**De Cloud Resume Challenge**

Hi! Mijn naam is Tristan Graaff. Ik ben op dit moment werkzaam als Test Automation Engineer en doe dit werk met plezier. De laatste tijd ben ik echter steeds meer geïnteresseerd geraakt in cloud computing en tech-infrastructuur en overweeg ik om een carrièreswitch te maken naar Cloud Engineer. Wat lijkt me nou zo leuk aan het werk als Cloud Engineer? Dat kan ik met alle liefde uitleggen.

* Van nature ben ik een jack-of-all-trades. Ik heb veel interesses en haal er voldoening uit met divers bezig te zijn. Het werk van een Cloud Engineer bestaat uit vele facetten: operating systems, networking, security, infrastructure as code, databases, pipelines, en zo voort.. Hier kan ik al mijn IT-interesses in kwijt, de ene dag ben ik misschien wat IaC aan het coderen, terwijl ik de andere dag werk aan de netwerkbeveiliging.
* State-of-the-art technologie. Cloud computing is een van de ‘hottest’ IT-vakgebieden. Je komt te werken met de nieuwste technologieën en staat aan het voorfront van de nieuwste ontwikkelingen binnen de IT, en al helemaal als je bijvoorbeeld AI-modellen gaat trainen en beheren in de cloud.
* Carrièreperspectief. Cloud computing is op dit moment een van de meest gevraagde vakgebieden in de IT en in de toekomst zullen alleen maar meer bedrijven cloud-native gaan werken. Er is veel werk en er zijn veel groeimogelijkheden te vinden in deze sector, zelfs meer dan in andere IT-vakgebieden, daar ben ik van overtuigd.

Maar hoe kom je er na wat te lezen en wat video’s te kijken achter of een bepaald onderwerp je echt ligt? Door het te doen natuurlijk. Ik was op zoek naar doe-het-zelf projecten om over de cloud te leren, en kwam toen de Cloud Resume Challenge tegen. Ik heb besloten om deze aan te gaan om te kijken hoe ik het werken in de cloud zelf ervaar, en om te kunnen laten zien wat ik kan, mocht ik door willen gaan met mijn reis tot Cloud Engineer. Ik heb besloten te gaan werken met AWS, omdat dit wereldwijd de grootste cloud provider is, en omdat de look en feel mij het meest bevalt van de 3 grote providers (AWS, Azure en GCP). Het belangrijkst is om aan de slag te gaan en te beginnen met leren. Uiteindelijk zijn deze platformen slechts tools om problemen in de echte wereld op te lossen. Zie hier meer info over de Cloud Resume Challenge: <https://cloudresumechallenge.dev/docs/the-challenge/aws/>. De komende tijd zal ik documenteren wat ik heb geleerd en dat op mijn blog posten.

Tot snel!

**Het veilig hosten van een statische website**

De Cloud Resume Challenge (<https://cloudresumechallenge.dev/docs/the-challenge/aws/>) bestaat uit 16 beschreven stappen, maar eigenlijk zie ik hem in drie delen.

1. Het hosten van de website
2. Het toevoegen van dynamische functionaliteit aan de website
3. Het implementeren van DevOps principes zoals Infrastructure as Code en CI/CD pipelines

In deze blogpost laat ik zien hoe ik het hosten van de website heb gerealiseerd.

**Gebruikte services:**

* **AWS S3**
* **AWS CloudFront**
* **AWS Route 53**
* **AWS Certificate Manager (ACM)**

**Static website hosting met Amazon S3 buckets**

Ik ben begonnen met het maken van een html pagina (die waar je nu op kijkt) en heb hier wat css voor de styling aan toegevoegd. Vervolgens heb ik 3 nieuwe buckets in Amazon S3 aangemaakt. Zoals je kan zien heb ik de regio eu-central-1 gekozen. Dit is de regio waar mijn potentiële websitebezoekers zicht het dichts bij bevinden.

A screenshot of a computer

AI-generated content may be incorrect.

We uploaden de content van de website naar de tristantech.org bucket. Dit kan met de AWS console, maar als je de AWS CLI hebt geïnstalleerd, kan dit ook met het volgende commando: *aws s3 cp <file-path> s3://tristantech.org.* Na het uitvoeren van dit commando zijn de upgeloade files zichtbaar in de bucket.

