

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

## Project Description:

TravelGo is a comprehensive, 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 for backend development, 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 various 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 userfriendly 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.

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

TravelGo offers a unified platform for booking buses, trains, flights, and hotels, allowing users to seamlessly plan entire trips. For instance, a user can log in, select their preferred transport, and then book accommodation, all within one interface. Flask efficiently manages real-time retrieval of diverse travel listings and processes user input. Hosted on AWS EC2, the platform remains highly responsive even during peak traffic, ensuring multiple users can browse and book without any delays, providing a truly integrated travel planning experience.

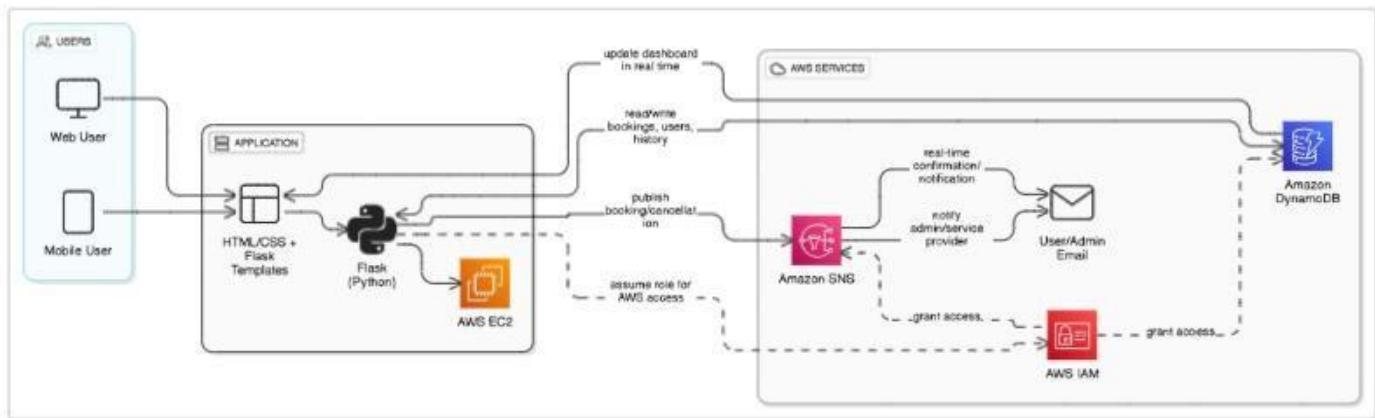
### Scenario 2: Real-Time Booking Confirmation and Updates with AWS SNS

Upon successful booking, TravelGo instantly notifies the user using AWS SNS, sending real-time email confirmations with all relevant details. For example, after a student books a hotel, they immediately receive an email confirming their reservation. This notification is triggered from the Flask backend once the booking is securely recorded in DynamoDB. AWS SNS can also alert admin staff or service providers, ensuring transparency and real-time updates across all stakeholders for every transaction, from initial booking to any subsequent cancellations.

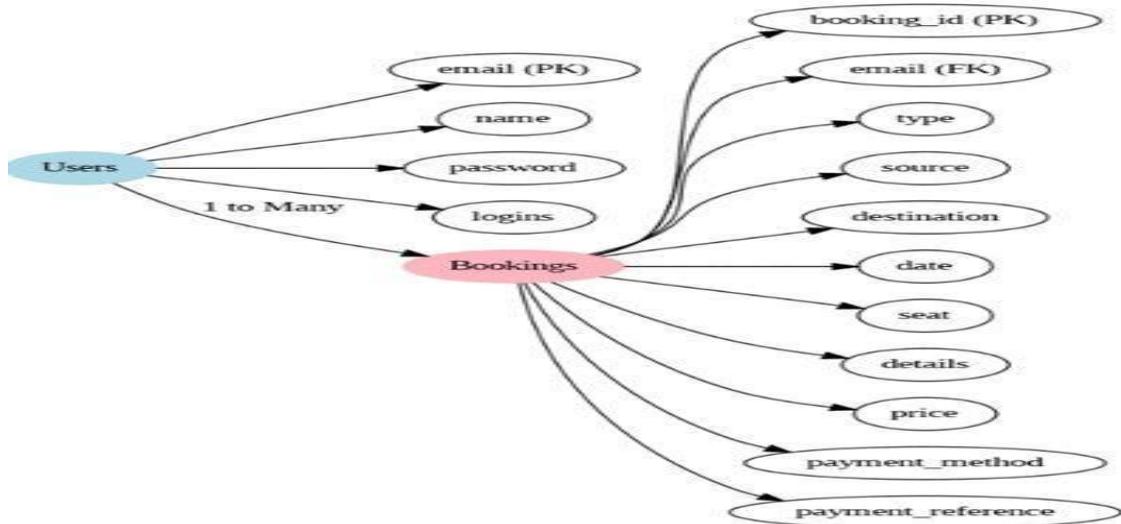
### Scenario 3: Dynamic User Dashboard with Comprehensive Travel History & Management

TravelGo features a dynamic and personalized user dashboard that provides a comprehensive overview of all past and upcoming bookings. A logged-in user can easily view their flight and hotel reservations, categorized by type, along with essential details like dates and prices. Flask efficiently fetches this personalized data from AWS DynamoDB, which persistently stores all user bookings. The responsive UI, powered by HTML/CSS and Flask templates, ensures users can review, manage, and quickly cancel their bookings anytime, from any device, with real-time updates.

## AWS ARCHITECTURE



## Entity Relationship (ER) Diagram:



## Pre-requisites:

1. **AWS Account Setup:** [AWS Account Setup](#)
2. **Understanding IAM:** [IAM Overview](#)
3. **Amazon EC2 Basics:** [EC2 Tutorial](#)
4. **DynamoDB Basics:** [DynamoDB Introduction](#)
5. **SNS Overview:** [SNS Documentation](#)
6. **Git Version Control:** [Git Documentation](#)

## Project WorkFlow:

### 1. Backend Development and Application Setup

**Activity 1.1:** Develop the Backend Using Flask.

**Activity 1.2:** Integrate AWS Services Using boto3

## 2. . AWS Account Setup and Login

**Activity 2.1:** . AWS Account Setup and Login

**Activity 2.2:** Log in to the AWS Management Console.

## 3. DynamoDB Database Creation and Setup

**Activity 3.1:** Create a DynamoDB Table.

**Activity 3.2:** Configure Attributes for User Data and Book Requests.

## 4. SNS Notification Setup

**Activity 4.1:** Create SNS topics for book request notifications.

**Activity 4.2:** Subscribe users and library staff to SNS email notifications.

## 5. IAM Role Setup

**Activity 5.1:** Create IAM Role

**Activity 5.2:** Attach Policies

## 6. EC2 Instance Setup

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

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

## 7. Deployment on EC2

**Activity 7.1:** Upload Flask Files

**Activity 7.2:** Run the Flask App

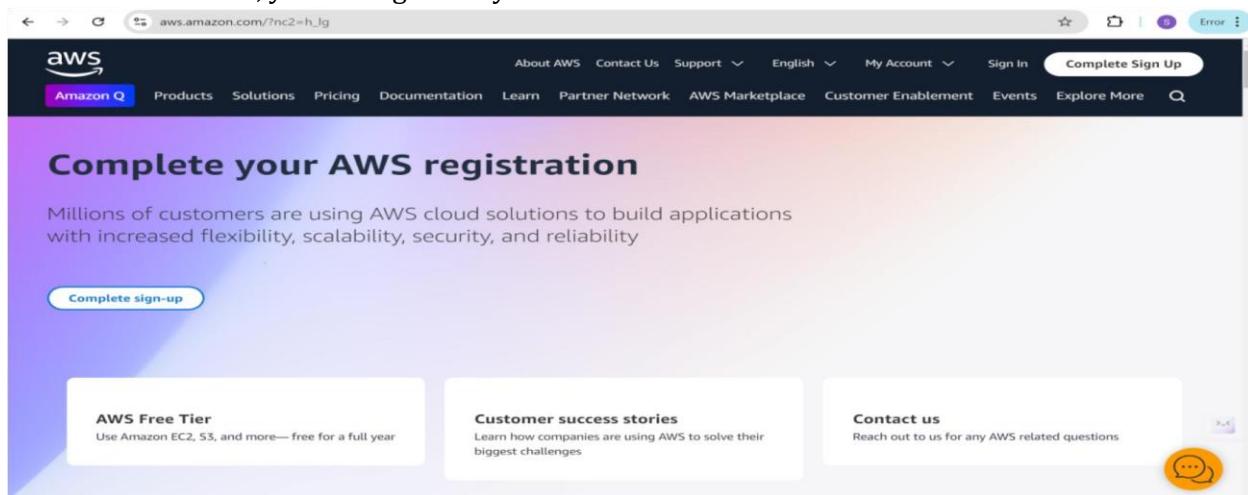
## 8. Testing and Deployment

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

## Milestone 1: AWS Account Setup and Login

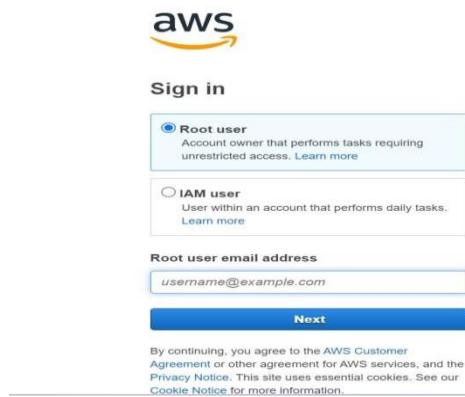
This is for your understanding only, please refrain from creating an AWS account. A temporary account will be provided via Troven.

- 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.



