Youtube naar MP3

Maak eenvoudig preek-mp3's van een YouTube-video! 🎧✨

Nooit meer gedoe met het handmatig knippen en converteren van preken uit je zondagsdiensten op YouTube! Met onze slimme oplossing transformeer je elke YouTube-opname moeiteloos naar een compacte, heldere MP3 van alleen de preek via een website.

Ontdek de Voordelen:

- ✂️ Preek perfect geknipt: Voer simpelweg de YouTube-link en de start- en eindtijd van de preek in. Wij doen de rest!
- 🎵 Kristalheldere audio, klein formaat: ontvang een MP3-bestand dat geoptimaliseerd is voor spraak: uitstekende kwaliteit zonder onnodig grote bestanden. Ideaal om te delen of te archiveren.
- 💻 Eenvoudige web interface: geen technische kennis nodig! Een duidelijke website leidt je door het proces: link plakken, tijden invoeren, en downloaden maar.
- 💰 Extreem kostenefficiënt: Gebouwd op de nieuwste serverless technologie van AWS. Je betaalt alleen voor wat je daadwerkelijk gebruikt – geen dure servers, geen onnodige kosten. Perfect voor budgetbewuste (vrijwilligers)organisaties!
- 🧹 Automatisch opgeruimd: De MP3-bestanden worden na 24 uur automatisch verwijderd. Zo blijft je opslag netjes en de kosten minimaal, zonder dat je er omkijken naar hebt.
- 🛠️ Minimaal onderhoud: serverless betekent geen servers beheren, geen updates installeren. Focus op wat echt telt, niet op IT-beheer.

Hoe Werkt Het?

- Plak de YouTube-link van de dienst.
- Specificeer de start- en eindtijd van de preek.
- Klik op "Verwerken" en wacht even.
- Download je kant-en-klare preek-MP3!

Zet vandaag nog de stap naar moeiteloze preek-archivering en -verspreiding!

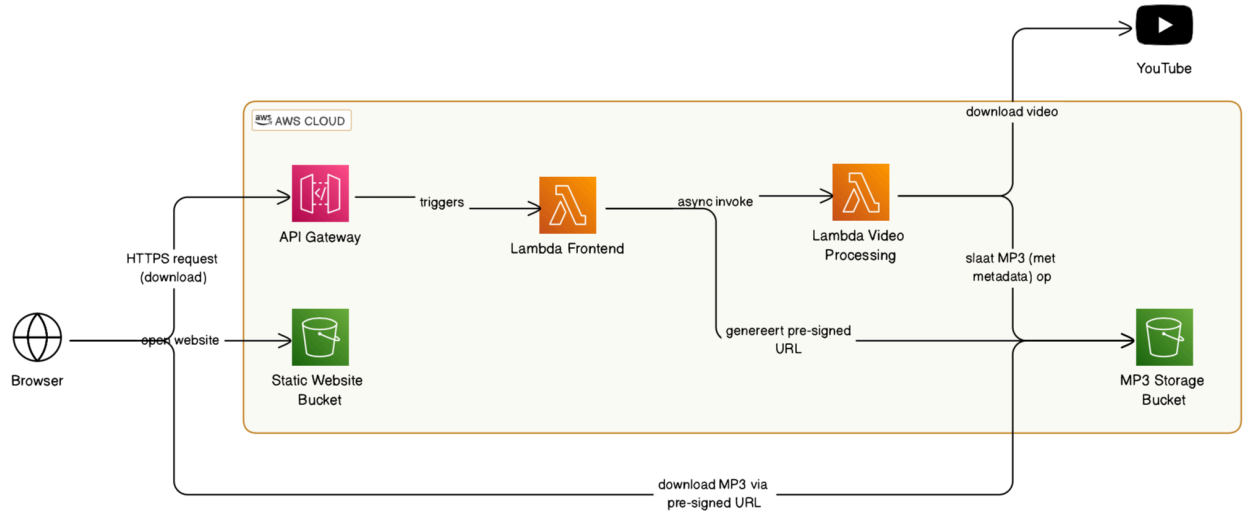
Projectplan: Preek naar MP3 Service (met IaC)

Dit is een Agile projectplan om de AWS-architectuur iteratief te bouwen, met een focus op Python waar mogelijk, inclusief het implementeren van Infrastructure as Code (IaC) met AWS SAM. We gebruiken een aanpak die lijkt op Scrum, met korte iteraties (sprints) om snel werkende onderdelen op te leveren.

