COSC2639 Assignment 3

Matthew Kellock - s3812552

Links

- The live application URL: https://cosc26313.kellock.com.au
- Repository URL: https://github.com/mkellock/cosc2639-a3
- QuickSight board: https://ap-southeast-2.quicksight.aws.amazon.com/sn/accounts/723465791019/dashboards/6ada968b-c2ea-47ab-828b-01174cf682c8?directory_alias=cosc26313

Summary

This assignment submission is a short form video sharing site that reduces the resolution of video files after upload.

Introduction

What are the motivations behind your idea?

One of the issues with modern social media platforms is the concept of followers and influencers rather than a focus on content and sharing. This site removes the concept of a log-in, only allowing uploading and up/downvoting of video files.

What it does?

After uploading video (at this time, a short video), the site will transcode the file and reduce/standardise the resolution. You can upvote and downvote video files, the system figures out your location based off your IP, and on the backend offers metrics on content uploaded.

Why is it required?

This site focuses on content rather than followers, removing the concept of celebrity and influencers. The uploader video location allows viewers to see the origins of a video, providing necessary context around a video's creator.

How it can be used as real-life application?

The site requires more work, including video descriptions, categorisation, sorting, and filtering, however with a month or two of work (and maybe someone with better eye for UI than me), could turn this application into a functional social media platform.

With exception of the RDS MySQL database, all backend infrastructure and code are infinitely horizontally scalable, utilising technologies like CloudFront, ECS Fargate, S3, and Lambda. To make the application more scalable and suitable for large volumes of traffic, the MySQL database could be replaced with DynamoDB or augmented with Elastic Cache/CloudFront for the API to eliminate/reduce load on the MySQL database, respectively.

The advantages/positive/new things of your application.

One of the big advantages a platform like this is to reduce the emphasis of celebrity from a social media platform, there is no ability to identify uploaders (aside from within the content), this addresses several issues with social media sites including but not limited to:

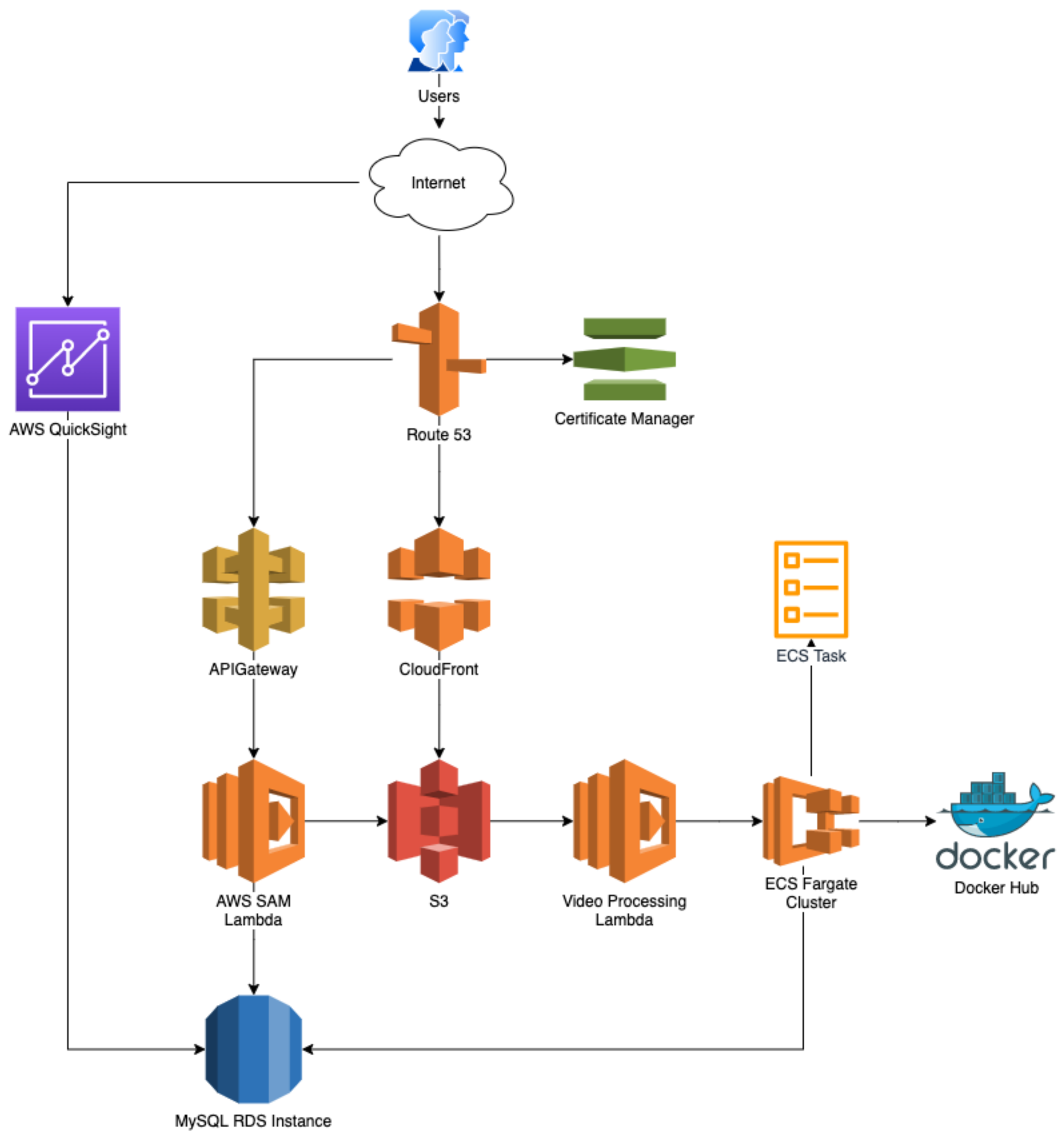
- Fake news and echo chambers
- Lacklustre content
- Celebrity idolisation and toxic media

