

STAGE CEGEKA

Realisatiedocument

Arne Madalijns – 3CCS01

2024- 2025

Inhoudstafel

1. INLEIDING	3
2. ANALYSE	4
2.1. Azure Network Fundamentals	4
2.2. Load Balancing concepts	5
2.3. F5 HA-cluster in Azure	6
2.4. Vergelijkende Analyse ALB en F5	9
2.5. Automatisatie met Terraform	10
2.6. Configuratie met Ansible	11
2.7. Application Gateway	12
2.8. Traffic Manager	13
2.9. Load Balancing in de Praktijk (eindanalyse)	15
3. REALISATIE	18
3.1. Introductie cloud networking & load balancing	18
3.2. Cross-Region Load Balancer	19
3.3. F5 HA-cluster in Azure	20
3.4. Vergelijkende Analyse ALB en F5	21
3.5. Automatisatie met Terraform	22
3.6. Configuratie via Ansible	24
3.7. Application Gateway	25
3.8. Traffic Manager	26
3.9. Load Balancing in de Praktijk (einddocument)	27
4. EXTRA OEFENING	28
5. BESLUIT	29
6. BIJLAGE	30
7. LITERATUURLIJST	31

1. Inleiding

Dit document beschrijft de **analyse** en **realisatie** van mijn stage die gericht was op het uittesten van verschillende load balancers en het kiezen van **de juiste oplossing binnen Azure**. In het **projectplan** werd de nadruk gelegd op de **stijgende vraag** naar schaalbare en efficiënte cloudoplossingen en het belang van **het kiezen van de juiste load balancing-methode** per use case aangevuld met een WRM (Weighted Ranking Method).

Dit onderzoek richt zich niet alleen op **Azure's native load balancing-oplossingen**, maar ook op **F5 load balancers**, die binnen Cegeka in on-premises omgevingen worden gebruikt, maar nog niet eerder in Azure zijn opgezet of onderzocht.

In dit document zal ik stap voor stap de uitvoering van mijn onderzoek toelichten. Dit omvat:

1. **Onderzoek en analyse**
 - Bestuderen van **Azure network fundamentals** en **load balancing-concepten**.
 - Vergelijking van de verschillende **Azure load balancers** en hun functionaliteiten.
 - Introductie tot **F5 load balancers** en hoe deze werken binnen een cloudomgeving.
2. **Praktische implementatie en tests**
 - Opzetten van een **cross-region Azure Load Balancer** om verkeer te verdelen over meerdere regio's en automatische failover te testen.
 - Implementeren van een **High Availability (HA) F5 Load Balancer** in Azure en configureren in **active/standby modus**.
 - Configureren en testen van **failover-scenario's** tussen Azure Load Balancer en F5.
 - Automatische opstelling met behulp van Terraform en Ansible.
3. **Vergelijkende analyse en evaluatie**
 - Gedetailleerde vergelijking tussen **F5 in het Cegeka datacenter** en in **Azure** op basis van **performance, features, kosten en use cases**.
 - Testen van **automatiseringsmogelijkheden met Terraform** voor een efficiënte en herhaalbare implementatie.

Door middel van deze realisatie wordt niet alleen onderzocht hoe F5 geïmplementeerd kan worden in Azure, maar wordt ook een diepgaand inzicht verkregen in **de voordelen en beperkingen van verschillende load balancers** binnen Azure. Dit document dient als korte samenvatting, referentiepunt voor toekomstige implementaties en als basis voor **mogelijke uitbreiding van F5-diensten binnen Azure** bij Cegeka.

2. Analyse

Vooraleer ik startte met de praktische uitvoering van mijn stageopdracht, heb ik meerdere analyses gemaakt over de verschillende load balancing-oplossingen binnen Microsoft Azure en heb ik F5 BIG-IP onderzocht, met als doel de meest geschikte technologieën te identificeren voor toekomstige toepassingen binnen Cegeka.

Zowel Azure-native oplossingen (Load Balancer, Application Gateway, Traffic Manager) als F5, momenteel enkel on-premises gebruikt, worden besproken. De focus ligt op de kenmerken, typische use cases, inzetbaarheid én een objectieve vergelijking via WRM.

Tot slot worden er diepgaande vergelijkingen gemaakt tussen Application Gateway (AGW) en F5, en tussen F5 on-premises en F5 in Azure, inclusief kosten, schaalbaarheid en security.

Deze analyse vormt de inhoudelijke basis van de keuzes en aanbevelingen die tijdens de stage werden gemaakt.

2.1. Azure Network Fundamentals

Om een goed onderbouwd onderzoek te voeren naar load balancing in Azure, was het belangrijk om eerst de netwerkarchitectuur binnen Azure grondig te begrijpen. Ik verdiepte me in de **Azure Networking Fundamentals**, met een focus op de belangrijkste componenten die de basis vormen voor elk netwerk in Azure: **Virtual Networks (VNETs)**, **Subnets** en **Network Security Groups (NSG's)**.

- **Virtual Networks (VNETs)** vormen de ruggengraat van het Azure-netwerk. Ze zorgen voor een logisch geïsoleerde omgeving waarin resources met elkaar kunnen communiceren.
- **Subnets** bieden de mogelijkheid om een VNET te segmenteren in kleinere delen. Dit vergemakkelijkt het organiseren van resources, het beperken van verkeer en het beter toepassen van beveiliging.
- **Network Security Groups (NSG's)** maken het mogelijk om het verkeer te controleren aan de hand van regels die inbound- en outboundverkeer toelaten of blokkeren. NSG's vormen dus een belangrijk onderdeel van de beveiliging van het netwerk.

Deze theoretische heropfrissing was noodzakelijk om in latere fases load balancers **correct en veilig** te kunnen configureren binnen een bestaande Azure-omgeving.

2.2. Load Balancing concepts

Aangezien mijn stageproject zich richt op het opzetten en vergelijken van load balancing-oplossingen, was het belangrijk om eerst te begrijpen wat load balancing precies inhoudt, op welke lagen het plaatsvindt en welke technologieën beschikbaar zijn binnen Azure en daarbuiten.

Wat is Load Balancing?

Bij Load Balancing wordt binnenvkomend netwerkverkeer gelijkmatig verdeeld over meerdere backend-servers. Dit verhoogt de **beschikbaarheid**, **betrouwbaarheid** en **schaalbaarheid** van applicaties. Er bestaan twee belangrijke lagen waarop load balancing wordt toegepast:

- **Layer 4 (transport):** Verkeer wordt verdeeld op basis van **IP-adres en poortnummer** (bijv. TCP/UDP). Deze aanpak is snel en efficiënt, maar biedt beperkte mogelijkheden tot inspectie of herschrijven van het verkeer.
- **Layer 7 (application):** Verkeer wordt verdeeld op basis van **inhoud** zoals HTTP-headers, URL's of cookies. Dit laat meer geavanceerde routing toe, zoals path-based routing of session affinity.

Oplossingen binnen Azure

Azure biedt verschillende native oplossingen voor load balancing, elk met hun eigen focus en voordelen:

- **Azure Load Balancer (L4)**
 - Verdeelt verkeer op basis van IP-adressen / poorten.
 - Voornamelijk geschikt voor intern verkeer, maar kan ook voor extern verkeer gebruikt worden.
 - Ideaal voor eenvoudige scenario's.
- **Application Gateway (L7)**
 - HTTP(S)-based routing met features als SSL-termination, URL-routing en session affinity.
 - Makkelijk te integreren met Web Application Firewall (WAF) voor extra beveiliging.
- **Traffic Manager (DNS-based)**
 - Verdeelt verkeer op basis van DNS naar meerdere regio's.
 - Routingmethodes zoals geographic, weighted, performance, etc.
 - Wordt gebruikt voor globale failover en regionale verspreiding van resources.

Onderzoek naar F5 BIG-IP

Naast de Azure-native tools heb ik ook **F5 BIG-IP** onderzocht als externe, geavanceerde load balancing-oplossing. F5 wordt vaak ingezet in complexe omgevingen waar diepgaande pakketcontrole vereist is. Het ondersteunt zowel **layer 4 als layer 7** load balancing, en biedt:

- Krachtige traffic management via iRules (scripting).
- Sterke ingebouwde beveiliging.
- High Availability (HA) via Active-Active of Active-Standby.
- Flexibel inzetbaar in fysieke, virtuele of cloudomgevingen.

Door deze concepten te onderzoeken, heb ik de technische basis gelegd voor de volgende stappen van mijn stage: het **vergelijken van load balancers in concrete scenario's**, het **documenteren van hun inzetbaarheid binnen Cegeka** en het **uitwerken van een technische handleiding**.

2.3. F5 HA-cluster in Azure

In dit deel van mijn stageproject onderzocht ik hoe een enterprise-grade load balancing-oplossing zoals **F5 BIG-IP** kan worden geïntegreerd binnen een **Azure-cloudomgeving**, met specifieke focus op het opzetten van een **HA-architectuur**.

Waarom dit onderzoek?

In tegenstelling tot traditionele on-premises omgevingen waar **floating IP-adressen** gebruikt kunnen worden voor failover tussen load balancers, biedt Azure deze functionaliteit **niet** op dezelfde manier aan. Aangezien Cegeka ook klanten ondersteunt in hybride en cloud-only omgevingen, is het essentieel om te begrijpen hoe enterprise load balancers zoals F5 correct geconfigureerd kunnen worden binnen de beperkingen en mogelijkheden van Azure.

