Jordi Strybos
Reinoud Neijzen
Sander Vandamme
Bavo Van Meel

Inhoudstafel

Nuttige links

Publieke link: http://edu.ap.bucketapphost.s3-website-us-east-1.amazonaws.com/
Github repository: https://github.com/reinoud-neijzen-ap/ict-architecture

Samenvatting

Vandaag de dag is de wereld meer en meer digitaal. De corona pandemie heeft er nog sterker voor gezorgd dat er een nood ontstaat om bestanden op een platform te kunnen delen met andere gebruikers.
Er zijn al soortgelijke services op de markt, maar vragen vaak een premiumprijs. De service die wij gaan maken, houdt bestanden slechts 24 uur bij zodat de opslagkosten gedrukt blijven. Op deze manier kunnen we de service tegen een zeer lage prijs aanbieden.

Situatie to-be

Authenticatie

Personen die de service willen gebruiken moeten kunnen inloggen. Je moet ingelogd zijn om bestanden te kunnen uploaden en downloaden.

Om in te loggen heb je de volgende gegevens nodig:

- Username
- Wachtwoord

Als de inloggegevens correct zijn, krijg je een token terug. Nadat je deze token hebt, kan je de gateway contacteren.

Opslag

Bestanden worden opgeslagen in een database. Als een gebruiker een bestand upload, krijgt het bestand een zogenaamde TTL (Time To Live) mee. Na 24 uur worden deze bestanden verwijderd van de database. Dit wordt gedaan om de opslagkosten laag te houden.

Bestanden uploaden

Enkel geregistreerde gebruikers kunnen bestanden uploaden. Wanneer een gebruiker een bestand uploadt, wordt een unieke code (UUID) aangemaakt. Ook zal er een checksum worden bijgehouden zodat de gebruikers hun download kunnen verifiëren. Deze informatie (UUID en checksum) moet dan uiteraard worden bijgehouden in de file.

Nice to have

Bij het uploaden van een bestand kan de gebruiker kiezen hoeveel keer het bestand gedownload mag worden. Als de gebruiker een negatieve waarde ingeeft, mag het bestand oneindig veel gedownload worden.

Bij het uploaden van het bestand, kan de gebruiker ook kiezen om een extra wachtwoord in te stellen. Als een gebruiker dit bestand dan wilt downloaden, moet hij een dubbele beveiliging uitvoeren:

- De gebruiker moet de UUID kennen.
- De gebruiker moet het extra wachtwoord kennen.

Bestanden downloaden

Enkel geregistreerde gebruikers kunnen bestanden downloaden. De gebruiker die een bestand wenst te downloaden, moet de UUID kennen. Wanneer de gebruiker de download actie uitvoert, wordt de gebruiker die de download uitvoert tezamen met het gedownloade bestand opgeslagen in een log. Zo kan men zien wie een download heeft uitgevoerd en over welke file dit ging. Natuurlijk krijgt de gebruiker na de request ook de gewenste file.

