**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 Sanders Sven voor de fantastische begeleiding tijdens mijn stageperiode. In bijzonder wil ik de heer Donné Johan 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](#_Toc103508824)

[Figurenlijst 2](#_Toc103508825)

[Tabellenlijst 3](#_Toc103508826)

[Afkortingenlijst 4](#_Toc103508827)

[Begrippenlijst 5](#_Toc103508828)

[Inleiding 6](#_Toc103508829)

[1 Analyse opstelling 7](#_Toc103508830)

[1.1 Platform flowchart 7](#_Toc103508831)

[1.2 Relatie diagram 9](#_Toc103508832)

[1.3 Netwerk diagram 10](#_Toc103508833)

[2 Framework 11](#_Toc103508834)

[2.1 Django geschiedenis 11](#_Toc103508835)

[2.2 Django framework 13](#_Toc103508836)

[2.3 Voor- en nadelen Django framework 14](#_Toc103508837)

[3 Django alternatieven 15](#_Toc103508838)

[3.1 Flask 15](#_Toc103508839)

[3.1.1 Voor- en nadelen Flask framework 16](#_Toc103508840)

[3.2 Web2py 16](#_Toc103508841)

[3.2.1 Voor-en nadelen Web2Py 17](#_Toc103508842)

[3.3 CherryPy 18](#_Toc103508843)

[3.3.1 Voor-en nadelen CherryPy 18](#_Toc103508844)

[3.3.2 Conclusie framework verschillen 19](#_Toc103508845)

[5 Python virtuele omgevingen 19](#_Toc103508846)

[5.2 Voordelen Python virtuele omgevingen 20](#_Toc103508847)

[6 FortiManager 20](#_Toc103508848)

[6.1 Zero Touch provisioning 20](#_Toc103508849)

[6.2 Network automatisatie 21](#_Toc103508850)

[6.3 Central management 21](#_Toc103508851)

[7 Front-end frameworks 22](#_Toc103508852)

[8 Ansible 23](#_Toc103508853)

[9.1 Ansible toepassingen 23](#_Toc103508854)

[9.1.1 Provisioning 23](#_Toc103508855)

[9.1.2 Configuratie management 24](#_Toc103508856)

[9.1.3 Applicatie deployment 24](#_Toc103508857)

[9.2 Continue levering 25](#_Toc103508858)

[9.3 Conclusie Ansible 25](#_Toc103508859)

[10 Jenkins 26](#_Toc103508860)

[10.1 Jenkins minimumvoorwaarden 27](#_Toc103508861)

[11 Praktische uitwerking 28](#_Toc103508862)

[12 Hosting 28](#_Toc103508863)

[13 Handleiding 29](#_Toc103508864)

[Conclusie 30](#_Toc103508865)

[Nawoord 31](#_Toc103508866)

[Literatuurlijst 32](#_Toc103508867)

[Bijlagenoverzicht 34](#_Toc103508868)

[Bijlage 1: Platform flowchart 35](#_Toc103508869)

[Bijlage 2: Relatie diagram 36](#_Toc103508870)

[Bijlage 3: Netwerk diagram 37](#_Toc103508871)

# Codefragmentlijst

**Geen gegevens voor lijst met afbeeldingen gevonden.**

# Figurenlijst

[Figuur 1: Platform logica overzicht 8](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508803)

[Figuur 2: Connectie relatie diagram 9](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508804)

[Figuur 3: Netwerk/hosting diagram 10](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508805)

[Figuur 4: Django tijdlijn [16] 11](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508806)

[Figuur 5: Django/Python logo [17] 13](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508807)

[Figuur 6: Flask logo [5] 15](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508808)

[Figuur 7: Voorbeeld Web2py IDE [19] 17](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508809)

[Figuur 8: CherryPy voorbeeld code [20] 18](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508810)

[Figuur 9: Python virtuele omgevingen visueel voorbeeld [9] 19](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508811)

[Figuur 10: Zero touch provisioning (FortiManager) [10] 21](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508812)

[Figuur 11: Ansible stage levenscyclus [12] 23](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508813)

[Figuur 12: Zero downtime applicatie-update [18] 24](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508814)

