**Capture Moments - AWS-Powered Photographer Booking System**

**Project Description:**

Capture Moments is a cloud-native photographer booking platform that connects professional photographers with clients in real time. Built using scalable AWS infrastructure, the system enables users to explore photographer portfolios, filter by category (wedding, portrait, event, etc.), and book appointments securely and efficiently.

The system is built using **Amazon EC2** for hosting the application backend, ensuring reliable performance and full control over the server environment. **Amazon DynamoDB** is used database to manage user data, bookings, and photographer listings with high availability AWS **IAM** is implemented to manage secure access between services, define user roles (admin, photographer, client), and enforce fine-grained permission control.

**Scenario 1:**

Riya is planning a wedding and needs to book a professional photographer. She visits the Capture Moments platform, logs into her account, and begins her search. On the home page, she sees a search bar and filtering options such as: Photography Type (Wedding, Event, Portrait, etc.), Location, Availability Dates, Ratings etc.

Riya selects the **"**Wedding**"** category and filters photographers available in her city.

The platform displays a list of photographers in a card layout. Each photographer's card includes:

Profile Picture

Photographer Name (e.g., *Amit Lensman*)

Specialties (e.g., Wedding, Portrait)

Location (e.g., Hyderabad, India)

Average Rating (e.g., ★★★★☆)

Years of Experience (e.g., 5+ years)

Mini Portfolio Preview (3–4 sample images)

**Scenario 2:**

A client (like Riya) visits the platform and searches for wedding photographers. She sees Amit’s profile in the results. On the webpage, his photographer card includes:

Name: *Amit Lensman*

Specialties: Wedding, Portrait

Rating: 4.8/5

Location: Hyderabad

3–4 sample portfolio image

**AWS ARCHITECTURE**

This AWS-based architecture powers a scalable and secure web application using Amazon EC2 for hosting the backend, with a lightweight framework like Flask handling core logic. Application data is stored in Amazon DynamoDB, ensuring fast, reliable access, while user access is managed through AWS IAM for secure authentication and control. Real-time alerts and system and the communication and user engagement.

Entity

Relationship

(

ER)Diagram

:

**Pre**

**-**

**requisites:**

**1.**

.

**AWS**

**Account**

**Setup**

:

[AW](https://youtu.be/CjKhQoYeR4Q?si=ui8Bvk_M4FfVM-Dh)

[S](https://youtu.be/CjKhQoYeR4Q?si=ui8Bvk_M4FfVM-Dh)

[t](https://youtu.be/CjKhQoYeR4Q?si=ui8Bvk_M4FfVM-Dh)

[Accoun](https://youtu.be/CjKhQoYeR4Q?si=ui8Bvk_M4FfVM-Dh)

[Setu](https://youtu.be/CjKhQoYeR4Q?si=ui8Bvk_M4FfVM-Dh)

[p](https://youtu.be/CjKhQoYeR4Q?si=ui8Bvk_M4FfVM-Dh)

**2.**

**Understanding**

**IAM**

:

[IA](https://youtu.be/gsgdAyGhV0o?si=3qg-bULgkD4LXNvR)

[M](https://youtu.be/gsgdAyGhV0o?si=3qg-bULgkD4LXNvR)

[Overvie](https://youtu.be/gsgdAyGhV0o?si=3qg-bULgkD4LXNvR)

[w](https://youtu.be/gsgdAyGhV0o?si=3qg-bULgkD4LXNvR)

**3.**

**Amazon**

**EC2**

**Basics**

[:](https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz)

[EC](https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz)

[2](https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz)

[Tutoria](https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz)

[l](https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz)

**4.**

**DynamoDB**

**Basics**

:

[DynamoD](https://docs.aws.amazon.com/dynamodb)

[B](https://docs.aws.amazon.com/dynamodb)

[Introductio](https://docs.aws.amazon.com/dynamodb)

[n](https://docs.aws.amazon.com/dynamodb)

**5.**

**SNS**

**Overview**

:

[SN](https://docs.aws.amazon.com/sns)

[S](https://docs.aws.amazon.com/sns)

[Documentatio](https://docs.aws.amazon.com/sns)

[n](https://docs.aws.amazon.com/sns)

**6.**

**Git**

**Version**

**Control**

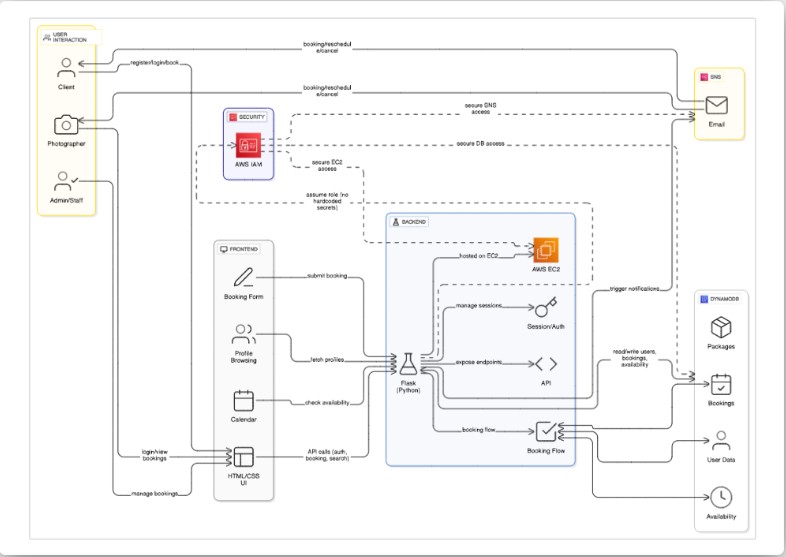
[:](https://git-scm.com/doc)

[Gi](https://git-scm.com/doc)

[t](https://git-scm.com/doc)

[Documentatio](https://git-scm.com/doc)

[n](https://git-scm.com/doc)



**Project Workflow:**

# AWS Account Setup and Login

**Activity 1.1:** Set up an AWS account if not already done**.**

**Activity 1.2:** Log in to the AWS Management Console

# DynamoDB Database Creation and Setup

**Activity 2.1**: Create a DynamoDB Table.

**Activity 2.2**: Configure Attributes for User Data and Book Requests.

# IAM Role Setup

**Activity 3.1**: Create IAM Role

**Activity 3.2**: Attach Policies

# EC2 Instance Setup

**Activity 4.1**: Launch **6** an EC2 instance to host the Flask application.

**Activity 4.2**: Configure security groups for HTTP, and SSH access.

# 5.Deployment on EC2

**Activity 5.1**: install

dependencies

# Milestone 6: Testing and Deployment

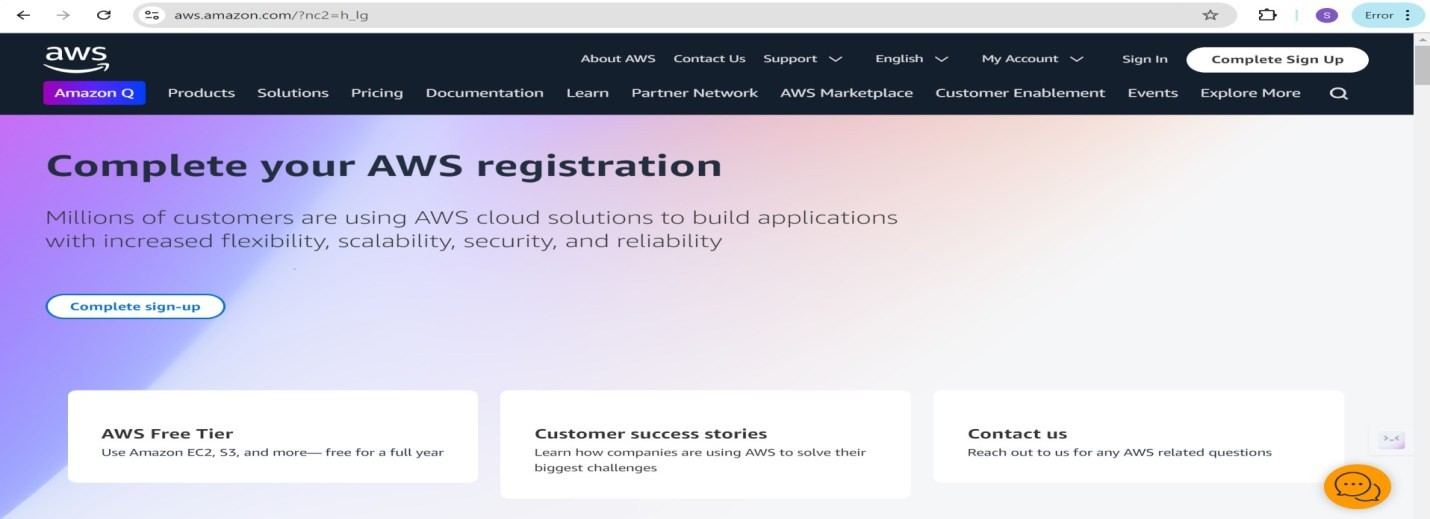
**Activity 6.1** Conduct functional testing to verify booking a photographer and view photographers:

**Milestone 1: AWS Account Setup and Login**

**Activity 1.1: Set up an AWS account if not already done.**

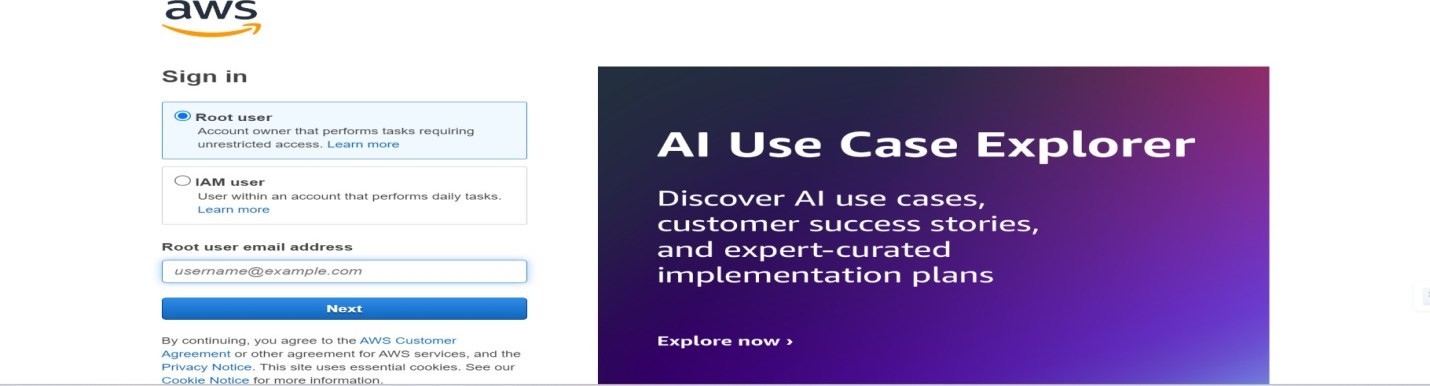
○ Sign up for an AWS account and configure billing settings.

○



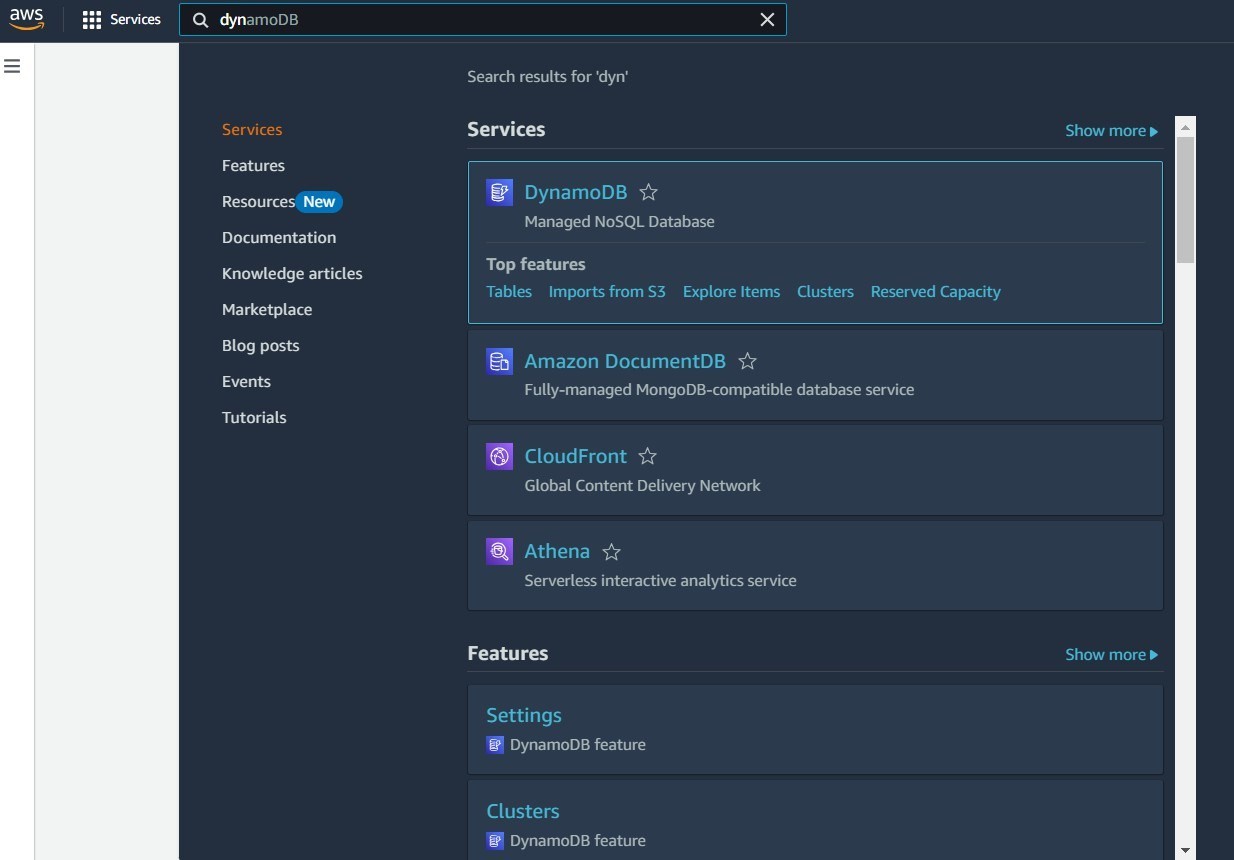
## ● Activity 1.2: Log in to the AWS Management Console

