Project Hosting Designrapport

Yorrit Belmans
Tars van Roey
Sepp Eyckmans
Daan Snijders
Robbe Keppens

Table of Contents

1 Beschrijving van het probleem

De klant heeft een applicatie die gehost moet worden op een betrouwbare, veilige en schaalbare omgeving. Momenteel ontbreekt een gestructureerde hostingoplossing die voldoet aan de vereisten voor beveiliging, automatisering en hoge beschikbaarheid.

Zonder de juiste implementatie van securitymaatregelen, zoals de CIS Controls, loopt de applicatie risico op cyberdreigingen en datalekken. Daarnaast is het essentieel dat de hostingomgeving flexibel kan opschalen en hoge beschikbaarheid biedt, zodat de applicatie stabiel blijft functioneren bij wisselende belasting. Automatisering is noodzakelijk om onderhoud en beheer efficiënt te laten verlopen en menselijke fouten te minimaliseren.

Het project richt zich op het opzetten van een hostingoplossing die voldoet aan deze eisen en de applicatie optimaal ondersteunt.

2 Toepassingsgebied van het project

Must haves:

- Implementatie van CIS Controls 1-8, 10 en 11 voor beveiliging.
- Hosting via een gedeelde serveromgeving (shared hosting) met gegarandeerde prestaties en beveiliging.
- Automatisering van updates, back-ups en monitoring om de operationele efficiëntie te verbeteren.
- Hoge beschikbaarheid (HA) door middel van redundante infrastructuur.

Should haves:

- Schaalbare infrastructuur die automatisch resources toevoegt bij verhoogde belasting.
- Uitgebreide logging en monitoring voor beveiliging en prestaties.

Could haves:

- Extra beveiligingsmaatregelen zoals netwerksegmentatie en geavanceerde DDoS-bescherming.
- Optimalisaties voor performance tuning en caching.

Won’t haves:

- Ontwikkeling of wijziging van de applicatie zelf; dit project richt zich uitsluitend op de hosting en beveiliging.

3 Stakeholder analyse

- **De klant (leerkracht als opdrachtgever)**
  - **Verwachtingen**: Een duidelijke, goed onderbouwde hostingoplossing met de juiste beveiligingsmaatregelen.
  - **Verantwoordelijkheden**: Beoordeling van het project en feedback geven op de implementatie.
- **Het projectteam (studenten)**
  - **Verwachtingen**: Een leerzame ervaring in hosting, security en automatisering.
  - **Verantwoordelijkheden**: Ontwerpen, implementeren en documenteren van de hostingomgeving, inclusief beveiligingsmaatregelen en automatisering.

4 Analyse van de behoeften

4.1 Functionele eisen

- De klant moet een applicatie kunnen deployen.
- De klant moet meerdere applicaties kunnen hosten gescheiden van elkaar.
- De klant moet zijn applicatie offline kunnen halen als hij dit wil.
- De klant moet een account kunnen aanmaken/beheren/verwijderen.
- De klant moet zijn applicaties kunnen beheren via een API.
- De klant moet de mogelijkheid hebben tot het aanmaken van een database.
- De klant moet de mogelijkheid om de database te bereiken en modificeren.

4.2 Niet-functionele eisen

4.2.1 Performantie

- ***Responsetijden***

Ervoor zorgen dat de applicatie van de klant altijd snel en stabiel toegankelijk is, zelfs tijdens piekbelastingen en bij kritisch verkeer. Dit kan worden gerealiseerd met behulp van load balancers en Kubernetes Ingress.

- ***Efficiëntie***

Verzekeren dat alle applicaties die op het hostingplatform worden gehost, optimaal presteren en efficiënt draaien. Dit volbrengen we via het implementeren van monitoring en dit continu te monitoren van onze prestaties en het implementeren van verbeteringen om de stabiliteit, snelheid en betrouwbaarheid van het platform te waarborgen.

- ***Resource beheer***

Het efficiënt beheren en eerlijk verdelen van onze serverresources over verschillende applicaties. Zo zorgen we ervoor dat elke applicatie over voldoende capaciteit beschikt om optimaal te presteren en beheerd te worden. Dit kan realiseert worden via Kubernetes Deployment.

