THOMAS MORE

EEN PROOF OF CONCEPT VOOR DE MIGRATIE

Cloud-native Drupal on AWS

Door: Ahmet Nuri Uygun

Hogeschoolpromoter: Lars Struyf

Bedrijfspromoter: Sylvie De Paepe

PBA Elektronica-ICT.
Afstudeerrichting ICT

Bachelorproef voorgedragen tot het behalen van de graad en het diploma van bachelor

Campus De Nayer

Academiejaar 2025 - 2026

Woord vooraf

Deze scriptie is geschreven in het kader van mijn bachelorproef ter afronding van de opleiding Elektronica-ICT – Application Development aan Thomas More Campus De Nayer te Sint-Katelijne-Waver. Van februari tot op heden heb ik intensief gewerkt aan de realisatie van dit project.

De stageperiode was voor mij een uitstekende gelegenheid om mijn kennis op het gebied van AWS en Infrastructure as Code (IaC) aanzienlijk uit te breiden. Daarnaast bood het mij de kans om de theoretische fundamenten van netwerken, databases en applicatiebeveiliging om te zetten in de praktijk.

Uiteraard heb ik dit proces niet alleen doorlopen. In het bijzonder wil ik Michali Tsompanoglu bedanken, die mij gedurende vier maanden als technisch begeleider heeft bijgestaan. Hij stond niet alleen dagelijks klaar voor technische ondersteuning, maar speelde ook een cruciale rol in mijn integratie binnen het bedrijf.

Daarnaast gaat mijn dank uit naar Sylvie, mijn hogeschoolpromotor, en Klaas, mijn teamleader, voor hun voortdurende steun en advies tijdens deze stage. Tot slot wil ik alle collega's bij Randstad Digital bedanken voor de prettige samenwerking en de leerrijke omgeving die zij mij hebben geboden.

Samenvatting

Moderne softwarebedrijven staan onder toenemende druk om hun infrastructuur schaalbaar, veilig en kostenefficiënt te maken. Randstad Digital host haar Drupal-applicaties momenteel op private on-premise servers, wat leidt tot problemen met schaalbaarheid bij verkeerspieken, hoge beheerslast en een trage time-to-market. Om deze beperkingen aan te pakken, werd onderzocht hoe een cloud-native architectuur op Amazon Web Services (AWS) deze pijnpunten kan oplossen.

Als methode werd een Proof of Concept (POC) opgezet waarbij de Drupal-applicatie van "Kom op Tegen Kanker" werd gecontaineriseerd via Docker en gedeployed op AWS ECS Fargate. De volledige infrastructuur — netwerk, database, opslag en caching — werd geautomatiseerd beheerd via Terraform als Infrastructure as Code. Een CI/CD-pipeline in Bitbucket Pipelines, aangevuld met Snyk-beveiligingsscans, zorgde voor een volledig geautomatiseerde build- en deploymentcyclus. De werking werd gevalideerd via connectiviteitstesten, een load-test met 100 gelijktijdige gebruikers en een live deploymentvalidatie.

De POC bewijst dat een cloud-native Drupal-architectuur op AWS een haalbaar en robuust alternatief is voor de huidige on-premise aanpak. Het systeem schaalde automatisch op van 1 naar 3 instanties bij hoge belasting, met een foutmarge onder 5,45%. De gelaagde beveiliging via Security Groups en AWS Secrets Manager garandeert een veilige omgeving. De gerealiseerde infrastructuur biedt Randstad Digital een concrete blauwdruk voor een schaalbare, geautomatiseerde en kostenefficiënte Drupal-hosting in de cloud.

Lijst met tabellen

Lijst met figuren

Lijst met afkortingen en symbolen

Afkorting	Beschrijving
IAC	Infrastructure As Code
AWS	Amazon Web Services
CMS	Content Management System
POC	Proof Of Concept
CI / CD	Continuous Integration / Continuous Development
ECS	Elastic Container Service
EC2	Elastic Compute Cloud
VPS	Virtual Private Server
HCL	Hashicorp Configuration Language
RDS	Relation Database Service
SG	Security Group
EFS	Elastic File System
VPC	Virtual Private Cloud
AZ	Availability Zone
ALB	Application Load Balancer
IGW	Internet Gateway

Opstartverslag

Gegevens student(en)	
Student 1: Ahmet Nuri Uygun **Opleiding:** Elektronica-ICT **GSM:** +32 491 20 72 92	
Gegevens bedrijf / instelling	
Naam: Randstad Digital Belgium **Adres:** Kolonel Begautlaan 1a, 3012 Leuven **Tel:** 016 65 20 16	
Gegevens bedrijfspromoter	
Naam: Sylvie De Paepe **Functie:** Project Manager **Afdeling:** Klik of tik om tekst in te voeren.	**Tel/GSM:** +32 490 01 41 18 **Email:** sylvie.depaepe@randstaddigital.com

Gegevens bachelorproef

Titel bachelorproef

Devops Randstad Digital

Omschrijving van het probleem

Er is momenteel geen gestandaardiseerde, schaalbare en veilige AWS-infrastructuur beschikbaar om onze lagere (DTA) omgevingen van Drupal-websites te hosten adhv Docker

Doelstellingen van de bachelorproef (kwantitatief en meetbaar)

De stagiair gaat aan de slag om de minimale requirements te achterhalen om een Drupal website in een Docker omgeving op te zetten. Hierbij houdt hij rekening met persistente data en caching. Success betekent dat al onze huidige integraties ook in deze POC omgeving op AWS beschikbaar zijn.

CI/CD

De AWS-omgeving wordt opgezet met Infrastructure as Code (Terraform). De Drupal- webapplicatie wordt gebouwd en gedeployed in een Docker-container. De stagiair onderzoekt de minimale vereisten om dit proces te realiseren en zorgt ervoor dat de opgeleverde Docker- container “productiewaardig” is door enkel de essentiële componenten te behouden.

Documentatie en onderzoek

De documentatie dient als leidraad voor het project en biedt verdieping van de gemaakte keuzes, zodat deze inzichtelijk en herbruikbaar blijven.

Testen

Testen van de webapp met een van onze developers. Eventueel doen we ook een loadtest (50- 100 bezoekers).

Gewenst resultaat: wat moet er (minimaal) opgeleverd worden?

De Drupal-website draait volledig in AWS met geautomatiseerde pipelines, met nadruk op integratiemogelijkheden.

Duidelijke documentatie met betrekking tot de gemaakte keuzes.Verdere uitbreidingen kunnen worden uitgevoerd indien er extra tijd beschikbaar is.

Wat is de planning (activiteiten van de student + includeer ook planning m.b.t. het schrijven van de scriptie)?

- **Week 1: Kick-off & Scope** – Verzamelen van requirements, bepalen van de integraties en het opzetten van de high-level architectuur (bij voorkeur ECS Fargate) en repo-structuur.
- **Week 2: Lokale Drupal-omgeving** – Opzetten van een lokale Docker-stack met Drupal/PHP en een database.
- **Week 3: Productiewaardige Docker-images** – Optimaliseren van Dockerfiles (multi-stage, non-root) en het inregelen van healthchecks.
- **Week 4-5: Terraform Fundament & Database** – Opzetten van de cloud-infrastructuur (VPC, ECR, netwerk) en het configureren van een RDS-database in een privaat subnet.
- **Week 6: ECS Fargate + ALB** – Uitrollen van de ECS-cluster en service, inclusief een Application Load Balancer (ALB) en logging via CloudWatch.
- **Week 7: Persistent Files (EFS)** – Configureren van Amazon EFS voor permanente opslag van Drupal-bestanden.
- **Week 8: Caching + Cron** – Implementatie van Redis (ElastiCache) voor caching en het instellen van Drupal-cronjobs.
- **Week 9-10: CI/CD Pipelines** – Bouwen van pipelines voor zowel de applicatie (build, scan, deploy) als de infrastructuur (Terraform plan/apply).
- **Week 11: Integraties & Testen** – Uitvoeren van end-to-end testen en een loadtest (50-100 gebruikers) om de schaalbaarheid te controleren.
- **Week 12: Documentatie & Afronding** – Opstellen van runbooks, ADR's (Architectural Decision Records), een kostenindicatie en het geven van de einddemo

Opgemaakt te **op** 09/02/2026

Klik of tik om een datum in te voeren.

Handtekeningen

Bedrijfspromotor	Hogeschoolpromotor	Student 1
Sylvie De Paepe	Lars Struyf	Ahmet Nuri Uygun

Thomas More
Campus De Nayer
Jan De Nayerlaan 5, 2860 Sint-Katelijne-Waver, België
+ 32 15 31 69 44 | info.denayer@thomasmore.be | www.thomasmore.be

1. Inleiding

Drupal is een krachtig Content Management Systeem (CMS) geschreven in PHP, dat sinds de eerste release in 2001 is uitgegroeid tot een van de meest betrouwbare platformen voor ondernemingen. Hoewel het marktaandeel van Drupal kleiner is in vergelijking met andere frameworks, wordt het platform vertrouwd door vooraanstaande instanties zoals NASA, Harvard University, Tesla en The Economist.

Deze voorkeur komt voort uit de focus van Drupal op veiligheid, flexibiliteit en het vermogen om complexe inhoudsstructuren op een professionele manier te beheren. Bovendien biedt Drupal moderne architecturale mogelijkheden, zoals een ‘headless’ opzet, waardoor het naadloos integreert met frontend-frameworks zoals Vue.js of Next.js.

In de hedendaagse ontwikkelomgevingen speelt containerisatie een cruciale rol. De combinatie van Drupal en Docker is een bewezen 'best practice' die consistentie garandeert tussen lokale ontwikkelomgevingen en productie-omgevingen. Echter, de manier waarop deze omgevingen gehost worden, is de afgelopen jaren drastisch veranderd. Met de opkomst van cloud-technologieën zoals Amazon Web Services (AWS) in 2006, gevolgd door Google Cloud Platform en Microsoft Azure, is de focus verschoven van statische servers naar vluchtige, schaalbare en geautomatiseerde infrastructuren.

Binnen de moderne DevOps-wereld is de adoptie van Infrastructure as Code (IaC) een van de belangrijkste trends. In plaats van servers handmatig te configureren, wordt de volledige infrastructuur gedefinieerd in scripttaal via tools zoals Terraform. Dit zorgt ervoor dat complexe cloud-omgevingen op een repetitieve en foutloze manier kunnen worden uitgerold.

Daarnaast is een robuuste CI/CD-pipeline (Continuous Integration & Continuous Deployment) essentieel geworden voor softwarebedrijven. Door gebruik te maken van tools zoals Bitbucket Pipelines, kan het proces van code-analyse, het bouwen van Docker-images en de uiteindelijke deployment naar AWS volledig geautomatiseerd worden. Dit vermindert niet alleen de kans op menselijke fouten, maar versnelt ook de levering van nieuwe functionaliteiten aanzienlijk.

Deze bachelorproef focust op de realisatie van een dergelijke moderne infrastructuur voor Randstad Digital. Het doel is om een Proof of Concept (POC) op te zetten binnen AWS, waarbij Drupal-applicaties op een schaalbare en veilige manier worden gehost met behulp van Docker, Terraform en geautomatiseerde pipelines.

2. Probleemstelling

Momenteel host en beheert Randstad Digital haar Drupal-applicaties op private servers binnen een traditionele on-premise infrastructuur. Hoewel deze methode als veilig wordt beschouwd, brengt ze aanzienlijke beperkingen met zich mee die de groei en efficiëntie van de organisatie belemmeren. In de huidige context van snelle digitale transformatie schiet de huidige aanpak tekort op het gebied van schaalbaarheid, flexibiliteit en kostenefficiëntie.

De kern van het probleem ligt in het gebrek aan een gestandaardiseerde, schaalbare en veilige AWS-infrastructuur voor de lagere omgevingen (Development, Test, Acceptance - DTA). Wanneer een Drupal-website te maken krijgt met een plotselinge piek in het verkeer, kunnen private servers vaak niet snel genoeg resources bijschakelen, wat leidt tot instabiliteit of downtime. Bovendien vereist het handmatig beheren van deze servers veel tijd van systeembeheerders, wat de kans op menselijke fouten vergroot en de 'time-to-market' voor nieuwe features vertraagt.