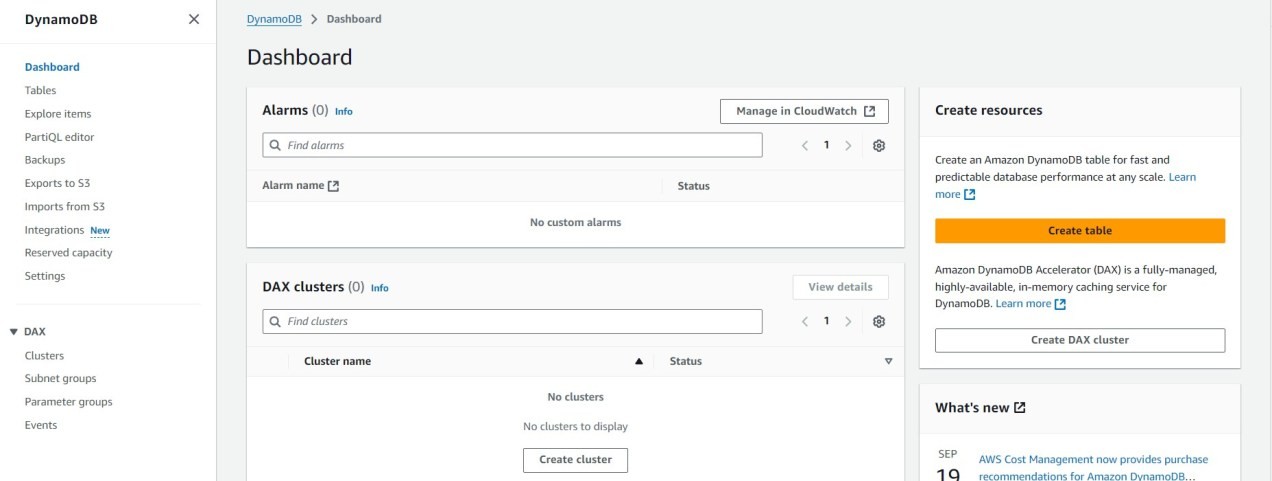
○ After setting up your account, log in to the [AWS Management Console](https://aws.amazon.com/console/)

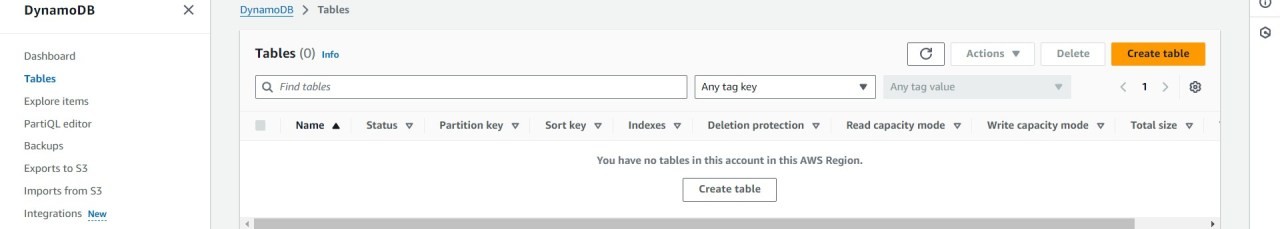


**Milestone 2: DynamoDB Database Creation and Setup**

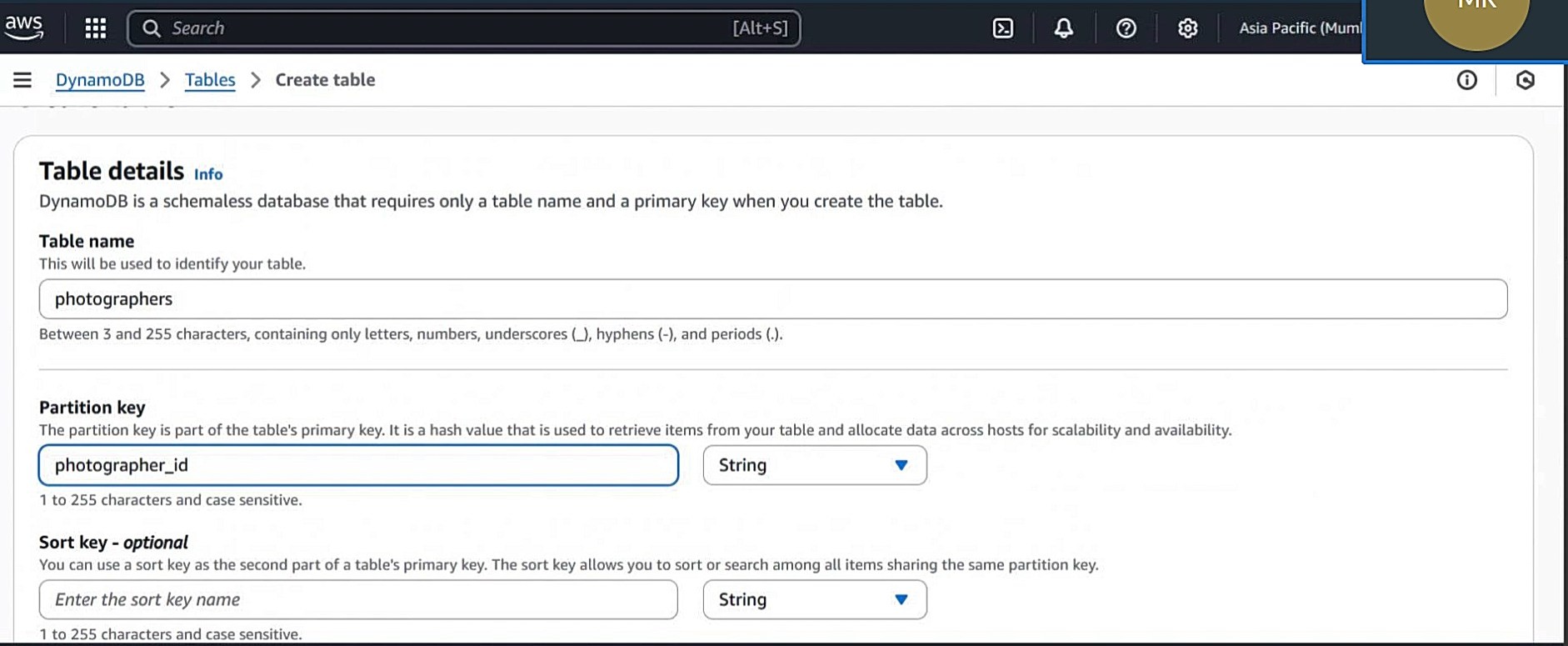
* **Activity 2.1: Create a DynamoDB Table.** 
  1. In the AWS Console, navigate to DynamoDB and click on create tables.







* **Activity 2.2: Configure Attributes for User Data and Book Request.** 
  1. Create photographers table with partition key ”photographer\_id ”with type String and click on create tables.



o

Create

booking table

with

partition

key”

