebben we ook nog phpMyAdmin draaien. Dit kunnen we raadplegen door de IP van de loadbalancer op te zoeken. Deze service draait op poort 80, dit hebben we gedaan omdat phpMyAdmin toch een andere IP krijgt dus maakt het niet echt uit welke poort we gebruiken want deze gaat geen conflicten geven. Als username moet je "root" gebruiken en het password hebben we in de deployment file aangemaakt en dat is "password".

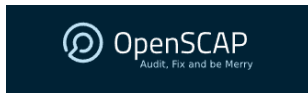


### PhpMyAdmin login

## 2.2 Security

### 2.2.1 OpenSCAP

Om de Virtual Machine te testen op insecurities gebruiken we een Cis compliance check uitgevoerd door OpenSCAP. Dit is een programma dat een SSH connectie maakt op de VM om zo de CIS test uit te voeren. Om een OpenSCAP test uit te voeren is er een installatie op de VM nodig en een systeem waar de SCAP workbench tool op staat om remote access uit te voeren. ons doel is om boven de 80% op de score te komen.



Logo OpenSCAP

#### 2.2.1.1 Installatie Openscap

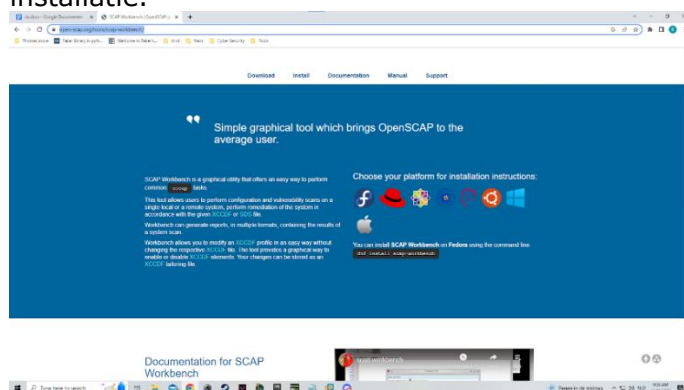
Om Openscap te installeren op de Virtual Machine voer je het volgende commando uit:

```
apt-get install libopenscap8
```

#### 2.2.1.2 Install SCAP workbench

SCAP workbench kan geïnstalleerd worden via <https://www.open-scap.org/tools/scap-workbench/>. Kies de software voor het Operating System dat u gebruikt.

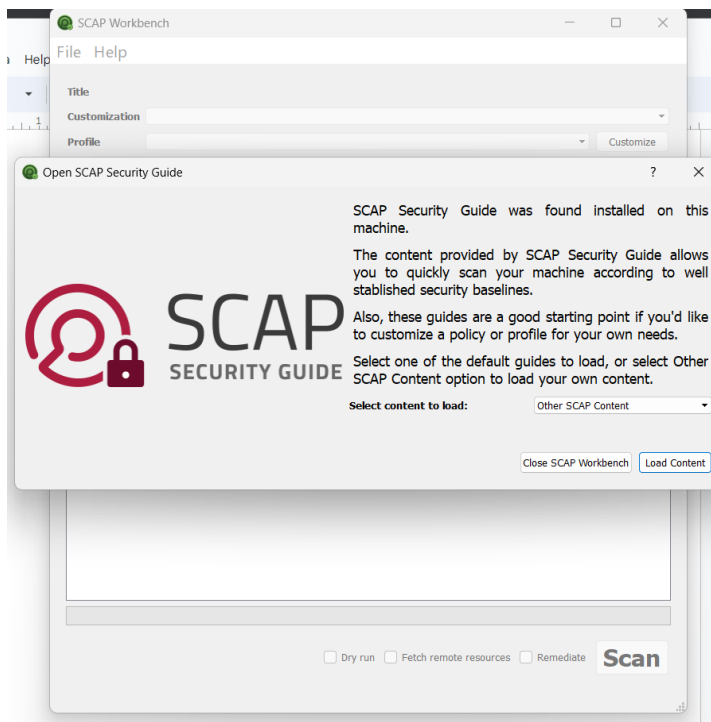
Daarna klikt u dubbel klik links op het windows icoontje en dan begint de installatie.



Installatie SCAP

#### 2.2.1.3 Scan uitvoeren

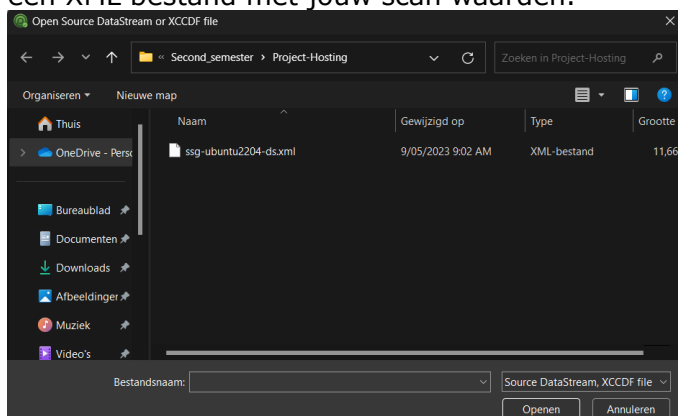
bij het openen van de workbench tool wordt er gevraagd naar het operating system van de Virtual Machine dat je gaat scannen. Omdat wij Ubuntu gebruiken kiezen we hier Other SCAP Content.



### *Uitvoeren van de scan SCAP*

Daarna klik je op Load Content.

Hierna moet je een bestand kiezen met de scan die je wilt uitvoeren, kies hier een XML bestand met jouw scan waarden.



### *Kies bestand SCAP*

Selecteer het systeem dat je wilt scannen door bij 'User and host' het volgende in te geven:

"username@ip-address". Zorg ervoor dat SSH ingeschakeld is, anders kan de tool geen verbinding maken met de machine.

**Title** Guide to the Secure Configuration of Ubuntu 22.04

**Customization** None selected

**Profile** CIS Ubuntu 22.04 Level 1 Server Benchmark (284) Customize

**Target** ☐ Local Machine ☒ Remote Machine (over SSH)

**User and host** admin-ccs04@172.26.192.142 **Port** 22 admin-ccs04@172.26.192.142:22

*Guide to the secure configuration of ubuntu 22.04*

Daarna druk je op 'Scan' om de scan te starten.

0% (0 results, 284 rules selected)

☐ Dry run ☐ Fetch remote resources ☐ Remediate **Scan**

*Scan knop SCAP*

Als je de scan start wordt er gevraagd om je wachtwoord in te voeren.

win-ssh-askpass

The authenticity of host '172.26.192.142 (172.26.192.142)' c

.....

OK Cancel

*Wachtwoord ingeven SCAP*

Daarna om "Yes or No" te typen om te bevestigen om de scan te starten.

win-ssh-askpass

Please type 'yes' or 'no':

...

OK Cancel

*Askpass SCAP*

Hierna wordt je nogmaals gevraagd om het wachtwoord in te voeren. Hierna start de scan.

win-ssh-askpass

admin-ccs04@172.26.192.142's password:

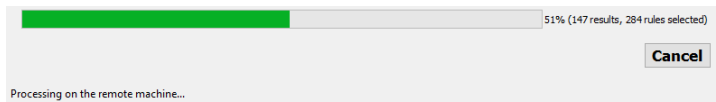
.....

OK Cancel

*Askpass wachtwoord SCAP*

### 2.2.1.4 Resultaten

De scan begint na het ingeven van het wachtwoord van de machine. Hierna begint de Workbench Tool connectie te maken en de tests uit te voeren, dit kan een tijdje duren.



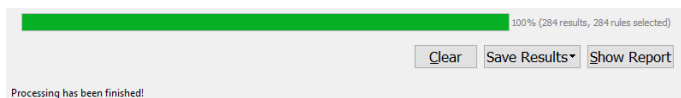
#### SCAP scan bezig

Wanneer de scan klaar is krijg je meerdere opties om uit te kiezen.

'Clear' om alles te verwijderen.

'Save results' om het verslag op te slaan.

'Show Report' om het verslag te bekijken.

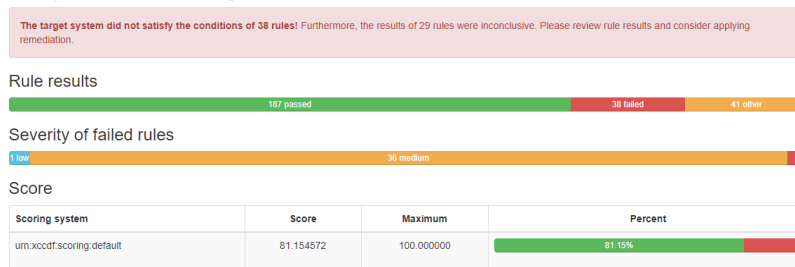


#### SCAP scan klaar

Wanneer we op 'Show Report' klikken krijgen we een html pagina met de resultaten.

Hier zien we de uiteindelijke score, in dit geval 81%.

#### Compliance and Scoring



#### SCAP scan resultaat

### 2.2.1.5 Commando's

Om de score 80% + te behalen hebben wij volgende commando's uitgevoerd.

```
sudo apt-get install aide -y
sudo apt-get remove rsync -y
sudo chown root:root /var/log/audit/audit.log
sudo chmod 640 /var/log/audit/audit.log
sudo chmod 0750 /var/log/audit/audit.log
sudo chown root:root /var/log/audit/audit.log
sudo chmod 0640 /var/log/audit/audit.log
sudo chown root:root /var/log/audit/audit.log
apt-get install iptables-persistent
apt-get remove iptables-persistent
sudo nano /etc/motd (DO NOT ACCESS WITHOUT PERMISSION!)
```

Hiernaast voeren we ook nog een script uit. Dit script komt van een Github repository, lees deze pagina eerst even door om te begrijpen wat er juist gebeurt.

<https://github.com/konstruktoid/hardening>

Je kan dit script uitvoeren volgens deze stappen:

1. `sudo apt-get update`
2. `sudo apt-get upgrade`
3. `sudo apt-get -y install git net-tools procps --no-install-recommends`
4. `git clone github.com/konstruktoid/hardening.git`
5. `sudo nano hardening/ubuntu.cfg`
6. verander in deze file het volgende
7. `CHANGEME=""` → `CHANGEME="ccs04"` in ons geval
8. `sudo bash ubuntu.sh`

Hierna wordt het script uitgevoerd, dit kan een tijdje duren.

Als je hierna nogmaals een scan uitvoert op de VM zal deze aanzienlijk gestegen zijn in percentage, net zoals de VM een stuk veiliger en meer beschermd zal zijn.



## 2.3 API

### 2.3.1 Systemd

Dit is een service die onze API draait met behulp van uvicorn, een snelle ASGI-server.

#### 2.3.1.1 Installatie

- Om de service te installeren, moet je eerst de volgende stappen uitvoeren:
- Zorg ervoor dat je Python 3 en pip geïnstalleerd hebt op je systeem.
- Installeer de benodigde pakketten met het commando `pip install -r requirements.txt` in de map `/home/admin-ccs04/API`. Het bestand `requirements.txt` bevat de namen en versies van de pakketten die nodig zijn voor de API, zoals uvicorn en fastapi.
- Maak een bestand genaamd `api.service` in de map `/etc/systemd/system` met de volgende inhoud:

```
Unit]
Description=My API Service
After=network.target

[Service]
User=admin-ccs04
WorkingDirectory=/home/admin-ccs04/API
ExecStart=/usr/bin/uvicorn main:app --host 172.26.104.0 --port 8060
Restart=always

[Install]
WantedBy=multi-user.target
```

Dit bestand definieert de service die de API zal draaien. Het specificeert de beschrijving, de afhankelijkheden, de gebruiker, de werkmap, het uitvoerbare commando en het herstartbeleid van de service.

#### 2.3.1.2 De service starten, stoppen en herstarten

Om de service te starten, stoppen of herstarten, kun je gebruik maken van de volgende commando's:

- `sudo systemctl start api.service` om de service te starten.
- `sudo systemctl stop api.service` om de service te stoppen.
- `sudo systemctl restart api.service` om de service te herstarten.

