Part 1 – Architecture design

Design consideration

We need to take into account some considerations to build the infra for these functionalities:

- Ensure scalability: API GW allows routing and handling multiple requests without bottlenecks.
- Maintain modularity: each module such us Jobs, SQL and Files, are separate components to decouple responsibilities.
- High availability: Deploy components in a hosting with failovers to ensure their availability.
- Efficiency: Jobs handle background tasks like file transformations, enabling non-blocking operations for the API.
- Security: No users have been mentioned, but I guess not everyone can access to all the files

How would you build it

To build the architecture I would use AWS services.

API GW

For API GW I would use API Gateway. This service allows you to define endpoints so the App can access to them and execute specific tasks. It can be linked with Cognito to handle authentication and authorization.

Jobs

For Jobs I would use Lambda. This service is lightweight and cost reduced as they just charge you for use and you don't need to have an EC2 instance or something similar running the whole day.

Lambdas are flexible and scale really well. They can be triggered under some conditions on other AWS services. This will allow to run a lambda right after the CSV has been uploaded to Jobs so it can be processed and stored in the SQL table.

Another approach I have used for this in the past for solving something like this was to attach and SQS queue to the S3 bucket so that queue was ingesting a Worker from Elastic Beanstalk. This was interesting because of the dead letter queue that generates the Worker. So, if there are specific requirements for it, in case of a file failing to be

formatted, additional processing can be added to the workflow. Without further information I would opt for the Lambda approach as is cheaper and more scalable.

SQL

For SQL I would use Amazon RDS to create the table to store the information from the transformed CSV file. There are no more details to pick an specific database so I'll opt for PostgreSQL as tends to scale better.

Files

For files I would use S3 for sure. It is cheap and flexible. It can be linked with lambdas to execute some jobs. There is also integration with Cloudfront as a CDN to distribute the content around the globe and keep it closer and faster to access for the users of App. S3 allows to create presigned URLs to share the objects stored in files in a secure way.

Another files system that I would use would be Glacier to keep long lasting files, depending on how long we would need to store files with more than a year of life and how frequent would the users will access them.

Additional services

Not in the diagram, but would be interesting to consider AWS Cloudfront to handle logs from the different services for audibility and debugging.

Something extra to consider would be Route 53 and the certificates.

I'm mentioning these because the cost of them will be added to the platform.

How the messaging and orchestration will work

API Gateway will handle most of the communication part. As it will be the router to the different endpoints to execute every job. Depending on the implementation in the jobs part, and additional SQS queue will be added to the communication flow.

Workflow 1

API GW gets the request, if it is a valid request, a Lambda will process the request storing the file in S3. Storing a file in the bucket for the CSVs will trigger another lambda that will handle the transformation into the SQL table. When the processing is done, I'm assuming here the success message is a push notification for the app. It will push a message into an SNS queue connected to Firebase to send the success notification to the user.

In case of some kind of spinner or progress bar is needed to be shown to the user, it will be needed to build a web socket so the success message is pushed through it and the frontend is refreshed to show the message there or update the interface completing the progress bar.

Workflow 2

API Gateway will get the request from the App, will check the authorization and start a Lambda to get the file from SRV-GW to store the CSV file into the right S3 bucket. After that a presigned URL will be returned through API Gateway response to the APP so the user can access and download the file they were requesting.

Which environment would you choose and why

My choice is a cloud environment due to scalability, managed services and cost-effectiveness (AWS as I mentioned before). Docker-based deployment would be interesting to have in local the closest conditions while developing.

Kubernetes seems to be an overkill here and will just complicate the development process. API Gateway and Lambda are quite flexible and easy to use to don't need it.

In terms of programming languages. I would choose Typescript with NodeJS for the lambdas as it is fast for processing requests and works really well along with Lambdas.

Which security elements can be used

For the authorization we will use OAuth for for API Gateway connected with Cognito to check on authentication.

IAM service will be necessary to handle users and roles in the application and properly handle authorization.

For the API security AWS provides some interesting services. WAF can be used as a firewall and to filter some requests. API Gateway and lambda handle their own security as you can define to what resources they have access either through ARN role or group. AWS secrets store will be use to secure keys used inside the jobs.

I almost forget about it as it tends to be created at the beginning of the infra creation process, but a proper VPC must be created to secure the requests between the different components. This will match with the pink area.