



Faculteit Bedrijf en Organisatie

Infrastructure as Code op AWS: analyse en vergelijkende studie van de beschikbare services

Antonio Pizarro

Scriptie voorgedragen tot het bekomen van de graad van  
professionele bachelor in de toegepaste informatica

Promotor:  
Gertjan Bosteels  
Co-promotor:  
Sander Adam

Instelling: —

Academiejaar: 2021-2022

Derde examenperiode



Faculteit Bedrijf en Organisatie

Infrastructure as Code op AWS: analyse en vergelijkende studie van de beschikbare services

Antonio Pizarro

Scriptie voorgedragen tot het bekomen van de graad van  
professionele bachelor in de toegepaste informatica

Promotor:  
Gertjan Bosteels  
Co-promotor:  
Sander Adam

Instelling: —

Academiejaar: 2021-2022

Derde examenperiode



# Woord vooraf

TODO voorwoord



## Samenvatting

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus.

Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.



# Inhoudsopgave

<b>1</b>	<b>Inleiding</b>	<b>17</b>
1.1	Probleemstelling	17
1.2	Onderzoeksvraag	18
1.3	Onderzoeksdoelstelling	18
1.4	Opzet van deze bachelorproef	18
<b>2</b>	<b>Stand van zaken</b>	<b>19</b>
2.1	AWS AppRunner	19
2.1.1	Marktevaluatie	20
2.2	AWS Amplify	20
2.2.1	Amplify libraries	20
2.2.2	Amplify Studio	20
2.2.3	Amplify CLI	21

2.2.4	Amplify Hosting .....	21
2.2.5	Marktevaluatie .....	21
<b>2.3</b>	<b>AWS Lightsail</b>	<b>21</b>
2.3.1	Marktevaluatie .....	22
<b>2.4</b>	<b>AWS Copilot</b>	<b>22</b>
2.4.1	Applications .....	22
2.4.2	Environments .....	22
2.4.3	Services .....	22
2.4.4	Jobs .....	23
2.4.5	Pipelines .....	23
<b>2.5</b>	<b>AWS Elastic Beanstalk</b>	<b>23</b>
<b>2.6</b>	<b>AWS Cloudformation</b>	<b>24</b>
2.6.1	Template .....	24
2.6.2	Stack .....	24
2.6.3	Change set .....	25
2.6.4	Delete stack .....	25
<b>2.7</b>	<b>Terraform</b>	<b>25</b>
<b>2.8</b>	<b>Ansible</b>	<b>26</b>
<b>2.9</b>	<b>AWS SAM</b>	<b>26</b>
<b>2.10</b>	<b>AWS CDK</b>	<b>26</b>
<b>2.11</b>	<b>Troposphere</b>	<b>27</b>
<b>2.12</b>	<b>Pulumi</b>	<b>27</b>

<b>3</b>	<b>Methodologie</b>	<b>29</b>
3.1	Criteria evaluatie	30
<b>4</b>	<b>Vergelijking services en tools</b>	<b>33</b>
4.1	Initiële inzet	33
<b>5</b>	<b>Business case</b>	<b>37</b>
5.1	Functionele requirements	37
5.2	AWS Serverless oplossing	37
<b>6</b>	<b>Uitwerking business case</b>	<b>39</b>
6.1	Uitwerking AWS CDK	39
6.2	Uitwerking AWS Cloudformation	39
6.3	Uitwerking Terraform	39
<b>7</b>	<b>Conclusie</b>	<b>41</b>
<b>A</b>	<b>Onderzoeksvoorstel</b>	<b>43</b>
A.1	Introductie	43
A.2	State-of-the-art	43
A.3	Methodologie	44
A.4	Verwachte resultaten	44
A.5	Verwachte conclusies	44
	<b>Acroniemen</b>	<b>45</b>
	<b>Woordenlijst</b>	<b>47</b>

<b>Bibliografie</b> .....	<b>49</b>
---------------------------	-----------

## Lijst van figuren



## Lijst van tabellen

4.1	Initiële inzet test . . . . .	34
4.2	Initiële inzet declaratieve tools . . . . .	35
4.3	Initiële inzet programmatieve tools . . . . .	35





## Listings



# 1. Inleiding

De inleiding moet de lezer net genoeg informatie verschaffen om het onderwerp te begrijpen en in te zien waarom de onderzoeksvraag de moeite waard is om te onderzoeken. In de inleiding ga je literatuurverwijzingen beperken, zodat de tekst vlot leesbaar blijft. Je kan de inleiding verder onderverdelen in secties als dit de tekst verduidelijkt. Zaken die aan bod kunnen komen in de inleiding (Pollefliet, 2011):

- context, achtergrond
- afbakenen van het onderwerp
- verantwoording van het onderwerp, methodologie
- probleemstelling
- onderzoeksdoelstelling
- onderzoeksvraag
- ...

## 1.1 Probleemstelling

Uit je probleemstelling moet duidelijk zijn dat je onderzoek een meerwaarde heeft voor een concrete doelgroep. De doelgroep moet goed gedefinieerd en afgeleid zijn. Doelgroepen als “bedrijven,” “KMO’s,” systeembeheerders, enz. zijn nog te vaag. Als je een lijstje kan maken van de personen/organisaties die een meerwaarde zullen vinden in deze bachelorproef (dit is eigenlijk je steekproefkader), dan is dat een indicatie dat de doelgroep goed gedefinieerd is. Dit kan een enkel bedrijf zijn of zelfs één persoon (je co-promotor/opdrachtgever).

