**Nomios**

Excelsiorlaan 89.

1930 Zaventem.

**Van Mol Gerrit**

Professionele Bachelor Elektronica-ICT, afstudeerrichting ICT

Academiejaar 2021/2022

**Platform voor automatisatie van firewall configuraties en deployment**

**Stagegegevens**

**Stagiair**

Gerrit Van Mol

**Opleiding**

Elektronica-ICT, afstudeerrichting ICT

**Academiejaar**

2021/2022

**Stageperiode**

31/01 - 30/05/2022

**Stagebegeleider**

Stijn Vanbinnebeeck

**Stageplaats**

Nomios N.V. Belgium

Excelsiorlaan 89.  
1930 Zaventem.

**Mentor(en)**

Sven Sanders

**Automation platform for quick deployment of firewalls**

G. Van Mol

The purpose of this project is automating deployment of firewalls. It will simplify the job of a technician that normally installs and deploys these devices. Automating firewall deployment prevents long and difficult procedures that would otherwise take a long time to get up and running with the device.

The first chapter of this report will cover research about the best possible frameworks available, afterwards a comparison is made between frameworks. Finally, one of these frameworks will selected and used to create a platform. Based on functionality requests, the platform is made to perform the most important tasks to make automated deployment of firewalls possible.

In conclusion this platform will provide a workflow that is quick and ease of use. It will eliminate repetitive tasks and long configuration hours. Eventually the platform will be saving time of technicians and improving efficiency of the company.

Keywords: Platform, automation, framework, time saving

Voorwoord

Voor het behalen van een diploma in het derde jaar en Elektronica-ICT, is het maken van een bachelorproef een vereiste. Deze bundel is het schriftelijke verslag ervan.

Deze bachelorproef is het resultaat van de kennis en vaardigheden die ik heb verzameld zowel op school als op mijn stageplaats. Het was een leerrijke uitdaging, waarbij ik ook heel wat nieuwe ervaringen opdeed. Bij de uitwerking van de bachelorproef kreeg ik tips en ondersteuning van mijn docenten, stagebegeleiders, familie en vrienden. In de eerste plaats zou ik graag de directie van de co-hogeschool Odisee willen bedanken voor de kans die ik kreeg er mijn opleiding te volgen.

Daarnaast wil ik alle docenten bedanken voor hun fantastische begeleiding door heen de jaren van mijn opleiding, en Sven Sanders voor de fantastische begeleiding tijdens mijn stageperiode. In bijzonder wil ik de heer Johan Donné mijn docent applied-programming, data security en tal van andere vakken bedanken, om onder zijn deskundig toezicht theoretische informatie en praktische ervaring te verschaffen.

Ook wil ik graag de mensen bedanken die niet in mij geloofden voor het versterken van mijn motivatie en doorzettingsvermogen.

Tot slot gaat mijn dank naar mijn ouders Youri en Kathleen Van Mol, Ilona Van Mol en mijn grote zus Laure Van Mol voor hun aanmoediging en ondersteuning.

Met het rapport hoop ik de lezer een boeiend en leerrijk verslag te brengen van mijn bachelorproef.

Gerrit Van Mol

Kapelstraat 57 Lochristi, Juni 2022

Inhoud

[Codefragmentlijst 1](#_Toc104240729)

[Figurenlijst 2](#_Toc104240730)

[Tabellenlijst 3](#_Toc104240731)

[Afkortingenlijst 4](#_Toc104240732)

[Begrippenlijst 5](#_Toc104240733)

[Inleiding 6](#_Toc104240734)

[1 Analyse opstelling 7](#_Toc104240735)

[1.1 Huidige werkwijze 7](#_Toc104240736)

[1.2 Snellere werkwijze 8](#_Toc104240737)

[2 Platform structuur 9](#_Toc104240738)

[2.1 Platform logica 9](#_Toc104240739)

[3 Framework 12](#_Toc104240740)

[3.1 Django geschiedenis 12](#_Toc104240741)

[3.2 Django framework 13](#_Toc104240742)

[3.3 Django architectuur (MVC/MVT) 13](#_Toc104240743)

[3.3.1 Models 14](#_Toc104240744)

[3.3.2 Views 15](#_Toc104240745)

[3.3.3 Templates 16](#_Toc104240746)

[3.1 Netwerk diagram 19](#_Toc104240747)

[3.3.2 Conclusie framework verschillen 19](#_Toc104240748)

[5 Python virtuele omgevingen 20](#_Toc104240749)

[5.2 Voordelen Python virtuele omgevingen 21](#_Toc104240750)

[6 FortiManager 22](#_Toc104240751)

[6.1 Zero Touch provisioning 22](#_Toc104240752)

[6.2 Network automatisatie 23](#_Toc104240753)

[6.3 Central management 23](#_Toc104240754)

[7 Front-end frameworks 23](#_Toc104240755)

[8 Ansible 24](#_Toc104240756)

[9.1 Ansible toepassingen 24](#_Toc104240757)

[9.1.1 Provisioning 24](#_Toc104240758)

[9.1.2 Configuratie management 25](#_Toc104240759)

[9.1.3 Applicatie deployment 25](#_Toc104240760)

[9.2 Continuous delivery 26](#_Toc104240761)

[9.3 Conclusie Ansible 26](#_Toc104240762)

[10 Jenkins 27](#_Toc104240763)

[10.1 Jenkins minimumvoorwaarden 28](#_Toc104240764)

[10.2 Jenkins best practices 29](#_Toc104240765)

[11 Praktische uitwerking 30](#_Toc104240766)

[12 Hosting 31](#_Toc104240767)

[13 Handleiding 32](#_Toc104240768)

[Conclusie 33](#_Toc104240769)

[Nawoord 34](#_Toc104240770)

[Literatuurlijst 35](#_Toc104240771)

[Bijlagenoverzicht 37](#_Toc104240772)

[Bijlage 1: Platform flowchart 38](#_Toc104240773)

[Bijlage 2: Relatie diagram 39](#_Toc104240774)

[Bijlage 3: Netwerk diagram 40](#_Toc104240775)

# Codefragmentlijst

[Codefragment 1: Django model code voorbeeld [5] 14](#_Toc104240723)

[Codefragment 2: Voorbeeld inhoud views.py (datetime functie) [7] 15](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240724)

[Codefragment 3: urls.py voorbeeld URL [7] 16](#_Toc104240725)

[Codefragment 4: views.py met context dictionary [8] 17](#_Toc104240726)

[Codefragment 5: Template voorbeeld 17](#_Toc104240727)

[Codefragment 6: Folder structuur virtuele omgeving 22](#_Toc104240728)

# Figurenlijst

[Figuur 1: Huidig “tijd-consumerend” proces 7](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240848)

[Figuur 2: Nieuwe werkwijze a.d.h.v. platform “grafische interface” 8](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240849)

[Figuur 3: Django platform pagina structuur 10](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240850)

[Figuur 4: Django tijdlijn [18] 12](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240851)

[Figuur 5: Django MVT-werking [25] 13](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240852)

[Figuur 6: Visuele voorstelling model in databank [5] 14](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240853)

[Figuur 7: Logica achter view (MVT) [6] 15](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240854)

[Figuur 8: View functie resultaat [7] 16](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240855)

[Figuur 9: Template output voorbeeld 17](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240856)

[Figuur 10: Python virtuele omgevingen visueel voorbeeld [9] 20](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240857)

[Figuur 11: Zero touch provisioning (FortiManager) [10] 22](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240858)

[Figuur 12: Ansible stage levenscyclus [12] 24](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240859)

[Figuur 13: Zero downtime applicatie-update [20] 25](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240860)

[Figuur 14: Jenkins master-slave architectuur [15] 27](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240861)

[Figuur 15: Jenkins visuele werking [23] 28](file:////Users/gerritvanmol/Documents/Odisee_repos/automationplatform/Documents/Papers/V3.3_AutomatisatiePlatform-GerritVanMol.docx#_Toc104240862)

# Tabellenlijst

[Tabel 1: Jenkins minimum hardware voorwaarden 26](#_Toc104131043)

[Tabel 2: Jenkins minimumsoftware voorwaarden [17] 27](#_Toc104131044)