### • Activity 1: Log in to the AWS Management Console

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

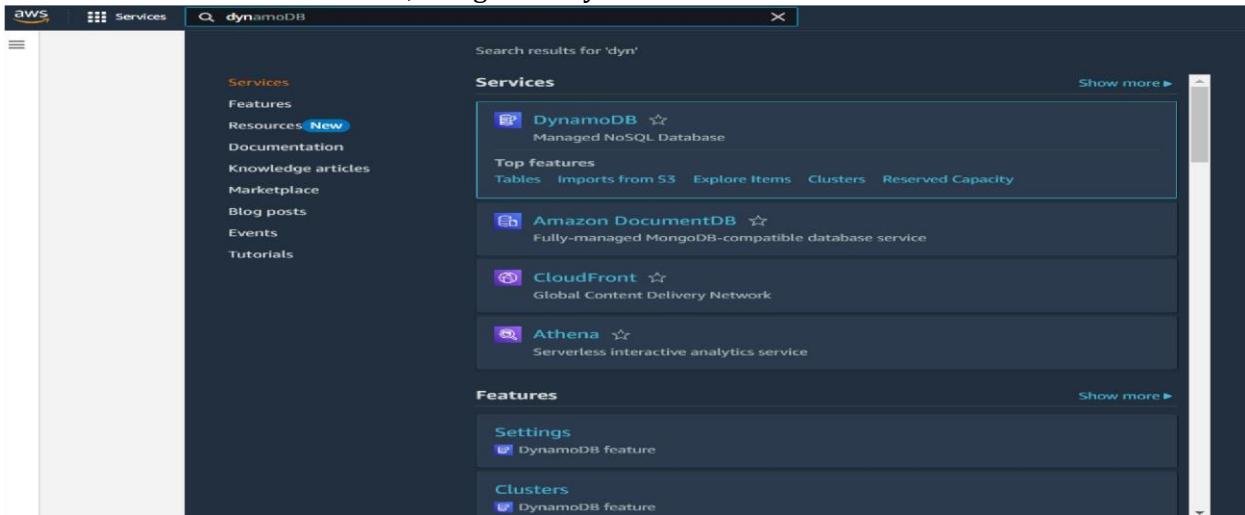


## Milestone 2: DynamoDB Database Creation and Setup

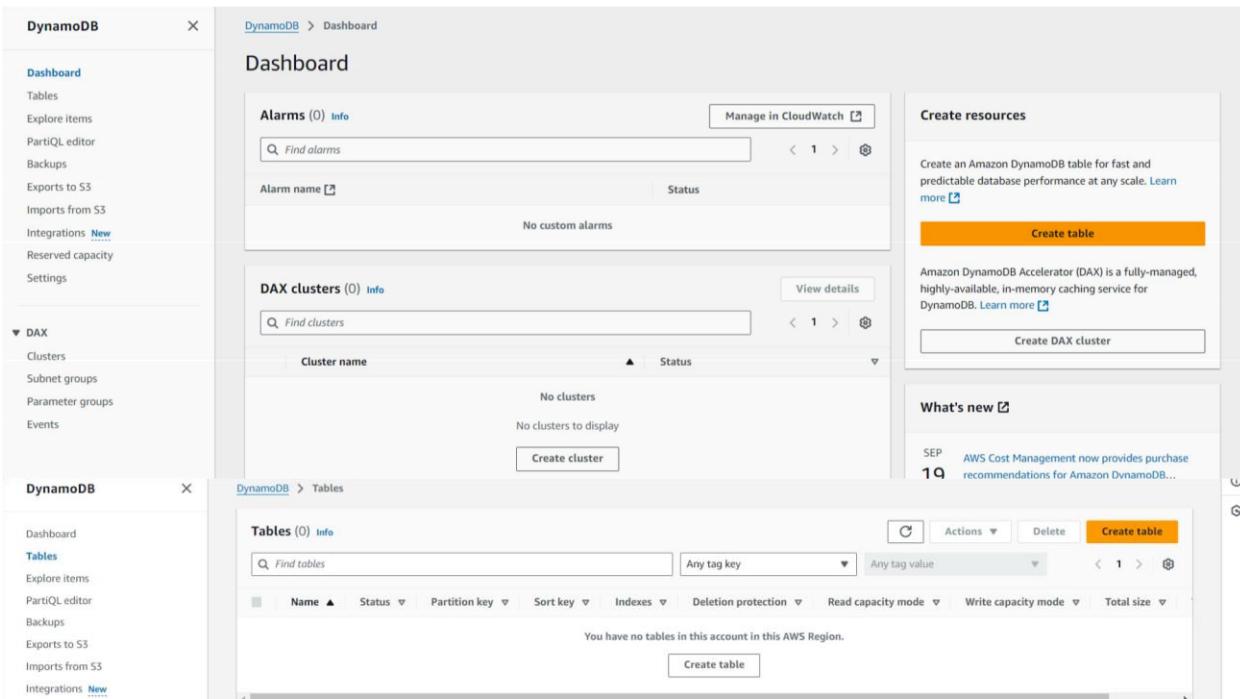
Database Creation and Setup involves initializing a cloud-based NoSQL database to store and manage application data efficiently. This step includes defining tables, setting primary keys, and configuring read/write capacities. It ensures scalable, high-performance data storage for seamless backend operations.

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

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



The image is a screenshot of the AWS Services search results. The search bar at the top contains 'dynamodb'. On the left, a sidebar lists various AWS resources: Services, Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area shows search results for 'dynamodb'. Under the 'Services' heading, 'DynamoDB' is listed as a 'Managed NoSQL Database'. Below it, under 'Top features', are 'Tables', 'Imports from S3', 'Explore Items', 'Clusters', and 'Reserved Capacity'. Further down, under 'Features', are 'Settings' (with 'DynamoDB feature') and 'Clusters' (with 'DynamoDB feature'). Other services listed include Amazon DocumentDB, CloudFront, and Athena.



DynamoDB > Dashboard

## Dashboard

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

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

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

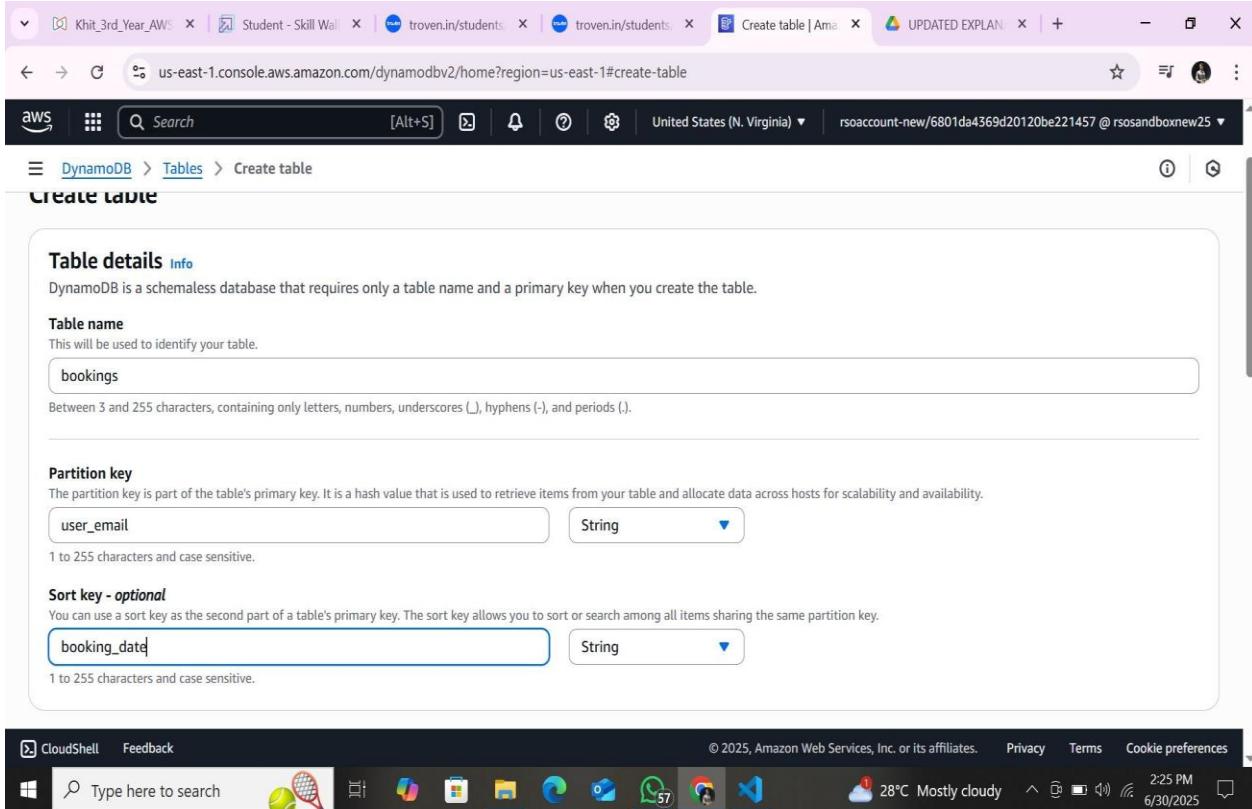
SEP AWS Cost Management now provides purchase recommendations for Amazon DynamoDB... [Learn more](#)

DynamoDB > Tables

Tables (0) [Info](#)

Create table

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



khit\_3rd\_Year\_AWS | Student - Skill Wall | troven.in/students | troven.in/students | Create table | Amazon | UPDATED EXPLAIN | +

us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#create-table

aws Search [Alt+S]

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.

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.

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.

1 to 255 characters and case sensitive.

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

Type here to search 28°C Mostly cloudy 2:25 PM 6/30/2025

Khit\_3rd\_Year\_AWS | Student - Skill Wall | troven.in/students | troven.in/students | List tables | Amazon | UPDATED EXPLAIN | + | - | X

us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#tables

aws | Search [Alt+S] | United States (N. Virginia) | rsoaccount-new/6801da4369d20120be221457 @ rsosandboxnew25

DynamoDB > Tables

**DynamoDB**

- Dashboard
- Tables**
- Explore items
- PartiQL editor
- Backups
- Exports to S3
- Imports from S3
- Integrations [New](#)
- Reserved capacity
- Settings

**DAX**

- Clusters
- Subnet groups

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 2:25 PM 6/30/2025

Share feedback X

The bookings table was created successfully. X

**Tables (1) Info**

	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection
<input type="checkbox"/>	<a href="#">bookings</a>	Active	user_email (\$)	booking_date (\$)	0	0	<input checked="" type="checkbox"/> Off

Khit\_3rd\_Year\_AWS | Student - Skill Wall | troven.in/students | troven.in/students | Create table | Amazon | UPDATED EXPLAIN | + | - | X

us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#create-table

aws | Search [Alt+S] | United States (N. Virginia) | rsoaccount-new/6801da4369d20120be221457 @ rsosandboxnew25

DynamoDB > 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.