## 1.2 Onderzoeksvraag

Wees zo concreet mogelijk bij het formuleren van je onderzoeksvraag. Een onderzoeksvraag is trouwens iets waar nog niemand op dit moment een antwoord heeft (voor zover je kan nagaan). Het opzoeken van bestaande informatie (bv. “welke tools bestaan er voor deze toepassing?”) is dus geen onderzoeksvraag. Je kan de onderzoeksvraag verder specificeren in deelvragen. Bv. als je onderzoek gaat over performantiemetingen, dan

## 1.3 Onderzoeksdoelstelling

Wat is het beoogde resultaat van je bachelorproef? Wat zijn de criteria voor succes? Beschrijf die zo concreet mogelijk. Gaat het bv. om een proof-of-concept, een prototype, een verslag met aanbevelingen, een vergelijkende studie, enz.

## 1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 7, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

## 2. Stand van zaken

Infrastructure as Code (IaC) is een essentiële bouwsteen binnen de DevOps methodologie. De infrastructuur code wordt behandeld op dezelfde manier als een softwareontwikkelaar zijn code zou behandelen. Met andere woorden, de code wordt op een ordelijke manier binnen een versiebeheer systeem bewaard. Daarnaast moet het ook mogelijk zijn om Continuous Integration (CI) en Continuous Deployment (CD) toe te passen op de infrastructuur code. De code moet een consistent en repliceerbaar resultaat leveren die automatisatie toelaat (Mansoor, 2014).

### 2.1 AWS AppRunner

AppRunner is een AWS-service die de functionaliteit verschaft om broncode of een container image te inzetten op een snelle, simpele en kostefficiënte manier. De toepassing wordt op een schaalbare en veilige AWS-cloud omgeving opgezet. De gebruiker hoeft de configuratie van AWS-resources niet zelf in handen te nemen, alle achterliggende infrastructuur wordt door AppRunner beheerd (Khen, 2022b).

In een artikel van Aussems (2021) beschrijft hij AWS AppRunner als een geautomatiseerde versie van AWS Fargate, de serverless containerdienst van de provider. Daarom is deze service bestemd voor ontwikkelaars die geen tijd willen besteden in de uitrol, configuratie of beheer van hun containertoepassingen. Ze hoeven enkel de broncode af te leveren, in de vorm van een code repository of een container image (Khen, 2022b).

Toepassingen die via AppRunner zijn ingezet, worden op container instanties geplaatst. Deze instanties verbruiken computer- en geheugenbronnen, waarvoor de gebruiker moet

betalen. Het geheugen en vCPU worden bij het opzet van AppRunner bepaald en op basis daarvan wordt de kost per GB bepaald. De kosten omvatten aan de ene kant de systeembronnen van een bevoorraadde container instantie. Deze beperkte systeembronnen zijn nodig om de container instantie op een inactief staat te behouden. Daarnaast moet er ook betaald worden voor de systeembronnen wanneer een container actief wordt gebruikt, bijvoorbeeld tijdens het verwerken van requests. Er wordt enkel betaald als de toepassing draaiende is. Het is heel eenvoudig om een AppRunner toepassing tijdelijk te stoppen en terug te starten. Deze acties kunnen via de console, CLI of API uitgevoerd worden (AWS, g.d.-a).

### 2.1.1 Marktevaluatie

## 2.2 AWS Amplify

AWS Amplify bestaat uit een set van tools gericht voor het ontwikkelen van frontend web en mobiele toepassingen. Met deze tools wordt het opzetten van een full-stack toepassing op AWS-infrastructuur op een intuïtieve manier mogelijk (AWS, 2022c). De gebruiker heeft de flexibiliteit om het grote aanbod aan AWS-componenten te integreren met zijn oplossing. Integratie met backend componenten wordt mogelijk gemaakt via serverless technologieën (Kandaswamy, 2022).

Vervolgens worden de door Amplify aangeboden tools verder uitgelicht.

### 2.2.1 Amplify libraries

De Amplify open-source client libraries voorzien use-case gebaseerde, declaratieve en gemakkelijk te gebruiken interfaces doorheen verschillende categorieën van cloud ondersteunende operaties. Deze stellen mobiele en web ontwikkelaars in staat om integraties met hun backend infrastructuur te realiseren. De libraries worden ondersteund door de AWS-cloud en bieden een inplugbaar model aan, dat ook door andere cloud leveranciers kan worden gebruikt. Verder kunnen de libraries gebruikt worden met beide, nieuwe backend aangemaakt via de Amplify CLI en bestaande backend bronnen (AWS, 2022b).

### 2.2.2 Amplify Studio

AWS Amplify Studio is een visuele ontwikkeling omgeving voor het bouwen van full-stack web en mobiele toepassingen. Studio bouwt verder op bestaande backend-bouw mogelijkheden in AWS Simplify, UI-ontwikkeling wordt aanzienlijk versneld. Met Studio is het mogelijk om een webtoepassing in zijn geheel te bouwen, front-to-back, met minimale codeer werk. En toch behoudt de gebruiker volledige controle over de toepassing ontwerp en gedrag via code (AWS, 2022b).