A screenshot of a computer

AI-generated content may be incorrect.

In de tristantech.org bucket heb ik de volgende instellingen aangepast:

* *Static website hosting* is voor nu **enabled** met de volgende instellingen. De homepage van de website wordt de index.html file, en we voegen ook een error document toe, mocht de website niet correct kunnen laden.

A screenshot of a computer

AI-generated content may be incorrect.

* *Block all public access* is voor nu **disabled**. Let op, dit is geen best practise om te doen en is enkel tijdelijk om te controleren of het static website endpoint bereikbaar is.

A close-up of a text

AI-generated content may be incorrect.

* Ik voeg een bucket policy to die alle public read access toestaat. Dit is ook geen best practise en is wederom enkel om te controleren of de website correct geladen kan worden.

A computer screen shot of a program code

AI-generated content may be incorrect.

Nu de relevante inrichting is gedaan, navigeer ik naar *Properties -> Static website hosting* (helemaal onderaan de pagina). Het endpoint van mijn website is hier te zien.

A screenshot of a computer

AI-generated content may be incorrect.

Als ik nu op het endpoint klik, wordt mijn website geladen. Succes!

A screenshot of a computer

AI-generated content may be incorrect.

Hiermee krijgt CloudFront expliciet wel toegang tot de bucket en zal de website dus alleen via deze route kunnen worden bezocht. Daarnaast is alle toegang verboden tot de folder */other*. Hier heb ik andere delen van mijn website staan die ik nog niet met het publiek wil delen.

Nu de volgende bucket. De www.tristantech.org bucket blijft leeg en de *Block all public access* setting heb ik hier op **enabled** staan.

A screenshot of a computer

AI-generated content may be incorrect.

A close-up of a computer screen

AI-generated content may be incorrect.

De reden hiervoor is dat deze enkel moet redirecten naar de tristantech.org bucket, waar mijn content staat. Dit kan worden ingesteld onder *Properties -> Static website hosting.* Ik voer bij de Host name het endpoint van de tristantech.org bucket in en kies voor het http protocol en sla mijn wijzigingen op.  
  
A screenshot of a computer

AI-generated content may be incorrect.

Als ik nu onder het kopje *Static website hosting* op de endpoint van mijn www.tristantech.org bucket klik, wordt ik doorverbonden naar het endpoint van mijn tristantech.org bucket. Succes!

A screenshot of a computer

AI-generated content may be incorrect.

Nu nog kort naar de laatste bucket, logs.tristantech.org. Hier is enkel een *logs* folder aanwezig.

A screenshot of a computer

AI-generated content may be incorrect.

Deze folder wordt automatisch toegevoegd wanneer in de tristantech.org bucket de instelling *Server access logging* op **enabled** wordt gezet en de logs.tristantech.org bucket wordt gekozen om de logging weg te schrijven.

A screenshot of a computer

AI-generated content may be incorrect.

Nu ik de endpoints van mijn twee buckets werkend heb gemaakt en logging heb weggeschreven in de derde, is het tijd voor wat *best practises* voordat ik verder ga. In mijn tristantech.org bucket pas ik de volgende instellingen aan:

* *Block all public access* ze tik nu op **enabled**. Ik wil vanuit security-overwegingen dat mijn website enkel beschikbaar wordt gesteld via mijn CloudFront distributie (daar zo meer over). Het endpoint van mijn S3 bucket mag niet publiekelijk te bereiken zijn.

A close-up of a text

AI-generated content may be incorrect.

* Het *PublicReadGetObject* bucket policy, dat ik eerder heb toegevoegd, verwijder ik dan ook vanuit de bovestaande overweging. Er is op dit moment geen bucket policy ingericht.

A screenshot of a computer

AI-generated content may be incorrect.

Als ik nu naar mijn bucket website endpoint navigeer, krijg ik een 403 melding aangezien public access niet meer is toegestaan. Dit is dus naar verwachting.

A screenshot of a computer

AI-generated content may be incorrect.

* Aangezien ik met CloudFront ga werken, is het niet meer nodig om met een bucket website endpoint te werken. CloudFront communiceert met mijn S3 bucket door middel van een REST API, en niet door middel van het website endpoint. *Static website hosting* zet ik dan ook op **disabled**.

