

# TravelGo: A Cloud-Powered Real-Time Travel Booking Platform Using AWS:

## 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:

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.).

## 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:

TravelGo is a full-stack, cloud-based travel booking platform designed to simplify the process of reserving buses, trains, flights, and hotels through a unified interface. Built using Flask as the backend framework, the application is deployed on Amazon EC2 and leverages DynamoDB for efficient storage of user data and bookings. TravelGo allows users to register, log in, search for transportation and accommodation options, and book their travel with ease. Once a booking is confirmed or cancelled, users receive real-time email notifications powered by AWS Simple Notification Service (SNS), keeping them informed throughout their journey.

The platform's user-friendly interface supports dynamic seat selection for buses, hotel filtering based on preferences such as luxury or budget, and provides booking summaries along with centralized cancellation management. By combining cloud scalability, responsive design, and secure session handling, TravelGo delivers a seamless and real-time travel planning experience for users.

## Scenarios:

### Scenario 1: Hassle-Free Multi-Mode Travel Booking Experience

TravelGo offers users a unified platform to search and book buses, trains, flights, and hotels all in one place. For instance, a user planning a trip from Hyderabad to Bangalore can log in, select their preferred mode of transport, choose from available options, and proceed to booking. Flask manages the backend operations such as retrieving travel listings and processing user input in real-time. Hosted on AWS EC2, the platform remains responsive even during high-traffic hours like weekends or holiday seasons, allowing multiple users to browse and book without delay.

## Scenario 2: Real-Time Booking Confirmation with AWS SNS

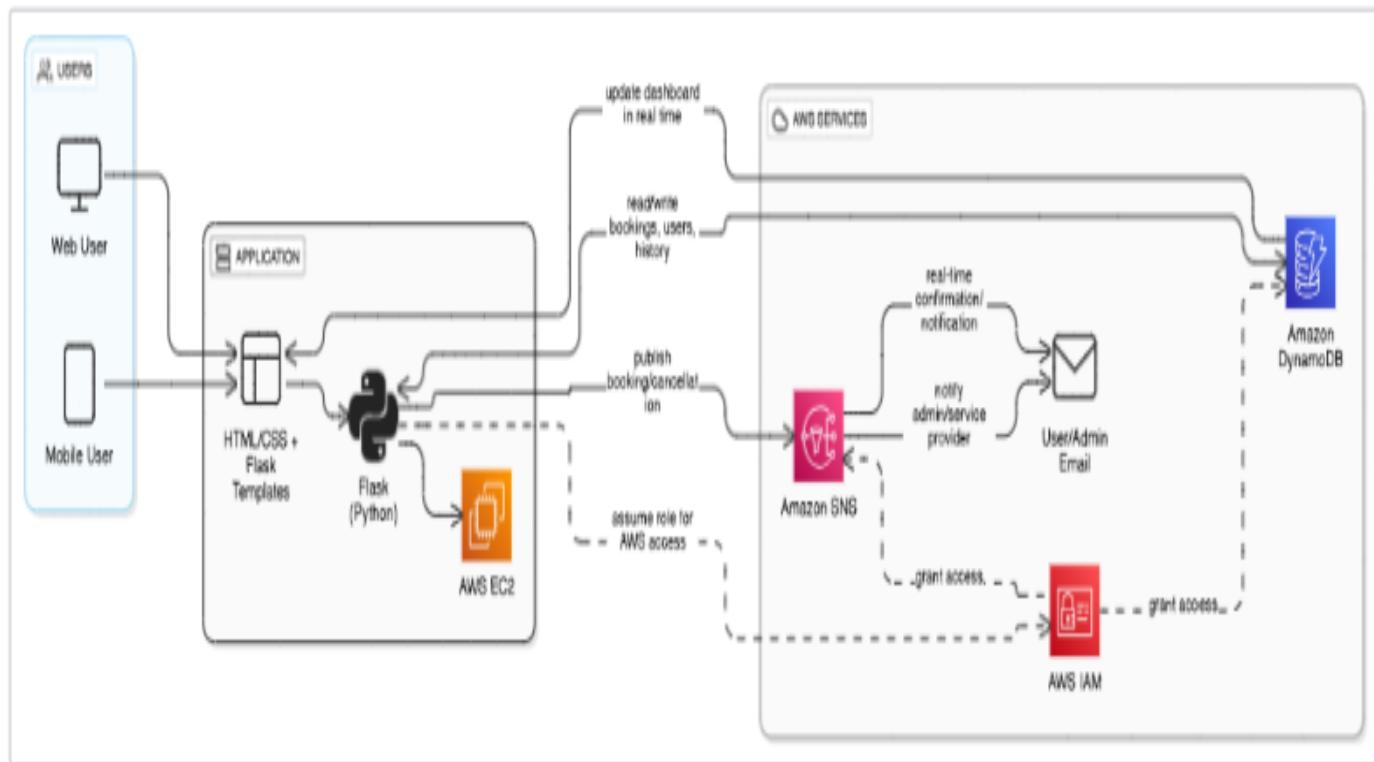
Once a booking is made—whether it's a train ticket or a hotel stay—TravelGo uses AWS SNS to instantly notify the user. For example, after a student books a hotel in Chennai, SNS sends a real-time email notification confirming the booking with all the relevant details. This notification is triggered from the Flask backend after the booking is successfully recorded in DynamoDB. Additionally, SNS can alert admin or service providers, ensuring transparency and real-time updates on every transaction.

## Scenario 3: Dynamic Dashboard with Personal Travel History

TravelGo features a dynamic user dashboard that displays all past and upcoming bookings for the logged-in user. For example, a user who has booked a flight and a hotel can view these bookings categorized by type, along with dates, price, and cancellation options. Flask fetches this data from AWS DynamoDB, which persistently stores all user bookings. The dashboard UI, powered by responsive HTML/CSS and Flask templates, ensures users can review or manage bookings anytime, from any device, with real-time updates and quick cancellation workflows supported.

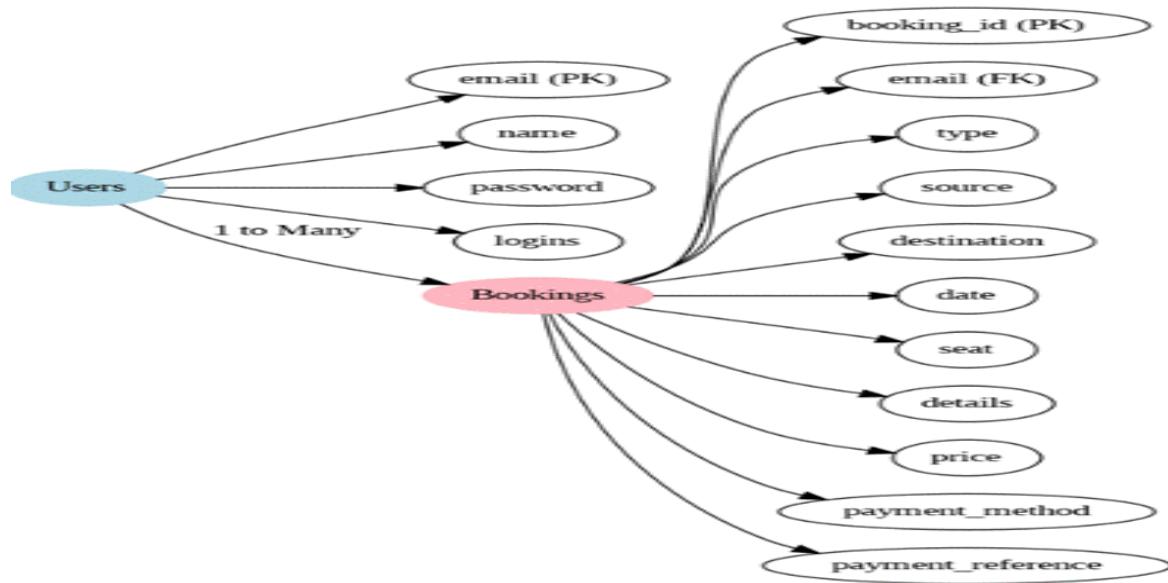
### Architecture:

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.



## 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.



## 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/latest/developerguide/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. Backend Development and Application 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.*
- *Log in to the 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 on 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.

## Milestone 1: Web Application Development and Setup

Backend Development and Application Setup focuses on establishing the core structure of the application. This includes configuring the backend framework, setting up routing, and integrating database connectivity. It lays the groundwork for handling user interactions, data management, and secure access.

### Important Instructions:

- Start by creating the necessary HTML pages and Flask routes (app.py) to build the core functionality of your application.
- During the initial development phase, store and retrieve data using Python dictionaries or lists locally. This will allow you to design, test, and validate your application logic without external database dependencies.
- Ensure your app runs smoothly with local data structures before integrating any cloud services.

### Post Troven Access Activation:

- Once Troven Labs access is provided (valid for 3 hours), you must immediately proceed with Milestone 1 of your Guided Project instructions.
- At this point, modify your app.py and replace local dictionary/list operations with AWS services (such as DynamoDB, RDS, or others as per project requirements).
- Using the temporary credentials provided by Troven Labs, securely connect your application to AWS resources
- Since the AWS configuration is lightweight and already instructed in the milestones, you should be able to complete the cloud integration efficiently within the allotted time.

### LOCAL DEPLOYMENT: File explorer:



```
TRAVELGO
├── static
│   └── templates
│       ├── bookings.html
│       ├── bus.html
│       ├── confirm_bus_details...
│       ├── confirm_flight_detail...
│       ├── confirm_hotel_detail...
│       ├── confirm_train_detail...
│       ├── dashboard.html
│       ├── flight.html
│       ├── home.html
│       ├── hotel.html
│       ├── index.html
│       ├── login.html
│       ├── order.html
│       ├── register.html
│       ├── select_seats.html
│       ├── train.html
│       └── welcome.html
└── venv
    ├── app.py
    ├── README.md
    └── requirements.txt
```



**Description:** Organize the project with HTML templates for each feature (e.g., login, wishlist, quiz, checkout) under the templates folder and manage backend logic in application.py.

### Activity 3.1: Develop the Backend Using Flask

#### Import libraries:

```
from flask import Flask, render_template, request, redirect, session, url_for, jsonify, flash
import pymongo
import boto3
from boto3.dynamodb.conditions import Key, Attr
from werkzeug.security import generate_password_hash, check_password_hash
from datetime import datetime
from decimal import Decimal
import uuid
import random
import urllib.parse
from bson.objectid import ObjectId
```

**Description:** Import essential Flask modules for web handling, Boto3 for AWS integration, Werkzeug for password hashing, and date time for timestamp management.

#### Flask App Initialization:

```
# Initialize the Flask application
application = Flask(__name__)
application.secret_key = 'your_secret_key'
```

**Description:** Initialize the Flask application and set a secret key to securely manage user sessions and form data.

#### Database Configuration

```
#Initialize dynamodb resource
dynamodb = boto3.resource('dynamodb', region_name='us-east-1')

#Define dynamodb tables
users_table = dynamodb.Table('travelgo_users')
trains_table = dynamodb.Table('trains')
bookings_table = dynamodb.Table('bookings')
```

**Description:** Connect to DynamoDB using Boto3 and define references to the User Table, trains and Bookings table

#### Routes for Core Functionalities:

##### Home and registration routes:

```
@app.route('/')
def home():
    return render_template('home.html', logged_in='email' in session)

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        email = request.form['email']
        username = request.form['username']
        password = generate_password_hash(request.form['password'])
        users.insert_one({'email': email, 'username': username, 'hashed_password': password, "login_count": 0})
        return redirect(url_for('login'))
    return render_template('register.html')
```



**Description:** Create the home and registration routes, where the registration route securely hashes user passwords and stores user data in DynamoDB upon form submission.