booking\_id

“with

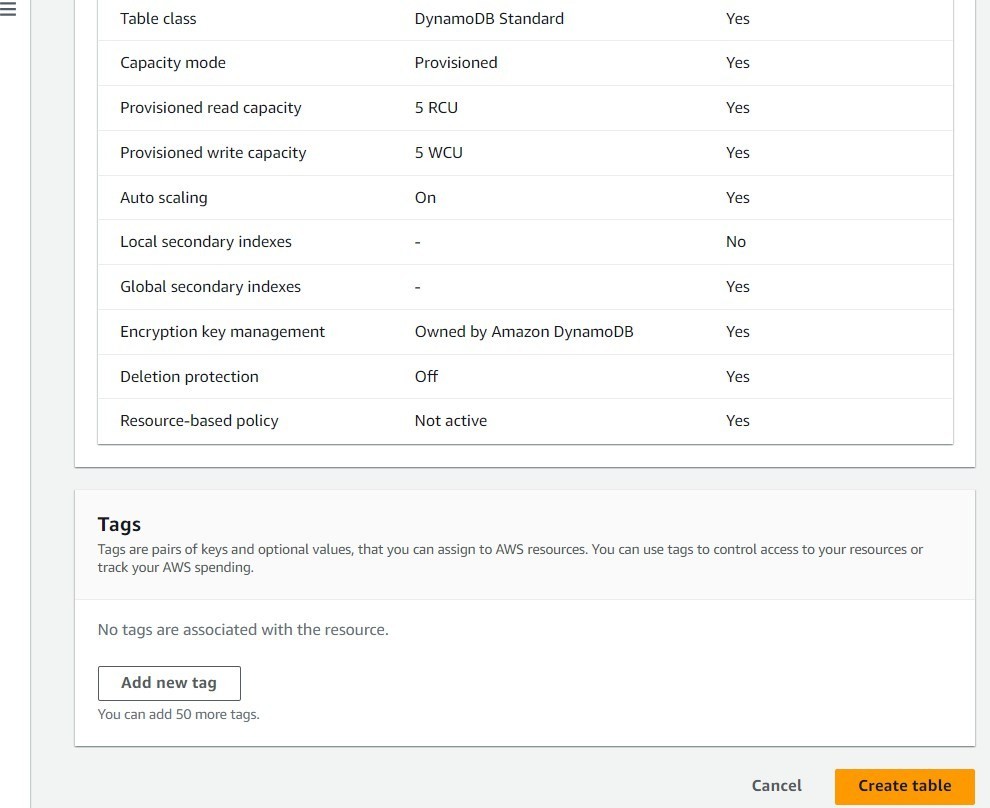
type

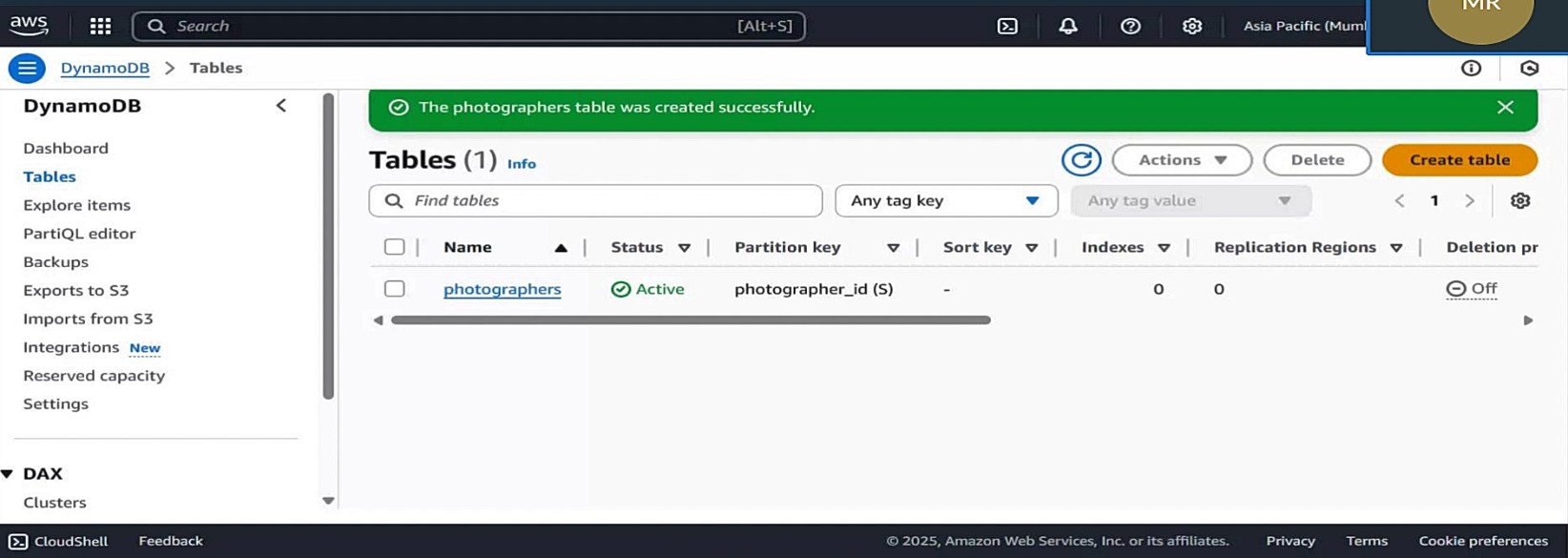
String

and

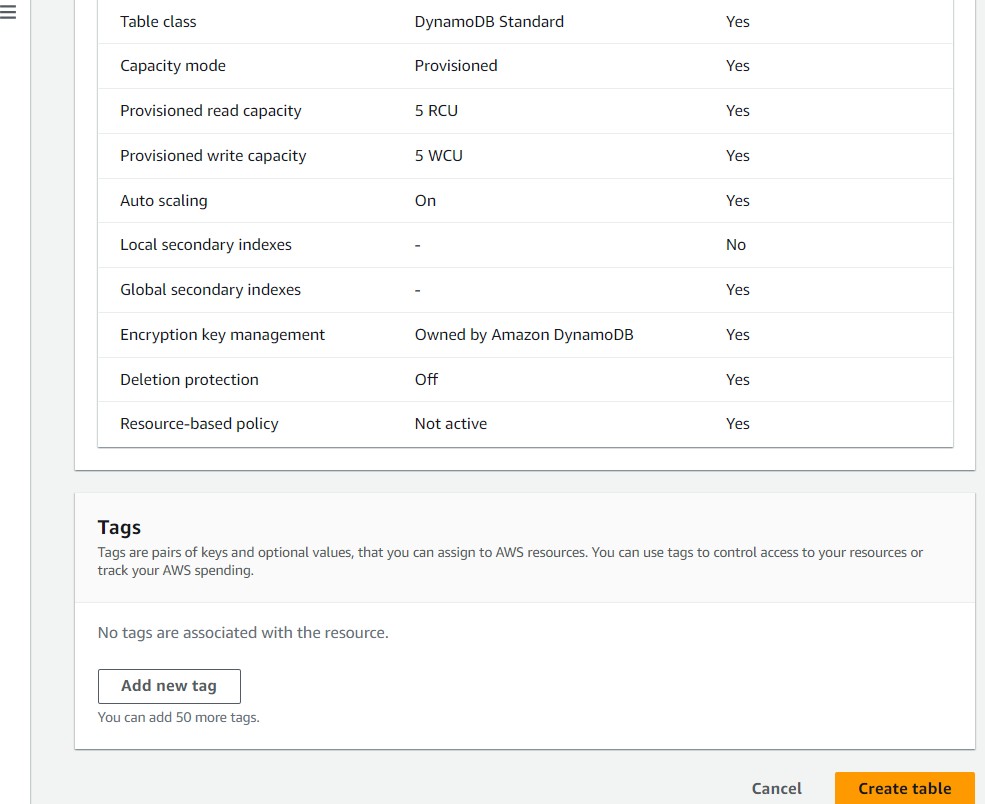
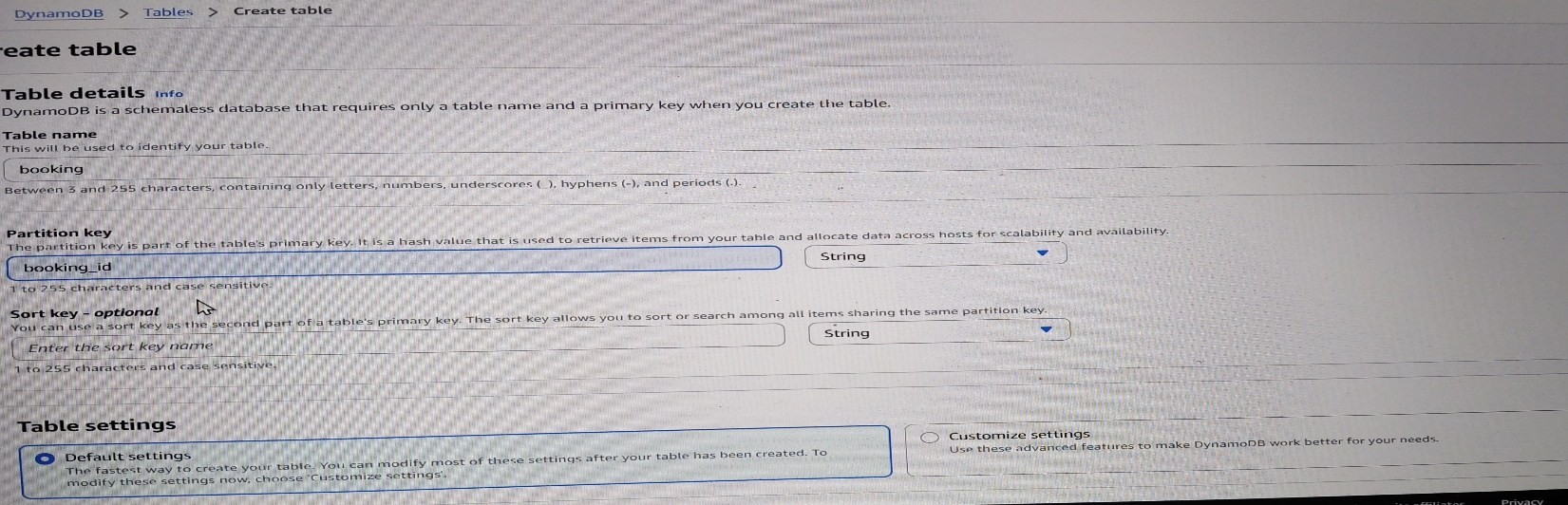
click

on create





○ Create booking table with partition key” booking \_id “with type String and click on create tables.



○

Create

booking table

with

partition

key” booking \_id

“with

type

String

and

click

on create tables.

○

Create

booking table

with

partition

key” booking \_id

“with

type

String

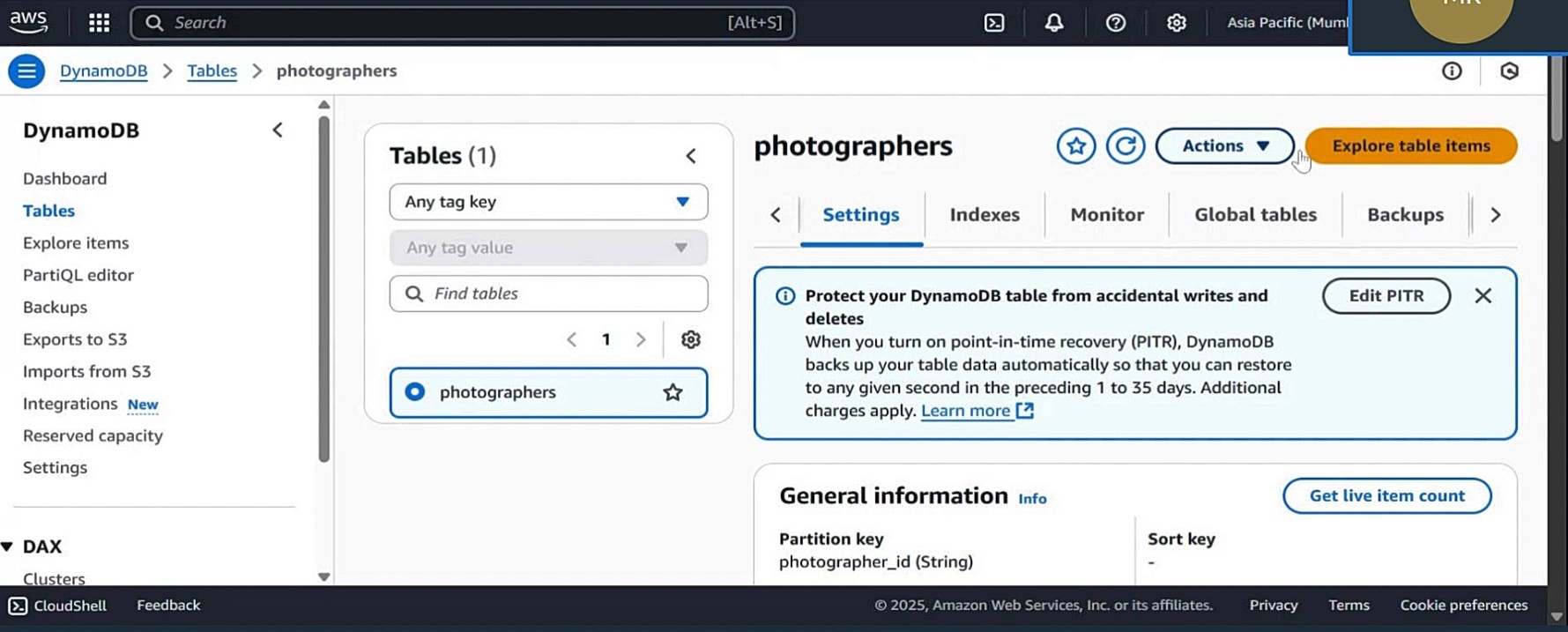
and

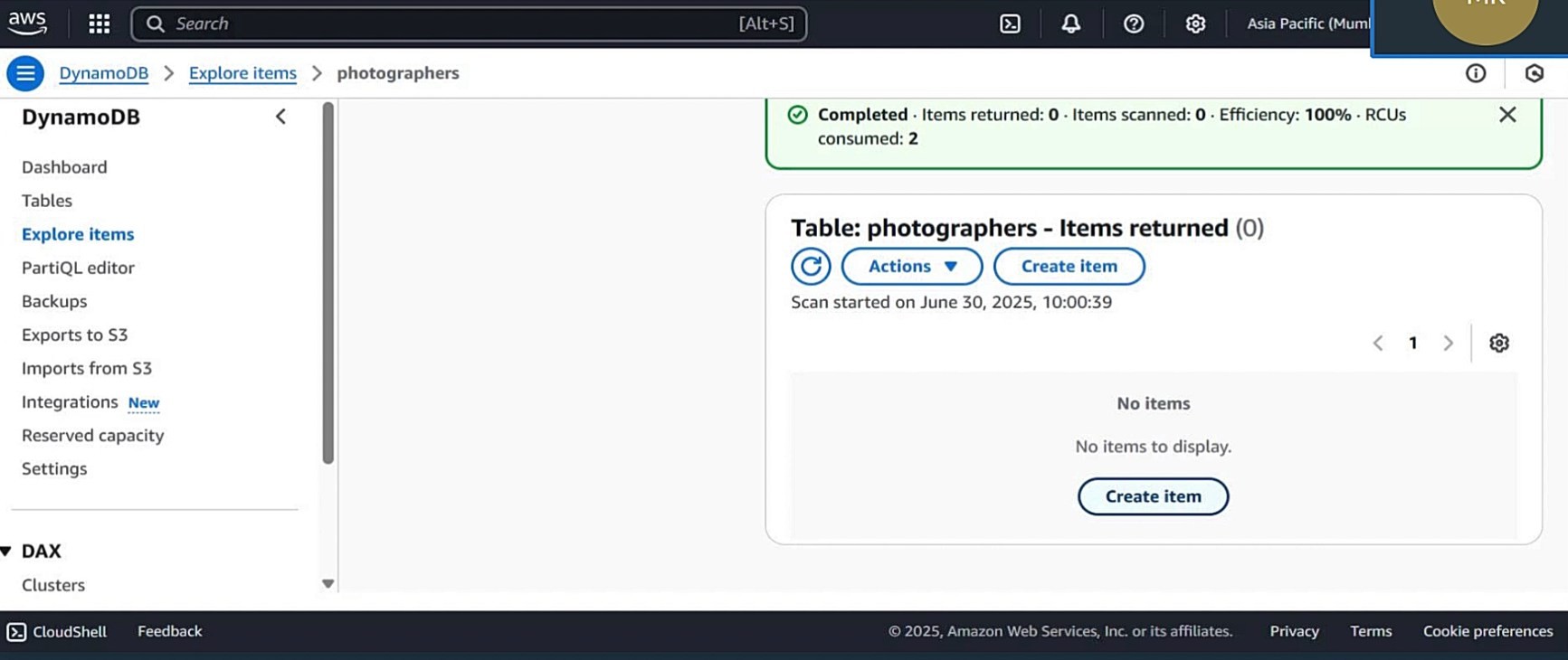
click

on create tables.

