EXPERIMENT 1

DATE: 11/07/23

Aim:

Learning about IAM role and usage of MFA.

Description:

AWS Identity and Access Management (IAM) is a web service that enables Amazon Web Services (AWS) customers to manage users and user permissions in AWS. The service is targeted at organisations with multiple users or systems in the cloud that use AWS products such as Amazon EC2, Amazon SimpleDB, and the AWS Management Console. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users can access.

The main purpose of the IAM is to enable securely control access to AWS services and resources. With the help of AWS, we can manage, create policies for different users. there are many accesses authorised to users like giving access keys, passwords, and multi-factor authentication devices.

It is to create roles and add permissions to control which operation can be performed by the entity.

In this experiment the following real-time work is shown:

- pre-created users are added to the groups
- Updating Passwords for the users
- How we can edit a group policy
- Locating and using the IAM sign-in URL

Working:

List of instructions in creating a IAM in Amazon Web Services

Task 1: Explore the Users and Groups

In this task, you will explore the Users and Groups that have already been created for you in IAM.

1. In the **AWS Management Console**, on the **Services** menu, select **IAM**.

2. In the navigation pane on the left, choose **Users**.
3. Choose **user-1**.
4. Notice that user-1 does not have any permissions.
5. Choose the **Groups** tab.
6. Choose the **Security credentials** tab.
7. In the navigation pane on the left, choose **User groups**.
8. Choose the **EC2-Support** group.
9. Choose the **Permissions** tab.

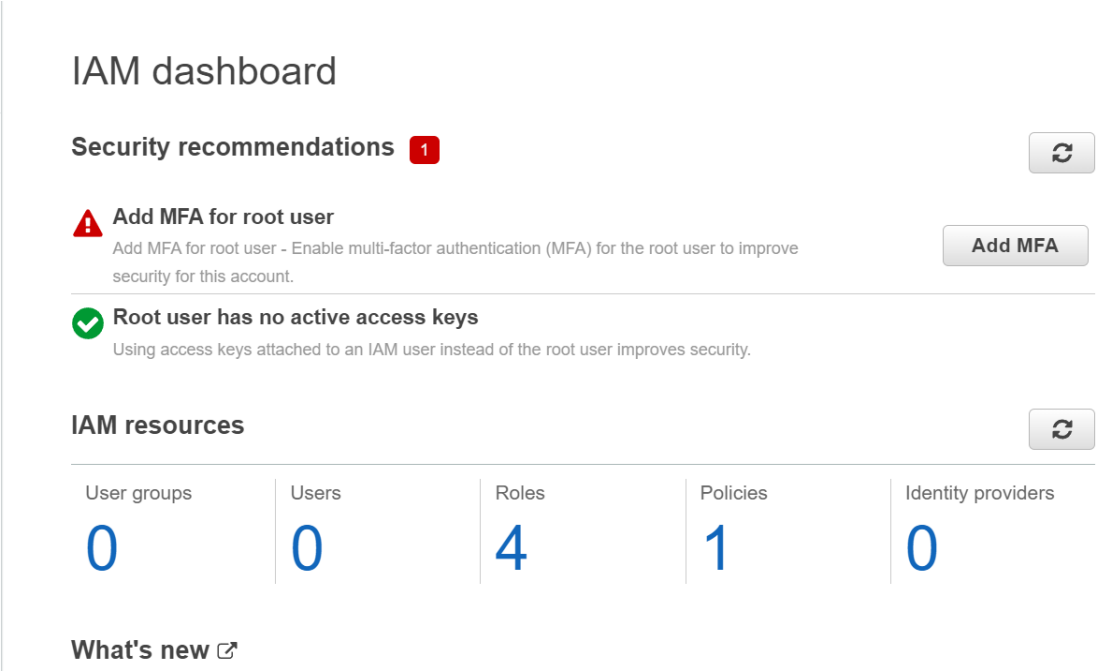


Fig.-1: Dashboard of IAM

10. Choose the plus (+) icon next to the AmazonEC2ReadOnlyAccess policy to view the policy details.
11. Choose the minus icon (-) to hide the policy details.
12. In the navigation pane on the left, choose **User groups**.
13. Choose the **S3-Support** group and then choose the **Permissions** tab.

The S3-Support group has the **AmazonS3ReadOnlyAccess** policy attached

14. Choose the plus (+) icon to view the policy details.

This policy grants permissions to Get and List resources in Amazon S3.

15. Choose the minus icon (-) to hide the policy details.

16. In the navigation pane on the left, choose **User groups**.

17. Choose the **EC2-Admin** group and then choose the **Permissions** tab.

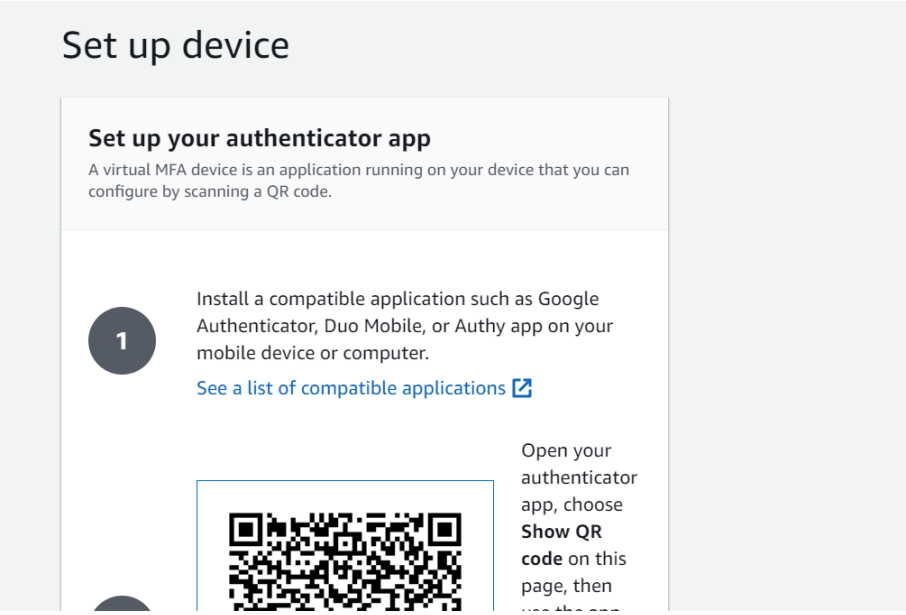


Fig.-2: Multi Factor Authentication of IAM

18. Choose the plus (+) icon to view the policy details.

This policy grants permission to view (Describe) information about Amazon EC2 and also the ability to Start and Stop instances

19. Choose the minus icon (-) to hide the policy details.

20. In the navigation pane on the left, choose **User groups**.

21. Choose the **EC2-Admin** group and then choose the **Permissions** tab.

This Group is slightly different from the other two. Instead of a *Managed Policy*, it has an **Inline Policy**, which is a policy assigned to just one User or Group. Inline Policies are typically used to apply permissions for one-off situations.

22. Choose the plus (+) icon to view the policy details.

This policy grants permission to view (Describe) information about Amazon EC2 and the ability to Start and Stop instances.

23. Choose the minus icon (-) to hide the policy details.

Task 2: Add Users to Groups

You have recently hired **user-1** into a role where they will provide support for Amazon S3. You will add them to the **S3-Support** group so that they inherit the necessary permissions via the attached *AmazonS3ReadOnlyAccess* policy.

You can ignore any "not authorised" errors that appear during this task. They are caused by your lab account having limited permissions and will not impact your ability to complete the lab.

Add user-1 to the S3-Support Group

24. In the left navigation pane, choose **User groups**.

25. Choose the **S3-Support** group.

26. Choose the **Users** tab.

27. In the **Users** tab, choose **Add users**.

28. In the **Add Users to S3-Support** window, configure the following:

In the **Users** tab you will see that user-1 has been added to the group.

29. In the navigation pane on the left, choose **User groups**.

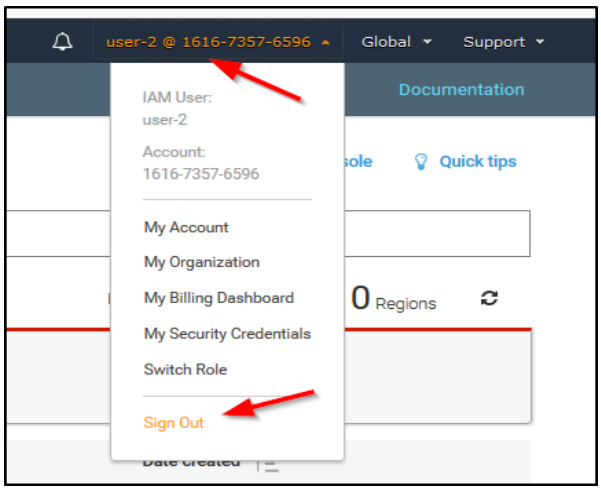


Fig.-3: IAM Users dropdown options

32. Paste the **IAM users sign-in** link into your private window and press **Enter**.

33. Paste the sign-in link into the address bar of your private web browser tab again. If it is not in your clipboard, retrieve it from the text editor where you stored it earlier.

34. Sign-in with:
    - **IAM user name:** user-3
    - **Password:** Lab-Password3

35. In the **Services** menu, choose **EC2**.

36. In the navigation pane on the left, choose **Instances**.

As an EC2 Administrator, you should now have permissions to Stop the Amazon EC2 instance.

Select the instance named *LabHost* .

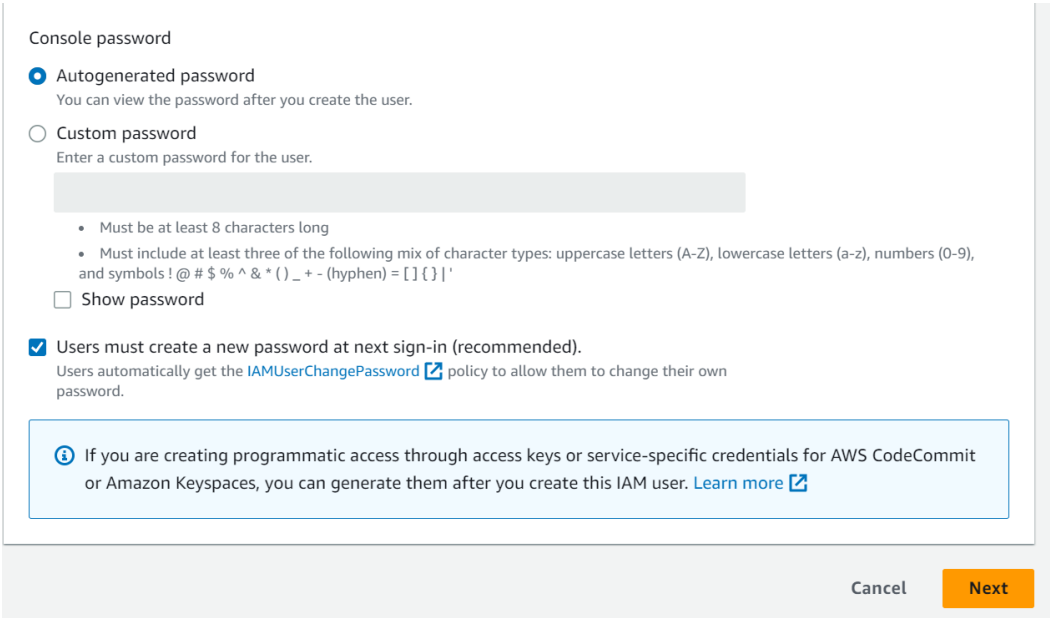


Fig.-4: Creating a user in Identity and access management.

Result:

The experiment was successfully executed on AWS Console in a real-time environment.

EXPERIMENT 2

DATE: 25/07/23

Aim:

Launching and connecting Amazon Elastic Compute Cloud (EC2) of Windows.

Description:

Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 reduces hardware costs so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. You can add capacity (scale up) to handle compute-heavy tasks, such as monthly or yearly processes, or spikes in website traffic. When usage decreases, you can reduce capacity (scale down) again.

An instance is a virtual server in the AWS Cloud. With Amazon EC2, you can set up and configure the operating system and applications that run on your instance.

An Amazon EC2 Windows instance is similar to the traditional Windows Server. After you launch an instance, it briefly goes into the pending state while registration takes place, then it goes into the running state. The instance remains active until you stop or terminate it. You can't restart an instance after you terminate it. You can create a backup image of your instance while it's running, and launch a new instance from that backup image.

Working: Refer to the following steps to launch an EC2 Windows Instance and connect it.

Step 1 - Log into your AWS account via https://aws.amazon.com/console/.

Step 2 - Open EC2 Dashboard

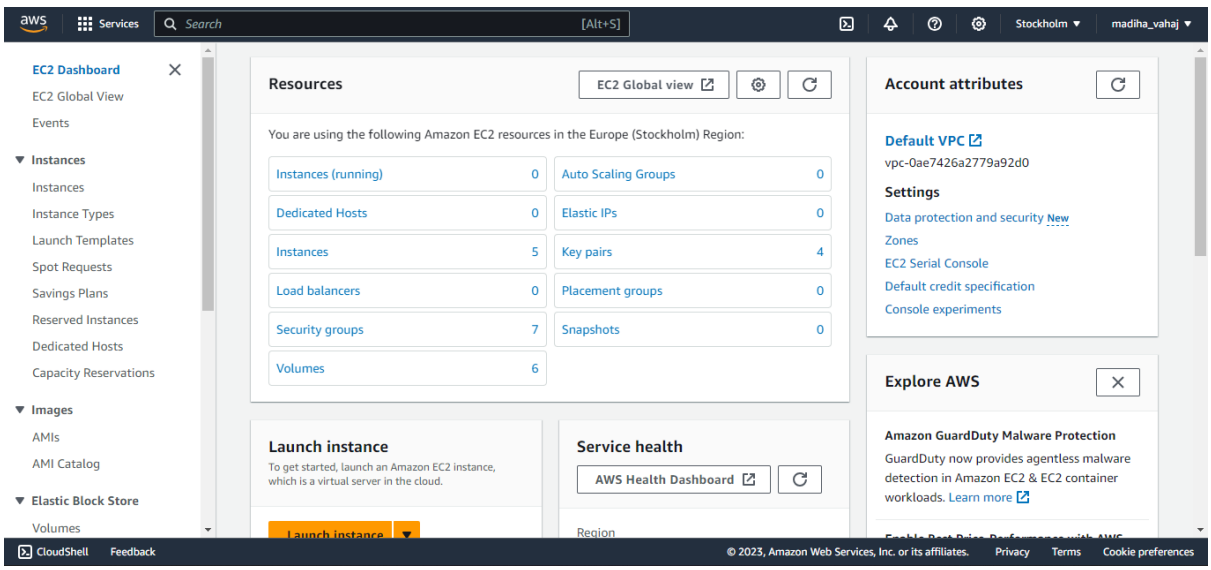


Figure 2.1 EC2 Dashboard

Step 3 – Click on Launch Instance.

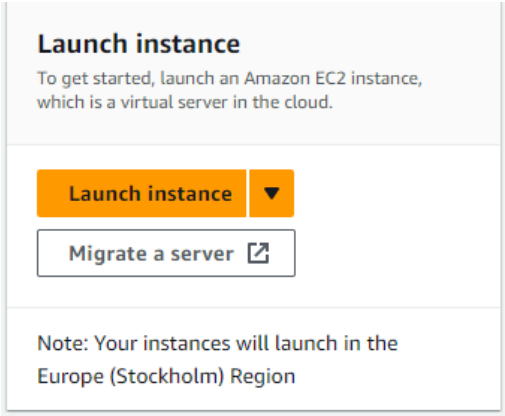


Figure 2.2 Launch Instance Button

Step 4 – Under Name and Tags give your instance a name (Example: EC2_Instance).

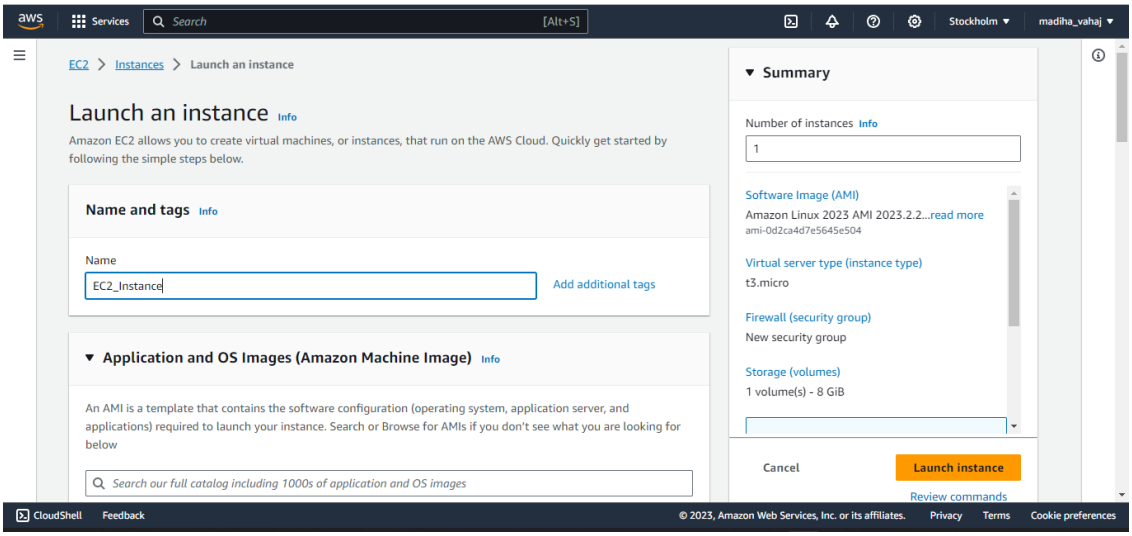


Figure 2.3 Enter name for the Instance

Step 5 - Under Application and OS Images (Amazon Machine Image) choose Quick Start, and then choose Windows. This is the operating system (OS) for your instance.

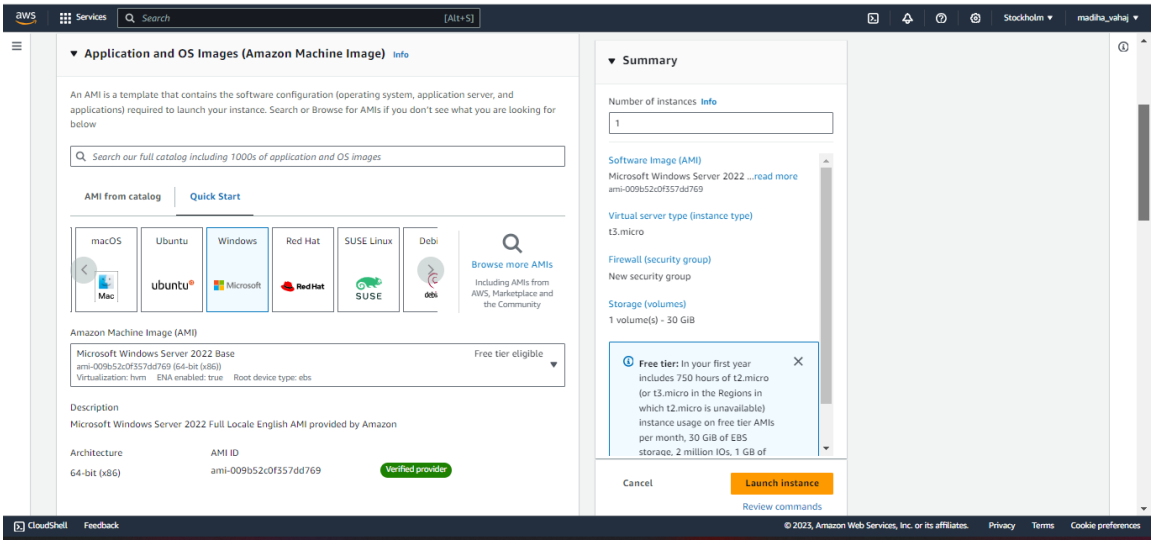


Figure 2.4 Amazon Machine Image

Step 6 – Under Instance Type choose Free Tier Eligible t3.micro instance type.

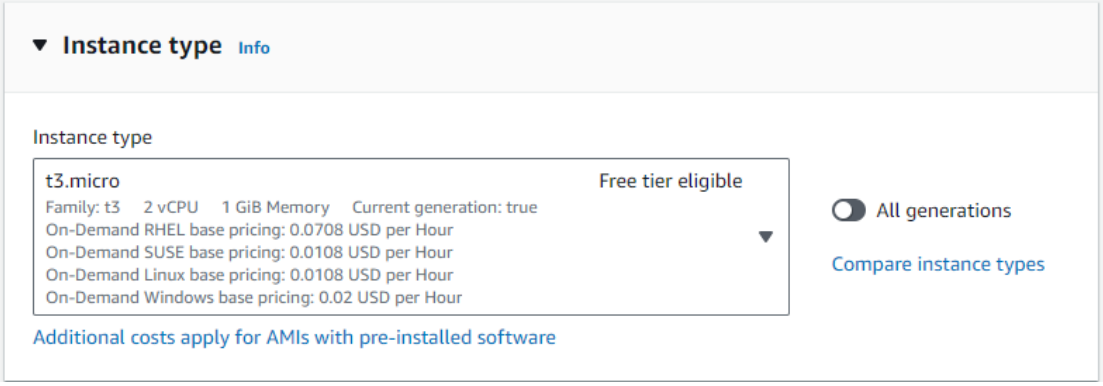


Figure 2.5 Instance Type

Step 7 – Under Key Pair (login), create a new key pair.