### 2.2.3 Amplify CLI

De Amplify Command Line Interface (CLI) is een eendrachtig tool die het bouwen, integreren en beheren van AWS-services voor een toepassing faciliteert (AWS, 2022b).

### 2.2.4 Amplify Hosting

Amplify Hosting biedt een git-gebaseerde workflow aan voor het hosten van een full-stack serverless webtoepassing met Continuous Deployment (CD)).

### 2.2.5 Marktevaluatie

In een AWS blog van Spittel (2022) legt hij uit hoe AWS Amplify Studio gemaakt is om het leven van ontwikkelaars gemakkelijker te maken. De ontwerp-ontwikkelaar overdracht gebeurt vlotter d.m.v. Amplify. Bovendien is de code aanpassen of uitbreiden heel eenvoudig met de gegenereerde componenten van Amplify Studio.

Ook Nwamba (2022) vertelt in zijn artikel hoe ontwikkelaars steeds meer low-code tools adopteren. Ontwikkelaars die het ontwikkelwerk voor het bouwen van een toepassing interface willen vereenvoudigen moeten AWS Amplify overwegen.

## 2.3 AWS Lightsail

AWS Lightsail verschaft cloud systeembronnen om een toepassing of webtoepassing moeiteloos te kunnen opzetten. Lightsail biedt vereenvoudigde diensten aan zoals instanties, containers, databanken, opslag, enz. Niet alleen is Lightsail compatibel met voorgeconfigureerde blueprints zoals Wordpress, Prestashop of LAMP. Maar het is ook mogelijk om statische content te hosten met Lightsail en die globaal ter beschikking te stellen (AWS, 2022a).

Verder worden een aantal kenmerken van Lightsail uitgelicht.

**Instances:** Lightsail virtuele servers voor het hosten van toepassingen.

**Containers:** Inzetten van Docker containers via Lightsail.

**Simplified load balancers:** Vereenvoudigde load balancing functionaliteit.

**Managed databases:** Volledig geconfigureerde MySQL of PostgreSQL databanken.

**Block and object storage:** Schaalbaar block en object opslag op de AWS-cloud.

**CDN distributions:** Content delivery network (CDN) voor de globale distributie van toepassingen.

### 2.3.1 Marktevaluatie

In een blog van Warriier (2022) beschrijft hij Lightsail als een gemakkelijke tool om eenvoudige workloads te beheren en voorgeconfigureerde infrastructuur, zoals toepassingen, databanken en andere populaire stacks, te provisioneren.

## 2.4 AWS Copilot

AWS Copilot is een Command Line Interface (CLI) voor ontwikkelaars die gecontaineriseerde toepassingen willen inzetten op AWS-services zoals AWS AppRunner, AWS Elastic container service (ECS) en AWS Fargate. De volledige levenscyclus van de toepassing kan via Copilot worden beheerd. Componenten zoals omgevingen en services kunnen m.b.v. de tool worden geconfigureerd. Vervolgens worden de hoofdconcepten van Copilot uitgelicht (Karakus, 2022).

### 2.4.1 Applications

Een applicatie is een collectie van services en omgevingen. De applicatienaam is dan meestal een high-level beschrijving van het product. Bijvoorbeeld een applicatie met naam "Chat" kan twee services bevatten, "frontend" en "api". Verder kunnen de services ingezet worden op een "test" en "production" omgeving (Karakus, 2022).

### 2.4.2 Environments

Het concept van omgevingen is door iedere ontwikkelaar gekend. Voordat de toepassing op een productieve omgeving kan worden vrijgesteld is het aangeraden om eerst een test omgeving te voorzien. Deze testomgeving is dan een weerspiegeling van wat er uiteindelijk in productie zal staan.

Elke omgeving kan zijn eigen versie van een service bevatten. De optimale werkwijze zou zijn om, voordat een service ingezet wordt op de productieomgeving, die eerst te testen op de testomgeving.

Verder bestaat ook de mogelijkheid om een nieuwe service aan een bestaande omgeving toe te voegen. Elke omgeving bevat een set van AWS-resources die gedeeld wordt binnen die omgeving. Deze omvatten netwerk-resources, de ECS-cluster en de load-balancer. Een nieuwe service zou gebruik maken van deze gedeelde resources (Karakus, 2022).

### 2.4.3 Services

Een service bestaat uit de code van de toepassing en alle ondersteunende infrastructuur die nodig is om de toepassing op AWS te kunnen hosten. De gebruiker kiest via de CLI



welke type service moet worden aangemaakt. Afhankelijk van het type service worden de nodige AWS-resources bevoorrad. Ter illustratie, een gebruiker kan kiezen voor een service om trafiek van het internet te bedienen. Daarbij horen AWS-resources zoals een Application Load Balancer (ALB) en een ECS service op AWS Fargate.

Als de gebruiker een service type heeft gekozen, wordt er een Dockerfile gegenereerd en de container images worden bewaard op Elastic Container Repository (ECR). Ook wordt er een *manifest* bestand aangemaakt, deze bevat configuratie parameters voor de service zoals hoeveelheid geheugen, CPU, het aantal kopieën van de service en andere (Karakus, 2022).

#### 2.4.4 Jobs

Jobs zijn vluchtig ECS tasks die getriggerd worden door een event. Wanneer een job voldaan is, wordt de ECS task beëindigd. De gebruiker dient alle nodige configuratie te voorzien voor de opzet van de ECS task. Daarnaast wordt er ook een *manifest* bestand voorzien waar geavanceerde configuratie kan worden ingesteld (Karakus, 2022).

