

Configuration Manual

MSc Research Project MSc in Cloud Computing

Vijayanand Somavaram Student ID: x22147802

> School of Computing National College of Ireland

Supervisor: Aqeel Kazmi

National College of Ireland Project Submission Sheet School of Computing



Student Name:	Vijayanand Somavaram
Student ID:	x22147802
Programme:	MSc in Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Aqeel Kazmi
Submission Due Date:	31/01/2024
Project Title:	Configuration Manual
Word Count:	1020
Page Count:	14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

<u>ALL</u> internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	S.Vijayanand
Date:	30th January 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).Attach a Moodle submission receipt of the online project submission, to
each project (including multiple copies).You must ensure that you retain a HARD COPY of the project, both for

your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Vijayanand Somavaram x22147802

This project can be divided into two phases

- Phase One: Data collection phase
- Phase One: Prediction phase

Due to resource limitations for this research project, please note that all the AWS services should be set in the same AWS Region.

1 Collecting Latency Data

In this phase, a methodology to collect the time taken to execute queries by DynamoDB will be collected. This methodology will include setting up AWS Services like EC2, API Gateway, Lambda, DynamoDB, and CloudWatch.

1.0.1 Step One: Create a Lambda function

Open the AWS console, select AWS Lambda from the search, and click on the "Create Lambda" option. Enter the name of the Lambda function, and select the runtime as Python 3.11 as shown in 1. Click on the Create option.



Figure 1: Create a lambda function

Remove the existing code from the code editor. From the submitted code artifacts copy the "lambdaPython" code onto the editor as shown in 2. Click On Deploy



Figure 2: Add the Python Code

1.1 Step Two: Create API Gateway

Search for API Gateway, from the console, and click on create. As shown in 3, select HTTP API and click on the Build option. Enter the name of the API, and then click on "review and create" as shown in 4. As Shown in 5 enter the name of a route, by clicking on the "Create" option. From 6, select the "POST" route. Then from the route details click on "Attach Integration". As shown in 7, click on the "create" button. Then on the next window as shown in 8, under the integration target select the integration "created from step one and save. From the left nav as shown in 9, select the "Stages" option. On the next window as shown in 10 enter the name of the stage and save the information. Then on the next window as shown in 11, click on the "CORS" option to deploy the API. From the left nav as shown in 12, select the "CORS" option to avoid CORS errors. On the next window as shown on 13 add the values and save them. From the left nav as shown in 14, select the name of your API, and the URLs under the stages for "your api name" is the API URL that will be used in later stages.

1.1.1 Step Three: Setup Web Application

Launch an EC2 instance from the AWS console, for this application "t2.micro" ubuntu instance will be enough. Connect to the instance through SSH client. For this, an application called MobaXterm has been used. It is a tool for remote computing. Follow the instructions as displayed on SSH client and connect to EC2 instance as shown on 15. Then install apache2 server on the instance using the commands

- "sudo apt-get install apache2"
- "cd var"
- "cd www"
- "cd html"



Figure 3: Create API Gateway

Create an API			
Create and configure integrations			
Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.			
Integrations (0) Info			
Add integration			
API name			
An HTTP API must have a name. This name is cosmetic and does not have to be unique; you will use the API's ID (generated later) to programmatically refer to this API.			
Cancel Review and Create Next			

Figure 4: Api Gateway

Open another window on Mobaxterm. Download the web application zip on the local folder where the ".pem" file is present. Enter the following commands

- "scp -i /path/to/your/.pemkey /copy/from/path.zip user@server:/copy/to/path"
- "sudo apt-get install unzip"
- "unzip yourfile.zip"
- "cd assets"
- "cd js"
- "sudo nano server.js"
- Copy and paste the API URL from step 2 in the place of URL as shown in 16

Routes	
Routes for test_api	Create
Q Search	

Figure 5: Api Gateway

Routes for test_api Create	Route details POST / (ID: wunbt60)
▼ / POST	Authorization Authorizers protect your API against unauthorized requests. Routes with no authorization attached are open. No authorizer attached to this route. Attach authorization Integration The integration is the backend resource that this route calls when it receives a request. No integration attached to this route. Attach integration

Figure 6: Api Gateway

Attach integrations to routes	Manage integrations	
Integrations for test_api		Create
Q Search		



Attach this integration to a	route	
Q POST /		×
Integration target		
Integration type		
Lambda function		▼
Integration details		
Integration target Choose the Lambda function that API Gate		
AWS Region	Lambda function	
us-east-1 🔻	Q arn:aws:lambda:us-east-1:631982972732:function:testFunction X	
Advanced settings		
Description - optional		

Figure 8: Api Gateway

- press "CTRL + O" press enter
- press "CTRL + x" to exit

With this the application is up, click on the EC2 Instance id, and open the public address, change the URL on the browser from "https" to "http". The application should be as shown in 17

▼ Develop	
Routes	
Authorization	
Integrations	
CORS	
Reimport	
Export	
▼ Deploy	
Stages	
 Monitor 	
Metrics	
Logging	
▼ Protect	
Throttling	

Figure 9: Api Gateway

Stage details	
Name	
test_stag	
Description	
Enter description	

Figure 10: Api Gateway

1.1.2 Step 4: Setup DyanmoDB:

From the AWS Console enter DyanmoDB, and click on the "Create table" button. Enter the name of the table, "test partition key", under table settings select "Customize settings", and under "Read/write capacity settings" select "Provisioned" For both Read capacity and Write capacity turn off Auto Scaling. For now, enter minimum capacity under read capacity units as 51 and 408, scroll to the bottom, and click on the "Create Table" button as shown in images 18 and 19

From the Lambda function create in Step one, open the editor to connect it with

	Stage: test_stag ▼	Deploy
	Delete	Edit
;	ast updated	

Figure 11: Api Gateway



Figure 12: Api Gateway

DyanamoDB. To do so, enter the name of the table for the variables shown in 20. In this case, enter the same name for both the "table" and "writeTable" variables. The connection now is completed. The DynamoDB table now created is capable of handling 24KB or read-and-write workloads. To test it click on "Read Payload size 24kb" or "Write payload size 24kb" from the app. By doing so the application will send continuous requests to DynamoDB for 250 seconds.

To check the logs and query times from the lambda console click on the "Monitor" option and then click on "View cloudwatch logs" and then by clicking on log files the

Cross-Origin Resource Sharing				
Configure CORS info CORS allows resources from different domains to be loaded by browsers. If you configure CORS for an API, API Gateway ignores CORS headers returned from your backend integration. See our CORS documentation for more details.				
Access-Control-Allow-Origin		Access-Control-Allow-Headers		
Enter a value for Allowed Origins	Add	Enter a value for Allowed Headers	Add	
·×		·×		
Access-Control-Allow-Methods		Access-Control-Expose-Headers		
Choose Allowed Methods	•	Enter a value for Exposed Headers	Add	
POST X				
Access-Control-Max-Age		Access-Control-Allow-Credentials		
0		NO		
			Cancel	Save

Figure 13: Api Gateway

APIs	API details					Edit
Custom domain names						
VPC links	API ID					
	hfkejrmfz4			2023-12-14		
API: test_api(hfkeirmfz4)	Description		Default and colot			
	No Description		Enabled			
▼ Develop						
Routes	Stages for test_a	api (2)				
Authorization						
Integrations	Q. Find resources					
CORS	Stage name	Invoko IIRI		Attached deployment	Auto deploy	Lart undated
Reimport	Jtage name	INVOLE OIL		Attached deptoyment	Auto deploy	Last updated
Export	<u>\$default</u>			xd7hdr	enabled	2023-12-14
▼ Daplay	test_stag		t-1.amazonaws.com/test_stag	9xl9al	disabled	2023-12-14
Stage						
Judges	- (0)					
▼ Monitor	Tags (0)					Manage tags
Metrics	Q Find resources					
Logging						
▼ Protect	Key 🔺 Va					

Figure 14: Api Gateway

data can be viewed as shown in 21

1.1.3 Conclusion

This way the experiment was conducted for 49 KB, and 98 KB for 2000, 3000, 4000, and 5000 requests per minute. Data was collected from the logs and was formatted for the machine learning model training.

2 Prediction Phase

After collecting the data from phase one, the data was formatted which have been submitted in the artifacts by name, "24KBdata" and "49KBdata" and "98KBdata".

2.0.1 Step one: Model training:

As shown in images 22 and 23, the notebook code can be downloaded from artifacts.



Figure 15: Mobaxterm Application



Figure 16: Mobaxterm Application

2.0.2 step two: Integration and Predictions:

For Predictions, a web application was created using the Streamlit framework. The code can be downloaded from the artifacts. Just like the steps followed in phase one, step three, an ec2 instance should be launched. The ec2 security port should be open to port: 80. The downloaded code should be uploaded to ec2 after creating a folder on ec2 by any SSH tool. Enter the following commands

• "sudo mkdir newFolder"

Click The Buttons To	Send Requests With Par	yload Sizes To AWS	
Lambaa Serveriess Fi	unctions		
Read Payload Size 24KB	Read Payload Size 48KB	Read Payload Size 96KB	
Write Payload Size 24KB	Write Payload Size 48KB	Write Payload Size 96KB	

Figure 17: Web Application for data collection

Create table				
Table details Info DynamoDB is a schemaless database that requires only a table name	ne and a primary key when you create the table.			
Table name This will be used to identify your table.				
Enter name for table				
Between 3 and 255 characters, containing only letters, numbers, u	nderscores (_), hyphens (-), and periods (.).			
Partition key The partition key is part of the table's primary key. It is a hash value hosts for scalability and availability.	e that is used to retrieve items from your table and allocate data across			
Enter the partition key name	String v			
1 to 255 characters and case sensitive.				
Sort key - optional You can use a sort key as the second part of a table's primary key. T same partition key.	The sort key allows you to sort or search among all items sharing the			
Enter the sort key name	String			
1 to 255 characters and case sensitive.				

Figure 18: setup DynamoDB

- upload the code to this folder
- "cd newFolder"
- "sudo apt-get update"
- "python3 -m venv env"
- "cd env"
- "cd bin"

Auto scaling Info Dynamically adjusts provisioned throug	ghput capacity on your behalf in response to act	tual traffic patterns.
 On 		
⊖ Off		
Minimum capacity units	Maximum capacity units	Target utilization (%)
51	10	70
Write capacity Auto scaling Info Dynamically adjusts provisioned throug On Off	ghput capacity on your behalf in response to ac	tual traffic patterns.
Write capacity Auto scaling Info Dynamically adjusts provisioned throug On Off Minimum capacity units	ghput capacity on your behalf in response to act Maximum capacity units	tual traffic patterns. Target utilization (%)

Figure 19: setup DynamoDB

dynamodb= boto3.resource('dynamodb') table= dynamodb.Table('x22147802 data layer 24kb v1') writeTable= dynamodb.Table('x22147802_data_layer_write_intensive_v1')



- "source activate"
- "cd .."
- "cd .."
- "pip install" all the packages mentioned in image 24
- "streamlit run streamlit.py -server.port=80"

START RequestId: d50744df-e7ac-49cb-a4d3-8157151ea66a Version: \$LATEST						
63 writes per second at 4013 at 30 % for wcu novel config at 250 loops : 2.0737640857696533 seconds						
END RequestId: d50744df-e7ac-49cb-a4d3-8157151ea66a						
REPORT RequestId: d50744df-e7ac-49cb-a4d3-8157151ea66a Duration: 2094.31 ms Billed Duration: 2095 ms	Memory Size: 128 MB	Max Memory Used: 80 MB				
START RequestId: 78b2e7f8-c96d-4b29-9950-5c7b055b82a5 Version: \$LATEST						
63 writes per second at 4013 at 30 % for wcu novel config at 250 loops : 2.074413299560547 seconds						
END RequestId: 78b2e7f8-c96d-4b29-9950-5c7b055b82a5						
REPORT RequestId: 78b2e7f8-c96d-4b29-9950-5c7b055b82a5 Duration: 2095.09 ms Billed Duration: 2096 ms	Memory Size: 128 MB	Max Memory Used: 80 MB				

Figure 21: Cloudwatch



Figure 22: notebook

• Click the public URL of the Instance, change the URL from "https" to "http", and the application will run as shown in the image 25



Figure 23: Notebook



Figure 24: python env



Figure 25: Application