

VARETAS GEORGIOS REPORT

N2G Evaluation Project for Cloud Architect Position

1. General Overview

At first, I started reading the assignment carefully in order to understand the general purpose of the solution as well as to examine the several parts of it. I tried to extract the essential requirements, decompose it into small “microservices” and have a first impression on the AWS Services that I could use. I came up with some small architectures and how these ones could help me build the solution.

The next step was to search for AWS reference architectures related to Cloud Analytics. The key features of the project like ‘real-time’ or ‘registration of the user’ offered a list of possible AWS Services I could leverage.

2. Defining Architectural Designs

The most difficult part was to opt for the best solution based on the five pillars of interests as mentioned in the assignment. My final architectural decision along with the reasons are shown below:

S3 bucket

Reason: Amazon S3 bucket is used widely as Data Lakes as a result of the low cost of storing data and the options that they offer. Some features that they offer include S3 Lifecycle policies and transitions into S3 Glacier. In addition, S3 events can be quite useful in event-driven architectures.

AWS IoT Core

Reason: AWS IoT offers a variety of services. Choosing AWS IoT Core for connecting the smart meters seemed the best solution. Even though the only reason we use that service is to capture the real-time data stream of the desired granularity, it is possible that we will use AWS IoT Analytics in the future.

AWS SQS

Reason: SQS is a fully managed service that helps us to decouple the different parts of processing the data. Consumers such as AWS Lambdas or EC2 instances can receive the messages, process them and delete them.

Amazon DynamoDB

Reason: By examining the structure of the data inputs, it was obvious that a NoSQL database should store the profile of a user. DynamoDB integrates with other AWS Services and also leverages IAM Roles and policies.

Amazon Cognito

Reason: Amazon Cognito offers simple and secure sign-up, sign-in operations. It scales to million of users and support sign-in with social known identity provides such as Google and Facebook.

Amazon API Gateway + AWS Lambdas vs ALB + EC2

The biggest challenge was to decide the API implementation in the AWS Cloud. Serverless Architecture offer:

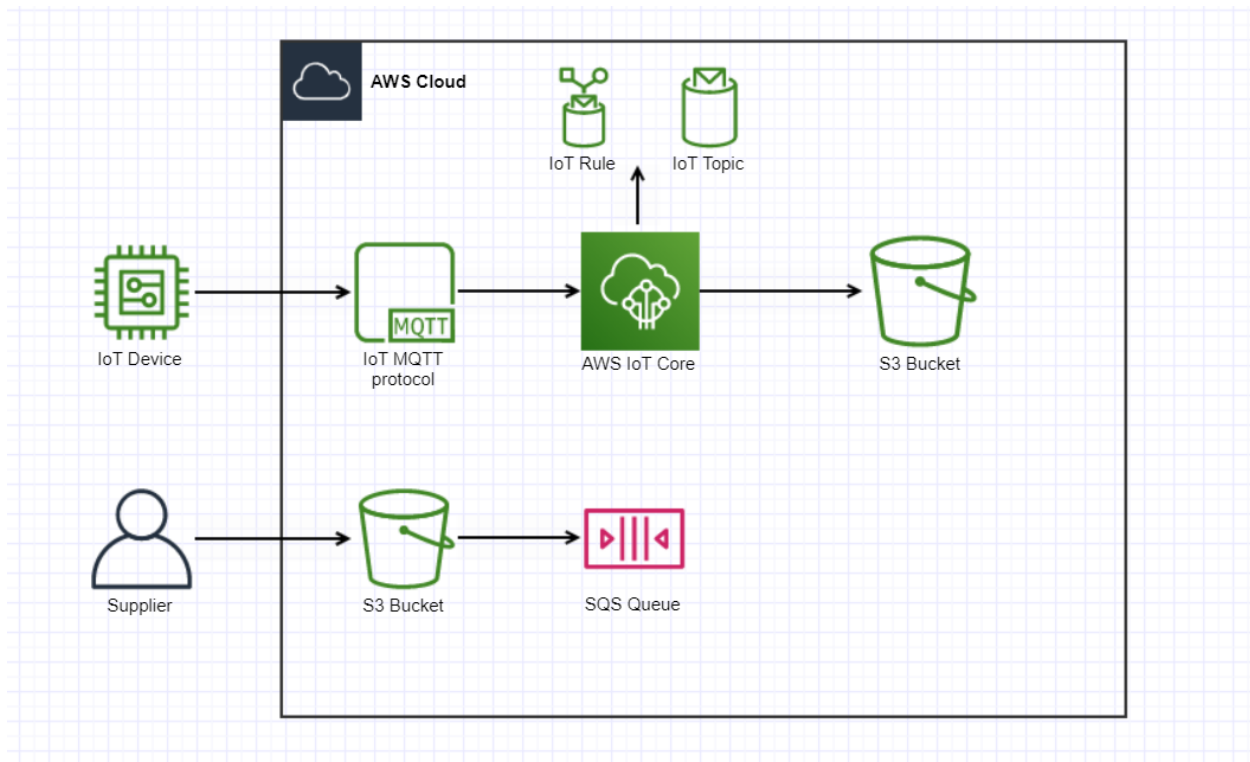
- Reliability
- Efficiency
- Maintenance
- High Availability