#### 2.4.5 Pipelines

Een Continuous Deployment (CD) pipeline kan worden opgezet via Copilot. Copilot is compatibel met de volgende git repositories: GitHub, BitBucket en CodeCommit. Wanneer een push wordt gedetecteerd, de pipeline bouwt de service, bewaart de container image op ECR en begint de inzet naar een omgeving (Karakus, 2022).

## 2.5 AWS Elastic Beanstalk

Elastic Beanstalk (EB) is een andere AWS service die het inzetten van toepassingen op AWS-infrastructuur aanzienlijk vereenvoudigt. Het beheer van AWS-infrastructuur wordt grotendeels door EB uitgevoerd. En toch behoudt de gebruiker de mogelijkheid om configuratiekeuzen te maken.

Elastic Beanstalk ondersteunt toepassingen ontwikkeld in Go, Java, .NET, Node.js, PHP, Python en Ruby. Wanneer een toepassing wordt ingezet, bouwt EB de geselecteerd platform versie en voorziet de nodige AWS-resources. De gebruiker kan Elastic Beanstalk gebruiken via de AWS Command Line Interface (CLI), of *eb*, een high-level CLI doelbewust voor Elastic Beanstalk ontworpen.

Bovendien is het ook mogelijk om via EB alledaagse inzet gerelateerde taken uit te voeren. Zoals de grootte van de EC2 fleet instanties aanpassen of het monitoren van de toepassingen. Deze acties kunnen rechtstreeks vanuit het EB webinterface (console) uitgevoerd worden (Khen, 2022a).

Het gebruik van Elastic Beanstalk bedraagt geen bijkomende kosten. De gebruiker betaalt

voor de AWS-resources die ingezet worden om de toepassing op AWS te hosten (Khen, 2022a).

## 2.6 AWS Cloudformation

Cloudformation is een AWS-service voor het modelleren en opzetten van AWS-resources. Complexe opstellingen, waarbij meerdere AWS-resources van elkaar afhankelijk zijn en samen functioneren, kunnen als één geheel geconfigureerd worden. Een Cloudformation template moet meerdere keren kunnen gebruikt worden. Het is daarom ook mogelijk om via parameters de template dynamisch te maken. De gebruiker kan a.d.h.v. deze parameters de configuratie van de AWS-resources anders instellen zonder de template elke keer te moeten aanpassen. Voor infrastructuur en AWS-resources die via Cloudformation zijn opgezet bestaat de mogelijkheid om aanpassingen te brengen via change-sets. Een alternatief is om bestaande infrastructuurcomponenten in een Cloudformation stack te opnemen. Daarnaast biedt Cloudformation ook de functionaliteit aan om infrastructuur eenvoudig af te bouwen (Mansoor, 2014).

### 2.6.1 Template

Cloudformation werkt op basis van templates. Een template is een JSON- of YAML-bestand waarin de gewenste AWS-resources en configuratie worden beschreven. Deze templates moeten op een S3 bucket beschikbaar gesteld worden om door Cloudformation gebruikt te kunnen worden. De mogelijkheid bestaat ook om naar een bestand op een lokaal systeem te verwijzen via de console, API of CLI. AWS zorgt in dat geval dat het bestand wordt opgeladen op een S3 bucket. Templates kunnen ook dynamisch geïmplementeerd worden door gebruik te maken van parameters. Deze parameters worden gespecificeerd bij het aanmaken of actualiseren van een stack. Zo wordt er vermeden om telkens opnieuw de template te moeten aanpassen (AWS, g.d.-b).

### 2.6.2 Stack

Een Cloudformation stack is een collectie van AWS-resources die beheerd kunnen worden als één geheel. De stack wordt aangemaakt op basis van een Cloudformation template. Indien de template parameters bevat, dan kunnen die gespecificeerd worden bij het aanmaken van de stack. Anders bestaat ook de mogelijkheid om default waarden in te stellen voor de parameters. Nadat een stack is aangemaakt kan die geactualiseerd worden a.d.h.v. change-sets. De aangemaakte resources kunnen eenvoudig afgebouwd worden door de stack te verwijderen (AWS, g.d.-b).

### 2.6.3 Change set

De resources van een bestaande stack kunnen aangepast worden via een change set. Daarvoor moet één of beide van de volgende acties uitgevoerd worden:

- Aanpassingen brengen aan de bestaande template.
- Nieuwe input parameters specificeren.

Een change set is de vergelijking van de huidige template t.o.v. de aangepaste versie. In de change set worden de voorgestelde aanpassingen opgesomd. Eens goedgekeurd worden de resources voor die stack bijgewerkt (AWS, g.d.-b).

### 2.6.4 Delete stack

Bij het verwijderen van een stack worden de betreffende AWS-resources afgebouwd. Het is mogelijk om bepaalde resources niet te verwijderen door een deletion policy te configureren waarbij de gewenste resources behouden blijven. Indien één van de resources niet verwijderd kan worden dan kan de stack ook niet verwijderd worden. In dat geval blijft de stack bestaan tot de systeemcomponenten succesvol verwijderd kunnen worden, of als alternatief kan aangegeven worden om deze te behouden buiten de stack (AWS, g.d.-b).

