

## HomeMade Pickles & Snacks: Taste the Best Resources

### Project Description:

Home Made Pickles & Snacks — Taste the Best is a cloud-based culinary platform revolutionizing access to authentic, handcrafted pickles and snacks. Addressing the growing demand for preservative-free, traditional recipes, this initiative combines artisanal craftsmanship with cutting-edge technology to deliver farm-fresh flavors directly to consumers. Built on Flask for backend efficiency and hosted on AWS EC2 for scalable performance, the platform offers seamless browsing, ordering, and subscription management. DynamoDB ensures real-time inventory tracking and personalized user experiences, while fostering sustainability through partnerships with local farmers and eco-friendly packaging. From tangy regional pickles to wholesome snacks, every product celebrates heritage recipes, nutritional integrity, and convenience—proving that tradition and innovation can coexist deliciously. "Preserving Traditions, One Jar at a Time."

### Scenario 1: Scalable Order Management for High Demand

A cloud-based system ensures seamless order processing during peak user activity. For instance, during a promotional event, hundreds of users simultaneously access the platform to place orders. The backend efficiently processes requests, updates inventory in real-time, and manages user sessions. The cloud infrastructure handles traffic spikes without performance degradation, ensuring smooth transactions and minimizing wait times.

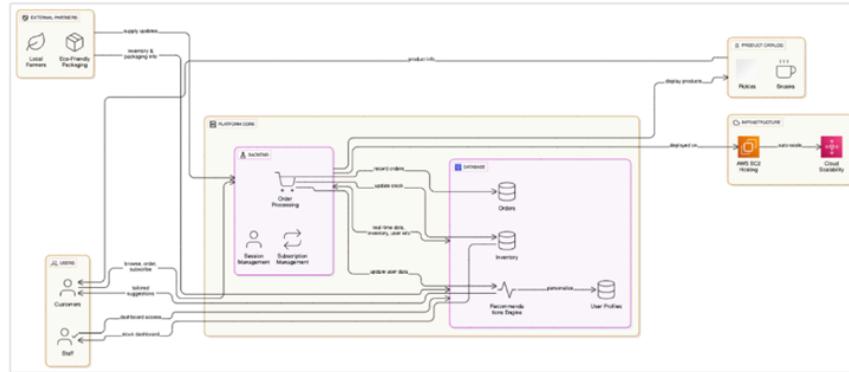
### Scenario 2: Real-Time Inventory Tracking and Updates

When a customer places an order for a product, the system instantly updates stock levels and records transaction details. For example, a user purchases an item, triggering automatic inventory deduction and order confirmation. Staff members receive updated dashboards to monitor stock availability and fulfillment progress, ensuring timely restocking and minimizing overselling risks.

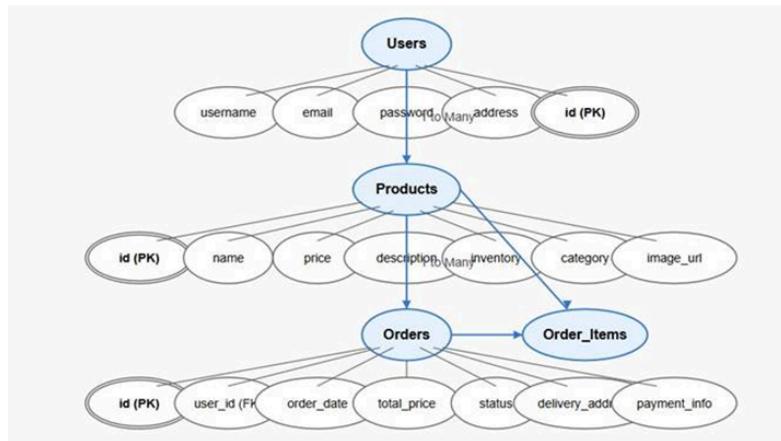
### Scenario 3: Personalized User Experience and Recommendations

The platform leverages user behavior data to enhance engagement. A returning customer, for instance, views tailored recommendations based on past purchases and browsing history. The system dynamically adjusts suggestions in real-time, while maintaining fast response rates even during high traffic, creating a frictionless and intuitive shopping experience.

## AWS ARCHITECTURE



Entity Relationship (ER)Diagram:



## Pre-requisites:

1. AWS Account Setup: [AWS Account Setup](#)
2. AWS IAM (Identity and Access Management): [IAM Overview](#)
3. AWS EC2 (Elastic Compute Cloud): [EC2 Tutorial](#)
4. AWS DynamoDB: [DynamoDB Introduction](#)
5. Git Documentation: [GIT Documentation](#)
6. VS Code Installation: [Download Visual Studio Code](#)

## 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

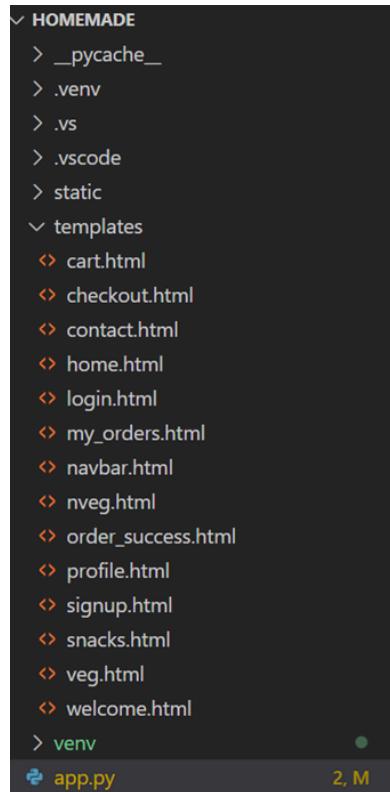
### Milestone 8. Testing and Deployment

- Conduct functional testing to verify user signup, login, buy/sell stocks and notifications.

## Milestone 1:Backend Development and Application Setup

## 1. Activity 1.1: Develop the backend using Flask

### a. File Explorer Structure



**Description:** set up the Home Made pickles and snacks 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., veg\_pickles.html, non\_veg\_pickles.html), and utility pages (e.g., checkout-form.html, profile.html).

### Description of the code :

#### 1. Flask App Initialization

```
from flask import Flask, render_template, request, redirect, url_for
import boto3
from boto3.dynamodb.conditions import Key
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from bcrypt import hashpw, gensalt, checkpw
```

**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.

## 2. Dynamodb Setup:

```
# Dynamodb setup
AWS_REGION = os.environ.get('AWS_REGION', 'us-east-1')
dynamodb = boto3.resource('dynamodb', region_name=AWS_REGION)
users_table = dynamodb.Table('users')
carts_table = dynamodb.Table('carts')
orders_table = dynamodb.Table('orders')
```

**Description:** initialize the DynamoDB resource for the ap-south-1 region and set up access to the Users and Requests tables for storing user details and order requests.

## 3. SNS Connection

```

SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
SENDER_EMAIL = os.environ.get('SENDER_EMAIL', "dileep6@gmail.com")
SENDER_PASSWORD = os.environ.get('SENDER_PASSWORD', "vcbt qcsp yvgi dyhn")

def send_email(to_email, subject, body):
    msg = MIMEText(body)
    msg['From'] = SENDER_EMAIL
    msg['To'] = to_email
    msg['Subject'] = subject
    msg.attach(MIMEText(body, 'plain'))
    try:
        import smtplib
        server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
        server.starttls()
        server.login(SENDER_EMAIL, SENDER_PASSWORD)
        text = msg.as_string()
        server.sendmail(SENDER_EMAIL, to_email, text)
        server.quit()
        print("Email sent successfully")
    except Exception as e:
        print(f"Failed to send email: {e}")

```

**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.

## 4. Routes for Web Pages

- **Home Route:**

```

@app.route('/home')
def home():
    if 'user' in session:
        return render_template('home.html', username=session['user'])
    return redirect(url_for('login'))

```

**Description:** Home page contains the routing for different categories which are Veg\_pickles, Non\_Veg\_pickles, Snacks

- **Signup Route:**

```
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        hashed_password = generate_password_hash(password)
        try:
            c.execute('INSERT INTO users (username, email, password) VALUES (?, ?, ?)', (username, email, hashed_password))
            conn.commit()
            flash('Signup successful! Please log in.')
            return redirect(url_for('login'))
        except sqlite3.IntegrityError:
            error = 'Username or email already exists.'
            return render_template('signup.html', error=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 routing:**

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        c.execute('SELECT password FROM users WHERE username = ?', (username,))
        user = c.fetchone()
        if user and check_password_hash(user[0], password):
            session['user'] = username
            return redirect(url_for('home'))
        else:
            return render_template('login.html', error='Invalid credentials')
    return render_template('login.html')
```

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

### Add Cart routes

```

@app.route('/add_to_cart', methods=['POST'])
def add_to_cart():
    if 'user' not in session:
        flash("You need to login first")
        return redirect(url_for('login'))

    username = session['user']
    name = request.form['name']
    price = int(request.form['price'])
    image = request.form['image']

    # Check if item already in cart
    carts_c.execute('SELECT qty FROM carts WHERE username = ? AND name = ?', (username, name))
    row = carts_c.fetchone()
    if row:
        carts_c.execute('UPDATE carts SET qty = qty + 1 WHERE username = ? AND name = ?', (username, name))
    else:
        carts_c.execute('INSERT INTO carts (username, name, price, image, qty) VALUES (?, ?, ?, ?, ?)', (username, name, price, image, 1))
    carts_conn.commit()
    flash(f"{name} added to cart!")
    return redirect(request.referrer or url_for('home'))

```

## Checkout Routes:

```

@app.route('/checkout', methods=['GET', 'POST'])
def checkout():
    if 'user' not in session:
        flash("You must be logged in to checkout.")
        return redirect(url_for('login'))

    if request.method == 'POST':
        name = request.form['name']
        address = request.form['address']
        phone = request.form['phone']
        payment = request.form['payment']

        if not name or not address or not phone or not payment:
            error = "All fields are required."
            return render_template('checkout.html', error=error)

        username = session['user']
        carts_c.execute('SELECT name, price, image, qty FROM carts WHERE username = ?', (username,))
        cart = [
            {'name': row[0], 'price': row[1], 'image': row[2], 'qty': row[3]}
            for row in carts_c.fetchall()
        ]
        for item in cart:
            orders_c.execute('INSERT INTO orders (username, name, price, image, qty) VALUES (?, ?, ?, ?, ?)', (username, item['name'], item['price'], item['image'], item['qty']))
        orders_conn.commit()
        # Clear user's cart after order
        carts_c.execute('DELETE FROM carts WHERE username = ?', (username,))
        carts_conn.commit()
        return redirect(url_for('order_success'))

    return render_template('checkout.html')

```

**Description:** These are the image routes are used to fall back in the local host for the static images in the static folder .

## Deployment Code:

```

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

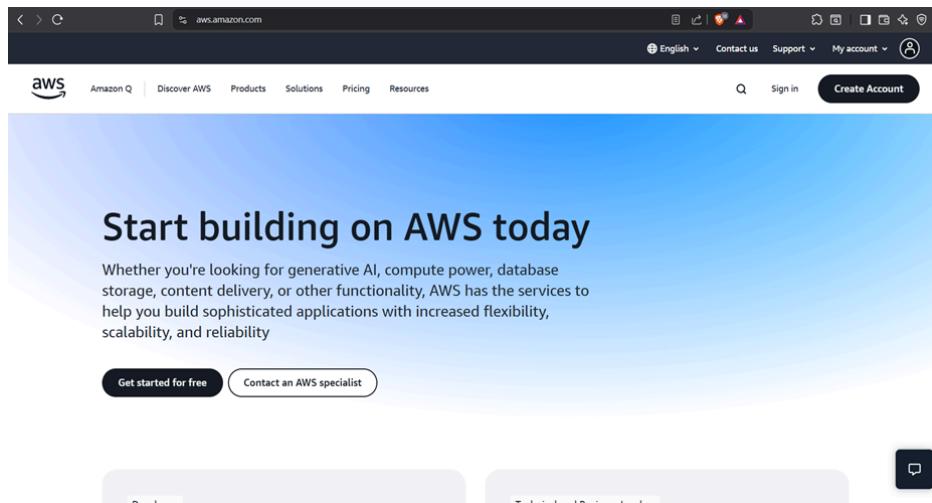
```

**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.

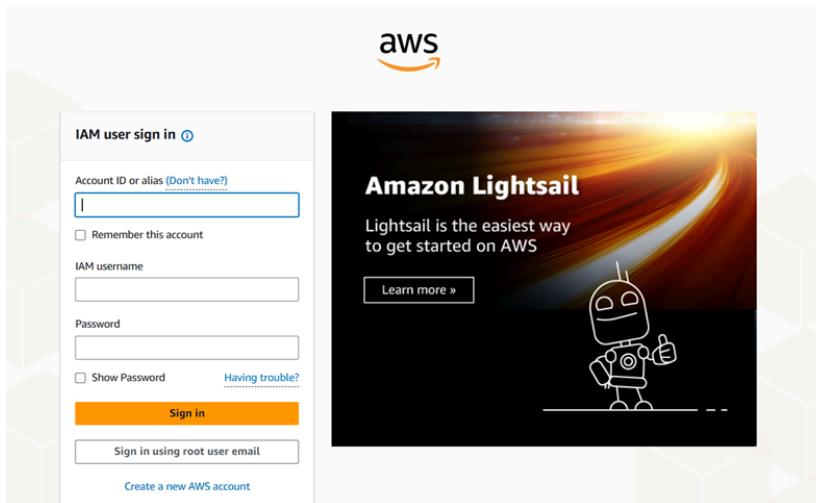
## Milestone 2: AWS Account Setup and Login

### a. Activity 2.1: Set up an AWS account if not already done.

- i. Sign up for an AWS account and configure billing settings.