# Afkortingenlijst

ZTP Zero Touch Provisioning

DB Database

MySQL My Structured Query Language

HTML Hyper Text Markup Language

UI User Interface

MVC Model View Controller

MVT Model View Template

IDE Integrated Development Environment

SSL Secure Sockets Layer

API Application Programming Interface

JSON JavaScript Object Notation

ORM Object Relational Mapper

CRUD Create, Read, Update, Delete

CSRF Cross-site Request Forgery

XSS Cross-site Scripting

IDE Integrated Development Environment

DVCS Distribution Version Control System

CI Continuous Integration

CD Continuous Development

CLI Command Line Interface

CSRF Cross-Site Request Forgery

ORM Object Relational Mapper

RAM Random-access memory

# Begrippenlijst

Variabele Element van bepaald type dat een tijdelijke waarde bijhoudt

Redundant Herhaling/dubbele gegevens

Property Eigenschap van een object

Provisioning Het proces voor opstellen van IT-infrastructuur

Batch processing Runnen van “jobs” dat geen gebruikers interactie nodig heeft

Staging Een replica van de productie omgeving om testen uit te voeren

# Inleiding

Dit project wordt een platform dat de opzet van firewalls automatiseert. Momenteel werkt de field engineer op een tijd consumerende manier waarbij alle configuraties van elk toestel handmatig gebeuren. Het platform geeft de technieker(s) de mogelijkheid snel en efficiënt meerdere toestellen te plaatsen binnen een bepaald tijdsframe. Aan de hand van zero touch provisioning zullen de techniekers weinig of geen voorkennis moeten hebben om een firewall toestel te plaatsen en operationeel te krijgen. Als een field engineer on-site is en geen basis configuratie aanwezig is op het toestel kan deze worden gedownload van het platform naar een USB-stick. Eenmaal de configuratie op de stick staat, kan deze worden overgezet naar de firewall. Met de basis configuratie aanwezig kan het platform verdere connectie maken met de firewall, om updates uit te sturen met verdere configuratie.

Er wordt onderzocht met welke technologieën en protocollen het platform best werkt om communicatie met de externe apparaten mogelijk te maken. Of het mogelijk is apparaten van verschillenden vendoren (tegelijk) te configureren. Hoe configuraties het efficiëntst kunnen worden beheerd. Of het mogelijk is meerdere connecties tegelijk tot stand te brengen tussen het platform en toestellen ter plekke. De opdrachtgever vraagt om gebruik te maken van het Django framework, Fortimanager, Ansible en/of alternatieven. Andere frameworks en automatisatie engines worden daarnaast ook onderzocht.

Bij deze opdracht heeft het bedrijf een aantal eisen. Deze omvatten dan vooral een volledig functionele login voor administrator en engineers. Alsook had de opdrachtgever graag een geautomatiseerde basis set-up van een firewall waarbij connectiviteit zoals een ping naar Fortigate-manager functioneert zoals het hoort.

Daarnaast zijn er geen vooropgestelde beperkingen of technische randvoorwaarden waaraan moet worden voldaan.

# 1 Analyse opstelling

In volgende subhoofdstukken worden processen en relatie diagrammen beschreven die bij aanvang van het project worden gehandhaafd. Met behulp van onderzoek en bevraging is deze flowchart opgebouwd. Om het huidige proces te gaan optimaliseren, zal er een deel worden geautomatiseerd. Ook voor de geoptimaliseerde werkwijze is er een flowchart opgemaakt om het proces visueel te gaan verduidelijken. Deze automatische processen zijn vastgelegd en vooropgesteld door het externe bedrijf (bank) waar deze opdracht voor uitwerkt, wordt. Voor welke bank dit project wordt uitgewerkt zal niet worden beschreven aangezien dit in conflict komt met vertrouwelijke informatie die niet kan/mag worden gedeeld. Op vlak van infrastructuur en hosting van het platform is de volledige vrijheid gegeven. Hosting van het platform wordt later in detail besproken.

## 1.1 Huidige werkwijze

Tijdens een interview kon er een huidige werkwijze worden vastgesteld voor deployment van firewalls. Waarbij duidelijk werd dat binnen het bedrijf deze werk structuur is vooropgesteld en als standaard wordt aangenomen voor het registreren en uitrollen van firewalls. Daarnaast werd ook duidelijk dat de engineer een minimum aantal stappen (Figuur 1: Huidig “tijd-consumerend” proces) moet ondernemen, voor een firewall van een labo naar deployment omgeving (on-site) kan worden verhuisd. Deze stappen moeten steeds worden herhaald voor elke individueele firewall, wat heel tijd-consumerend is.

Diagram

Description automatically generated

Figuur : Huidig “tijd-consumerend” proces

## 1.2 Snellere werkwijze

Omdat het configureren en testen van firewalls een steeds herhalend proces is, kan er een deel worden geautomatiseerd en versoepeld waardoor dit proces vlotter zal kunnen verlopen. Aan de hand van een platform met een grafische interface, worden handmatige stappen waar origineel steeds een terminal voor nodig was overbodig. Aan de hand van dit platform moet de engineer niet meer wisselen tussen terminal vensters en dergelijke. Dit is veel gebruiksvriendelijker en zorgt voor efficiëntie van gepresteerde werk uren. Alle configuraties of aanpassingen zal een engineer nu kunnen vanuit deze nieuwe interface.

Diagram

Description automatically generatedIn onderstaande illustratie (Figuur 2: Nieuwe werkwijze a.d.h.v. platform “grafische interface”) zijn de zo goed als alle stappen van het voorgaande proces in het nieuwe platform verwerkt.

Figuur : Nieuwe werkwijze a.d.h.v. platform “grafische interface”

# 2 Platform structuur

Omdat meerdere stappen worden samengesmolten voor het configureren van firewalls, zal aan de hand van het Django framework een grafische interface worden gemaakt om dit mogelijk te maken. Hoe dan ook voor een platform kan worden uitgewerkt moet er een structuur zijn van de pagina’s die geen gebruiker kan raadplegen, wat hun functie is en hoe ze aan elkaar hangen. In dit hoofdstuk zal elk deel zorgvuldig worden overlopen, met behulp van onderstaande figuur (Figuur 3: Django platform pagina structuur) waarbij de logica start vanaf de gebruiker (bovenaan) tot aan elk eindpunt/pagina.

## 2.1 Platform logica

Voor iemand eventuele data kan raadplegen op een van de beschikbare pagina’s moet er eerst worden ingelogd. Accounts worden aangemaakt door de administrator van het platform. Er mag geen mogelijkheid zijn om zomaar te registreren, door de eenvoudige reden dat dit erg onveilig kan zijn voor een publieke applicatie, waarbij iedereen zomaar gevoelige data van een bedrijf kan raadplegen en aanpassen. De administrator van het platform maakt aan de hand van een vooropgestelde lijst aan gekwalificeerde personen een aantal accounts aan. Dit kan de administrator aan de hand van een admin-paneel dat wordt voorzien door het Django framework zelf. Het account van de administrator zelf wordt aangemaakt bij creatie van de databank waar het platform zijn data op zal bewaren.  
  
Eenmaal de gekwalificeerde gebruiker is aangemeld krijgt hij of zij een overzicht pagina te zien waarbij men minimum onderstaande puntjes kan raadplegen.

* MGMT Interface/IP
* State (active/inactive)
* Hostname
* Site/premise code
* Software version
* Vendor/type

De vernoemde puntjes geven een snel overzicht van alle beschikbare firewalls en hun basis configuratie. Als er een aanpassing/update moet gebeuren op een van de reeds beschikbare firewalls op de overzichtspagina, kan er worden verder gegaan door op een firewall te klikken. De gebruiker zal worden omgeleid naar de firewall configuratie pagina. Daar kan de gebruiker de gekozen firewall aanpassen en/of verwijderen. Indien er nog geen firewalls beschikbaar zijn in het overzicht kan er via dezelfde pagina (firewall configuratie) een firewall worden toegevoegd. Aan de hand van een vooropgestelde CSV/Excel bestand worden firewall configuratie parameters ingeladen om een basis configuratie te genereren voor de specifieke firewall. De nieuw gegenereerde firewall configuratie wordt opgeslagen op de server van het platform. Later kan de gebruiker kiezen om het bestand te downloaden en manueel te uploaden naar de firewall, of indien mogelijk automatisch de nieuwe configuratie te pushen naar de firewall(s) waarvoor de specifieke configuratie werd gecreëerd. Wanneer een configuratie wordt aangemaakt krijgt de gebruiker een succes status terug, zo weet de gebruiker zeker dat het genereren van de configuratie succesvol was of niet.