#### Login routes:

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        user = users.find_one({'email': email})
        if user and check_password_hash(user['hashed_password'], password):
            session['email'] = email
            session['username'] = user['username']
            users.update_one({'email': email}, {'$inc': {'login_count': 1}})
            return redirect(url_for('welcome'))
        else:
            return render_template('login.html', error="Invalid credentials.")
    return render_template('login.html')
```

**Description:** Implement the user login route to validate credentials using DynamoDB and securely manage session data while updating the user's login count.

#### Welcome:

```
@app.route('/welcome')
def welcome():
    if 'email' not in session:
        return redirect(url_for('login'))
    return render_template('welcome.html', username=session.get('username'))
```

**Description:** Implement the welcome route to greet users upon login, fetch personalized user details from DynamoDB, and provide access to key TravelGo features through a user-friendly interface.

#### Dashboard:

```
@app.route('/dashboard')
def dashboard():
    if 'email' not in session:
        return redirect(url_for('login'))
    username = session.get('username', 'User')
    user_email = session['email']
    bookings = []
    for b in bookings_collection.find({'user_email': user_email}):
        booking_type = b.get('booking_type', 'Unknown').capitalize()
        if b.get('booking_type') in ['bus', 'train', 'flight']:
            details = f'{b.get("source")} → {b.get("destination")} @ {b.get("travel_date")}'
        elif b.get('booking_type') == 'hotel':
            details = f'{b.get("hotel_name")} from {b.get("checkin_date")}'
        else:
            details = "Details not available"
        bookings.append({
            'type': booking_type,
            'details': details,
            'date': b.get('booking_date', 'N/A')[0:16].replace('T', ' '),
            'booking_id': str(b.get('_id'))
        })
    return render_template('dashboard.html', username=username, bookings=bookings)
```



**Description:** Secure the user dashboard and implement route to store them in DynamoDB with item details and a timestamp.

### Bus,Train,Flight,Hotel routes:

```
@app.route('/bus')
def bus_booking_page():
    return render_template('bus.html') if 'email' in session else redirect(url_for('login'))

@app.route('/train')
def train_booking_page():
    return render_template('train.html') if 'email' in session else redirect(url_for('login'))

@app.route('/flight')
def flight_booking_page():
    return render_template('flight.html') if 'email' in session else redirect(url_for('login'))

@app.route('/hotel')
def hotel_booking_page():
    if 'email' not in session:
        return redirect(url_for('login'))
    return render_template('hotel.html')
```

**Description:** Implement add routes to allow admins to insert new bus, train, flight, and hotel details into DynamoDB, ensuring proper data validation and seamless integration with the booking system.

### Bookings route:

```
@app.route('/bookings')
def bookings():
    if 'email' not in session:
        return redirect(url_for('login'))

    email = session['email']
    bookings = list(bookings_collection.find({'user_email': email}))
    return render_template('bookings.html', bookings=bookings)
```

**Description:** Implement booking routes to capture user booking details for buses, trains, flights, and hotels, store them in DynamoDB, and trigger AWS SNS notifications upon successful booking.



## Select Seats Routes:

```
@app.route('/select_seats', methods=['GET', 'POST'])
def select_seats():
    if 'pending_booking' not in session:
        return "No pending booking found.", 400

    booking = session['pending_booking']

    if request.method == 'POST':
        selected = request.form.get('selected_seats', '')
        if selected:
            booking['selected_seats'] = selected
            booking['booking_date'] = datetime.now().isoformat()
            db.bookings.insert_one(booking)
            session.pop('pending_booking', None)
        return redirect(url_for('bookings')) # or dashboard