### b. Activity 2.2: Log in to the AWS Management Console

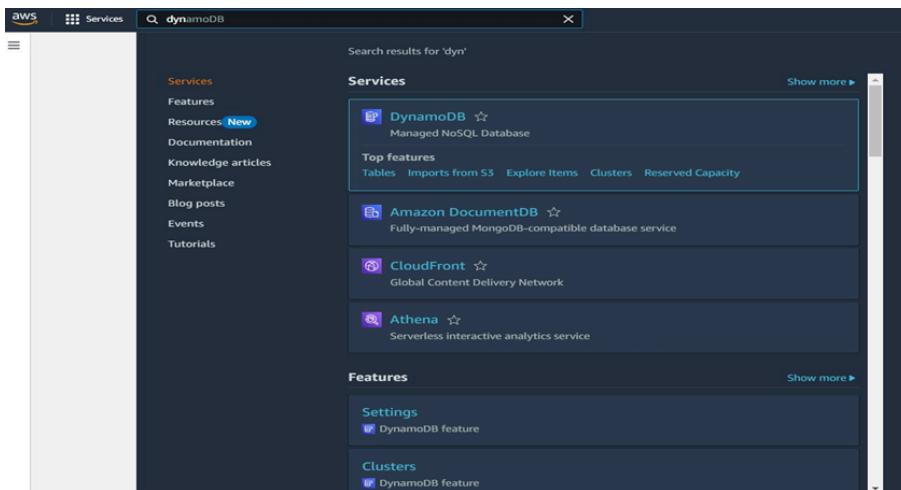
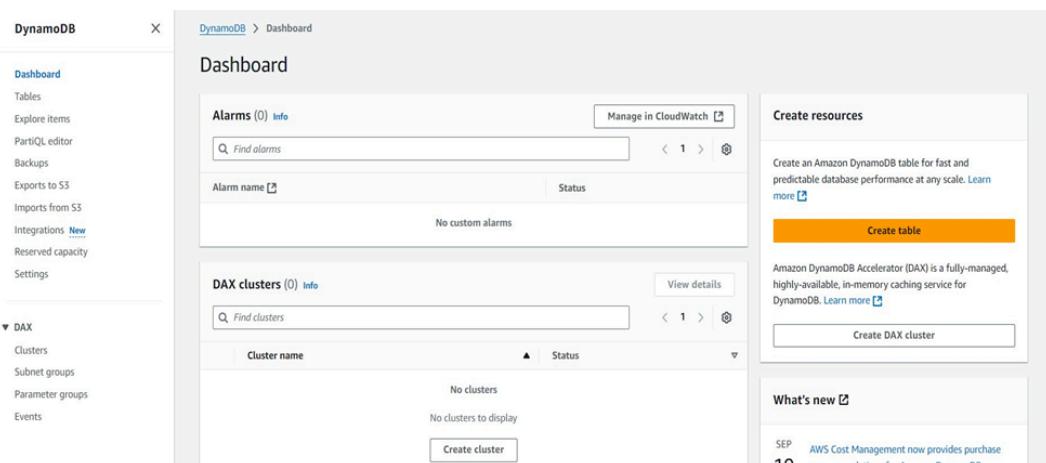


After setting up your account, log in to the [AWS Management Console](#).

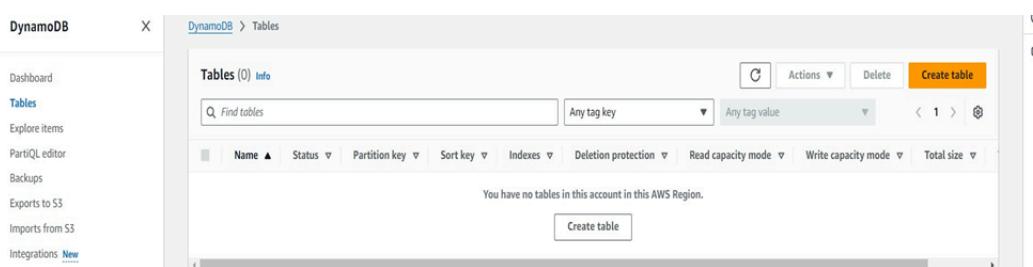
## Milestone 3: DynamoDB Database Creation and Setup

### C. Activity 3.1: Navigate to the DynamoDB

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

The screenshot shows the AWS DynamoDB Dashboard. On the left, there is a sidebar with links for Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, Settings, DAX (Clusters, Subnet groups, Parameter groups, Events), and a 'Tables' link under the main navigation. The main area is titled 'Dashboard' and contains sections for 'Alarms (0) Info' (with a 'Find alarms' search bar), 'DAX clusters (0) Info' (with a 'Find clusters' search bar and a 'Create cluster' button), and a 'Create resources' section. The 'Create resources' section includes a sub-section for creating an Amazon DynamoDB table with a 'Create table' button, information about Amazon DynamoDB Accelerator (DAX), and a 'Create DAX cluster' button. A 'What's new' section at the bottom right shows a recent update about AWS Cost Management providing purchase recommendations for Amazon DynamoDB.



The screenshot shows the AWS DynamoDB Tables page. The left sidebar has the same structure as the previous dashboard. The main area is titled 'Tables' and contains a 'Tables (0) Info' section with a 'Find tables' search bar and a 'Create table' button. Below this, a message states 'You have no tables in this account in this AWS Region.' There are also buttons for Actions, Delete, and Create table.

### d. Activity 3.2: Create a DynamoDB table for

## storing registration details and book requests.

- i. Create Users table with partition key “Email” with type String and click on create tables.

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.

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.

You can add 50 more tags.

DynamoDB &gt; Tables &gt; 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.

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.

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

### Tags

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

No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

[Cancel](#)
[Create table](#)

DynamoDB > 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.  
 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.

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**

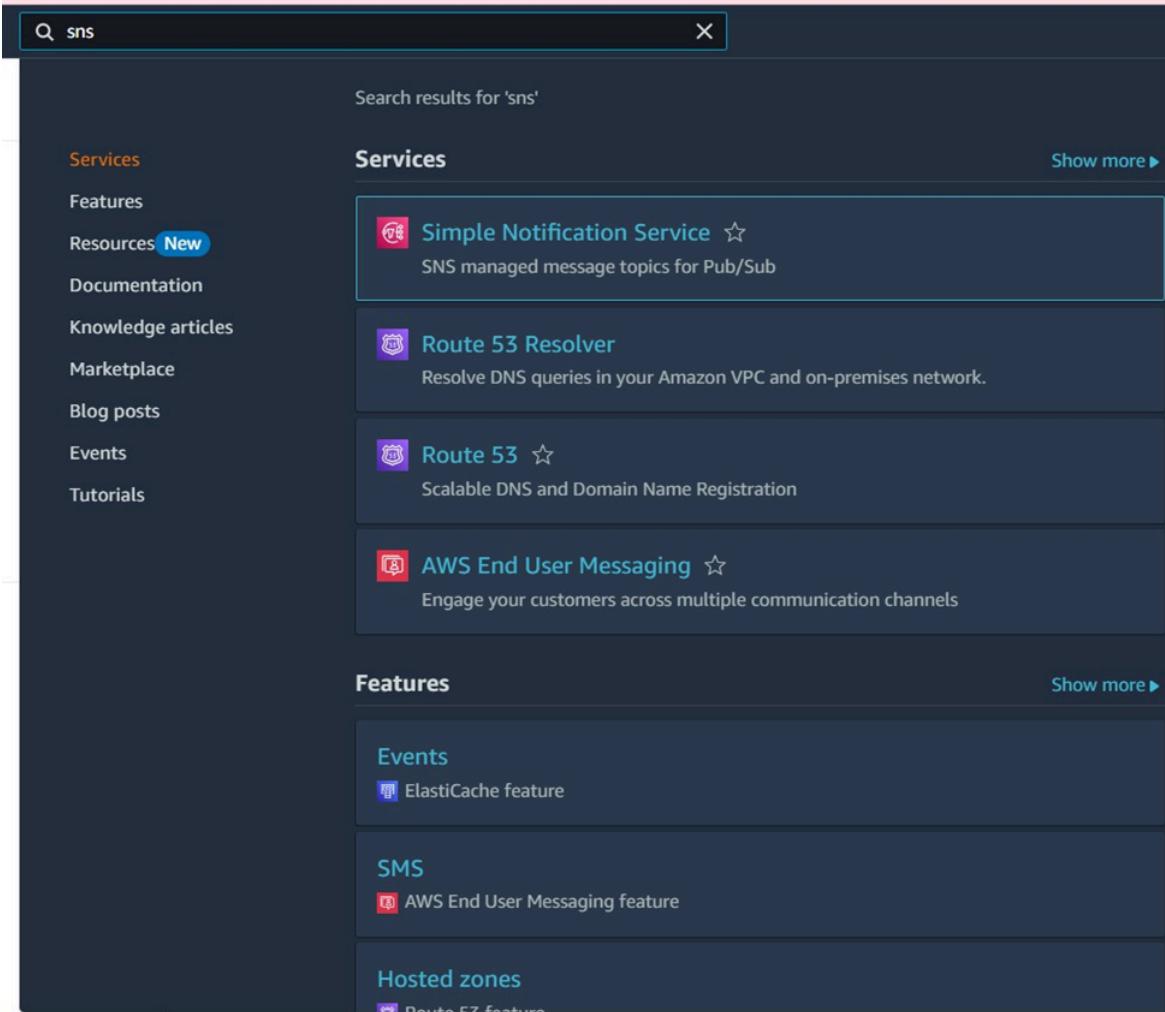
**Tables (2)** Info

<input type="checkbox"/>	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protecti
<input type="checkbox"/>	<a href="#">Orders</a>	<span>Active</span>	username (\$)	-	0	0	<input type="checkbox"/> Off
<input type="checkbox"/>	<a href="#">Users</a>	<span>Active</span>	username (\$)	-	0	0	<input type="checkbox"/> Off

## Milestone 4: SNS Notification Setup

### 1. Activity 4.1: Create SNS topics for sending email notifications for ordering alerts.

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

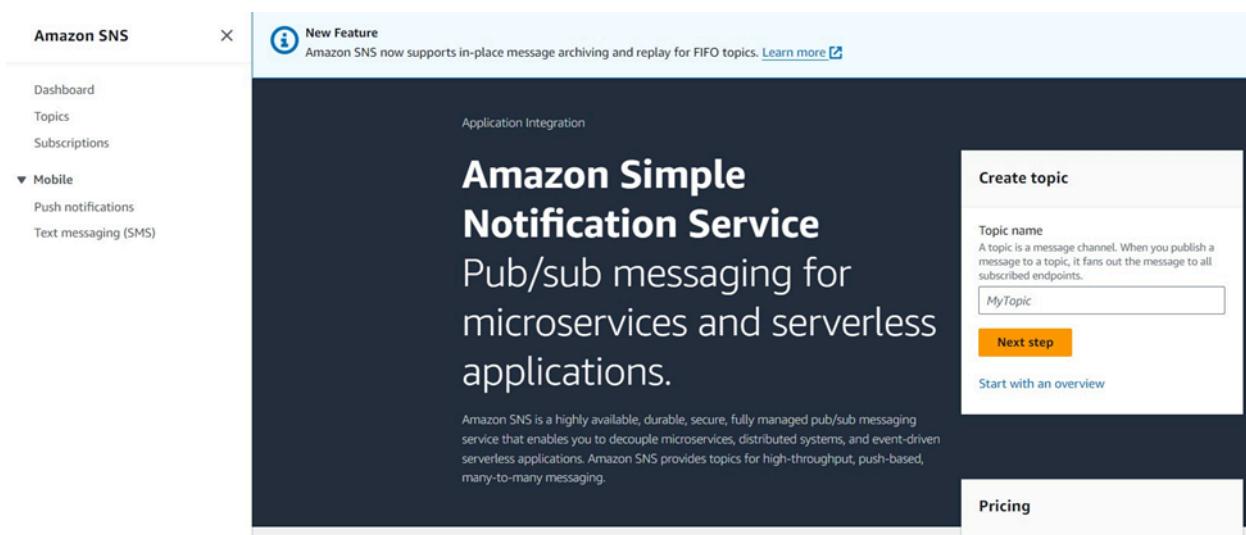


The screenshot shows the AWS Services search results for 'sns'. The search bar at the top contains 'sns'. Below it, there is a sidebar with links to 'Services', 'Features', 'Resources New', 'Documentation', 'Knowledge articles', 'Marketplace', 'Blog posts', 'Events', and 'Tutorials'. The main content area is titled 'Services' and shows the following results:

- Simple Notification Service** (star icon) - SNS managed message topics for Pub/Sub
- Route 53 Resolver** (shield icon) - Resolve DNS queries in your Amazon VPC and on-premises network.
- Route 53** (shield icon) - Scalable DNS and Domain Name Registration
- AWS End User Messaging** (phone icon) - Engage your customers across multiple communication channels

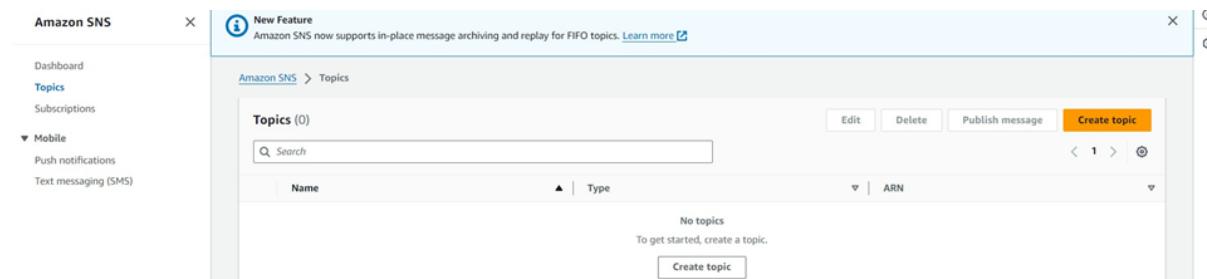
Below this, there is another section titled 'Features' with the following items:

- Events** (calendar icon) - ElastiCache feature
- SMS** (phone icon) - AWS End User Messaging feature
- Hosted zones** (shield icon) - Route 53 feature



The screenshot shows the Amazon SNS dashboard. On the left, a sidebar menu includes 'Dashboard', 'Topics', 'Subscriptions', and a 'Mobile' section with 'Push notifications' and 'Text messaging (SMS)'. The main content area features a 'New Feature' banner at the top stating 'Amazon SNS now supports in-place message archiving and replay for FIFO topics.' Below this is a large title 'Amazon Simple Notification Service' followed by a subtitle 'Pub/sub messaging for microservices and serverless applications.' A descriptive paragraph explains the service's purpose. To the right, a modal window titled 'Create topic' prompts the user to enter a 'Topic name' (with 'MyTopic' entered) and provides a 'Next step' button. A 'Pricing' link is visible at the bottom right.

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



The screenshot shows the 'Topics' page within the Amazon SNS service. The left sidebar remains the same. The main area displays a table header for 'Topics (0)' with columns for 'Name', 'Type', and 'ARN'. A search bar is at the top of the table. Below the table, a message says 'No topics' and 'To get started, create a topic.' A prominent orange 'Create topic' button is located at the bottom center of the page.

- c. Choose Standard type for general notification use cases and Click on Create Topic.

Amazon SNS > Topics > Create topic

## Create topic

### Details

#### Type [Info](#)

Topic type cannot be modified after topic is created.

FIFO (first-in, first-out)

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

Standard

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

#### Name

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](#)

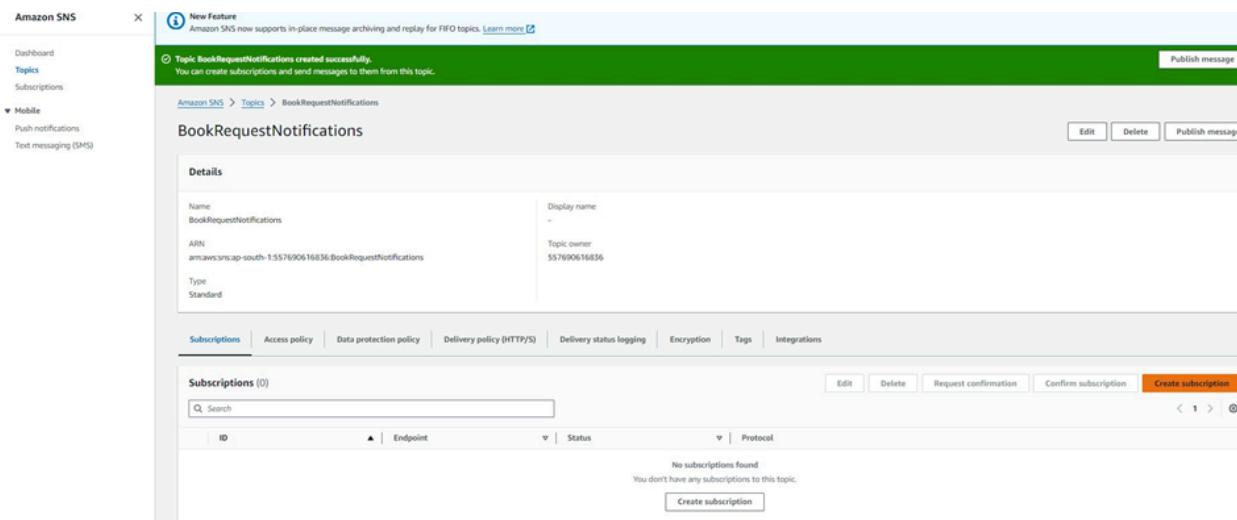
► **Active tracing - optional** [Info](#)

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**

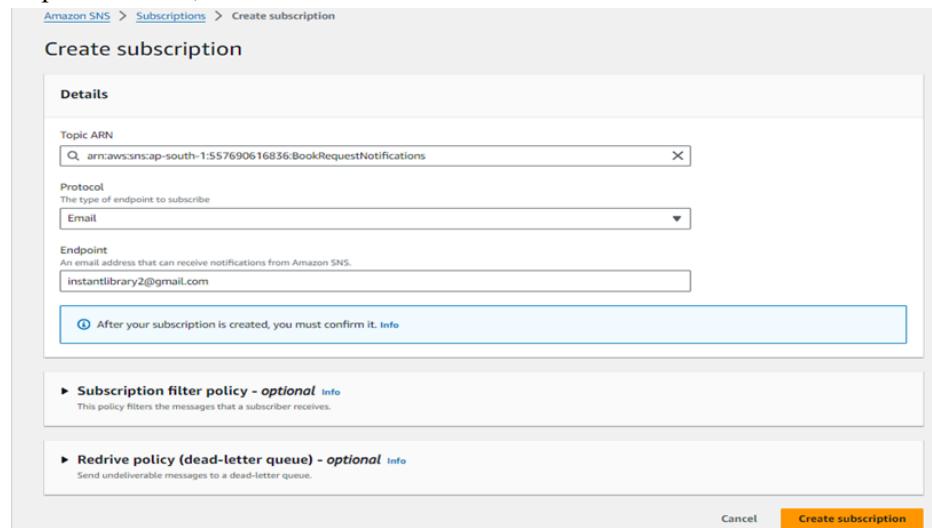
- d. Configure the SNS topic and note down the **Topic ARN**.



The screenshot shows the 'Amazon SNS' interface. On the left, a sidebar lists 'Topics' under 'Amazon SNS'. The main area displays a success message: 'Topic BookRequestNotifications created successfully. You can create subscriptions and send messages to them from this topic.' Below this, the 'BookRequestNotifications' topic details are shown, including its ARN (arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications) and type (Standard). A 'Subscriptions' tab is selected, showing a table with one row: 'instantlibrary2@gmail.com' (Email protocol). A 'Create subscription' button is visible at the bottom.

## 2. Activity 4.2: 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.



The screenshot shows the 'Create subscription' dialog. It has a 'Details' section where 'Topic ARN' is set to 'arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications' and 'Protocol' is set to 'Email'. The 'Endpoint' field contains 'instantlibrary2@gmail.com'. Below these fields, there are two optional sections: 'Subscription filter policy - optional' (with a note about filtering messages) and 'Redrive policy (dead-letter queue) - optional' (with a note about sending undeliverable messages to a dead-letter queue). At the bottom right, there are 'Cancel' and 'Create subscription' buttons, with the latter being orange.

Amazon SNS

**New Feature**  
Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#)

**Subscription to BookRequestNotifications created successfully.**  
The ARN of the subscription is arn:aws:sns:ap-south-1:557690616856:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4.

Amazon SNS > Topics > BookRequestNotifications > Subscription: d78e0371-9235-404d-952c-85c2743607c4

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

**Details**

ARN	arn:aws:sns:ap-south-1:557690616856:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4	Status	Pending confirmation
Endpoint	instantlibrary2@gmail.com	Protocol	EMAIL
Topic	BookRequestNotifications		
Subscription Principal	arn:aws:iam::557690616856:root		

[Edit](#) [Delete](#)

[Subscription filter policy](#) [Redrive policy \(dead-letter queue\)](#)

**Subscription filter policy** Info  
This policy filters the messages that a subscriber receives.

No filter policy configured for this subscription.  
To apply a filter policy, edit this subscription.

[Edit](#)

**b. After subscription requestfor the mail confirmation**

Amazon SNS

**New Feature**  
Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#)

**Confirmation request was sent successfully.**  
The ARN of the subscription is arn:aws:sns:ap-south-1:557690616856:BookRequestNotifications:7aeef5f16-12ad-4731-9f7d-c342c2113a55.

Amazon SNS > Topics > BookRequestNotifications

**BookRequestNotifications**

**Details**

Name	BookRequestNotifications	Display name	-
ARN	arn:aws:sns:ap-south-1:557690616856:BookRequestNotifications	Topic owner	557690616856
Type	Standard		

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

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

**Subscriptions (2)**

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

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

**c. Navigate to the subscribed Emailaccount and Clickon 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!

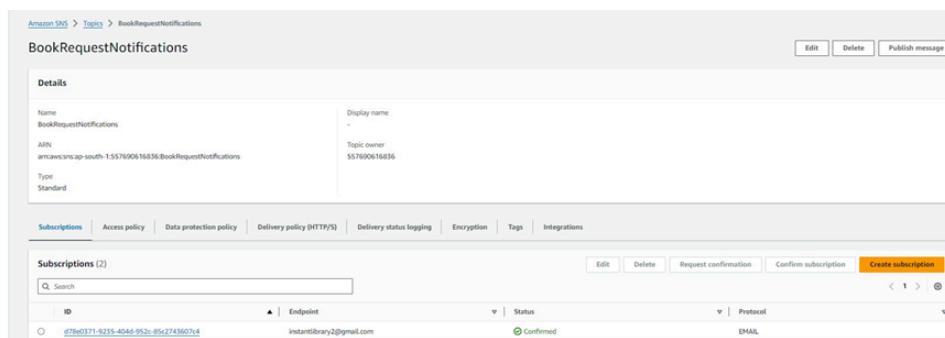
You have successfully subscribed.

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](#).