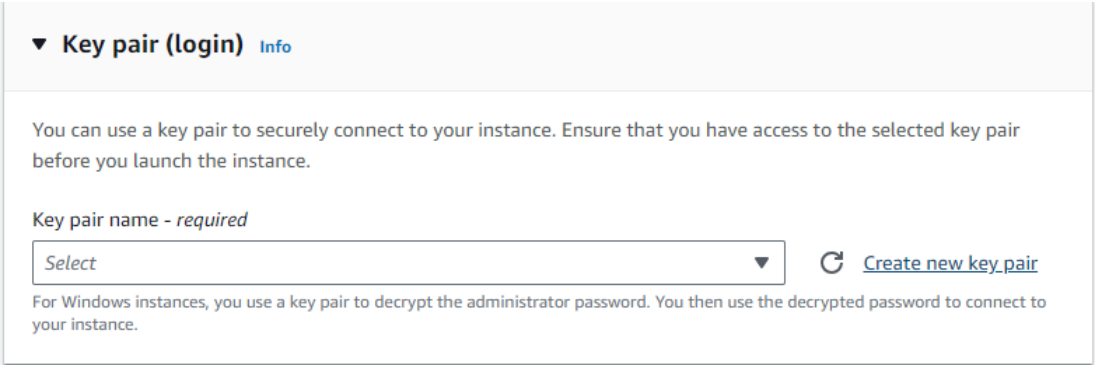


Figure 2.6 New Key Pair Login

After clicking on create a new key pair, give your key a name (Example: EC2_Instance_Key)
Choose RSA as key pair type and pem as private key file format.
Select Create Key Pair

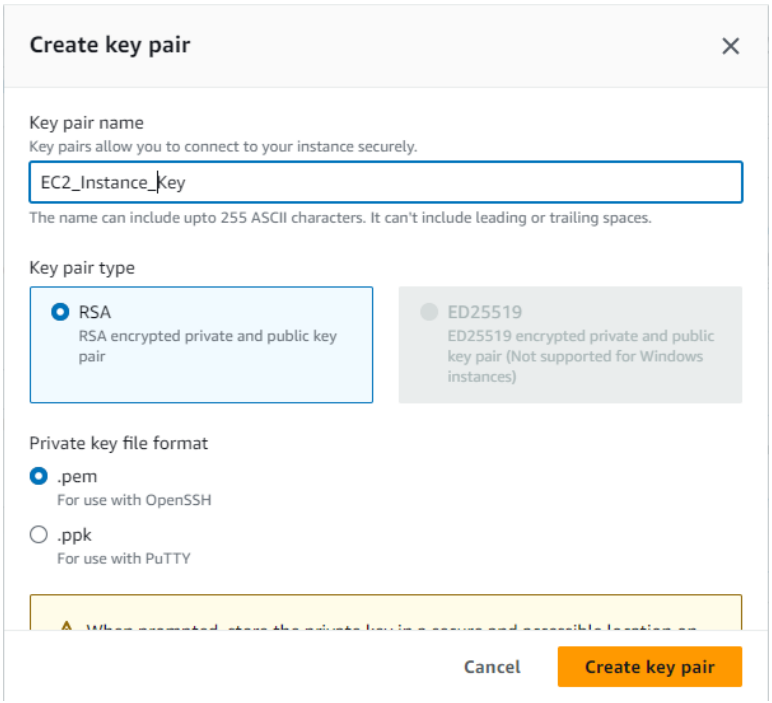


Figure 2.7 Create Key Pair

Step 8 – Under Network Settings, choose create security group and select all three options and in 'Allow RDP traffic from' select Anywhere.

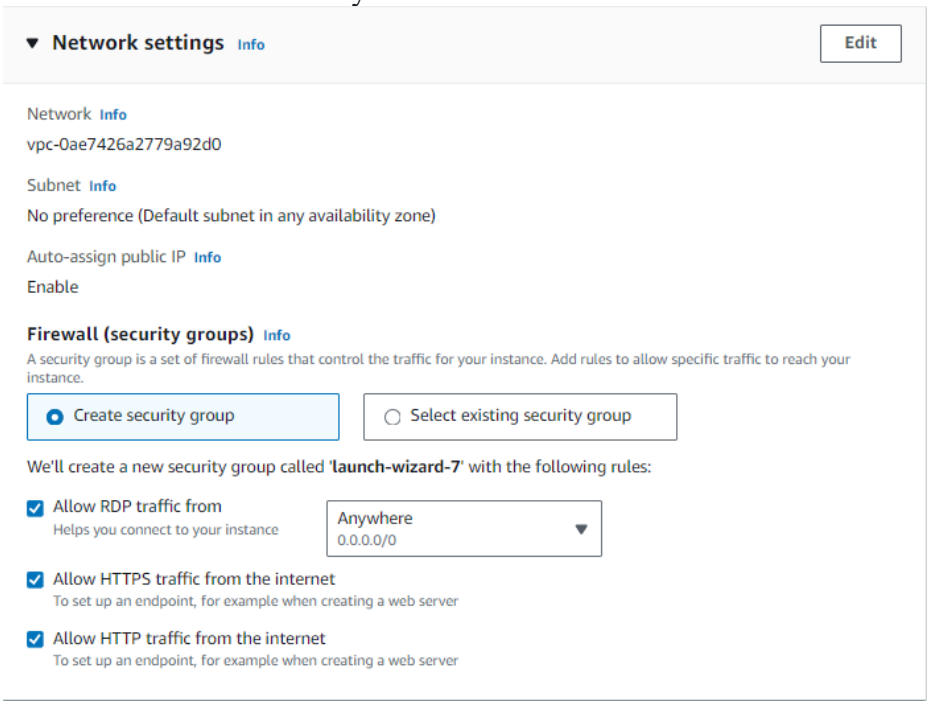


Figure 2.8 Network Settings

Step 9 – Under Configure Storage, choose storage accordingly and Launch the Instance.

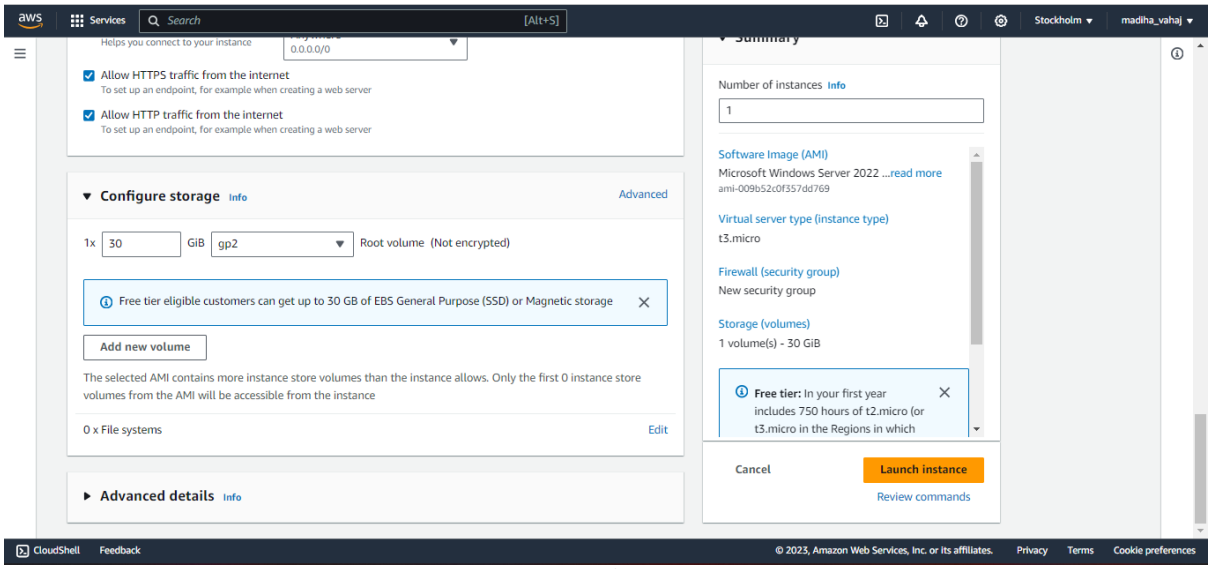


Figure 2.9 Configure Storage

Step 10 – The Instance has been created and is in running state.

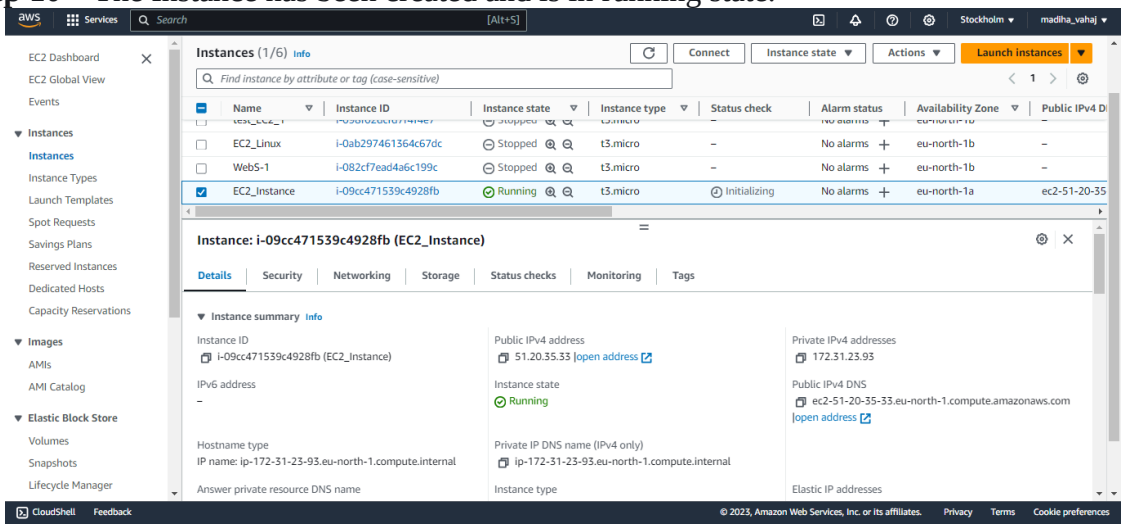


Figure 2.10 Running Instance

Step 11 – Now Connect the Instance with the Windows. Select Connect from the instances dashboard. When you click on connect, a new window will open. There select RDP client and under it choose ‘Connect using RDP client’.
It shows Public DNS, User name, and Password.
Click on ‘Get Password’ to decrypt it.

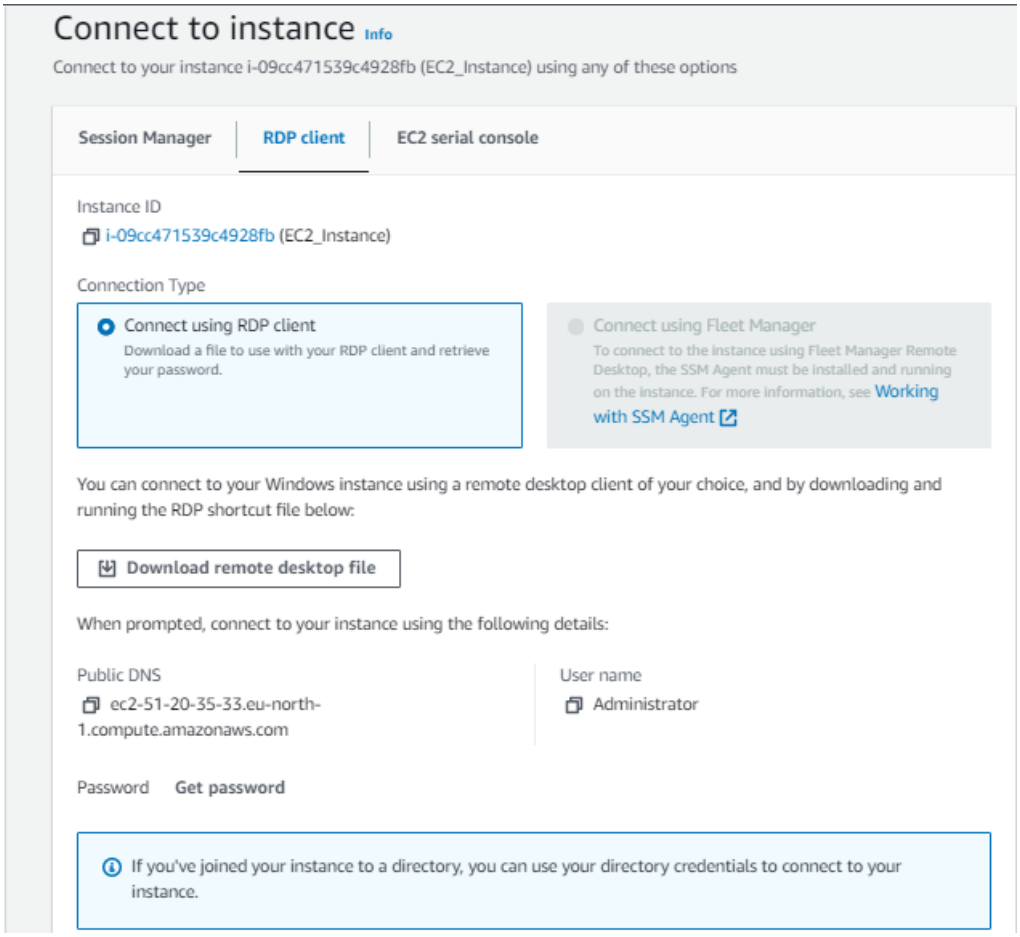


Figure 2.11 Connect to Instance

Step 12 – Upload the private key file that we created previously. The key was downloaded in your computer, select it from there. The contents of the file are displayed. Now click on 'Decrypt Password'.

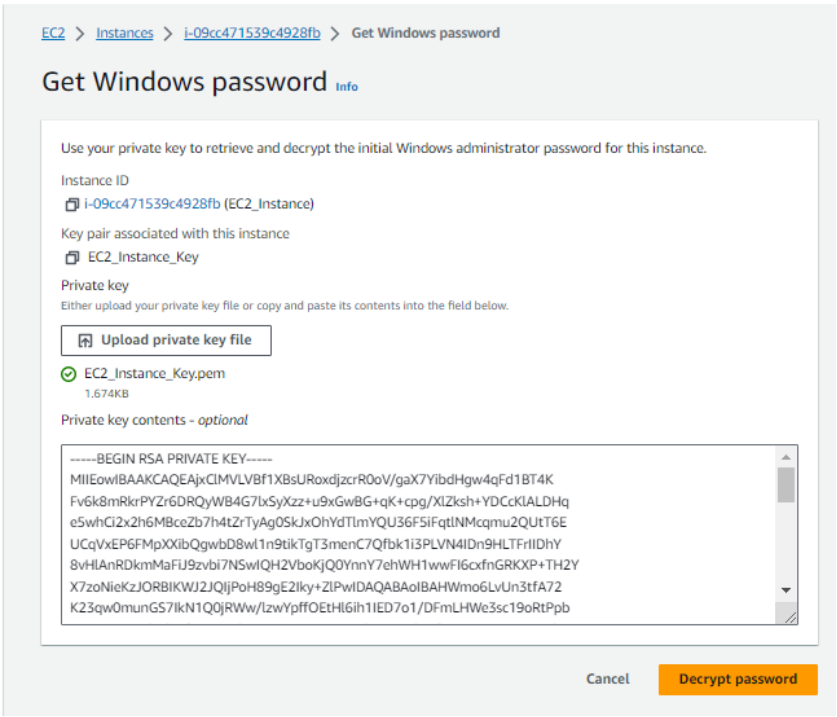


Figure 2.12 Get Windows Password

Step 13 – The password is successfully decrypted and now the password is shown instead of Get Password.

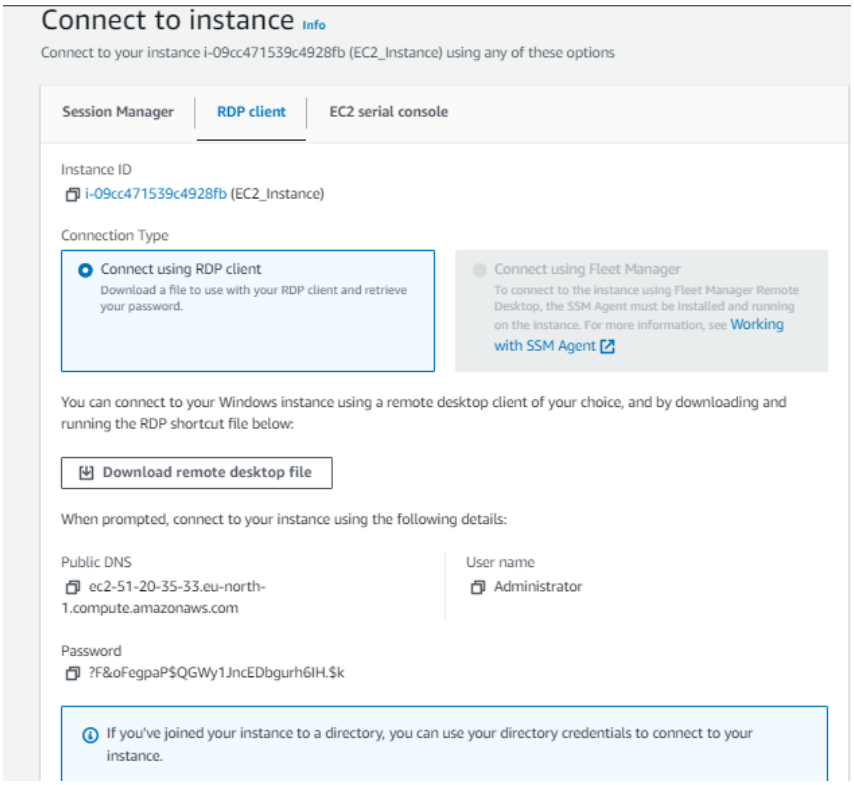


Figure 2.13 Shows Decrypted Password

Step 14 – Now open your Windows and open Remote Desktop Connection and enter the details.

Computer: In this, copy the Public DNS from ‘Connect to instance’ and paste it.

User name: In this. Copy the User name from ‘Connect to instance’ and paste it.

Now click on connect.

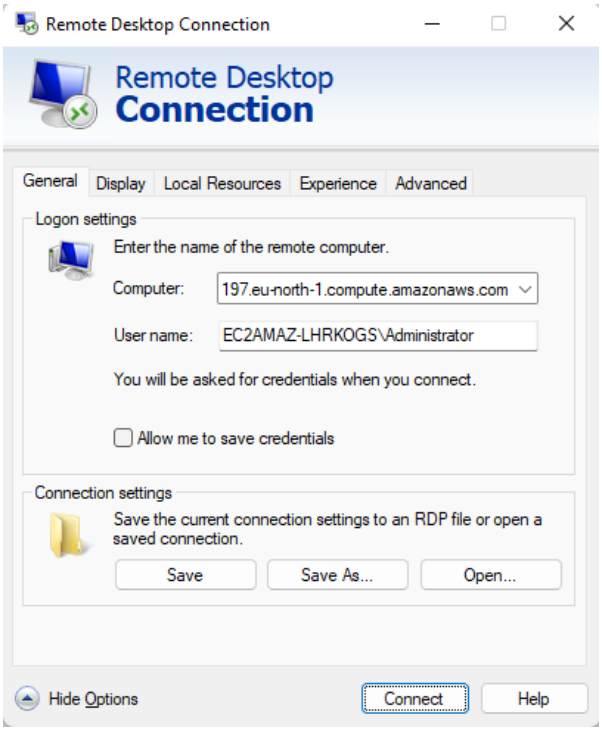


Figure 2.14 Remote Desktop Connection

Step 15 - Copy the Password from 'Connect to instance' and paste it. Click OK.

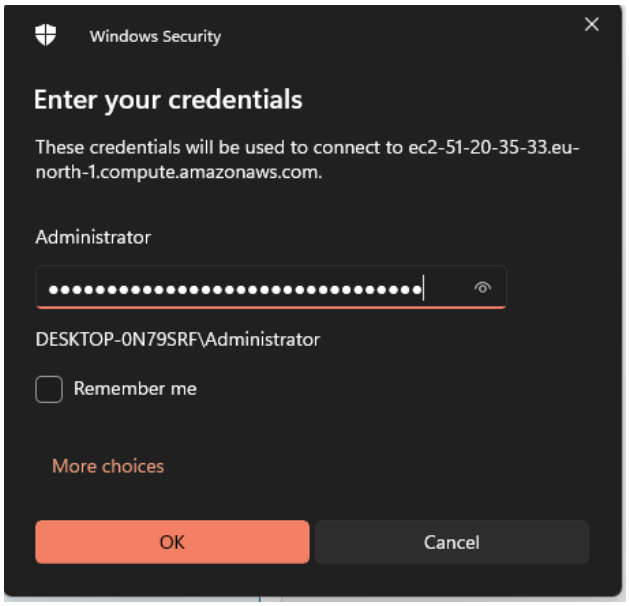


Figure 2.15 Enter your Credentials

Result: The Amazon EC2 Windows Instance has been successfully created.

EXPERIMENT 3

DATE: 1/08/23

Aim:

To explore the various features and create a Lambda Function.

Description:

AWS Lambda is a serverless computing service offered by Amazon Web Services (AWS) that allows developers to run code in response to events without the need to provision or manage servers. It is a key component of the serverless architecture, which enables developers to focus on writing code and building applications without worrying about the underlying infrastructure.

Here are some key points about AWS Lambda:

1) Event-Driven Execution: AWS Lambda is designed to execute code in response to various events, such as HTTP requests, changes in data stored in AWS services like Amazon S3, database updates, and more. It can be triggered by a wide range of AWS services and custom events.
2) Supported Languages: Lambda supports multiple programming languages, including Node.js, Python, Java, Ruby, C#, Go, and custom runtimes. This flexibility allows developers to choose the language that best fits their needs.
3) Scalability: Lambda automatically scales the execution environment based on the incoming workload. It can handle a single request or scale to thousands of requests per second, making it suitable for both small-scale and large-scale applications.
4) Pay-as-You-Go Pricing: AWS Lambda follows a pay-as-you-go pricing model, where you only pay for the compute time consumed during the execution of your code. There are no upfront fees or charges for idle time.
5) Statelessness: Lambda functions are stateless, meaning they do not retain information between executions. Any state information must be stored externally, such as in a database or a file storage service.
6) Custom Execution Roles: You can define execution roles for Lambda functions, granting them specific permissions to interact with other AWS services securely.
7) Monitoring and Logging: AWS provides tools like Amazon CloudWatch for monitoring and logging Lambda functions. You can set up alarms, track execution metrics, and troubleshoot any issues.

Architecture:

The instruction set architecture of a Lambda function determines the type of computer processor that Lambda uses to run the function. Lambda provides 2 types of instruction set architectures:

1) Arm64 – 64-bit ARM architecture, for the AWS Graviton2 processor.
2) x86_64 – 64-bit x86 architecture, for x86-based processors. X86_64 is the default architecture.

Functions that use arm64 architecture offer lower cost per Gb/s compared with the equivalent function running on an x86-based CPU.

Runtime:

A runtime is a version of a programming language or framework that you can use to write Lambda functions. Lambda supports runtime versions for Node.js, Python, Ruby, Go, Java, C# (.NET Core), and PowerShell (.NET Core)

To use other languages in Lambda, you can create your own runtime.

Advanced Options:

1) Code Signing: Use code signing configurations to enable code signing for a function. With code signing, you can ensure that the code has been signed by an approved source and has not been altered since signing, and that the code signature has not expired or been revoked.
2) Enable function URL: A function URL is a dedicated HTTP(S) endpoint for your function. When your function URL is configured, you can use it to invoke your function through a browser, curl, Postman, or any HTTP client.
3) Enable VPC: All Lambda functions run securely inside a default system-managed virtual private cloud (VPC). However, you can also configure your Lambda function to access resources in a custom VPC.
4) Enable Tags: Tags are optional key-value pairs that you can attach to your Lambda function. They can be used for various purposes such as function organization, cost allocation, access control.