Er is daarom nood aan een POC die aantoont hoe een overstap naar een cloud-native architectuur deze pijnpunten kan oplossen. Dit project onderzoekt hoe containerisatie en geautomatiseerde cloud-infrastructuren kunnen zorgen voor een omgeving die niet alleen robuuster is, maar ook voldoet aan de moderne DevOps-eisen van reproduceerbaarheid en automatische schaling.

3. Methode

3.1. Docker

Docker fungeert als het fundament voor dit project door applicaties te verpakken in containers. Een container bevat de volledige runtime-omgeving: de code, bibliotheken en systeemtools. Dit lost het bekende probleem van "het werkt op mijn computer, maar niet op de server" definitief op.

Waarom Docker cruciaal is voor Randstad Digital:

- Gestandaardiseerde Omgevingen: Ontwikkelaars hoeven niet langer handmatig pakketten te installeren op hun lokale machine. De container garandeert dat de applicatie overal identiek draait, wat de samenwerking tussen verschillende teams aanzienlijk vereenvoudigt.
- Naadloze CI/CD Integratie: Docker-images vormen de perfecte 'artifact' voor pipelines. Elke build is voorspelbaar en betrouwbaar omdat de omgeving waarin getest wordt dezelfde is als de productieomgeving.
- Cloud-Native Voorbereiding: Door Drupal te containeriseren, kunnen we gebruikmaken van moderne orkestratiediensten zoals AWS Elastic Container Service(ECS) en Fargate, waarbij we de applicatie schalen zonder ons zorgen te maken over de onderliggende serverinfrastructuur.

3.2. AWS

Elke webapplicatie heeft rekenkracht nodig om toegankelijk te zijn. Voor softwarebedrijven vormen de kosten en het beheer van deze infrastructuur een van de grootste uitgavenposten. We maken hierbij het onderscheid tussen twee hoofdvormen van hosting:

3.2.1. Traditionele Hosting

In traditionele modellen betaalt een bedrijf vaak vooraf voor een vaste hoeveelheid resources.

- Dedicated hosting biedt volledige controle over een fysieke server, maar is duur en traag om op te schalen.
- Shared hosting deelt resources met andere websites, wat risico's inhoudt voor de prestaties (verkeerspieken bij buren) en de veiligheid (lekken op andere sites). Dit model vormt een 'single point of failure': als de fysieke server uitvalt, gaat alles offline

3.2.2. Cloud Hosting

Cloud hosting, zoals AWS, biedt een elasticiteit die traditionele hosting niet kan evenaren. In plaats van vooraf te betalen, hanteert men een Pay-as-you-go model waarbij men enkel betaalt voor wat men daadwerkelijk verbruikt. Resources worden mirrored over een cluster van servers. Als één server uitvalt, neemt een andere het direct over zonder downtime. Dit maakt de infrastructuur veel veerkrachtiger.

3.2.3. Waarom de keuze voor AWS?

Hoewel Microsoft Azure bekend staat om zijn sterke reputatie bij grote ondernemingen en Google Cloud uitblinkt in innovatieve micro-projecten, is voor Randstad Digital gekozen voor AWS omdat dit de interne standaard is. AWS was de eerste op de markt en biedt het meest volwassen ecosysteem voor schaalbare Drupal-architecturen.

Table 1 Vergelijking van Cloud Service Providers

Kenmerk	AWS	Azure	Google Cloud
Aantal regio's	36	60+	27+
Best geschikt voor	Breed scala aan workloads en diverse diensten	Microsoft-georiënteerde workloads en hybride cloudomgevingen	Kostengevoelige workloads, AI/ML- en big data-projecten, gecontaineriseerde applicaties
Voordelen	Uitgebreid aanbod aan diensten	Sterke integratie met Microsoft-producten	Sterke focus op AI/ML en big data
Beperkingen	Niet altijd de meest kostenefficiënte optie	Kan minder gebruiksvriendelijk zijn dan AWS	Kleiner wereldwijd netwerk

3.2.4. Vergelijking van AWS Implementatie-approaches

Binnen AWS zijn er verschillende manieren om onze Drupal-container te draaien.

3.2.4.1. Amazon Elastic Compute Cloud (EC2)

Dit zijn virtuele servers die je zelf volledig moet beheren. Hoewel krachtig, vereist het veel onderhoud (security patches, operating system-updates). Het grootste nadeel is de beperkte schaalbaarheid; omdat de database en bestanden vaak aan één virtuele machine gekoppeld zijn, schendt dit de 'high-availability' best practices van AWS.

3.2.4.2. AWS Lightsail

Een vereenvoudigde Virtual Private Server (VPS) oplossing met voorspelbare kosten. Hoewel dit ideaal is voor kleine projecten, is het niet geschikt voor enterprise-workflows of complexe CI/CD-pipelines waarbij meerdere ontwikkelaars betrokken zijn.

3.2.4.3. Amazon Elastic Compute Service(ECS) + Fargate (De Gekozen Oplossing)

ECS is een volledig beheerde orkestratiedienst voor Docker-containers. In combinatie met Fargate wordt dit "serverless".

Met Fargate hoeven we geen onderliggende EC2-instances meer te beheren. AWS regelt de infrastructuur, beveiliging en schaling volledig automatisch. Dit vermindert de operationele lasten voor Randstad Digital aanzienlijk en zorgt voor een kostenbesparing door facturatie per seconde en automatische inperking tijdens rustige periodes.

Table 2 Vergelijking van AWS Implementatie-approaches

Kenmerk	Amazon EC2	AWS Lightsail	Amazon ECS + Fargate
Beheermodel	Infrastructure as a Service	Virtual Private Server	Serverless Container Orchestration
Onderhoud	Hoog (OS patches, updates)	Gemiddeld (Managed VPS)	Minimaal (AWS beheert de infra)
Kostenstructuur	Per uur/seconde (altijd aan)	Vast maandbedrag	Pay-as-you-go (per seconde)
Best geschikt voor	Volledige controle over OS	Kleine, eenvoudige projecten	Enterprise workflows & CI/CD

3.3. Infrastructure as Code

Vóór de opkomst van DevOps was het opzetten van IT-omgevingen een proces dat weken in beslag nam. Systeembeheerders moesten handmatig servers configureren, netwerken instellen en softwarestacks voorbereiden, wat extreem foutgevoelig was. Infrastructure as Code (IaC) lost dit op door infrastructuurvereisten vast te leggen in code in plaats van handmatige handelingen.

3.3.1. Terraform

Voor dit project is gekozen voor Terraform, een open-source tool van HashiCorp die gebruikmaakt van de HashiCorp Configuration Language (HCL). Hoewel cloudproviders zoals AWS eigen tools hebben (bijv. CloudFormation), biedt Terraform cruciale voordelen voor Randstad Digital:

- Consistentie: Het garandeert dat de ontwikkel-, staging- en productieomgevingen identieke kopieën van elkaar zijn.
- Samenwerking: Meerdere teamleden kunnen tegelijk aan de infrastructuur werken via versiebeheer (Git), waarbij elke wijziging gedocumenteerd en traceerbaar is.
- Automatisering: Terraform voert pre-execution checks uit om te zien of de instellingen voldoen aan de verwachtingen voordat de cloud-resources daadwerkelijk worden aangemaakt

3.4. CI / CD Pipelines

Een moderne DevOps-workflow is niet compleet zonder een geautomatiseerde pipeline die Continuous Integration (CI) en Continuous Delivery (CD) faciliteert.

- Continuous Integration (CI): Ontwikkelaars pushen hun code regelmatig naar de main branch. De pipeline voert direct automatische tests en scans uit. Indien een test faalt, wordt de code niet gemerged, wat integratieproblemen in een laat stadium voorkomt.
- Continuous Delivery (CD): Na een succesvolle CI-fase wordt de code automatisch klaargezet voor deployment naar de doelomgeving. Met één druk op de knop kan de nieuwe versie van de Drupal-applicatie naar AWS worden gepusht.
- Continuous Deployment: Dit is de laatste stap waarbij gevalideerde wijzigingen zonder menselijke tussenkomst direct naar productie gaan. Dit versnelt de levering van patches en nieuwe functies aanzienlijk.

Dit project maakt gebruik van Bitbucket Pipelines om de volledige cyclus van code tot AWS-deployment te automatiseren.

3.4.1. Bitbucket Pipelines

De volledige pipeline-configuratie wordt gedefinieerd in een `bitbucket-pipelines.yml` bestand in de root van het project. Dit betekent dat de automatisering mee evolueert met de applicatiecode

Aangezien Randstad Digital Bitbucket gebruikt als hun primaire Git-repository, biedt Bitbucket Pipelines een geïntegreerde oplossing zonder dat er extra externe servers (zoals Jenkins) beheerd hoeven te worden.

4. Onderzoek

Voor de realisatie van dit project is gewerkt met een demo-omgeving van de webpagina "Kom op Tegen Kanker" (KOTK), aangeleverd door Randstad Digital. De focus van dit onderzoek ligt op het transformeren van deze applicatie naar een schaalbare, cloud-native infrastructuur.

4.1. Dockerizen van de Drupal-applicatie

De eerste cruciale stap in het proces is het containeriseren van de applicatie. Dit garandeert dat de software identiek functioneert in elke omgeving, van lokale ontwikkeling tot de productieomgeving in AWS. Aangezien de KOTK-applicatie gebruikmaakt van Drupal (PHP) voor de backend en Vue.js (Node.js) voor de frontend, is een geavanceerde configuratie van de Docker-images noodzakelijk.

4.1.1. Multi-Stage Build Strategie

Om de efficiëntie van de uiteindelijke image te optimaliseren, is gekozen voor de Multi-Stage build methode. Deze aanpak scheidt de bouwomgeving (build environment) strikt van de runtime-omgeving. Hierdoor worden enkel de noodzakelijke artefacten gekopieerd naar de definitieve image, wat de omvang aanzienlijk beperkt en de beveiliging verhoogt door build-tools en caches te verwijderen.

De bouw van de Docker-image is onderverdeeld in drie fasen:

Fase 1: PHP Builder (Backend)

In deze fase wordt een Composer-image gebruikt om alle PHP-afhankelijkheden te installeren. Hierbij wordt gebruikgemaakt van een Cloudsmith API-key voor authenticatie bij private repositories. De autoloader wordt geoptimaliseerd voor productie door de *--optimize-autoloader* vlag te gebruiken. De relevante mappen zoals *web*, *scripts*, *drush* en *etc* worden hier voorbereid.

Fase 2: Node.js Builder (Frontend Assets)

In deze fase worden de frontend-assets gegenereerd met behulp van *Node.js 24-alpine*. De bibliotheken voor Vue.js en de specifieke thema's (zoals *gin_fizz* en *ocelot_aok)* worden gecompileerd. De resulterende CSS- en JavaScript-bestanden worden klaargezet voor de runtime-fase.

Fase 3: Runtime Image

In de laatste fase worden enkel de noodzakelijke bestanden uit de voorgaande stappen gekopieerd naar een schone PHP-FPM runtime image. Om de veiligheid te waarborgen, wordt de gebruiker gewijzigd naar `www-data`, waardoor de applicatie altijd onder een `non-root` user draait. De applicatie luistert op poort 8080, wat overeenstemt met de configuratie van de toekomstige AWS Load Balancer.

4.1.2. Lokale Validatie via Docker Compose

Na het bouwen van de image is de functionaliteit lokaal getest via een Docker Compose stack. Deze stack simuleert de volledige architectuur door verschillende containers met elkaar te verbinden in een gedeeld netwerk:

- **Web Container**: Bevat de geoptimaliseerde Drupal-omgeving waarbij de map `/files` is gemount voor persistente content.
- **Database Container**: Maakt gebruik van een officiële `MySQL 8.0` image waarbij een database-dump van de KOTK-applicatie automatisch wordt ingeladen bij de eerste opstart.
- **Redis Container**: Wordt ingezet voor caching om de prestaties van de Drupal-applicatie te verbeteren.

- **Solr Container**: Aangezien AWS geen native beheerde dienst heeft voor de specifieke Solr-configuratie van dit project, is Solr opgezet als een sidecar-container. Deze container gebruikt de KOTK-specifieke zoekconfiguratie en is essentieel voor de zoekfunctionaliteit van de website.