Diagram

Description automatically generated

Figuur : Django platform pagina structuur

Ook op de firewall configuratie pagina kan de gebruiker een back-upconfiguratie van de gekozen firewall(s) opvragen. Hierbij worden de op dat moment actieve firewall configuraties opgevraagd, en terug opgeslagen op de server van het platform. Indien er reeds een back-up is gecreëerd van een specifieke firewall zal deze worden overschreven. Opnieuw, om de gebruiker op de hoogte te brengen of de back-up succesvol is aangemaakt, wordt er terug een succes status bijgehouden en weergegeven.

Soms is het mogelijk dat een firewall moet her-labeld/verplaatst worden naar een andere site. Daarom is er de mogelijkheid om firewall hostnames en adres objecten aan te passen. Een hostname kan rechtstreeks worden aangepast op de interface van het platform. Adres objecten daarentegen moeten terug worden voorzien aan de hand van een CSV/Excel bestand. Dit bestand wordt geüpload en opgeslagen op de server van het platform. De gedefinieerde waarden worden uit het Csv-bestand opgehaald en omgevormd in een JSON-bestand zo dat het via een API-call of alternatief kan worden geüpload naar de firewall. Na een hostnaam of adres object update, krijgt de gebruiker opnieuw een succes statusmelding. Op die manier weet de gebruiker met zekerheid dat de aanpassing is doorgevoerd.

Wanneer een firewall voor het eerst wordt geconfigureerd en toegevoegd aan het platform moet de engineer zeker zijn dat het apparaat operationeel is, daarom is er een pagina voorzien waar dit kan worden gecontroleerd. Op de “operational” pagina wordt er een overzicht gecreëerd van de actief en inactieve firewalls. De staat van een firewall wordt bepaald afhankelijk van een ping (ICMP-protocol). Wanneer er een antwoord is op de ping zal op het platform worden weergegeven dat het toestel actief, ook wanneer geen reactie is op de verzonden ping zal er inactief worden weergegeven. Een firewall dat op inactief staat kan twee dingen betekenen;

1. De firewall is net toegevoegd en dus nog niet getest
2. De firewall heeft mogelijks een fout en kan niet (meer) worden getest

Wanneer een test wordt uitgevoerd krijgt de gebruiker een melding op de grafische interface die aangeeft of de ping is uitgestuurd of niet. Als dan de status van die specifieke firewall op inactief staat, en zijn status niet update weet de engineer dat het toestel moet worden gecontroleerd op configuratie fouten of andere oorzaken.

# 3 Framework

Om het platform uit te werken is er een framework nodig dat kan worden gebruik als fundamenteel middelpunt om verder op te bouwen. Aan de hand van dit framework zal het platform worden uitgewerkt. Er werd voor dit project het Django framework opgelegd door de opdracht gever waarbij een potentiële overweging voor eventuele andere frameworks wordt uitgesloten.

Hoe dan ook zal er in dit hoofdstuk worden terug gekeken op wat de geschiedenis is van Django, waar het vandaan komt en hoe het is ontstaan, en zal kort vergeleken worden met andere populaire frameworks met hun voor-/nadelen. Op die manier wordt er een beter zicht gecreëerd over de verschillende frameworks en bij welke toepassingen ze best ingezet kunnen worden.

## 3.1 Django geschiedenis

Origineel is het Django framework ontstaan en ontworpen door “Lauwrence journal world” in 2003 [1]. In 2004 werd het een open-source project en kreeg het de naam Django. De naam wordt uitgesproken als “Jang-oh” waarbij de “D” stil is [2]. Het is vernoemd achter de jazzgitarist Django Reinhardt. Een web team dat nieuwswebsites beheerden maakten gebruik van veel herhaaldelijke code en ontwerpen. Deze herhaaldelijke code werd uiteindelijk uitgewerkt als een algemeen web development framework. Het framework werd steeds meer gekend en verbeterd waarbij de eerste officiële versie (v0.90) werd uitgerold in november 2005. Ondertussen zitten ze aan versie (v3.2.12) sinds december 2021. Bij elke versie worden mogelijke problemen weggewerkt en features toegevoegd zoals nieuwe templates, database typen, etc. Django is intussen een veelzijdig framework dat de mogelijkheid biedt meerdere soorten websites te creëren.



Figuur : Django tijdlijn [18]

## 3.2 Django framework

Django is een gratis open-source framework, gemaakt om te focussen op het ontwikkelen van onderhoudbare websites. Sinds 2005 geeft het de mogelijkheid om snel maar ook veilige en schaalbare websites te ontwikkelen in de Python programmeertaal. Het framework heeft een zeer actieve gemeenschap waar elke ontwikkelaar steeds terecht kan voor vragen. Daarnaast is de documentatie ook zeer uitgebreid en goed onderhouden. Het grootste voordeel aan dit framework is dat het over een rijke feature set beschikt van meer dan 10 000 packages. Handig ingebouwde packages zijn bijvoorbeeld, API’s, CMS, UA, CAPTCHA-ondersteuning en form validatie. Django is gekend voor de “batteries-included” filosofie. Het principe achter “batteries-included” is dat de algemene functionaliteit voor het bouwen van een web-applicatie zou moeten mee geleverd worden in het framework, in plaats van aparte bibliotheken die deze functionaliteit leveren. Daarnaast wordt ontwikkeling ondersteund door een non-profit organisatie, wat de ontwikkeling vooruit helpt en bedrijven/individuen zekerheid geeft over de levensduur en vooruitzichten van het framework [3] [4].

## 3.3 Django architectuur (MVC/MVT)

Het makkelijke aan Django is dat alle achterliggende delen al inbegrepen zijn, zo is het niet nodig om een backend te maken aan de hand van API’s, JavaScript, etc. Omdat Django gebruik maakt van het MVC/MVT-software design is het niet meteen nodig om gebruik te maken van alternatieve backend opties. Dit software design is een collectie van drie belangrijke componenten Model, View en Template (controller). In andere programmeertalen is de “Model View Controller” (MVC) architectuur een gekende standaard. Het Django framework daar in tegen beschikt over de “Model View Template” architectuur. Het grootste verschil zit in de “view” deel van beide architecturen.

Bij Django wordt de view van MVC gezien als de data logica, en beschrijft de data die wordt gepresenteerd aan de gebruiker. In tegenstelling tot hoe iets wordt gepresenteerd naar de gebruiker. De view beschrijft dus welke data wordt vertoond, en niet hoe de data kan wordt vertoond aan de gebruiker.

Diagram

Description automatically generatedHet is dus logisch om de inhoud te scheiden van de presentatie, dat is waar templates een rol spelen. In Django beschrijft een "view" welke gegevens worden gepresenteerd, maar een view draagt hier dus data over naar een template, die op zijn beurt beschrijft hoe de gegevens worden gepresenteerd. De controller in Django is het framework op zich [2]. In onderstaande afbeelding (Figuur 5: Django MVT-werking [25]) kan men een visuele weergave terugvinden van hoe de MVT-structuur in zijn werk gaat.

Figuur : Django MVT-werking [25]

### 3.3.1 Models

Aan de slag gaan met een gewone databank is vrij complex en omslachtig, omdat de ontwikkelaar moet weten welke vele query's nodig zijn voor het aanmaken, verwijderen of aanpassen van tabellen en/of andere database-gerelateerde zaken. Modellen brengt de complexiteit van SQL (Structured Query Language) databanken naar beneden door taken en tabellen te structureren in modellen.

