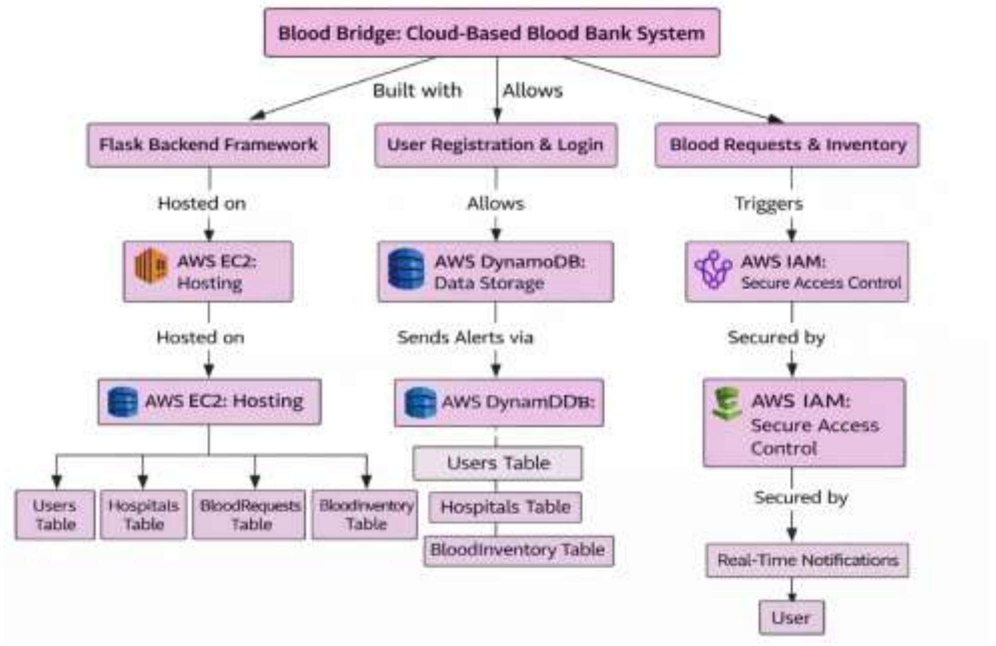
Blood Bridge: Optimizing Lifesaving resources using AWS Services

Project Description:

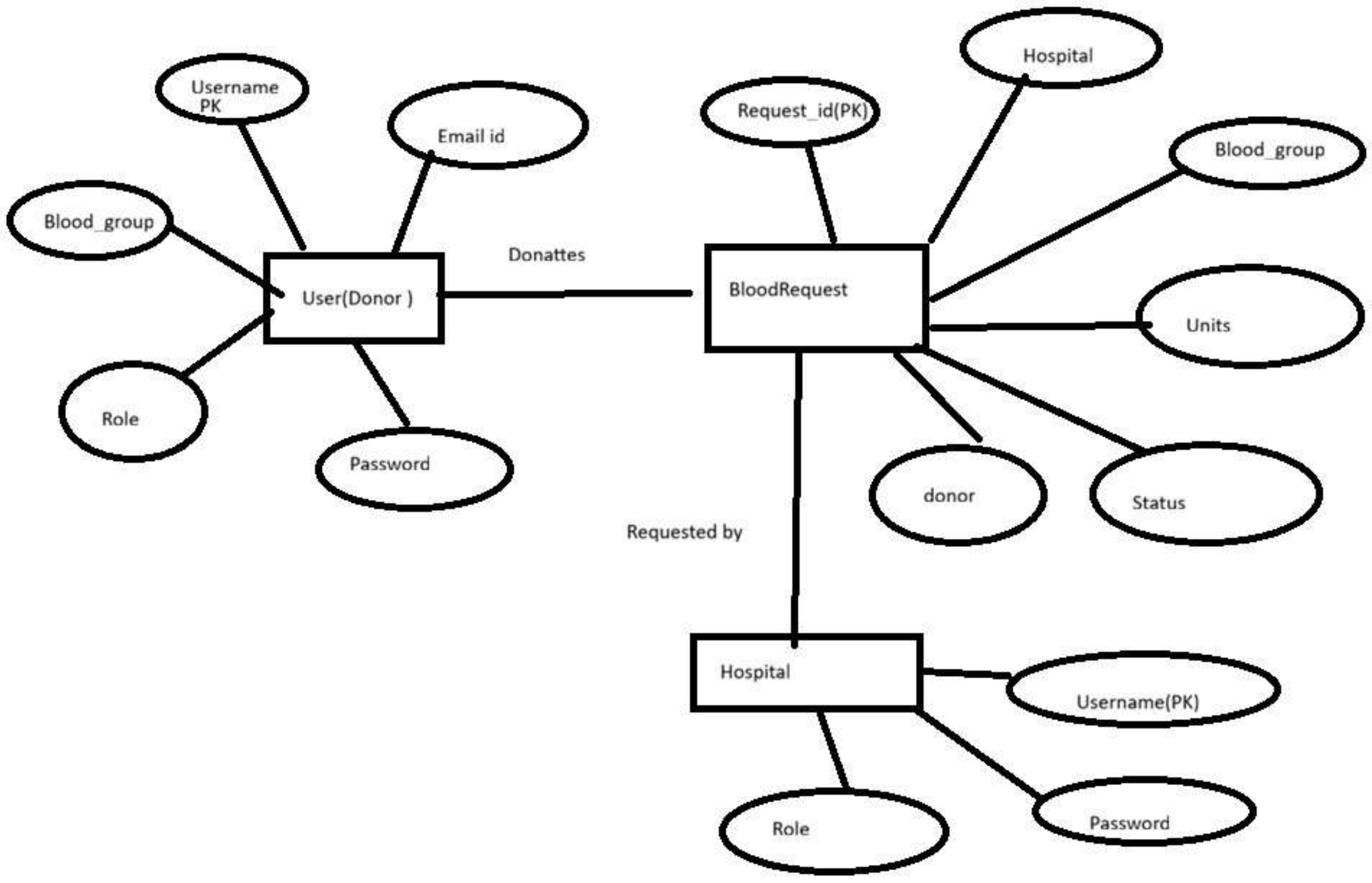
"BloodBridge" is a comprehensive web-based blood bank management system designed to streamline the process of blood donation and distribution. The project leverages Amazon Web Services (AWS) for robust and scalable infrastructure, utilizing Amazon DynamoDB for secure and efficient data storage and Amazon EC2 for reliable web hosting. The user-friendly web interface allows individuals to register and log in to their personal accounts, creating a seamless experience for both donors and recipients. Once logged in, users are presented with a dashboard that serves as a central hub for all blood-related activities.
The dashboard prominently features current blood requests, allowing users to view real-time needs in their community. Additionally, registered users can easily submit their own blood requests, specifying blood type, quantity, and urgency. This system not only facilitates quick responses to critical blood needs but also fosters a sense of community engagement in the life-saving act of blood donation. By combining modern cloud technology with an intuitive user interface, "BloodBridge" aims to bridge the gap between blood donors and those in need, ultimately saving lives and improving healthcare outcomes.