Working:

Refer to the following steps to create a Lambda function.

1) Log into your AWS account via https://aws.amazon.com/console/.

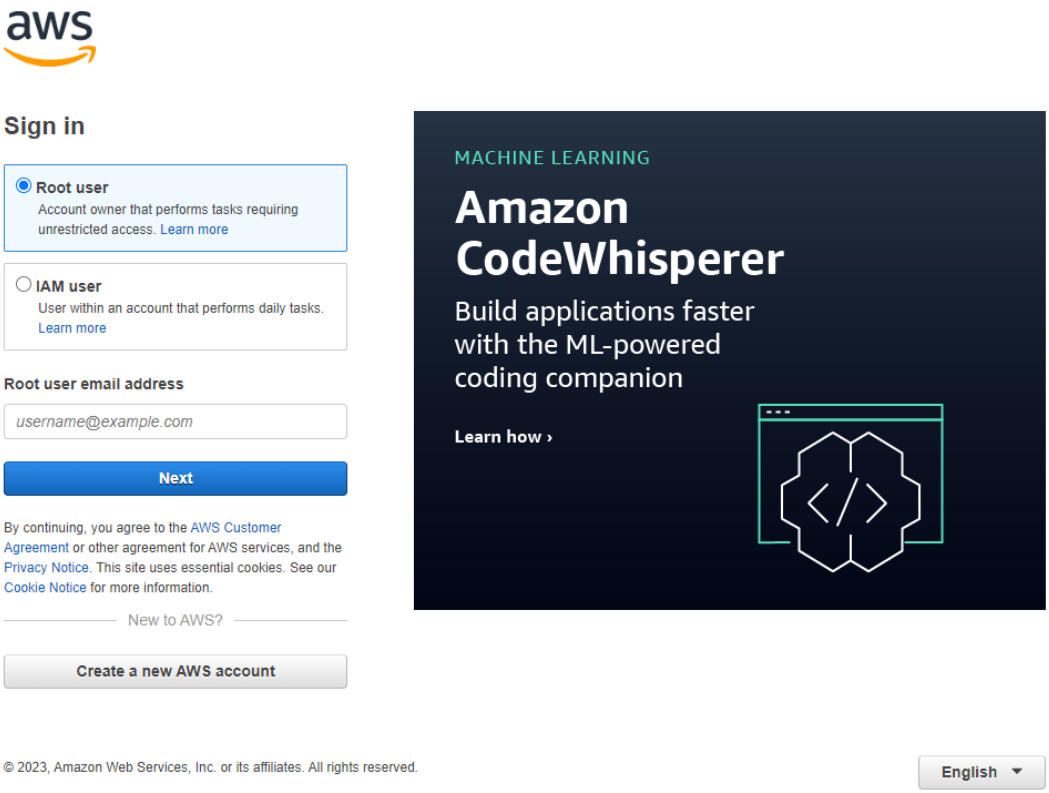


Figure 3.1: Login Window

2) In the AWS Management Console, search for "Lambda" in the services search bar or navigate to "Compute" and then select "Lambda."

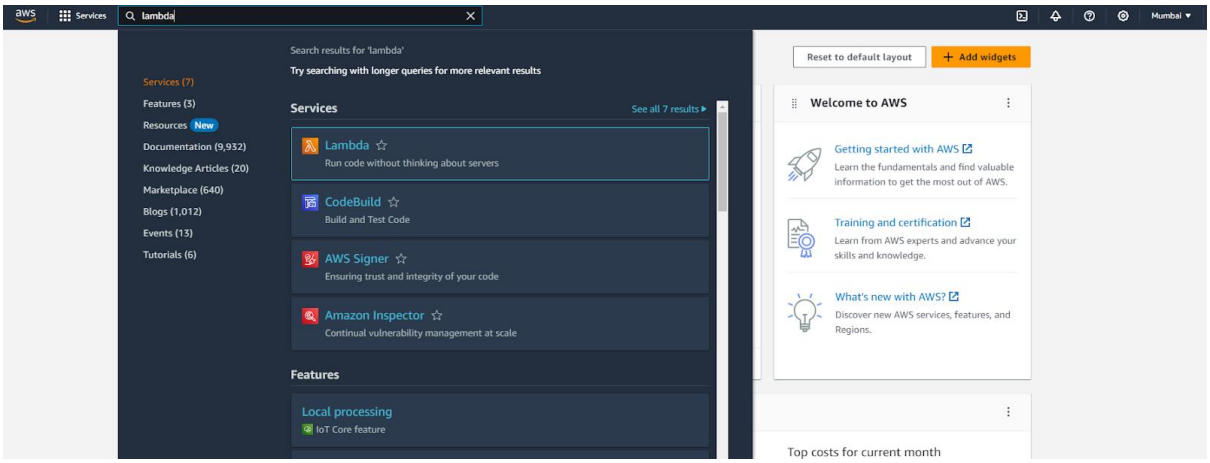


Figure 3.2: Search Window

3) Click on “Create function” button.

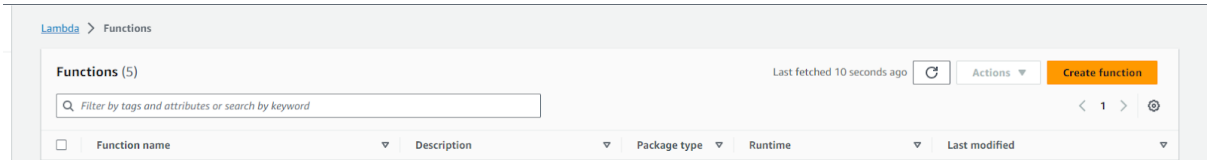


Figure 3.3: Create Function Window

4) There are two options, either a user can choose to create a function from scratch or use a blueprint/template. Select “Author from Scratch”.

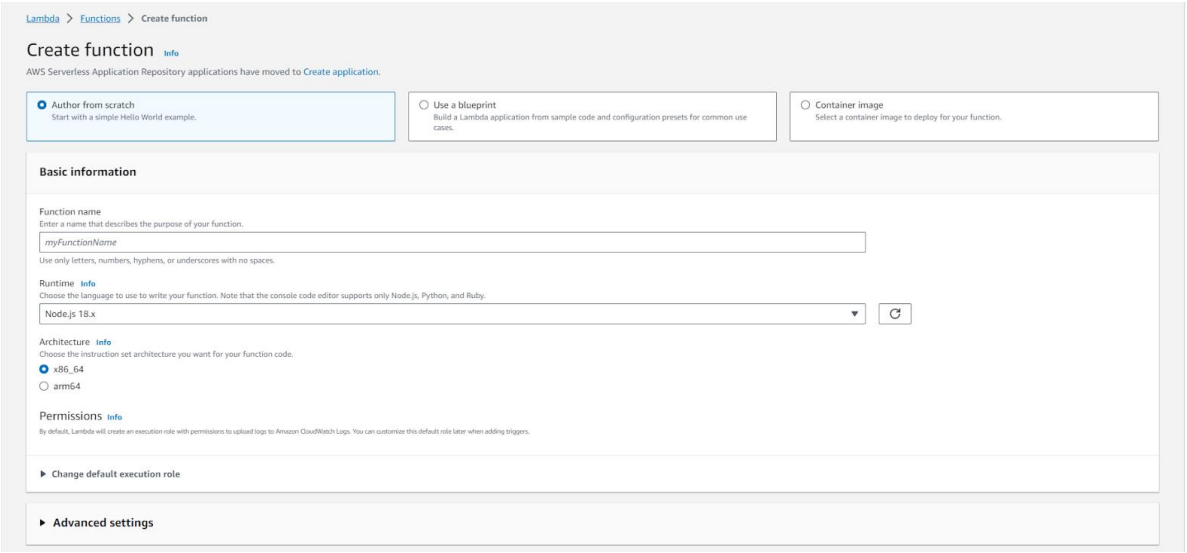


Figure 3.4: Create Function Window

5) Give the lambda function a name, there are a few options to choose the runtime environment. Select Python 3.11.
6) There are two different types of architecture set that can be used. Select “x86_64”.
7) Select the default execution role.

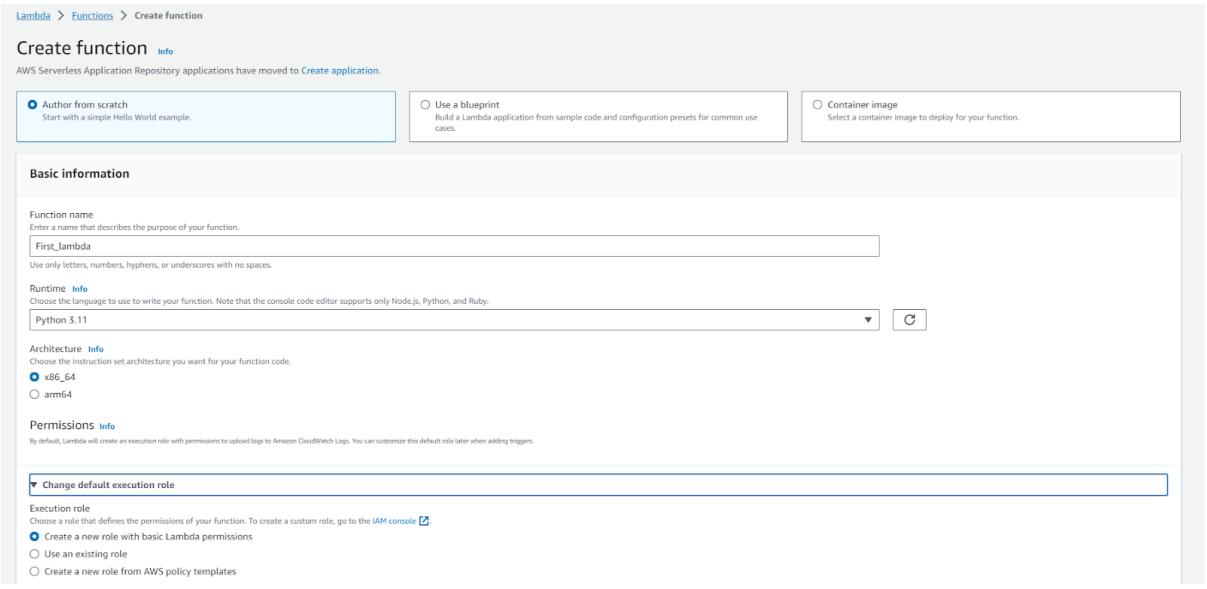


Figure 3.5: Create Function Window

8) Select “Create function”.

In the code editor, use the following code for the lambda function.

Code:

```
import json

def lambda_handler(event, context):
    numbers = event['numbers']
    a=numbers['number_1']
    b=numbers['number_2']
    # Concatenate lists 'a' and 'b' to create 'c'.
    c = a + b
    # Print the concatenated list 'c'.
    print(c)
    return {
        'statusCode': 200,
        'body': json.dumps(c) }
```

The above code configures a lambda function to add 2 numbers received as a json from an event.

Configure Test Event:

1) Select the drop down menu from Test field.

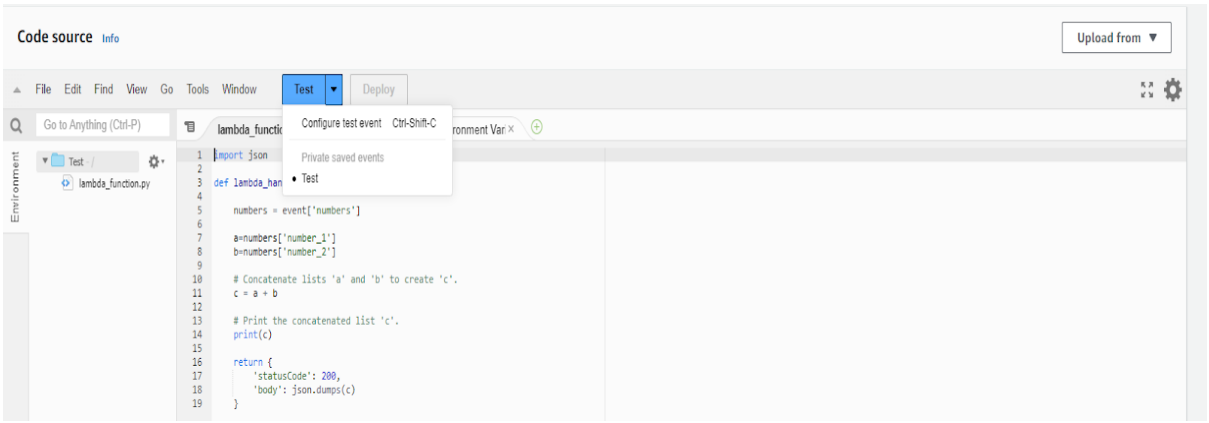


Figure 3.6: Test Function Window

2) Select “Configure test event”.
3) Give the event a name and use the following code to configure a test event.

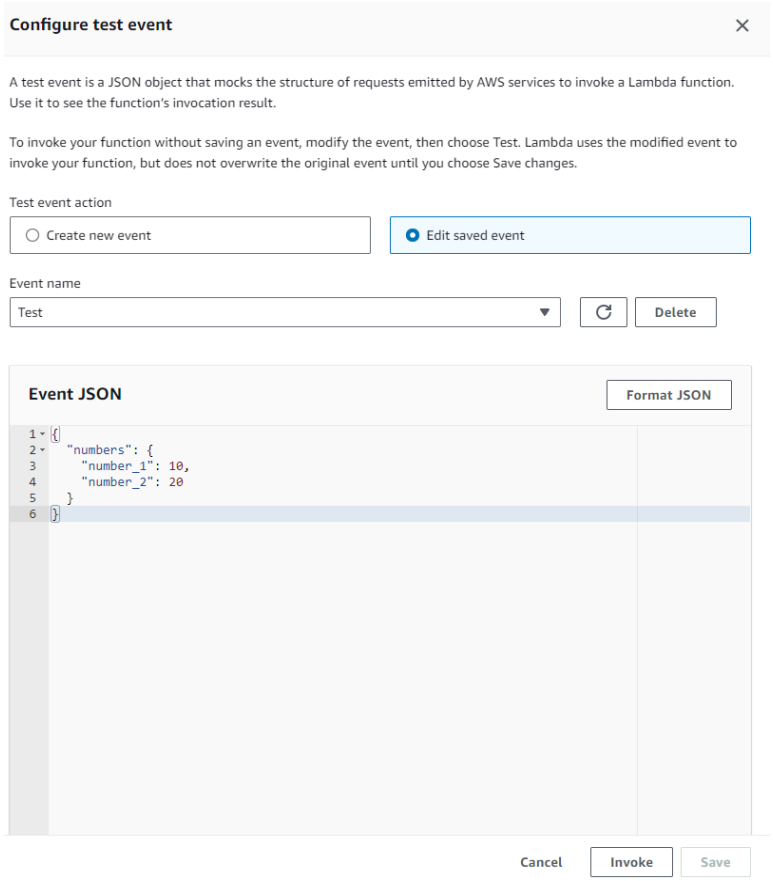


Figure 3.7: Test Function Window

Test Event:

```
{
  "numbers": {
    "number_1": 10,
    "number_2": 20
  }
}
```

4) Select “Save”

Executing Lambda Function:

1) Select the Test button.
2) In the response section and function logs the sum of two numbers configured in the test event is visible.

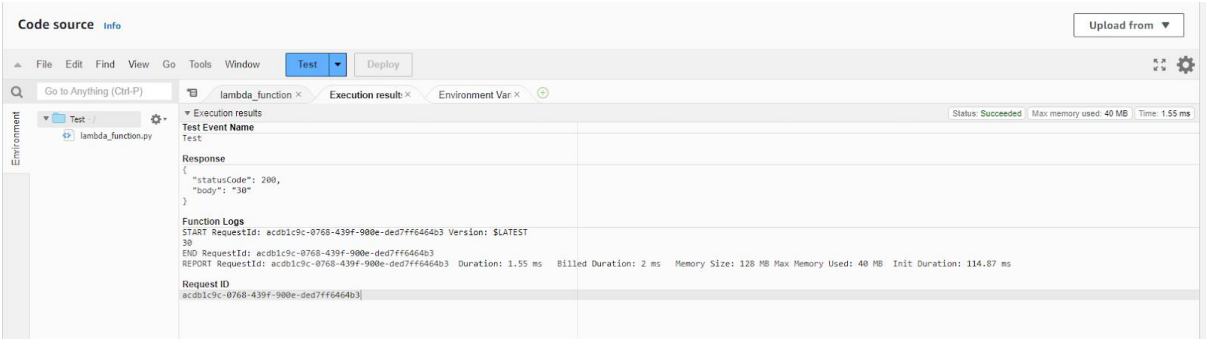


Figure 3.8: Lambda Function Window

Result:

Various features of lambda function were explored and a Lambda function to add two numbers using a Test Event was configured.

EXPERIMENT 4

DATE: 22/08/23

Aim:

Working with EBS (Elastic Block Store)

Description:

Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances. EBS volumes that are attached to an instance are exposed as storage volumes that persist independently from the life of the instance. You can create a file system on top of these volumes or use them in any way you would use a block device (such as a hard drive). You can dynamically change the configuration of a volume attached to an instance.

We recommend Amazon EBS for data that must be quickly accessible and requires long-term persistence. EBS volumes are particularly well-suited for use as the primary storage for file systems, databases, or for any applications that require fine granular updates and access to raw, unformatted, block-level storage. Amazon EBS is well suited to both database-style applications that rely on random reads and writes, and to throughput-intensive applications that perform long, continuous reads and writes.

Fig.-1: Description of EBS

Working:

Task 1: Create a New EBS Volume

In this task, you will create and attach an Amazon EBS volume to a new Amazon EC2 instance.

1. In the **AWS Management Console**, on the **Services** menu, click **EC2**.
2. In the left navigation pane, choose **Instances**.

An Amazon EC2 instance named **Lab** has already been launched for your lab.

3. Note the **Availability Zone** of the instance. It will look like *us-east-1a*.
4. In the left navigation pane, choose **Volumes**.

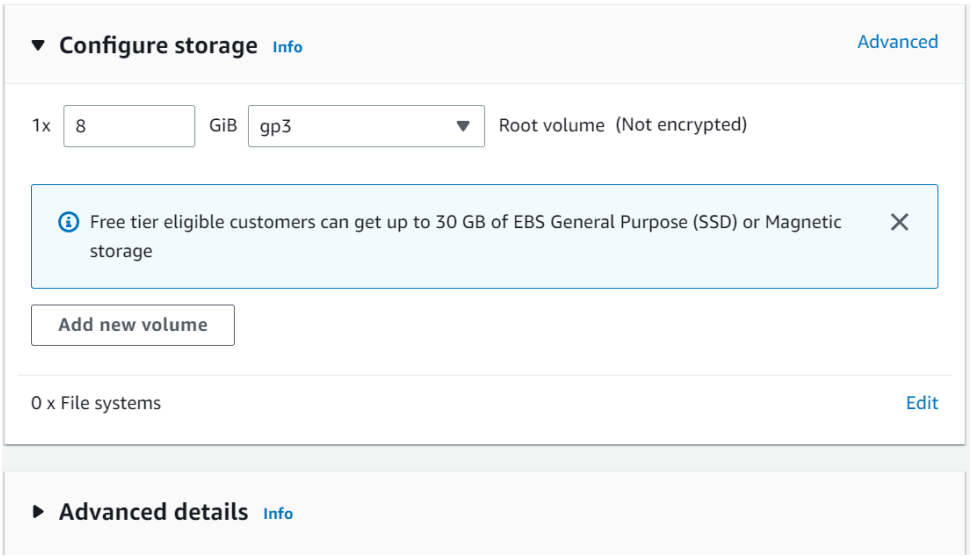


Fig.-2: Configuring storage for EBS

You will see an existing volume that is being used by the Amazon EC2 instance. This volume has a size of 8 GiB, which makes it easy to distinguish from the volume you will create next, which will be 1 GiB in size.

Choose **Create Volume**

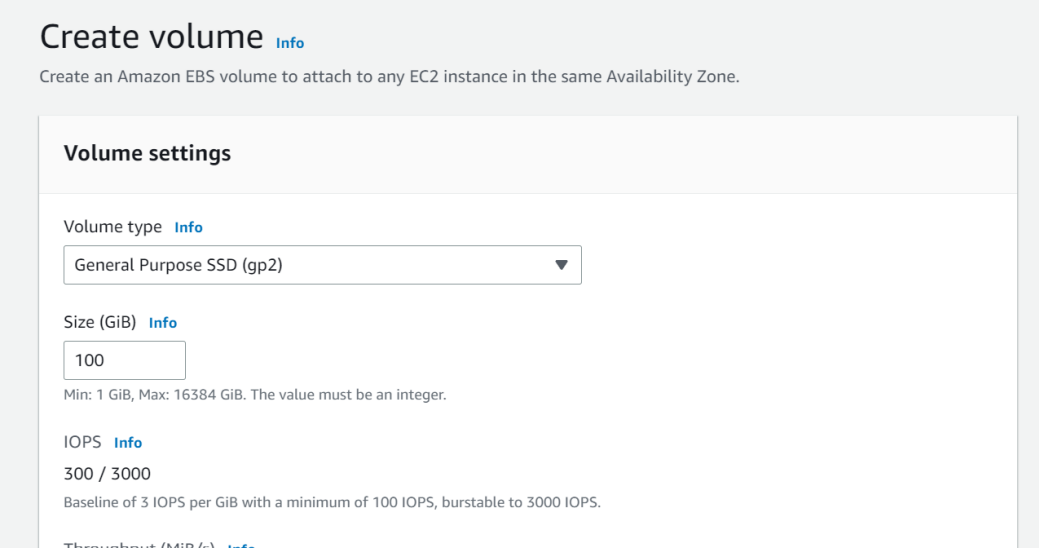


Fig.-3: Creating a volume for EBS

Your new volume will appear in the list and will move from the *Creating* state to the *Available* state. You may need to choose **to refresh** to see your new volume.

Task 2: Attach the Volume to an Instance

You can now attach your new volume to the Amazon EC2 instance.