Projectdoel: Een kosteneffectieve, onderhoudsarme AWS-oplossing ontwikkelen om preken uit YouTube-video's te extraheren als MP3-bestanden, toegankelijk via een eenvoudige webinterface, en beheerd via Infrastructure as Code.

Methodologie: Agile (Scrum-achtig)

- **Sprintlengte:** 1-2 weken (afhankelijk van je beschikbare tijd als vrijwilliger).
- **Iteratieve Oplevering:** Elke sprint levert een werkend (deel)product op.



Sprint 0: Basis Setup & Kernfunctionaliteit (Proof of Concept) 🚀

- **Doel:** De kern van de video-naar-audio conversie lokaal en daarna in een Lambda-functie werkend krijgen.
- **Taken:**
  1. **AWS Account & IAM Setup:**
     - Zorg dat je AWS-account klaar is.
     - Maak een IAM-gebruiker aan voor development met beperkte rechten.
     - Maak initiële IAM-rollen voor Lambda (toegang tot S3, CloudWatch Logs).
  2. **Lokale Python Scripting (Onderzoek & Prototyping):**
     - Python script om een YouTube-video te downloaden (gebruik yt-dlp).
     - Python script om met FFmpeg (lokaal geïnstalleerd):
       - Audio uit een videobestand te extraheren.
       - Het audiofragment te knippen op basis van timestamps.
       - Te converteren naar MP3 (test verschillende bitrates voor kwaliteit vs. grootte, bijv. 64kbps/96kbps mono).
  3. **"Video Processing" Lambda Functie (Python) - Eerste Versie:**
     - Project opzetten (voor nu handmatig, later via SAM).
     - yt-dlp en FFmpeg beschikbaar maken in Lambda:
       - **Aanpak:** Maak een Lambda Layer met een statische build van FFmpeg en de yt-dlp library.
       - Code om input (YouTube URL, timestamps - voor nu hardcoded of via test event) te ontvangen.
       - Code om de video te downloaden.
       - Code om audio te extraheren, knippen en converteren naar MP3.
  4. **S3 Bucket voor MP3 Opslag:**
     - Maak een S3 bucket aan (handmatig).

        - Lambda-functie slaat de gegenereerde MP3 op in deze bucket.
- **Definition of Done (DoD) Sprint 0:**
    - Een MP3-bestand kan succesvol worden gegenereerd en opgeslagen in S3 door de "Video Processing" Lambda handmatig (of via een test event) aan te roepen met een YouTube URL en timestamps.
    - FFmpeg en yt-dlp werken binnen de Lambda-omgeving.

Sprint 1: Web Interface & API Koppeling 🌐

- **Doel:** Een basis webinterface waarmee de gebruiker de YouTube-link en timestamps kan invoeren en het proces kan starten.
- **Taken:**
    1. **Eenvoudige Web Interface (HTML, CSS, JavaScript):**
        - HTML-formulier met invoervelden voor: YouTube URL, Starttijd, Eindtijd.
        - Een "Verwerk" knop.
        - Basis JavaScript om de formulierdata te verzamelen.
    2. **S3 Static Website Hosting:**
        - Configureer een S3 bucket voor het hosten van de statische webbestanden (handmatig).
    3. **API Gateway (HTTP API):**
        - Stel een HTTP API endpoint op (handmatig).
        - Route voor het indienen van de preekdetails.
    4. **"Frontend Logic" Lambda Functie (Python):**
        - Ontvangt data van API Gateway.
        - Valideert de input (eenvoudige validatie, bijv. of velden ingevuld zijn).
        - Roept de "Video Processing" Lambda **asynchroon** aan (bijv. met boto3 invoke met InvocationType='Event').
        - Geeft een directe bevestiging terug aan de frontend (bijv. "Verzoek ontvangen, verwerking gestart").
    5. **Frontend-Backend Integratie:**
        - JavaScript op de webpagina om een POST request naar het API Gateway endpoint te sturen met de formulierdata.
        - Toon de bevestiging van de "Frontend Logic" Lambda.
- **Definition of Done (DoD) Sprint 1:**
    - De gebruiker kan via de webinterface een YouTube URL en timestamps invoeren.
    - De "Verwerk" knop triggert de "Video Processing" Lambda via API Gateway en de "Frontend Logic" Lambda.
    - De gebruiker krijgt een bevestiging dat het verzoek is ontvangen.
    - De MP3 wordt (nog steeds) in S3 opgeslagen (geen downloadlink in UI nog).