A close-up of a computer screen

AI-generated content may be incorrect.

Als ik nu naar het website endpoint navigeer, krijg ik een 404 melding, omdat het website endpoint niet meer bestaat. Dit is dus ook naar verwachting.

A screenshot of a computer

AI-generated content may be incorrect.

Nu staat alles klaar om door te gaan met het volgende onderdeel.

**Het hosten van onze statische website op een snelle, veilige en schaalbare manier**

**Gebruikte services:**

* **AWS S3**
* **AWS CloudFront**
* **AWS Route 53**
* **AWS Certificate Manager (ACM)**

Nu de S3 buckets zijn aangemaakt en ingericht, wordt het tijd om te gaan nadenken over een stukje architectuur. Ik ga namelijk meerdere AWS services met elkaar laten communiceren. Laten we eerst een high-level overview maken zodat duidelijk wordt welke service met welke communiceert en waarom.

A diagram of a diagram

AI-generated content may be incorrect.

Helemaal links zien we de user, oftewel de websitebezoeker. Deze zal het adres van de website intoetsen, tristantech.org, en verwacht vervolgens mijn website te zien. Dit is de route die wordt afgelegd, van user naar de content in de S3 bucket:

1. De user toetst een website in. Deze moet worden omgezet naar een IP-adres. AWS Route 53 is een DNS-service waarmee ik de domeinnaam koppel aan mijn CloudFront-distributie. Er worden DNS-servers geraadpleegd om het IP-adres te vinden dat bij het domein hoort.
2. Wanneer het IP-adres is gevonden, wordt het verzoek van de gebruiker doorgestuurd naar Amazon CloudFront, een content delivery network. CloudFront heeft zogeheten *edge locations* over de hele wereld, waardoor het content kan leveren vanaf een locatie die het dichtst bij de gebruiker ligt. Kopieën van de websitecontent worden in cache opgeslagen om deze zo snel mogelijk te kunnen leveren aan de eindgebruiker.
3. Gezien ik een veilige verbinding willen opstellen, wil ik gebruik maken van HTTPS zodat het verkeer tussen de webbrowser van de gebruiker en onze website versleuteld is. Hiervoor is een SSL/TLS certificaat nodig dat kan worden uitgegeven en beheerd door Amazon Certificate Manager (ACM). Wanneer we een geldig certificaat is opgevraagd en gekoppeld, kan HTTPS worden ingeschakeld in CloudFront.
4. Uiteindelijk haalt CloudFront de statische content van de website op uit de S3 bucket die ik in de vorige blogpost heb aangemaakt en waar ik mijn websitecontent heb opgeslagen. De S3 bucket zal zo worden ingericht dat enkel mijn CloudFront distributie hier toegang toe heeft. CloudFront zal bij ieder nieuw websitebezoek de content eerst uit de cache proberen te laden, en indien de content hier niet aanwezig is, dan zal de bucket worden geraadpleegd.

Laten we nu de services gaan opzetten en inrichten.

**Het registreren van een domein en het aanmaken van een hosted zone in Route 53**

De eerste stap is om een domein aan te schaffen, als je dit nog niet bezit. Dit kan via Amazon, maar ook via andere aanbieders zoals GoDaddy. Ik heb mijn domein via Amazon gekocht. Zie hieronder hoe je een domein aanschaft (ik gebruik hier een voorbeeldnaam omdat ik tristantech.org al heb gekocht).

A screenshot of a computer

AI-generated content may be incorrect.

Na het aanschaffen zal het gekozen domein in *Route 53 -> Registered domains*  verschijnen.

A screenshot of a computer

AI-generated content may be incorrect.

Vervolgens heb ik in *Route 53 -> Hosted zones -> Create hosted zone* een nieuwe hosted zone gecreëerd. In een hosted zone kunnen meerdere records worden beheerd die onder hetzelfde domein vallen, hier zal ik zo op terugkomen om dit in praktijk te laten zien hoe dit werkt.