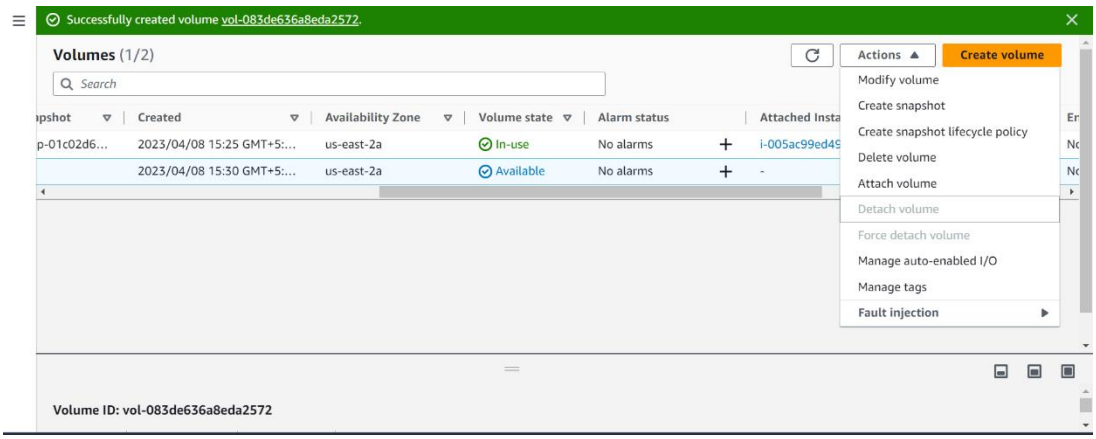


Fig.-4: Volume created

11. Select **My Volume**.
12. In the **Actions** menu, choose **Attach volume**.
13. Choose the **Instance** field, then select the instance that appears (Lab).

Note that the **Device** field is set to */dev/sdf*. You will use this device identifier in a later task.

14. Choose **Attach volume** The volume state is now *In-use*.

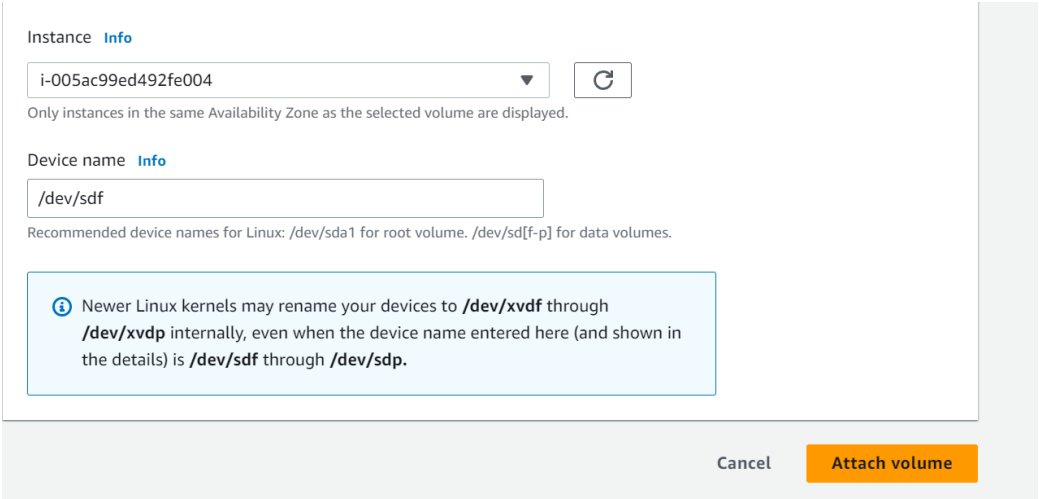


Fig.-5: Attaching the new volume to the instance

Result:

A volume of 10GB has been attached along with the given 8GB volume for a Linux instance as we can see in the above image also.

EXPERIMENT 5

DATE: 29/08/23

AIM: Exploring the S3 bucket.

DESCRIPTION:

Amazon Simple Storage Service (Amazon S3) is a logical storage unit in AWS. S3 bucket is a container for objects like files and metadata. It is created in specific AWS regions. To create an S3 bucket, we have to follow these steps:

Step 1: Sign into the AWS Management Console and open the Amazon S3 console.

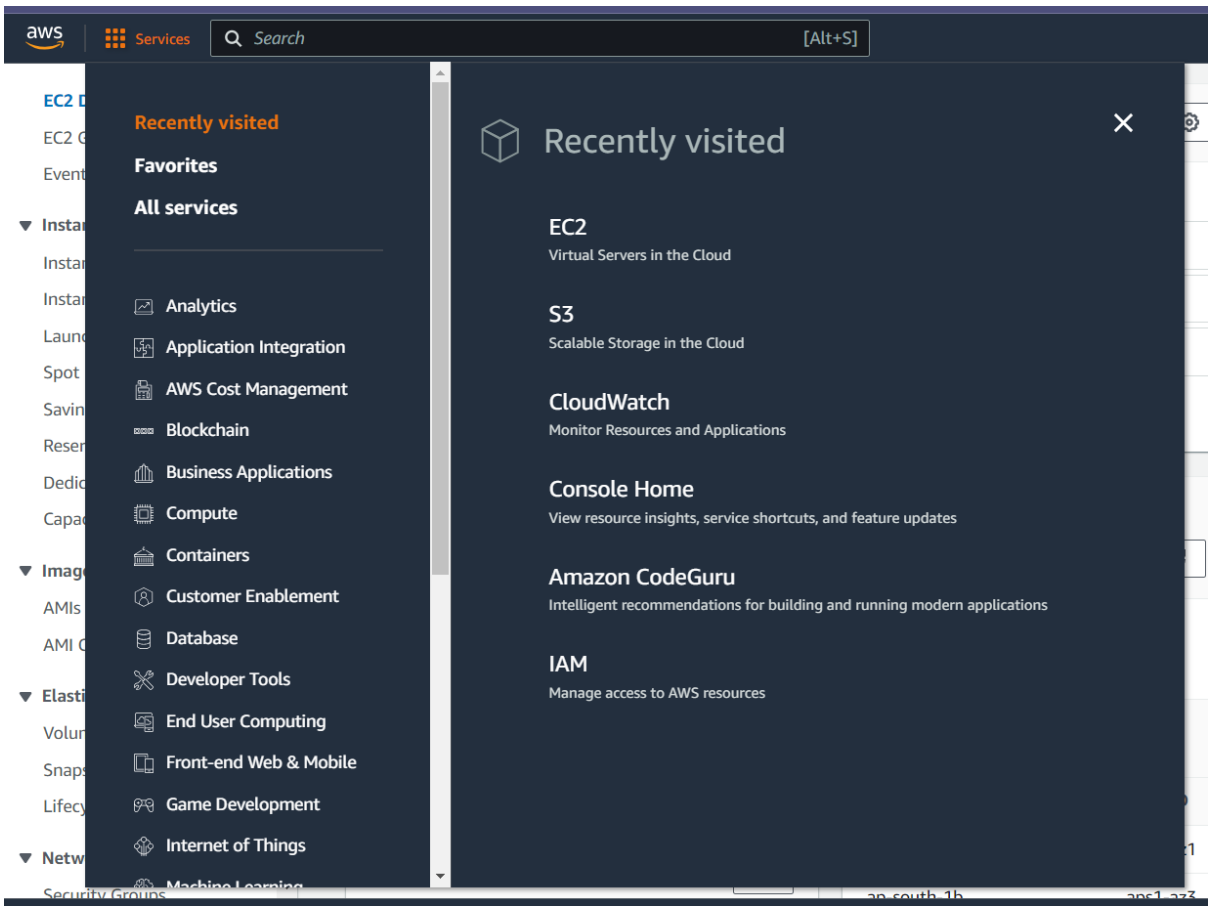


Figure 1: AWS Management Console

Step 2: In the Buckets page, click the “Create Bucket” button. For Bucket name, enter a name for your bucket. The bucket name must be globally unique across all existing bucket names in Amazon S3.

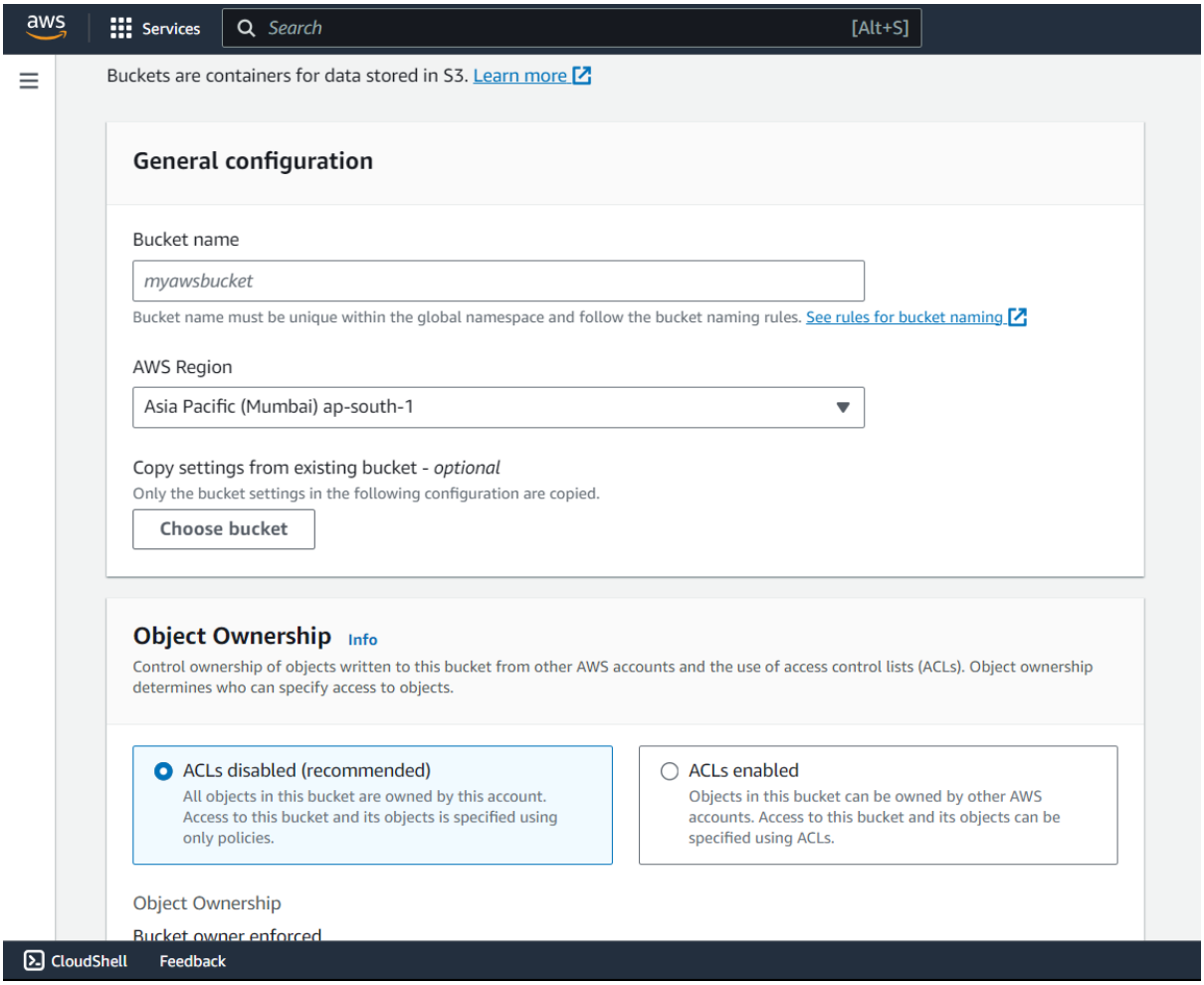


Figure 2: Bucket Configuration Page

Step 3: For Region, choose the AWS Region where you want the bucket to reside.

Step 4: Optionally, you can configure other storage management options for the bucket, such as:

a. Object Ownership: This setting controls who owns the objects in your bucket.

b. Block Public Access settings: These settings control whether the public can access the objects in your bucket.

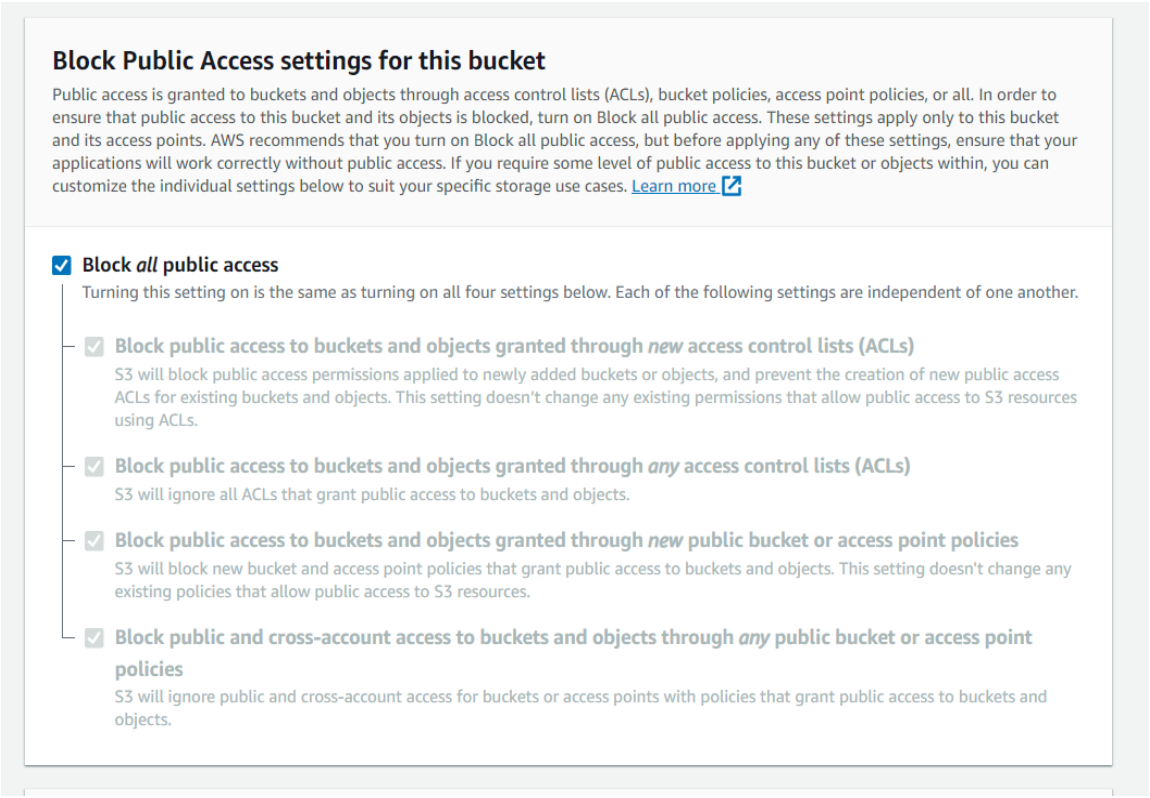


Figure 3: Public access control page

Step 5: Choose Create bucket.

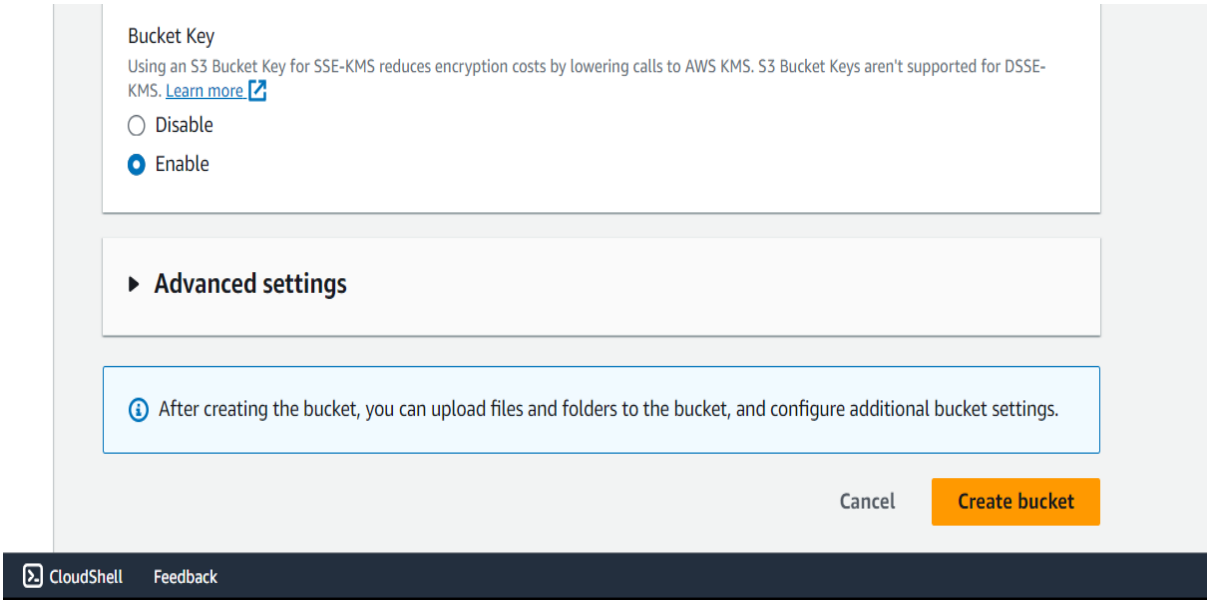


Figure 4: Bucket Creation Conformation

Step 6: Click the upload button.

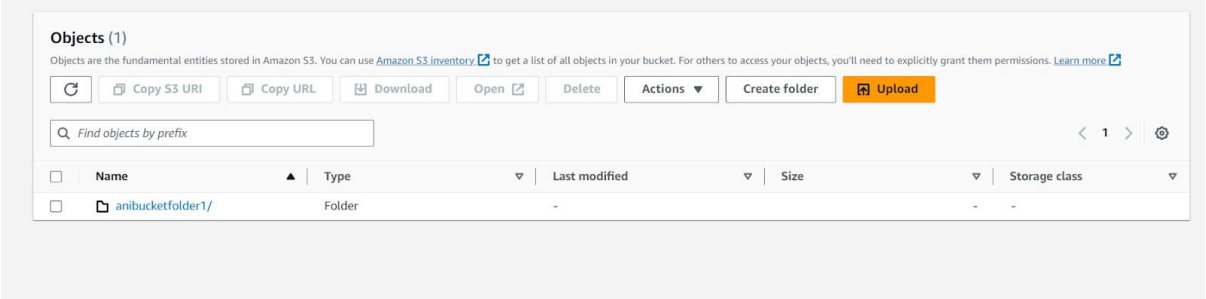


Figure 5: Uploading file to the S3 Bucket

Step 7: Choose the file you want to upload to S3.

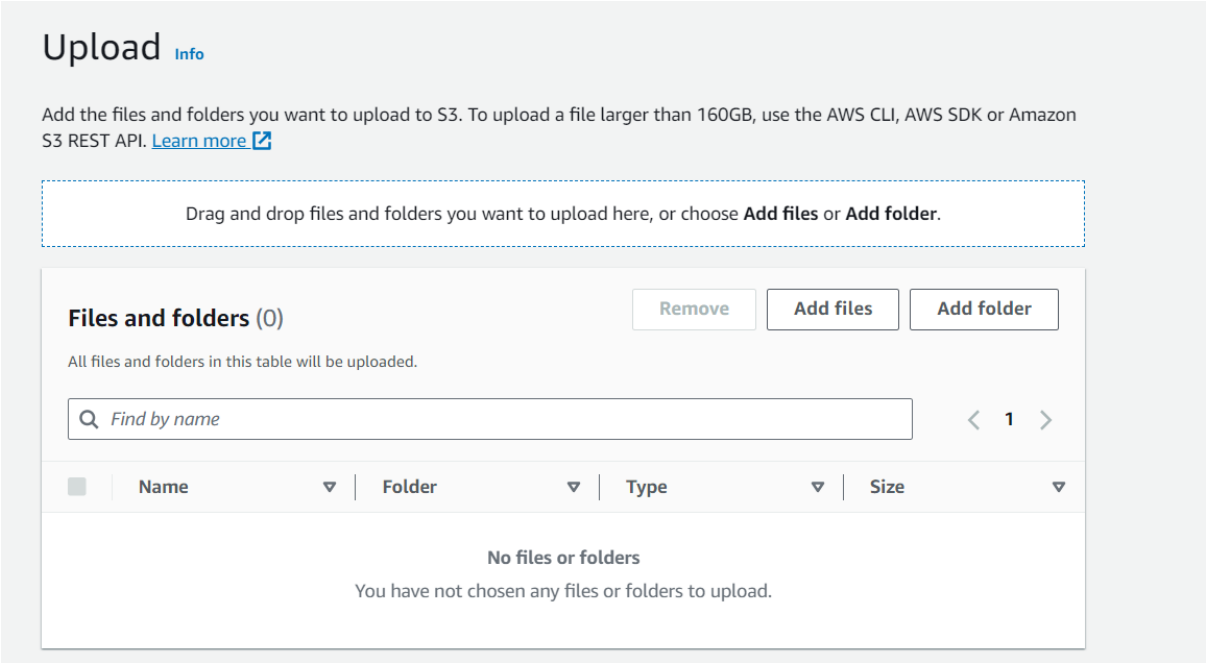


Figure 6: File Select option

Step 8: Click the upload button.

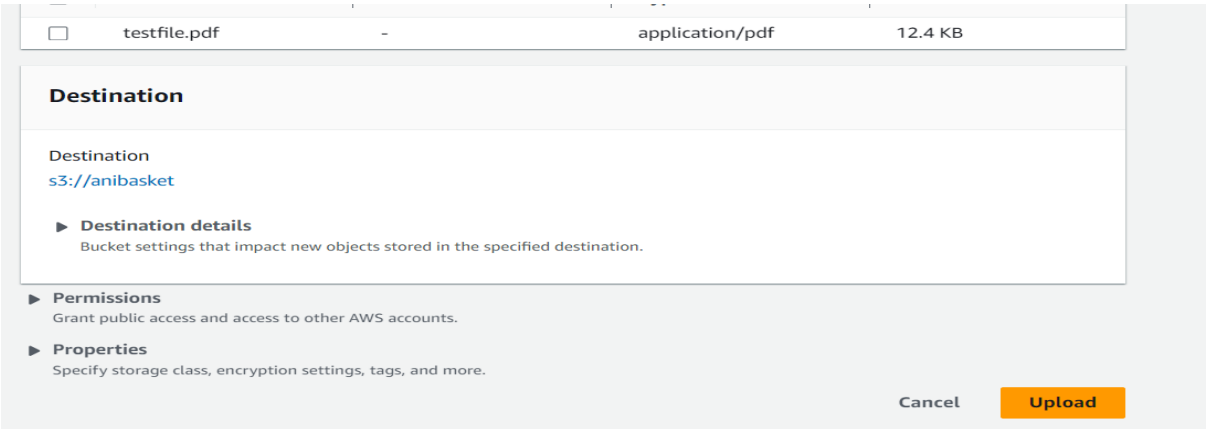


Figure 7: Uploading file

Step 9: Check the upload to make sure that the file is uploaded correctly.