[Figuur 13: Jenkins master-slave architectuur [22] 26](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508815)

[Figuur 14: Jenkins visuele werking [21] 27](https://hubkaho-my.sharepoint.com/personal/gerrit_vanmol_student_odisee_be/Documents/Bureaublad/Repos/automationplatform/Documents/Papers/V3_AutomatisatiePlatform-GerritVanMol.docx#_Toc103508816)

# Tabellenlijst

[Tabel 1: Django versie tijdlijn [2] 12](#_Toc103508817)

[Tabel 2: Voor-en nadelen Django framework 14](#_Toc103508818)

[Tabel 3: Voor-en nadelen Flask framework [6] 16](#_Toc103508819)

[Tabel 4: Voor-en nadelen Web2Py [7] 17](#_Toc103508820)

[Tabel 5: Voor-en nadelen CherryPy [8] 18](#_Toc103508821)

[Tabel 6: Jenkins minimum hardware voorwaarden 27](#_Toc103508822)

[Tabel 7: Jenkins minimumsoftware voorwaarden [15] 28](#_Toc103508823)

# 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. Hoe kunnen configuraties het efficiëntst worden beheerd. Of het mogelijk is meerdere connecties tegelijk tot stand te brengen tussen het platform en toestellen terplekke. De klant vraagt om gebruik te maken van het Django framework, Fortimanager en/of Ansible. Andere frameworks en automatisatie engines worden daarnaast ook nog vergeleken.

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 setup van een firewall waarbij connectiviteit zoals een ping naar Fortigatemanager functioneert zoals het hoort.

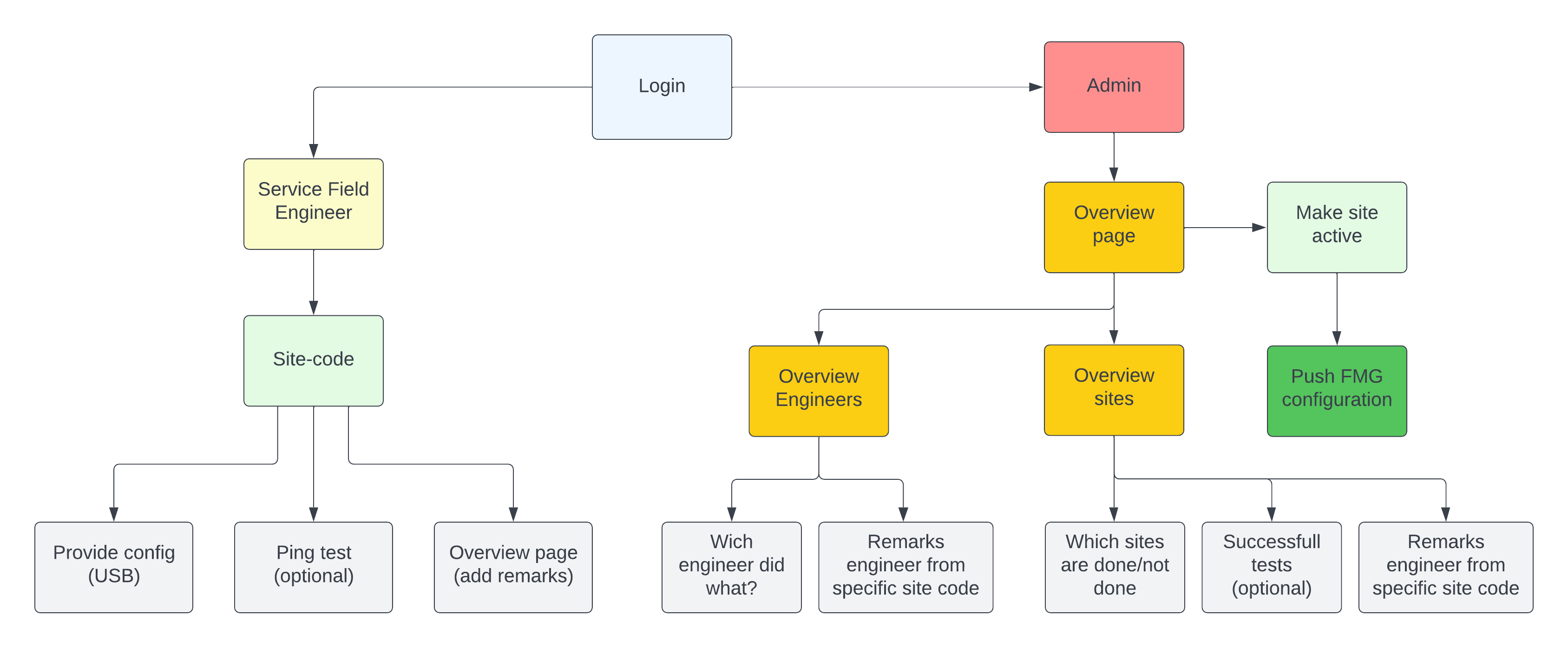
De klant heeft daarnaast geen vooropgestelde beperkingen of technische randvoorwaarden waaraan moet worden voldaan.