Een model is een soort interface voor de data, en is verantwoordelijk voor het onderhouden van de data. Het zorgt voor een logische structuur achter de volledige applicatie en wordt gerepresenteerd door de databank (relationele databanken zoals MySQL of Postgres). Een model in Django wordt gebruikt voor het maken van tabellen, data velden en eventuele extra parameters. Algemeen zal elk model in Django een enkele tabel voorstellen in de databank.

Aan de hand van het meegeleverde admin-paneel van Django, kunnen gemakkelijk datavelden worden aangemaakt, aangepast, verwijderd of opgehaald van een model samen met vele bijkomende operaties. Django modellen bieden simpliciteit, consistentie en version-control. De basis eigenschappen van een model zijn als volgt;

* Elk model is een Python klasse dat subklassen creëert van het “django.db.models.Model”
* Elk attribuut van een model representeert een database veld
* Geeft een automatisch gegenereerd/toegang tot de database

Een Django model voorbeeld kan in onderstaand codefragment worden geraadpleegd.



Codefragment 1: Django model code voorbeeld [5]

Diagram

Description automatically generatedWanneer een model is aangemaakt in Django moet de applicatie voor opstart worden gemigreerd. Wat betekend dat alle modellen dat nog niet aanwezig waren als tabel in de databank op dat moment worden aangemaakt. In onderstaande afbeelding (Figuur 6: Visuele voorstelling model in databank [5]) is een visuele voorstelling van wat er gebeurt als een model wordt gemigreerd naar de databank [5].

Figuur : Visuele voorstelling model in databank [5]

### 3.3.2 Views

Django views zijn één van de meest belangrijke onderdelen van de MVT-structuur. Zoals eerder al werd vermeld, is een view een methode dat web requests verwerkt en een antwoord terug geeft. Dit antwoord kan HTML-inhoud, redirect, 404 error of eender wat zijn dat een web browser kan afbeelden.

Django views zijn een deel van de interface, en renderen de HTML/CSS/JavaScript in een template om uiteindelijk weer te geven op een webpagina [6]. Onderstaande afbeelding (Figuur 7: Logica achter view (MVT) [6]) geeft visueel beeld van hoe de logica van een view in elkaar zit.

Figuur 7: Logica achter view (MVT) [6]

Wanneer er een project is aangemaakt is er een standaard aantal bestanden die worden meegeleverd. Eén van die bestanden is de “views.py”, daarin worden methoden gedefinieerd die de binnenkomende web requests verwerken. In onderstaand codefragment wordt de inhoud van een voorbeeld “views.py” bestand weergegeven.

Codefragment : Voorbeeld inhoud views.py (datetime functie) [7]

Om de gedefinieerde view functie weer te geven op een webpagina moet de functie worden opgeroepen in het “urls.py” bestand samen met een passende URL. Wanneer de gebruiker dan die specifieke URL bezoekt, krijgt hij of zij data terug gedefinieerd in de voorgaande functie. Onderstaand codefragment roept de functie “CurrentDate\_view” op uit het “views.py” bestand (Codefragment 2: Voorbeeld inhoud views.py (datetime functie)).  
****

Codefragment 3: urls.py voorbeeld URL [7]

Wanneer de thuispagina (“#”) wordt bezocht door de gebruiker, zal de voorgaand vermelde functie (“CurrentDate\_view”) worden opgeroepen en gelijkaardig resultaat als onderstaande afbeelding zal worden weergegeven.

Figuur 8: View functie resultaat [7]

### 3.3.3 Templates

Een template bestaat uit statische inhoud zoals standaard-HTML/CSS en JavaScript in een HTML-bestand. Naast de standaard-HTML en CSS bestaat een template ook speciale syntax, die beschrijft hoe dynamisch inhoud wordt geïmplementeerd. Het Django framework genereert en behandeld dynamisch webpagina's zichtbaar voor de eindgebruiker. Aan de hand van de Django backend (models/views) wordt er data opgehaald en doorgegeven aan een template. De data wordt dan door behulp van een template mooi afgebeeld naar de gebruiker toe. Templates kunnen van elkaar over erven, bijvoorbeeld een header template kan worden verwerkt in een standaard webpagina template waarbij ze samen een geheel vormen voor de eindgebruiker [8].

Om data van de view (views.py) in een template te refereren moet er een (dictionary) context worden gecreëerd in een bepaalde view functie, in onderstaand codefragment (Codefragment 4: views.py met context dictionary [8]) is een voorbeeld zichtbaar.

****

Codefragment 4: views.py met context dictionary [8]

Nu er een context is gecreerd in de view functie en is gekoppeld aan een template (base.html) in de “return”, kan de gedefinieerde data worden opgehaald in het template. In onderstaand code fragment worden de variabelen “data” en “example” opgeroepen, waarna de tekst achter deze variabelen zal worden afgebeeld.

****

Codefragment 5: Template voorbeeld [8]

Graphical user interface, text, application, chat or text message

Description automatically generatedVolgende figuur (Figuur 9: Template output voorbeeld) wordt als resultaat weergegeven aan de gebruiker;

Figuur 9: Template output voorbeeld

Nu er een beter beeld is gecreëerd van wat de MVT-architectuur is, kan er met zekerheid worden gezegd dat het een cruciaal en onmisbaar deel van Django is waardoor vele taken eenvoudig en zeer dynamisch worden.

Als er wordt terug gekeken op voorgaande afbeelding (Figuur 5: Django MVT-werking [25]) kan de MVT-architectuur als volgt worden samengevat;  
Het Model helpt met beheer van de databank en is de laag die zorgt voor toegang tot de data. De View wordt gebruikt als logica laag die zorgt voor interactie met de modellen om data over te dragen, deze data wordt later weergegeven in een Template. De Template is de presentatie laag die de gebruiker interface volledig behandeld.

Er is geen aparte controller en de complete applicatie is gebaseerd op het Model View Template design, de applicatie wordt dus volledig behandeld door het framework zelf [25]. Als een model is aangemaakt in Django kan het worden gemigreerd in de databank waarna het meteen in gebruik kan worden gesteld. Deze opstelling maakt Django een zeer krachtig framework waardoor er kan worden gefocust op de ontwikkeling van de applicatie zelf en is minder werk is vereist van de ontwikkelaar waardoor hij of zij van scratch, zou moeten beginnen. Andere features zoals user management, software administratie features en anderen zijn steeds integreerbaar via packages [3] [4].

Django is voorbestemd voor snelle webontwikkeling, deze snelheid is één van de beste eigenschappen van Django en de kracht die die het de ontwikkelaar biedt zonder de functionaliteit en beveiliging naar beneden te brengen. Dat alles in één van de eenvoudigste en populairste talen ter wereld gekend als Python.

Django is het framework van de toekomst dat integratie biedt met de meeste hedendaagse technologieën. Ook ontwikkelt de gemeenschap voortdurend nieuwe en interessante functies en functionaliteiten om toe te voegen aan dit reeds fantastisch framework.

Voor dit project zal het Django framework dus een centraal punt zijn waar niet alleen de gebruiker maar ook de te onderhouden toestellen zullen samen komen. In een volgend hoofdstuk zal er worden onderzocht hoe er kan gecommuniceerd worden met deze toestellen, op welke manier en met behulp van welke technologie/protocol dat zal zijn.

# 4 Firewall configuratie

Om verschillende firewalls van andere vendoren te kunnen configureren/updaten moet er een manier zijn om deze aan te spreken. Omdat er tussen fabrikanten geen gedefinieerde standaard is van hoe er met firewalls kan worden gecommuniceerd (buiten SSH), is er een automatisatie engine, API-aanspreekpunt of ander alternatief nodig om dit te kunnen realiseren. Het is zo dat een gebruiker bij het toevoegen van een firewall op het Django platform moet kunnen aanduiden van welke vendor het product is. Op basis daarvan kan er achterliggend een andere connectie of basis configuratie worden opgesteld.

De reden dat er een mogelijkheid moet zijn om meerdere firewall vendors te kunnen configureren is als volgt; Indien de gebruiker ooit van vendor wisselt, is het platform nog steeds bruikbaar. Daarnaast hangt de gebruiker niet vast om te kiezen voor een specifieke vendor, omdat het platform enkel die vendor ondersteunt. Er werd bij aanvang van het project voorgesteld om het platform samen te laten werken met Fortimanager. Hoe dan ook als Fortimanager in gebruik wordt genomen komt men in de voorgaand vernoemde situatie terecht waarbij de gebruiker vast hangt aan een vendor (Fortinet in dit geval).