Je kunt ook de status van de service controleren met het commando `sudo systemctl status api.service`.

### 2.3.2 API gebruiken

De API biedt een aantal endpoints die je kunt aanroepen met behulp van HTTP-verzoeken. Je kunt ook de interactieve documentatie bekijken op <http://172.26.104.0:8060/docs> of <http://172.26.104.0:8060/redoc>.

Hier zijn enkele voorbeelden van endpoints die je kunt gebruiken:

- GET /items om een lijst van items op te halen.
- POST /items om een nieuw item toe te voegen.
- GET /items/{item\_id} om een specifiek item op te halen op basis van zijn id.
- PUT /items/{item\_id} om een bestaand item bij te werken op basis van zijn id.
- DELETE /items/{item\_id} om een bestaand item te verwijderen op basis van zijn id.

Voor elk endpoint kun je ook optionele queryparameters of bodyparameters meegeven om het verzoek aan te passen.

### 2.3.3 Main.py

Dit bestand bevat de hoofdcode van de applicatie. Hier worden FastAPI-applicaties geconfigureerd en worden verschillende eindpunten gedefinieerd. Deze code bevat ook enkele hulpprogramma's en initialisatie van databases en andere services.

#### 2.3.3.1 Afhankelijkheden

- FastAPI: Het FastAPI-framework wordt gebruikt om web-API's te maken.
- HTTPException: FastAPI-klasse voor het afhandelen van HTTP-fouten.
- UploadFile, File: FastAPI-modellen voor het verwerken van bestandsuploads.
- Body: een FastAPI-decorateur die de body van het verzoek definieert.
- OAuth2
- PasswordBearer: FastAPI-beveiligingsklasse die OAuth2-verificatie afhandelt.
- Sessies: SQLAlchemy-klassen voor het beheren van databasesessies.
- Lijst: hint voor het specificeren van een lijst.
- Pad: klasse voor het werken met bestandspaden.
- JWTErrror, jwt: functies die JSON-webtokens verwerken.
- SendGridAPIClient, Mail: klassen voor het verzenden van e-mails via de SendGrid-service.
- Geheimen: een module die willekeurige tokens en wachtwoorden genereert.
- paramiko: SSH-clientimplementatie voor het uploaden van bestanden via SFTP.
- os: Modules voor het manipuleren van functies van het besturingssysteem.

#### 2.3.3.2 Hulpprogramma's

- get\_db(): afhankelijkheidsfunctie om de databasesessie op te halen.
- send\_email(to\_email: str, subject: str, content: str): hulpprogramma voor het verzenden van e-mails via SendGrid.

- `get_password_hash(wachtwoord: str)`: hulpprogramma voor wachtwoordhash.

### 2.3.3.3 Endpoints

- `/token`: POST-eindpunt voor het ophalen van een toegangstoken op basis van gebruikersnaam en wachtwoord.
- `/send-email/`: POST-eindpunt voor het verzenden van e-mail.
- `/users/`: POST-eindpunt voor het aanmaken van gebruikers.
- `/users/`: GET-eindpunt voor het ophalen van alle gebruikers.
- `/users/me`: GET-eindpunt voor het ophalen van de huidige gebruiker.
- `/users/{user_id}`: GET eindpunt om een specifieke gebruiker te krijgen.
- `/users/{user_id}/items/`: POST-eindpunt voor het maken van items voor een gebruiker.
- `/items/`: GET-eindpunt voor het ophalen van alle items.
- `/users/{user_id}`: PUT-eindpunt voor het updaten van gebruikers.
- `/users/{user_id}/`: DELETE eindpunt voor het verwijderen van gebruikers.
- `/upload`: Einde van POST.

### 2.3.4 Crud.py

Deze code bevat enkele functies en importfuncties die u kunt gebruiken bij het ontwikkelen van FastAPI-toepassingen met authenticatie, database-interactie en e-mailfunctionaliteit. Hieronder worden verschillende delen van de code uitgelegd en gedocumenteerd.

#### 2.3.4.1 Imports

- `OAuth2PasswordBearer` wordt geïmporteerd uit `fastapi.security` en gebruikt om de `OAuth2`-authenticatiemethode te configureren om toegangstoken te krijgen.
- Sessies worden geïmporteerd uit `sqlalchemy.orm` en gebruikt om databasesessies te maken en te gebruiken.
- Geheimen worden geïmporteerd om willekeurige tokens voor gebruikers te genereren.
- `SendGridAPIClient` en `Mail` worden geïmporteerd uit respectievelijk `sendgrid` en `sendgrid.helpers.mail` en worden gebruikt om e-mails te verzenden.
- Verificatie, modellen en schema's verwijzen naar andere modules in het project die implementaties bevatten van verificatie, databasemodellen en gegevensschema's.

#### 2.3.4.2 Functies

#### **`get_user(db: Session, user_id: int) -> models.User`**

Deze functie haalt een gebruiker op uit de database met het gebruikers-ID.

- `db`: Sessie-instantie die wordt gebruikt om te communiceren met de database.
- `user_id`: ID van de op te halen gebruiker.
- Retourneert: een exemplaar van het gebruikersmodel, of `Geen` als de gebruiker niet wordt gevonden.

**get\_user\_by\_email(db: Session, email: str) -> models.User**

Deze functie haalt gebruikers uit de database op basis van hun e-mailadres.

- db: Sessie-instantie die wordt gebruikt om te communiceren met de database.
- E-mail: het e-mailadres van de gebruiker om op te halen.
- Retourneert: een exemplaar van het gebruikersmodel, of Geen als de gebruiker niet wordt gevonden.

**get\_users(db: Session, skip: int = 0, limit: int = 100) -> List[models.User]**

Deze functie haalt de lijst met gebruikers uit de database.

- db: Sessie-instantie die wordt gebruikt om te communiceren met de database.
- Overslaan: aantal gebruikers dat moet worden overgeslagen (optioneel, standaard is 0).
- limiet: Maximum aantal gebruikers om op te halen (optioneel, standaard ingesteld op 100).
- ReturnValue: een lijst met instanties van gebruikersmodellen.

**create\_user(db: Session, user: schemas.UserCreate) -> models.User**

Deze functie creëert een nieuwe gebruiker in de database.

- db: Een instantie van Session die wordt gebruikt om met de database te communiceren.
- user: Een instantie van het gegevensschema UserCreate met de gegevens van de nieuwe gebruiker.
- Returnwaarde: Een instantie van het model User die is aangemaakt in de database.

**get\_items(db: Session, skip: int = 0, limit: int = 100) -> List[models.Item]**

Deze functie haalt een lijst van items op uit de database.

- db: Een instantie van Session die wordt gebruikt om met de database te communiceren.
- skip: Het aantal items dat moet worden overgeslagen (optioneel, standaardwaarde is 0).
- limit: Het maximale aantal items dat moet worden opgehaald (optioneel, standaardwaarde is 100).
- Returnwaarde: Een lijst van instanties van het model Item.

**create\_user\_item(db: Session, item: schemas.ItemCreate, user\_id: int) -> models.Item**

Deze functie creëert een nieuw item in de database dat toebehoort aan een specifieke gebruiker.

- db: Een instantie van Session die wordt gebruikt om met de database te communiceren.
- item: Een instantie van het gegevensschema ItemCreate met de gegevens van het nieuwe item.
- user\_id: Het ID van de gebruiker aan wie het item toebehoort.
- Returnwaarde: Een instantie van het model Item die is aangemaakt in de database.

**update\_user(db: Session, user: schemas.UserUpdate, db\_user: models.User) -> models.User**

Deze functie werkt de gegevens van een bestaande gebruiker bij in de database.

- db: Een instantie van Session die wordt gebruikt om met de database te communiceren.
- user: Een instantie van het gegevensschema UserUpdate met de bijgewerkte gegevens van de gebruiker.
- db\_user: Een instantie van het model User die moet worden bijgewerkt.
- Returnwaarde: Een instantie van het model User met de bijgewerkte gegevens.

**delete\_user(db: Session, user\_id: int) -> models.User**

Deze functie verwijdert een gebruiker uit de database op basis van een gebruikers-ID.

- db: Een instantie van Session die wordt gebruikt om met de database te communiceren.
- user\_id: Het ID van de gebruiker die moet worden verwijderd.
- Returnwaarde: Een instantie van het model User die is verwijderd uit de database.

**send\_email(to\_email: str, subject: str, content: str) -> Any**

Deze functie verzendt een e-mail naar een specifiek e-mailadres.

- to\_email: Het e-mailadres van de ontvanger.
- subject: Het onderwerp van de e-mail.
- content: De inhoud van de e-mail.
- Returnwaarde: Een willekeurige waarde die aangeeft of het verzenden van de e-mail is gelukt.

### **get\_user\_dir(db: Session, email: str) -> str**

Deze functie retourneert het directorypad van een gebruiker op basis van het e-mailadres.

- db: Een instantie van Session die wordt gebruikt om met de database te communiceren.
- email: Het e-mailadres van de gebruiker.
- Returnwaarde: Een string die het directorypad van de gebruiker voorstelt.

## **2.3.5 Models.py**

### **2.3.5.1 User model**

De klasse User vertegenwoordigt een gebruiker in het systeem. Het heeft de volgende velden:

- id (geheel getal, primaire sleutel): unieke identificatie voor de gebruiker.
- email (tekenreeks, uniek): het e-mailadres van de gebruiker.
- hashed\_password (tekenreeks): het gehashte wachtwoord van de gebruiker.
- is\_active (boolean, default = True): Geeft aan of de gebruiker actief is.
- files\_directory (tekenreeks, standaard=f"./user\_files/{id}"): pad naar de map waar gebruikersbestanden zijn opgeslagen.
- token (String): Nieuw veld voor het token van de gebruiker.

Het gebruikersmodel heeft ook een relatie met het itemmodel via het itemveld. In het artikelmodel wordt dit nader toegelicht.

### **2.3.5.2 Item model**

De klasse Item vertegenwoordigt items die eigendom zijn van een gebruiker. Het heeft de volgende velden:

- id (geheel getal, primaire sleutel): unieke identificatie voor het item.
- titel (string): De titel van het element.
- Beschrijving (tekenreeks): Een beschrijving van het item. owner\_id
- (geheel getal, externe sleutel voor 'users.id'): ID van de gebruiker die eigenaar is van het item.

Het artikelmodel is gerelateerd aan het gebruikersmodel via het veld Eigenaar. Dit maakt het gemakkelijk om het eigendom van het artikel te verifiëren.

## **2.3.6 Schemas.py**

### **2.3.6.1 item basis**

De klasse ItemBase definieert basiseigenschappen voor items. Het heeft de volgende kenmerken:

titel: Een verplicht attribuut van het type str dat de titel van het element vertegenwoordigt.

description: Een optioneel attribuut van het type str dat de beschrijving van het element vertegenwoordigt.

### **2.3.6.2 item create**

De klasse ItemCreate is een subklasse van ItemBase en erft al zijn attributen.

### **2.3.6.3 item**

De klasse Item vertegenwoordigt een item en bevat de volgende attributen.

id: Een vereist attribuut van het type int dat de unieke ID van het element vertegenwoordigt.

owner\_id: een vereist kenmerk van het type int dat de ID van de eigenaar van het item vertegenwoordigt.

De klasse Item heeft ook een interne Config-klasse die wordt gebruikt om de configuratie-opties van het model te wijzigen. In dit geval wordt orm\_mode ingesteld op True om aan te geven dat het model wordt gebruikt voor ORM (Object Relational Mapping).

### **2.3.6.4 UserBase**

De klasse UserBase definieert basiseigenschappen voor gebruikers. Het heeft de volgende kenmerken:

email: een vereist kenmerk van het type str dat het e-mailadres van de gebruiker vertegenwoordigt.

### **2.3.6.5 UserCreate**

De klasse UserCreate is een subklasse van UserBase en erft al zijn attributen. Het heeft echter extra kenmerken.

Wachtwoord: een vereist kenmerk van het type str dat het wachtwoord van de gebruiker vertegenwoordigt.

