Data Analytics on Big Data Using AWS Serverless

Abishek Arumugam Thiruselvi
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
abishekarumugam@icloud.com

Kausthuba Vanam
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
kausthubavanam90@gmail.com

***Abstract*— To implement the concepts, methods, and communications for data models and the resources they represent using the data sets from the GitHub directory. This assignment aims to develop a client/server application and incorporate two data communication methods.**

Keywords—AWS, Function, Lambda, S3, Serverless, runtime, NodeJS, API, Stack, Postman.

I. Introduction

A model for convenient on-demand network access to a shared pool of reconfigurable computing resources, such as networks, servers, storage, applications, and services, is known as cloud computing. It allows quick provisioning and release of these resources with little management work or service provider interaction [1].

NIST has three cloud service modes: SaaS, PaaS, and Iaas. In this assignment, we are using Infrastructure as a Service using AWS.

Resources used from the AWS infrastructure are AWS Lambda, API Gateway, CodeCommit, CodeBuild, CodePipeline, Cloud Formation, S3 and IAM.

The NodeJS16.x runtime is used in VS Code IDE for application development and deployed in a Serverless framework.

II. Data Model Design

The application model is a Microservices-based serverless model. Serverless computing features the cloud provider allocating machine resources as needed and managing the servers on behalf of its users. Fig. 1. Shows the data model of the application developed.

A. *CloudFormation*

The stack of all modules required is created through CloudFormation.

B. *CI/CD*

The pipeline is done through CodePipeline. The code pipeline uses the resources below,

- CodeCommit as its repository
- CodeBuild as its build module

C. *CloudWatch*

CloudWatch is used for monitoring and debugging purposes. Any changes in the stack created by CloudFormation will get logged in the CloudWatch.

D. *API Gateway*

It satisfies the REST API functionalities by communicating with the Lambda functions, and the roles are set in IAM.

E. *S3*

These buckets are NoSQL-based DB which gets and puts objects in them. Here we use "get" to get the datasets from S3.

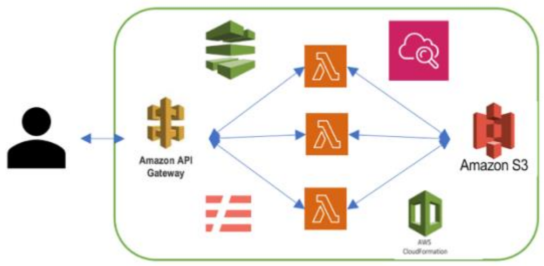


Fig. 1. Data Implementation Model

III. Data Serialization/ De-Serialization Methods

A. *Protobuf*

- Protocol buffers are just another way of serializing your messages into binary format.
- The serialized output is a packed sequence of bytes that attempts to conserve space. Sending bytes over the wire takes much less time than transferring the same data in XML or JSON forms. Deserializing the data after it has been serialised with protobuf is even quicker.

B. *JSON*

- JavaScript Object Notation (JSON) is an open-standard file format, and data interchange format that employs text humans can read to store and transport data objects made up of arrays and attribute-value pairs (or other serializable values).
- Data must be byte strings to be transmitted or stored in a file, yet complex things are rarely in this format. These complicated objects can be turned into byte strings by serialization. Deserialization is the receiver that must extract the original object from the byte strings once sent.

IV. TECHNICAL IMPLEMENTATIONS

1. *POST Operation*

Using API Gateway, the REST/ POST operation is performed with “Content-Type” as “application/json”, and the request parameters are passed as json body in the post request. The JSON body can be found in “event.json” in the code repository [2]. Fig. 2. Represents the data communication throughout the application.

2. *Event Operation*

The sent POST request from the endpoint [3] calls the lambda function, passes the request parameters as an event, and carries out the operation.

3. *Lambda Function*

3.1. Libraries

The external libraries like protobuf, jwt and structjson, which are not available in the inbuilt lambda functions, are added externally using AWS lambda layer functionality. The layer is called to the lambda function using the serverless.yml file through arn (Amazon Resource Names).