Related work

The closest social media platform I know of is Reddit, where sub-Reddits built around community and culture rather than followers and celebrity offer a different take on Facebook, Instagram and TicTok. One issue with Reddit though is that sub-Reddits can build toxic communities and echo chambers, these communities are an alternative for influencers on other platforms but have the same effect, my project focuses on content rather than individual/community popularity, solely relying on the video's merits and engagement.

System Architecture

Architectural Diagram



System Description

1. Data ingress happens through Route53 and is directed to either CloudFront or API Gateway depending on the type of request.
2. In the case of static content (web pages or videos), requests are directed to CloudFront which will grab content from S3 when there is a cache miss
3. In the case of an API call, requests are directed at API Gateway.
4. The Lambda function behind API Gateway saves video data to S3 (raw video content) and RDS MySQL database (video and IP information queried from an IP info service)
5. Once a video is put in the S3 `/uploads` folder, a Lambda is triggered to queue an ECS Task on an ECS Fargate cluster, the cluster pulls an image from Docker Hub to convert the video. Once the video is converted, its state is updated in the RDS MySQL database and made available in the application.
6. Video metrics are made available from the RDS Database in QuickSight

Description of dataset and APIs

Database

There is only one table (videos) in the database:

Column	Data Type	Primary Key	Not Null	Default	Description
id	VARCHAR(36)	✓	✓	UUID()	Id of the video
ip	VARCHAR(16)				IP address of the uploader
city	VARCHAR(255)				City of the uploader
region	VARCHAR(255)				Region of the uploader (e.g., state)
country	VARCHAR(2)				Country of the uploader
upload_date	DATETIME		✓	CURRENT_TIMESTAMP	The upload date
upvotes	INT		✓	0	Number of upvotes
downvotes	INT		✓	0	Number of downvotes
is_processing	TINYINT		✓	1	Weather the video is processing

API

Below are the API methods and their schema

Endpoint	Method	Request/Response	Description
get_videos	GET	ID (GUID) Upvotes (Integer) Downvotes (Integer) City (String) Region (String) Country (String)	Return the videos ready to view
upvote/{ID}	POST	None	Upvotes a video
downvote/{ID}	POST	None	Downvotes a video
upload	POST	IP (GUID) contents (string)	Uploads a video to the platform

Developer Manual

The project tries to use as much infrastructure as code as possible, below are the deployment steps:

1. The base infrastructure uses an AWS CloudFormation template, and can be deployed via the CLI, console, or other supported tools. One adjustment to the `cosc26393ProcessVideo` lambda code needs to be made prior to deployment, the subnets are hardcoded and need to be adjusted to suit your environment's public subnet configuration.
2. The API is available in the `./api` folder of the code and is deployed via AWS Serverless Application Model (SAM), deploy this via the CLI tooling into the same account as the CloudFormation template.
3. After deployment of the API, create a on the newly deployed SAM Lambda titled `api-AspNetCoreFunction-[unique code here]`, and add the following environment variables:
   - **Name:** BUCKET_NAME
     Value: The S3 bucket name deployed in step 1
   - **Name:** CONNECTION_STRING
     Value: The MySQL connection string to the RDS instance deployed in step 1 excluding database name

4. The DNS records are unique to each deployment requirements, so the steps may vary depending on where your domain and DNS records are hosted, and where your SSL certificates are stored. Below are the steps for AWS services:
   - Under Route53, register a domain
   - Under Certificates Manager, register a SSL certificate
   - Under API Gateway, create a custom domain name and point it to your SAM deployment
   - Under Route 53, create an alias A record and point it to your CloudFormation deployment
   - Under Route 53, create an alias A record and point it to your API Gateway custom domain
5. In the React UI is available in the `./frontend` folder, adjust the references in the `./frontend/src/App.tsx` file to point to the A record for your API created in step 4.
6. Build the front end by running `npx build`. Using the console or CLI, copy the newly created files in the `./frontend/build` to the S3 bucket created in step 1.
7. Upon the first video submission, the table schema will be built out by the application.
8. In your QuickSight account, add a data source to the RDS instance created in step 1, add metrics as necessary.

User manual

- The application is simple to use, select a (small) video file and hit the submit button, in a few minutes the video will appear on the site.
- Hit the refresh button on the top right of the page to refresh content.
- A theme button is available to toggle light/dark mode
- The up arrow on each video is to upvote
- The down arrow on each video is to downvote

References

The two main sources of information were:

- AWS Documentation for information on API's, technology usage, etc. - https://docs.aws.amazon.com
- Stack Overflow for information on resolving undocumented problems using the AWS platform - https://stackoverflow.com