### **2.3.6.6 User**

De klasse User vertegenwoordigt een gebruiker en bevat de volgende eigenschappen:

id: een vereist kenmerk van het type int, dat de unieke ID van de gebruiker vertegenwoordigt.

`is_active`: Vereist bool-attribuut dat aangeeft of de gebruiker actief is of niet.

`items`: Een eigenschap van het type `List` die de items van de gebruiker bevat.

De `User`-klasse heeft ook een interne `Config`-klasse voor het wijzigen van de configuratie-opties van het model. In dit geval wordt `orm_mode` ingesteld op `True`, wat aangeeft dat het model wordt gebruikt voor ORM (Object Relational Mapping). Het uitsluitingsattribuut is ook aanwezig, wat aangeeft dat het "token" attribuut niet wordt opgenomen in de gegenereerde JSON-uitvoer.

#### **2.3.6.7 userUpdate**

De klasse `UserUpdate` definieert kenmerken die voor een gebruiker kunnen worden bijgewerkt. Het heeft de volgende optionele attributen:

- `E-mail`: het e-mailadres van de gebruiker.
- `full_name`: de volledige naam van de gebruiker.
- `Wachtwoord`: wachtwoord van de gebruiker.

Opmerking: elk van deze attributen is gedefinieerd als Optioneel, met een standaardwaarde van `Geen`, zodat u ze indien nodig kunt bijwerken.

### **2.3.7 Database.py**

Deze code demonstreert het opzetten van een SQLite-databaseverbinding en -configuratie met behulp van `SQLAlchemy`, een populaire Python SQL-toolkit en object-relational mapping (ORM)-bibliotheek. Dit document beschrijft de verschillende procedures en beschrijft de gebruikte bibliotheken en concepten.

#### **2.3.7.1 installeren**

Voordat u de code start, moet u ervoor zorgen dat `SQLAlchemy` is geïnstalleerd. Installeer `SQLAlchemy` via `pip` met de volgende opdracht:

```
pip installeer sqlalchemy
```

#### **2.3.7.2 importeer de vereiste bibliotheken**

Eerst moeten we de juiste bibliotheek in ons Python-script importeren.

```
Importeer vanuit sqlalchemy create_engine
```

```
importeer declarative_base uit sqlalchemy.ext.declarative
```

```
sqlalchemy.orm geïmporteerd uit Sessionmaker
```

- `create_engine`: maakt een database-engine die verbinding maakt met een database en SQL-query's uitvoert.
- `declarative_base`: Dit is een tool waarmee je een basisklasse kunt definiëren voor declaratieve klassen in `SQLAlchemy`.
- `sessionmaker`: laten we een sessiefabriek maken voor het maken van databasesessies.



### 2.3.7.3 De database-URL configureren

In dit voorbeeld wordt een SQLite-database gebruikt. Definieer de database-URL in een variabele met de naam `SQLALCHEMY_DATABASE_URL`.

```
SQLALCHEMY_DATABASE_URL = "sqlite:///./sqllitedb/sqllitedata.db"
```

Dit is de URL om verbinding te maken met de SQLite-database `sqllitedata.db` in de `sqllitedb`-map van de huidige werkdirectory.

### 2.3.7.4 Bouw een database-engine

Maak een database-engine met behulp van de functie `create_engine` en de eerder gedefinieerde database-URL.

```
engine = create_engine(  
    SQLALCHEMY_DATABASE_URL, echo=True,  
    connect_args={"check_same_thread": False},  
)
```

- `SQLALCHEMY_DATABASE_URL`: URL van de eerder gedefinieerde database. `echo=True`: hiermee worden alle uitgevoerde SQL-query's naar de console afgedrukt voor foutopsporing en ontwikkelingsdoeleinden.
- `connect_args={"check_same_thread": False}`: Dit is een SQLite-specifieke configuratie om de veiligheid van de databaseverbinding te garanderen bij gebruik in een multithreaded context.

### 2.3.7.5 Maak een sessiefabriek

Gebruik vervolgens de functie `sessionmaker` om een session factory te maken en deze te koppelen aan uw database-engine.

```
SessionLocal = sessionmaker(autocommit=False, autoflush=False,  
    bind=engine)
```

- `autocommit=False`: geeft aan dat de sessie handmatig wordt beheerd en niet automatisch wordt vastgelegd bij elke wijziging.
- `autoflush=False`: geeft aan dat sessies handmatig worden beheerd en niet automatisch worden leeggemaakt voordat de query wordt uitgevoerd.
- `bind=engine`: Dit bindt de session factory aan de database-engine.

### 2.3.7.6 declareer een basisklasse

Gebruik ten slotte `declarative_base` om de basisklasse te definiëren. Deze basisklasse wordt gebruikt om modelklassen te definiëren.

```
Basis = declarative_base()
```

Hierdoor kunt u de declaratieve syntaxis van SQLAlchemy gebruiken bij het definiëren van tabellen en hun relaties.

## 2.3.8 Auth.py

### 2.3.8.1 Importeer de vereiste bibliotheken.

- Importeer de OAuth2PasswordBearer-klasse van het fastapi.security-pakket voor het beheren van OAuth2-wachtwoordverificatie.
- Importeer de Depends, HTTPException en status klassen van het fastapi-pakket voor het afhandelen van afhankelijkheden en HTTP-fouten.
- Importeer de datetime en timedelta klassen van het datetime-pakket voor het werken met datums en tijden.
- Importeer de CryptContext klasse van het passlib.context-pakket voor het hashen en verifiëren van wachtwoorden.
- Importeer de jwt-functie en JWTError-klasse van het jose-pakket voor het genereren en decoderen van JWT's.
- Importeer de Session klasse van het sqlalchemy.orm-pakket voor het werken met database-sessies.

### 2.3.8.2 Configuratievariabelen:

- SECRET\_KEY: geheime sleutel die wordt gebruikt om de JWT te ondertekenen.
- Algorithm: algoritme dat wordt gebruikt om de JWT te ondertekenen en te decoderen.
- ACCESS\_TOKEN\_EXPIRE\_MINUTES: Het aantal minuten dat het toegangstoken geldig is.

### 2.3.8.3 Wachtwoordcontext:

Maak een instantie van de klasse CryptContext met de gewenste configuratie. In dit geval wordt het bcrypt-algoritme gebruikt. Deze context wordt gebruikt voor wachtwoord-hashing en verificatie.

### 2.3.8.4 Wachtwoord hash-functie:

De functie get\_password\_hash neemt een wachtwoord als invoer en retourneert het gehashte wachtwoord met behulp van de pwd\_context-instantie.

### 2.3.8.5 Mogelijkheid om wachtwoorden te valideren:

De functie Explore\_password accepteert onbewerkte en gehashte wachtwoorden als invoer. De functie vergelijkt het gehashte wachtwoord met het onbewerkte wachtwoord en retourneert True als ze overeenkomen, False als ze niet overeenkomen.

### 2.3.8.6 Functie om toegangstoken aan te maken:

- De functie create\_access\_token accepteert een datadictionary als invoer.
- Maak een kopie van de gegevens en stel de vervaldatum van het token in op basis van de variabele ACCESS\_TOKEN\_EXPIRE\_MINUTES.
- Onderteken de JWT met SECRET\_KEY en ALGORITHM en stuur de gecodeerde JWT terug.

### **2.3.8.7 Mogelijkheid om gebruikers te authenticeren:**

De functie `authenticate_user` accepteert een databasesessie (`db`), gebruikersnaam en wachtwoord als invoer. Hij haalt een gebruiker uit de database op basis van de gebruikersnaam. Als de gebruiker niet bestaat of als de wachtwoorden niet overeenkomen, retourneert de functie `False`. anders wordt de gebruiker geretourneerd.

### **2.3.8.8 Een functie om de huidige gebruiker te krijgen:**

De functie `get_current_user` accepteert een databasesessie (`db`) en token als invoer. De token wordt ontsleutelt met behulp van `SECRET_KEY` en `ALGORITHM`. Haal de gebruikersnaam op uit het gedecodeerde token. Als de gebruikersnaam niet bestaat, wordt een HTTP-foutcode 401 (Ongeautoriseerd) gegenereerd. Haal een gebruiker uit de database op basis van de gebruikersnaam. Als de gebruiker niet bestaat, wordt een HTTP-foutcode 401 (Ongeautoriseerd) gegenereerd. anders wordt de gebruiker geretourneerd.

### **2.3.8.9 Functie om momenteel actieve gebruikers te krijgen:**

De functie `get_current_active_user` accepteert een databasesessie (`db`) en token als invoer. Hij haalt de huidige gebruiker op door de functie `get_current_user` aan te roepen. Als de gebruiker inactief is, wordt een HTTP-foutcode 400 (Bad Request) gegenereerd anders wordt de gebruiker geretourneerd.

## **2.3.9 Reset\_database**

### **2.3.9.1 importeer de vereiste modules**

Zorg ervoor dat de vereiste modules correct in uw code zijn geïmporteerd. Importeer in dit geval de `motormodule` uit het databasebestand en de `basismodule` uit het modelbestand. Deze modules zijn vereist om toegang te krijgen tot de database-engine en bieden de functionaliteit die nodig is om de database te beheren.

### **2.3.9.2 Laat de bestaande tabel vallen**

Voordat u de database opnieuw maakt, moet u de bestaande tabellen verwijderen. Dit kan worden bereikt door de methode `drop_all()` van de `metadata-eigenschap` van het basisobject aan te roepen. Deze methode laat alle tabellen vallen die in het basisobject zijn gedefinieerd.

### **2.3.9.3 Bouw de database opnieuw op**

Nadat u de bestaande tabellen hebt verwijderd, bouwt u de database opnieuw op. Dit kunt u doen door de methode `create_all()` van de `metadata-eigenschap` van het basisobject aan te roepen. Met deze methode worden alle tabellen gemaakt die in het basisobject zijn gedefinieerd.

#### **2.3.9.4 Test uw code**

Na het voltooien van de bovenstaande stappen, is het belangrijk om uw code te testen om er zeker van te zijn dat de database correct is gereset. Voer de code uit en controleer of deze het gewenste resultaat oplevert. Als u een foutmelding krijgt, controleert u uw code en zorgt u ervoor dat alle stappen correct zijn geïmplementeerd.

## 2.4 NFS Shared Map Setup

Instructies voor het opzetten van een gedeelde map met behulp van Network File System (NFS) tussen een server (172.26.192.197) en een client (172.26.192.219). De gedeelde map geeft de client toegang tot bestanden op de server. De hostnaam van de server is admin-ccs04, en de gedeelde map bevindt zich in /home/admin-ccs04/API/files. De setup omvat ook automatische synchronisatie van bestanden tussen de server en de client.

### 2.4.1 Server Configuratie

1. Install NFS op de server:

```
sudo apt update sudo apt install nfs-kernel-server
```

2. Configureer de NFS export op de server:

```
sudo nano /etc/exports
```

Voeg de volgende regel toe aan het bestand:

```
/home/admin-ccs04/API/files 172.26.192.219(rw,sync,no_subtree_check)
```

Save the file and exit the editor.

3. Restart de NFS server:

```
sudo systemctl restart nfs-kernel-server
```

### 2.4.2 Client Configuratie

1. Installeer NFS op de client:

```
sudo apt update sudo apt install nfs-common
```

2. Maak de shared map aan op de client:

```
sudo mkdir /mnt/shared
```

```
sudo mount 172.26.192.197:/home/admin-ccs04/API/files /mnt/shared
```

De gedeelde map van de server is nu toegankelijk in /mnt/shared op de client.

### 2.4.3 Automount van de NFS share:

Om de NFS share automatisch te mounten tijdens het opstarten, voegt u een entry toe aan het bestand /etc/fstab:

```
sudo nano /etc/fstab
```

Voeg de volgende lijn toe:

```
172.26.192.197:/home/admin-ccs04/API/files /mnt/shared nfs
auto,nofail,noatime,nolock,intr,tcp,actimeo=1800 0 0
```

Save the file and exit the editor.

Om de share te mounten zonder opnieuw op te starten, voert u het volgende commando uit:

```
sudo mount -a
```

De NFS-share wordt nu automatisch gemount op de client tijdens het opstarten.