A screenshot of a computer

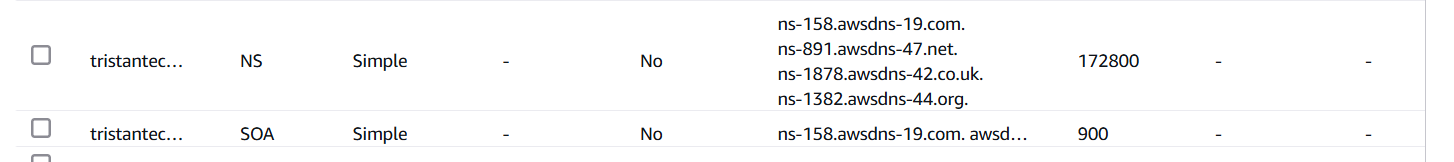
AI-generated content may be incorrect.

De hosted zone is nu zichtbaar onder het overzicht *Hosted zones*.

A screenshot of a computer

AI-generated content may be incorrect.

Bij het openen van de aangemaakte hosted zone, zijn er twee records automatisch aangemaakt.



Het name server (NS) record bevat 4 name servers. Dit zijn de servers die de DNS-queries naar mijn domein afhandelen. Deze name servers moeten gelijk zijn aan de name servers in het geregistreerde domein. Route 53 zal in de eerste instantie dezelfde name servers toewijzen, maar mocht je ze ooit moeten wijzigen, dan kan je dit bij *Registered domains* doen (dus niet in de hosted zone zelf).

A close-up of a computer screen

AI-generated content may be incorrect.

Dan is er nog het start of authority (SOA) record. Hierin staat belangrijke informatie over de hosted zone, zoals de primaire name server en het emailadres van de administrator van het domein. Zowel de gegevens van de NS als de SOA zijn publiek beschikbaar. Met een simpel PowerShell command kan ik ze opvragen.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Het aanvragen van een certificaat in ACM**

Nu is het tijd om een certificaat aan te vragen in de AWS Certificate Manager (ACM). Let op! Ik werk vanuit de AWS region eu-central-1. Het certificaat moet echter vanuit us-east-1 worden aangevraagd, anders zal het niet worden goedgekeurd.

A screenshot of a computer

AI-generated content may be incorrect.

Ik vraag een public certificate aan.

A screenshot of a computer

AI-generated content may be incorrect.

Ik voer mijn domeinnaam in, en daarnaast voeg ik ook twee subdomeinen toe, en kies verder voor de standaardinstellingen. Dat betekent dat we ons certificaat gaan valideren door middel van de DNS.

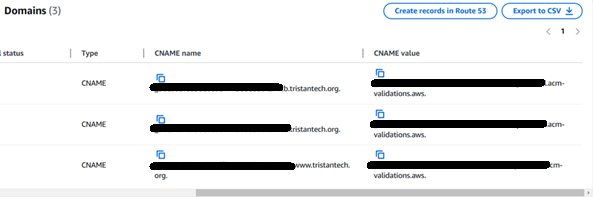


Nu het certificaat is aangemaakt, kan ik het openen, maar is de status nog *pending*. (Hieronder is een voorbeeldcertificaat te zien, niet het werkelijke dat ik gebruik).

A screenshot of a computer

AI-generated content may be incorrect.

In het certificaat staan onder *Domains* mijn domeinen met een CNAME name en een CNAME value. Deze waarden heb ik zo nodig in de *Route 53 Hosted zone* om het certificaat te valideren, dit is de DNS validatie. Zoals je ziet zijn twee CNAME names en values gelijk aan elkaar. Dit is omdat dat domeinen tristantech.org en \*.tristantech.org dezelfde names en values toegewezen krijgen. Het is dus niet nodig om ook een \*.tristantech.org aan te maken.



Nu maak ik in *Route 53 -> Hosted zones -> tristantech.org -> Create record* voor beide bovenstaande domeinen een nieuw CNAME record aan.

A screenshot of a computer

AI-generated content may be incorrect.

Nu ga ik terug naar mijn certificaat en de status is nu *Issued* in plaats van *Pending*. Het certificaat is gevalideerd op mijn domein. Hiermee heb ik bewezen dat ik de eigenaar ben van het domein.