# 1 Analyse opstelling

In volgende subhoofdstukken worden relatie diagrammen en logica processen beschreven. Deze processen zijn vastgelegd en vooropgesteld door het externe bedrijf waar deze opdracht voor uitwerkt wordt. Met de opgestelde schema’s, flowcharts en diagrammen worden processen visueel verduidelijkt. 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 Platform flowchart

Omdat er binnen het bedrijf een bepaalde werk structuur moet worden aangenomen voor het uitrollen en registreren van firewalls is deze workflow ook vastgelegd en gevisualiseerd in een flowchart. In onderstaande flowchart (Figuur 1: Platform logica overzicht) is de huidige logica zichtbaar, met daarbij de minimumpagina’s die gebruikers kunnen zien op het platform afhankelijk van welke soort rol een gebruiker heeft. Daarnaast zal het schema ook worden gebruikt om de volledige projectstructuur op te stellen en te rangschikken op de Django server. Zo zal de flowchart later ook worden gebruik als basis om logische SQL-relaties te op te bouwen in het Django framework. Deze relaties zullen definiëren welke componenten iets met elkaar gemeen hebben alsook hoe data kan vloeien om uiteindelijk logisch weer te geven aan de eindgebruiker.



Figuur 1: Platform logica overzicht

## 1.2 Relatie diagram

Het relatie diagram (Figuur 2: Connectie relatie diagram) visualiseert hoe de verschillende technologieën zullen samenwerken en geconnecteerd worden. Onder deze technologieën worden dan het backend, front-end en netwerk gedeelte verstaan. Er wordt een beter beeld gegeven van de cruciale componenten en velden die in het gehele project zullen moeten bijgehouden worden. Op die manier kan door de field engineer sneller maar ook beter, een correcte en werkende omgeving opgebouwd worden.

Figuur 2: Connectie relatie diagram

## 1.3 Netwerk diagram

Om het platform beschikbaar te maken naar de eindgebruiker moet deze 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. De 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.

Figuur 3: Netwerk/hosting diagram



# 2 Framework

Om het platform uit te werken is er een framework nodig dat kan worden gebruik als fundamenteel center punt om verder op te bouwen. Aan de hand van dit framework zal het platform worden uitgewerkt. Er werd voor dit project extern gekozen voor het Django framework waarbij een potentiële overweging voor eventuele andere frameworks worst 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. Alsook zullen meerdere voor-/nadelen worden overlopen en vergelijken met andere populaire frameworks. Op die manier wordt er een beter zicht gecreëerd over de verschillende frameworks en voor welke toepassingen ze best ingezet kunnen worden.