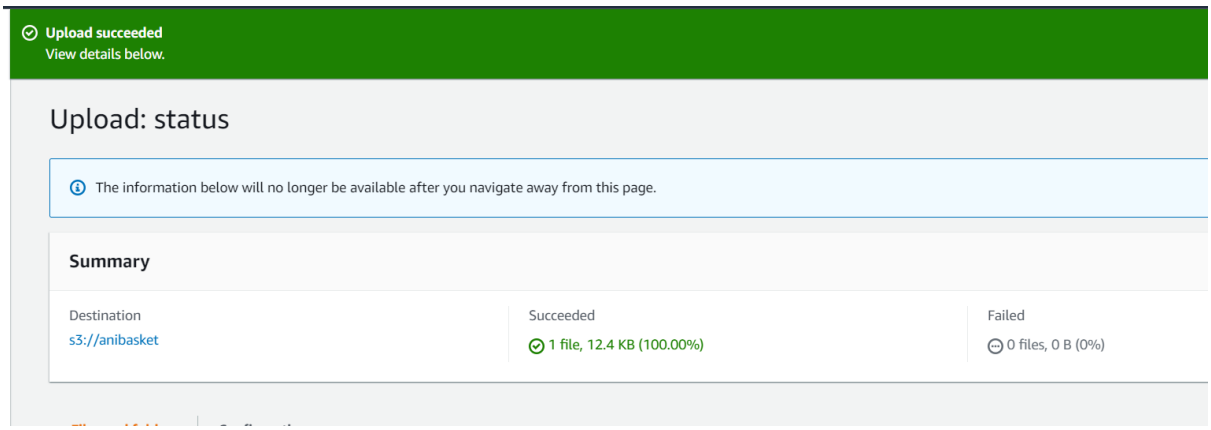


Figure 8: Upload Status page

Result:

The S3 bucket was created successfully in the AP-South-1 regional server and a file was uploaded to the created bucket.

EXPERIMENT 6

DATE: 05/09/23

Aim:

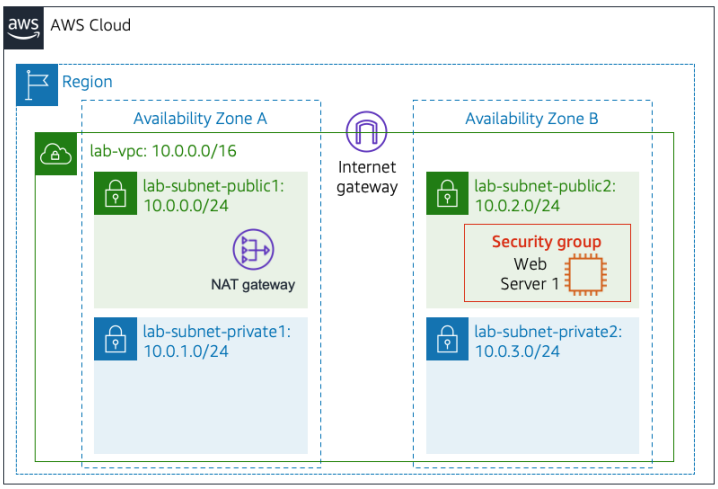
Build your VPC and launch a web server.

Description:

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch Amazon Web Services (AWS) resources into a virtual network that you defined. This virtual network closely resembles a traditional network that you would operate in your own data center, with the benefits of using the scalable infrastructure of AWS. You can create a VPC that spans multiple Availability Zones.

After completing this lab, you should be able to do the following:

- Create a VPC.
- Create subnets.
- Configure a security group.
- Launch an EC2 instance into a VPC.



Public Route Table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	Internet gateway

Private Route Tables

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	NAT gateway

Fig.-1: AWS Cloud Architecture

Working:

Task 1: Create Your VPC

1. In the *Preview* panel on the right, confirm the settings you have configured.
    - **VPC:** lab-vpc
    - **Subnets**:
        - us-east-1a
            - ***Public* subnet name:** lab-subnet-public1-us-east-1a
            - ***Private* subnet name:** lab-subnet-private1-us-east-1a
    - **Route tables**
        - lab-rtb-public
        - lab-rtb-private1-us-east-1a
    - **Network connections**
        - lab-igw
        - lab-nat-public1-us-east-1a

2. At the bottom of the screen, choose **Create VPC**

The VPC resources are created. The NAT Gateway will take a few minutes to activate. Please wait until *all* the resources are created before proceding to the next step.

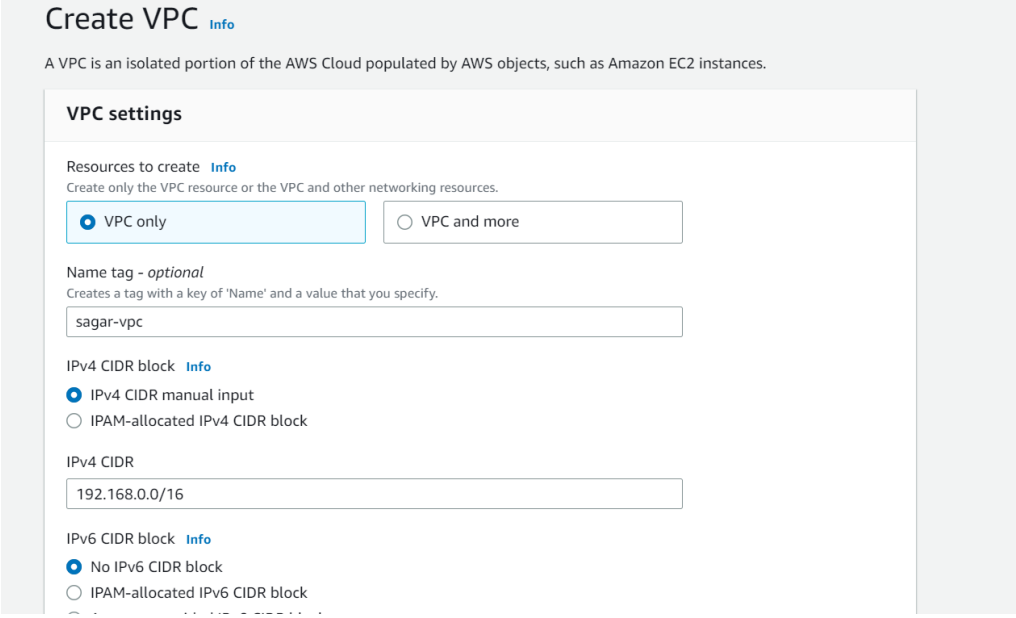


Fig.-2: Creating a VPC

Task 2: Create Additional Subnets

In this task, you will create two additional subnets for the VPC in a second Availability Zone. Having subnets in multiple Availability Zones within a VPC is useful for deploying solutions that provide *High Availability*.

After creating a VPC as you have already done, you can still configure it further, for example, by adding more **subnets**. Each subnet you create resides entirely within one Availability Zone.

11. In the left navigation pane, choose **Subnets**.

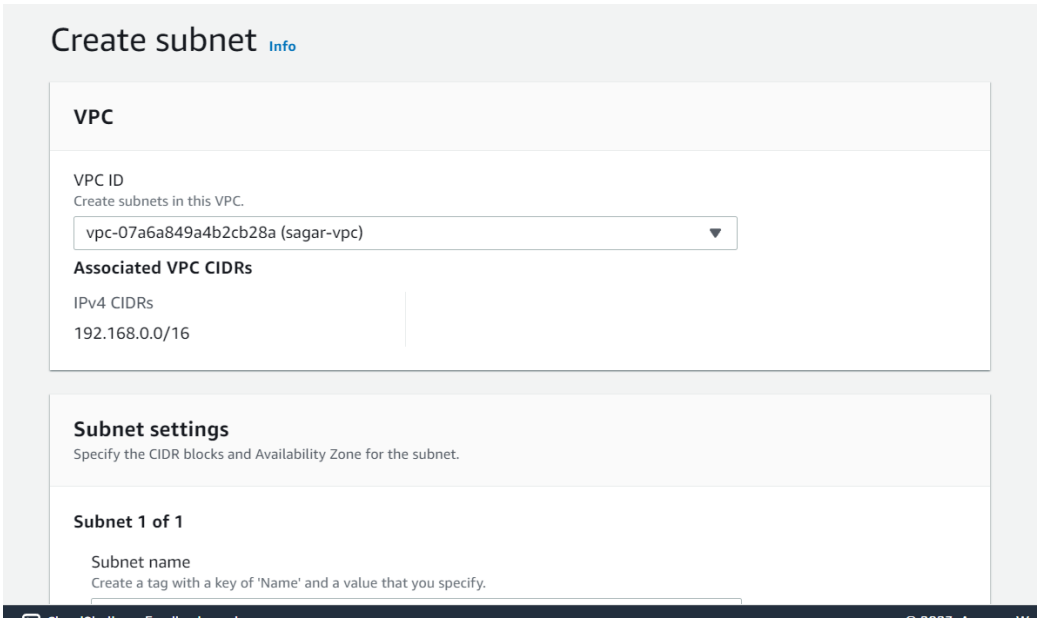


Fig.-3: Creating a subnet

The second *public* subnet was created. You will now create a second *private* subnet.

12. Choose **Create subnet** then configure:
    - **VPC ID:** lab-vpc
    - **Subnet name:** lab-subnet-private2
    - **Availability Zone:** Select the *second* Availability Zone (for example, us-east-1b)
    - **IPv4 CIDR block:** 10.0.3.0/24

The subnet will have all IP addresses starting with **10.0.3.x**.

13. Choose **Create subnet**

14. In the left navigation pane, choose **Route tables**.

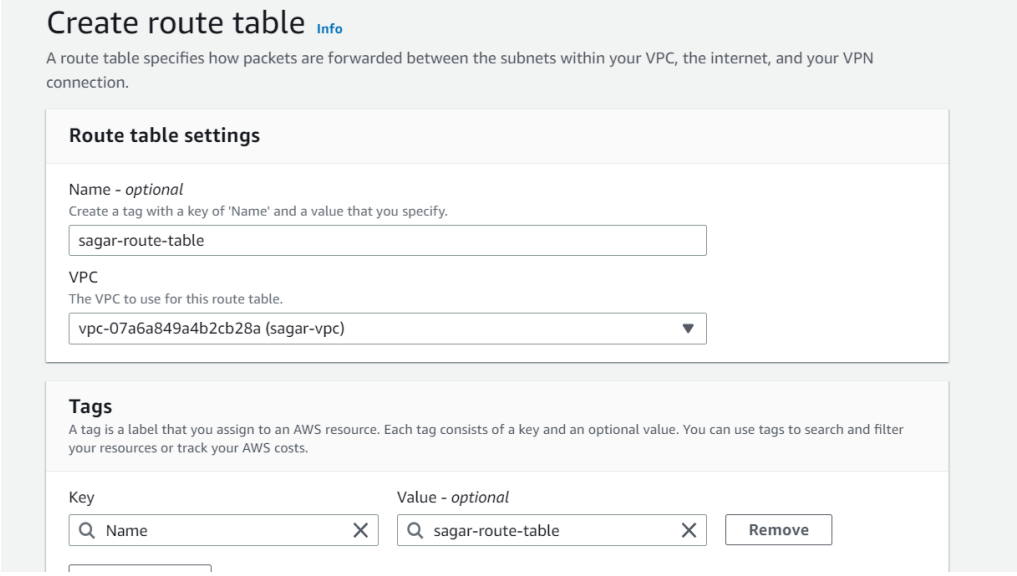


Fig.-4: Creating the route table

15. Select the **lab-rtb-private1-us-east-1a** route table.

16. In the lower pane, choose the **Routes** tab.

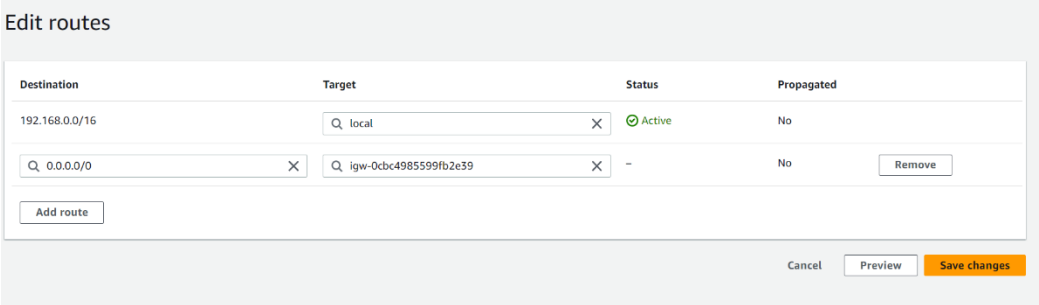


Fig.-5: Editing routes in the route table

21. Choose the **Subnet associations** tab.

22. In the Explicit subnet associations area, choose **Edit subnet associations**

23. Leave **lab-subnet-public1-us-east-1a** selected, but also select **lab-subnet-public2**.

24. Choose **Save associations**

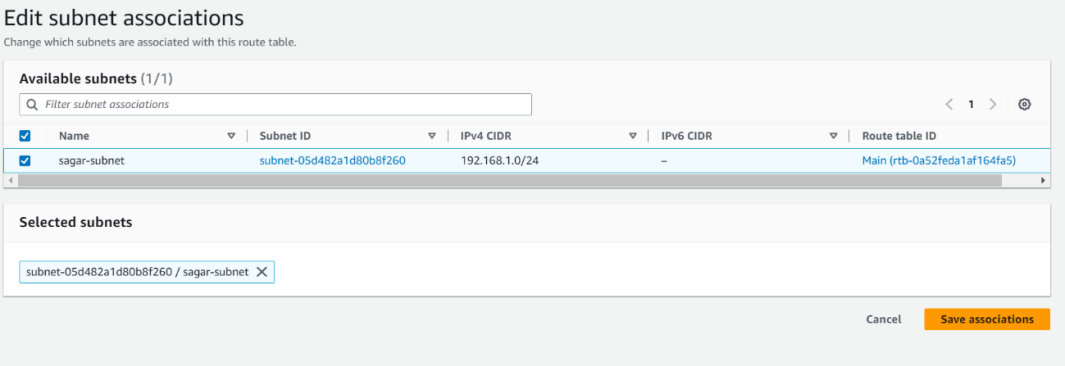


Fig.-6: Changing subnet associations and saving it

Your VPC now has public and private subnets configured in two Availability Zones. The route tables you created in task 1 have also been updated to route network traffic for the two new subnets.

Task 3: Create a VPC Security Group

In this task, you will create a VPC security group, which acts as a virtual firewall. When you launch an instance, you associate one or more security groups with the instance. You can add rules to each security group that allow traffic to or from its associated instances.

29. In the left navigation pane, choose **Security groups**.

30. Choose **Create security group** and then configure:
    - **Security group name:** Web Security Group
    - **Description:** Enable HTTP access
    - **VPC:** choose the X to remove the currently selected VPC, then from the drop-down list choose **lab-vpc**

31. In the **Inbound rules** pane, choose **Add rule**

32. Configure the following settings:
    - **Type:** *HTTP*
    - **Source:** *Anywhere-IPv4*
    - **Description:** Permit web requests

33. Scroll to the bottom of the page and choose **Create security group**

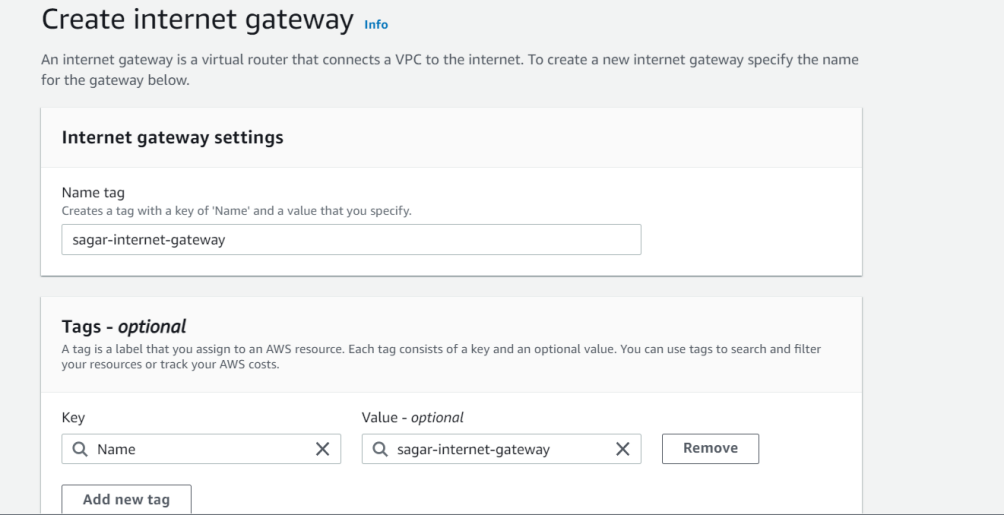


Fig.-7: Creating the internet gateway

You will use this security group in the next task when launching an Amazon EC2 instance.

Task 4: Launch a Web Server Instance

When you name your instance, AWS creates a tag and associates it with the instance. A tag is a key value pair. The key for this pair is ***Name***, and the value is the name you enter for your EC2 instance.

34. Choose an AMI from which to create the instance:
    - In the list of available *Quick Start* AMIs, keep the default **Amazon Linux** selected.
    - Also keep the default **Amazon Linux 2023 AMI** selected.

The type of *Amazon Machine Image (AMI)* you choose determines the Operating System that will run on the EC2 instance that you launch.

35. Choose an Instance type:
    - In the *Instance type* panel, keep the default **t2.micro** selected.

The *Instance Type* defines the hardware resources assigned to the instance.

36. Select the key pair to associate with the instance:
    - From the **Key pair name** menu, select **vockey**.

37. Configure the Network settings:

    - Next to Network settings, choose **Edit**, then configure:
        - **Network:** *lab-vpc*
        - **Subnet:** *lab-subnet-public2* (*not* Private!)
        - **Auto-assign public IP:** *Enable*
        -

38. Configure a script to run on the instance when it launches:
    - Expand the **Advanced details** panel.
    - Scroll to the bottom of the page and then copy and paste the code shown below into the **User data** box:

```
#!/bin/bash
# Install Apache Web Server and PHP
sudo dnf install -y httpd wget php mariadb105-server
# Download Lab files
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-100-ACCLFO-2-9026/2-lab2-vpc/s3/lab-app.zip
unzip lab-app.zip -d /var/www/html/
# Turn on web server
chkconfig httpd on
service httpd start
```

This script will run with root user permissions on the guest OS of the instance. It will run automatically when the instance launches for the first time. The script installs a web server, a database, and PHP libraries, and then it downloads and installs a PHP web application on the web server.

39. At the bottom of the **Summary** panel on the right side of the screen choose **Launch instance**

40. Select **Web Server 1**.

41. Copy the **Public IPv4 DNS** value shown in the **Details** tab at the bottom of the page.

42. Open a new web browser tab, paste the **Public DNS** value and press Enter.

You should see a web page displaying the AWS logo and instance meta-data values.
The complete architecture you deploy

Result

A VPC has been created successfully and an Instance has been launched into it. Launching of an instance is the same as compared to the earlier lab.

EXPERIMENT 7

DATE: 12/09/23

Aim:

Launching and connecting Amazon Elastic Compute Cloud (EC2) of Windows.

Description:

Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 reduces hardware costs so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. You can add capacity (scale up) to handle compute-heavy tasks, such as monthly or yearly processes, or spikes in website traffic. When usage decreases, you can reduce capacity (scale down) again.

An instance is a virtual server in the AWS Cloud. With Amazon EC2, you can set up and configure the operating system and applications that run on your instance.

An Amazon EC2 Windows instance is similar to the traditional Windows Server. After you launch an instance, it briefly goes into the pending state while registration takes place, then it goes into the running state. The instance remains active until you stop or terminate it. You can't restart an instance after you terminate it. You can create a backup image of your instance while it's running, and launch a new instance from that backup image.

Working:

Refer to the following steps to launch an EC2 Windows Instance and connect it.

Step 1 - Log into your AWS account via https://aws.amazon.com/console/.

Step 2 - Open EC2 Dashboard.