    return render_template('select_seats.html', booking=booking)
```

**Description:** Create seat selection routes that display available seats dynamically, allow users to choose their preferred seats, and update the selected status in DynamoDB upon confirmation.

## Initiating Flask app:

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

**Description:** Start the Flask application on host **0.0.0.0** and port **5000** in debug mode to enable live reloads and detailed error logs during development.

## Milestone 2 : AWS Account Setup

### Important Notice: Use Troven Labs for AWS Access

Students are strictly advised not to create their own AWS accounts, as doing so may incur charges. Instead, we have set up a dedicated section called “Labs” on the Troven platform, which provides temporary and cost-free access to AWS services.

Once your website is locally deployed and fully functional, you must proceed with integrating AWS services only through the Troven Labs environment. This ensures secure, controlled access to AWS resources without any risk of personal billing.

All steps involving AWS (such as deploying to EC2, connecting to DynamoDB, or using SNS) must be carried out within the Troven Labs platform, as we've configured temporary credentials for each student.



**Reminder: You must complete the Web Development task before gaining access to Troven. Once accessed, the AWS Console via Troven is available for only 3 hours—please plan your work accordingly.**

Please follow the provided guidelines and access AWS exclusively through Troven to avoid unnecessary issues.

Please refer the below link -

<https://drive.google.com/file/d/1HzWc7AMJ2BrxhV-uaw5s0vWtcd-28qgl/view?usp=sharing>

## AWS Account Setup and Login:

**This is for your understanding only, please refrain from creating an AWS account. A temporary account will be provided via Troven.**

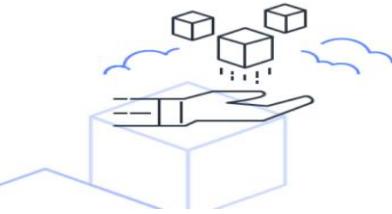
- Go to the AWS website (<https://aws.amazon.com/>).
- Click on the "Create an AWS Account" button.
- Follow the prompts to enter your email address and choose a password.
- Provide the required account information, including your name, address, and phone number.
- Enter your payment information. (Note: While AWS offers a free tier, a credit card or debit card is required for verification.)
- Complete the identity verification process.
- Choose a support plan (the basic plan is free and sufficient for starting).
- Once verified, you can sign in to your new AWS accounts.

The screenshot shows the AWS registration process. At the top, there's a navigation bar with links like 'About AWS', 'Contact Us', 'Support', 'English', 'My Account', 'Sign In', and 'Complete Sign Up'. Below the navigation, the main heading is 'Complete your AWS registration'. A sub-headline says 'Millions of customers are using AWS cloud solutions to build applications with increased flexibility, scalability, security, and reliability'. There's a prominent 'Complete sign-up' button. Three callout boxes provide additional information: 'AWS Free Tier' (describing free access to EC2, S3, etc.), 'Customer success stories' (showcasing how companies use AWS), and 'Contact us' (offering support). A small orange speech bubble icon is also visible.



**Explore Free Tier products with a new AWS account.**

To learn more, visit [aws.amazon.com/free](https://aws.amazon.com/free).



## Sign up for AWS

Root user email address  
Used for account recovery and as described in the [AWS Privacy Notice](#)

AWS account name  
Choose a name for your account. You can change this name in your account settings after you sign up.

**Verify email address**

OR

[Sign in to an existing AWS account](#)

## Log in to the AWS Management Console

- After setting up your account, log in to the [AWS Management Console](#).



### Sign in

#### Root user

Account owner that performs tasks requiring unrestricted access. [Learn more](#)

#### IAM user

User within an account that performs daily tasks. [Learn more](#)

Root user email address

**Next**

By continuing, you agree to the [AWS Customer Agreement](#) or other agreement for AWS services, and the [Privacy Notice](#). This site uses essential cookies. See our [Cookie Notice](#) for more information.

## AI Use Case Explorer

Discover AI use cases, customer success stories, and expert-curated implementation plans

[Explore now >](#)

## Milestone 3 : DynamoDB Database Creation and Setup

Database Creation and Setup involves initializing a cloud-based NoSQL database to store and manage application data efficiently. This step includes defining tables, setting primary keys, and configuring read/write capacities. It ensures scalable, high-performance data storage for seamless backend operations.

### Navigate to the DynamoDB:

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



aws Services  X

Search results for 'dyn'

Services

- Features
- Resources **New**
- Documentation
- Knowledge articles
- Marketplace
- Blog posts
- Events
- Tutorials

Services

DynamoDB ☆  
Managed NoSQL Database

Top features

Tables Imports from S3 Explore Items Clusters Reserved Capacity

Amazon DocumentDB ☆  
Fully-managed MongoDB-compatible database service

CloudFront ☆  
Global Content Delivery Network

Athena ☆  
Serverless interactive analytics service

Show more ▶

Features

Settings

DynamoDB feature

Clusters

DynamoDB feature

Show more ▶

DynamoDB X [DynamoDB](#) > Dashboard

## Dashboard

Dashboard

Tables

Explore items

PartiQL editor

Backups

Exports to S3

Imports from S3

Integrations **New**

Reserved capacity

Settings

DAX

Clusters

Subnet groups

Parameter groups

Events

Alarms (0) [Info](#) Manage in CloudWatch [\[ \]](#)

Find alarms

Alarm name [\[ \]](#) Status

No custom alarms

DAX clusters (0) [Info](#) View details

Find clusters

Cluster name [▲](#) Status [▼](#)

No clusters

No clusters to display

Create cluster

Create resources

Create an Amazon DynamoDB table for fast and predictable database performance at any scale. [Learn more \[ \]](#)

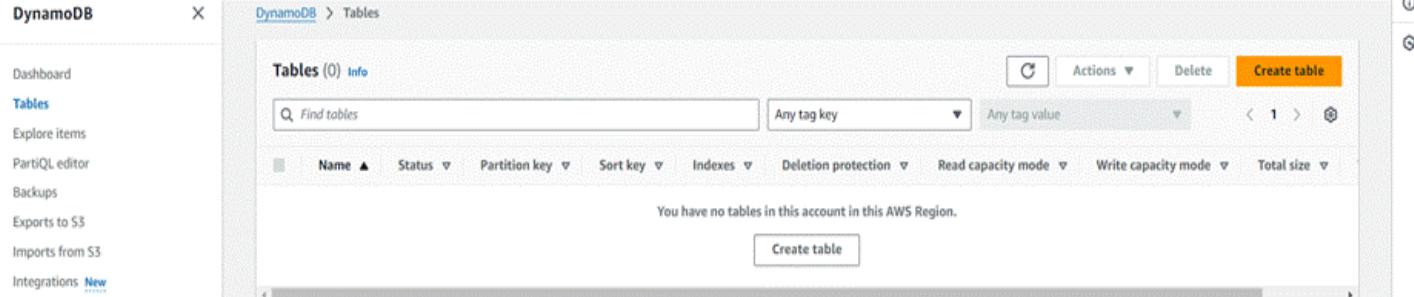
Create table

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

Create DAX cluster

What's new [\[ \]](#)

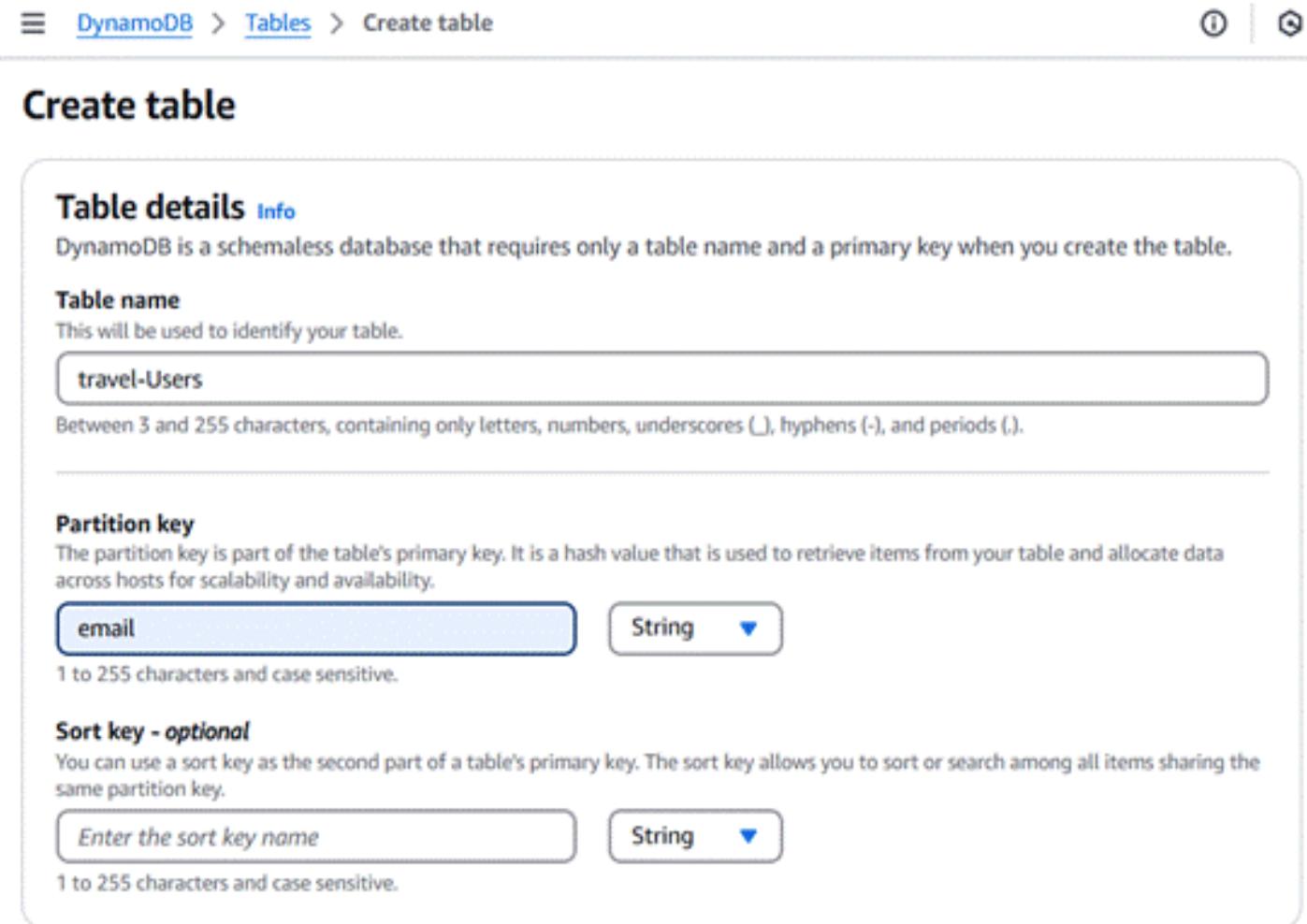
SEP 19 AWS Cost Management now provides purchase recommendations for Amazon DynamoDB...



The screenshot shows the AWS DynamoDB 'Tables' page. On the left, there is a navigation sidebar with links like 'Dashboard', 'Tables' (which is selected and highlighted in blue), 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', and 'Integrations'. The main content area has a heading 'Tables (0) Info' with a search bar labeled 'Find tables'. Below the search bar are several filter dropdowns: 'Any tag key', 'Any tag value', 'Name' (sorted ascending), 'Status', 'Partition key', 'Sort key', 'Indexes', 'Deletion protection', 'Read capacity mode', 'Write capacity mode', and 'Total size'. A message at the bottom states 'You have no tables in this account in this AWS Region.' There is a prominent orange 'Create table' button at the bottom right.

## Create an DynamoDB table for storing data

Create a travel-Users table for storing registered user details with partition key “Email” with type String and click on create tables.



The screenshot shows the 'Create table' wizard. At the top, there is a breadcrumb navigation: 'DynamoDB > Tables > Create table'. To the right of the breadcrumb are three small icons: a help icon, a refresh icon, and a back/forward icon. The main section is titled 'Create table' and contains a sub-section titled 'Table details' with a blue 'Info' link. It says: 'DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.' Under 'Table name', it says: 'This will be used to identify your table.' with a text input field containing 'travel-Users'. Below it, a note says: 'Between 3 and 255 characters, containing only letters, numbers, underscores (\_), hyphens (-), and periods (.).'. Under 'Partition key', it says: '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.' with a text input field containing 'email' and a dropdown menu set to 'String'. Below it, a note says: '1 to 255 characters and case sensitive.' Under 'Sort key - optional', it says: '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.' with a text input field containing 'Enter the sort key name' and a dropdown menu set to 'String'. Below it, a note says: '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](#)

Follow the same steps to create a Bookings for storing booking records with email as the partition key and booking\_id as the sort key.

## 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.

Bookings

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.

booking\_id

String ▾

1 to 255 characters and case sensitive.

### Table settings

Default settings

The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose 'Customize settings'.

Customize settings

Use these advanced features to make DynamoDB work better for your needs.

| 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](#)

**Tables (1/7) [Info](#)**

2 matched results [Any tag key](#) [Any tag value](#)

[Actions ▾](#) [Delete](#) [Create table](#)

[Name = travel-Users](#) [X](#) [or ▾](#) [Name = Bookings](#) [X](#) [Clear filters](#)

|                          | Name                         | Status                                    | Partition key | Sort key | Indexes | Replication Regions                   | Deletion protection                   | Favorite                               | Read capacity mode | Write capac |
|--------------------------|------------------------------|---|---------------|----------|---------|---------------------------------------|---------------------------------------|--|--------------------|-------------|
| <input type="checkbox"/> | <a href="#">Bookings</a>     | <span style="color: green;">Active</span> | email (\$)    | -        | 0 0     | <span style="color: grey;">Off</span> | <span style="color: grey;">Off</span> | <span style="color: grey;">Star</span> | On-demand          | On-demand   |
| <input type="checkbox"/> | <a href="#">travel-Users</a> | <span style="color: green;">Active</span> | email (\$)    | -        | 0 0     | <span style="color: grey;">Off</span> | <span style="color: grey;">Off</span> | <span style="color: grey;">Star</span> | On-demand          | On-demand   |

## Milestone 4 : SNS Notification Setup:

Amazon SNS is a fully managed messaging service that enables real-time notifications through channels like SMS, email, or app endpoints. You create topics, configure subscriptions, and integrate SNS into your app to send notifications based on specific events.

### SNS Topics for email notifications

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

Search results for 'sns'

Services
Show more ▶

- Features
- Resources New
- Documentation
- Knowledge articles
- Marketplace

 **Simple Notification Service** ☆  
SNS managed message topics for Pub/Sub

 **Route 53 Resolver**  
Resolve DNS queries in your Amazon VPC and on-premises network.

Amazon SNS

**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.

**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)                                   |  | Edit | Delete | Publish message | Create topic |
|--|--|------|--------|-----------------|--------------|
| Name   |  | ▲    | ▼      | Type            | ARN          |
| No topics<br>To get started, create a topic. |  |      |        |                 |              |
| <a href="#">Create topic</a>                 |  |      |        |                 |              |

- Choose Standard type for general notification use cases and Click on 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
- Subscription protocols: SQS

Standard

- Best-effort message ordering
- At-least once message delivery
- Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

#### Name

BookingConfirmation

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.

[Cancel](#)

[Create topic](#)

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



Amazon SNS > Topics > BookingConfirmation

**New Feature**  
Amazon SNS now supports High Throughput FIFO topics. [Learn more](#)

**Topic BookingConfirmation created successfully.**  
You can create subscriptions and send messages to them from this topic.

**BookingConfirmation**

**Details**

|      |   |              |              |
|------|---|--------------|--------------|
| Name | BookingConfirmation                                     | Display name | -            |
| ARN  | arn:aws:sns:ap-south-1:557690616836:BookingConfirmation | Topic owner  | 557690616836 |
| Type | Standard  |              |              |

**Subscriptions** | Access policy | Data protection policy | Delivery policy (HTTP/S) | Delivery status logging | Encryption | Tags | Integrations

**Subscriptions (0)**

| ID  | Endpoint | Status | Protocol |
|---|----------|--------|----------|
| No subscriptions found<br>You don't have any subscriptions to this topic. |          |        |          |

[Create subscription](#)

## Subscribe users and admin

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

### Create subscription

**Details**

**Topic ARN**  
 [X](#)

**Protocol**  
The type of endpoint to subscribe  
 [▼](#)

**Endpoint**  
An email address that can receive notifications from Amazon SNS.

**Info** 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.

[Cancel](#) [Create subscription](#)



After subscription request for the mail confirmation

**BookingConfirmation**

[Edit](#) [Delete](#) [Publish message](#)

| Details      |   |
|--------------|---|
| Name         | BookingConfirmation                                     |
| ARN          | arn:aws:sns:ap-south-1:557690616836:BookingConfirmation |
| Type         | Standard  |
| Display name | -   |
| Topic owner  | 557690616836  |

[Subscriptions](#) [Access policy](#) [Data protection policy](#) [Delivery policy \(HTTP/S\)](#) [Delivery status logging](#) [Encryption](#) [Tags](#) [Integrations](#)

**Subscriptions (1)**

| ID                   | Endpoint                  | Status               | Protocol |
|----------------------|---------------------------|----------------------|----------|
| Pending confirmation | instantlibrary2@gmail.com | Pending confirmation | EMAIL    |

[Edit](#) [Delete](#) [Request confirmation](#) [Confirm subscription](#) [Create subscription](#)

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

AWS Notifications <no-reply@sns.amazonaws.com>  
to me ▾Mon, Apr 14,

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

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.

## BookingConfirmation

[Edit](#) [Delete](#) [Publish message](#)

**Details**

|      |   |              |              |
|------|---|--------------|--------------|
| Name | BookingConfirmation                                     | Display name |              |
| ARN  | arn:aws:sns:ap-south-1:557690616836:BookingConfirmation | Topic owner  | 557690616836 |
| Type | Standard  |              |              |

---

[Subscriptions](#) [Access policy](#) [Data protection policy](#) [Delivery policy \(HTTP/S\)](#) [Delivery status logging](#) [Encryption](#) [Tags](#) [Integrations](#)

**Subscriptions (1)**

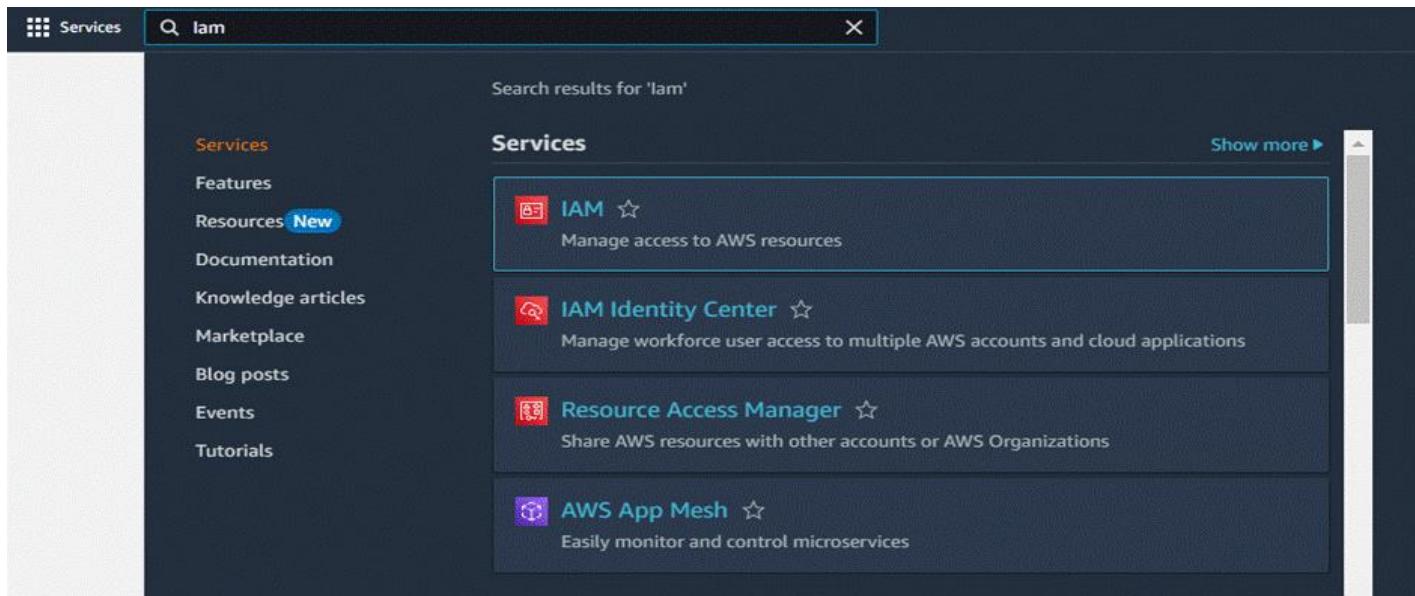
| ID                                   | Endpoint                  | Status    | Protocol |
|--------------------------------------|---------------------------|-----------|----------|
| da5ad931-7e8f-4ac8-b771-ec2e0cfbaef3 | instantlibrary2@gmail.com | Confirmed | EMAIL    |

## Milestone 5 : IAM Role Setup:

IAM (Identity and Access Management) role setup involves creating roles that define specific permissions for AWS services. To set it up, you create a role with the required policies, assign it to users or services, and ensure the role has appropriate access to resources like EC2, S3, or RDS. This allows controlled access and ensures security best practices in managing AWS resources.

### Create IAM Role.

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



The screenshot shows the AWS Lambda service page. At the top, there is a search bar with the text 'Iam'. Below the search bar, there is a sidebar with links to 'Services', 'Features', 'Resources New', 'Documentation', 'Knowledge articles', 'Marketplace', 'Blog posts', 'Events', and 'Tutorials'. The main content area displays 'Search results for 'Iam'' and a list of services under 'Services'. The services listed are: '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), and 'AWS App Mesh' (Easily monitor and control microservices). Each service item includes a small icon and a star rating.

Identity and Access Management (IAM)

IAM > Roles

### Roles (6) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

| Role name                | Trusted entities | Last activity       |
|--------------------------|------------------|---------------------|
| EC2 Role                 | AmazonEC2        | 2023-09-01 12:00:00 |
| AmazonSNSFullAccess      | AmazonSNS        | 2023-09-01 12:00:00 |
| AmazonDynamoDBFullAccess | AmazonDynamoDB   | 2023-09-01 12:00:00 |
| AmazonSQSFullAccess      | AmazonSQS        | 2023-09-01 12:00:00 |
| AmazonS3FullAccess       | AmazonS3         | 2023-09-01 12:00:00 |
| AmazonLambdaFullAccess   | AmazonLambda     | 2023-09-01 12:00:00 |

Step 1 Select trusted entity Info

Step 2 Add permissions

Step 3 Name, review, and create

Select trusted entity type

AWS service Allow AWS services like S3, Lambda, or others to perform actions in the account.

AWS account Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Web identity Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

EC2

Choose a use case for the specified service.

Use case

EC2 - EC2 instances to call AWS services on your behalf.

EC2 Role for AWS Systems Manager

EC2 - EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.

EC2 Spot Fleet

EC2 - EC2 Spot Fleet to request and terminate Spot Instances on your behalf.

EC2 - Spot Fleet Auto Scaling

EC2 - Auto Scaling to create and update EC2 spot fleets on your behalf.

EC2 - Spot Fleet Tagging

EC2 - EC2 launch spot instances and attach tags to the launched instances on your behalf.

EC2 - Spot Instances

EC2 - EC2 Spot Instances to launch and manage spot instances on your behalf.

EC2 - Spot Fleet

EC2 - EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.

EC2 - Scheduled Instances

EC2 - Scheduled Instances to manage instances on your behalf.

## 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

### Add permissions info

Step 2: Add permissions

Step 3: Name, review, and create

**Permissions policies (1/955) info**

Choose one or more policies to attach to your new role.

Filter by Type: All types

Policy name: AmazonDynamoDB

| Policy name  | Type        | Attached as        |
|--|-------------|--------------------|
| <input checked="" type="checkbox"/>  AmazonDynamoDBFullAccess | AWS managed | Permissions policy |
| <input type="checkbox"/>  AmazonDynamoDBReadOnlyAccess        | AWS managed | Permissions policy |

Set permissions boundary - optional

Cancel Previous Next

Name, review, and create

Role details

Role name: sns\_Dynamodb\_role

Description: Allows EC2 instances to call AWS services on your behalf.

Step 1: Select trusted entities

Trust policy:

```
1: { "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": "AWS", "Action": "sts:AssumeRole" } ] }
```

Step 2: Add permissions

Permissions policy summary

| Policy name   | Type        | Attached as        |
|---|-------------|--------------------|
| <input type="checkbox"/>  AmazonDynamoDBFullAccess | AWS managed | Permissions policy |
| <input type="checkbox"/>  AmazonSNSFullAccess      | AWS managed | Permissions policy |

Step 3: Add tags

Add tags - optional info

No tags associated with the resource.

Add new tag

Cancel Previous Create role

IAM > Roles > sns\_Dynamodb\_role

**sns\_Dynamodb\_role info**

Allows EC2 instances to call AWS services on your behalf.

Delete

**Summary**

|  |   |  |
|--|---|--|
| Creation date: October 13, 2024, 23:06 (UTC+05:30) | ARN: arn:aws:iam::557690616836:role/sns_Dynamodb_role | Instance profile ARN: arn:aws:iam::557690616836:instance-profile/sns_Dynamodb_role |
| Last activity: 6 days ago                          | Maximum session duration: 1 hour                      |  |

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

**Permissions policies (2) info**

You can attach up to 10 managed policies.

Filter by Type: All types

| Policy name   | Type        | Attached entities |
|---|-------------|-------------------|
| <input type="checkbox"/>  AmazonDynamoDBFullAccess | AWS managed | 4                 |
| <input type="checkbox"/>  AmazonSNSFullAccess      | AWS managed | 2                 |



## Milestone 6 : EC2 Instance Setup

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

To set up a public EC2 instance, choose an appropriate Amazon Machine Image (AMI) and instance type. Ensure the security group allows inbound traffic on necessary ports (e.g., HTTP/HTTPS for web applications). After launching the instance, associate it with an elastic IP for consistent public access, and configure your application or services to be publicly accessible.

The screenshot shows a GitHub repository named "travelgo" which is public. The repository has 1 branch and 0 tags. The commit history shows the following activity:

| Commit           | Message   | Date                 |
|------------------|---|----------------------|
| SIRI             | Updated app.py with DynamoDB integration and fixed issues | a482cf8 · 4 days ago |
| static/images    | Added project files                                       | 4 days ago           |
| templates        | Added project files                                       | 4 days ago           |
| venv             | Added project files                                       | 4 days ago           |
| README.md        | first commit  | 4 days ago           |
| app.py           | Updated app.py with DynamoDB integration and fixed issues | 4 days ago           |
| requirements.txt | Added project files                                       | 4 days ago           |

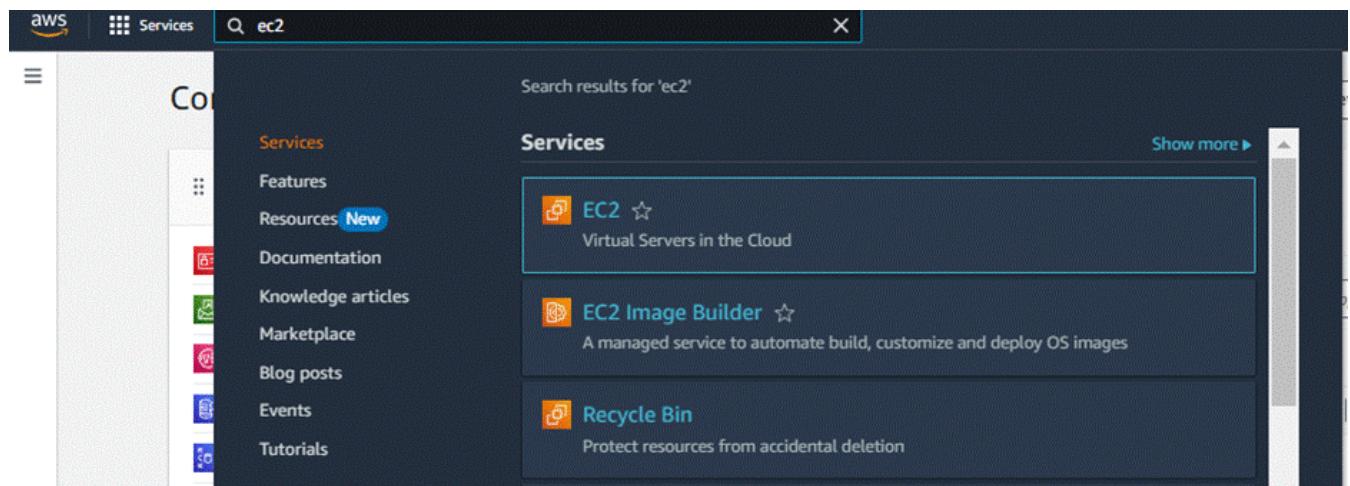
The screenshot shows the same GitHub repository "travelgo". The commit history is identical to the previous screenshot. On the right side, there is a "Clone" section with the following options:

- Local
- Codespaces
- Clone (selected)
- HTTPS (selected)
- SSH
- GitHub CLI
- Clone using the web URL: <https://github.com/ladi-siri18/travelgo.git>
- Open with GitHub Desktop
- Download ZIP

## Load your Project Files to Github:

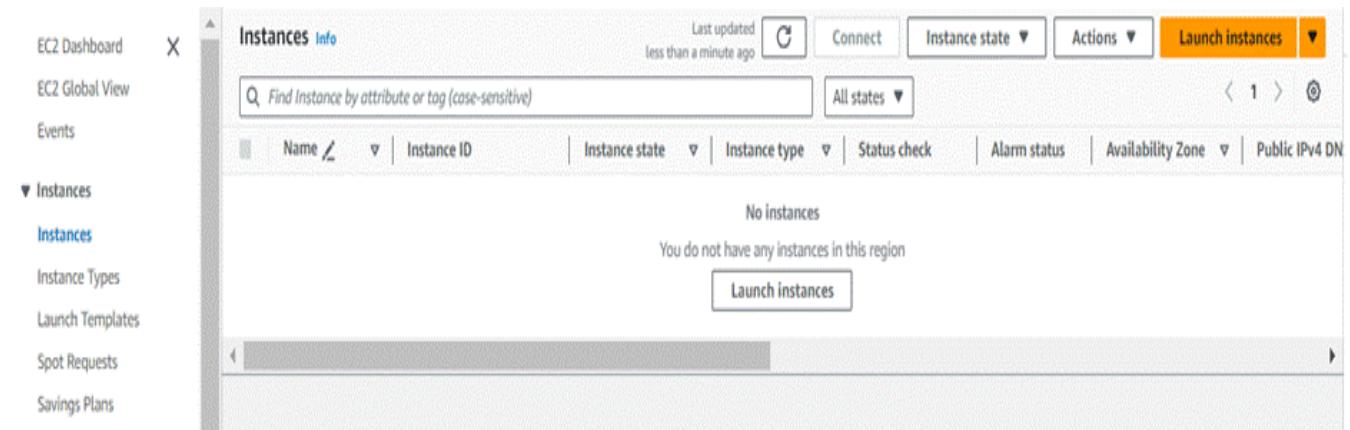
Launch EC2 Instance

In the AWS Console, navigate to EC2 and launch a new instance.



The screenshot shows the AWS CloudSearch interface. In the top navigation bar, 'Services' is selected, and a search bar contains the query 'ec2'. Below the search bar, a sidebar lists various AWS services: Services, Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area displays search results for 'ec2', including:

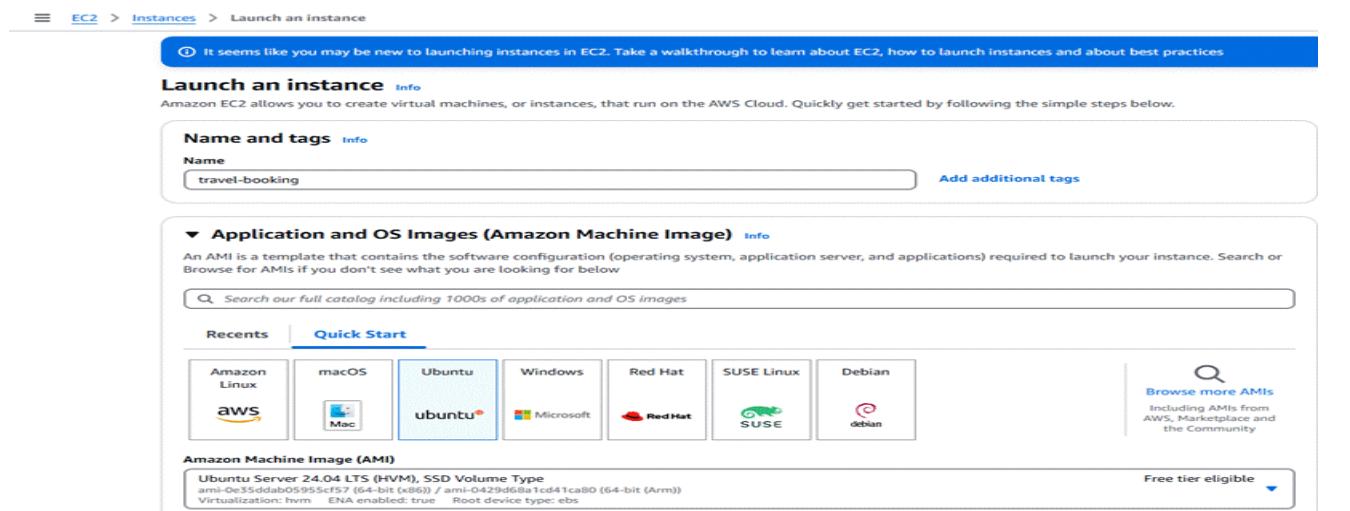
- EC2** ★ Virtual Servers in the Cloud
- EC2 Image Builder** ★ A managed service to automate build, customize and deploy OS images
- Recycle Bin** Protect resources from accidental deletion



The screenshot shows the AWS EC2 Instances page. The left sidebar includes links for EC2 Dashboard, EC2 Global View, Events, and a expanded section for Instances with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, and Savings Plans. The main content area has a header 'Instances Info' with a search bar and filters for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. It displays the message 'No instances' and 'You do not have any instances in this region' with a 'Launch instances' button.

Click on Launch instance to launch EC2 instance

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



The screenshot shows the 'Launch an instance' wizard. The top bar includes 'EC2 > Instances > Launch an instance'. A blue banner at the top says: 'It seems like you may be new to launching instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices'. The main steps are:

- Name and tags**: A field labeled 'Name' contains 'travel-booking'.
- Application and OS Images (Amazon Machine Image)**: A search bar and a list of recent AMIs including Amazon Linux, macOS, Ubuntu (selected), Windows, Red Hat, SUSE Linux, and Debian.
- Amazon Machine Image (AMI)**: A detailed view of the selected 'Ubuntu Server 24.04 LTS (HVM), SSD Volume Type' AMI, which is 'Free tier eligible'. Other details shown include 'ami-0e35ddab0595cf57 (64-bit (x86)) / ami-0429d68a1cd41ca80 (64-bit (Arm))', 'Virtualization: hvm', 'ENAv2 enabled: true', and 'Root device type: ebs'.

- Create and download the key pair for Server access.

**▼ Instance type** [Info](#) | [Get advice](#)

Instance type

|   |   |
|---|---|
| <b>t2.micro</b>   | Free tier eligible  |
| Family: t2 1 vCPU 1 GiB Memory Current generation: true | <input checked="" type="checkbox"/> All generations<br><a href="#">Compare instance types</a> |
| 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

**▼ 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*

Select  [Create new key pair](#)

**Create key pair** X

**Key pair name**  
Key pairs allow you to connect to your instance securely.

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

**Key pair type**

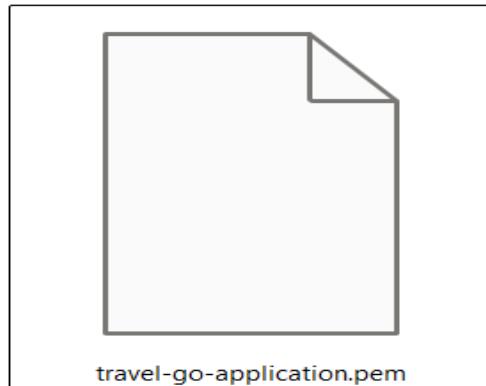
|   |  |
|---|--|
| <input checked="" type="radio"/> RSA<br>RSA encrypted private and public key pair | <input type="radio"/> ED25519<br>ED25519 encrypted private and public key pair |
|---|--|

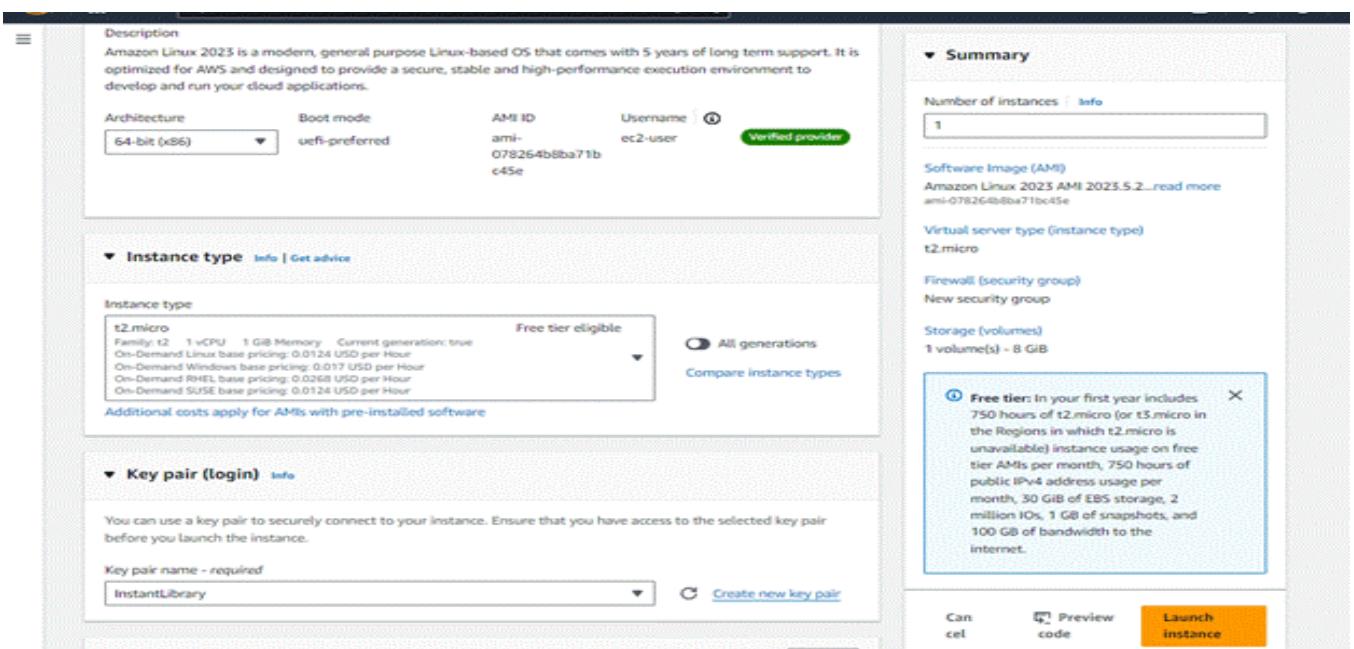
**Private key file format**

|   |  |
|---|--|
| <input checked="" type="radio"/> .pem<br>For use with OpenSSH | <input type="radio"/> .ppk<br>For use with PuTTY |
|---|--|

**⚠️** When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#) 

[Cancel](#) [Create key pair](#)