4.2.2 Automatisatie

- ***Deployment***

De klant moet eenvoudig zijn applicatie kunnen uploaden naar onze repository, waarna automatisch een volledig gehoste applicatie met een eigen omgeving wordt opgezet, zonder dat handmatige acties van de klant nodig zijn voor de deployment. Dit word volbracht via GIT Deployment Templates en met hulp van automatie tools (Ansible, Puppet,…).

- ***Monitoring & Alerts***

Elke applicatie die op ons platform wordt gehost, krijgt continue uptime-monitoring. Mocht er een probleem optreden met de applicatie van de klant, dan ontvangen we direct een alert, zodat we proactief kunnen ingrijpen. We maken hiervoor gebruik van Uptime Kuma en verzamelen waardevolle loggegevens via Grafana Loki, wat essentieel is voor een effectief monitoringsproces.

- ***Automatisch (Security) Updates***

Elk klantsysteem zal een automatisch systeem- en beveiligingsupdateproces implementatie krijgen. Dit zorgt ervoor dat de klantomgeving altijd up-to-date blijft zonder dat de klant dit manueel moet doen. We garanderen dit door gebruik te maken van automatiseringstools en Systemd-functionaliteiten zoals timers en services.

- ***Automatisch Back-up***

Elke klantapplicatie krijgt een uniek geautomatiseerd back-upproces. Dit zorgt ervoor dat klantgegevens bij dataverlies kunnen worden hersteld en totale data loss wordt voorkomen met minimum manuele input. Dit realiseren we met geautomatiseerde tools en een secundaire back-upopslag om de back-ups op te bewaren.

- ***Automatisch Recovery***

Bij dataverlies moet er naast het back-upproces ook een recoveryproces zijn om de data correct te herstellen. Wij garanderen voor elke klantapplicatie een uniek recoveryproces met minimale manuele input. Dit realiseren we met geautomatiseerde tools en een secundaire back-upopslag, waar de back-ups veilig worden bewaard voor herstel.

- ***Automatisch Schaling***

Ons hostplatform maakt gebruik van automatische schaalvergroting, zodat de applicatie zich bij piekverkeer dynamisch aanpast. Dit garandeert een naadloze gebruikerservaring en constante optimale prestaties, zonder impact voor de klant. Via Kubernetes Deployment zal het schaling proces automatisch kunnen verlopen.

4.2.3 Gebruikerservaring

- ***Gebruiksvriendelijk***

We zorgen ervoor dat elke interactie met ons platform zo gebruiksvriendelijk mogelijk is, zodat klanten hun applicaties moeiteloos kunnen uploaden en beheren. Dit doen we door het GIT Deployment proces en de GIT actions duidelijk te documenteren en uit te leggen zodat de klant hier eenvoudig mee kan werken.

- ***Deploy Templates***

Om het deploymentproces van applicaties zo soepel en gebruiksvriendelijk mogelijk te laten verlopen, met minimale handmatige input van de klant, maken we gebruik van GIT Deployment Templates en GIT actions.

4.2.4 Schaalbaarheid

- ***Horizontaal/Verticaal***

Het platform is ontworpen om zowel horizontale als verticale schaalbaarheid te ondersteunen. Kubernetes kan automatisch extra containers starten om piekverkeer op te vangen, en de onderliggende infrastructuur is elastisch opgezet. Dit zorgt ervoor dat applicaties altijd voldoende resources hebben, zonder onnodige verspilling.

- ***Load Balancing***

Om een gelijkmatige verdeling van het verkeer te garanderen en overbelasting van individuele servers te voorkomen, maken we gebruik van load balancing. Inkomend verkeer wordt slim gerouteerd naar de beschikbare containers. Hierdoor blijven applicaties responsief en stabiel, zelfs onder zware belasting.

- ***Elastisch***

De infrastructuur van ons platform is elastisch, wat betekent dat resources automatisch kunnen worden toe- of afgenomen op basis van de actuele belasting. Dit voorkomt verspilling van capaciteit en zorgt ervoor dat applicaties altijd optimaal presteren.

4.2.5 Herstelbaarheid

- ***Back-up***

Om het risico op volledig dataverlies te elimineren, bieden wij elke klant een op maat gemaakt en betrouwbaar back-upplan. Dit plan wordt afgestemd op de specifieke behoeften van uw bedrijf en omvat beveiligingsmaatregelen, automatische back-ups en recovery. Door deze back-ups te bewaren op een veilige 2de gecentraliseerd storage solution, bent u altijd verzekerd van een veilige en snelle gegevensherstelopties, ongeacht de situatie.