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



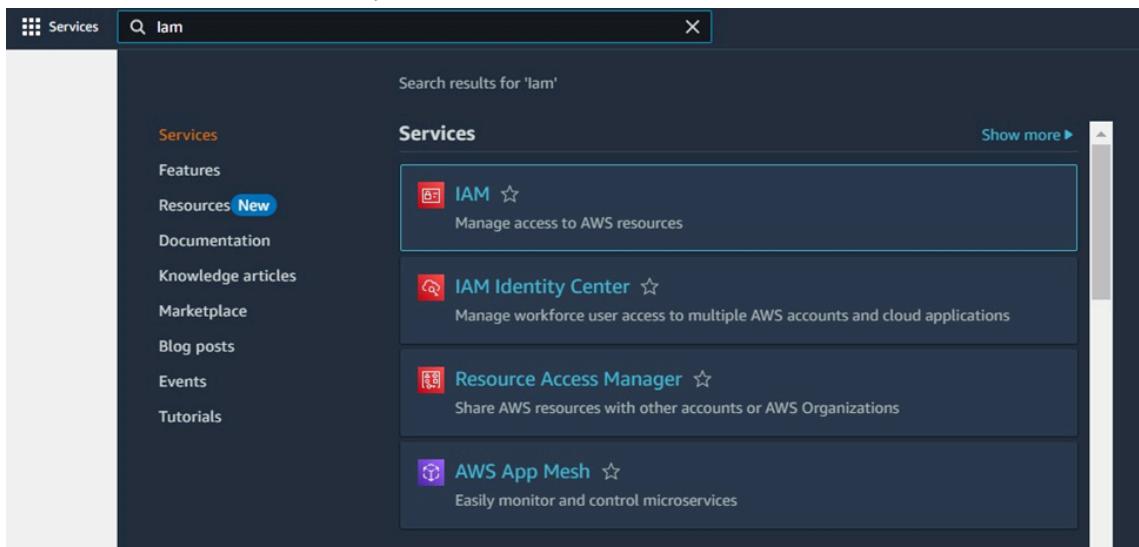
The screenshot shows the AWS SNS console with the 'BookRequestNotifications' topic selected. The 'Details' section displays the topic's name, ARN, and type. Below it, the 'Subscriptions' section lists one confirmed subscription to an email address.

ID	Endpoint	Status	Protocol
d7be0171-9215-404d-952c-85c274360714	instantlibrary2@gmail.com	Confirmed	EMAIL

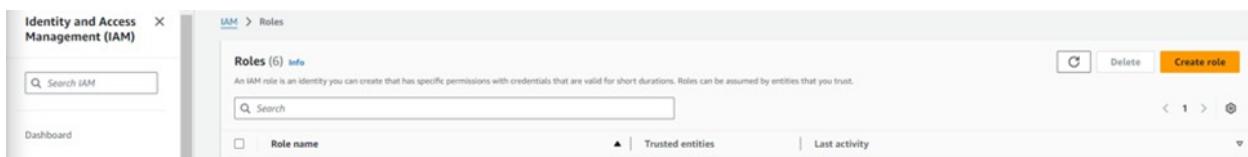
## Milestone 5: IAM Role Setup

### 1. 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.

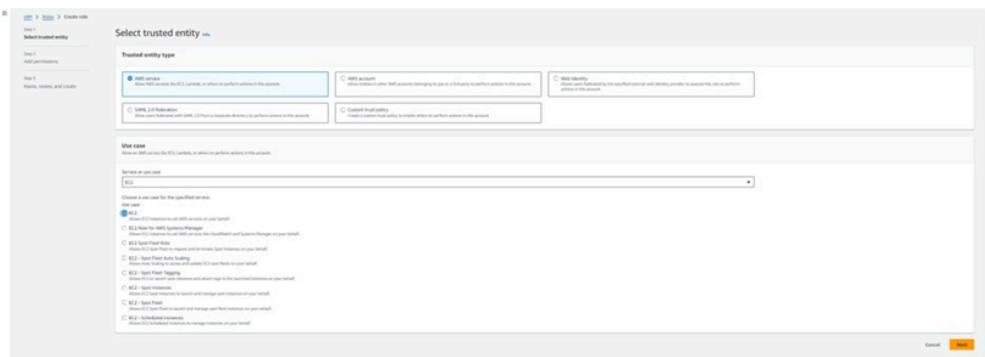
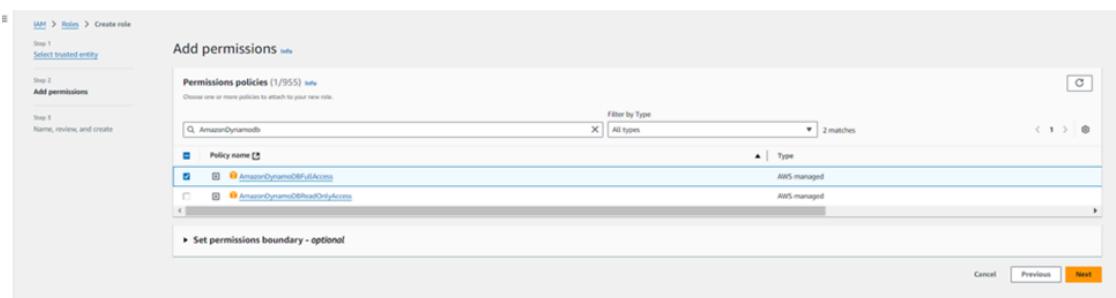


The screenshot shows the AWS search results for 'iam'. The 'Services' section on the left lists various AWS services. The 'IAM' service is highlighted and listed first in the main search results, with a description of 'Manage access to AWS resources'.



The screenshot shows the 'Roles' page in the AWS IAM console. It displays a list of six existing IAM roles. The top role, 'AdministratorAccess', is highlighted. The page includes a search bar and buttons for creating a new role or deleting an existing one.

Role name	Trusted entities	Last activity
AdministratorAccess		
AmazonEC2ContainerServiceForLambdaRole		
AmazonS3FullAccess		
AmazonSNSFullAccess		
AmazonSQSFullAccess		
AmazonVPCFullAccess		

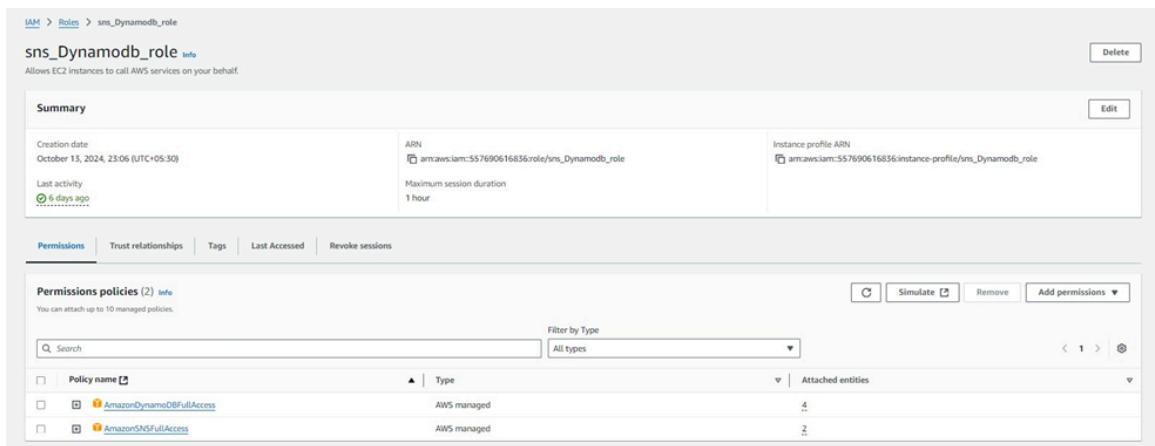



## 2. Activity 5.2: Attach Policies.

- Attach the following policies to the role:
1. AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.
  2. AmazonSNSFullAccess: Grants EC2 the ability to send notifications via SNS.



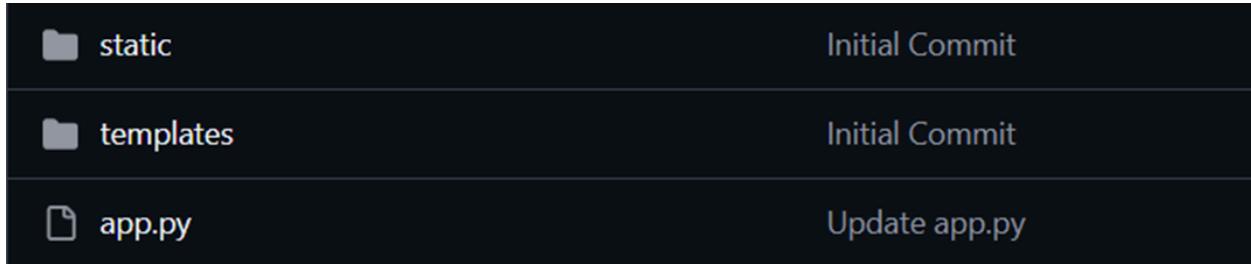


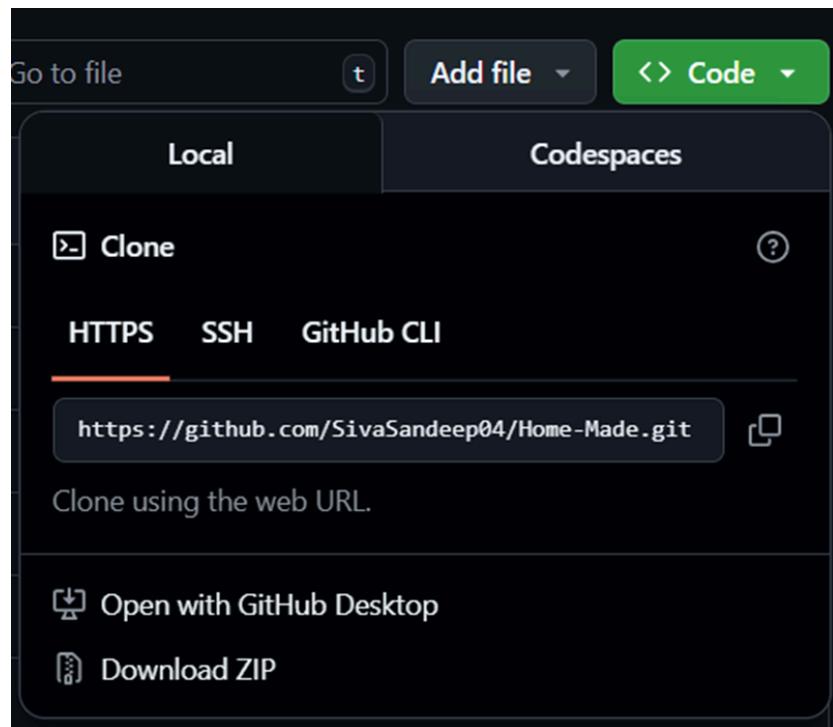


The screenshot shows the AWS IAM Roles page with the role 'sns\_Dynamodb\_role' selected. The 'Summary' tab is active, displaying details like ARN, Creation date (October 13, 2024), Last activity (6 days ago), and Maximum session duration (1 hour). The 'Permissions' tab is also visible, showing attached policies: 'AmazonDynamoDBFullAccess' and 'AmazonSNSFullAccess'. Both are AWS managed policies.

## Milestone 6: EC2 InstanceSetup

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

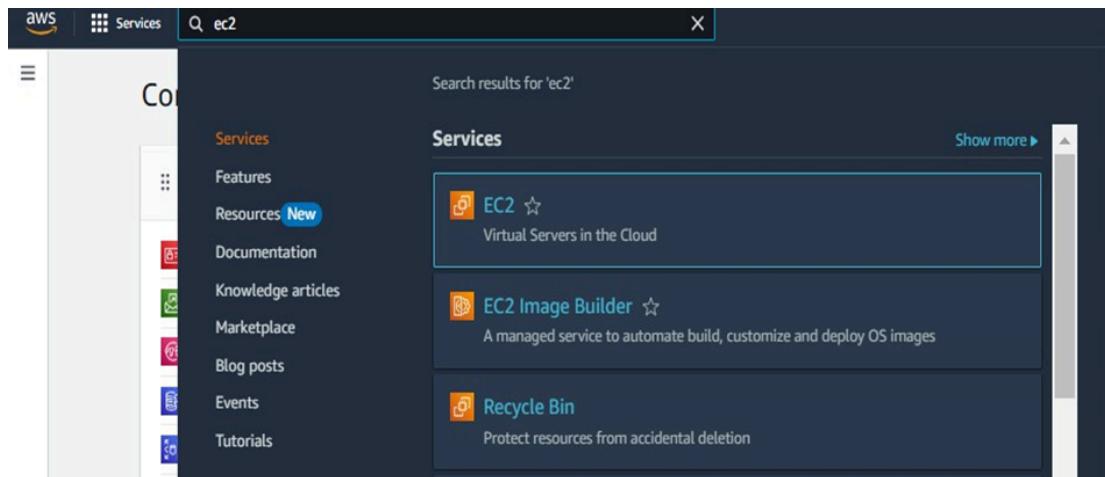




## 2. Activity 6.1: Launch an EC2 instance to host the Flask application.

### a. Launch EC2 Instance

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



- Click on Launch instance to launch EC2 instance



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.

Do not show me this message again [Take a walkthrough](#)

**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

**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**

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian [Browse more AMIs](#)

**Summary**

Number of instances [Info](#)

1

Software Image (AMI)  
Amazon Linux 2023 AMI 2023.7.2... [read more](#)  
ami-05ffe3c48a9991133

Virtual server type (instance type)  
t2.micro

Firewall (security group)  
New security group

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

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

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)

All states

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

No instances

You do not have any instances in this region

[Launch instances](#)

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.

