

PROGRAM 1: Exploring AWS CloudShell and the AWS Cloud9 IDE

Steps for creating CloudShell Step 1:

Login to AWS Account

Step 2: Open CloudShell

Step 3: Execute shell commands in the terminal

Step 4: Choose Download file from Actions dropdown menu

Step 5: Provide the path to the file created (demo.txt)

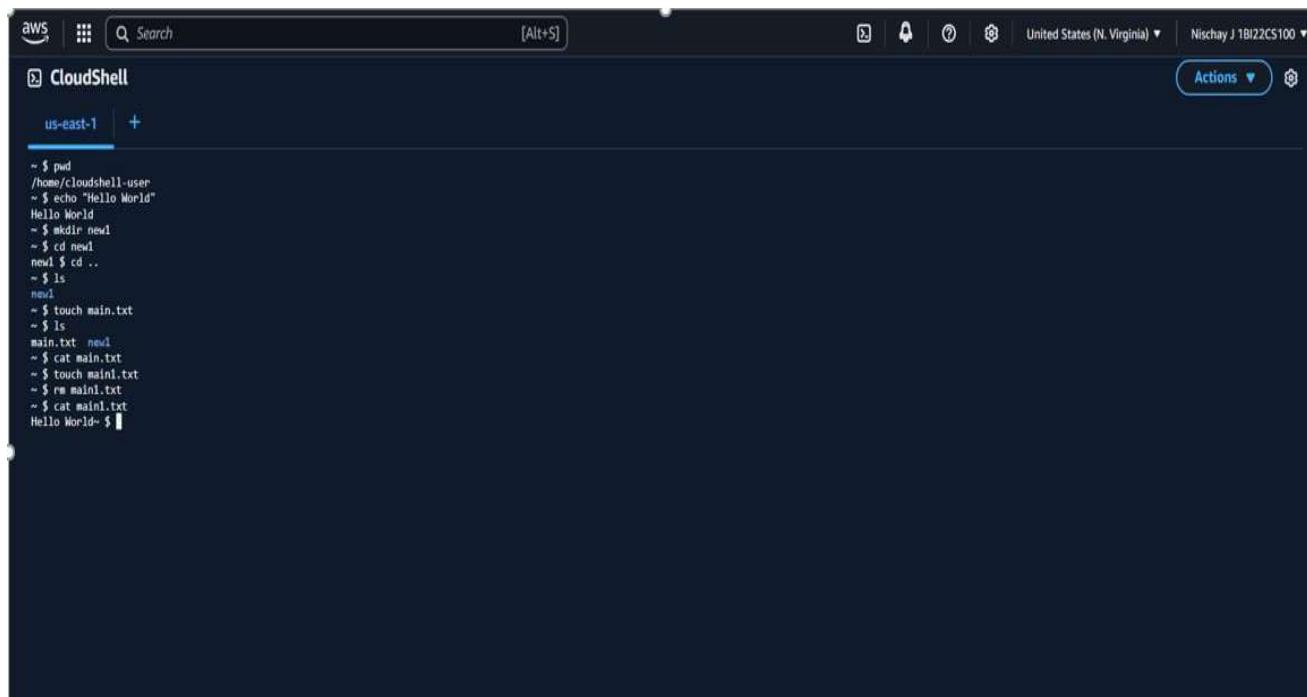
Step 6: An empty file “demo.txt” is downloaded

Step 7: Add content in the downloaded file(demo.txt) and save

Step 8: Execute ‘rm’ command. To upload file, click on Upload file option from the Actions dropdown menu

Step 9: Upload demo.txt file

Step 10: Once uploaded successfully, check the contents of demo.txt using the ‘cat’ command



The screenshot shows the AWS CloudShell interface. At the top, there's a navigation bar with the AWS logo, a search bar, and account information for 'United States (N. Virginia)' and 'Nischay J 1B122CS100'. Below the navigation bar is a header with 'CloudShell' and a dropdown for 'us-east-1'. The main area is a terminal window displaying the following command history:

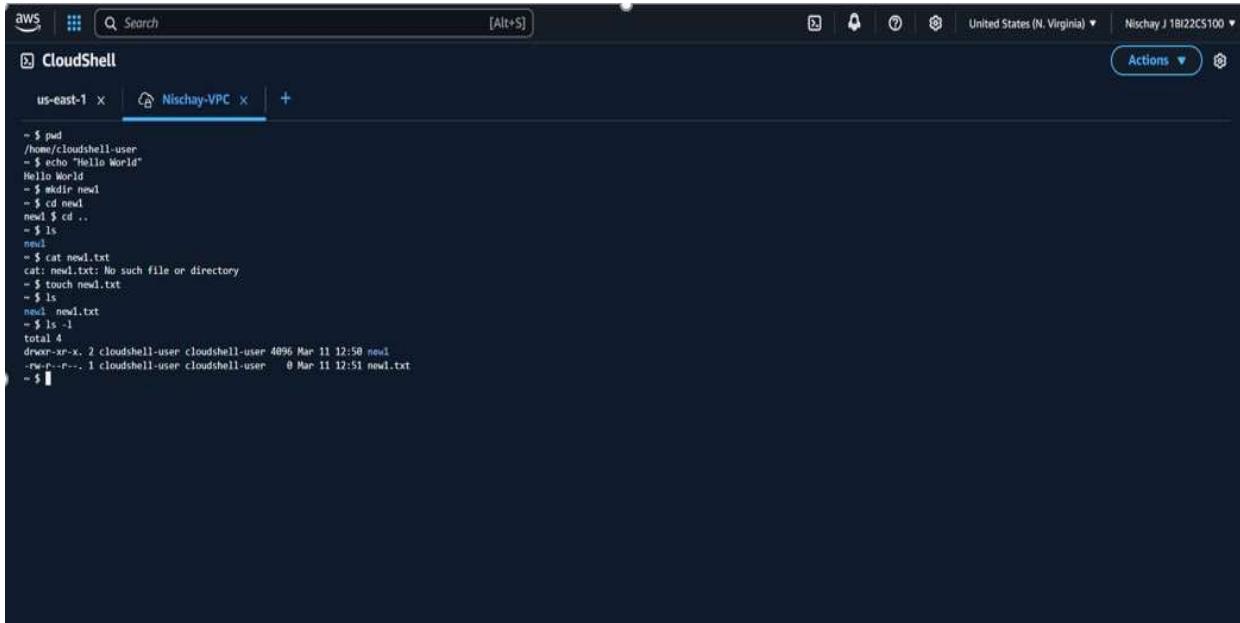
```
~ $ pwd
/home/cloudshell-user
~ $ echo "Hello World"
Hello World
~ $ mkdir new1
~ $ cd new1
new1 $ cd ..
~ $ ls
new1
~ $ touch main.txt
~ $ ls
main.txt new1
~ $ cat main.txt
~ $ rm main1.txt
~ $ cat main1.txt
Hello World- $
```

Steps for creating VPC Environment

Step 1: From the Actions menu choose ‘Create VPC environment’

Step 2: Give a VPC name and choose VPC, Subnet and the default security group. Click on Create

Step 3: Execute the same commands as of CloudShell in the VPC window except for download and upload file options



The screenshot shows the AWS CloudShell interface. At the top, there's a navigation bar with the AWS logo, a search bar, and various icons. Below it, the title 'CloudShell' is displayed, followed by a tab bar with 'us-east-1' and 'Nischay-VPC'. The main area is a terminal window showing a series of commands and their outputs:

```
- $ pwd  
/home/cloudshell-user  
- $ echo "Hello World"  
Hello World  
- $ cd newl  
- $ ls newl  
newl $ cd ..  
- $ ls  
newl  
- $ cat newl.txt  
cat: newl.txt: No such file or directory  
- $ touch newl.txt  
- $ ls  
newl newl.txt  
- $ ls -l  
total 4  
drwxr-xr-x. 2 cloudshell-user cloudshell-user 4096 Mar 11 12:58 newl  
-rw-r--r--. 1 cloudshell-user cloudshell-user 0 Mar 11 12:51 newl.txt  
- $
```

Step 4: Once all the commands are executed delete the VPC

Step 5: Also delete the CloudShell once the commands are executed

Steps to create EC2 instance

Step 1: Open EC2 in AWS Console and click on Launch Instance

Step 2: Name the instance and proceed below

Step 3: Click on create new key pair .Create a new key pair by providing a name

Step 4: Keep the default options for the rest and click on Launch Instance

Step 5: Confirmation of the launch of our new instance

Step 6: Click on Instances. It displays the instances that are running. Click on Instance ID to know about a particular instance.

Step 7: Click on the Connect option on the top-right to connect to an instance

Step 8: Keep the default options and click on Connect

Step 9: An Amazon-Linux terminal is displayed

Step 10: Execute the commands executed in VPC in this terminal

Step 11: The CloudShell button on the bottom-left corner can be clicked to open a shell terminal. We can execute commands and create VPC here

The screenshot shows a CloudShell interface on an AWS account. The terminal window displays a Linux command-line session:

```
~$ pwd
/home/cloudshell-user
~$ echo "Hello World"
Hello World
~$ mkdir new1
~$ cd new1
new1$ cd ..
~$ ls
new1
~$ cat main1.txt
cat: main1.txt: No such file or directory
~$ touch main1.txt
~$ ls
main1.txt  new1
~$ ls -l
total 4
-rw-r--r-- 1 cloudshell-user cloudshell-user 0 Mar 11 13:00 main1.txt
drwxr-xr-x  2 cloudshell-user cloudshell-user 4096 Mar 11 12:59 new1
~$ rm main1.txt
~$ ls
new1
~$
```

At the bottom of the terminal, there is a prompt: `~$`.

Step 12: Delete the EC2 instance after execution by clicking on Terminate(delete) instance

Step 13: Indicates the successful termination of the instance. Also delete the corresponding key pair and Security groups associated with this instance

User can sign-out of the AWS account after these steps.

