

**CSCI 5410**

**Serverless Data Processing (Summer 2023)**

## MASTER OF APPLIED COMPUTER SCIENCE

Assignment-3 (Part B):

Name: **Jainil Sevalia |** Banner Id : **B00925445 |** Email:[**jn498899@dal.ca**](mailto:jn498899@dal.ca)

**Table of figures:**

[Figure 1 : Architecture of application 3](#_Toc140653352)

[Figure 2 : "sample-data-b00925445" s3 bucket creation (AWS Console) 3](#_Toc140653353)

[Figure 3 : Successfully created s3(sample-data-b00925445) bucket. (AWS Console). 4](#_Toc140653354)

[Figure 4 : Creating "tag-b00925445" bucket for Named Entity text files. 4](#_Toc140653355)

[Figure 5: Successfully created tag-b00925445 bucket. 5](#_Toc140653356)

[Figure 6 : Creating Dynamo DB for Final output. 5](#_Toc140653357)

[Figure 7 : List of DynamoDB Table. 6](#_Toc140653358)

[Figure 8 : Successfully Dynamo DB table created. 6](#_Toc140653359)

[Figure 9 : Creating ECR repository for extractfeatures lambda function. 7](#_Toc140653360)

[Figure 10 : Code for extractfeatures lambda function(1) 8](#_Toc140653361)

[Figure 11 : Code for extractfeatures lambda function(2) 9](#_Toc140653362)

[Figure 12 : Docker image build, tagged and pushed to ECR for extractfeatures lambda function. 10](#_Toc140653363)

[Figure 13 : Successfully pushed docker image of extractfeatures lambda function to ECR. 10](#_Toc140653364)

[Figure 14 : Successfully extractfeatures lambda function created. 11](#_Toc140653365)

[Figure 15 : Trigger for extractfeatures lambda function. 12](#_Toc140653366)

[Figure 16 : Creating ECR repository for accessDB lambda function. 13](#_Toc140653367)

[Figure 17 : Code for accessDB lambda function 15](#_Toc140653368)

[Figure 18 : Docker image build, tagged and pushed to ECR for accessDB lambda function. 16](#_Toc140653369)

[Figure 19 : Successfully pushed docker image of access DB lambda function to ECR. 16](#_Toc140653370)

[Figure 20 : Successfully accessDB lambda function created. 17](#_Toc140653371)

[Figure 21 : Trigger for accessDB lambda function. 18](#_Toc140653372)

[Figure 22 : Testing: Running Script to upload all the files tos3 bucket. 19](#_Toc140653373)

[Figure 23 : Testing: Successfully uploaded all the files to s3 bucket. 19](#_Toc140653374)

[Figure 24: Testing: All files successfully uploaded to s3 bucket(sample-data-b00925445).\ 20](#_Toc140653375)

[Figure 25 : Testing: Content of uploaded file in sample-data-b00925445. 20](#_Toc140653376)

[Figure 26 : Testing: Processed files are stored in the tags-b00925445 bucket. 21](#_Toc140653377)

[Figure 27 : Testing: Content of processed files. These files contain Named Entity and its frequency. 21](#_Toc140653378)

[Figure 28 : Testing: Entity and Frequency stored in DynamoDB. 22](#_Toc140653379)

**Event-driven serverless application using AWS Lambda.**

**Architecture of application:**

A screen shot of a video game

Description automatically generated

Figure : Architecture of application

**Creating architecture of serverless application**

* Created “SampleDataB00925445” s3 bucket for uploading raw txt files.

A screenshot of a computer

Description automatically generated

Figure : "sample-data-b00925445" s3 bucket creation (AWS Console)

* Empty “sample-data-b00925445” bucket shown below after creation.

A screenshot of a computer

Description automatically generated

Figure : Successfully created s3(sample-data-b00925445) bucket. (AWS Console).

* Creating second bucket: “tag-b00925445” bucket for Named Entity text files.

A screenshot of a computer

Description automatically generated

Figure : Creating "tag-b00925445" bucket for Named Entity text files.

* Successfully created "tag-b00925445" bucket for processed data.

A screenshot of a computer

Description automatically generated

Figure : Successfully created tag-b00925445 bucket.

* Creating Dynamo Db Table for storing final Named entity and it’s frequency.

A screenshot of a computer

Description automatically generated

Figure : Creating Dynamo DB for Final output.

A screenshot of a computer

Description automatically generated

Figure : List of DynamoDB Table.

* Successfully created Dynamo DB table with default configurations.