De installatie van de NFS gedeelde map is nu voltooid. De gedeelde map in /home/admin-ccs04/API/files op de server is toegankelijk op de client in /mnt/shared. Alle wijzigingen op de server worden automatisch gesynchroniseerd met de client.

## 2.5 Duplicatie back-ups

### 2.5.1 Installatie op VMs

#### 2.5.1.1 Stap 1: Installatie van Mono

- Voeg de apt-sleutel toe:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF
```

- Voeg de bronlijst toe:

```
echo "deb https://download.mono-project.com/repo/ubuntu stable-focal main" | sudo tee /etc/apt/sources.list.d/mono-official-stable.list
```

- Werk de apt-lijst bij:

```
sudo apt update
```

- Installeer Mono en de vereiste extra's:

```
sudo apt install mono-devel gtk-sharp2 libappindicator0.1-cil libmono-2.0-1
```

#### 2.5.1.2 Stap 2: Installatie van Duplicati

- Installeer de vereisten:

```
sudo apt install apt-transport-https nano git-core software-properties-common dirmngr -y
```

- Download de nieuwste versie:

```
wget https://updates.duplicati.com/beta/duplicati\_2.0.6.3-1\_all.deb
```

- Installeer het gedownloade bestand:

```
sudo apt install ./duplicati_2.0.6.3-1_all.deb -y
```

- Maak de systemd-service aan:

```
sudo nano /etc/systemd/system/duplicati.service
```

- Plak de volgende tekst en sla het bestand op:

```
[Unit]
```

```
Description=Duplicati web-server
```

```
After=network.target
```

```
[Service]
```

```
Nice=19

IOSchedulingClass=idle

EnvironmentFile=-/etc/default/duplicati

ExecStart=/usr/bin/duplicati-server $DAEMON_OPTS

Restart=always
```

```
[Install]
```

```
WantedBy=multi-user.target
```

- Bewerk het configuratiebestand voor Duplicati:

```
sudo nano /etc/default/duplicati
```

- Bewerk DAEMON\_OPTS="" als volgt. Je kunt de webservice-poort wijzigen als deze conflicteert met andere services, en het webservice-interface als je een specifiek adres wilt binden.

```
DAEMON_OPTS="--webinterface=any --webinterface-port=8200 --portable-mode"
```

- Activeer en start de Daemon:

```
sudo systemctl enable duplicati.service
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl start duplicati.service
```

- Controleer de status:

```
sudo systemctl status duplicati.service
```

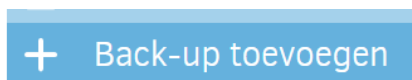
### **2.5.1.3 Eerste uitvoering:**

Blader naar `http://serverip:8200` en je zou de eerste installatiewizard moeten zien.



## 2.5.2 Back-ups maken in webinterface

### Stap 1: Klik op Back-up toevoegen



*Back-up toevoegen Duplicati*

**Stap2: Zorg ervoor dat er een nieuwe back-up gemaakt wordt door het bolletje aan te duiden dat hier voor zorgt.**

#### Nieuwe back-up toevoegen

- ☒ Een nieuwe back-up instellen  
Voer configuratie-details in
- ☐ Importeer vanuit een bestand  
Laad een configuratie vanuit een geëxporteerde taak of een opslagprovider

Volgende >

*Nieuwe back-up Duplicati*

### Stap 3: Vul de algemene back-upinstellingen in

#### Algemene back-upinstellingen

Naam

Backup-primaryVM

Omschrijving (optioneel)

Versleuteling

AES-256 encryptie, ingebouwd

Wachtwoordzin

\*\*\*\*\*

Herhaal wachtwoordzin

\*\*\*\*\*

[Tonen](#) | [Genereer](#) | Strength: Sterk

Volgende >

*Algemene back-upinstellingen Duplicati*

## Stap 4: Vul het back-updoel in

Klik op authID om je ID te bemachtigen

Back-updoel ⋮

Opslagtype  
Google Drive ▼

Pad op server  
back-ups(primaryVM)

AuthID  
239a7f5500676505381aee80ca876d79:wVaW6BbUjRe5f0

Test verbinding

Geavanceerde opties ▼

< Vorige Volgende >

### Back-updoel Duplicati

Test de verbinding om zeker te zijn dat er een connectie is met je google drive account

Back-updoel ⋮

Opslagtype  
Google Drive ▼

Pad op server  
back-ups(primaryVM)

AuthID  
239a7f5500676505381aee80ca876d79:wVaW6BbUjRe5f0

Testen ...

Testen van de verbinding ...

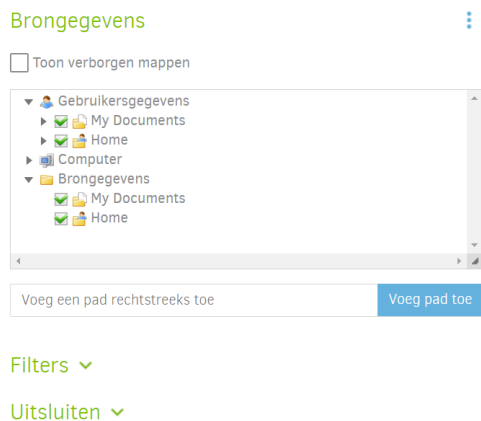
Testen ...

Geavanceerde opties ▼

### Connectie testen Duplicati

## Stap 5: Brongegevens

Klik de gewenste mappen aan die je wil back-uppen.



## Stap 6: automatisatie

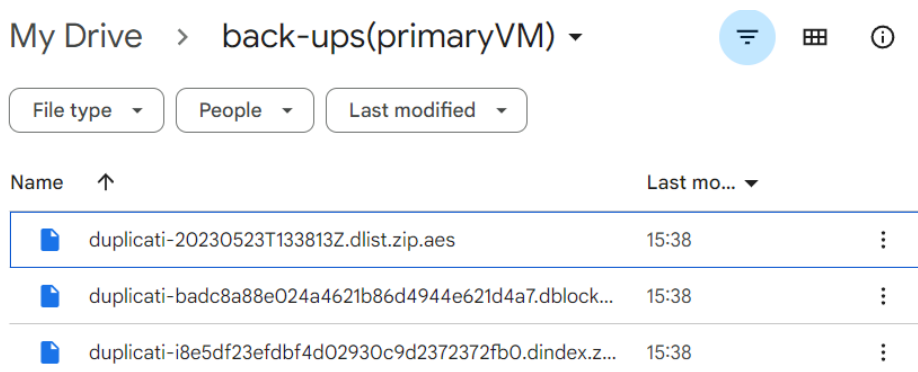
Zet de automatische back-ups uit.

## Stap 7: uitvoeren van de back-up



*Back-up uitvoeren Duplicati*

## Stap 8: De back-ups zijn terug te vinden in de onedrive.



*Back-ups bekijken in onedrive Duplicati*

## 2.6 Monitoring

### 2.6.1 Zabbix-monitoring tool

zabbix is een tool om je netwerk, applicaties, machines, cloud en veel meer te monitoren. met zabbix kan je meerdere parameters van je machines bekijken zoals RAM, CPU en netwerkverkeer. Wij gebruiken Zabbix op een server om ons netwerk van Virtual Machines te monitoren, hierbij gebruiken wij een MySQL database en een Apache webserver voor de frontend.



*Zabbix logo*

### 2.6.2 Zabbix installeren

Om Zabbix te installeren moet je eerst een paar packets installeren op je VM. Met deze commando's ga je de zabbix repository binnenhalen.

```
wget https://repo.zabbix.com/zabbix/6.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.4-1+ubuntu22.04_all.deb
```

```
dpkg -i zabbix-release_6.4-1+ubuntu22.04_all.deb
```

```
apt update
```

Met volgende commando's installeer je de front end agent van zabbix. Hier installeer je tegelijkertijd MySQL.

```
apt install zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf zabbix-sql-scripts zabbix-agent
```

Werk de database uit, maak een gebruiker.

```
mysql -uroot -p  
password
```

```
mysql> create database zabbix character set utf8mb4 collate  
utf8mb4_bin;
```

```
mysql> create user zabbix@localhost identified by 'password';
```

```
mysql> grant all privileges on zabbix.* to zabbix@localhost;
```

```
mysql> set global log_bin_trust_function_creators = 1;
```

```
mysql> quit;
```

Importeer op de Zabbix server host het initiële schema en de gegevens. U wordt gevraagd uw nieuw aangemaakte wachtwoord in te voeren.

```
zcat /usr/share/zabbix-sql-scripts/mysql/server.sql.gz | mysql --
default-character-set=utf8mb4 -uzabbix -p zabbix
```

Schakel de optie "log\_bin\_trust\_function\_creators" uit na het importeren van database schema's.

```
mysql -uroot -p
```

password

```
mysql> set global log_bin_trust_function_creators = 0;
```

```
mysql> quit;
```

Configureer de database voor de Zabbix-server.

Bewerk het bestand /etc/zabbix/zabbix\_server.conf

```
DBPassword=password
```

```
StartVMwareCollectors=1
```

Zabbix server en agent processen starten.

Start de Zabbix server en agent processen en laat ze starten bij het opstarten van het systeem.