PROGRAM 2: Working with DynamoDB which makes use of PartiQL

Steps for creation of DynamoDB User

Step 1: Login to AWS account. Search for DynamoDB

Step 2: Select “Tables” from the left panel and click on “Create Table”

Step 3: Enter the details by giving the table name as “student” and partition key as “USN”

Step 4: Select “Customize Settings” under Table Settings and select “DynamoDB standard- IA” under Table Class

Step 5: Click on “Create Table”. Once the table is created successfully click on the highlighted table link

Step 6: Click on the “Explore table items”

Step 7: Click on “Create item”

The screenshot shows the 'Create item' interface in the AWS DynamoDB console. The table 'student' is selected. The 'Attributes' section contains the following data:

Attribute name	Value	Type
USN - Partition Key	1B122CS100	String
Name	Nischay	String
Department	CSE	String
Semester	6	Number
Section	B	String
Batch	2022	Number

Buttons at the bottom right include 'Cancel' and 'Create item' (which is highlighted).

Step 8: Create a new item by filling up the Attributes such as Name, Dept, Sem, Sec, Batch. And click on “Create item”

Step 9: The Table shows the item created

Step 10: Create another item using JSON view (Java Script Object Notation)

Step 11: Create a total of 4-5 items using Form or JSON view

The screenshot shows the AWS DynamoDB 'Explore Items' interface for the 'student' table. The table has the following data:

	USN (String)	Batch	Department	Name	Section	Semester
1	1B022CS132	2022	CSE	Ajith	B	6
2	1B022CS080	2022	CSE	Lakshay	B	6
3	1B022CS127	2022	CSE	Rishav	B	6
4	1B022CS118	2022	CSE	Prince	B	6
5	1B022CS100	2022	CSE	Nischay	B	6

Step 12: Click on the “PartiQL editor” on the left panel. Click on the 3 dots next to the students and it will provide ways in the table can be viewed

Step 13: If we click on “scan table” the whole table will be visible

The screenshot shows the AWS DynamoDB 'PartiQL editor' interface. A query is run against the 'student' table:

```
1 SELECT * FROM "student"
```

The results show the same 5 items from the previous screenshot:

	USN (String)	Batch	Department	Name	Section	Semester
1	1B022CS132	2022	CSE	Ajith	B	6
2	1B022CS080	2022	CSE	Lakshay	B	6
3	1B022CS127	2022	CSE	Rishav	B	6
4	1B022CS118	2022	CSE	Prince	B	6
5	1B022CS100	2022	CSE	Nischay	B	6

Step 14: Click on “Query table” -> Clear ->Replace the USN value with any of the given values->Run

Step 15: Now Click on “Set item”, once it is executed, select “scan table” for the updated table

Step 16: Click on “Drop item”, and once again select the “scan table” for the updated table

Step 17: Delete the table “student”, by clicking on “Delete” under “Actions”

PROGRAM 3 : Developing REST APIs with Amazon API Gateway

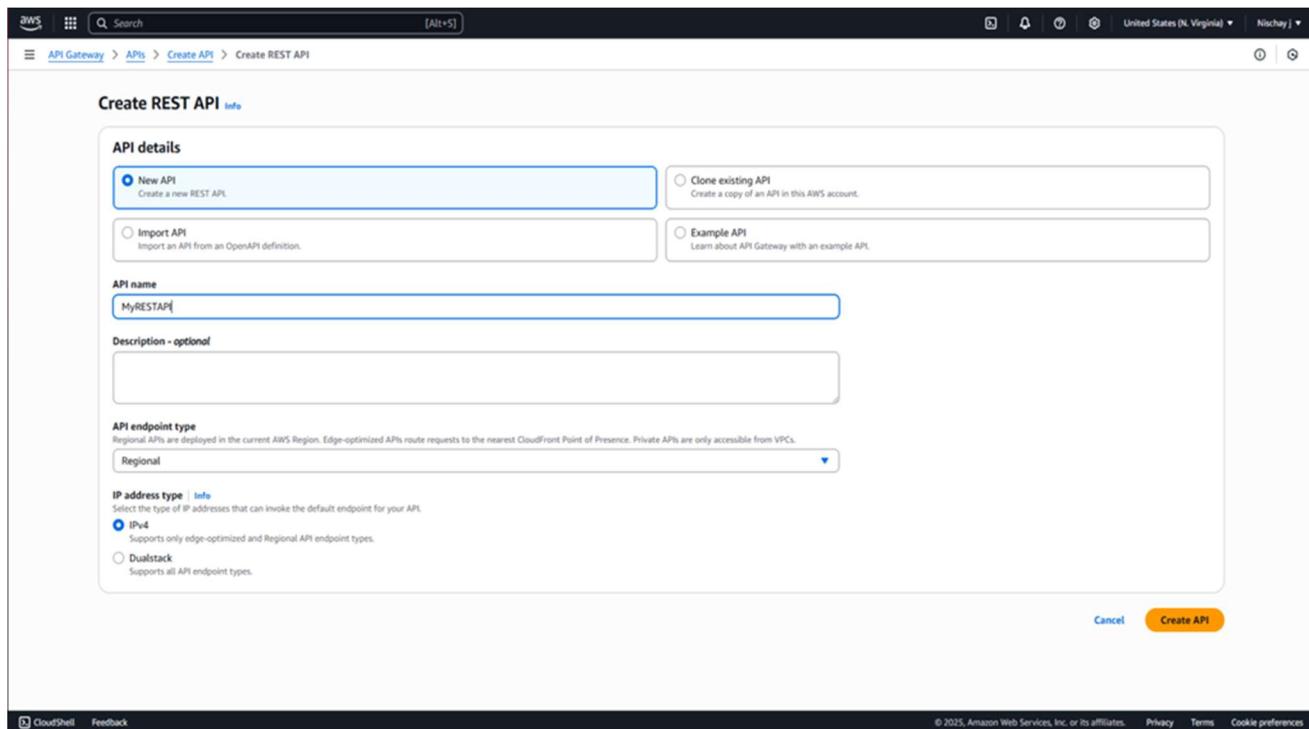
Steps for creating a REST API:

Step 1: Open AWS and Sign In to the Console.