Denkproces

Tijdens deze taak onderzocht ik:

- Welke Azure VM-series geschikt zijn voor F5 en ondersteuning bieden voor 3 en 4 NIC's.
- Hoe NSG's correct moeten worden ingesteld per interface (management, external, internal en HA).
- Welke manieren er bestaan om HA te realiseren zonder floating IP's in Azure, zoals:
 - Azure Load Balancer (ALB) vóór de F5-instances
 - Cloud Failover Extension (CFE) van F5 zelf.

Een belangrijk onderdeel was ook het analyseren van de **verschillen in complexiteit, onderhoud en betrouwbaarheid** tussen deze HA-oplossingen.

Vergelijking HA met ALB en CFE

Tijdens het opzetten van een High Availability (HA) architectuur voor F5 BIG-IP in Azure onderzocht ik twee mogelijke methodes:

1. **Azure Load Balancer:** native oplossing die verkeer naar meerdere F5-instances stuurt op basis van health probes.
2. **Cloud Failover Extension:** Door F5 ontwikkelde extensie die via Azure API automatisch de secondary IP van een NIC verplaatst naar de active machine bij failover.

Kenmerken	Azure Load Balancer	Cloud Failover Extension
Failover	Health probe-based, ALB detecteert uitval	API-call verplaatst secondary IP naar actieve NIC
Actieve node	Active-Active of Active-Standby	Actieve node verwerkt altijd verkeer, standby is passief
Failover tijd	+/- 5 - 15sec	+/- 10-20sec (afhankelijk van de API latency en instellingen)
Complexiteit	Gemiddeld (standaard ALB setup)	Hoog (vereist IAM, managed identity, storage, tagging)
Automation mogelijkheid	Weinig extra configuratie	Meer extra configuratie
Betrouwbaarheid	Hoog (Azure-native)	Hoog, afhankelijk van de config
Monitoring	Azure Monitor + ALB metrics	CFE logs via storage account

Opzetvereisten van CFE

Voor CFE moet extra infrastructuur worden opgezet:

- **User Assigned Managed Identity** met specifieke RBAC-rollen.
- **Storage Account** waar CFE statusinformatie opslaat (failover triggers, IP-bindings, etc).
- **Correcte tagging** van resources zodat CFE weet welke resources er moet worden overgenomen bij een failover.
- **Installatie RPM package.**
- **Correct** gestructureerde **JSON file** op beide F5-instances en moet CFE worden geïnitialiseerd.

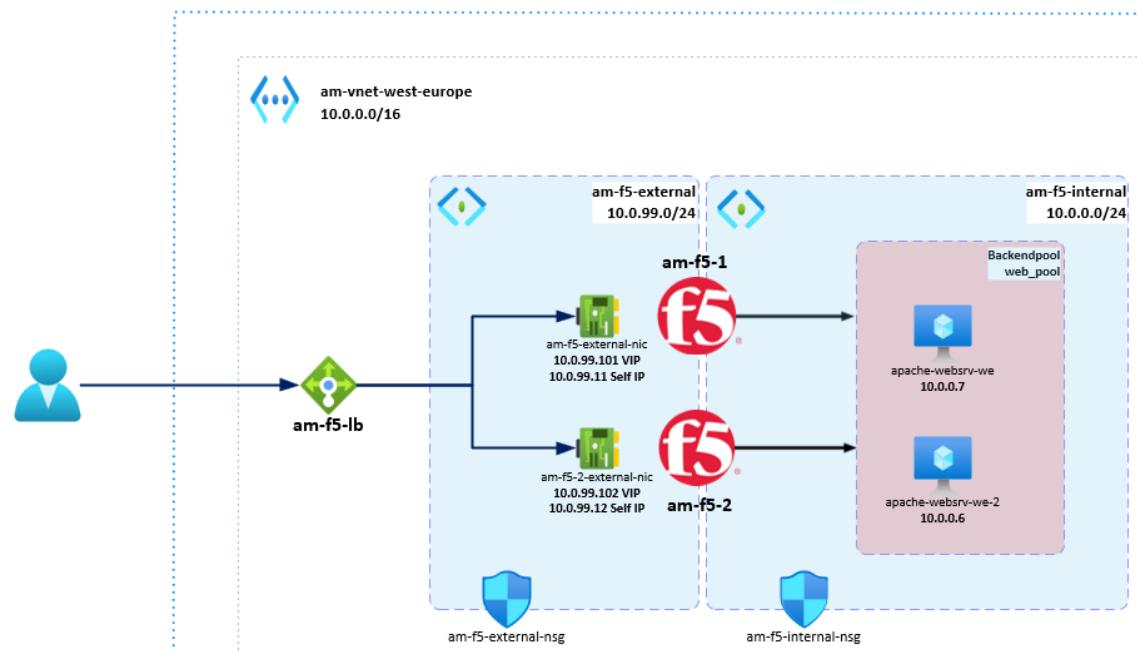
Afweging tussen ALB en CFE

ALB is **eenvoudiger** en **snel** in setup en goed geïntegreerd met andere Azure-services. Het is vooral geschikt wanneer eenvoud cruciaal is en het niet erg is dat beide F5-instances verkeer kunnen ontvangen zolang de backend-configuratie dat ondersteunt. Er dient wel telkens een uniek IP-adres te zijn voor elke Virtual Server, wat beheer en overzicht in F5 moeilijker maakt.

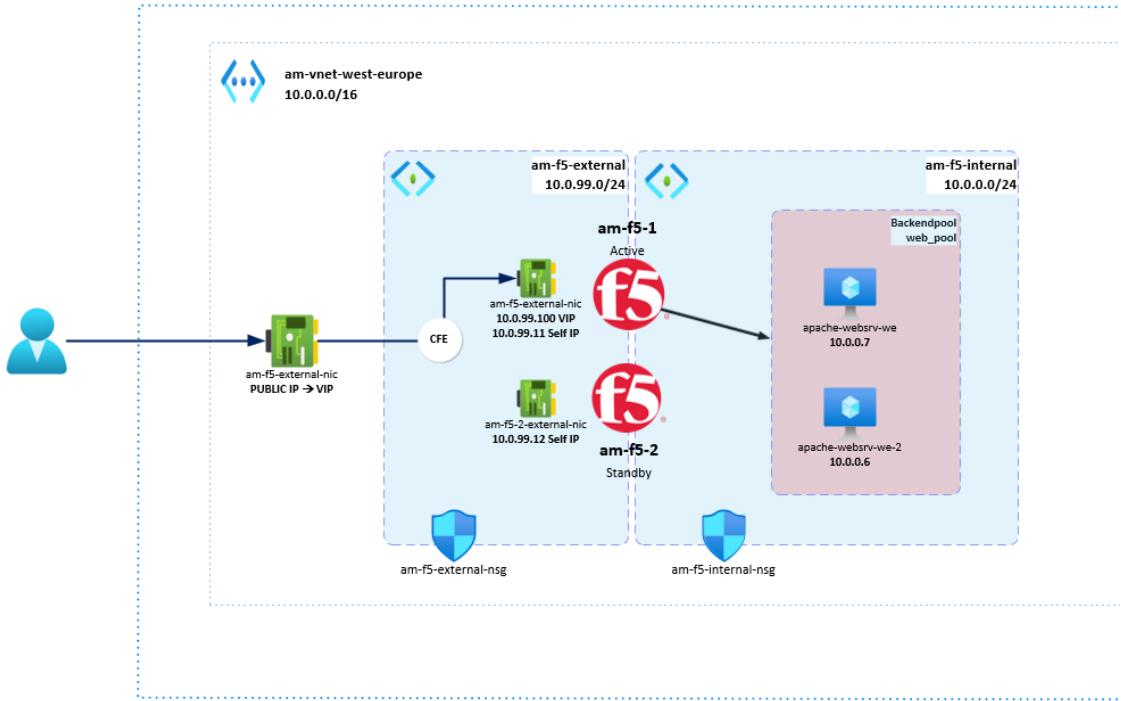
CFE biedt **een oplossing die dichter aanleunt bij een on-prem floating IP-concept**. Het zorgt ervoor dat de actieve node letterlijk het IP overneemt, en dus consistent blijft als enige echte toegangspunt. Dit is nuttig wanneer applicaties of backends verwachten dat het verkeer **steeds vanaf hetzelfde IP** komt. Nadelen zijn dat de opstart en configuratie complexer zijn en dat het overschakelen van IP-adressen niet altijd even snel verloopt.