cc

ddskffxdvxkvjkfdvkflkf





○ Create items with data and click on create items

○

Click

on

**Create**

**Topic**

and

choose

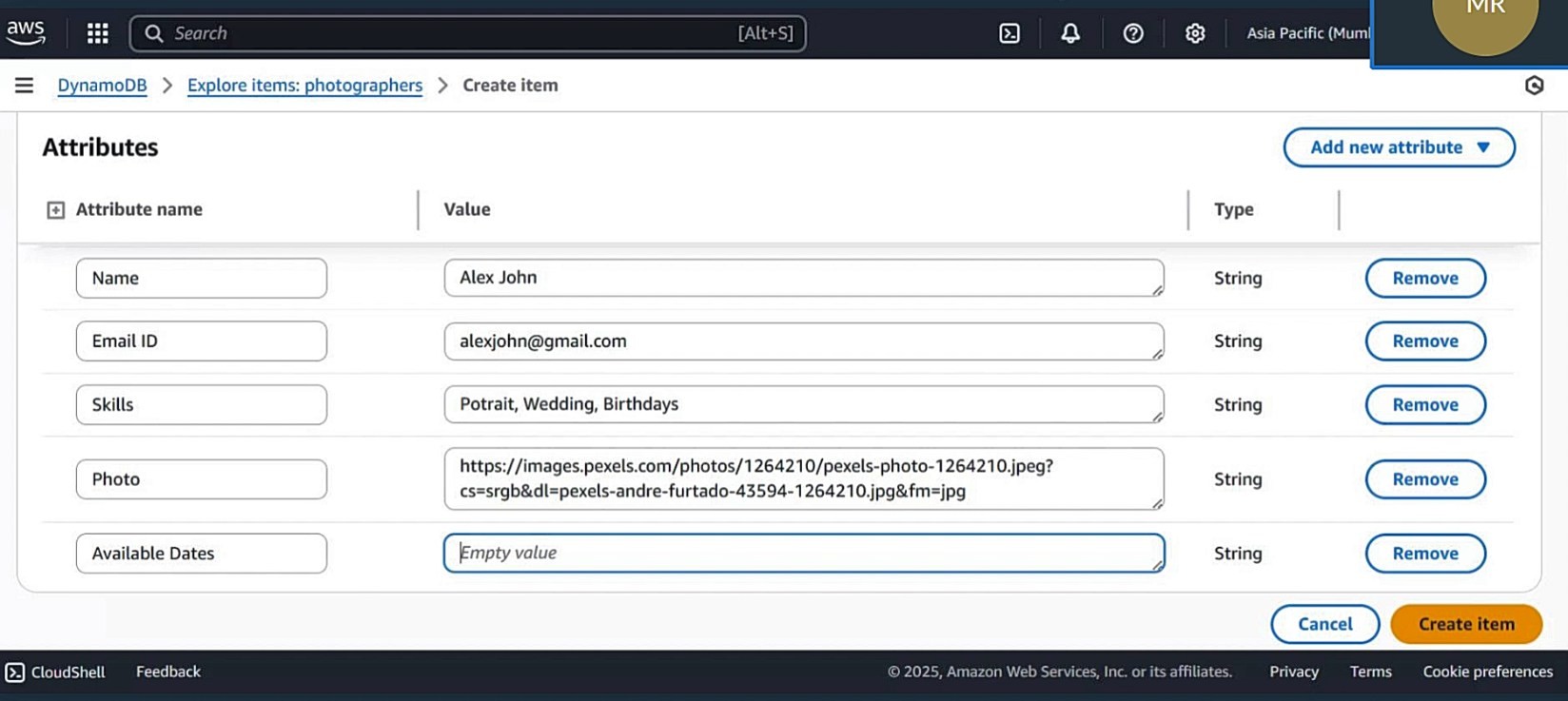
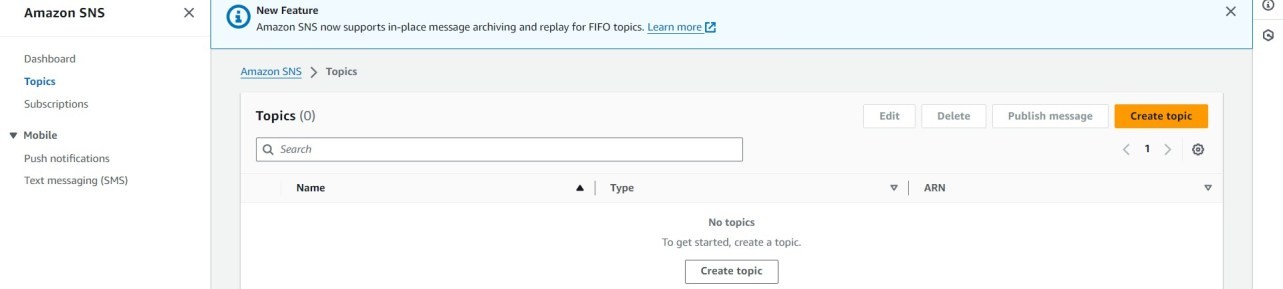
a

name

for

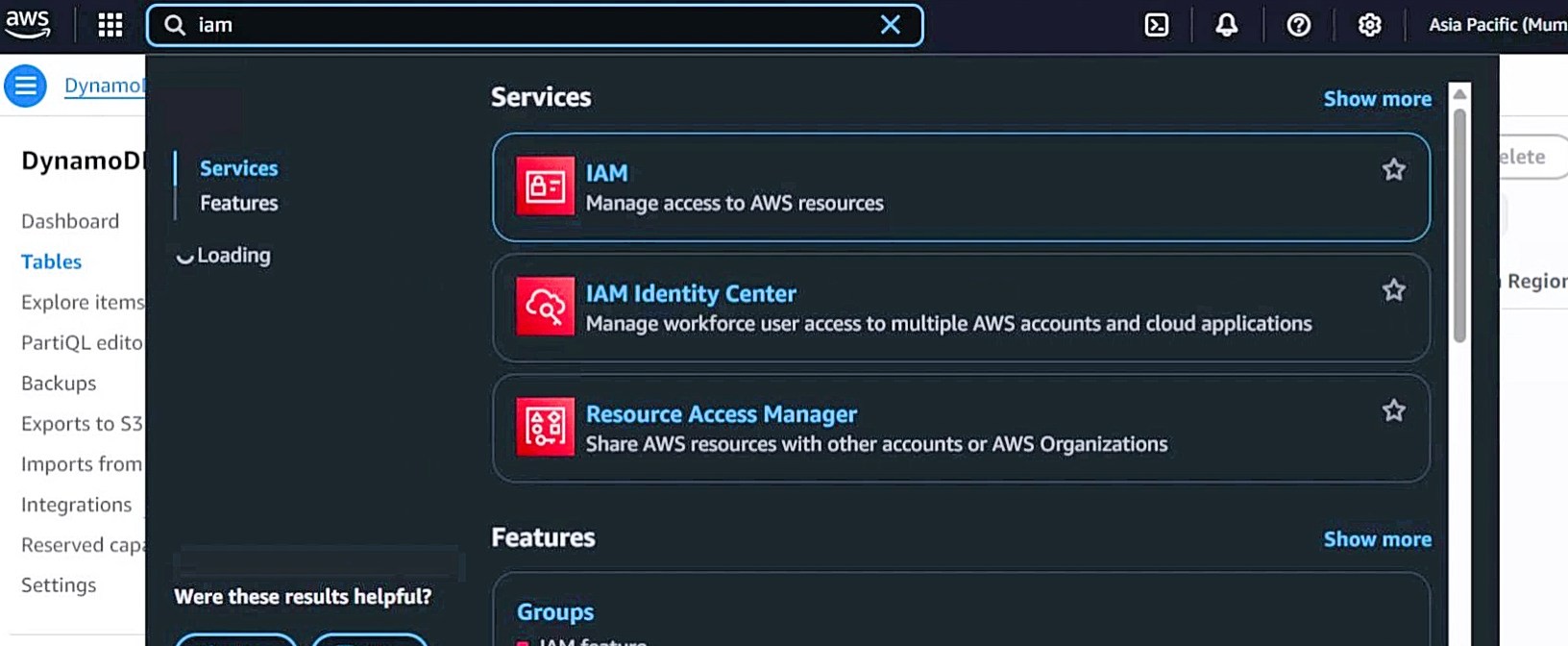
the

topic.



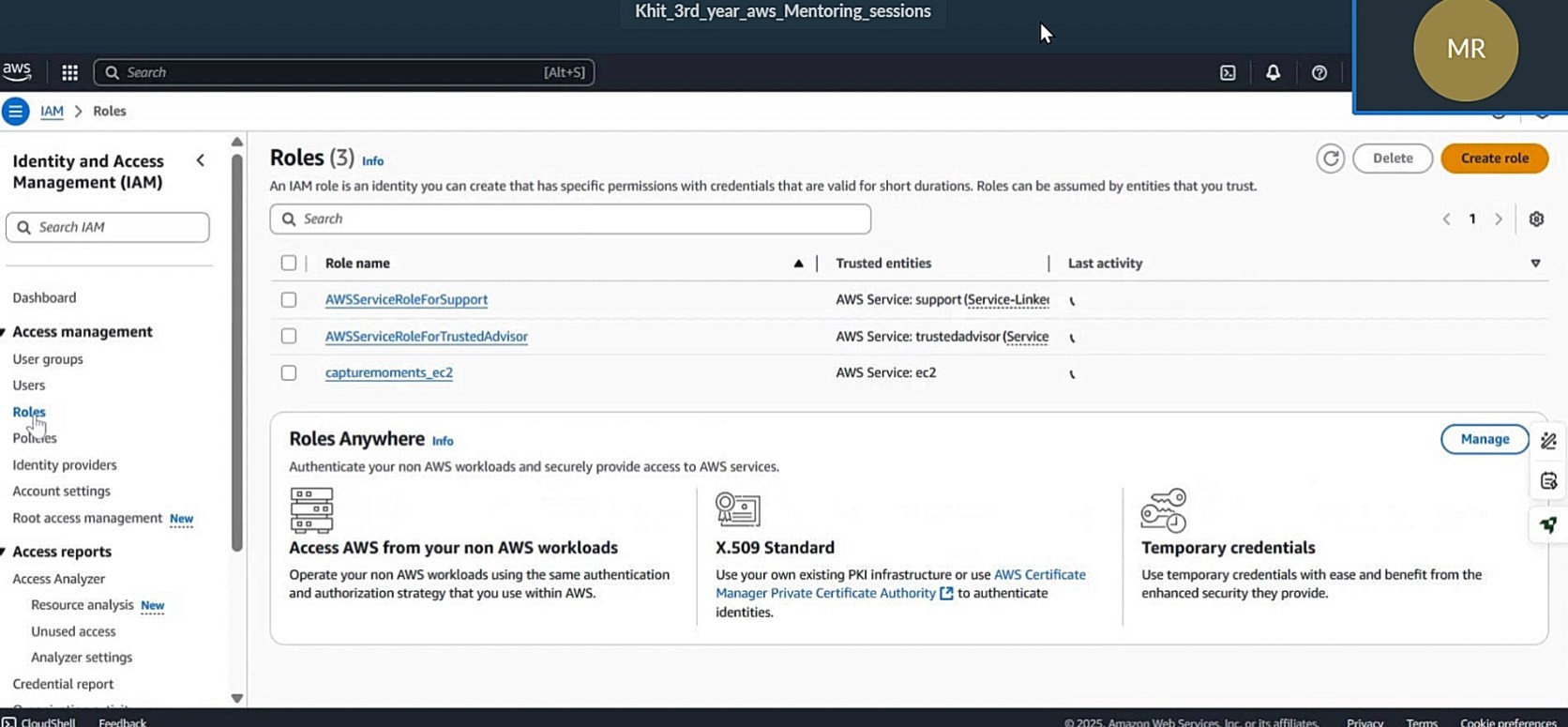
**Milestone 3: IAM Role Setup**

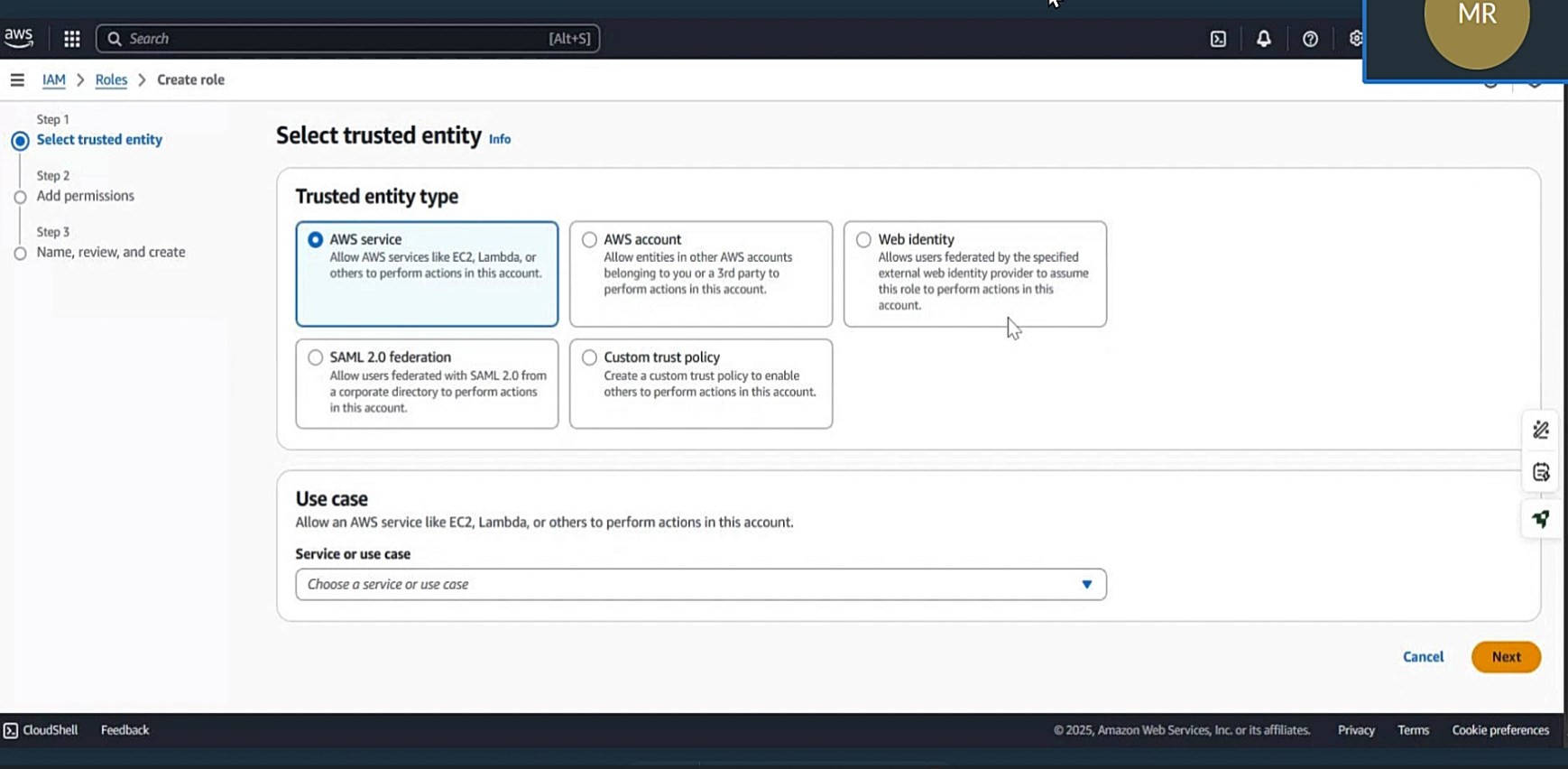
**Activity 3.1**: Create IAM Role o go to search bar and search the iam and click on the iam below given figure