Scenarios
Scenario 1: Emergency Blood Request
Sarah, a hospital administrator, logs into Life Link during a critical situation. A patient needs a rare blood type urgently. Using her dashboard, Sarah quickly submits a high-priority blood request, specifying the required blood type and quantity. The system immediately notifies potential donors in the area, significantly reducing the time to find a match and potentially saving the patient's life.
Scenario 2: Regular Donor Management
John, a regular blood donor, uses Life Link to manage his donations. After logging in, he checks his dashboard to see when he's eligible to donate again. He notices a nearby blood drive event listed in the requests section. John uses the system to schedule his next donation, helping maintain a steady supply of blood for the local hospitals.
Scenario 3: Blood Bank Inventory Update
A blood bank manager, Lisa, uses Life Link to update the current blood inventory. She logs into her specialized account and accesses a feature to input the latest stock levels for each blood type. The system automatically updates the dashboard for all users, reflecting the current needs. This real-time update helps prioritize requests for blood types that are running low, ensuring efficient distribution of this vital resource.

AWS ARCHITECTURE



Entity Relationship (ER)Diagram:



Pre-requisites:

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

Project WorkFlow:

1. AWS Account Setup and Login

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

Activity 1.2: Log in to the AWS Management Console

2. DynamoDB Database Creation and Setup

Activity 2.1: Create a DynamoDB Table.

Activity 2.2: Configure Attributes for User Data and Book Requests.

3. SNS Notification Setup

Activity 3.1: Create SNS topics for book request notifications.

Activity 3.2: Subscribe users and library staff to SNS email notifications.

4. Backend Development and Application Setup

Activity 4.1:Develop the Backend Using Flask.

Activity 4.2: Integrate AWS Services Using boto3.

5. IAM Role Setup

Activity 5.1: Create IAM Role

Activity 5.2: Attach Policies

6. EC2 Instance Setup

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

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

7. Deployment on EC2

Activity 7.1:Upload Flask Files

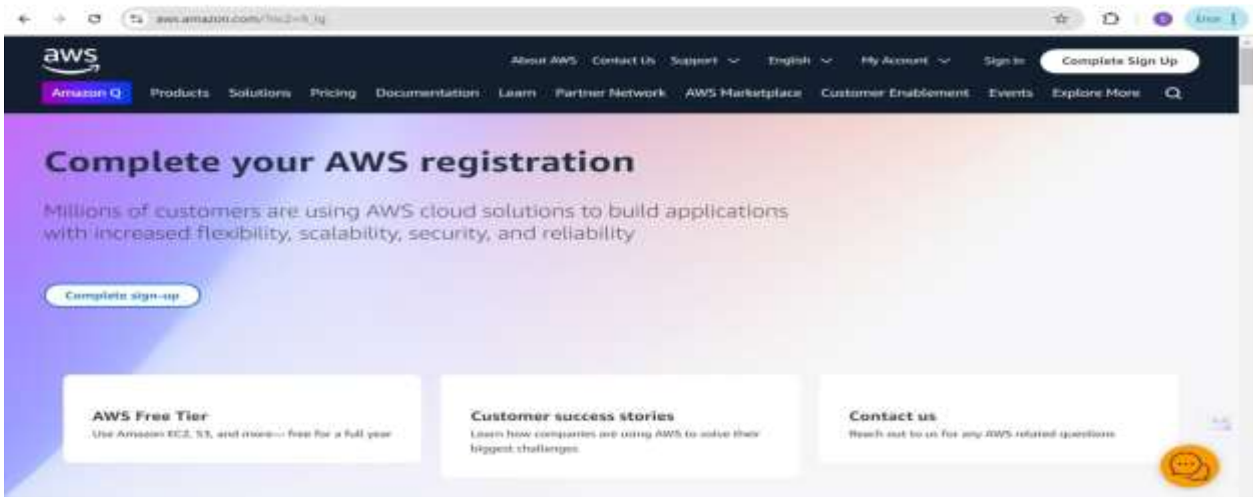
Activity 7.2: Run the Flask App

8. Testing and Deployment

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

Milestone 1: AWS Account Setup and Login

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



Complete your AWS registration

Millions of customers are using AWS cloud solutions to build applications with increased flexibility, scalability, security, and reliability