## 2.7 Terraform

Terraform is een HashiCorp open-source Infrastructure as Code (IaC) tool. Hashicorp is een Advanced Technology Partner bij AWS Partner Network (APN) en ook een lid van de AWS DevOps Competency (Campbell, 2018). Terraform is een gelijkaardige tool aan Cloudformation. Beide werken met declaratieve configuratie bestanden waarmee AWS-infrastructuur kan worden ingezet en geactualiseerd.

Terraform is platform-agnostisch; de tool is compatibel met on-site servers of cloud servers zoals AWS, Google Cloud Platform, OpenStack en Azure. Het is mogelijk om meerdere cloud-leveranciers, voor verschillende doeleinden, te gebruiken. Dit betekent niet dat de gebruiker zonder gevolgen van cloud-leverancier kan schakelen. De cloud-providers hebben elk een andere werkmethode en aanbod aan resources. Daarom is spontaan van cloud-provider nooit een optimale oplossing. Toch is de flexibiliteit en compatibiliteit van Terraform een grote troef (Szalski, 2019).

Terraform werkt met configuratie bestanden zoals de meeste declaratieve IaC tools. Deze bestanden worden geschreven in HashiCorp Configuration Language (HCL), de configuratie taal van HashiCorp. De syntax heeft eigenschappen zoals variabelen en interpolatie, en de documentatie is zeer goed. In een Terraform configuratie bestand worden resources en data sources beschreven. Een resource vertegenwoordigt een component van de infrastructuur zoals een AWS EC2 instantie, een RDS instantie, een Route53 DNS record of een regel in een security group. Op deze manier kan de gebruiker de resources

configureren en inzetten op de cloud architectuur (Szalski, 2019).

## 2.8 Ansible

## 2.9 AWS SAM

## 2.10 AWS CDK

Cloud Development Kit (CDK) is een open source software development framework gebruikt voor het schrijven van Infrastructure as Code (IaC) in een voor de ontwikkelaar vertrouwde programmeertaal zoals:

- TypeScript
- JavaScript
- Python
- Java
- C#

AWS CDK gebruikt de voordelen van een programmeertaal om infrastructuur en toepassingen te schrijven, lezen en begrijpen (Mansoor, 2014).

De CDK-framework compileert de code en genereert een Cloudformation stack. Daarom gelden dezelfde principes op deze tool als bij Cloudformation. Via de CDK-Toolkit, een CLI-interface voor CDK, kunnen de standaard acties zoals een stack creëren, wijzigen en verwijderen uitgevoerd worden. Daarnaast zijn troubleshooting functionaliteiten ook beschikbaar via de CDK-Toolkit (Mansoor, 2014).

### App

De App construct is de root klasse van alle constructs. De klasse heeft geen argumenten nodig. Bij het aanmaken van een Stack construct moet er verwezen naar een App construct.

### Stack

Een stack is de eenheid van inzet voor AWS CDK. Een CDK-stack is gelijkwaardig aan een Cloudformation-stack en beschikt over dezelfde beperkingen. Alle AWS-resources die gedefinieerd worden binnen een stack worden als één geheel beschouwd.

### Construct

Een construct ligt aan de basis van een CDK-app. Het staat voor een AWS-resource en bevat alle configuratie die nodig is voor Cloudformation om de component te bouwen. Een

construct kan één AWS-resource vertegenwoordigen zoals een AWS S3 bucket of kan bestaan uit meerdere resources waarbij men spreekt over een higher-level abstraction construct. De AWS CDK library bevat een collectie van constructs voor elke AWS-resource. Er bestaan ook meer complexe third-party constructs die beschikbaar gesteld worden via de Construct Hub.

### CDK-Toolkit

Met behulp van de toolkit CLI kunnen alle functionaliteiten uitgevoerd worden die nodig zijn om de applicatie en infrastructuur te kunnen inzetten en beheren zoals:

**Synthesis:** De synth commando compileert de code en genereert een Cloudformation template.

**Deploy:** De deploy commando zal de stack inzetten naar de geconfigureerde AWS-account, voor elke inzet wordt de code gesynthetiseerd.

**Diff:** De diff commando vergelijkt de laatste stack met de laatste gesynthetiseerde template, de output is een overzicht van de verschillen tussen de twee.

**Destroy:** De destroy commando verwijdert de stack en de bijhorende AWS-resources, dit is gelijkwaardig aan het verwijderen van een Cloudformation stack.

## 2.11 Troposphere

## 2.12 Pulumi



## 3. Methodologie

Dit onderzoek bestaat uit een literatuurstudie waarbij de verschillende Infrastructure as Code (IaC) services in kaart worden gebracht. In dit onderzoek wordt specifiek gekeken naar tools en services die compatibel zijn met AWS. Het is dan ook geen verrassing dat de meeste services door AWS zelf worden aangeboden. De literatuurstudie is te vinden in Hoofdstuk 3.1.