Sprint 2: MP3 Download & Status Feedback 📥

- **Doel:** De gebruiker in staat stellen de gegenereerde MP3 te downloaden en feedback te geven over de status.
- **Taken:**
    1. **Unieke Bestandsnamen & Metadata:**
        - "Video Processing" Lambda: Sla MP3's op met een unieke identifier (bijv. gebaseerd op een UUID of de request ID) om conflicten te voorkomen.

2. **Download Mechanisme:**
    - "Frontend Logic" Lambda (of een nieuwe, kleine Lambda voor status/download): Genereer een **pre-signed S3 URL** voor het MP3-bestand.
    - Frontend JavaScript:
        - Implementeer polling. De frontend vraagt periodiek (elke X seconden) aan een nieuw API Gateway endpoint (gekoppeld aan een Lambda) of het bestand klaar is. Deze status-Lambda checkt of het S3 object (met de unieke identifier) bestaat.
        - Zodra het bestand klaar is, toon een downloadlink (de pre-signed S3 URL) op de webpagina.
3. **Basis Status Feedback:**
    - Frontend: Toon berichten zoals "Verwerking bezig...", "MP3 is klaar voor download!", "Fout opgetreden".

- **Definition of Done (DoD) Sprint 2:**
    - Nadat de verwerking is voltooid, kan de gebruiker via de webinterface een downloadlink voor de MP3 ontvangen en het bestand downloaden.
    - De gebruiker krijgt basis feedback over de status van het proces.

Sprint 3: Geautomatiseerde Cleanup & UI/UX Verbeteringen ✨

- **Doel:** Automatische verwijdering van MP3's implementeren en de gebruikerservaring verder verbeteren.
- **Taken:**
    1. **Automatische MP3 Verwijdering (Handmatige Configuratie voor nu):**
        - Configureer een **S3 Lifecycle Policy** op de MP3-opslagbucket (handmatig via de AWS console) om objecten automatisch te verwijderen na 24 uur. *Dit wordt later geautomatiseerd met SAM.*
    2. **Verbeterde Input Validatie (Frontend & Backend):**
        - Frontend (JavaScript): Valideer YouTube URL formaat, zorg dat timestamps logisch zijn (eind > start). Geef duidelijke foutmeldingen bij de invoervelden.
        - Backend ("Frontend Logic" Lambda): Robuustere validatie van de input.
    3. **Error Handling & Logging:**
        - "Video Processing" Lambda & "Frontend Logic" Lambda: Implementeer try-except blokken voor foutafhandeling.
        - Log duidelijke fouten naar Amazon CloudWatch Logs.
        - Geef gebruikersvriendelijke foutmeldingen terug aan de frontend als er iets misgaat.
    4. **UI Verbeteringen:**
        - Verbeter de styling van de webpagina (eenvoudig, functioneel).
        - Duidelijkere instructies voor de gebruiker.
- **Definition of Done (DoD) Sprint 3:**
    - MP3-bestanden worden automatisch na 24 uur verwijderd uit S3 (via handmatige lifecycle policy).
    - De webinterface heeft verbeterde inputvalidatie en geeft betere feedback/foutmeldingen.
    - De Lambda-functies hebben basis error handling en logging.

