MedTrack: AWS Cloud-Enabled Healthcare Management System

1. Project Description

MedTrack is a full-stack, cloud-enabled healthcare management system using Flask for backend APIs, hosted on AWS EC2, with DynamoDB as its database. It provides a centralized platform for patients and doctors to register, book appointments, view medical history, and receive real-time notifications via AWS SNS. AWS IAM ensures secure, role-based access to system resources.

Hardware Required:

Processor: Intel i5 or equivalent (minimum). RAM: 4 GB (8 GB recommended for Full Stack MERN). Storage: 128 GB SSD or 128 GB HDD. Internet Connectivity: High-speed internet (minimum 10 Mbps per system). Additional: Audio-visual setup for interactive sessions (microphone, speakers, etc.).

Software Required:

Updated web browser (Google Chrome, Firefox, or Microsoft Edge). Visual Studio Code (or any preferred IDE). Git (latest version).

System Required:

Projector and Audio System for presentations in all labs/classrooms Classrooms/Labs are equipped with systems or provisions for students to join sessions with their own laptops.

Description:

In today's fast-evolving healthcare landscape, efficient communication and coordination between doctors and patients are crucial. MedTrack is a cloud-based healthcare management system that streamlines patient doctor interactions by providing a centralized platform for booking appointments, managing medical histories, and enabling diagnosis submissions. To address these challenges, the project utilizes Flask for backend development, AWS EC2 for hosting, and DynamoDB for managing data. MedTrack allows patients to register, log in, book appointments, and submit diagnosis reports online. The system ensures real-time notifications, enhancing communication between doctors and patients regarding appointments and medical submissions. Additionally, AWS Identity and Access Management (IAM) is employed to ensure secure access control to AWS resources, allowing only authorized users to access sensitive data. This cloud-based solution improves accessibility and efficiency in healthcare services for all users.

Scenarios:

Scenario 1: Efficient Appointment Booking System for Patients

In the MedTrack system, AWS EC2 provides a reliable infrastructure to manage multiple patients accessing the platform simultaneously. For example, a patient can log in, navigate to the appointment booking page, and easily submit a request for an appointment. Flask handles backend operations, efficiently retrieving and processing user data in real-time. The cloud-based architecture allows the platform to handle a high volume of appointment requests during peak periods, ensuring smooth operation without delays.

Scenario 2: Secure User Management with IAM

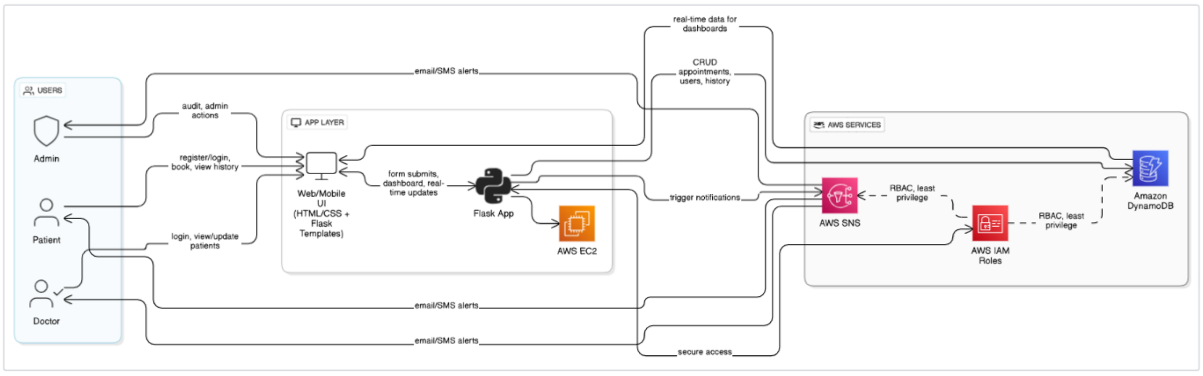
MedTrack utilizes AWS IAM to manage user permissions and ensure secure access to the system. For instance, when a new patient registers, an IAM user is created with specific roles and permissions to access only the features relevant to them. Doctors have their own IAM configurations, allowing them access to patient records and appointment details while maintaining strict security protocols. This setup ensures that sensitive data is accessible only to authorized users.

Scenario 3: Easy Access to Medical History and Resources

The MedTrack system provides doctors and patients with easy access to medical histories and relevant resources. For example, a doctor logs in to view a patient's medical history and upcoming appointments. They can quickly access, and update records as needed. Flask manages real-time data fetching from DynamoDB, while EC2 hosting ensures the platform performs seamlessly even when multiple users access it simultaneously, offering a smooth and uninterrupted user experience.

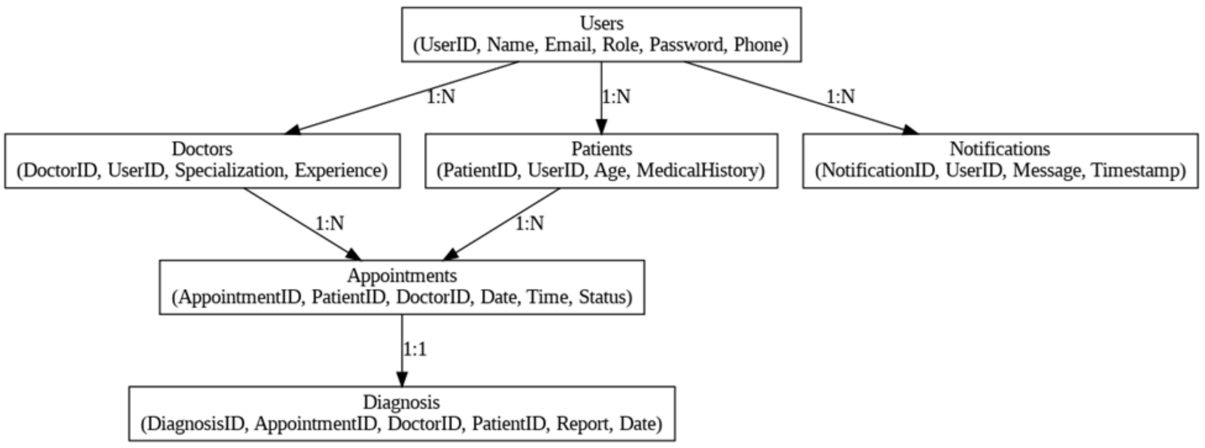
Architecture

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 notifications are enabled via Amazon SNS, enhancing communication and user engagement.



Entity Relationship (ER) Diagram

An ER (Entity-Relationship) diagram visually represents the logical structure of a database by defining entities, their attributes, and the relationships between them. It helps organize data efficiently by illustrating how different components of the system interact and relate. This structured approach supports effective database normalization, data integrity, and simplified query design.


Users
(UserID, Name, Email, Role, Password, Phone)
1:N
1:N
1:N
Doctors
(DoctorID, UserID, Specialization, Experience)
Patients
(PatientID, UserID, Age, MedicalHistory)
Notifications
(NotificationID, UserID, Message, Timestamp)
1:N
1:N
Appointments
(AppointmentID, PatientID, DoctorID, Date, Time, Status)
1:1
Diagnosis
(DiagnosisID, AppointmentID, DoctorID, PatientID, Report, Date)

Pre-requisites

- AWS Account Setup: https://docs.aws.amazon.com/accounts/latest/reference/getting-started.html
- AWS IAM (Identity and Access Management): https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html
- AWS EC2 (Elastic Compute Cloud): https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
- AWS DynamoDB: https://docs.aws.amazon.com/amazondynamodb/Introduction.html
- Amazon SNS: https://docs.aws.amazon.com/sns/latest/dg/welcome.html
- Git Documentation: https://git-scm.com/doc
- VS Code Installation: (download the VS Code using the below link or you can get that in Microsoft store) https://code.visualstudio.com/download

Project WorkFlow

Milestone 1. Web Application Development and Setup

- Develop the Backend Using Flask.
- Integrate AWS Services Using boto3.

Milestone 2. AWS Account Setup and Login

- Set up an AWS account if not already done.
- Login to AWS Management Console.

Milestone 3. DynamoDB Database Creation and Setup

- Create a DynamoDB Table.
- Configure Attributes for User Data and Book Requests.

Milestone 4. SNS Notification Setup

- Create SNS topics for book request notifications.
- Subscribe users and library staff to SNS email notifications.

Milestone 5. IAM Role Setup

- Create IAM Role
- Attach Policies

Milestone 6. EC2 Instance Setup

- Launch an EC2 instance to host the Flask application.
- Configure security groups for HTTP, and SSH access.

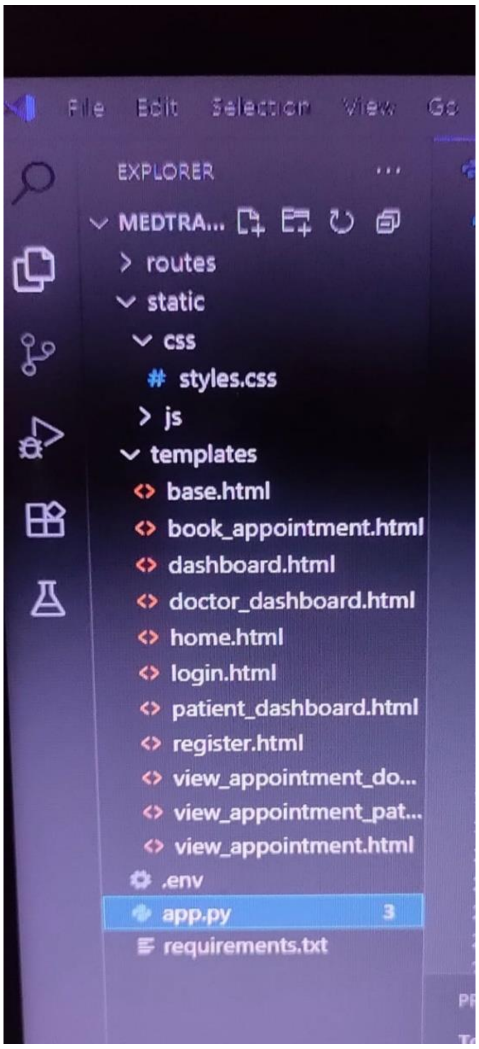
Milestone 7. Deployment using EC2

- Upload Flask Files
- Run the Flask App

Milestone 8. Testing and Deployment

- Conduct functional testing to verify user registration, login, book requests, and notifications.

1.Develop the backend using Flask



```
MEDTRA...
  > routes
  v static
    v css
      # styles.css
    > js
  v templates
    <> base.html
    <> book_appointment.html
    <> dashboard.html
    <> doctor_dashboard.html
    <> home.html
    <> login.html
    <> patient_dashboard.html
    <> register.html
    <> view_appointment_do...
    <> view_appointment_pat...
    <> view_appointment.html
  .env
  app.py                3
  requirements.txt
```

Description of the code:

1. Imports:

- Import necessary modules like Flask, render_template, request, redirect, url_for, session, and others as needed for your app.

2. Flask App Initialization:

- app = Flask(__name__): Starts the Flask web application.
- app.secret_key: Used for securely signing the session cookie and enabling flash messages.

3. Temporary In-Memory Storage:

- users: A list or data structure to store registered user information temporarily.
- bookings: A list to store appointment or ticket bookings.
- booking_counter: A counter to uniquely identify each booking.

4. Authentication Routes:

4.1 Homepage /:

- Displays the landing page or redirects users based on login status.

4.2 Login /login:

- **GET request**: Shows the login form.
- **POST request**:
    - Validates user credentials.
    - Checks if email exists and password matches.
    - On success, stores user info in session.
    - On failure, flashes an "Invalid login" message.

4.3 Signup /signup:

- **GET request**: Shows the signup form.
- **POST request**:
    - Collects new user details from the form.
    - Checks if the email is already registered.
    - Hashes the password and stores the user data.
    - Redirects to the login page on successful signup.

4.4 Logout /logout:

- Clears the user session to log out.
- Flashes a logout confirmation message.
- Redirects to the homepage.

5. Main Application Pages:

5.1 Home after Login /home1, About /about, Contact /contact_us:

- Checks if the user is logged in.
- If logged in, renders the home1.html dashboard.

- If not logged in, redirects to /login.
- Serves static pages like About and Contact Us.

6. Doctor and Patient Routes:

- **Doctor Routes (doctor.py)**:
  Handles doctor-specific views such as doctor dashboard, viewing appointments, and managing schedules.
- **Patient Routes (patient.py)**:
  Manages patient-specific functionality like booking appointments and viewing their own appointments.

7. Appointment Booking:

- Page to book new appointments.
- Stores booking info in bookings.
- Associates bookings with users (patients) and doctors.
- Allows viewing and managing appointments based on user roles.

8. Static Files:

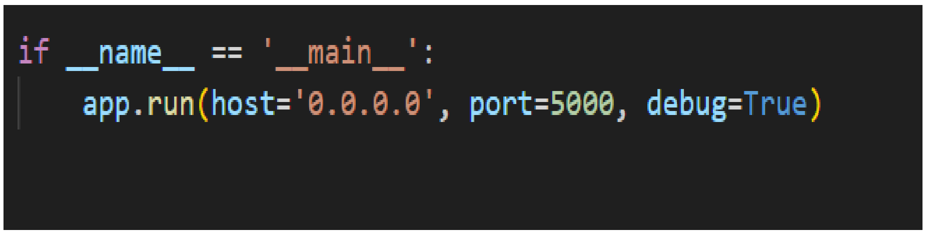
- CSS and JavaScript files stored under static/css and static/js for styling and interactive behavior.
- HTML templates under templates/ folder for rendering views.

9. Environment Variables (.env):

- Stores sensitive information like SECRET_KEY and database credentials.
- Loaded into the app to configure secret keys and other settings.