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.

String

1 to 255 characters and case sensitive.

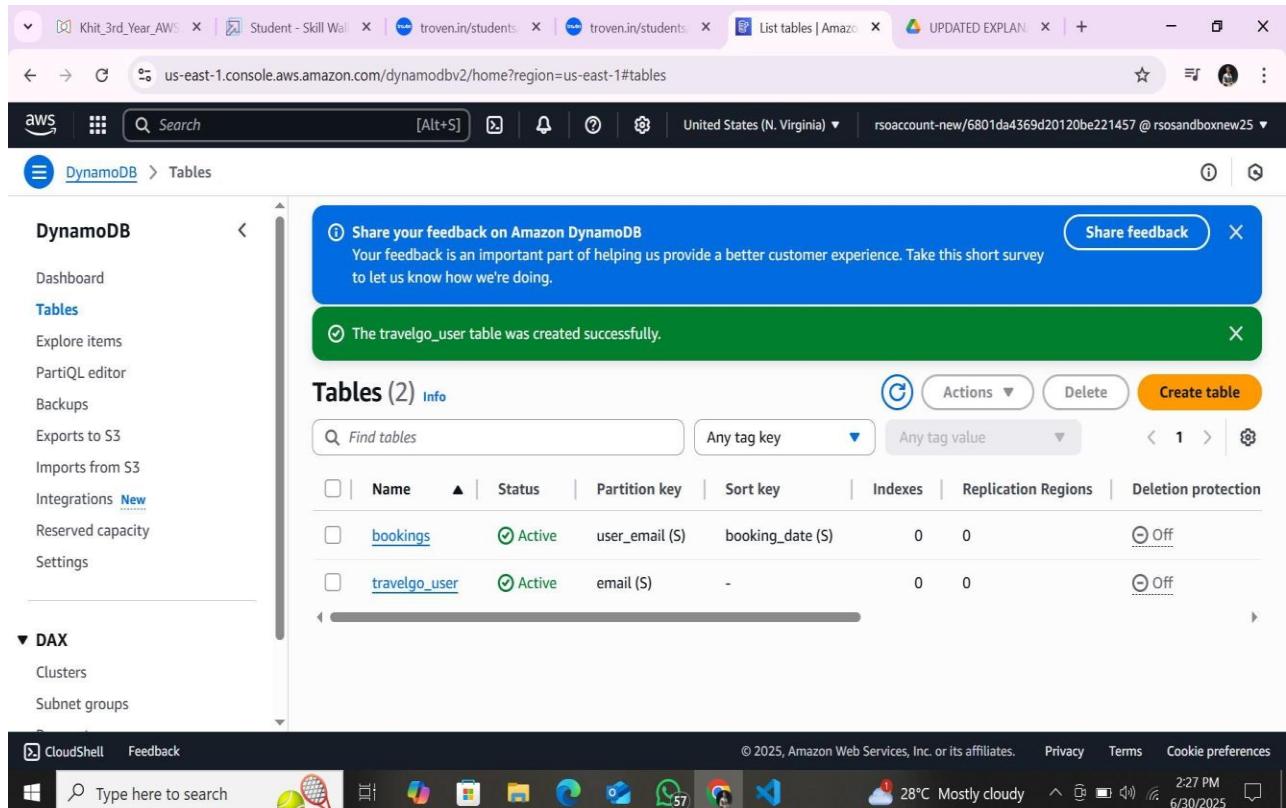
**Sort key - optional**

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.

String

1 to 255 characters and case sensitive.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 2:26 PM 6/30/2025



The screenshot shows the AWS DynamoDB console interface. On the left, a sidebar menu includes options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations (New), Reserved capacity, and Settings. Below this is a section for DAX with Clusters and Subnet groups. The main content area displays a success message: "The travelgo\_user table was created successfully." A table titled "Tables (2)" lists two entries:

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection
<a href="#">bookings</a>	Active	user_email (\$)	booking_date (\$)	0	0	Off
<a href="#">travelgo_user</a>	Active	email (\$)	-	0	0	Off

At the bottom, the Windows taskbar shows various pinned icons and the system tray indicates it's 2:27 PM on 6/30/2025.

I created three dynamo Tables for my project they are bookings, trains and travelgo user .

### Milestone 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.

Search results for 'sns'

**Services**

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

**Services**

- 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

**Show more ▶**

**Features**

- Events
- ElastiCache feature
- SMS
- AWS End User Messaging feature

**Show more ▶**

**Hosted zones**

Elastic Load Balancing feature

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**

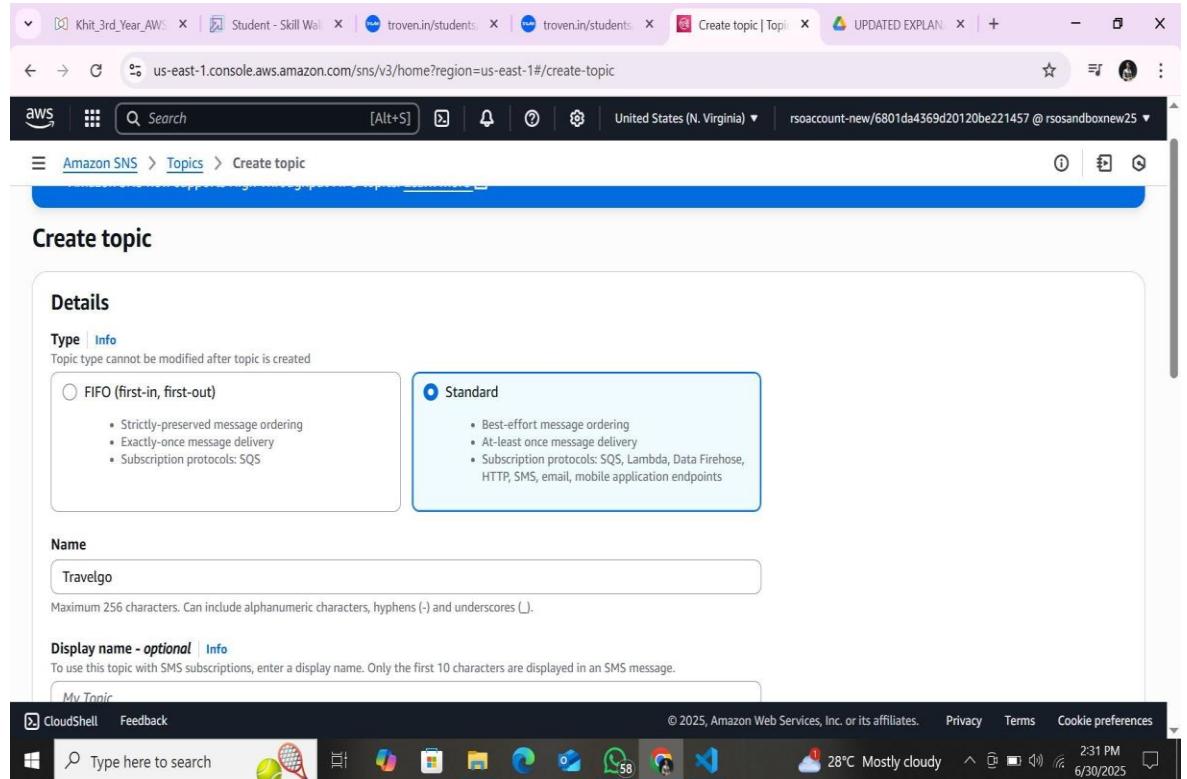
Topic name  
A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

**Next step**

Start with an overview

**Pricing**

- Click on **Create Topic** and choose a name for the topic
- Choose Standard type for general notification use cases and Click on Create Topic.



The screenshot shows a web browser window with multiple tabs open, including "Create topic | Topics" and "UPDATED EXPLANATION". The main content is the "Create topic" page for Amazon SNS. It displays two options: "FIFO (first-in, first-out)" and "Standard". The "Standard" option is selected and highlighted with a blue border. Below the selection, there is a list of features: Best-effort message ordering, At-least once message delivery, and Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints. The "Name" field contains "Travelgo". The "Display name - optional" field is empty. The Windows taskbar at the bottom shows various pinned icons and the current date and time as 6/30/2025.

► **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.

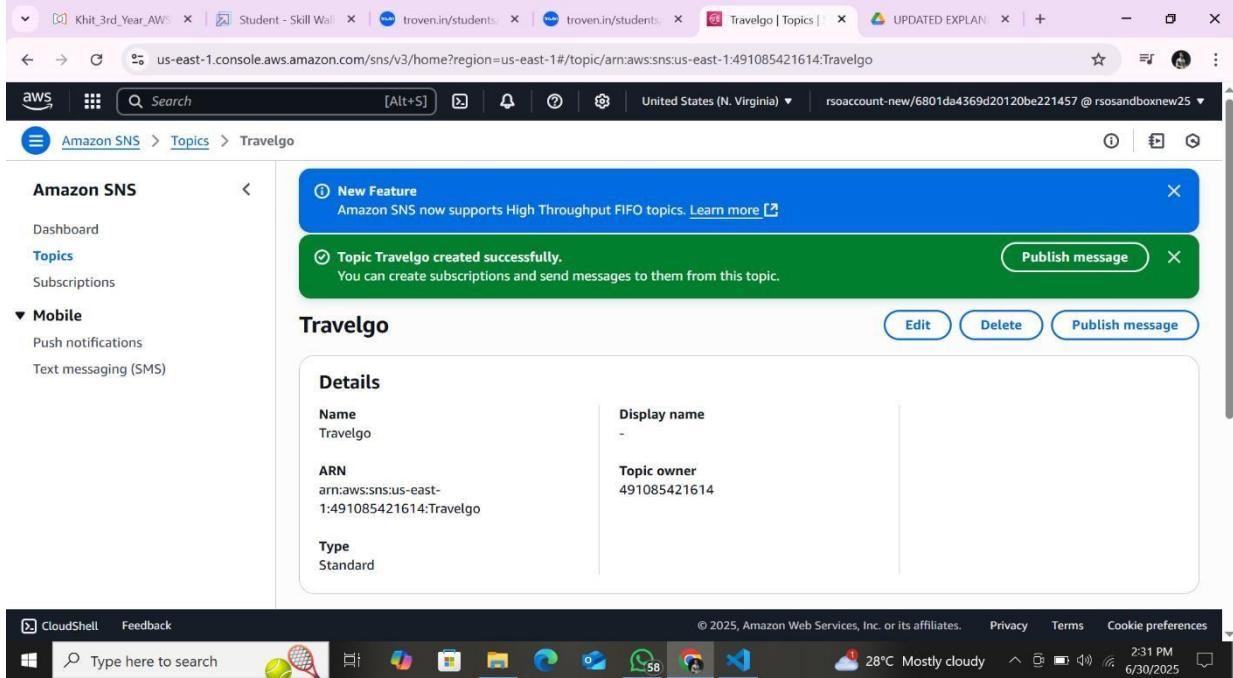
► **Delivery status logging - optional** [Info](#)  
These settings configure the logging of message delivery status to CloudWatch Logs.