Do not show me this message again [Take a walkthrough](#)

**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

**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

**Summary**

Number of instances [Info](#)

1

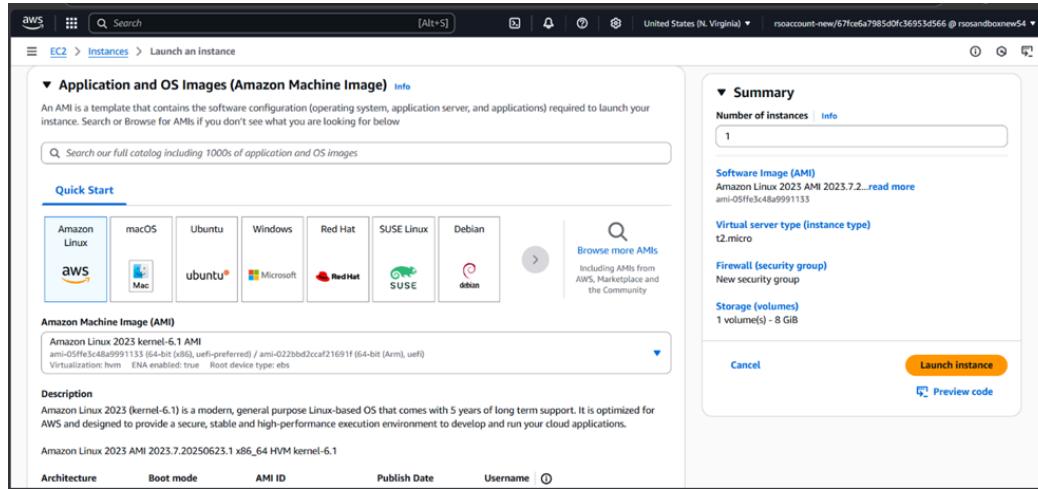
Software Image (AMI)  
Amazon Linux 2023 AMI 2023.7.2... [read more](#)  
ami-05ffe3c48a9991133

Virtual server type (instance type)  
t2.micro

Firewall (security group)  
New security group

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

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



**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**

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian

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

**Amazon Machine Image (AMI)**

Amazon Linux 2023 kernel-6.1 AMI  
ami-05ffec4c8a95991133 (64-bit (x86), uefi-preferred) / ami-022bbd2ccaf21691f (64-bit (Arm), uefi)  
Virtualization: hvm ENA enabled: true Root device type: ebs

**Description**

Amazon Linux 2023 (kernel-6.1) 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.

Amazon Linux 2023 AMI 2023.7.20250623.1 x86\_64 HVM kernel-6.1

Architecture Boot mode AMI ID Publish Date Username

**Summary**

Number of instances [Info](#)  
1

Software Image (AMI)  
Amazon Linux 2023 AMI 2023.7.2... [read more](#)  
ami-05ffec4c8a95991133

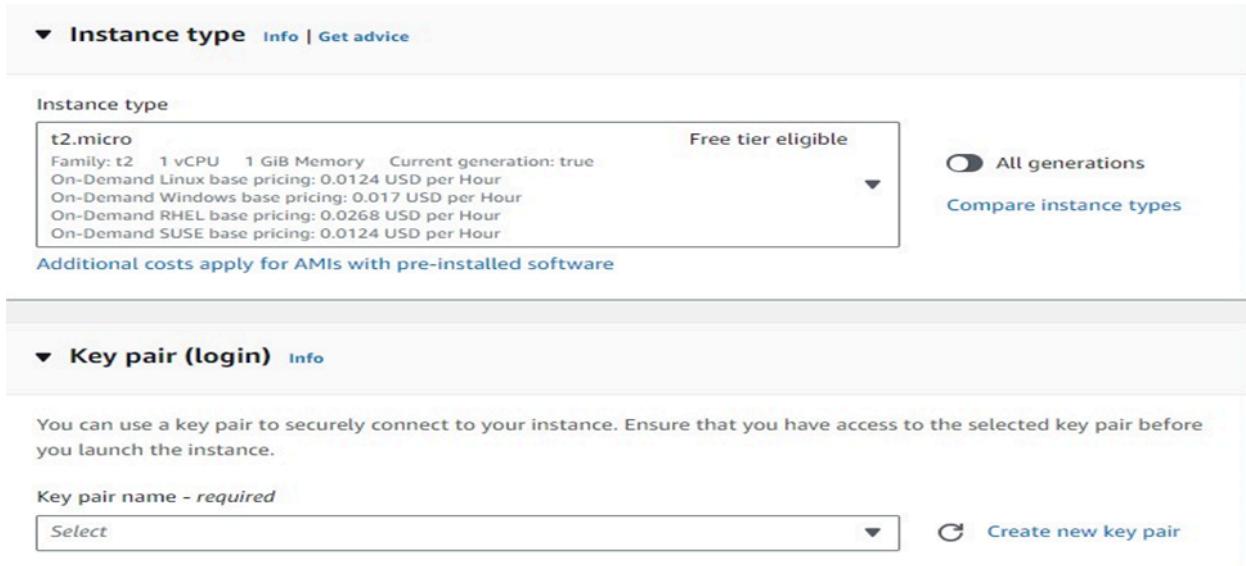
Virtual server type (instance type)  
t2.micro

Firewall (security group)  
New security group

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

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

#### 4. 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

Additional costs apply for AMIs with pre-installed software

**Free tier eligible**

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

Select [Create new key pair](#)

### Create key pair

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

homeMadeKey

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**

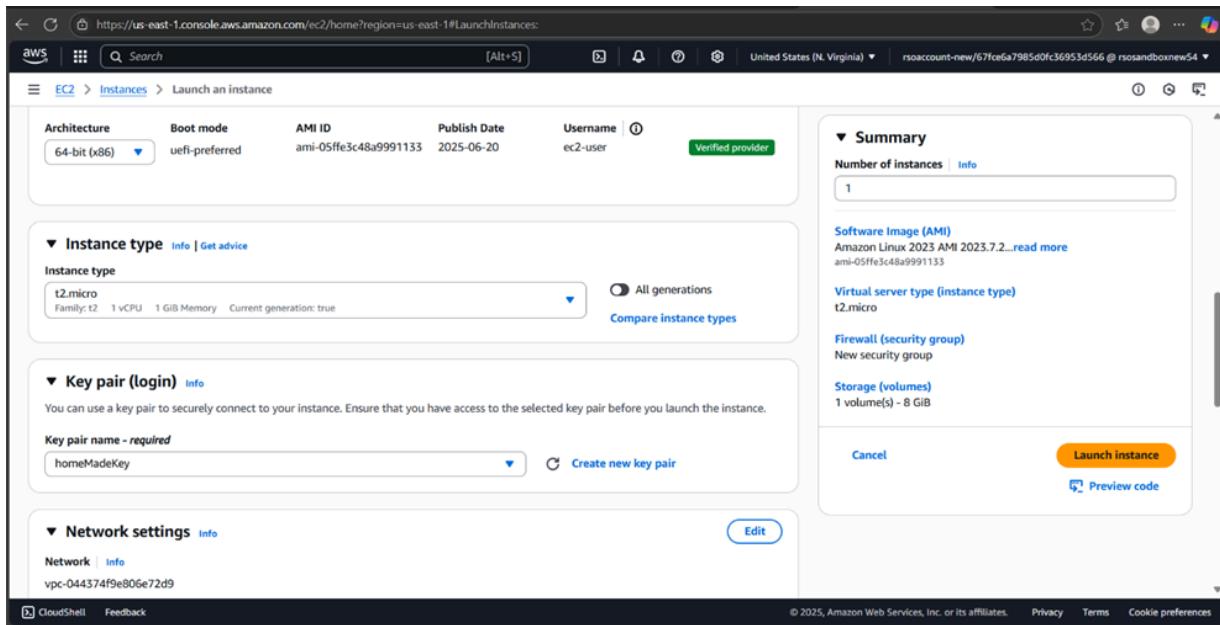
.pem  
For use with OpenSSH

.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](#)



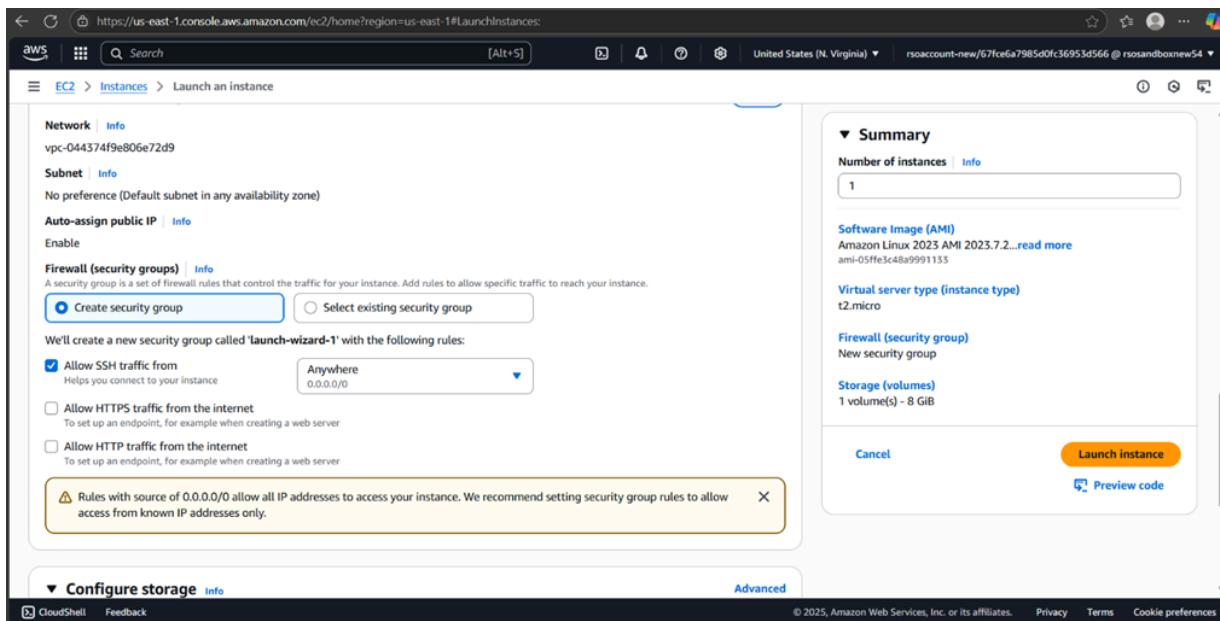


This screenshot shows the 'Launch an instance' wizard in the AWS Management Console. The user has selected the following configurations:

- Architecture:** 64-bit (x86)
- Boot mode:** uefi-preferred
- AMI ID:** ami-05ffe3c48a9991133
- Publish Date:** 2025-06-20
- Username:** ec2-user
- Software Image (AMI):** Amazon Linux 2023 AMI 2023.7.2... (ami-05ffe3c48a9991133)
- Virtual server type (instance type):** t2.micro
- Firewall (security group):** New security group
- Storage (volumes):** 1 volume(s) - 8 GiB
- Key pair (login):** homeMadeKey
- Network settings:** Network (Info) - vpc-044374f9e806e72d9

The 'Summary' section indicates 1 instance will be launched. Buttons for 'Launch instance' and 'Preview code' are visible.