10. Running the Flask App:

- The app is run using:

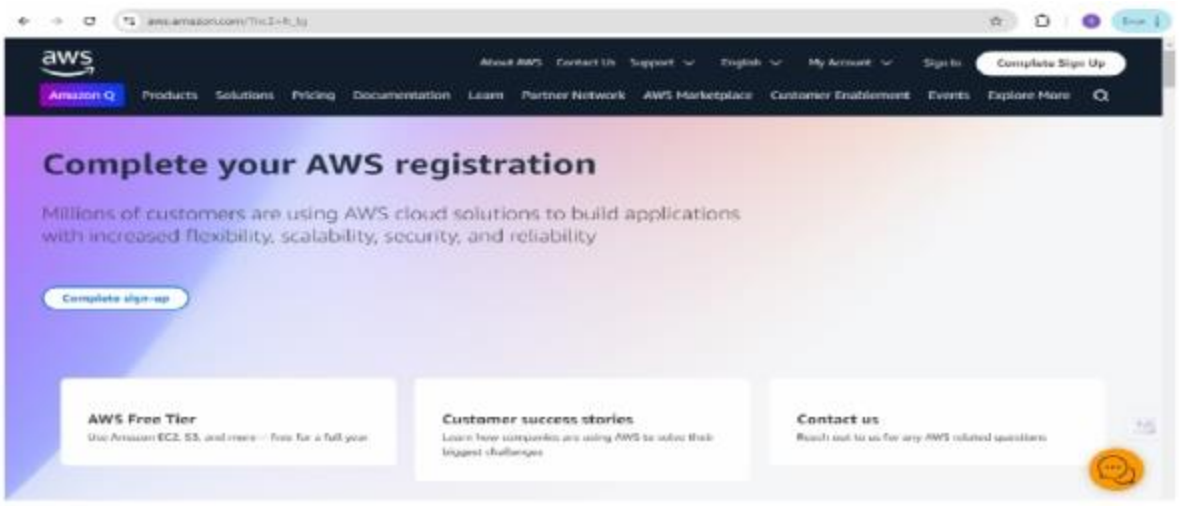


```python
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

2.AWS Account Setup and Login

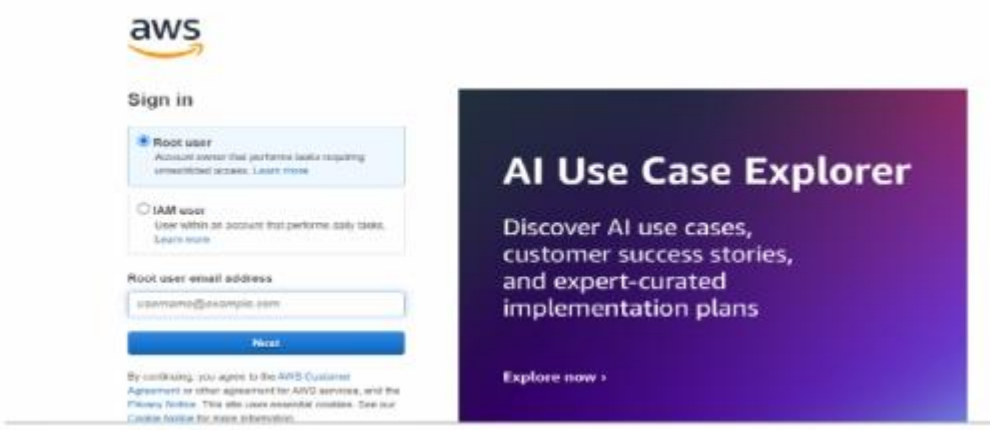
● Activity 1.1: Set up an AWS account if not already done.

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



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

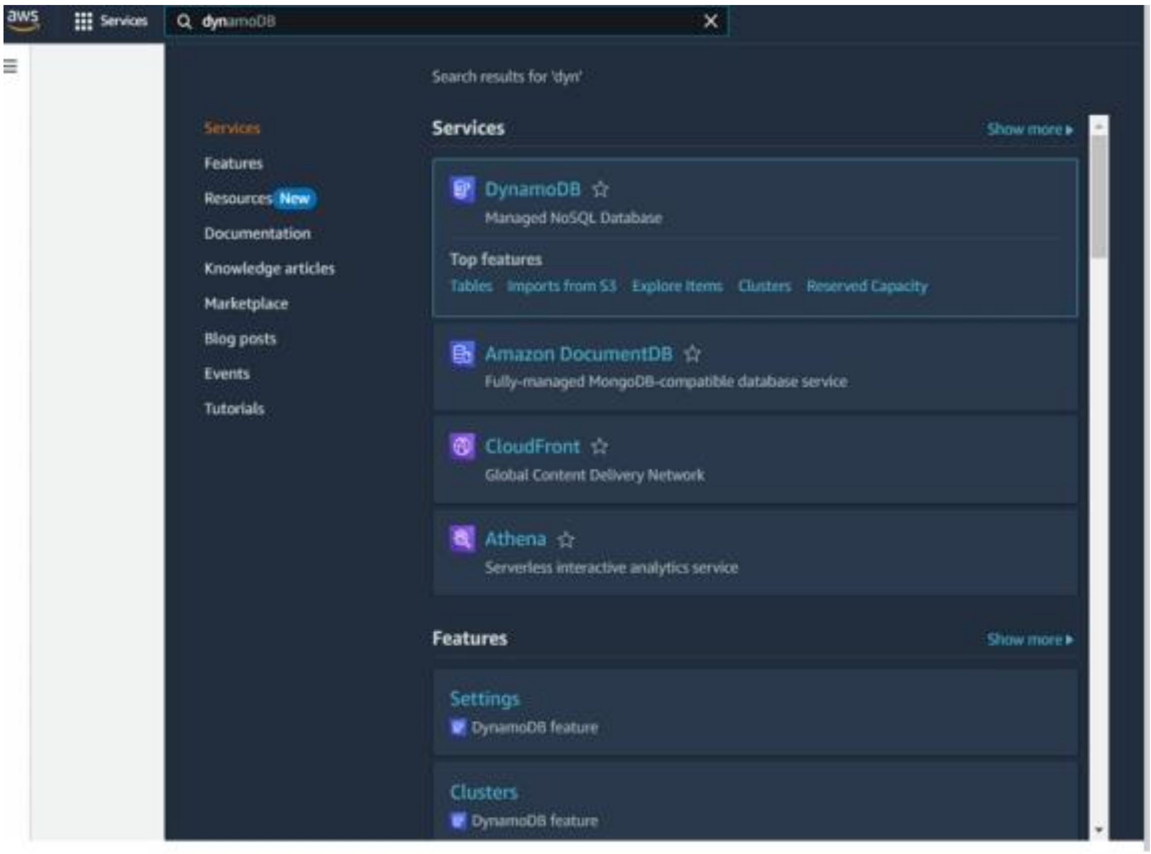
○ **After setting up your account, log in to the AWS Management Console.**

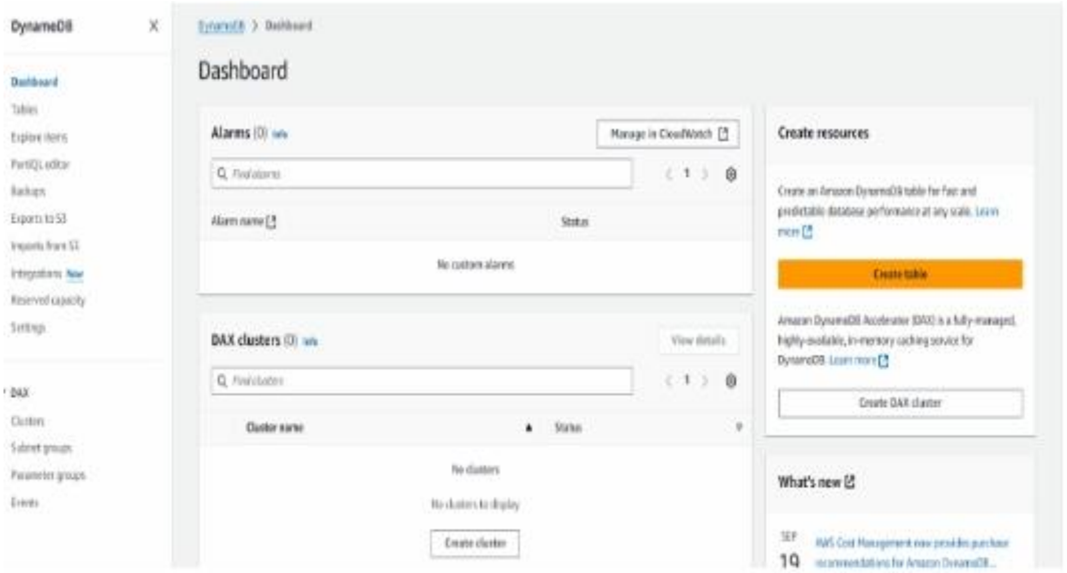


2: DynamoDB Database Creation and Setup

● **Activity 2.1:Navigate to the DynamoDB**

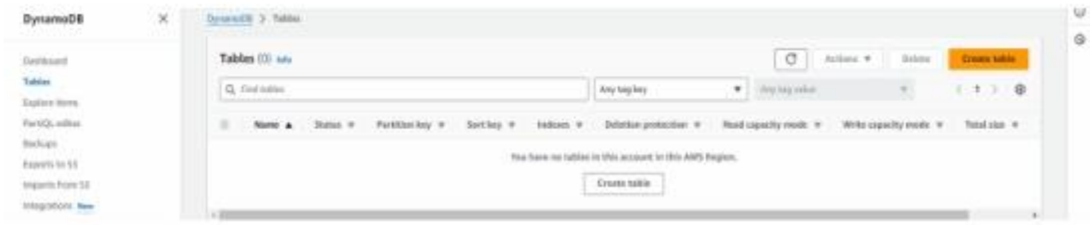
○ **In the AWS Console, navigate to DynamoDB and click on create tables.**





Create an Amazon DynamoDB table for fast and predictable database performance at any scale. Learn more

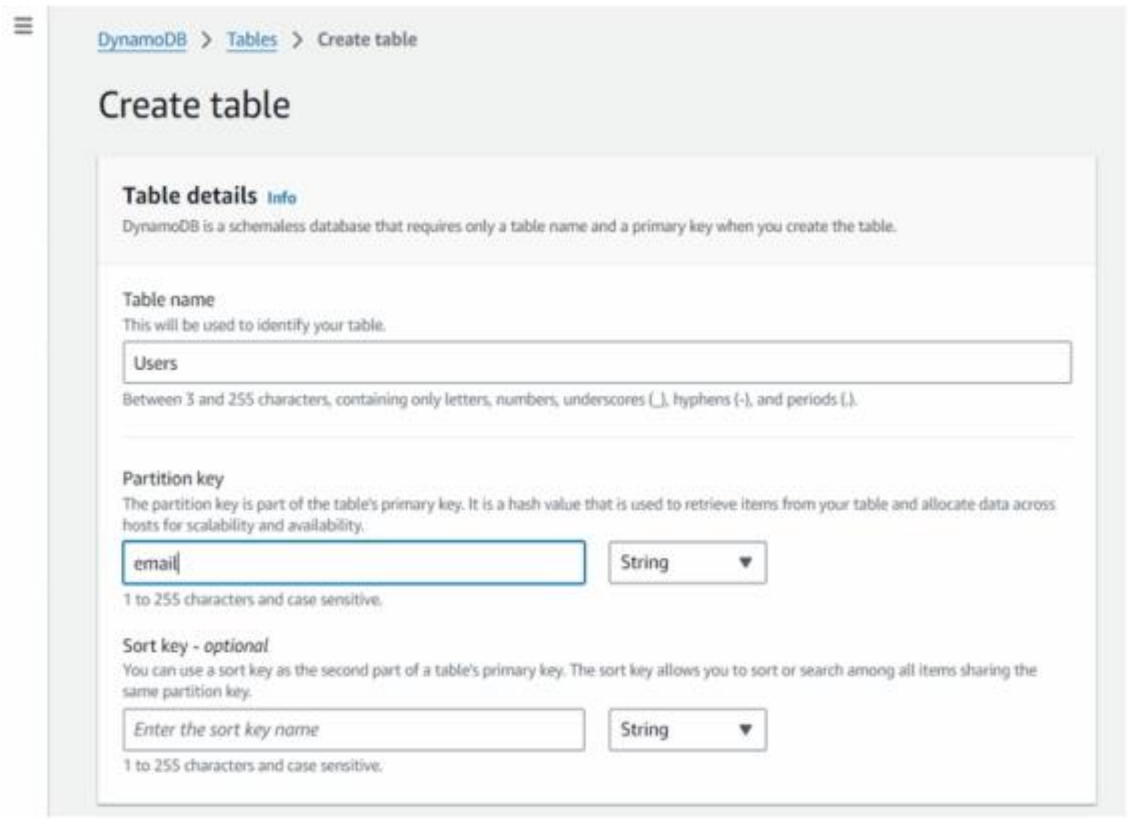
Amazon DynamoDB Accelerator (DAX) is a fully-managed, highly-available, in-memory caching service for DynamoDB. Learn more



You have no tables in this account in this AWS Region.

● **Activity 2.2:Create a DynamoDB table for storing registration details and book requests.**

○ **Create Users table with partition key "Email" with type String and click on create tables.**



Create table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

Users

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

email | String

1 to 255 characters and case sensitive.

Sort key - *optional*

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

Enter the sort key name | String

1 to 255 characters and case sensitive.

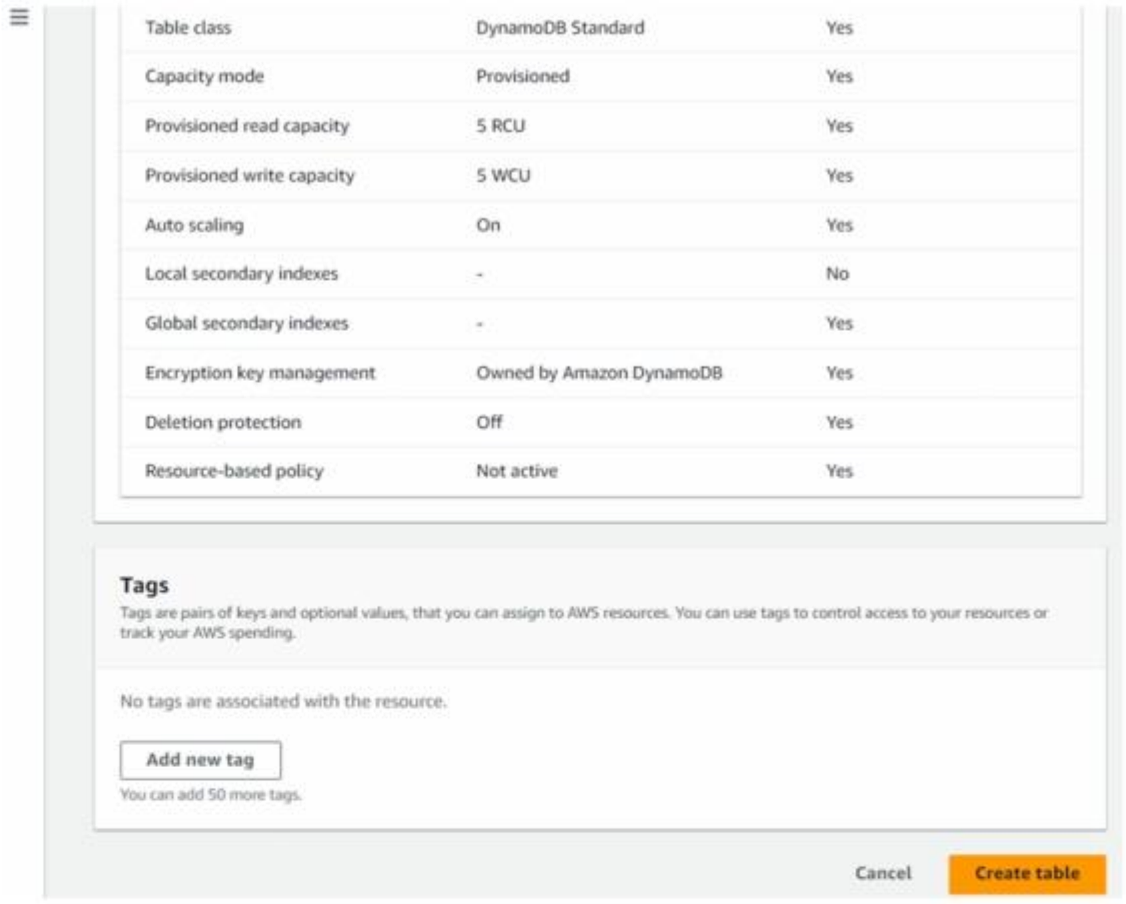


Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

Tags

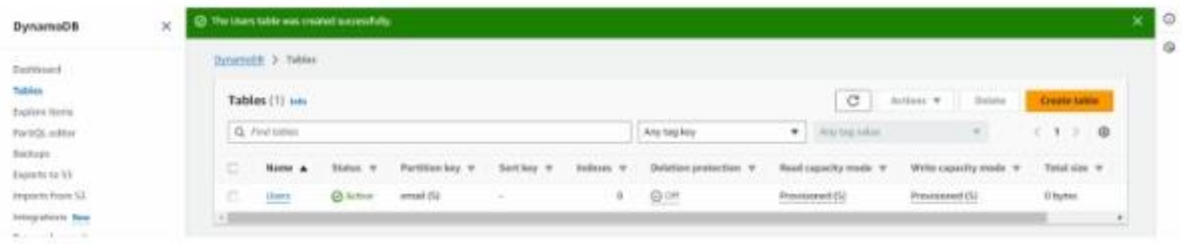
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

Add new tag

You can add 50 more tags.

Cancel Create table



DynamoDB

Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New

The Users table was created successfully.

DynamoDB > Tables

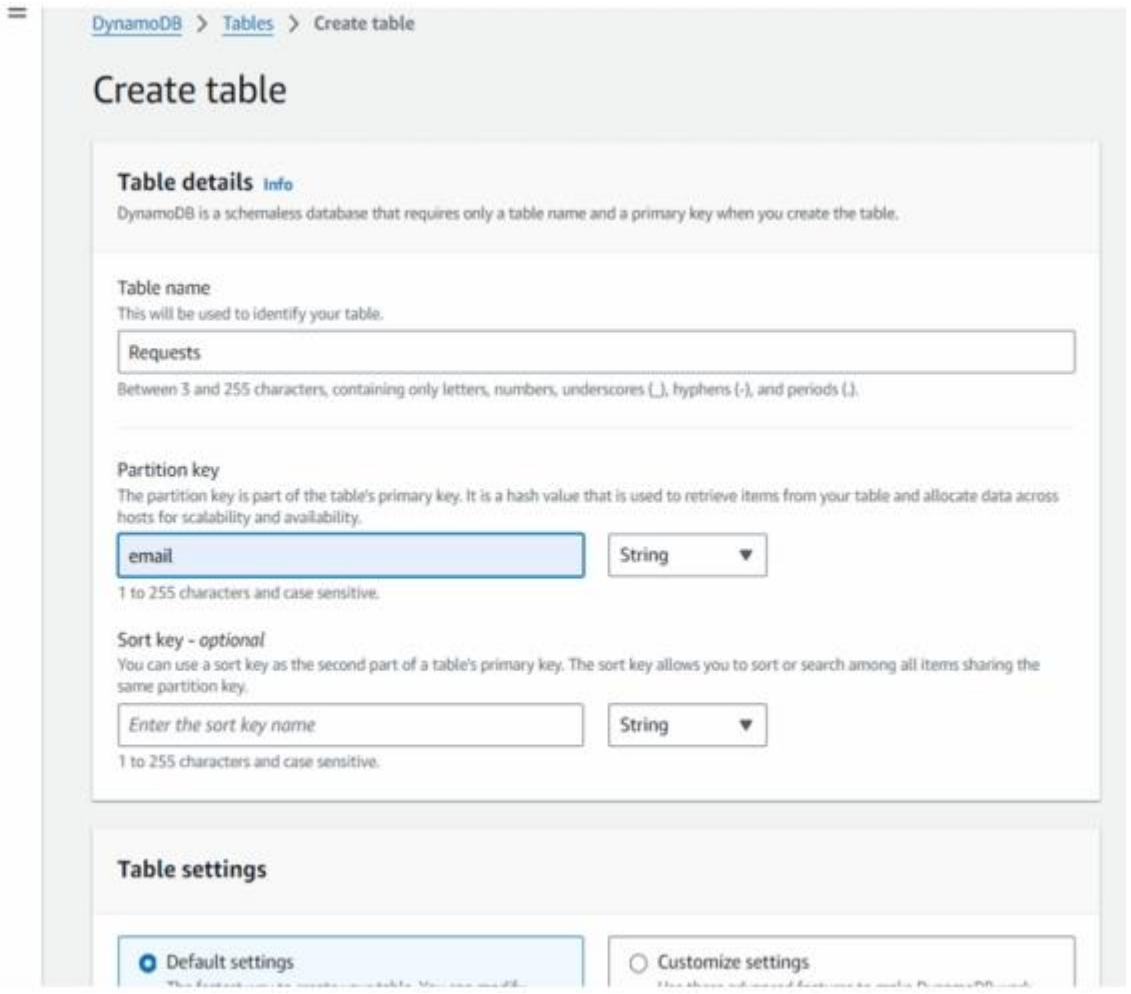
Tables (1) Info

Actions Delete Create table

Find tables | Any tag key | Any tag value

	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size
	Users	Active	email (S)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes

○ **Follow the same steps to create a requests table with Email as the primary key for book requests data.**



DynamoDB > Tables > Create table

Create table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

Requests

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

email | String

1 to 255 characters and case sensitive.

Sort key - *optional*

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

Enter the sort key name | String

1 to 255 characters and case sensitive.

Table settings

[✓ Default settings] Customize settings