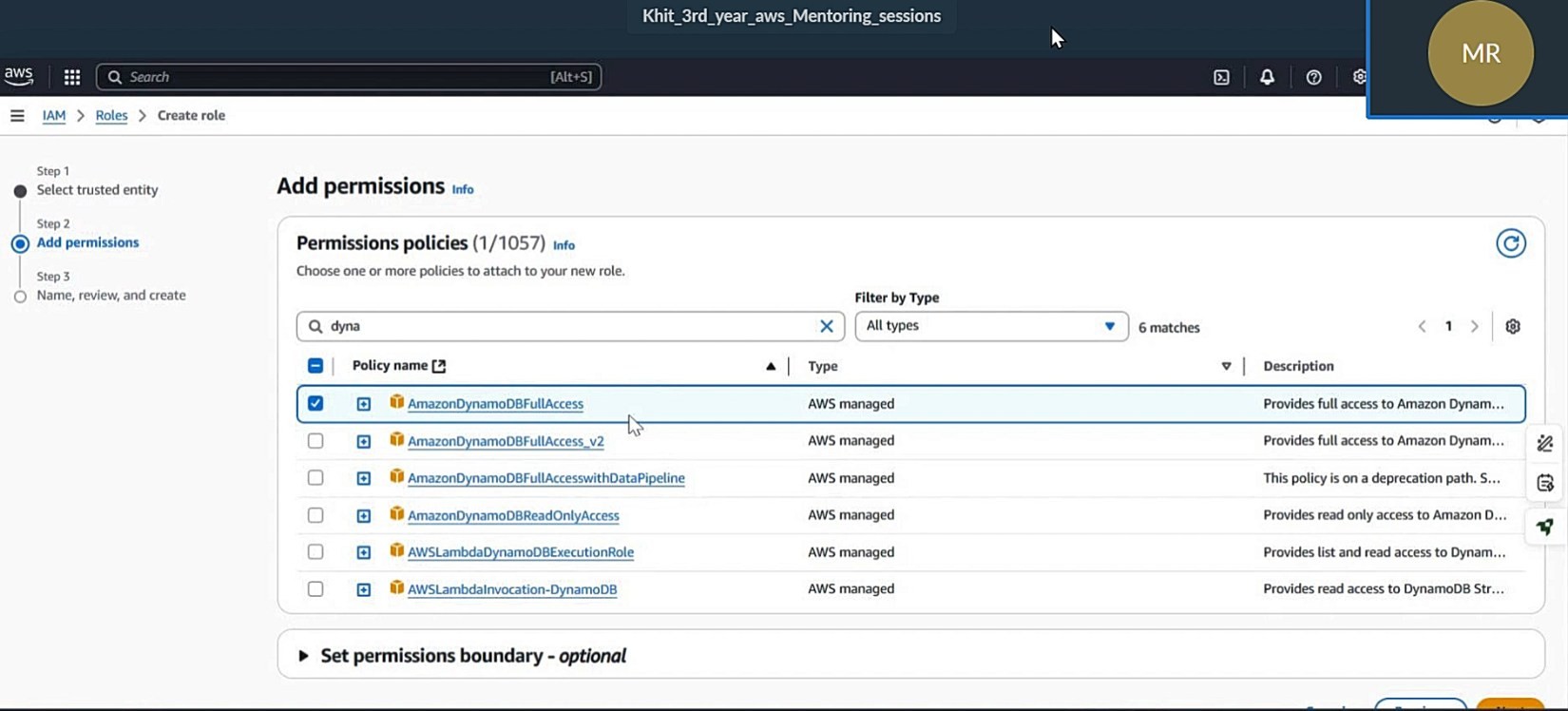
**Activity 3.2**: Attach Policies

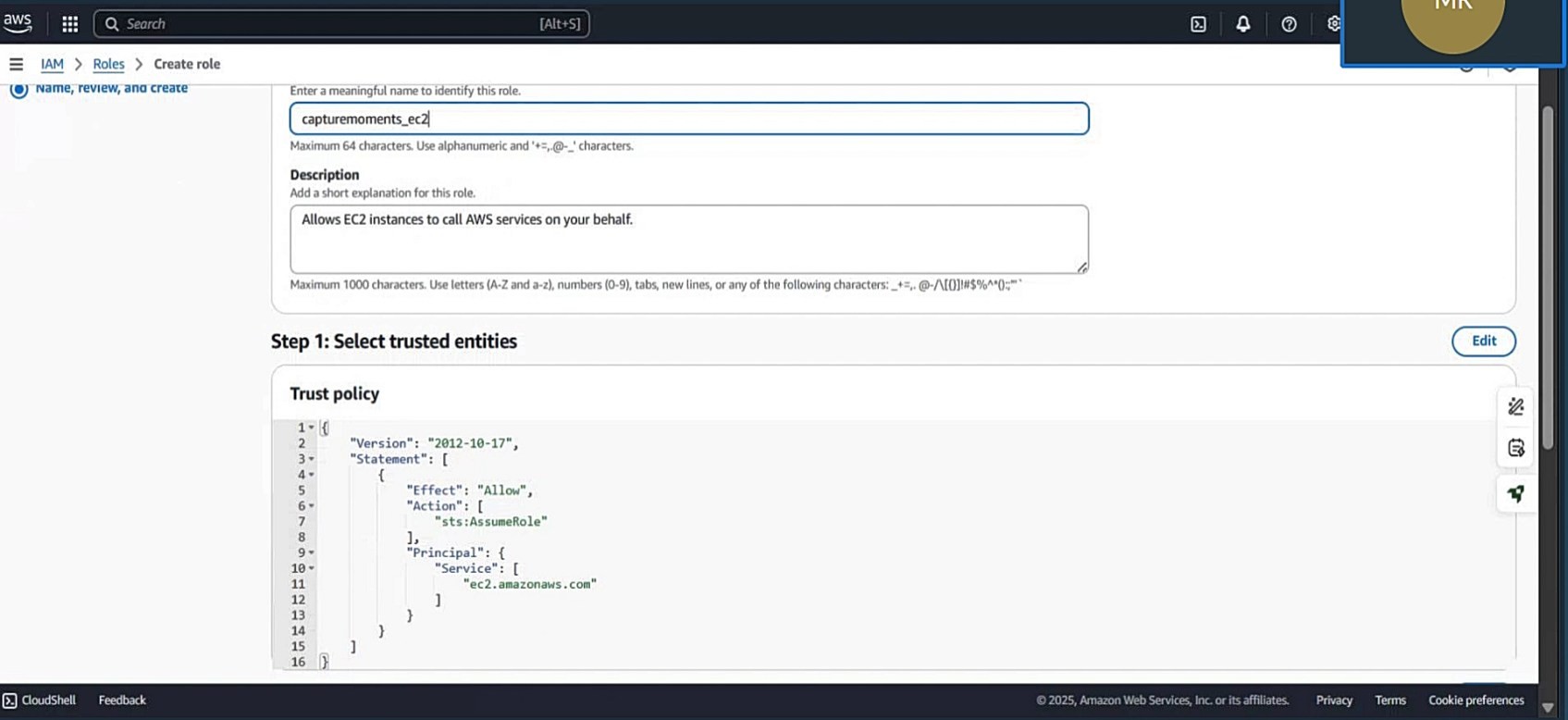
* Click on the iam and select the roles and create the roles with step by step with given figure





* In this step we select the amazondynmadbfullaccess and go to next steps





* successfully created iam role

**sddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd**

o

**cerr**

successfully created iam role

**1.**

**EC2**

**Instance**

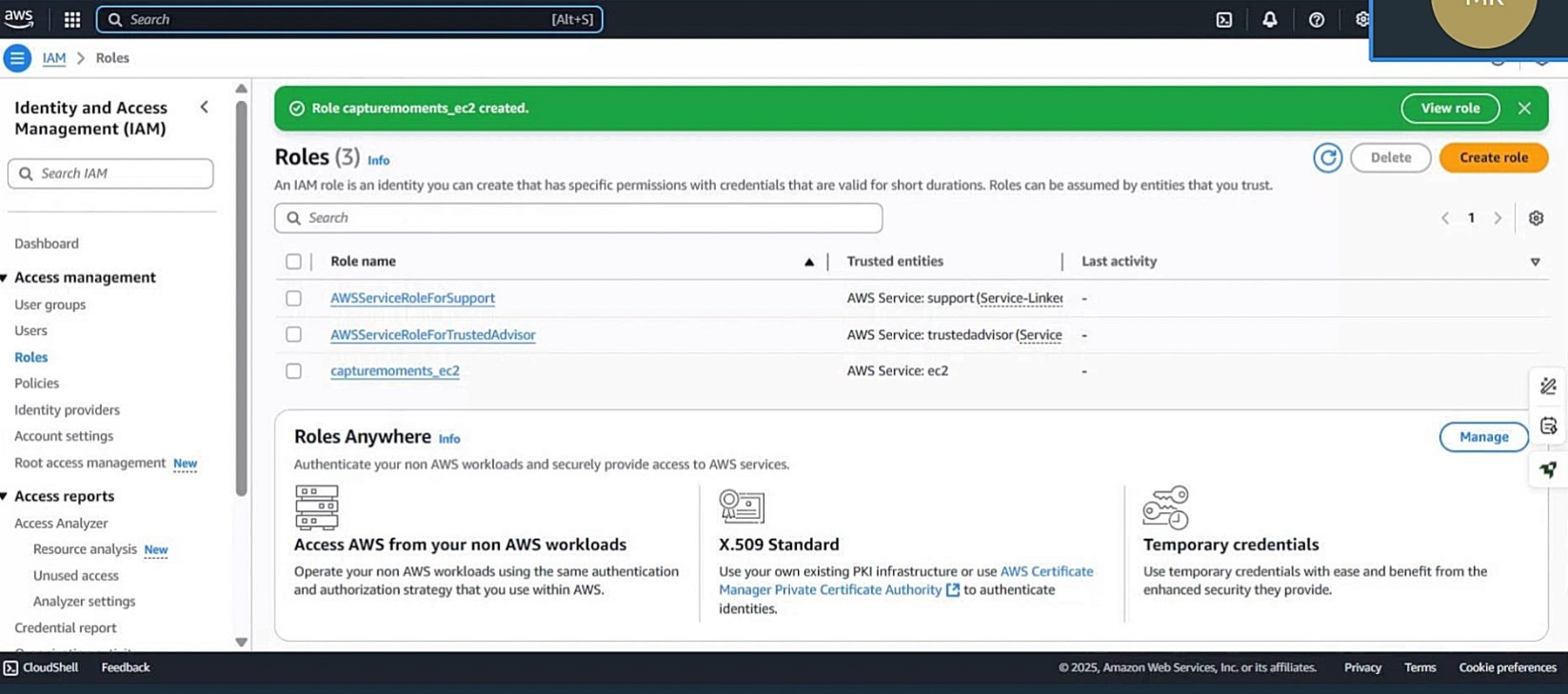
**Setup**

**fr**

**;;;**

**fff**

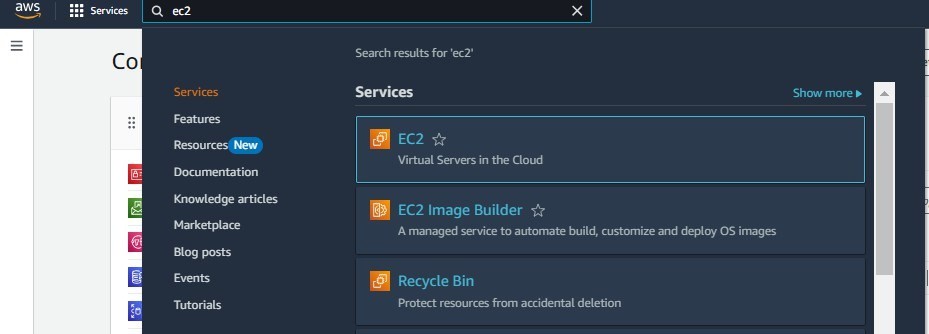
**Milestone 2:**



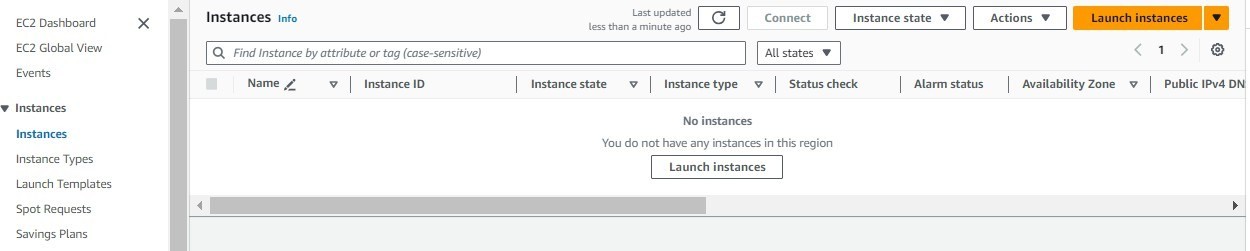
**Milestone 4: EC2 Instance Setup**

**Activity 4.1**: Launch **6** an EC2 instance to host the Flask application.

* go to search bar and search the iam and click on the iam below given figure



* Launch the instance and create the instance.