In Hoofdstuk 4 wordt de evaluatie en vergelijking gedaan van de eerder besproken iac tools. De criteria, als beschreven in Hoofdstuk 3.1, wordt toegepast op de volgende tools:

- AWS AppRunner
- AWS Amplify
- AWS Lightsail
- AWS Copilot
- AWS Elastic Beanstalk
- AWS Cloudformation
- Terraform
- Ansible
- AWS SAM
- AWS CDK
- Troposphere
- Pulumi

Om op de criteria te kunnen antwoorden wordt er gekeken naar verschillende bronnen:

- Documentatie tools/services
- Artikels geschreven door gebruikers

- Startersgidsen

In Hoofdstuk 5 wordt een business case uitgewerkt. De business case zal bestaan uit een aantal requirements waaraan de IaC tool moet voldoen. De tools die de business case werkelijk kunnen uitwerken, zonder uitbreidingen naar andere services, zullen geselecteerd worden. In Hoofdstuk 6 wordt de business case uitgewerkt met de geselecteerde tools.

### 3.1 Criteria evaluatie

De services en tools zullen op bepaalde criteria geëvalueerd worden. De criteria is grotendeels afhankelijk van het type onderneming, een startup, een KBO of een grote onderneming. Aan de hand van een vergelijkende studie wordt er bepaald welke tool in welke situatie het best wordt gebruikt.

1. Initiële inzet
  - Training
  - Kost opzet
  - Documentatie en/of support
2. Operationele effectiviteit
  - State Validatie Rollback Drift elimination
  - Maturiteit
  - Onderhoudbaarheid
  - Veiligheid (least privilege implementation, RBAC, transparantie)
  - Categorisatie (tagging, resource groups, omgevingen)
3. Gebruiksgemak/Gebruikscomfort
  - Herbruikbaarheid
  - Ready-made (abstractieniveau)
  - Cross-platform support
  - Syntax
  - Open-source
4. Total cost of ownership (TCO)
  - Infrastructuur
  - Support
5. Flexibiliteit
  - Integratie met een bestaande omgeving
  - Integratie/collaboratie met andere IaC tools
  - Integratie met andere DevOps tools
  - Integratie met andere development frameworks

In de eerste plaats wordt er gekeken naar de eerste inzet die nodig is om de tool of service in gebruik te nemen. Dit houdt in de kosten gebonden aan het gebruik van de tool, eventuele trainingen als het gaat over een complexe tools die voorkennis vereisen. De documentatie is ook een grote aanwijzing voor de waarde van de tool. Goed gestructureerde en gesubstantieerde documentatie maakt een eerste kennismaking met de tool moeiteloos.



De operationele effectiviteit van de tool wordt ook onder de loep genomen. De maturiteit en de onderhoudbaarheid van de tool spelen een grote rol als een gebruiker een keuze moet maken tussen de verschillende tools. Een tool met een lange levensduur die nog steeds wordt onderhouden zal eerder als betrouwbaar geacht worden. Binnen IaC zijn er bepaalde functionaliteiten die de gebruiker op een georganiseerde manier laten werken en het beheer van de infrastructuur ondersteunen. Het al dan niet beschikken van deze functionaliteiten zal ook wegen bij het evalueren van de tools. Een aantal van deze functionaliteiten wordt hieronder beschreven.

**State:** De tool geeft de mogelijkheid om de status van de gereproduceerde infrastructuur op te volgen. Als er wijzigingen zouden plaatsvinden dan worden deze bij de state opgevangen.

**Validatie:** Als er sprake is van code dan wordt er verwacht dat deze aan bepaalde syntax of formaat eisen voldoet. De validatie moet ervoor zorgen dat voor het genereren van de infrastructuur de nodige meldingen worden weergegeven als iets niet klopt.

**Rollback:** De mogelijkheid om de gegenereerd infrastructuur te verwijderen of wijzigingen terug te draaien.

**Drift elimination:** De functionaliteit om afwijkingen tussen de state en de huidige implementatie te kunnen detecteren. Wijzigingen die buiten de scope van de IaC tool worden aangebracht zijn in dit geval een afwijking.

**Veiligheid:** Op vlak van veiligheid wordt er gekeken naar de functionaliteit rond toegankelijkheid. Het is van uiterst belang dat enkel toegestane gebruikers infrastructuur kunnen genereren of wijzigen. Concepten zoals least privilege, roll based access control (RBAC) en policies horen hierbij.

**Categorisatie:** Functionaliteit om infrastructuur te beheren op een ordelijke manier zoals tagging, resource groups en omgevingen.

Onder de categorie gebruiksgemak worden zaken zoals de herbruikbaarheid van de tool bestudeerd. De mogelijkheid om verschillende omgevingen te kunnen opzetten vertrekende van dezelfde infrastructuur template. De complexiteit van de tool is een determinerende factor. Sommige tools eisen de volledige configuratie van de gedefinieerde AWS-resources. Andere tools vragen helemaal geen configuratie, de gebruiker moet zich dan enkel bezighouden met de ontwikkeling van de toepassing. Afhankelijk van het niveau van abstractie kan een tool heel complex of heel eenvoudig zijn. De syntax is een factor die meespeelt bij tools waar het vertrekpunt een template of code is. Als de code in een programmeertaal, die niet exclusief is voor die tool, is geschreven dan moet de code voldoen aan de regels van dat programmeertaal. Bepaalde tools beschikken over hun eigen syntax, andere gebruiken generieke bestandsformaten zoals JSON en YML.

De Total cost of ownership (TCO) is een andere belangrijke factor. Het budget waarover een onderneming beschikt zal ook meespelen bij de keuze van een IaC tool. Niet alle ondernemingen kunnen de kosten om een complexe tool te gebruiken verantwoorden, de expertise die nodig is om de tool in gebruik te nemen kan extra kosten betekenen.