- ***High Availability***

We implementeren High Availability op ons platform, zodat bij een crash of storing van de klantapplicatie direct een vervangende kopie wordt gebruikt. Deze kopie versie is altijd up-to-date met de huidige primaire klantapplicatie. Dit volledige proces gebeurt automatisch en zonder downtime, waardoor de klant geen hinder ondervindt van prestatieproblemen.

- ***Failover***

Bij een storing wordt automatisch een failover-mechanisme geactiveerd. Dit betekent dat als een server of container uitvalt, het systeem onmiddellijk overschakelt naar een andere gezonde node.

- ***Redundantie***

Om de betrouwbaarheid en beschikbaarheid van het platform te garanderen, is redundantie een essentieel onderdeel. Dit houdt in dat kritische componenten meervoudig aanwezig zijn, zodat er geen single point of failure bestaat.

4.2.6 Compatibiliteit

- **Software support (OS, Frameworks, Opslag, ...)**

Momenteel ondersteunen we enkel PHP versies die nog ondersteund worden door het PHP Development Team. Op deze manier zorgen we ervoor dat we security maximaliseren binnen ons platform aangezien de ondersteunde versies nog regelmatig security updates krijgen. De database van de klanten zullen MySQL databases, op deze manier zorgen wij voor een uniform database systeem is binnen ons platform.

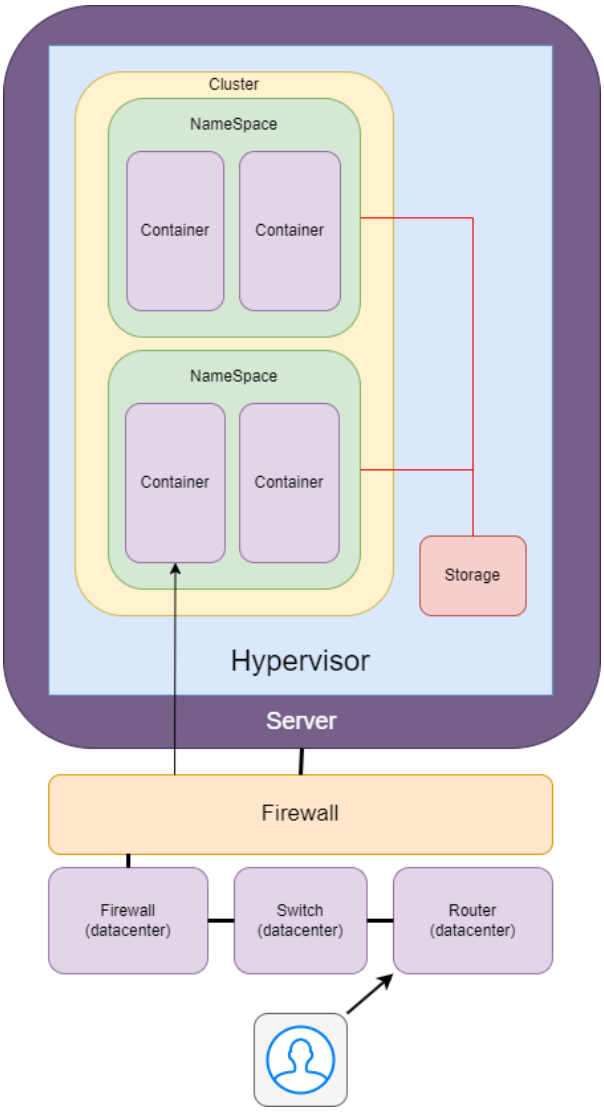
4.3 Security

4.3.1 CIS Controls

Om de veiligheid van het platform te garanderen, worden verschillende CIS Controls geïmplementeerd:

- **CIS Control 1 & 2 (Asset Management):** Automatische inventarisatie van alle hardware en softwarecomponenten binnen het platform.
- **CIS Control 5 (Account Management):** Toegangsbeheer via **Authentik** en Role-Based Access Control (RBAC).
- **CIS Control 7 (Web en E-mailbeveiliging):** Netwerkverkeer wordt gefilterd via **pfSense** om ongewenste verbindingen te blokkeren.
- **CIS Control 8 (Malware Defense): ClamAV** wordt gebruikt om bestanden op malware te scannen.