●

Create

and

download

the

key

pair

for

Server

access.

○

Suc

cessfully

done

with

the

SNS

mail

subscription

and

setup,

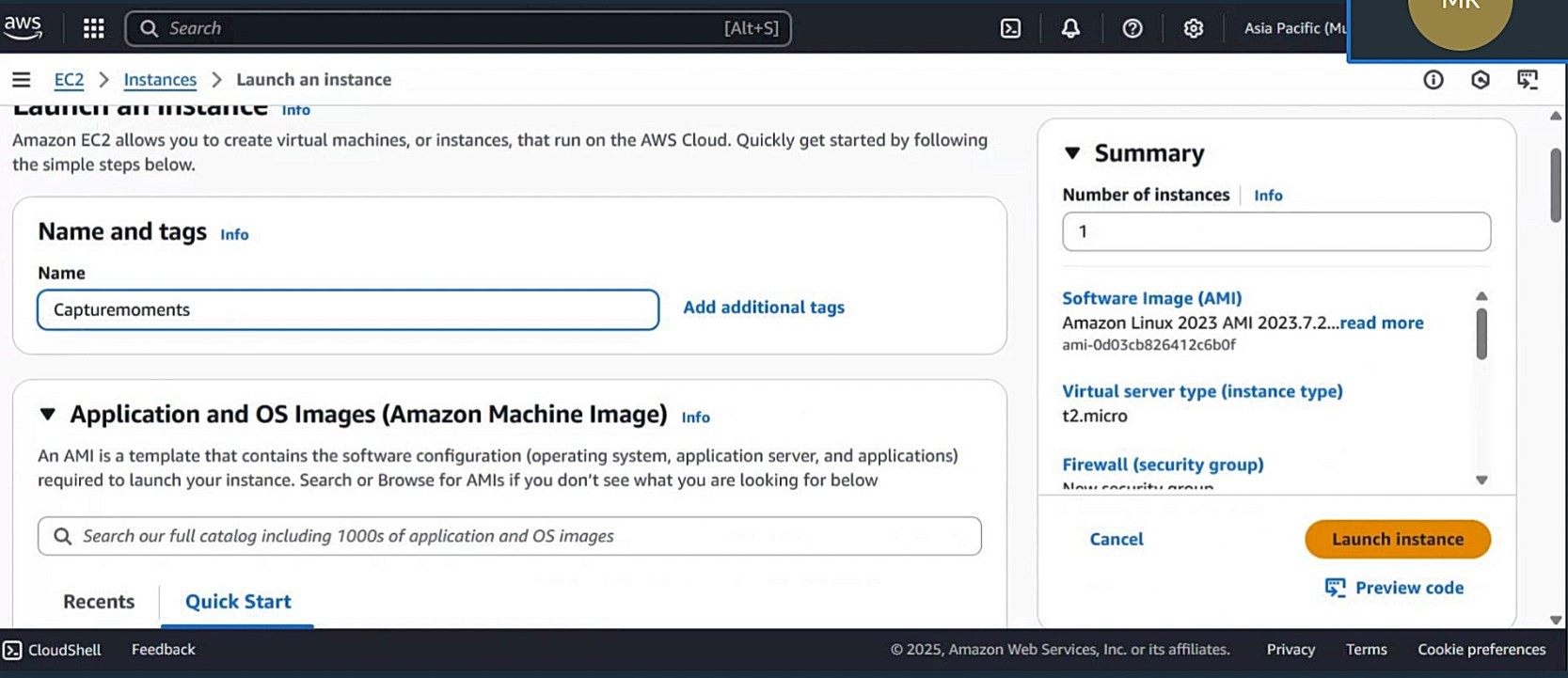
now

store

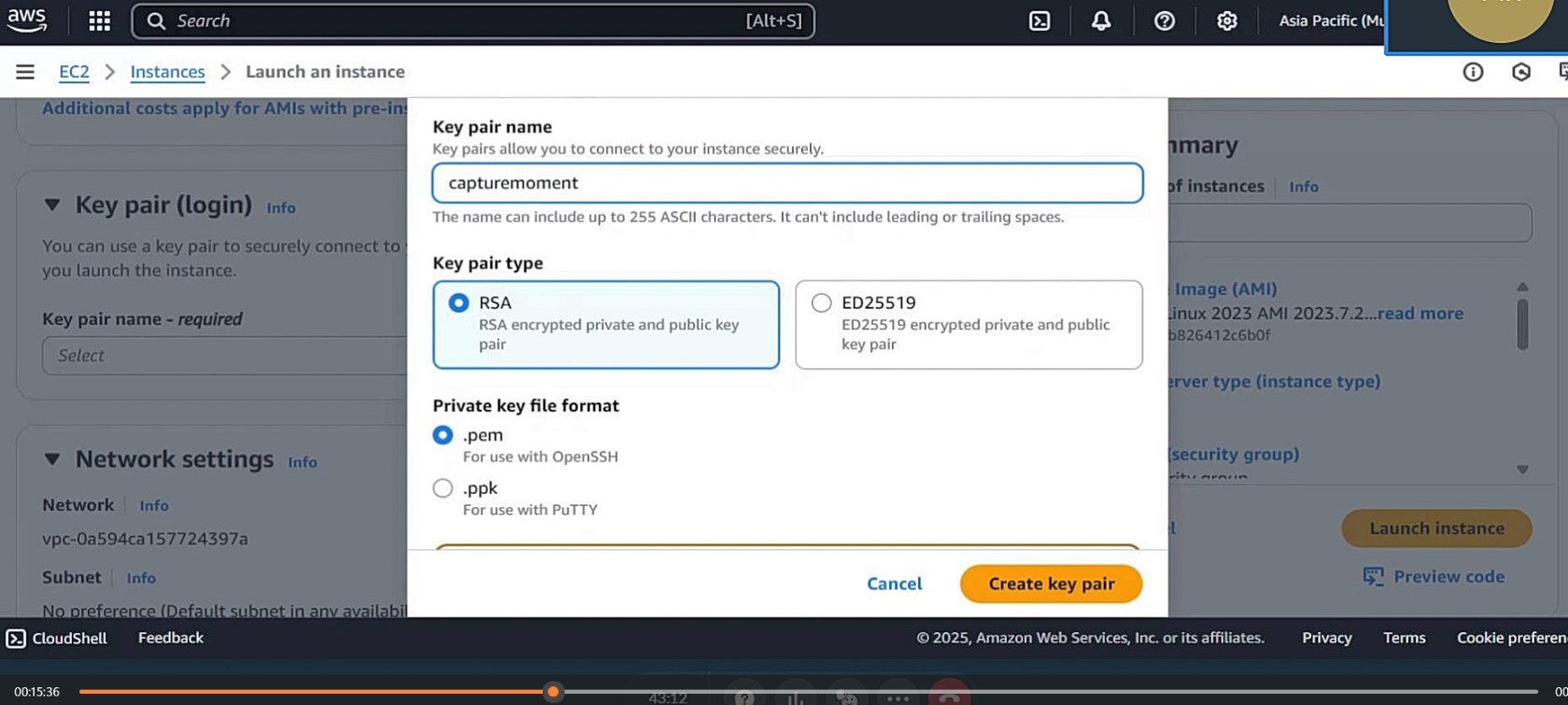
the

ARN

link.

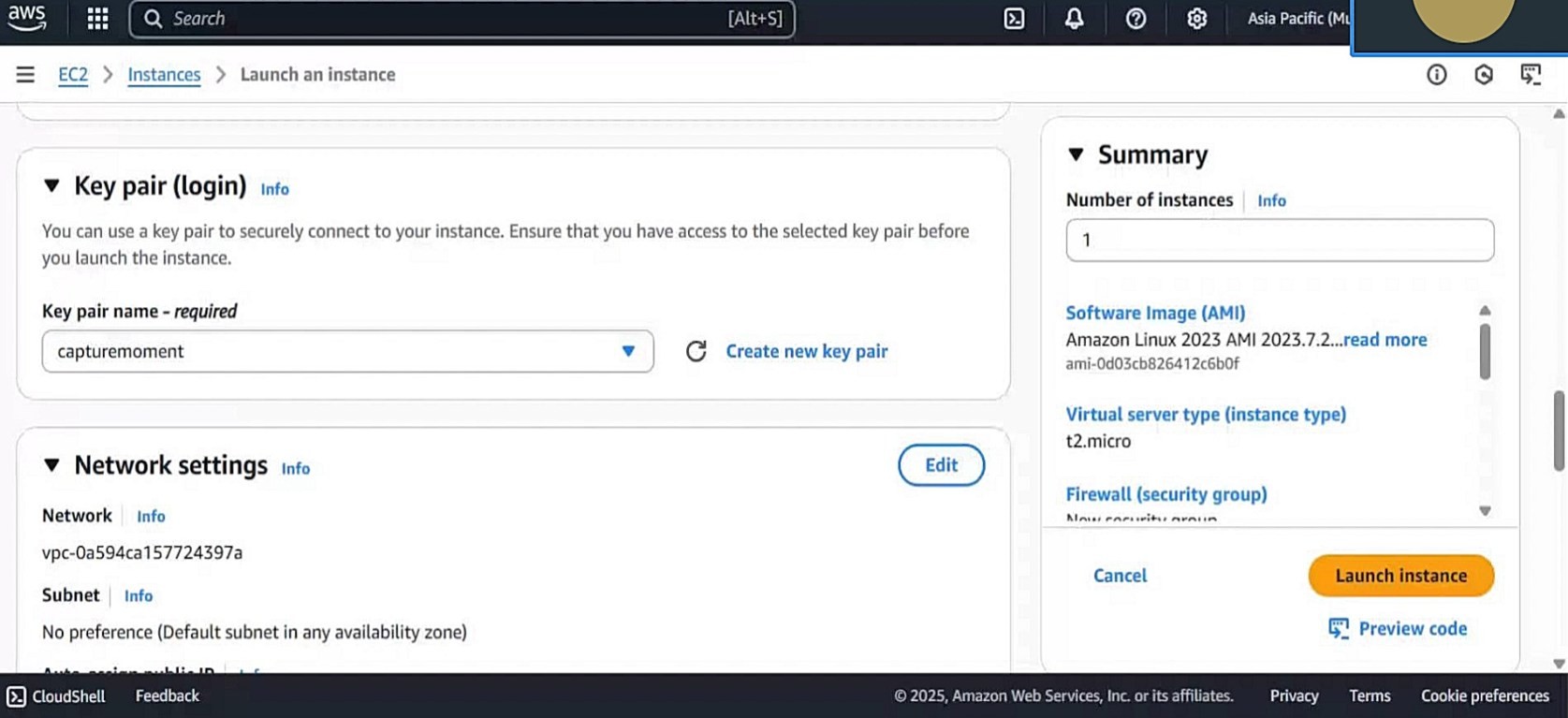


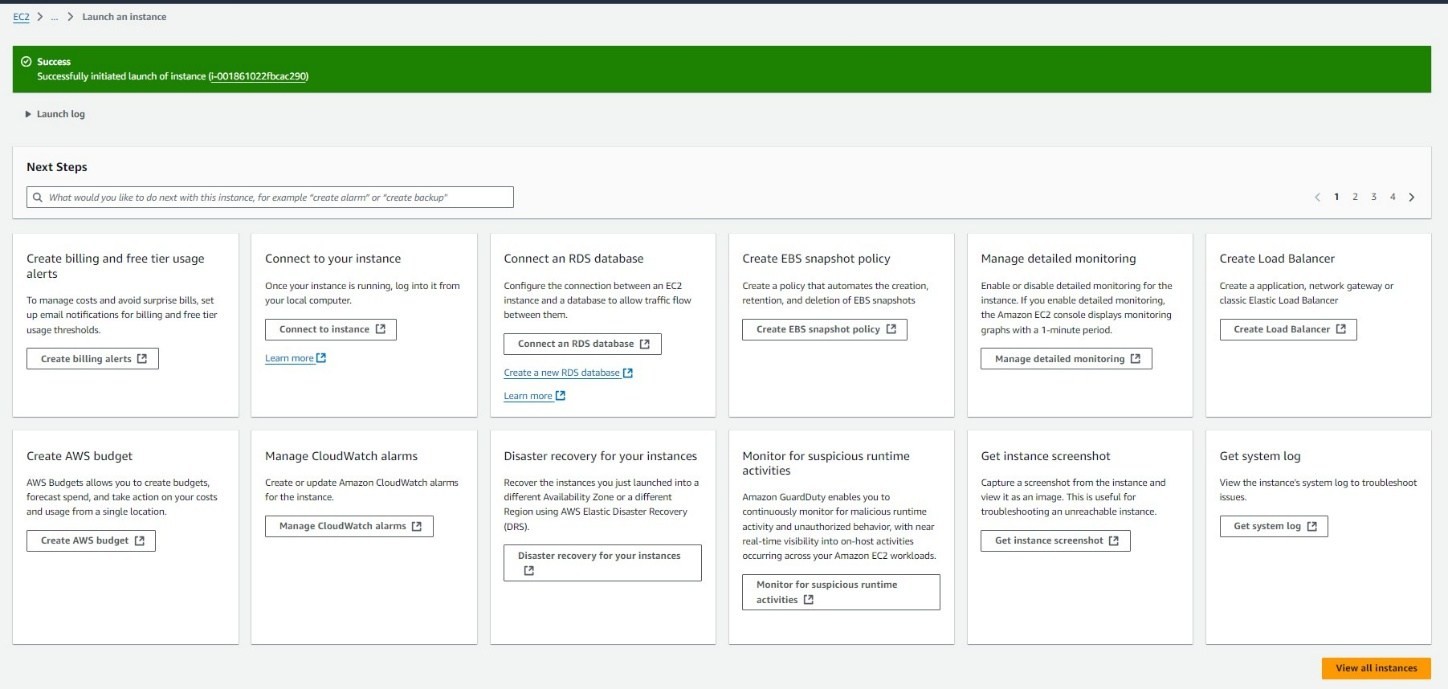
* Create the key pair name and download the key pair.