3.2. JWT Decode

The event body is parsed as JSON and stored in the file. In handler.js, lines 28:29 decodes the jwt sent from the post and logged. From the event body, we get the following details "workLoadMetric", "benchMarkType", "batch unit", "batchSize": "1", "batched": "1", and "rwfID".

3.3. Protobuf Operation

Line 50:55, line 155:168 and line 170:171 encodes/decode using the protobufjs library and perform structural conversion of JSON to proto and log the result to CloudWatch.

3.4. Data Analytics

The Average of the workLoadMetric is computed and logged. The process happens from line 101:141.

3.5. Exceptions

The 4XX/5XX exceptions are handled using try/catch blocks and are logged. Unhandled exceptions are mostly returned in CloudWatch to perform debugging.

3.6. Return

The result is returned in line 178 and returned in post response.

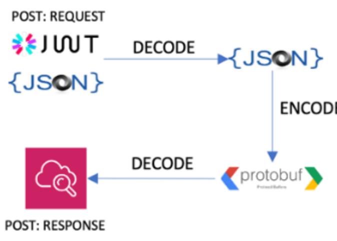


Fig. 2. Data Communication

V. CLOUD DEPLOYMENT

Serverless and CloudFormation help in the formation of the stack and helps in cloud deployment. This application is pipelined to auto-deploy all the time when commits are done. The serverless microservices-based architecture sets the infrastructure using the “serverless.yml” file.

A. *CodeCommit*

The code repository [2] has two branches, master and dev. Any change in the master branch will trigger the CodePipeline to perform CI/CD.

B. *CodeBuild*

The CodeBuild uses the “buildspec.yml” file as a node to start deploying whenever new commits are introduced in CodeCommit.

C. *CodePipeline*

Integrates CodeCommit aa s repository, CodeBuild with “buildspec.yml” as build node, and we select no web deployment as our application is serverless.

After the CI/CD is complete, the build logs will have an endpoint, and the same endpoint can be viewed in API Gateway.

VI. TESTING

Testing of the application is done using,

- Postman
- serverless-offline plugin
- Lambda test functions.

VII. INSTRUCTIONS TO RUN THE APPLICATION

Steps to install, run and test the application. Prerequisites are AWS login and npm installed on the local machine.

STEP 1. Install AWS CLI in the local terminal with the necessary user access [4].

STEP 2. Configure AWS CLI for CodeCommit [5].

STEP 3. Create a layer with the required library for running the application, i.e., protobuf and jwt [6].

STEP 4. In the local terminal, run the command “npm install -g serverless”

STEP 5. In the local terminal, run the command, “sls create -t aws-nodejs -p <project-name>”

STEP 6. Run “npm init -y”

STEP 7. Run “npm install --save aws-sdk protobuf jwt serverless-offline”

STEP 8. Put the files “handler.js”, “buildspec.yml”, “userpackage.proto” and “serverless.yml”

a. To test offline, continue these steps.

   i. Run “npm install serverless-offline” in the project folder.

   ii. Run “sls offline”

iii. Use the endpoint generated in postman with "event.json" as postparmaters.

iv. Postman gives valid results in the response body.

STEP 9. Create a CodeCommit repository in AWS

STEP 10. Run "git init"

STEP 11. Run "git add ."

STEP 12. Run "git commit -am "first commit"".

STEP 13. Run "git remote add origin <repo url>".

STEP 14. Run "git push --set-upstream origin"

STEP 15. Create a pipeline with the below steps [7].

STEP 16. Add CodeCommit as a repository.

STEP 17. Create a CodeBuild using environment variables as "buildspec.yml" and ENV_NAME=prod-env".

STEP 18. Select "skip stage" in the deployment stage, as it's serverless.

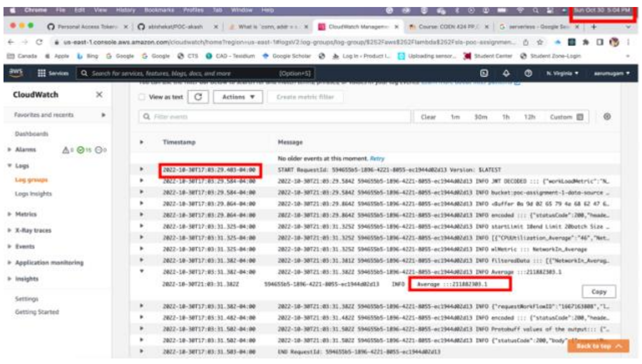
STEP 19. CodePipeline starts building, and an endpoint will be generated for the application.