Netwerkdiagrammen

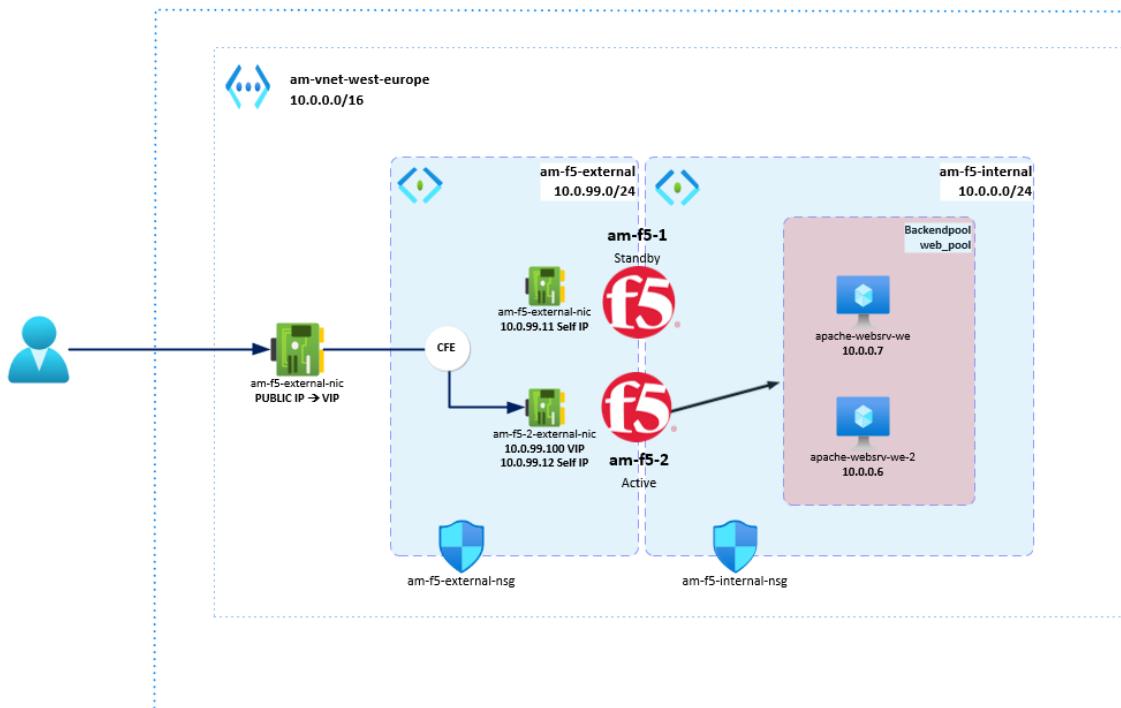
Hieronder zie je het netwerkdiagram voor een HA-opstelling met een **ALB**. De gebruiker maakt verbinding via het **publieke IP-adres van de ALB**, die op zijn beurt in de backend twee adressen heeft: de **Virtual Server IP's van beide F5-instances**. De **actieve F5 ontvangt het verkeer** en stuurt een **gezonde health check** terug. De **standby F5 daarentegen ontvangt geen verkeer** en geeft een **ongezonde health check** door, waardoor de ALB weet dat deze machine niet actief is.



De onderstaande netwerkdiagrammen tonen de **HA-opstelling via de CFE**. In deze configuratie is er slechts **één Virtual Server IP ingesteld**, dat aanvankelijk als secondary IP op de netwerkinterface van F5-1 staat. Bij een failover zorgt de CFE ervoor dat dit IP-adres **automatisch wordt verplaatst**: via een API-call naar Azure wordt het secondary IP dynamisch verplaatst naar de interface van de andere F5.



In het tweede diagram zien we dat **F5-2** inmiddels de **actieve machine** is geworden. Hij heeft het IP-adres **overgenomen** van F5-1 en ontvangt nu al het verkeer dat binnenkomt via **hetzelfde publieke IP-adres**.



Besluit

Beide methodes zijn toepasbaar binnen Azure, maar vereisen verschillende niveaus van kennis en beheer. Door beide opstellingen te documenteren en CFE te automatiseren via Terraform en Ansible, laat ik bij Cegeka code achter die herbruikbaar is en waarmee de hele omgeving eenvoudig kan worden opgezet.

2.4. Vergelijkende Analyse ALB en F5

In dit deel heb ik een gedetailleerde **analyse** gemaakt tussen **Azure Load Balancer (ALB)** en **F5** op het gebied van performance, features, configuratie & beheer, kostprijs en verduidelijkt met use cases.

Wat load balancing nu precies inhoudt en wat de verschillen zijn tussen layer 4 en layer 7, ga ik niet opnieuw uitleggen, want dit heb ik al eerder aangehaald. Daarom bespreek ik kort mijn belangrijkste ontdekkingen.

Concrete verschillen

Azure Load Balancer

- **Lage latency** binnen een regio door de integratie met Azure's backbone.
- Standaard beveiligd.
- Biedt **geen SSL-offloading** of andere layer 7-mogelijkheden.
- **Makkelijk** te configureren via Azure Portal, CLI of Terraform.
- **Zeer goedkope** optie, je betaalt enkel een paar euro's per maand voor de verbruikte resources.

F5 BIG-IP

- **Geavanceerde** traffic management.
- Biedt **wel SSL-offloading**.
- Afhankelijk van de licentie biedt F5 **ingebouwde** WAF, DDoS-bescherming en andere security features.
- **Global load balancing** met Global Traffic Manager (GTM).
- **Complexere configuratie** waardoor er specifieke kennis vereist is.
- Kan worden geconfigureerd via de WebUI, CLI, API, Ansible, ...
- **Zeer dure optie**, licenties variëren van enkele duizenden tot tienduizenden euro's.

Use Cases ALB

Stel dat u een simpele webserver hebt en redundantie wilt garanderen. U besluit om drie identieke websites op verschillende machines te implementeren. Om binnenkomend verkeer gelijkmatig over deze servers te verdelen, gebruikt u ALB om het verkeer over **TCP-poort 80** te verdelen.

ALB is ook zeer effectief voor interne load balancing. Als u bijvoorbeeld **database-failover** nodig hebt, kan een interne ALB verkeer omleiden naar een secundaire database in geval van downtime. In dit scenario verdeelt de ALB efficiënt verkeer op **TCP-poort 5432 (PostgreSQL)** binnen het interne netwerk.

Use Cases F5

Een iRule in F5 die mobiele gebruikers naar een mobiele versie van de website stuurt, terwijl desktopgebruikers de volledige versie krijgen.

Maak met behulp van iControl een script in Python dat automatisch een IP-adres op een blocklist plaatst als het te veel 403-fouten veroorzaakt.

2.5. Automatisatie met Terraform

In dit deel van mijn stage heb ik onderzocht hoe ik mijn eerder opgebouwde F5 HA-omgeving, inclusief de CFE, kon automatiseren met behulp van Terraform. De motivatie achter deze analyse lag in het streven naar een **makkelijke, dubbele F5 deployment** die meteen geconfigureerd kon worden.

Onderzoek aanpak

Mijn eerste stap was het in kaart brengen van de mogelijkheden om F5 BIG-IP-omgevingen via Terraform te deployen. Al snel kwam ik terecht op de [officiële GitHub-pagina van F5](#), waar enkele modules beschikbaar zijn voor verschillende soorten F5-deployments.

Hoewel deze modules in theorie een snelle oplossing zouden moeten bieden, merkte ik dat de documentatie **onvolledig** en vaak **verwarrend** was. Vooral het correct toevoegen van extra NICs bleek complex. Hierdoor besloot ik om niet volledig afhankelijk te zijn van de bestaande module, maar in plaats daarvan het infrastructuur **stapsgewijs zelf te bouwen** met de Azure-provider in Terraform.

Tijdens dit proces onderzocht ik welke resources en parameters exact nodig waren voor het opzetten van een F5-instance in Azure. Een belangrijk onderdeel daarvan was het achterhalen van de juiste **image en SKU**.

Dit kon ik doen via het Azure CLI-commando:

```
az vm image list --publisher f5-networks --offer f5-big-ip-good --all --out table
```

Met deze informatie kon ik verder een eerste versie van mijn Terraform-configuratie opstellen die een enkele Virtuele Machine (VM) met een F5-image aanmaakt, inclusief het beheer van de NIC's, IP-forwarding en diskconfiguratie.

Vaststellingen

De grootste les uit dit deel van mijn stage was dat het gebruik van **externe modules niet altijd** de beste keuze is, zeker wanneer de documentatie gebrekig is. Door het infrastructuur van nul op te bouwen, kreeg ik **een veel beter inzicht** in hoe de onderliggende resources samenwerken, hoe ze gekoppeld worden en hoe je ze correct moet configureren voor specifieke use cases zoals F5 HA.

Daarnaast leerde ik **herbruikbare Terraform-code** schrijven. De eerste versie van mijn configuratie was lang, repetitief en moeilijk te onderhouden. Daarom maakte ik een eigen Terraform-module waarin ongeveer **90%** van de benodigde configuratie was verwerkt. Dankzij het gebruik van **variabelen** kon ik met slechts een vijftiental lijnen code een volledig nieuwe F5-machine aanmaken.

2.6. Configuratie met Ansible

In dit deel van mijn stage onderzocht ik hoe ik de volledige configuratie van een F5 HA-omgeving in Azure kon automatiseren met **Ansible**, aansluitend op het infrastructuur dat ik eerder met Terraform had opgezet. De motivatie voor deze analyse lag in het streven naar **volledige automatisatie**: van provisioning tot configuratie, zonder manuele stappen. Cegeka werkt ook met een “automation first”-mentaliteit om fouten te voorkomen, tijd te besparen en om sneller op schaal te kunnen werken.

Onderzoek aanpak

Mijn eerste stap was het verkennen van bestaande Ansible-modules voor F5 BIG-IP via de officiële documentatie van Ansible zelf. De bedoeling was om bestaande modules te gebruiken voor het opzetten van VLANs, Self IPs, Pools, Virtual Servers, HA-configuratie en de installatie van de **CFE**. Tijdens het testen merkte ik echter dat veel van deze modules slechts **gedeeltelijk** werkten of **foutmeldingen** gaven, vooral bij het configureren van meer geavanceerde functionaliteiten zoals ConfigSync of device trust.