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-078264b8ba71b Username: ec2-user Verified provider

Instance type: t2.micro 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

Key pair (login): InstantLibrary

Summary  
Number of instances: 1 Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2... read more ami-078264b8ba71bc45e Virtual server type (instance type): t2.micro Firewall (security group): New security group Storage (volumes): 1 volume(s) - 8 GiB

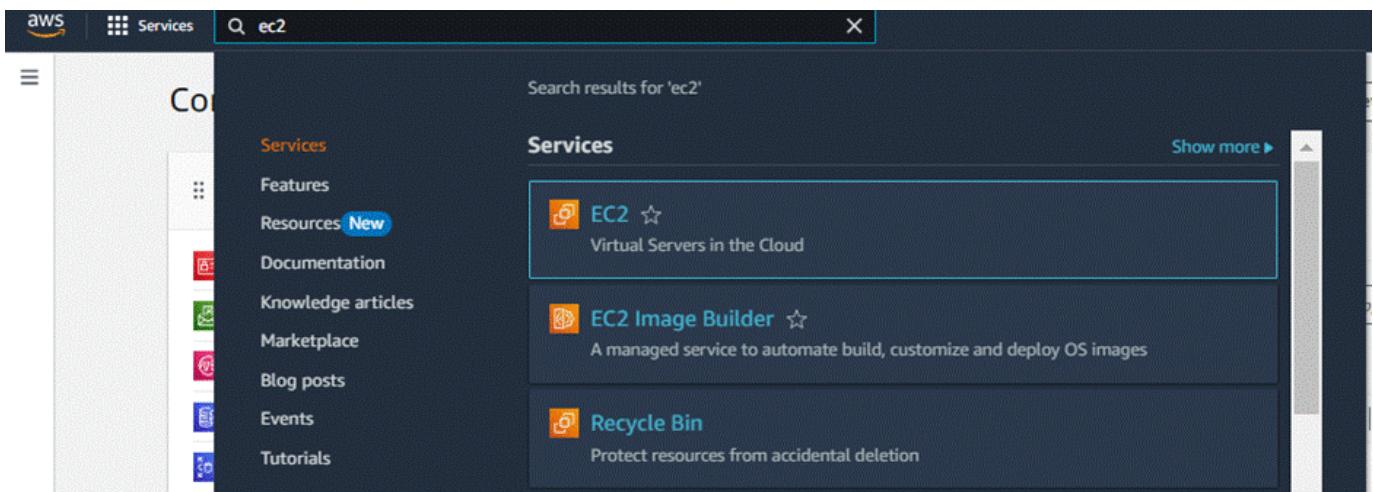
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet.

Cancel Preview code Launch instance

## Launch An EC2 instance to host The flask

### Launch EC2 Instance

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



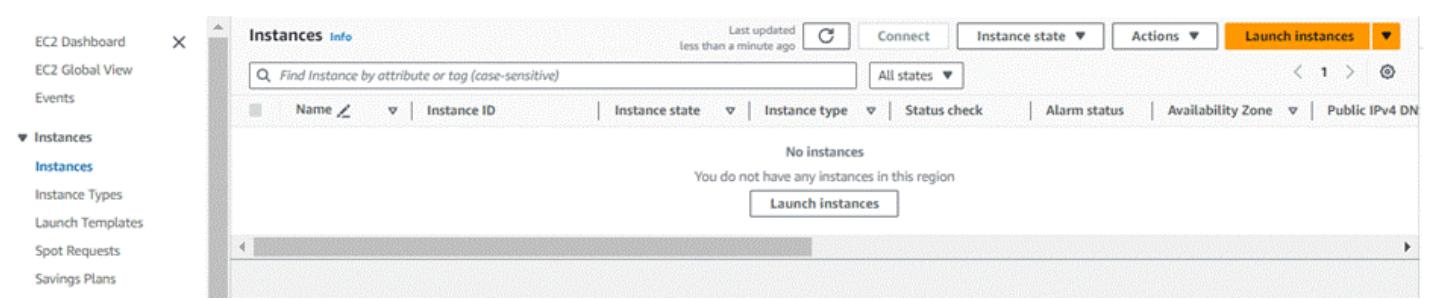
Search results for 'ec2'

Services

Show more ▾

- 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



EC2 Dashboard EC2 Global View Events

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Instances Info

Last updated less than a minute ago Connect Instance state Actions Launch Instances

Find Instance by attribute or tag (case-sensitive)

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

No instances

You do not have any instances in this region

Launch instances

## 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

[Add additional tags](#)

### ▼ Summary

#### Number of instances [Info](#)

#### Software Image (AMI)

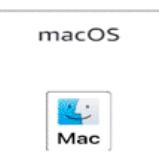
Amazon Linux 2023 AMI 2023.5.2...[read more](#)  
 ami-078264b8ba71bc45e

#### 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).


[Browse more AMIs](#)

Including AMIs from AWS, Marketplace and the Community

### Amazon Machine Image (AMI)

#### Amazon Linux 2023 AMI

ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)  
 Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

#### 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-02b49a24cfb95941c

**Verified provider**

- Create and download the key pair for Server access.

### ▼ Instance type [Info](#) | [Get advice](#)

#### Instance type

**t2.micro**  
 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

Free tier eligible

 All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

### ▼ 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

[Create new key pair](#)

### Create key pair

**Key pair name**  
Key pairs allow you to connect to your instance securely.

InstantLibrary

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

**Key pair type**

RSA  
RSA encrypted private and public key pair

ED25519  
ED25519 encrypted private and public key pair

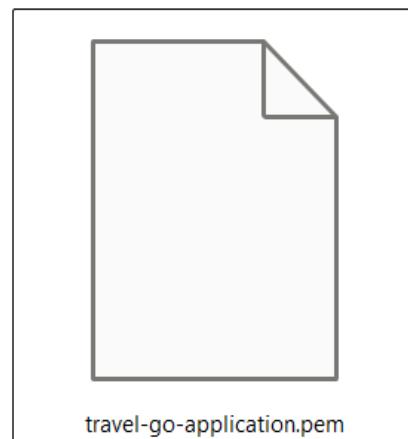
**Private key file format**

-pem  
For use with OpenSSH

-ppk  
For use with PuTTY

**When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)**

[Cancel](#) [Create key pair](#)



**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                | Username |
| 64-bit (x86) | uefi-preferred | ami-078264b8ba71bc45e | 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

**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](#)

ami-078264b8ba71bc45e

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 I/Os, 1 GiB of snapshots, and 100 GB of bandwidth to the internet.

[Cancel](#) [Preview code](#) [Launch instance](#)

## Configure security groups for HTTP, and SSH access:

**Network settings**

VPC - required | [info](#)  
 vpc-035ab276bf19dd7211  
 172.31.0.0/16 (default)

Subnet | [info](#)  
 No preference

Auto-assign public IP | [info](#)  
 Enabled

Additional charges apply when outside of free tier allowance.

**Firewall (security groups)** | [info](#)  
 A security group is a set of 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  
 Search-wizard-S

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](#)  
 Search-wizard-S created: 2025-04-14T07:20:40.970Z

**Inbound Security Group Rules**

**Security group rule 1 (TCP, 22, 0.0.0.0/0)**

|  |   |   |                               |
|--|---|---|-------------------------------|
| Type   <a href="#">info</a><br>ssh             | Protocol   <a href="#">info</a><br>TCP  | Port range   <a href="#">info</a><br>22                                     | <input type="button"/> Remove |
| Source type   <a href="#">info</a><br>Anywhere | Source   <a href="#">info</a><br><input type="button"/> Add CIDR, prefix list or security group | Description - optional   <a href="#">info</a><br>e.g. SSH for admin desktop |                               |
| 0.0.0.0/0 <input type="button"/>               |   |   |                               |

**Security group rule 2 (TCP, 80, 0.0.0.0/0)**

|  |   |   |                               |
|--|---|---|-------------------------------|
| Type   <a href="#">info</a><br>HTTP            | Protocol   <a href="#">info</a><br>TCP  | Port range   <a href="#">info</a><br>80                                     | <input type="button"/> Remove |
| Source type   <a href="#">info</a><br>Anywhere | Source   <a href="#">info</a><br><input type="button"/> Add CIDR, prefix list or security group | Description - optional   <a href="#">info</a><br>e.g. SSH for admin desktop |                               |
| 0.0.0.0/0 <input type="button"/>               |   |   |                               |

**Security group rule 3 (TCP, 5000, 0.0.0.0/0)**

|  |   |   |                               |
|--|---|---|-------------------------------|
| Type   <a href="#">info</a><br>Custom TCP      | Protocol   <a href="#">info</a><br>TCP  | Port range   <a href="#">info</a><br>5000                                   | <input type="button"/> Remove |
| Source type   <a href="#">info</a><br>Anywhere | Source   <a href="#">info</a><br><input type="button"/> Add CIDR, prefix list or security group | Description - optional   <a href="#">info</a><br>e.g. SSH for admin desktop |                               |
| 0.0.0.0/0 <input type="button"/>               |   |   |                               |

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Add security group rule

### Summary

Number of instances | [Info](#)

1

#### Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64... [read more](#)  
 ami-0e35ddab05955cf57

**Virtual server type (instance type)**  
 t2.micro

**Firewall (security group)**  
 New security group

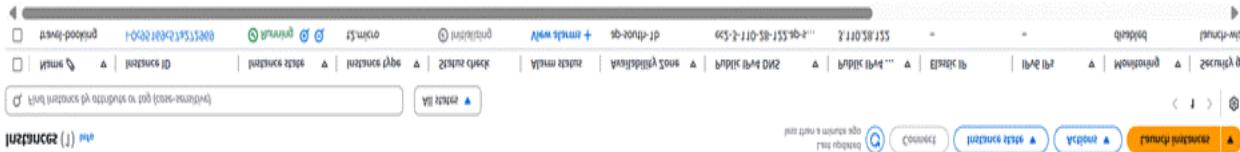
**Storage (volumes)**  
 1 volume(s) - 8 GiB

**Free tier:** In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

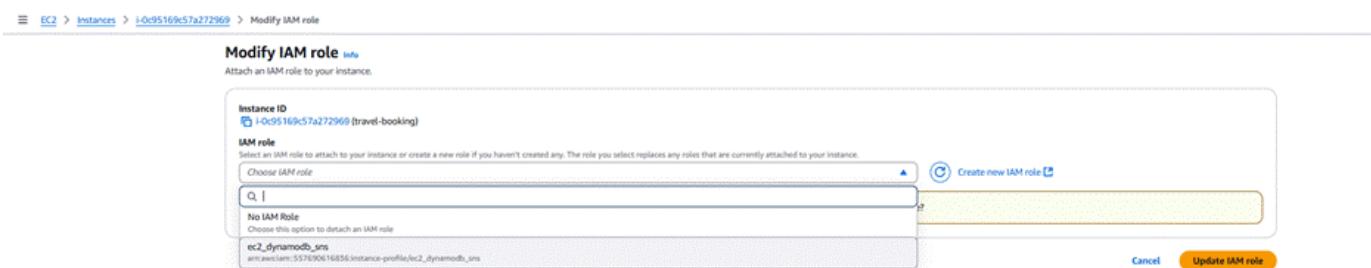
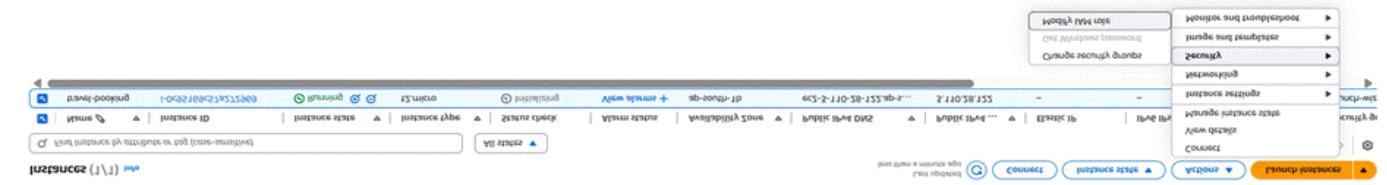
[Cancel](#)

[Launch instance](#)

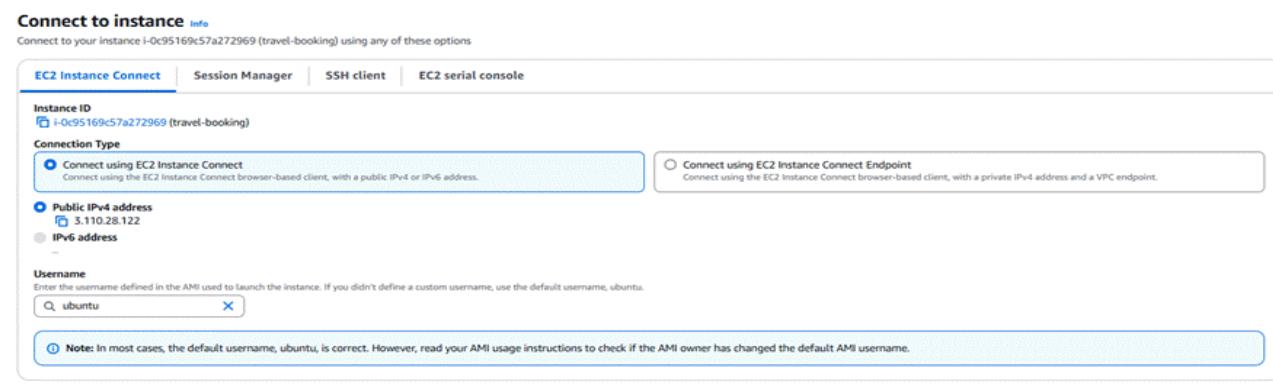
[!\[\]\(d6bcfab96a67558cb40dad541320fc50\_img.jpg\) Preview code](#)



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.



Now connect the EC2 with the files



