

Architecture

- The MQTT software stream payloads to the AWS IoT Core message broker to a specific MQTT topic.
- The AWS IoT rule activates upon detecting a payload within its designated topic. The rule is configured with an Amazon Kinesis Data Firehose and Dynamodb action.
- Amazon Kinesis Data Firehose buffers the incoming payloads, then it will deliver them to the data store once either size or time thresholds are met, whichever comes first.
- Before delivering to the destination, Kinesis Data Firehose execute an AWS Lambda function to transform the payloads in batches. In this case, the lambda function will add and attributed named "processedAt" with current timestamp as the value and then return the modified payload to Kinesis Data Firehose.
- The modified payloads are compressed and put into an Amazon S3 bucket.
- API Gateway proxies request to S3 and Dynamodb
- Frontend application deployed on EC2 instance and call the API Gateway endpoint

Information

The code is in this link: <https://github.com/mecode12/iot-s3-app/tree/master>

Task

S3

Create an S3 bucket with the following configuration:

- Block all public access.
- Bucket Versioning: Disable

Lambda

Create and deploy the lambda function with the following configuration:

- Lambda runtime: python3.12
- Timeout: 1 minutes
- Memory size: 256 MB
- Ephemeral storage: 512 MB
- Architecture: arm64

Kinesis

Create Kinesis Data Firehose with the following configuration:

- Source: Direct PUT
- Destination: S3
- Enable transform source records with AWS Lambda:
 - Choose the Lambda function you have previously deployed.
 - Version or alias: `$$$LATEST`
 - Buffer size: 1 MB
 - Buffer interval: 60 seconds
- New line delimiter: Enabled
- S3 bucket prefix: data/
- S3 bucket error output prefix: error/
- S3 bucket and S3 error output prefix time zone: UTC
- S3 buffer hints:
 - Buffer size: 1 MB
 - Buffer interval: 60 seconds
- Compression for data records: GZIP

Dynamodb

- Create dynamodb with partition key **msg** type string

IOT

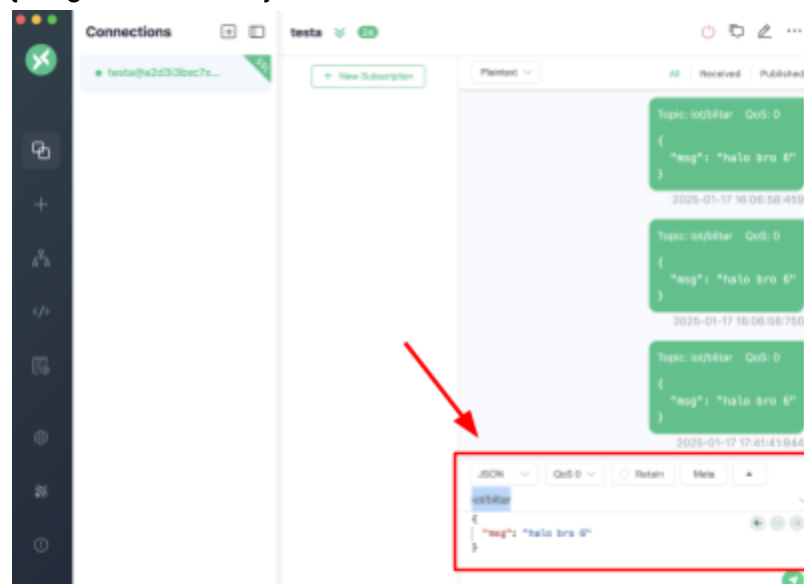
Configure IOT thing and IOT Rule with the following details:

- Use the “iot-blitar” as thing name
- Use “iot/blitar” as topic
- Create iot rule to send the iot data to the kinesis data firehouse
 - Use “\n” as separator in iot rule
- Create iot rule to send the iot data to dynamodb
 - Use dynamodbv2

MQTTx


- Download mqttx on your laptop: <https://mqttx.app/>
- This application will use to simulate the iot devices. We will send data to the aws iot using this application
- Connect to the aws iot core endpoint
 - Use protocol mqtt
 - Use port 8883
 - Refer to this docs: <https://mqttx.app/docs/get-started>
- Send test data from mqttx to the AWS IOT
 - Topic: iot/blitar
 - message

`{"msg": "test bebas"}`



Result

- Make sure after sending test message from mqttx, you have data in the s3
- If you don't have error folder, you are ok. But if you don't have data folder, you are not ok!