Step 2: Open Amazon API Gateway dashboard.

Step 3: Choose an API type window appears scroll down to find REST API and click Build.

Step 4: Select New API, give a name and scroll down, click on Create API.



Step 5: Successful creation notification is displayed, click on Create method.

Step 6: Open link in new tab and create Lambda Function.

Step 7: Enter the function name and 'Python 3.13' as runtime.

Step 8: Enable function URL and keep Auth type as NONE.

Step 9: Now, go to REST API, choose create method, choose HTTP and HTTP method, and choose lambda function, paste function ARN and create method.

Step 10: Successfully created 'GET' method.

Step 11: In Deploy API dialog box give Stage as 'New Stage' and give it name then click on Deploy

Step 12: Successfully created deployment for MyRESTAPI.

Step 13: copy the Invoke URL to new Tab and view Output.

Additional Configurations

- Enable Code signing | [Info](#)
- Use code signing configurations to ensure that the code has been signed by an approved source and has not been altered since signing.
- Enable encryption with an AWS KMS customer managed key | [Info](#)
- By default, Lambda encrypts the .zip file archive using an AWS-owned key.
- Enable function URL | [Info](#)
- Use function URLs to assign HTTPS endpoints to your Lambda function.

Auth type

Choose the auth type for your function URL. [Learn more](#)

- AWS_IAM Only authenticated IAM users and roles can make requests to your function URL.
- NONE Lambda won't perform IAM authentication on requests to your function URL. The URL endpoint will be public unless you implement your own authorization logic in your function.

Function URL permissions

When you choose auth type **NONE**, Lambda automatically creates the following resource-based policy and attaches it to your function. This policy makes your function public to anyone with the function URL. You can edit the policy later.

```
1 * []
2 "Version": "2012-10-17",
3 "Statement": [
4   {
5     "StatementId": "FunctionURLAllowPublicAccess",
6     "Effect": "Allow",
7     "Principal": "*",
8     "Action": "lambda:InvokeFunctionUrl",
9     "Resource": "arn:aws:lambda:us-east-1:209479306029:function:<Function Name>",
10    "Condition": {
11      "StringEquals": {
12        "lambda:FunctionUrlAuthType": "NONE"
13      }
14    }
15  }
]
```

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Successfully created deployment for MyRESTAPI. This deployment is active for MyStage1.

Stages

MyStage1	Stage actions	Create stage
Stage details Info <p>Stage name MyStage1</p> <p>Cache cluster Info <input type="radio"/> Inactive</p> <p>Default method-level caching <input type="radio"/> Inactive</p> <p>Invoke URL https://lb3xuso0ng.execute-api.us-east-1.amazonaws.com/MyStage1</p> <p>Active deployment 2kvhrn on May 19, 2025, 16:48 (UTC+05:30)</p>	Edit	
Logs and tracing Info <p>CloudWatch logs <input type="radio"/> Inactive</p> <p>X-Ray tracing <input type="radio"/> Inactive</p> <p>Custom access logging <input type="radio"/> Inactive</p>	Edit	

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Steps for Deletion of REST API:

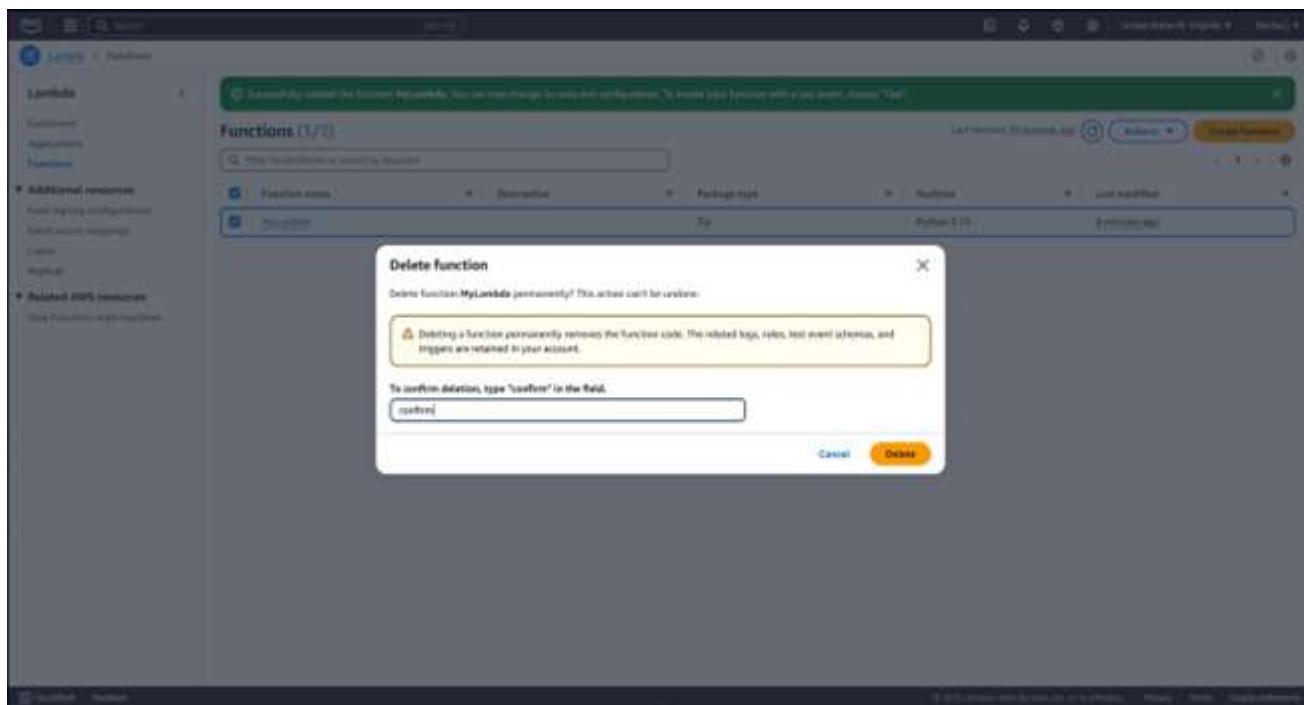
Step 1: Select the API been created and click on ‘Delete’.