**Activity 4.2**: Configure security groups for HTTP, and SSH access.

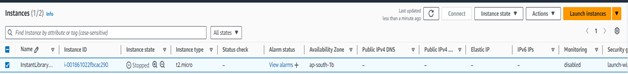
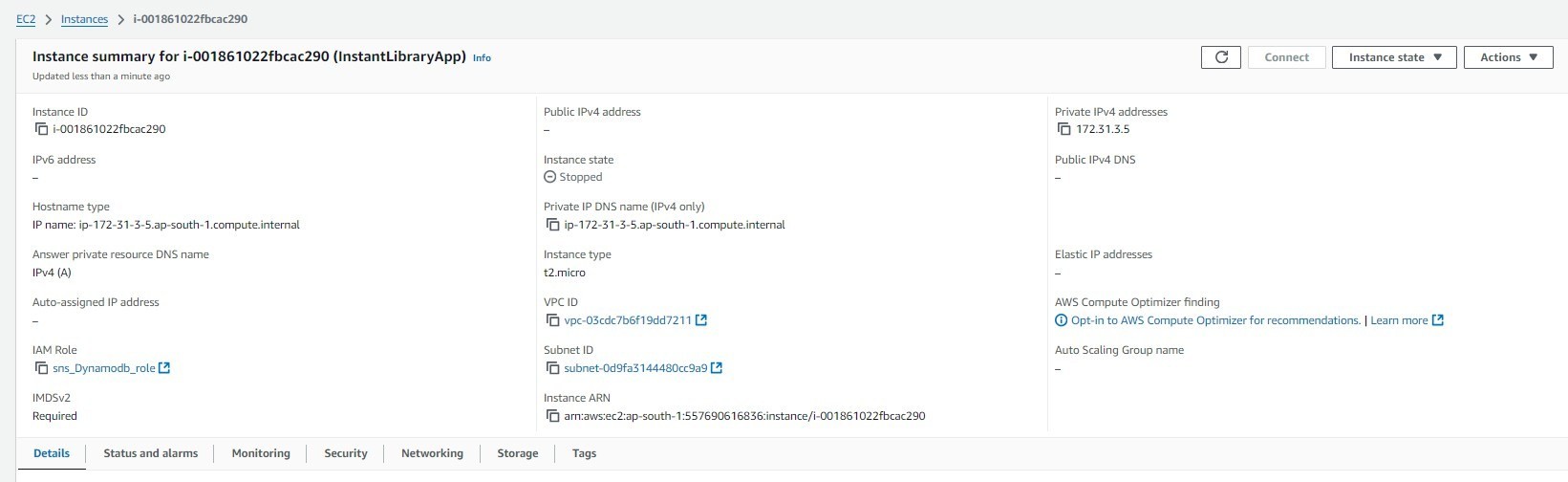
* Go to network settings and click on the edit and to down and click the network security rule.

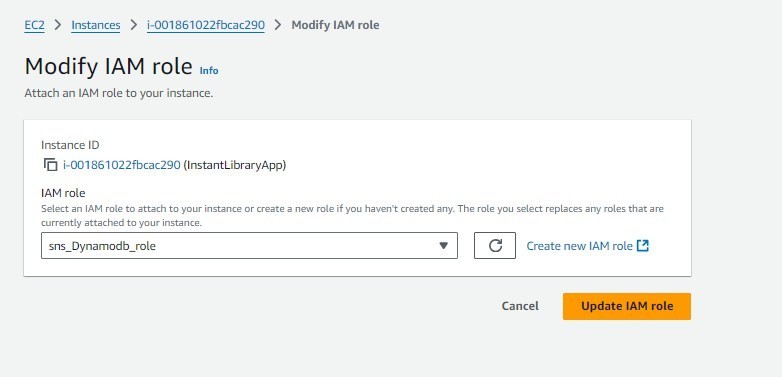


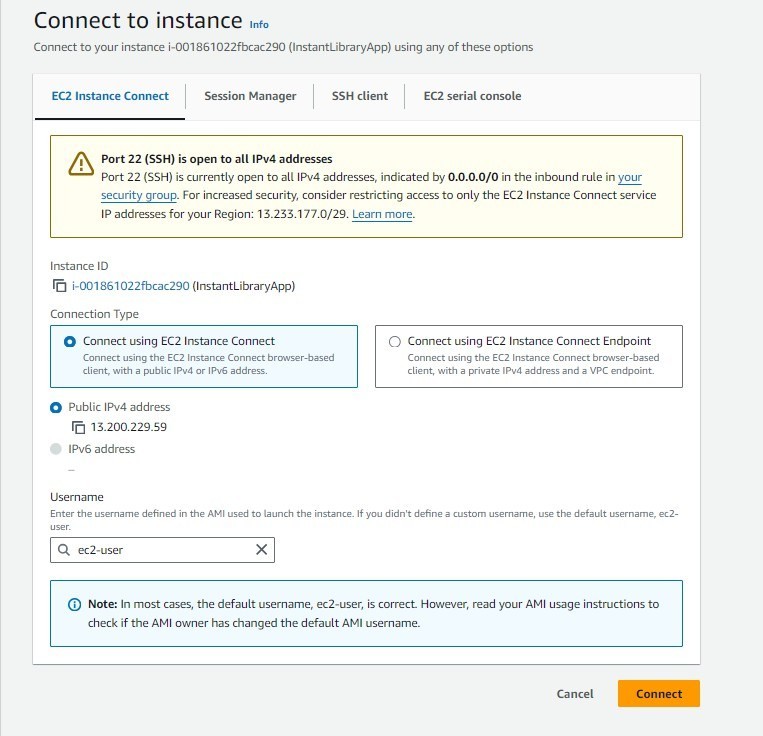


**Description:**

To connect to EC2 using EC2 Instance Connect, start by ensuring that an IAM role is attached to your EC2 instance. You can do this by selecting your instance, clicking on Actions, then navigating to Security and selecting Modify IAM Role to attach the appropriate role. After the IAM role is connected, navigate to the EC2 section in the AWS Management Console. Select the EC2 instance you wish to connect to. At the top of the EC2 Dashboard, click the Connect button. From the connection methods presented, choose EC2 Instance Connect. Finally, click Connect again, and a new browser-based terminal will open, allowing you to access your EC2 instance directly from your browser.

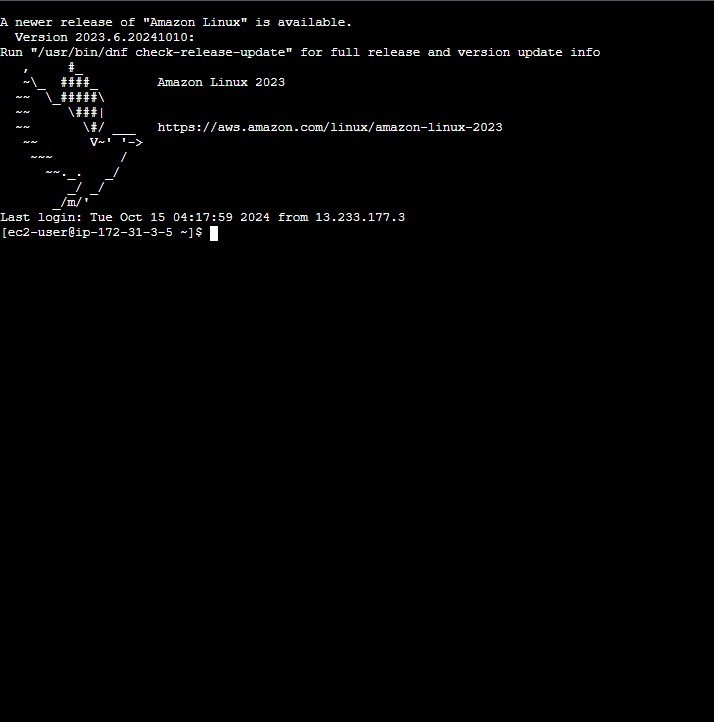






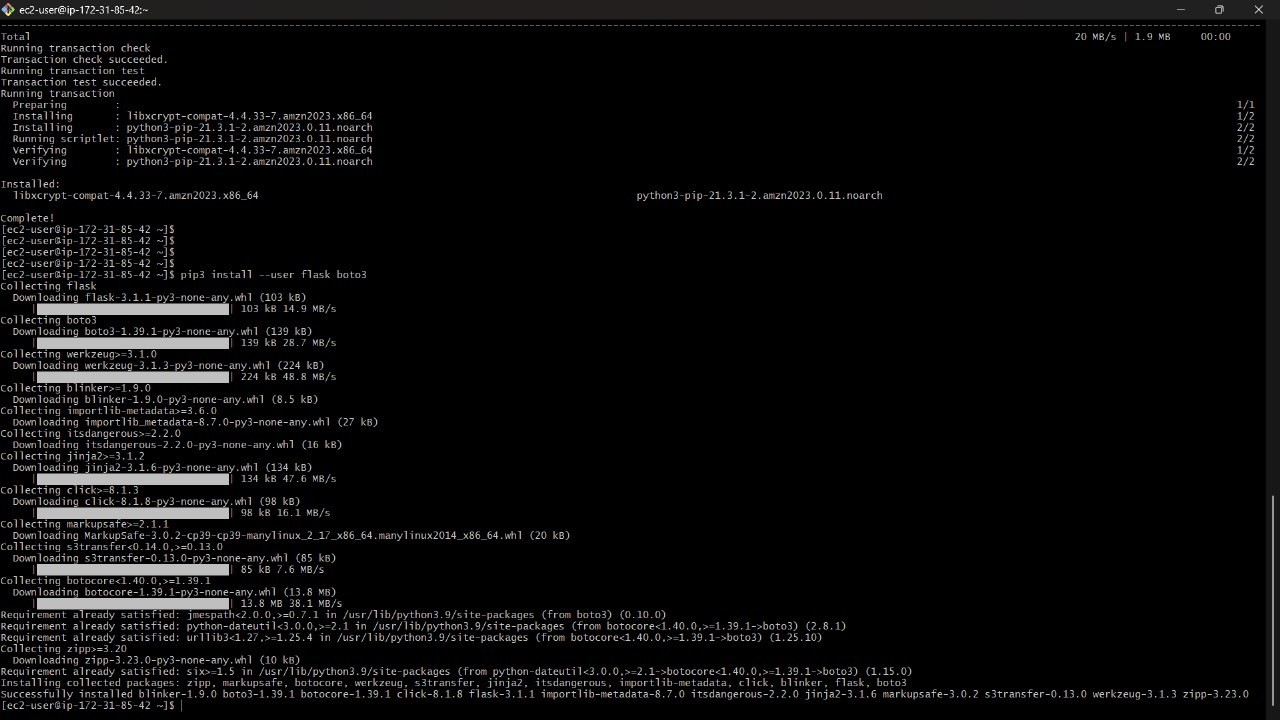
**Milestone 5: Deployment on EC2**

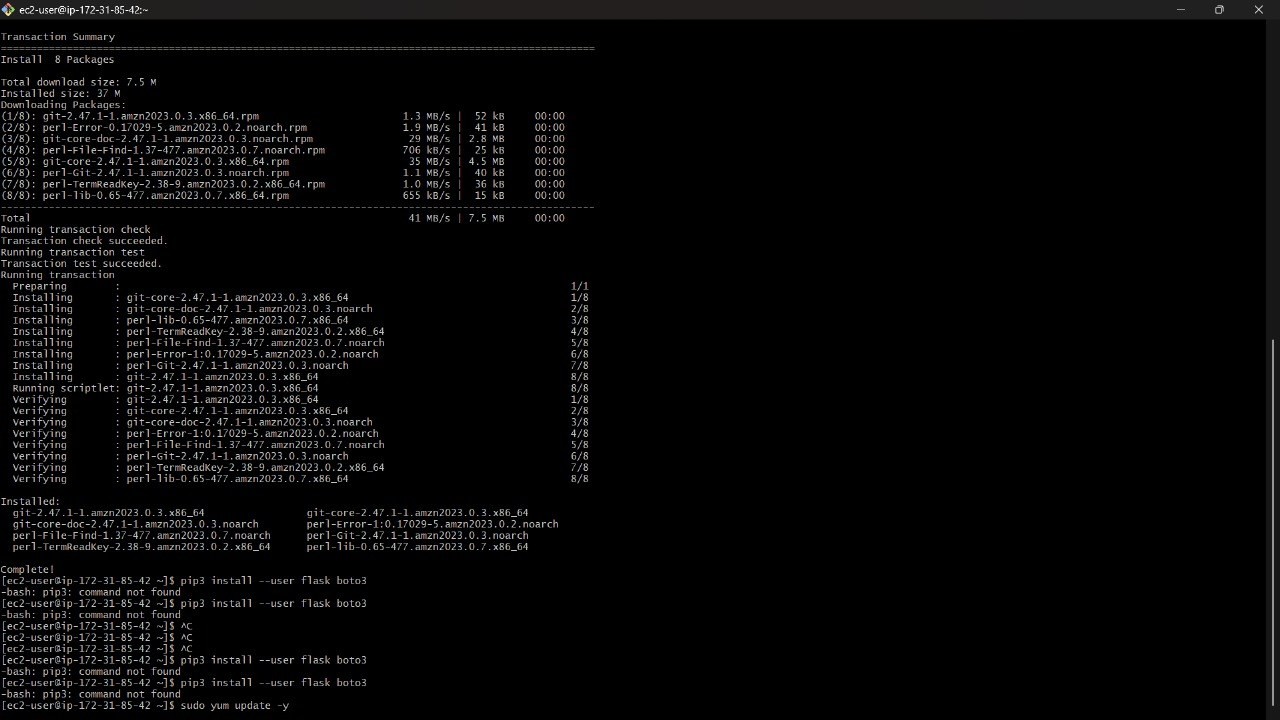
**Activity 5.1:** install dependencies



This will download your project to the EC2 instance. Now install dependencies, so type following commands.