Logica diagram ?; vendor x => connect using API-request  
 vendor y => connect using SSH

## 3.1 Netwerk diagram

Diagram

Description automatically generatedOm het platform beschikbaar te maken naar de eindgebruiker moet dit ook toegankelijk zijn. Daarom is er een productieserver (Django-srv) in (Figuur 3: Netwerk/hosting diagram) aanwezig die het platform zal hosten. Productieversies worden aan de hand van een CI/CD-server (Jenkins-srv) getest en beschikbaar gemaakt. Op die manier kunnen werknemers en andere partners gebruik maken van het gerealiseerde platform/service. Het netwerk diagram (Figuur 3: Netwerk/hosting diagram) zal ook opnieuw een beter beeld geven van welke componenten samen werken. Omdat de toestellen in een labo omgeving zijn opgesteld zal er een VPN nodig zijn om het platform en beschikbare toestellen te consulteren. Na een aantal test fasen zal onderstaande opstelling later worden gerecreëerd in productieomgeving op voorziene infrastructuur van de opdrachtgever (bank).

### 3.3.2 Conclusie framework verschillen

In bovenstaande tabellen werd een overzicht gecreëerd over de voor- en nadelen van elk opgesomd framework in voorgaand hoofdstuk. Hoe dan ook, als er wordt vergeleken is Django toch één van de beste web frameworks voor het ontwikkelen van schaalbare complexe applicaties. Maar toch, blijven er een aantal toepassingen waar Django niet geschikt voor is zoals volledig eigen aanpasbaarheid waar Flask dan sterker uitkomt.

In dergelijke gevallen kunnen alternatieven voor Django worden gebruikt, zoals Flask, CherryPy, etc. Dit zijn web frameworks die kunnen worden gebruikt voor meer eenvoudig flexibele websites. Daarnaast zijn er nog enkele niet-Python frameworks zoals Node.JS, Laravel, React die ook een goed alternatief zijn voor Django. Maar dan is kennis van deze andere programmeertalen vereist [5].

# 5 Python virtuele omgevingen

Een virtuele Python omgeving bestaat uit twee componenten. De Python interpreter waarop de virtuele omgeving draait en een map met bibliotheken van derden die in de virtuele omgeving zijn geïnstalleerd. Deze virtuele omgevingen zijn gescheiden van de andere virtuele omgevingen, waardoor eventuele wijzigingen of geïnstalleerde packages aangebracht in die virtuele omgeving geen invloed hebben op de andere virtuele omgevingen of de systeembibliotheken. Er kunnen dus meerdere virtuele omgevingen worden gecreëerd met verschillende Python versies, bibliotheken of dezelfde bibliotheken in verschillende versies omdat elke virtuele omgeving onafhankelijk is van elkaar.

Figuur 10: Python virtuele omgevingen visueel voorbeeld [9]

In bovenstaande afbeelding (Figuur 9: Python virtuele omgevingen visueel voorbeeld [9]) is een voorbeeld van wat er op een systeem staat wanneer meerdere virtuele Python-omgevingen zijn gemaakt. Eigenlijk is een virtuele omgeving een afgezonderde mappenboom die een specifieke Python-versie, bibliotheken en andere scripts bevat. Er zijn geen beperking op het aantal virtuele omgevingen op een systeem, omdat het slechts mappen zijn die enkele bestanden bevatten [9].

## 5.2 Voordelen Python virtuele omgevingen

Het voordeel van virtuele Python omgevingen wordt duidelijk wanneer verschillende Python projecten op een dezelfde machine worden ontwikkeld. Sommige projecten zijn afhankelijk van verschillende versies of pakketten. Bijvoorbeeld; een ontwikkelaar werkt aan twee verschillende Django applicaties, het eerste project maakt gebruik van een ander front-end framework pakket dan het tweede project. Daarnaast zal het eerste project met Python versie 2.X en de andere met versie 3.X worden uitgerold. Dit zou leiden tot compatibiliteitsproblemen omdat Python niet tegelijkertijd meerdere versies van hetzelfde pakket kan gebruiken. Het andere voordeel dat het gebruik van virtuele Python omgevingen vergroot, is wanneer er wordt gewerkt op beheerde servers of productieomgevingen waar algemene systeem pakketten worden vereist en niet kunnen worden gewijzigd.

Virtuele Python omgevingen creëren afgezonderde contexten, om afhankelijkheden die nodig zijn voor verschillende projecten gescheiden te houden. Op die manier worden conflicten met andere projecten of systeem pakketten gecreëerd. Het komt erop neer dat het opzetten van virtuele omgevingen de beste manier is om verschillende Python projecten te isoleren. Vooral wanneer deze projecten verschillende en tegenstrijdige afhankelijkheden hebben. Het is aangeraden aan elke ontwikkelaar om altijd een virtuele omgeving te creëren voor elk Python project, waarbij installatie van alle vereiste pakketten daarin terecht komen [9]. Wanneer een project van machine moet migreren of dergelijke situatie is het minder lastig om de nodige pakketten over te zetten, dan een machine te klonen of pakketten individueel over zetten. In dit project worden ook virtuele omgevingen gebruikt, op die manier kan de Django omgeving en zijn packages later eventueel worden uitgerold aan de hand van Docker containers.

Het maken van een virtuele omgeving gebeurt met behulp van de module “venv”, wordt een folderstructuur nagemaakt van een standaard Python installatie op een systeem. Tegelijk worden symlinks gekopieerd in de folderstructuur naar de Python executable.   
De folderstructuur van een virtuele omgeving wordt in onderstaand fragment weergegeven (Codefragment 1: Folder structuur virtuele omgeving).



Codefragment 6: Folder structuur virtuele omgeving

# 6 FortiManager

FortiManager is een platform gemaakt om gecentraliseerd en automatisch beheer te kunnen voorzien voor meerdere FortiGate toestellen. Het platform voorziet controle, segmentatie en algemeen consistente bescherming voor toestellen, applicaties en gebruikers. Dit platform werd gekozen door de klant omdat het product specifiek is (Fortinet) en meer functionaliteit integreert dan andere alternatieven zoals IBM Security Guardium en WatchGuard Dimension. Het FortiManager platform voorziet ondersteuning om één tot honderdduizend toestellen te configureren vanuit één centrale console.

FortiManager beheert indirect de FortiAP, FortiSwitches en andere Secure Access toestellen via de FortiGate. Om de zero touch-configuratie te ondersteunen, maakt FortiManager gebruik van de functie "Add Model Device" waarmee gebruikers een toestel kunnen toevoegen. Eenmaal een FortiGate met een overeenkomende ID is geregistreerd bij de FortiManager wordt er automatisch de configuratie toegepast op dat specifieke toestel.

## 6.1 Zero Touch provisioning

Zero touch provisioning betekent dat een configuratie aan een apparaat wordt toegewezen zonder tussenkomst van de gebruiker, bij voorkeur voordat er toegang is tot de fysieke hardware. Er zijn meerdere methoden om dit soort functionaliteit te bereiken.

Zero touch provisioning is bedoeld om een ​​interessant probleem op te lossen. Het probeert de "menselijke fout" te verminderen bij het configureren van een toestel voor/tijdens productie. Of nu een fout wordt veroorzaakt door stress en tijdsdruk, er is altijd kans dat de persoon die de configuratie installeert een ​​kleine fout of storing van twee minuten kan veranderen in een storing van twee uur.

Het hebben van een niet-stressvolle, lagedrukomgeving om vooraf een configuratie goed te plannen en toe te passen, vergroot de kansen op een succesvolle implementatie enorm. In zekere zin geeft deze functie een kort moment tijdens de implementatie waar de engineer de "handen van het stuur" kan halen terwijl de configuratie naar het apparaat wordt geüpload.