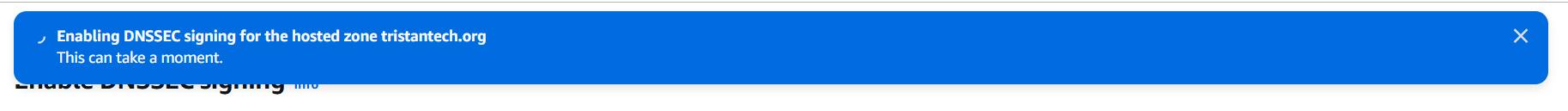
Wat ik ook nog wil doen, is het implementeren van DNSSEC in mijn hosted zone. Met DNSSEC worden cryptografische handtekeningen toegevoegd aan de records in mijn hosted zone. Door de handtekening te controleren bij een DNS-request, kan worden geverifieerd dat het opgevraagde DNS-record afkomstig is van de juiste nameserver en niet is gewijzigd door een partij met kwade intenties. Wat DNSSEC **niet** doet, is het versleutelen van de inhoud van het verkeer, er kan enkel mee geverifieerd worden of het teruggegeven record authentiek is.

In de tab *DNSSEC signing* klik ik op *Enable DNSSEC signing*.

A screenshot of a computer

AI-generated content may be incorrect.

Ik selecteer de optie om een nieuwe Customer Managed Key aan te maken en voer de naam in in de *Provide KSK name* en *Create customer managed CMK* velden. Dan klik ik op *Create KSK and enable signing*. DNSSEC wordt nu ingeschakeld.



Na enkele momenten zien je dat het signen van DNSSEC succesvol is verlopen. Nu moet een chain of trust nog tot stand gezet worden. Ik klik op *View information to create DS record*.

A screenshot of a chat

AI-generated content may be incorrect.

Ik klap de velden *Establish a chain of trust* en *Route 53 registrar* open en kopieer de public key.

A screenshot of a computer

AI-generated content may be incorrect.

Nu navigeer ik naar *Route 53 -> Registered domains -> tristantech.org* en klik ik op de tab *DNSSEC keys*.

A screenshot of a computer

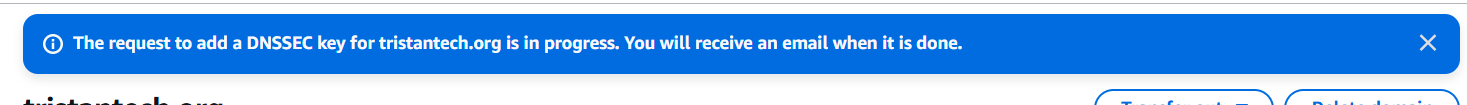
AI-generated content may be incorrect.

Ik klik op *Add key*, plak de Public key in het veld en klik weer op *Add*. De Key type en het algoritme hoeven niet aangepast te worden.

A screenshot of a computer

AI-generated content may be incorrect.

Een melding verschijnt dat de aanvraag om en DNSSEC key aan mijn domein toe te voegen is gedaan.



Enkele moment later ontvang ik een email.

A close-up of a computer screen

AI-generated content may be incorrect.

Nu is ook te zien dat de DNSSEC key is toegevoegd aan mijn domein.

A screenshot of a computer

AI-generated content may be incorrect.

Als ik nu naar <https://dnssec-debugger.verisignlabs.com/> navigeer en daar mijn domein invoer, zie ik dat het inrichten van DNSSEC succesvol is verlopen en er geen problemen zijn waargenomen.

A screenshot of a computer

AI-generated content may be incorrect.

**Content uit een S3 bucket ophalen met een CloudFront distributie**

Ik heb nu een domein aangemaakt in Route 53, en ik heb content in mijn bucket staan. Is het dan niet voldoende om mijn domein naar mijn bucket endpoint te laten verwijzen? **Nee.** Het gebruik van CloudFront biedt voordelen wat betreft performance, scalability, security, kosten en flexibiliteit. Hier kan je er meer over lezen: [Amazon CloudFront Key Features](https://aws.amazon.com/cloudfront/features/?nc=sn&loc=2&whats-new-cloudfront.sort-by=item.additionalFields.postDateTime&whats-new-cloudfront.sort-order=desc).

Ik ga naar Amazon CloudFront en klik op *Create distribution*.

A screenshot of a computer

AI-generated content may be incorrect.