* sudo yum update -y
* sudo yum install python3 git -y
* pip3 install --user flask boto3
* git clone https://github.com/<your-username>/<repo-name>.git
* cd <repository-name>





**Verify the Flask app is running**:



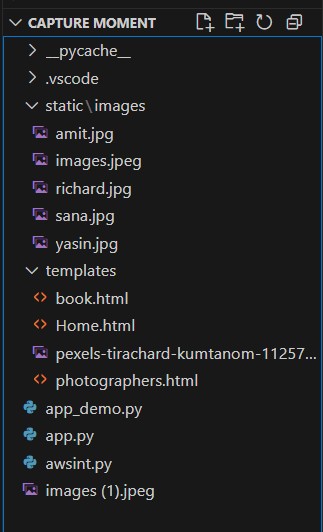
**Access the website through:**

**PublicIPs**54.234.69.121/

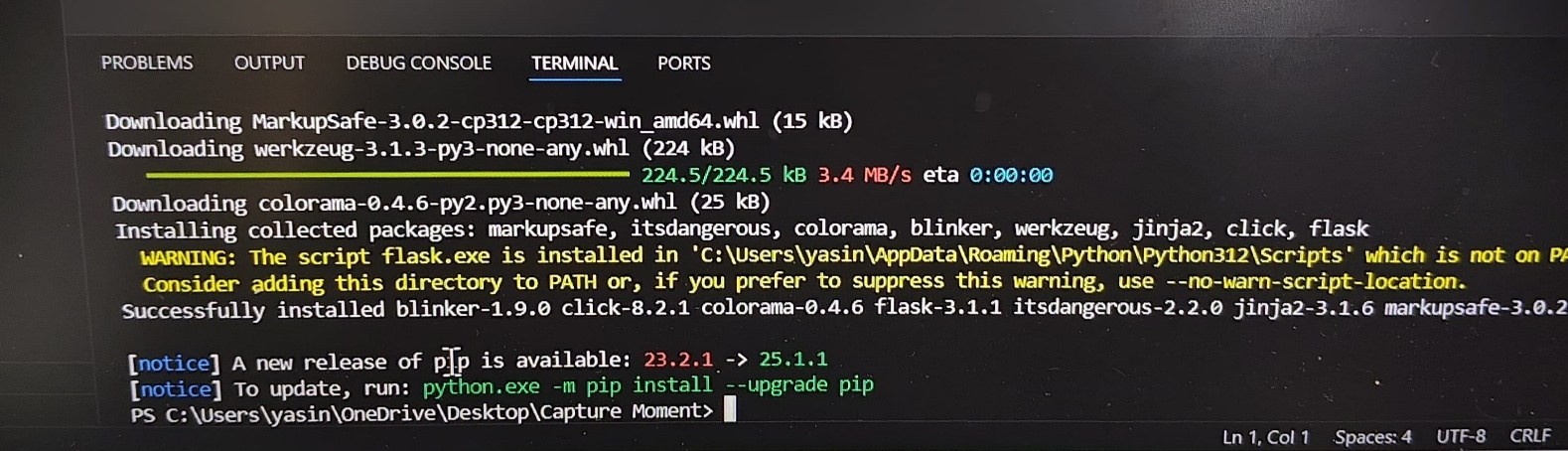
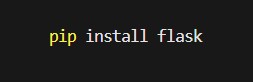
**Description: -** project is named **"**CAPTURE MOMENT**"**, likely a web-based photographer booking system.

It includes a static/images folder that holds image files for photographers and other visuals. Images such as Richard, sana, and yasin, suggest profiles of individual photographers. The templates folder contains HTML files used for rendering web pages in a Flask applicationHome.html serves as the main landing page of the website. and book.html is likely used for booking photographers and photographers.html probably displays a list or grid view of available photographers. The Python file app.py is most likely the main Flask application handling routes and logic and awsint.py seems to handle AWS integration. Overall, the structure reflects a wellorganized web app combining Flask, HTML templates, and AWS functionality.

**File Explorer Structure**



* **Flask App Initialization**



**Description:** This image shows the output of a successful pip install flask.Flask and its, as indicated by the Downloading and Installing collected packages dependencies are being installedlines. Packages include: flask, werkzeug, jinja2, click, its dangerous, colorama, blinker, and markup safe.



**Description:** initialize the Flask application instance using Flask ( name\_) to start building the web app.

* **Boto3 Initialization:**

You typically install it via the terminal using pip, like this:



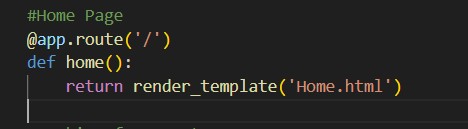
**Discription: -**This command tells Python’s package manager (pip) to download and install the boto3 library from the Python Package Index.Dependencies Installed Along with boto3 When you run the above command, it also installs.

* botocore – Low-level core functionality shared across AWS SDKs
* jmespath – Used for filtering JSON responses
* s3transfer – For efficient file uploads and downloads from S3
* urllib3, python-dateutil, and docutils – Used internally by botocore

After installation, the terminal will show:

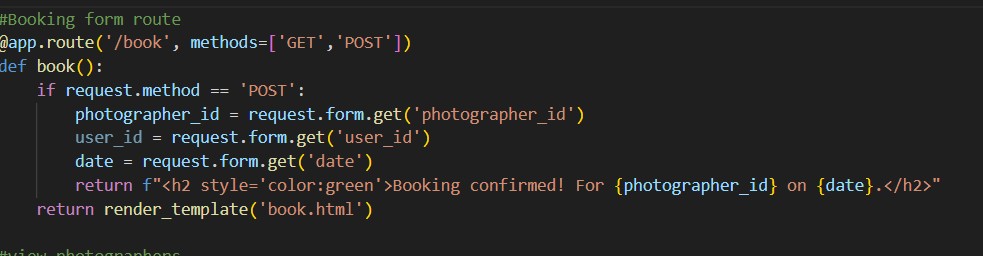
Successfully installed boto3-x.x.x botocore-x.x.x jmespath-x.x.x ...

* **Routes for Web Pages**
* **Home Route:**



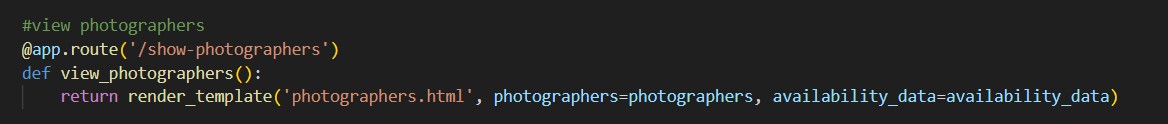
**Description:** define the home route / to automatically redirect users to the home page when they access the base URL

**Book Route:**



**Description:** The image shows a Flask route definition in Python that handles booking a photographer on the /book URL path. This defines a route /book which accepts both GET and POST HTTP methods. This route displays a booking form (book.html) and processes form submissions to confirm bookings by showing a success message with the photographer ID and date.

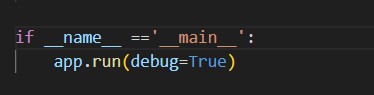
**View Photographer Route:**



**Description:** This route allows users to see all photographers and their availability.

It sends data from the backend to the photographers.html page using render template. The script is set to run in debug mode when executed as the main program

**Deployment Code:**



**Description:** start the Flask server to listen on all network interface at port with debug mode enabled for development and test

**Milestone 6: Testing and Deployment**

● **Activity 6.1** Conduct functional testing to verify booking a photographer and view photographers:



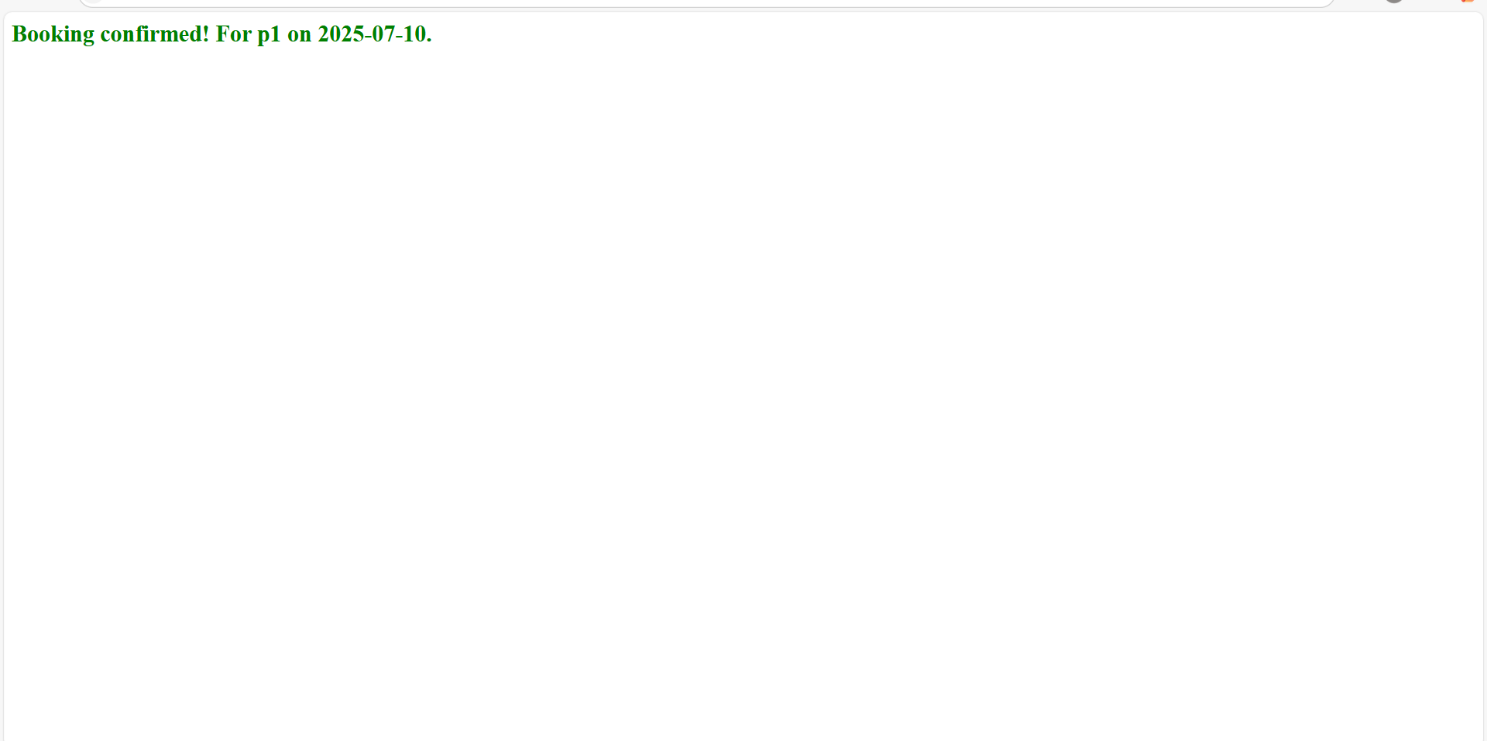
A screenshot of a computer

AI-generated content may be incorrect.

A white background with green text

AI-generated content may be incorrect.

A group of people holding cameras

AI-generated content may be incorrect.

**Conclusion: -** The Capture Moments – AWS-Powered Photographer Booking System is a robust and innovative platform designed to simplify and modernize the process of booking professional photographers. By utilizing AWS services such as DynamoDB for database management, S3 for secure image storage, and a Flask-based web application, the system ensures high performance, reliability, and scalability.

This project provides an efficient interface for users to browse, select, and book photographers based on their preferences, while also offering photographers an organized platform to manage their profiles and bookings. It effectively demonstrates how cloud computing can enhance traditional service-based industries by automating tasks, improving accessibility, and ensuring data security.