Deze lokale stack vormt de basis voor de verdere uitrol naar de AWS-omgeving via Terraform en ECS Fargate.

4.2. Infrastructure as Code (IaC) met Terraform

In dit project wordt de AWS-infrastructuur volledig beheerd via Terraform. Dit laat toe om de omgeving op een declaratieve wijze te definiëren, wat fouten minimaliseert en de reproduceerbaarheid vergroot.

4.2.1. De Terraform Workflow

De kern van de Terraform-werkwijze rust op drie essentiële commando's die de levenscyclus van de infrastructuur beheren:

- `terraform init`: Initialiseert het project en downloadt de benodigde AWS provider-plug-ins.
- `terraform plan`: Voert een "dry run" uit om wijzigingen te bekijken zonder ze daadwerkelijk door te voeren.
- `terraform apply`: Voert het plan uit en creëert de effectieve resources in de AWS-cloud.

4.2.2. Architecturale Structuur: Persistent versus Temporary

Om de implementatiesnelheid te verhogen en de kosten te beheersen, is de infrastructuur opgedeeld in twee logische mappen:

- /persistent: Bevat kritieke resources die niet vernietigd mogen worden, zoals de Virtual Private Cloud(VPC), Relational Database Service(RDS), Elastic File System (EFS) opslag en de Load Balancer. Dit voorkomt dataverlies en zorgt ervoor dat Domain Name Service (DNS) instellingen (Load Balancer endpoint) stabiel blijven.
- /temporary: Bevat vluchtige resources die verwijderd kunnen worden om kosten te besparen, zoals de ECS Cluster, AutoScaling en Redis.

4.2.3. State Management en Samenwerking

Om effectief in teamverband te werken, is een Remote State strategie geïmplementeerd. In plaats van de infrastructuurstatus lokaal op te slaan, wordt gebruikgemaakt van:

- S3 Backend**:** Een versleutelde S3-bucket dient als de *Single Source of Truth* voor de infrastructuurstatus.
- State Locking via DynamoDB: Om corruptie van configuratiebestanden te voorkomen wanneer twee ontwikkelaars tegelijkertijd wijzigingen doorvoeren, wordt een DynamoDB-tabel gebruikt om de 'state' te vergrendelen tijdens uitvoering.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 5.46.0"
    }
  }

  backend "s3" {
    bucket       = "production-drupal-terraform-state"
    key          = "persistent/terraform.tfstate"
    region       = "eu-north-1"
    use_lockfile = true
  }
}
```

4.3. Netwerk en Beveiliging Niveau

4.3.1. VPC

De fundering van de infrastructuur is de VPC. Dit is een logisch geïsoleerd netwerk waarin alle Drupal-gerelateerde resources worden ondergebracht.

```
resource "aws_vpc" "main" {
  cidr_block           = var.vpc_cidr
  instance_tenancy     = "default"
  enable_dns_hostnames = true
  enable_dns_support   = true
}
```

4.3.2. Subnets en Availability Zones

Er is gekozen voor een gelaagde subnet-strategie over meerdere Availability Zones (AZ's) om de beveiliging en beschikbaarheid te optimaliseren:

- **Publieke Subnets**: Bevatten de Internet Gateway (IGW) en de Application Load Balancer (ALB). Deze zijn direct bereikbaar vanaf het internet.

```
resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id
}
```

- **Private Subnets:** Bevatten de ECS Fargate taken en de RDS database. Deze hebben geen directe route vanaf het internet, wat de *attack surface* aanzienlijk verkleint

Om redundantie te garanderen, worden de subnets verspreid over verschillende fysieke locaties (AZ's). Uitgaand verkeer vanuit de private subnets (voor updates) verloopt via een NAT Gateway in de publieke zone.

```
resource "aws_nat_gateway" "main" {
  allocation_id = aws_eip.nat.id
  subnet_id     = aws_subnet.public_a.id
  depends_on = [aws_internet_gateway.main]
}
```

4.3.3. Application Load Balancer (ALB)

De ALB fungeert als het centrale toegangspunt. Het verdeelt inkomend verkeer intelligent over de Drupal-containers in de private subnets.

```
resource "aws_lb" "main" {
  name               = "${var.project_name}-alb"
  load_balancer_type = "application"
  security_groups    = [var.alb_security_group_id]
  subnets            = var.public_subnet_ids
  internal           = false
}
```

Target Groups : De Target Groups voeren continu health checks uit om te garanderen dat verkeer enkel naar gezonde containers wordt gestuurd.

```
resource "aws_lb_target_group" "app" {
  name        = "${var.project_name}-tg"
  port        = var.target_group_port
  protocol    = var.target_group_protocol
  vpc_id      = var.vpc_id
  target_type = var.target_type

  health_check {
    path                = var.health_check_path
    protocol            = var.health_check_protocol
    healthy_threshold   = 2
    unhealthy_threshold = 5
    timeout             = 10
    interval            = 30
    matcher             = var.health_check_matcher
  }
}
```

Listeners : De routering dwingt veiligheid af door HTTP-verkeer (poort 80) direct om te leiden naar HTTPS (poort 443)

De volledige netwerkarchitectuur, inclusief de verdeling over verschillende zones, is weergegeven in Figuur 1.

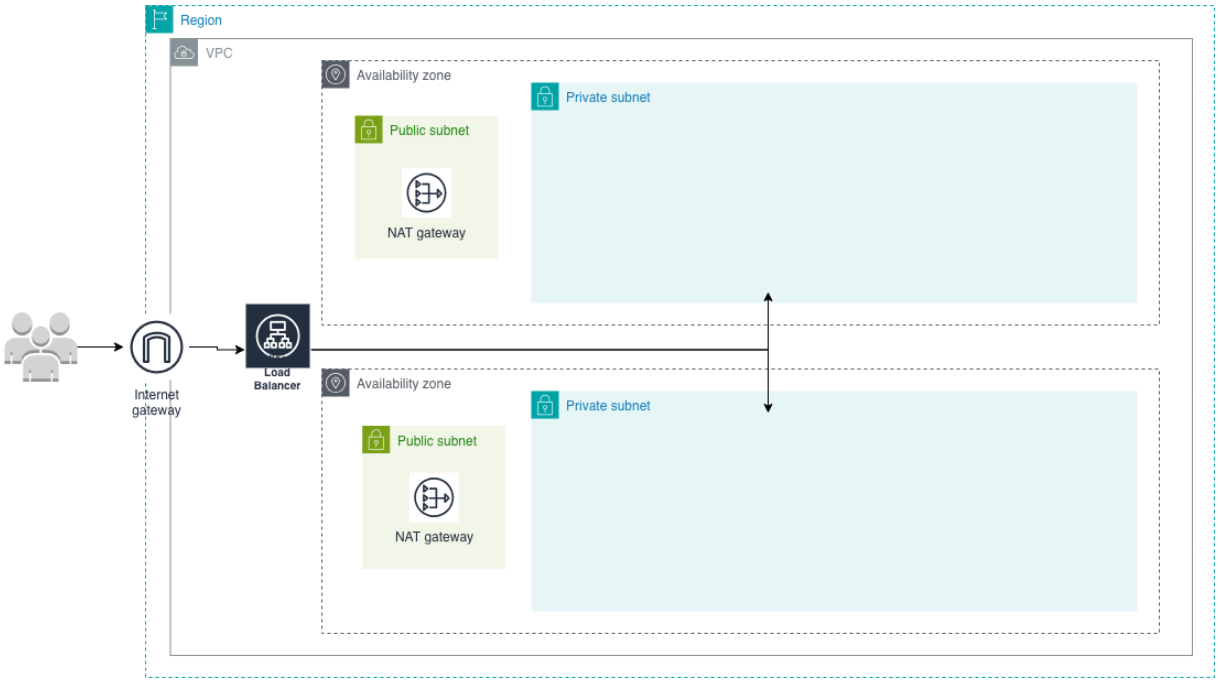

Figure 1 Overzicht van de AWS Netwerkarchitectuur

4.3.4. Security Groups

Beveiligingsgroepen fungeren als virtuele firewalls die het inkomende en uitgaande verkeer controleren. Er is een strikte hiërarchie toegepast:

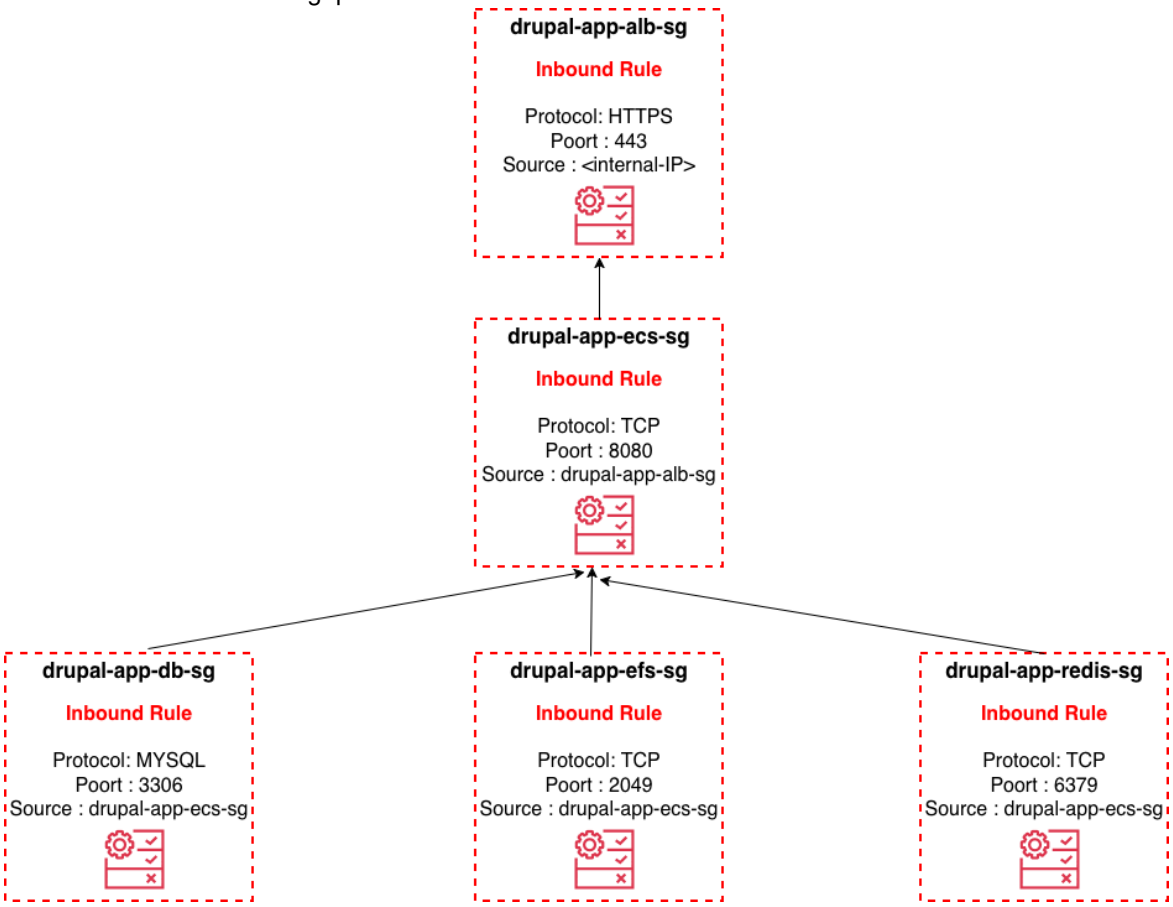

Figure 2 Layered security: ALB, ECS en RDS

De verkeersregels zijn strikt geconfigureerd volgens het principe van *least privilege*:

- **ALB Security Group:** Accepteert enkel HTTP (80) en HTTPS (443) verkeer vanaf de vertrouwde IP-range van het bedrijf.
- **ECS Tasks Security Group:** Laat enkel inkomend verkeer toe dat afkomstig is van de ALB. Uitgaand verkeer is toegestaan voor updates en communicatie met de database.
- **Database & EFS & Redis Security Groups:** Deze laten uitsluitend verkeer toe op hun specifieke poorten (3306 voor MySQL en 2049 voor EFS, 6379 voor Redis) mits dit verkeer afkomstig is van de ECS Security Group.

Door het Stateful karakter van AWS Security Groups hoeven we enkel inkomende regels strikt te definiëren; de respons op een toegestane verbinding wordt automatisch goedgekeurd, wat de configuratie overzichtelijk en veilig houdt.

4.4. De Database Niveau

De database layer vormt het hart van de Drupal-applicatie waar alle kritieke data, van gebruikersinformatie tot contentstructuren, wordt opgeslagen. In dit project is de focus gelegd op een combinatie van beveiliging door isolatie en geautomatiseerd geheimbeheer.

4.4.1. AWS Secrets Manager: Veilig beheer van inloggegevens

Het hardcoderen van wachtwoorden in scripts is een groot beveiligingsrisico. Om dit te voorkomen, is gebruikgemaakt van AWS Secrets Manager. Dit is gecentraliseerde dienst voor het veilig opslaan en beheren van gevoelige informatie zoals database-credentials ve API-keys.

Tijdens het Terraform-proces wordt een wachtwoord gegenereerd met een hoge entropie via de `random_password` resource. Deze gegevens worden vervolgens in JSON-formaat opgeslagen. Dit stelt de applicatie in staat om op een programmatische manier alle benodigde verbindingsdetails — zoals het endpoint, de poort en de gebruikersnaam — op te halen uit één beveiligde bron.