Step 2: To confirm the action, enter ‘confirm’ and click on delete.

Step 3: Select the lambda function been created and click on ‘Actions’ then click on delete.

Step 4: To confirm the action, enter ‘confirm’ and click on delete

Step 5: Lambda function is deleted successfully



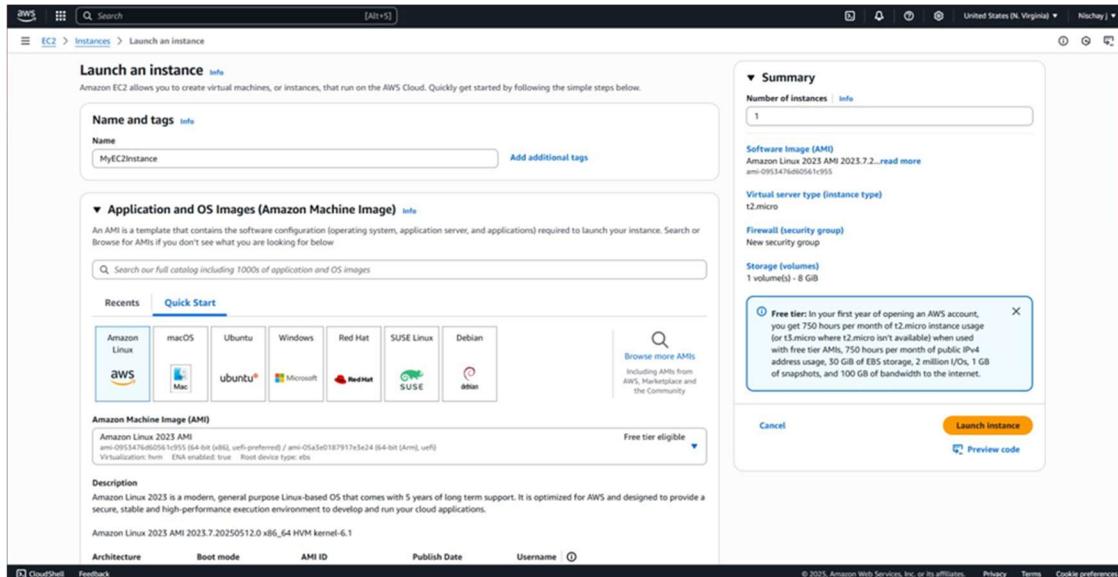
PROGRAM 4: Migrating a Web Application to Docker

Steps for creating an EC2 Instance

Step 1: Open AWS and Sign In to the Console.

Step 2: Open Amazon EC2 dashboard.

Step 3: Fill the Name and select ‘Amazon Linux’