```
systemctl restart zabbix-server zabbix-agent apache2
```

```
systemctl enable zabbix-server zabbix-agent apache2
```

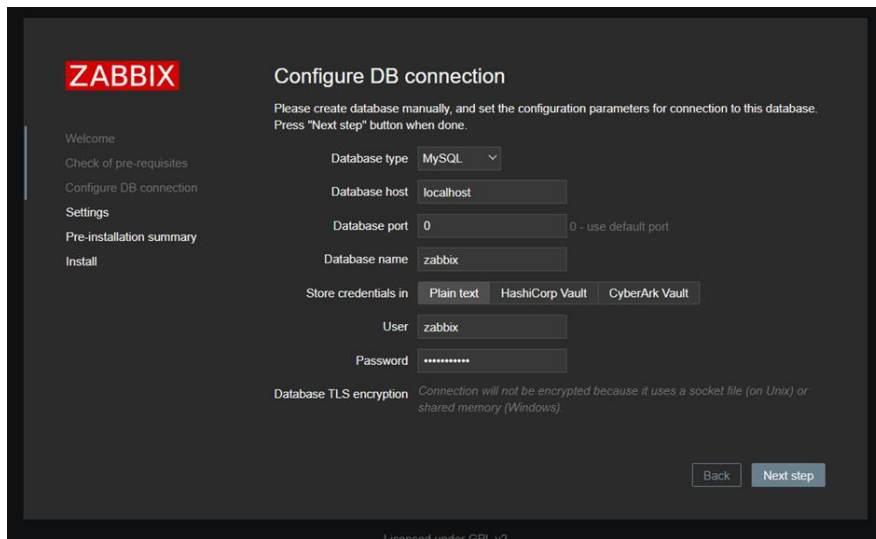
Open Zabbix UI webpagina door te surfen naar De standaard URL voor de Zabbix UI bij gebruik van de Apache webserver is:

<http://host/zabbix>.

In ons geval:

<http://172.26.104.1/zabbix>

Als we nu surfen naar dit adres komen we terecht op de front end van Zabbix, hier moeten we de front end connecteren met de database. Hier kunnen we alles default laten staan. we moeten enkel het wachtwoord gekozen ingevuld worden.



**ZABBIX** Configure DB connection

Please create database manually, and set the configuration parameters for connection to this database. Press "Next step" button when done.

Database type:

Database host:

Database port:  0 - use default port

Database name:

Store credentials in:

User:

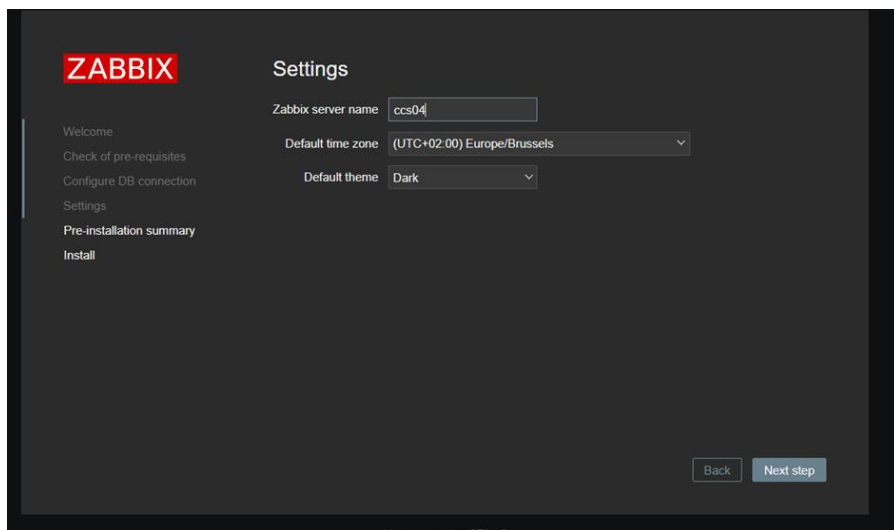
Password:

Database TLS encryption:

Licensed under GPL v2

### Connectie zabbix

Kies voor een passende servernaam en de juiste tijdzone, hierna is de installatie klaar.



**ZABBIX** Settings

Zabbix server name:

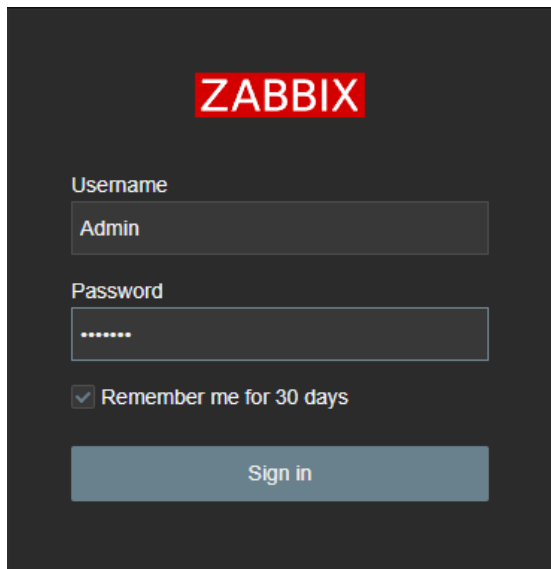
Default time zone:

Default theme:

Licensed under GPL v2

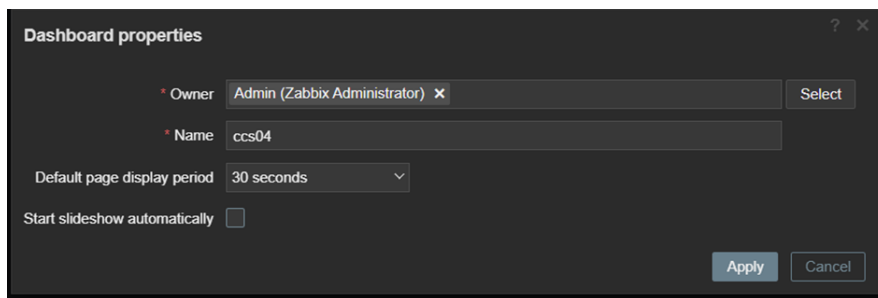
### Tijdzone selectie zabbix

Na nogmaals inloggen, kunnen we ons dashboard aanmaken.



The image shows the Zabbix login interface. At the top, the ZABBIX logo is displayed in red. Below it, there are two input fields: 'Username' with the value 'Admin' and 'Password' with masked characters. A checkbox labeled 'Remember me for 30 days' is checked. At the bottom, there is a 'Sign in' button.

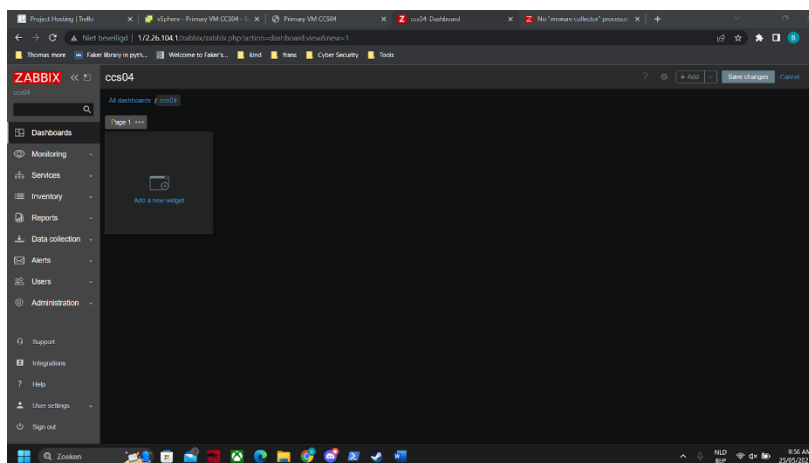
*Inloggen zabbix*



The image shows the 'Dashboard properties' dialog box. It contains the following fields: 'Owner' set to 'Admin (Zabbix Administrator)' with a 'Select' button, 'Name' set to 'ccs04', 'Default page display period' set to '30 seconds' with a dropdown arrow, and 'Start slideshow automatically' with an unchecked checkbox. At the bottom right, there are 'Apply' and 'Cancel' buttons.

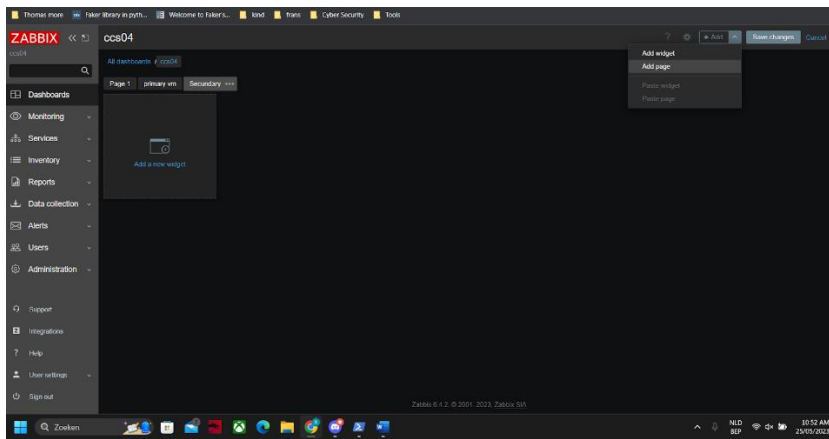
*Dashboard properties zabbix*

Ons dashboard is nu nog leeg, nu kunnen we handmatig onze machines toevoegen.



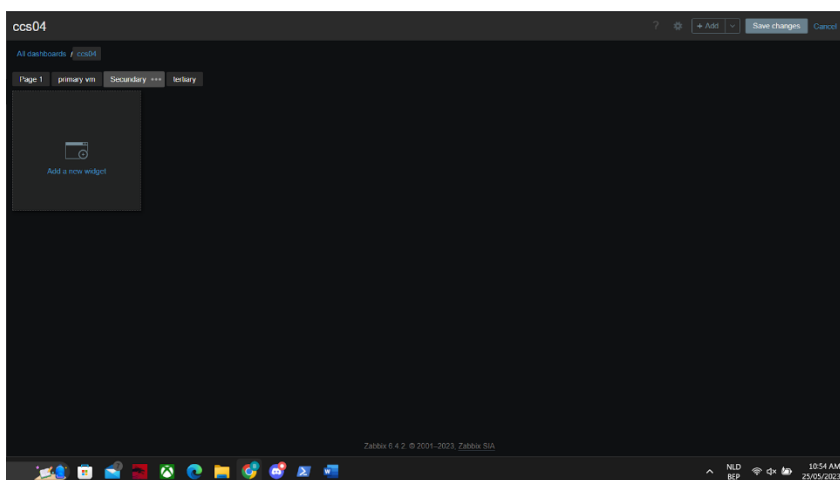
*Dashboard zabbix*

We kunnen een pagina toevoegen die samenhangt met een VM.



### *Pagina toevoegen zabbix*

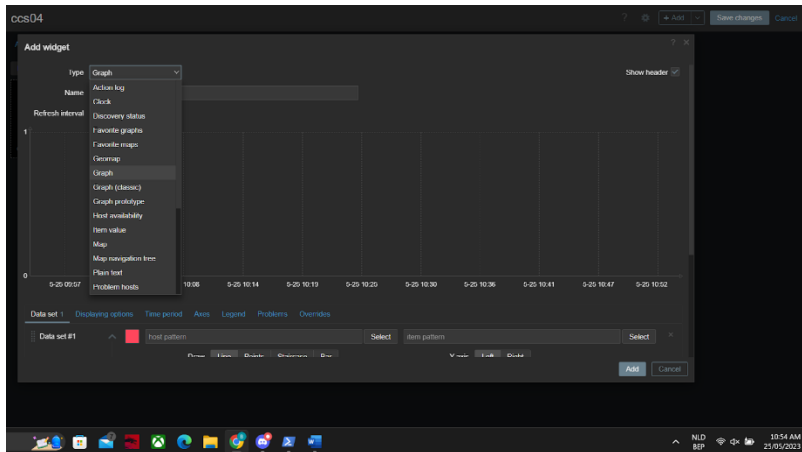
Nu kunnen we widgets toevoegen die gekozen parameters monitoren.



### *Widgets zabbix*

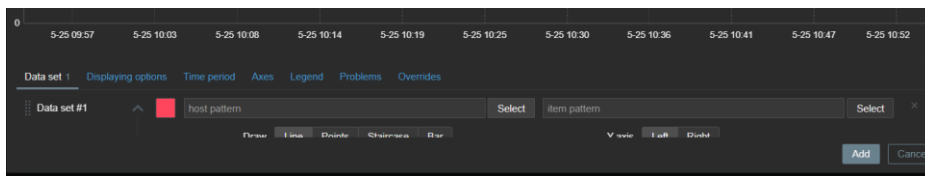
Dan kunnen we verschillende soorten diagrammen kiezen. We kiezen hier nu voor graph om een grafiek te krijgen.





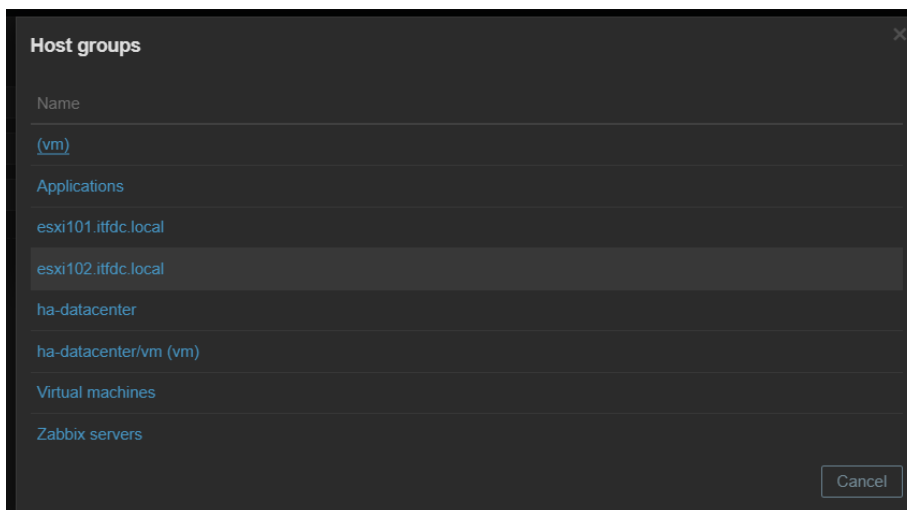
*Diagrammen kiezen zabbix*

Klik hier op de eerste Select onderaan.



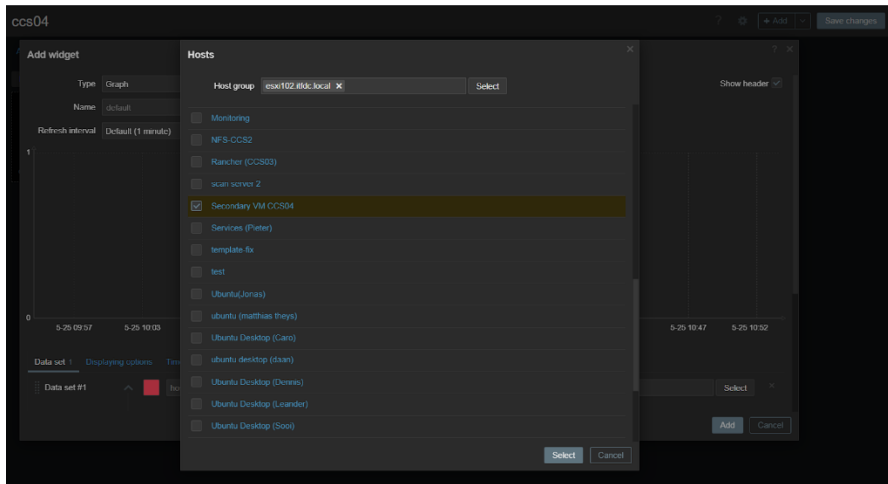
*Selecteren zabbix*

Hier kiezen we een host waar onze virtuele machine op draait. Onze secundaire VM draait in host 172.26.0.102 dus we klikken hier op esxi102.itfdc.local .



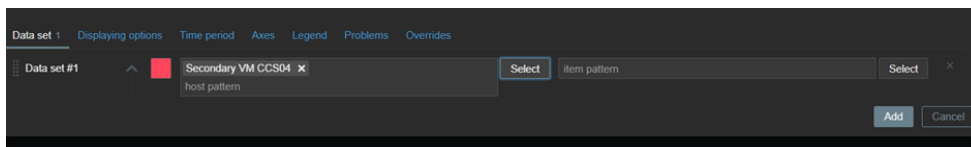
*Host zabbix*

Dan kiezen we onze Secundaire VM of VM naar keuze.



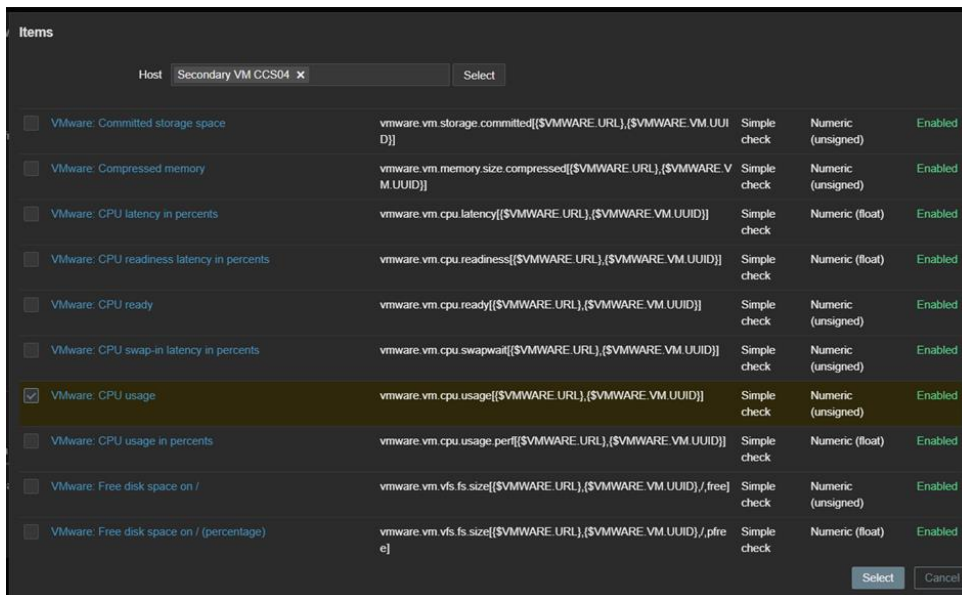
*Gekozen host zabbix*

Dan kunnen we onze grafiek aanpassen naar onze eigen keuze. Hier kiezen we de select links onderaan.



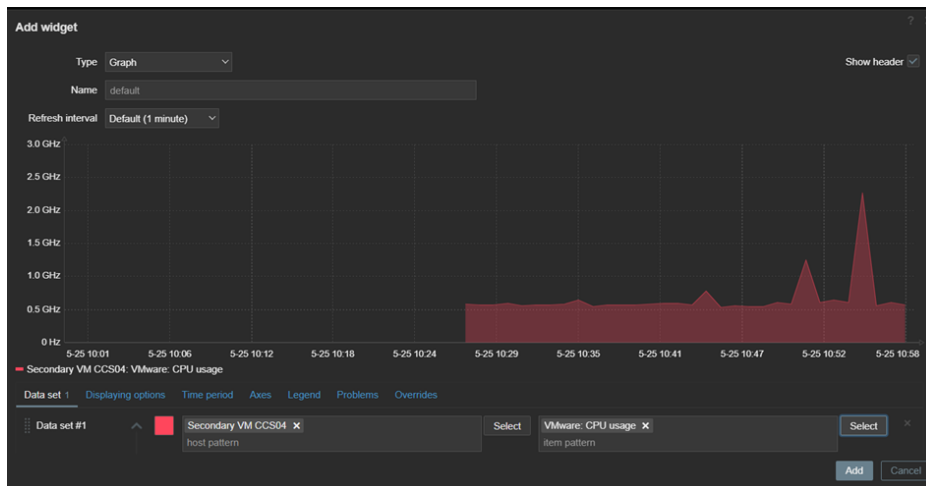
*Aanpassen grafiek zabbix*

Dan kunnen we kiezen wat we willen monitoren. We kiezen hier CPU usage.



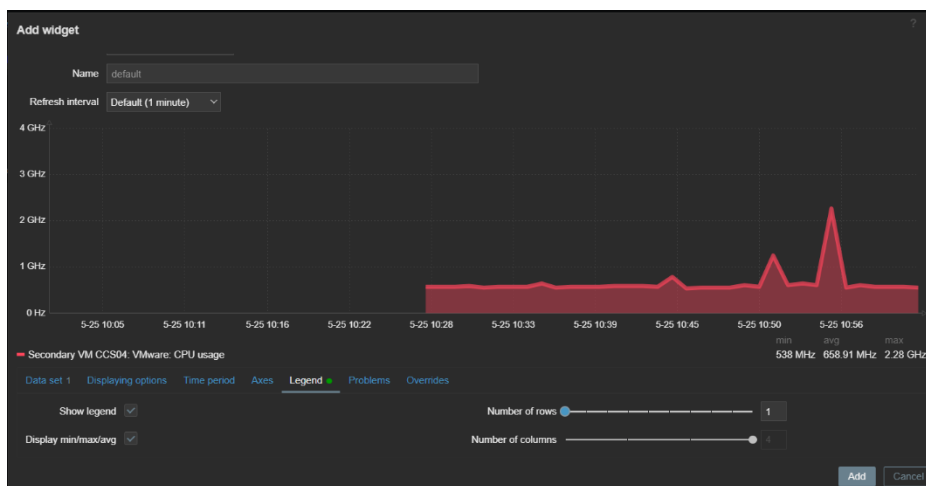
*Monitoren zabbix*

dan krijgen we de volgende grafiek.



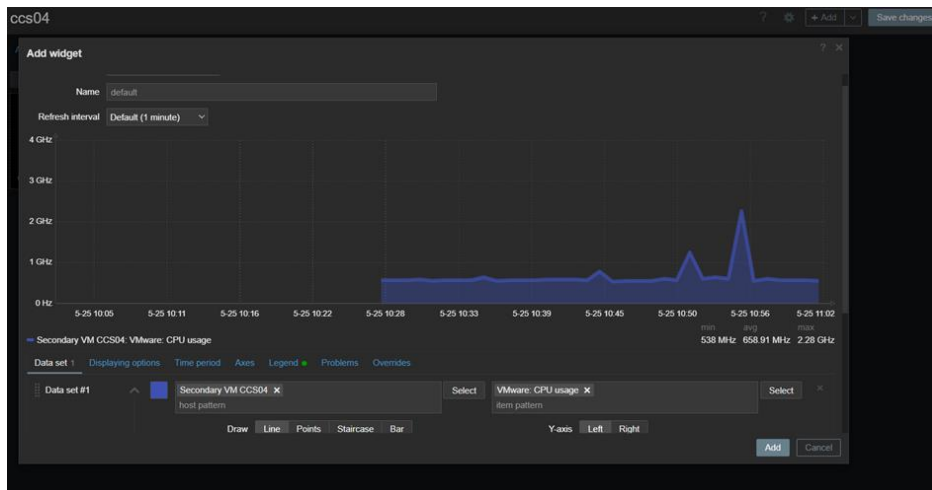
*Volgende grafiek zabbix*

We kunnen ook onze legenda aanpassen. Hier tonen we dan nog het minimum gemiddelde en maximum cpu usage onder de grafiek



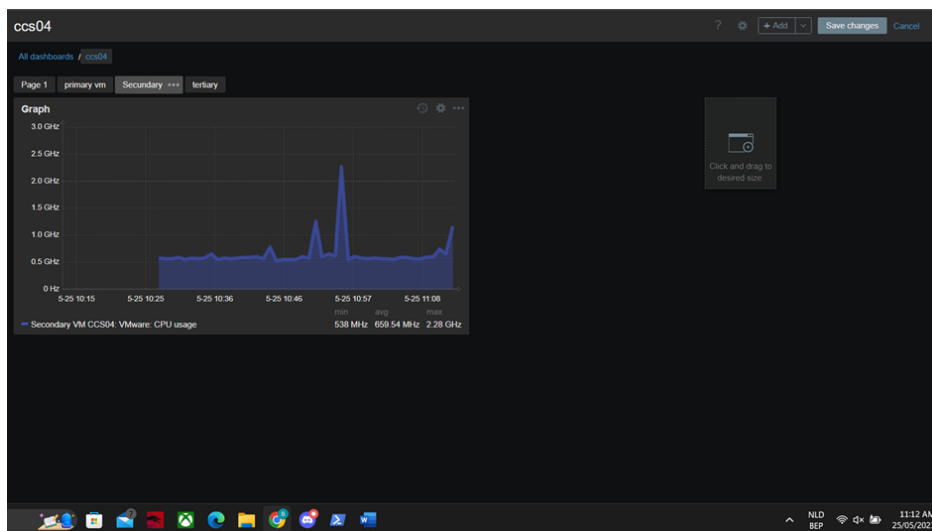
*Legenda aanpassen zabbix*

Dan kunnen we de kleur nog aanpassen naar keuze. Wij hebben voor blauw gekozen.



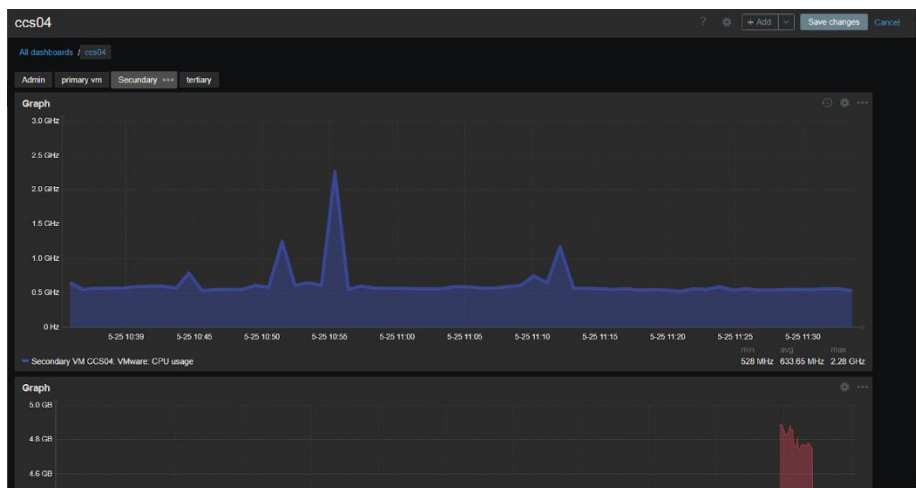
### Kleur aanpassen zabbix

Dan klikken we add en krijgen we onze grafiek in het dashboard te zien onder de pagina Secondary. Als je al de gewilde grafieken hebt gemaakt dan kun je rechtboven op save changes klikken



### Add grafiek zabbix

Dan kun je door de pagina's gaan en kijken naar de monitoring.



*Kijken door pagina's zabbix*

Nu kan dit meerdere malen herhaald worden voor andere parameters naar keuze.

## **3 HANDLEIDING VOOR DE KLANTEN**

### **3.1 API**

#### **3.1.1 Gebruiker aanmaken**

Maak een POST-verzoek aan `/users/` met de volgende gegevens in de aanvraagtekst:

E-mail: het e-mailadres van de gebruiker die u wil aanmaken

Wachtwoord: wachtwoord van de gebruiker die u wil aanmaken

U ontvangt een JSON-antwoord met de gemaakte gebruiker. Hierna krijgt u een mail waar dat het aanmaken van uw nieuwe gebruiker bevestigd wordt. Uw toegangstoken wordt ook naar dit e-mailadres gestuurd.

#### **3.1.2 Gebruiker valideren**

Log in om een toegangstoken te krijgen.

Stuur een POST-verzoek naar `/token` met behulp van het volgende formulier.

Gebruikersnaam: gebruikersnaam

wachtwoord: wachtwoord

Zonder validatie kan u niets doen op het hostingsplatform.

#### **3.1.3 Gebruiker bijwerken**

Doe een PUT-verzoek aan `/users/{user_id}`. Vervang `{user_id}` door de ID van de gebruiker. Geef de gegevens op die moeten worden bijgewerkt in de hoofdtekst van het verzoek. Ontvang een JSON-antwoord met de bijgewerkte informatie van de gebruiker.

#### **3.1.4 Gebruiker verwijderen**

Dien een DELETE-verzoek in bij `/users/{user_id}` en vervang `{user_id}` door uw gebruikers-ID. Ontvang een JSON-antwoord met de verwijderde gebruiker.