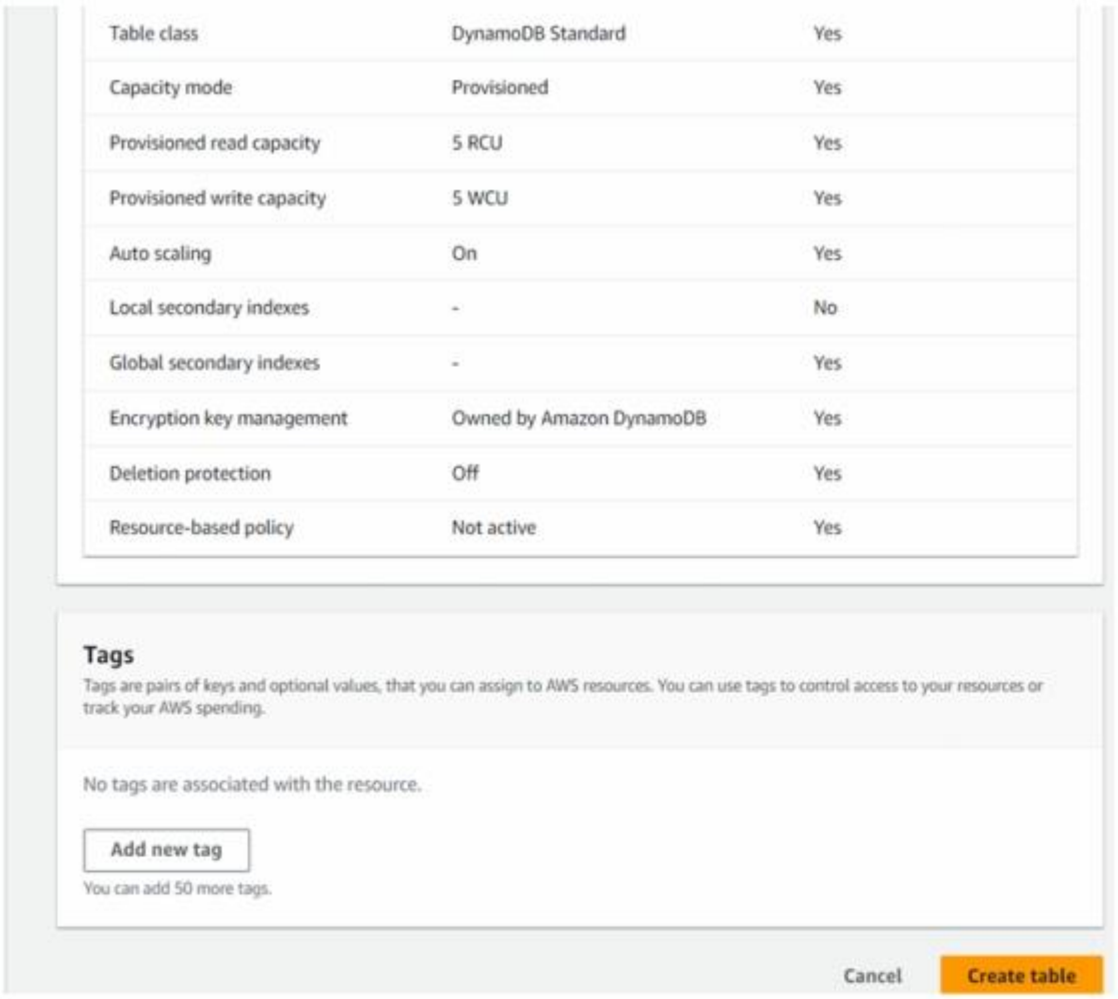


Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

Tags

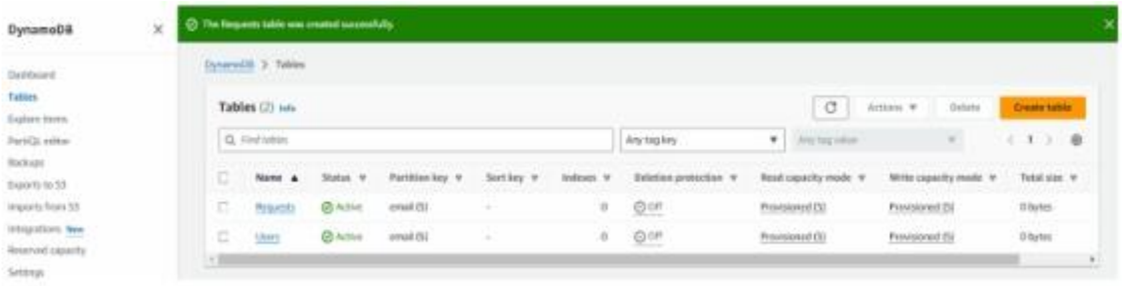
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

Add new tag

You can add 50 more tags.

Cancel Create table



The Requests table was created successfully.

DynamoDB > Tables

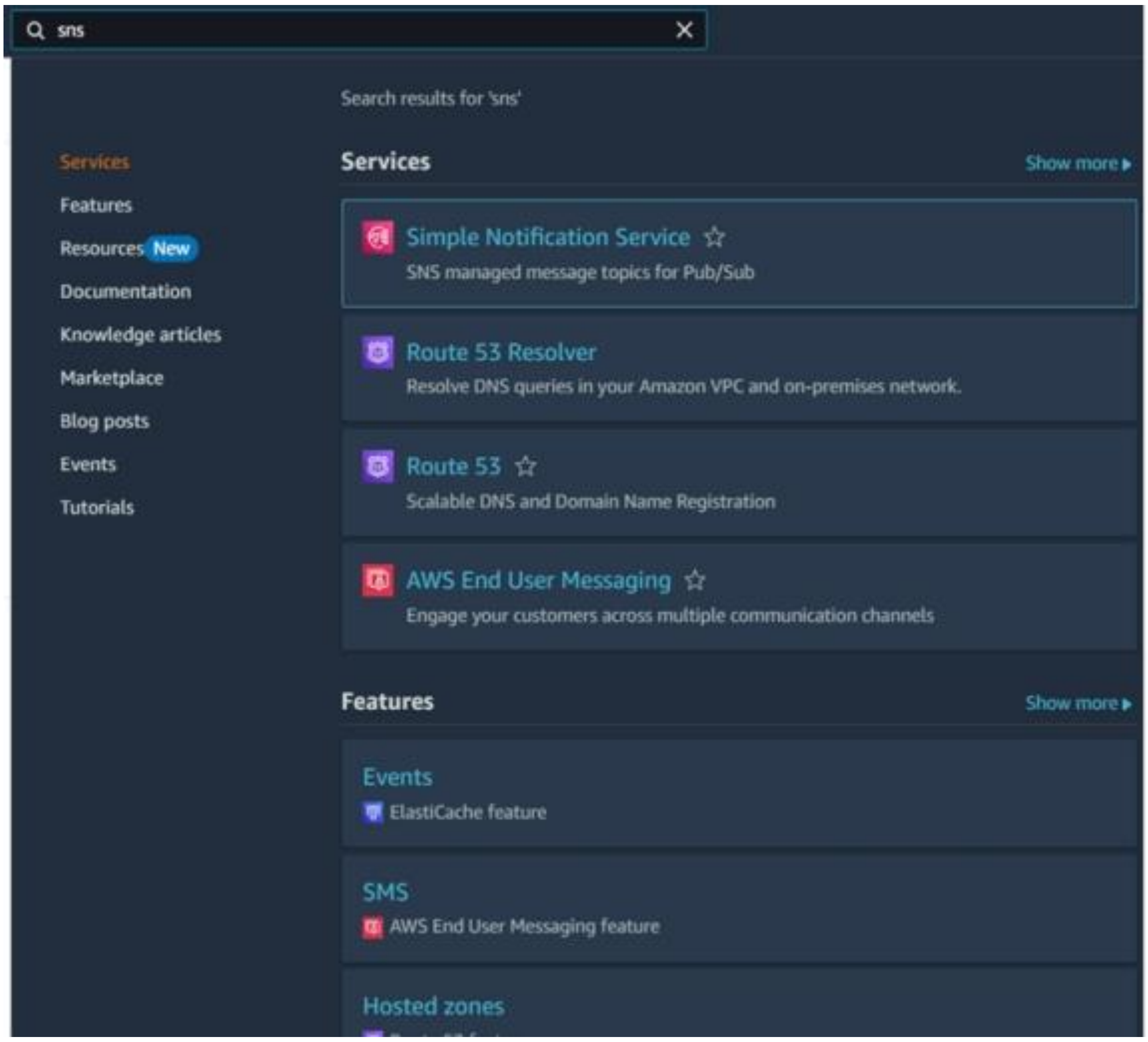
Tables (2) Info

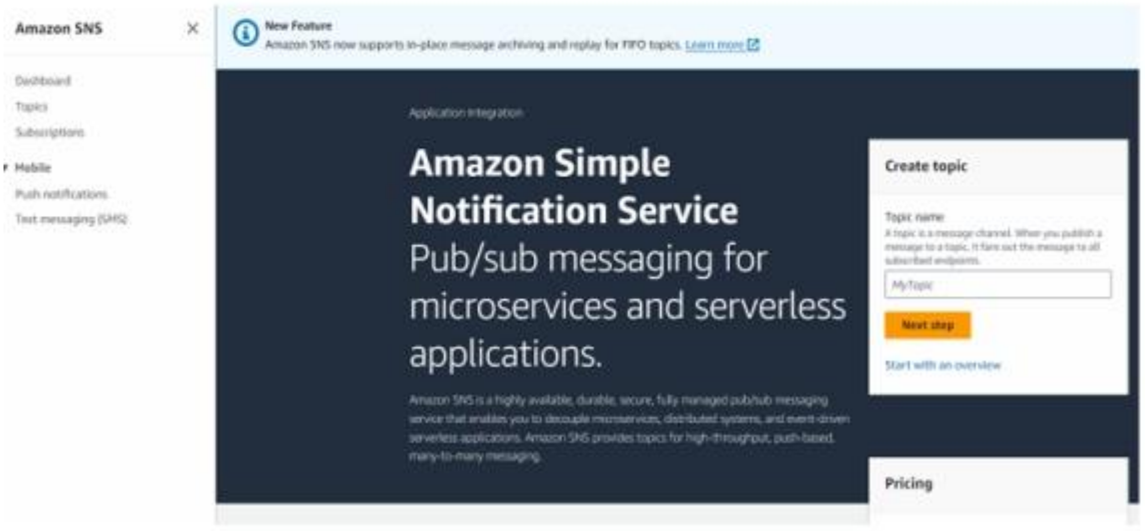
Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size
Requests	Active	email (S)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes
Users	Active	email (S)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes

3: SNS Notification Setup

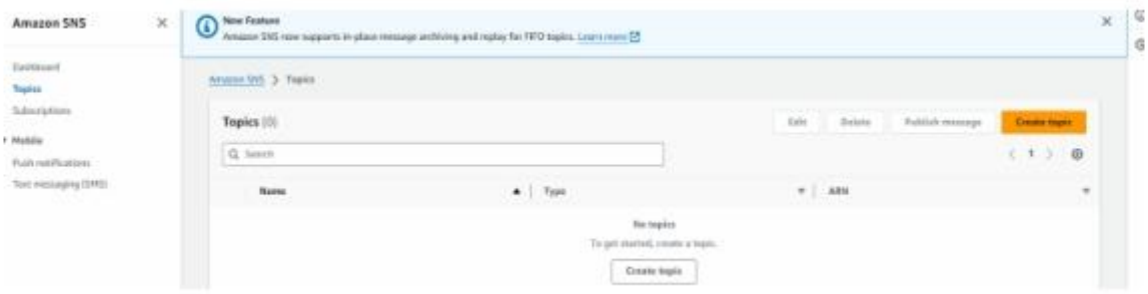
- **Activity 3.1: Create SNS topics for sending email notifications to users and library staff.**

    ○ **In the AWS Console, search for SNS and navigate to the SNS Dashboard.**


sns
Search results for 'sns'
Services
Features
Resources New
Documentation
Knowledge articles
Marketplace
Blog posts
Events
Tutorials
Services
Show more
Simple Notification Service
SNS managed message topics for Pub/Sub
Route 53 Resolver
Resolve DNS queries in your Amazon VPC and on-premises network.
Route 53
Scalable DNS and Domain Name Registration
AWS End User Messaging
Engage your customers across multiple communication channels
Features
Show more
Events
ElastiCache feature
SMS
AWS End User Messaging feature
Hosted zones


Amazon SNS
Dashboard
Topics
Subscriptions
Mobile
Push notifications
Text messaging (SMS)
New Feature
Amazon SNS now supports in-place message archiving and replay for FIFO topics. Learn more
Application Integration
Amazon Simple Notification Service
Pub/sub messaging for microservices and serverless applications.
Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.
Create topic
Topic name
A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.
MyTopic
Next step
Start with an overview
Pricing

○ Click on **Create Topic** and choose a name for the topic**.**



Amazon SNS

New Feature

Amazon SNS now supports in-place message archiving and replay for FIFO topics. Learn more

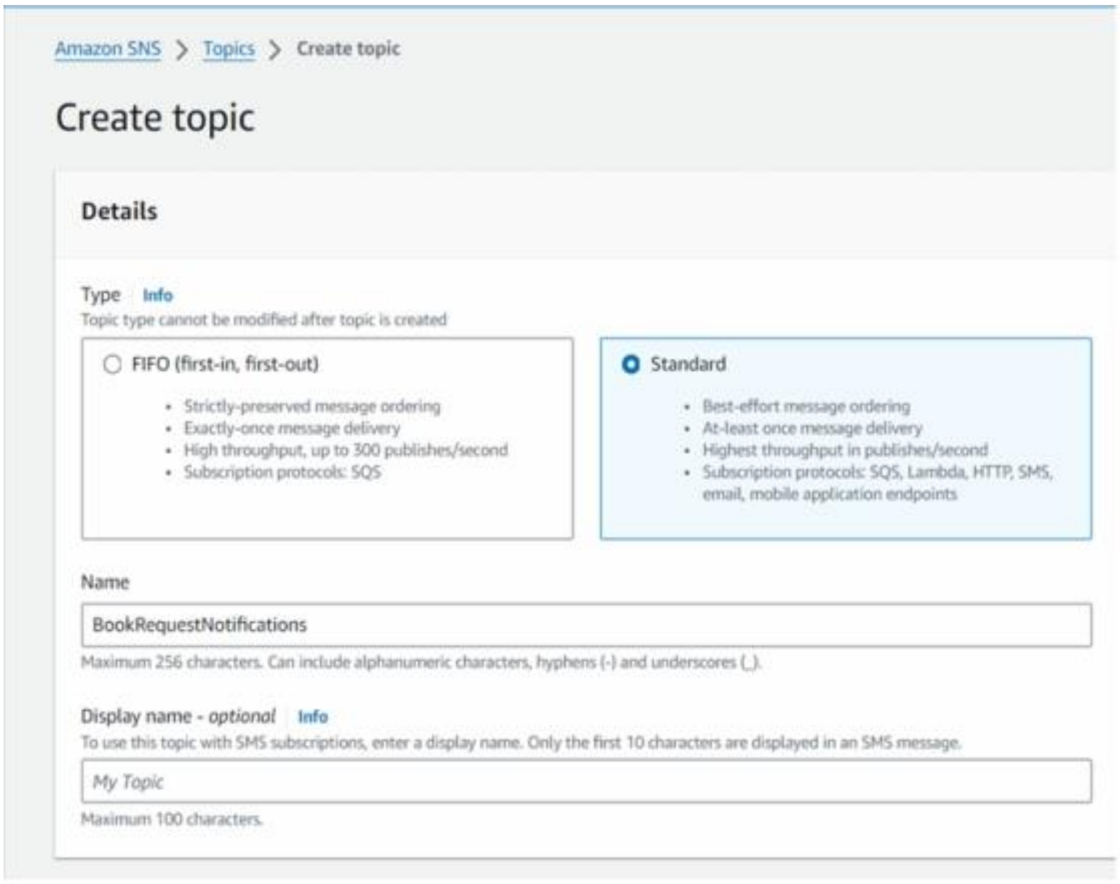
Amazon SNS > Topics

Topics (0)

No topics

To get started, create a topic.

○ **Choose Standard type for general notification use cases and Click on Create Topic.**



Amazon SNS > Topics > Create topic

Create topic

Details

Type Info

Topic type cannot be modified after topic is created

FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

[✓ Standard]

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name

BookRequestNotifications

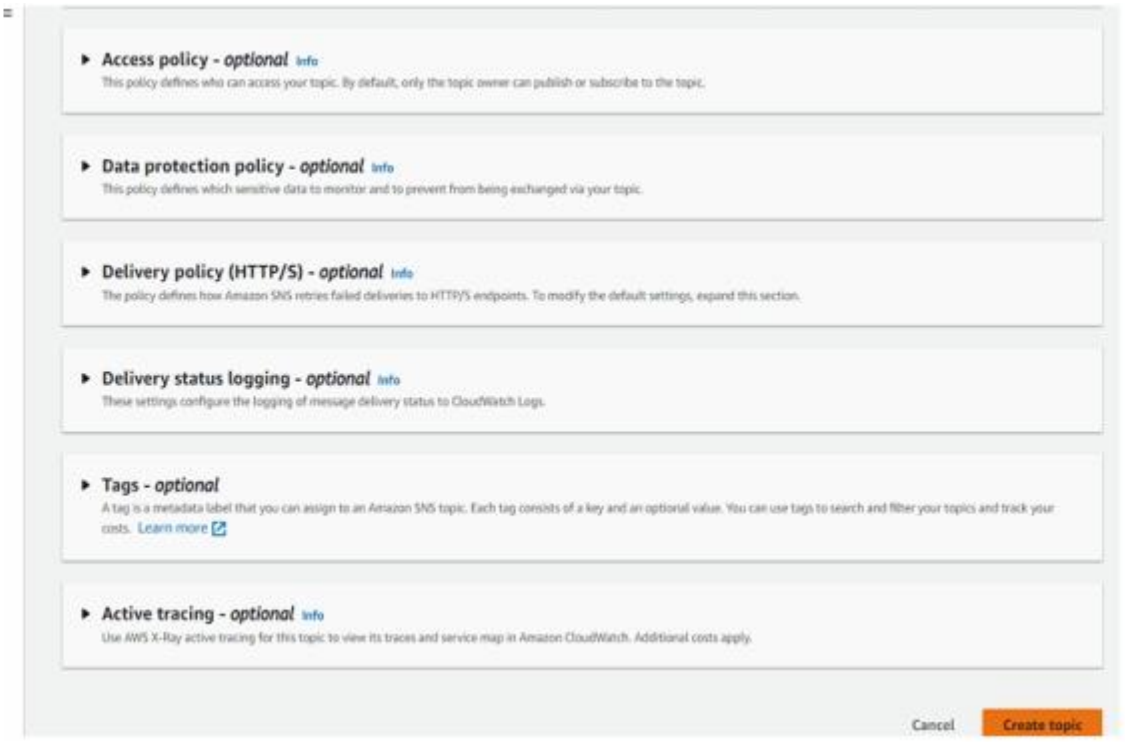
Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional* Info

To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

My Topic

Maximum 100 characters.



▸ **Access policy - *optional*** Info