```
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.6.0-1024-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Tue Apr 15 15:33:31 UTC 2025

  System load: 0.0          Processes:           105
  Usage of /: 36.2% of 6.71GB  Users logged in: 0
  Memory usage: 20%
  Swap usage: 0%
  * Ubuntu Pro delivers the most comprehensive open source security and
    compliance features.
  https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

63 updates can be applied immediately.
33 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Tue Apr 15 15:33:32 2025 from 13.233.177.5
ubuntu@ip-172-31-12-110:~$
```

i-0c95169c57a272969 (travel-booking)  
PublicIP: 13.203.222.149 PrivateIP: 172.31.12.110

## Milestone 7: Deployment using EC2:

Deployment on an EC2 instance involves launching a server, configuring security groups for public access, and uploading your application files. After setting up necessary dependencies and environment variables, start your application and ensure it's running on the correct port. Finally, bind your domain or use the public IP to make the application accessible online.

### 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
```

### 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](https://github.com/your-github-username/your-repository-name.git)

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



- here: '[https://github.com/AlekhyaPenubakula/Travel-Booking-website\\_Ec2\\_Dynamodb\\_SNS.git](https://github.com/AlekhyaPenubakula/Travel-Booking-website_Ec2_Dynamodb_SNS.git)'

**This will download your project to the EC2 instance.**

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

```
cd <your-folder>
```

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

## Run the Flask Application

- export SNS\_TOPIC\_ARN=arn:aws:sns:ap-south-1:557690616836:BookingConfirmation
  - git pull origin main ( Only if you made changes in git and want to reflect them in EC2 terminal.)

## # Install requirements

pip install flask boto3

- Launch the Flask app:

```
sudo -E venv/bin/python3 app.py
```

### **Verify the Flask app is running:**

<http://your-ec2-public-ip>

- Run the Flask app on the EC2 instance

```
Last login: Mon Apr 14 16:27:44 2025 from 13.233.177.5
ubuntu@ip-172-31-12-110:~$ cd Travel-Booking-website_Ec2_Dynamodb_SNS
ubuntu@ip-172-31-12-110:~/Travel-Booking-website_Ec2_Dynamodb_SNS$ cd travel_booking_system
ubuntu@ip-172-31-12-110:~/Travel-Booking-website_Ec2_Dynamodb_SNS/travel_booking_system$ export SNS_TOPIC_ARN=arn:aws:sns:ap-south-1:557690616836:BookingConfirmation
ubuntu@ip-172-31-12-110:~/Travel-Booking-website_Ec2_Dynamodb_SNS/travel_booking_system$ git pull origin main
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (5/5), 1.68 KiB | 429.00 KiB/s, done.
From https://github.com/alekhyapenubakula/Travel-Booking-website_Ec2_Dynamodb_SNS
 * branch      main       -> FETCH HEAD
  2289f47..397e3a1  main       -> origin/main
Updating 2289f47..397e3a1
Fast-forward
 travel_booking_system/templates/hotels.html | 107 ++++++-
 1 file changed, 63 insertions(+), 44 deletions(-)
ubuntu@ip-172-31-12-110:~/Travel-Booking-website_Ec2_Dynamodb_SNS/travel_booking_system$ sudo -E venv/bin/python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
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://172.31.12.110:80
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 661-186-119
```

**Access the website through:**

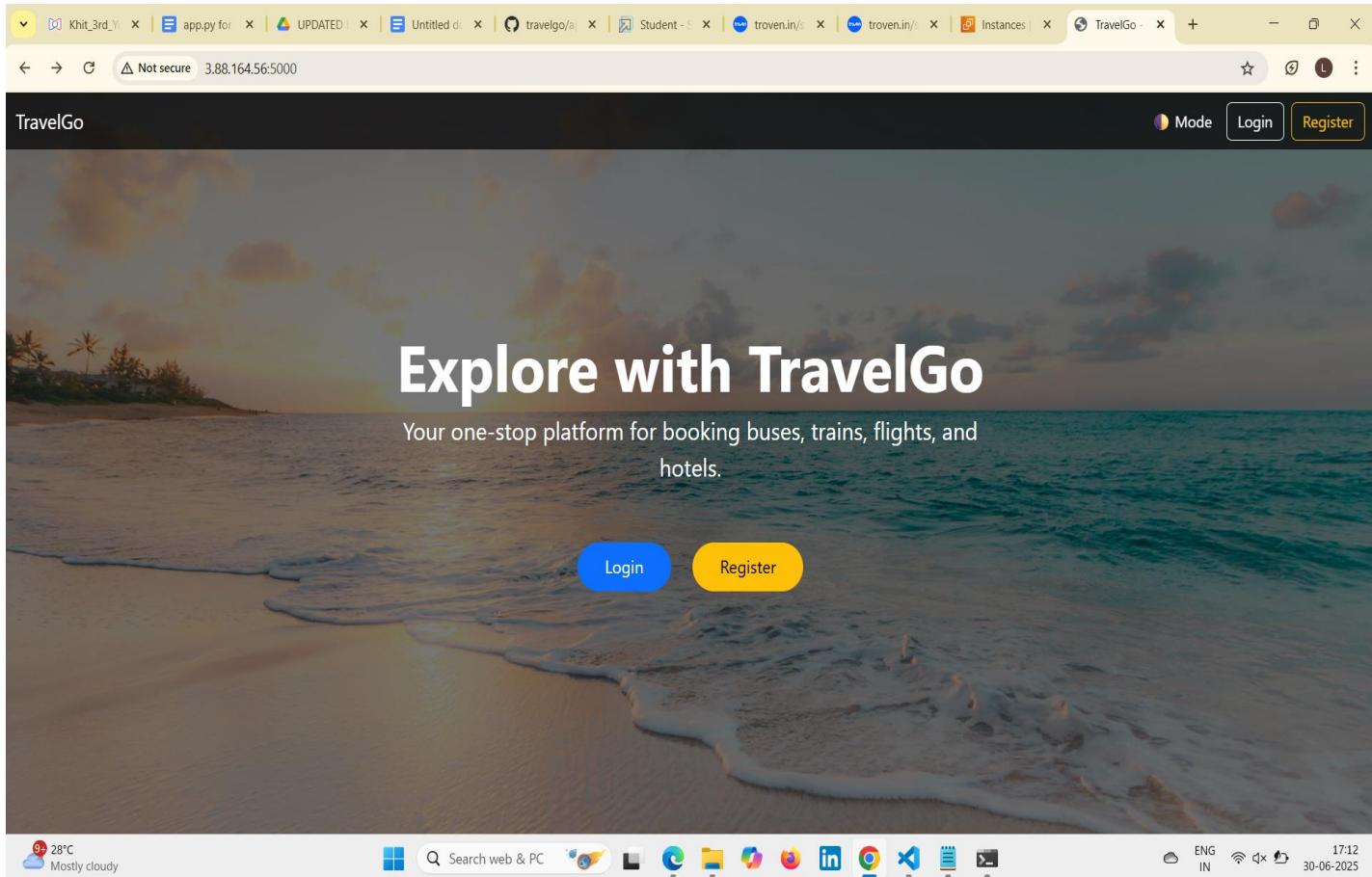
**Public IPs: <http://3.88.164.56:5000/>**

## Milestone 8 : Testing and Deployment

Testing and deployment involve verifying that your application works as expected before making it publicly accessible. Start by testing locally or on a staging environment to catch bugs and ensure functionality. Once tested, deploy the application to an EC2 instance, configure necessary services, and perform a final round of live testing to confirm everything runs smoothly in the production environment.