STEP 20. Add the data sets to the S3 bucket created.

STEP 21. Copy the "event.json" from the repository and use postman or API Gateway or the lambda function generated to test the application functionality.

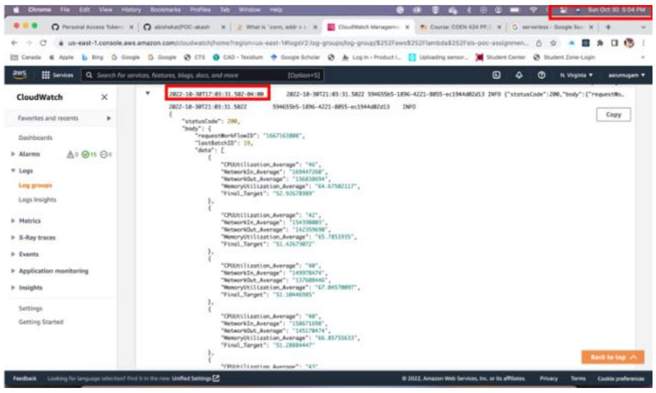
VIII. RESULTS

One main difficulty while testing the application is an "Internal Server Error", as shown in Fig. 3. This error is due to the CloudFront. As per AWS documentation and the StackOverflow developer's comments, we need to buy a domain and route through route53 to resolve this error. Instead, we logged all the details in CloudWatch logs to test the functionality. For results viewing purposes, we have attached build logs and CloudWatch logs in the repository [2] for scrutiny.



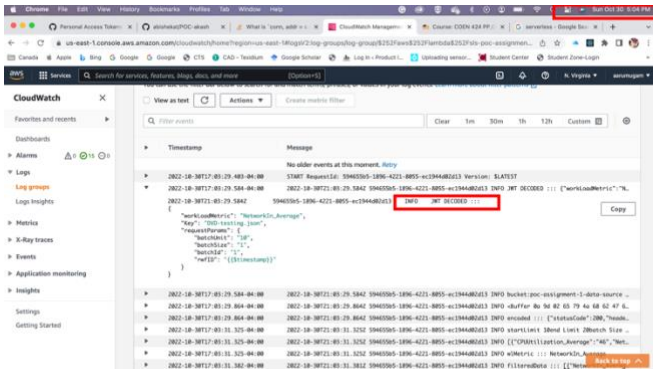
```
Timestamp                       Message
                                No older events at this moment. Retry
2022-10-30T17:03:29.403-04:00   START RequestId: 594655b5-1896-4221-8055-ec1944d02d13 Version: $LATEST
2022-10-30T17:03:29.584-04:00   2022-10-30T21:03:29.584Z 594655b5-1896-4221-8055-ec1944d02d13 INFO JWT DECODED ::: {"workLoadMetric":"N…
2022-10-30T17:03:29.584-04:00   2022-10-30T21:03:29.584Z 594655b5-1896-4221-8055-ec1944d02d13 INFO bucket:poc-assignment-1-data-source …
2022-10-30T17:03:29.864-04:00   2022-10-30T21:03:29.864Z 594655b5-1896-4221-8055-ec1944d02d13 INFO <Buffer 0a 9d 02 65 79 4a 68 62 47 6…
2022-10-30T17:03:29.864-04:00   2022-10-30T21:03:29.864Z 594655b5-1896-4221-8055-ec1944d02d13 INFO encoded ::: {"statusCode":200,"heade…
2022-10-30T17:03:31.325-04:00   2022-10-30T21:03:31.325Z 594655b5-1896-4221-8055-ec1944d02d13 INFO startLimit 10end Limit 20batch Size …
2022-10-30T17:03:31.325-04:00   2022-10-30T21:03:31.325Z 594655b5-1896-4221-8055-ec1944d02d13 INFO [{"CPUUtilization_Average":"46","Net…
2022-10-30T17:03:31.325-04:00   2022-10-30T21:03:31.325Z 594655b5-1896-4221-8055-ec1944d02d13 INFO wlMetric ::: NetworkIn_Average
2022-10-30T17:03:31.382-04:00   2022-10-30T21:03:31.381Z 594655b5-1896-4221-8055-ec1944d02d13 INFO filteredData ::: [{"NetworkIn_Averag…
2022-10-30T17:03:31.382-04:00   2022-10-30T21:03:31.382Z 594655b5-1896-4221-8055-ec1944d02d13 INFO Average :::211882303.1
2022-10-30T21:03:31.382Z        594655b5-1896-4221-8055-ec1944d02d13   INFO   Average :::211882303.1
2022-10-30T17:03:31.382-04:00   2022-10-30T21:03:31.382Z 594655b5-1896-4221-8055-ec1944d02d13 INFO {"requestWorkFlowID":"1667163808","l…
2022-10-30T17:03:31.482-04:00   2022-10-30T21:03:31.482Z 594655b5-1896-4221-8055-ec1944d02d13 INFO encoded ::: {"statusCode":200,"heade…
2022-10-30T17:03:31.502-04:00   2022-10-30T21:03:31.502Z 594655b5-1896-4221-8055-ec1944d02d13 INFO Protobuff values of the output::: {"…
2022-10-30T17:03:31.502-04:00   2022-10-30T21:03:31.502Z 594655b5-1896-4221-8055-ec1944d02d13 INFO {"statusCode":200,"body"
2022-10-30T17:03:31.503-04:00   END RequestId: 594655b5-1896-4221-8055-ec1944d02d13
```