#### **3.1.5 Bestanden opladen**

Stuur een POST-verzoek naar `/upload` met het te uploaden bestand als bijlage. Het bestand wordt geüpload naar de opgegeven doelserver. Als het uploaden is gelukt, ontvangt u een JSON-antwoord met een succesbericht.

#### **3.1.6 Bestanden bijwerken**

Stuur een PUT-verzoek naar `/updatefile/{file_id}`. Vervang `{file_id}` door de ID van het bestand dat moet worden bijgewerkt. Specificeer het bestand dat moet worden bijgewerkt als bijlage in het verzoek. Ontvang een JSON-antwoord met bijgewerkte bestandsinformatie.

### **3.1.7 Bestanden verwijderen**

Doe een DELETE-verzoek aan /deletefile/{file\_id}. Vervang {file\_id} door de ID van het te verwijderen bestand. U ontvangt een JSON-antwoord waarin wordt bevestigd dat het bestand is verwijderd.

### **3.1.8 Project deployen**

Voer een script uit op een externe server door een POST-verzoek naar /execute-script te sturen. U ontvangt een JSON-antwoord met de uitvoer van uw script.

## 4 TROUBLESHOOTING

### 4.1 Zabbix

Wanneer we zabbix aan het installeren waren kregen we telkens de error dat er beschadigde pakketten werden bijgehouden waardoor we een aantal commando's niet konden uitvoeren. Dit kwam door de VM zelf en hiervoor hebben we al de pakketten wat er mee te maken had verwijderd en opnieuw geïnstalleerd waardoor dit uiteindelijk zichzelf oplosten.

### 4.2 Disaster recovery plan

In dit deel van de documentatie gaan we het hebben over stappenplan dat je moet uitvoeren mochten er zich problemen voordoen.

#### 4.2.1 Stap 1:

Als eerste kun je proberen door de laatste genomen snapshot van de VM's terug te zetten. Er worden om de paar uur snapshots genomen, zo kan u altijd enkele iteraties terug gaan, en zo terug een werkend platform creëren.

#### 4.2.2 Stap 2:

Als vorige stap niet voldoende / niet van toepassing is, moet je alle servers verwijderen uit het datacenter en kan je deze terug deployen van de voorgemaakte templates in vSphere.

Bij deze stap heb je 2 opties, oftewel van een template met schone installatie te beginnen, of beginnen van een template waarbij kubernetes al is geïnstalleerd, voor het geval dat er problemen mochten zijn met de installatie van microk8s en docker.

#### 4.2.3 Stap 3:

De volgende stap is inloggen in de Admin VM en vervolgens de 2 ansible playbooks in de map `"/home/admin-ccs04/kubernetes/working-playbooks"` uitvoeren. Deze gaan kubernetes en de andere dependencies installeren die nodig zijn. De andere playbook gaat de cluster maken.

Voor de volgende commando's uit te voeren moet je in de bovenstaand vermelde map staan.

- Commando 1: `sudo ansible-playbook test-make-cluster.yml --ask-become-pass`
- Commando 2: `sudo ansible-playbook tes-join-cluster.yml --ask-become-pass`

Om na te kijken of dat de nodes gemaakt zijn voer je volgend commando uit:

*microk8s kubectl get nodes*



#### 4.2.4 Stap 4:

Log in in de Primary VM en ga naar de folder `"/home/admin-ccs04/kubernetes"`. Vervolgens voer je dit commando uit: `microk8s kubectl apply -f storageclass.yaml`

Hierna voer je het script `deploy.sh` uit om de rest van de deployment te verkrijgen. Dit doe je door volgend commando uit te voeren: `./deploy.sh`

#### 4.2.5 Stap 5:

Om na te kijken of dat alles succesvol is uitgevoerd kunt u volgende commando's uitvoeren:

- Commando 1 om de namespace na te kijken:

*microk8s kubectl get namespaces*

- Commando 2 om de pods na te kijken:

*microk8s kubectl get pods -n project-1*

- Commando 3 om de services na te kijken:

*microk8s kubectl get svc -o wide -n project-1*

Voor in te loggen op de VM's, raadpleeg de password vault voor de correcte passwords.

## 5 GEBRUIKTE TOOLS

In het project zijn diverse technologieën gebruikt om het hostingplatform te ontwikkelen. Elk van deze technologieën biedt unieke functies en voordelen die bijdragen aan de veiligheid en betrouwbaarheid van het hostingplatform. In de volgende paragrafen worden deze technologieën nader toegelicht en hun rol in het project beschreven.

### 5.1 Zabbix

#### *WRM Monitoring Tools*

Monitoring tools	Schaalbaarheid (40%)	Visualisatie (25%)	Gebruiksgemak (25%)	Gegevensverzameling (10%)	Totaal Score
Prometheus	9	7	8	7	8
Zabbix	7	7	7	9	7,5

Uit de WRM blijkt dat Prometheus een betere keuze zou zijn, echter is er besloten om Zabbix te gebruiken voor het hostingsproject vanwege de ondersteuning die het biedt.

Zabbix is een monitoring tool die wordt gebruikt om de prestaties en beschikbaarheid van IT-infrastructuurcomponenten, zoals servers, netwerkkapparaten, databases en applicaties, te bewaken. Het verzamelt informatie van verschillende bronnen en geeft deze weer in grafieken, diagrammen en waarschuwingen.

Met Zabbix kunnen beheerders de status van hun IT-infrastructuur bewaken en problemen proactief identificeren en oplossen voordat deze tot uitval leiden. Het biedt ook mogelijkheden voor trendanalyse, capaciteitsplanning en rapportage.

Zabbix kan worden geïnstalleerd op verschillende besturingssystemen en kan worden geconfigureerd om verschillende soorten monitoring uit te voeren, waaronder netwerkmonitoring, servermonitoring, logboekbewaking en applicatiemonitoring. Het ondersteunt ook verschillende protocollen en methoden voor gegevensverzameling, zoals SNMP, JMX, ICMP, SSH, agenten en aangepaste scripts.

Zabbix is een open source monitoring tool en is beschikbaar onder de GPL-licentie. Het biedt een rijke set van functies en kan worden uitgebreid met aangepaste scripts en plug-ins. Zabbix heeft ook een actieve gemeenschap van gebruikers en ontwikkelaars die samenwerken om de tool te verbeteren en nieuwe functies toe te voegen (Das D. 2022).