This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic.

▸ **Data protection policy - *optional*** Info
This policy defines which sensitive data to monitor and to prevent from being exchanged via your topic.

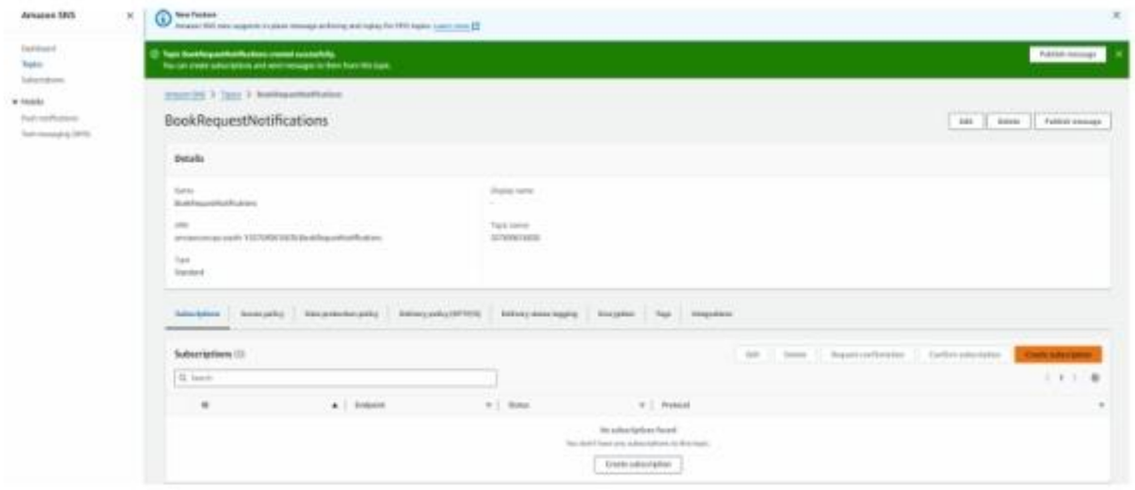
▸ **Delivery policy (HTTP/S) - *optional*** Info
The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section.

▸ **Delivery status logging - *optional*** Info
These settings configure the logging of message delivery status to CloudWatch Logs.

▸ **Tags - *optional***
A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. Learn more

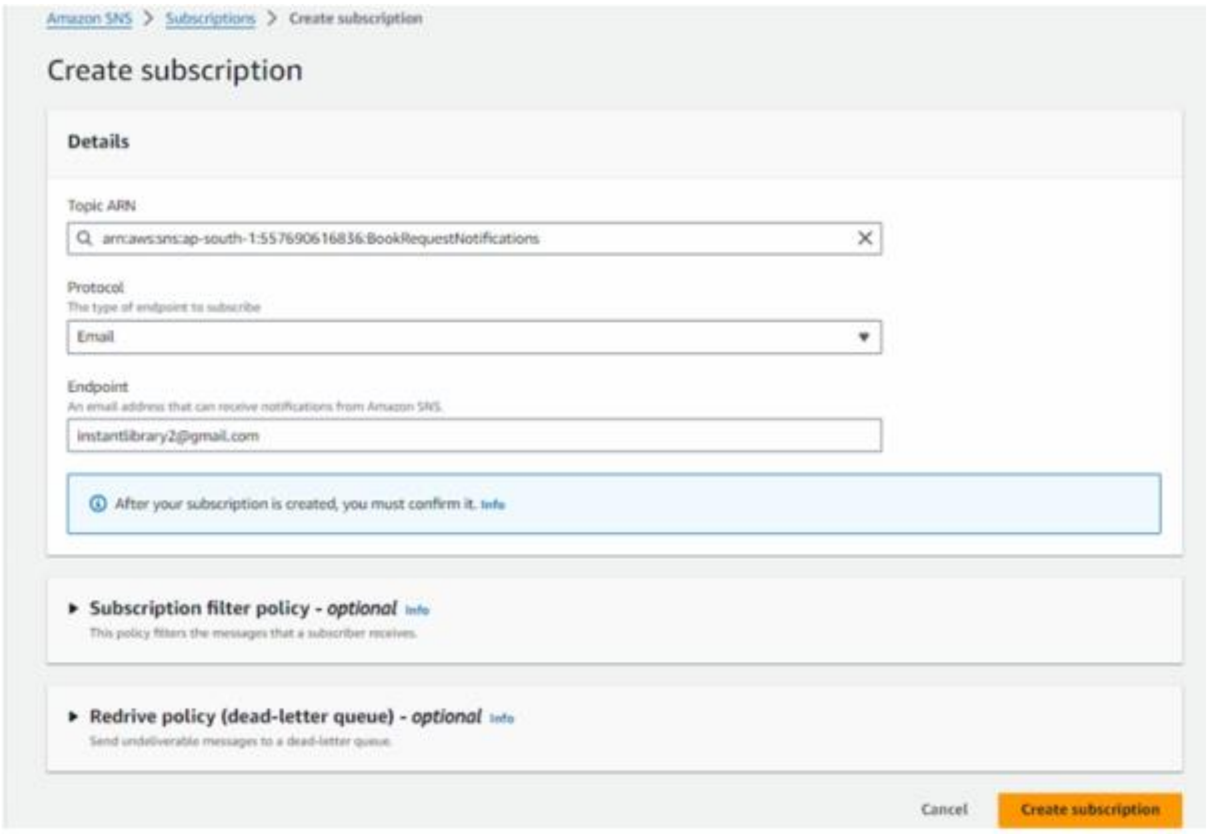
▸ **Active tracing - *optional*** Info
Use AWS X-Ray active tracing for this topic to view its traces and service map in Amazon CloudWatch. Additional costs apply.

○ Configure the SNS topic and note down the Topic ARN.



● Activity 3.2: Subscribe users and staff to relevant SNS topics to receive real-time notifications when a book request is made.

○ Subscribe users (or admin staff) to this topic via Email. When a book request is made, notifications will be sent to the subscribed emails.



Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN

arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications

Protocol

The type of endpoint to subscribe

Email

Endpoint

An email address that can receive notifications from Amazon SNS.

instantlibrary2@gmail.com

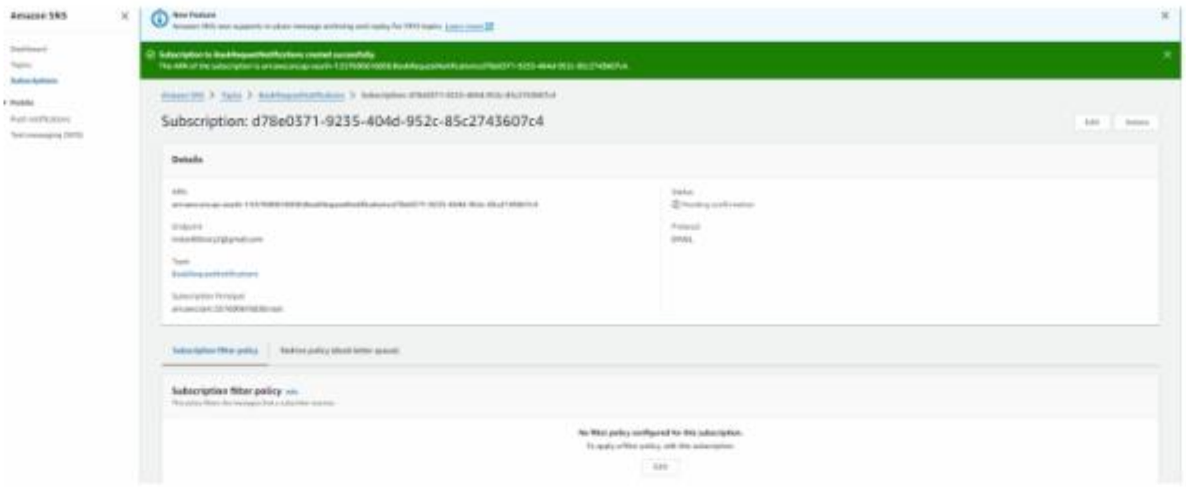
After your subscription is created, you must confirm it. Info

▸ **Subscription filter policy - *optional*** Info

This policy filters the messages that a subscriber receives.

▸ **Redrive policy (dead-letter queue) - *optional*** Info

Send undeliverable messages to a dead-letter queue.

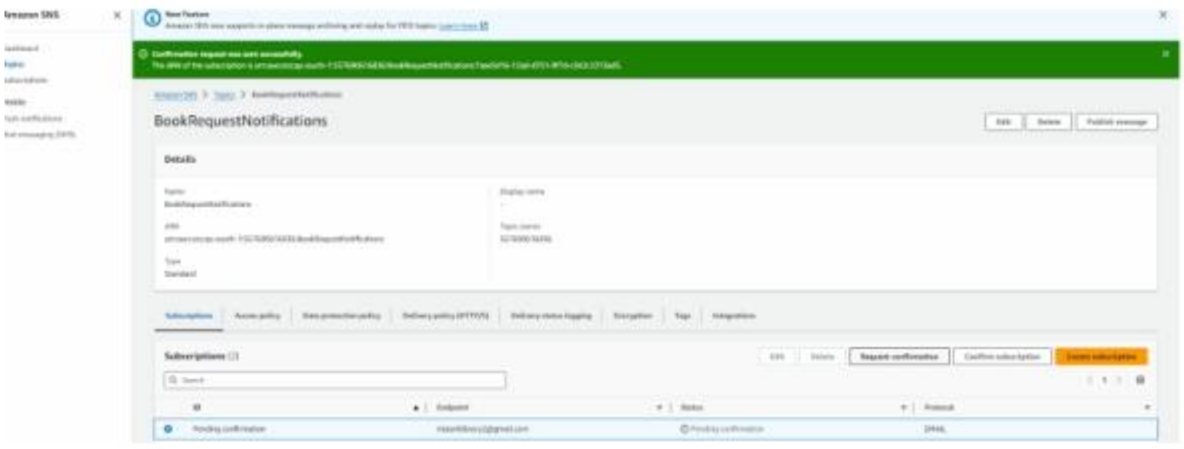


Subscription: d78e0371-9235-404d-952c-85c2743607c4

Details

Subscription filter policy

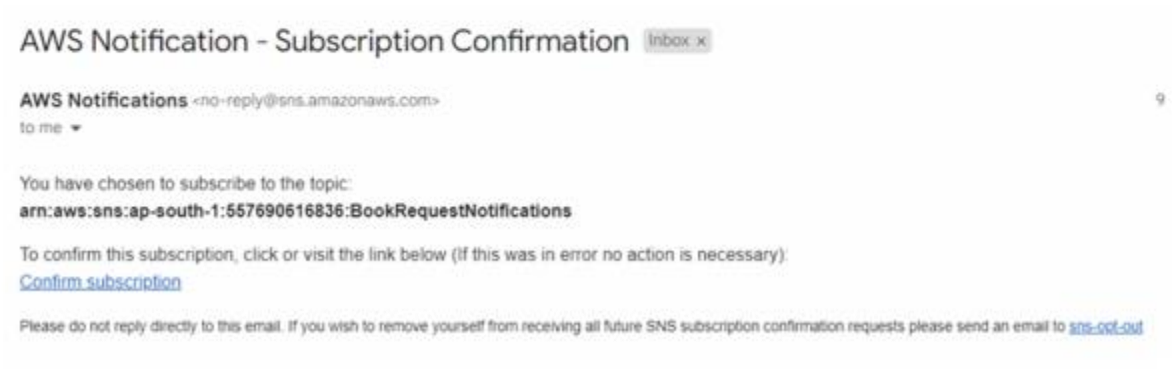
○ After subscription request for the mail confirmation


BookRequestNotifications

Details

Subscriptions

○ **Navigate to the subscribed Email account and Click on the confirm subscription in the AWS Notification- Subscription Confirmation mail.**

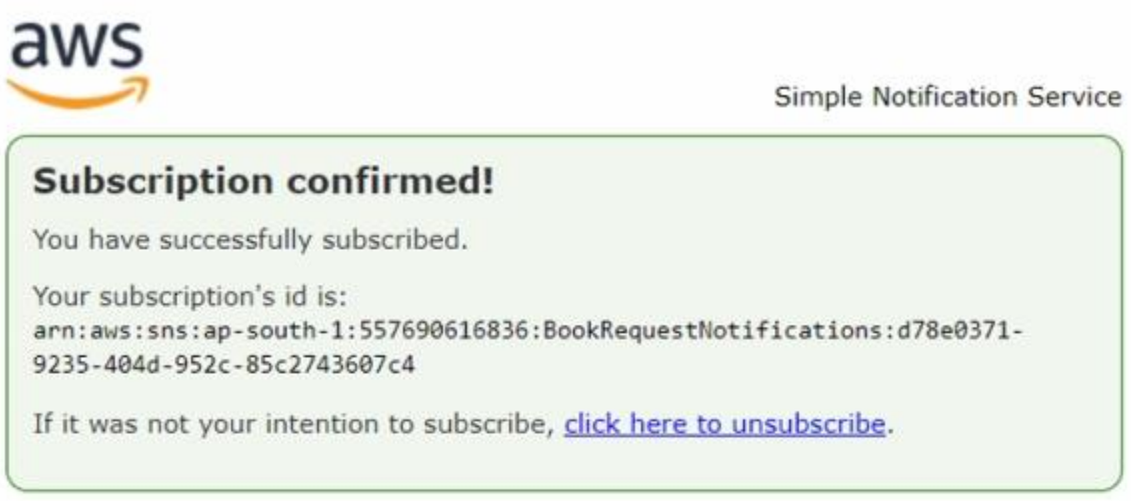

AWS Notification - Subscription Confirmation Inbox ×

AWS Notifications <no-reply@sns.amazonaws.com>
to me

You have chosen to subscribe to the topic:
arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to sns-opt-out


Simple Notification Service

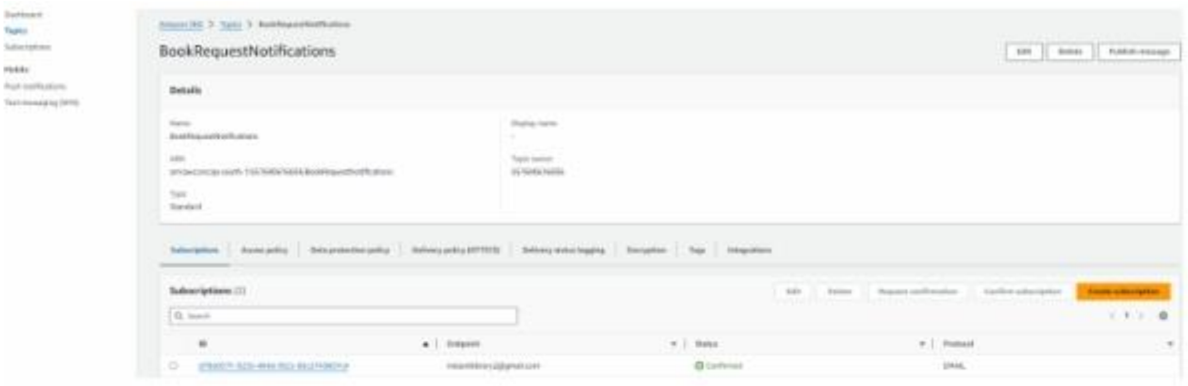
Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:
`arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4`

If it was not your intention to subscribe, click here to unsubscribe.

- Successfully done with the SNS mail subscription and setup, now store the ARN link.



BookRequestNotifications

Details

Name
BookRequestNotifications

Type
Standard

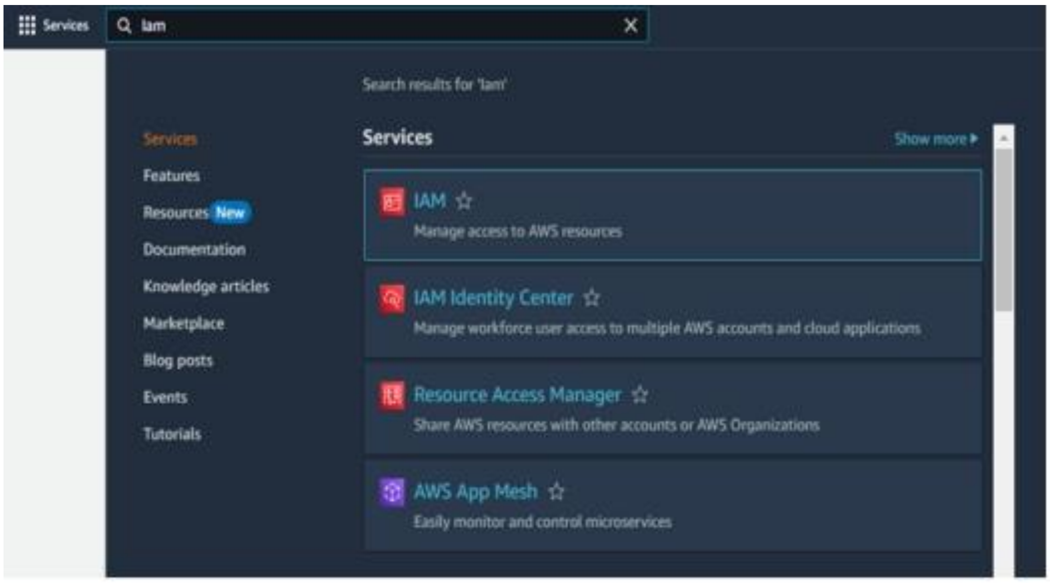