Fig. 3. Data Analytics



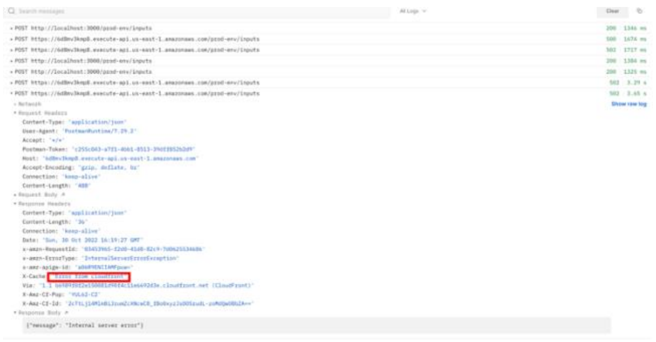
```
2022-10-30T17:03:31.502-04:00   2022-10-30T21:03:31.502Z 594655b5-1896-4221-8055-ec1944d02d13 INFO {"statusCode":200,"body":{"requestWo…
2022-10-30T21:03:31.502Z        594655b5-1896-4221-8055-ec1944d02d13   INFO
{
    "statusCode": 200,
    "body": {
        "requestWorkFlowID": "1667163808",
        "lastBatchID": 19,
        "data": [
            {
                "CPUUtilization_Average": "46",
                "NetworkIn_Average": "169447268",
                "NetworkOut_Average": "156838694",
                "MemoryUtilization_Average": "64.67502117",
                "Final_Target": "52.92678989"
            },
            {
                "CPUUtilization_Average": "42",
                "NetworkIn_Average": "154398003",
                "NetworkOut_Average": "142359690",
                "MemoryUtilization_Average": "65.7851935",
                "Final_Target": "51.42679072"
            },
            {
                "CPUUtilization_Average": "40",
                "NetworkIn_Average": "149978474",
                "NetworkOut_Average": "137608446",
                "MemoryUtilization_Average": "67.04570097",
                "Final_Target": "51.10446985"
            },
            {
                "CPUUtilization_Average": "40",
                "NetworkIn_Average": "158671698",
                "NetworkOut_Average": "145170474",
                "MemoryUtilization_Average": "66.85755633",
                "Final_Target": "51.28884447"
            },
            {
                "CPUUtilization_Average": "43",
```