<input type="checkbox"/>	Name
<input type="checkbox"/>	 data/
<input type="checkbox"/>	 error/

- Make sure you also have data in dynamodb. Try to send multiple different message to have more than one data in dynamodb

Items returned (2)

<input type="checkbox"/>	msg (<i>String</i>)
<input type="checkbox"/>	halo bro 6
<input type="checkbox"/>	halo bro 7

API Gateway

Configure and deploy API Gateway with the following details:

- Use rest api
- Create resource named data-s3
 - Create get method
 - Integration type aws service
 - Aws service: s3
 - Action name: ListResource
 - Execution role: labrole
- Create resource named data-dynamodb
 - Create post method
 - Integration type aws service
 - Aws service: dynamodb
 - Action name: Scan
 - Execution role: labrole
 - Edit integration request
 - Mapping template
 - Content-type = application/json
 - Template body = `{"TableName": "Nama Tabel Dynamodb"}`

VPC

Configure VPC with the following details:

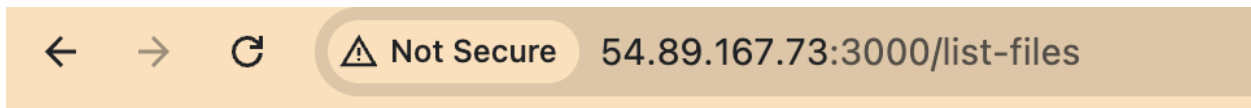
- Use the prefix 172.16.0.0/23
- Create two public subnet and two private subnet.
 - 172.16.0.0/25 pubsubnet-az-a
 - 172.16.0.128/25 pubsubnet-az-b
 - 172.16.1.0/25 privsubnet-az-a
 - 172.16.1.128/25 privsubnet-az-b
- Create two routetable, one for private and one for public
- Create internet gateway and nat gateway
- Make sure instance in private subnet is able to access internet.

Security Group

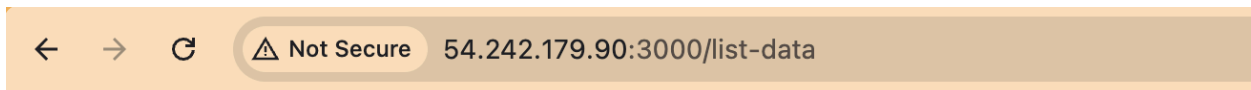
- Create two security groups
 - sg-lb will be used for load balancer
 - sg-ec2 will be used for ec2 instance
- Configure the security group as secure as possible

EC2

- Deploy the application on ec2 instance
- You need to use auto scaling group and load balancer
- Use launch template to define the ec2 configuration
- Use t2.micro and amazon linux 2023
- **YOU ARE PROHIBITED TO SSH TO THE INSTANCE, OR YOU WILL DISQUALIFIED**
- Configure auto scaling to have minimum 2 instance and maximum 4 instance.
- Make sure to deploy the ec2 instance into private subnet and different AZ
- Configure the load balancer to listen on port 8080, so you will access the load balancer by **endpoint_loadbalancer:8080**
- **Tips: the application is only accessible from /list-files, /list-data, and /status.**
 - You can't access the root url, for example ec2_ip:3000 or endpoint_loadbalancer:8080. If you access that, you will get error 404 not found
 - You can access ec2_ip:3000/list-files, ec2_ip:3000/list-data, ec2_ip:3000/status, or endpoint_loadbalancer:8080/list-files, endpoint_loadbalancer:8080/list-data, endpoint_loadbalancer:8080/status



Your bucket is: test-rosid-blitar



Messages: halo bro 6, halo bro 7