Subscriptions (1)

ID	Endpoint	Status	Protocol
[illegible]	[illegible]@gmail.com	Confirmed	EMAIL

5: IAM Role Setup

● **Activity 5.1:Create IAM Role.**

○ **In the AWS Console, go to IAM and create a new IAM Role for EC2 to interact with DynamoDB and SNS.**



Search results for 'iam'

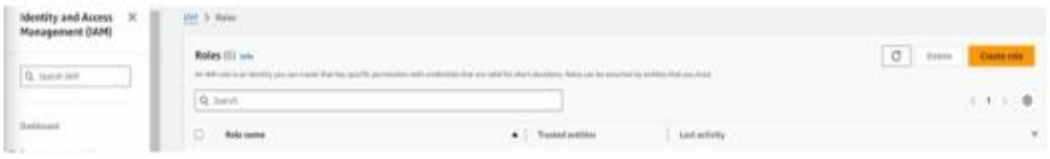
Services

IAM
Manage access to AWS resources

IAM Identity Center
Manage workforce user access to multiple AWS accounts and cloud applications

Resource Access Manager
Share AWS resources with other accounts or AWS Organizations

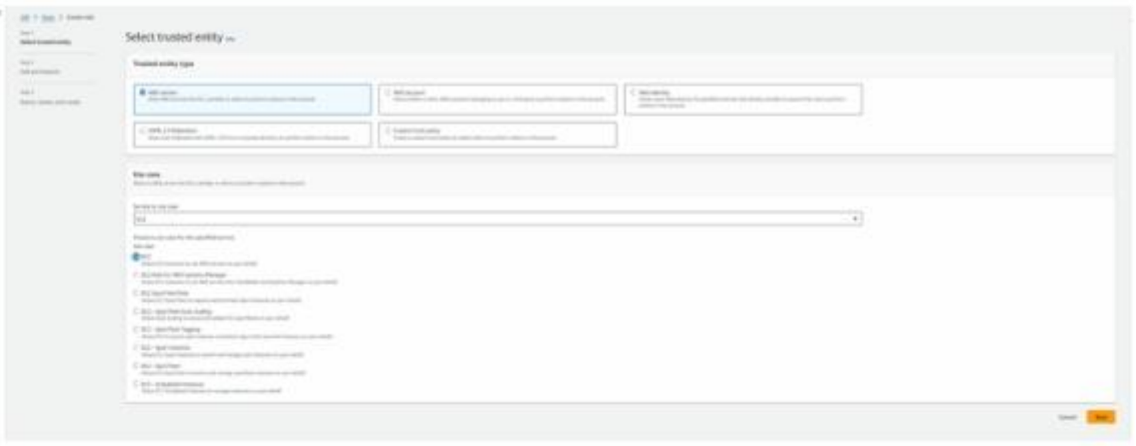
AWS App Mesh
Easily monitor and control microservices

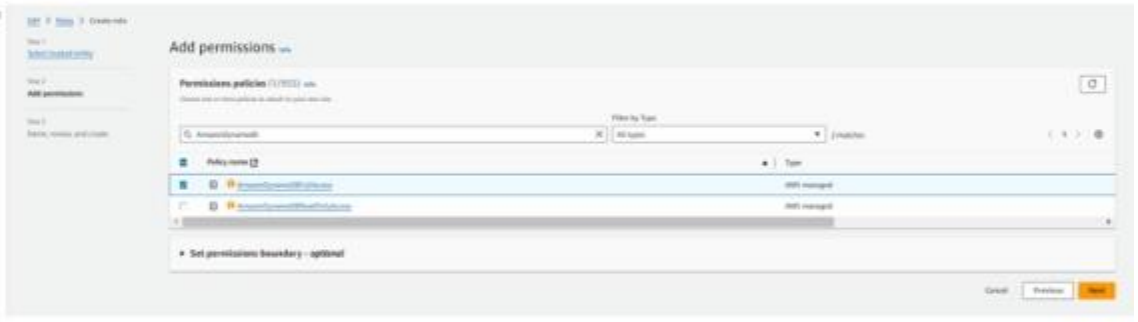


Identity and Access Management (IAM)

Roles (5) Info

Role name	Trusted entities	Last activity


Select trusted entity


Add permissions

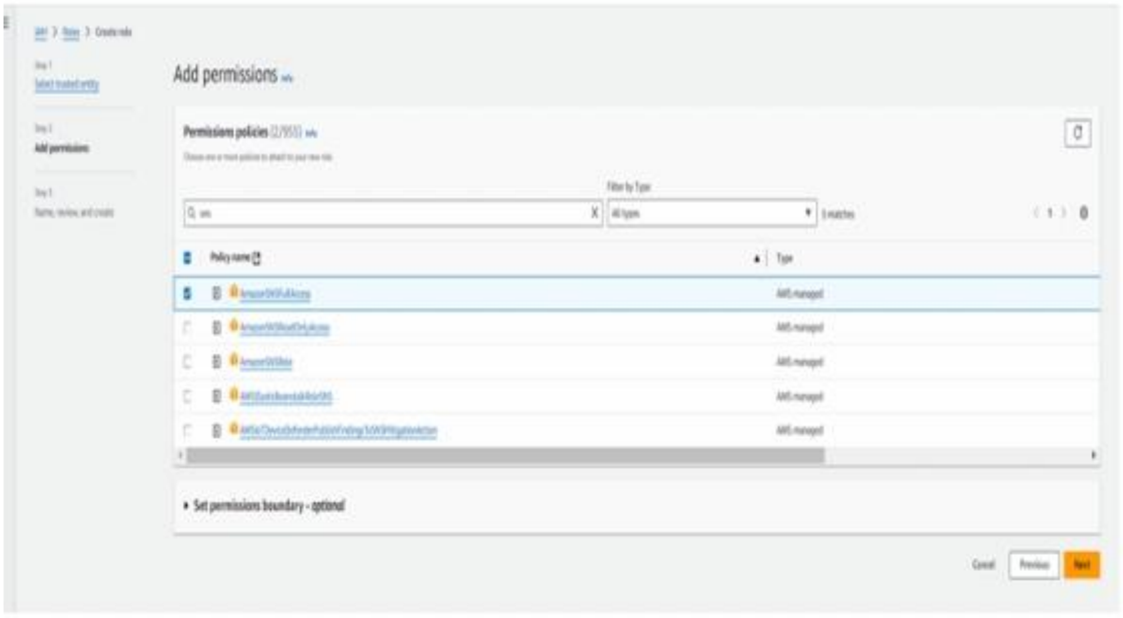
Permissions policies

Set permissions boundary - optional

- **Activity 5.2: Attach Policies.**

  Attach the following policies to the role:

  - AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.
  - AmazonSNSFullAccess: Grants EC2 the ability to send notifications via SNS.



IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Add permissions Info

Permissions policies (2/955) Info

[illegible]

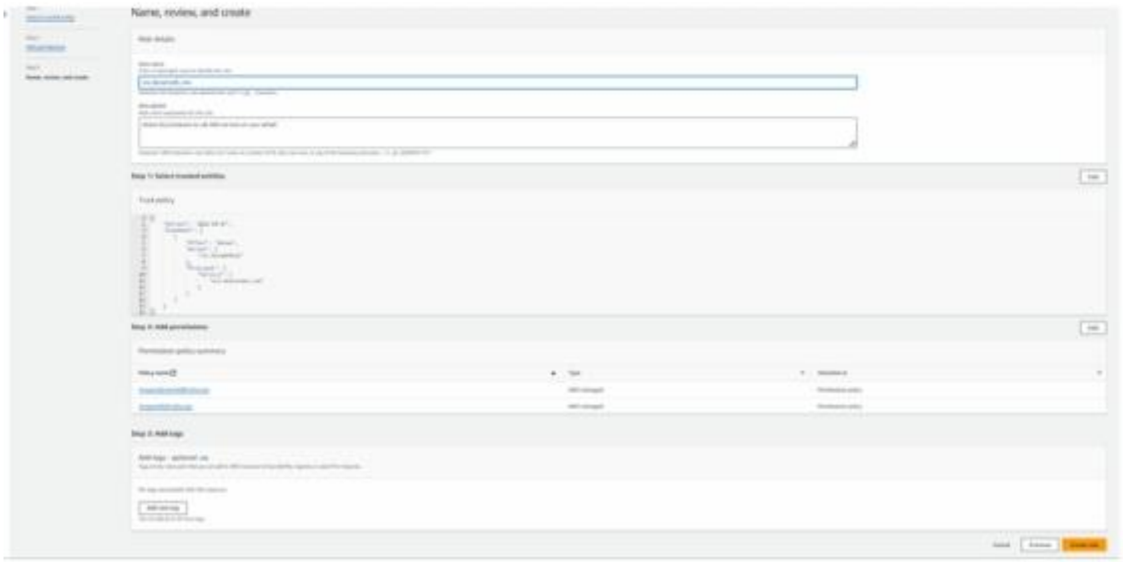
Filter by Type

sns | All types | 5 matches

	Policy name	Type
✓	AmazonSNSFullAccess	AWS managed
	AmazonSNSReadOnlyAccess	AWS managed
	AmazonSNSRole	AWS managed
	AWSElasticBeanstalkRoleSNS	AWS managed
	AWSIoTDeviceDefenderPublishFindingsToSNSMitigationAction	AWS managed

▸ Set permissions boundary - optional

Cancel | Previous | Next



Name, review, and create

[illegible]

Step 1: Select trusted entities

Trust policy

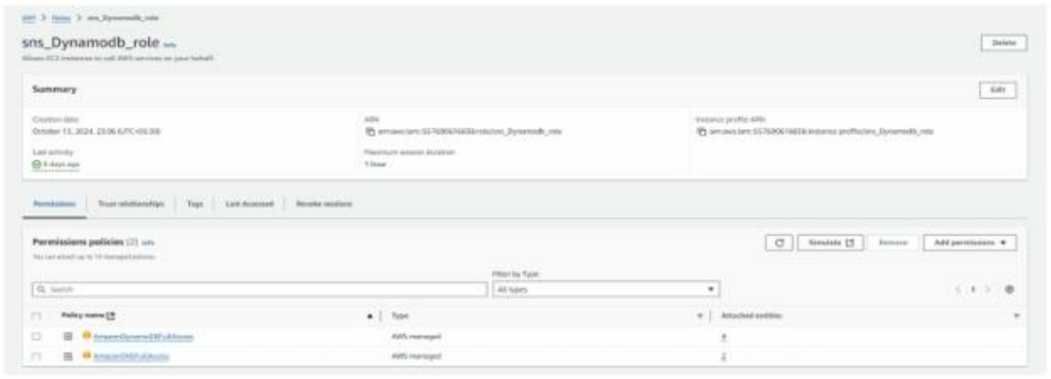
[illegible]

Step 2: Add permissions

[illegible]

Step 3: Add tags

[illegible]



IAM > Roles > sns_Dynamodb_role

sns_Dynamodb_role Info

Allows EC2 instances to call AWS services on your behalf.

Delete

Summary

Edit

Creation date: October 13, 2024, 23:06 (UTC+05:30)

Last activity: [illegible]

ARN: arn:aws:iam::[illegible]:role/sns_Dynamodb_role

Maximum session duration: 1 hour

Instance profile ARN: arn:aws:iam::[illegible]:instance-profile/sns_Dynamodb_role

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (2) Info

You can attach up to 10 managed policies.

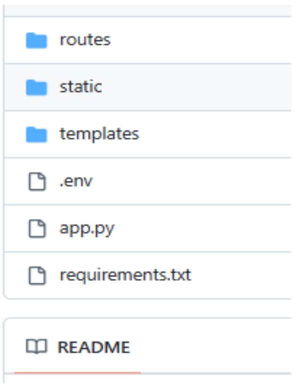
Simulate | Remove | Add permissions

Filter by Type: All types

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	1
AmazonSNSFullAccess	AWS managed	2

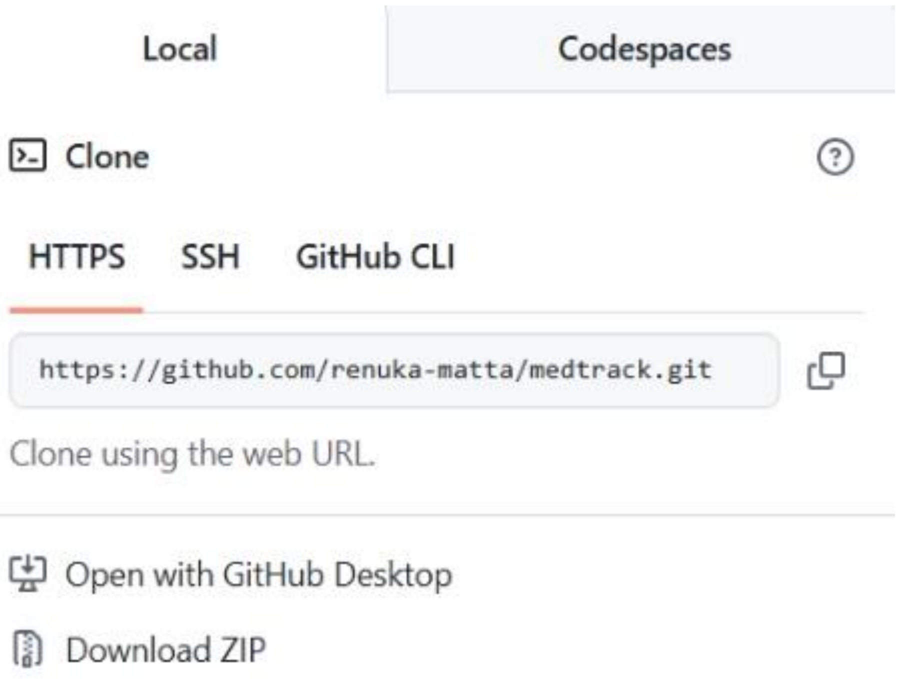
Milestone 6: EC2 Instance Setup

- **Note: Load your Flask app and Html files into GitHub repository.**



routes
static
templates
.env
app.py
requirements.txt

README



Local | Codespaces

Clone

HTTPS | SSH | GitHub CLI