Daarnaast is zero touch provisioning belangrijk omdat het de beheerder in staat stelt een configuratie voor meerdere apparaten in te stellen voordat er toegang wordt gegeven tot het fysieke of virtuele toestel. Hierdoor kan de beheerder offline tools gebruiken om snel apparaten in bulk te configureren en implementeren. Fortinet biedt een aantal tools om dit te bereiken bij het implementeren van FortiGate-firewalls [10].

Figuur 11: Zero touch provisioning (FortiManager) [10]

<https://www.avfirewalls.com/FortiManager-Series.asp>

## 6.2 Network automatisatie

<https://www.avfirewalls.com/FortiManager-Series.asp>

## 6.3 Central management

<https://www.avfirewalls.com/FortiManager-Series.asp>

# 7 Front-end frameworks

<https://stackoverflow.com/questions/59317193/what-is-the-best-way-to-build-a-django-webapp-frontend>

<https://www.reddit.com/r/django/comments/hlzv4x/most_compatible_css_framework_with_django/>

<https://www.youtube.com/watch?v=g73Nv8m9dno>

(Moet nog worden uitgewerkt)

Moet/kan er nog een extra front-end framework op Django? Welke zijn integreer baar?

# 8 Ansible

Ansible is een open source automatisatie engine die repetitieve taken zoals configureren van servers of uitrol van applicaties kan gaan automatiseren. Aan de hand van CI/CD kunnen snel vernieuwde versies van een applicatie online gebracht worden zonder enige downtime. Ansible is vooral gekend in de tak van “infrastructure as code” en is vergelijkbaar met het zero touch provisioning principe van FortiManager. Het verschil met FortiManager en Ansible is dan dat Ansible zo goed als alle soorten toestellen kan configureren, waarbij FortiManager specifiek is voor Fortinet producten. Ansible brengt meer consistentie, betrouwbaarheid en schaalbaarheid naar eender welke IT-omgeving. Met de YAML-taal is het ook gemakkelijker om scripts/configuraties te lezen en schrijven als standaard Engels.

Gebruikmakende van een enkele controle machine kunnen meerdere machines worden aangestuurd. Alle machines die onder de controle machine vallen worden aangesproken via OpenSSH. Met SSH kan de controle machine updates en configuraties uitvoeren op andere machines. Het is mogelijk om naast SSH-authenticatie gebruik te maken van Kerberos, LDAP of andere authenticatie managementsystemen [11].

Ansible kan dus worden gebruikt door meerdere applicaties zoals provisioning van virtuele machines, configuratie management, applicatie uitrol, beveiliging automatisatie en meer. Om de Django applicatie vlot uit te rollen en online te krijgen zal de Ansible engine een belangrijke rol spelen. Mogelijks in samenwerking met andere services of achterliggende Linux machines.

Figuur : Ansible stage levenscyclus [12]

## 9.1 Ansible toepassingen

### 9.1.1 Provisioning

Bare metal-servers hebben af en toe nood aan provisioning. Met Ansible kan iedereen veel zaken integreren zoals datacenterbeheer tools om servers aan te spreken en een provisioning te starten. Bij gebruik van verschillende hypervisor(s), virtuele opslag/netwerken kan Ansible een diverse selectie van modules en omgevingen ondersteunen. Door gebruik van sommige modules is het mogelijk om gemakkelijker te gaan schalen.

Naast bare metal servers en hypervisor(s) is het ook mogelijk om netwerkinfrastructuur en cloudoplossingen te leveren. De netwerk automatiseringsmogelijkheden zorgen voor validatie en continue werking van fysieke netwerkapparaten.

Wanneer Ansible een rol speelt voor een (publieke of private) Cloud omgeving is het mogelijk om niet alleen de clouddiensten te leveren, maar ook de onderliggende infrastructuur en applicaties binnen de Cloud te onderhouden [12].

### 9.1.2 Configuratie management

Ansible is een state-driven resourcemodel, wat inhoudt dat het de toestand van de systemen en services beschrijft. De huidige status van de doelmachine is niet relevant, aangezien Ansible de status van de machine naar wens zal transformeren. Dit leidt tot minder potentiële fouten in vergelijking met script gebaseerde oplossingen, omdat deze vaak onomkeerbare acties veroorzaken. Ansible heeft ook de mogelijkheid om de configuratie te valideren op mogelijke syntaxisfouten [13].

### 9.1.3 Applicatie deployment

Zoals eerder al werd vermeld is het mogelijk om te beschrijven wat de gewenste staat is van een machine. De gewenste staat van een machine is gedefinieerd in een Ansible “Playbook”. Playbooks worden uitgevoerd tegen een groep/enkele host waarbij de gewenste staat voor elke host hetzelfde is. Op die manier kan elke ontwikkelaar configuratie fouten vermijden en is de configuratie met zekerheid op elke machine hetzelfde. Op die manier maakt Ansible herhalende opdrachten zoals jaarlijks onderhoud en andere configuraties betrouwbaarder.

Diagram

Description automatically generatedPlaybooks worden verzonden via SSH naar de gedefinieerde doel host met de OS-compatibele modules, toestel specifieke modules zijn vergelijkbaar met opdrachten in de CLI voor het besturingssysteem van de doelmachine. Zodra de machine de module heeft ontvangen, worden ze ter plaatse uitgevoerd, eenmaal de taak is voltooid (staat geüpdatet) wordt de module verwijderd van de doel host [13].

Figuur : Zero downtime applicatie-update [20]

Zoals te zien is in bovenstaande afbeelding (Figuur 9) is het mogelijk om applicaties en updates uit te rollen, zonder dat de gebruiker enige downtime opmerkt. De gebruiker merkt geen downtime omdat elke server of servergroep apart worden geüpdatet met behulp van de load balancer. Wanneer de ontwikkelaar gebruik maakt van een Distributed Version Control System is het mogelijk om downloadbare artifacts te maken van actieve applicaties. Hoe dan ook, het is ook mogelijk om gebruik te maken van REST API’s waarbij een aantal variabelen worden aangepast op een andere service. Als er eenmaal aanpassingen zijn of de uitrol is van een nieuwe update, kunnen mail services worden opgesteld of een pushnotificatie sturen. Op die manier is de ontwikkelaar steeds up-to-date en weet hij/zij of de aanpassing al is doorgevoerd of niet.

## 9.2 Continuous delivery

Met de stage en testmethode kunnen Ansible-"plays" worden gevalideerd en getest. In het inventory bestand is het mogelijk om omgevingen in te delen in verschillende groepen van machines. Dit maakt het mogelijk om test en productiemachines te definiëren. Het is echter aangeraden om productie en testmachines te definiëren in aparte invenotory bestanden, op die manier worden enige verrassingen voorkomen wanneer een configuratie wordt uitgerold. Wanneer een "play" klaar is, kan het worden getest op de staging machines. Als de test slaagt, kan ervoor gekozen worden om ook automatisch de “play” op productiemachines uit te voeren waarbij het eindproduct beschikbaar wordt voor gebruikers.

Weet dat een “play” bestaat uit een set aan geschreven instructies in de YAML-taal om een ​​gewenste machineconfiguratie in te tellen. Deze machines kunnen bestaan uit monitoringsystemen, netwerk toestellen, load balancers, webservers en andere soorten toestellen. Een veelvoorkomende manier om Ansible te gebruiken is met een CI-systeem waarbij succesvolle builds van een play worden gepubliceerd [14].

Taken die plaatsvinden bij het testen:

* De CI zorgt ervoor dat Ansible een playbook uitvoert om een staging omgeving te implementeren met de nieuwe apparaat configuratie of toepassing
* Wanneer de test slaagt, kan er een melding worden verzonden om de uitrol te bevestigen naar productieomgeving
* Na publicatie kan de configuratie of applicatieversie worden opgeslagen in een artefact op de CI-server voor latere raadpleging of andere doeleinden.