5 Hardware Schema



Ons hardware schema is een high level representatie van de hardware die we gebruiken, ook wordt er aangegeven welk traject de data zal volgen wanneer een klant connecteert met zijn applicatie.

Als we dit traject volgen komen we eerst bij de netwerkvereisten van ons datacenter. Hier staat een router met firewall en verschillende switchen.

Na deze initiële datacenter apparaten komen we aan onze fysieke servers. Hier zal opnieuw een firewall staan zowel onze servers als de klant applicaties zal beveiligen. Verder bestaat onze server uit een hypervisor (Proxmox). Hierop zullen al onze applicaties gedraaid worden.

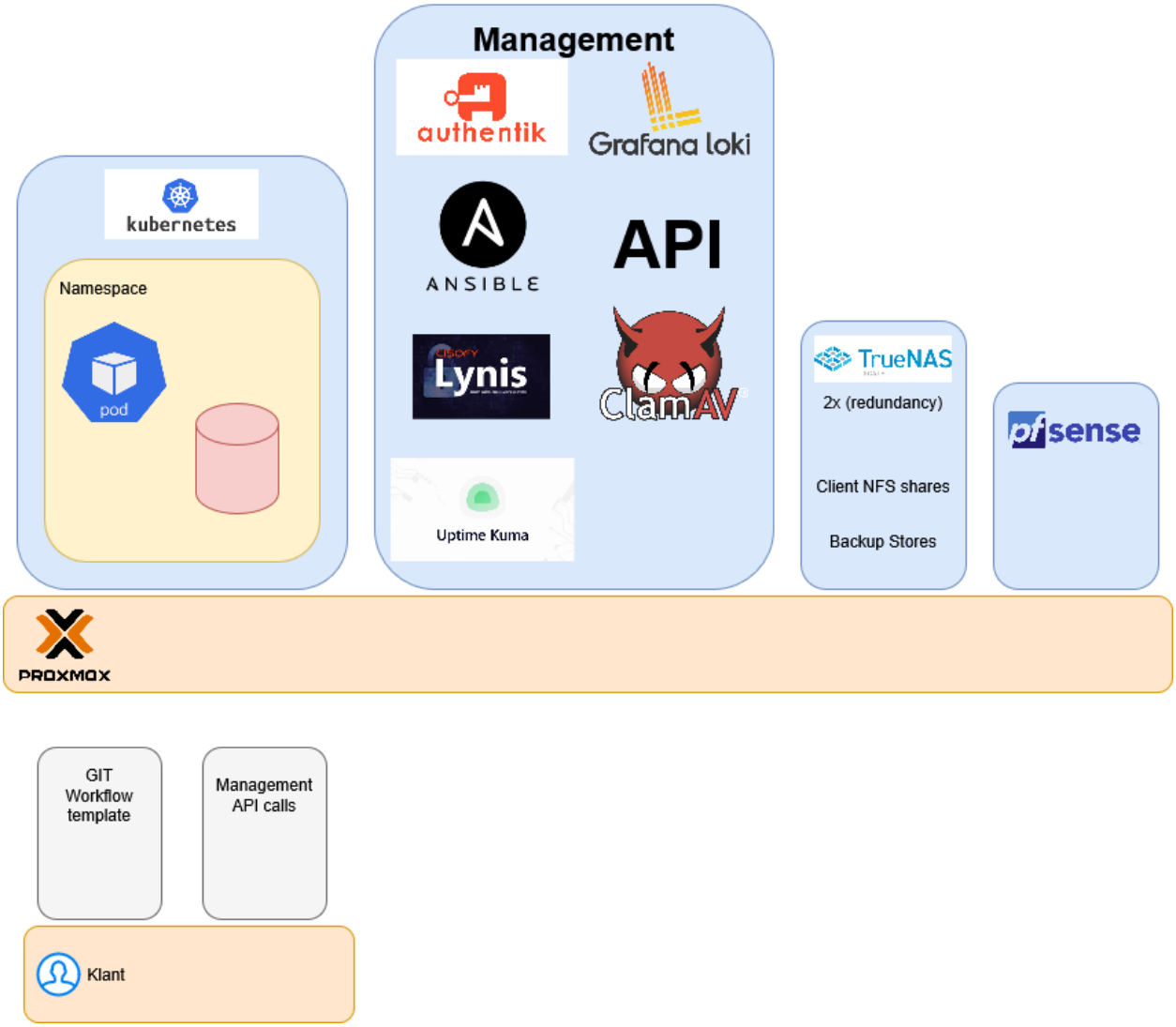
Onze hypervisor zal beschikken over meerdere virtuele machines. Zoals onze Kubernetes cluster, storage machines, firewall en andere managementapplicaties.