► **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**.



The screenshot shows the AWS SNS Topics page. On the left, there's a sidebar with 'Amazon SNS' navigation: Dashboard, Topics (which is selected), and Subscriptions. Below that is a 'Mobile' section with Push notifications and Text messaging (SMS). The main content area shows a 'Travelgo' topic. A blue banner at the top says 'New Feature: Amazon SNS now supports High Throughput FIFO topics. Learn more'. Below it, a green banner says 'Topic Travelgo created successfully. You can create subscriptions and send messages to them from this topic.' with a 'Publish message' button. The 'Travelgo' topic card has sections for 'Details': Name (Travelgo), Display name (-), ARN (arn:aws:sns:us-east-1:491085421614:Travelgo), Topic owner (491085421614), and Type (Standard). At the bottom of the page, there's a Windows taskbar with various icons and system status information.

- **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.
- After subscription request for the mail confirmation
- 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 ▾

9

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](#)

**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**

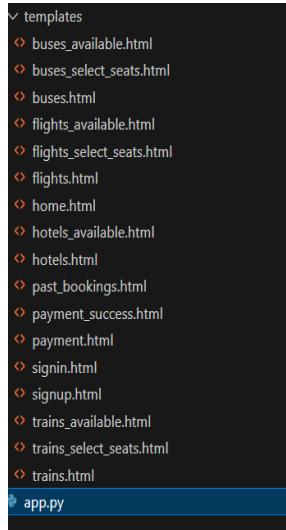
If it was not your intention to subscribe, [click here to unsubscribe](#).

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

## Milestone 4:Backend Development and Application Setup

- **Activity 4.1: Develop the backend using Flask**

- File Explorer Structure



**Description:** set up the TRAVEL GO project with an app.py file, a static/ folder for assets, and a templates/ directory containing all required HTML pages like home, login, register, subject-specific pages (e.g. buse.html, hotel.html, train.html, flight.html).

### Description of the code :

- **Flask App Initialization**

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
import random
import string
import datetime
import boto3
from botocore.exceptions import ClientError
```

**Description:** import essential libraries including Flask utilities for routing, Boto3 for DynamoDB operations, SMTP and email modules for sending mails, and Bcrypt for password hashing and verification

```
app = Flask(__name__)
```

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

- **Dynamodb Setup:**

```
# AWS Configuration
AWS_REGION = 'us-west-2' # Update to your desired region
DYNAMODB_TABLE_USERS = 'Users'
DYNAMODB_TABLE_BOOKINGS = 'Bookings'
SNS_TOPIC_ARN = 'arn:aws:sns:your-region:your-account-id:your-sns-topic' # Update with your ARN
```

**Description:** Initialize the DynamoDB resource for the us-east-1 region and set up access to the Users and Requests tables for storing user details and book requests.

- **SNS Connection**

```
def send_notification(message):
    try:
        response = sns_client.publish(
            TopicArn=SNS_TOPIC_ARN,
            Message=message,
            Subject='Booking Notification'
        )
        return response
    except ClientError as e:
        print(f"Error sending notification: {e}")
        return None
```

**Description:** Configure **SNS** to send notifications when a book request is submitted. Paste your stored ARN link in the sns\_topic\_arn space, along with the region name where the SNS topic is created. Also, specify the chosen email service in SMTP\_SERVER (e.g., Gmail, Yahoo, etc.) and enter the subscribed email in the SENDER\_EMAIL section. Create an 'App password' for the email ID and store it in the SENDER\_PASSWORD section. ● **Routes for Web Pages**

- **Home Route:**

```
@app.route('/home')
def home():
    if 'username' not in session:
        return redirect(url_for('signin'))
    return render_template('home.html', username=session['username'])
```

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

- **Register Route:**

```
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        if get_user(username):
            flash('Username already exists', 'error')
        elif add_user(username, password):
            flash('Account created! Please sign in', 'success')
            return redirect(url_for('signin'))
        else:
            flash('Error creating account', 'error')

    return render_template('signup.html')
```

**Description:** define /register route to validate registration form fields, hash the user password using Bcrypt, store the new user in DynamoDB with a login count, and send an SNS notification on successful registration

- **login Route (GET/POST):**

```
@app.route('/signin', methods=['GET', 'POST'])
def signin():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        user = get_user(username)
        if user and user['password'] == password:
            session['username'] = username
            flash('Login successful!', 'success')
            return redirect(url_for('home'))
        else:
            flash('Invalid credentials', 'error')

    return render_template('signin.html')
```

**Description:** define /login route to validate user credentials against DynamoDB, check the password using Bcrypt, update the login count on successful authentication, and redirect users to the home page

- **Train , Bus and Hotel routes:**

```

mock_travel_options = {
    'flights': [
        {'id': 'flight1', 'flight_number': 'AC101', 'airline': 'AirConnect',
         'departure_time': '06:00 AM', 'arrival_time': '08:00 AM', 'price': 150.00, 'duration': '2h 00m'},
        {'id': 'flight2', 'flight_number': 'SH202', 'airline': 'SkyHigh',
         'departure_time': '09:30 AM', 'arrival_time': '11:30 AM', 'price': 180.00, 'duration': '2h 00m'},
    ],
    'buses': [
        {'id': 'bus1', 'name': 'Express Lines', 'departure_time': '08:00 AM',
         'arrival_time': '12:00 PM', 'price': 25.00, 'total_seats': 20},
    ],
    'trains': [
        {'id': 'train1', 'number': 'REX001', 'name': 'Rail Express',
         'departure_time': '07:00 AM', 'arrival_time': '01:00 PM', 'price': 60.00},
    ],
    'hotels': [
        {'id': 'hotel1', 'name': 'Grand Plaza Hotel', 'location': 'New York',
         'price_per_night': 120.00, 'rating': 4.5},
    ]
}

```

**Description:** define /home-page to render the main homepage, /ebook-buttons to handle subject selection and redirection, and /<subject>.html dynamic route to render subjectspecific pages like Mathematics or English.

- **Request Routes:**

```

@app.route('/book/<transport_type>', methods=['GET', 'POST'])
def book_transport(transport_type):
    if 'username' not in session:
        return redirect(url_for('signin'))

    if request.method == 'POST':
        session['current_booking'] = {
            'type': transport_type,
            'source': request.form.get('source'),
            'destination': request.form.get('destination'),
            'date': request.form.get('date'),
        }
        return redirect(url_for(f'{transport_type}_available'))

    return render_template(f'{transport_type}.html')

@app.route('/book/<transport_type>/available')
def transport_available(transport_type):
    if 'username' not in session:
        return redirect(url_for('signin'))

    booking = session.get('current_booking')
    if not booking or booking['type'] != transport_type:
        return redirect(url_for(f'book_{transport_type}'))

    options = mock_travel_options.get(transport_type, [])
    return render_template(f'{transport_type}_available.html',
                           options=options,
                           booking=booking)

@app.route('/payment', methods=['GET', 'POST'])
def payment():
    if 'username' not in session:
        return redirect(url_for('signin'))

    booking = session.get('current_booking')
    if not booking:
        return redirect(url_for('home'))

    if request.method == 'POST':
        booking_id = generate_unique_id()
        booking_details = {
            'booking_id': booking_id,
            'username': session['username'],
            'type': booking['type'],
            'status': 'confirmed',
            'created_at': datetime.datetime.now().isoformat(),
            'details': booking
        }

        if add_booking(booking_details):
            send_notification(
                f"New booking: {booking_id}\n"
                f"Type: {booking['type']}\n"
                f"User: {session['username']}"

```

### Deployment Code:

```

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)