In terms of security, we refer to the shared responsibility model where AWS is responsible for the security of the cloud and we are responsible for the security in the cloud. In Serverless solutions, AWS takes care of the security concepts while we take care of the security of our applications. A solution with VPC, EC2, ALB, Databases and several security layers seems to be more secure but it needs careful configuration and continuous monitoring of the data flows.

In terms of cost efficiency, a serverless architecture seems the best solution taking into account the total cost of ownership (TCO) as a sum of development cost + infrastructure cost + maintenance cost. If the number of customers keeps increasing, we should re-evaluate the situation as the cost of the serverless solution would be really high.

As a result of the above analysis, I chose to implement a serverless architecture.

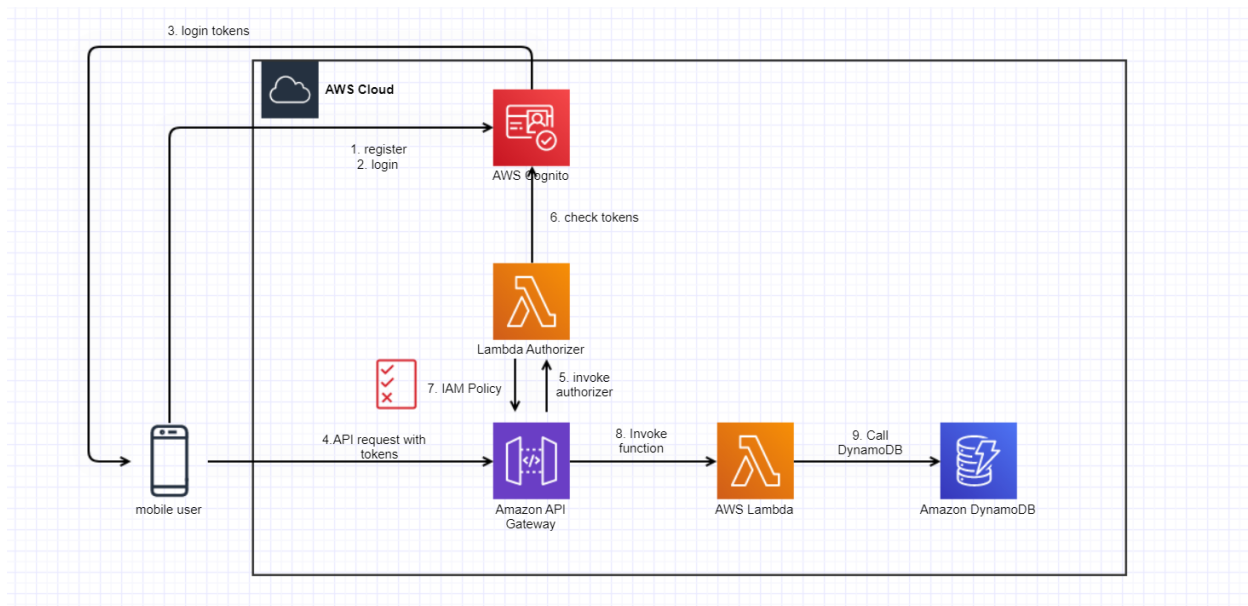
3. Architectures



The above image shows the AWS infrastructure for the IoT Stack.

We connect the IoT device to AWS IoT Core and send the real time data to S3 buckets. It is vital that we keep a format for the s3 objects as {ClientID}/{YYYYMMDD}/timestamp in order to be grouped for the analysis.

In addition, we created the S3 bucket where the Supplier can upload the .csv files. Once a file is uploaded to that bucket, it sends a metadata message to the SQS in order to notify that the daily consumption of an installation is available for processing. Also here we should ask the supplier to keep a file format {YYYYMMDD}/{ClientID}.csv



The above image shows the AWS infrastructure for the mobile app.

A customer can sign-up and sign-in through AWS Cognito. After receiving the id token, they can get or update their profiles in the mobile app. API Gateway invokes a lambda authorizer which checks the tokens and returns a policy. If the policy allows this API call, API Gateway invokes the lambda function in order to satisfy the request.