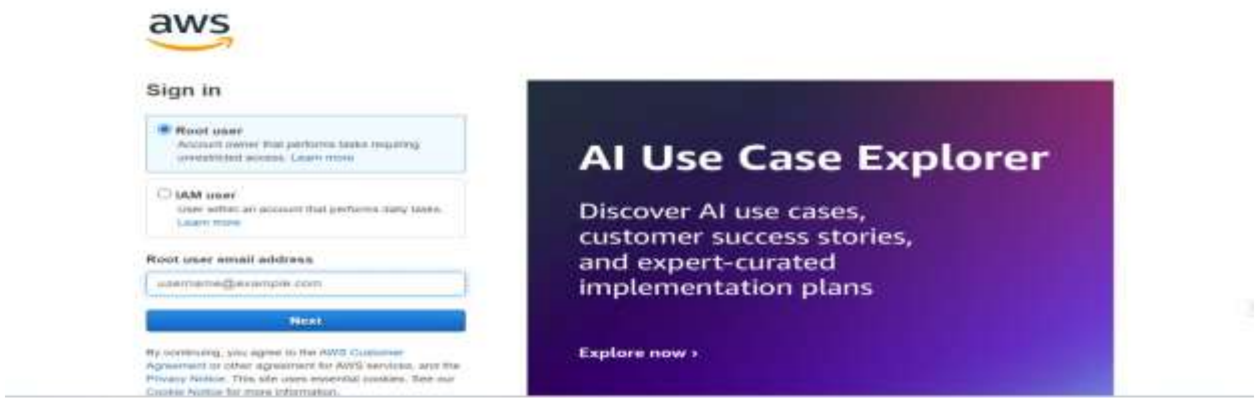
Complete sign-up

AWS Free Tier

Customer success stories

Contact us

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



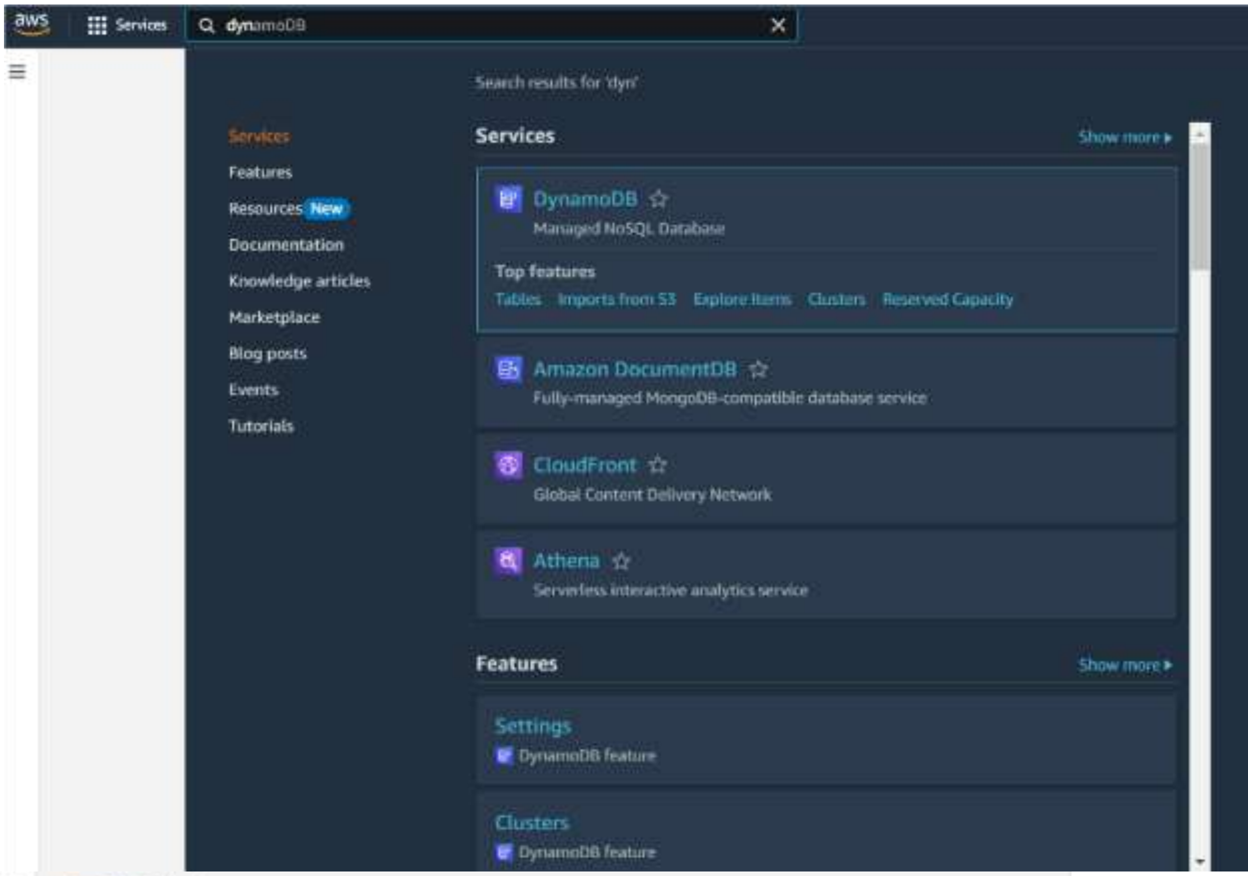
AI Use Case Explorer

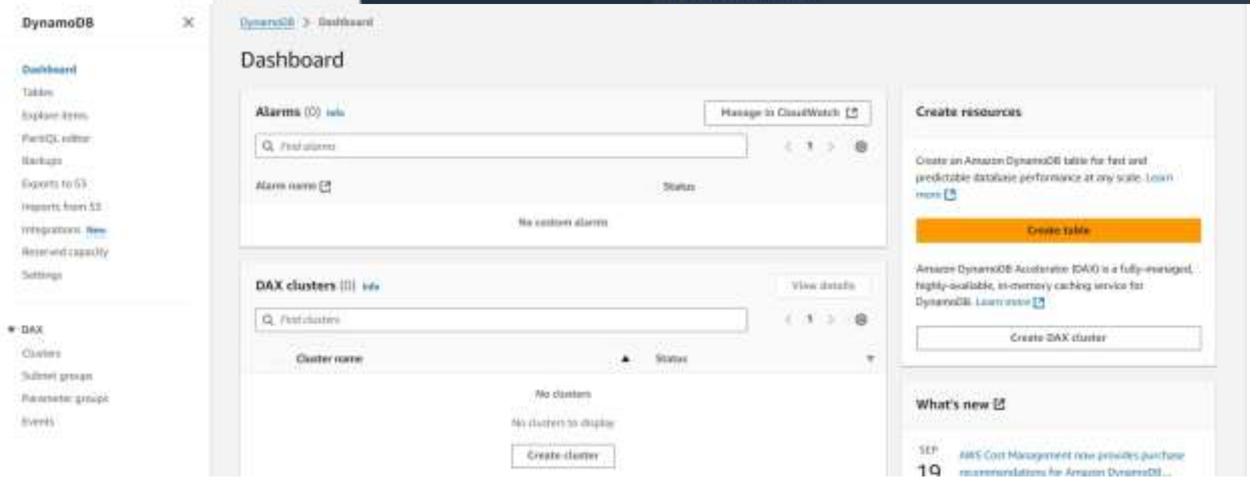
Discover AI use cases, customer success stories, and expert-curated implementation plans