A screenshot of a computer

Description automatically generated

Figure : Successfully Dynamo DB table created.

* Creating ECR Repository(extractfeatures) for deploying image of “extractfeature” lambda function.

A screenshot of a computer screen

Description automatically generated

Figure : Creating ECR repository for extractfeatures lambda function.

* extractfeatures Lambda code.

A screen shot of a computer program

Description automatically generated

Figure : Code for extractfeatures lambda function(1)

Continue…..

A screen shot of a computer program

Description automatically generated

Figure : Code for extractfeatures lambda function(2)

* Pushing this code to ECR repository. For that I created Docker image. Run the script which build docker image, tagged docker image and pushed it to ECR.

A screenshot of a computer program

Description automatically generated

Figure : Docker image build, tagged and pushed to ECR for extractfeatures lambda function.

* Successfully pushed docker image of extractfeatures lambda function to ECR.

A screenshot of a computer

Description automatically generated

Figure : Successfully pushed docker image of extractfeatures lambda function to ECR.

* Created extractfeatures lambda function using docker image from ECR.

A screenshot of a computer

Description automatically generated

Figure : Successfully extractfeatures lambda function created.

* Adding trigger event for extractfeatures lambda function. If any object added in to sample-data-b00925445 s3 bucket this lambda will trigger.

A screenshot of a computer

Description automatically generated

Figure : Trigger for extractfeatures lambda function.

* Creating ECR Repository(accessdb) for deploying “accessDB” lambda function.

A screenshot of a computer

Description automatically generated

Figure : Creating ECR repository for accessDB lambda function.

* accessDB Lambda code.

A screenshot of a computer program

Description automatically generated

Figure : Code for accessDB lambda function

* Pushing this code to ECR repository. For that I created Docker image. Run the script which build docker image, tagged docker image and pushed it to ECR.

A screenshot of a computer program

Description automatically generated

Figure : Docker image build, tagged and pushed to ECR for accessDB lambda function.

* Successfully pushed docker image of accessDB lambda function to ECR.

A screenshot of a computer

Description automatically generated

Figure : Successfully pushed docker image of access DB lambda function to ECR.

* Created accessDB lambda function using docker image from ECR.

A screenshot of a computer screen

Description automatically generated

Figure : Successfully accessDB lambda function created.

* Adding trigger event for accessDB lambda function. If any object added in to tag-b00925445 s3 bucket this lambda will trigger.

A screenshot of a computer

Description automatically generated

Figure : Trigger for accessDB lambda function.

**TESTING:**

* Running Script that would upload each file in the tech folder to sample-data-b00925445 s3 bucket.

A screenshot of a computer program

Description automatically generated

Figure : Testing: Running Script to upload all the files tos3 bucket.

* Successfully uploaded all the files to s3 bucket.

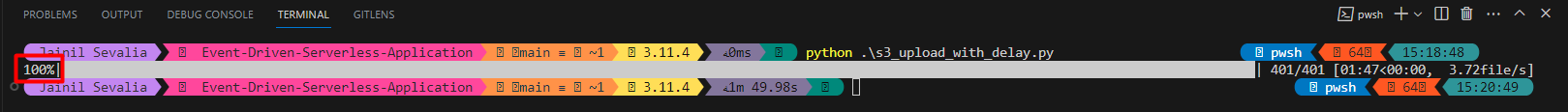


Figure : Testing: Successfully uploaded all the files to s3 bucket.

* sample-data-b00925445 s3 bucket successfully got all files(401 files).

A screenshot of a computer

Description automatically generated

Figure : Testing: All files successfully uploaded to s3 bucket(sample-data-b00925445).\

* Content of uploaded file.

A screenshot of a computer

Description automatically generated

Figure : Testing: Content of uploaded file in sample-data-b00925445.

* During the upload of raw files to sample-data-b00925445, Lambda triggers and fetch Named entities form text. Processed files are stored in the tags-b00925445 bucket.

A screenshot of a computer

Description automatically generated

Figure : Testing: Processed files are stored in the tags-b00925445 bucket.

* Content of processed files. These files contain Named Entity and its frequency.

A screenshot of a computer

Description automatically generated

Figure : Testing: Content of processed files. These files contain Named Entity and its frequency.

* When Processed files are stored in tags-b00925445 s3 buckets accessDB lambda will trigger and fetch Named Entity and its frequency and store Entity as a key and frequency as value in Dynamo DB table.

A screenshot of a computer

Description automatically generated

Figure : Testing: Entity and Frequency stored in DynamoDB.