```
resource "aws_secretsmanager_secret" "db_secret" {
  name                    = "${var.project_name}-db-credentials"
  description             = "RDS database credentials for ${var.project_name}"
  recovery_window_in_days = var.recovery_window_in_days
}

resource "aws_secretsmanager_secret_version" "secret" {
  secret_id = aws_secretsmanager_secret.db_secret.id
  secret_string = jsonencode({
    username      = var.db_username
    password      = random_password.password.result
    dbname        = var.db_name
  })
}
```

4.4.2. Amazon RDS (Relational Database Service)

Voor de opslag van de Drupal-data is gekozen voor Amazon RDS met de MySQL-engine. Het gebruik van een managed service zoals RDS biedt het voordeel dat AWS taken zoals hardware-onderhoud, software-

patching en backups automatiseert, waardoor de focus volledig op de ontwikkeling van de applicatie kan blijven.

Veiligheidsmaatregelen voor de database:

Netwerkisolatie: De database is ondergebracht in de private subnets via een aws_db_subnet_group. Door de parameter publicly_accessible op false te zetten, heeft de database geen publiek IP-adres en is deze onbereikbaar vanaf het internet.

Toegangscontrole: De toegang is strikt beperkt via de eerder besproken Security Groups. Enkel verkeer op poort 3306 dat afkomstig is van de ECS-taken wordt toegelaten.

Encryptie: Alle data "at rest" wordt versleuteld om te voldoen aan de enterprise-standaarden voor gegevensbeveiliging.

```
resource "aws_db_instance" "rds_database" {
  identifier              = "${var.project_name}-db"
  allocated_storage       = 20
  storage_type            = "gp3"
  engine                  = var.db_engine
  engine_version          = "8.4"
  instance_class          = "db.t3.micro"
  storage_encrypted       = true

  db_name  = var.db_name
  username = var.db_username
  password = var.db_password # Afkomstig uit Secrets Manager

  db_subnet_group_name   = aws_db_subnet_group.drupal_db_sng.name
  vpc_security_group_ids = [var.db_sg_id]

  publicly_accessible = false
  deletion_protection = var.deletion_protection
}
```

De integratie tussen Secrets Manager en RDS zorgt voor een "zero-trust" benadering waarbij inloggegevens dynamisch worden beheerd en nooit onversleuteld in de broncode of de console verschijnen.
De interactie tussen het geautomatiseerd wachtwoordbeheer en de database-instantie wordt visueel toegelicht in Figuur 3.

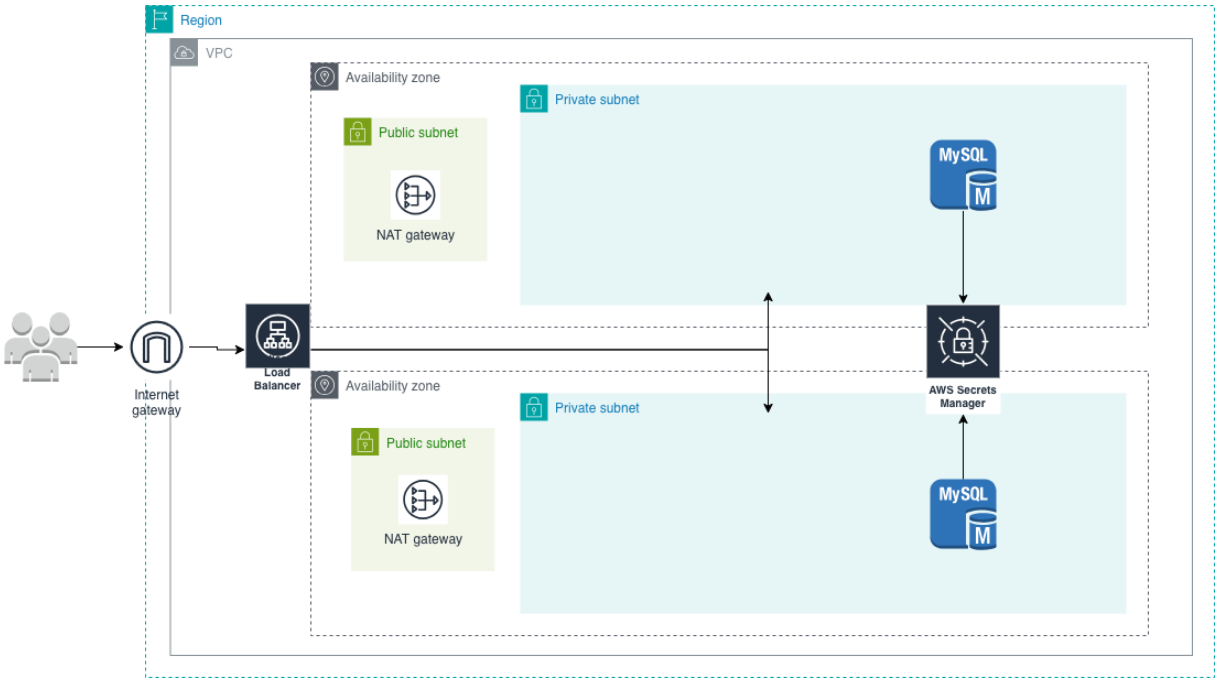

Figure 3 Architectuur van de Database Layer

4.5. De Applicatie Niveau

Nu de database- en netwerkfaciliteiten zijn opgezet, wordt in deze sectie de feitelijke applicatie-omgeving besproken. Hierbij wordt gebruikgemaakt van Amazon Elastic Container Service (ECS) met de AWS Fargate engine om de Drupal-applicatie op een serverloze en schaalbare manier te hosten.

4.5.1. Container Management: ECR en ECS Cluster

Alle gecontaineriseerde images worden opgeslagen in de Amazon Elastic Container Registry (ECR). Deze private registry garandeert dat enkel geautoriseerde AWS-services de images kunnen ophalen.

Voor dit project zijn twee afzonderlijke ECR-repositories geconfigureerd: één voor de Drupal-image en één voor de Solr-image. Deze scheiding maakt onafhankelijk versiebeheer en afzonderlijke beveiligingsscans per component mogelijk. Een cruciale veiligheidsmaatregel is de instelling van immutable image tags: eenmaal gepusht, kan een image-tag nooit meer worden overschreven. Dit garandeert dat elke deployment volledig reproduceerbaar is en dat een rollback naar een vorige stabiele versie altijd mogelijk blijft zonder risico op stille overschrijvingen.

Aanvullend is scan on push geactiveerd op beide repositories. Elke nieuw gepushte image wordt automatisch gescand op bekende kwetsbaarheden (CVE's) in het besturingssysteem en de geïnstalleerde libraries. Deze ingebouwde scan vormt een tweede verdedigingslinie naast de Snyk-containerscans in de CI/CD-pipeline.

De uitvoering vindt plaats binnen een ECS Cluster. Een cruciaal onderdeel van de configuratie is de Capacity Provider Strategy. Door Fargate een gewicht (weight) van 1 te geven, dwingen we Terraform af om altijd serverloze rekenkracht te gebruiken. Dit vereenvoudigt de automatisering en voorkomt onvoorziene kosten door ongebruikte servers.

Tot slot is Amazon CloudWatch Container Insights geactiveerd op het cluster. Container Insights verzamelt gedetailleerde metrische gegevens per container.

4.5.2. Task Definition en IAM Roles

De Task Definition fungeert als de blauwdruk voor de containers. Hierin worden de CPU- en RAM-limieten vastgelegd. Voor dit project is gekozen voor 2 vCPU en 4 GB geheugen: de Solr JVM vereist minimaal ~2 GB heap, terwijl PHP-FPM ~1,5 GB nodig heeft bij piekbelasting. Door beide processen voldoende ruimte te geven, vermijden we CPU-contentieproblemen die anders tot time-outs leiden.

De Task Definition bevat twee containers: Drupal (essential: true) en Solr (essential: false). De markering essential: false voor Solr is een bewuste architecturale keuze: indien de Solr-container crasht, blijft de volledige Drupal-applicatie beschikbaar. Enkel de zoekfunctionaliteit valt tijdelijk weg. ECS zal de taak herstarten zodra Solr niet meer reageert, zonder de gehele service te onderbreken.

Drie IAM-rollen: het principe van least privilege;

- **ecs-task-execution-role** : ECS-infrastructuurlaag: image ophalen uit ECR, logs schrijven naar CloudWatch, credentials lezen uit Secrets Manager
- **ecs-task-role :** Applicatielaag: S3-bestandstoegang, SSM Session Manager voor debugsessies in de container
- **ecs-cron-role :** EventBridge Scheduler: starten van cron-taken, doorgeven van de execution- en task-roles

Deze scheiding volgt het principe van least privilege: de applicatiecode (task role) heeft geen toegang tot de rechten van de execution role. Een gecompromitteerde container kan daardoor geen images overschrijven of secrets beheren.

```
resource "aws_iam_policy" "get_db_credentials" {
  name        = "${var.project_name}-get-db-secrets-policy"
  description = "Toegang voor ECS om database secrets te lezen"