Fig. 4. Final Response



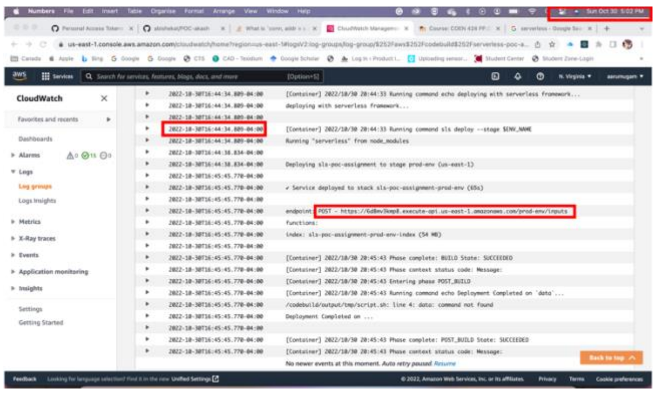
```
Timestamp                       Message
                                No older events at this moment. Retry
2022-10-30T17:03:29.403-04:00   START RequestId: 594655b5-1896-4221-8055-ec1944d02d13 Version: $LATEST
2022-10-30T17:03:29.584-04:00   2022-10-30T21:03:29.584Z 594655b5-1896-4221-8055-ec1944d02d13 INFO JWT DECODED ::: {"workLoadMetric":"N…
2022-10-30T21:03:29.584Z        594655b5-1896-4221-8055-ec1944d02d13   INFO   JWT DECODED :::
{
    "workLoadMetric": "NetworkIn_Average",
    "Key": "DVD-testing.json",
    "requestParams": {
        "batchUnit": "10",
        "batchSize": "1",
        "batchId": "1",
        "rwfID": "{{$timestamp}}"
    }
}
2022-10-30T17:03:29.584-04:00   2022-10-30T21:03:29.584Z 594655b5-1896-4221-8055-ec1944d02d13 INFO bucket:poc-assignment-1-data-source …
2022-10-30T17:03:29.864-04:00   2022-10-30T21:03:29.864Z 594655b5-1896-4221-8055-ec1944d02d13 INFO <Buffer 0a 9d 02 65 79 4a 68 62 47 6…
2022-10-30T17:03:29.864-04:00   2022-10-30T21:03:29.864Z 594655b5-1896-4221-8055-ec1944d02d13 INFO encoded ::: {"statusCode":200,"heade…
2022-10-30T17:03:31.325-04:00   2022-10-30T21:03:31.325Z 594655b5-1896-4221-8055-ec1944d02d13 INFO startLimit 10end Limit 20batch Size …
2022-10-30T17:03:31.325-04:00   2022-10-30T21:03:31.325Z 594655b5-1896-4221-8055-ec1944d02d13 INFO [{"CPUUtilization_Average":"46","Net…
2022-10-30T17:03:31.325-04:00   2022-10-30T21:03:31.325Z 594655b5-1896-4221-8055-ec1944d02d13 INFO wlMetric ::: NetworkIn_Average
2022-10-30T17:03:31.382-04:00   2022-10-30T21:03:31.381Z 594655b5-1896-4221-8055-ec1944d02d13 INFO filteredData ::: [{"Netw…
```