4. Requirements not fulfilled

It is clear that not all the requirements are fulfilled by the above architectures due to lack of time. In this section I will try to give some comments in order to explain my understanding on those requirements.

- The requirement for the overview of the daily reports.

This was an additional operation that mobile application should offer. The results of the analysis would be stored in an S3 bucket and the user could authenticate and receive a presigned URL in order to get it.

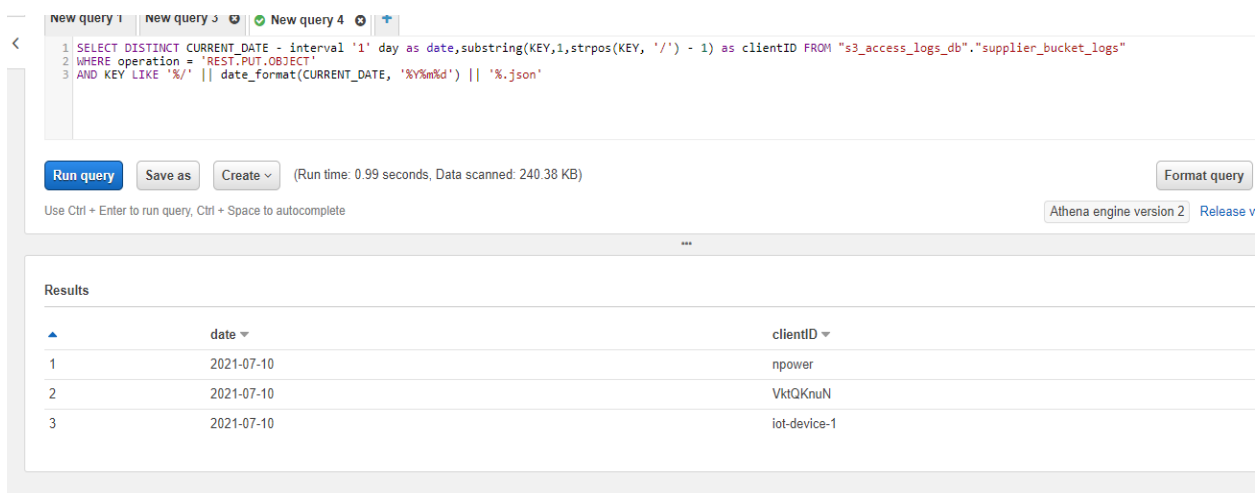
- The requirement for the delivery of the real-time data to analysis.

We could configure a daily event in Amazon EventBridge to invoke a lambda in order to poll the S3 bucket for available data streams from previous day and publish a metadata message to the SQS.

- The requirement for the summary report for each day.

This report should contain the information about the usage of our service.

Regarding the number of installations that provide real-time data, we can enable the S3 Access Logs to monitor the access in iot and supplier bucket. An example of that process is shown in the following image:



The screenshot shows the Amazon Athena console interface. At the top, there are tabs for 'new query 1', 'new query 3', and 'New query 4'. The active query is displayed in a text area:

```
1 SELECT DISTINCT CURRENT_DATE - interval '1' day as date, substring(KEY,1, strpos(KEY, '/') - 1) as clientID FROM "s3_access_logs_db"."supplier_bucket_logs"
2 WHERE operation = 'REST.PUT.OBJECT'
3 AND KEY LIKE '%/' || date_format(CURRENT_DATE, '%Y%m%d') || '%.json'
```

Below the query editor, there are buttons for 'Run query', 'Save as', and 'Create'. A status bar indicates '(Run time: 0.99 seconds, Data scanned: 240.38 KB)'. A 'Format query' button is also present. A hint below the buttons says 'Use Ctrl + Enter to run query, Ctrl + Space to autocomplete'. The bottom right corner shows 'Athena engine version 2' and a 'Release v' link.

The 'Results' section displays a table with two columns: 'date' and 'clientID'. The table contains three rows of data:

	date	clientID
1	2021-07-10	npower
2	2021-07-10	VktQKnuN
3	2021-07-10	iot-device-1

Accordingly, we can analyze the access logs of the bucket with the reports for the requests that are made to its objects.

Other information can be obtained by the AWS CloudTrail logs. We can track update operations to the Clients DynamoDB table and output valuable results for the summary report.

5. Potential Services to be added

- **AWS IoT Greengrass**

Perform machine learning inference locally on devices using models trained in cloud.

- **AWS Lake Formation**

It is a fully managed service related to data lakes. It also automates many manual steps.

- **Amazon EMR**

Processing vast amounts of data using open source tools such as Apache Spark, Apache Hive.