## 9.3 Conclusie Ansible

Nu er een beter zicht is gecreëerd over de functionaliteiten van Ansible, kan er met zekerheid wordt gezegd dat Ansible een groter spectrum aan mogelijkheden naar boven brengt vergeleken met FortiManager. Op dit moment is nog niet duidelijk in welke tak Ansible een rol zal spelen in dit project, maar er kan met zekerheid wordt gezegd dat het niet voor CI/CD zal zijn en eerder voor pure firewall deployment. Daarvoor zal een ander automation tool voor worden gebruikt genaamd Jenkins. Hoe dan ook Ansible is een waardige kandidaat voor eender welk project dat men wil gaan automatiseren. De mogelijkheden en ondersteuning van alle verschillende modules maken de beschikbare automatisatie opties eindeloos.

# 10 Jenkins

Het enige constante in technologie is verandering, dus het hele proces begint opnieuw zodra er een release, nieuwe functie, update is zullen veranderingen uiteindelijk in productie worden gesteld. Daarom is er zoiets als CI/CD, als er wordt gedacht om CI/CD te implementeren zijn er meerdere oplossingen die dit kunnen bieden. Daaronder vallen automatisatie tools zoals Jenkins, GitLab CI/CD, Ansible, TeamCity, etc. Het doel achter continuous delivery is dat projecten en ideeën uiteindelijk in gebruik worden genomen door de eindgebruiker, wanneer een update gebeurt dat ze deze niet opmerken en het project beschikbaar blijft.

Het maakt externe gebruikers niet uit hoe een product of idee werd afgeleverd, zo lang ze eenmaal toegang hebben tot de functie of applicatie. Voor het interne team zoals een DevOps-engineer, is het wel belangrijk hoe een product wordt geleverd, omdat het proces van functielevering rechtstreeks invloed heeft op het team. Een product dat niet op de juiste manier wordt afgeleverd, kan later voor heel wat zorgen en problemen zorgen die kunnen worden vermeden, daarom maakt men best gebruik van CI/CD.

Zoals eerder vermeld is Ansible ook in staat om CI/CD te voorzien. Hoe dan ook is dit niet het kerndoel van Ansible, daarom zal gebruik gemaakt worden van Jenkins. Enige tekortkomingen of andere zaken dat Jenkins en/of Ansible hebben, kunnen worden opgevangen door ze eventueel te laten samen werken. De samenwerking van deze twee technologieën is een vaak voorkomend fenomeen.

Jenkins is een open source platform en heeft net als Django een sterke gemeenschap achter zich staan, voor enige bijstand die een ontwikkelaar of DevOps-engineer zou nodig hebben kan worden gerekend op publieke forums. Opnieuw omdat het een zeer populair open source engine is worden steeds meer en betere functies naar Jenkins gebracht na controle van meerdere ontwikkelaars, waardoor ook de nieuwste veiligheids standaarden worden toegepast. Omdat Jenkins een gedistribueerde werking heeft is het voldoende om slechts één server te voorzien, zelfs voor complexe projecten. De reden dat maar één server nodig is, is omdat Jenkins gebruik maakt van de Master-Slave architectuur voor het beheren van gedistribueerde builds. De Jenkins server is de “Master”, waarna het mogelijk is om werklast te verdelen over meerdere “slave” servers. Afhankelijk van hoe Jenkins is geconfigureerd is het mogelijk om op deze “slave” servers meerdere build/test omgevingen tegelijkertijd te laten lopen. Het testen van builds op meerdere servers kan handig zijn om bijvoorbeeld compatibiliteit te testen op verschillende besturingssystemen (Figuur 13: Jenkins master-slave architectuur [15]) [16].

Figuur : Jenkins master-slave architectuur [15]

De reden achter het gebruik van Jenkins in plaats van de ingebouwde CI/CD-service van GitLab, is dat er later nog steeds zonder problemen kan worden gewisseld van DVCS-service of platform. Enige aanpassingen die moeten gebeuren in Jenkins om pipeline(s) operationeel/actief te houden is de repository bron URL-aanpassen.



Figuur : Jenkins visuele werking [23]

## 10.1 Jenkins minimumvoorwaarden

|  |  |
| --- | --- |
| **Type hardware** | **Minimumvoorwaarde** |
| RAM | 256MB |
| Schijf opsalg | 1GB (10GB bij gebruik docker container) |

Voor Jenkins operationeel kan zijn op een machine zijn er een aantal minimumvoorwaarden die moeten voldoen. Onder deze voorwaarden vallen dan vooral hardware maar ook software voorwaarden. Een belangrijke factor op vlak van hardware is het aantal beschikbare tijdelijk geheugen (RAM) en harde schijf opslagcapaciteit in Gigabytes (GB).

Tabel 1: Jenkins minimum hardware voorwaarden

De hardware voorwaarden van bovenstaande tabel (Tabel 6: Jenkins minimum hardware voorwaarden) zijn vaak niet genoeg wanneer meerdere jobs tegelijk moeten kunnen draaien. Ook wanneer wordt gewerkt met een team van mensen zijn bovenstaande minimumaantallen niet voldoende. Voor een klein team wordt al sneller één Gigabyte aan RAM en vijftig Gigabytes aan opslag aangeraden. Hoe dan ook verschillen de minimumvoorwaarden van team tot team en van project tot project.

Naast hardware voorwaarden is het software gedeelte waarbij wordt verwacht dat een aantal packages zijn geïnstalleerd, en compatibel zijn met het systeem waarop ze worden geïnstalleerd. In onderstaande tabel (Tabel 7: Jenkins minimumsoftware voorwaarden) is een oplijsting van de verschillende minimumvoorwaarden op valk van software [17].

|  |  |
| --- | --- |
| **Software** | **Beschrijving** |
| Java | Java Development Kit ([JDK](https://www.oracle.com/java/technologies/downloads/)) of (JRE) |
| Webbrowser (Enkel voor systeem met GUI) | Google chrome, Firefox, Safari, etc. |
| Besturingssysteem | Windows, macOS, Linux (Ubuntu, etc) |

Tabel 2: Jenkins minimumsoftware voorwaarden [17]

In bovenstaande tabel kan iedereen de nodige software/packages raadplagen wanneer zal gebruik worden gemaakt van Jenkins. Omdat niet alle machines een GUI ondersteunen is een webbrowser niet echt een vereiste. Zolang de eindgebruiker de Jenkins interface maar kan raadplegen is er geen probleem. Er kan bijvoorbeeld aan de hand van een andere machine met een webbrowser naar het IP-adres van de Jenkins server worden gegaan om dit op te lossen.

10.2 Jenkins best practices

De gebruikersinterface van Jenkins is niet zo heel gebruiksvriendelijk als sommige modernere alternatieven. Sommige geavanceerdere configuraties kunnen lastig zijn om in te configureren voor DevOps-engineers zonder veel ervaring, hoewel de community support sterk is om hierbij te helpen. Voor de vele problemen is de kans groot dat er een plug-in is ontwikkeld die kan helpen in eender welke situatie.

Om ervoor te zorgen de engineer het beste uit Jenkins haalt, zijn er toch een aantal zaken die de engineer best opvolgt;

* Houd het veilig: zoals bij elke belangrijke softwaretool, is het van belang om Jenkins-server(s) veilig te houden met robuuste gebruikersauthenticatieprocessen. Daarnaast is een back-up van de map 'Jenkins Home' steeds aangeraden, deze bevat alle configuratie- en loggegevens die een engineer zeker niet wilt verliezen.
* Wees selectief met plug-ins: het aantal beschikbare plugins is groot, daarom kan het verleidelijk zijn om er te veel te proberen. Dit zou processen kunnen vertragen, waardoor het hoofddoel van een snelle levering teniet wordt gedaan.
* Vermijd complexe Groovy-code in pijplijnen: Groovy-code wordt altijd uitgevoerd op de master/controller, dus als het te ingewikkeld is, zal dit een aanslag zijn op de computerbronnen. Onthoud dat de pipeline is ontworpen om het team te helpen, sneller te bouwen en dus het ontwerp best zo eenvoudig mogelijk houdt [16].

# 11 Praktische uitwerking

(Moet nog worden uitgewerkt)

* ESXi opstellen van VM’s
* Opstellen van virtuele omgeving
* Installeren/exporteren (pip freeze) van nodige packages
* Jenkins opstelling/piplines
* Opstelling Django project (structuur/connecties)

# 12 Hosting

Hoe kan dit platform meerdere malen gaan opstellen op andere locaties en de configuratie automatiseren? Is dat mogelijk met Ansible en wat voor servers of services zijn nodig om een Django platform te lanceren? Hoe zal een service engineer verbinding kunnen maken met het platform (via VPN of andere manier)?