Fig. 5. JWT decoded



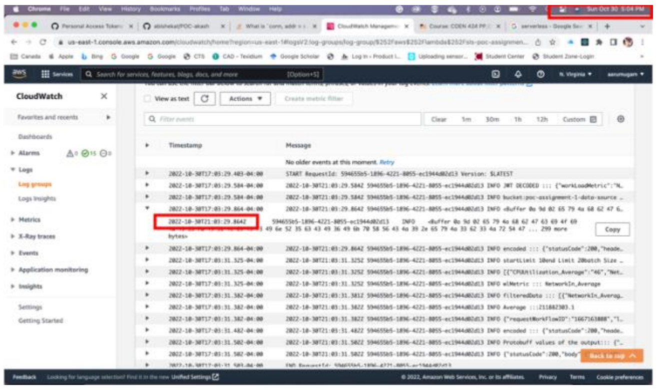
```
POST http://localhost:3000/prod-env/inputs                                            200  1346 ms
POST https://6dBmv3kmp8.execute-api.us-east-1.amazonaws.com/prod-env/inputs            500  1674 ms
POST https://6dBmv3kmp8.execute-api.us-east-1.amazonaws.com/prod-env/inputs            502  1717 ms
POST http://localhost:3000/prod-env/inputs                                            200  1384 ms
POST http://localhost:3000/prod-env/inputs                                            200  1325 ms
POST https://6dBmv3kmp8.execute-api.us-east-1.amazonaws.com/prod-env/inputs            502  3.29 s
POST https://6dBmv3kmp8.execute-api.us-east-1.amazonaws.com/prod-env/inputs            502  3.65 s
  Network                                                                                Show raw log
  Request Headers
    Content-Type: "application/json"
    User-Agent: "PostmanRuntime/7.29.2"
    Accept: "*/*"
    Postman-Token: "c255c043-a7f1-40b1-8513-3f6ff8526d9"
    Host: "6dBmv3kmp8.execute-api.us-east-1.amazonaws.com"
    Accept-Encoding: "gzip, deflate, br"
    Connection: "keep-alive"
    Content-Length: "480"
  Request Body
  Response Headers
    Content-Type: "application/json"
    Content-Length: "36"
    Connection: "keep-alive"
    Date: "Sun, 30 Oct 2022 16:19:27 GMT"
    x-amzn-RequestId: "83453965-f2d0-41d0-82c9-7d0625534686"
    x-amzn-ErrorType: "InternalServerErrorException"
    x-amz-apigw-id: "a0kM9ENI1AMFpuw="
    X-Cache: "Error from cloudfront"
    Via: "1.1 ne4093012e150081d96f4c11e6492d3e.cloudfront.net (CloudFront)"
    X-Amz-Cf-Pop: "YUL62-C2"
    X-Amz-Cf-Id: "2cTtLj14MlmBiJzumZcXNcwC8_IBo0xyzJu005zudL-zoMdQw0BbZA=="
  Response Body
    {"message": "Internal server error"}
```

Fig. 6. Internal Server Error Postman but logs have results.



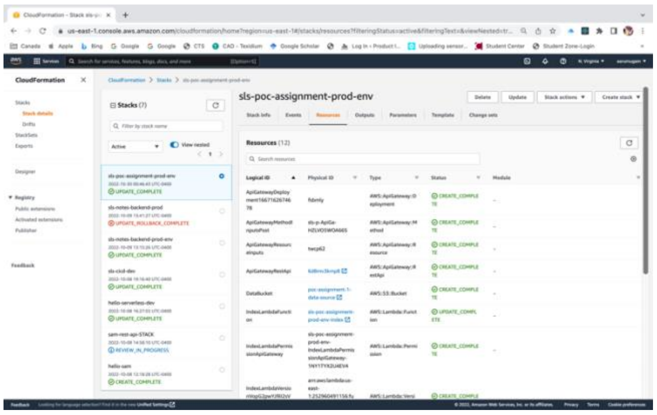
```
2022-10-30T16:44:34.809-04:00   [Container] 2022/10/30 20:44:33 Running command echo deploying with serverless framework...
2022-10-30T16:44:34.809-04:00   deploying with serverless framework...
2022-10-30T16:44:34.809-04:00
2022-10-30T16:44:34.809-04:00   [Container] 2022/10/30 20:44:33 Running command sls deploy --stage $ENV_NAME
2022-10-30T16:44:34.809-04:00   Running "serverless" from node_modules
2022-10-30T16:44:38.834-04:00
2022-10-30T16:44:38.834-04:00   Deploying sls-poc-assignment to stage prod-env (us-east-1)
2022-10-30T16:45:45.770-04:00
2022-10-30T16:45:45.770-04:00   ✔ Service deployed to stack sls-poc-assignment-prod-env (65s)
2022-10-30T16:45:45.770-04:00
2022-10-30T16:45:45.770-04:00   endpoint: POST - https://6dBmv3kmp8.execute-api.us-east-1.amazonaws.com/prod-env/inputs
2022-10-30T16:45:45.770-04:00   functions:
2022-10-30T16:45:45.770-04:00   index: sls-poc-assignment-prod-env-index (54 MB)
2022-10-30T16:45:45.770-04:00
2022-10-30T16:45:45.770-04:00   [Container] 2022/10/30 20:45:43 Phase complete: BUILD State: SUCCEEDED
2022-10-30T16:45:45.770-04:00   [Container] 2022/10/30 20:45:43 Phase context status code: Message:
2022-10-30T16:45:45.770-04:00   [Container] 2022/10/30 20:45:43 Entering phase POST_BUILD
2022-10-30T16:45:45.770-04:00   [Container] 2022/10/30 20:45:43 Running command echo Deployment Completed on `date`...
2022-10-30T16:45:45.770-04:00   /codebuild/output/tmp/script.sh: line 4: date: command not found
2022-10-30T16:45:45.770-04:00   Deployment Completed on ...
2022-10-30T16:45:45.770-04:00
2022-10-30T16:45:45.770-04:00   [Container] 2022/10/30 20:45:43 Phase complete: POST_BUILD State: SUCCEEDED
2022-10-30T16:45:45.770-04:00   [Container] 2022/10/30 20:45:43 Phase context status code: Message:
                                No newer events at this moment. Auto retry paused. Resume
```