## 2.1 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 “Jango” en 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 4: Django tijdlijn [18]

Volgende tabel (Tabel 1: Django versie tijdlijn [2]) geeft een gedetailleerdere weergave van de voorbije Django versies.

|  |  |  |
| --- | --- | --- |
| **Versie** | **Datum** | **Beschrijving** |
| 0.90 | 16 Nov 2005 |  |
| 0.91 | 11 Jan 2006 | "new-admin" |
| 0.95 | 29 Jul 2006 | "magic removal" |
| 0.96 | 23 Mar 2007 | "newforms", testing tools |
| 1.0 | 3 Sep 2008 | API stability, decoupled admin, unicode |
| 1.1 | 29 Jul 2009 | Aggregates, transaction based tests |
| 1.2 | 17 May 2010 | Multiple db connections, CSRF, model validation |
| 1.3 | 23 Mar 2011 | Class based views, staticfiles |
| 1.5 | 26 Feb 2013 | Time zones, in browser testing, app templates. |
| 1.6 | 6 Nov 2013 | Python 3 Support, configurable user model |
| 1.7 | 2 Sep 2014 | db transaction management, connection pooling. |
| 1.8 LTS | 2 Sep 2014 | Migrations, application loading and configuration. |
| 1.8 LTS | 1 Apr 2015 | Native support template(s) engines. |
| 1.9 | 1 Dec 2015 | Automatic password validation. New styling for admin UI. |
| 1.10 | 1 Aug 2016 | Full text search for PostgreSQL. New-style middleware. |
| 1.11 LTS | 4 Apr 2017 | Last support Python 2.7. |
| 2.0 | 2 Dec 2017 | First Python 3-only release, Simplified URL routing syntax, Mobile friendly admin. |
| 2.1 | 1 Aug 2018 | Model "view" permission. |
| 2.2 LTS | 1 Apr 2019 | Security release. Supported until at least April 2022 |
| 3.0 | 2 Dec 2019 | ASGI-support |
| 3.1 | 4 Aug 2020 | Asynchronous views and middleware |
| 3.2 LTS | 6 Apr 2021 | Extended Support until at least April 2024 |
| 4.0 | 7 Dec 2021 | Extended Support until at least April 2023 |

Tabel 1: Django versie tijdlijn [2]

## 2.2 Django framework

Django is een gratis open source framework, gemaakt om te focussen op de development 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 framework heeft ook een non-profit waarbij meer dan 10 000 packages worden ondersteund dat voor een rijke set aan features zorgt. Alle packages/features worden steeds frequent onderhouden en ondersteund. Met een rijke set aan features en packages, bijvoorbeeld ingebouwde API’s, CMS, UA, CAPTCHA-ondersteuning en form validatie.

Graphical user interface, application

Description automatically generatedHet 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. Met Django kan er worden gefocust op de applicatie zelf en is minder werk vereist dan dat de ontwikkelaar van scratch zou moeten beginnen [3] [4].

Figuur 5: Django/Python logo [19]

Zoals eerder vermeld werd door de opdrachtgever opgelegd om gebruik te maken van het Django framework. Hierdoor is de eigen keuze voor gebruik van een alternatief framework onmogelijk. Hoe dan ook worden mogelijke alternatieve frameworks niet uitgesloten en in een later hoofdstuk in de kijker gezet met hun voor- en nadelen. Hoe dan ook is het gekozen framework voor de uitwerking van dit project beslist door meerdere achterliggende redenen. Het framework laat toe om een volledig datamodel op te bouwen in Python zonder dat men MySQL nodig heeft. Het JSON-formaat en andere vaak voorkomende data typen worden actief ondersteund door Django MySQL. Gebruik makende van een ORM kan Django traditionele databank structuren en tabellen converteren in Python klassen. Op deze manier kan men sneller en gemakkelijker werken in één en dezelfde Python omgeving.

Django heeft de structuur van MVC-MVT waarbij men krachtige componenten van het met het framework kan maken [5]. Aan de hand van de modellen in MVC kunnen de gecreëerde webapplicaties worden geconnecteerd met Django en kan de data worden beheerd. Database velden worden vaak omgezet in klasse attributen. Deze attributen zijn ook gedefinieerd in Python, waardoor ze gemakkelijk worden beheerd met de ingebouwde Django database. Het web framework biedt een snelle mogelijkheid om te integreren met een applicatie database. Aan de hand van CRUD kan men meerdere operaties uitvoeren op duurzame gegevens zoals het maken, lezen, bijwerken en verwijderen van gegevens. Naast CRUD zijn andere features zoals user management, software administratie features en anderen steeds integreer baar via packages [3].