```
https://github.com/renuka-matta/medtrack.git
```

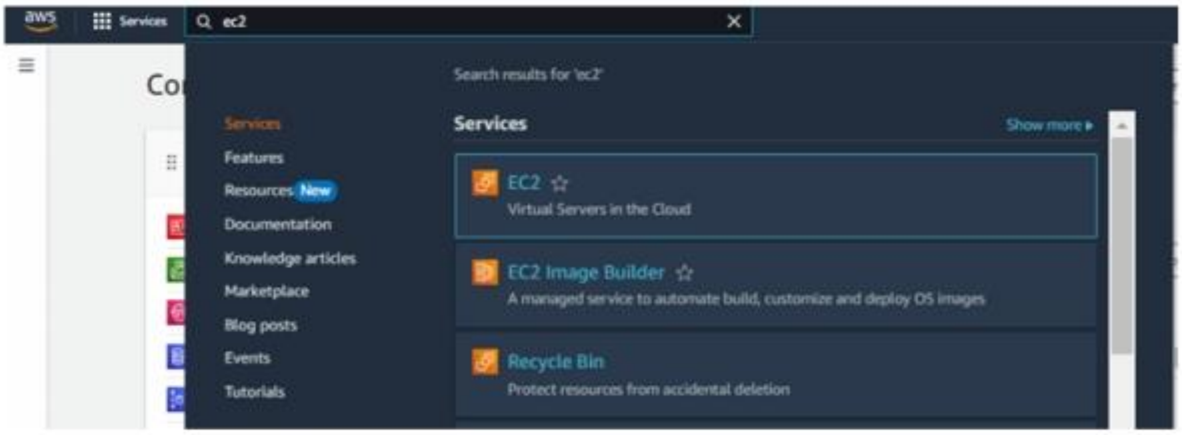
Clone using the web URL.

Open with GitHub Desktop

Download ZIP

- **Activity 6.1: Launch an EC2 instance to host the Flask application.**

  - **Launch EC2 Instance**
    - In the AWS Console, navigate to EC2 and launch a new instance.



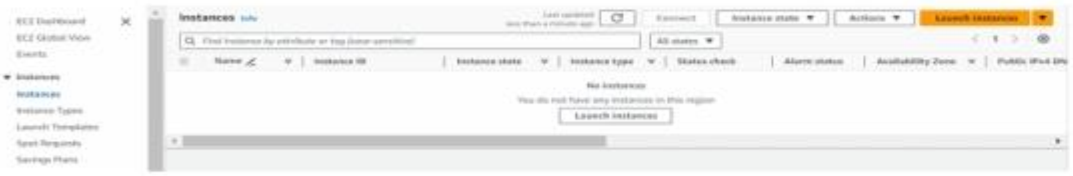
Services

EC2
Virtual Servers in the Cloud

EC2 Image Builder
A managed service to automate build, customize and deploy OS images

Recycle Bin
Protect resources from accidental deletion

- Click on Launch instance to launch EC2 instance

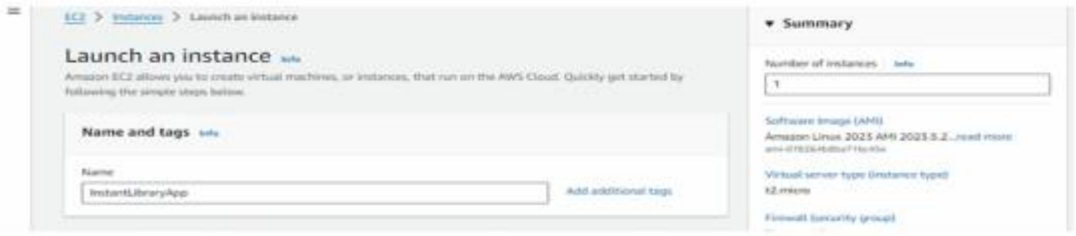


Instances Info

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS

No instances
You do not have any instances in this region
Launch instances


EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: InstantLibraryApp

Add additional tags

Summary

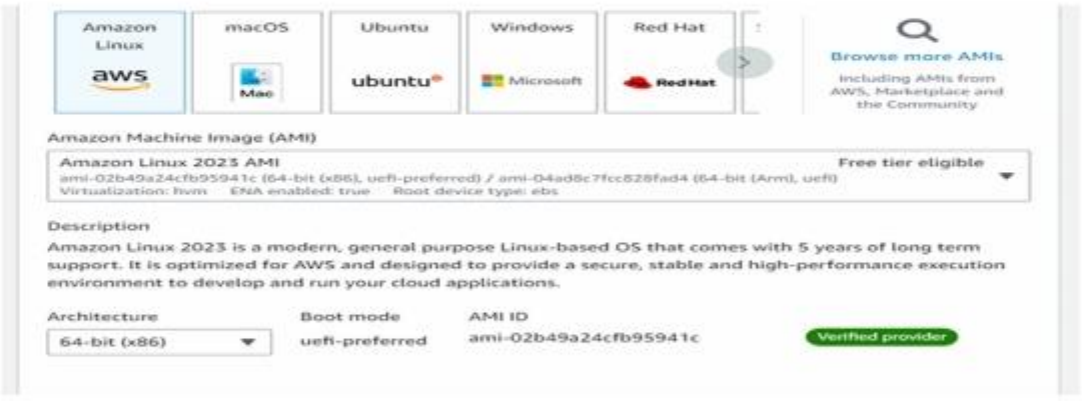
Number of instances: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2...read more

Virtual server type (instance type): t2.micro

Firewall (security group)

- Choose Amazon Linux 2 or Ubuntu as the AMI and t2.micro as the instance type (free-tier eligible).



Amazon Machine Image (AMI)

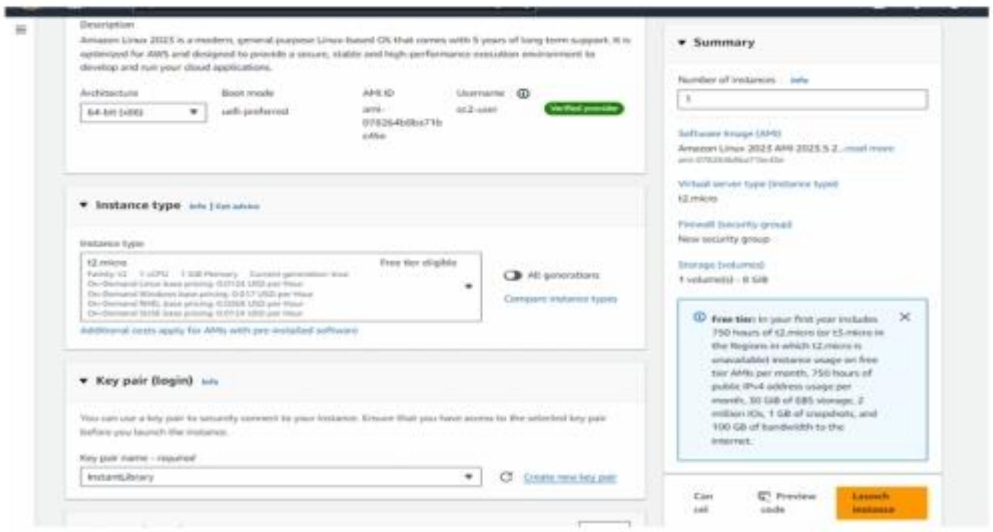
Amazon Linux 2023 AMI — Free tier eligible
ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)
Virtualization: hvm   ENA enabled: true   Root device type: ebs

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Architecture	Boot mode	AMI ID	
64-bit (x86)	uefi-preferred	ami-02b49a24cfb95941c	Verified provider

InstantLibrary.pem


Description
Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Architecture: 64-bit (x86)
Boot mode: uefi-preferred
AMI ID: ami-078264b8ba71bc45e
Username: ec2-user
Verified provider

Instance type Info | Get advice

Instance type
t2.micro — Free tier eligible
Family: t2 · 1 vCPU · 1 GiB Memory · Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0268 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour
Additional costs apply for AMIs with pre-installed software

All generations
Compare instance types

Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required
InstantLibrary
Create new key pair

Summary

Number of instances Info
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.5.2...read more
[illegible]

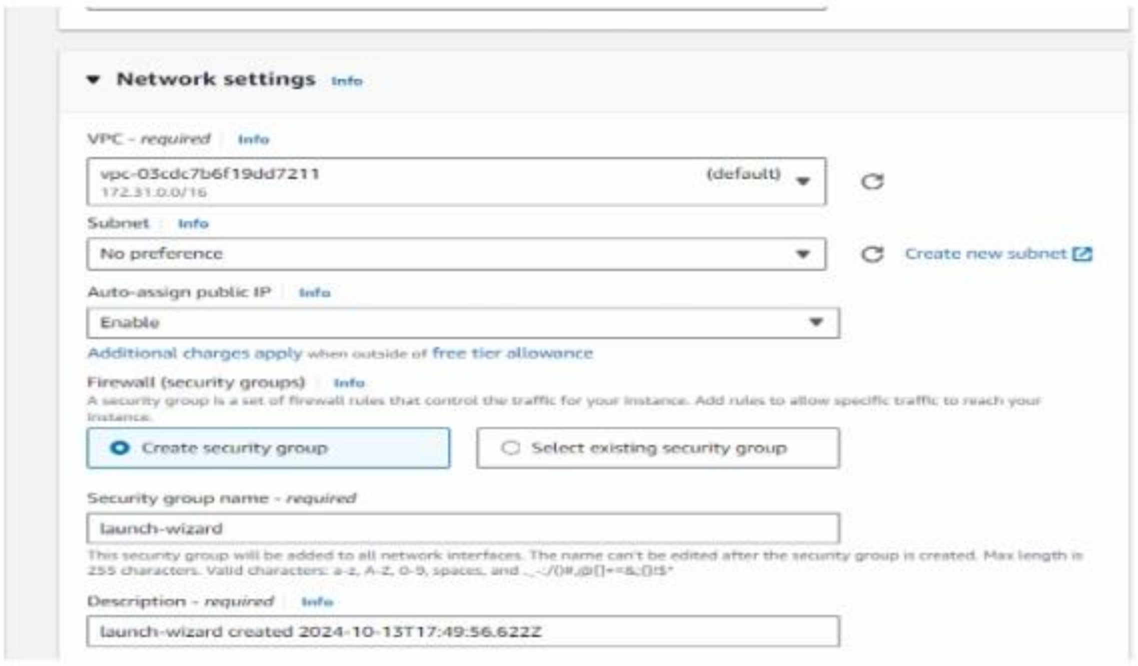
Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

- **Activity 6.2:Configure security groups for HTTP, and SSH access.**


Network settings Info

VPC - *required* Info
vpc-03cdc7b6f19dd7211 (default)
172.31.0.0/16

Subnet Info
No preference
Create new subnet

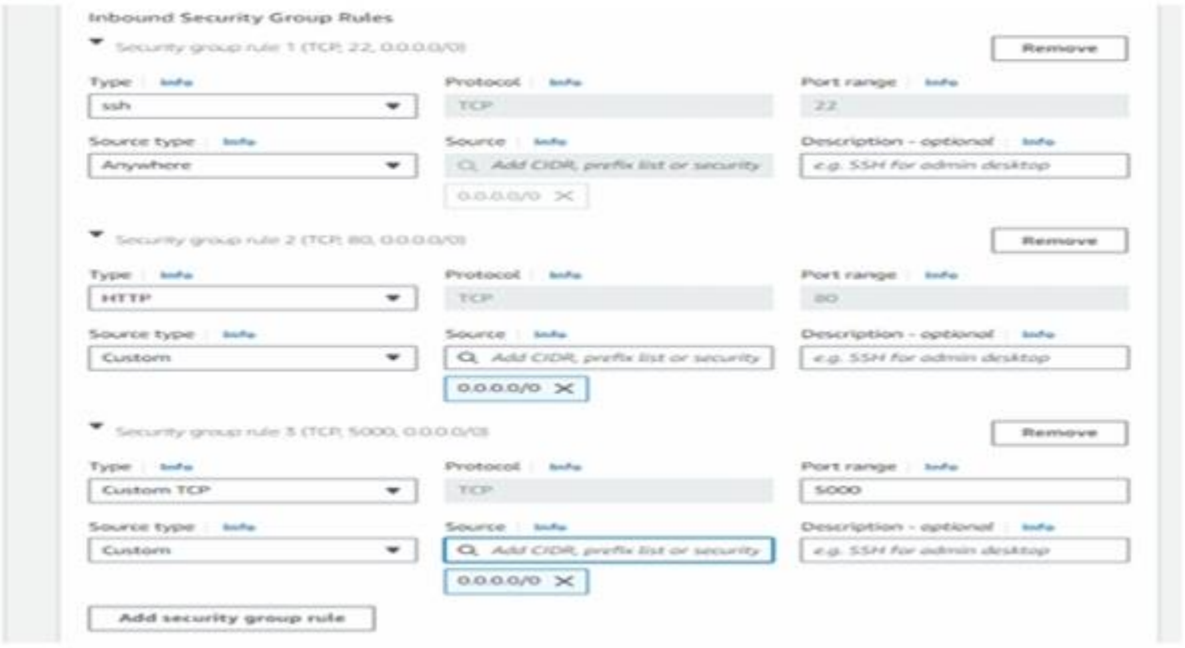
Auto-assign public IP Info
Enable
Additional charges apply when outside of free tier allowance

Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

[✓ Create security group] Select existing security group

Security group name - *required*
launch-wizard
This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required* Info
launch-wizard created 2024-10-13T17:49:56.622Z



Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0) — Remove

Type	Protocol	Port range
ssh	TCP	22

Source type	Source	Description - optional
Anywhere	Add CIDR, prefix list or security; 0.0.0.0/0	e.g. SSH for admin desktop

▼ Security group rule 2 (TCP, 80, 0.0.0.0/0) — Remove

Type	Protocol	Port range
HTTP	TCP	80

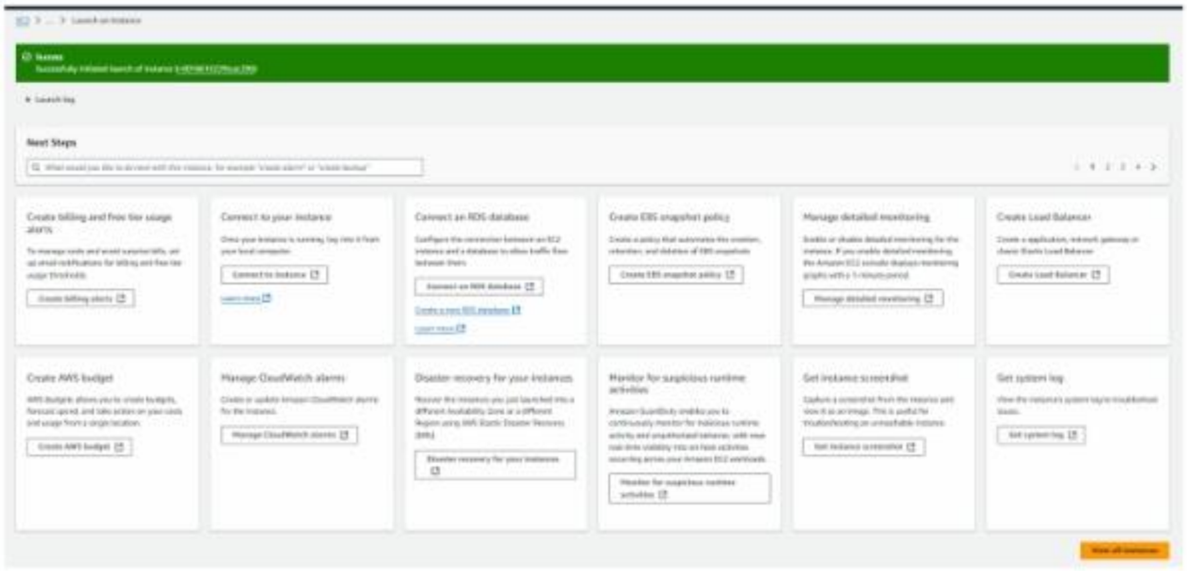
Source type	Source	Description - optional
Custom	Add CIDR, prefix list or security; 0.0.0.0/0	e.g. SSH for admin desktop

▼ Security group rule 3 (TCP, 5000, 0.0.0.0/0) — Remove

Type	Protocol	Port range
Custom TCP	TCP	5000

Source type	Source	Description - optional
Custom	Add CIDR, prefix list or security; 0.0.0.0/0	e.g. SSH for admin desktop

Add security group rule



EC2 > ... > Launch an instance

Success
Successfully initiated launch of instance [illegible]

Launch log

Next Steps

Create billing and free tier usage alerts — Create billing alerts

Connect to your instance — Connect to instance — Learn more

Connect an RDS database — Connect an RDS database — Create a new RDS database — Learn more

Create EBS snapshot policy — Create EBS snapshot policy

Manage detailed monitoring — Manage detailed monitoring

Create Load Balancer — Create Load Balancer

Create AWS budget — Create AWS budget

Manage CloudWatch alarms — Manage CloudWatch alarms

Disaster recovery for your instances — Disaster recovery for your instances

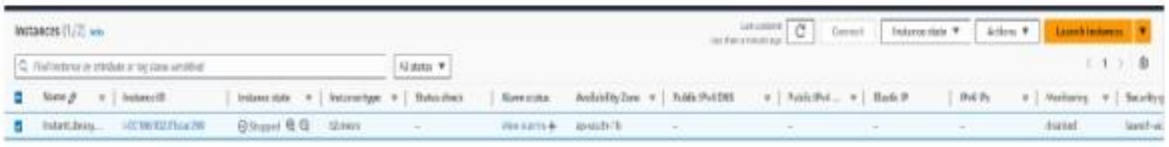
Monitor for suspicious runtime activities — Monitor for suspicious runtime activities

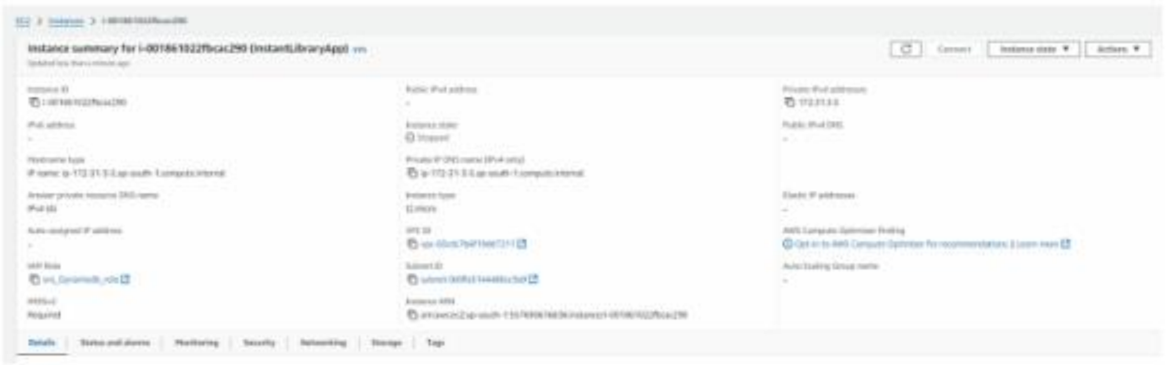
Get instance screenshot — Get instance screenshot

Get system log — Get system log

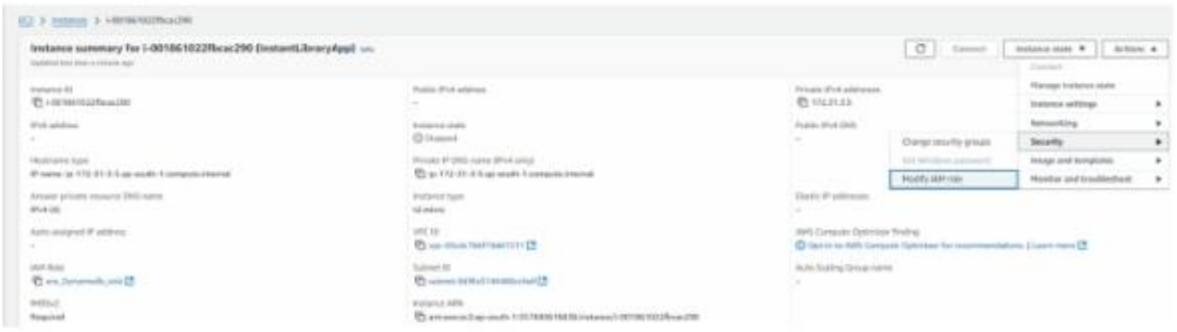
View all instances

- 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.

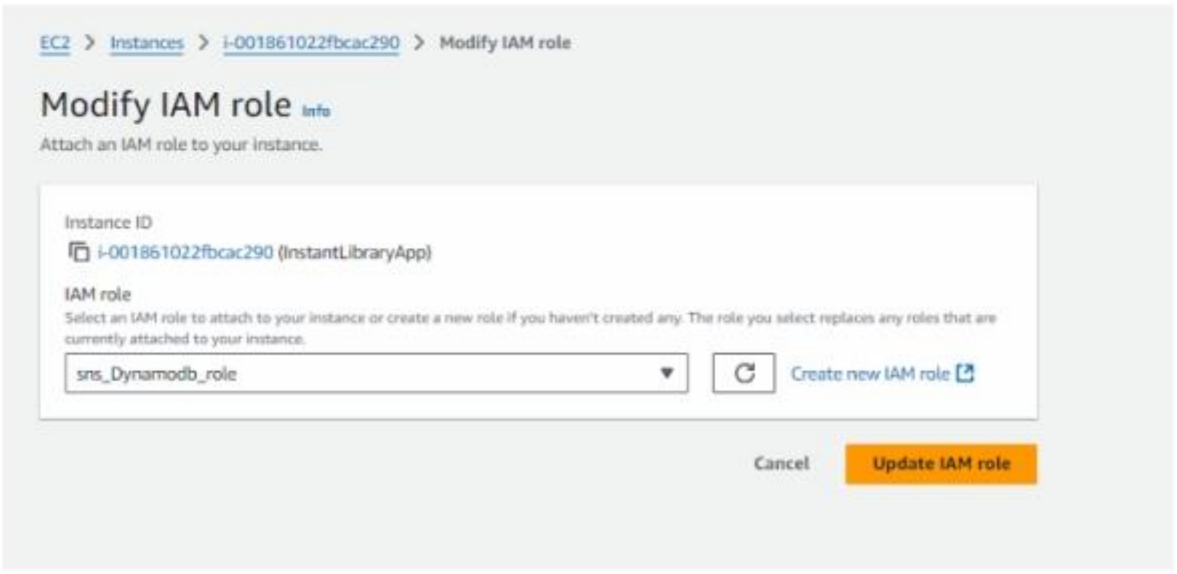




Instance summary for i-001861022fbcac290 (InstantLibraryApp)



Instance summary for i-001861022fbcac290 (InstantLibraryApp)



EC2 > Instances > i-001861022fbcac290 > Modify IAM role

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID

i-001861022fbcac290 (InstantLibraryApp)

IAM role

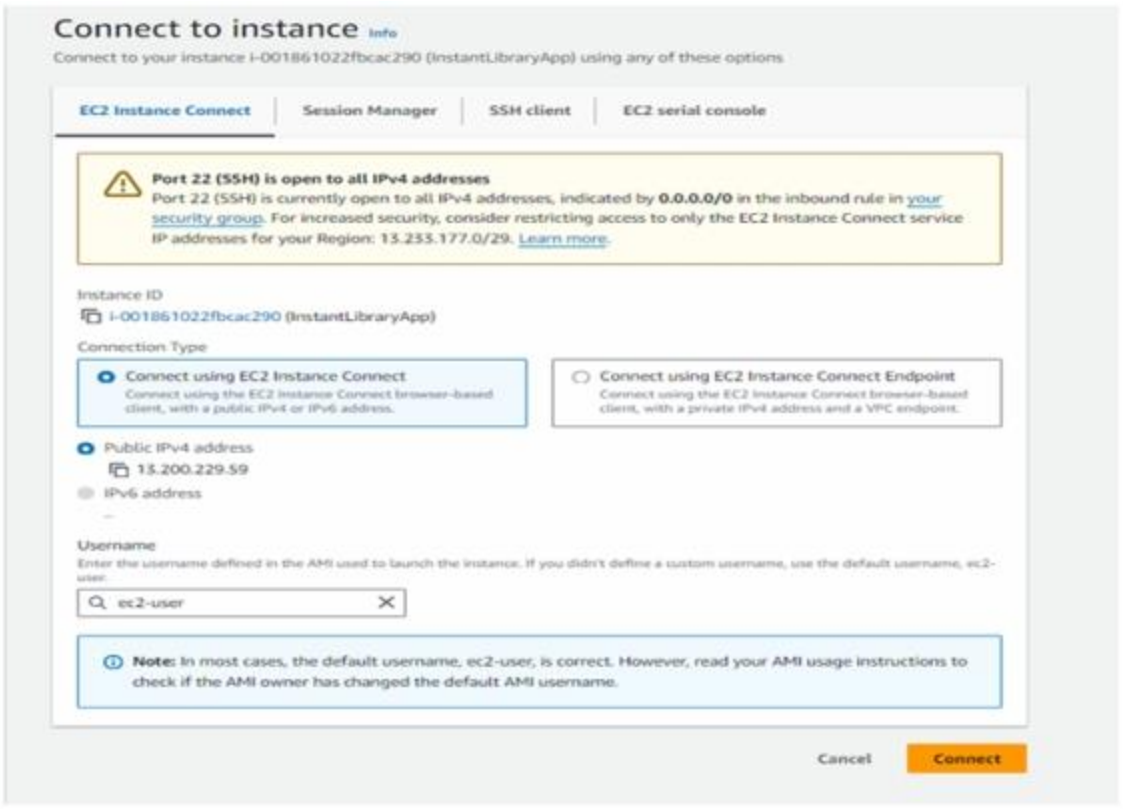
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

sns_Dynamodb_role

Create new IAM role

Cancel | Update IAM role

- **Now connect the EC2 with the files**



Connect to instance

Connect to your instance i-001861022fbcac290 (InstantLibraryApp) using any of these options

Port 22 (SSH) is open to all IPv4 addresses
Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in your security group. For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 13.233.177.0/29. Learn more.

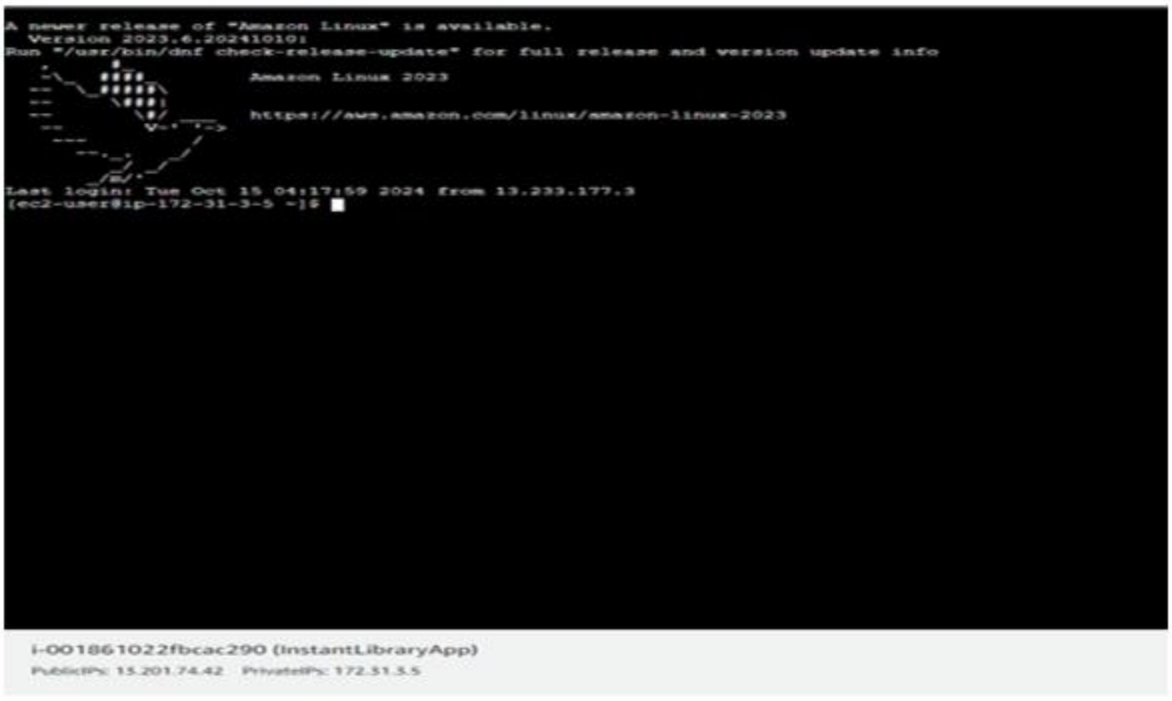
Instance ID
i-001861022fbcac290 (InstantLibraryApp)

Connection Type
● Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.
○ Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

● Public IPv4 address
13.200.229.59
○ IPv6 address
–

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.
ec2-user

Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.