Bij het creëren van de distribution kies ik het bucket endpoint van mijn eerder aangemaakte bucket *tristantech.org*. Het is dus van belang om hier het bucket endpoint en niet het website endpoint te kiezen (daarom heb ik het website endpoint eerder verwijderd). Het bucket endpoint is namelijk een REST API endpoint en hiermee kunnen *Origin access control setting*s worden ingericht. Op deze manier kan ik de bucket enkel publiek toegankelijk maken via mijn CloudFront distributie. Ik vink *Origin access control settings* dan ook aan en voeg hier mijn bucket aan toe.

A screenshot of a computer

AI-generated content may be incorrect.

Ik selecteer de optie *Redirect HTTP to HTTPS*.

A close-up of a computer

AI-generated content may be incorrect.

De *Web Application Firewall* zet ik aan.

A screenshot of a computer

AI-generated content may be incorrect.

Ik voeg twee *Alternate domain names* toe.

A screenshot of a computer

AI-generated content may be incorrect.

Ik voeg het SSL certificaat dat ik eerder heb aangemaakt in ACM toe zodat de verbinding wordt beveiligd met HTTPS.

A screenshot of a computer

AI-generated content may be incorrect.

En *Standard logging* schakel ik in.

A screenshot of a computer

AI-generated content may be incorrect.

Dan klik ik op *Create distribution.* De distribution is aangemaakt maar er verschijnt een melding dat de S3 bucket policy moet worden geupdate.

A screenshot of a computer

AI-generated content may be incorrect.

Ik kopieer de bucket policy en voeg hem toe aan mijn *tristantech.org* S3 bucket onder *Permissions -> Bucket policy*.

A screenshot of a computer

AI-generated content may be incorrect.

Als ik nu weer terugnavigeer naar mijn CloudFront distributie, dan vind ik onder *General -> Details* de domeinnaam van mijn distributie. Mijn wordt succesvol geladen via het distribution endpoint.

A screenshot of a computer

AI-generated content may be incorrect.

Ook is de website nu succesvol beveiligd met HTTPS.

A screenshot of a computer

AI-generated content may be incorrect.

En om nog te bevestigen dat we website enkel via de CloudFront distributie te bereiken is, navigeer ik naar mijn tristantech.org bucket en klik ik op de index.html file.

A screenshot of a computer

AI-generated content may be incorrect.

Hier is een *Object URL* zichtbaar. Door deze URL te openen, kan de content van de html file bekeken worden, ook wanneer *Static website hosting* is disabled. Als ik de URL open, krijg ik echter ook een AccessDenied melding.

A screenshot of a computer

AI-generated content may be incorrect.

De content van de website is nu echt alleen toegankelijk via CloudFront. Succes!

**Het doorverbinden van mijn Route 53 domein naar CloudFront**

Nu CloudFront en S3 zijn ingericht, moet het Route 53 domein nog worden gekoppeld aan de CloudFront distributie. Ik wil namelijk in de browser de adressen *tristantech.org* en *www.tristantech.org* kunnen intypen, en automatisch worden doorverbonden naar mijn distributie. De eerste stap om dit te bereiken, heb ik al gezet bij het aanmaken van de distributie toen ik de *Alternate domain names* heb gedefinieerd.

A screenshot of a computer

AI-generated content may be incorrect.

Ik moet in Route 53 echter nog inrichten dat het verkeer naar mijn gekochte domein naar CloudFront moet worden gestuurd.

Nu is het tijd om meer records toe te voegen aan de hosted zone, zodat we uiteindelijk de websitecontent uit de S3 bucket kunnen ophalen. Ik navigeer naar *Route 53 -> Hosted zones -> tristantech.org*. Je ziet in de onderstaande afbeelding de eerder besproken NS, SOA en CNAME records staan. Ook heb ik twee A records toegevoegd. (Deze staan hier nu al in, maar ik zal laten zien hoe ik ze gecreëerd heb.)

A screenshot of a computer

AI-generated content may be incorrect.

Een A record koppelt een domeinnaam aan een specifiek IPv4-adres. Zo kan ik mijn domeinnaam dus koppelen aan mijn CloudFront distributie, en ik doe het ook meteen voor het subdomein *www.tristantech.org*. Ik klik op *Create record* en kies voor de volgende inrichting:

* Het *subdomain* laat ik leeg
* Het *Record type* is dus A
* Ik zet de switch *Alias* **aan**
* Ik selecteer bij *Route traffic to* de optie *Alias to CloudFront distribution*
* Dan selecteer ik mijn eerder aangemaakte distributie
* En de *Routing policy* laat ik op *Simple routing* staan

A screenshot of a computer

AI-generated content may be incorrect.