  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action   = ["secretsmanager:GetSecretValue"]
        Effect   = "Allow"
        Resource = [var.secret_arn] # Alleen toegang tot dit specifieke geheim
      }
    ]
  })
}
```

In de Task Definition wordt expliciet onderscheid gemaakt tussen gevoelige en niet-gevoelige configuratiewaarden:

- Secrets (via Secrets Manager): `DB_PASSWORD`, `DB_USER`, `DB_NAME`, `SMTP_USER`, `SMTP_PASSWORD` worden versleuteld opgeslagen en pas tijdens runtime ontsleuteld in het geheugen van de container. Ze verschijnen nooit in de ECS-console of CloudWatch-logs.

- Plaintext omgevingsvariabelen: `REDIS_HOST`, `DB_HOST`, `ALB_DNS_NAME`, `SMTP_HOST` eindpuntadressen zonder beveiligingswaarde die ook zichtbaar mogen zijn.

De EFS-volumes zijn geconfigureerd met `transit_encryption = "ENABLED"`. Dit betekent dat alle data tussen de Fargate-container en het EFS-bestandssysteem via TLS wordt versleuteld, zelfs binnen het VPC-netwerk. Dit voldoet aan de enterprise-beveiligingsstandaarden voor data in transit.

4.5.3. Schaalberheid: AutoScaling Policies

Om in te spelen op wisselende verkeersdrukte, is een Autoscaling Policy geïmplementeerd op basis van Step Scaling. De service schaalt automatisch op (scale-out) wanneer het gemiddelde CPU- of geheugengebruik de drempelwaarde overschrijdt, en schaalt weer af (scale-in) tijdens rustige periodes. De automatische schaalbaarheid en de verdeling van de werklast over de verschillende Availability Zones worden gevisualiseerd in Figuur 4.

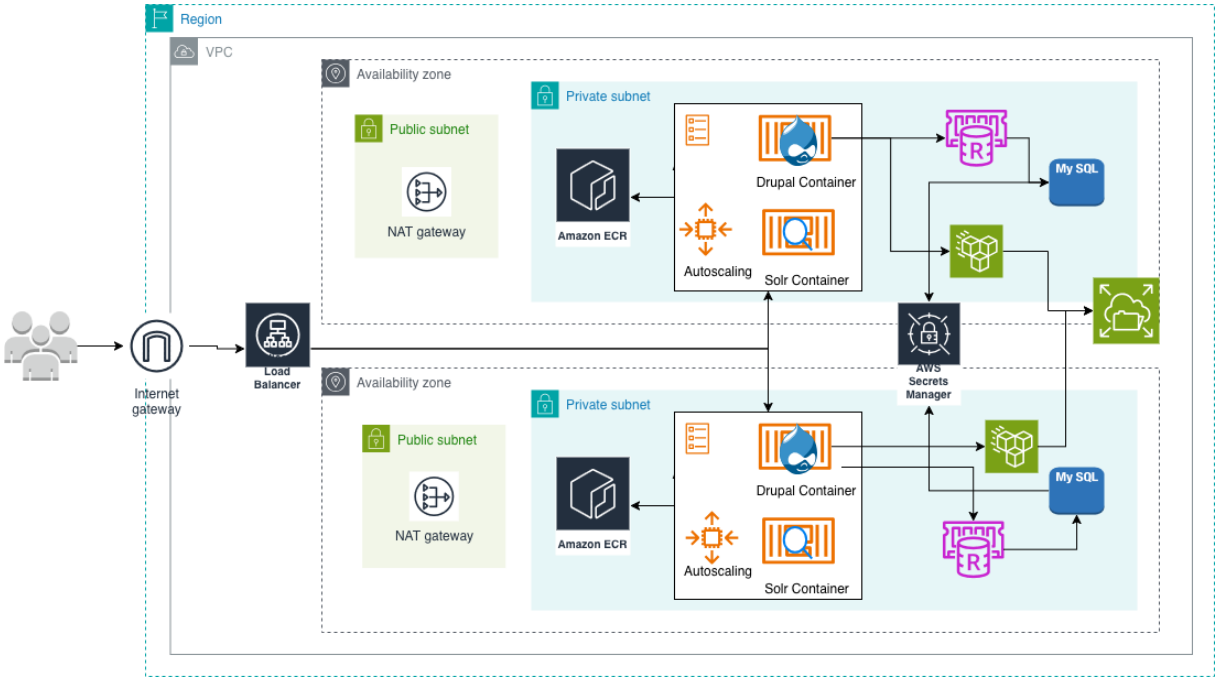

Figure 4 ECS AutoScaling in Multi-AZ infrastructuur

Dit diagram illustreert hoe de Application Load Balancer de inkomende verzoeken verdeelt over de beschikbare ECS Tasks. Wanneer de CPU-drempelwaarde wordt bereikt, start de Auto Scaling Group extra containers op in beide Availability Zones, wat de Drupal-applicatie performant houdt onder zware belasting.

In tegenstelling tot een eenvoudige Target Tracking policy, maakt deze implementatie gebruik van een verfijnd twee-traps mechanisme voor de CPU scale-out:

Table 3 Twee-traps step scaling voor scale-out

CPU-gebruik boven drempel	Actie
0 – 20% boven drempel (60–80%)	+1 taak (gematigde belasting)
> 20% boven drempel (> 80%)	+2 taken (zware belasting)

Dit zorgt voor een snellere reactie bij plotse pieken, zonder bij elke kleine overschrijding direct twee taken op te starten.

4.5.4. Persistente Opslag : Amazon EFS

Aangezien Fargate-containers 'ephemeral' (vluchtig) zijn, gaan alle uploads in Drupal verloren bij bir herstart. Om dit op te lossen, wordt Amazon EFS (Elastic File System) ingezet.

Een uniek aspect van deze implementatie is het gebruik van Access Points. We hebben twee aparte toegangspunten gecreëerd: `/files` voor publieke media en `/private` voor gevoelige configuratiebestanden. Dankzij de `creation_info` in Terraform worden de juiste Linux-permissies (UID 33 voor `www-data`) automatisch toegepast bij de eerste koppeling. De koppeling tussen de vluchtige Fargate-containers en het persistente Amazon EFS-bestandssysteem is gedetailleerd weergegeven in Figuur 5

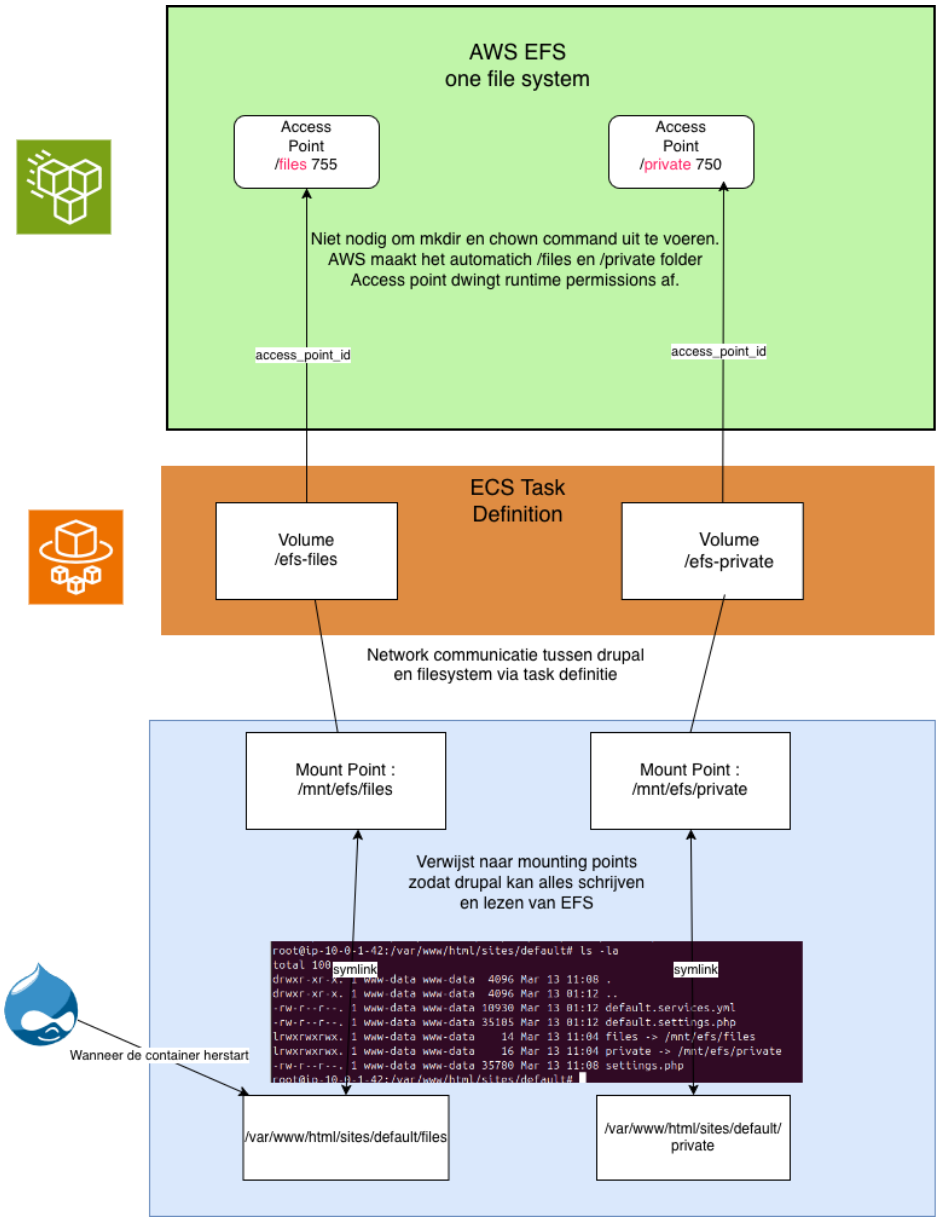

Figure 5 EFS Access Points en Mount Targets

In Figuur 5 is te zien hoe de EFS Mount Targets in elk privaat subnet fungeren als netwerkinterface voor de opslag. De Access Points (`/files` en `/private`) zorgen voor de noodzakelijke isolatie en handhaving van POSIX-permissies. Dit ontwerp stelt de Drupal-applicatie in staat om media-bestanden en configuraties veilig te delen tussen alle actieve container-instances, ongeacht hun levenscyclus.

De SymLink Strategie:
Fargate-containers zijn vluchtig (ephemeral): elke herstart begint vanuit een schone image. Drupal verwacht zijn bestanden op vaste paden zoals `/opt/drupal/web/sites/default/files`, maar EFS is gemount op het generieke pad `/mnt/efs/files`. Een rechtstreekse koppeling via de Task Definition volstaat niet, omdat Drupal de map zelf moet bezitten.

De oplossing is een startup command dat bij elke containeropstart het bestaande Drupal-bestandspad verwijdert en vervangt door een symbolische koppeling naar het EFS-mountpunt:

```
command = [
        "sh",
        "-c",
        "mkdir -p /mnt/efs/private/config-sync && rm -rf
/opt/drupal/web/sites/default/files && ln -s /mnt/efs/files
/opt/drupal/web/sites/default/files && rm -rf /opt/drupal/web/sites/default/private
&& ln -s /mnt/efs/private /opt/drupal/web/sites/default/private && exec apache2-
foreground"
      ]
```

Table 4 Symlink Stappen Uitleggen

Stap	Reden
mkdir -p /mnt/efs/private/config-sync	Verzekert dat de config-sync map op EFS aanwezig is vóór Drupal deze nodig heeft
rm -rf .../files	Verwijdert de lege map uit de image die anders de symlink blokkeert
ln -s /mnt/efs/files .../files	Koppelt Drupal's bestandspad aan het persistente EFS-volume
ln -s /mnt/efs/private .../private	Zelfde koppeling voor het private bestandspad (gevoelige uploads)
/opt/drupal/scripts/solr-reindex.sh &	Start de Solr-herindexering als achtergrondproces (de & zorgt dat Apache niet wacht)
exec apache2-foreground	Start Apache als hoofdproces; ECS bewaakt dit proces voor health checks

Dankzij deze aanpak deelt elke actieve ECS-taak dezelfde bestanden via EFS, ongeacht wanneer de taak gestart is of hoeveel taken gelijktijdig draaien.

4.5.5. Prestatieoptimalisatie: Amazon ElastiCache voor Redis

Om de responstijd van de website te verbeteren en de database-load te verlagen, is Redis geïmplementeerd als caching-laag via Amazon ElastiCache. De instantie draait op een cache.t3.micro node in de private subnets, enkel bereikbaar via poort 6379 vanuit de ECS Security Group.

Geheugenbeheer**:** Via de `allkeys-lru` policy (Least Recently Used) verwijdert Redis automatisch de oudste cache-items wanneer het geheugen vol is, wat "Out of Memory" crashes voorkomt.

```
resource "aws_elasticache_parameter_group" "this" {
```

```
  name   = "${var.project_name}-redis7-params"
  family = "redis7"
  parameter {
    name  = "maxmemory-policy"
    value = "allkeys-lru"
  }
}
```

4.6. Monitoring en Onderhoud

In een serverloze omgeving zoals AWS Fargate is directe toegang tot de container (bijv. via SSH) niet mogelijk. Daarom is het essentieel om robuuste mechanismen te implementeren voor logging, taakautomatisering en communicatie.

4.6.1. CloudWatch en gecentraliseerde Logging

Om de status van de applicatie te bewaken, is gebruikgemaakt van Amazon CloudWatch. Alle systeemuitvoer (`stdout`) en foutmeldingen (`stderr`) van de Drupal-container worden via de `awslogs` driver direct naar een centrale loggroep gestroomlijnd. Dit is cruciaal voor het diagnosticeren van falende health checks of prestatieproblemen.

4.6.2. Serverless Cron Jobs via Amazon EventBridge

In plaats van een continu draaiend proces binnen de container te onderhouden, is gekozen voor een cloud-native "Serverless Cron" benadering met de Amazon EventBridge Scheduler.

- Werking: Op basis van een ingestelde `schedule_expression` activeert EventBridge op vaste tijdstippen een taak.
- Target: De scheduler start een nieuwe, kortstondige Fargate-taak in het private subnet.
- Command Override: Via Terraform wordt het standaard startcommando overschreven om het script `/opt/drupal/scripts/cron/drush-cron.sh` uit te voeren. Dit script voert de noodzakelijke Drupal-onderhoudstaken uit via Drush.