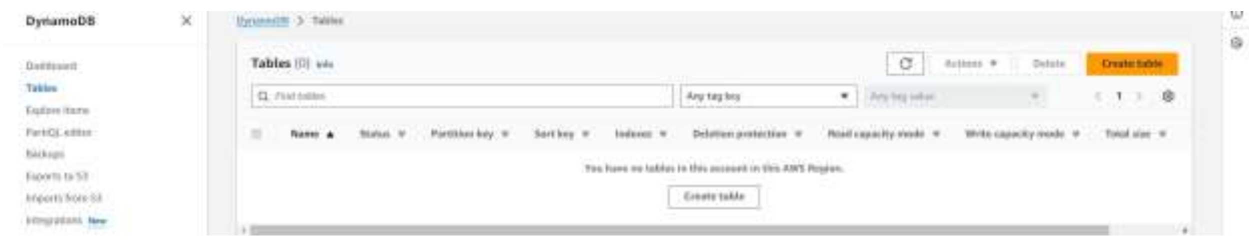
Explore now ›

Milestone 2: mongoDatabase Creation and Setup

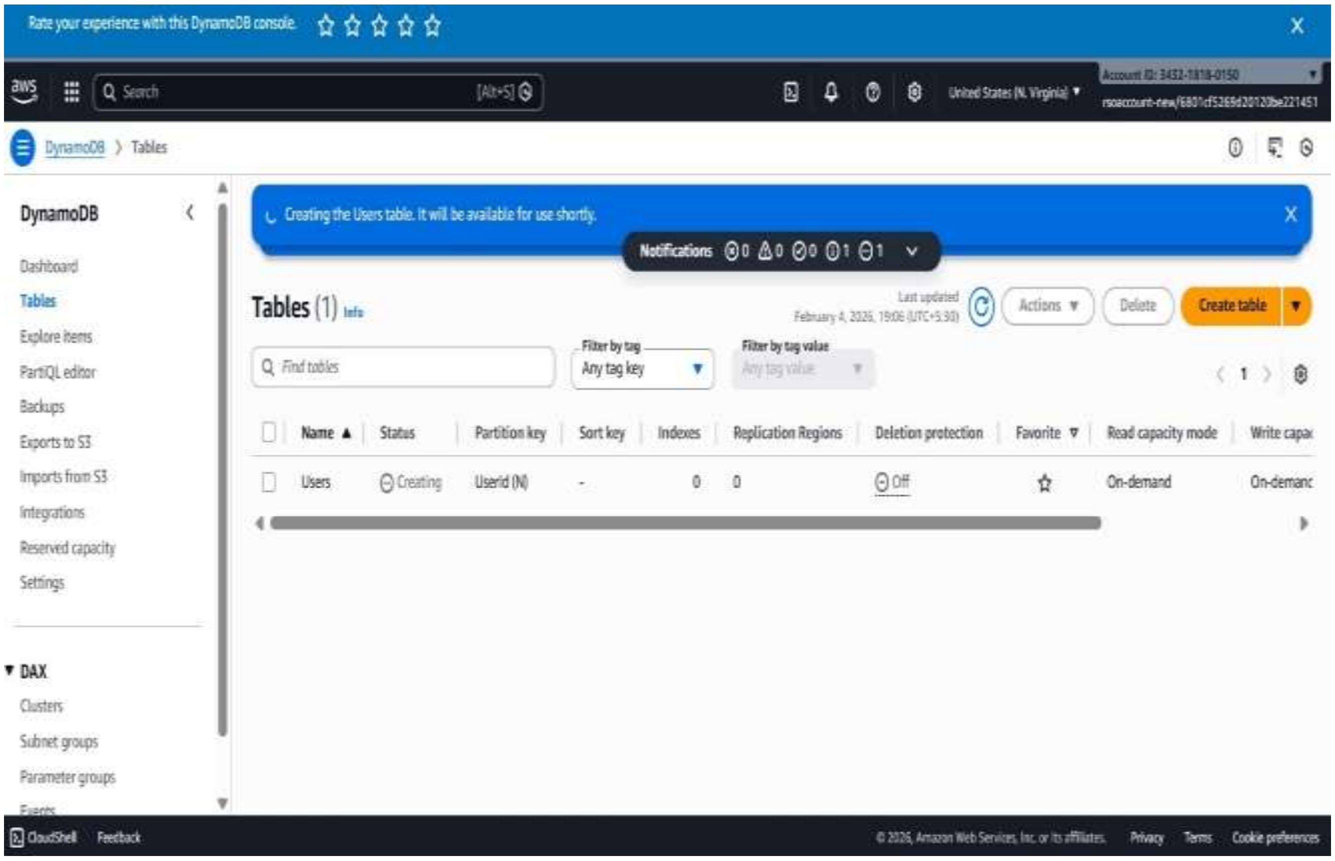
- **Activity 2.1:mongo to the DynamoDB**
  - In the AWS Console, navigate to DynamoDB and click on create tables.

○ 
aws
Services
dynamoDB
Search results for 'dyn'
Services
Features
Resources New
Documentation
Knowledge articles
Marketplace
Blog posts
Events
Tutorials
Services
Show more ▸
DynamoDB
Managed NoSQL Database
Top features
Tables
Imports from S3
Explore items
Clusters
Reserved Capacity
Amazon DocumentDB
Fully-managed MongoDB-compatible database service
CloudFront
Global Content Delivery Network
Athena
Serverless interactive analytics service
Features
Show more ▸
Settings
DynamoDB feature
Clusters
DynamoDB feature


DynamoDB
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New
Reserved capacity
Settings
DAX
Clusters
Subnet groups
Parameter groups
Events
DynamoDB > Dashboard
Dashboard
Alarms (0) Info
Manage in CloudWatch
Find alarms
Alarm name
Status
No custom alarms
DAX clusters (0) Info
View details
Find clusters
Cluster name
Status
No clusters
No clusters to display
Create cluster
Create resources
Create an Amazon DynamoDB table for fast and predictable database performance at any scale. Learn more
Create table
Amazon DynamoDB Accelerator (DAX) is a fully-managed, highly-available, in-memory caching service for DynamoDB. Learn more
Create DAX cluster
What's new
SEP 19
AWS Cost Management now provides purchase recommendations for Amazon DynamoDB...


DynamoDB
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New
DynamoDB > Tables
Tables (0) Info
Actions
Delete
Create table
Find tables
Any tag key
Any tag value
Name
Status
Partition key
Sort key
Indexes
Deletion protection
Read capacity mode
Write capacity mode
Total size
You have no tables in this account in this AWS Region.
Create table

- **Activity 2.2:Create a DynamoDB table for storing Users details and Blood _request details.**
    - Create Users table with partition key “userId” with type String and click on create tables.


Creating the Users table. It will be available for use shortly.

Tables (1)

Last updated February 4, 2026, 19:06 (UTC+5:30)

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode	Write capac
Users	Creating	Userid (N)	-	0	0	Off	☆	On-demand	On-demanc

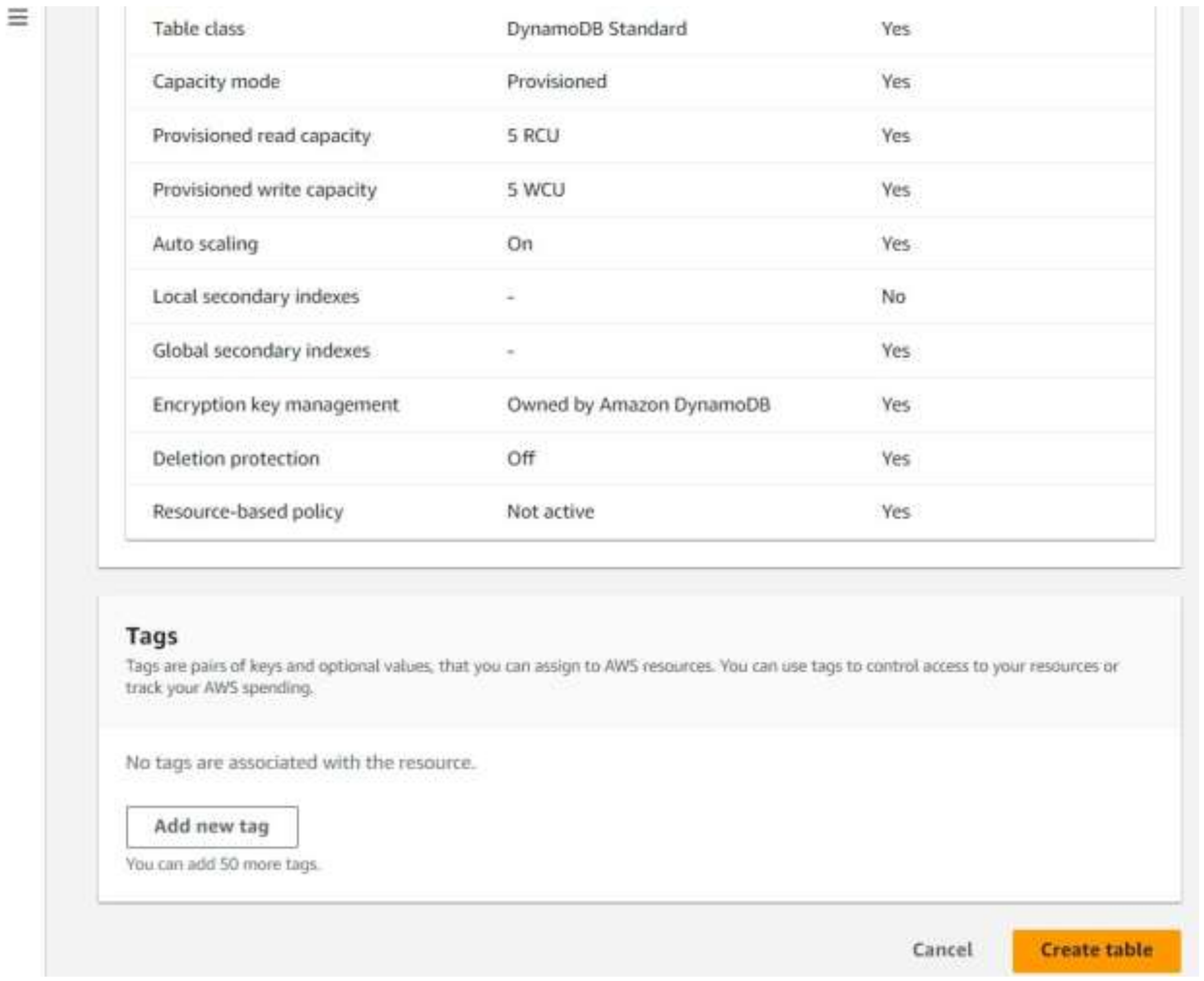


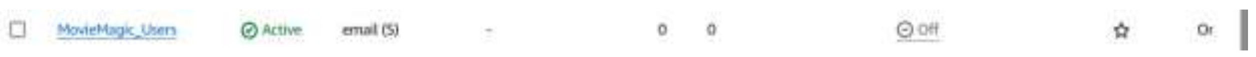
Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based 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.