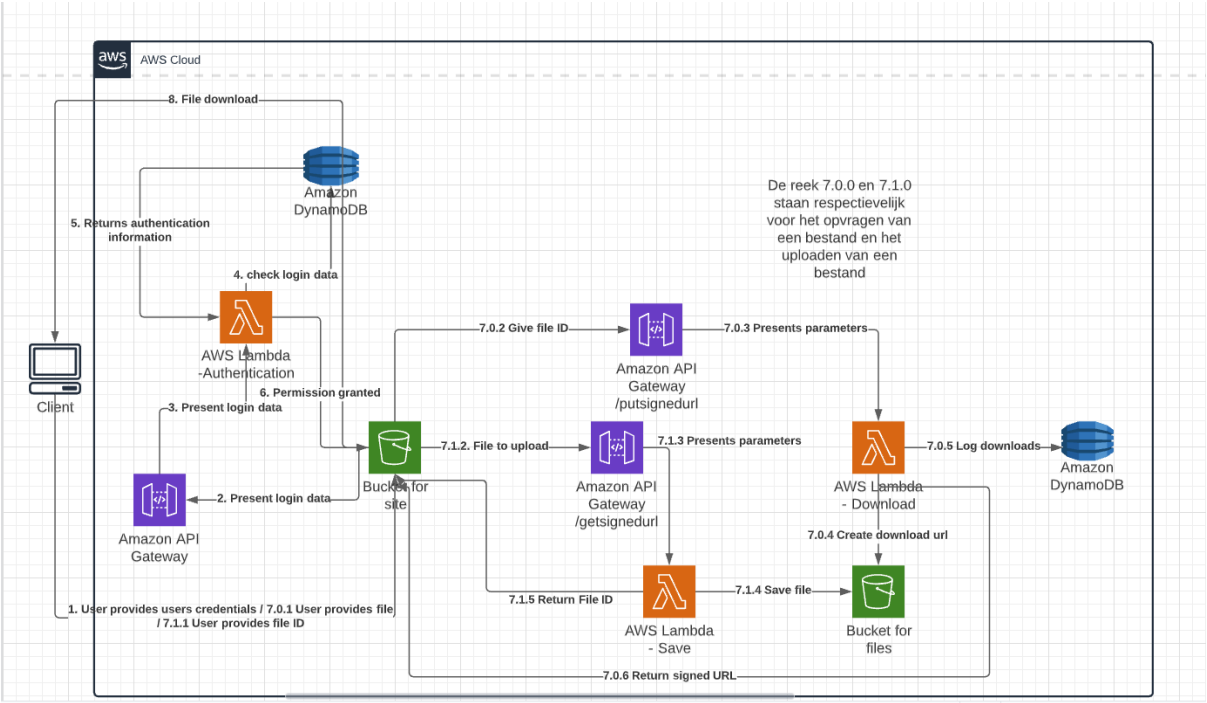
Access log

Telkens wanneer een gebruiker een bestand downloadt, wordt de gebruiker toegevoegd aan een log met daarin zijn naam, de opgevraagde file en de datum en tijd van opvraging.

Nice to have

De gebruiker die het bestand heeft geüpload, kan ook de access log opvragen. Er kan ook een optie worden voorzien bij het uploaden van de file om de access log naar de gebruiker te mailen als het bestand van de database verwijderd wordt.

Architectuur



We zullen kort even de architectuur toelichten.
Alles begint bij de client. De client zal eerst bij de site moeten inloggen bij de site. Deze site is gehost op een bucket. Deze inloggegevens zullen via een API en een lambda functie verwerkt en gecontroleerd worden. Hiervoor wordt ook een dynamoDB betrokken voor het opslaan van de wachtwoorden.

Als de user credentials zijn goedgekeurd door de lambda functie kan de gebruiker beginnen aan de volgende stappen. De client kan dan een file die hij wil uploaden aan de site geven. De site zal deze dan aan de hand van een API en een lambda functie opslaan in een bucket. De lambda functie zal de ID van de file dan ook terugsturen zodat een gebruiker deze achteraf met deze ID kan downloaden.

De andere mogelijkheid die de gebruiker heeft is aan de hand van een ID een file downloaden. Dit doet hij door de ID in te geven en dan zal de site ervoor zorgen dat een API call gemaakt wordt. Deze API call zal een lambda functie triggeren en zo zal een presigned url tezamen met een checksum worden teruggestuurd naar de site. Deze lambda zal ook deze download loggen. De site back-end zal dan de file weldegelijk downloaden en zo krijgt de gebruiker een file en een checksum terug.

Verantwoording keuzes

Login

Om in te loggen, maken we gebruik van een lambda functie en dynamoDB in de plaats van cognito. Cognito is een zeer goede optie, beter dan waar wij voor hebben gekozen, maar het valt buiten de scope van het project.
Om in te loggen moet je een username en password opgeven. Deze worden dan gecontroleerd in de lambda functie aan de hand van dynamoDB waar de inloggegevens opgeslagen zijn.

Op vlak van security biedt het opslaan van de inloggegevens in een aparte database ook nog een voordeel. Deze staan dan niet op site niveau en er kan extra beveiliging voor worden ingebouwd zodat deze gegevens moeilijker te achterhalen zijn.

Upload

Voor de save-operatie hebben we verkozen met drie verschillende zaken te werken: een lambda, een API en een bucket. De lambda moet functies uitvoeren wanneer die een bestand binnen krijgt. Dan gaat hij een UUID berekenen. Ook moet deze lambda de functie hebben om uiteindelijk deze zaken en het document op te slaan in de bucket. De klant moet ook een antwoord krijgen als de operatie succesvol is.