De Kubernetes cluster zal bestaan uit namespaces met daarin de applicaties van de klant. Elke klant zal ook zijn eigen deel opslag krijgen op de storage servers waar hun applicaties toegang tot hebben.

In het software diagram zullen deze zaken nog dieper toegelicht worden.

6 Softwareschema

Ons softwareschema bestaat uit al de verschillende software waarvan we gebruik maken in het hosting platform.



6.1 Kubernetes Cluster

Het hosting platform maakt gebruik van Kubernetes clusters. In deze cluster worden verschillende namespaces aangemaakt, elke klant zal zijn eigen namespace toegewezen krijgen waarin hij zijn applicaties kan opzetten.

Er wordt gebruik gemaakt van Kubernetes omdat dit veel mogelijkheden biedt. Zo maken we gebruik van high availability, failover, autoscaling. Dit maakt het mogelijk om een robuuste infrastructuur op te zetten.

7 Management Software

7.1.1 Authentik

Deze software zal gebruikt worden voor authenticatie. Alle verschillende klanten zullen hier een account kunnen aanmaken. Door middel van Acces Control Lists zullen de verschillende klanten en werknemers gescheiden worden van elkaar.

Authentik zal ook geïntegreerd worden in de andere tools die we gebruiken. Zo zal elke aparte namespace authenticatie met Authentik vereisten voordat er aanpassingen gedaan kunnen worden. Ook voor Proxmox, Uptime Kuma, API, Ansible en Grafana Loki zullen we eerst moeten authentiseren met Authentik voordat we toegang hebben tot de tool.

7.1.2 Grafana Loki

Grafana Loki zullen we gebruiken om alle logging van alle verschillende machines en applicaties te verzamelen. Deze tool maakt het ons gemakkelijk om na te gaan wanneer er fouten voorvallen en deze te vermeiden in de toekomst.

7.1.3 Ansible

Ansible zal de automatisatie doen voor ons hosting platform. We zullen het gebruiken om automatisch nieuwe applicaties te deployen, het aanmaken van nieuwe databases en het opzetten van nieuwe domeinnamen.

We maken hiervoor verschillende playbooks die we zullen laten uitvoeren door een API-call.

7.1.4 API

Er wordt gebruik gemaakt van een API geprogrammeerd in Python. De API zal gebruikt worden voor het managen van de verschillende applicaties door de klant. Dit wordt later nog verder toegelicht.

7.1.5 Lynis

Deze tool zal gebruikt worden om vulnerability scans uit te voeren op al onze systemen. Dit helpt de security hoog te houden.

7.1.6 ClamAV

ClamAV zal gebruikt worden om malware scans uit te voeren. Op al onze systemen zal dit periodiek uitgevoerd worden om te scannen naar bekende malware bestanden. Deze tool help ook voor de security hoog te houden.

7.1.7 Uptime Kuma

Deze tool zal een de uptime van al onze systemen bijhouden. Ook zal deze tool automatisch meldingen sturen wanneer een systeem down is. Dit zal ons helpen de uptime te maximaliseren.

Verder zal er ook een status pagina beschikbaar zijn voor elke klant waar al zijn applicaties met uptime te bekijken zijn.

7.1.8 Storage

In ons hosting platform zullen we TrueNAS gebruiken om onze storage te beheren. We zullen twee TrueNAS virtuele machines aanmaken voor redundancy. De TrueNAS zal nfs shares aanmaken per klant en verbonden worden met zijn applicaties in kubernetes.

Ook zullen onze backups opgeslagen worden op TrueNAS.