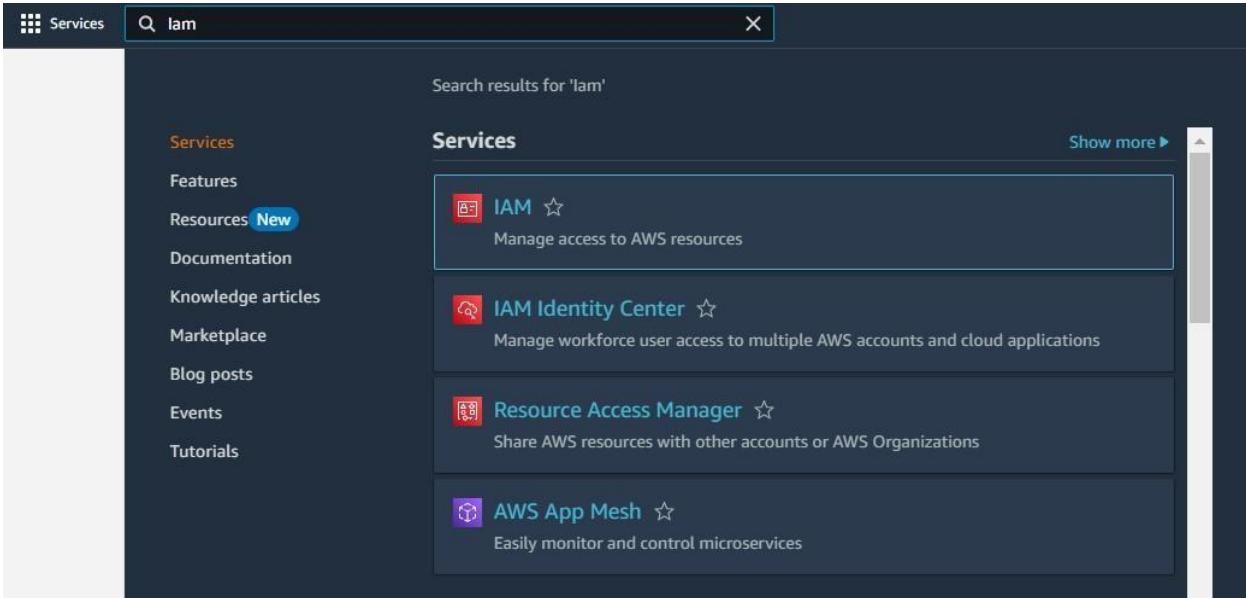
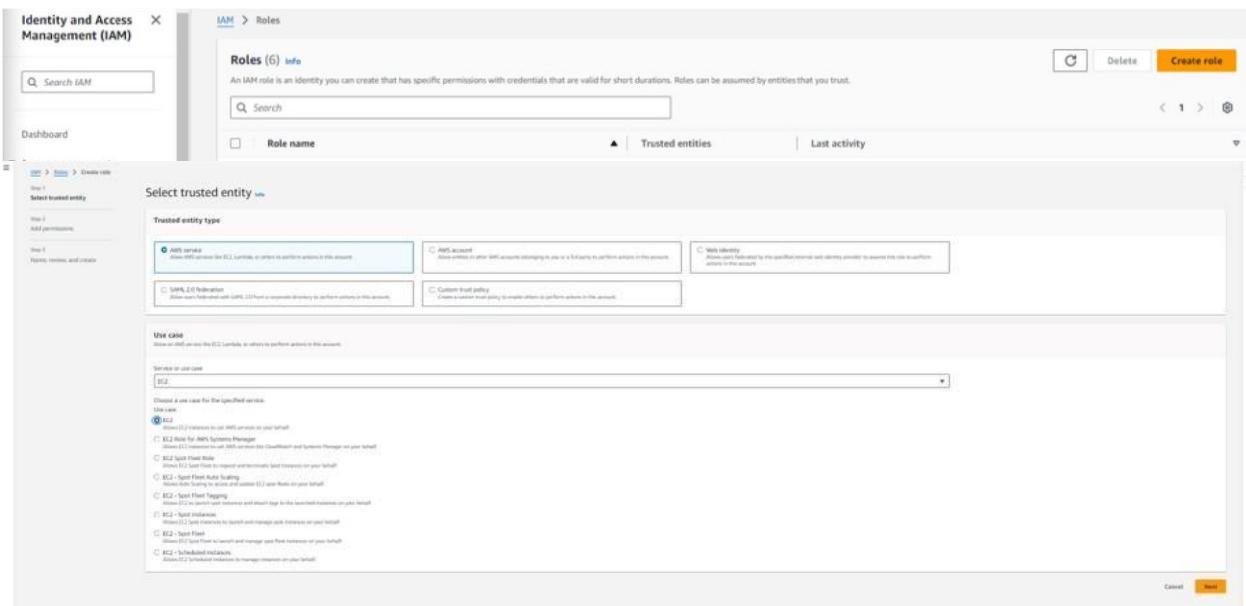
```

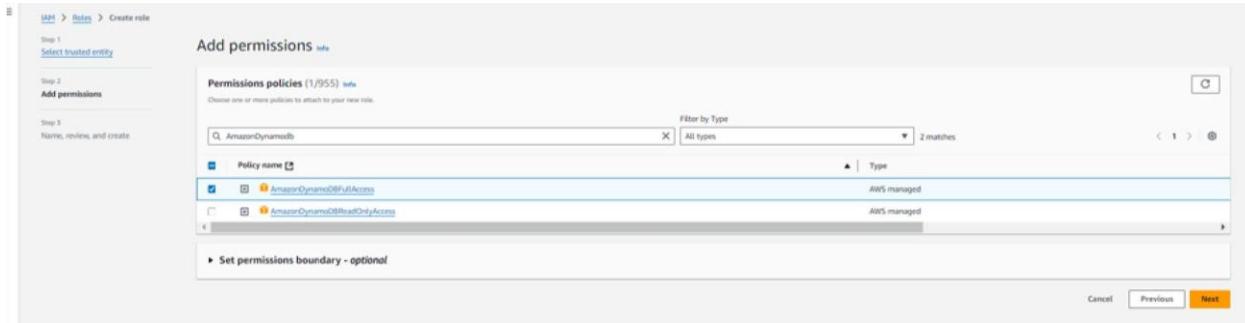
**Description:** start the Flask server to listen on all network interfaces (0.0.0.0) at port 80 with debug mode enabled for development and testing.

## Milestone 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.

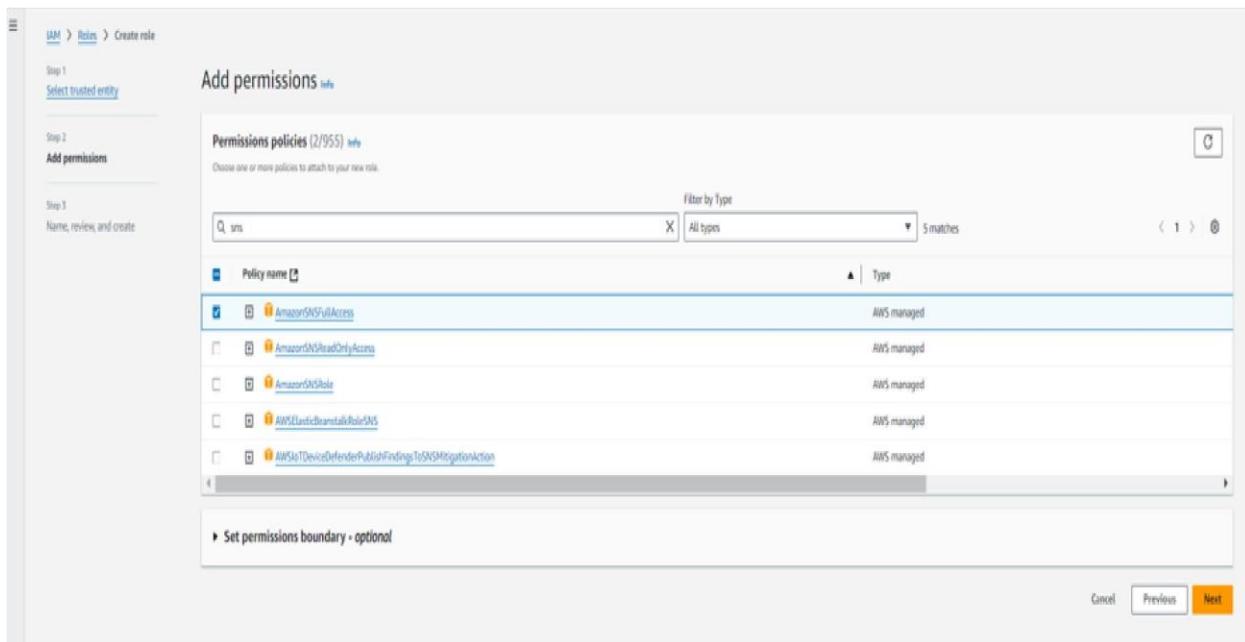





## ● Activity 5.2: Attach Policies.

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.





[Select trusted entities](#)

**Name, review, and create**

[Step 1: Select trusted entities](#)

**Role details**

**Role name**  
Enter a meaningful name to identify this role.  
`sns_Dynamodb_role`  
Maximum 64 characters, the characters used: alphanumeric and \_-+=@.

**Amazon SNS permissions**  
Allows EC2 instances to call AWS services on your behalf.  
Allows EC2 instances to call AWS services on your behalf.  
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), underscores (\_), or any of the following characters: ~-=!@#\$%^&\*()

**Step 1: Select trusted entities**

**Trust policy**