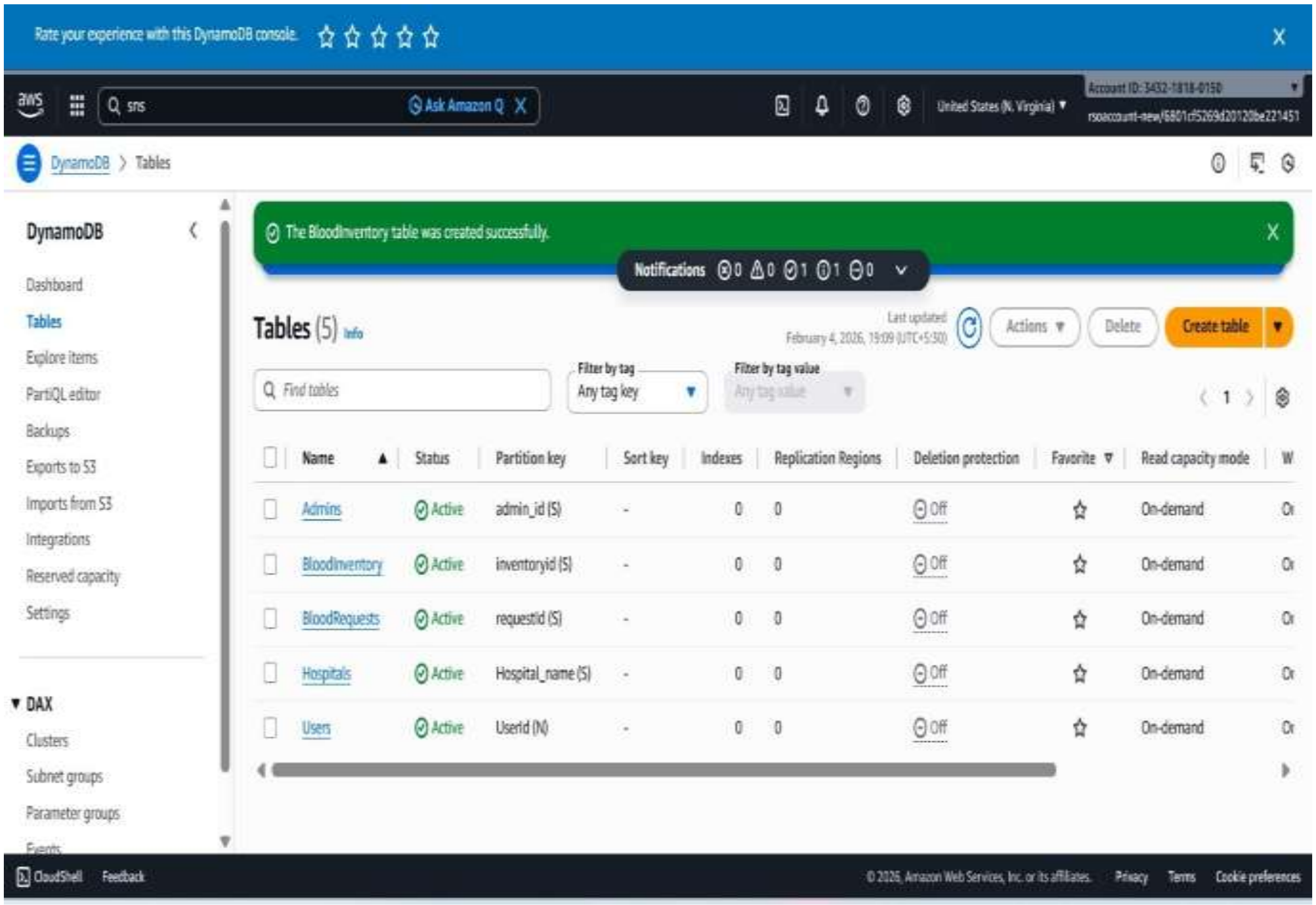
No tags are associated with the resource.

You can add 50 more tags.



☐	MovieMagic_Users	Active	email (S)	-	0	0	Off	☆	On

○ Follow the same steps to create a tables with the primary keys.



Rate your experience with this DynamoDB console.

DynamoDB > Tables

The BloodInventory table was created successfully.

Notifications 0 0 1 1 0

Tables (5) Info

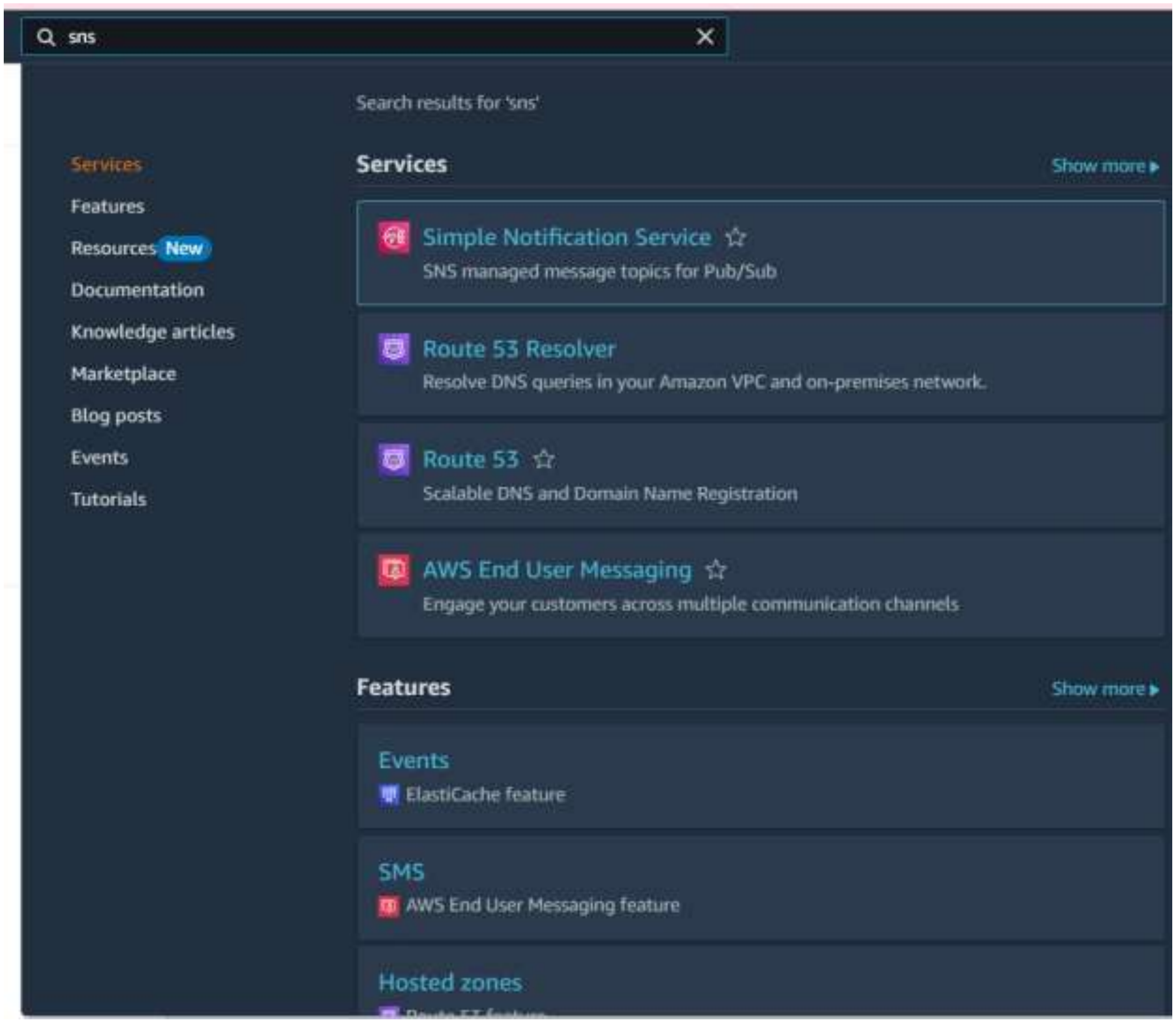
Last updated February 4, 2026, 19:09 (UTC+5:30)