De bucket is handig en gemakkelijk te bereiken vanuit een lambda. De bestanden worden hier rechtstreeks geplaatst met de uuid als naam.

Wanneer de klant een bestand heeft gekozen kan hij op de “upload” knop drukken. Wanneer dit gebeurt zal de website een get-request sturen naar de API gateway die verbonden is met de lambda. Bij deze get request stuurt de website nog 2 parameters mee. Deze zijn: het bestandstype en het contenttype van het bestand. De lambda gebruikt deze dan om een uuid aan te maken en het bestand op te slaan in de bucket. Door deze 2 parameters mee te sturen kunnen we het bestand met de juiste extensie opslaan en erna ook weer downloaden.

Eens de lambda dit allemaal heeft verwerkt stuurt deze een url en een uuid terug. De uuid moet de gebruiken bijhouden aangezien hij deze zal nodig hebben indien hij het bestand terug wilt downloaden. De url wordt dan in de website gebruikt om een put-request uit te voeren op de S3 bucket waar de file wordt opgeslagen. Hierna is heel het upload proces afgerond en zit de file in de bucket.

API gateway

De API gateway zorgt er voor dat er nog meer security is. Zo kan de client niet direct aan de buckets komen. De client moet dan specifieke requests gebruiken om toegang te krijgen tot zaken die al ingesteld zijn.

Download

Voor de download hebben we drie belangrijke componenten: een API, een lambda functie en een dynamoDB database.

Als eerste komt de API. De API zal een lambda functie aanroepen en ervoor zorgen dat de lambda functie de ID zal weten van de file die gedownload moet worden uit de s3 bucket en eveneens ook de naam van de gebruiker die de download uitvoert.

De lambda functie zal dan aan de hand van de info die wordt meegegeven door de API twee zaken doen. Ten eerste zal deze functie een signed url creëren voor het opgevraagde object en deze signed url dan ook terugsturen, zodat deze gebruikt kan worden om te het object uiteindelijk te downloaden. Ook wordt de bestandsextensie van het opgevraagde document mee opgestuurd mits deze nodig is voor het correct downloaden van de file elders.

Als tweede zal de lambda functie de download loggen naar een dynamoDB tabel. Voor dit te doen zal de lambda functie een uuid creëren en deze dan met de ID van de opgevraagde file,de user en het tijdstip van opvraging opslagen in de database. Op deze manier kan men in deze database de specifieke file en user terugvinden die betrokken waren bij de download.

Door een API te gebruiken kunnen we makkelijk en gecontroleerd onze lambda functie aanspreken(triggeren) zonder de cliënt hier rechtstreeks in verbinding mee te moeten zetten. Eveneens zorgt de werking van het download gedeelte ervoor dat mensen die niet beschikken over de exacte uuid van het document deze ook niet makkelijk kunnen gokken, het is namelijk een uuid van 32 karakters. Verder maakt de API het ons ook gemakkelijk de juiste lambda functies op de juiste moment aan te spreken.

Om de lambda die de download verzorgt te laten werken moeten we deze zijn permissions zo instellen dat deze toegang hebben tot juiste dynamoDB en bucket. Dit doen we door in de permission policy deze AmazonS3FullAccess te geven alsook DynamoDBFullAccess.

De exacte settings voor de services van dit onderdeel zijn terug te vinden in de bijlage met alle schermafbeeldingen. Deze schermafbeeldingen bevatten enkel afbeelding van configuratie onderdelen die tijdens het maken van het project zijn aangepast.

S3 File Bucket

Om de files op te slaan hebben we gekozen gebruik te maken van een S3 bucket. Hierop voeren we zowel “get” als “put” requests uit. We kunnen deze acties toelaten via een specifieke link(signed url) en alle andere calls weigeren. Zo zit er toch een beveiliging op. In de bucket worden de files ook verwijderd na 24 uur via een lifecycle rule. Verder hebben we ook aan de hand van de cors configuratie getracht ervoor te zorgen dat we enkel de eerder vernoemde “put” en “get” requests toelaten.