## Voor- en nadelen Django framework

|  |  |
| --- | --- |
| **Voordelen** | **Nadelen** |
| Heeft een MVT (Model View Template) architectuur | Kan meerdere requests niet gelijktijdig verwerken |
| Zeer goed beveiligd en up-to-date met laatste security patches | Niet geschikt voor kleine projecten |
| Aangedreven door Python programmeertaal | Steile leer curve |
| Beschikt over ORM (Object Relational Mapper) | ORM-Systeem mist functies (Python toolkits) |
| Beschik over standaard administrator interface | URL-referenties zijn niet vanzelfsprekend |
| Snelle ontwikkeling | Het is monolistisch (om arm volledig pakket of niets) |
| Zeer schaalbaar | Nauw gekoppeld (packages en bibliotheken kunnen niet zonder elkaar) |
| Actieve gemeenschap, goed onderhouden documentatie en voorbeelden |  |
| Batterijen inbegrepen filosofie |  |
| Performantie |  |
| Object georiënteerd |  |

Tabel 2: Voor-en nadelen Django framework

Uiteindelijk komt het erop neer dat Django voorbestemd is 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 de eenvoudigste en één van de 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. Dit framework is geschikt voor beginners als voor ervaren ontwikkelaars en kan iedereen zijn webontwikkeling vaardigheid tillen naar een nieuw niveau.

# 3 Django alternatieven

Naast Django zijn er tal van andere Python frameworks die de klant had kunnen kiezen. Flask is bijvoorbeeld ook één van de meest populaire frameworks voor web development naast Django. Het gekozen framework (Django) heeft tal van voordelen, maar ook een aantal nadelen waarom het juist niet interessant zou kunnen zijn. Zo is bijvoorbeeld de volledige projectstructuur niet te vermeiden wanneer de ontwikkelaar een enkele webpagina wil maken. Onderstaand worden een drietal populaire frameworks bekeken, zo dat in een later hoofdstuk kan worden vergeleken met Django [5]. Op die manier wordt er een algemeen zicht gecreëerd van andere mogelijkheden en voor welke toepassingen een bepaald framework voordeliger kan zijn.

## 3.1 Flask

Afbeelding met tekst

Automatisch gegenereerde beschrijvingFlask is een micro-framework dat geschikt is voor kleine projecten die een snelle manier nodig hebben om een ​​eenvoudige, door Python aangedreven website te maken. Het is zeer schaalbaar en zeer geschikt voor startups. Zo kan iedereen beginnen met het ontwikkelen van een basistoepassing en vervolgens opschalen naar afhankelijk van het toenemend gebruikersverkeer. Het drijft veel kleine tools en eenvoudige web interfaces aan die zijn gebouwd op bestaande API's. Het maakt gebruik van microservices en micro-front-end(s) om de laadtijd van een webpagina te verkorten. Flask wordt geleverd met veel configuratiewaarden, samen met gemakkelijk aanpasbare standaardinstellingen en conventies. Het implementeert ook algemene beveiligingsmaatregelen zoals sessie gebaseerd beheer, wachtwoord-hashing, HTTP-authenticatie en cross-site request forgery (CSRF) voor bescherming tegen kwetsbaarheden. Flask heeft een ingebouwde ontwikkelserver en is volledig geschikt voor Uni code, terwijl het ook unittesten, REST en WSGI-compliance ondersteunt. Het is een zeer snel framework voor het ontwikkelen van front-end designs en vereist zeer weinig configuraties waarbij een standaard web interface beschikbaar is bij backend ontwikkeling [5].

Figuur 6: Flask logo [5]

### 3.1.1 Voor- en nadelen Flask framework

|  |  |
| --- | --- |
| **Voordelen** | **Nadelen** |
| Hogere flexibiliteit | Meer veiligheids gevaren |
| Hogere compatibiliteit met de nieuwste technologieën | Trage MVT (Model View Template) ontwikkeling |
| Hoge schaalbaarheid voor eenvoudige webapplicaties | Complexere tech stack |
| Gemakkelijk voor technische expirimenten | Hoge onderhoudskosten |
| Ligt hogere performantie | Lastig te onderhouden op lange termijn |
| Gemekkelijk voor eenvoudige doeleinden | Gecompliceerd voor grote implementaties |
| Kleinere code base | Afhankelijk van externe plugins/bibliotheken |
| Zeer aanpasbaar (plugins en andere in eigen handen) | Geen ingebouwd admin paneel |