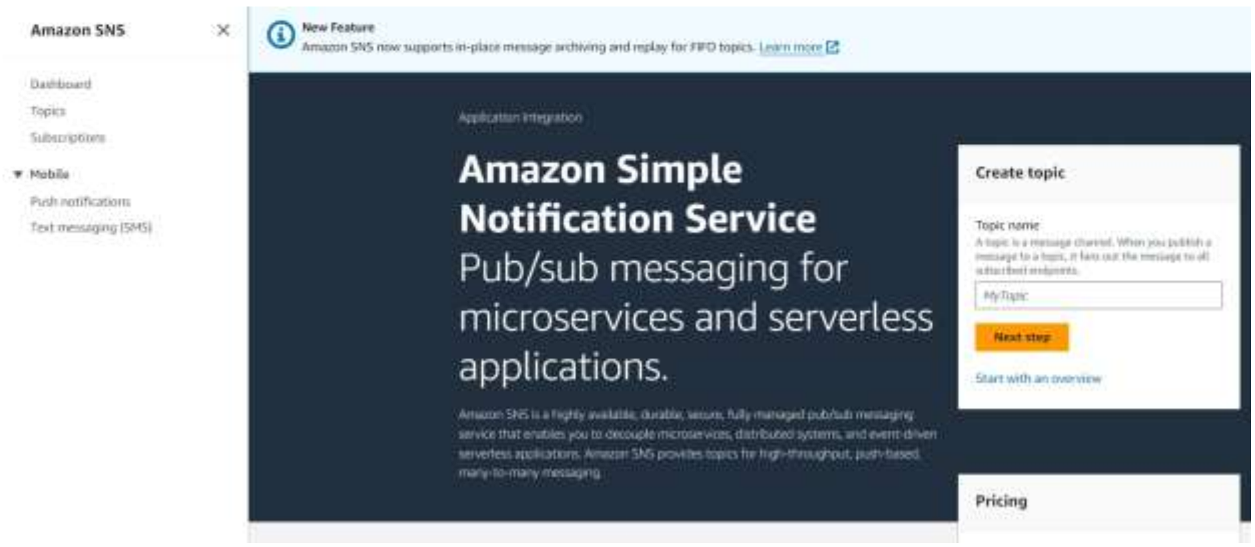
Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode
Admins	Active	admin_id (S)	-	0	0	Off	☆	On-demand
BloodInventory	Active	inventoryid (S)	-	0	0	Off	☆	On-demand
BloodRequests	Active	requestid (S)	-	0	0	Off	☆	On-demand
Hospitals	Active	Hospital_name (S)	-	0	0	Off	☆	On-demand
Users	Active	Userid (N)	-	0	0	Off	☆	On-demand

© 2026, Amazon Web Services, Inc. or its affiliates.

Milestone 3: SNS Notification Setup

- **Activity 3.1: Create SNS topics for sending email notifications to users regarding booking confirmation of their ticket.**
    - In the AWS Console, search for SNS and navigate to the SNS Dashboard.


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



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

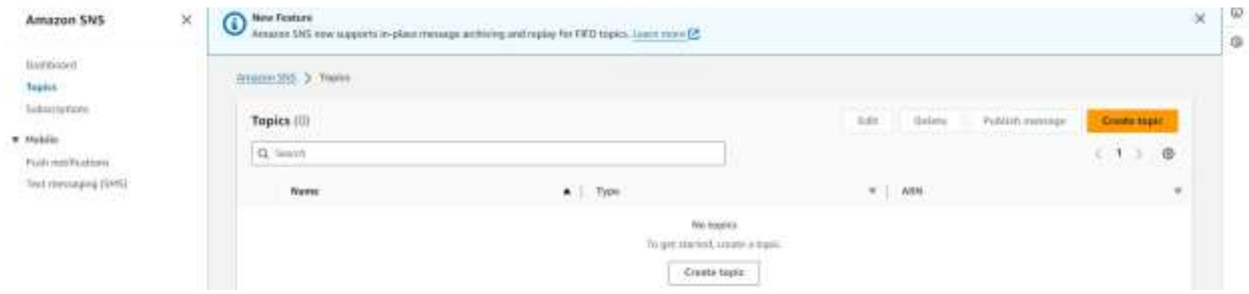
Application integration

Amazon Simple Notification Service
Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