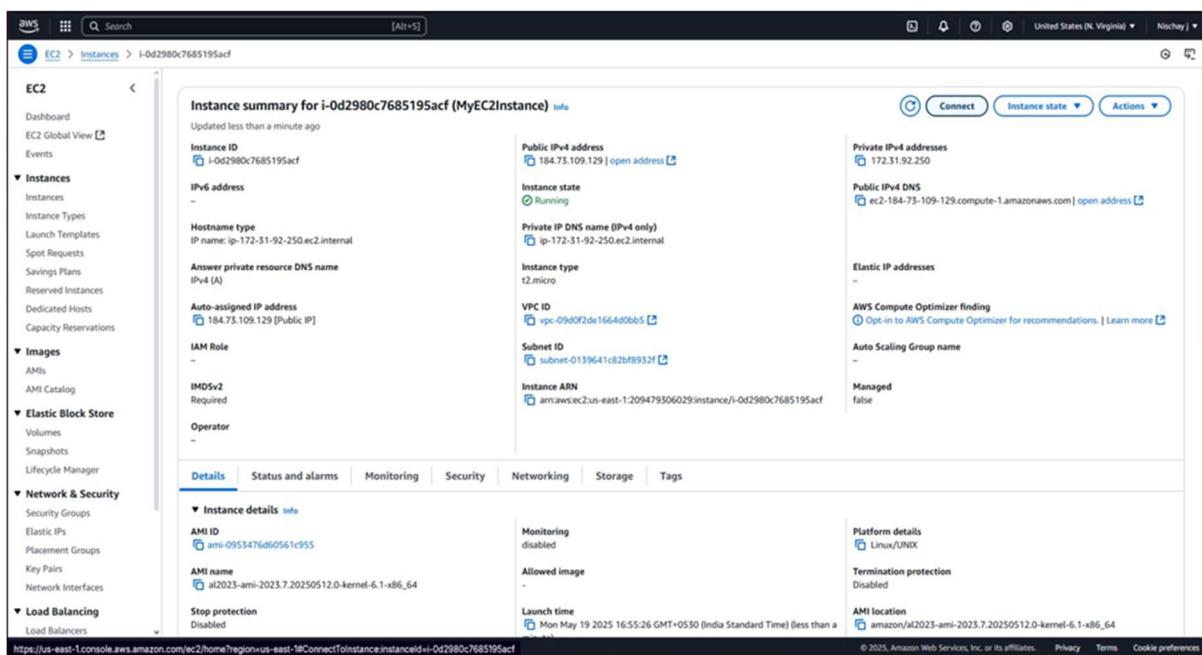
Step 4: Click on create key pair and generate new pair.

Step 5: Enable HTTPS and HTTP traffic.

Step 6: Instance has been successfully launched

Step 7: Click on instance and select ‘connect’.

Step 8: Then again click on ‘connect’.



Steps for migrating a Web Application to Dockers:

Step 1: Amazon Linux Shell is opened

Step 2: Run ‘sudo yum update -y’ command to update or get newer features if available

Step 3: Run ‘sudo yum install docker -y’ command to install Docker application on our instance.

Step 4: Run ‘sudo service docker start’ command to start docker service on our instance.

Step 5: Run ‘sudo service docker status’ command to check if Docker is running.

Step 6: Run ‘sudo su’ and ‘docker version’ command to go to root directory on our instance and to check version of Docker installed.

Step 7: Run ‘docker pull nginx’ command to download the nginx web application from Docker’s repository.

Step 8: Run ‘docker images’ command to see downloaded application images.

Step 9: Run ‘docker run -d -p 80:80 nginx’ and ‘docker ps’ command to start the nginx application and to check the status of running processes under docker.

```

OS/Arch:      linux/amd64
Context:      default

Server:
Engine:
Version:      25.0.0
API version:  1.44 (minimum version 1.24)
Go version:   go1.23.8
Git commit:   71907ca
Built:        Fri Apr 11 00:00:00 2025
OS/Arch:      linux/amd64
Experimental: false
containerd:
Version:      1.7.27
GitCommit:    05044ec09a75232cad458027ca83437aee3f4da
runc:
Version:      1.2.4
GitCommit:    6c52b3fc541fb26fe8c374d5f58112a0a5dbda66
docker-init:
Version:      0.19.0
GitCommit:    de40ad0
[root@ip-172-31-92-250 ec2-user]# docker pull nginx
Status: Downloaded newer image for nginx:latest
latest: Pulling from library/nginx
254e724d7786: Pull complete
913115292750: Pull complete
3e544d53ce99: Pull complete
f18ed624f211: Pull complete
38826f246f21: Pull complete
404a69f4e456: Pull complete
d3ddc5e71e8d: Pull complete
Digest: sha256:c15d46c91debd2f436196f3a768483ad32c258ed4elbeb3d367a27ed67253e66
Status: Downloaded newer image for nginx:latest
[root@ip-172-31-92-250 ec2-user]# docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
nginx              latest   a830707172e8  4 weeks ago   192MB
[root@ip-172-31-92-250 ec2-user]# docker run -d -p 80:80 nginx
d8d47f5d47dfef1f149308117ae3e02abb7e10ad6767492d147f98b454c
[root@ip-172-31-92-250 ec2-user]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
d8d47f5d47df      "nginx -g 'daemon off;'"
11 seconds ago     Up 10 seconds         0.0.0.80->80/tcp, ::1:80->80/tcp   jovial_ardinghell
[root@ip-172-31-92-250 ec2-user]#

```

i-0d2980c7685195acf (MyEC2Instance)
PublicIPs: 184.73.109.129 PrivateIPs: 172.31.92.250

Step 10: Copy the public IP Address (e.g. 13.233.73.122/) to a new tab to view the running nginx application.

Steps for Deletion of EC2 instance:

Step 1: Select the EC2 instance been created and click on ‘Instance state’ and the click on ‘Terminate instance’.

Step 2: EC2 instance has been successfully terminated.

PROGRAM 5: Caching Application Data with ElasticCache, Caching with Amazon CloudFront, Caching Strategies.