```
A newer release of "Amazon Linux" is available.
  Version 2023.6.20241010:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$ 
```

i-001861022fbcac290 (InstantLibraryApp)
PublicIPs: 13.201.74.42   PrivateIPs: 172.31.3.5

Milestone 7: Deployment on EC2

Activity 7.1: Install Software on the EC2 Instance

Install Python3, Flask, and Git:

On Amazon Linux 2:

```
sudo yum update -y
sudo yum install python3 git
sudo pip3 install flask boto3
```

Verify Installations:

```
flask --version
git --version
```

Activity 7.2:Clone Your Flask Project from GitHub

Clone your project repository from GitHub into the EC2 instance using Git.

Run: 'git clone https://github.com/your-github-username/your-repository-name.git'

Note: change your-github-username and your-repository-name with your credentials

Here: 'git clone

https://github.com/KVeenaMadhuri/medtrack

This will download your project to the EC2 instance.

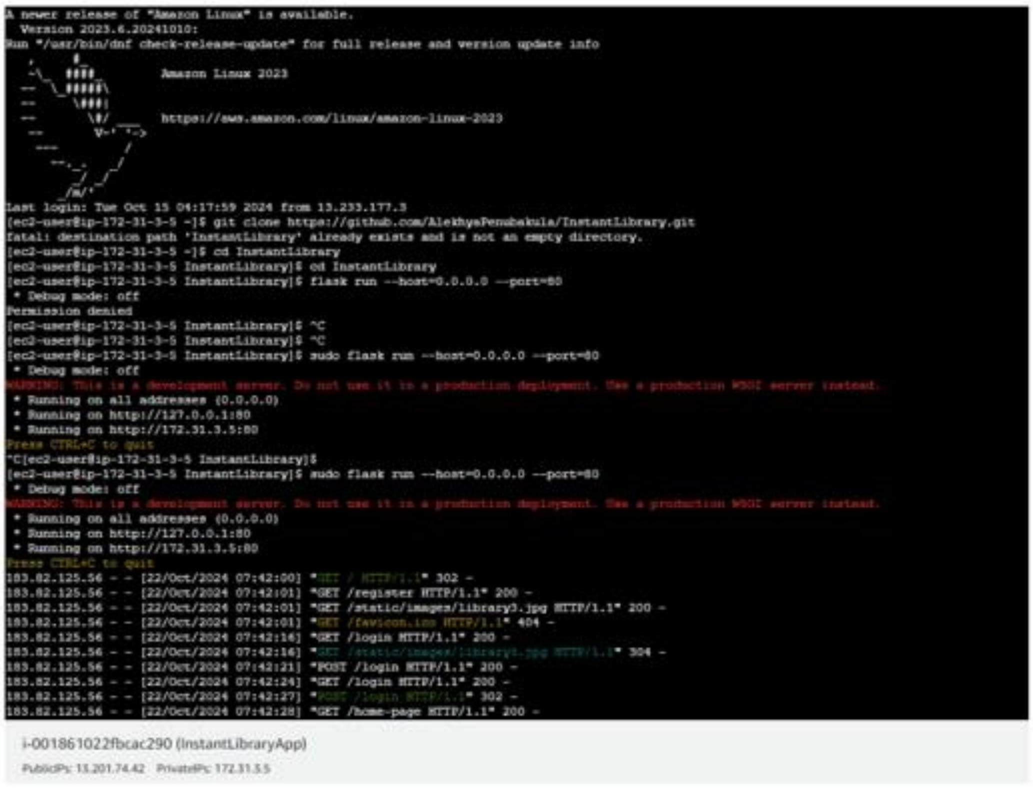
To navigate to the project directory, run the following command:

```
cd InstantLibrary
```

Once inside the project directory, configure and run the Flask application by executing the following command with elevated privileges:

Run the Flask Application

```
sudo flask run --host=0.0.0.0 --port=80
```



```
A newer release of "Amazon Linux" is available.
  Version 2023.6.20241010:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$ git clone https://github.com/AlekhyaPenubakula/InstantLibrary.git
fatal: destination path 'InstantLibrary' already exists and is not an empty directory.
[ec2-user@ip-172-31-3-5 ~]$ cd InstantLibrary
[ec2-user@ip-172-31-3-5 InstantLibrary]$ cd InstantLibrary
[ec2-user@ip-172-31-3-5 InstantLibrary]$ flask run --host=0.0.0.0 --port=80
 * Debug mode: off
Permission denied
[ec2-user@ip-172-31-3-5 InstantLibrary]$ ^C
[ec2-user@ip-172-31-3-5 InstantLibrary]$ ^C
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
^C[ec2-user@ip-172-31-3-5 InstantLibrary]$
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
183.82.125.56 - - [22/Oct/2024 07:42:00] "GET / HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /register HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /static/images/library3.jpg HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /favicon.ico HTTP/1.1" 404 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /static/images/library3.jpg HTTP/1.1" 304 -
183.82.125.56 - - [22/Oct/2024 07:42:21] "POST /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:24] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:27] "POST /login HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:28] "GET /home-page HTTP/1.1" 200 -
```

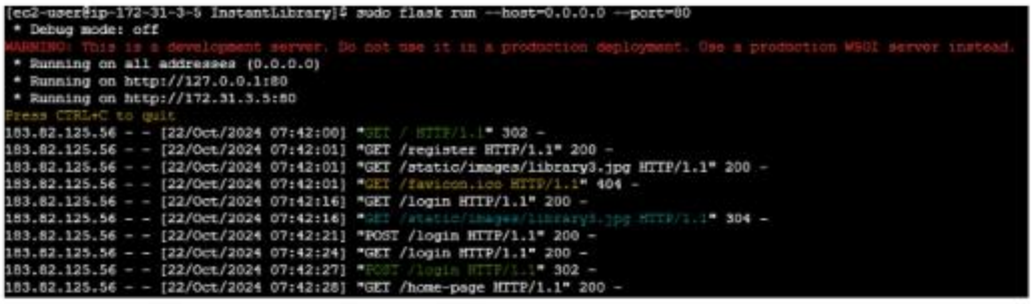
i-001861022fbcac290 (InstantLibraryApp)

PublicIPs: 13.201.74.42   PrivateIPs: 172.31.3.5

Verify the Flask app is running:

http://your-ec2-public-ip

- Run the Flask app on the EC2 instance



```
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
183.82.125.56 - - [22/Oct/2024 07:42:00] "GET / HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /register HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /static/images/library3.jpg HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /favicon.ico HTTP/1.1" 404 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /static/images/library3.jpg HTTP/1.1" 304 -
183.82.125.56 - - [22/Oct/2024 07:42:21] "POST /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:24] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:27] "POST /login HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:28] "GET /home-page HTTP/1.1" 200 -
```

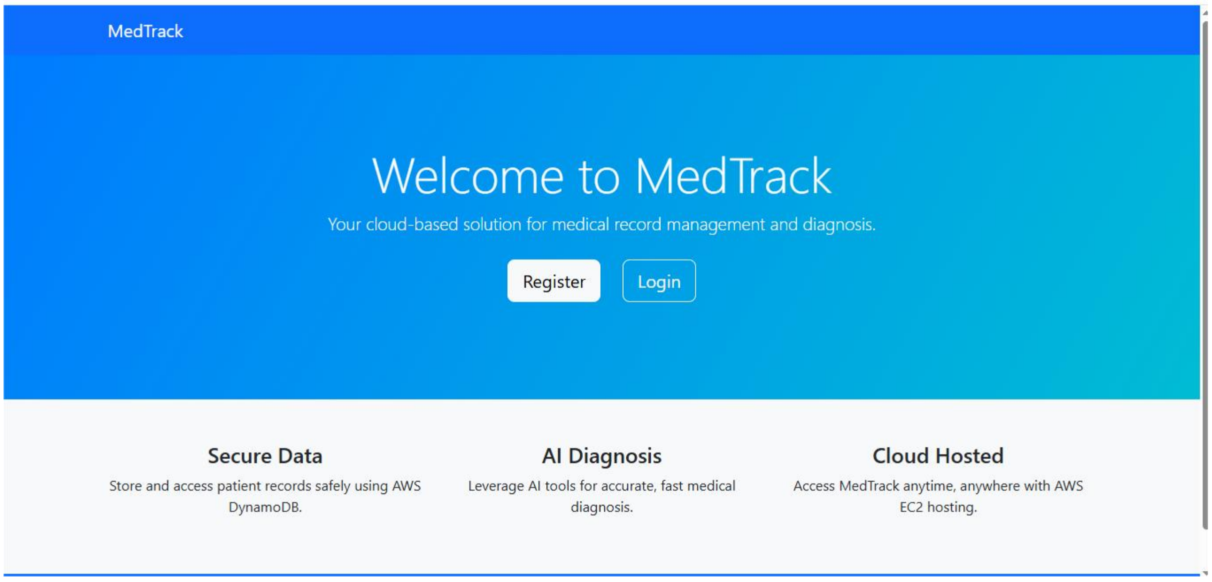
Access the Website Through

Running on http://127.0.0.1:5000

Milestone 8: Testing and Deployment

- **Activity 8.1: Conduct functional testing to verify user registration, login, book requests, and notifications.**

Home Page:



Welcome to MedTrack

Your cloud-based solution for medical record management and diagnosis.

Secure Data

Store and access patient records safely using AWS DynamoDB.

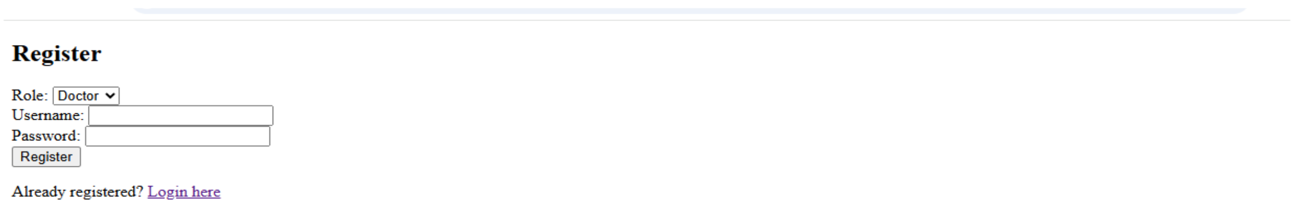
AI Diagnosis

Leverage AI tools for accurate, fast medical diagnosis.

Cloud Hosted

Access MedTrack anytime, anywhere with AWS EC2 hosting.

Register page:

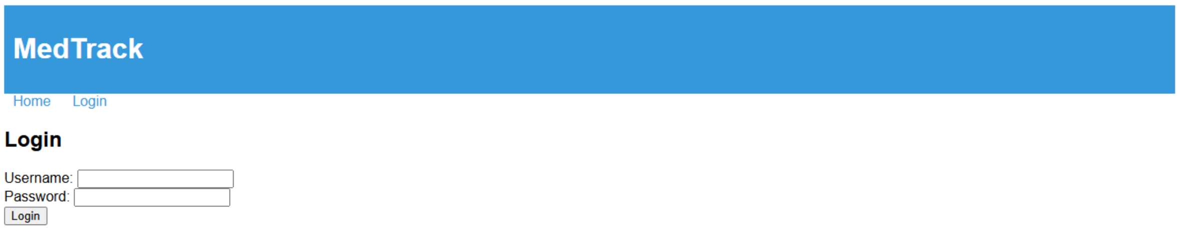


Register

Role: Doctor
Username:
Password:

Already registered? Login here

Login Page:



Google drive link: https://drive.google.com/file/d/1U5Ru1p4qodAHn1Ji7RRH61xnWj7iw5us/view?usp=drivesdk