Tabel 3: Voor-en nadelen Flask framework [6]

## 3.2 Web2py

Web2py is een web gebaseerd framework waarmee iedereen overal browser apps kan maken, wijzigen, implementeren en beheren. Het is een platformonafhankelijk framework dat op meerdere apparaten en browsers kan worden gebruikt. Het implementeert ook een krachtige foutregistratie en ticketing-systeem. Web2py heeft geen afhankelijkheden buiten de Python-standaardbibliotheek. Het framework vereist geen vereisten voor installatie of configuratie. Naast achterwaartse compatibiliteit heeft het ook beveiligingsmaatregelen tegen aanvallen zoals cross-site scripting, SQL-injectie en andere kwaadaardige aanvallen. Het framework beschik over een volledig uitgeruste IDE in de vorm van een web interface, waarmee de ontwikkelaar website-inhoud vanuit elke webbrowser kan wijzigen. Met Web2py kunnen snelle, schaalbare en veilige database gestuurde webapplicaties ontwikkeld worden [5].

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Figuur 7: Voorbeeld Web2py IDE [21]

### 3.2.1 Voor-en nadelen Web2Py

|  |  |
| --- | --- |
| **Voordelen** | **Nadelen** |
| Ingebouwde IDE (interface) | Ondersteund enkel Python2.X |
| Component configuratie (MVC) | Beperkte flexibiliteit |
| Ingebouwd admin paneel | Weinig tot geen up-to-date documentatie/voorbeelden |
| Ondersteund templates | Ondersteund enkel Google Datastore |
| Zeer performant | Ondersteund doctests (geen unit tests) |
|  | Productie/ontwikkeling omgeving zijn niet geschijden van elkaar |

Tabel 4: Voor-en nadelen Web2Py [7]

## 3.3 CherryPy

CherryPy is een op Python gebaseerd, object georiënteerd framework voor webontwikkeling. Het biedt ingebouwde plugins en een krachtig configuratiesysteem. CherryPy bevat ook een implementatie van de Ruby framework en bevat het Ruby on Rails-routeringssysteem in Python. URL’s kunnen worden gegenereerd en toegewezen aan controllers. CherryPy vereist minimale regels om de broncode te schrijven, wat leidt tot een korte ontwikkeltijd en hoge productiviteit. Het is een zeer licht alternatief voor Django en een van de oudste Python-frameworks. CherryPy heeft ingebouwde tools voor het creëren van statische webpagina's, URL-codering, gebruikersauthenticatie en sessiebeheer. Het is ingewerkt met een krachtig configuratiesysteem dat meerdere HTTP-verzoeken kan verwerken tegenover Django en meerdere HTTP-servers tegelijk kan uitvoeren [5].

Figuur 8: CherryPy voorbeeld code [22]

|  |  |
| --- | --- |
| **Voordelen** | **Nadelen** |
| Zeer flexibel | Geen ORM |
| Goede performantie | Performantie hangt af van aantal/soort geinstalleerde plugins |
| Kleine leercurve | Geen ingebouwd admin paneel |
| Ondersteund IIS, Apache en WSGI | Weinig tot geen up-to-date documentatie/voorbeelden |

### 3.3.1 Voor-en nadelen CherryPy

Tabel 5: Voor-en nadelen CherryPy [8]

### 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 productievere web frameworks die kunnen worden gebruikt voor een meer performante en eenvoudig flexibele website. 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 verrijst [5].

# 5 Python virtuele omgevingen

Wat zijn ze, wat is het voordeel en waarom worden ze gebruikt?

 Een virtuele Python omgeving bestaat uit twee essentiële 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 9: Python virtuele omgevingen visueel voorbeeld [9]

In bovenstaande afbeelding (Figuur 9: Python virtuele omgevingen visueel voorbeeld [9]) is een voorbeeld van wet 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.