Daarom schakelde ik over naar een alternatief: **de Ansible command module** waarmee ik rechtstreeks **tmsh-commando's** naar de F5 kon sturen. Deze aanpak bleek betrouwbaarder en gaf me meer controle over de configuratieflow.

Voordat ik effectief met de playbooks aan de slag kon, moest ik zorgen voor een **werkende SSH-verbinding** naar de F5-instances. Deze machines kregen via Terraform **publieke IP's** toegewezen, die ik automatisch liet weergeven via de output. De bijbehorende NSG's werden zo geconfigureerd dat SSH-verkeer enkel vanaf mijn IP-adres werd toegelaten, wat een belangrijk beveiligingspunt is.

Vaststellingen

Tijdens dit deel van mijn stage merkte ik dat de **bestaande Ansible-modules** voor F5 BIG-IP vaak **beperkt** zijn en **soms fouten geven**, zeker bij geavanceerde configuraties zoals HA en device trust. Door over te schakelen naar de **command module** met directe **tmsh-commando's** kon ik betrouwbaarder en flexibeler werken.

Deze aanpak gaf me **meer controle en inzicht** in de F5-configuratie. Tot slot leerde ik hoe ik gestructureerde en herbruikbare playbooks kon opbouwen voor een volledig geautomatiseerde configuratieflow.

2.7. Application Gateway

Aangezien **F5 BIG-IP** al als Layer 4/7 oplossing was geïmplementeerd voor de HA-opstelling, wilde ik onderzoeken of Azure-native alternatieven en in het bijzonder de **Application Gateway** complementair ingezet kan worden.

Waarom Application Gateway?

Application Gateway is een **Layer 7 load balancer** die speciaal ontwikkeld is om **webverkeer** (HTTP/HTTPS) intelligent te beheren. Dit maakt het niet alleen mogelijk om verkeer te verdelen over meerdere backends, maar ook om verkeer te verdelen op basis van **URL-paths of headers**. In tegenstelling tot klassieke load balancers of netwerkgerichte oplossingen, biedt de App Gateway ook **SSL-offloading, URL-rewrites en custom error handling**.

Onderzoek

Mijn aanpak was om de App Gateway stapsgewijs te verkennen, telkens gefocust op één specifieke functionaliteit. Zo kon ik de **use-cases, beperkingen en complexiteit** per onderdeel analyseren.

1. Basic Load Balancing

Als eerste test maakte ik een klassiek scenario waarin HTTP-verkeer op poort 80 wordt verdeeld over twee backend webservers. Deze stap hielp bij het inschatten van hoe moeilijk de configuratie was, en vormt een goede “baseline” voor de complexere features.

2. Path-Based routing

Een belangrijke functie is **path-based routing**. Hiermee kan inkomend verkeer naar verschillende backend pools gestuurd worden op basis van het URL-pad. Een typisch scenario is het hosten van meerdere webapplicaties op één domein, waarbij /images/* naar één server voor afbeeldingen gaat en /videos/* naar een andere.

3. SSL-Termination

Dit houdt in dat het HTTPS-verkeer bij de App Gateway wordt gedecrypteerd, waarna het intern via HTTP wordt doorgestuurd naar de backend. De backend servers hoeven geen encryptie / decryptie meer uit te voeren en alle certificaten worden op één plek beheerd en vernieuwd.

4. Redirection Rules

Hiermee kan verkeer automatisch worden doorgestuurd, bijvoorbeeld van HTTP naar HTTPS of van de ene website naar de andere.

5. Rewrite Rules

Deze functie laat toe om inkomende headers te wijzigen of extra headers toe te voegen voor security of optimalisatie. Ik gebruikte dit onder meer om headers zoals X-Frame-Options, HSTS en Content-Type-Options te injecteren, wat een **duidelijke verbetering van de beveiligingscore** opleverde.

6. Custom Error Pages

Een andere functie is de mogelijkheid **custom error pages** in te stellen. In plaats van standaard foutcodes zoals 403 of 502 te tonen, kan men gepersonaliseerde HTML-pagina's configureren.

Listeners vs rules

Een **listener** bepaalt welk verkeer door de gateway wordt geaccepteerd, bijvoorbeeld HTTP-verkeer op poort 80 of HTTPS-verkeer op poort 443.

Een **rule** verbindt deze listener met een backend pool en bepaalt wat er met het verkeer moet gebeuren, zoals doorsturen naar een backend, omleiden naar een andere URL of aanpassen van headers.

2.8. Traffic Manager

Deze load balancer biedt de mogelijkheid om inkomend verkeer intelligent te verdelen over meerdere endpoints, verspreid over verschillende regio's. In tegenstelling tot een traditionele layer 4 of 7 load balancer werkt Traffic Manager op het **DNS-niveau**: het stuurt gebruikersverkeer naar het beste endpoint nog **vóór** de verbinding met de webserver tot stand komt.

Waarom Traffic Manager?

Ik heb Traffic Manager onderzocht als aanvulling op de andere load balancing-oplossingen (zoals App Gateway en Azure Load Balancer), specifiek voor scenario's waarbij:

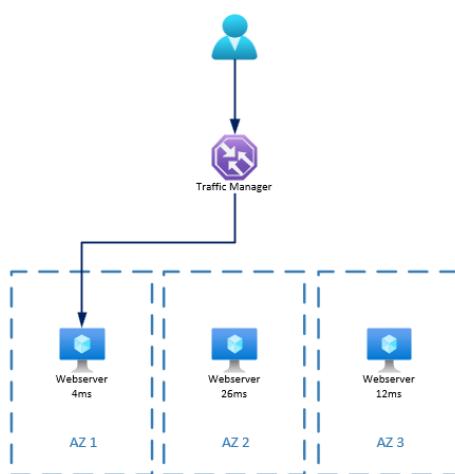
- Verkeer uit verschillende geografische regio's efficiënt en slim verdeeld moet worden.
- Applicaties die moeten reageren op wereldwijde gebruikers met zo min mogelijk latency.
- High availability of failover DNS-niveau gewenst is.

Onderzoek

Tijdens mijn onderzoek testte ik verschillende routing policies uit om hun toepassing te vergelijken:

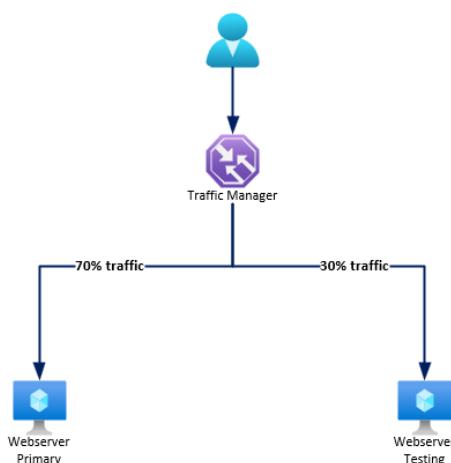
1. Performance

Verkeer wordt naar het endpoint met de laagste netwerklatency geleid. Ideaal voor low latency toepassingen met gebruikers over de hele wereld.



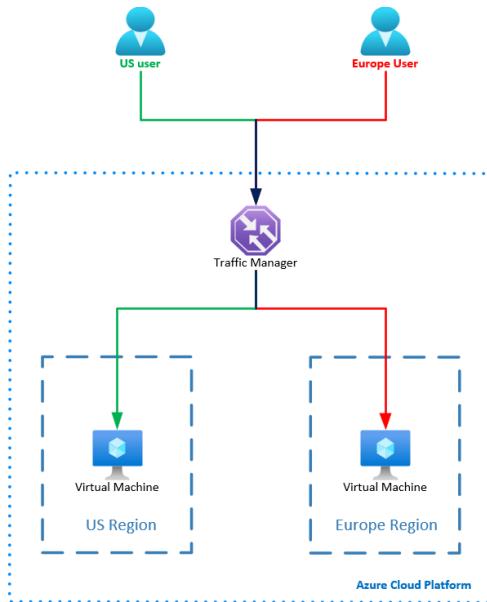
2. Weighted

Biedt de mogelijkheid om verkeer percentagegegewijs te verdelen over meerdere endpoints. Dit is nuttig bij canary deployments voor nieuwe versies van een applicatie.



3. Geographic

Routiert verkeer op basis van de fysieke locatie van de gebruiker. Dit is essentieel voor organisaties die databasescherming moeten garanderen, bijvoorbeeld in het kader van de GDPR of lokale wetgeving.



4. Priority

Zorgt voor een eenvoudige failover waarbij al het verkeer naar het primaire endpoint gaat en pas naar een secundair endpoint schakelt als de eerste niet beschikbaar is.

5. Multivalue

Reageert met meerdere IP-adressen voor redundancy op DNS-niveau, zodat de client zelf load balancing kan toepassen.

6. Subnet

Laat toe om verkeer te sturen op basis van IP-range van de client, nuttig bij interne segmentatie.

Besluit

Traffic Manager biedt vooral waarde in situaties waarbij **geografische load balancing** en **failover** belangrijk zijn. In tegenstelling tot App Gateway of Azure Load Balancer (die werken op transport- of applicatieniveau), is Traffic Manager bijzonder schaalbaar en geconfigureerd op DNS-niveau met het bijkomende voordeel dat het **platformonafhankelijk** is: het kan zowel Azure, on-premises als externe clouds aansturen.