# 13 Handleiding

Hier zullen delen van het platform worden overlopen, waarbij handelingen en stappen worden verduidelijk mocht dat nodig zijn om bepaalde acties te bereiken.

# Conclusie

(Moet nog worden uitgewerkt)

● Welke doelstellingen zijn bereikt?

● Als er bepaalde doelstellingen niet bereikt zijn, wat zijn de redenen hiervoor? Hoe kon dit voorkomen worden?

● Gaan jouw bereikte resultaten in de toekomst toegepast worden? Heb je daar stappen voor ondernomen?

● Hoe kunnen je resultaten in de toekomst nog verbeterd worden?

● Is er nog vervolgonderzoek nodig?

● Misschien heb je ervaren dat bepaalde werksituaties minder efficiënt aangepakt worden. Heb jij tips om deze op te lossen?

# Nawoord

(Moet nog worden uitgewerkt)

● Hoe verliep de samenwerking met de collega’s?

● Hoe verliep het rapporteren naar collega’s en verantwoordelijken?

● In welke mate kon je een probleem zelfstandig aanpakken?

● Wat heb je bijgeleerd over bedrijfseigen afspraken i.v.m. veiligheid, milieu, kwaliteit, …?

● Op welke manier heb je het eigen werk geëvalueerd en in welke mate deed je dat systematisch

# Literatuurlijst

|  |  |
| --- | --- |
| [1] | JavaPoint, „Django history - javapoint,” [Online]. Available: https://www.javatpoint.com/django-tutorial. [Geopend 22 03 2022]. |
| [2] | Django docs (community), „FAQ Django docs - History and pronounce,” Django community, [Online]. Available: https://docs.djangoproject.com/en/4.0/faq/general/#:~:text=Django%20is%20pronounced%20JANG%2Doh,audio%20clip%20of%20the%20pronunciation.. [Geopend 17 05 2022]. |
| [3] | I. Education, „What is Django - IBM,” 29 03 2021. [Online]. Available: https://www.ibm.com/cloud/learn/django-explained. [Geopend 21 03 2022]. |
| [4] | „Full Stack Python - Django framework,” [Online]. Available: https://www.fullstackpython.com/django.html. [Geopend 21 05 2022]. |
| [5] | Javapoint, „Django MVT - javatpoint,” [Online]. Available: https://www.javatpoint.com/django-mvt. [Geopend 17 05 2022]. |
| [6] | GeeksforGeeks, „GeeksforGeeks - Django Models,” 11 04 2022. [Online]. Available: https://www.geeksforgeeks.org/django-models/. [Geopend 22 05 2022]. |
| [7] | GeeksforGeeks, „GeeksforGeeks - Views In Django,” 16 09 2021. [Online]. Available: https://www.geeksforgeeks.org/views-in-django-python/. [Geopend 22 05 2022]. |
| [8] | GeeksforGeeks, „GeeksforGeeks - Django Templates,” 27 09 2021. [Online]. Available: https://www.geeksforgeeks.org/django-templates/. [Geopend 23 05 2022]. |
| [9] | „Django introduction,” MDN contributors, 18 02 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction. [Geopend 20 03 2022]. |
| [10] | J. Patel, „Django Alternatives: Top 5 Python Frameworks To Use in 2022,” 02 03 2022. [Online]. Available: https://www.monocubed.com/blog/django-alternatives/. [Geopend 22 03 2022]. |
| [11] | M. Lotfinejad, „Dataquest - A Complete Guide to Python Virtual Environments,” 09 03 2022. [Online]. Available: https://www.dataquest.io/blog/a-complete-guide-to-python-virtual-environments/. [Geopend 14 05 2022]. |
| [12] | J. Torian, „Zero(ish) Touch Provisioning with FortiManager – Explained,” 04 02 2019. [Online]. Available: https://www.historiantech.com/zeroish-touch-provisioning-with-fortimanager-explained/. [Geopend 01 05 2022]. |
| [13] | R.H Ansible, „About Ansible - Ansible documentation,” [Online]. Available: https://docs.ansible.com/ansible/latest/index.html. [Geopend 03 11 2021]. |
| [14] | R.H Ansible, „Red Hat Ansible use case - Provisioning,” [Online]. Available: https://www.ansible.com/use-cases/provisioning. [Geopend 03 11 2021]. |
| [15] | R.H. Ansible, „Ansible for Configuration management,” [Online]. Available: https://www.ansible.com/use-cases/configuration-management. [Geopend 03 11 2021]. |
| [16] | R.H. Ansible, „Ansible For Continuous Delivery,” [Online]. Available: https://www.ansible.com/use-cases/continuous-delivery. [Geopend 03 11 2021]. |
| [17] | J. McAllister, „packtpub - Mastering Jenkins,” [Online]. Available: https://subscription.packtpub.com/book/application-testing/9781784390891/2/ch02lvl1sec15/understanding-the-master-and-slave-architecture. [Geopend 15 05 2022]. |
| [18] | S. Mino, „Jobsity - What is Jenkins and Why Should You Use It?,” [Online]. Available: https://www.jobsity.com/blog/what-is-jenkins-and-why-should-you-use-it. [Geopend v 05 2022]. |
| [19] | J., „TOOLSQA - Jenkins What are the pre- requisites and procedure,” 14 05 2022. [Online]. Available: https://www.toolsqa.com/jenkins/install-jenkins/. [Geopend 14 05 2022]. |
| [20] | A. Technologies, „Django timeline – Skillsets, Budget and Benefits,” 05 01 2021. [Online]. Available: https://www.agiratech.com/django-web-development-skillsets-budget-and-benefits. [Geopend 21 03 2022]. |
| [21] | A. Martin, „Django logo,” 03 08 2021. [Online]. Available: https://njkhanh.com/flask-django-or-pyramid-choose-the-right-python-framework-for-your-project-p5f33363134. [Geopend 24 03 2022]. |
| [22] | R.H. Ansible, „Ansible for application deployment,” [Online]. Available: https://www.ansible.com/use-cases/application-deployment. [Geopend 03 11 2021]. |
| [23] | Wikipedia contributors, „Django (web framework),” 05 04 2022. [Online]. Available: https://en.wikipedia.org/wiki/Django\_(web\_framework). [Geopend 13 04 2022]. |
| [24] | „Web2py Python Framework (Youtube),” 01 08 2014. [Online]. Available: https://www.youtube.com/watch?v=yKQNmutC0WY. [Geopend 28 04 2022]. |
| [25] | P. ReviewDesk, „CherryPy - Python Web Framework,” 13 04 2022. [Online]. Available: https://www.predictiveanalyticstoday.com/cherrypy/. [Geopend 28 04 2022]. |
| [26] | W. Semik, „Flask vs. Django - STXNEXT,” 29 11 2021. [Online]. Available: https://www.stxnext.com/blog/flask-vs-django-comparison/. [Geopend 10 05 2022]. |
| [27] | „Web2Py pros and cons - Choosing a better framework,” [Online]. Available: https://www.tutorialspoint.com/python\_web\_development\_libraries/python\_web\_development\_libraries\_choosing\_a\_better\_framework.htm. [Geopend 10 05 2022]. |
| [28] | „Compare CherryPy vs web2py,” 13 02 2021. [Online]. Available: https://codeahoy.com/compare/cherrypy-vs-web2py. [Geopend 10 05 2022]. |
| [29] | „Edureka - Afbeelding Jenkins werking,” [Online]. Available: https://www.edureka.co/blog/content/ver.1531719070/uploads/2018/07/Asset-36-1.png. [Geopend 15 05 2022]. |

# Bijlagenoverzicht

Bijlage 1: Platform flowchart

Bijlage 2: Relatie diagram

Bijlage 3: Netwerk/hosting diagram

## Teams Description automatically generated with low confidenceBijlage 1: Platform flowchart

## Full relation diagramBijlage 2: Relatie diagram

## Bijlage 3: Netwerk diagram