Steps for creating an ElasticCache:

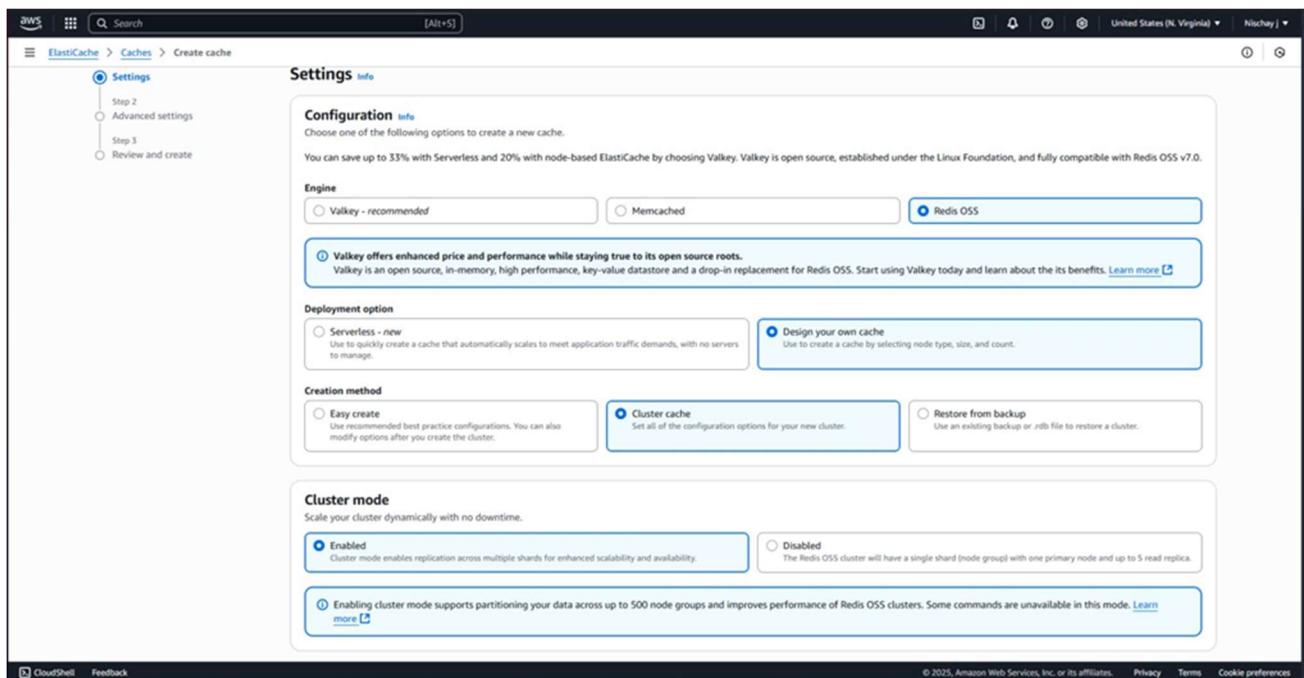
Step 1: Open AWS and Sign In to the Console.

Step 2: Open Amazon ElasticCache

Step 3: On the left side of the screen click on ‘Redis OSS caches’ and click on ‘create cache’.

Step 4: Click on ‘Continue with Redis OSS’.

Step 5: Select ‘Redis OSS’ as an Engine, select ‘Design your own cache’ for Deployment option, select ‘Cluster cache’ for Creation method and enable the cluster mode.



Step 6: Fill the Cluster name and disable ‘Multi-AZ’.

Step 7: Select ‘7.0’ as an Engine version, port as ‘6379’ and Node type as ‘cache.t3.micro’.

The ‘Number of shards’ should be 2 and ‘Replicas per shard’ should be 1.

Step 8: Click on ‘Create a new subnet group’ and fill the name of the subnet.

Step 9: Click on Manage and select ‘us-east-1a and us-east-1b’.

Step 10: Select ‘Specify Availability Zones’ option under Availability Zone placements.

Step 11: Open a new tab and open EC2.

