Video Processing Application Design with AWS Lambda and FFMPEG

Team member: 120090155 Zheng Li

122090118 Gan Zhengyang

121090883 Zhu Yihui

# Background:

In the era of digital content consumption, video streaming has become one of the most popular forms of online entertainment. However, the quality of video playback is often hindered by varying internet bandwidth, leading to buffering, pixelation, and an overall poor user experience. To address this issue, we propose an innovative cloud-based video processing application that intelligently optimizes video content to adapt to users' network conditions, thereby providing a smoother and higher-quality viewing experience.

# Overview:

This document outlines the design of a video processing application that triggers an AWS Lambda function when a video is uploaded to S3. The Lambda function processes the video using FFmpeg to convert it into different resolutions based on the user's network conditions and stores the processed video back in S3. Below is the detailed architecture and implementation steps.

# 1. General Process Overview:

- The user uploads a video file via the frontend.  
- The frontend detects the user's network conditions (bandwidth, network type, etc.).  
- The video is uploaded to S3 using a pre-signed URL.  
- The S3 upload triggers an AWS Lambda function to process the video.  
- The Lambda function uses FFmpeg to convert the video to a resolution suitable for the user's network.  
- The converted video is uploaded back to S3.  
- The frontend provides a download link for the processed video.

# 2. Frontend Implementation:

## a. File Upload Interface:

- A file selector allows the user to choose the video file for upload.

## b. Detecting Network Information:

- JavaScript is used to detect the user's network information: navigator.connection.effectiveType and navigator.connection.downlink  
- Fallback to default network values if the browser does not support the API.

## c. Pre-signed Upload URL Request:

- The frontend sends a request to the backend for a pre-signed S3 URL.  
- Network information is included in the request.

## d. Upload File to S3:

- Using the pre-signed URL, the video is uploaded to S3.

## e. User Notification:

- The user is informed that the video is being processed.

# 3. Backend Implementation:

## a. Pre-signed Upload URL API:

- An API generates a pre-signed S3 URL for secure file upload.  
- Network information is stored in DynamoDB alongside the file key.

## b. S3 Event Notification:

- S3 is configured to trigger a Lambda function when a video is uploaded.

# 4. Lambda Function Video Processing:

- FFmpeg is bundled as a Lambda Layer to process videos.  
- The Lambda function determines the appropriate resolution based on the user's network.  
- The video is downloaded from S3, processed, and uploaded back to S3.

# 5. Providing a Download Link:

- The frontend polls or uses WebSocket to check the processing status.  
- Once processing is complete, the frontend requests a pre-signed download link.

# 6. Security and Permissions:

- Follow the principle of least privilege for IAM roles.  
- Use pre-signed URLs for secure S3 access.  
- Ensure S3 bucket policies are correctly configured.

# 7. Performance and Cost Optimization:

- Increase Lambda memory for faster processing.  
- Limit the video size or processing time to avoid Lambda timeouts.  
- Test the system under different conditions to validate performance.

Our application is working for:  
1. Video Upload: Users upload videos to Amazon S3 through the front-end.  
2. Network Detection: The front-end detects the user's network conditions and sends relevant information.  
3. Automatic processing: S3 upload triggers AWS Lambda, which uses FFmpeg to convert the video resolution according to the network conditions.  
4. Storage and Download: The processed video is stored back to S3 and the front-end provides a download link to the user.  
5. Security & Optimization: Uses pre-signed URLs and the principle of least privilege to optimize performance and cost.