```
1  "Version": "2012-10-17",
2  "Statement": [
3      {
4          "Effect": "Allow",
5          "Principal": "*",
6          "Action": "sts:AssumeRole"
7      }
8  ]
```

**Step 2: Add permissions**

**Permissions policy summary**

Policy name	Type	Attached as
<a href="#">AmazonDynamoDBFullAccess</a>	AWS managed	Permissions policy
<a href="#">AmazonSNSFullAccess</a>	AWS managed	Permissions policy

**Step 3: Add tags**

Add tags - optional info.  
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

[Add new tag](#)  
You can add up to ten tags.

[Cancel](#) [Previous](#) [Create next](#)

**IAM > Roles > sns\_Dynamodb\_role**

**sns\_Dynamodb\_role** [Info](#) [Delete](#)

**Summary**

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

**Last activity**  
 6 days ago

**ARN**  
 arn:aws:iam::557690616836:role/sns\_Dynamodb\_role

**Instance profile ARN**  
 arn:aws:iam::557690616836: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.

**Filter by Type**

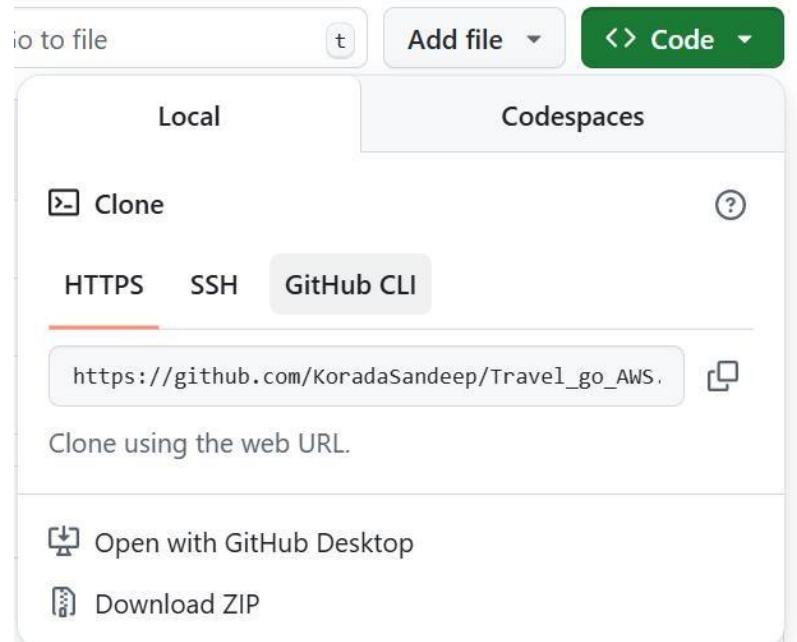
**Search**

Policy name	Type	Attached entities
<a href="#">AmazonDynamoDBFullAccess</a>	AWS managed	4
<a href="#">AmazonSNSFullAccess</a>	AWS managed	2

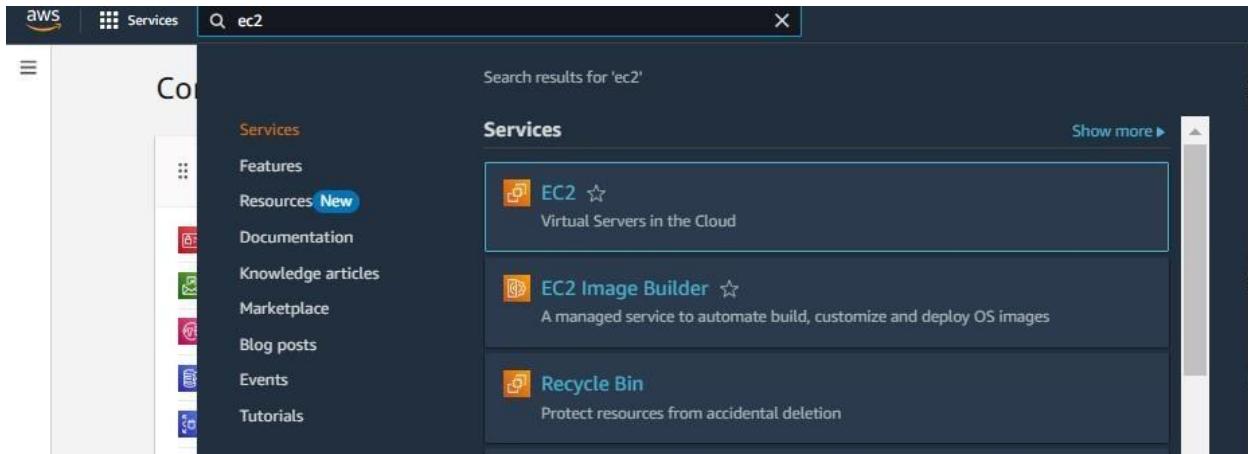
## Milestone 6: EC2 Instance Setup

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

 static	Initial commit
 templates	Update statistics.html
 app.py	Update app.py



- 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.

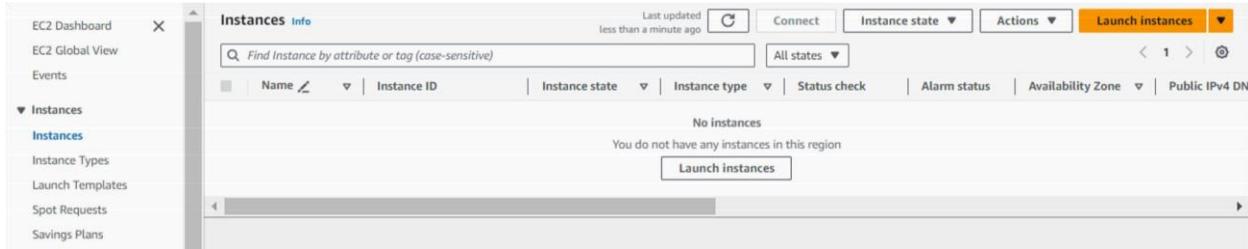


Search results for 'ec2'

**Services**

- EC2
- EC2 Image Builder
- Recycle Bin

- Click on Launch instance to launch EC2 instance



Instances Info

Last updated less than a minute ago

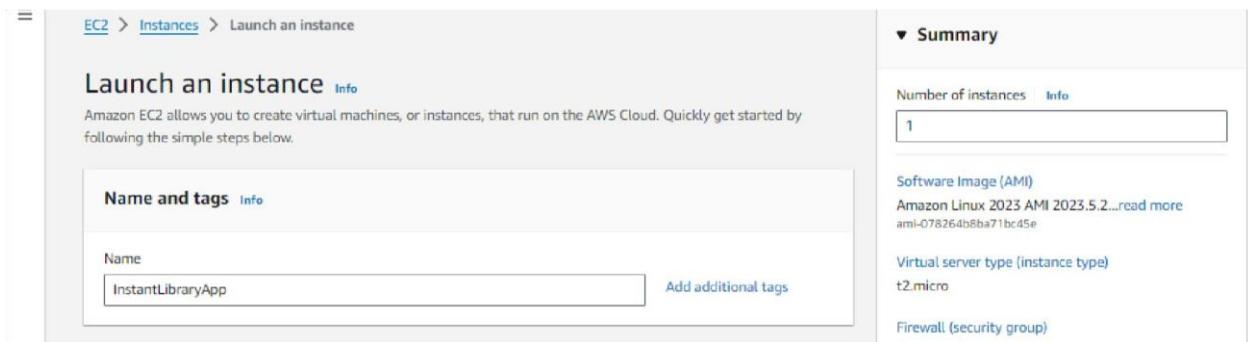
Find Instance by attribute or tag (case-sensitive)

All states

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 Info

1

Software Image (AMI)

Amazon Linux 2023.5.2...read more

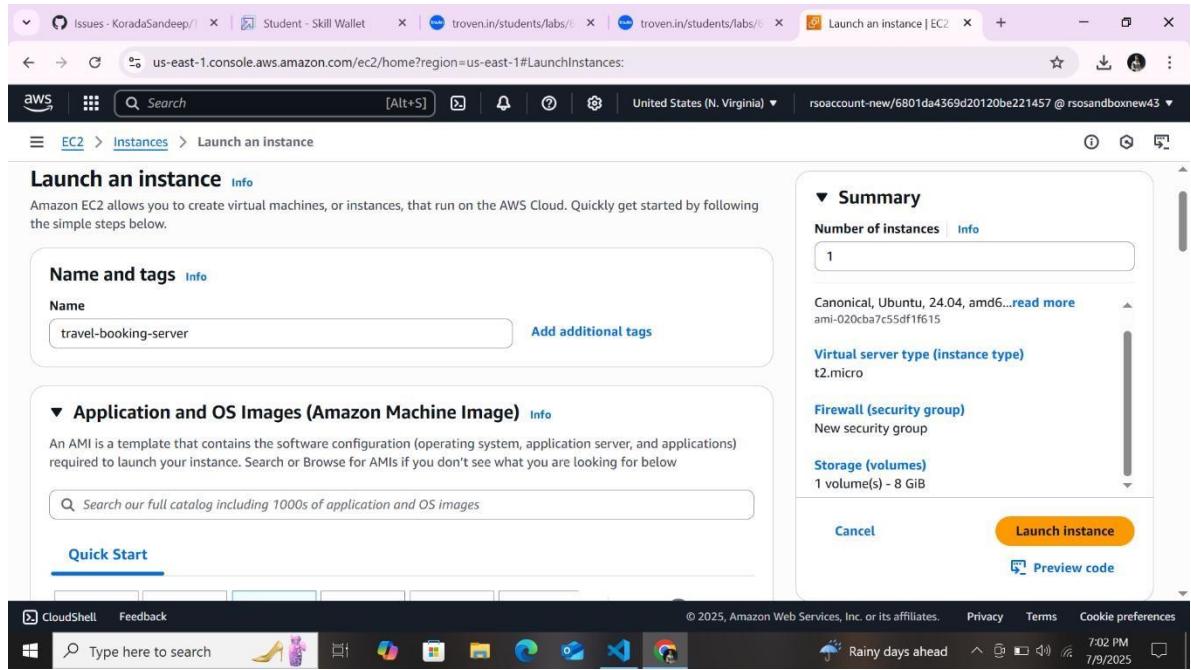
ami-078264b8ba71bc45e

Virtual server type (instance type)

t2.micro

Firewall (security group)

- Choose Amazon Linux 2



Issues - KoradaSandip/ | Student - Skill Wallet | troven.in/students/labs/ | troven.in/students/labs/ | Launch an instance | EC2 | + | - | × | ↗ us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances: | Search [Alt+S] | Search | United States (N. Virginia) | rsoaccount-new/6801da4369d20120be221457 @ rsosandboxnew43 |

aws | 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**  
travel-booking-server Add additional tags

**Application and OS Images (Amazon Machine Image) Info**

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.

Search our full catalog including 1000s of application and OS images

**Quick Start**

CloudShell Feedback Type here to search       

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences Rainy days ahead 7:00 PM 7/9/2025

**Summary**

Number of instances Info  
1

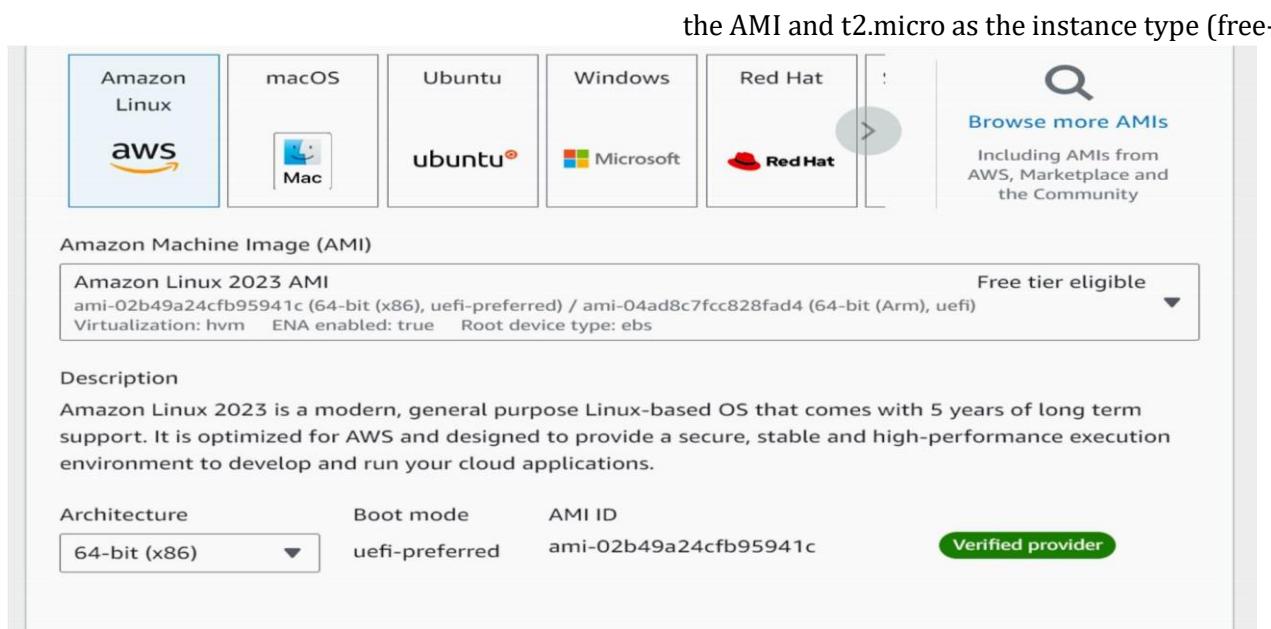
Canonical, Ubuntu, 24.04, amd64... read more  
ami-020cba7c55df1f615

Virtual server type (instance type)  
t2.micro

Firewall (security group)  
New security group

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

Cancel **Launch instance** Preview code



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	<b>Verified provider</b>
64-bit (x86)	uefi-preferred	ami-02b49a24cfb95941c	

**Browse more AMIs**  
Including AMIs from AWS, Marketplace and the Community

- Create and download the key pair for Server access.

Issues - KoradaSandeep/ | Student - Skill Wallet | troven.in/students/labs/ | troven.in/students/labs/ | Launch an instance | EC2 | +

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

aws Search [Alt+S] United States (N. Virginia) rsoaccount-new/6801da4369d20120be221457 @ rpsandboxnew43

EC2 > Instances > Launch an instance

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

Instance type

t2.micro Family: t2 1 vCPU 1 GiB Memory Current generation: true

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

travel\_go [Create new key pair](#)

**Summary**

Number of instances [Info](#)

1

Canonical, Ubuntu, 24.04, amd64... [read more](#)

ami-020cba7c55df1f615

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

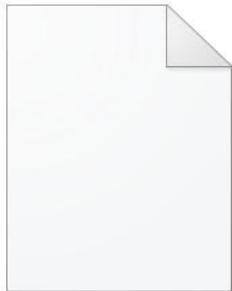
Storage (volumes)

1 volume(s) - 8 GiB

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