## 4. Vergelijking services en tools

In dit hoofdstuk worden de IaC tools geëvalueerd en vergeleken op basis van de criteria in Hoofdstuk 3.1. De tools worden in drie categorieën verdeeld:

- Declaratieve tools
- Programmatieve tools
- Intuïtieve/externe tools

Declaratieve tools

### 4.1 Initiële inzet

Tool/Service	Training	Kost
AWS AppRunner	De gebruiker moet een basis kennis hebben van AWS. Een ontwikkelaar zou redelijk snel de tool in gebruik kunnen nemen zonder speciale trainingen.	De kost voor een geprovisioneerde container instantie is \$ 0.007/GB-uur. Voor een actieve container instantie is de kost \$ 0.064/vCPU-uur en \$ 0.007/GB-uur. Daarnaast zijn er optionele add-ons en extra kosten als de toepassing andere AWS-services gebruikt.
AWS Amplify	TODO	Voor het gebruik van de Amplify Framework (libraries, CLI, UI Components) betaalt de gebruiker enkel voor de onderliggende AWS-resources. De kosten voor het hosten van een statische webtoepassing zijn de volgende: <ul style="list-style-type: none"> <li>• \$ 0.01 per build minuut</li> <li>• \$ 0.023 per GB per maand</li> <li>• \$ 0.15 per GB</li> </ul> Publieke SSL-certificaten inclusief.
AWS Lightsail	TODO	Met Lightsail is de kost laag en voorspelbaar. Deze service bundelt resources zoals geheugen, vCPU en SSD in een plan. Alle plannen includeren een statisch IP-adres, DNS, SSH-terminaal toegang, SSD-opslag en monitoring.
AWS Copilot	TODO	Er zijn geen kosten verbonden aan het gebruik van deze tool. De gebruiker betaalt enkel voor de onderliggende AWS-resources.
AWS Elastic Beanstalk	TODO	Er zijn geen kosten verbonden aan het gebruik van deze service. De gebruiker betaalt enkel voor de onderliggende AWS-resources. Het is aan de gebruiker om de grootte en het aantal AWS-resources te bepalen.

Tabel 4.1: Initiële inzet test

<b>Tool</b>	<b>Documentatie</b>	<b>Training</b>	<b>Kost</b>
AWS Cloudformation	test	test	test
AWS Terraform	test	test	test
AWS SAM	test	test	test

Tabel 4.2: Initiële inzet declaratieve tools

<b>Tool</b>	<b>Documentatie</b>	<b>Training</b>	<b>Kost</b>
AWS CDK	test	test	test
Troposphere	test	test	test
Pulumi	test	test	test

Tabel 4.3: Initiële inzet programmatieve tools



## 5. Business case

### 5.1 Functionele requirements

- Een opslagplaats waar bestanden langdurig en veilig kunnen bewaard worden.
- De metadata van de bewaarde bestanden wordt automatisch opgeslagen.
- Een API-service om de metadata van de bewaarde bestanden te kunnen raadplegen en/of te bewerken.

### 5.2 AWS Serverless oplossing

De serverless oplossing bestaan uit de volgende AWS-resources:

**AWS S3 bucket** opslag voor de bestanden

**AWS Lambda functie 1** S3 events verwerker - bij het toevoegen van een bestand aan de S3 bucket

**AWS Lambda functie 2** Backend voor de API-gateway, verwerker van de API-calls

**AWS API-gateway** API-service voor het raadplegen of bewerken van de metadata

**AWS DynamoDB** NoSQL databank voor het opslaan van de metadata





## 6. Uitwerking business case

TODO uitwerking cloudformation / AWS CDK / Terraform

### 6.1 Uitwerking AWS CDK

TODO

### 6.2 Uitwerking AWS Cloudformation

TODO

### 6.3 Uitwerking Terraform

TODO



## 7. Conclusie

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer

blandit lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetur libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem.

Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

# A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

## A.1 Introductie

Hier introduceer je werk. Je hoeft hier nog niet te technisch te gaan.

Je beschrijft zeker:

- de probleemstelling en context
- de motivatie en relevantie voor het onderzoek
- de doelstelling en onderzoeksvraag/-vragen

## A.2 State-of-the-art

Hier beschrijf je de *state-of-the-art* rondom je gekozen onderzoeksdomein. Dit kan bijvoorbeeld een literatuurstudie zijn. Je mag de titel van deze sectie ook aanpassen (literatuurstudie, stand van zaken, enz.). Zijn er al gelijkaardige onderzoeken gevoerd? Wat concluderen ze? Wat is het verschil met jouw onderzoek? Wat is de relevantie met jouw onderzoek?

Verwijs bij elke introductie van een term of bewering over het domein naar de vakliteratuur, bijvoorbeeld (Doll & Hill, 1954)! Denk zeker goed na welke werken je refereert en

waarom.

Je mag gerust gebruik maken van subsecties in dit onderdeel.

### A.3 Methodologie

Hier beschrijf je hoe je van plan bent het onderzoek te voeren. Welke onderzoekstechniek ga je toepassen om elk van je onderzoeksvragen te beantwoorden? Gebruik je hiervoor experimenten, vragenlijsten, simulaties? Je beschrijft ook al welke tools je denkt hiervoor te gebruiken of te ontwikkelen.