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

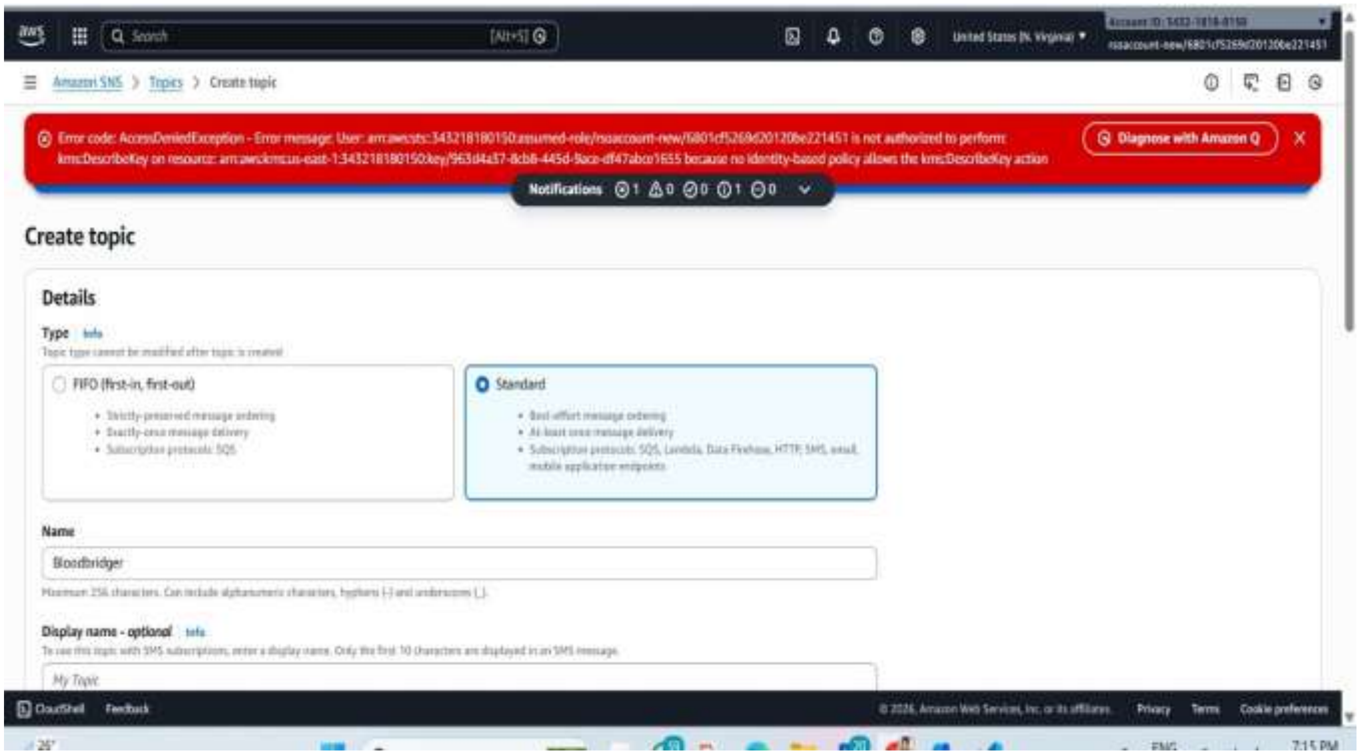
○ Click on **Create Topic** and choose a name for the topic.



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

No topics
To get started, create a topic.

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



Amazon SNS > Topics > Create topic

Error code: AccessDeniedException - Error message: User: arn:aws:sts::343218180150:assumed-role/rsoaccount-new/6801cf5269d20120be221451 is not authorized to perform: kms:DescribeKey on resource: arn:aws:kms:us-east-1:343218180150:key/963d4a37-8cb8-445d-9ace-df47abce1655 because no identity-based policy allows the kms:DescribeKey action

Create topic

Details

Type Info

Topic type cannot be modified after topic is created

( ) FIFO (first-in, first-out)
- Strictly-preserved message ordering
- Exactly-once message delivery
- Subscription protocols: SQS