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

Type here to search Rainy days ahead 7:02 PM 7/9/2025



travel\_go.pem

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

Issues · KoradaSandeep/ · Student - Skill Wallet · troven.in/students/labs/ · troven.in/students/labs/ · Launch an instance | EC2 · +

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

aws Search [Alt+S] United States (N. Virginia) rsoaccount-new/6801da4369d20120be221457 @ rsosandboxnew43

EC2 > Instances > Launch an instance

Type | Info Protocol | Info Port range | Info

ssh TCP 22

Source type | Info Source | Info Description - optional | Info

Anywhere Add CIDR, prefix list or security e.g. SSH for admin desktop

0.0.0.0/0 X

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

Type | Info Protocol | Info Port range | Info

HTTP TCP 80

Source type | Info Source | Info Description - optional | Info

Anywhere Add CIDR, prefix list or security e.g. SSH for admin desktop

0.0.0.0/0 X

⚠ 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. X

▼ Summary

Number of instances | Info 1

Canonical, Ubuntu, 24.04, amd6...read more ami-020cba7c55df1f615

Virtual server type (instance type) t2.micro

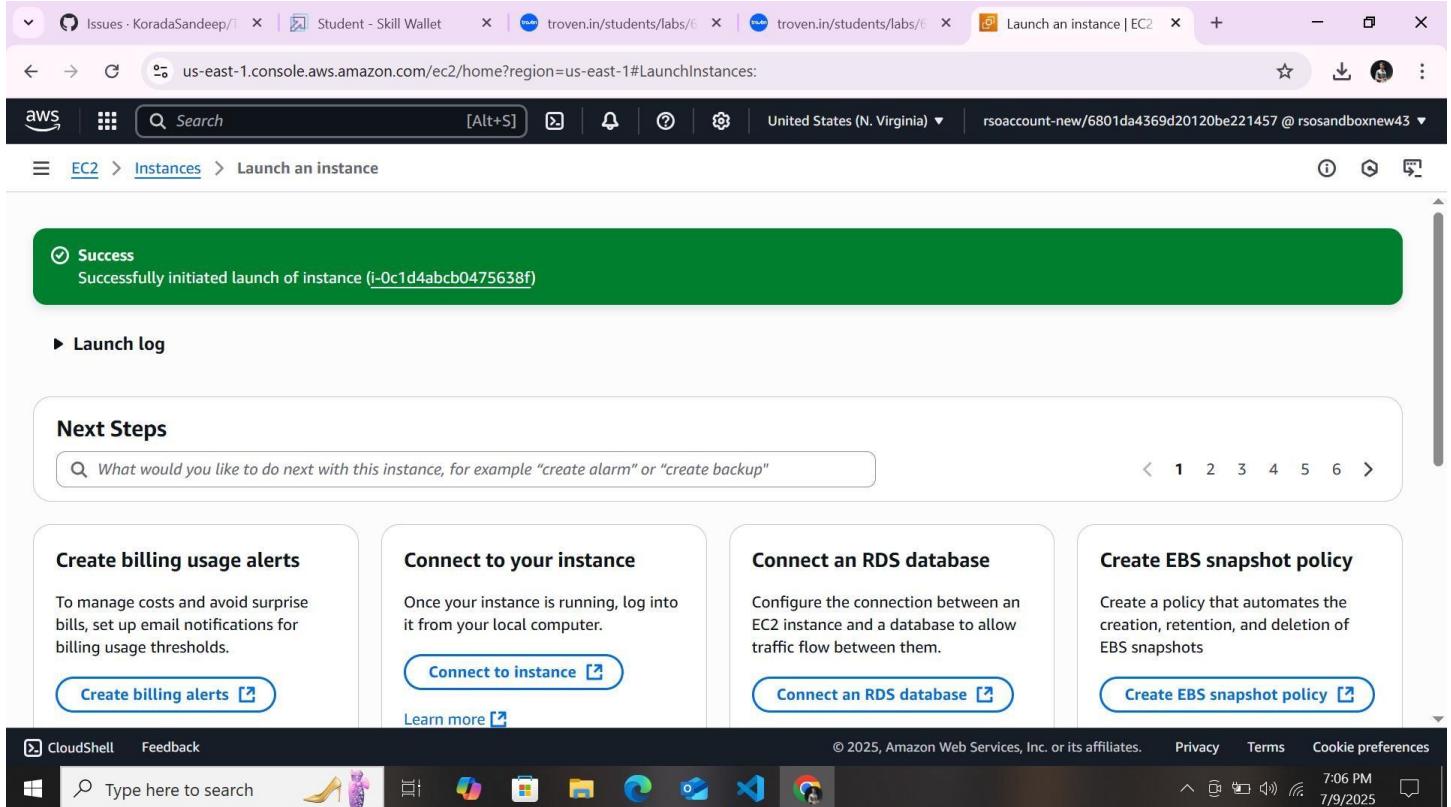
Firewall (security group) New security group

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

Cancel Launch instance Preview code

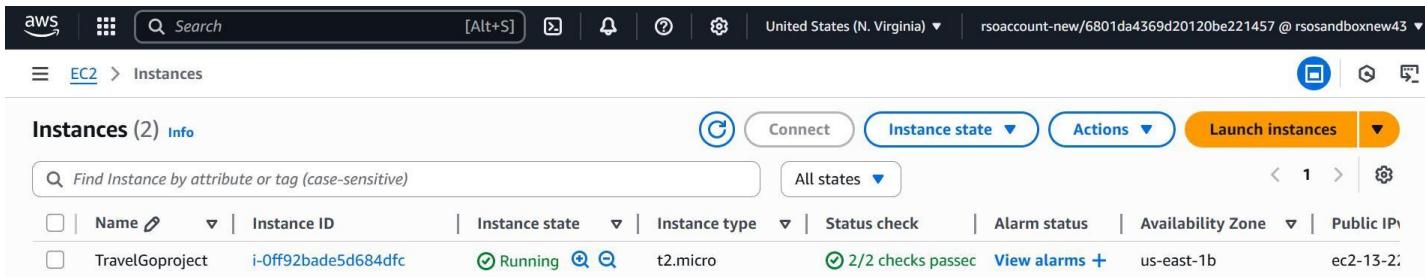
CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Rainy days ahead 7:02 PM 7/9/2025



The screenshot shows the AWS EC2 'Launch an instance' page. At the top, there is a green success message box stating: "Success Successfully initiated launch of instance (i-0c1d4abcb0475638f)". Below this, there is a "Launch log" section. Under "Next Steps", there are four cards: "Create billing usage alerts", "Connect to your instance", "Connect an RDS database", and "Create EBS snapshot policy". Each card has a corresponding "Create [service] alerts" or "Create [service] policy" button. The bottom of the page shows the Windows taskbar.

- 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.



The screenshot shows the AWS EC2 Instances page. It displays two instances: "TravelGoproject" (Instance ID: i-0ff92bade5d684dfc, State: Running, Type: t2.micro) and another instance (Instance ID: ec2-13-2, State: Pending). The page includes filters for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP. There are also buttons for Connect, Instance state, Actions, and Launch instances.

**i-Off92bade5d684dfc (TravelGoproject)**

[Details](#) | [Status and alarms](#) | [Monitoring](#) | [Security](#) | [Networking](#) | [Storage](#) | [Tags](#)
**▼ Instance summary [Info](#)**
**Instance ID**  
 i-Off92bade5d684dfc

**Public IPv4 address**  
 13.220.213.173 | [open address](#)
**Private IPv4 addresses**  
 172.31.94.144

**IPv6 address**  
 -

**Instance state**  
 Running

**Public DNS**  
 ec2-13-220-213-173.compute-1.amazonaws.com  
[| open address](#)
**Hostname type**  
 IP name: ip-172-31-94-144.ec2.internal

**Private IP DNS name (IPv4 only)**  
 ip-172-31-94-144.ec2.internal

Screenshot of a web browser showing the AWS EC2 Instances Modify IAM Role page for instance i-Off92bade5d684dfc.

The browser tabs include: KoradaSan, Student - S, troven.in/s, troven.in/s, Modify IAM Role, EC2 Instances, Google Ge, botcore.e, and others.

The URL is: us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyIAMRole:instanceId=i-Off92bade5d684dfc

The AWS navigation bar shows: aws, Search, [Alt+S], United States (N. Virginia), rsoaccount-new/6801da4369d20120be221457 @ rsosandboxnew43.

The breadcrumb navigation is: EC2 > Instances > i-Off92bade5d684dfc > Modify IAM role

### Modify IAM role [Info](#)

Attach an IAM role to your instance.

**Instance ID**  
 i-Off92bade5d684dfc (TravelGoproject)

**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.

EC2\_DynamoDB\_SNS\_Role1 Create new IAM role

[Cancel](#) [Update IAM role](#)

- Now connect the EC2 with the files

## Connect to instance Info

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

[EC2 Instance Connect](#)

[Session Manager](#)

[SSH client](#)

[EC2 serial console](#)



### 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 X

i **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.

[Cancel](#)

[Connect](#)



Smart  
Internz