### A.4 Verwachte resultaten

Hier beschrijf je welke resultaten je verwacht. Als je metingen en simulaties uitvoert, kan je hier al mock-ups maken van de grafieken samen met de verwachte conclusies. Benoem zeker al je assen en de stukken van de grafiek die je gaat gebruiken. Dit zorgt ervoor dat je concreet weet hoe je je data gaat moeten structureren.

### A.5 Verwachte conclusies

Hier beschrijf je wat je verwacht uit je onderzoek, met de motivatie waarom. Het is **niet** erg indien uit je onderzoek andere resultaten en conclusies vloeien dan dat je hier beschrijft: het is dan juist interessant om te onderzoeken waarom jouw hypothesen niet overeenkomen met de resultaten.

# Acroniemen

**ALB** Application Load Balancer. 23  
**API** Application programming interface. 20, 24  
**APN** AWS Partner Network. 25  
**AWS** Amazon Web Services. 23, 29

**CD** Continuous Deployment. 19, 21, 23  
**CDK** Cloud Development Kit. 26, 27  
**CDN** Content delivery network. 21  
**CI** Continuous Integration. 19  
**CLI** Command Line Interface. 20–24, 26, 27

**DNS** Domain Name System. 25

**EB** Elastic Beanstalk. 23  
**EC2** Elastic Compute Cloud. 23, 25  
**ECR** Elastic Container Repository. 23  
**ECS** Elastic Container Service. 22, 23

**HCL** HashiCorp Configuration Language. 25

**IaC** Infrastructure as Code. 19, 25, 26, 29, 30, 33

**RDS** Relational Database Service. 25

**S3** Simple Storage Service. 24

**TCO** Total cost of ownership. 30, 31





## Woordenlijst

**image** Een snapshot van een container waar er al bepaalde software is geïnstalleerd en/of geconfigureerd. 19, 23

**security group** Een security group is een virtuele firewall van AWS, deze bepaalt welke trafiek wordt toegelaten. 25



## Bibliografie

- Aussems, M. (2021, mei 19). *AWS App Runner ontfermt zich over uitrol en beheer container-apps* (ITdaily, Red.). <https://itdaily.be/nieuws/cloud/aws-app-runner-ontfermt-zich-over-uitrol-en-beheer-container-apps/>
- AWS. (g.d.-a). *AWS AppRunner Pricing* (AWS, Red.). <https://aws.amazon.com/apprunner/pricing/>
- AWS. (g.d.-b). *What is AWS CloudFormation?* (AWS, Red.). <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-what-is-concepts.html>
- AWS. (2022a, januari 1). *Amazon Lightsail features* (AWS, Red.). <https://aws.amazon.com/lightsail/features/>
- AWS. (2022b, januari 1). *Amplify Framework Documentation* (AWS, Red.). <https://docs.amplify.aws/#amplify-javascript>
- AWS. (2022c, januari 1). *AWS Amplify Overview* (AWS, Red.). <https://aws.amazon.com/amplify/>
- Campbell, J. (2018, maart 1). *Terraform: Beyond the Basics with AWS* (AWS, Red.). <https://aws.amazon.com/blogs/apn/terraform-beyond-the-basics-with-aws/>
- Doll, R., & Hill, A. B. (1954). The mortality of doctors in relation to their smoking habits: a preliminary report. *British Medical Journal*, 328(7455), 1529–1533.
- Kandaswamy, S. (2022, maart 21). *Build a Serverless Full-Stack Registration App in minutes using AWS Amplify*. <https://aws.amazon.com/blogs/mobile/build-a-serverless-full-stack-registration-app-in-minutes-using-aws-amplify/>
- Karakus, E. (2022, januari 1). *AWS Copilot CLI Documentation* (AWS, Red.). <https://aws.github.io/copilot-cli/docs/overview/>
- Khen, D. (2022a, januari 20). *AWS Elastic Beanstalk Developer Guide*. <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/awseb-dg.pdf#Welcome>

- Khen, D. (2022b, februari 28). *AWS App Runner Developer Guide* (AWS, Red.). <https://docs.aws.amazon.com/apprunner/latest/dg/apprunner-guide.pdf#what-is-apprunner>
- Mansoor. (2014, december 1). *Introduction to DevOps on AWS*. <https://docs.aws.amazon.com/whitepapers/latest/introduction-devops-aws/introduction-devops-aws.pdf#infrastructure-as-code>
- Nwamba, C. (2022, april 22). *From Figma to Next.js App in Minutes*. <https://dev.to/codebeast/from-figma-to-nextjs-app-in-minutes-4jp9>
- Pollefliet, L. (2011). *Schrijven van verslag tot eindwerk: do's en don'ts*. Academia Press.
- Spittel, A. (2022, mei 4). *Write Your Own Code with AWS Amplify Studio* (AWSBlogs, Red.). <https://aws.amazon.com/blogs/mobile/write-your-own-code-with-aws-amplify-studio/>
- Szalski, R. (2019, juli 12). *Terraform AWS Cloud: Sane Infrastructure Management* (Top-tal, Red.). <https://www.toptal.com/devops/terraform-aws-cloud-iac>
- Warrier, R. (2022, maart 30). *Monitoring the Health of Amazon Lightsail Instances with Datadog* (AWSBlog, Red.). <https://aws.amazon.com/blogs/apn/monitoring-the-health-of-amazon-lightsail-instances-with-datadog/>