## 5. Activity 6.2:Configure securitygroups for HTTP and SSH access.

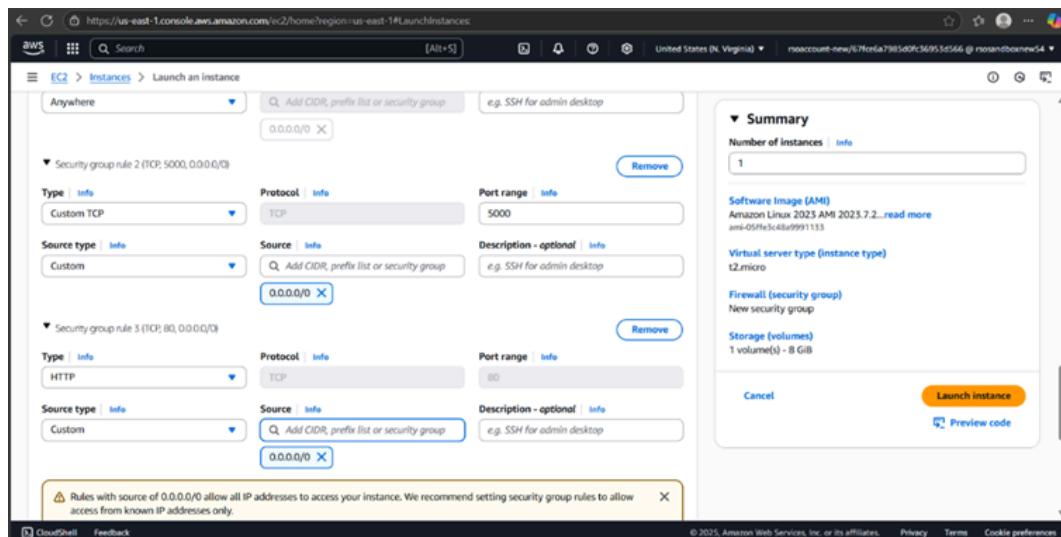
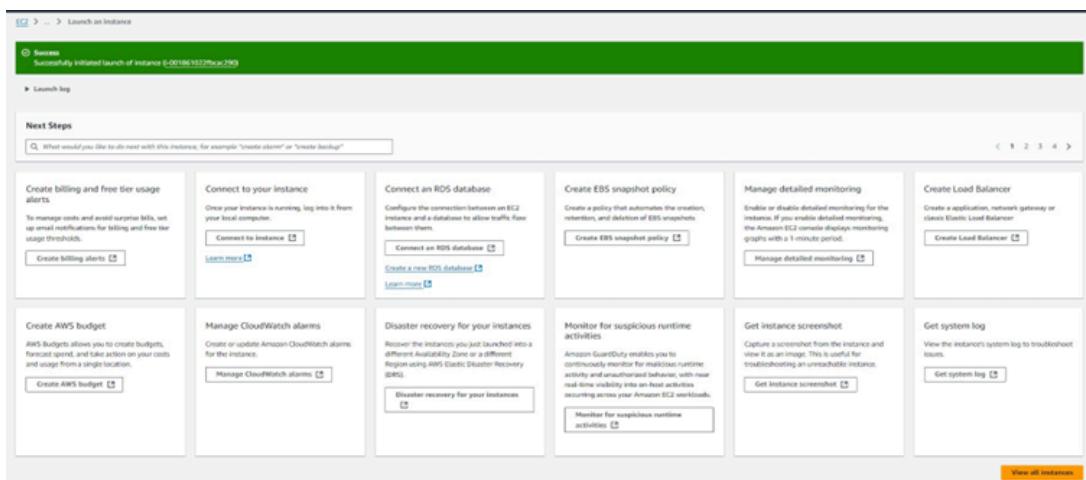


This screenshot shows the continuation of the 'Launch an instance' wizard, specifically the 'Network' configuration step. The user has created a new security group named 'launch-wizard-1' with the following rules:

- Allow SSH traffic from anywhere** (checked): Helps you connect to your instance.
- Allow HTTPS traffic from the internet** (unchecked): To set up an endpoint, for example when creating a web server.
- Allow HTTP traffic from the internet** (unchecked): To set up an endpoint, for example when creating a web server.

A note at the bottom states: "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."

The 'Summary' section indicates 1 instance will be launched. Buttons for 'Launch instance' and 'Preview code' are visible.

6. 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.

Successfully attached EC2\_DynamoDB\_Role to instance i-0a759096dd59bc99d

**Instances (1/2) Info**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input checked="" type="checkbox"/> HomeMadePickles	i-0a759096dd59bc99d	<span>Running</span>	t2.micro	<span>2/2 checks passed</span>	<a href="#">View alarms +</a>	us-east-1d
<input type="checkbox"/> HomeDeleted	i-0d8e6b4b7c4708f03	<span>Terminated</span>	t2.micro	-	<a href="#">View alarms +</a>	us-east-1d

**i-0a759096dd59bc99d (HomeMadePickles)**

**Instance summary for i-001861022fbcac290 (InstantLibraryApp) Info**

Updated less than a minute ago

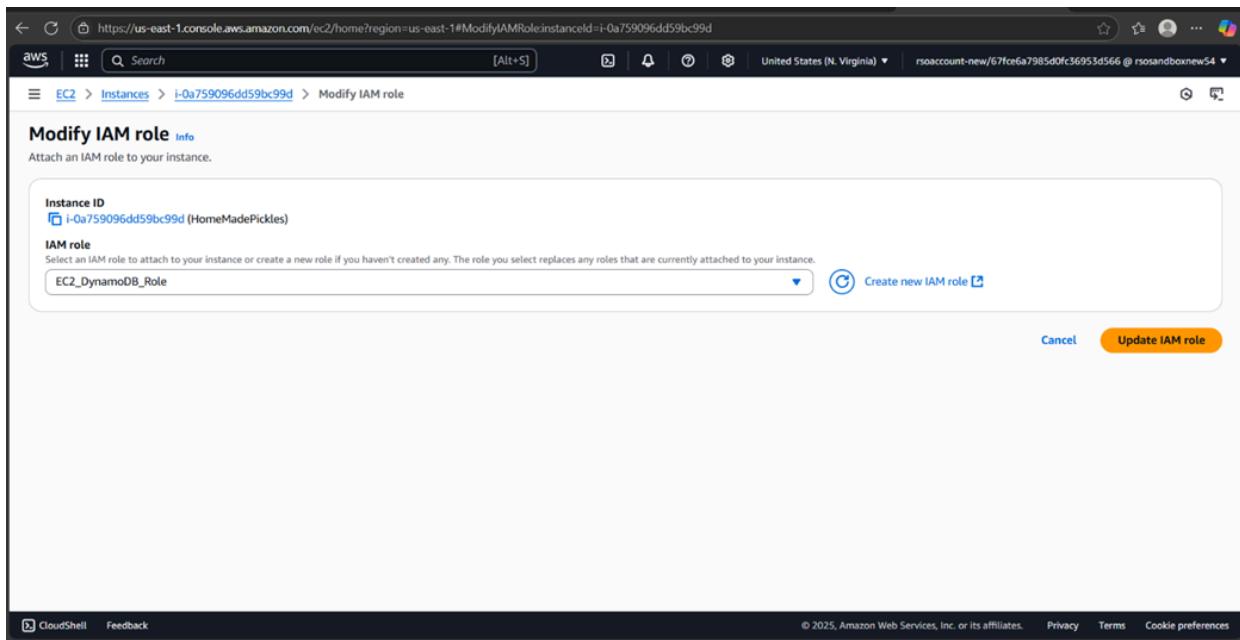
Instance ID <a href="#">i-001861022fbcac290</a>	Public IPv4 address -	Private IPv4 addresses <a href="#">172.31.3.5</a>
IPv6 address -	Instance state <span>Stopped</span>	Public IPv4 DNS -
Hostname type IP name: ip-172-31-5-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) <a href="#">ip-172-31-5-5.ap-south-1.compute.internal</a>	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding <a href="#">Opt-in to AWS Compute Optimizer for recommendations   Learn more</a>
Auto-assigned IP address -	VPC ID <a href="#">vpc-05cdc7b6f19dd7211</a>	Auto Scaling Group name -
IAM Role <a href="#">sns_Dynamodb_role</a>	Subnet ID <a href="#">subnet-0d9fa3144480cc9a9</a>	
IMDsv2 Required	Instance ARN <a href="#">arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290</a>	

**Details** | Status and alarms | Monitoring | Security | Networking | Storage | Tags

**Instance summary for i-001861022fbcac290 (InstantLibraryApp) Info**

Updated less than a minute ago

Instance ID <a href="#">i-001861022fbcac290</a>	Public IPv4 address -	Private IPv4 addresses <a href="#">172.31.3.5</a>	Manage instance state <a href="#">Stop</a>
IPv6 address -	Instance state <span>Stopped</span>	Public IPv4 DNS -	Instance settings <a href="#">Edit</a>
Hostname type IP name: ip-172-31-5-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) <a href="#">ip-172-31-5-5.ap-south-1.compute.internal</a>	Elastic IP addresses -	Networking <a href="#">Edit</a>
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding <a href="#">Opt-in to AWS Compute Optimizer for recommendations   Learn more</a>	Security <a href="#">Edit</a>
Auto-assigned IP address -	VPC ID <a href="#">vpc-05cdc7b6f19dd7211</a>	Change security groups <a href="#">Edit</a>	
IAM Role <a href="#">sns_Dynamodb_role</a>	Subnet ID <a href="#">subnet-0d9fa3144480cc9a9</a>	Get Windows password <a href="#">Edit</a>	
IMDsv2 Required	Instance ARN <a href="#">arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290</a>	Modify IAM role <a href="#">Edit</a>	
		Image and templates <a href="#">Edit</a>	Monitor and troubleshoot <a href="#">Edit</a>



Modify IAM role [Info](#)

Attach an IAM role to your instance.

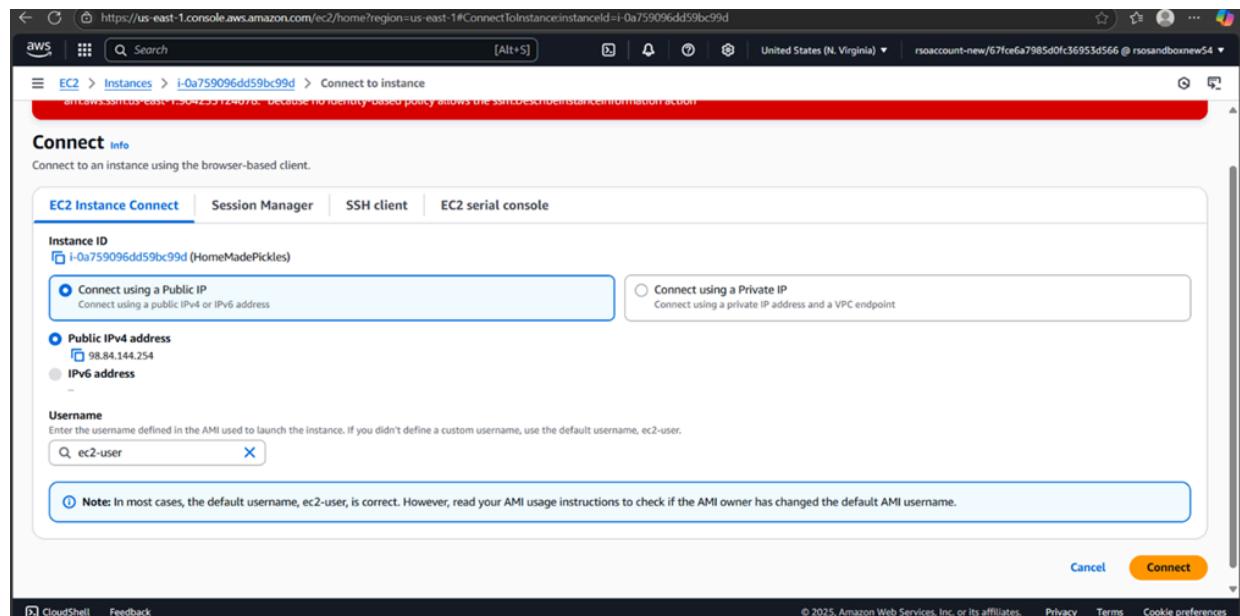
Instance ID  
 i-0a759096dd59bc99d (HomeMadePickles)

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\_Role [Create new IAM role](#)

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

## 7. Now connect the EC2 with the files.



EC2 > Instances > i-0a759096dd59bc99d > Connect to instance

arn:aws:ec2:us-east-1:759096dd59bc99d:instance/i-0a759096dd59bc99d: because no identity-based policy allows the ssm:DescribeInstanceInformation action

**Connect** [Info](#)

Connect to an instance using the browser-based client.