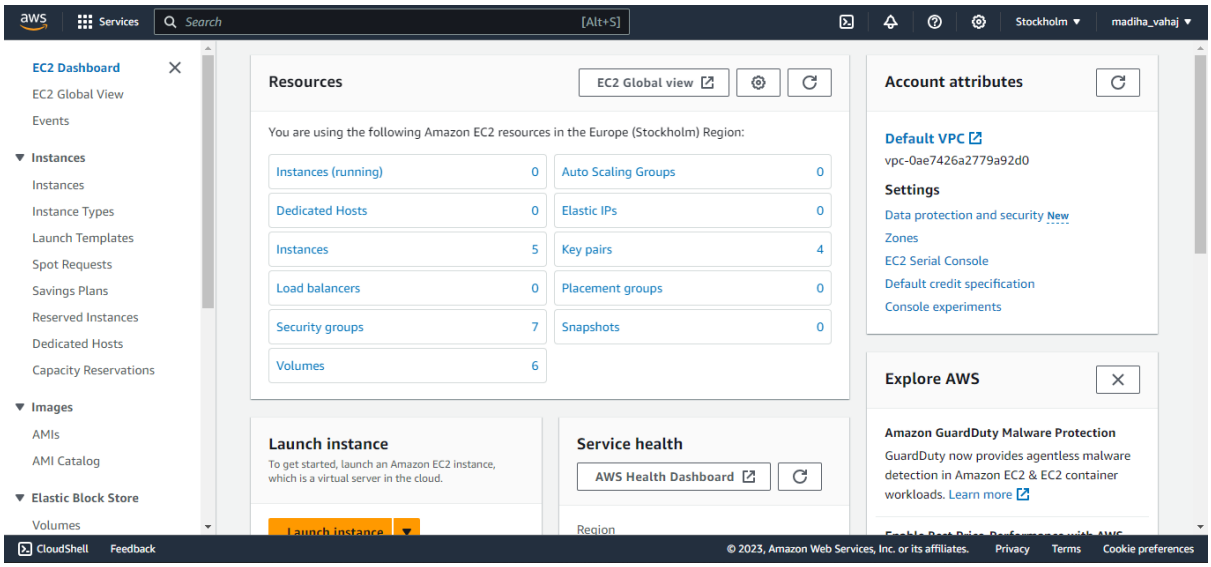


Figure 7.1 EC2 Dashboard

Step 3 – Click on Launch Instance

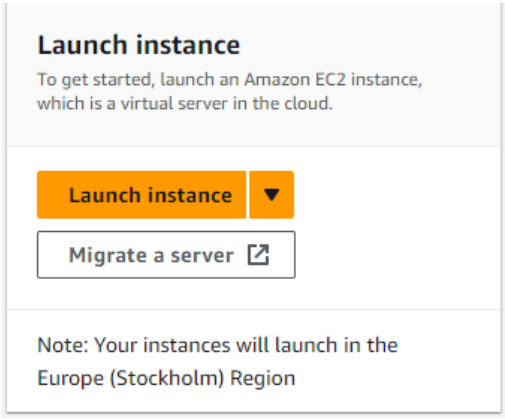


Figure 7.2 Launch Instance Button

Step 4 – Under Name and Tags give your instance a name (Example: EC2_Instance)

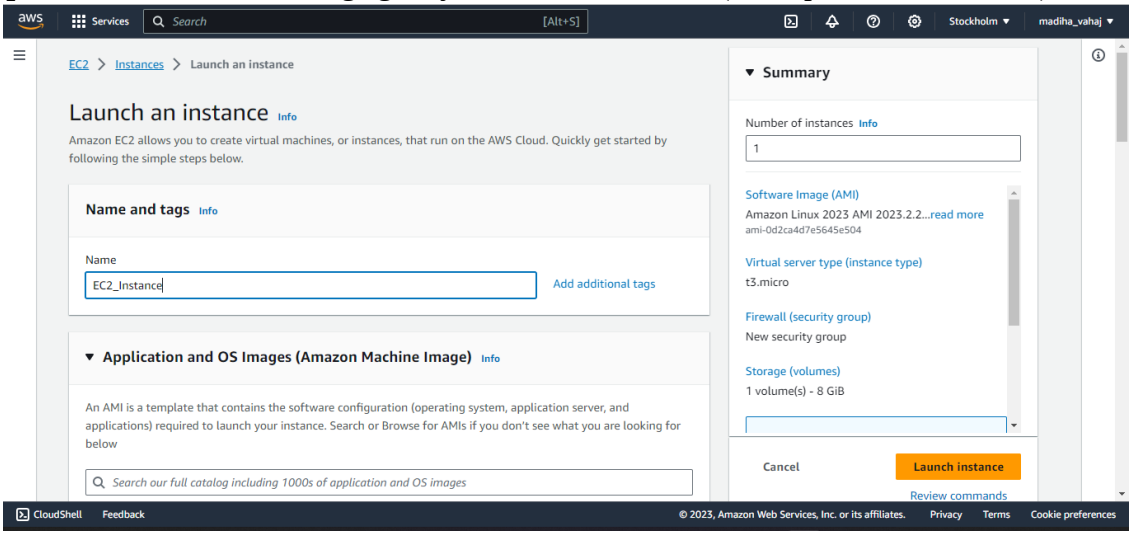


Figure 7.3 Enter name for the Instance

Step 5 - Under Application and OS Images (Amazon Machine Image) choose Quick Start, and then choose Windows. This is the operating system (OS) for your instance.

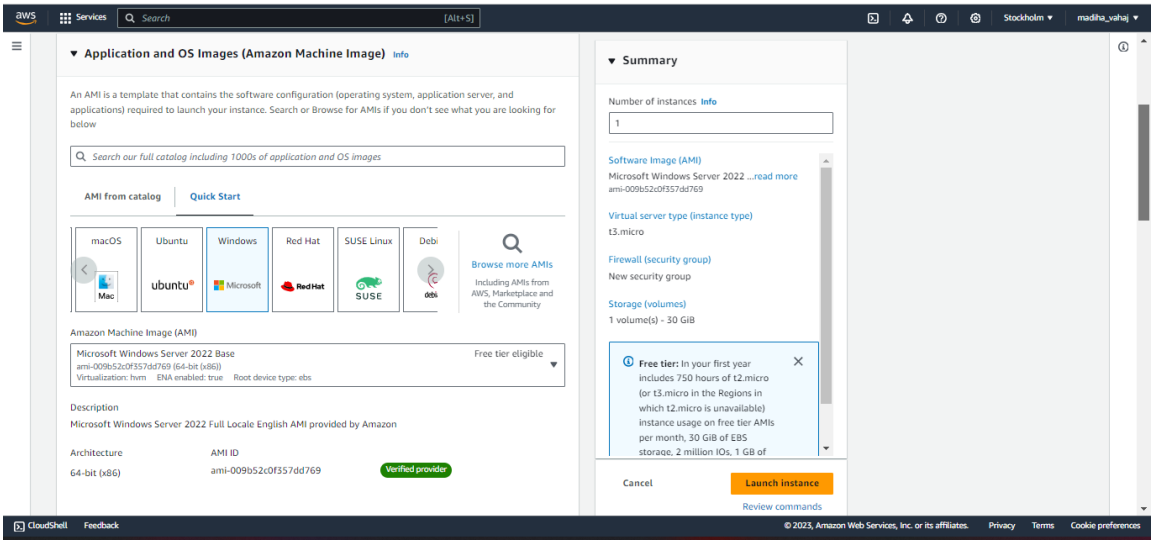


Figure 7.4 Amazon Machine Image

Step 6 – Under Instance Type choose Free Tier Eligible t3.micro instance type.

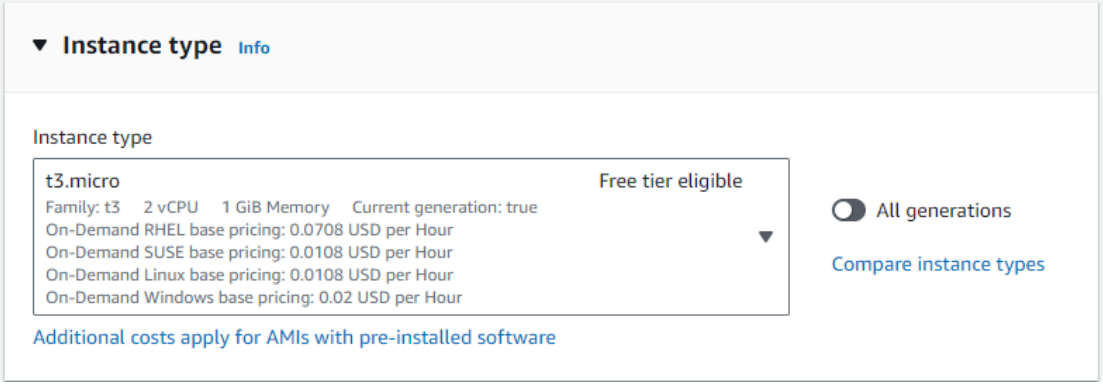


Figure 7.5 Instance Type

Step 7 – Under Key Pair (login), create a new key pair

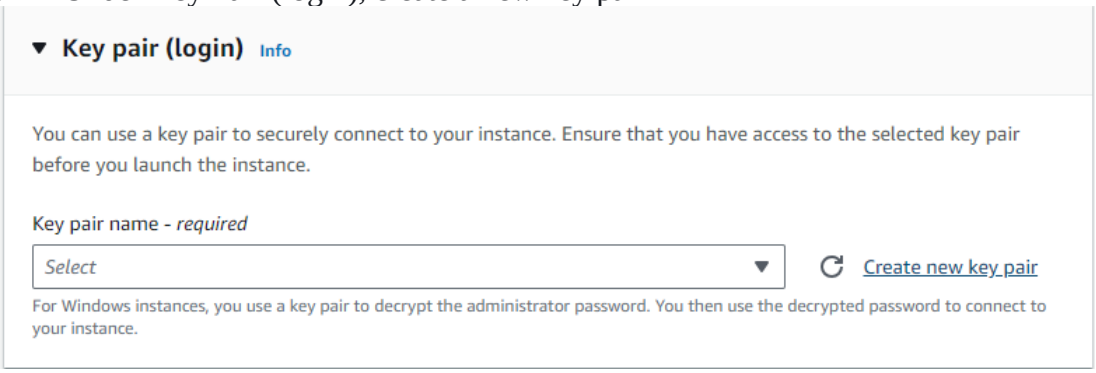


Figure 7.6 New Key Pair Login

After clicking on create a new key pair, give your key a name (Example: EC2_Instance_Key)
Choose RSA as key pair type and .pem as private key file format.
Now click on Create Key Pair

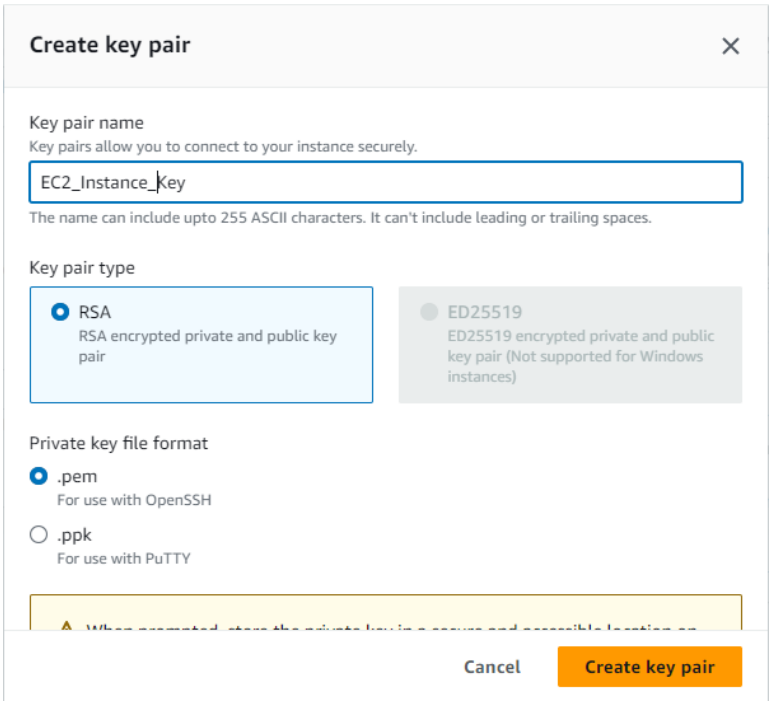


Figure 7.7 Create Key Pair

Step 8 – Under Network Settings, choose create security group and select all three options and in 'Allow RDP traffic from' select Anywhere.

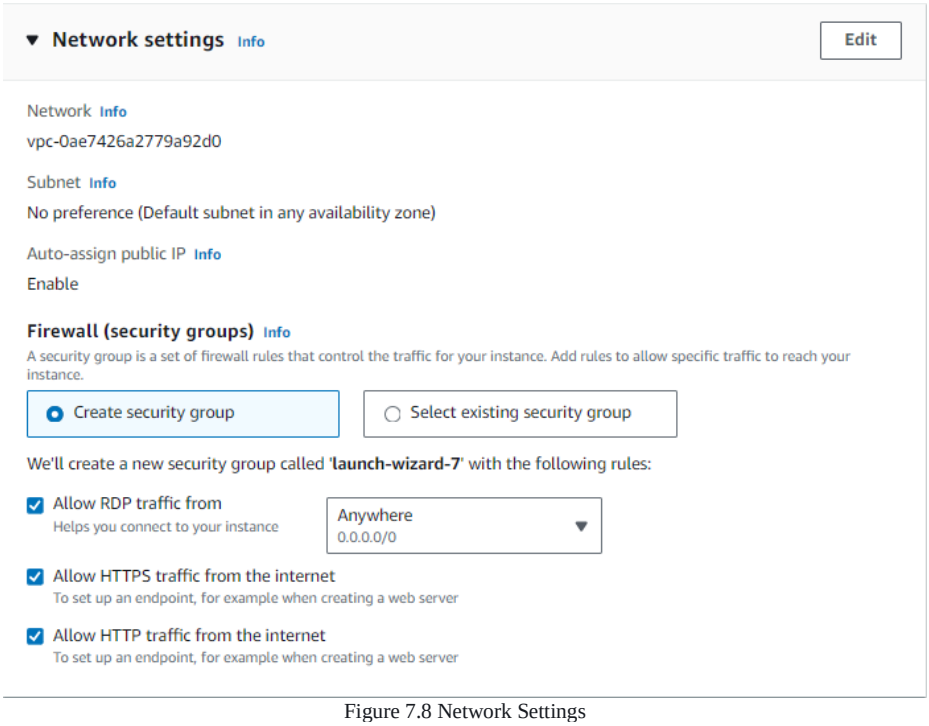


Figure 7.8 Network Settings

Step 9 – Under Configure Storage, choose storage accordingly and Launch the Instance.

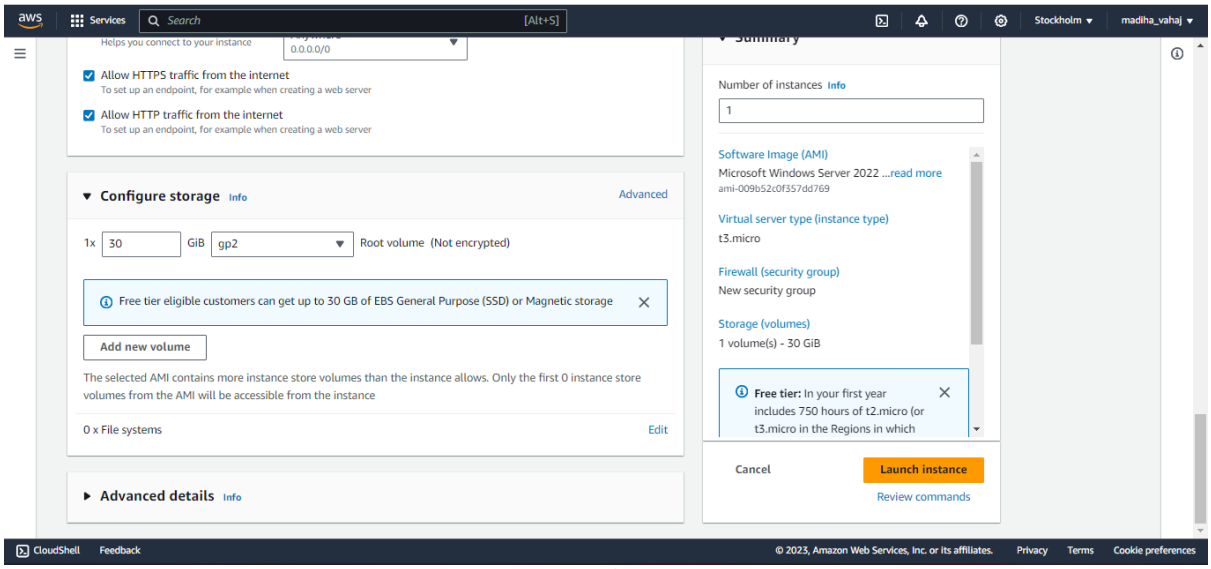


Figure 7.9 Configure Storage

Step 10 – The Instance has been created and is in running state.

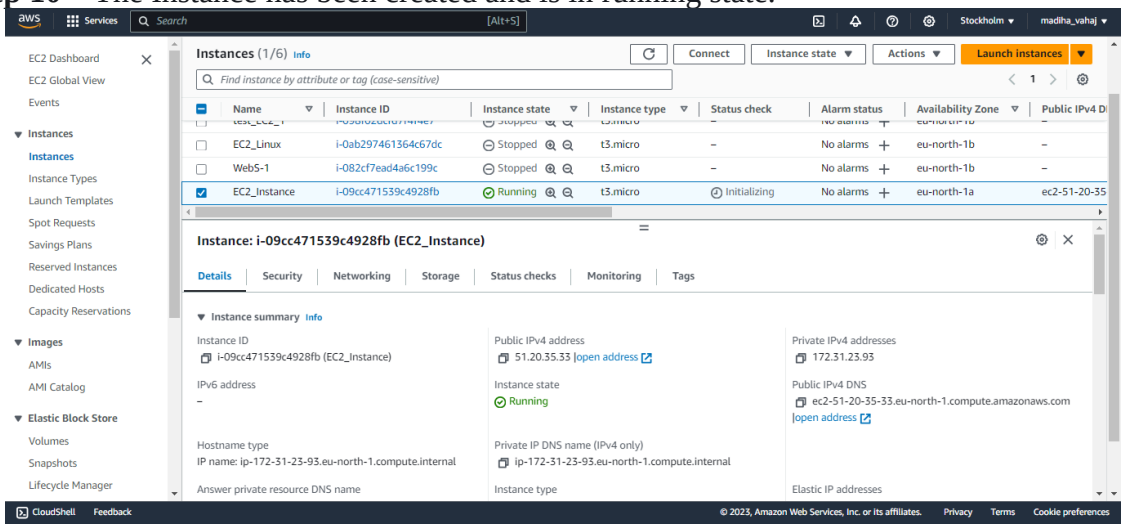


Figure 7.10 Running Instance

Step 11 – Now Connect the Instance with the Windows. Select Connect from the instances dashboard. When you click on connect, a new window will open. There select RDP client and under it choose ‘Connect using RDP client’.
It shows Public DNS, User name, and Password.
Click on ‘Get Password’ to decrypt it.

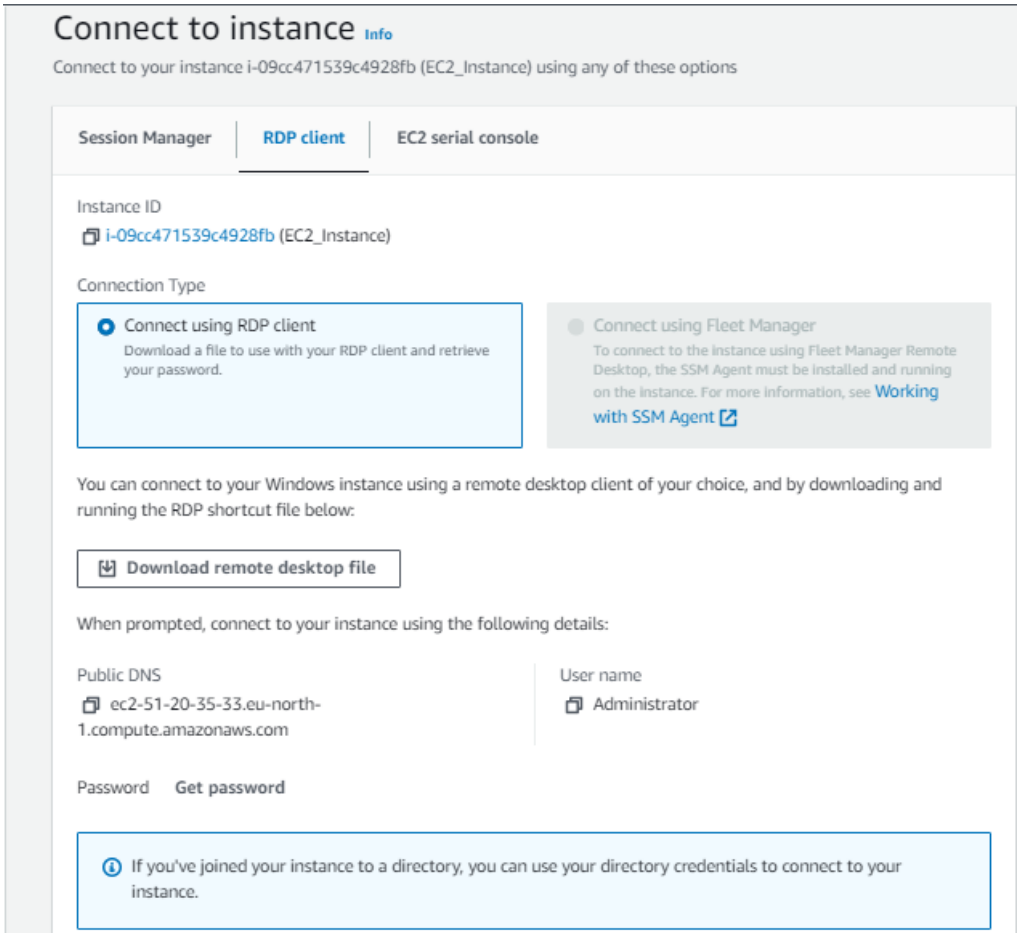


Figure 7.11 Connect to Instance

Step 12 – Upload the private key file that we created previously. The key was downloaded in your computer, select it from there. The contents of the file are displayed. Now click on 'Decrypt Password'.

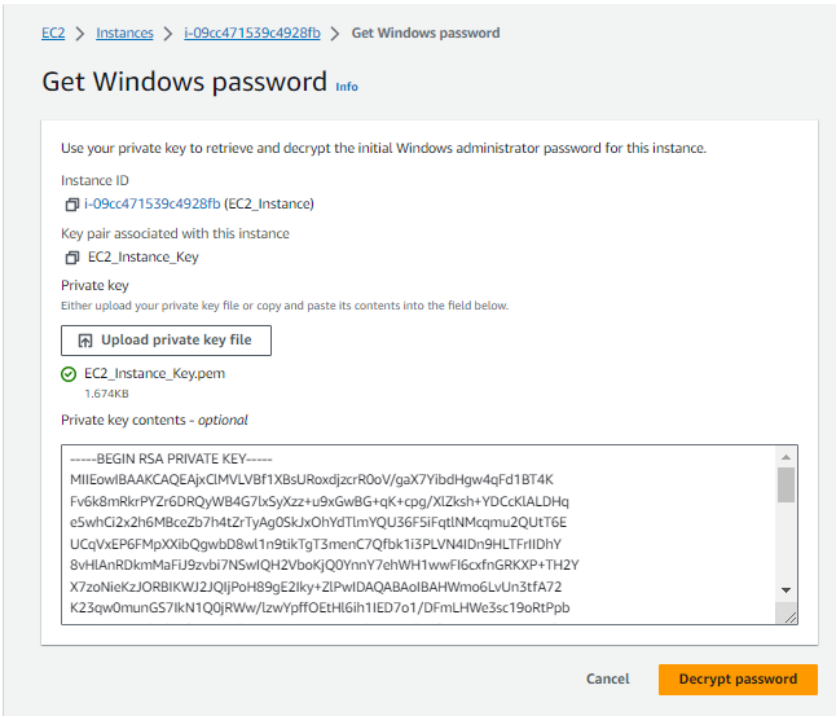


Figure 7.12 Get Windows Password

Step 13 – The password is successfully decrypted and now the password is shown instead of Get Password.