```
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/AlekhyaPenubakula/InstantLibrary.git' • 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
      #_
      _\###_
     ~\_\###\_
     ~~\###|_
     ~~ \#/   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-publicip>**

- 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 -
```

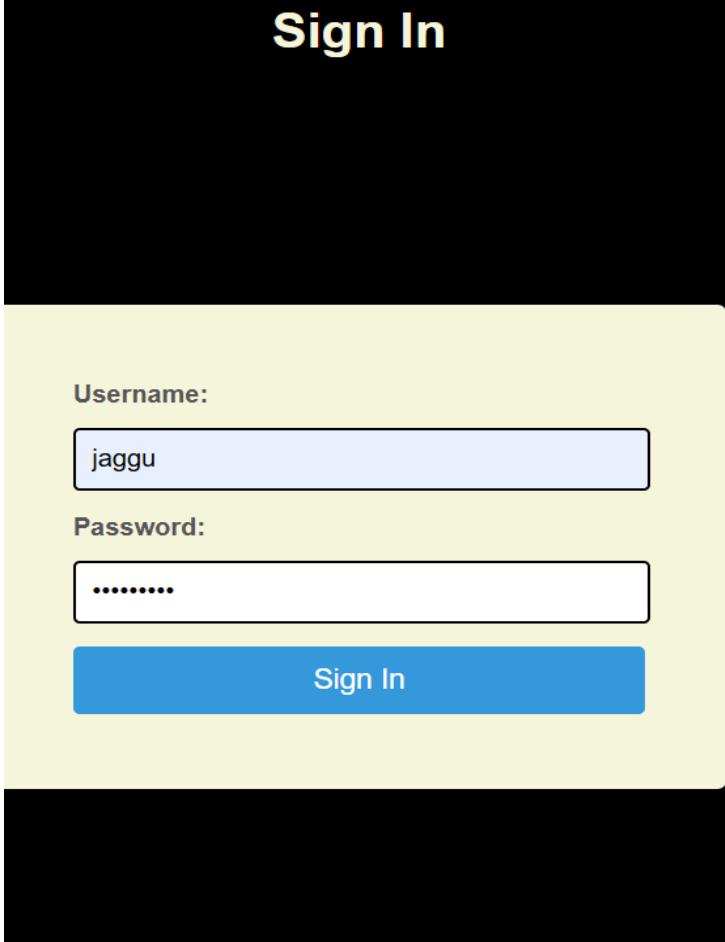
### Access the website through:

Public IPs: <http://3.92.196.29:5000>

## Milestone 8: Testing and Deployment

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

**Login Page:**



The image shows a login interface. At the top, there is a black header with the word "Sign In" in white. Below the header is a light yellow rectangular form. Inside the form, there is a label "Username:" followed by a text input field containing the text "jaggu". Below that is a label "Password:" followed by another text input field containing six dots ("....."). At the bottom of the form is a blue rectangular button with the text "Sign In" in white. The entire form is set against a black background.

**Signup page:**

### Create Account

Email:

Username:

Password:

Confirm Password:

[Sign Up](#)

**Home page:**

Welcome, jaggu!

Book Your Next Journey

**Flights**

Flights

[Book Now](#)

**Hotels**

Hotels

[Book Now](#)

**Buses**

Buses

[Book Now](#)

**Trains**

Trains

[Book Now](#)

**Past Bookings**

Past Bookings

[View All](#)

**About Bookings page:**

### Your Past Bookings

[Back to Home](#)

**Flights Booking** \$150.00

Booking ID: UT01NSEWMS  
Booked On: 2025-07-15

**Flight:** AirConnect AC101      **Route:** srikakulam to vishakapatnam

**Travel Date:** 2025-07-24      **Seat Number:** T7YZG4

**Payment Method:** upi

**Hotels Booking** \$1200.00

Booking ID: OT7YAIHC7V  
Booked On: 2025-07-15

**Hotel:** Berlin Central Hotel (Berlin)      **Location:** Berlin

**Check-in:** 2025-07-17      **Check-out:** 2025-07-29

**Payment Method:** upi

**Trains Booking** \$75.00

Booking ID: OCY0W0JEPV  
Booked On: 2025-07-15

**Train:** Metro Rail MET002      **Route:** srikakulam to kakinada

**Travel Date:** 2025-07-18      **Ticket Number:** V991E9

**Payment Method:** upi

## Contact Page:

### Contact Us

Have questions, feedback, or need assistance? Our dedicated support team is here to help you.

Email: [support@travelease.com](mailto:support@travelease.com)

Phone: +1 (800) 123-4567

Address: 123 Travel Lane, Adventure City, Global 56789

Customer Service Hours: Monday - Friday, 9:00 AM - 6:00 PM (Your Local Time)

We look forward to hearing from you and helping you with your travel needs!

© 2025 TravelEase. All rights reserved.

f t o in

## About page:

### About TravelEase

Welcome to **TravelEase**, your one-stop solution for seamless travel planning and booking. We believe that exploring the world should be an effortless and enjoyable experience from start to finish. Our platform is designed to provide you with the best deals on flights, hotels, buses, and trains, all in one convenient place.

Our mission is to empower travelers with easy access to comprehensive travel options, personalized recommendations, and a user-friendly interface. Whether you're planning a quick getaway or an elaborate international adventure, TravelEase is here to make your journey stress-free.

Join the growing community of happy travelers who trust TravelEase for their adventures. Start your journey with us today!

### Hotels Booking :

**Hotel Booking**

Location:

Check-in Date:  dd-mm-yyyy

Check-out Date:  dd-mm-yyyy

**Find Hotels**

[Back to Home](#)

### Trains Booking:

### Book a Train

From:

To:

Travel Date:  dd - mm - yyyy

**Book Train**

[Back to Home](#)

## Available Trains:

### Available Trains

[Back to Home](#)

**Rail Express REX001** \$60.0  
Departing from: vishakapatnam to delhi on 2025-07-25  
🕒 Departure: 07:00 AM 🕒 Arrival: 01:00 PM

**Metro Rail MET002** \$75.0  
Departing from: vishakapatnam to delhi on 2025-07-25  
🕒 Departure: 11:00 AM 🕒 Arrival: 05:00 PM

**Speedy Freight SFR003** \$65.0  
Departing from: vishakapatnam to delhi on 2025-07-25  
🕒 Departure: 09:00 AM 🕒 Arrival: 03:00 PM

## Buses Booking:

### Bus Booking

Source City

Destination City

Travel Date  
 📅

**Find Buses**

[Back to Home](#)

## Available Buses for vishakapatnam to Mumbai on 2025-07-16

**Express Lines**  
Departure: 08:00 AM  
Arrival: 12:00 PM  
Price: \$25.0

**Select**

**City Link**  
Departure: 10:00 AM  
Arrival: 02:00 PM  
Price: \$30.0

**Select**

**Green Transport**  
Departure: 03:00 PM  
Arrival: 07:30 PM  
Price: \$28.0

**Select**

[Back to Bus Booking](#)

### Select Your Seats

#### Express Lines

08:00 AM - 12:00 PM

Price: \$25.0 per seat



Available    Selected    Occupied

**Continue to Payment**

[Back to Available Buses](#)

## Flights Booking:

### Book a Flight

From:

To:

Departure Date:

[Back to Home](#)

### Available Flights

[Back to Home](#)

**AirConnect AC101**

Departing from: vishakapatnam to delhi on 2025-07-18

🕒 Departure: 06:00 AM      🕒 Arrival: 08:00 AM      🕒 Duration: 2h 00m

**SkyHigh SH202**

Departing from: vishakapatnam to delhi on 2025-07-18

🕒 Departure: 09:30 AM      🕒 Arrival: 11:30 AM      🕒 Duration: 2h 00m

**BlueJet BL303**

Departing from: vishakapatnam to delhi on 2025-07-18

🕒 Departure: 01:00 PM      🕒 Arrival: 03:30 PM      🕒 Duration: 2h 30m

## Complete Your Booking

### Booking Details:

Type: flights

From: vishakapatnam

To: delhi

Date: 2025-07-18

#### Select Payment Method:

- Credit Card
- PayPal
- Bank Transfer
- upi

**Complete Payment**

[Cancel and Return Home](#)

## Conclusion:

The TravelGo Website has been successfully developed and deployed using a scalable and cloudnative architecture. Leveraging AWS services such as EC2 for hosting, DynamoDB for real-time data management, and SNS for instant booking and cancellation notifications, the platform provides a seamless travel booking experience for users. TravelGo enables registered users to search and book buses, trains, flights, and hotels in a centralized, intuitive interface, eliminating the complexities of navigating multiple travel services.

The cloud infrastructure ensures high availability and smooth performance even during peak usage, while the Flask backend ensures efficient handling of user authentication, dynamic booking flows, and data transactions. Real-time notification integration via AWS SNS allows users to receive booking confirmations and cancellations immediately via email, improving communication and user engagement.

In summary, the TravelGo Website offers a modern, reliable, and user-friendly solution for managing travel and accommodation needs. It highlights the potential of cloud-based platforms in building unified travel systems, simplifying operations, and enhancing the overall user experience.