EC2 Instance Connect Session Manager SSH client EC2 serial console

Instance ID  
 i-0a759096dd59bc99d (HomeMadePickles)

Connect using a Public IP  
 Connect using a public IPv4 or IPv6 address

Connect using a Private IP  
 Connect using a private IP address and a VPC endpoint

Public IPv4 address  
 98.84.144.254

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](#)

**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](#)

```
[ec2-user@ip-172-31-23-180 Home-Made]$ 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://127.0.0.1:5000
 * Running on http://172.31.23.180:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 108-811-646
115.244.132.21 - - [04/Jul/2025 09:39:11] "GET / HTTP/1.1" 200 -
115.244.132.21 - - [04/Jul/2025 09:39:13] "GET /favicon.ico HTTP/1.1" 404 -

```

i-0f76b672ed1a95795 (HomeMade1)  
 PublicIPs: 98.84.144.254 PrivateIPs: 172.31.23.180

## 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:

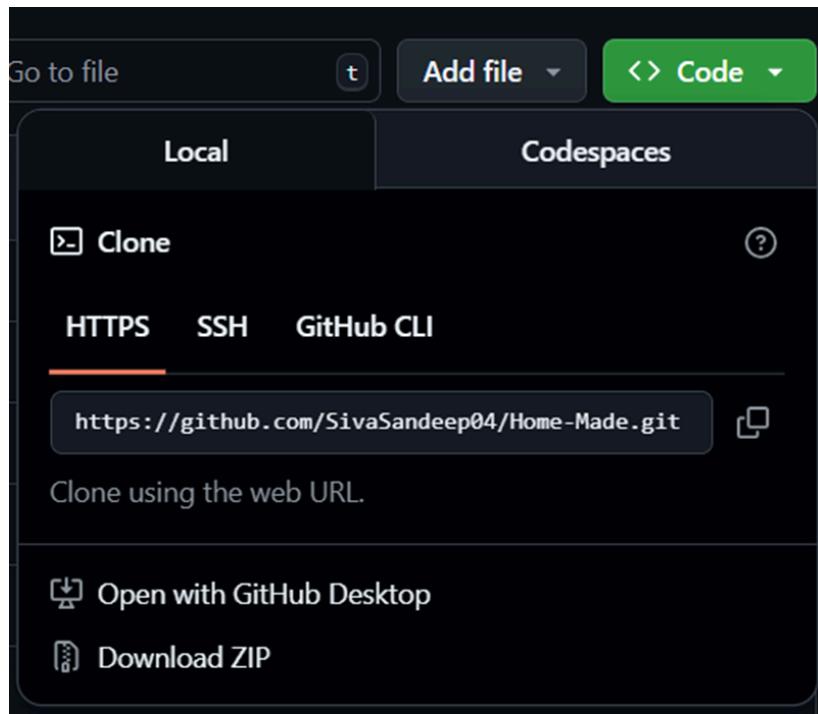
```
flask --
```

```
version git
```

```
-version
```

## Activity 7.2:Clone Your Flask Projectfrom GitHub

Clone your projectrepository from GitHubinto the EC2 instance using Git.



Run: 'git clone <https://github.com/SivaSandeep04/Home-Made.git>'

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

Here : 'git clone <https://github.com/SivaSandeep04/Home-Made.git>'

**To navigate to the projectdirectory, run the following command:**

```
cd home_made
```

**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=5000
```

aws [Alt+T] Search United States (N. Virginia) rsosaccount-new/67fc6a7985d0fc36953d566 @ rsosandboxnew109 rsosaccount-new/67fc6a7985d0fc36953d566 @rsosandboxnew109

```

collecting s3transfer<0.14.0,>=0.13.0
  Downloading s3transfer-0.13.0-py3-none-any.whl (85 kB)
[...]
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.3)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.3)
Requirement already satisfied: six<1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1>botocore<1.40.0,>=1.39.3)
Installing collected packages: botocore, s3transfer, boto3
Successfully installed botocore-1.39.3 s3transfer-0.13.0
botocore-1.39.3 is the latest version
botocore-1.39.3 s3transfer-0.13.0
In�ntalling to user installation because normal site-packages is not writable
Collecting python-dotenv
  Downloading python_dotenv-1.1.1-py3-none-any.whl (20 kB)
Installing collected packages: python-dotenv
Successfully installed python-dotenv-1.1.1
[ec2-user@ip-172-31-23-180 ~]$ git clone https://github.com/SivaSandeep04/Home-Made.git
Cloning into 'Home-Made'...
remote: Enumerating objects: 2031, done.
remote: Counting objects: 100% (2031/2031), done.
remote: Compressing objects: 100% (1829/1829), done.
remote: Total 2031 (delta 180), reused 2023 (delta 180), pack-reused 0 (from 0)
Receiving objects: 100% (2031/2031), 17.98 MiB | 32.12 MiB/s, done.
Resolving deltas: 100% (184/184), done.
[ec2-user@ip-172-31-23-180 Home-Made]$ 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://127.0.0.1:5000
 * Running on http://172.31.23.180:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 108-811-646
15.244.132.21 - - [04/Jul/2025 09:39:11] "GET / HTTP/1.1" 200 -
15.244.132.21 - - [04/Jul/2025 09:39:13] "GET /favicon.ico HTTP/1.1" 404 -
[...]
i-0f76b672ed1a95795 (HomeMade)
PublicIPs: 98.84.144.254 PrivateIPs: 172.31.23.180

```

Verify the Flask app is running: <http://your-ec2-public-ip>

## 1. Run the Flask app on the EC2 instance