2.9. Load Balancing in de Praktijk (eindanalyse)

In de laatste fase van mijn stage ben ik gestart met het maken van een overkoepelend document dat ik specifiek voor Cegeka heb opgesteld. Dit document vormt een **diepgaande analyse van alle onderzochte load balancers** binnen Azure en F5 BIG-IP, inclusief hun werking, toepassingen en inzetbaarheid binnen de context van Cegeka.

Inhoud van de analyse

Tijdens de analyse maakte ik gebruik van mijn eerder opgemaakte documenten over de verschillende load balancers:

- Azure Load Balancer (L4 – TCP/UDP)
- Application Gateway (L7 – HTTP/HTTPS)
- Traffic Manager (DNS-based)
- F5 BIG-IP

Voor elk van deze load balancers zijn de volgende zaken onderzocht:

- De belangrijkste kenmerken (key features)
- Typische use cases (telkens twee per load balancer)
- Diepgaande vergelijking tussen AGW & F5, alsook F5 in datacenter & Azure

WRM-analyse & use cases

Bij de vergelijking hield ik rekening met vijf belangrijke criteria:

- **Simplicity:** Hoe makkelijk is de load balancer te configureren en beheren?
- **Performance:** Hoe snel en betrouwbaar verwerkt de load balancer het netwerkverkeer?
- **Costs:** Hoe duur is de oplossing?
- **Security:** Welke beveiliging is standaard voorzien of hoe makkelijk is dit toe te voegen?
- **High Availability:** Hoe goed ondersteunt de load balancer een failover bij storingen?

Azure Load Balancer

Een bedrijf host een aantal databaseservers in Azure. Om downtime door hardware- of netwerkstoringen te voorkomen, willen ze het verkeer automatisch overzetten naar een secundaire database.

Criteria	Weight	ALB	ALB Weighted	App Gateway	App Gateway Weighted	Traffic Manager	Traffic Manager Weighted
Simplicity	15%	80	12	60	9	50	7,5
Performance	25%	90	22,5	60	15	50	12,5
Costs	15%	90	13,5	60	9	85	12,75
Security	15%	50	7,5	75	11,25	30	4,5
High Availability	30%	90	27	70	21	50	15
Total	100%		82,5		65,25		52,25

ALB scoort het **hoogst** in deze use case omdat het **goedkoop**, **snel** en **eenvoudig** te configureren is. Het biedt **zeer goede prestaties** en beschikbaarheid, wat ideaal is voor het automatisch doorschakelen tussen databases. Het enige minpunt is de beperkte ingebouwde security, maar dat is in deze use case minder belangrijk.

Application Gateway

Een bedrijf heeft een webshop met duizenden dagelijkse bezoekers. Ze willen SSL-offloading, path-based routing en makkelijke bescherming tegen web aanvallen. Ze vinden het ook belangrijk om een snelle, schaalbare oplossing te hebben.

Criteria	Weight	ALB	ALB Weighted	App Gateway	App Gateway Weighted	Traffic Manager	Traffic Manager Weighted
Simplicity	10%	90	9	70	7	75	7,5
Performance	30%	80	24	85	25,5	60	18
Costs	10%	90	9	60	6	85	8,5
Security	30%	30	9	90	27	30	9
High Availability	20%	80	16	85	17	60	12
Total	100%		67		82,5		55

De **Application Gateway** scoort hier het beste omdat het **sterke securityfuncties** biedt en **zeer goede prestaties** levert. Het is iets **duurder** en **moeilijker** om op te zetten dan de ALB, maar vanwege de focus op beveiliging en beschikbaarheid is het de beste keuze in deze situatie.

Traffic Manager

Een bedrijf beheert meerdere webservers verspreid over Europa, Amerika en Azië. Ze willen dat bezoekers via DNS automatisch worden verbonden met de geografisch dichtstbijzijnde server.

Criteria	Weight	ALB	ALB Weighted	App Gateway	App Gateway Weighted	Traffic Manager	Traffic Manager Weighted
Simplicity	20%	65	13	55	11	90	18
Performance	25%	70	17,5	65	16,25	90	22,5
Costs	15%	90	13,5	60	9	75	11,25
Security	15%	40	6	85	12,75	40	6
High Availability	25%	80	20	75	18,75	75	18,75
Total	100%		70		67,75		76,5

Traffic Manager is hier de logische keuze omdat het speciaal ontworpen is voor **wereldwijde verdeling** van verkeer via **DNS**. Hoewel het op vlak van performance en security minder scoort dan de andere opties, is het eenvoudig in gebruik, goedkoop en perfect geschikt om gebruikers automatisch te verbinden met de dichtstbijzijnde locatie. Dat maakt het ideaal voor deze use case.

Analyseproces

Om tot een overkoepelende vergelijkende analyse te komen, ben ik opnieuw doorheen mijn fasen gegaan (1–4) waarin ik telkens een load balancer in detail onderzocht en praktisch testte. In fase 5 werden alle resultaten gebundeld tot een overkoepelend vergelijgingsdocument.

Daarnaast heb ik de **Weighted Ranking Method (WRM)-analyse** gedaan, om op een objectieve manier de beste oplossing te selecteren in functie van specifieke scenario's. Elke load balancer werd gescoord op basis van criteria zoals performantie, flexibiliteit, kosten en security.

Vergelijking AGW vs F5 BIG-IP

Aangezien deze twee oplossingen het dichtst bij elkaar aanleunen qua functionaliteit, heb ik een aparte vergelijking gemaakt. Hierbij lag de focus op:

- Securitymogelijkheden (WAF, SSL inspection, DDoS protection, etc.)
- Programmability (iRules vs ingebouwde features)
- Protocolondersteuning
- IAM-integratie

Vergelijking F5 Cegeka vs F5 Azure

Tot slot heb ik het belangrijkste onderzoek gedaan: **is het voordelig als Cegeka hun F5-diensten uitbreiden richting Azure?**

Dit heb ik onderzocht aan de belangrijkste factoren zoals:

- **Opstarttijd**, hoe snel is een volledige HA-omgeving gebruiksklaar?
- **Schaalbaarheid**, is er nood aan schaalbare oplossingen?
- **Kostprijs**, wat zal de klant moeten betalen voor deze nieuwe F5-dienst?
- **Netwerksnelheid**, wat zijn de maximum snelheden die Cegeka nu gaat kunnen aanbieden?
- **Security**, hoe veilig is Azure en tot welk niveau heeft Cegeka controle over de data?

3. Realisatie

In dit hoofdstuk wordt het **eindresultaat** van de opdrachten tijdens mijn stage besproken. Dit omvat de **concrete uitwerking** van de onderzochte load balancing-oplossingen, **de technische opzet**, en **de toegepaste tools**. Er wordt stilgestaan bij de werking van de oplossingen, hoe ze geconfigureerd werden, welke technologieën zijn ingezet (zoals Terraform en Ansible), en hoe dit samenkomt in een vergelijkende analyse.

3.1. Introductie cloud networking & load balancing

Voor ik aan de technische uitwerking van mijn stage begon, heb ik twee voorbereidende onderzoeken gedaan die essentieel waren voor een goede uitvoering van de rest van mijn stage. Deze onderzoeken resulteerden in **concrete documentatie** die ik tijdens het verdere verloop van de stage kon raadplegen.

Azure Networking Fundamentals

Om correct met Azure-load balancers te kunnen werken, was een duidelijke kennis van het Azure netwerkmodel noodzakelijk. Daarom maakte ik een overzichtelijke documentatie waarin ik de belangrijkste componenten binnen Azure Networking toelicht:

- Virtual Networks (VNets)
- Subnets
- Network Security Groups (NSGs)

Load Balancing Concepts

Daarnaast maakte ik ook documentatie over de **basisprincipes van load balancing**. Dit document hielp me om inzicht te krijgen in de werking van load balancers, het verschil tussen layer 4 en layer 7 load balancing, en de verschillende mogelijkheden die Azure op dat vlak biedt.

Oplevering

Beide documenten vormen de basis waarop ik mijn stage gebouwd heb. Hoewel ze geen directe productiewaarde hebben voor Cegeka, dragen ze bij aan een **gesubstraatde kennisopbouw** en kunnen ze in de toekomst gebruikt worden als startpunt voor interne opleidingen, kennisdeling of projectvoorbereiding.

Zie bijlage 1: “**Task 1 - Azure Networking Study & Presentation**”

3.2. Cross-Region Load Balancer

Als eerste praktische opdracht van mijn stage heb ik een **cross-region load balancing omgeving opgezet met Azure Load Balancer**. Deze oefening diende om de eerder opgedane theoretische kennis om te zetten naar een realistisch scenario binnen Azure, en vormde een opstap naar complexere configuraties.

Architectuur

De infrastructuur bestaat uit twee regio's: **West-Europa** en **Noord-Europa**. In beide regio's heb ik identieke componenten opgezet, waaronder:

- Twee **Virtuele Machines** per regio, voorzien van een Apache-webserver.
- Één Virtual Network + subnet per regio.
- Één **Public Regional Load Balancer** per regio die het verkeer lokaal verdeelt.
- Één **Cross-Region Load Balancer** die het globale verkeer tussen regio's balanceert.