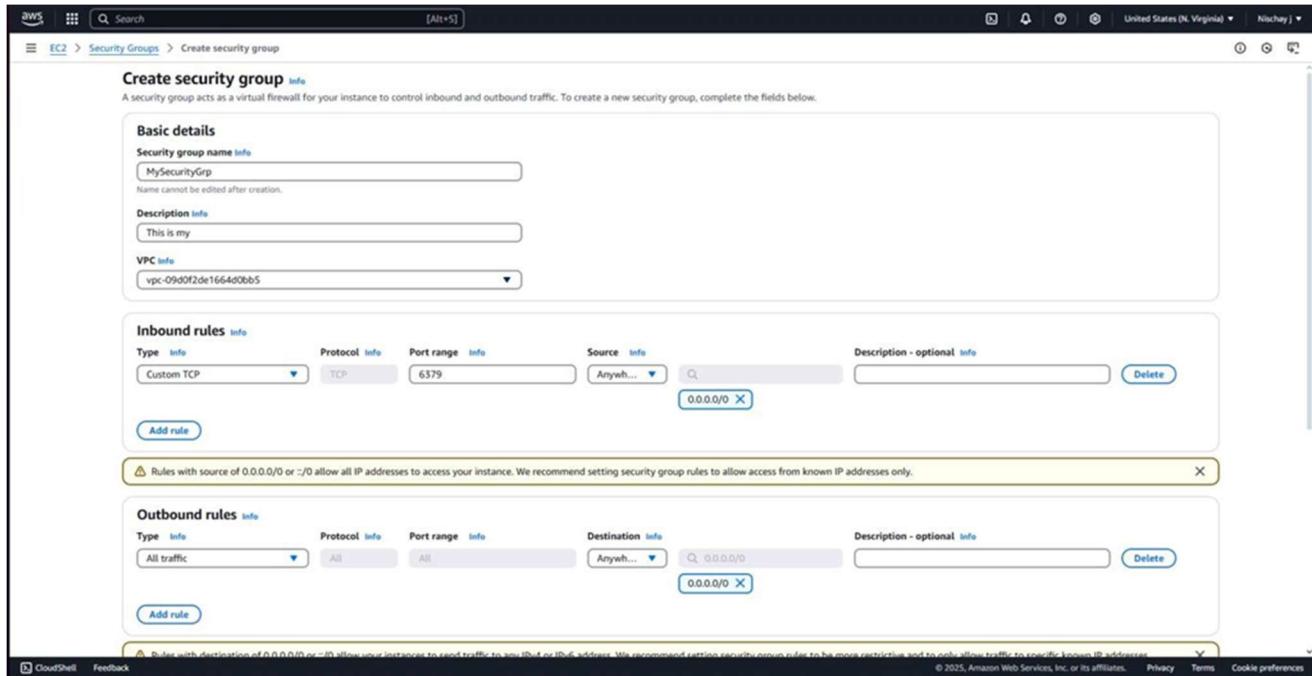
Step 12: On the left side of the screen click on ‘Security Groups’ and click on ‘create security Groups’.

Step 13: Fill the name, port range as ‘6379’ and source as ‘Anywhere IP4v’.

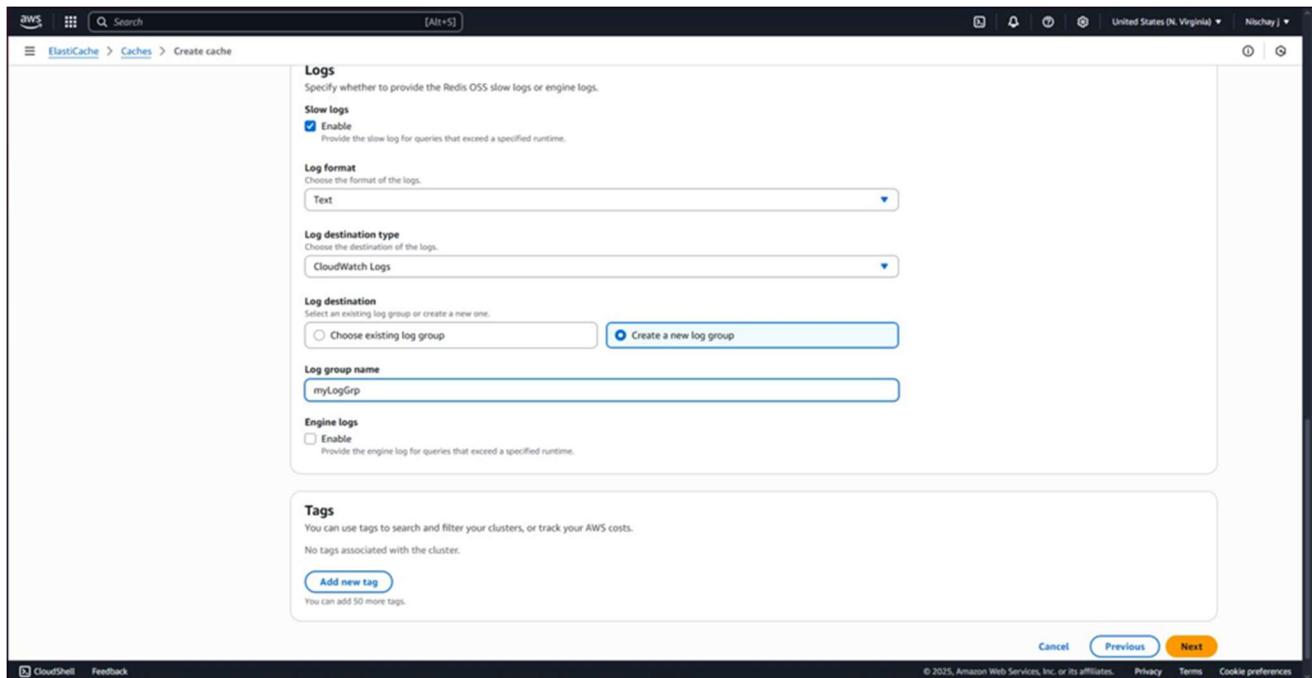
Step 14: Security group has been created successfully.

Step 15: Now come back to the ElasticCache, select ‘AUTH default user access’ in Access control and enter the ‘AUTH token’.

Step 16: Disable the ‘Backup’ and enable ‘slow logs’.



Step 17: Select ‘Text’ for the Log format, click on ‘Create a new log group’ and then enter the log group name.



Step 18: Click on ‘Create’ to create a cluster.

Step 19: Cluster has been created successfully.

Steps for Deletion of Cluster:

Step 1: Select the cluster been created and click on ‘Actions’ and the click on ‘Delete’.

Step 2: The cluster been created has been successfully deleted.