## 5.2 OpenVPN

OpenVPN is een open source softwaretoepassing die wordt gebruikt voor het opzetten van een veilige en privé virtueel particulier netwerk (VPN) verbinding. Met OpenVPN kunnen gebruikers op een veilige manier verbinding maken met internet en toegang krijgen tot bronnen die op afstand beschikbaar zijn, zoals een bedrijfsnetwerk, bestanden of printers. OpenVPN kan op verschillende besturingssystemen worden geïnstalleerd, zoals Windows, macOS en Linux, en kan worden geconfigureerd om verschillende beveiligingsprotocollen te gebruiken, zoals Transport Layer Security (TLS) en User Datagram Protocol (UDP).

Het gebruik van OpenVPN helpt bij het beschermen van privacy en beveiliging omdat het de communicatie tussen de client en de server versleutelt, waardoor de gegevens worden beschermd tegen afluisteren of manipulatie door kwaadwillende derden. OpenVPN is populair vanwege de flexibiliteit en de mogelijkheid om aangepaste configuraties te maken voor verschillende toepassingen en situaties (OpenVPN. 2022).

## 5.3 Uncomplicated Firewall

*WRM Firewalls*

Firewalls	Bescherming (40%)	Prestaties (25%)	Gebruiksgemak (25%)	Prijs (10%)	Totaal Score
UFW	8	7	9	10	7,9
IPFire	9	8	7	8	7,8
pfSense	10	9	8	9	9,1

De keuze is gevallen op Uncomplicated Firewall (UFW) vanwege de capaciteit die nodig is voor het project.

UFW is een gebruiksvriendelijke front-end voor het beheren van firewall-regels op Linux gebaseerde systemen. Het is ontworpen om het configureren en beheren van een firewall gemakkelijker te maken voor gebruikers die mogelijk geen geavanceerde kennis hebben (Das D. 2022).

## 5.4 Ansible

Ansible is een open-source automatiseringstool die wordt gebruikt voor het vereenvoudigen van IT-taken, het beheren van configuraties en het implementeren van applicaties op grote schaal. Het biedt een eenvoudige en menselijke leesbare taal, genaamd YAML, voor het definiëren van configuratiebestanden en speelboeken. Met Ansible kunnen systeembeheerders en ontwikkelaars de configuratie van systemen, implementaties en orkestratie van taken automatiseren, waardoor het beheer van complexe infrastructuur eenvoudiger wordt.

## **5.5 Poste.io**

Poste.io is een open source softwarepakket waarmee je eenvoudig een eigen e-mailserver kunt opzetten en beheren. Het biedt ondersteuning voor verschillende e-mailprotocollen en -diensten, waaronder SMTP, IMAP en POP3, evenals spamfiltering en virusdetectie. Poste.io wordt vaak gebruikt door ontwikkelaars, startups en kleine bedrijven die op zoek zijn naar een kosteneffectieve en veilige manier om hun e-mail te beheren (complete mail server. (n.d.)).

## **5.6 PhpMyAdmin**

PHPMyAdmin is een webgebaseerde applicatie die wordt gebruikt voor het beheren van MySQL-databases. Het biedt een grafische gebruikersinterface waarmee gebruikers gemakkelijk databases, tabellen, velden, gebruikers en rechten kunnen maken, bewerken en verwijderen. PHPMyAdmin kan worden gebruikt om query's uit te voeren, back-ups te maken en gegevens te importeren en exporteren. Het is beschikbaar in verschillende talen en kan worden geïnstalleerd op verschillende platforms. PHPMyAdmin is een veelgebruikte tool voor webontwikkelaars en systeembeheerders die met MySQL-databases werken (Contributors, P. (n.d.)).

## **5.7 Discord**

Ons helpdeskteam maakt gebruik van Discord als communicatiekanaal om klanten te ondersteunen bij vragen en problemen. Daarnaast hebben we ook een FAQ-sectie op ons hostingsplatform die via Discord toegankelijk is. Op deze manier kunnen klanten gemakkelijk en snel antwoorden vinden op veelgestelde vragen en hebben ze toegang tot relevante informatie die ze nodig hebben om hun hostingervaring soepel te laten verlopen.

## **5.8 Duplicati**

Duplicati is een back-upclient die veilig gecodeerde, incrementele, gecomprimeerde externe back-ups van lokale bestanden op cloudopslagservices en externe bestandsservers opslaat. Duplicati ondersteunt niet alleen verschillende online back-upservices zoals OneDrive, Amazon S3, Backblaze, Rackspace Cloud Files, Tahoe LAFS en Google Drive, maar ondersteunt ook elke server die SSH/SFTP, WebDAV of FTP ondersteunt. (Wikipedia contributors, 2022)

## **5.9 SCAP Workbench**

SCAP Workbench is een grafisch hulpprogramma dat een gemakkelijke manier biedt om algemene oscap-taken uit te voeren. Met de tool kunnen gebruikers configuratie- en kwetsbaarheidsscans uitvoeren op een enkel lokaal of extern systeem en het systeem herstellen op basis van een bepaald Extensible Configuration Checklist Description Format (XCCDF)- of SDS-bestand. Workbench kan rapporten in verschillende formaten genereren, samen met de resultaten van systeemscans.

Workbench biedt een eenvoudige manier om XCCDF-configuratiebestanden te wijzigen zonder de betreffende XCCDF-bestanden te wijzigen. Deze tool biedt een grafische manier om XCCDF-elementen in of uit te schakelen. Uw wijzigingen kunnen worden opgeslagen als XCCDF-aanpassingsbestanden. (*SCAP Workbench* | *OpenSCAP portal*, z.d.)

## **BESLUIT**

Het hostingsproject bleek een waardevolle ervaring te zijn die ons een diepgaand inzicht gaf in het opzetten van een hostingplatform. Eerst wisten we niet precies waar we moesten beginnen en stonden we voor de uitdaging om de juiste aanpak te vinden. Door nauw samen te werken en expertise te delen, hebben we echter de nodige stappen kunnen zetten om onze doelen te bereiken.