Vervolgens klik ik op *Add another record* en herhaal ik de net uitgevoerde stappen, met als enige verschil dat ik nu bij het *subdomain* de tekst ‘www’ invoer.

A screenshot of a computer

AI-generated content may be incorrect.

Als ik nu op *Create record* klik, worden de A records toegevoegd aan mijn hosted zone. Tijd om te testen of het doel bereikt is. Ik navigeer in de browser naar *tristantech.org*. De website wordt geladen. Succes!

A screenshot of a computer

AI-generated content may be incorrect.

En ook als ik in mijn browser naar *www.tristantech.org* navigeer, wordt mijn website geladen. Yes, het is gelukt!

A screenshot of a computer

AI-generated content may be incorrect.

**Het toevoegen van dynamische functionaliteit mijn gehoste website**

Mijn statische website draait nu succesvol in mijn AWS cloud. Ik wil er echter dynamische content aan toevoegen: een counter die bijhoudt hoeveel keer mijn homepage is bezocht. Statische content is, zoals uit de naam valt te concluderen, altijd hetzelfde. Hiervoor is dus geen server-side verwerking nodig. Dynamische content daarentegen, wordt in real-time gegenereerd en kan veranderen op basis van gebruikersinteractie. Dit vereist vaak server-side verwerking en een database.

Om mijn counter in mijn website te implementeren, ga ik *serverless* werken. Dit betekent niet dat er geen servers zijn, maar simpelweg dat ik ze niet hoef in te richten en te onderhouden, dat doet AWS voor me.

**Gebruikte services:**

* **AWS S3**
* **AWS Lambda**
* **AWS DynamoDB**
* **AWS API Gateway**

Voordat ik het inrichten van de services induik, wil ik kijken naar de architectuur van mijn project.

A diagram of a software company

AI-generated content may be incorrect.

A diagram of a company

AI-generated content may be incorrect.

De services *Route 53*, *CloudFront*, *ACM* en *S3* heb ik natuurlijk al opgezet en ingericht. Maar wat gebeurt er precies met de nieuw toegevoegde services?

1. Ik begin bij een bekende: S3. Er moet wat JavaScript code worden toegevoegd aan de bucket zodat het aantal websitebezoeken op de hoofdpagina kan worden getoond. Ook moet er code worden toegevoegd die kan doorgeven wanneer de websitebezoeken uit de database moeten worden opgehaald en wanneer het aantal websitebezoeken in de database moet worden geupdate, oftewel HTTP requests aan een API verstuurt.
2. Wanneer de code in de S3 bucket getriggerd wordt, wordt er een request gestuurd naar een REST API die ik deploy met de API Gateway service. De API Gateway fungeert als een ‘tussenpersoon’ tussen de client en de back-end services.
3. API Gateway richt ik in om op zijn beurt een Lambda-functie te triggeren wanneer de API een request ontvangt. AWS Lambda is een service waarmee (back-end) code kan worden gerund zonder te hoeven denken aan het inrichten en onderhouden van servers, vandaar de naam *serverless*. In Lambda kan ik de logica schrijven die communiceert met de database en dus het aantal websitebezoekers ophaalt of update.
4. DynamoDB is een NoSQL database service waarmee data kan worden beheerd zonder servers te hoeven managen, dus wederom *serverless*. In DynamoDB houd ik de telling van het aantal websitebezoeken bij.

**Het aanmaken van een DynamoDB table**

Ik open de service DynamoDB en klik op *Create table*. Ik hoef de database zelf niet te creëren vanwege het *serverless* concept, dus onderliggende infrastructuur wordt door AWS verzorgd. Bij het aanmaken van de table kies ik voor de volgende instellingen:

* Table name: website-visits
* Partition key: count\_id
* Table settings: Default settings

A screenshot of a computer

AI-generated content may be incorrect.

Als ik vervolgens nog een keer op *Create table* klik, zie ik dat mijn tabel in het *Tables* overzicht is verschenen.

A screenshot of a computer

AI-generated content may be incorrect.

Nu open ik de tabel, klik ik op *Explore table items* en klik ik op *Create item* in het *Items returned* overzicht.