```
resource "aws_scheduler_schedule" "this" {
  name                = "${var.project_name}-cron-schedule"
  schedule_expression = var.cron_schedule_expression

  target {
    arn      = var.ecs_cluster_arn
    role_arn = var.ecs_scheduler_invoke_role_arn

    ecs_parameters {
      task_definition_arn = var.task_definition_arn
      launch_type         = "FARGATE"
    }

    input = jsonencode({
      containerOverrides = [
        {
          name    = "drupal"
          command = ["/bin/sh", "-c", "/opt/drupal/scripts/cron/drush-cron.sh"]
        }
```

```
      ]
    })
  }
}
```

4.6.3. Foutafhandeling en Notificaties via Amazon SES

Een essentieel onderdeel van het onderhoudsproces is de automatische rapportage van systeemgebeurtenissen. Binnen dit project is Amazon SES niet enkel geconfigureerd voor standaard gebruikersinteracties, maar fungeert het ook als een cruciaal instrument voor foutafhandeling (error handling).

Wanneer de EventBridge Scheduler een Drupal cron-job activeert, voert de Fargate-container specifieke onderhoudstaken uit. Indien er tijdens dit proces een kritieke fout optreedt — bijvoorbeeld een mislukte database-update of een falende module-indexering — is het systeem zodanig geconfigureerd dat er direct een notificatie wordt verzonden naar de systeembeheerder.

4.7. Drupal Configuratie

Nadat de cloud-infrastructuur is opgezet via Terraform, moet de Drupal-applicatie worden geconfigureerd om verbinding te maken met de verschillende AWS-services zoals RDS, ElastiCache (Redis), Solr en SES. Dit gebeurt via dynamische configuratiebestanden: `settings.php` en `host-config.php`.

Om een naadloze overgang tussen lokale ontwikkeling en de cloud-infrastructuur te garanderen, is een abstractielaag in de configuratie noodzakelijk. Figuur 6 illustreert hoe de Drupal-applicatie dynamisch de benodigde services (DB, Redis, Solr, SMTP) koppelt, afhankelijk van de runtime-omgeving

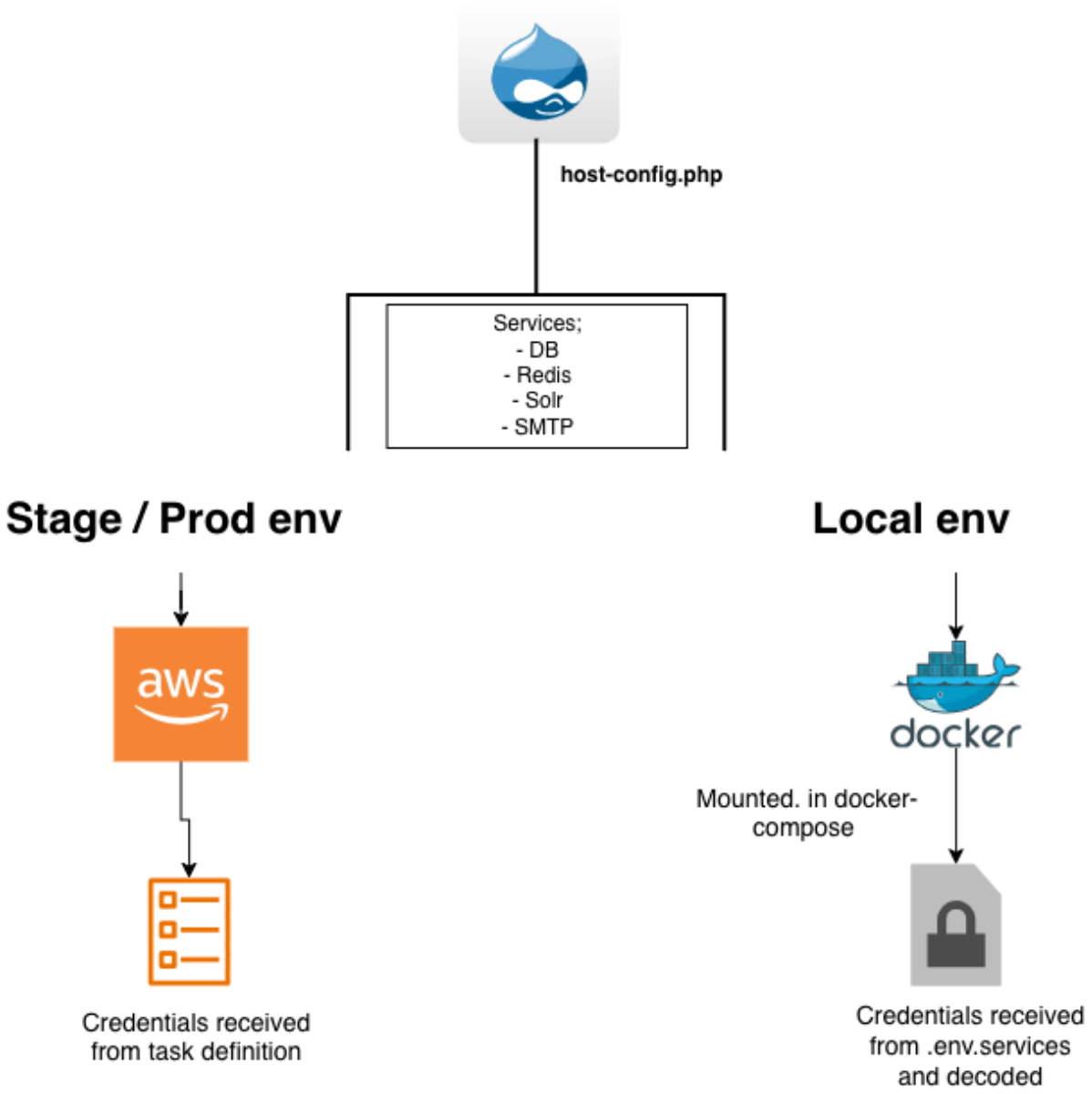

Figure 6 Configuratieflow: Lokaal vs. AWS Cloud

4.7.1. Dynamische Environment Discovery

Een kernaspect van de configuratie is de flexibiliteit tussen verschillende omgevingen (Local, Stage, Prod). In de settings.php is een mechanisme ingebouwd dat zoekt naar een `.env.services` bestand. Dit bestand bevat base64-gecodeerde JSON-data met alle omgevingsvariabelen die door de infrastructuur worden aangeleverd.

```
$candidateEnvFiles = [
  getenv('ENV_SERVICES_FILE'),
  '/var/www/.env.services.aws-stage-kotk-aok',
  // ... andere locaties
];
// Het script decodeert de JSON-payload om database- en cache-instellingen te laden.
```

4.7.2. Database-en Cache-integratie

De database-instellingen worden niet statisch gedefinieerd, maar overgenomen uit omgevingsvariabelen (`DB_HOST`, `DB_USER`, vb.) die veilig vanuit de AWS Secrets Manager in de container worden geïnjecteerd.

Voor de prestaties is de integratie met Redis cruciaal. De configuratie zorgt ervoor dat Drupal de Redis-backend gebruikt voor caching, lock-management en flood-control, mits de PHP-extensie geladen is.

```
if (extension_loaded('redis') && class_exists('Drupal\redis\ClientFactory')) {
  $settings['cache']['default'] = 'cache.backend.redis';
  $settings['redis.connection']['host'] = $instance['host'];
  $settings['redis.connection']['port'] = $instance['port'];
}
```

4.7.3. Beveiliging: Trusted Host Patterns

Om "Host Header Injection" aanvallen te voorkomen, dwingt Drupal Trusted Host Patterns af. In een cloud-omgeving met een ALB is dit complexer, omdat de hostnames dynamisch kunnen zijn. De configuratie in dit project genereert automatisch regex-patronen voor:

- De ALB DNS-naam.
- De specifieke domeinnaam
- Interne VPC IP-ranges voor de AWS health checkers.

4.7.4. Sidecar Solr en SMTP

- **Search API (Solr):** Omdat Solr als een sidecar-container binnen dezelfde Fargate-taak draait, is de configuratie ingesteld om verbinding te maken via localhost op poort 8983.

- **SMTP (Amazon SES):** Zodra de SMTP_USER variabele aanwezig is, schakelt Drupal automatisch over van de standaard PHP-mail naar de SMTPMailSystem. Hierbij worden de SES-endpoints van de regio `eu-north-1` gebruikt.

4.8. Automatisering: CI/CD via Bitbucket Pipelines

De laatste fase van het onderzoek richt zich op de automatisering van het build- en deploymentproces. Hiervoor is gebruikgemaakt van Bitbucket Pipelines. Dit systeem fungeert als de centrale orchestrator die bij elke wijziging in de broncode de noodzakelijke stappen uitvoert om de applicatie veilig en consistent naar AWS te pushen.

4.8.1. DevSecOps: Veiligheid door Snyk-integratie

Beveiliging is een integraal onderdeel van de pipeline (DevSecOps). Voordat er resources worden aangemaakt, voert de pipeline een Snyk IaC Scan uit. Deze scan analyseert de Terraform-scripts op beveiligingsrisico's, zoals te openstaande poorten of onversleutelde databases. Pas wanneer de configuratie voldoet aan de veiligheidseisen, gaat het proces verder.

Daarnaast wordt er na het bouwen van de Docker-images een Snyk Container Scan uitgevoerd op zowel de Drupal- als de Solr-image om kwetsbaarheden in de gebruikte libraries of het base-OS te detecteren.

4.8.2. De Pipeline Flow: Build, Scan en Deploy

De pipeline is opgedeeld in logische stappen die parallel of sequentieel worden uitgevoerd:

- **Infrastructure Initialization:** Terraform initialiseert de backend en bereidt de Amazon ECR (Elastic Container Registry) voor.
- **Docker Build & Push:** De Drupal- en Solr-images worden gebouwd. Hierbij wordt het specifieke build-nummer van Bitbucket (*BITBUCKET_BUILD_NUMBER*) of de Git-tag gebruikt als versiebeheer voor de images.
- **Deployment:** Na goedkeuring van de scans voert Terraform de apply actie uit, waarbij de nieuwe images naar de ECS Fargate clusters worden gepusht.

4.8.3. Release Management via Git Tags

Voor productieomgevingen maakt de pipeline gebruik van Git Tags (bijv. v1.0.0). Dit zorgt voor een strikt releasebeheer:

- **Immutable Tags:** Door images te taggen met versienummers in plaats van een generieke 'latest' tag, is het altijd mogelijk om terug te keren naar een vorige stabiele versie (rollback).
- **Manuele Approval:** In de productie-pipeline is een manual trigger ingebouwd voor de uiteindelijke terraform apply. Dit dient als een menselijke 'gatekeeper' om onbedoelde wijzigingen in kritieke infrastructuur te voorkomen.

5. Resultaten

In dit hoofdstuk worden de resultaten van de geïmplementeerde PoC gepresenteerd. Door middel van connectiviteitstesten en load-tests wordt aangetoond dat de AWS-infrastructuur niet alleen functioneel is, maar ook voldoet aan de gestelde eisen op het gebied van beveiliging en schaalbaarheid.

5.1. Validatie van Netwerkbeveiliging en Database Connectiviteit

De eerste testfase richtte zich op het valideren van de gelaagde beveiliging binnen de VPC. Hierbij is gecontroleerd of de Security Groups effectief verkeer filteren op basis van het least privilege principe.

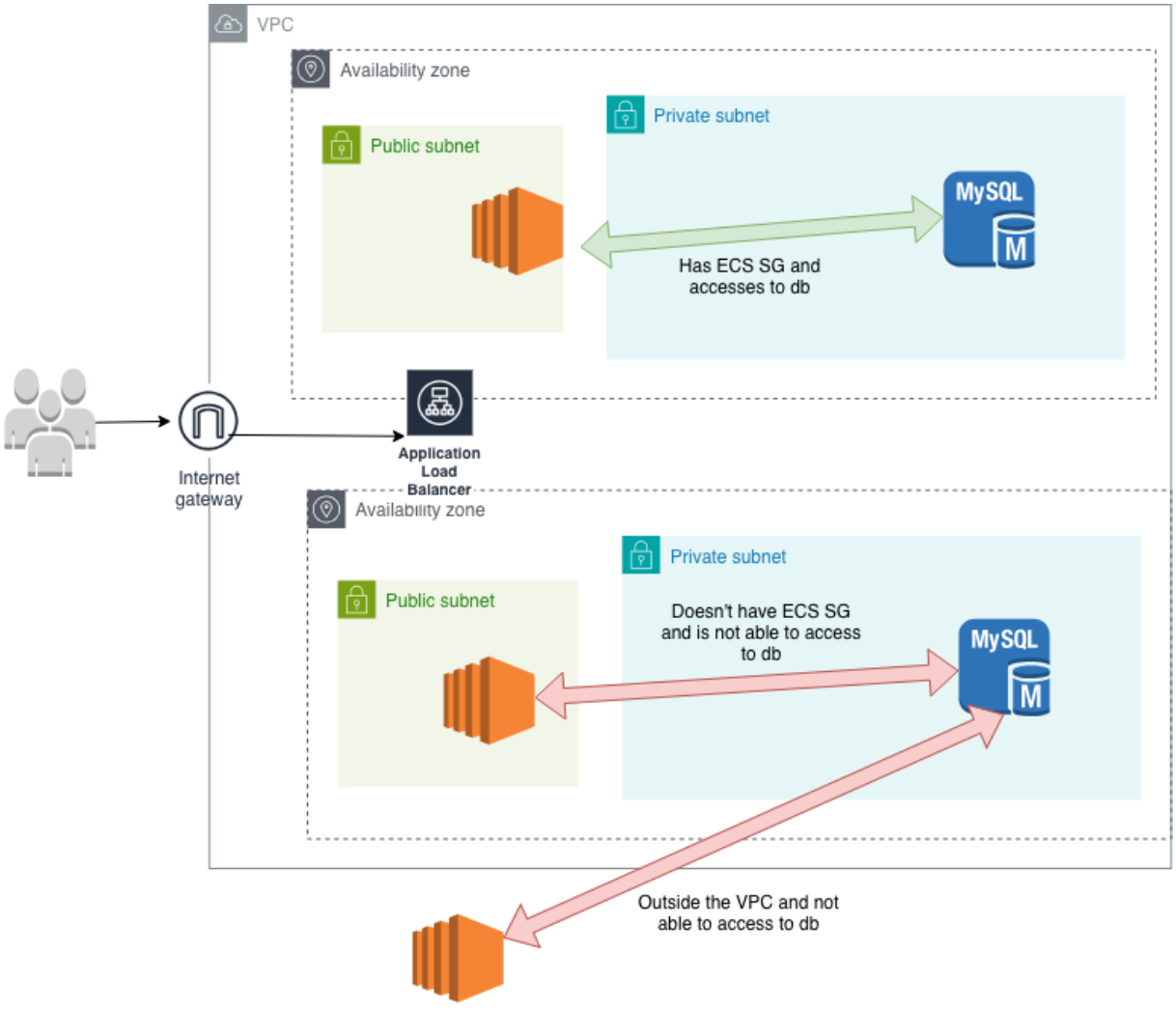

Figure 7 Validatie van netwerkisolatie binnen de VPC

Analyse van Figuur 7:

- Succesvolle verbinding**:** De Drupal-container binnen de *private subnet* kan succesvol communiceren met de MySQL-database omdat deze over de juiste Security Group (ECS SG) beschikt.

- Geblokkeerde toegang**:** Instances in de *public subnet* zonder de juiste SG, evenals alle externe bronnen buiten de VPC, worden onmiddellijk geblokkeerd door de firewall-regels van de database. Dit bewijst dat de data-integriteit gewaarborgd is.

5.2. Schaalbaarheid en Performance (Load Testing)

Om de robuustheid van de ECS Fargate Auto Scaling te testen, is een load-test uitgevoerd met de tool k6. Het scenario drie realistisch gewogen request-groepen die de volledige applicatiestapel belasten:

- Uncached (40%): Elke request wordt voorzien van een unieke cache-buster (?_vu=&_i=), waardoor Drupal telkens een volledige PHP-render uitvoert. Dit maximaliseert de CPU-belasting en triggert het auto-scaling alarm.
- Cached (40%): Dezelfde pagina's worden zonder cache-buster opgevraagd. Na een initiële setup()-warmup worden deze verzoeken bediend vanuit de Redis page cache, wat de prestaties van de caching-laag valideert.
- Search/Solr (20%): Zoekopdrachten via de Solr-backend testen de database-integratie en de zoekindex onder belastin

Het scenario gebruikt 75 gelijktijdige gebruikers (VUs) — ruim binnen het vereiste bereik van 50–100 — verdeeld over vijf fasen over een totale duur van 9 minuten:

```
export const options = {
  stages: [
    { duration: '1m', target: 20 },  // Warmup   – triggert scale-out alarm bij ~60%
CPU
    { duration: '3m', target: 20 },  // Hold laag – tweede taak provisioneert en
koppelt aan ALB
    { duration: '1m', target: 75 },  // Opvoeren  – volledige belasting op 2 gezonde
taken
    { duration: '3m', target: 75 },  // Aanhouden – gespreide belasting op
opgeschaald cluster
    { duration: '1m', target: 0  },  // Afbouwen  – triggert scale-in countdown
  ],
  thresholds: {
    error_rate:            ['rate<0.10'],
    http_req_duration:     ['p(95)<12000'],
    uncached_resp_time_ms: ['p(95)<10000'],
    cached_resp_time_ms:   ['p(95)<3000'],
    search_resp_time_ms:   ['p(95)<15000'],
  },
};
```

Figure 8 Console resultaat van K6 Load Testing

De foutmarge van 5,45% is volledig toe te schrijven aan de scale-out transitieperiode (~2 minuten) waarin de nieuwe taak provisioneert en registreert bij de ALB maar nog geen verzoeken afhandelt. Zodra de tweede taak actief was, stabiliseerde de foutmarge naar nul. De cached responstijd van gemiddeld 771ms tegenover 1,72s voor uncached requests bevestigt dat de Redis page cache correct functioneert en de Drupal-backend effectief ontlast.

Zoals te zien in Figuur 8, overschreed de CPU-utilisatie de drempel van 60% gedurende twee opeenvolgende evaluatieperiodes. Het systeem activeerde de scale-out policy, wat resulteerde in een opschaling van 1 naar 3 actieve taken (zie Figuur 9). In totaal werden 9.381 HTTP-verzoeken verwerkt (17,2 req/s gemiddeld), waarbij 727 MB aan data werd ontvangen.

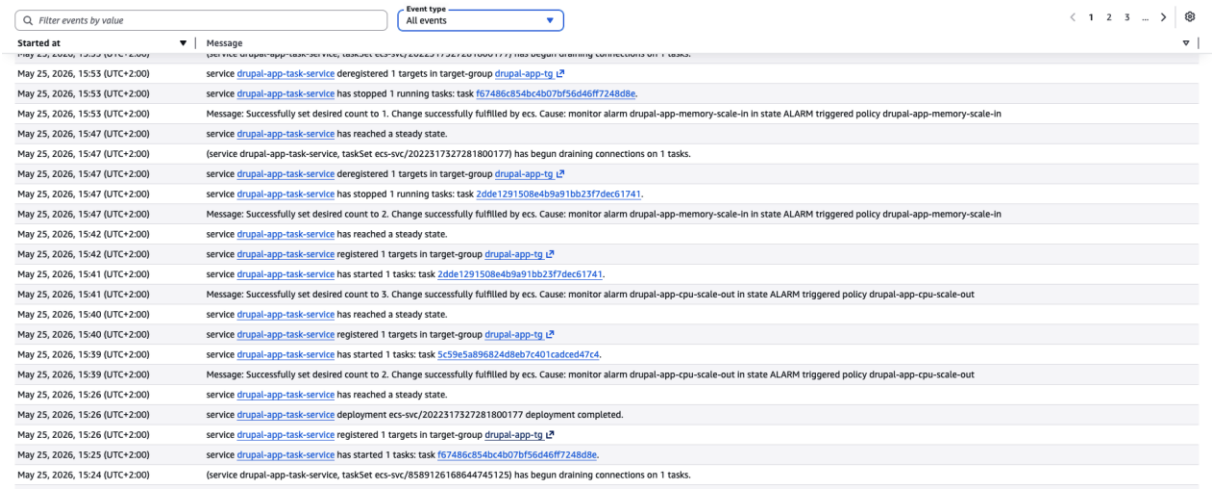

Figure 9 Opschaling naar 3 actieve ECS-instances

5.3. Sidecar Container Integratie : Apache Solr

Een essentieel onderdeel van de Drupal-architectuur is de zoekfunctionaliteit. In de PoC draait Solr als een sidecar-container binnen dezelfde ECS-taak als Drupal.

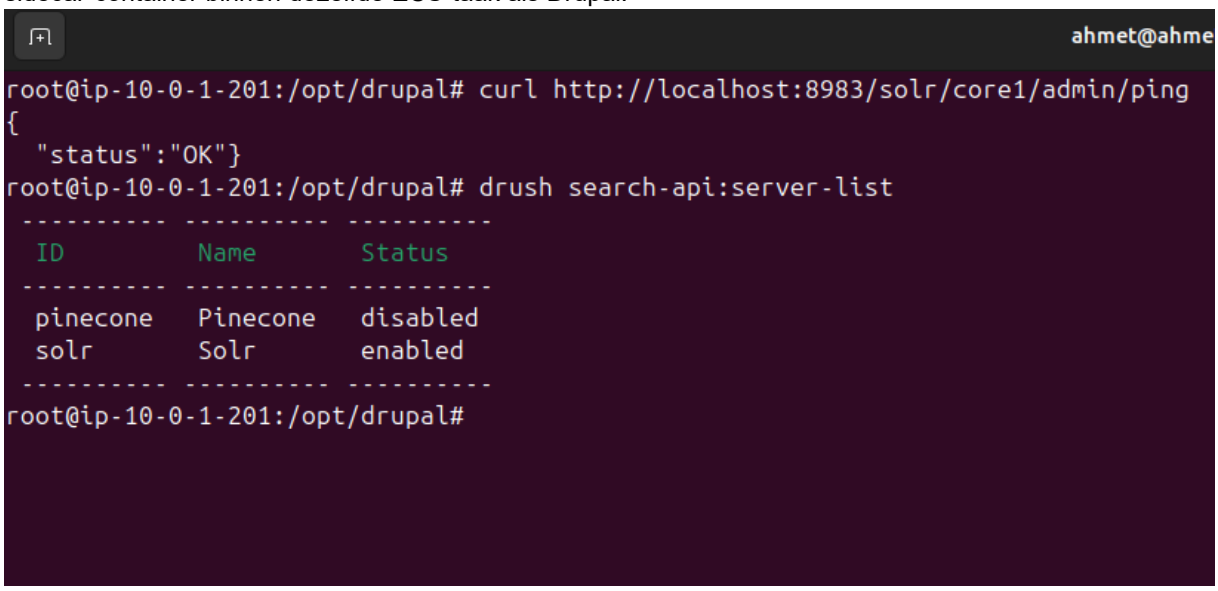

Figure 10 Validatie van Solr-status via Drush

De bovenstaande terminal-output (Figuur 10) bevestigt dat de Drupal-applicatie via *localhost:8983* succesvol kan communiceren met de Solr-backend. Het commando drush *search-api:server-list* toont aan dat de Solr-server de status "enabled" heeft, wat de correcte werking van de sidecar-configuratie bewijst.

5.4. Deployment en Runtime Validatie

De uiteindelijke bevestiging van het project is de succesvolle weergave van de Kom op Tegen Kanker (KOTK) webpagina op het live cloud-domein.

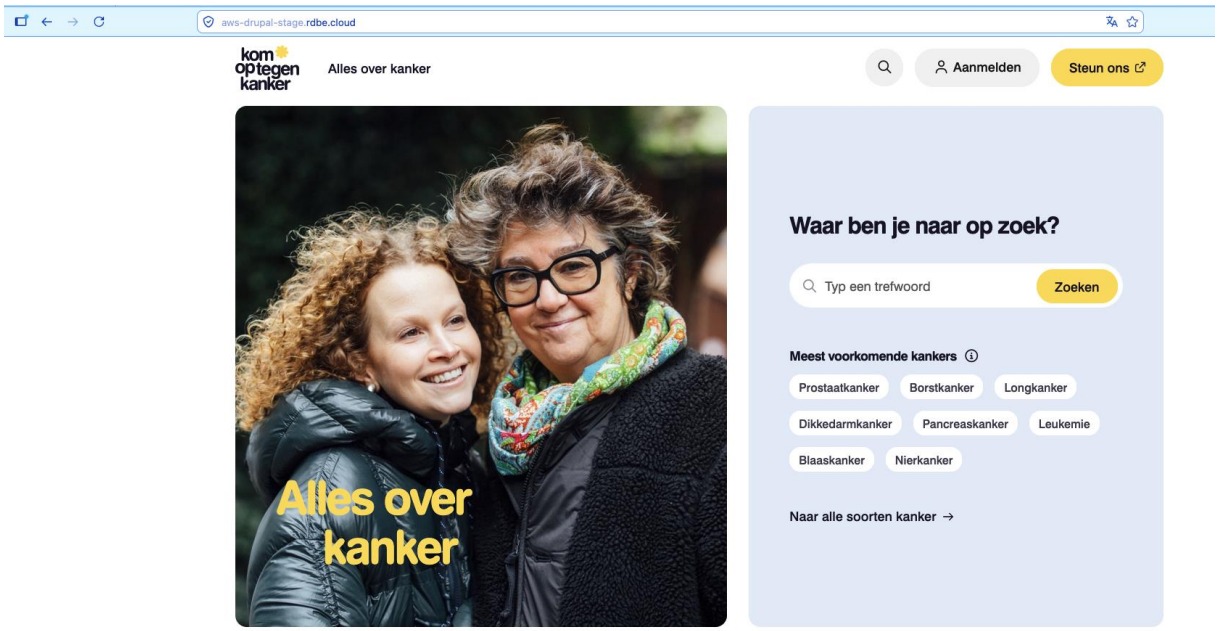

Figure 11 Live weergave van de KOTK-webpagina

6. Interpretatie van de resultaten

De testresultaten tonen aan dat de gerealiseerde AWS-infrastructuur op alle vlakken voldoet aan de vooropgestelde doelstellingen.

De netwerkbeveiligingstesten bevestigen dat het least privilege-principe correct is toegepast. Enkel ECS-taken met de juiste Security Group kunnen de database bereiken, terwijl alle andere toegangspogingen — vanuit publieke subnets of het internet — geblokkeerd worden. Dit betekent dat de gevoelige data van Drupal effectief geïsoleerd is, wat een fundamentele verbetering is ten opzichte van de traditionele on-premise aanpak waarbij netwerkgrenzen minder strikt zijn afgebakend.

De load-test met k6 bevestigt dat het auto-scaling mechanisme correct reageert op verhoogde verkeersdruk. Bij een CPU-gebruik boven 60% gedurende twee opeenvolgende evaluatieperiodes schaalde het systeem automatisch op van 1 naar 3 ECS-instanties, zonder menselijke tussenkomst. De foutmarge bedroeg 5,45% bij 75 gelijktijdige gebruikers — ruim binnen de acceptabele grens van 10%. De foutmarge is uitsluitend toe te schrijven aan de transitieperiode van ~2 minuten tijdens het provisioneren van de nieuwe taak, en niet aan structurele instabiliteit. Dit toont aan dat de architectuur de verkeerspieken die de huidige private servers destabiliseren, probleemloos kan opvangen. Bovendien bevestigen de drie afzonderlijke testscenario's (uncached, cached en search) dat zowel de Redis caching-laag als de Solr zoekintegratie correct functioneren onder belasting.

De sidecar-integratie van Apache Solr bevestigt dat ook niet-standaard AWS-diensten succesvol in een containeromgeving kunnen worden opgenomen, zonder in te boeten op functionaliteit.

7. Besluit/Discussie

Deze bachelorproef onderzocht of een cloud-native Drupal-architectuur op AWS de schaalbaarheidsproblemen, de hoge beheerslast en de trage time-to-market van de huidige on-premise infrastructuur van Randstad Digital kan oplossen. De resultaten van de POC beantwoorden deze vraag bevestigend.

Docker, ECS Fargate en Terraform vormen samen een robuuste basis voor een reproduceerbare, schaalbare en veilige Drupal-omgeving. De geautomatiseerde CI/CD-pipeline elimineert handmatige stappen in het deploymentproces, wat de kans op menselijke fouten verkleint en de leveringssnelheid verhoogt. De live werking van de KOTK-applicatie op het AWS-clouddomein bewijst dat alle componenten correct samenwerken als één geïntegreerd systeem.

De conclusie is dan ook dat de cloud-native aanpak niet alleen de technische pijnpunten oplost, maar ook een schaalbaar en gedocumenteerd model biedt dat Randstad Digital direct kan toepassen voor toekomstige Drupal-projecten.

Voor verder onderzoek zijn er enkele interessante pistes. Een uitbreiding met een Web Application Firewall (WAF) . Daarnaast zou een gedetailleerde kostenanalyse via AWS Cost Explorer, gecombineerd met Reserved Instances voor de database, de financiële haalbaarheid op lange termijn in kaart kunnen brengen.

8. Literatuurlijst

Alkhatib, A., Shaheen, A., & Albustanji, R. N. (2025). A Comparative Analysis of Cloud Computing Services: AWS, Azure, and GCP. International Journal of Computing and Digital System, 18(1), 1–15. https://doi.org/10.12785/ijcds/1571111846