De code hieronder werd gebruikt om op elke VM een Apache-webserver te installeren:

```
1. #cloud-config
2. package_update: true
3. packages:
4.   - apache2
5. runcmd:
6.   - systemctl enable apache2
7.   - systemctl start apache2
```

Oplevering

Een configuratie document waar ik stap voor stap uitleg hoe deze opstelling werkt en hoe alles geconfigureerd moet worden, alsook een netwerkschema om het visueel makkelijk over te brengen.

Hoewel dit een leeroefening voor mezelf was, biedt het resultaat een eenvoudig startpunt voor het opzetten van een **cross-region failover**. Dit kan in toekomstige klantprojecten worden ingezet als basisconfiguratie of voor testomgevingen.

Zie bijlage 2: “*Task 2 - Set Up a Basic Load Balancer in Azure*”

3.3. F5 HA-cluster in Azure

Als tweede praktische opdracht ben ik aan de slag gegaan met het opzetten van een **F5 BIG-IP Active/Standby High Availability (HA) cluster in Azure**. Dit was eigenlijk het **hoofddoel** van mijn stage en had als doel **na te gaan** of F5 binnen Azure **een haalbare en schaalbare load balancing-oplossing** biedt voor bedrijfskritische toepassingen.

Enkele F5-opstelling

Als eerste stap heb ik een **enkele F5 BIG-IP** opgezet om de **basisfunctionaliteit te testen** zoals het opzetten van de verschillende interfaces, VLAN, pools en virtual servers.

In Azure heb ik **drie nieuwe subnets en NIC's aangemaakt**, allemaal gekoppeld aan de juiste NSG en F5-VM. Om alles veilig te kunnen instellen, heb ik ook een **jumphost opgezet** die alleen gebruikt werd om verbinding te maken met de F5 en deze te configureren.

Dubbele HA F5-opstelling

Daarna werd de configuratie uitgebred naar een **Active/Standby HA-cluster** met twee F5 BIG-IP instances. Hiervoor werden de volgende componenten toegevoegd:

- Tweede F5-VM + extra HA NIC's, subnets en NSG's
- **ConfigSync, Trust en Device Group** ingesteld tussen de twee instances.
- **Traffic Group** aangemaakt voor automatische failover.

Om ervoor te zorgen dat de failover ook echt werkt, heb ik in Azure **twee mogelijke opties uitgeprobeerd**: eentje via een **Azure Load Balancer** en eentje via de **Cloud Failover Extension** van F5. Meer uitleg hierover vind je terug bij de analyse.

Oplevering

Als **proof of concept** heb ik een **volledige F5 HA-cluster opgezet** in Azure, met twee F5-instances in een **Active/Standby-configuratie**. Ik werkte zowel de setup uit voor failover via een Azure Load Balancer als via de Cloud Failover Extension. Alles werd gebundeld in een duidelijke **stap-voor-stap handleiding**, aangevuld met **technische documentatie** die het hele proces ondersteunt.

Zie bijlage 3: "Task 3 - Set Up and Configure F5 Load Balancer"

Waarde voor Cegeka

Deze proof of concept toont aan dat F5 BIG-IP met HA wél degelijk inzetbaar is binnen Azure, mits de juiste configuratie. Dit opent nieuwe mogelijkheden voor Cegeka om **enterprise-grade load balancing aan te bieden in Azure** voor klanten die meer nodig hebben dan de native oplossingen.

3.4. Vergelijkende Analyse ALB en F5

In dit deel van mijn stage heb ik een vergelijkende analyse gemaakt tussen Azure Load Balancer en F5, met de focus op performance, functionaliteiten en configuratie. Aangezien deze onderwerpen eerder al uitgebreid zijn besproken, herhaal ik de technische details hier niet. Voor meer informatie kan je altijd het document via de bijlage openen.

Zie bijlage 4: “Task 4 - Comparative Analysis”

3.5. Automatisatie met Terraform

Na het handmatig opzetten van een F5 HA-omgeving, heb ik in deze opdracht dezelfde infrastructuur geautomatiseerd met **Terraform**. Het doel was om op een herbruikbare en efficiënte manier een volledige **high availability setup van F5 BIG-IP** in Azure te kunnen deployen, inclusief ondersteuning voor de **Cloud Failover Extension**.

Nadat ik mijn analyse had afgerond en alle nodige informatie verzameld had, kon ik effectief aan de slag met Terraform.

Eigen Terraform module

De door mij gemaakte module bevat ongeveer **90%** van de nodige code om een F5 VM te maken. Parameters zoals VM-naam, locatie, netwerkinterfaces, diskconfiguratie en image-instellingen kunnen via variabelen worden aangepast. Hierdoor wordt het mogelijk om met slechts een tiental lijnen code een nieuwe F5-instance te deployen.

```
resource "azurerm_linux_virtual_machine" "f5_vm" {
  name          = var.vm_name
  location      = var.location
  resource_group_name = var.resource_group_name
  size          = var.size
  computer_name = var.computer_name
  admin_username = var.admin_username
  admin_password = var.admin_password
  disable_password_authentication = var.disable_password_authentication
  network_interface_ids = var.network_interface_ids
  tags          = var.vm_tags

  os_disk {
    name          = var.os_disk_name
    storage_account_type = var.os_disk_storage_account_type
    caching        = var.os_disk_caching
    disk_size_gb   = var.os_disk_size_gb
  }

  source_image_reference {
    offer      = var.image_offer
    publisher = var.image_publisher
    sku       = var.image_sku
    version   = var.image_version
  }

  plan {
    name      = var.plan_name
    product   = var.plan_product
    publisher = var.plan_publisher
  }

  identity {
    type      = "UserAssigned"
    identity_ids = var.identity_ids
  }

  custom_data = base64encode(templatefile("${path.module}/cloud-init.yml", {
    ssh_public_key = var.ansible_machine_ssh_key
  }))
}
```

De code hieronder wordt **nu** gebruikt voor het maken van een F5 VM die de onderliggende module gebruikt:

```
module "f5_vm_1" {
  source = "./modules/f5_bigip"
  vm_name = var.f5_1_hostname # The name of the VM in Azure
  computer_name = var.f5_1_computername # The internal computer name of the Linux environment
  admin_username = var.admin_username # The username to login in F5
  os_disk_name = var.f5_1_os_disk_name
  identity_ids = [azurerm_user_assigned_identity.terraform_identity.id]

  network_interface_ids = [
    azurerm_network_interface.management_nic.id,
    azurerm_network_interface.external_nic.id,
    azurerm_network_interface.internal_nic.id,
    azurerm_network_interface.ha_nic.id
  ]
}
```

Toegevoegde functies

De Terraform code is verder uitgebreid met enkele extra's om het geheel meer production-ready te maken:

- Cloud-Init code voor het automatisch configureren van SSH-instellingen.
- User Identity met nodige rollen voor correcte API-toegang door CFE.
- Storage Account voor het opslaan van de CFE state-bestanden.

Oplevering

Het eindresultaat van deze opdracht is een **volledig gedocumenteerde Terraform-oplossing** voor het opzetten van een **F5 HA-cluster met CFE** in Azure. Daarnaast heb ik een **technische guide opgesteld** waarin stapsgewijs alles wordt uitgelegd.

Door de automatisatie via Terraform is het mogelijk geworden om **binnen enkele minuten** een volledig werkende HA-omgeving op te zetten. Dit levert niet alleen **tijdswinst** op, maar vermindert ook de kans op menselijke fouten bij de configuratie.

Zie bijlage 5: “Task 5 - Automation with Terraform”

Waarde voor Cegeka

Het document dat ik heb opgesteld is bedoeld voor intern gebruik bij **Cegeka** en kan dienen als **referentie** of **basis** voor toekomstige projecten waarbij F5 BIG-IP via Terraform geautomatiseerd uitgerold moet worden in de Azure Cloud.

3.6. Configuratie via Ansible

Na het succesvol opzetten van een F5 HA-cluster met behulp van Terraform, heb ik deze infrastructuur verder geautomatiseerd met Ansible. Het doel was om de configuratie van de F5 BIG-IP-instances volledig automatisch te laten verlopen na de provisioning, inclusief netwerkconfiguratie, failover-instellingen en installatie van de Cloud Failover Extension.

Playbooks

Ik maakte een reeks **playbooks** die elk een onderdeel van de **F5-configuratie uitvoeren**. Deze playbooks zijn modulair opgebouwd en kunnen **individueel** worden uitgevoerd, **of in één keer** via een “master” playbook. Dit zorgt ervoor dat de volledige F5-omgeving op minder dan **vier minuten** automatisch geconfigureerd is.

Beveiliging

Om gevoelige data, zoals admin-wachtwoorden, veilig te beheren, heb ik gebruik gemaakt van **Ansible Vault**. Hiermee kunnen gevoelige variabelen versleuteld worden opgeslagen in een apart bestand:

`ansible-vault create vault.yml`

Tijdens het uitvoeren van het playbook wordt de gebruiker gevraagd de vault te openen met:

`ansible-playbook -i inventory.ini --ask-vault-pass master-playbook.yml`

Oplevering

Ik heb een set van **zeven playbooks** gemaakt voor de **volledige HA-configuratie** van F5, aangestuurd via een **master playbook**. De installatie en configuratie van CFE is volledig geautomatiseerd, inclusief het downloaden, installeren en configureren via de REST API zonder handmatige tussenkomst.