### Functional Testing to verify the Project:

#### Home Page:





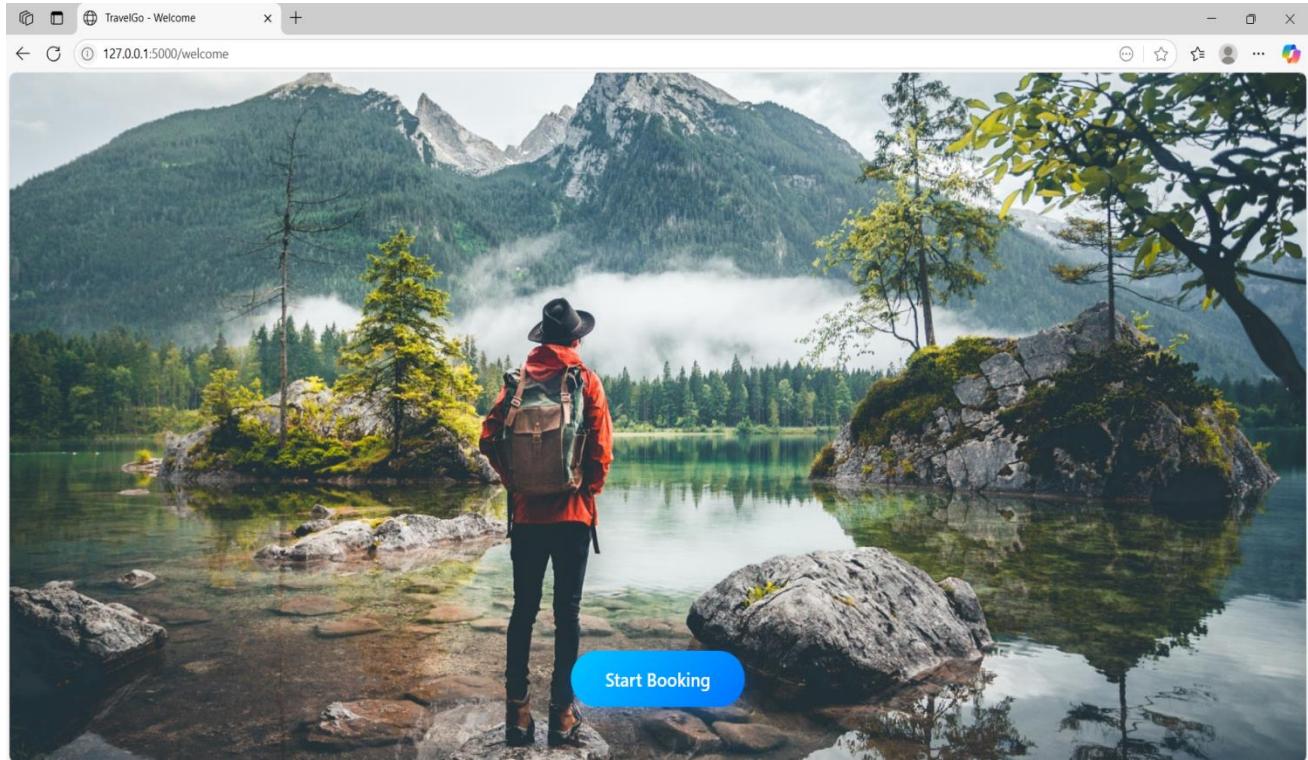
## Login Page:

The screenshot shows a web browser window titled "Login - TravelGo". The address bar displays the URL "127.0.0.1:5000/login". The main content area features a white rectangular form with rounded corners. At the top center is the text "Login to TravelGo". Below it are two input fields: the first for "Email" and the second for "Password", both with placeholder text. Underneath these fields is a large purple rectangular button with the word "Login" in white. At the bottom of the form, there is a small line of text that reads "New user? [Register here](#)".

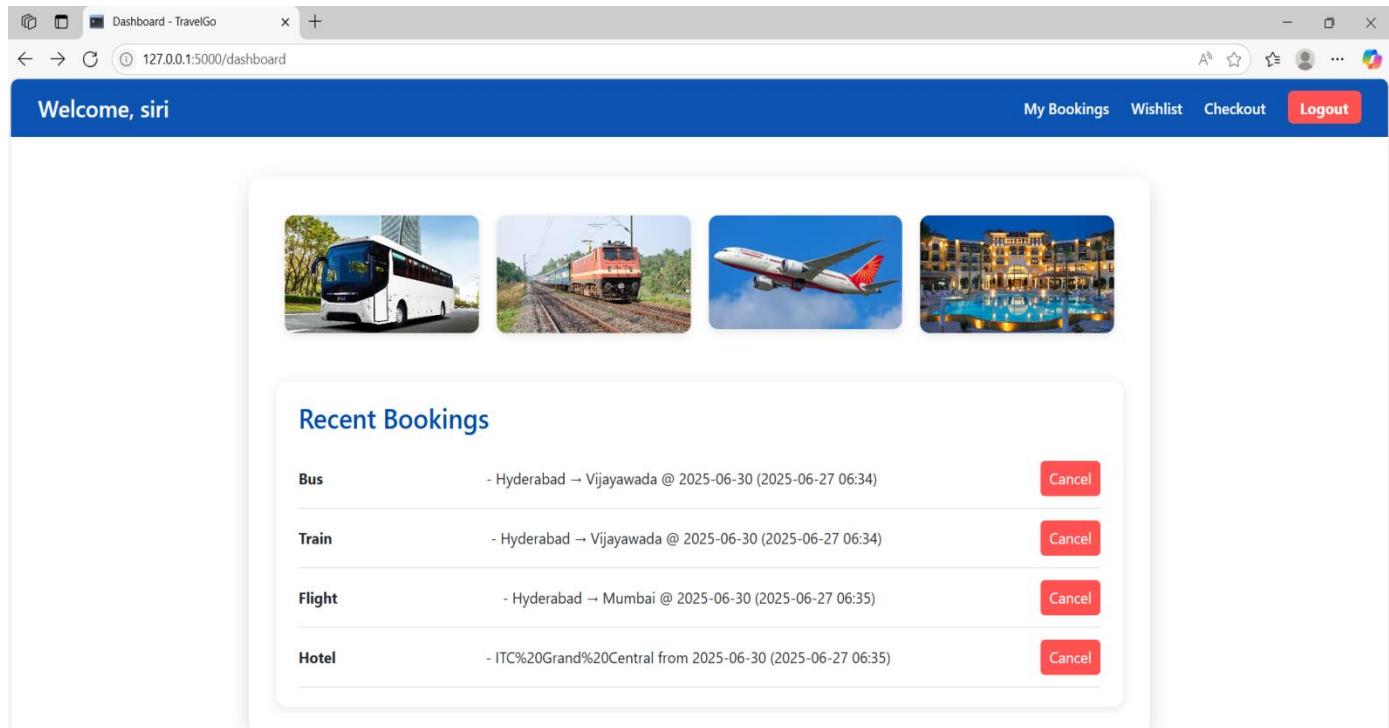
## Register Page:

The screenshot shows a web browser window titled "Register - TravelGo". The address bar displays the URL "127.0.0.1:5000/register". The main content area features a white rectangular form with rounded corners. At the top center is the text "Create an Account". Below it are three input fields: "Username", "Email", and "Password", each with placeholder text. Underneath these fields is a large blue rectangular button with the word "Register" in white. At the bottom of the form, there is a small line of text that reads "Already registered? [Login here](#)".

## Welcome Page:



## Dashboard Page:



Welcome, siri

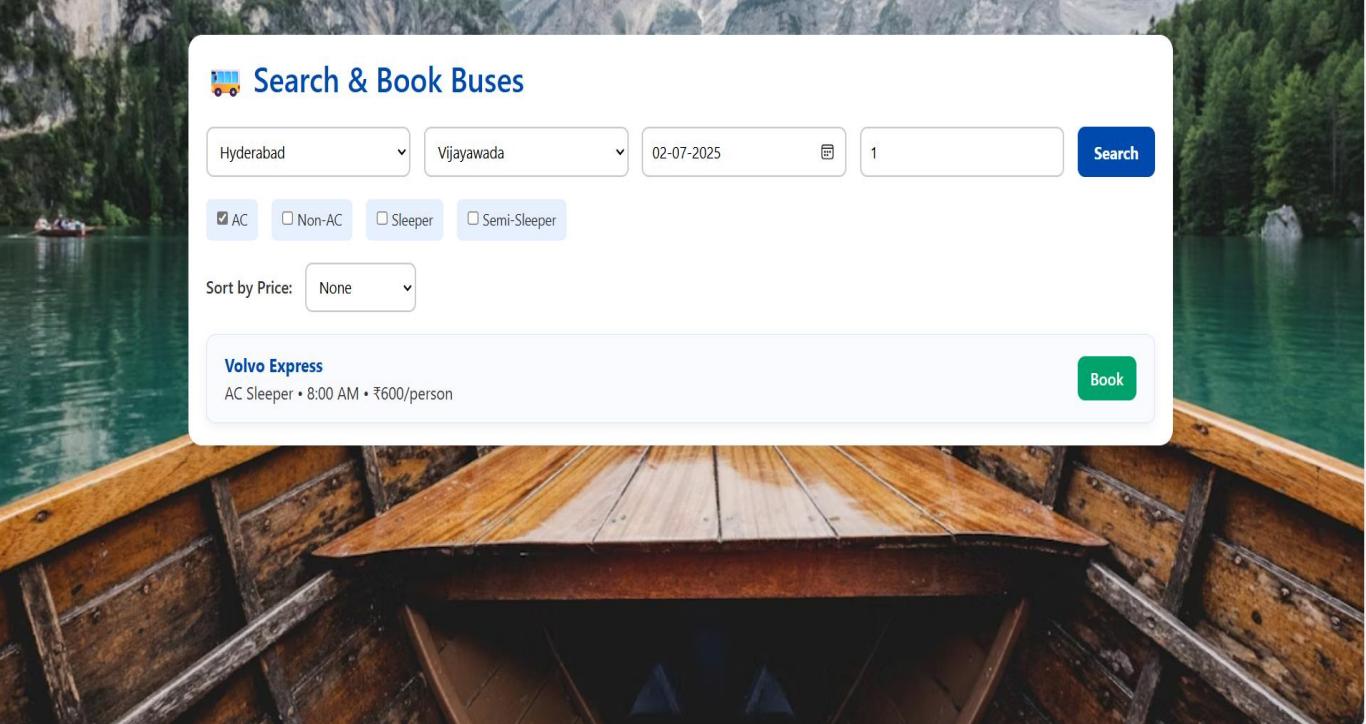
My Bookings   Wishlist   Checkout   Logout

Recent Bookings

|        |  |        |
|--------|--|--------|
| Bus    | - Hyderabad → Vijayawada @ 2025-06-30 (2025-06-27 06:34)   | Cancel |
| Train  | - Hyderabad → Vijayawada @ 2025-06-30 (2025-06-27 06:34)   | Cancel |
| Flight | - Hyderabad → Mumbai @ 2025-06-30 (2025-06-27 06:35)       | Cancel |
| Hotel  | - ITC%20Grand%20Central from 2025-06-30 (2025-06-27 06:35) | Cancel |



