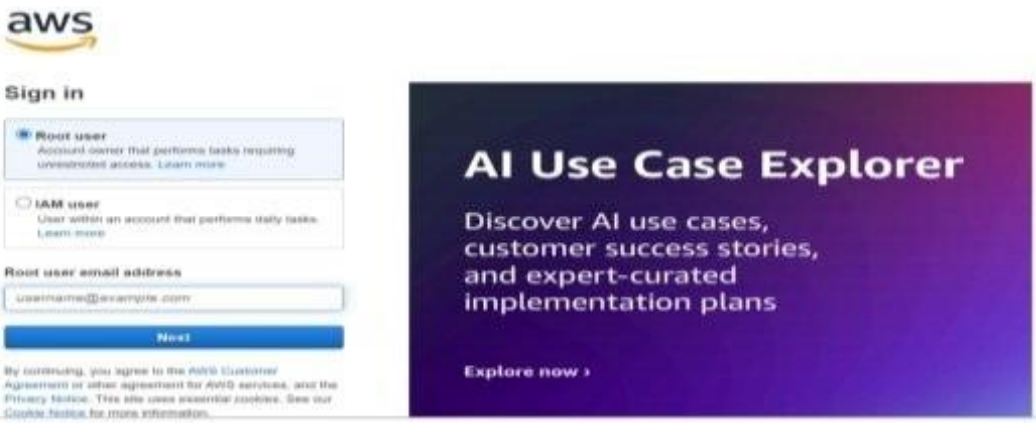
Milestone 1: AWS Account Setup and Login

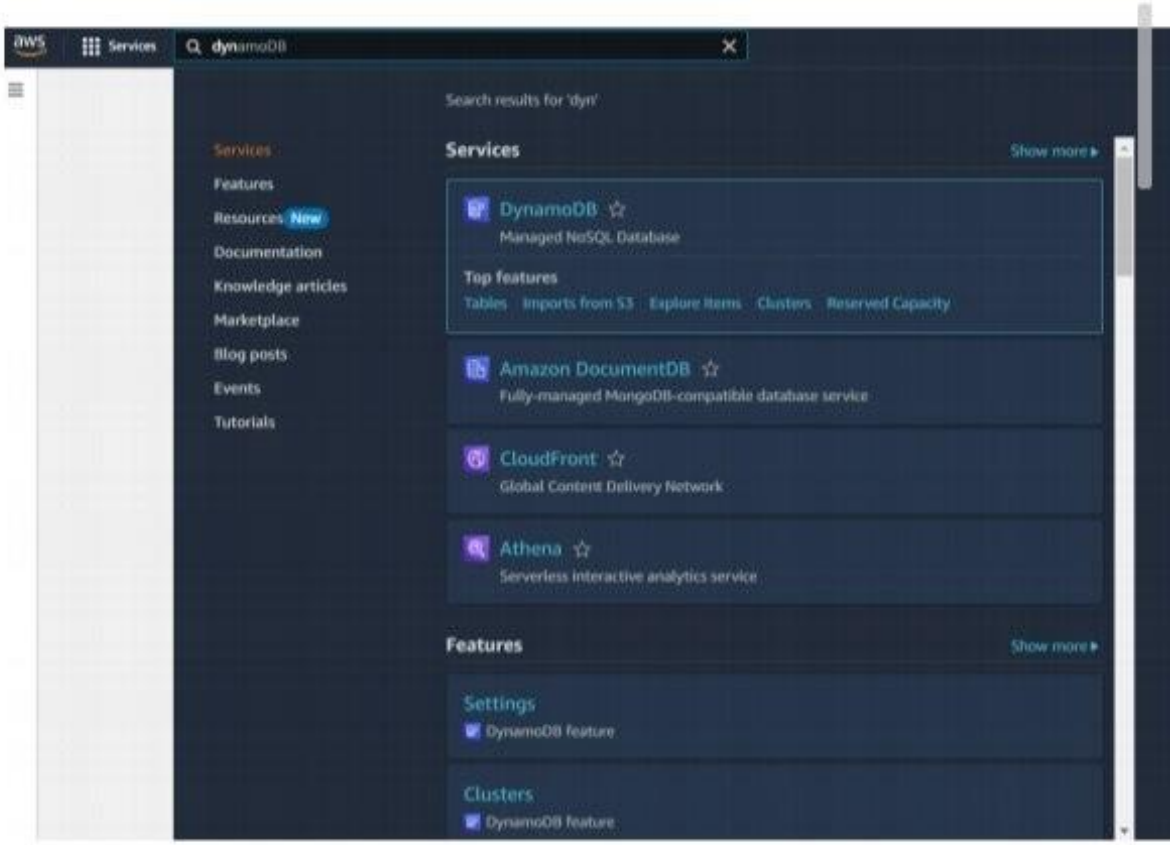
- **Activity 1.1: Set up an AWS account if not already done.**
  - Sign up for an AWS account and configure billing settings.

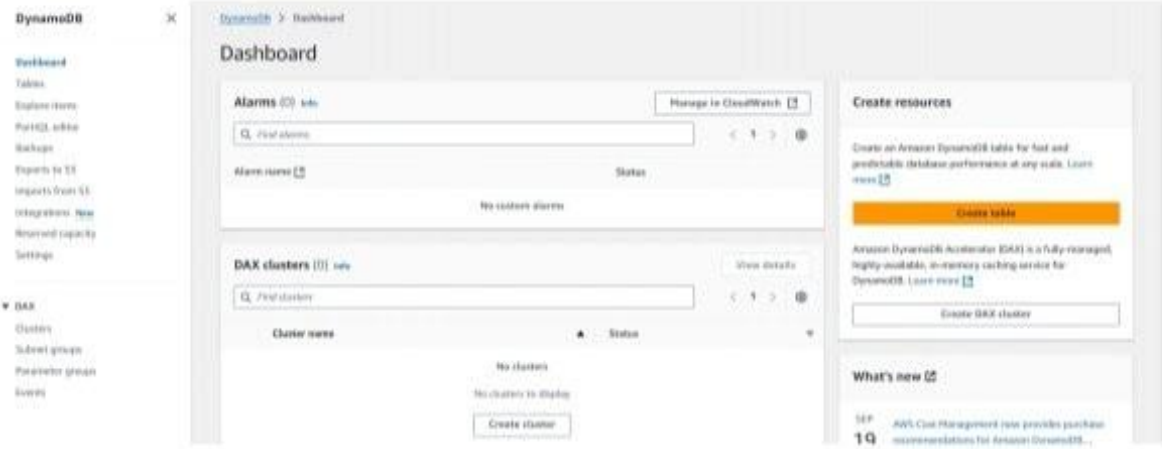
- **Activity 1.2: Log in to the AWS Management Console**
  - After setting up your account, log in to the AWS Management Console.



Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

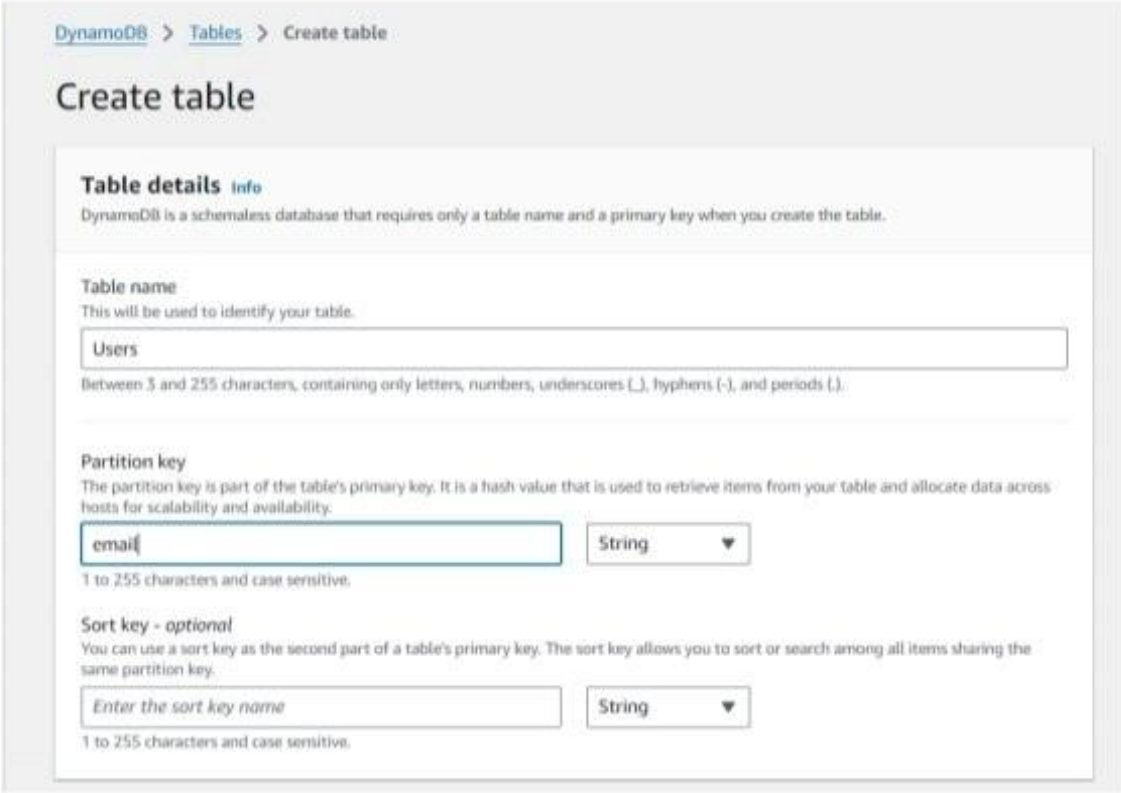
Milestone 2: DynamoDB Database Creation and Setup

- **Activity 2.1:Navigate to the DynamoDB**
  - In the AWS Console, navigate to DynamoDB and click on create tables.
  - 



Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

- **Activity 2.2:Create a DynamoDB table for storing registration details and book requests.**
    - Create Users table with partition key "Email" with type String and click on create tables.



Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

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

Tags

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

No tags are associated with the resource.

Add new tag

You can add 50 more tags.

Cancel Create table

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

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

DynamoDB > Tables > Create table

Create table

Table details Info

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

Table name

This will be used to identify your table.

Requests

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

Partition key

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

email | String

1 to 255 characters and case sensitive.

Sort key - *optional*

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

Enter the sort key name | String

1 to 255 characters and case sensitive.

Table settings

Default settings

Customize settings

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

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

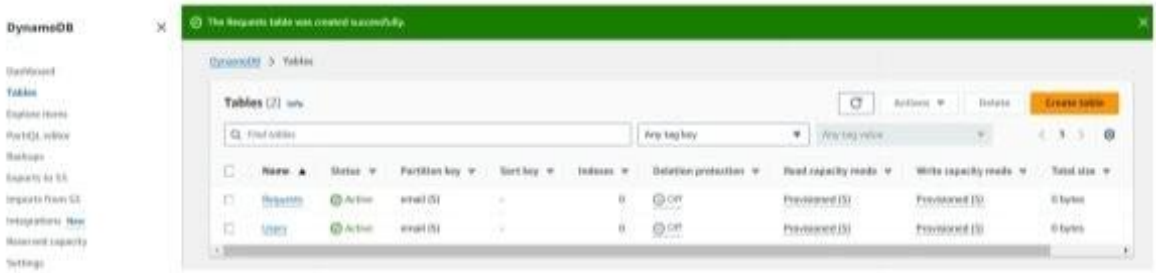
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

Add new tag

You can add 50 more tags.

Cancel | Create table

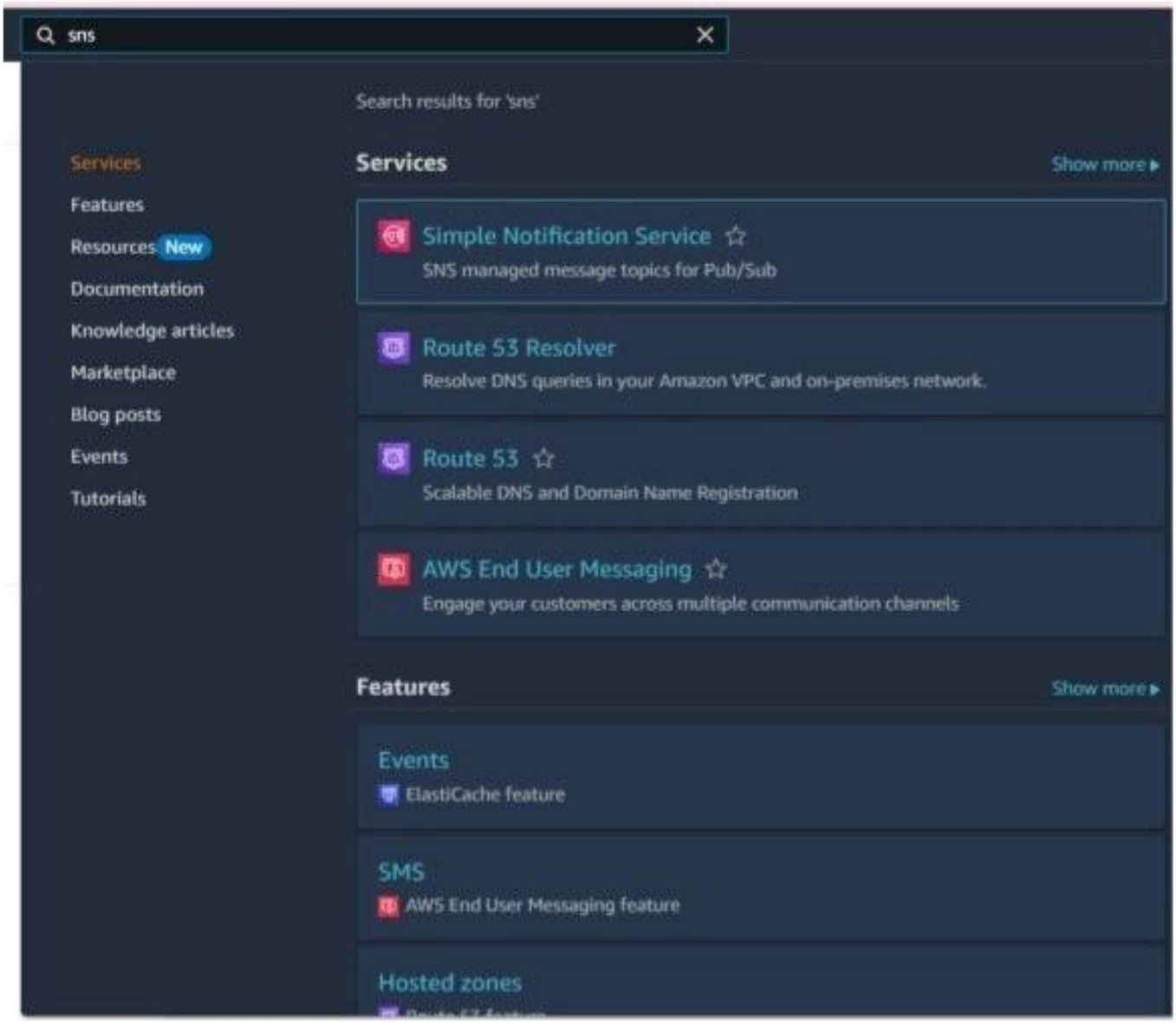


Milestone 3: SNS Notification Setup

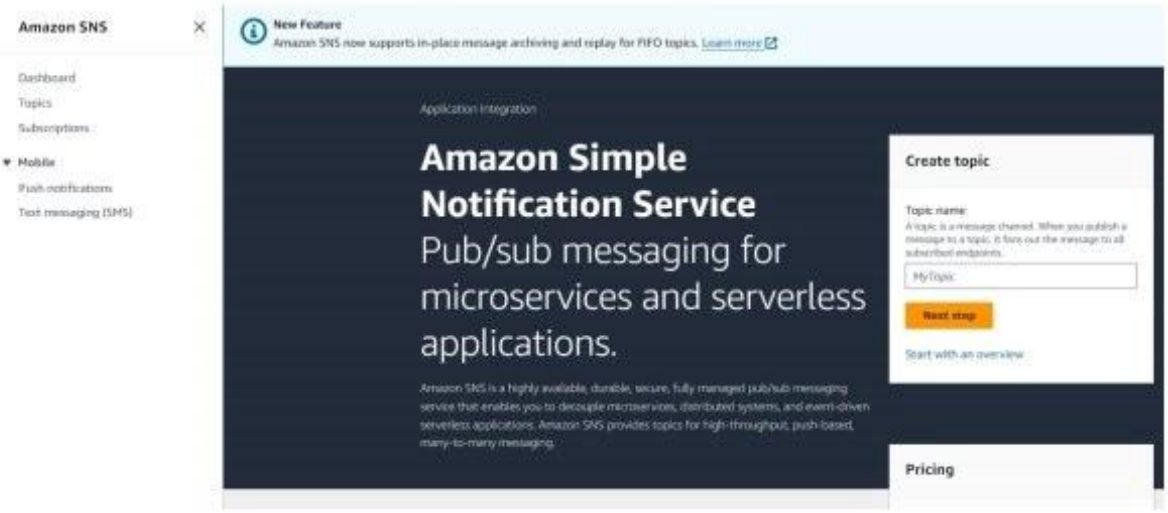
- **Activity 3.1: Create SNS topics for sending email notifications to users and library staff.**

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

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



Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/



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

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

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

Amazon SNS > Topics > Create topic

Create topic

Details

Type | Info

Topic type cannot be modified after topic is created

○ FIFO (first-in, first-out)

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

● Standard

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

Name

BookRequestNotifications

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

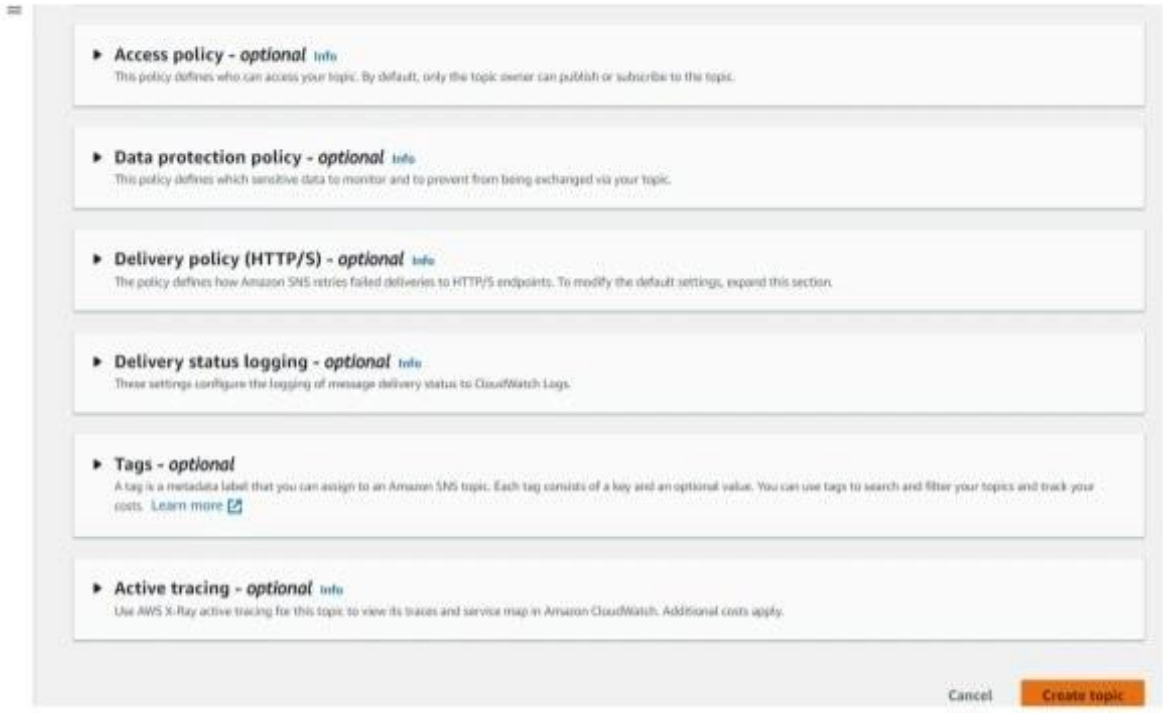
Display name - *optional* | Info

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

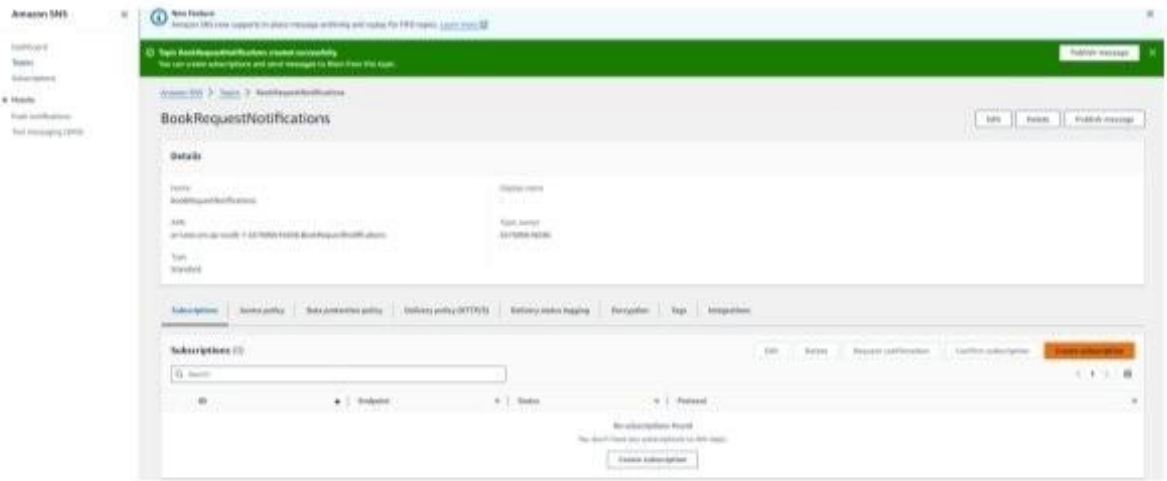
My Topic

Maximum 100 characters.

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/



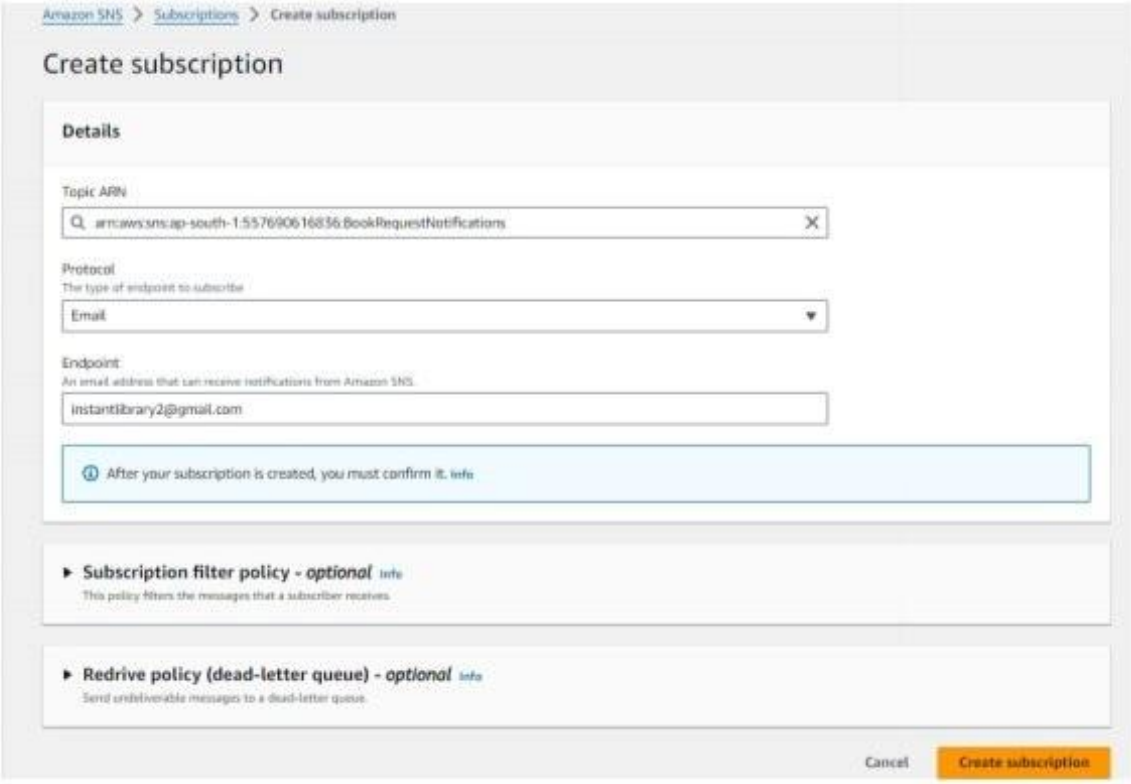
- Configure the SNS topic and note down the **Topic ARN**.



Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

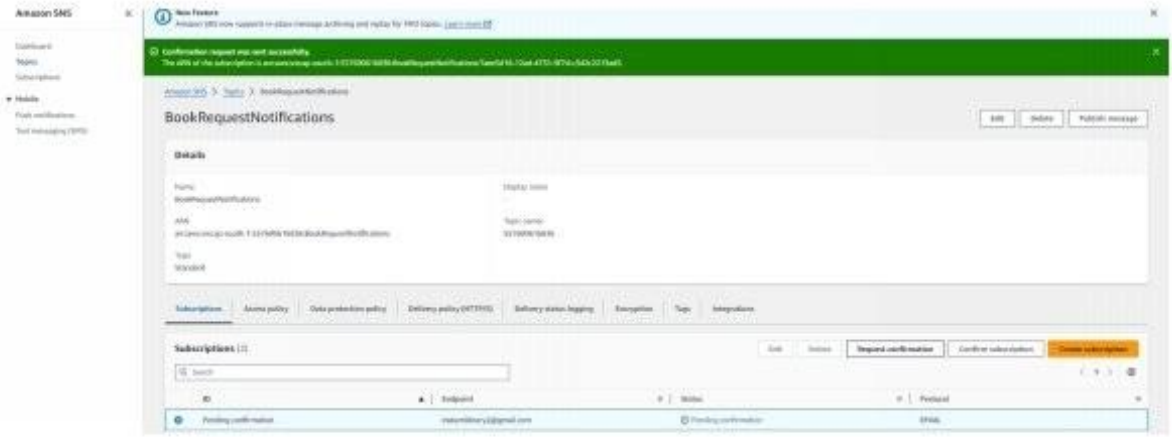
- **Activity 3.2: Subscribe users and staff to relevant SNS topics to receive real-time notifications when a book request is made.**

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



Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

- 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.

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

AWS Notification - Subscription Confirmation Inbox x

AWS Notifications <no-reply@sns.amazonaws.com> 9
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

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

aws

Simple Notification Service

Subscription confirmed!

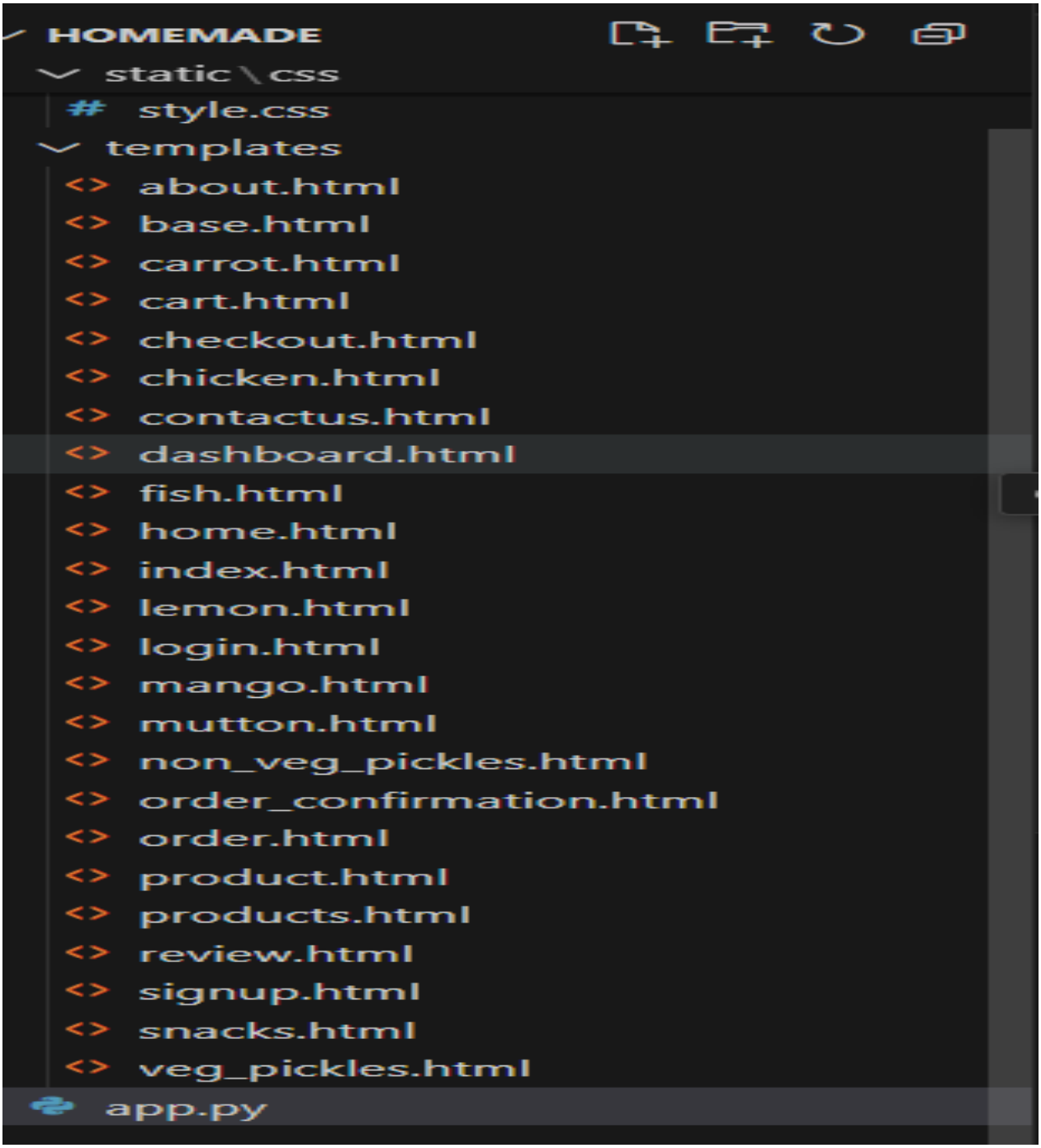
You have successfully subscribed.

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.

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/



Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

Description of the code :

? Flask App Initialization

```
from flask import Flask, render_template, request, redirect, url_for, session
from werkzeug.security import generate_password_hash, check_password_hash
import boto3
from datetime import datetime
import json,uuid
```

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

- Use boto3 to connect to DynamoDB for handling user registration, Order details database operations and also mention region_name where Dynamodb tables are created.

```
dynamodb = boto3.resource('dynamodb', region_name='ap-south-1')  # e.g., 'us-east-1'
users_table = dynamodb.Table('Users')
```

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

- Use boto3 to connect to DynamoDB for handling user registration, Order details database operations and also mention region_name where Dynamodb tables are created.

```
dynamodb = boto3.resource('dynamodb', region_name='ap-south-1')
users_table = dynamodb.Table('Users')
orders_table = dynamodb.Table('Orders')
```

```
products = {
    'non_veg_pickles': [
        {'id': 1, 'name': 'Chicken Pickle', 'weights': {'250': 600, '500': 1200, '1000': 1800}},
        {'id': 2, 'name': 'Fish Pickle', 'weights': {'250': 200, '500': 400, '1000': 800}},
        {'id': 3, 'name': 'Gongura Mutton', 'weights': {'250': 400, '500': 800, '1000': 1600}},
        {'id': 4, 'name': 'Mutton Pickle', 'weights': {'250': 400, '500': 800, '1000': 1600}},
        {'id': 5, 'name': 'Gongura Prawns', 'weights': {'250': 600, '500': 1200, '1000': 1800}},
        {'id': 6, 'name': 'Chicken Pickle (Gongura)', 'weights': {'250': 350, '500': 700, '1000': 1050}}
    ],
    'veg_pickles': [
        {'id': 7, 'name': 'Traditional Mango Pickle', 'weights': {'250': 150, '500': 280, '1000': 500}},
        {'id': 8, 'name': 'Zesty Lemon Pickle', 'weights': {'250': 120, '500': 220, '1000': 400}},
        {'id': 9, 'name': 'Tomato Pickle', 'weights': {'250': 130, '500': 240, '1000': 450}},
        {'id': 10, 'name': 'Kakarakaya Pickle', 'weights': {'250': 130, '500': 240, '1000': 450}},
        {'id': 11, 'name': 'Chintakaya Pickle', 'weights': {'250': 130, '500': 240, '1000': 450}},
        {'id': 12, 'name': 'Spicy Pandu Mirchi', 'weights': {'250': 130, '500': 240, '1000': 450}}
    ],  # Add your veg pickle products here
    'snacks': [
        {'id': 7, 'name': 'Banana Chips', 'weights': {'250': 300, '500': 600, '1000': 800}},
        {'id': 8, 'name': 'Crispy Aam-Papad', 'weights': {'250': 150, '500': 300, '1000': 600}},
        {'id': 9, 'name': 'Crispy Chekka Pakodi', 'weights': {'250': 50, '500': 100, '1000': 200}},
        {'id': 10, 'name': 'Boondhi Acchu', 'weights': {'250': 300, '500': 600, '1000': 900}},
        {'id': 11, 'name': 'Chekkalu', 'weights': {'250': 350, '500': 700, '1000': 1000}},
        {'id': 12, 'name': 'Ragi Laddu', 'weights': {'250': 350, '500': 700, '1000': 1000}},
        {'id': 13, 'name': 'Dry Fruit Laddu', 'weights': {'250': 500, '500': 1000, '1000': 1500}},
        {'id': 14, 'name': 'Kara Boondi', 'weights': {'250': 250, '500': 500, '1000': 750}},
        {'id': 15, 'name': 'Gavvalu', 'weights': {'250': 250, '500': 500, '1000': 750}},
        {'id': 16, 'name': 'Kaju Chikki', 'weights': {'250': 250, '500': 500, '1000': 750}},
```

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

```python
@app.route("/login", methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        try:
            response = users_table.get_item(Key={'username': username})
            if 'Item' not in response:
                return render_template('login.html', error="User not found")

            user = response['Item']
            if check_password_hash(user['password'], password):
                session['logged_in'] = True
                session['username'] = username
                session.setdefault('home', [])
                return redirect(url_for('home'))  # ✅ Add this to redirect after login
            else:
                return render_template('login.html', error="Incorrect password")
        except Exception as e:
            return render_template('login.html', error=f"An error occurred: {str(e)}")

    # ✅ This was missing
    return render_template('login.html')



@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        username = request.form['username'].strip()
        email =request.form['email'].strip()
        password= request.form['password']
        try:
            response = users_table.get_item(Key={'username': username})
            if 'Item' in response:
                return render_template('signup.html', error = 'Username already exists')
```

Web Page routing and login

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

```python
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        username = request.form['username'].strip()
        email =request.form['email'].strip()
        password= request.form['password']
        try:
            response = users_table.get_item(Key={'username': username})
            if 'Item' in response:
                return render_template('signup.html', error = 'Username already exists')

            hashed_password = generate_password_hash(password)

            users_table.put_item(
                Item={
                    'username': username,
                    'email': email,
                    'password': hashed_password,
                }
            )

            return redirect(url_for('login'))

        except Exception as e:
            app.logger.error(f"Signup error: {str(e)}")
            return render_template('signup.html', error='Registration failed. Please try again.')
    return render_template('signup.html')

@app.route('/logout')
def logout():
        session.clear()
        return redirect(url_for('login'))

@app.route('/home')
def home():
```

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

Homepage Routing: home page contains different routings veg_pickles,non_veg pickles,snacks,cart etc

pp.py > {} Flask

```python
@app.route('/non_vegpickles')
def non_vegpickles():
    return render_template('non_vegpickles.html', products=products ['non_vegpickles'])

@app.route('/veg_pickles')
def veg_pickles():
    # Simply pass all products without filtering
    return render_template('veg_pickles.html', products=products ['veg_pickles'])

@app.route('/snacks')
def snacks():
    return render_template('snacks.html', products=products['snacks'])

@app.route('/check_out', methods=['GET', 'POST'])
def check_out():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    error_message = None # Variable to hold error messages
    if request.method == 'POST':
        try:
            name = request.form.get('name', '').strip()
            address = request.form.get('address', '').strip()
            phone = request.form.get('phone', '').strip()
            payment_method = request.form.get('payment', '').strip()
            #Validate inputs
            if not all([name, address, phone, payment_method]):
                return render_template('check_out.html', error="All fields are required.")

            if not phone.isdigit() or len(phone) != 10:
                return render_template('check_out.html', error="Phone number must be exactly 10 digits.")
            #Get cart data from hidden inputs
            cart_data = request.form.get('cart_data', '[]')
            total_amount = request.form.get('total_amount', '0')
            try:
                cart_items = json.loads(cart_data)
                total_amount = float(total_amount)
```

templates > checkout.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Checkout</title>
</head>
<body>
    <h1>Checkout</h1>
    <p>Enter your shipping and payment details to complete the purchase.</p>
</body>
</html>
```

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

```
pp.py
            except (json.JSONDecodeError, ValueError):
                return render_template('check_out.html', error="Invalid cart data format.")
            # Ensure cart is not empty
            if not cart_items:
                return render_template('check_out.html', error="Your cart is empty.")
            try:
                orders_table.put_item(
                    Item={
                        'order_id': str(uuid.uuid4()),
                        'username': session.get('username', 'Guest'),
                        'name': name,
                        'address': address,
                        'phone': phone,
                        'items': cart_items,
                        'total_amount': total_amount,
                        'payment_method': payment_method,
                        'timestamp': datetime.now().isoformat()
                    }
                )
            except Exception as db_error:
                print(f"DynamoDB Error: {db_error}")
                return render_template('check_out.html', error="Failed to save order. Please try again later.")
            # Redirect to success page with success message
            return redirect(url_for('success', message="Your order has been placed successfully!"))
        except Exception as e:
            print(f"Checkout error: {str(e)}")
            return render_template('checkout.html', error="An unexpected error occurred. Please try again.")
    return render_template('check_out.html')

@app.route('/cart', methods=['GET', 'POST'])
def cart():
    if request.method == 'POST':
        if 'cart' not in session:
            session['cart'] = []

        # Fetch form data
        product_id = request.form.get('product_id')
```

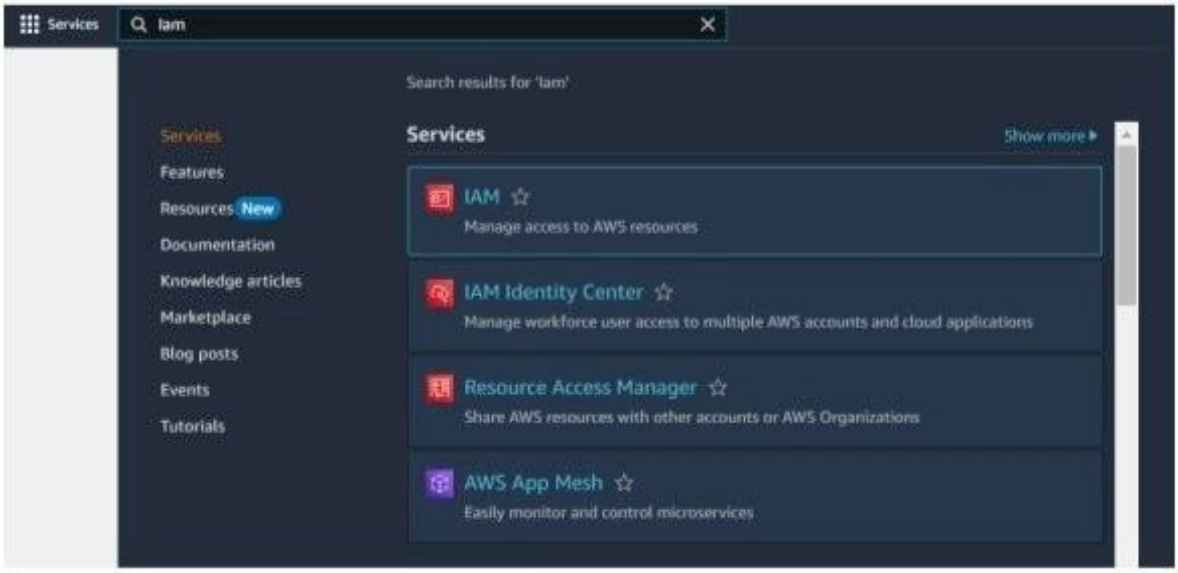
Ln 2, Col 1 Spaces: 4 UTF-8

```
pp.py
            session.modified = True


    return render_template('cart.html', cart=session.get('cart', []))


@app.route('/success')
def success():
    return render_template('success.html')

@app.route('/about')
def about():
    return render_template('about.html')

def send_email(to_email, subject, body):
    try:
        msg = MIMEMultipart()
        msg['From'] = EMAIL_ADDRESS
        msg['To'] = to_email
        msg['Subject'] = subject
        msg.attach(MIMEText(body, 'plain'))

        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login(EMAIL_ADDRESS, EMAIL_PASSWORD)
        server.send_message(msg)
        server.quit()
    except Exception as e:
        print(f"Failed to send email: {e}")

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

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

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.



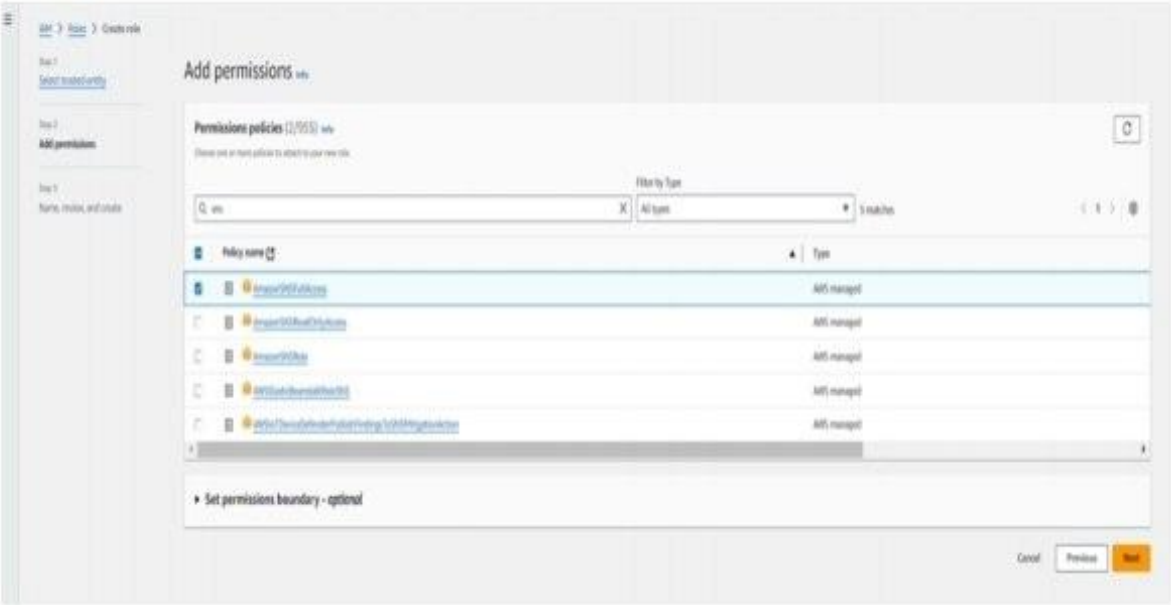
Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

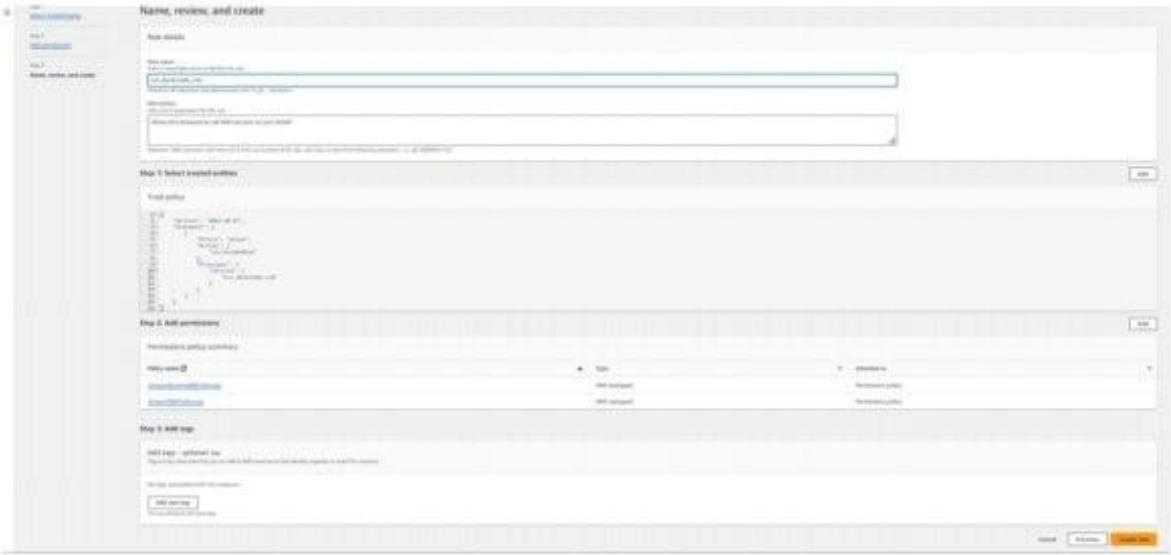
- **Activity 5.2: Attach Policies.**

  Attach the following policies to the role:

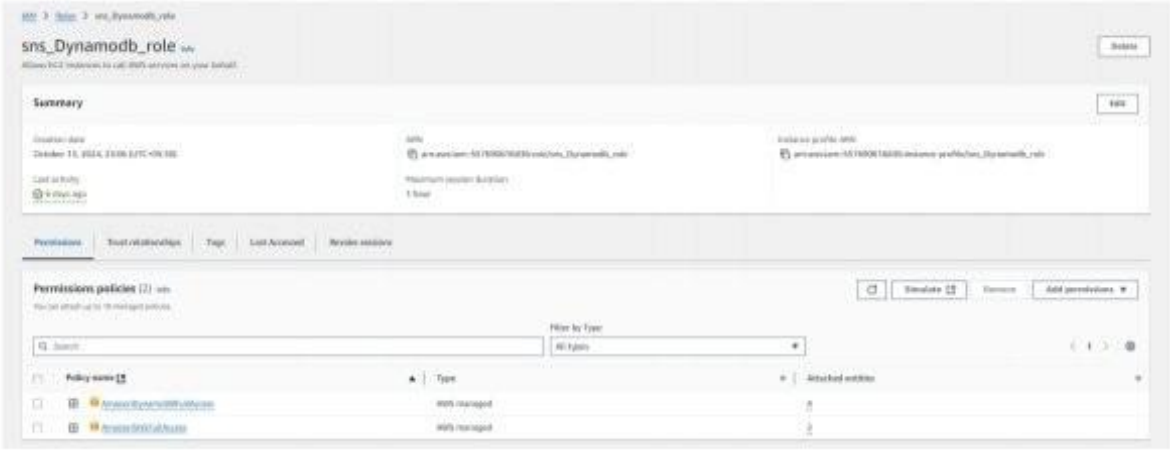
  - AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.
  - AmazonSNSFullAccess: Grants EC2 the ability to send notifications via SNS.

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/





Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/



Milestone 6: EC2 Instance Setup

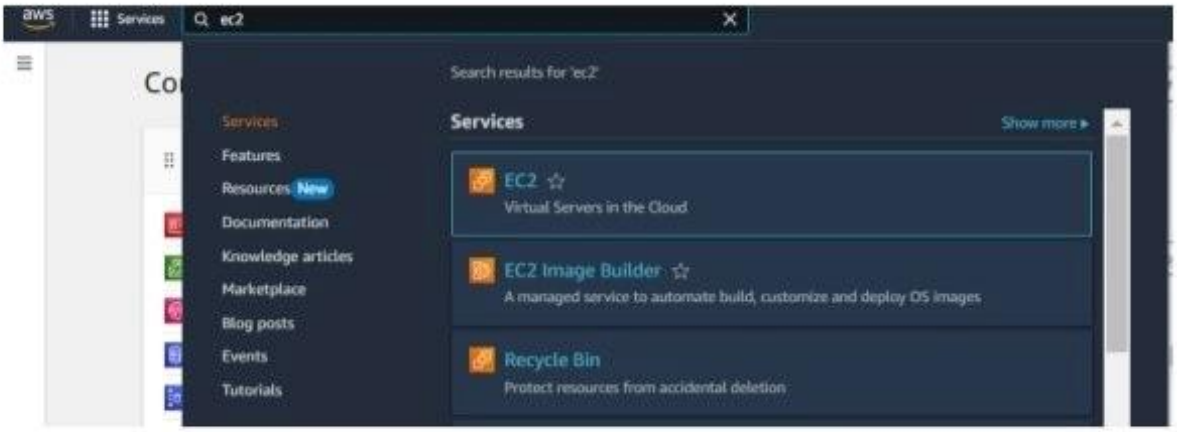
- **Note: Load your Flask app and Html files into GitHub repository.**

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

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

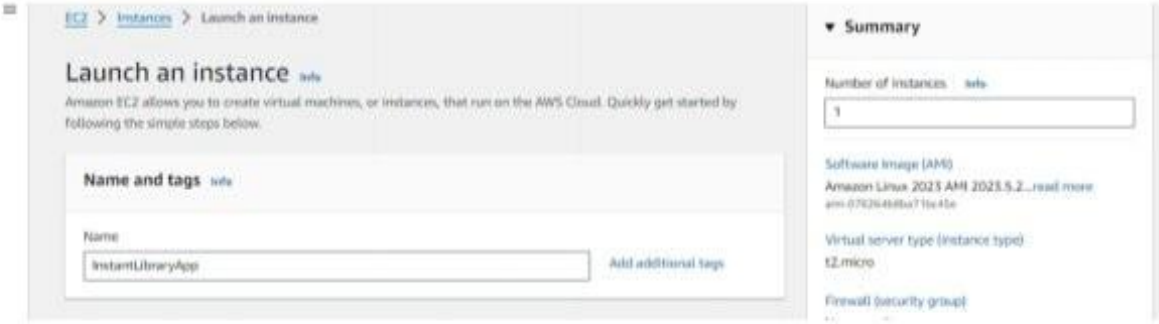
- **Activity 6.1: Launch an EC2 instance to host the Flask application.**

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

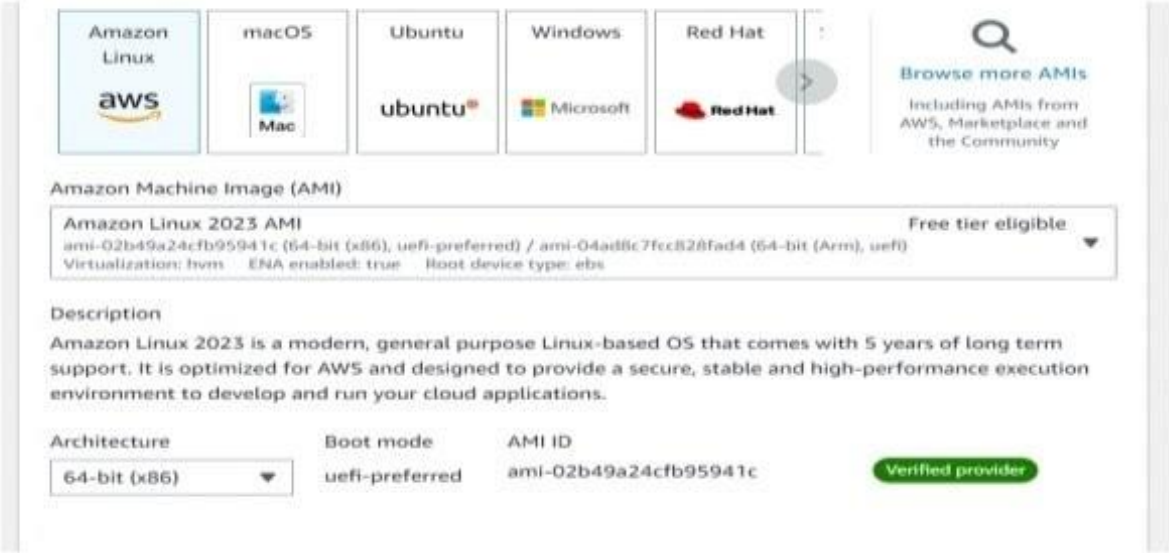


Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

- Click on Launch instance to launch EC2 instance

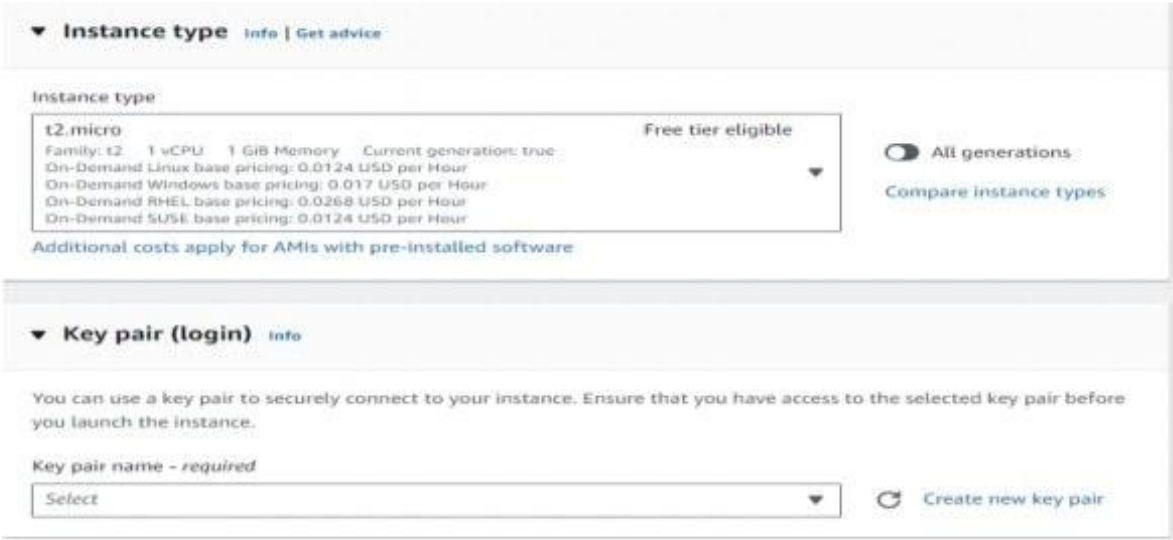


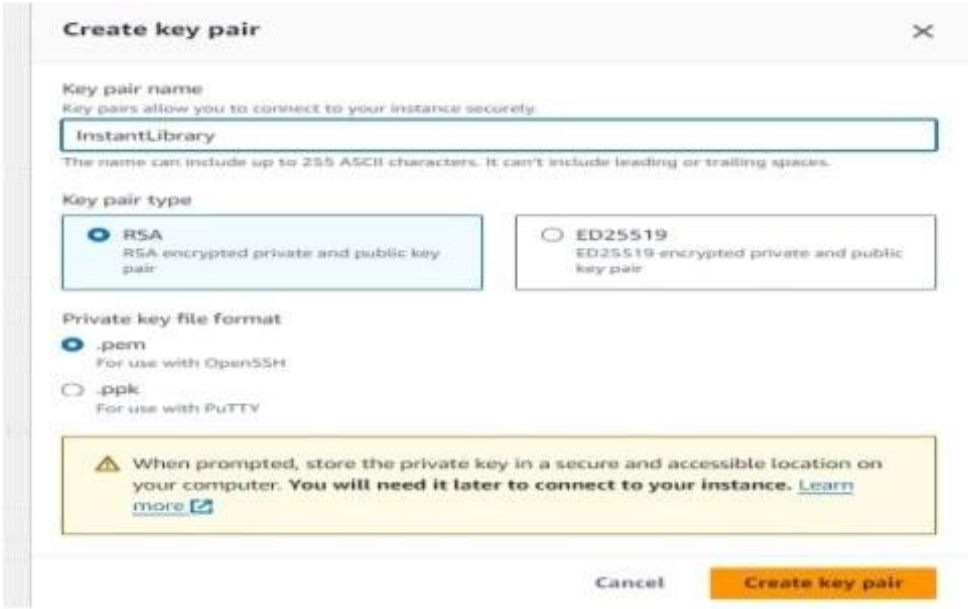
- Choose Amazon Linux 2 or Ubuntu as the AMI and t2.micro as the instance type (free-tier eligible).



Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

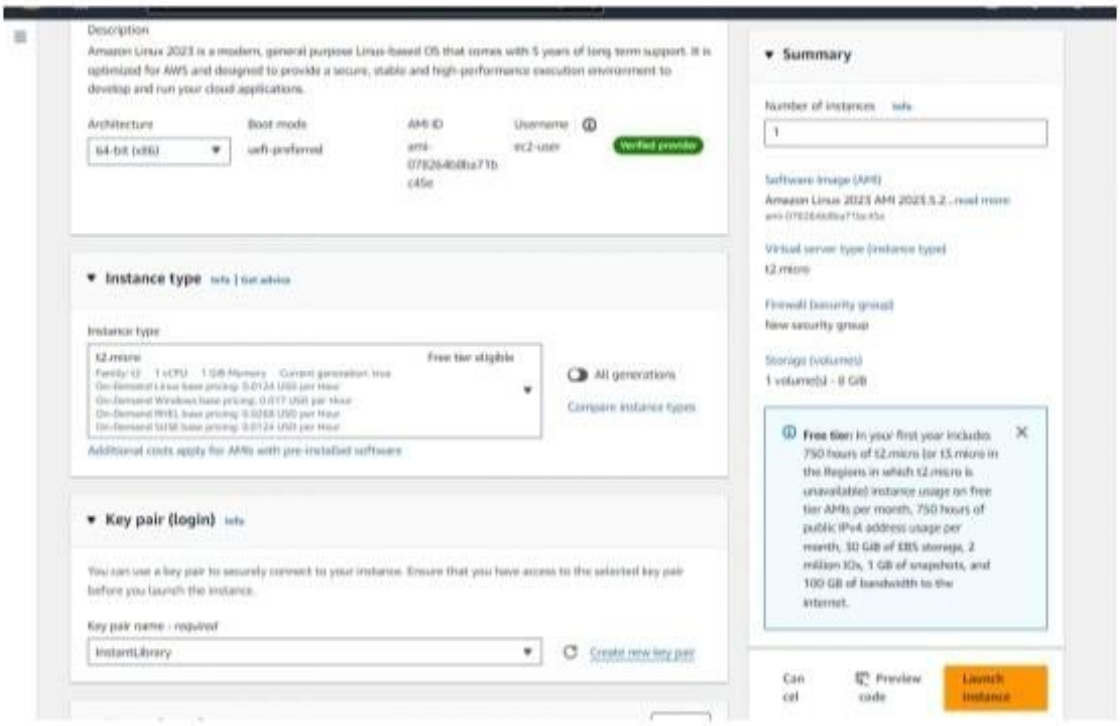
- Create and download the key pair for server access.





Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

InstantLibrary.pem



Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

Network settings Info

VPC - *required* Info

vpc-03cdc7b6f19dd7211 (default)
172.31.0.0/16

Subnet Info

No preference — Create new subnet

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

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

- (•) Create security group
- ( ) Select existing security group

Security group name - *required*

launch-wizard

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

Description - *required* Info

launch-wizard created 2024-10-13T17:49:56.622Z

Inbound Security Group Rules

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

Type	Protocol	Port range
ssh	TCP	22

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

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

Type	Protocol	Port range
HTTP	TCP	80

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

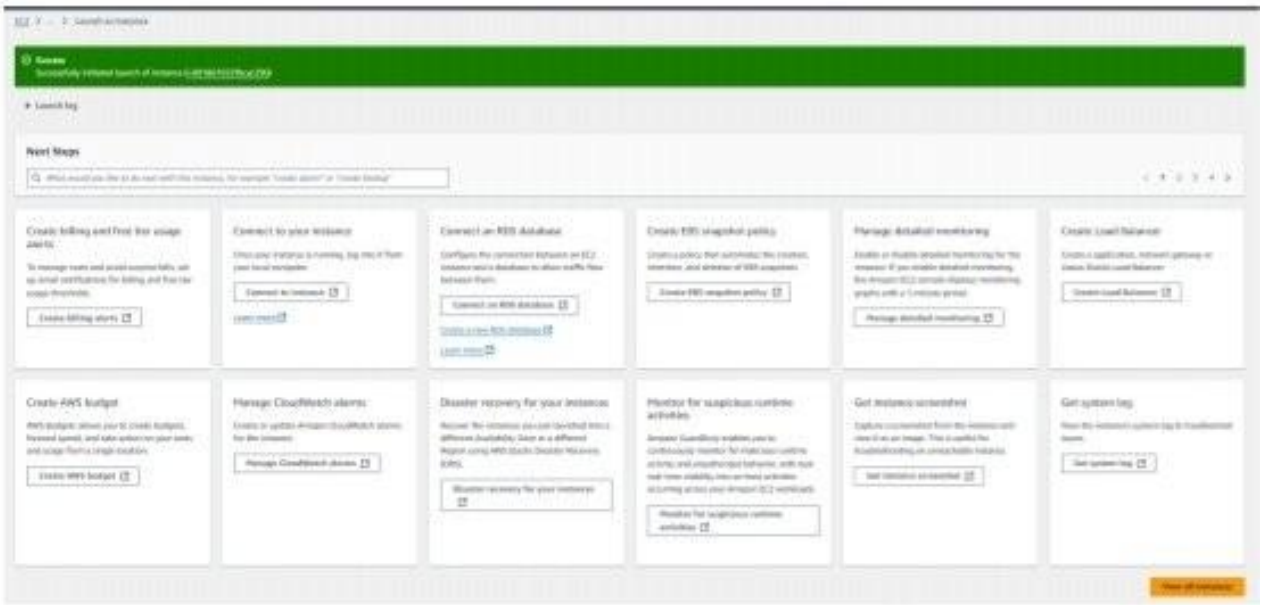
Security group rule 3 (TCP, 5000, 0.0.0.0/0) — Remove

Type	Protocol	Port range
Custom TCP	TCP	5000

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

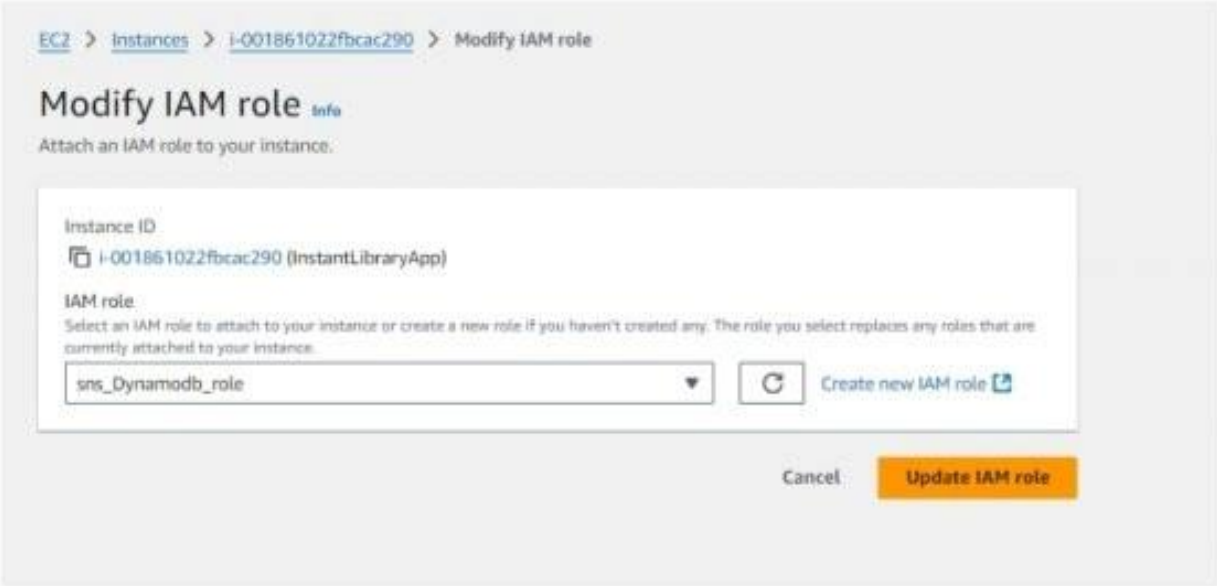
Add security group rule

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/



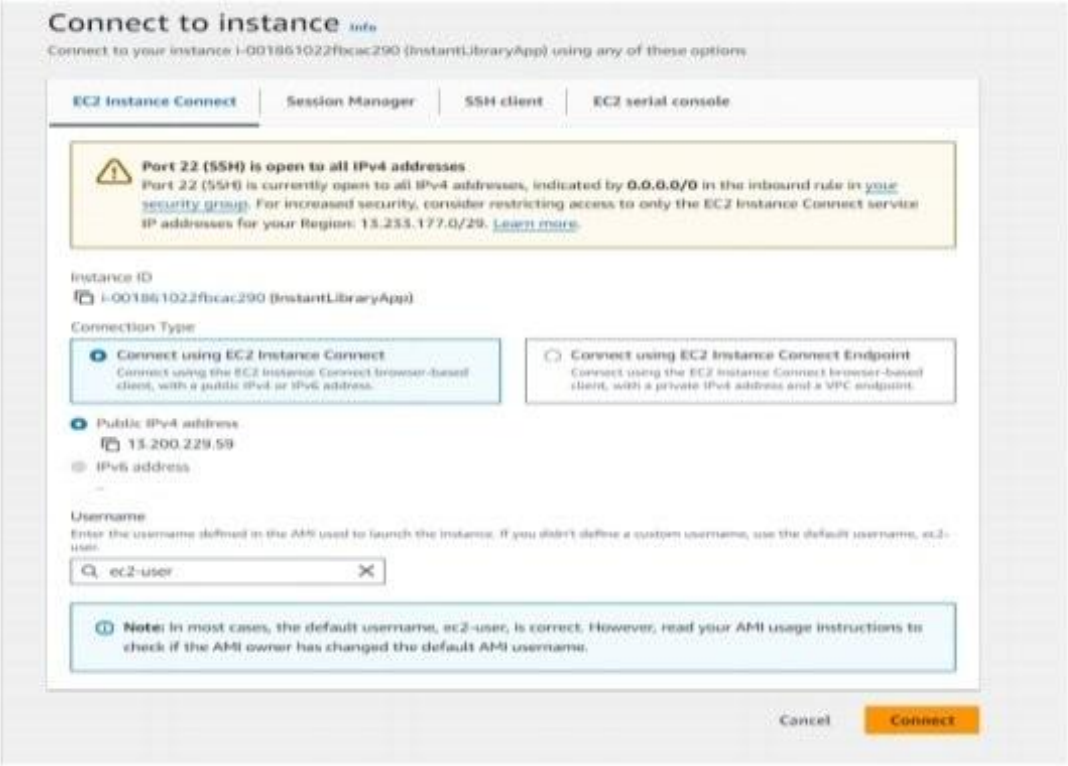
- To connect to EC2 using **EC2 Instance Connect**, start by ensuring that an **IAM role** is attached to your EC2 instance. You can do this by selecting your instance, clicking on **Actions**, then navigating to **Security** and selecting **Modify IAM Role** to attach the appropriate role. After the 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.

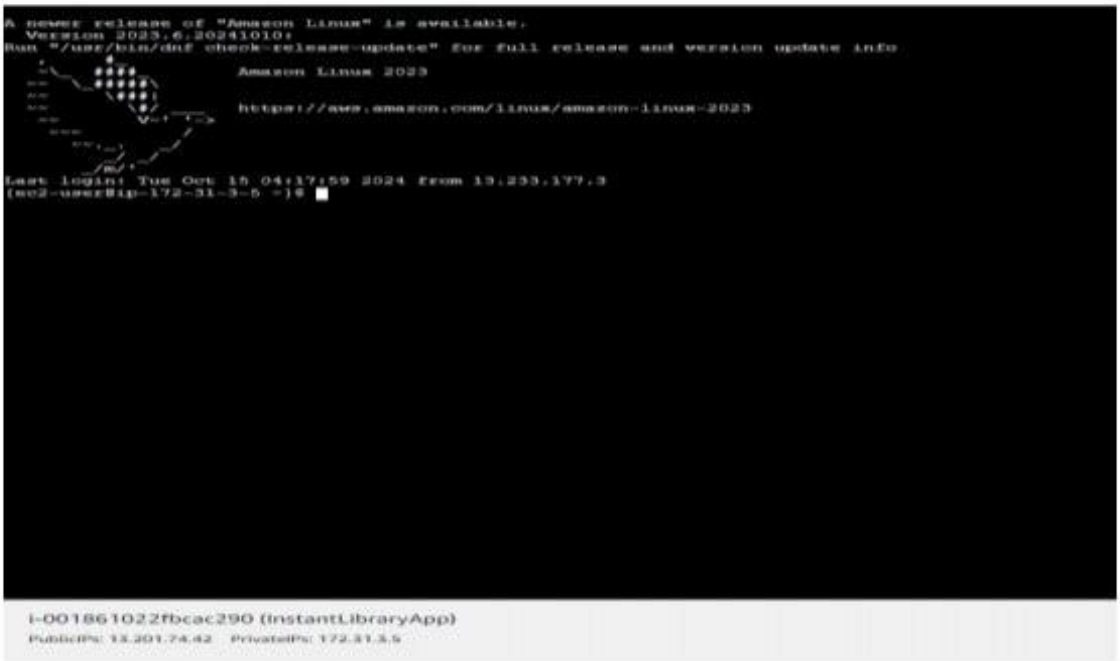
Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/



- Now connect the EC2 with the files

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/





Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/

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
```

Created in Photo-to-PDF One Click Converter. Download here: https://firehawk.ai/phototopdf/