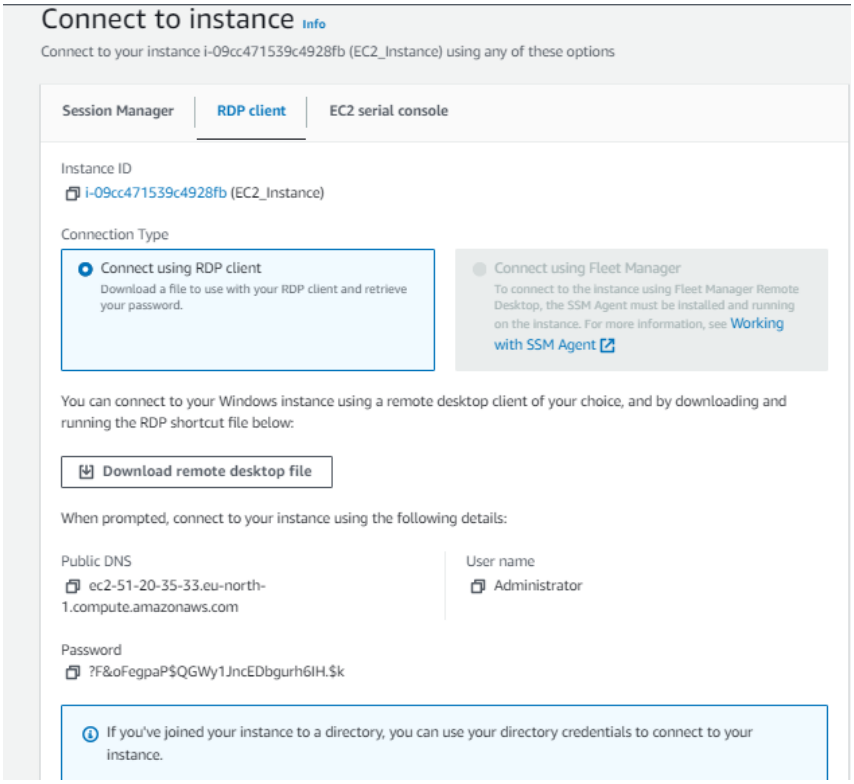


Figure 2.13 Shows Decrypted Password

Step 14 – Now open your Windows and open Remote Desktop Connection and enter the details.

Computer: In this, copy the Public DNS from 'Connect to instance' and paste it.

User name: In this. Copy the User name from 'Connect to instance' and paste it.

Now click on connect.

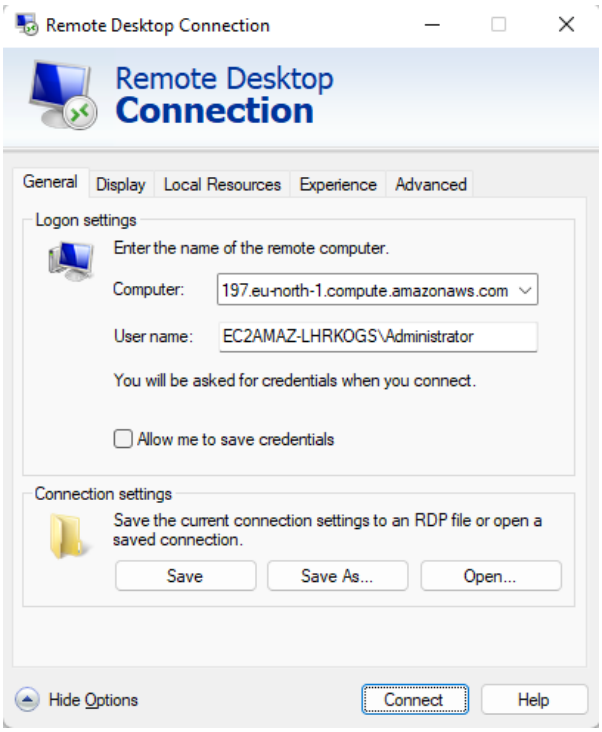


Figure 2.14 Remote Desktop Connection

Step 15 - Copy the Password from ‘Connect to instance’ and paste it. Click OK.

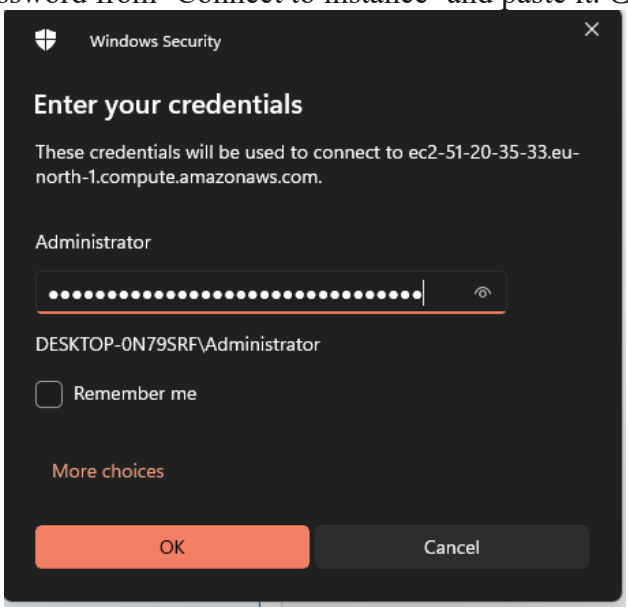


Figure 2.15 Enter your Credentials

Result: The Amazon EC2 Windows Instance has been successfully created.

EXPERIMENT 8

DATE: 26/09/23

Aim:

Build A Database Server in AWS

Description:

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, which allows you to focus on your applications and business. Amazon RDS provides you with six familiar database engines to choose from: Amazon Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MySQL and MariaDB. mazon Relational Database Service (RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. With RDS, users can choose from several popular database engines, including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle, and Microsoft SQL Server.

RDS takes care of time-consuming administrative tasks such as patching and backups, allowing users to focus on their applications. It also provides automated backups, multi-AZ deployments, and point-in-time restore, which helps to improve data durability and availability.

RDS offers flexible scaling options to handle changes in application traffic, allowing users to easily scale up or down their database instance size or replica count. Users can also choose between on-demand, reserved, and spot instances, giving them control over their costs.

RDS provides several security features to help protect sensitive data. It supports encryption at rest and in transit, network isolation, and integrates with AWS Identity and Access Management (IAM) to manage user access.

RDS can be managed using the AWS Management Console, CLI, or API, and integrates with other AWS services, including AWS CloudFormation, AWS Lambda, and Amazon CloudWatch.

Working:

Task 1: Create a Security Group for the RDS DB Instance

In this task, you will create a security group to allow your web server to access your RDS DB instance. The security group will be used when you launch the database instance.

1. In the **AWS Management Console**, on the Services menu, choose **VPC**.
2. In the left navigation pane, choose **Security Groups**.
3. Choose Create security group and then configure:
    - **Security group name:** DB Security Group
    - **Description:** Permit access from Web Security Group
    - **VPC:** *Lab VPC*

    You will now add a rule to the security group to permit inbound database requests.

4. In the **Inbound rules** pane, choose Add rule

    The security group currently has no rules. You will add a rule to permit access from the *Web Security Group*.

5. Configure the following settings:
    - **Type:** *MySQL/Aurora (3306)*
    - **CIDR, IP, Security Group or Prefix List:** Type sg and then select *Web Security Group*.

    This configures the Database security group to permit inbound traffic on port 3306 from any EC2 instance that is associated with the *Web Security Group*.

6. Choose Create security group

    You will use this security group when launching the Amazon RDS database.

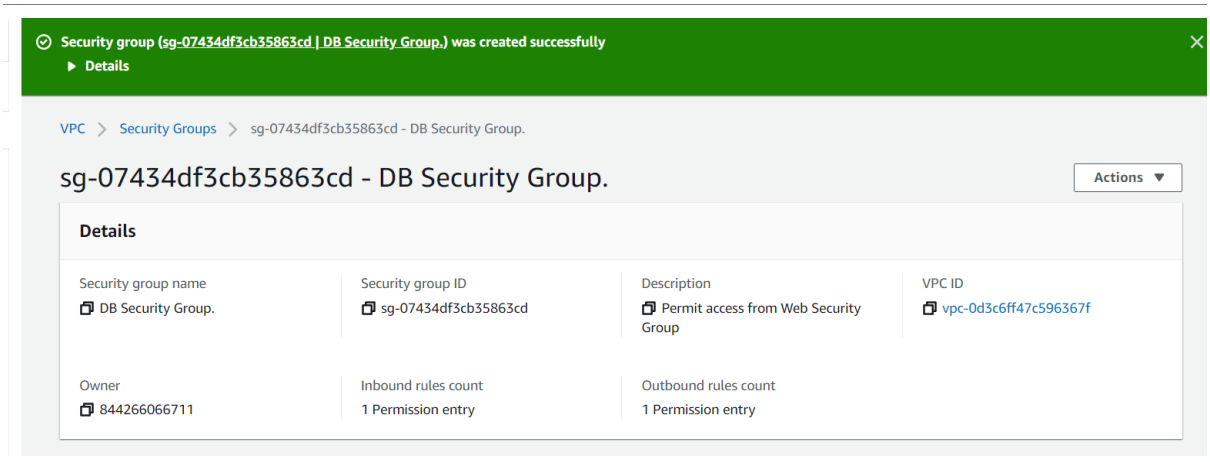


Fig.-1: Creating the DB security group

Task 2: Create a DB Subnet Group

In this task, you will create a *DB subnet group* that is used to tell RDS which subnets can be used for the database. Each DB subnet group requires subnets in at least two Availability Zones.

1. On the Services menu, choose **RDS**.
2. In the left navigation pane, choose **Subnet groups**.

   If the navigation pane is not visible, choose the menu icon in the top-left corner.

3. Choose Create DB Subnet Group then configure:
   - **Name:** DB-Subnet-Group
   - **Description:** DB Subnet Group
   - **VPC:** *Lab VPC*
4. Scroll down to the **Add Subnets** section.
5. Expand the list of values under **Availability Zones** and select the first two zones: **us-east-1a** and **us-east-1b**.
6. Expand the list of values under **Subnets** and select the subnets associated with the CIDR ranges **10.0.1.0/24** and **10.0.3.0/24**.

   These subnets should now be shown in the **Subnets selected** table.

7. Choose Create

   You will use this DB subnet group when creating the database in the next task.

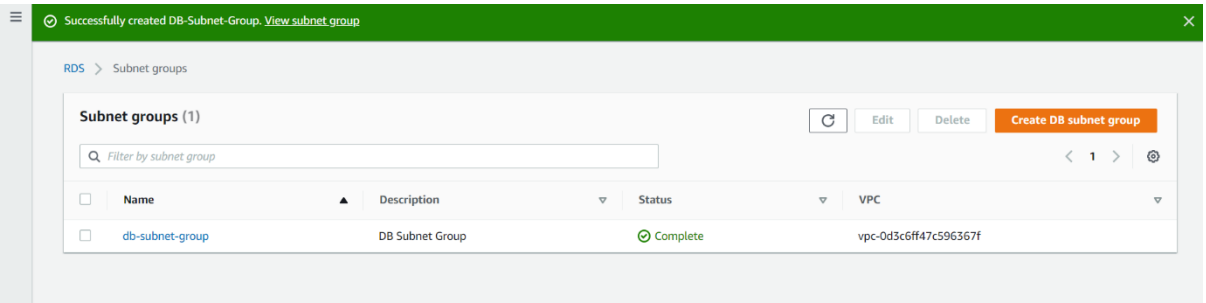


Fig.-2: Creating the DB subnet group

Task 3: Create an Amazon RDS DB Instance

In this task, you will configure and launch a Multi-AZ Amazon RDS for MySQL database instance.

Amazon RDS **Multi-AZ** deployments provide enhanced availability and durability for Database (DB) instances, making them a natural fit for production database workloads. When you provision a multi-AZ DB instance, Amazon RDS automatically creates a primary DB instance and synchronously replicates the data to a standby instance in a different Availability Zone (AZ).

1. In the left navigation pane, choose **Databases**.
2. Choose Create database

   If you see **Switch to the new database creation flow** at the top of the screen, please choose it.

3. Select **MySQL**.
4. Under **Settings**, configure:
   - **DB instance identifier:** lab-db
   - **Master username:** main
   - **Master password:** lab-password
   - **Confirm password:** lab-password

5. Under **DB instance class**, configure:
   - Select **Burstable classes (includes t classes)**.
   - Select *db.t3.micro*
6. Under **Storage**, configure:
   - **Storage type:** *General Purpose (SSD)*
   - **Allocated storage:** *20*
7. Under **Connectivity**, configure:
   - **Virtual Private Cloud (VPC):** *Lab VPC*
8. Under **Existing VPC security groups**, from the dropdown list:
   - Choose *DB Security Group*.
   - Deselect *default*.
9. Expand **Additional configuration**, then configure:
   - **Initial database name:** lab
   - Uncheck **Enable automatic backups**.
   - Uncheck **Enable encryption**
   - Uncheck **Enable Enhanced monitoring**.

   This will turn off backups, which is not normally recommended, but will make the database deploy faster for this lab.

10. Choose Create database

    Your database will now be launched.
    If you receive an error that mentions "not authorized to perform: iam:CreateRole", make sure you unchecked *Enable Enhanced monitoring* in the previous step.

11. Choose **lab-db** (choose the link itself).

    You will now need to wait **approximately 4 minutes** for the database to be available. The deployment process is deploying a database in two different Availability zones. While you are waiting, you might want to review the Amazon RDS FAQs or grab a cup of coffee.

12. Wait until **Info** changes to **Modifying** or **Available**.
13. Scroll down to the **Connectivity & security** section and copy the **Endpoint** field.

It will look like: *lab-db.cggq8lhnxvnv.us-west-2.rds.amazonaws.com*

14. Paste the Endpoint value into a text editor. You will use it later in the lab.

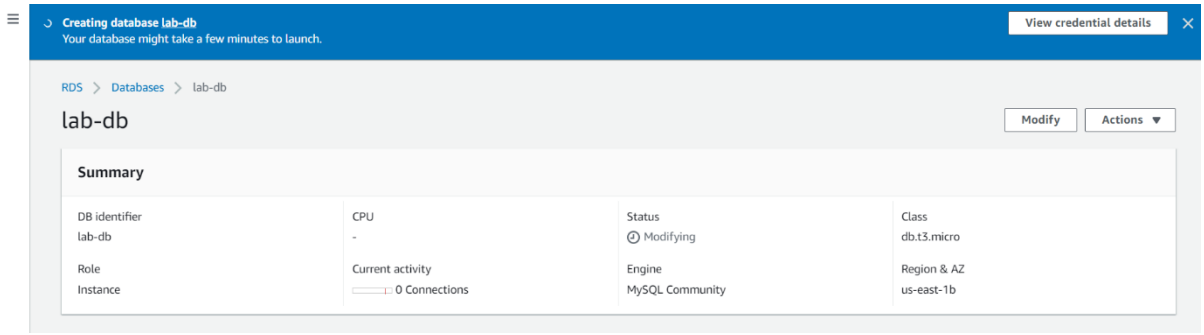


Fig.-3: Running the lab database

Task 4: Interact with Your Database

In this task, you will open a web application running on your web server and configure it to use the database.

1. To copy the **WebServer** IP address, choose on the Details drop down menu above these instructions, and then choose Show.
2. Open a new web browser tab, paste the *WebServer* IP address and press Enter.

   The web application will be displayed, showing information about the EC2 instance.

3. Choose the **RDS** link at the top of the page.

   You will now configure the application to connect to your database.

4. Configure the following settings:
   - **Endpoint:** Paste the Endpoint you copied to a text editor earlier
   - **Database:** lab
   - **Username:** main
   - **Password:** lab-password
   - Choose **Submit**

   A message will appear explaining that the application is running a command to copy information to the database. After a few seconds the application will display an **Address Book**.
   The Address Book application is using the RDS database to store information.

5. Test the web application by adding, editing and removing contacts.

   The data is being persisted to the database and is automatically replicating to the second Availability Zone.

Fig.-4: Displaying the Address Book database

Fig.-5: Displaying the Address Book Database

Result:

A database server has been created successfully on the cloud.

EXPERIMENT 9

DATE: 10/10/23

Aim:

To demonstrate the working of CloudWatch using AWS.

Description:

Amazon CloudWatch is a comprehensive monitoring and observability service provided by Amazon Web Services (AWS). It helps you collect and track various metrics, collect and monitor log files, and set alarms. This service is primarily designed for managing and monitoring AWS resources and applications, but it can also be extended to on-premises resources and other cloud environments.

Here's a brief description of key features and components of Amazon CloudWatch:

1. **Metrics:** CloudWatch allows you to collect and store data about the performance and health of your resources. These metrics can be generated by various AWS services, custom applications, or any other data source you want to monitor. Metrics are stored for up to 15 months, and you can visualize them using graphs and dashboards.
2. **Alarms:** You can set up alarms based on CloudWatch metrics to trigger automated actions or notifications when certain thresholds are breached. For example, you can set up an alarm to notify you when CPU utilization on an EC2 instance exceeds a certain percentage.
3. **Dashboards:** CloudWatch provides customizable dashboards that allow you to visualize your application's performance data and metrics in one place. You can create widgets and arrange them to create a custom monitoring dashboard that suits your needs.
4. **Logs:** CloudWatch Logs allows you to store, monitor, and access log files from your AWS resources. You can create log groups to organize log streams, and set up filters to extract and forward specific log events to other AWS services for analysis.
5. **Events:** CloudWatch Events allows you to respond to changes in your AWS environment. You can create rules that match events and trigger actions, such as running an AWS Lambda function or notifying an Amazon SNS topic.
6. **Synthetics:** CloudWatch Synthetics allows you to create canaries, which are scripts that monitor your endpoints and APIs. These canaries can be set up to mimic user interactions and ensure your applications are performing as expected.

7. **Application Insights**: For applications that use AWS resources, CloudWatch Application Insights provides a consolidated view of your application's health and performance. It can automatically detect issues and provide root cause analysis.
8. **X-Ray Integration:** CloudWatch seamlessly integrates with AWS X-Ray, which provides end-to-end tracing for distributed applications. This integration helps you understand how requests are flowing through your services.
9. **Cross-Account and Cross-Region Monitoring**: You can set up CloudWatch to monitor resources in different AWS accounts and regions, allowing you to have centralized visibility and control over your entire AWS infrastructure.
10. **Custom Metrics:** In addition to AWS-generated metrics, you can publish your own custom metrics to CloudWatch, allowing you to monitor specific application-level data and performance statistics.

Amazon CloudWatch is a crucial tool for DevOps, system administrators, and developers who need to ensure the availability, reliability, and performance of their AWS resources and applications. It provides the necessary insights and tools to manage, monitor, and troubleshoot complex cloud environments.

Procedure:

1) Log into your AWS account via https://aws.amazon.com/console/.

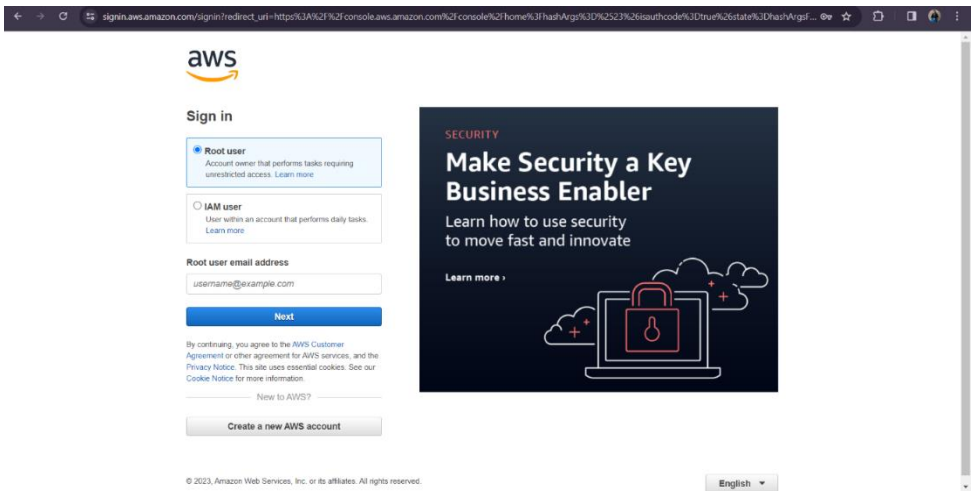


Fig 9.1: Login to the console

2) Launch an Instance.

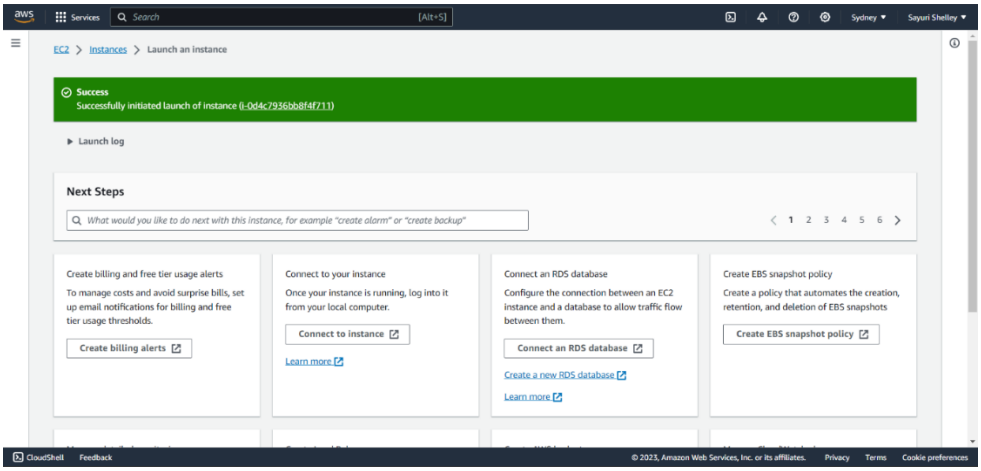


Fig 9.2: Launch an instance

3) Navigate to CloudWatch service, Click on Alarms in the left pane and select Create Alarms-> Select Metric -> EC2 -> Per-Instance Metrics and select the metric of CPU Utilization and click on Select Metric .