Fig. 7. Build Logs



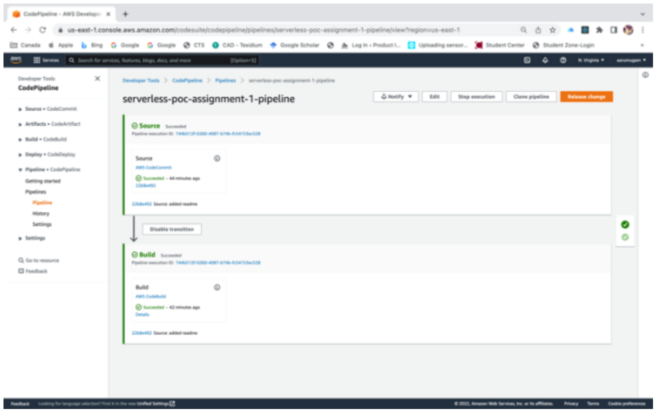
```
Timestamp                       Message
                                No older events at this moment. Retry
2022-10-30T17:03:29.403-04:00   START RequestId: 594655b5-1896-4221-8055-ec1944d02d13 Version: $LATEST
2022-10-30T17:03:29.584-04:00   2022-10-30T21:03:29.584Z 594655b5-1896-4221-8055-ec1944d02d13 INFO JWT DECODED ::: {"workLoadMetric":"N…
2022-10-30T17:03:29.584-04:00   2022-10-30T21:03:29.584Z 594655b5-1896-4221-8055-ec1944d02d13 INFO bucket:poc-assignment-1-data-source …
2022-10-30T17:03:29.864-04:00   2022-10-30T21:03:29.864Z 594655b5-1896-4221-8055-ec1944d02d13 INFO <Buffer 0a 9d 02 65 79 4a 68 62 47 6…
    2022-10-30T21:03:29.864Z    594655b5-1896-4221-8055-ec1944d02d13    INFO    <Buffer 0a 9d 02 65 79 4a 68 62 47 63 69 4f 69
    … 49 6e 52 35 63 43 49 36 49 6b 70 58 56 43 4a 39 2e 65 79 4a 33 62 33 4a 72 54 47 ... 299 more
    bytes>
2022-10-30T17:03:29.864-04:00   2022-10-30T21:03:29.864Z 594655b5-1896-4221-8055-ec1944d02d13 INFO encoded ::: {"statusCode":200,"heade…
2022-10-30T17:03:31.325-04:00   2022-10-30T21:03:31.325Z 594655b5-1896-4221-8055-ec1944d02d13 INFO startLimit 10end Limit 20batch Size …
2022-10-30T17:03:31.325-04:00   2022-10-30T21:03:31.325Z 594655b5-1896-4221-8055-ec1944d02d13 INFO [{"CPUUtilization_Average":"46","Net…
2022-10-30T17:03:31.325-04:00   2022-10-30T21:03:31.325Z 594655b5-1896-4221-8055-ec1944d02d13 INFO alMetric ::: NetworkIn_Average
2022-10-30T17:03:31.382-04:00   2022-10-30T21:03:31.381Z 594655b5-1896-4221-8055-ec1944d02d13 INFO filteredData ::: [{"NetworkIn_Averag…
2022-10-30T17:03:31.382-04:00   2022-10-30T21:03:31.382Z 594655b5-1896-4221-8055-ec1944d02d13 INFO Average :::211882303.1
2022-10-30T17:03:31.382-04:00   2022-10-30T21:03:31.382Z 594655b5-1896-4221-8055-ec1944d02d13 INFO {"requestWorkFlowID":"1667163808","I…
2022-10-30T17:03:31.482-04:00   2022-10-30T21:03:31.482Z 594655b5-1896-4221-8055-ec1944d02d13 INFO encoded ::: {"statusCode":200,"heade…
2022-10-30T17:03:31.502-04:00   2022-10-30T21:03:31.502Z 594655b5-1896-4221-8055-ec1944d02d13 INFO Protobuff values of the output::: {"…
2022-10-30T17:03:31.502-04:00   2022-10-30T21:03:31.502Z 594655b5-1896-4221-8055-ec1944d02d13 INFO {"statusCode":200,"body"…
```