Sprint 4: Infrastructure as Code (AWS SAM) 🏗️

- **Doel:** Alle AWS-resources definiëren en beheren met AWS SAM.
- **Taken:**
    1. **AWS SAM CLI Installatie & Setup:**
        - Installeer de AWS SAM CLI lokaal.
        - Initialiseer een nieuw SAM-project (sam init) of integreer SAM in je bestaande projectstructuur.
    2. **SAM Template (template.yaml) Creatie:**
        - Definieer de "Video Processing" Lambda functie, inclusief:
            - Runtime, handler, code locatie.
            - Geheugen, timeout.
            - IAM-permissies (rol automatisch gegenereerd door SAM of expliciet gedefinieerd).
            - Verwijzing naar de FFmpeg/yt-dlp Lambda Layer.
        - Definieer de "Frontend Logic" Lambda functie (en eventuele status Lambda), inclusief configuraties.
        - Definieer de API Gateway (HTTP API) en de routes naar de Lambda-functies.
        - Definieer de S3 bucket voor MP3-opslag.
        - **Belangrijk:** Definieer de **S3 Lifecycle Policy** voor automatische verwijdering binnen de SAM-template voor de S3 bucket.
        - Definieer de S3 bucket voor static website hosting.
        - Definieer benodigde IAM-rollen en -policies binnen SAM, met het principe van least privilege.
    3. **Lambda Functie Code Aanpassing:**
        - Organiseer je Python code zodat deze makkelijk te packagen en deployen is met SAM.
    4. **Bouwen en Deployen met SAM:**
        - Gebruik sam build om je applicatie te bouwen.
        - Gebruik sam deploy --guided (initieel) en sam deploy om de stack naar AWS te deployen.
    5. **Testen van de SAM Deployment:**
        - Verifieer dat alle resources correct zijn aangemaakt in AWS.
        - Test de end-to-end functionaliteit na deployment via SAM.
- **Definition of Done (DoD) Sprint 4:**
    - Alle benodigde AWS-resources (Lambda's, API Gateway, S3 buckets, IAM-rollen, S3 Lifecycle Policy) zijn gedefinieerd in een template.yaml bestand.
    - De volledige applicatie kan worden gebouwd en gedeployed met AWS SAM CLI commando's.
    - De S3 Lifecycle Policy voor automatische verwijdering is nu onderdeel van de IaC.
    - De functionaliteit die in eerdere sprints is gebouwd, werkt nog steeds na deployment via SAM.

Sprint 5: Testen, Documentatie & "Deployment" Afronding 🧪📜✅

- **Doel:** De oplossing grondig testen, documenteren en de "livegang" binnen de kerk

afronden.

- **Taken:**
    1. **Grondig Testen (End-to-End na SAM):**
        - Herhaal testen van Sprint 3, nu met de via SAM gedeployde infrastructuur.
        - Test met verschillende YouTube-video's (kort, lang, verschillende kanalen).
        - Test randgevallen: ongeldige URL's, onmogelijke timestamps, video's die niet gedownload mogen worden.
        - Test de downloadfunctionaliteit op verschillende browsers/apparaten (indien relevant).
        - Verifieer de automatische cleanup die nu via SAM is geconfigureerd.
    2. **Security Review (SAM context):**
        - Controleer de IAM-permissies zoals gedefinieerd in de SAM-template.
        - Zorg dat S3 buckets correct geconfigureerd zijn (bijv. geen onnodige publieke toegang).
    3. **Documentatie:**
        - Werk het README.md bestand bij met:
            - Instructies voor het deployen en updaten van de applicatie met AWS SAM (sam build, sam deploy).
            - Overzicht van de SAM-template structuur.
            - Hoe de webinterface te gebruiken.
            - Architectuuroverzicht.
            - Instructies voor eventuele toekomstige beheerders/ontwikkelaars.
    4. **Kostenoptimalisatie Review:**
        - Controleer de Lambda-geheugeninstellingen en execution times in de SAM-template.
        - Verifieer dat je de goedkoopste geschikte S3-opslagklasse gebruikt in de SAM-template.
    5. **Finale "Deployment" en Overdracht:**
        - Zorg dat de laatste versie via SAM is gedeployed.
        - Loop de werking en het beheer (via SAM) eventueel door met een andere vrijwilliger.
- **Definition of Done (DoD) Sprint 5:**
    - De oplossing is grondig getest en stabiel bevonden na deployment met SAM.
    - Uitgebreide documentatie, inclusief IaC-beheer, is aanwezig.
    - Security en kostenaspecten zijn overwogen en geconfigureerd via SAM.
    - De oplossing is klaar voor gebruik door de vrijwilligers van de kerk en kan consistent worden beheerd en gedeployed.

Algemene Overwegingen tijdens het Project:

- **Python Versie:** Gebruik een recente, ondersteunde Python-versie voor je Lambda-functies (bijv. Python 3.9+).
- **Secrets Management:** Vermijd hardcoding van gevoelige informatie.
- **Broncodebeheer:** Gebruik Git (bijv. met GitHub, GitLab, of AWS CodeCommit) om je code en SAM-template te beheren.
- **Regelmatige Feedback:** Als er andere vrijwilligers zijn die dit gaan gebruiken, vraag tussentijds om feedback.

Dit aangepaste plan integreert IaC op een logisch moment, nadat de kernfunctionaliteit is

gevalideerd, waardoor je een robuustere en beter beheerbare oplossing krijgt. Veel succes!