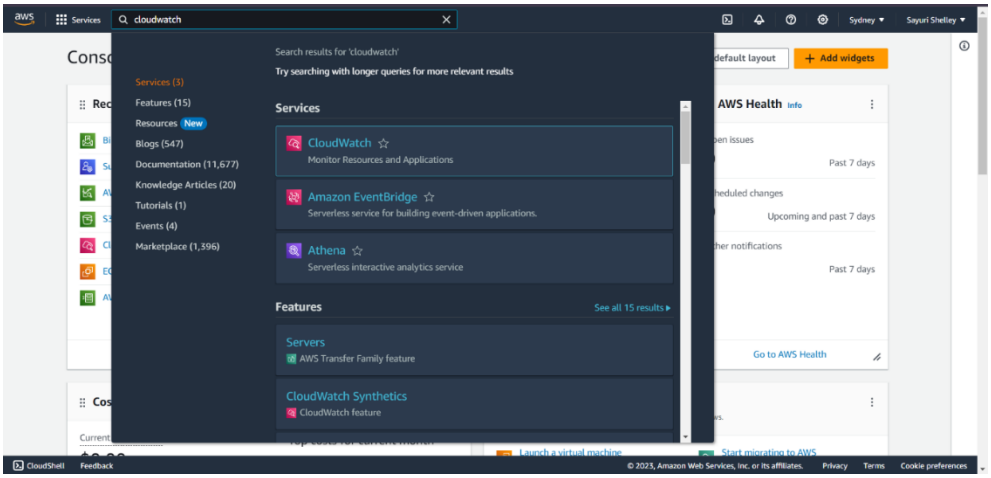


Fig 9.3: Search for CloudWatch services

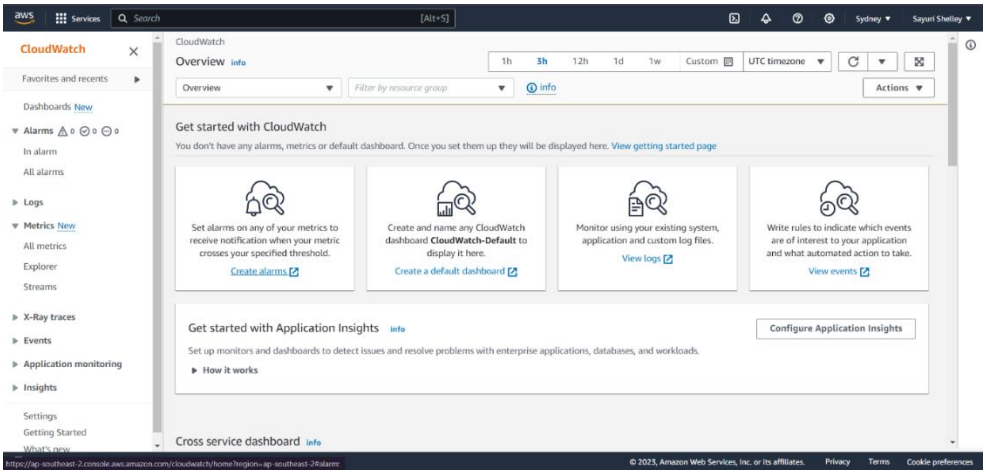


Fig 9.4: check the CloudWatch dashboard

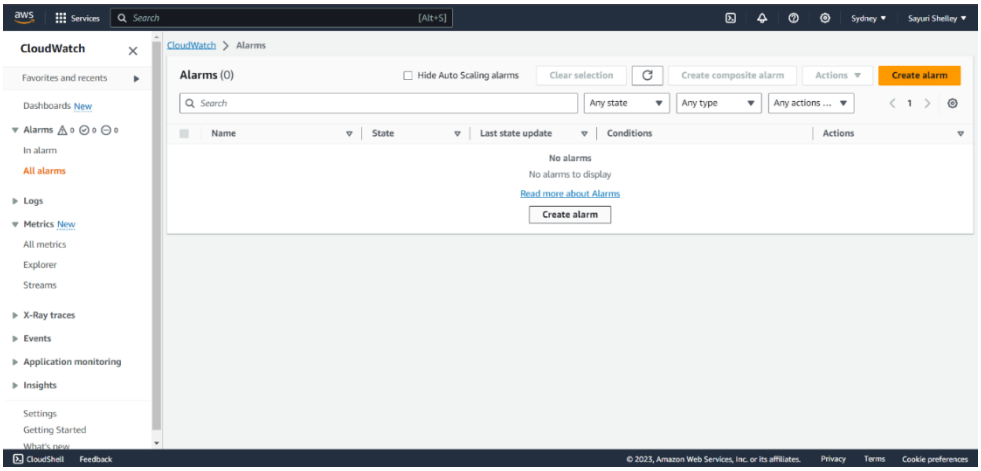


Fig 9.5: click create alarm

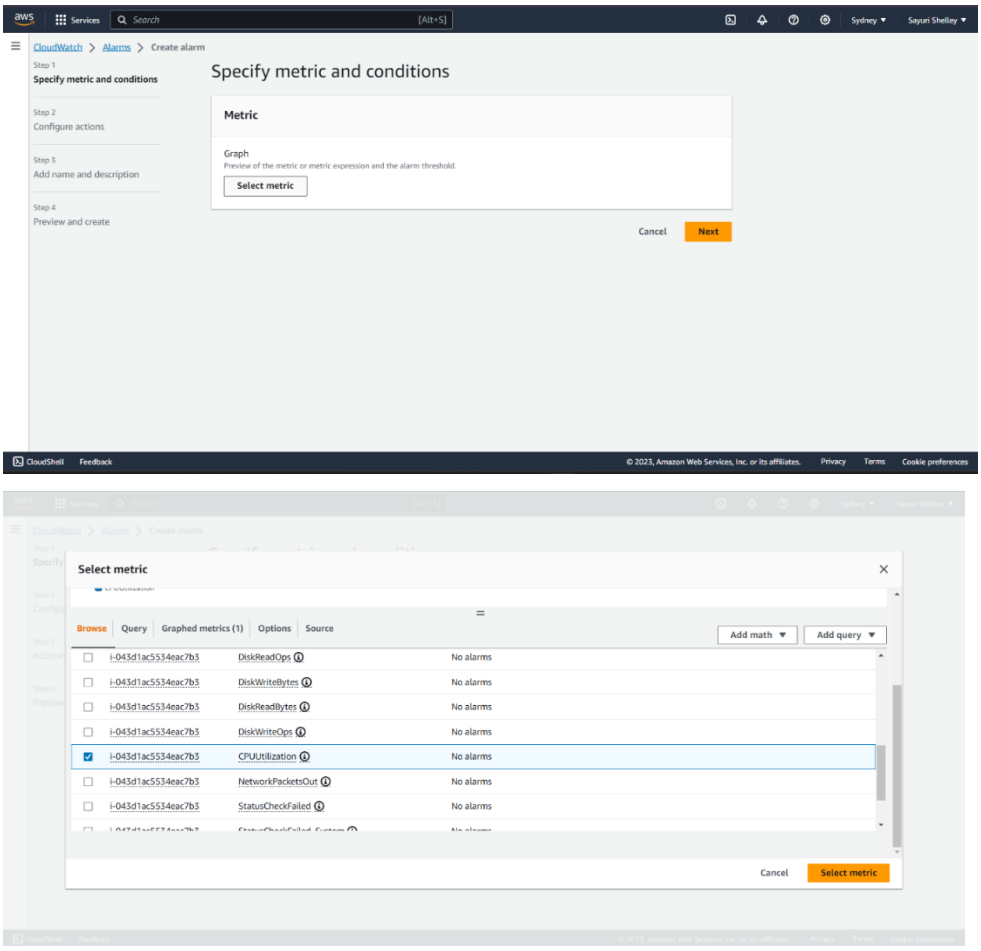
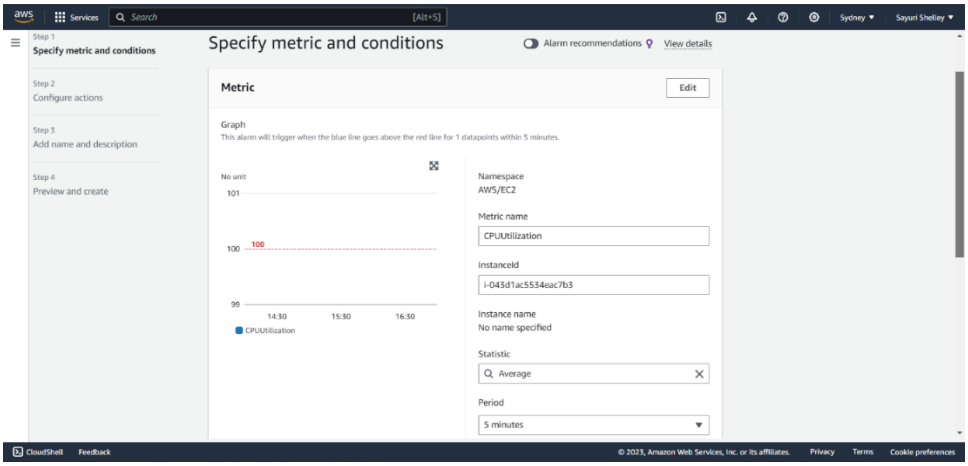


Fig 9.6: search for the instance you made

4) Specify the metric and conditions, Period=5 minutes, threshold greater than 100 -> Next



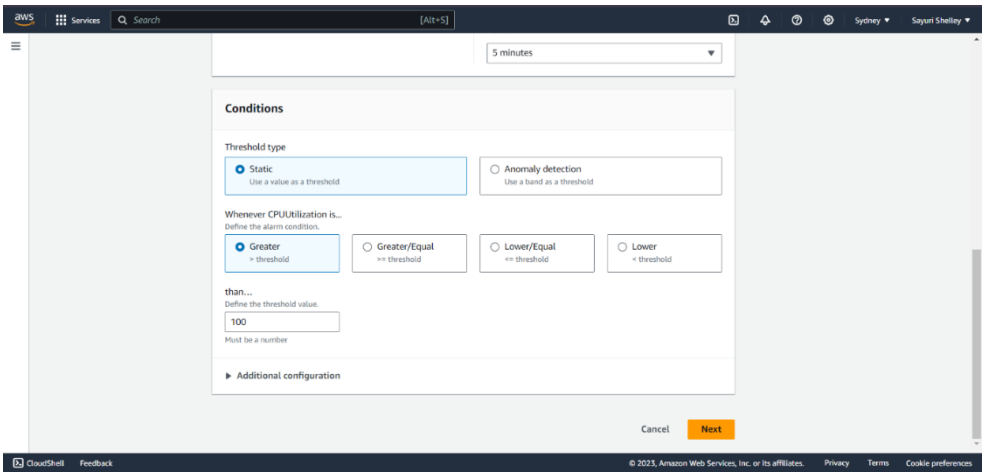


Fig 9.7: Set the conditions

5) For Configure Actions, Create a new topic, enter email id and select Create Topic , click Next

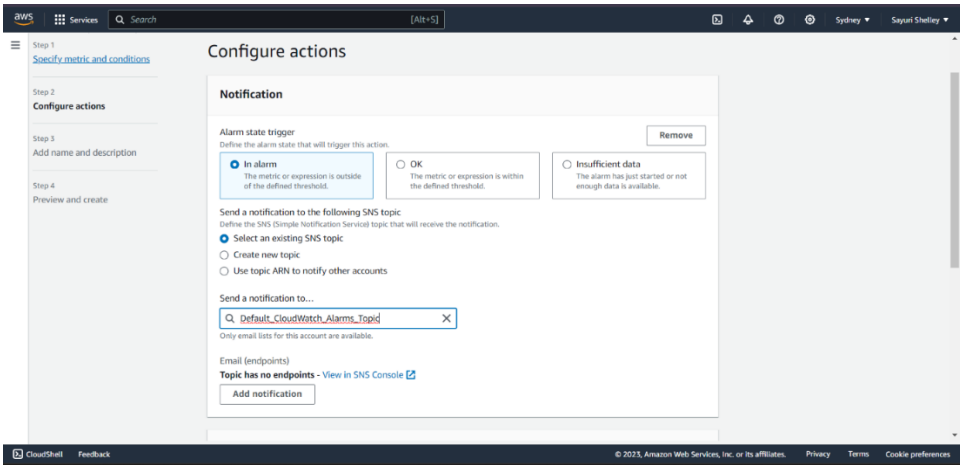


Fig 9.8: Configure the actions

6) Give a name to the alarm -> Create alarm

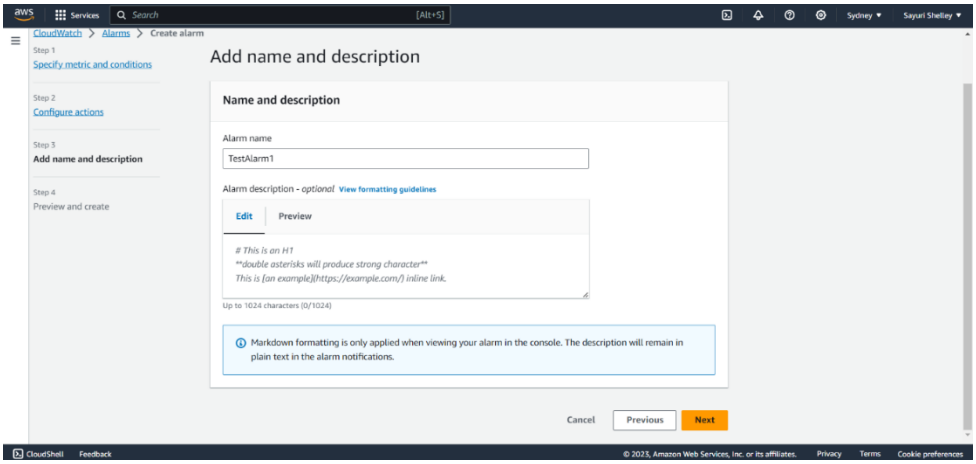


Fig 9.9: Give a name to alarm

7) Alarm is successfully created.

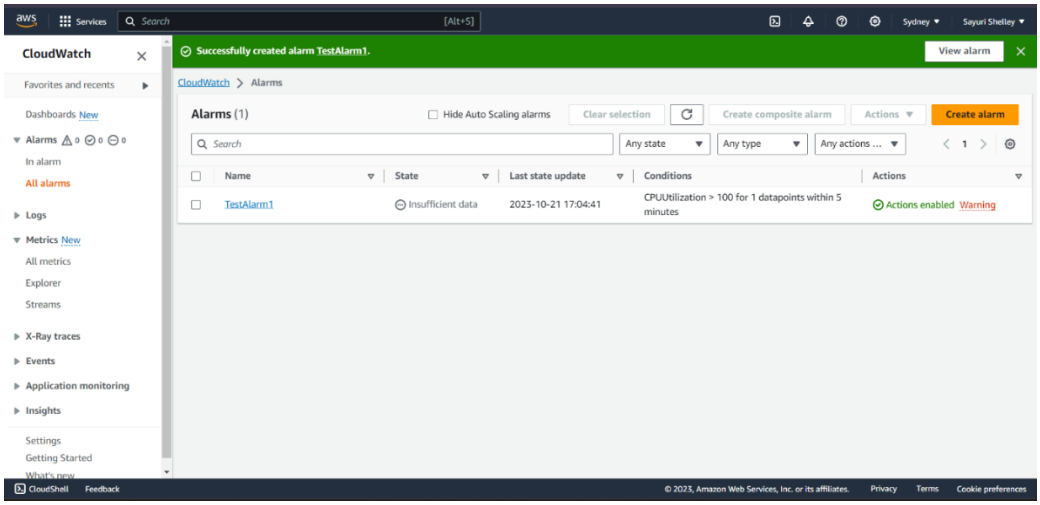


Fig 9.10: Alarm is created

Result: Alarm feature of AWS Cloud Watch was explored.

EXPERIMENT 10

DATE: 17/10/23

Aim:

Working on Cloud Trail

Description:

AWS CloudTrail is an AWS service that helps you enable operational and risk auditing, governance, and compliance of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail. Events include actions taken in the AWS Management Console, AWS Command Line Interface, and AWS SDKs and APIs.

CloudTrail is enabled on your AWS account when you create it. When activity occurs in your AWS account, that activity is recorded in a CloudTrail event. You can easily view recent events in the CloudTrail console by going to Event history. For an ongoing record of activity and events in your AWS account, create an event data store or create a trail. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

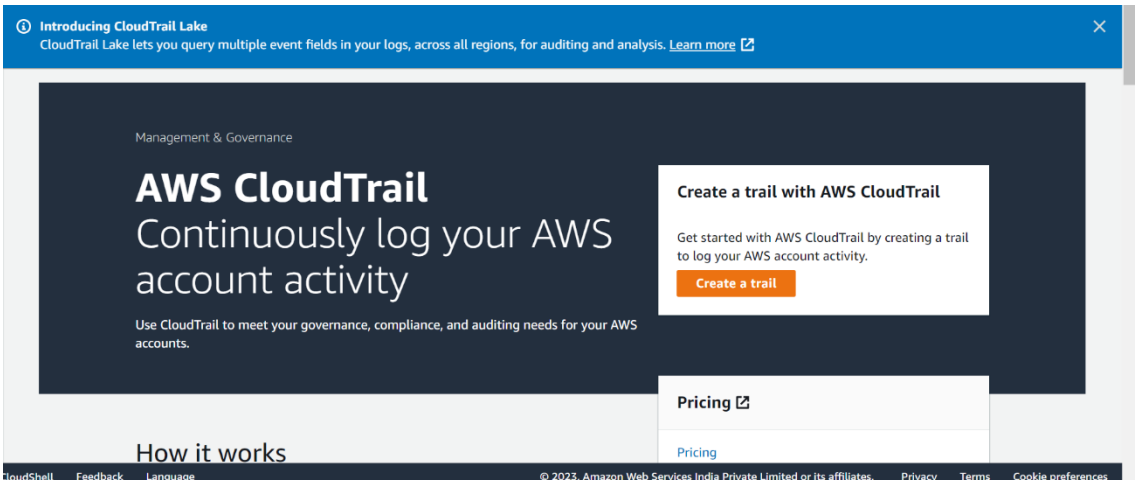


Fig. 10.1: AWS Cloud Trail Dashboard

Working:

1. Log in to your AWS Management Console and navigate to the CloudTrail service.
2. Click on the "Create trail" button to create a new CloudTrail trail.
3. Choose a name for your trail and select the region where you want to store your CloudTrail logs.

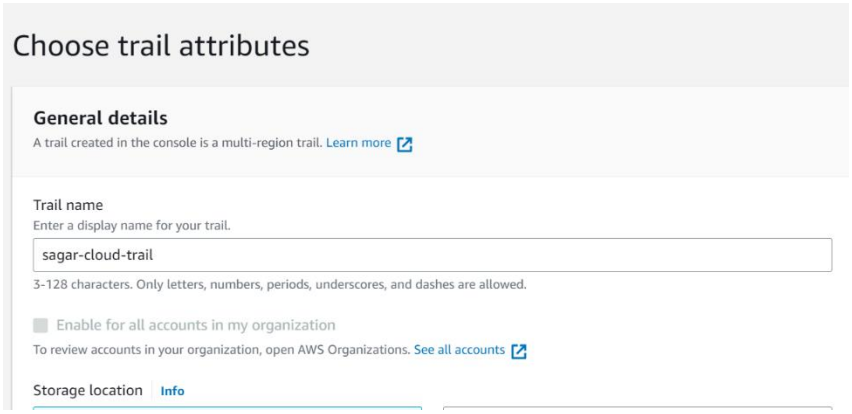


Fig.10.2: Choosing the trail attributes

4. Decide which events you want to log. You can log all events or select specific ones based on the services and resources you want to track.
5. Choose a storage location for your logs. You can store them in an S3 bucket, a CloudWatch Logs group, or both.

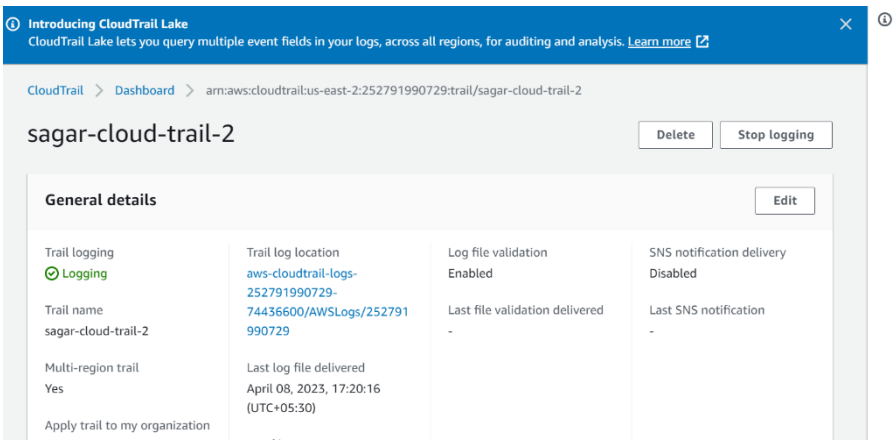


Fig.10.3: Creation of the cloud trail

6. Choose any additional options you want to enable, such as log file validation or cloud watch event delivery.

7. Review your settings and click "Create trail" to create your CloudTrail trail.

8. Once your trail is created, you can view and manage it from the CloudTrail dashboard. You can also configure CloudWatch alarms to notify you when certain events occur or when your log storage reaches a certain capacity.

9. To test your CloudTrail configuration, perform some actions in your AWS account that will generate CloudTrail logs, such as creating a new EC2 instance or modifying a security group.

10. Navigate to your S3 bucket or CloudWatch Logs group to view your CloudTrail logs. You should see a log file for each event that was generated in your account.

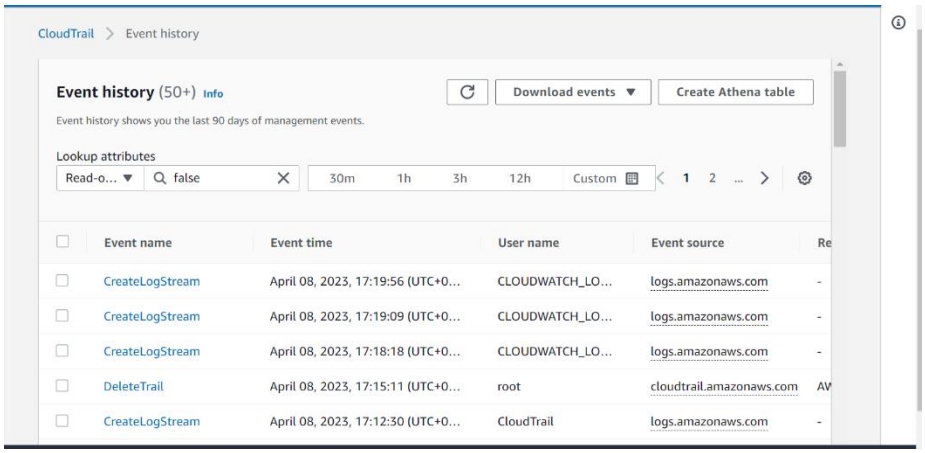


Fig.10.4: Viewing various logs of the AWS account using cloud trail service

Result:

AWS Cloud trail has been understood successfully and the logs were seen which showed the different activities done in the root account.