Daarnaast heb ik een technisch document opgesteld met uitleg over de playbooks, Ansible-configuratie en uitleg over stukken code voor een vlotte automatisatie.

Zie bijlage 6: “Task 6 - Configuring F5 with Ansible”

Waarde voor Cegeka

Cegeka beschikt nu over een **volledig geautomatiseerde oplossing** voor de deployment en configuratie van een **F5 HA-cluster** in Azure. Bovendien blijft de automatisering toepasbaar op een algemene HA-cluster, zelfs zonder CFE.

De combinatie van Terraform voor provisioning en Ansible voor configuratiebeheer maakt het mogelijk om in minder dan 20 minuten een productieklare F5 HA-cluster inclusief CFE op te zetten. Met deze oplossing heeft Cegeka een basis om F5-automatisatie op grotere schaal toe te passen.

3.7. Application Gateway

Voor de vijfde praktische opdracht ben ik aan de slag gegaan met het opzetten, configureren en testen van een **Application Gateway**, een Layer 7 load balancer die geavanceerde routeringsmogelijkheden ondersteunt.

Onderstaande functies heb ik bekeken en zelf opgezet om zo meer inzicht te krijgen in de mogelijkheden die deze load balancer te bieden had:

- Listener en rules.
- Path-Based routing en verkeer gestuurd naar de juiste backend server.
- Nieuw SSL certificaat laten genereren en omgezet naar een .pfx bestand.
- Rewrite rules om info uit de headers te verwijderen om fingerprinting te vermijden.
- Extra security headers laten toevoegen zoals HSTS, X-Frame-Options, etc.
- ...

Om mijn **SSL-certificaat** om te zetten naar een **.pfx-bestand** voor de upload naar Azure, heb ik het volgende commando gebruikt:

```
openssl pkcs12 -export -out agw-nas25.pfx -inkey /etc/letsencrypt/live/agw.nas25.be/privkey.pem -in  
/etc/letsencrypt/live/agw.nas25.be/fullchain.pem -certfile  
/etc/letsencrypt/live/agw.nas25.be/chain.pem
```

Oplevering

Alle **uitgevoerde stappen en testresultaten** zijn gedocumenteerd in een **technisch document**, waar per feature stap voor stap wordt uitgelegd hoe de configuratie werkt. Naast tekst bevat dit document een **netwerkschema** om het makkelijk visueel over te brengen.

Dit document draagt mee aan de vergelijkende analyse die Cegeka later kan gaan gebruiken bij het vergelijken van de verschillende load balancers in Azure.

Zie bijlage 7: “Task 7 - Azure Application Gateway”

Waarde voor Cegeka

Met deze configuratie en het bijbehorend **technisch document** beschikt **Cegeka** over een **praktische referentie en proof of concept** voor het inzetten van Application Gateway. De verschillende routingopties tonen hoe simpel en krachtig de Application Gateway kan zijn in scenario's waar webverkeerbeheer, performantie en security centraal staan.

De documentatie kan dienen als **interne handleiding of trainingsmateriaal** voor toekomstige medewerkers.

3.8. Traffic Manager

Als zesde praktische opdracht heb ik Azure Traffic Manager geconfigureerd en getest. Deze DNS-based load balancer laat toe om verkeer wereldwijd te verdelen op basis van verschillende routing policies. Net als bij de vorige opdracht lag de focus op **praktisch onderzoek**, het testen van **routingmethodes** en het **documenteren** van de resultaten.

Opstelling

Om de werking van Traffic Manager correct te testen, heb ik twee webservers opgezet:

- Webserver 1: Gehost in Europa
- Webserver 2: Gehost in de Verenigde Staten

Beide VM's heb ik uitgerust met Apache en een eenvoudige HTML-pagina. Via een public IP en gekoppeld DNS-label kregen beide servers een publiek toegankelijke FQDN. Vervolgens zijn deze als endpoints toegevoegd aan Traffic Manager via de configuratieoptie "Public IP address".

Vervolgens heb ik onderstaande routingmethodes getest en de configuratie gedocumenteerd:

- Performance
- Weighted
- Geographic

Oplevering

Alle **uitgevoerde stappen en testresultaten** zijn gedocumenteerd in een **technisch document**, waar per feature stap voor stap wordt uitgelegd hoe de configuratie is opgezet. Naast tekst bevat dit document ook **netwerkschema's** om het concept visueel duidelijk over te brengen.

Dit document draagt mee aan de vergelijkende analyse die Cegeka later kan gaan gebruiken bij het vergelijken van de verschillende load balancers in Azure.

Zie bijlage 8: "*Task 8 - Azure Traffic Manager*"

Waarde voor Cegeka

Met deze configuratie en het bijbehorende technisch document beschikt Cegeka over een **praktische referentie en proof of concept** voor het inzetten van **Traffic Manager**. De verschillende routingmethodes tonen hoe flexibel en krachtig Traffic Manager is in scenario's waarbij **wereldwijde beschikbaarheid, performance en failover** cruciaal zijn.

De documentatie is geschikt om in te zetten als **interne handleiding** voor toekomstige medewerkers die met Azure Load Balancing-oplossingen aan de slag gaan.

3.9. Load Balancing in de Praktijk (einddocument)

Het resultaat van dit laatste deel is een **volledig, gestructureerd technisch document** dat Cegeka concreet helpt bij het **bepalen van de juiste load balancing strategie** binnen hun cloudomgeving.

Oplevering

- Een **gestructureerd en overzichtelijk document** waarin alle onderzochte load balancers (Azure Load Balancer, Application Gateway, Traffic Manager en F5 BIG-IP) worden vergeleken op vlak van werking, functies, typische use cases en voor- en nadelen.
- **Visuele tabellen** die snel inzicht geven in de sterktes en zwaktes van elke oplossing.
- **WRM-analyses** die op basis van relevante criteria tot objectieve aanbevelingen komen.
- Een **diepgaande vergelijking** tussen het gebruik van F5 BIG-IP in het Cegeka-datacenter en in Azure, met aandacht voor **kosten, deploymenttijden, schaalbaarheid** en automatisatie.

Zie bijlage 9: “*Task 9 & 10 - Load Balancing in Practice*”

Waarde voor Cegeka

Met dit einddocument bundel ik alle opgedane kennis uit mijn volledige stageperiode in één overzichtelijk en praktisch bruikbaar rapport. Hiermee laat ik bij Cegeka waardevolle documentatie achter die toelaat om:

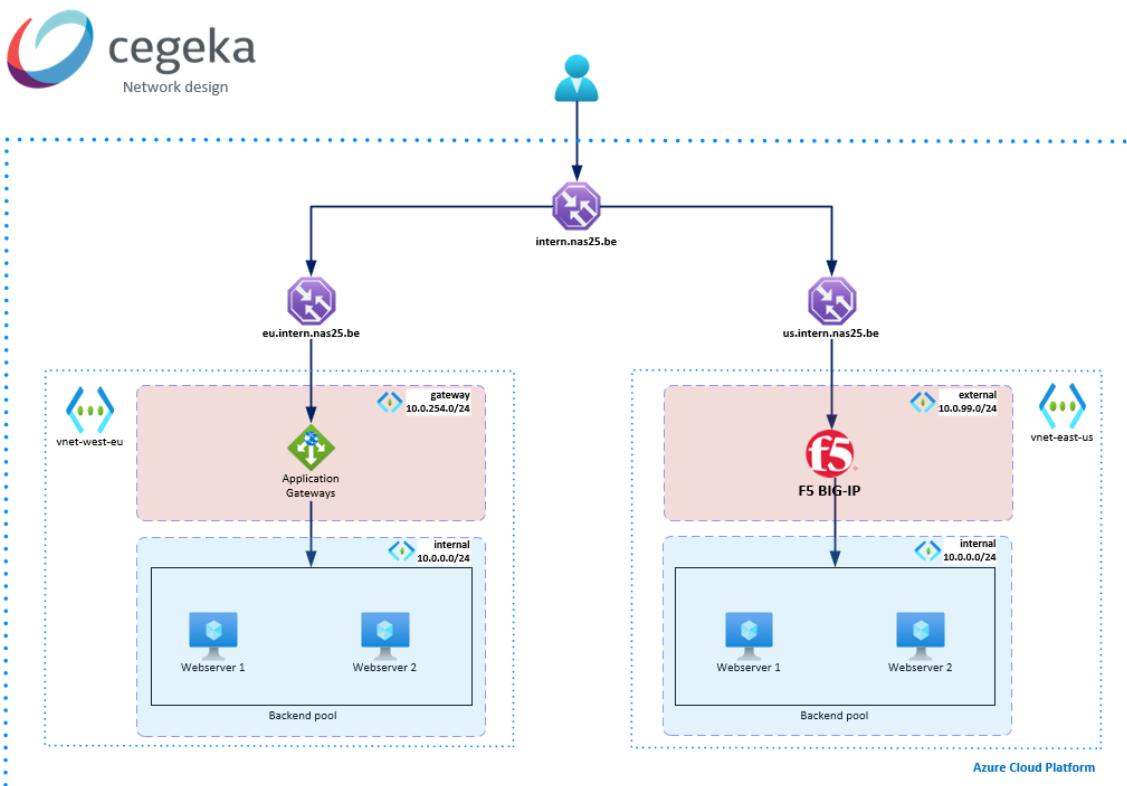
- **Volledig inzicht te krijgen** in zowel Azure-native als externe load balancing oplossingen.
- **Onderbouwde keuzes maken** voor toekomstige implementaties.
- **Een duidelijke basis te leggen voor het overwegen van een nieuwe F5-dienst in Azure**, inclusief de voordelen, haalbaarheid en impact op bestaande architecturen.