```

223.228.99.42 - - [03/Jul/2025 07:43:22] "GET / HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET /static/css/general.css HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET /static/css/nav.css HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET /static/css/footer.css HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET /static/css/produce_card.css HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET /static/css/cart_checkout.css HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET /static/js/utils.js HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:23] "GET /static/img/logo.png HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:23] "GET /static/img/hero-banner.jpg HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:23] "GET /static/js/product.js HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:23] "GET /static/js/cart.js HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:23] "GET /static/js/main.js HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:23] "GET /static/js/profile.js HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:24] "GET /favicon.ico HTTP/1.1" 404 -
DynamoDB error: An error occurred (ResourceNotFoundException) when calling the Scan operation: Requested resource not found
223.228.99.42 - - [03/Jul/2025 07:43:26] "GET /api/products HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:28] "GET /static/img/chicken-pickle.jpg HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:28] "GET /static/img/fish-pickle.jpg HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:49] "GET / HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:49] "GET /static/css/general.css HTTP/1.1" 304 -

```

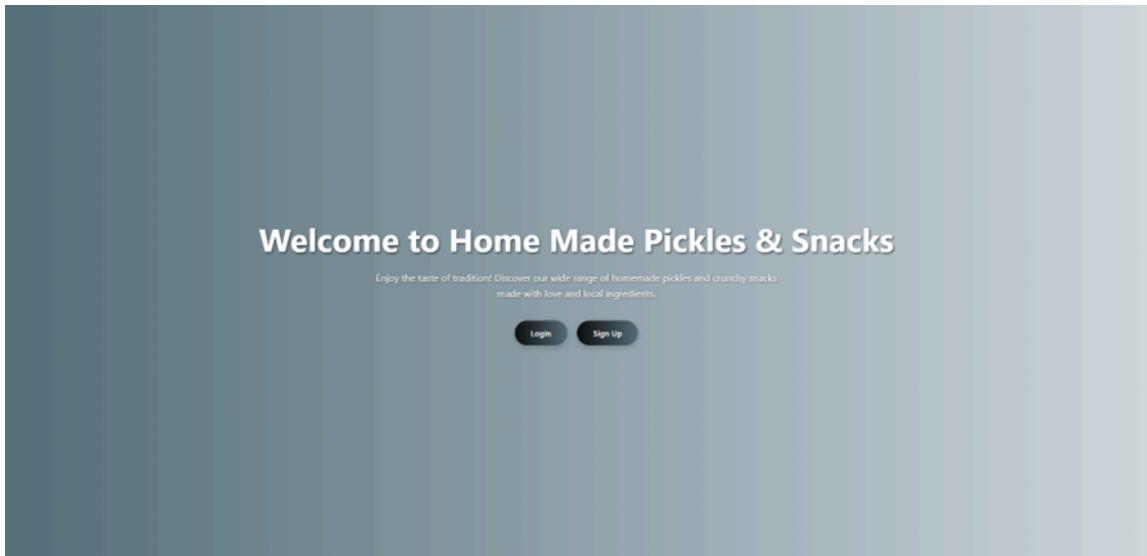
Access the website through:

PublicIPs:<http://98.84.144.254:5000/>

## Milestone 8: Testing and Deployment

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

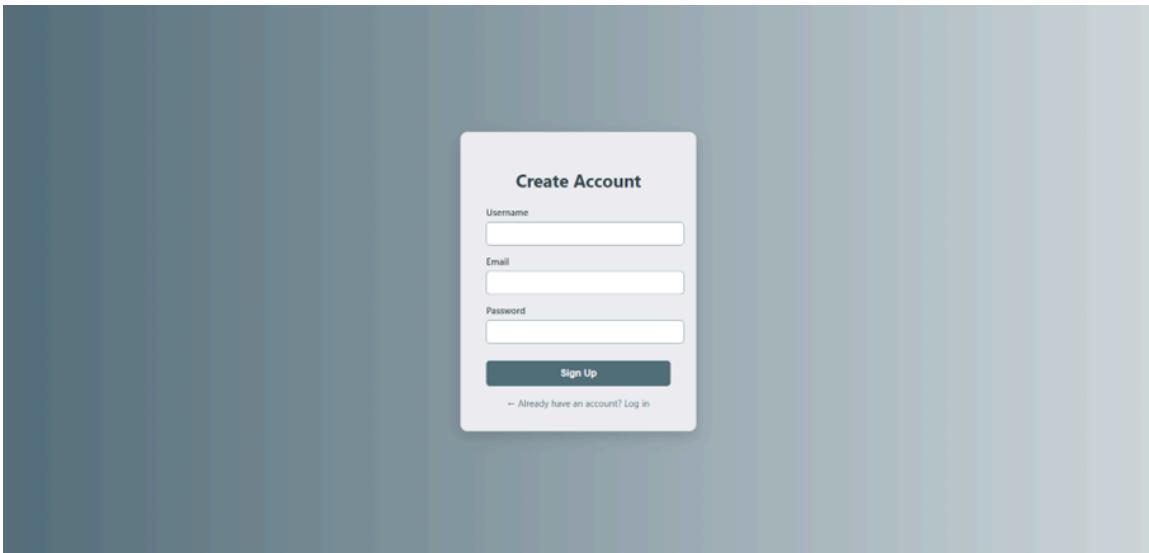
**Welcome Page:**



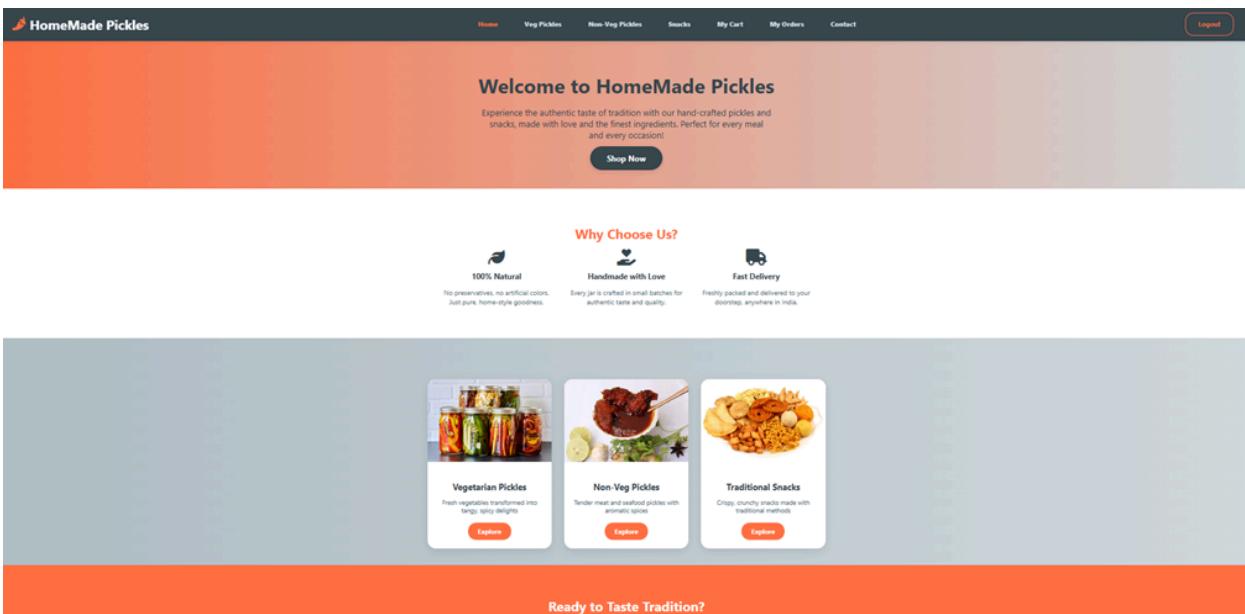
**Login Page:**



### Register Page:



### Home page:



### Veg Pickles Page :

**HomeMade Pickles**

Home    **Veg Pickles**    Non-Veg Pickles    Snacks    My Cart    My Orders    Contact    [Logout](#)



**Mango Pickle**  
Classic Andhra-style raw mango pickle.  
★★★★★  
₹149

[Add To Cart](#)



**Lemon Pickle**  
Zesty and tangy with a hint of spice.  
★★★★★  
₹129

[Add To Cart](#)



**Ginger Pickle**  
Spicy and tangy ginger pickle.  
★★★★★  
₹149

[Add To Cart](#)



**Mixed Vegetable Pickle**  
A medley of seasonal vegetables in spices.  
★★★★★  
₹159

[Add To Cart](#)



**Carrot Pickle**  
Crunchy carrots with a spicy kick.  
★★★★★  
₹119

[Add To Cart](#)



**Green Chilli Pickle**  
Ferky green chilies in tangy spices.  
★★★★★  
₹139

[Add To Cart](#)

## Non-Veg pickles Page:

**HomeMade Pickles**

Home    [Veg Pickles](#)    **Non-Veg Pickles**    Snacks    My Cart    My Orders    Contact    [Logout](#)



**CHICKEN PICKLE**  
Classic Andhra-style chicken pickle.  
★★★★★  
₹149

[Add To Cart](#)



**Mutton Pickle**  
Zesty and tangy with a hint of spice.  
★★★★★  
₹129

[Add To Cart](#)



**Prawns Pickle**  
Spicy and tangy prawns pickle.  
★★★★★  
₹149

[Add To Cart](#)



**Fish Pickle**  
Aromatic fish with a spicy twist.  
★★★★★  
₹159

[Add To Cart](#)



**Egg Pickle**  
Spicy and tangy egg pickle.  
★★★★★  
₹119

[Add To Cart](#)



**Keema Pickle**  
Spicy minced meat pickle.  
★★★★★  
₹139

[Add To Cart](#)

## Snacks Page:

**HomeMade Pickles**

Home   Veg Pickles   Non-Veg Pickles   Snacks   My Cart   My Orders   Contact   Logout



**Murukku**  
Crispy and savory traditional snack.  
★★★★★  
₹149   [Add To Cart](#)



**Chekkalu**  
Crispy rice flour snacks with spices.  
★★★★★  
₹129   [Add To Cart](#)



**Chakli**  
Crispy and savory spiral snack.  
★★★★★  
₹149   [Add To Cart](#)



**Mixture**  
Aromatic mixture with a spicy twist.  
★★★★★  
₹159   [Add To Cart](#)



**Boondi**  
Crispy and savory gram flour balls.  
★★★★★  
₹119   [Add To Cart](#)



**Banana Chips**  
Crispy and crunchy banana chips.  
★★★★★  
₹139   [Add To Cart](#)

## My Cart:

**HomeMade Pickles**

Home   Veg Pickles   Non-Veg Pickles   Snacks   **My Cart**   My Orders   Contact   Logout

Mango Pickle added to cart! Carrot Pickle added to cart! Prawns Pickle added to cart! Boondi added to cart!

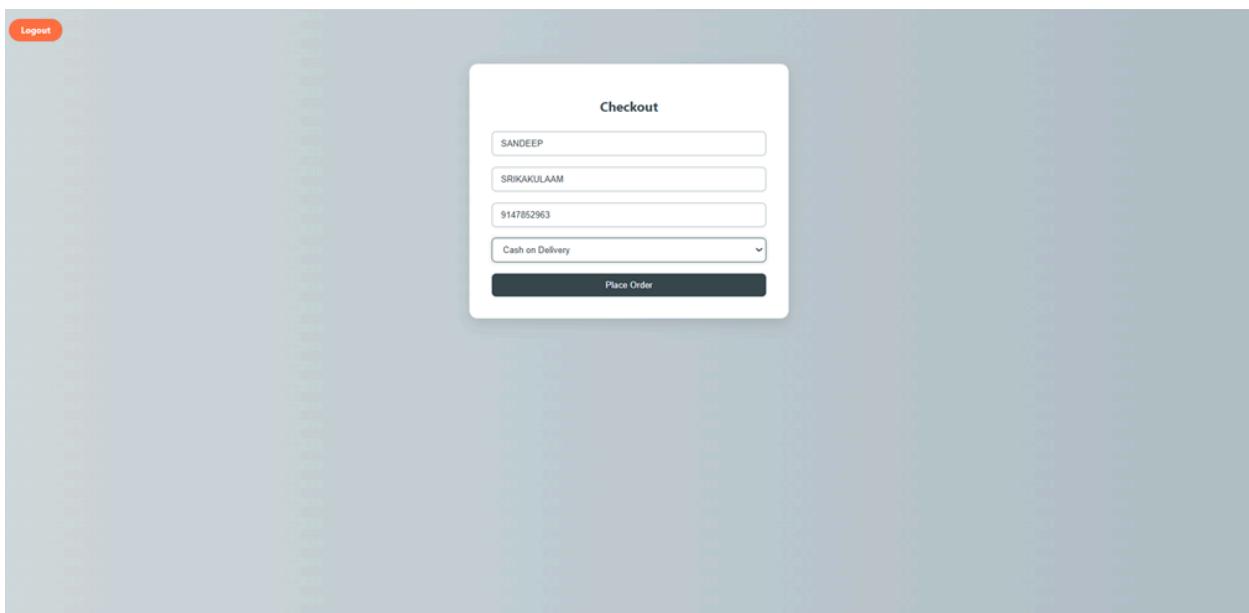
	<b>Mango Pickle</b> Quantity: 1	₹149 <a href="#">Remove</a>
	<b>Carrot Pickle</b> Quantity: 1	₹119 <a href="#">Remove</a>
	<b>Prawns Pickle</b> Quantity: 1	₹149 <a href="#">Remove</a>
	<b>Boondi</b> Quantity: 1	₹119 <a href="#">Remove</a>

Total: ₹536

[Proceed to Checkout](#)

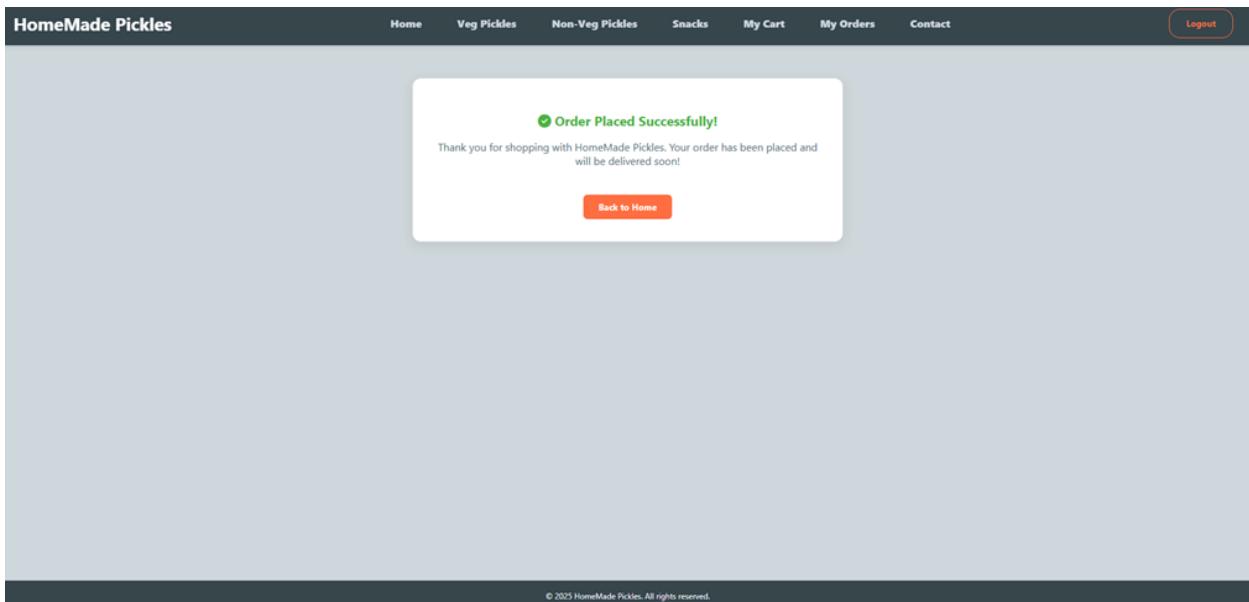
© 2025 HomeMade Pickles. All rights reserved.

## Checkout:



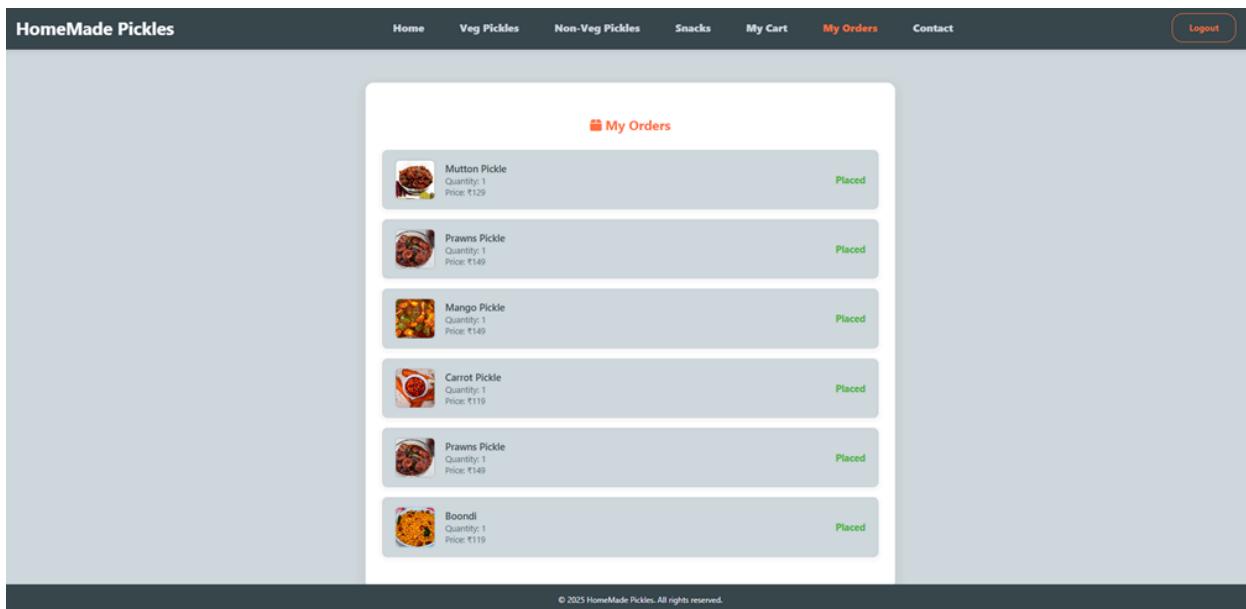
A screenshot of a checkout interface. At the top left is a "Logout" button. The main area is titled "Checkout" and contains four input fields: "SANDEEP", "SRIKAKULAAM", and "9147852963". Below these is a dropdown menu set to "Cash on Delivery". At the bottom is a large "Place Order" button.

### Conformation Page :



A screenshot of a confirmation page for an order placed successfully. The page has a dark header bar with the website name "HomeMade Pickles" and navigation links: Home, Veg Pickles, Non-Veg Pickles, Snacks, My Cart, My Orders, and Contact. A "Logout" button is also in the header. The main content area features a success message: "Order Placed Successfully!" with a checkmark icon, followed by a note: "Thank you for shopping with HomeMade Pickles. Your order has been placed and will be delivered soon!". A "Back to Home" button is at the bottom of this message. At the very bottom of the page is a small copyright notice: "© 2025 HomeMade Pickles. All rights reserved."

### Orders Page:



The screenshot shows the 'My Orders' section of the HomeMade Pickles website. It lists six orders, each with a small thumbnail image, the product name, quantity, price, and a green 'Placed' status indicator.

Product	Quantity	Price	Status
Mutton Pickle	1	₹129	Placed
Prawns Pickle	1	₹149	Placed
Mango Pickle	1	₹149	Placed
Carrot Pickle	1	₹119	Placed
Prawns Pickle	1	₹149	Placed
Boondi	1	₹119	Placed

## Conclusion:

The Home Made Pickles platform has been successfully developed and deployed using a robust, cloud-native architecture. By leveraging AWS EC2 for hosting, DynamoDB for data management, and SNS for real-time order and registration notifications, the application ensures reliable and scalable access to artisan pickle products. This system overcomes the limitations of traditional sales channels by providing customers with a seamless way to browse offerings, manage carts, and place orders—while makers can efficiently track and fulfill requests.

The cloud-native approach guarantees that as customer demand grows, the platform scales automatically without compromising performance. Integration of Flask with AWS services ensures that backend processes—including user authentication, product browsing, and order processing—run smoothly. Rigorous testing has validated every workflow, from signup through checkout notifications, confirming end-to-end reliability.

In summary, the Home Made Pickles platform offers a modern, efficient solution for small-scale food artisans, enhancing customer experience and operational efficiency. This project showcases the power of cloud-based architectures to transform niche businesses and drive growth.