## Bus Booking Page:



Bus Booking - TravelGo

127.0.0.1:5000/bus

TravelGo

Home Dashboard

### Search & Book Buses

Hyderabad Vijayawada 02-07-2025 1 Search

AC  Non-AC  Sleeper  Semi-Sleeper

Sort by Price: None

**Volvo Express**  
AC Sleeper • 8:00 AM • ₹600/person Book

## Confirm Bus booking:

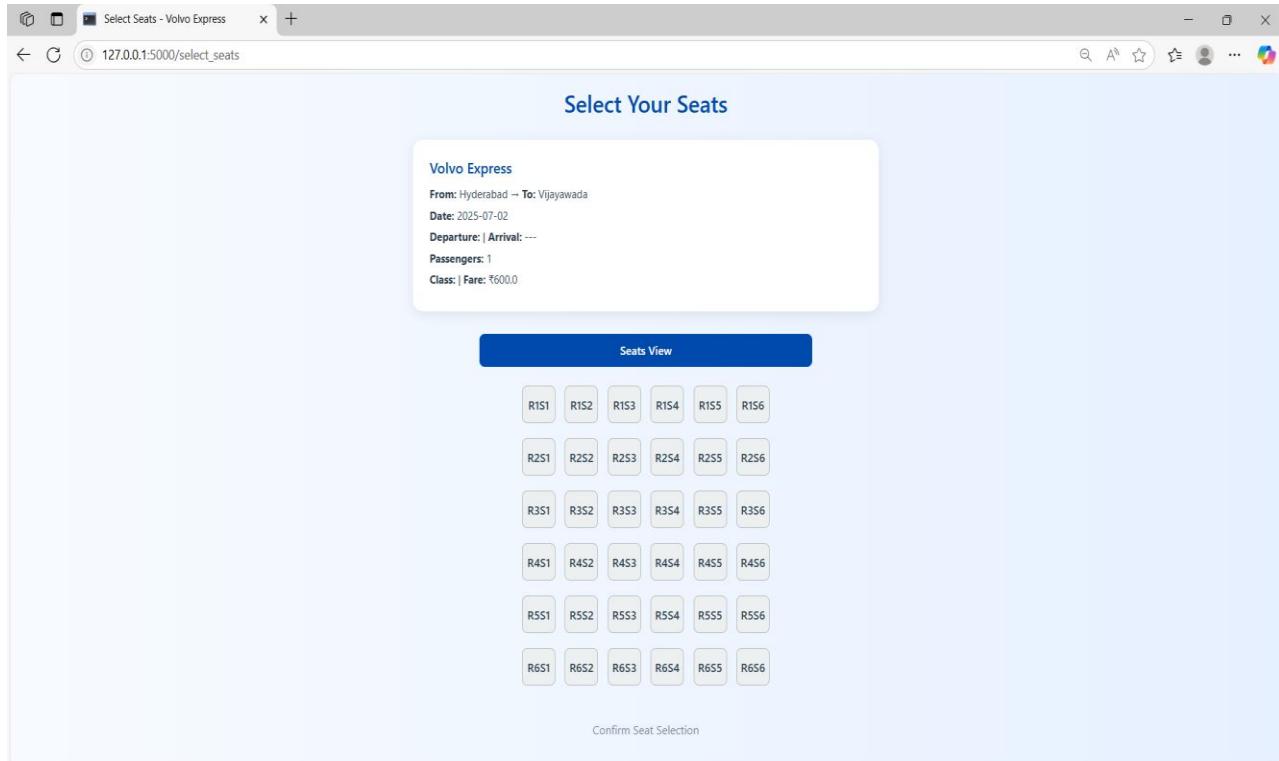
Confirm Bus Booking

Bus Name: Volvo Express  
Route: Hyderabad → Vijayawada  
Travel Date: 2025-07-02  
Departure Time: 8:00 AM  
Bus Type: AC Sleeper  
Number of Seats: 1  
Price per Seat: ₹600.0

Total Price: ₹600.0

Proceed to Select Seats

## Selecting Seats:



**Select Your Seats**

**Volvo Express**

From: Hyderabad — To: Vijayawada

Date: 2025-07-02

Departure: | Arrival: ---

Passengers: 1

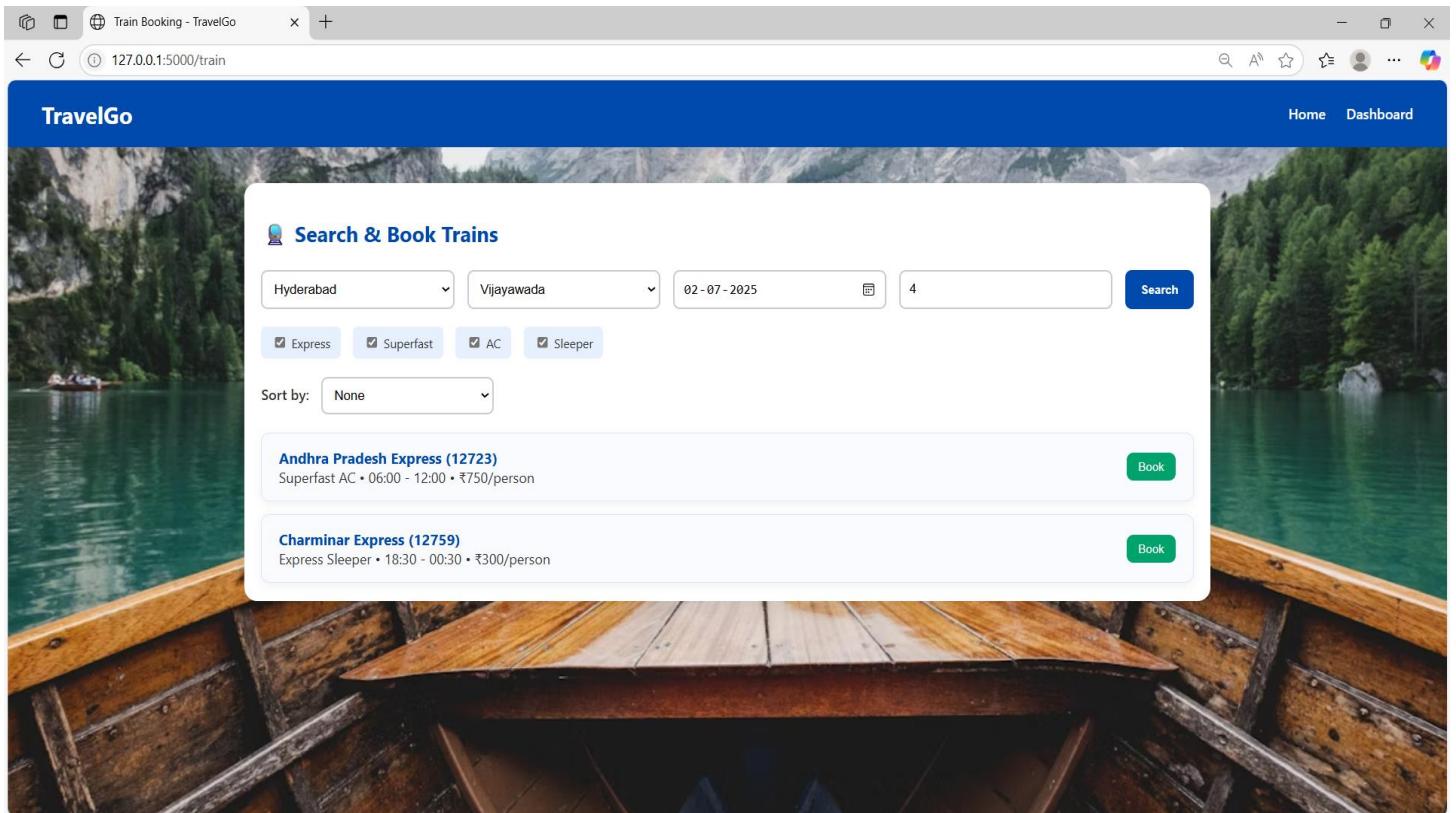
Class: | Fare: ₹600.0

**Seats View**

|      |      |      |      |      |      |
|------|------|------|------|------|------|
| R1S1 | R1S2 | R1S3 | R1S4 | R1S5 | R1S6 |
| R2S1 | R2S2 | R2S3 | R2S4 | R2S5 | R2S6 |
| R3S1 | R3S2 | R3S3 | R3S4 | R3S5 | R3S6 |
| R4S1 | R4S2 | R4S3 | R4S4 | R4S5 | R4S6 |
| R5S1 | R5S2 | R5S3 | R5S4 | R5S5 | R5S6 |
| R6S1 | R6S2 | R6S3 | R6S4 | R6S5 | R6S6 |

Confirm Seat Selection

## Train Booking Page:



**TravelGo**

Home Dashboard

**Search & Book Trains**

Hyderabad — Vijayawada • 02-07-2025 • 4

Express  Superfast  AC  Sleeper

Sort by: None

**Andhra Pradesh Express (12723)**  
Superfast AC • 06:00 - 12:00 • ₹750/person **Book**

**Charminar Express (12759)**  
Express Sleeper • 18:30 - 00:30 • ₹300/person **Book**



## Selecting Seats:

Volvo Express  
From: Hyderabad → To: Vijayawada  
Date: 2025-07-02  
Departure | Arrival: ---  
Passengers: 1  
Class: | Fare: ₹600.0

Seats View

R1S1 R1S2 R1S3 R1S4 R1S5 R1S6  
R2S1 R2S2 R2S3 R2S4 R2S5 R2S6  
R3S1 R3S2 R3S3 R3S4 R3S5 R3S6  
R4S1 R4S2 R4S3 R4S4 R4S5 R4S6  
R5S1 R5S2 R5S3 R5S4 R5S5 R5S6  
R6S1 R6S2 R6S3 R6S4 R6S5 R6S6

Confirm Seat Selection

## Flight Booking Page

Flight Booking - TravelGo

TravelGo

Home Dashboard

Search & Book Flights

Hyderabad (HYD) Mumbai (BOM) 02 - 07 - 2025 04 - 07 - 2025 4 Search

Indigo  Vistara  Air India  Direct  1 Stop

Sort by: None

**Indigo (6E 234)**  
Direct Economy • 08:00 - 09:30 • Direct • ₹3500/person **Book**

**Vistara (UK 876)**  
Direct Economy • 10:00 - 11:45 • Direct • ₹4200/person **Book**

**Air India (AI 543)**  
1 Stop Economy • 14:00 - 16:00 • 1 Stop(s) • ₹3000/person **Book**



## Selecting Seats:

Indigo

From: Hyderabad → To: Mumbai

Date: 2025-07-02

Departure: | Arrival: 09:30

Passengers: 4

Class: Direct Economy | Fare: ₹3500.0

Seats View

R1S1 R1S2 R1S3 R1S4 R1S5 R1S6  
R2S1 R2S2 R2S3 R2S4 R2S5 R2S6  
R3S1 R3S2 R3S3 R3S4 R3S5 R3S6  
R4S1 R4S2 R4S3 R4S4 R4S5 R4S6  
R5S1 R5S2 R5S3 R5S4 R5S5 R5S6  
R6S1 R6S2 R6S3 R6S4 R6S5 R6S6

Confirm Seat Selection

## Hotel Booking page:

TravelGo

Find & Book Hotel Rooms

mumbai 30 - 06 - 2025 02 - 07 - 2025 1

Search

5-Star 4-Star 3-Star WiFi Pool Parking

Sort by: None

ITC Grand Central

Mumbai • 5-Star • ₹12000/night

Amenities: WiFi, Pool, Parking, Spa

Book



## My Bookings:

Booking Date: N/A

Type: Bus

Bus Name: Vihin Express

Route: Hyderabad → Vijayawada

Travel Date: 2023-07-02

Departure Time: 08:00 AM

Bus Type: AC Seater

Seats Booked: 4

Selected Seats: R 1, S 1, R 1, S 2, R 1, S 3, R 1, S 4

Total Price: ₹2400.00

[Cancel Booking](#)

Booking Date: N/A

Type: Train

Train Name: Andhra Pradesh Express

Route: Hyderabad → Vijayawada

Travel Date: 2023-07-02

Departure

Class:

Seats Booked: 4

Selected Seats: R 2, S 2, R 2, S 3, R 2, S 4, R 2, S 5

Total Price: ₹3000.00

[Cancel Booking](#)

Booking Date: N/A

Type: Flight

Flight Name: Indigo

Route: Hyderabad → Mumbai

Travel Date: 2023-07-02

Time:

Seats Booked: 4

Selected Seats: R 6, S 1, R 6, S 2, R 6, S 3, R 6, S 4

Total Price: ₹4000.00

[Cancel Booking](#)

Booking Date: N/A

Type: Hotel

Hotel Name: ITC Grand Central

Check-in: 2023-06-30

Check-out: 2023-07-02

Guests: 1

Total Price: ₹12000.00

[Cancel Booking](#)

## Conclusion for TravelGo Cloud-Based Booking Platform Project:

The TravelGo Cloud-Based Booking Platform has been successfully developed and deployed using a scalable architecture built on AWS services. Leveraging EC2 for application hosting, DynamoDB for efficient data management, and SNS for real-time booking notifications, the platform delivers a seamless and reliable experience for users booking bus, train, flight, and hotel services.

This cloud-native solution addresses the common challenges of fragmented travel platforms by offering a unified interface for users to search, book, and manage their travel itineraries. With secure user authentication, dynamic seat selection, and centralized booking history, TravelGo ensures high usability and responsiveness. The system has been thoroughly tested for functionality, ensuring smooth operations from login to booking confirmation.

In conclusion, TravelGo demonstrates the power of cloud-based applications in streamlining multi-modal travel management, enhancing user convenience, and ensuring robust performance under growing demand—making it a scalable solution for the evolving travel industry.