Tijdens het project hebben we niet alleen nieuwe tools zoals Zabbix en Duplicati leren kennen, maar ook hoe we onze bestaande kennis konden toepassen op nieuwe situaties. We hebben onze vaardigheden aangescherpt met bekende programma's zoals Kubernetes en FastAPI om de applicaties in een hostingomgeving te begrijpen. Als groep hebben we samen gekeken naar welke aanpak het beste zou werken voor het project. We hebben meerdere tests uitgevoerd om de juiste software te vinden die naadloos in ons systeem zou passen. We werken momenteel aan de implementatie van ons project met de software van uw keuze, met als doel een optimaal resultaat.

Over het algemeen was het uitvoeren van het groepswerkproject erg nuttig en uitdagend. Het was een eer om met zo'n geweldige groep te werken. We zijn trots op wat we tot nu toe hebben bereikt en kijken ernaar uit om onze nieuwe kennis en vaardigheden toe te passen op toekomstige projecten.

## BIBLIOGRAFIE

- ClamAVNet. (n.d.). <https://www.clamav.net/>
- Contributors, P. (n.d.). phpMyAdmin. phpMyAdmin. <https://www.phpmyadmin.net/>
- Das, D. (2022, August 8). The 10 Best Free Linux Firewall Tools. MUO. <https://www.makeuseof.com/best-free-firewall-linux/>
- Fisher, T. (2023, March 1). *The 8 Best Commercial Backup Software Programs of 2023*. Lifewire. <https://www.lifewire.com/raid-levels-2624711>
- Gillis, A. S., Sullivan, E., & Posey, B. (2021, August 18). *RAID (redundant array of independent disks)*. Storage. <https://searchstorage.techtarget.com/definition/RAID>
- how-to-enable-scp. (2021). Inomotionhosting. Retrieved March 27, 2021, from <https://www.inomotionhosting.com/support/website/ssh/how-to-enable-scp-in-cpanel/>
- Loshin, P. (2022). Network File System (NFS). Enterprise Desktop. <https://www.techtarget.com/searchenterprisedesktop/definition/Network-File-System>
- MySQL. (n.d.). <https://www.mysql.com/>
- NetApp. (2019). What Is Persistent Storage? How Does It Work? | NetApp. <https://www.netapp.com/data-management/max-memory-accelerated-data/persistent-storage/#:~:text=Persistent%20storage%20is%20any%20data,referred%20to%20as%20nonvolatile%20storage.>
- OpenVPN. (2022, April 20). What Is OpenVPN? | OpenVPN. <https://openvpn.net/faq/what-is-openvpn/>
- Patch Management Using Automation | Puppet by Perforce.* (n.d.). <https://www.puppet.com/why-puppet/use-cases/patch-management>
- Poste.io ~ complete mail server. (n.d.). <https://poste.io/>
- Production-Grade Container Orchestration. (n.d.-b). Kubernetes. <https://kubernetes.io/>
- Raid setup guide.* (2020). Pupetsystems. Retrieved March 6, 2020, from [https://www.pugetsystems.com/help/RAID\\_Setup\\_Guide.php](https://www.pugetsystems.com/help/RAID_Setup_Guide.php)
- SCAP Workbench | OpenSCAP portal. (z.d.). <https://www.open-scap.org/tools/scap-workbench/>
- Sophos. (2023, March 2). Endpoint Protection with Sophos Intercept X. SOPHOS. <https://www.sophos.com/en-us/products/endpoint-antivirus.aspx>
- Using Patch Management.* (n.d.). Puppet Enterprise Guide. <https://puppet-enterprise-guide.com/theory/using-patch-management.html#how-does-it-work>

VMware Remote Console for vSphere. (n.d.).  
<https://docs.vmware.com/en/VMware-Remote-Console/12.0/com.vmware.vsmrc.vsphere.doc/GUID-BD88AB4A-6FF4-44A2-AEC2-F5BF8F932085.html>

VMware vSphere | Enterprise Workload Platform. (2023, March 21). VMware.  
<https://www.vmware.com/products/vsphere.html>

Wat is een API? (n.d.). Apple Support. <https://support.apple.com/nl-nl/guide/shortcuts-mac/apd2e30c9d45/mac#:~:text=Een%20API%2C%20of%20een%20Applicatie,van%20de%20voorziening%20kan%20ontvangen.>

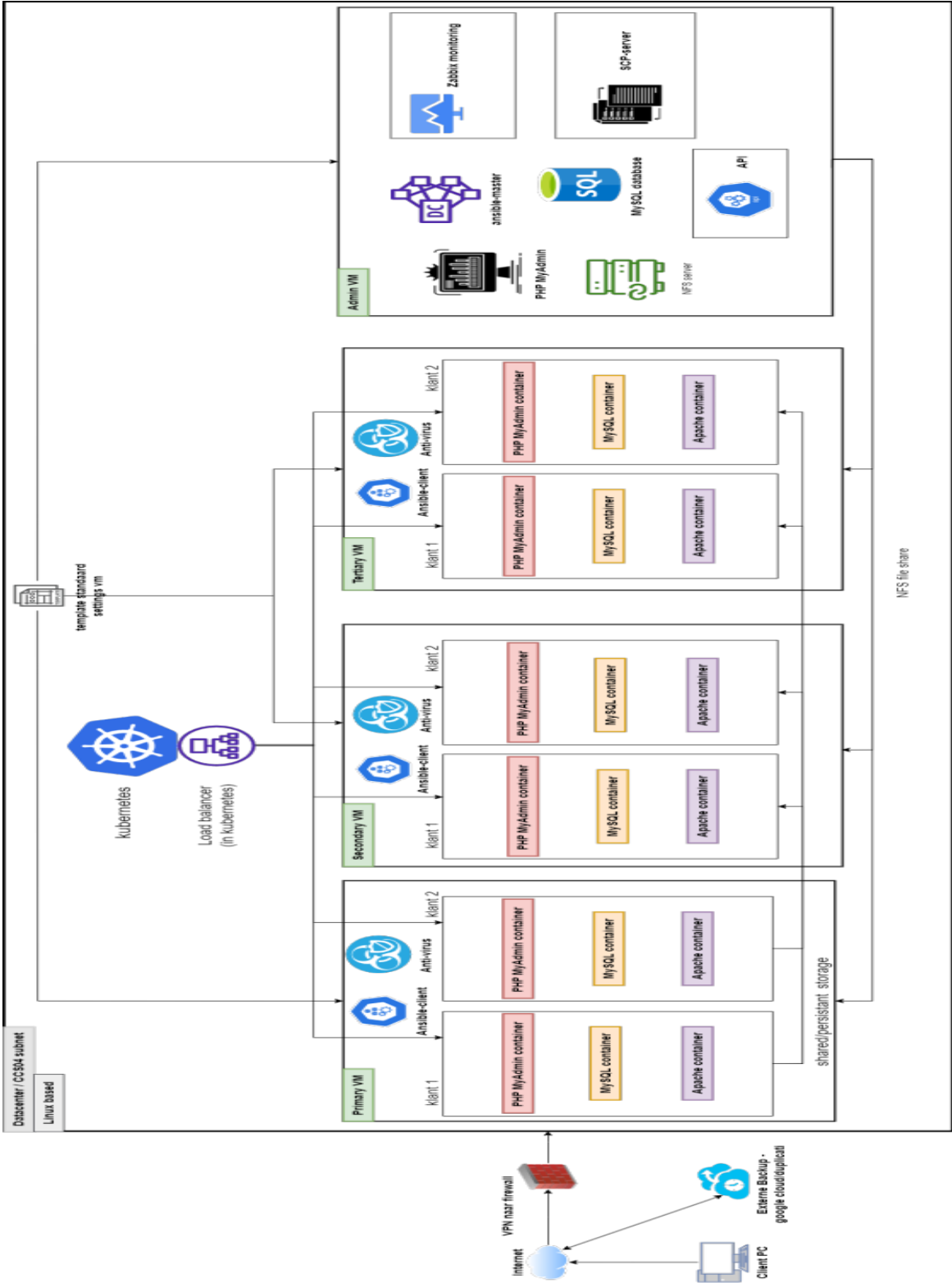
Wikipedia contributors. (2022). Duplicati. Wikipedia.  
<https://en.wikipedia.org/wiki/Duplicati>

Yusuf, N. (2023, 6 januari). Wat is sFTP. . . AccessPay legt uit hoe dit beveiligingsmiddel werkt. AccessPay. <https://accesspay.com/nl/knowledge-hub/veiligheid/wat-is-sftp-en-hoe-werkt-het/#:~:text=SFTP%20staat%20voor%20%22Secure%20File%20Transfer%20Protocol%22.>

Zelleke, L., & Zelleke, L. (2023, February 28). *The Best Server Patch Management Tools for 2023*. Comparitech. <https://www.comparitech.com/net-admin/best-server-patch-management-tools/>



BIJLAGEN



Schematische voorstelling van het systeem uitvergroet

## VERKLARENDE WOORDENLIJST

Afkorting	Woord voluit	Betekenis
ICMP	Internet Control Message Protocol	ICMP is een protocol dat gebruikt wordt om fouten en statusberichten tussen netwerkkapparaten te verzenden, met name tussen routers en hosts.
JMX	Java Management Extensions	JMX is een Java-technologie die gebruikt wordt voor het beheer en de monitoring van Java-toepassingen en Java virtuele machines (JVM's).
KPI	Key Performance Indicators	KPI's zijn meetbare waarden die worden gebruikt om te beoordelen of een organisatie, afdeling of individu presteert volgens de gestelde doelen en doelstellingen.
RAID	Redundant Array of Independent Disks	RAID combineert meerdere harde schijven om betrouwbaarheid, prestaties en/of capaciteit te verbeteren.
SNMP	Simple Network Management Protocol	SNMP is een protocol dat gebruikt wordt om netwerkkapparaten op afstand te monitoren, te configureren en te beheren.
SSH	Secure Shell	SSH is een netwerkprotocol dat gebruikt wordt voor beveiligde communicatie tussen twee computers.
XCCDF	Extensible Configuration Checklist Description Format	Het Extensible Configuration Checklist Description Format is een XML-formaat waarin veiligheidscontrolelijsten, benchmarks en configuratiedocumentatie worden gespecificeerd.