# 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 betekend 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 van een kort venster of niet, er is altijd kans dat de persoon die de configuratie installeert een ​​kleine fout of storing van 2 minuten kan veranderen in een storing van 2 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 10: 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 11: 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 12: Zero downtime applicatie-update [20]

Zoals de lezer kan zien 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 door behulp van de load balancer. Wanneer de ontwikkelaar gebruik maakt van DVCS 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 variabele 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 Continue levering

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. Om 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 continue levering is dat projecten en ideeën uiteindelijk in gerbuik worden genomen door de eindgebruiker, wanneer een update gebreurt 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 er wel belang aan 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 in de algemene industrie.

Jenkins is een open source platform en heeft net als Django een sterke gemeenschap achter zich staan, voor enige bijstand dat een ontwikkelaar of DevOps-engineer zou nodig hebben kan worden gerekend op publieke forums. Opnieuw omdat het open source 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 [22] [15]) CITATION SMi22 \l 2067 [16].

Figuur 13: 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 dat moeten gebeuren in Jenkins om pipeline(s) operationeel/actief te houden is de repository bron URL-aanpassen.



Figuur 14: 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 SEQ Tabel \\* ARABIC 6: Jenkins minimum hardware voorwaarden

De hardware voorwaarden van bovenstaande tabel ( REF \_Ref103437174 \h 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 ( REF \_Ref103438413 \h Tabel 7: Jenkins minimumsoftware voorwaarden) is een oplijsting van de verschillende minimumvoorwaarden op valk van software CITATION J22 \l 2067 [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 SEQ Tabel \\* ARABIC 7: Jenkins minimumsoftware voorwaarden CITATION J22 \l 2067 [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 gemeenschapsondersteuning 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 CITATION SMi22 \l 2067 [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

BIBLIOGRAPHY

|  |  |
| --- | --- |
| [1] | JavaPoint, „Django history - javapoint,” [Online]. Available: https://www.javatpoint.com/django-tutorial. [Geopend 22 03 2022]. |
| [2] | Wikipedia contributors, „Django (web framework),” 05 04 2022. [Online]. Available: https://en.wikipedia.org/wiki/Django\_(web\_framework). [Geopend 13 04 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] | „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]. |
| [5] | 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]. |
| [6] | W. Semik, „Flask vs. Django - STXNEXT,” 29 11 2021. [Online]. Available: https://www.stxnext.com/blog/flask-vs-django-comparison/. [Geopend 10 05 2022]. |
| [7] | „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]. |
| [8] | „Compare CherryPy vs web2py,” 13 02 2021. [Online]. Available: https://codeahoy.com/compare/cherrypy-vs-web2py. [Geopend 10 05 2022]. |
| [9] | 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]. |
| [10] | 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]. |
| [11] | R.H Ansible, „About Ansible - Ansible documentation,” [Online]. Available: https://docs.ansible.com/ansible/latest/index.html. [Geopend 03 11 2021]. |
| [12] | R.H Ansible, „Red Hat Ansible use case - Provisioning,” [Online]. Available: https://www.ansible.com/use-cases/provisioning. [Geopend 03 11 2021]. |
| [13] | R.H. Ansible, „Ansible for Configuration management,” [Online]. Available: https://www.ansible.com/use-cases/configuration-management. [Geopend 03 11 2021]. |
| [14] | R.H. Ansible, „Ansible For Continuous Delivery,” [Online]. Available: https://www.ansible.com/use-cases/continuous-delivery. [Geopend 03 11 2021]. |
| [15] | 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]. |
| [16] | 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]. |
| [17] | 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]. |
| [18] | 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]. |
| [19] | 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]. |
| [20] | R.H. Ansible, „Ansible for application deployment,” [Online]. Available: https://www.ansible.com/use-cases/application-deployment. [Geopend 03 11 2021]. |
| [21] | „Web2py Python Framework (Youtube),” 01 08 2014. [Online]. Available: https://www.youtube.com/watch?v=yKQNmutC0WY. [Geopend 28 04 2022]. |
| [22] | P. ReviewDesk, „CherryPy - Python Web Framework,” 13 04 2022. [Online]. Available: https://www.predictiveanalyticstoday.com/cherrypy/. [Geopend 28 04 2022]. |
| [23] | „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