A close-up of a computer screen

AI-generated content may be incorrect.

Ik geef de *count\_id* de waarde ‘total’ en ik voeg een nieuw Number attribuut toe met de naam *visit\_count* en de waarde 0. Zoals je eerder zag, is *count\_id* de Partition key. Deze wordt gebruikt om items in een een tabel te kunnen identificeren. Daarnaast wil ik natuurlijk ook de telling bijhouden, vandaar dat ik een *visit\_count* attribuut heb aangemaakt. Vervolgens klik ik weer op *Create item*.

A screenshot of a computer

AI-generated content may be incorrect.

De *count\_id* en *visitor\_count* zijn nu zichtbaar in het *table items* overzicht.

A screenshot of a computer

AI-generated content may be incorrect.

Nu navigeer ik terug naar het *Table* details overzicht, ga ik in mijn tabel *website-visits* naar de tab *Backups* eb klik ik in het overzicht *Point-in-time recovery* op *Edit*. Ik zet de *Backup recovery period* op 35 dagen en klik op *Save changes*. *Point-in-time recovery* is nu **ingeschakeld**. Mocht ik per ongeluk data uit mijn tabel verwijderen of verkeerd aanpassen, dan kan ik gebruik maken van de backup.

A screenshot of a computer

AI-generated content may be incorrect.

Mijn DynamoDB tabel is nu succesvol ingericht. Tijd om hiermee te gaan communiceren.

**Het schrijven van een Lambda functie om de DynamoDB tabel te kunnen bevragen en updaten**

Lambda is zoals eerder genoemd een serverless oplossing om back-end code te kunnen draaien. In Lambda moeten losse functies worden aangemaakt. Dit heeft een aantal voordelen:

* Je wordt gedwongen om modulair te werken. Iedere functie bestaat uit een kort script dat verantwoordelijk is voor slechts één taak.
* Omdat scripts kort blijven en zich op 1 taak richten, blijf je een goed overzicht houden van hoeveel geheugen er wordt gebruikt bij het draaien van de code, waardoor performance en kosten geoptimaliseerd kunnen worden.
* Up- en downscaling kan gemakkelijk gerealiseerd worden omdat AWS de onderliggende infrastructuur al heeft verzorgd. Je hoeft enkel functies toe te voegen of te verwijderen.

Nu open ik de Lambda service en klik ik op *Create function*. Ik kies voor de volgende instellingen:

* *Author from scratch*, ik ga zelf een functie schrijven en heb geen template nodig.
* *Function name:* countNumberOfWebsiteVisits
* *Runtime*: Node.js 22.x
* *Architecture*: x86\_64
* De *default execution role* en *Additional Configurations* pas ik nu niet aan.

A screenshot of a chat

AI-generated content may be incorrect.

Als ik nu nog een keer op *Create function* klik, wordt de functie aangemaakt.

A screenshot of a computer

AI-generated content may be incorrect.

Ook staat er wat default code in het *Code source* overzicht.

A screenshot of a computer

AI-generated content may be incorrect.

Nu is het tijd om mijn eigen code toe te voegen.

A screen shot of a computer program

AI-generated content may be incorrect.

Ik zal in grote lijnen uitleggen wat hier gebeurt:

* Regels 1 en 2: ik importeer de DynamoDB client en commandos om hiermee te kunnen communiceren vanuit de AWS Software Development Kit. Dit is nodig om met mijn aangemaakte DynamoDB tabel te kunnen ‘praten’.
* Regel 4: de handler functie wordt getriggered wanneer Lambda de code gaat runnen. De functie is asynchroon omdat ik met API calls ga werken. Op deze manier kan het werken met API calls netjes worden verwerkt en afgerond. Denk bijvoorbeeld aan wanneer er moet gewacht worden op een response of wanneer er een error moet worden afgehandeld.
* Regel 6: als de http-method gelijk is aan ‘POST’, dan wil ik mijn visitor\_count gaan updaten.
* Regel 7 tot en met 17: hier geef ik in welke tabel en partition key er geupdate moet worden. Ook geef ik aan welk attribuut ik moet updaten en met welke waarde. In dit geval incrementeer ik *visitor\_count* met de waarde 1. Bij ieder POST request wordt de counter dus met 1 verhoogd.
* Regel 19 tot en met 32: