CLOUD SERVERLESS COMPUTING

21CS3281R/ 21CS3281A/ 21CS3281P

LAB WORKBOOK

STUDENT ID:

STUDENT NAME:

ACADEMIC YEAR: 2023-24

Table of Contents

1. **Session 01: Introductory Session ..... NA**
2. **Session 02: Build a standalone static web hosting on EC2 Linux machine #1 ..... #**
3. **Session 03: Build a Static web hosting on AWS S3 bucket with name KLUNIVERSITY by creating bucket policy for grant public read access with cloudfront #2 ..... #**
4. **Session 04: Implementation of Autoscaling to Manage more than two EC2 #3 ..... #**
5. **Session 05: Understand the basics of Lambda and Create your first lambda function in Python ,Java, Node.Js #4 ..... #**
6. **Session 06: Implement lambda to start and stop EC2 services #5 ..... #**
7. **Session 07: Integration of lambda with S3 for objects create event #6 ..... #**
8. **Session 08: Create an Amazon S3 bucket and upload a test file to your new bucket.#7 ..... #**
9. **Session 09: write Your Lambda function to retrieves information about this file when you upload it in s3 bucket #8 ..... #**
10. **Session 10: test the lambda function from the aws management console for java. #9 ..... #**
11. **Session 11: Inter region transfer of Table in DynamoDB Integration of Lambda with DynamoDB #10 ..... #**
12. **Session 12: Configuring a bucket for notifications SQS queue #11 ..... #**
13. **Session 13: Build serverless application using Athena Architecture #12 ..... #**
14. **Session 14: Implementation of Autoscaling to Manage two different target groups with each having atleast two target group members with load balancer and Deploy a Web Application #13 (Adv/Peer) ..... #**
15. **Session 15: implement power of maths calculator application using lambda functions #14 (Adv/Peer) ..... #**
16. **Session 16: List all the EC2 instances and Status across all regions in an AWS account using Python boto3 #15 (Adv/Peer) ..... #**
17. **Session 17: Create an SNS topic, subscribe an endpoint to the SNS Topic, publish a message to the SNS Topic, Check receipt of message, Delete the Subscription and the SNS topic #16 (Adv/Peer) ..... #**
18. **Session 18: Introduction of Kinesis as notification Services #17 (Adv/Peer) ..... #**
19. **Session 19: Build serverless application using Athena , S3 , Glue and Quicksight Architecture #18 (Adv/Peer) ..... #**
20. **Session 20: Build end to end serverless web application for Get and post method using CRUD operations #19 (Adv/Peer) ..... #**
21. **Session 21: Build Real time CICD Serveless web Application #20 (Adv/Peer) ..... #**

A.Y. 2023-24 LAB/SKILL CONTINUOUS EVALUATION

S.No	Date	Experiment Name	Pre-Lab (10M)	In-Lab (25M)			Post-Lab (10M)	Viva Voce (5M)	Total (50M)	Faculty Signature
				Program/ Procedure (5M)	Data and Results (10M)	Analysis & Inference (10M)				
1.		Introductory Session	-NA-							
2.		Build a standalone static web hosting on EC2 Linux machine #1								
3.		Build a Static web hosting on AWS S3 bucket with name KLUNIVERSITY by creating bucket policy for grant public read access with cloudfront#2								
4.		Implementation of Autoscaling to Manage more than two EC2 #3								
5.		Understand the basics of Lambda and Create your first lambda function in Python ,Java, Node.Js #4								
6.		Implement lambda to start and stop EC2 services #5								
7.		Integration of lambda with S3 for objects create event #6								
8.		Create an Amazon S3 bucket and upload a test file to your new bucket #7								
9.		write Your Lambda function to retrieves information about this file when you upload it in s3 bucket #8								
10.		test the lambda function from the aws management console for java. #9								
11.		Inter region transfer of Table in DynamoDB Integration of Lambda with DynamoDB #10								

S.No	Date	Experiment Name	Pre-Lab (10M)	In-Lab (25M)			Post-Lab (10M)	Viva Voce (5M)	Total (50M)	Faculty Signature
				Program/ Procedure (5M)	Data and Results (10M)	Analysis & Inference (10M)				
12		Configuring a bucket for notifications SQS queue #11								
13.		Build serverless application using Athena Architecture #12								
14.		Implementation of Autoscaling to Manage two different target groups with each having atleast two target group members with load balancer and Deploy a Web Application #13 (Adv/Peer)								
15.		implement power of maths calculator application using lambda #14 (Adv/Peer)								
16.		List all the EC2 instances and Status across all regions in an AWS account using Python boto3 #15 (Adv/Peer)								
17.		Create an SNS topic, subscribe an endpoint to the SNS Topic, publish a message to the SNS Topic, Check receipt of message, Delete the Subscription and the SNS topic #16 (Adv/Peer)								
18.		Introduction of Kinesis as notification Services #17 (Adv/Peer)								
19.		Build serverless application using Athena , S3 , Glue and Quicksight Architecture #18 (Adv/Peer)								
20.		Build end to end serverless web application for Get and post method using CRUD operations #19 (Adv/Peer)								

S.No	Date	Experiment Name	Pre-Lab (10M)	In-Lab (25M)			Post-Lab (10M)	Viva Voce (5M)	Total (50M)	Faculty Signature
				Program/ Procedure (5M)	Data and Results (10M)	Analysis & Inference (10M)				
21		Build Real time CICD Serveless web Application #20 (Adv/Peer)								

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

#1. Experiment Title: Build a standalone static web hosting on EC2 Linux machine

Aim/Objective:

To build a standalone static web hosting on EC2 Linux machine.

Description:

Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS to lower costs, become more agile, and innovate faster. It has following features: (i) most functionality, (ii) largest community of customers and partners, (iii) most secure, (iv) fastest pace of innovation, (v) and most proven operational expertise.

Amazon Web Services offers a broad set of global cloud-based products including compute, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security and enterprise applications. These services help organizations move faster, lower IT costs, and scale. AWS is trusted by the largest enterprises and the hottest start-ups to power a wide variety of workloads including: web and mobile applications, game development, data processing and warehousing, storage, archive, and many others.

Pre-Requisites:

Now that we have highlighted a few reasons why you should see a future with AWS, let's explore the options that make this possible. This AWS services catalog will supply you with the fundamentals to help you get started.

1. Amazon IAM (Identity and Access Management)

AWS Identity and Access Management provides secure access and management of resources in a secure and compliant manner. By leveraging IAM, you can create and manage users and groups by allowing and denying their permissions for individual resources. There are no additional costs, people only get charged for the use of other services by their users.

2. Amazon EC2 (Elastic Compute Cloud)

EC2 is a cloud platform provided by Amazon that offers secure, and resizable compute capacity. Its purpose is to enable easy access and usability to developers for web-scale cloud computing, while allowing for total control of your compute resources. Deploy applications rapidly without the need for investing in hardware upfront; all the while able to launch virtual servers as-needed and at scale.

3. Amazon RDS (Relational Database Services)

Amazon Relational Database Service (Amazon RDS) makes database configuration, management, and scaling easy in the cloud. Automate tedious tasks such as hardware provisioning, database arrangement, patching, and backups – cost-effectively and proportionate to your needs. RDS is available on various database instances which are optimized for performance and memory, providing six familiar database engines including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle. database, and SQL server. By leveraging the AWS Database Migration Service, you can easily migrate or reproduce your existing databases to Amazon RDS. Visit Amazon's RDS page.

4. Amazon VPC (Virtual Private Cloud)

Amazon VPC enables you to set up a reasonably isolated section of the AWS Cloud where you can deploy AWS resources at scale in a virtual environment. VPC gives you total control over your environment, which includes the option to choose your own IP address range, creation of subsets, and arrangement of route tables and network access points. Easily customize the network configuration of your VPC with flexible dashboard management controls designed for maximum usability. For example, users can launch public-facing subnet for web servers with internet access.

Pre-Lab:

1.What is the relation between the Availability Zone and Region?

Ans:

2.What is an AMI, and why is it necessary for EC2 instances?

Ans:

3. Explain the characteristics of different instance types (e.g., t2.micro, m5.large)?

Ans:

4. In which year, Amazon Web Services founded?

A. 2005
B. 2006
C. 2007
D. 2008

5. EC2 stands for ?

A. Elastic Compute Cloud
B. Elastic Configuration Cloud
C. Elastic Cloud Configuration
D. Elastic Cloud Compute

In-Lab: Build a standalone static web hosting on EC2 Linux machine

- **Launching an ec2 instance**

  An instance is a virtual server in the AWS Cloud. You launch an instance from an Amazon Machine Image (AMI). The AMI provides the operating system, application server, and applications for your instance.

  When you sign up for AWS, you can get started with Amazon EC2 for free using the AWS Free Tier. You can use the free tier to launch and use a t2.micro instance for free for 12 months (in Regions where t2.micro is unavailable, you can use a t3.micro instance under the free tier). If you launch an instance that is not within the free tier, you incur the standard Amazon EC2 usage fees for the instance. For more information, see Amazon EC2 pricing.

  After you launch your instance, you can connect to it and use it. To begin, the instance state is pending. When the instance state is running, the instance has started booting. There might be a short time before you can connect to the instance. Note that bare metal instance types might take longer to launch. For more information about bare metal instances, see Instances built on the Nitro System.

  The instance receives a public DNS name that you can use to contact the instance from the internet. The instance also receives a private DNS name that other instances within the same VPC can use to contact the instance. For more information about connecting to your instance, see Connect to your Linux instance.

  When you are finished with an instance, be sure to terminate it.

- **Procedure/Program:**

Q1. Creating First Linux Instance and hosting an Apache web Server

This lab provides you with a basic overview of launching, resizing, managing, and monitoring an Amazon EC2 Linux instance and how to make it as an Apache web server.

By the end of this lab, you will be able to:

- Launch an Apache web server
- Monitor Your EC2 instance
- Modify the security group that your web server is using to allow HTTP access
- Resize your Amazon EC2 instance to scale
- Terminate your EC2 instance

Steps required:

Task 1: Launch an Apache web server

1. Login to AWS Management console through login credential
2. In the **AWS Management Console** on the **Services** menu, click **EC2**.
3. Choose Launch Instance, then select Launch Instance
4. Click Select next to Amazon Linux 2 AMI
5. Click Next: Configure Instance Details

6. Copy the following commands and paste them into the User data field:
   #!/bin/bash
   sudo yum install httpd –y
   sudo service httpd start
   echo '<html><h1>Welcome to Apache Web Server!</h1></html>' >
   /var/www/html/index.html
7. Click Next: Add Storage
8. Click Next: Add Tags
9. Click Add Tag then configure:
   Key: Name
   Value: Apache Web Server
10. Click Next: Configure Security Group
11. Configure Security Group, configure:
    Security group name: Apache Web Server security group
    Description: Security group for my web server
12. Click Review and Launch
13. Click Launch
14. Click the Choose an existing key pair drop-down and select Proceed without a key pair.
15. Select I acknowledge that ....
16. Click Launch Instances
17. Your instance will now be launched.
18. Click View Instances
19. Wait for your instance to display the following:
20. Instance State: running
21. Status Checks: 2/2 checks passed

Task 2: Monitor Your EC2 instance

22. Click the Status Checks tab.
23. Click the Monitoring tab.
24. In the Actions menu, select Monitor and troubleshoot Get System Log.
25. Scroll through the output and note that the HTTP package was installed from the user data that you added when you created the instance.
26. Choose Cancel.
27. In the Actions menu, select Monitor and troubleshoot Get Instance Screenshot.
28. Choose Cancel.

Task 3: Update Your Security Group and Access the Web Server

29. Click the Details tab.
30. Copy the IPv4 Public IP of your instance to your clipboard.
31. Open a new tab in your web browser, paste the IP address you just copied, then press Enter.
32. Keep the browser tab open, but return to the EC2 Management Console tab.
33. In the left navigation pane, click Security Groups.
34. Select Web Server security group.
35. Click the Inbound tab.

The security group currently has no rules.

Click Edit inbound rules then configure:

Type: HTTP
Source: Anywhere

36. Click Save rules
37. Return to the web server tab that you previously opened and refresh the page.
38. You should see the message Welcome to Apache Web Server!

Task 4: Resize your Amazon EC2 instance to scale

39. On the EC2 Management Console, in the left navigation pane, click Instances.
40. Web Server should already be selected.
41. In the Instance state menu, select Stop instance.
42. Choose Stop

Your instance will perform a normal shutdown and then will stop running.
Wait for the Instance State to display: stopped

43. In the Actions menu, select Instance Settings Change Instance Type, then configure:

Instance Type: t2.small

44. Choose Apply
45. In the left navigation menu, click Volumes.
46. In the Actions menu, select Modify Volume.
47. The disk volume currently has a size of 8 GiB. You will now increase the size of this disk.
48. Change the size to: 10

NOTE: You may be restricted from creating large Amazon EBS volumes in this lab.

49. Choose Modify
50. Choose Yes to confirm and increase the size of the volume.
51. Choose Close
52. In left navigation pane, click Instances.
53. In the Instance State menu, select Start instance.
54. Choose Start
55. In the left navigation pane, click Limits.
56. From the drop-down list, choose Running instances.

Task 5: Terminate the instance

57. In left navigation pane, click Instances.
58. In the Instance State menu, select Terminate instance.
59. Then choose Terminate

Q2. Connecting ec2 instance through PuTTy s/w, git bash, CLI, Amazon Linux, Remote desktop connection

This lab provides you with a basic overview of how an Amazon EC2 Linux instance is connected through different client software

By the end of this lab, you will be able to:

- (a) Connect your Linux instance through PuTTy
- (b) Connect your Linux instance through Git bash
- (c) Connect your Linux instance through AWS CLI
- (d) Connect your Linux instance through Amazon Linux
- (e) Connect your Windows instance through Remote desktop connection

(a) Connect your Linux instance through PuTTy

Task 1: Create the private key file format (.ppk file) compatible to PuTTy

1. Open PuTTygen window
2. Click on Load to upload MyKeyPair.pem file
3. Click on Save private key
4. Type yes in confirmation
5. Browse the location where you want to save MyKeyPair.ppk file

Task 2: Authenticate the private key file

6. Open PuTTy configuration window
7. Paste public ipv4 address of ec2 instance that you want to ssh under the text field of host name
8. Click on SSH
9. Click on Auth
10. Click on Browse to select MyKeyPair.ppk file for private key file authentication
11. Now input username as ec2-user
12. Ec2 instance is connected

(b) Connect your Linux instance through git bash

13. Open gitbash window
14. Change the current working directory where your MyKeyPair.pem file is stored
15. Type the following command ssh -i MyKeyPair.pem file name ec2 user name@public ipv4 address For example, ssh -i Cloud-keypair.pem ec2-user@13.235.24.156
16. Type yes in confirmation
17. Ec2 instance should be connected

(c) Connect your Linux instance through AWS CLI

18. Open AWS CLI window
19. Type the command aws –version to verify that whether CLI is successfully installed or not
20. Type the following command aws configure to set and configure user name
21. Input AWS access key ID
22. Input AWS secret access key ID
23. Input region name
24. Input default output format as JSON
25. Type the following command ssh -i MyKeyPair.pem file name ec2 user name@public ipv4 address For example, ssh -i Cloud-keypair.pem ec2-user@13.235.24.156
26. Type yes in confirmation
27. EC2 instance is connected

(d) Connect your Linux instance through Amazon Linux

1. In the **AWS Management Console** on the **Services** menu, click **EC2**.
2. Select running Instance that you want to connect
3. Click connect
4. EC2 instance is connected

(e) Connect your Windows server through Remote Desktop Connection

Task 1: Launch a windows server

5. In the **AWS Management Console** on the **Services** menu, click **EC2**.
6. Choose Launch Instance, then select Launch Instance
7. Click Select next to Amazon Microsoft Windows Server 2019 Base AMI instance.
8. Click Next: Configure Instance Details
9. Click Next: Add Storage
10. Click Next: Add Tags
11. Click Add Tag then configure:
    Key: Name
    Value: Windows Web Server
12. Click Next: Configure Security Group
13. Configure Security Group, configure:
    Security group name: Windows Web Server security group
    Description: Security group for my windows web server
14. Click Review and Launch
15. Click Launch
16. Click the Choose an existing key pair drop-down and select Proceed without a key pair.
17. Select I acknowledge that ....
18. Click Launch Instances
19. Your instance will now be launched.
20. Click View Instances
21. Wait for your instance to display the following:
22. Instance State: running
23. Status Checks: 2/2 checks passed

Task 2: Connect your Windows instance through Remote Desktop Connection

20. Click checked mark on Windows ec2 instance id from Instances
21. Click on connect
22. Click on RDP client
23. Click on Get password
24. Click on Browse button under Browse to your key pair
25. Select the private key that is stored into your local drive
26. Click on Decrypt Password
27. Copy the generated password and save it in secure place
28. Open remote desktop connection from the search bar of o/s.
29. Paste public ip address of your ec2 instance
30. Click on Show options
31. Specify Administrator as user name
32. Click on connect
33. Click on Yes to validate the identity of the server
34. Now, the windows server will be splashed on your screen

- **Data and Results:**

- **Analysis and Inferences:**

Sample VIVA-VOCE Questions (In-Lab):

1. What is a key pair in EC2?

Ans:

2. How to configure aws cli?

Ans:

3. What is an AMI, and why is it necessary for EC2 instances?

Ans:

Post-Lab: Launch KLUNIVERSITY standalone static web hosting on Amazon EC2 Linux machine

- **Procedure/Program:**

1. Open AWS CLI window
2. Type the command aws --version to verify that whether CLI is successfully installed or not
3. Type the following command aws configure to set and configure user name
4. Input AWS access key ID
5. Input AWS secret access key ID
6. Input region name
7. Input default output format as JSON
8. aws ec2 describe-key-pairs
9. aws ec2 create-key-pair –key-name 'My-Key-Pair' --query 'KeyMaterial' --output text > My-Key-Pair.pem
10. chmod 400 My-Key-Pair.pem
11. aws ec2 describe-key-pairs
12. aws ec2 describe-security-groups
13. aws ec2 create-security-group --group-name First-SG --description "This is first Security group"
14. aws ec2 describe-vpcs
15. aws ec2 describe-security-groups
16. Copy the security group-id and subnet-id into gid and sid
17. aws ec2 authorize-security-group-ingress --group-id gid --protocol tcp --port 22 –cidr publicid/32
    Note this public id can be known from checkip.amazon.aws.com
18. Copy the valid AMI-id under the specific region
19. aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t2.micro --key-name MyKeyPair --security-group-ids gid --subnet-id sid
20. Type the following command ssh -i MyKeyPair.pem file name ec2 user name@public ipv4 address
21. Type yes in confirmation
22. Ec2 instance should be connected

- **Data and Results:**

- **Analysis and Inferences:**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

#2 Experiment Title: Build a Static web hosting on AWS S3 bucket with name KLUNIVERSITY by creating bucket policy for grant public read access with cloudfront

Aim/Objective: using Amazon S3 to host a static website.

Description:

You can use Amazon S3 to host a static website. On a static website, individual webpages include static content. They might also contain client-side scripts. By contrast, a dynamic website relies on server-side processing, including server-side scripts such as PHP, JSP, or ASP.NET. Amazon S3 does not support server-side scripting, but AWS has other resources for hosting dynamic websites. To learn more about website hosting on AWS

Pre-Requisites:

For this you need to create a bucket, enable static website hosting, edit block public access settings, add a bucket policy that makes your bucket content publicly available, configure an index document, configure an error document, test your website endpoint, and clean up.

1. Amazon S3 (Simple Storage Service)

Amazon S3, at its core, facilitates object storage, providing leading scalability, data availability, security, and performance. Businesses of vast sizes can leverage S3 for storage and protect large sums of data for various use cases, such as websites, applications, backup, and more. Amazon S3's intuitive management features enable the friction less organization of data and configurable access controls.

Pre-Lab:

1. What is an S3 Bucket?Explain briefly.
Ans:

2. What happens if I exceed my quota on Amazon S3?
Ans:

3. How can you make objects public?
Ans:

4.The durability for Amazon S3 bucket (Standard, IA, One zone – IA and Glacier is ____

a) 99

b) 99.999999999

c)99.99

d)90.999999999

5.The number of accounts that can be given access to a bucket is _____.
Choose the correct option from below list
(1)10
(2)25
(3)100
(4)50

In-Lab:

You can use Amazon S3 to host a static website. On a static website, individual webpages include static content. They might also contain client-side scripts. By contrast, a dynamic website relies on server-side processing, including server-side scripts such as PHP, JSP, or ASP.NET. Amazon S3 does not support server-side scripting, but AWS has other resources for hosting dynamic websites. To learn more about website hosting on AWS

For this you need to create a bucket, enable static website hosting, edit block public access settings, add a bucket policy that makes your bucket content publicly available, configure an index document, configure an error document, test your website endpoint, and clean up.

- **Procedure/Program:**

Task1: Create a S3 bucket with name KLUNIVERSITY

1. Sign into the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose Create bucket.
3. Enter the Unique Bucket name (for example, KLUNIVERSITY<123>).
4. Choose the Region where you want to create the bucket.
5. Choose a Region that is geographically close to you to minimize latency and costs, or to address regulatory requirements. The Region that you choose determines your Amazon S3 website endpoint.
6. To accept the default settings and create the bucket, choose Create.

Task 2: Enable static website hosting

7. In the Buckets list, choose the name of the bucket that you want to enable static website hosting for.
8. Choose Properties, Under Static website hosting, choose Edit.
9. Choose Use this bucket to host a website, Under Static website hosting, choose Enable.
10. In Index document, enter the file name of the index document, typically index.html.
    The index document name is case sensitive and must exactly match the file name of the HTML index document that you plan to upload to your S3 bucket. When you configure a bucket for website hosting, you must specify an index document. Amazon S3 returns this index document when requests are made to the root domain or any of the subfolders. For more information, see Configuring an index document.
11. To provide your own custom error document for 4XX class errors, in Error document, enter the custom error document file name.
    The error document name is case sensitive and must exactly match the file name of the HTML error document that you plan to upload to your S3 bucket. If you don't specify a custom error document and an error occurs, Amazon S3 returns a default HTML error document. For more information, see Configuring a custom error document.
12. (Optional) If you want to specify advanced redirection rules, in Redirection rules, enter XML to describe the rules.

13. For example, you can conditionally route requests according to specific object key names or prefixes in the request. For more information, see Configure redirection rules to use advanced conditional redirects.
14. Choose Save changes.
15. Amazon S3 enables static website hosting for your bucket. At the bottom of the page, under Static website hosting, you see the website endpoint for your bucket.
16. Under Static website hosting, note the Endpoint.
    The Endpoint is the Amazon S3 website endpoint for your bucket. After you finish configuring your bucket as a static website, you can use this endpoint to test your website.

Task 3: Edit Block Public Access settings

17. Choose the name of the bucket that you have configured as a static website.
18. Choose Permissions, Under Block public access (bucket settings), choose Edit.
19. Clear Block all public access and choose Save changes.
    Amazon S3 turns off Block Public Access settings for your bucket. To create a public, static website, you might also have to edit the Block Public Access settings for your account before adding a bucket policy. If account settings for Block Public Access are currently turned on, you see a note under Block public access (bucket settings).

Task 4: Add a bucket policy that makes your bucket content publicly available

20. After you edit S3 Block Public Access settings, you can add a bucket policy to grant public read access to your bucket. When you grant public read access, anyone on the internet can access your bucket.
21. Under Buckets, choose the name of your bucket, Choose Permissions.
22. Under Bucket Policy, choose Edit. To grant public read access for your website, copy the following bucket policy, and paste it in the Bucket policy editor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::Bucket-Name/*"
      ]
    }
  ]
}
```

23. Update the Resource to your bucket name.
    In the preceding example bucket policy, Bucket-Name is a placeholder for the bucket name. To use this bucket policy with your own bucket, you must update this name to match your bucket name.
24. Choose Save changes.
    A message appears indicating that the bucket policy has been successfully added.
    If you see an error that says Policy has invalid resource, confirm that the bucket name in the bucket policy matches your bucket name. If you get an error message and cannot save the bucket policy, check your account and bucket Block Public Access settings to confirm that you allow public access to the bucket.

Task 5: Create an index.html file.

25. Create an index.html file.
    If you don't have an index.html file, you can use the following HTML to create one:

```
<html>
<head>
  <title>KLUNIVERSITY Home Page</title>
</head>
<body>
 <h1>Welcome to KLUNIVERSITY</h1>
 <p>Now hosted on Amazon S3!</p>
</body>
</html>
```

    Save the index file locally.
    The index document file name must exactly match the index document name that you enter in the Static website hosting dialog box. The index document name is case sensitive. For example, if you enter index.html for the Index document name in the Static website hosting dialog box, your index document file name must also be index.html and not Index.html.
26. In the Buckets list, choose the name of the bucket that you want to use to host a static website.
    Enable static website hosting for your bucket and enter the exact name of your index document (for example, index.html).
27. Drag and drop the index file into the console bucket listing.
28. Choose Upload and follow the prompts to choose and upload the index file.
    (Optional) Upload other website content to your bucket.

Task 6: Configure an error document

29. Create an error document, for example 404.html.
30. Save the error document file locally.
31. Upload this file into the same bucket

Task 7: Test your website endpoint

32. After you configure static website hosting for your bucket, you can test your website endpoint.
33. Under Buckets, choose the name of your bucket.
    Choose Properties.
34. At the bottom of the page, under Static website hosting, choose your Bucket website endpoint.
    Your index document opens in a separate browser window.

Ans:

- **Data and Results:**

- **Analysis and Inferences:**

Sample VIVA-VOCE Questions (In-Lab):

1.Which of the following is NOT an advantage of AWS cloud computing over on-premise computing?

a) Broad selection of hardware and software resources over the internet

b) Large upfront investments in physical servers

c) Provision compute capacity on-demand

d) Pay-as-you-go pricing

2.Which AWS service allows you to use compute capacity without worrying about provisioning or managing underlying hardware?

a) Amazon S3

b) Amazon ECS

c) Amazon DynamoDB

d) Amazon Lightsail

3.What does AMI stand for?

a) Amazon Modified Instance

b) Amazon Main Infrastructure

c) Amazon Machine Image

4.If you want to make changes to your EC2 instance, you must do it programmatically.

a) False

b) True

5.An AMI allows you to configure software, such as operating system, web servers, etc. on your EC2 instance.

a) False

b) True

Post-Lab:

- **Procedure/Program:**

1. Apply life cycle rule for bucket auto deletion?

Ans:

2. Add cloudfont for public access ?

Ans:

- **Data and Results:**

- **Analysis and Inferences:**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#3 Experiment Title: Implementation of Auto scaling to Manage more than two EC2

Aim/Objective: AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost

Description:

AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost. Using AWS Auto Scaling, it's easy to setup application scaling for multiple resources across multiple services in minutes. The service provides a simple, powerful user interface that lets you build scaling plans for resources including Amazon EC2 instances and Spot Fleets, Amazon ECS tasks, Amazon DynamoDB tables and indexes, and Amazon Aurora Replicas. AWS Auto Scaling makes scaling simple with recommendations that allow you to optimize performance, costs, or balance between them. If you're already using Amazon EC2 Auto Scaling to dynamically scale your Amazon EC2 instances, you can now combine it with AWS Auto Scaling to scale additional resources for other AWS services. With AWS Auto Scaling, your applications always have the right resources at the right time.

It's easy to get started with AWS Auto Scaling using the AWS Management Console, Command Line Interface (CLI), or SDK. AWS Auto Scaling is available at no additional charge. You pay only for the AWS resources needed to run your applications and Amazon CloudWatch monitoring fees.

Using AWS Auto Scaling, you maintain optimal application performance and availability, even when workloads are periodic, unpredictable, or continuously changing. AWS Auto Scaling continually monitors your applications to make sure that they are operating at your desired performance levels. When demand spikes, AWS Auto Scaling automatically increases the capacity of constrained resources so you maintain a high quality of service.

It lets you set target utilization levels for multiple resources in a single, intuitive interface. You can quickly see the average utilization of all of your scalable resources without having to navigate to other consoles. For example, if your application uses Amazon EC2 and Amazon DynamoDB, you can use AWS Auto Scaling to manage resource provisioning for all of the EC2 Auto Scaling groups and database tables in your application.

AWS Auto Scaling lets you build scaling plans that automate how groups of different resources respond to changes in demand. You can optimize availability, costs, or a balance of both. AWS Auto Scaling automatically creates all of the scaling policies and sets targets for you based on your preference. AWS Auto Scaling monitors your application and automatically adds or removes capacity from your resource groups in real-time as demands change.

Pre-Requisites:

AWS Auto Scaling, Amazon EC2

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Pre-Lab:

1. What is Elastic Load Balancer (ELB)?

Ans:

2.What is the difference between auto-scaling and ELB?

Ans:

3.How can we assign a static IP address to a ELB?

Ans:

4. What is VPC ?

Ans:

5.Difference between Ingress and Load Balancer?

Ans:

In-Lab:

AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost. Using AWS Auto Scaling, it's easy to setup application scaling for multiple resources across multiple services in minutes. The service provides a simple, powerful user interface that lets you build scaling plans for resources including Amazon EC2 instances and Spot Fleets, Amazon ECS tasks, Amazon DynamoDB tables and indexes, and Amazon Aurora Replicas. AWS Auto Scaling makes scaling simple with recommendations that allow you to optimize performance, costs, or balance between them. If you're already using Amazon EC2 Auto Scaling to dynamically scale your Amazon EC2 instances, you can now combine it with AWS Auto Scaling to scale additional resources for other AWS services. With AWS Auto Scaling, your applications always have the right resources at the right time.

It's easy to get started with AWS Auto Scaling using the AWS Management Console, Command Line Interface (CLI), or SDK. AWS Auto Scaling is available at no additional charge. You pay only for the AWS resources needed to run your applications and Amazon CloudWatch monitoring fees.

It lets you set target utilization levels for multiple resources in a single, intuitive interface. You can quickly see the average utilization of all of your scalable resources without having to navigate to other consoles. For example, if your application uses Amazon EC2 and Amazon DynamoDB, you can use AWS Auto Scaling to manage resource provisioning for all of the EC2 Auto Scaling groups and database tables in your application.

- **Procedure/Program:**

Task 1: Implementation of Load Balancer with 2 EC2 instances

1. Configure 1st ec2-instance having name ec2-server1 with the following user data

```
#!/bin/bash
sudo yum install httpd –y
sudo service httpd start
echo “<h1> Server 1>/h1>” > /var/www/html/index.html
```

2. Connect this ec2-server1 instance
   Configure 2nd ec2-instance with name ec2-server2 with the user data

```
#!/bin/bash
sudo yum install httpd –y
sudo service httpd start
echo “<h1> Server 2>/h1>” > /var/www/html/index.html
connect ec2-server2 instance
```

3. Click on search bar and type ec2, In the left navigation pane, under Load Balancing, select Load Balancers
4. Among all click on Classic Load Balancer - previous generation from Select load balancer type
5. Click on create
6. Specify load balancer name – only alphabet, number and hyphen are allowed
7. Create LB inside- by default
8. Create an internal load balancer – by default
9. Enable advanced VPC configuration – Check on this label

10. Under Select subnets, select and add two subnets - You will need to select a Subnet for each Availability Zone where you wish traffic to be routed by your load balancer. If you have instances in only one Availability Zone, please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

11. Here, either we can create a new security group or select an existing security group. In case of creating a new security group, you need to specify security group name, and Description (optional).
    Under security group creation. Select HTTP under type. Port range 80, protocol TCP, and source 0.0.0.0/0
12. Configure security settings
13. Click on next

Configure health check- Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer. Customize the health check to meet your specific needs.

14. You can also alter Ping protocol, Ping port, and Ping path, if you want to alter.
15. You can also alter advanced Details like Response Timeout, Interval, Unhealthy threshold, Healthy threshold as per your requirements.
16. Add EC2 instances- The table below lists all your running EC2 Instances. Check the boxes in the Select column to add those instances to this load balancer.
17. You can also avail Availability zone distribution property by enabling
    Enable cross-zone load balancing and
    Enable connection Draining
18. Add Tags- Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.
19. Creation of a tag is optional. But, If you want to create, then it will be viable.
20. Click on Create and Review
21. Close
22. Test load Balancer application by pasting its DNS into the browser search bar Under Description tab, copy DNS name LBL1-1346090609.ap-south-1.elb.amazonaws.com (A Record)
23. It will start run.

Task 2: Implementation of Auto Scaling with Load Balancer

24. Click on Launch configuration under Auto scaling in EC2 Dashboard
25. Specify the name of Launch configuration
26. Select AMI and select instance type
27. Assign security group (select existing one or create a new one)
28. Select an existing key pair
29. Click on the checkbox of “I acknowledge that I have access to the selected private key file (.pem), and that without this file, I won't be able to log into my instance”.
30. Click on Create Launch Configuration.
31. Click on Auto scaling group under Auto scaling in EC2 Dashboard

32. Click on Create Auto scaling group button
33. Specify Auto scaling group name
34. Click on switch to Launch Configuration
35. Select the created Launch Configuration name
36. Choose VPC and select all availability zones
37. Click on next button
38. Select Attach to an existing load balancer
39. Select ELB Checkbox
40. Specify Health check grace period value
41. Configure group size and scaling policies
42. Click on Next, Next, Next Buton
45. Click on Create Auto Scaling group
46. Copy its DNS name and paste it in new tab of your browser

- **Data and Results:**

- **Analysis and Inferences:**

Sample VIVA-VOCE Questions (In-Lab):

1. **What is AWS Auto Scaling?**

Ans:

2. **What is Load Balancer?**

Ans:

3. **What is the full form of EC2**

Ans:

4. What is an ELB?

Ans:

4. **List out the uses of AWS Auto Scaling?**

Ans:

Post-Lab:

AWS Auto Scaling lets you build scaling plans that automate how groups of different resources respond to changes in demand. You can optimize availability, costs, or a balance of both. AWS Auto Scaling automatically creates all of the scaling policies and sets targets for you based on your preference. AWS Auto Scaling monitors your application and automatically adds or removes capacity from your resource groups in real-time as demands change.

Using AWS Auto Scaling, you maintain optimal application performance and availability, even when workloads are periodic, unpredictable, or continuously changing. AWS Auto Scaling continually monitors your applications to make sure that they are operating at your desired performance levels. When demand spikes, AWS Auto Scaling automatically increases the capacity of constrained resources so you maintain a high quality of service.

- **Procedure/Program:**

  Task 3: Implementation of Auto Scaling without Load Balancer

  Step 1: Create a launch template

1. Open the Amazon EC2 console.
2. On the navigation bar at the top of the screen, select an AWS Region. The Amazon EC2 Auto Scaling resources that you create are tied to the Region that you specify.
3. In the left navigation pane, choose Launch Templates, and then choose Create launch template.
4. For Launch template name, enter my-template-for-auto-scaling.
5. Under Auto Scaling guidance, select the check box.
6. For Amazon machine image (AMI), choose a version of Amazon Linux 2 (HVM) from the Quick Start list. The AMI serves as a basic configuration template for your instances.
7. For Instance type, choose a hardware configuration that is compatible with the AMI that you specified.
8. (Optional) For Key pair name, choose an existing key pair. You use key pairs to connect to an Amazon EC2 instance with SSH. Connecting to an instance is not included as part of this tutorial. Therefore, you don't need to specify a key pair unless you intend to connect to your instance.
9. Leave Networking platform set to VPC.
10. For Security groups, choose a security group in the same VPC that you plan to use as the VPC for your Auto Scaling group. If you don't specify a security group, your instance is automatically associated with the default security group for the VPC.
11. You can leave Network interfaces empty. Leaving the setting empty creates a primary network interface with IP addresses that we select for your instance (based on the subnet to which the network interface is established). If instead you choose to specify a network interface, the security group must be a part of it.
12. Choose Create launch template.
13. On the confirmation page, choose Create Auto Scaling group.

14. On the Choose launch template or configuration page, for Auto Scaling group name, enter my-first-asg.
15. Choose Next.
16. The Choose instance launch options page appears, allowing you to choose the VPC network settings you want the Auto Scaling group to use and giving you options for launching On-Demand and Spot Instances (if you chose a launch template).
17. In the Network section, keep VPC set to the default VPC for your chosen AWS Region, or select your own VPC. The default VPC is automatically configured to provide internet connectivity to your instance. This VPC includes a public subnet in each Availability Zone in the Region.
18. For Availability Zones and subnets, choose a subnet from each Availability Zone that you want to include. Use subnets in multiple Availability Zones for high availability. For more information, see Considerations when choosing VPC subnets.
19. [Launch template only] In the Instance type requirements section, use the default setting to simplify this step. (Do not override the launch template.) For this tutorial, you will launch only one On-Demand Instance using the instance type specified in your launch template.
20. Keep the rest of the defaults and choose Skip to review.
21. On the Review page, review the information for the group, and then choose Create Auto Scaling group.
22. Select the check box next to the Auto Scaling group that you just created.
23. A split pane opens up in the bottom part of the Auto Scaling groups page, showing information about the group. The first tab available is the Details tab, showing information about the Auto Scaling group.
24. Choose the second tab, Activity. Under Activity history, you can view the progress of activities that are associated with the Auto Scaling group. The Status column shows the current status of your instance. While your instance is launching, the status column shows PreInService. The status changes to Successful after the instance is launched. You can also use the refresh button to see the current status of your instance.
25. On the Instance management tab, under Instances, you can view the status of the instance. Verify that your instance launched successfully. It takes a short time for an instance to launch. The Lifecycle column shows the state of your instance. Initially, your instance is in the Pending state. After an instance is ready to receive traffic, its state is InService. The health status column shows the result of the EC2 instance health check on your instance.
26. Go to the next step if you would like to delete the basic infrastructure for automatic scaling that you just created.

- **Data and Results:**

- **Analysis and Inferences:**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#4. Experiment Title: Understand the basics of Lambda and Create your first lambda function in Python ,Java, Node.Js

Aim/Objective:

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers.

Description:

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and software as a service (SaaS) applications, and only pay for what you use. It runs code without provisioning or managing infrastructure. Simply write and upload code as a .zip file or container image. It automatically respond to code execution requests at any scale, from a dozen events per day to hundreds of thousands per second. So, it saves costs by paying only for the compute time you use—by per-millisecond—instead of provisioning infrastructure upfront for peak capacity. It optimizes code execution time and performance with the right function memory size. Respond to high demand in double-digit milliseconds with Provisioned Concurrency

Pre-Requisites:

Lambda Function

Pre-Lab:

1. What are the features in AWS lambda that automate the deployment?

Ans:

2.What are the advantages of using the Serverless approach?

Ans:

3. What do you know about Zero downtime deployment?

Ans:

4. What is the time limit for execution in Lambda when you perform DDOS?

Ans:

5. What are the best practices for security in Lambda?

Ans:

In-Lab:

- **Procedure/Program:**

  Task 1: A sample Lambda Function lab demonstration

  This lab will demonstrate you about how to create a Lambda Function as following:

  1. Services- lambda
  2. Under create function, select Use a blueprint
  3. Write and select hello-world-python
  4. Click on configure
  5. Specify function name as Hello-World-Function
  6. Under Execution role, select service-role/HelloWorld-role
  7. Click on create function
  8. Click on Test
  9. Select configure Test event
  10. Type HelloWoldEvent
  11. Click on create
  12. Click on Test multiple times
  13. Click on Detail
  14. Click on Monitor tab
  15. Click on view logs in CloudWatch

Program 1:

Your first AWS Lambda function-planet Python 3.7 Code:

Step1:LAMBDA CODE

```
import json
def lambda_handler(event, context):
print('this is my aws lambda function')
if event['planet'] == 'earth':
    return 'moon'
elif event['planet'] == 'sun':
    return 'this is not the planet'
else:
     return ' we do nt recognize your argument'
```

Step2: Create TEST event

Json code:

```
 {
 'planet': 'earth'
}
```

Step3: deploy
Step4: test.
Note: if python 3.9 use double quotes instead of single quotes.
output: moon

- **Data and Results:**

- **Analysis and Inferences:**

Program 2:

Step1: create lambda function - python 3.7

Lambda code: cal.py

```
import the JSON utility package
import json
import the Python math library
import math
define the handler function that the Lambda service will use an entry point
def lambda_handler(event, context):
extract the two numbers from the Lambda service's event object
   mathResult = math.pow(int(event['base']), int(event['exponent']))
return a properly formatted JSON object
   return {
   'statusCode': 200,
```

'body': json.dumps('Your result is ' + str(mathResult))

}

Step2: Create TEST event

Json code:

{

'base': '2',

'exponent': '3'

}

Step3: deploy

Step4: test.

- **Data and Results:**

- **Analysis and Inferences:**

Sample VIVA-VOCE Questions (In-Lab):

1. **What is Lambda?**

Ans:

2. **How many different runtime libraries supported by Lambda?**

Ans:

3. What is Trigger in context of Lambda?

Ans:

4. What do you mean by event handling?

Ans:

Post-Lab:

- **Procedure/Program:**

1. How to create your first AWS Lambda Function in Java?

Ans:

- **Data and Results:**

- **Analysis and Inferences:**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#5. Experiment Title: Implement lambda to start and stop EC2 services

Aim/Objective:

To stop and start Amazon Elastic Compute Cloud (Amazon EC2) instances at regular intervals using Lambda.

Description:

You can reduce your Amazon Elastic Compute Cloud (Amazon EC2) usage by stopping and starting my EC2 instances automatically. To stop and start EC2 instances at regular intervals using Lambda, You create a custom AWS Identity and Access Management (IAM) policy and execution role for your Lambda function. Next, you create Lambda functions that stop and start your EC2 instances. Test your Lambda functions. And, thereafter, you create Cloud Watch Events rules that trigger your function on a schedule. You can also create rules that trigger on an event that takes place in your AWS account. Or, you can use the AWS Cloud Formation template provided at the end of this article to automate the procedure.

Pre-Requisites:

Lambda function, AWS account, AWS Cloud Formation template

Pre-Lab:

1. What Is Amazon EC2 Service ?

Ans:

2. Explain Storage For Amazon EC2 Instance ?

Ans:

3. What Is Amazon EC2 Root Device Volume ?

Ans:

4. Which of the following is not used for load balancing?

A. Zeus

B. HAProxy

C. Nginx

D. OpenSwan

5. Which of the following is a structured data store that supports indexing and data queries to both EC2 and S3?

A. CloudWatch

B. Amazon SimpleDB

C. Amazon Cloudfront

D. All of the above

In-Lab:

- **Procedure/Program:**

To stop and start EC2 instances at regular intervals using Lambda, do the following:

Task 1: Create a custom AWS Identity and Access Management (IAM) policy and execution role for your Lambda function.

1. Create an IAM policy using the JSON policy editor. Copy and paste the following JSON policy document into the policy editor:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogGroup",
    "logs:CreateLogStream",
    "logs:PutLogEvents"
   ],
   "Resource": "arn:aws:logs:*:*:*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:Start*",
    "ec2:Stop*"
   ],
   "Resource": "*"
  }
 ]
}
```

2. Create an IAM role for Lambda.
   Note: When attaching a permissions policy to Lambda, make sure that you choose the IAM policy you just created.

Task 2: Create Lambda functions that stop and start your EC2 instances.

3. In the AWS Lambda console, choose Create function.
4. Choose Author from scratch.
5. Under Basic information, add the following:
   For Function name, enter a name that identifies it as the function used to stop your EC2 instances. For example, "StopEC2Instances".
   For Runtime, choose Python 3.9.
   Under Permissions, expand Change default execution role.
   Under Execution role, choose Use an existing role.
   Under Existing role, choose the IAM role that you created.
6. Choose Create function.

7. Under Code, Code source, copy and paste the following code into the editor pane in the code editor (lambda_function). This code stops the EC2 instances that you identify.

```
import boto3
region = 'us-west-1'
instances = ['i-12345cb6de4f78g9h', 'i-08ce9b2d7eccf6d26']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
   ec2.stop_instances(InstanceIds=instances)
   print('stopped your instances: ' + str(instances))
```

Important: For region, replace "us-west-1" with the AWS Region that your instances are in. For instances, replace the example EC2 instance IDs with the IDs of the specific instances that you want to stop and start.

8. Choose Deploy.
9. On the Configuration tab, choose General configuration, Edit. Set Timeout to 10 seconds and then select Save.

   Note: Configure the Lambda function settings as needed for your use case. For example, if you want to stop and start multiple instances, you might need a different value for Timeout and Memory.

10. Repeat steps 1-9 to create another function. Do the following differently so that this function starts your EC2 instances:

    In step 3, enter a different Function name than the one you used before. For example, "StartEC2Instances".

    In step 5, copy and paste the following code into the editor pane in the code editor (lambda_function):

```
import boto3
region = 'us-west-1'
instances = ['i-12345cb6de4f78g9h', 'i-08ce9b2d7eccf6d26']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
   ec2.start_instances(InstanceIds=instances)
   print('started your instances: ' + str(instances))
```

Note: For region and instances, use the same values that you used for the code to stop your EC2 instances.

Task 3: Test your Lambda functions.

11. In the AWS Lambda console, choose Functions.
12. Choose one of the functions that you created.
13. Select the Code tab.
14. In the Code source section, select Test.
15. In the Configure test event dialog box, choose Create new test event.
16. Enter an Event name. Then, choose Create.

Note: You don't need to change the JSON code for the test event—the function doesn't use it.

17. Choose Test to run the function.
18. Repeat steps 1-6 for the other function that you created.
    Tip: You can check the status of your EC2 instances before and after testing to confirm that your functions work as expected.

- **Data and Results:**

- **Analysis and Inferences:**

Sample VIVA-VOCE Questions (In-Lab):

1. What is JSON policy document?

Ans:

2. What is the use of Instance in EC2?

Ans:

3. What is AWS Identity and Access Management (IAM) policy?

Ans:

4. What is the purpose of using Cloud Watch?

Ans:

5. Why we need to use Cloud Formation template?

Ans:

Post-Lab:

1. how to auto start / stop EC2 instances using AWS Lambda Function

- **Procedure/Program:**

Step1: initially will create 2 EC2 instances in AWS Region and take instances IDs.

Step2: create IAM ROLE with following JSON code(attach FULL EC2,CLOUDWATCH,LAMBDA)

```
{
  "Version": "2012-10-17",
   "Statement":[
                {
                  "Effect": "Allow",
                  "Action": [
                       "logs:CreateLogGroup",
                        "logs:CreateLogStream",
                         "logs:PutLogEvents"
                              ],
                    "Resource": "arn:aws:logs:*:*:*"
                },
               {
                   "Effect": "Allow",
                    "Action": [

                              "ec2:Start*",
                              "ec2:Stop*"
                                ],
                      "Resource": "*"
              }
        ]
```

```
        }
```

Step 3: then will create 2 AWS Lambdas in the same region

//Stops the EC2 instances for lambda1

```
import boto3 region = 'us-east-1'
instances = ['i-12345', 'i-2345']
ec2 = boto3.client('ec2', region_name=region)
def lambda_handler(event, context):
        ec2.stop_instances(InstanceIds=instances)
        print('stopped your instances: ' + str(instances))
```

// Starts the EC2 instances for lambda2

```
import boto3
region = 'us-east-1'
instances = ['i-12345', 'i-2345']
ec2 = boto3.client('ec2', region_name=region)
def lambda_handler(event, context):
        ec2.start_instances(InstanceIds=instances)
        print('started your instances: ' + str(instances))
```

Step 4: start and stop periodically or at given time using Lambda function and CloudWatch events. Please find the required Role and Lambda Scrips

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

- **Data and Results:**

- **Analysis and Inferences:**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#6. Experiment Title: Integration of lambda with S3 for objects create event

Aim/Objective:

To create a sample Lambda function to be triggered on an S3 Object upload event.

Description:

You can use Lambda to process event notifications from Amazon Simple Storage Service. Amazon S3 can send an event to a Lambda function when an object is created or deleted. You configure notification settings on a bucket, and grant Amazon S3 permission to invoke a function on the function's resource-based permissions policy. Amazon S3 invokes your function asynchronously with an event that contains details about the object.

This lab walks you through the creation and usage a serverless AWS service called AWS Lambda. In this lab, we will create a sample Lambda function to be triggered on an S3 Object upload event. The lambda function will make a copy of that object and place it in a different s3 bucket.

Pre-Requisites:

Lambda function, Amazon S3

Pre-Lab:

1. What is AWS Lambda?

Ans:

2. What's AWS Lambda's best language?

Ans:

3. What events can trigger an AWS Lambda function?

Ans:

4. What are different ways to trigger Lambda?

Ans:

5. What are the factors that decide how AWS Lambda is charged?

Ans:

In-Lab:

- **Procedure/Program:**

Task 1. Log in to the AWS Management Console.
Task 2. Create two S3 buckets. One for the source and one for the destination.
Task 3. Create a Lambda function to copy the object from one bucket to another bucket.
Task 4. Test the Lambda Function.

1. S3 Configuration
   Services -> S3
2. Create Amazon S3 Bucket (Source Bucket)
3. Click on Create bucket.
   - Bucket Name: your_source_bucket_name
   - Region: US East (N. Virginia)

   Note: Every S3 bucket name is unique globally, so create the bucket with a name not currently in use.
4. Leave other settings as default and click on the Create button. Once the bucket is created successfully, select your S3 bucket (click on the checkbox). Click on the Copy Bucket ARN to copy the ARN like arn:aws:s3::: source_bucket_name
5. Save the source bucket ARN in a text file for later use. Create Amazon S3 Bucket (Destination Bucket). Click on Create bucket.
   - Bucket Name: your_destination_bucket_name
   - Region: US East (N. Virginia)

   Note: Every S3 bucket name is unique globally, so create the bucket with a name not currently in use. Leave other settings as default and click on the Create button.
   Once the bucket is created successfully, select your S3 bucket (click on the checkbox).
6. Click on the Copy Bucket ARN to copy the ARN.
   - arn:aws:s3:::zacks-destination-bucket
7. Save the source bucket ARN in a text file for later use.
   Now we have two S3 buckets (Source and Destination). We will make use of our AWS Lambda function to copy the content from source bucket to destination bucket.
8. IAM Configuration
   Services -> IAM -> Policies
9. Create an IAM Policy
   As a pre-requisite for creating the Lambda function, we need to create a user role with a custom policy.
10. Click on Create policy.
11. Click on the JSON tab and copy-paste the below policy statement in the editor:

Policy JSON

```
 {
"Version":"2012-10-17",
"Statement":[
  {
    "Effect":"Allow",
    "Action":[
      "s3:GetObject"
    ],
    "Resource":[
      "arn:aws:s3:::sourcebukklu/*"
    ]
  },
  {
    "Effect":"Allow",
    "Action":[
      "s3:PutObject"
    ],
    "Resource":[
      "arn:aws:s3:::destinationbukklu/*"
    ]
  }
 ]
}
```

12. Make sure you have /* after the arn name. Click on Review policy. Click on the Create policy button. An IAM Policy with the name myS3policy is created.
13. Create an IAM Role- In the left menu, click on Roles. Click on the Create role button.
14. Select Lambda from AWS Services list.
15. Click on Next: Permissions.
16. Filter Policies: Now you can see a list of policies. Search for your policy by name (myS3policy).
17. Select your policy and click on the Next: Tags.
18. Add Tags: Provide key-value pair for the role:
    - Key: Name
    - Value: myS3role
19. Click on the Next: Review
20. Role Name:
    - Role name: myS3role
21. Click on the Create role button.

You have successfully created an IAM role by name myS3role.

22. Lambda Configuration- Services -> Lambda
23. Create a Lambda Function- Click on the Create a function button.
24. Choose Author from scratch.

Function name: mylambdafunction

25. Runtime: Select Node.js 12x
26. Role: In the permissions section, select use an existing role.
27. Existing role: Select myS3role, Click on Create function
28. Configuration Page: On this page, we need to configure our lambda function.
    If you scroll down a little bit, you can see the Function code section. Here we need to write a NodeJs function which copies the object from the source bucket and paste it into the destination bucket.
29. Remove the existing code in AWS lambda index.js. Copy the below code and paste it into your lambda index.js file.

```
var AWS = require("aws-sdk");
exports.handler = (event, context, callback) => {
var s3 = new AWS.S3();
var sourceBucket = "your_source_bucket_name";
var destinationBucket = "your_destination_bucket_name";
 var objectKey = event.Records[0].s3.object.key;
 var copySource = encodeURI(sourceBucket + "/" + objectKey);
            var copyParams = { Bucket: destinationBucket, CopySource: copySource,
Key: objectKey };
            s3.copyObject(copyParams, function(err, data) {
if (err) {
console.log(err, err.stack);
            } else {
                        console.log("S3 object copy successful.");
                       }
           });

   };
```

30. You need to change the source and destination bucket name (not ARN!) in the index.js file based on your bucket names.
31. Save the function by clicking on Deploy in the right corner.
32. Adding Triggers to Lambda Function, Go to the top and left page, click on + Add trigger under Designer, Scroll down the list and select S3 from the trigger list.
33. Once you select S3, a form will appear. Enter these details:
    - Bucket: Select your source bucket - your_source_bucket_name.
    - Event type: All object create events
34. Leave other fields as default.
35. And check this option of Recursive invocation to avoid failures in case you upload multiple files at once.
36. Click on Add.
37. Validation Test, Prepare an image on your local machine.
38. Go to Bucket list and click on source bucket - your_source_bucket_name.
39. Upload image to source S3 bucket. To do that:
    - Click on the Upload button.

- Click on Add files to add the files.
- Select the image and click on the Upload button to upload the image.

40. Now go back to the S3 list and open your destination bucket - your_destination_bucket_name.
41. To open the object, scroll down and change ACL - Everyone – Read
42. You can see a copy of your uploaded source bucket image in the destination bucket.

- **Data and Results:**

- **Analysis and Inferences:**

Sample VIVA-VOCE Questions (In-Lab):

1. What is the use of Amazon Simple Storage Service

Ans:

2. What is an event

Ans:

3. How an event can be triggered

Ans:

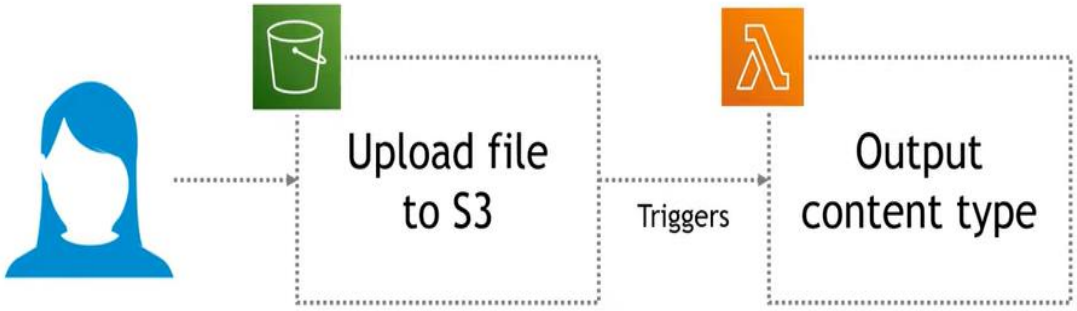
4. What is S3 buckets

Ans:

5. What is the use of IAM Policy

Ans:

Post-Lab: Integration of lambda with S3 for objects create event



- **Procedure/Program:**
- **STEP1: CREATE A BUCKET IN N.VIRGINIA**

  STEP2: LAMBDA PYTHON 3.9 SAME REGION (N.VIRGINIA)

  CREATE NEW ROLE

  -S3 READ ONLY PERMISSION

  -ALL REMAINING DEFAULT

  STEP3: ADD TRIGGER FROM LAMBDA TO S3

  -SELECT SAME BUCKET FROM S3

  -SELECT ALL OBJECT CREATE EVENTS

  - CHECK IN S3- EVENT NOTIFICATION ADDED OR NOT

  STEP4: UPLOAD ANY FILE IN S3 BUCKET AND CHECK LOGS IN LAMBDA

  SUCCESSFUL

LAMBDACODE:

```
import json
import urllib.parse
import boto3
print('Loading function')
s3 = boto3.client('s3')
def lambda_handler(event, context):
   #print("Received event: " + json.dumps(event, indent=2))
   Get the object from the event and show its content type
   bucket = event['Records'][0]['s3']['bucket']['name']
   key = urllib.parse.unquote_plus(event['Records'][0]['s3']  ['object']['key'],
encoding='utf-8')
   try:
      response = s3.get_object(Bucket=bucket, Key=key)
      print("CONTENT TYPE: " + response['ContentType'])
      return response['ContentType']
   except Exception as e:
      print(e)
      print('Error getting object {} from bucket {}. Make sure they exist and your bucket is
in the same region as this function.'.format(key, bucket))
      raise e
```

- **Data and Results:**

- **Analysis and Inferences:**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#7. Experiment Title: Create an Amazon S3 bucket and upload a test file to your new bucket

Aim/Objective:

To upload a test file to your new Amazon S3 bucket.

Description:

When you upload a test file to your new bucket, your Lambda function retrieves information about this file when you test the function from the console. To invoke your function, Amazon S3 needs permission from the function's resource-based policy. When you configure an Amazon S3 trigger in the Lambda console, the console modifies the resource-based policy to allow Amazon S3 to invoke the function if the bucket name and account ID match. If you configure the notification in Amazon S3, you use the Lambda API to update the policy. You can also use the Lambda API to grant permission to another account, or restrict permission to a designated alias. If your function uses the AWS SDK to manage Amazon S3 resources, it also needs Amazon S3 permissions in its execution role.

Pre-Requisites:

Amazon S3 bucket, Lambda function and AWS SDK.

Pre-Lab:

1. What is Auto Scaling in Lambda?

Ans:

2. What is Auto Scaling in Lambda?

Ans:

3. What do you think makes Lambda a timesaving approach?

Ans:

4. What are the best practices for security in Lambda?

Ans:

5. How does Lambda handle failure during event processing?

Ans:

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

In-Lab:

- **Procedure/Program:**

Task 1: Create bucket and upload file:

1. To create an Amazon S3 bucket using the console, Open the Amazon S3 console. Choose Create bucket.
2. For AWS Region, choose a Region.
   Note that you must create your Lambda function in the same Region. After creating the bucket, Amazon S3 opens the Buckets page, which displays a list of all buckets in your account in the current Region.
3. To upload a test object using the Amazon S3 console, On the Objects tab, choose Upload. Drag a test file from your local machine to the Upload page.
4. Choose Upload.

- **Data and Results:**

- **Analysis and Inferences:**

Sample VIVA-VOCE Questions (In-Lab):

1. **What is the purpose of using the line "console.log(message);"?**

Ans:

2. **What is Amazon S3 Replication?**

Ans:

3. **Why we use Amazon S3 getObject AP?**

Ans:

4. **What is source S3 bucket name?**

Ans:

5. What is the purpose of using function blueprint to create the Lambda function?

Ans:

Post-Lab: ***file creation in s3 bucket using lambda Trigger***

Procedure/Program:

STEP1: CREATE A BUCKET IN N.VIRGINIA

STEP2: LAMBDA PYTHON 3.9 SAME REGION (N.VIRGINIA)

CREATE NEW ROLE

-S3 READ ONLY PERMISSION

-LAMBDA BASIC EXECUTION

- ADD PERMISSIONS

-SERVICESES -S3

-LIST ALL

-READ ALL

-WRITE -PUT OBJECT

-RESOURCE ALL

-ALL REMAINING DEFAULT

STEP3: ADD TRIGGER FROM LAMBDA TO S3

-SELECT SAME BUCKET FROM S3

CHECK IN S3 EVENT NOTIFICATION ADDED OR NOT

STEP4: SUCCESSFUL , FILE UPLOADED IN S3 BUCKET AND CHECK LOGS IN LAMBDA

lambdacode:

```
import json
import boto3
s3 = boto3.client('s3')
def lambda_handler(event, context):
    bucket = 'hemanth16999'


    transactionToUpload = {}
    transactionToUpload [ 'facultyName'] = 'DR.Naresh sir'
    transactionToUpload [ 'CourseName'] = 'cloudserverlesscomputing'
    transactionToUpload ['practicalname'] = 'create a file with details which you have provide here'
    transactionToUpload ['Branch'] = 'CSE-CLOUDEDGECOMPUTING'
     fileName = 's3objectnew' + '.json'
    uploadByteStream = bytes(json.dumps(transactionToUpload).encode('UTF-8'))
     s3.put_object(Bucket=bucket, Key=fileName, Body=uploadByteStream)
     print('Put Complete')
```

- **Data and Results:**

- **Analysis and Inferences:**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#8. Experiment Title: write Your Lambda function to retrieves information about this file when you upload it in s3 bucket

Aim/Objective:

To retrieves information about this file when you upload it in s3 bucket.

Description:

When you upload a test file to your new bucket, your Lambda function retrieves information about this file when you test the function from the console. To invoke your function, Amazon S3 needs permission from the function's resource-based policy. When you configure an Amazon S3 trigger in the Lambda console, the console modifies the resource-based policy to allow Amazon S3 to invoke the function if the bucket name and account ID match. If you configure the notification in Amazon S3, you use the Lambda API to update the policy. You can also use the Lambda API to grant permission to another account, or restrict permission to a designated alias. If your function uses the AWS SDK to manage Amazon S3 resources, it also needs Amazon S3 permissions in its execution role.

Pre-Requisites:

Amazon S3 bucket, Lambda function and AWS SDK.

Pre-Lab:

1. What is Auto Scaling in Lambda?

Ans:

2. What is Auto Scaling in Lambda?

Ans:

3. What do you think makes Lambda a timesaving approach?

Ans:

4. What are the best practices for security in Lambda?

Ans:

5. How does Lambda handle failure during event processing?

Ans:

In-Lab:

- **Procedure/Program:**

Task 1: Create bucket and upload file:

5. To create an Amazon S3 bucket using the console, Open the Amazon S3 console. Choose Create bucket.
6. For AWS Region, choose a Region.
   Note that you must create your Lambda function in the same Region. After creating the bucket, Amazon S3 opens the Buckets page, which displays a list of all buckets in your account in the current Region.
7. To upload a test object using the Amazon S3 console, On the Objects tab, choose Upload. Drag a test file from your local machine to the Upload page.
8. Choose Upload.

Task 2: Creating the lambda function:

9. Create the Lambda function, Use a function blueprint to create the Lambda function.
   A blueprint provides a sample function that demonstrates how to use Lambda with other AWS services. Also, a blueprint includes sample code and function configuration presets for a certain runtime. For this tutorial, you can choose the blueprint for the Node.js or Python runtime.
10. To create a Lambda function from a blueprint in the console, Open the Functions page of the Lambda console. Choose Create function. On the Create function page, choose Use a blueprint.
11. Under Blueprints, enter s3 in the search box. In the search results, do one of the following:
    For a Node.js function, choose s3-get-object. For a Python function, choose s3-get-object-python. Choose Configure.
    Under Basic information, do the following:
    For Function name, enter my-s3-function. For Execution role, choose Create a new role from AWS policy templates. For Role name, enter my-s3-function-role. Under S3 trigger, choose the S3 bucket that you created previously. When you configure an S3 trigger using the Lambda console, the console modifies your function's resource-based policy to allow Amazon S3 to invoke the function. Choose Create function.

Task 3: Retrieve the file information:

12. Review the function code
    The Lambda function retrieves the source S3 bucket name and the key name of the uploaded object from the event parameter that it receives. The function uses the Amazon S3 getObject API to retrieve the content type of the object.
    While viewing your function in the Lambda console, you can review the function code on the Code tab, under Code source. The code looks like the following:

Node.js
Python
Example index.js

```
console.log('Loading function');

const aws = require('aws-sdk');

const s3 = new aws.S3({ apiVersion: '2006-03-01' });

exports.handler = async (event, context) => {
   //console.log('Received event:', JSON.stringify(event, null, 2));

   // Get the object from the event and show its content type
   const bucket = event.Records[0].s3.bucket.name;
   const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
'));
   const params = {
      Bucket: bucket,
      Key: key,
   };
   try {
      const { ContentType } = await s3.getObject(params).promise();
      console.log('CONTENT TYPE:', ContentType);
      return ContentType;
   } catch (err) {
      console.log(err);
      const message = `Error getting object ${key} from bucket ${bucket}. Make sure
they exist and your bucket is in the same region as this function.`;
      console.log(message);
      throw new Error(message);
   }
};
```

Note: We have to pass bucket name and object name in the above code based on S3 bucket name and object name.

13. Test in the console
14. Invoke the Lambda function manually using sample Amazon S3 event data.

15. To test the Lambda function using the console : On the Code tab, under Code source, choose the arrow next to Test, and then choose Configure test events from the dropdown list. In the Configure test event window, do the following: Choose Create new test event.

16. For Event template, choose Amazon S3 Put (s3-put). For Event name, enter a name for the test event. For example, mys3testevent.

In the test event JSON, replace the S3 bucket name (example-bucket) and object key (test/key) with your bucket name and test file name. Your test event should look similar to the following:

```
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-west-2",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2":
"EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEF
GH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "my-s3-bucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::example-bucket"
        },
        "object": {
          "key": "HappyFace.jpg",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901"
        }
      }
    }
  ]
}
```

17. Choose Create.

18. To invoke the function with your test event, under Code source, choose Test.

- **Data and Results:**

- **Analysis and Inferences:**

Sample VIVA-VOCE Questions (In-Lab):

1. What is the purpose of using the line "console.log(message);"?

Ans:

2. What is Amazon S3 Replication?

Ans:

3. Why we use Amazon S3 getObject AP?

Ans:

4. What is source S3 bucket name?

Ans:

5.What is the purpose of using function blueprint to create the Lambda function?

Ans:

Post-Lab:

- **Procedure/Program:**
  1. Replicate Data within and between AWS Regions Using Amazon S3 Replication

- **Data and Results:**

- **Analysis and Inferences:**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#9. Experiment Title: Test the Lambda function from the AWS management console.

Aim/Objective:

To test the created function from the AWS management console.

Description:

Testing a Lambda function from the AWS Management Console involves creating a test event and invoking the function to observe its behavior. To do this, navigate to the Lambda service in the AWS Management Console and select the specific Lambda function you want to test. Within the function's details page, locate the "Test" button. Clicking on it allows you to configure a test event.

Pre-Requisites:

AWS management console.

Pre-Lab:

1. What is the purpose of testing a function from the AWS management console?

Ans:

2. How can AWS management console testing help validate the functionality of a function?

Ans:

3. What steps are involved in executing a test from the AWS management console?

Ans:

4. How does testing a function from the console assist in troubleshooting and debugging?

Ans:

5. What key information should you gather before testing a function in the AWS management console?

Ans:

In-Lab:

- **Test the function from the AWS management console**

Post-Lab:

1. **What is the use of AWS management console?**

Ans:

2. **What is the full form of AWS?**

Ans:

3. **How to create a function in AWS?**

Ans:

4. What is an function in AWS?

Ans:

5. How to test an function in AWS?

Ans:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#10. Experiment Title: Inter region transfer of Table in DynamoDB Integration of Lambda with DynamoDB

Aim/Objective:

To perform Inter region transfer of a Table in DynamoDB and to reading/writing Item from or to DynamoDB Table using a serverless AWS service, AWS Lambda.

Description:

Two of the most frequent feature requests for Amazon DynamoDB involve backup/restore and cross-Region data transfer. Today we are addressing both of these requests with the introduction of a pair of scalable tools (export and import) that you can use to move data between a DynamoDB table and an Amazon S3 bucket. The export and import tools use the AWS Data Pipeline to schedule and supervise the data transfer process. The actual data transfer is run on an Elastic MapReduce cluster that is launched, supervised, and terminated as part of the import or export operation.

In other words, you simply set up the export (either one-shot or every day, at a time that you choose) or import (one-shot) operation, and the combination of AWS Data Pipeline and Elastic MapReduce will take care of the rest. You can even supply an email address that will be used to notify you of the status of each operation. Because the source bucket (for imports) and the destination bucket (for exports) can be in any AWS Region, you can use this feature for data migration and for disaster recovery.

Pre-Requisites:

Lambda function, DynamoDB.

Pre-Lab:

1. Can DynamoDB perform atomic updates in place?

Ans:

2. What sort of query functionality is supported by DynamoDB?

Ans:

3. What kinds of secondary indexes does Amazon's DynamoDB support?

Ans:

4. How many global secondary indexes are created for each table?

Ans:

5. How many global secondary indexes are created for each table?

Ans:

In-Lab:

- **Procedure/Program:**

a) Inter region transfer of a Table in DynamoDB

Task 1. Inter region transfer of a Table.

1. Open DynamoDB service with two different locations say one is Ohio and another one is Northern California
2. Click create table on one location
3. Given name for the table, partition key should be ID in Ohio region
4. Under view item, select create item, define ID value as one. Add an attribute message and define it as hello
5. Under global tables, choose and click on Create replica
6. Under regions, choose northern California
7. For creating replication, this will take a few minutes
8. Once completed, you can see the same table which will be available in Northern California
9. Create a new item Ohio region and see how it replicates in Northern California

b) Integration of Lambda with DynamoDB

This lab walks you through the reading and writing Item from or to DynamoDB Table using a serverless AWS service called AWS Lambda. In this lab, we will create a sample Lambda function to read and write Item from or to DynamoDB Table.

Task 1. Create a Table named Users in DynamoDB.

1. In Services, select dynamodb
2. Create Table
3. Specify Table name as *Student*
4. Specify ID as Partition key
5. Create
6. Click on Student Table
7. Actions
8. Click on create item
9. Click on Add new attribute to add string fields like firstname and lastname with values
10. Note- To view Item, click on Explore Table items
11. To copy arn, goto overview Tab in Student Table
12. Copy that arn into notepad

Task 2. Create a role.

13. In services, select IAM and click on Roles
14. Create role
15. Click on Lambda
16. Cilck on Next permission
17. Attach AWS LambdaBasicExecutionRole policy with this role

18. Specify this Role name as LambdaDynamoDBRole
19. Create role
20. To add another policy with this role, click on LambdaDynamoDBRole
21. Click on Add inline Policy
22. Under choose a service, search for DynamoDB
23. Click on Checkbox All DynamoDB actions
24. Click on Add ARN
25. Paste the copied dynamodb arn
26. Click on Add button
27. Click on Review policy
28. Specify name for new policy as DynamoDBReadWriteAccess
29. Click on create policy

Task 3. Create a Lambda function to write and read Item to/from DynamoDB.

30. Select lambda
31. Click on create function
32. Select author from Scratch
33. Specify name as getStudentData
34. Select Runtime as Node.js.14.0
35. Under Existing role, select LambdaDynamoDBRole
36. Click on create function
37. Paste the following code into code area of index.js

```
'use strict'
const AWS = require('aws-sdk');
AWS.config.update({ region: "us-east-1"});
exports.handler = async(event, context) => {
const db = new AWS.DynamoDB({ apiVersion: "2012-10-08"});
const documentClient = new AWS.DynamoDB.DocumentClient({ region: "us-east-
1"});
const params = {
  TableName: "Student",
  Key: {
       id: "9876"
     }
   }
   try {
        const data = await documentClient.get(params).promise();
        console.log(data);
       }
   catch(err) {
            console.log(err);
  }
}
```

Task 4. Test the Lambda Function.

38. Click on Deploy
39. Click on Test to create Test event
40. Select Configure Test event
41. Specify getStudentData as Event name
42. Click on Create
43. Click on Test

Note In case of Writing Item into DynamoDB Table, the following will be the Lambda Code Function

```
'use strict'
const AWS = require('aws-sdk');
AWS.config.update({ region: "us-east-1"});
exports.handler = async(event, context) => {
const db = new AWS.DynamoDB({ apiVersion: "2012-10-08"});
const documentClient = new AWS.DynamoDB.DocumentClient({ region: "us-east-1"});
const params = {
  TableName: "Students",
  Item: {
        id: "9876",
       firstname: “Naween”,
       lastname: “Kumar”
     }
   }
   try {
         const data = await documentClient.put(params).promise();
         console.log(data);
       }
   catch(err) {
            console.log(err);
  }
}
```

- **Data and Results:**

- **Analysis and Inferences:**

Sample VIVA-VOCE Questions (In-Lab):

1. **What is AWS Data Pipeline?**

Ans:

2. **What is Elastic MapReduce?**

Ans:

3. **What is data migration?**

Ans:

4. **What is disaster recovery?**

Ans:

5. What is cross region data transfer?

Ans:

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Post-Lab:

- **Procedure/Program:**

  1. Create and Manage a Nonrelational Database with Amazon DynamoDB

- **Data and Results:**

- **Analysis and Inferences:**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#11. Experiment Title: Configuring a bucket for notifications SQS queue

Aim/Objective:

In this lab, you add a notification configuration to your bucket using an an Amazon SQS queue.

Description:

The Fanout scenario is when a message published to an SNS topic is replicated and pushed to multiple endpoints, such as Kinesis Data Firehose delivery streams, Amazon SQS queues, HTTP(S) endpoints, and Lambda functions. This allows for parallel asynchronous processing.

For example, you can develop an application that publishes a message to an SNS topic whenever an order is placed for a product. Then, SQS queues that are subscribed to the SNS topic receive identical notifications for the new order. An Amazon Elastic Compute Cloud (Amazon EC2) server instance attached to one of the SQS queues can handle the processing or fulfillment of the order. And you can attach another Amazon EC2 server instance to a data warehouse for analysis of all orders received.

You can also use fanout to replicate data sent to your production environment with your test environment. Expanding upon the previous example, you can subscribe another SQS queue to the same SNS topic for new incoming orders. Then, by attaching this new SQS queue to your test environment, you can continue to improve and test your application using data received from your production environment.

Pre-Requisites:

SQS queue, EC2 server.

Pre-Lab:

1. What distinguishes Amazon SQS from Amazon MQ?

Ans:

2. How dependable is the data storage in Amazon SQS?

Ans:

3. What advantages does Amazon SQS have over custom-built or pre-packaged message queuing systems?

Ans:

4. Which of the following is a message queue or transaction system for distributed Internet-based applications?

a) Amazon Simple Notification Service

b) Amazon Elastic Compute Cloud

c) Amazon Simple Queue Service

d) Amazon Simple Storage System

In-Lab:

- **Procedure/Program:**

Tasks 1: add a notification configuration to your bucket using an an Amazon SQS queue.

Step 1: Create an Amazon SQS queue

Step 2: Create an Amazon SNS topic

Step 3: Add a notification configuration to your bucket

Step 4: Test the setup

1. Using the Amazon SQS console, create a queue.

```
{                                                    —
   "Version": "2012-10-17",
   "Id": "example-ID",
   "Statement": [
     {
       "Sid": "example-statement-ID",
       "Effect": "Allow",
       "Principal": {
          "Service": "s3.amazonaws.com"
       },
       "Action": [
          "SQS:SendMessage"
       ],
       "Resource": "SQS-queue-ARN",
       "Condition": {
          "ArnLike": {
             "aws:SourceArn": "arn:aws:s3:::s3snsbucketklu"
// Specify ARN of own created bucket
          },
          "StringEquals": {
             "aws:SourceAccount": "268160201852"
// Specify bucket owner account ID
          }
```

```
        }
      }
    ]
}
```

2. Select standard under Create queue, specify SQS name as SQS-SNS-S3, replace access policy to the queue with the following policy. In it, provide your Amazon SQS ARN, source bucket name, and bucket owner account ID.
3. Leave other detail as they are and click on create queue.
   Note the queue ARN as “arn:aws:sqs:us-east-1:268160201852:SQS-SNS-S3”.
4. Using Amazon SNS console, create a topic. Subscribe to the topic. For this exercise, use email as the communications protocol. Replace the access policy attached to the topic with the following policy. In it, provide your SNS topic ARN, bucket name, and bucket owner's account ID.

```
{
  "Version": "2012-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "Example SNS topic policy",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
        "SNS:Publish"
      ],
      "Resource": "SNS-topic-ARN",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:*:*:bucket-name"
        },
```

```
            "StringEquals": {
                "aws:SourceAccount": "bucket-owner-account-id"
            }
        }
    }
]
}
```

5. Click on Subscription
6. Click on Create subscription, select your Topic arn and also select Email as protocol, specify your email id as endpoint name. Leave others and click on Create subscription
7. Validate Pending confirmation by opening email inbox mail
   Note the topic ARN as “arn:aws:sns:us-east-1:268160201852:sns-sqs-s3:8af40cfe-d37f-4214-ba97-4a321b71e6ca”. The SNS topic you created is another resource in your AWS account, and it has a unique ARN.
8. Using the Amazon S3 console, add a notification configuration requesting Amazon S3 to do the following:
9. Click on Create event notification
10. Select All object creates events checkbox, select specify SQS queue and select SQS arn under SQS queue dropdown menu
11. Click on save changes
12. Publish events of the All object create events type to your Amazon SQS queue.
13. Publish events of the Object in RRS lost type to your Amazon SNS topic.
14. After you save the notification configuration, Amazon S3 posts a test message, which you get via email

- **Data and Results:**

- **Analysis and Inferences:**

Sample VIVA-VOCE Questions (In-Lab):

1. **What is SNS topic**

Ans:

2. **What is SQS queue**

Ans:

3. **What is HTTPS**

Ans:

4. What is EC2 server

Ans:

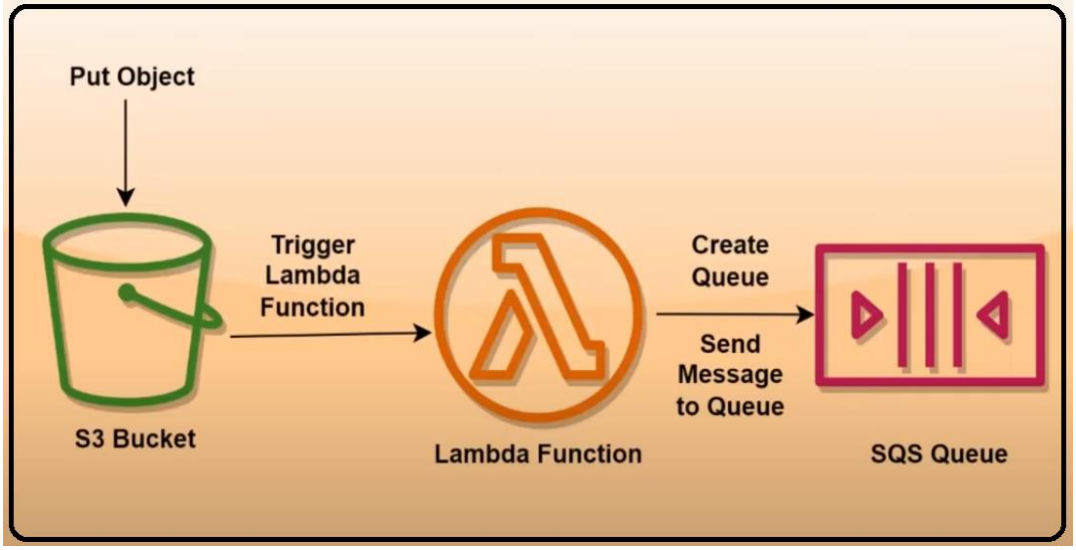
5. What is ARN

Ans:

Post-Lab:

- **Procedure/Program:**

  1. Trigger Lambda Function with S3 Event to Create and Send a message to SQS Queue



- **Data and Results:**

- **Analysis and Inferences:**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#12. Experiment Title: Build serverless application using Athena Architecture

Aim/Objective:

To use Amazon Athena to query data. and check the results of the query.

Description:

Amazon Athena, an interactive query service that makes it easy to search data in Amazon S3 using SQL, was launched at re:Invent 2016. Athena is a serverless service, meaning that you don't need to manage any infrastructure or perform any setup, and you only have to pay for as much as you use. You can store structured data in S3, for example as JSON or CSV, and then simply query that data using SQL, just as if your S3 bucket was a database. In this post, we will cover some details and get you started with Amazon Athena via a simple tutorial that uses Athena as infrastructure as code from a Serverless Framework project.

One of the best reasons for choosing Amazon Athena is that it provides serverless Querying of the data which is stored in Amazon S3 with the help of standard SQL. It also provides support to various data formats like structured, semi-structured and unstructured. Some of the other reasons for choosing Athena over others can be Data Formats - Amazon Athena service works with several different data formats as discussed above. Athena also supports data types like arrays and objects, but when comparing it with Redshift, it does not give support to such data types. So, here Athena edges out as compare to Redshift. User Experience - Coming to the user interface, Amazon Athena provides a simple UI.Getting started with Athena is much more comfortable, all need to do is create a database, select the table name and specify the location of the data on Amazon S3. We can easily add columns in bulk and also easily do the partitioning of the table in Athena, whereas Redshift requires to configure all the cluster properties, and also it takes much time for a cluster to get active. Speed and Performance - As Amazon Athena is serverless, which makes it quicker and easier to execute the queries on Amazon S3 without taking care of the server and the cluster to set up or manage. Another thing is the initialization time, in Athena, we can straight away query the data on Amazon S3, but in Redshift, we have to wait for the cluster to get active and once the cluster is activated, only then we are allowed to query the data.

When comparing to a data warehouse like Amazon Redshift, it should be best chosen when the data is to be taken from several different sources, like retail sales the system, financial systems or any other sources and we have to store the data for a A more extended period to build any report based on that data. The query the engine in Amazon Redshift has been optimized for performing well especially in the use cases where we need to run several complex queries like joining several large datasets. So, when we need to run the queries against extensive structured data and need to apply lots of joins across the tables, and then we should go for Amazon Redshift. But services like Amazon Athea makes it easier to run the interactive queries against the extensive data by directly uploading them in Amazon S3 and don't worry about managing the infrastructure and handling the data. Athena is best suited when we need to run the queries against some weblogs for troubleshooting the issues in the site. In this type of service, we need to define the tables for our data and start querying with standard SQL. Although we can use both the services(i.e., Amazon RedShift and Amazon Athena) Together. This can only be done by keeping the data on Amazon S3 before loading it to Redshift.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

This lab walks you through using Amazon Athena to query data. You'll create a table based on sample data stored in Amazon Simple Storage Service, query the table, and check the results of the query.**Pre-Requisites:**

Athena database, SQL

Pre-Lab:

1. **What is SQL query**

Ans:

2. What is relational database

Ans:

3. What is the speciality of athena database

Ans:

4.How to create a database

Ans:

5.What is structured data

Ans:

In-Lab:

- **Procedure/Program:**

Step 1: Create a Database

You first need to create a database in Athena. To create an Athena database

1. Open the Athena console at https://console.aws.amazon.com/athena/.
   If this is your first time to visit the Athena console in your current AWS Region, choose Explore the query editor to open the query editor. Otherwise, Athena opens in the query editor.
2. Choose View Settings to set up a query result location in Amazon S3.
3. On the Settings tab, choose Manage.
4. For Manage settings, do one of the following:
   In the Location of query result box, enter the path to the bucket that you created in Amazon S3 for your query results. Prefix the path with s3://.
5. Choose Browse S3, choose the Amazon S3 bucket that you created for your current Region, and then choose Choose.
6. Choose Save.
7. Choose Editor to switch to the query editor.
8. On the right of the navigation pane, you can use the Athena query editor to enter and run queries and statements.
9. To create a database named mydatabase, enter the following CREATE DATABASE statement.

```
CREATE DATABASE mydatabase
```

10. Choose Run or press Ctrl+ENTER.
11. From the Database list on the left, choose mydatabase to make it your current database.

Step 2: Create a Table

Now that you have a database, you can create an Athena table for it. The table that you create will be based on sample Amazon CloudFront log data in the location s3://athena-examples-myregion/cloudfront/plaintext/, where myregion is your current AWS Region.

12. In the navigation pane, for Database, make sure that mydatabase is selected.
13. To give yourself more room in the query editor, you can choose the arrow icon to collapse the navigation pane.
14. To create a tab for a new query, choose the plus (+) sign in the query editor. You can have up to ten query tabs open at once.
15. To close one or more query tabs, choose the arrow next to the plus sign. To close all tabs at once, choose the arrow, and then choose Close all tabs.
16. In the query pane, enter the following CREATE EXTERNAL TABLE statement. The regex breaks out the operating system, browser, and browser version information from the ClientInfo field in the log data.

```
CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront_logs (
 `Date` DATE,
 Time STRING,
 Location STRING,
```

```
Bytes INT,
RequestIP STRING,
Method STRING,
Host STRING,
Uri STRING,
Status INT,
Referrer STRING,
os STRING,
Browser STRING,
BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
"input.regex" = "^(?!#)([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^
]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^
]+)\\s+[^\(]+[\(]([^\;]+).*\%20([^\/]+)[\/](.*)$"
) LOCATION 's3://athena-examples-myregion/cloudfront/plaintext/';
```

17. In the LOCATION statement, replace myregion with the AWS Region that you are currently using (for example, us-west-1).
18. Choose Run.
19. The table cloudfront_logs is created and appears under the list of Tables for the mydatabase database.

Step 3: Query Data

Now that you have the cloudfront_logs table created in Athena based on the data in Amazon S3, you can run SQL queries on the table and see the results in Athena. For more information about using SQL in Athena, see SQL Reference for Amazon Athena.

20. Choose the plus (+) sign to open a new query tab and enter the following SQL statement in the query pane.

```
SELECT os, COUNT(*) count
FROM cloudfront_logs
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os
```

21. Choose Run.
22. To save the results of the query to a .csv file, choose Download results.
23. To view or run previous queries, choose the Recent queries tab.
24. To download the results of a previous query from the Recent queries tab, select the query, and then choose Download results. Queries are retained for 45 days.

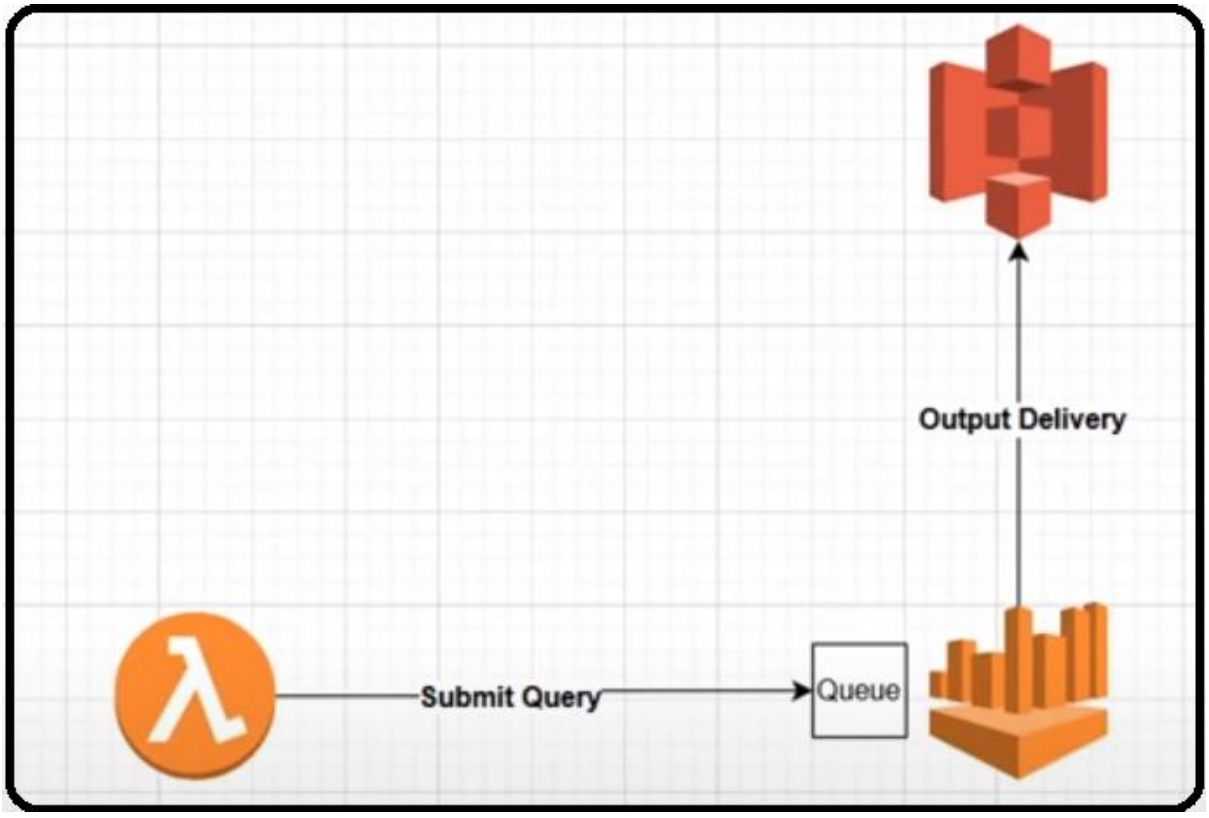
- **Data and Results:**

- **Analysis and Inferences:**

Post-Lab:

1. How to use SQL to Query S3 files with AWS Athena?

- **Procedure/Program:**



- **Data and Results:**

- **Analysis and Inferences:**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#13. Experiment Title: Implementation of Autoscaling to Manage two different target groups with each having atleast two target group members with load balancer and Deploy a Web Application.

Aim/Objective:

To Implement Autoscaling for Managing two different target groups with each having atleast two target group members with load balancer and Deploy a Web Application.

Description:

This experiment aims to address the challenge of efficiently managing and scaling web applications that have diverse components and varying traffic loads. At its core, this work involves the utilization of autoscaling, a dynamic provisioning technique that automatically adjusts the number of computing resources to match the demand. It specifically focuses on two different target groups, each with a minimum of two members, which represent different components or microservices of the web application. A load balancer is employed to distribute incoming traffic evenly across these target groups. The key objectives of this experiment include Achieving high availability, optimizing resource utilization, ensuring effective load distribution and Simplifying management. This work explores a critical aspect of modern web application deployment, demonstrating how autoscaling and load balancing can be employed to enhance performance, reliability, and cost-effectiveness in cloud-based environments.

Pre-Requisites:

AWS Management console and autoscaling and loadbalancing services on EC2

Pre-Lab:

1. Why is autoscaling important in managing web applications with varying traffic loads?

Ans:

2. How does the use of multiple target groups enhance load balancing capabilities?

Ans:

3. What are the criteria for determining the minimum number of target group members in each group?

Ans:

4. What is the role of a load balancer in the deployment of web applications?

Ans:

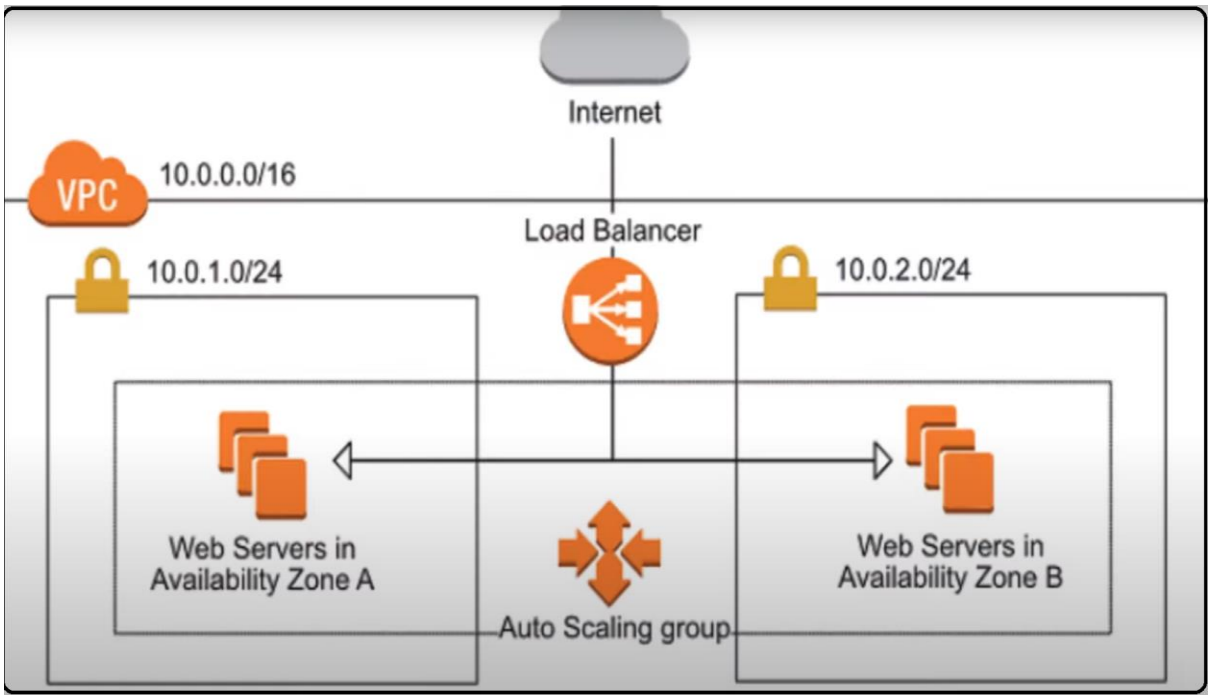
5. What factors should be considered when configuring autoscaling policies for web applications?

Ans:

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

In-Lab: Implementation of Autoscaling to Manage two different target groups with each having atleast two target group members with load balancer and Deploy a Web Application.

Architecture:



Write Procedure:

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Post-Lab:

1. **What is an Web Application?**

   Ans:

2. **What is Autoscaling?**

   Ans:

3. **What is load balancer?**

   Ans:

4. **What are the main advantages of AWS?**

   Ans:

5. **What are the main disadvantages of AWS?**

   Ans:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#14. Experiment Title: Implement power of maths calculator application using lambda

Aim/Objective:

To implement power of maths calculator application using lambda.

Description:

This implementation is a fascinating endeavor that combines the power of mathematics with serverless computing to create a versatile and efficient tool for mathematical calculations. Lambda, a serverless computing service offered by cloud providers like AWS, enables the execution of code in response to specific events without the need for managing traditional servers. In this context, Lambda functions are employed to execute mathematical operations requested by users within the calculator application. The "Power of Maths Calculator" is designed to perform a wide range of mathematical operations, such as arithmetic, algebraic, statistical, and even complex mathematical functions. Users input their mathematical expressions or formulas, and the Lambda functions process these inputs and return the results in real-time. The application can handle complex calculations, including solving equations, graphing functions, and performing advanced mathematical tasks. By utilizing Lambda, this calculator application benefits from auto-scaling, cost-efficiency, and high availability. It ensures that computing resources are allocated only when needed, reducing operational costs and providing a seamless user experience. Additionally, Lambda allows for easy scalability to accommodate varying workloads and user demands. This work leverages serverless computing to provide users with a powerful and responsive tool for performing diverse mathematical calculations, demonstrating the synergy between cloud-based technologies and mathematical problem-solving.

Pre-Requisites:

Lambda function.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Pre-Lab:

1. How can AWS Lambda be used to create a "Power of Maths" calculator application?

Ans:

2. What mathematical functions or operations will be included in the calculator application?

Ans:

3. What are the key features that Lambda provides for building this calculator?

Ans:

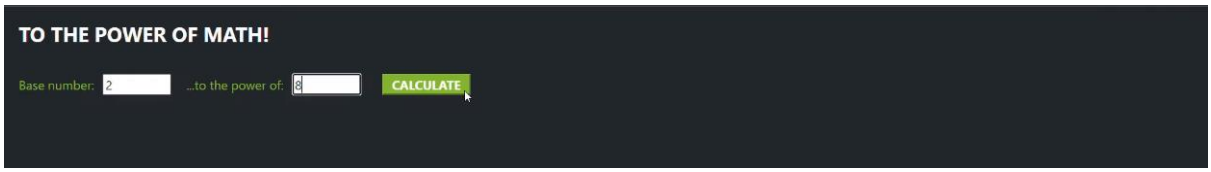
4. What kind of input and output formats are expected for this calculator application?

Ans:

5. What are the anticipated benefits of using Lambda for implementing this mathematical calculator?

Ans:

In-Lab: Implement to the power of maths calculator application using lambda



Post-Lab

1. **What is the use of lambda function?**

Ans:

2. **What is an web application?**

Ans:

3. **How to create/test an lambda function**

Ans:

4. What is the full form of AWS?

Ans:

5. What is an power of math calculator?

Ans:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#15. Experiment Title: List all the EC2 instances and Status across all regions in an AWS account using Python boto3

Aim/Objective: To List out all the EC2 instances and Status across all regions in an AWS account using Python boto3

Description:

This implementation is a powerful demonstration of how automation and cloud management can be streamlined for enhanced visibility and control in an Amazon Web Services (AWS) environment. Boto3 is a Python library that provides an easy and programmatic way to interact with AWS services, making it an excellent choice for this endeavor. To accomplish this task, the Python script would follow steps such as Authentication, Region Discovery, Iteration through Regions, Gathering Instance Information, Compilation of Results and finally Output or Action. This approach provides administrators and AWS users with a consolidated view of their EC2 instances across all regions, facilitating efficient monitoring and management, and can be an invaluable tool for large-scale AWS deployments.

Pre-Requisites:

This Section contains the list of Software/Tools or required knowledge (Glossary) to complete the task under the Laboratory Session.

Pre-Lab:

1. What is the objective of listing all EC2 instances and their status across AWS regions?
2. How does Python's boto3 library facilitate interaction with AWS services?
3. What are the necessary credentials and permissions needed to access EC2 instances across regions?
4. What Python scripts or functions will be utilized to achieve this task using boto3?
5. How can the gathered information about EC2 instances be useful for AWS account management?

In-Lab: List all the EC2 instances and Status across all regions in an AWS account using Python boto3

Post-Lab:

1. **What is an instance?**

Ans:

2. **What is an slot?**

Ans:

3. **What is an intent?**

Ans:

4. What is an Availability region?

Ans:

5. What is the full form of EC2?

Ans:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#16. Experiment Title: Create an SNS topic, subscribe an endpoint to the SNS Topic, publish a message to the SNS Topic, Check receipt of message, Delete the Subscription and the SNS topic

Aim/Objective:

To perform Create/subscribe/publish/Delete operations on the SNS topic.

Description:

You can use an AWS Lambda function to process records in an Amazon DynamoDB stream. With DynamoDB Streams, you can trigger a Lambda function to perform additional work each time a DynamoDB table is updated.

Lambda reads records from the stream and invokes your function synchronously with an event that contains stream records. Lambda reads records in batches and invokes your function to process records from the batch. Lambda polls shards in your DynamoDB stream for records at a base rate of 4 times per second. When records are available, Lambda invokes your function and waits for the result. If processing succeeds, Lambda resumes polling until it receives more records.

By default, Lambda invokes your function as soon as records are available in the stream. If the batch that Lambda reads from the stream only has one record in it, Lambda sends only one record to the function. To avoid invoking the function with a small number of records, you can tell the event source to buffer records for up to five minutes by configuring a batch window. Before invoking the function, Lambda continues to read records from the stream until it has gathered a full batch, or until the batch window expires.

If your function returns an error, Lambda retries the batch until processing succeeds or the data expires. To avoid stalled shards, you can configure the event source mapping to retry with a smaller batch size, limit the number of retries, or discard records that are too old. To retain discarded events, you can configure the event source mapping to send details about failed batches to an SQS queue or SNS topic.

You can also increase concurrency by processing multiple batches from each shard in parallel. Lambda can process up to 10 batches in each shard simultaneously. If you increase the number of concurrent batches per shared, Lambda still ensures in-order processing at the partition-key level.

Configure the Parallelization, Factor setting to process one shard of a Kinesis or DynamoDB data stream with more than one Lambda invocation simultaneously. You can specify the number of concurrent batches that Lambda polls from a shard via a parallelization factor from 1 (default) to 10. For example, when Parallelization Factor is set to 2, you can have 200 concurrent Lambda invocations at maximum to process 100 Kinesis data shards. This helps scale up the processing throughput when the data volume is volatile and the Iterator Age is high. Note that parallelization factor will not work if you are using Kinesis aggregation.

Pre-Requisites:

Lambda function, DynamoDB.

Pre-Lab:

1. What is AWS Simple Notification Service?

Ans:

2. What are some common use cases for SNS?

Ans:

3. How do you create a topic in SNS?

Ans:

4. How do you subscribe to an SNS topic?

Ans:

5. How do you send a message to an SNS topic?

Ans:

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

In-Lab:

- **Procedure/Program:**

Task 1: Create/subscribe/publish/Delete operations on the SNS topic.

1. Type SNS in the search bar, under Topic name, input yourTopicName say as Topic1
2. Click on next step
3. Select Standard for using email subscription protocol
4. Specify Topic name as Topic1 and display name as Best Wishes (Optional)
5. Specify optional parameters such as Encryption, Access policy, Delivery retry policy, Delivery status logging, and Tags if need to customize
6. Click on create topic
7. Click on Subscriptions under SNS Dashboard
8. After validating pending confirmation, click on Topic
9. Click on Topic1
10. Click on Publish message, input subject of the message and TTL value (optional)
11. Select message structure as Identical payload for all delivery protocols, also write the content of the message body to be sent to the endpoint
12. Specify message attributes such as timestamps, geospatial data, signatures, and identification for the message (optional)
13. Click on Publish message
14. Delete subscription by selecting the created subscription id and click on Delete option
15. Click on Topics, select the topic that you want to delete and click on Delete option Again, click on Delete option

- **Data and Results:**

- **Analysis and Inferences:**

Sample VIVA-VOCE Questions (In-Lab):

1. **What is Amazon SNS?**

Ans:

2. **What is Amazon SQS?**

Ans:

3. What is Delivery retry policy?

Ans:

4. What is Delivery status logging?

Ans:

5. What is endpoint?

Ans:

Post-Lab:

- **Procedure/Program:**

1. Filter Messages Published to Topics with Amazon SNS and Amazon SQS

- **Data and Results:**

- **Analysis and Inferences:**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#17. Experiment Title:Build a real-time Streaming App with Amazon Kinesis

Aim/Objective: To Introduce of Kinesis by using notification Services

Description:

The introduction of Amazon Kinesis as a notification service marks a powerful and versatile addition to the realm of real-time data processing and event-driven architecture. While Amazon Kinesis is traditionally known for its capabilities in streaming and analyzing large volumes of data, it can also serve as a highly effective notification service when appropriately configured. Amazon Kinesis excels in handling data streams, making it well-suited for scenarios where timely notifications are crucial. By using Kinesis as a notification service, organizations can set up event-driven workflows that respond to streaming data in real-time. It works through Data Ingestion, Processing and Filtering, Notification Generation, Integration with Other Services and finally Scalability/Reliability. In summary, leveraging Amazon Kinesis as a notification service enables organizations to harness the power of real-time data streams for proactive alerting, event detection, and automated responses, thereby enhancing operational efficiency and enabling timely decision-making. This approach is particularly valuable in use cases where instantaneous notifications are critical, such as in monitoring, security, and real-time analytics.

Pre-Requisites:

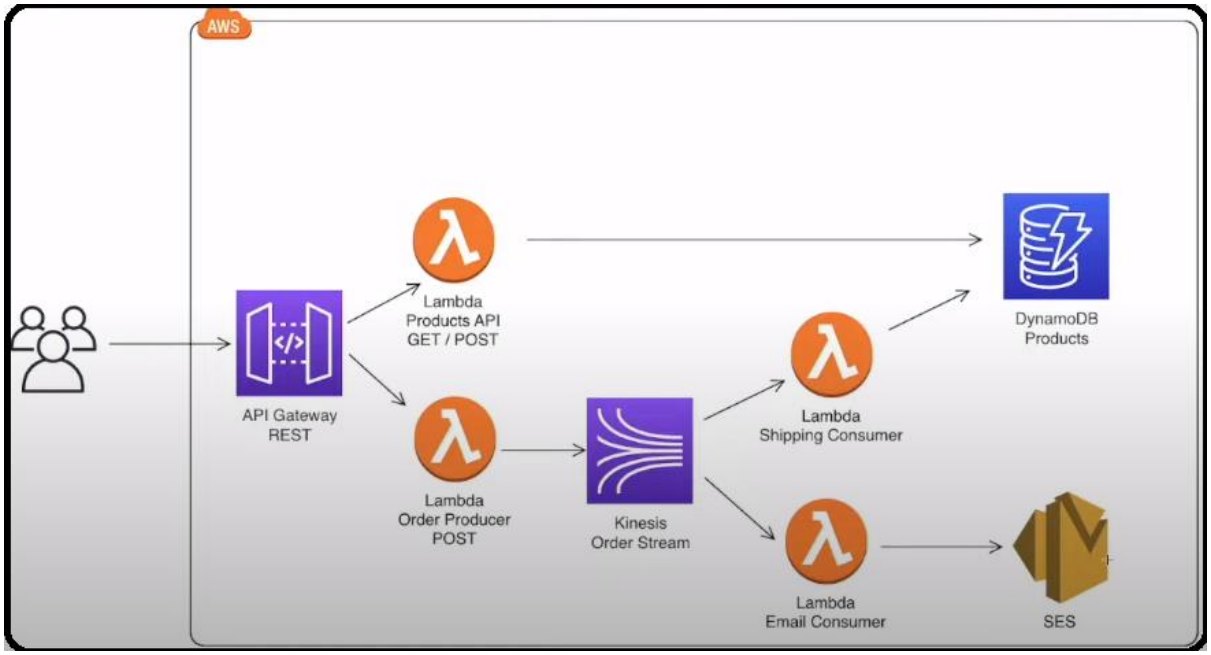
This Section contains the list of Software/Tools or required knowledge (Glossary) to complete the task under the Laboratory Session.

Pre-Lab:

1. What is the primary purpose of using Amazon Kinesis as a notification service?
2. How does Kinesis enable real-time data streaming for notifications?
3. What are the key components of Kinesis that make it suitable for notifications?
4. How can Kinesis be integrated with other AWS services for notification delivery?
5. What advantages does Kinesis offer over traditional notification services for handling large volumes of data?

In-Lab: Build a real-time Streaming App with Amazon Kinesis?

Architecture:



- **Procedure/Program:**

Post Lab:

1. **What is Kinesis?**

Ans:

2. **What is an notification service?**

Ans:

3. **What is meant by serverless cloud?**

Ans:

4. What is the difference between an data server and cloud?

Ans:

5. What are the applications of kinesis?

Ans:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#18. Experiment Title: Build serverless application using Athena , S3 , Glue and Quicksight Architecture

Aim/Objective: To build an serverless application using Athena , S3 , Glue and Quicksight Architecture.

Description:

Building a serverless application using Amazon Athena, Amazon S3, AWS Glue, and Amazon QuickSight architecture is a streamlined and cost-effective approach to create a data-driven application for analytics, reporting, and visualization. This architecture leverages serverless and managed services to handle data storage, processing, transformation, querying, and visualization. Amazon S3 serves as the central data repository, where you store your data in various formats like CSV, JSON, or Parquet. This scalable and durable object storage is designed to handle large datasets. AWS Glue is used for data ETL (Extract, Transform, Load) operations. It automatically discovers, catalogues, and transforms the data in your S3 buckets. It can convert data into query-friendly formats and schemas, making it easier for querying. Amazon Athena is a serverless query service that enables you to run SQL queries directly on your data stored in S3. It doesn't require any infrastructure management, and it's highly flexible for ad-hoc queries. Amazon QuickSight is a serverless, cloud-native BI (Business Intelligence) service for data visualization and dashboard creation. It connects to Athena and uses its query results to create interactive, visually appealing reports and dashboards. This architecture offers several advantages, including cost-efficiency through serverless services, scalability, and ease of use. It's particularly well-suited for organizations seeking to harness their data for insights, as it allows them to handle everything from data storage to reporting without the need for complex infrastructure management. It's a powerful solution for data-driven decision-making and reporting needs.

Pre-Requisites:

This Section contains the list of Software/Tools or required knowledge (Glossary) to complete the task under the Laboratory Session.

Pre-Lab:

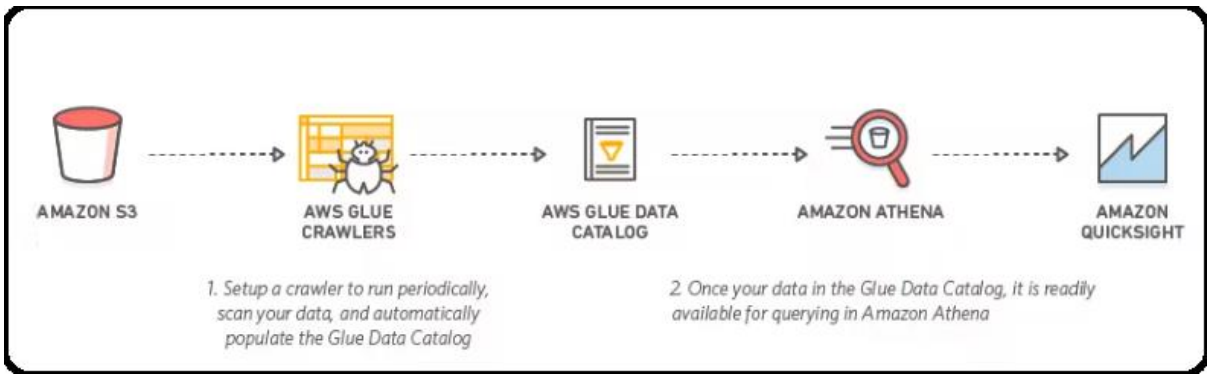
1. How can serverless architecture enhance the scalability of data applications?
2. What role does Amazon Athena play in querying data stored in Amazon S3?
3. How does AWS Glue assist in data preparation and transformation?
4. What is the purpose of Amazon QuickSight in this architecture?
5. What benefits does this architecture offer for cost-effective data analytics and visualization?

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

In-Lab:

Build serverless application using Athena , S3 , Glue and Quicksight Architecture

Architecture:



- **Procedure/Program:**

Post Lab:

1. **What is an Amazon S3 Bucket?**

Ans:

2. **What is the use of an Glue Crawler?**

Ans:

3. **What is an AWS Glue Data Catalog?**

Ans:

4. What is the purpose of Amazon Athena?

Ans:

5. List out the applications of Amazon Quicksight Dashboard?

Ans:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#19. Experiment Title: Build end to end serverless web application for Get and post method using CRUD operations

Aim/Objective: To build an end to end serverless web application for Get and post method using CRUD operations

Description:

Building an end-to-end serverless web application for CRUD (Create, Read, Update, Delete) operations is a robust approach to creating scalable and cost-effective web applications that interact with data. Leveraging serverless architecture components, such as AWS Lambda, Amazon API Gateway, and a NoSQL database like Amazon DynamoDB, can enable the development of a highly responsive and dynamic application.

1. **Create (POST):** To add new data records, a serverless function (AWS Lambda) can receive incoming POST requests from the web application. The data is then processed and stored in a NoSQL database like DynamoDB. The serverless function's scalability ensures it can handle varying levels of data entry.

2. **Read (GET):** The GET method retrieves data from the serverless application. When a user requests data, the application triggers a serverless function to query the database (DynamoDB). The queried data is then returned to the user, providing real-time access to information.

3. **Update (PUT/PATCH):** To modify existing data, a serverless function is invoked upon receiving PUT or PATCH requests. The function updates the specific record in the database, ensuring the application maintains data accuracy.

4. **Delete (DELETE):** When a DELETE request is received, the serverless function removes the designated record from the database. This process ensures that data management is both efficient and controlled.

By implementing these CRUD operations in a serverless web application, you can achieve several benefits, including auto-scaling, cost optimization, reduced infrastructure management, and rapid development. This architecture allows you to focus on application logic and functionality rather than dealing with server provisioning and maintenance. It's an excellent approach for developing responsive and efficient web applications that handle data seamlessly.

Pre-Requisites:

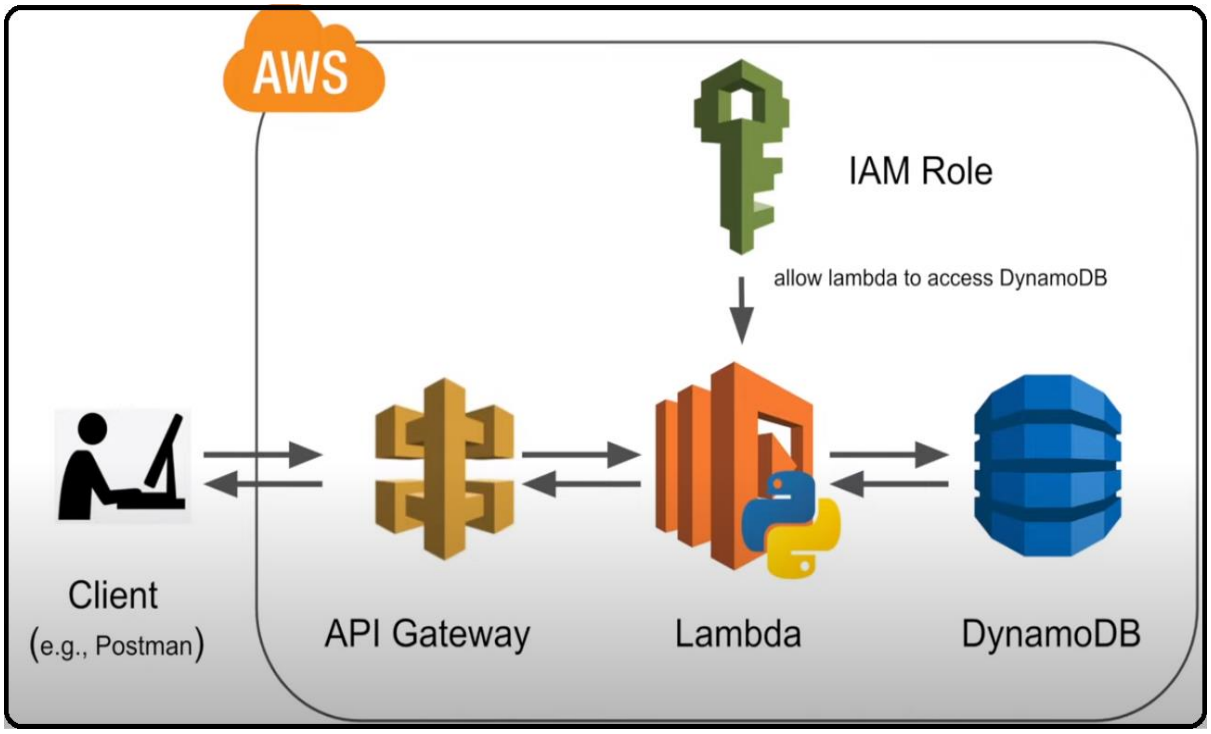
This Section contains the list of Software/Tools or required knowledge (Glossary) to complete the task under the Laboratory Session.

Pre-Lab:

1. What are the key components required for building an end-to-end serverless web application for CRUD operations?
2. How do GET and POST methods play a role in the application's CRUD functionality?
3. What is the significance of serverless architecture in developing this web application?
4. How can serverless functions ensure scalability and cost efficiency for CRUD operations?
5. What benefits does an end-to-end serverless approach offer in simplifying application development and management?

In-Lab: Build end to end serverless web application for Get and post method using CRUD operations

Architecture:



- **Procedure/Program:**

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Post Lab:

1. What is end to end serverless web application?

Ans:

2. List out the CRUD operations?

Ans:

3. What is an Get and post method?

Ans:

4. What is the full form of CRUD?

Ans:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

#20. Experiment Title: Build Real time CICD Serveless web Application

Aim/Objective: To build an Real time CICD Serveless web Application.

Description:

Building a real-time Continuous Integration and Continuous Deployment (CI/CD) serverless web application is an advanced and efficient approach to developing, deploying, and managing web applications that automatically adapt to changing requirements and deliver updates in real-time. Here's a high-level overview of this process:

1. **Continuous Integration (CI):** Developers commit code changes to a version control system (e.g., Git). A CI system, such as AWS CodePipeline or Jenkins, automatically builds and tests the code. For serverless applications, this may involve packaging AWS Lambda functions and other resources.

2. **Continuous Deployment (CD):** After successful CI, the CD pipeline deploys the updated application to a serverless environment. AWS Lambda, API Gateway, and other services can be used to host the application components. This serverless architecture ensures efficient scaling and automatic management of resources.

3. **Real-time Functionality:** Real-time features can be integrated using services like AWS WebSocket API, AWS AppSync, or Serverless Framework plugins. These services facilitate real-time communication, data synchronization, and instant updates within the web application.

4. **Monitoring and Logging:** Implementing robust monitoring and logging using services like Amazon CloudWatch and AWS X-Ray is crucial. This enables the detection of issues, troubleshooting, and performance optimization.

5. **Auto-scaling and High Availability:** Serverless architecture inherently provides auto-scaling and high availability. The application dynamically adjusts to changing loads, ensuring responsiveness and reliability.

6. **Security:** Security measures should be in place, including identity and access management (IAM) for controlling access to resources and data encryption in transit and at rest.

A real-time CI/CD serverless web application enables developers to rapidly iterate, deploy updates instantly, and provide a responsive and dynamic user experience. It leverages the benefits of serverless computing, reducing operational overhead and ensuring scalability and high availability. This approach is particularly well-suited for modern web applications requiring real-time features and agile development practices.

Pre-Requisites:

This Section contains the list of Software/Tools or required knowledge (Glossary) to complete the task under the Laboratory Session.

Pre-Lab:

1. What is the primary objective of building a real-time CI/CD serverless web application?

Ans:

2. How does serverless architecture enhance the CI/CD pipeline for web applications?

Ans:

3. What are the key components required for implementing real-time CI/CD in a serverless context?

Ans:

4. How can real-time features improve the deployment and testing processes?

Ans:

5. What advantages does a real-time CI/CD serverless web application offer for development and deployment speed?

Ans:

In-Lab: Develop Serverless CICD using SAM

Procedure:

AWS SAM CLI Installation: https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/install-sam-cli.html#install-sam-cli-instructions

Initialize the sample Hello World application

----

cmd -> sam --version

cmd -> sam init

cmd -> 1 (quick start)

cmd -> 1 (hello world app)

cmd -> y (xray tracing)

cmd -> y (cloudwatch Application Insights)

cmd -> sam-app (application name)

cmd -> cd sam-app

cmd -> tree

Some important files to highlight:

----

hello_world/app.py – Contains your Lambda function code.

hello_world/requirements.txt – Contains any Python dependencies that your Lambda function requires.

samconfig.toml – Configuration file for your application that stores default parameters used by the AWS SAM CLI.

template.yaml – The AWS SAM template that contains your application infrastructure code.

Build your application

----

python 9 Download: https://www.python.org/downloads/release/python-390/

//python 3.9 required

python --version

sam build

Some important files to highlight:

----

build/HelloWorldFunction – Contains your Lambda function code and dependencies. The AWS SAM CLI creates a directory for each function in your application.

build/template.yaml – Contains a copy of your AWS SAM template that is referenced by AWS CloudFormation at deployment.

build.toml – Configuration file that stores default parameter values referenced by the AWS SAM CLI when building and deploying your application.

Deploy your application to the AWS Cloud

----

aws configure

sam deploy --guided

Run your application

----

sam list endpoints --output json

browser -> https://8m64y8f2m7.execute-api.ap-south-1.amazonaws.com/Prod/hello

Modify and sync your application to the AWS Cloud

----

sam sync --watch

To modify and sync your application

In your IDE of choice, open the sam-app/hello_world/app.py file.

Change the message and save your file. The following is an example:

```
import json
...
def lambda_handler(event, context):
   ...
   return {
      "statusCode": 200,
      "body": json.dumps({
         "message": "hello everyone!",
         ...
      }),
   }
```

Run your application

----

browser -> https://8m64y8f2m7.execute-api.ap-south-1.amazonaws.com/Prod/hello

(Optional) Test your application locally (requires docker)

----

sam local invoke

To host your API locally

----

sam local start-api

Run your application locally

----

browser ->http://localhost/hello

Delete your application from the AWS Cloud

----

sam delete

Result:

Post-Lab:

1. **Differentiate between end to end serverless and real time CICD serverless web applications.**

Ans:

2. **What is an Real time CICD Serveless web Application?**

Ans:

3. **What is the full form of SAM?**

Ans:

4. What is end to end serverless web application?

Ans:

5. What is the full form of CICD?

Ans:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.