Amazon Web Services. (n.d.-a). Amazon CloudWatch: Component configuration examples for ECS. Geraadpleegd op 15 mei 2026, van https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/component-configuration-examples-ecs-task.html

Amazon Web Services. (n.d.-b). Amazon ECS: CloudWatch metrics. Geraadpleegd op 15 mei 2026, van https://docs.aws.amazon.com/AmazonECS/latest/developerguide/cloudwatch-metrics.html

Amazon Web Services. (n.d.-c). Amazon ECS: Task definition parameters. Geraadpleegd op 15 mei 2026, van https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definition_parameters.html

Amazon Web Services. (n.d.-d). Amazon ECS: Task execution IAM role. Geraadpleegd op 15 mei 2026, van https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_execution_IAM_role.html

Amazon Web Services. (n.d.-e). Amazon VPC: What is Amazon VPC?. Geraadpleegd op 15 mei 2026, van https://docs.aws.amazon.com/AmazonECS/latest/developerguide/security-network.html

Amazon Web Services. (n.d.-f). Connect to the internet using an internet gateway. Geraadpleegd op 15 mei 2026, van https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html

Amazon Web Services. (n.d.-g). Control traffic to resources using security groups. Geraadpleegd op 15 mei 2026, van https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-groups.html

Amazon Web Services. (n.d.-h). Pass sensitive data to a container. Geraadpleegd op 15 mei 2026, van https://docs.aws.amazon.com/AmazonECS/latest/developerguide/specifying-sensitive-data-tutorial.html

Amazon Web Services. (n.d.-i). Security in Amazon Elastic Container Service. Geraadpleegd op 15 mei 2026, van https://docs.aws.amazon.com/AmazonECS/latest/developerguide/security.html

Amazon Web Services. (n.d.-j). Use Secrets Manager secrets in Amazon ECS environment variables. Geraadpleegd op 15 mei 2026, van https://docs.aws.amazon.com/AmazonECS/latest/developerguide/secrets-envvar-secrets-manager.html

Binary Works, The. (2025). Is Drupal Dying in 2025? Why Enterprises Still Choose It Over Alternatives. The Binary Works Blog. Geraadpleegd op 4 mei 2026, van https://www.thebinaryworks.com/blog/drupal-dying-2025-why-enterprises-still-choose-it-over-alternatives/

Docker. (2024, 20 maart). Best practices for Dockerfile. Docker Documentation. Geraadpleegd op 5 mei 2026, van https://docs.docker.com/build/building/best-practices/

Docker Library. (n.d.). Official Repository for Drupal Docker Images. GitHub. Geraadpleegd op 5 mei 2026, van https://github.com/docker-library/drupal

Drupalize.me. (2024, 30 mei). Dockerize an Existing Project. Geraadpleegd op 5 mei 2026, van https://drupalize.me/tutorial/dockerize-existing-project

ElectroIQ. (2024). Drupal Statistics: Market Share, Usage, and Trends. ElectroIQ. Geraadpleegd op 4 mei 2026, van https://electroiq.com/stats/drupal-statistics/

Gbenga, O. (2023, 8 november). Deploying a Dockerized Web Application with AWS ECS and Fargate. Dev.to. Geraadpleegd op 5 mei 2026, van https://dev.to/gbenga700/deploying-a-dockerized-web-application-with-aws-ecs-and-fargate-29bb

Gupta, B., Mittal, P., & Mufti, T. (2021). A review on Amazon Web Service (AWS), Microsoft Azure & Google Cloud Platform (GCP) services. Proceedings of the 2nd International Conference on ICT for Digital, Smart, and Sustainable Development, ICIDSSD 2020 (Vol. 9). EAI. Geraadpleegd op 5 mei 2026, van https://eudl.eu/pdf/10.4108/eai.27-2-2020.2303255

Hashicorp. (n.d.). Terraform Language: S3 Backend. Geraadpleegd op 15 mei 2026, van https://developer.hashicorp.com/terraform/language/backend/s3

Hayward, L. (2018, 2 januari). Terraform and AWS Application Load Balancers. Medium. Geraadpleegd op 15 mei 2026, van https://medium.com/@leigh.hayward1/terraform-and-aws-application-load-balancers-62a6f8592bcf

Lorenc, P., & Woda, M. (2017). IaaS vs. Traditional Hosting for Web Applications - Cost Effectiveness Analysis for a Local Market. In W. Zamojski et al. (Eds.), Advances in Dependability Engineering of Complex Systems (Vol. 582, pp. 233–243). Springer. https://doi.org/10.1007/978-3-319-59415-6_23

Mustafa, O. (2023). A Complete Guide to DevOps with AWS: Deploy, Build, and Scale Services with AWS Tools and Techniques (1st ed.). Apress. https://doi.org/10.1007/9781484293034

Nawazdhandala. (2026a, 12 februari). How to Manage Secrets Manager Secrets with Terraform. OneUptime. Geraadpleegd op 15 mei 2026, van https://oneuptime.com/blog/post/2026-02-12-manage-secrets-manager-secrets-terraform/view

Nawazdhandala. (2026b, 12 februari). How to Mount Amazon EFS in ECS Fargate Tasks. OneUptime. Geraadpleegd op 15 mei 2026, van https://oneuptime.com/blog/post/2026-02-12-mount-efs-ecs-fargate-tasks/view

Open Source For You. (2020, 2 juli). The Benefits of Using Terraform as a Tool for Infrastructure as Code (IaC). Open Source For You. Geraadpleegd op 4 mei 2026, van https://www.opensourceforu.com/2020/07/the-benefits-of-using-terraform-as-a-tool-for-infrastructure-as-code-iac/

Pius, K. (2026, 5 januari). Drupal on AWS: Architecture Pitfalls & Best Practices (2026 Guide). DevPanel. Geraadpleegd op 5 mei 2026, van https://www.devpanel.com/blog/drupal-on-aws-architecture-pitfalls-best-practices-2026-guide/

Scott, J. (2024, 21 maart). 11 Years of Docker: Shaping the Next Decade of Development. Docker Blog. Geraadpleegd op 5 mei 2026, van https://www.docker.com/blog/docker-11-year-anniversary/

Shivkumar, R., Ananth, V., & Prem, N. (2021, 9 juli). Deploy serverless Drupal applications using AWS Fargate and Amazon EFS. AWS Storage Blog. Geraadpleegd op 5 mei 2026, van https://aws.amazon.com/blogs/storage/deploy-serverless-drupal-applications-using-aws-fargate-and-amazon-efs/

Vadim, V. (2024, 7 maart). Drupal on AWS. Medium. Geraadpleegd op 5 mei 2026, van https://medium.com/@novosibcool/drupal-on-aws-4f89b9bdd27b

9. Bijlagen

CONTACT

Ahmet Nuri Uygun | Student
r0937207@student.thomasmore.be

VOLG ONS

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

THOMAS MORE