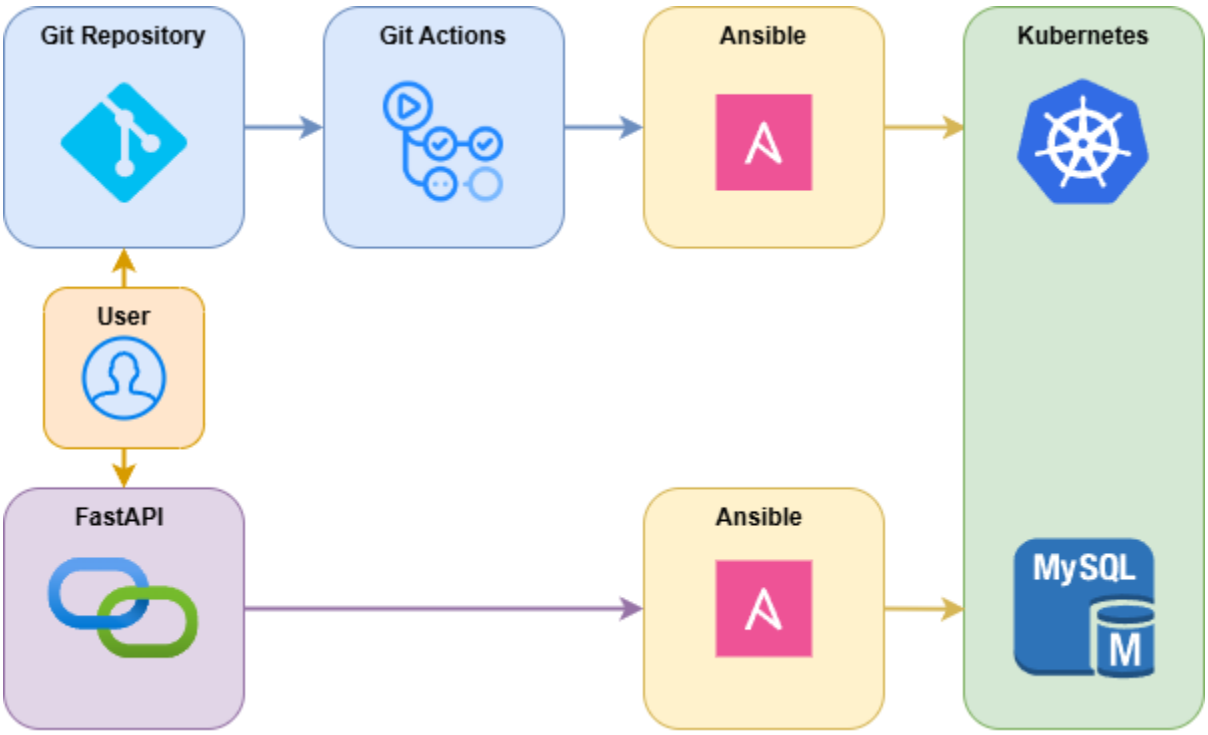
8 Interactie met klant

8.1 Git deployment process

De kant krijgt een Git repository. In deze repository heeft de klant de mogelijkheid voor zijn code in templates te plaatsen, deze wordt automatisch geïntegreerd in ons systeem. De automatische integratie gebeurt via Git actions, dit betekent dat de website altijd de code bevat die op dat moment in de templates staan. Op deze manier zorgen wij ervoor dat de klant op elk gegeven moment zijn webapplicatie kan bijwerken zonder dat hier enige extra moeite voor verreisd is.

8.2 Management process

De klant heeft ook de mogelijkheid voor gebruik te maken van API-calls. Via deze kan de klant de status van zijn applicatie beheren. Dit betekent dat de klant de mogelijkheid heeft voor zijn applicatie te starten, te pauzeren, opnieuw starten en live data van de applicatie op te vragen. De klant heeft ook de mogelijkheid voor via deze API-calls een domeinnaam en een database aan te vragen of te creëren.



Bibliography

Ansible. (n.d.). Retrieved from https://docs.ansible.com/

Authentik. (n.d.). Retrieved from https://goauthentik.io/

ClamAV. (n.d.). Retrieved from https://www.clamav.net/

Docker. (n.d.). Retrieved from https://www.docker.com/blog/how-to-use-your-own-registry-2/

FastAPI. (n.d.). Retrieved from https://fastapi.tiangolo.com/

Grafana Loki. (n.d.). Retrieved from https://grafana.com/oss/loki/

Kubernetes. (n.d.). Retrieved from https://kubernetes.io/

Lynis. (n.d.). Retrieved from https://cisofy.com/lynis/

PfSense. (n.d.). Retrieved from https://www.pfsense.org/

Proxmox. (n.d.). Retrieved from https://www.proxmox.com/en/

TrueNAS. (n.d.). Retrieved from https://www.truenas.com/

Uptime Kuma. (n.d.). Retrieved from https://uptime.kuma.pet/