4. Extra oefening

Als extra uitdaging heb ik een multi-load balancing omgeving opgezet in Azure. Het doel was om de verschillende load balancers waar ik de afgelopen drie maanden mee geëxperimenteerd heb, zoals Traffic Manager, Application Gateway en F5 BIG-IP, samen te brengen in één grootschalige en realistische opstelling.

Door dit te doen, wilde ik mij meer verdiepen in geografische regio's, verschillende load balancers en globale schaalbaarheid. Tegelijk bood deze oefening de kans om concrete scenario's uit de praktijk na te bootsen, zoals een wereldwijd beschikbare applicatie die automatisch verkeer stuurt naar de dichtstbijzijnde regio op basis van de locatie van de gebruiker.

Bovendien geeft deze opstelling een mooi totaalbeeld van hoe Azure-native oplossingen kunnen samenwerken met third-party load balancers zoals F5, een combinatie die je in veel enterprise-omgevingen ziet terugkomen.



De gebruiker maakt verbinding met de algemene domeinnaam van de applicatie, die beheerd wordt door Azure Traffic Manager. Traffic Manager is zo geconfigureerd dat het gebruikmaakt van geographic routing en heeft als endpoints de publieke IP-adressen van zowel de Application Gateway (voor Europa) als de F5 BIG-IP (voor de VS), allebei gekoppeld aan hun regionale domeinnamen.

Op basis van de geografische locatie van de gebruiker stuurt Traffic Manager het verkeer automatisch door naar de juiste regio, zodat elke gebruiker altijd verbonden wordt met de dichtstbijzijnde en meest geschikte omgeving.

5. Besluit

Tijdens mijn stage bij **Cegeka** kreeg ik de kans om te werken rond het opzetten, vergelijken en automatiseren van verschillende load balancing-oplossingen binnen **Azure**. De opdrachten waren uiteenlopend, zowel qua technologie als complexiteit, maar hadden steeds als doel om een schaalbare omgeving op te zetten.

In de opdracht rond **F5 load balancers** kreeg ik te maken met de praktische moeilijkheden van de **Cloud Failover Extension (CFE)** en de beperkingen van de Azure-omgeving. Toch slaagde ik erin om zowel de CFE- als de Azure Load Balancer-opstelling succesvol uit te werken. Deze hands-on ervaring leerde mij hoe belangrijk het is om om te gaan met technische uitdagingen en alternatieve oplossingen te onderzoeken.

Doorheen de andere opdrachten, zoals het opzetten van **Application Gateway**, **Traffic Manager** en het gebruik van tools zoals **Terraform** en **Ansible**, heb ik mijn technische kennis verbreed. Elke load balancer had zijn eigen sterkes, beperkingen en toepassingen. Deze werd duidelijk na het uitvoeren van **vergelijkende analyses**, waarbij onder andere de WRM-methode hielp om onderbouwde keuzes te maken.

Als ik terugkijk op de doelstellingen die in het projectplan werden voorgesteld, namelijk het **onderzoeken, testen en evalueren van verschillende load balancing-oplossingen** met als focus een **F5-omgeving in Azure**, kan ik met trots zeggen dat deze doelstellingen zijn behaald.

Voor de toekomst zou ik aanraden naar Cegeka toe om deze F5-omgeving in Azure verder uit te werken en te testen met hybride opstellingen waarbij on-premises en Azure-resources samenwerken via F5. Verder onderzoek van deze oplossingen kan Cegeka helpen om haar diensten nog beter te maken.

Deze stage bood mij niet alleen een **technische uitdaging**, maar was ook een waardevolle kans om mijn kennis en ervaring in cloudtechnologieën, en in het bijzonder **Azure**, verder uit te breiden.

6. Bijlage

Bijlage 1: “Task 1 – Study and Presentation”

Bevat een beknopte analyse over de Azure Network Fundamentals, alsook de verschillende load balancers binnen Azure en F5 BIG-IP.

Bijlage 2: “Task 2 – Set Up a Basic Load Balancer in Azure”

Technische stappen en testresultaten bij het opzetten van een eenvoudige Azure Load Balancer (Layer 4) met back-end pools en health probes over twee regio's.

Bijlage 3: “Task 3 – Set Up and Configure F5 Load Balancer”

Praktische handleiding voor het implementeren van een F5 BIG-IP in Azure, alsook het configureren van de verschillende HA-opties binnen Azure.

Bijlage 4: “Task 4 – Comparative Analysis”

Vergelijkende analyse tussen Azure Load Balancer en F5 BIG-IP inclusief functionaliteit, beheer, use cases en kostenanalyse.

Bijlage 5: “Task 5 – Automation with Terraform”

Documentatie over de geautomatiseerde uitrol van het cloud-infrastructuur zoals networking, VM's, security groups, etc.

Bijlage 6: “Task 6 – Configure F5 with Ansible”

Documentatie van hoe F5 BIG-IP kan worden geconfigureerd via Ansible zodat automatisch een HA-cluster wordt opgezet met de Cloud Failover Extension, inclusief gebruikte playbooks.

Bijlage 7: “Task 7 – Azure Application Gateway”

Analyse van Application Gateway over de configuratie, beveiligingsopties, routing policies en netwerkdiagram.

Bijlage 8: “Task 8 – Azure Traffic Manager”

Analyse van Traffic Manager over de configuratie, routing policies en netwerkdiagrammen.

Bijlage 9: “Task 9 & 10 – Load Balancing in Practice”

Overkoepelend en samenvattend document waar alle informatie wordt samengebracht met de belangrijkste features en scenario's waarin de verschillende load balancers worden toegepast in realistische situaties, inclusief de WRM-analyse.

Ook wordt hier de belangrijkste vergelijking gemaakt, die tussen F5 in het Cegeka-datacenter en F5 in Azure.

7. Literatuurlijst

MS Learn – What is Azure Load Balancer?

<https://learn.microsoft.com/en-us/azure/load-balancer/load-balancer-overview>

MS Learn – Load-balancing options

<https://learn.microsoft.com/en-us/azure/architecture/guide/technology-choices/load-balancing-overview>

MS Learn – What is Azure Application Gateway?

<https://learn.microsoft.com/en-us/azure/application-gateway/overview>

MS Learn – Azure Application Gateway features

<https://learn.microsoft.com/en-us/azure/application-gateway/features>

MS Learn – What is Traffic Manager?

<https://learn.microsoft.com/en-us/azure/traffic-manager/traffic-manager-overview>

MS Learn – How Traffic Manager Works

<https://learn.microsoft.com/en-us/azure/traffic-manager/traffic-manager-how-it-works>

MS Learn – Traffic Manager routing methods

<https://learn.microsoft.com/en-us/azure/traffic-manager/traffic-manager-routing-methods>

YouTube – Cloud Patashala – How to deploy F5 on Azure cloud

https://www.youtube.com/watch?v=7re37t3PA_4&t=1110s

YouTube – F5 DevCentral – BIG-IP VE in Microsoft Azure

<https://www.youtube.com/watch?v=wNSSFSENbOg&t=521s>

YouTube – Noor Networks – 6. License Activation and Initial Setup Wizard || F5 Big-IP LTM

<https://www.youtube.com/watch?v=-MXmJe4rrkw&t=1031s>

F5 Clouddocs – Deploy F5 BIG-IP Virtual Edition in Azure Classic

https://clouddocs.f5.com/cloud/public/v1/azure/Azure_single_classic.html

F5 Clouddocs – Deploy F5 BIG-IP Virtual Edition with Azure Gateway Load Balancer

https://clouddocs.f5.com/cloud/public/v1/azure/Azure_deploy_gwlb.html

YouTube – RAYKA – 17. F5 BIG-IP HA Active Standby Configuration

<https://www.youtube.com/watch?v=3IGqk2INvbQ>

F5 Clouddocs – F5 BIG-IP Cloud Failover

<https://clouddocs.f5.com/products/extensions/f5-cloud-failover/latest/>

<https://clouddocs.f5.com/products/extensions/f5-cloud-failover/latest/userguide/azure.html#define-the-failover-addresses-in-azure>

Stack Overflow - Terraform Azure network security

<https://stackoverflow.com/questions/48549577/terraform-azure-network-security>

GitHub F5Networks – Terraform Azure BIG-IP Module

<https://github.com/F5Networks/terraform-azure-bigip-module/tree/main>

F5 Techdocs – High Availability (HA) Failover

https://techdocs.f5.com/kb/en-us/products/big-ip_ltm/manuals/product/bigip-service-provider-sip-administration-13-1-0/6.html

F5 Clouddocs – Working With iControl LX/iApps LX RPM Packages

https://clouddocs.f5.com/products/iapp/iapp-lx/tmos-14_0/icontrollx_packages/working_with_icontrollx_packages.html

YouTube – John Savill's Technical Training

<https://www.youtube.com/@NTFAQGuy>



CONTACT

Arne Madalijns
r0937871@student.thomasmore.be
Tel. + 32 489 09 17 72

VOLG ONS

www.thomasmore.be
fb.com/ThomasMoreBE
#WeAreMore

THOMAS
MORE