Fig. 8. Protobuf Encoding



sls-poc-assignment-prod-env

Fig. 9. Stack creating resources through CloudFormation

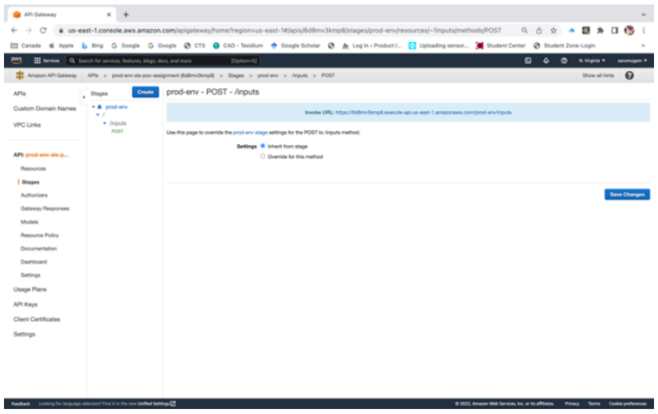


serverless-poc-assignment-1-pipeline

Source: Succeeded

Build: Succeeded

Fig. 10. CI/CD pipeline by CodePipeline



prod-env - POST - /inputs

Invoke URL: https://6d8mv3kmp8.execute-api.us-east-1.amazonaws.com/prod-env/inputs

Use this page to override the prod-env stage settings for the POST to /inputs method.

Settings: Inherit from stage / Override for this method

Fig. 10. API Gateway endpoint

IX. Resources

The code is available for access in the public S3 bucket [8].

X. Conclusions

We were able to implement all the specifications in the assignment description and incorporate additional functionalities like,

1. Language binding using "yml."
2. CI/CD integration using CodePipeline
3. NoSQL S3 connections.
4. Testing of the application.

References

[1] Zhang, Qi, Lu Cheng, and Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges." *Journal of internet services and applications* 1.1 (2010): 7-18.

[2] https://git-codecommit.us-east1.amazonaws.com/v1/repos/serverless-poc-assignment-1

[3] https://6d8mv3kmp8.execute-api.us-east-1.amazonaws.com/prod-env/inputs

[4] https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

[5] https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-https-unixes.html

[6] https://docs.aws.amazon.com/lambda/latest/dg/configuration-layers.html

[7] https://docs.aws.amazon.com/codepipeline/latest/userguide/pipelines-create.html

[8] https://s3.console.aws.amazon.com/s3/buckets/poc-assignment-1-data-source?region=us-east-1