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

Name

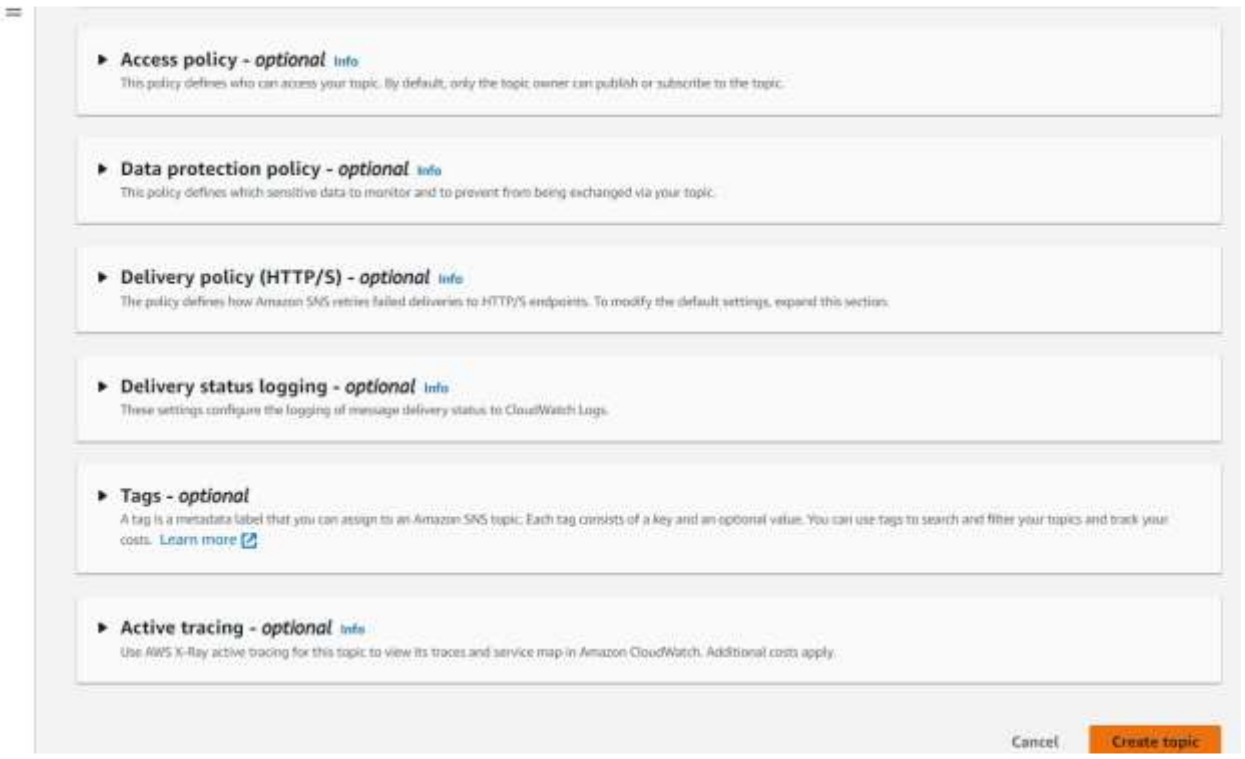
Bloodbridger

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



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

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

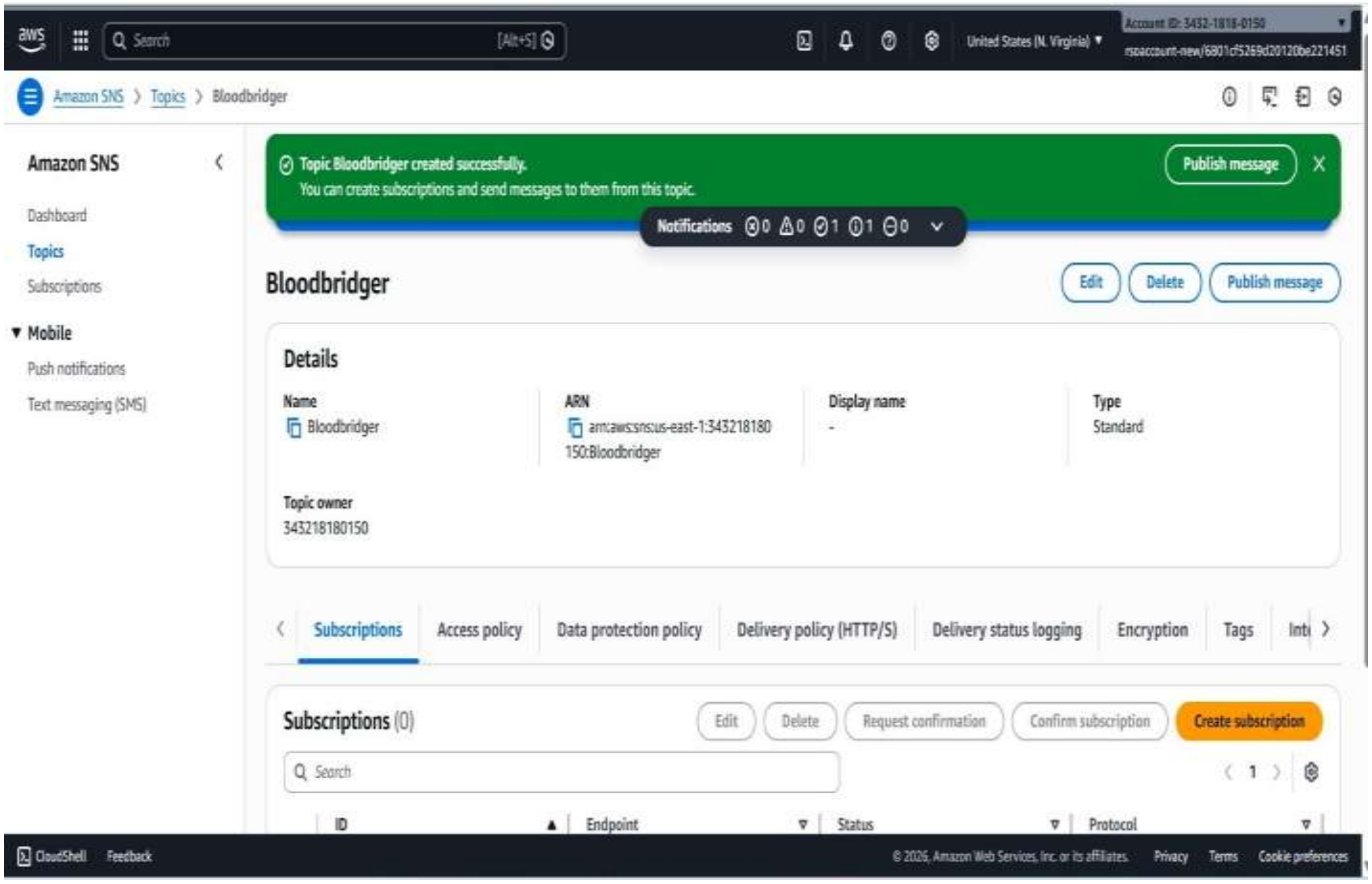
▸ Delivery policy (HTTP/S) - *optional* Info
The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section.

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

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

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

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


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

Bloodbridger

Details

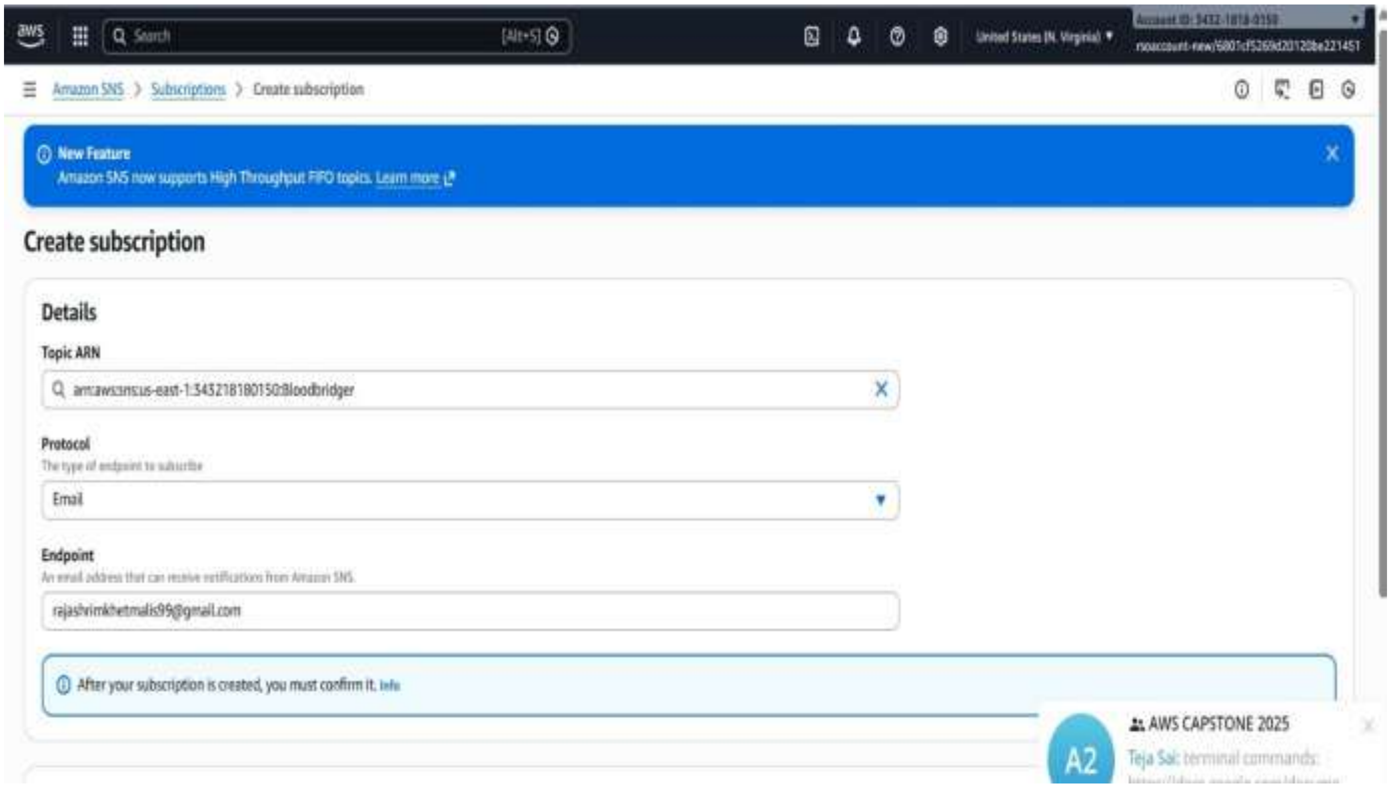
Name	ARN	Display name	Type
Bloodbridger	arn:aws:sns:us-east-1:343218180150:Bloodbridger	-	Standard

Topic owner
343218180150

Subscriptions (0)

ID	Endpoint	Status	Protocol

- **Activity 3.2: Subscribe users and staff to relevant SNS topics to receive real-time notifications when a blood request is made.**
    - Subscribe users (or admin staff) to this topic via Email. When a users logged in , notifications will be sent to the user's emails**.**



Amazon SNS > Subscriptions > Create subscription

New Feature
Amazon SNS now supports High Throughput FIFO topics. Learn more

Create subscription

Details

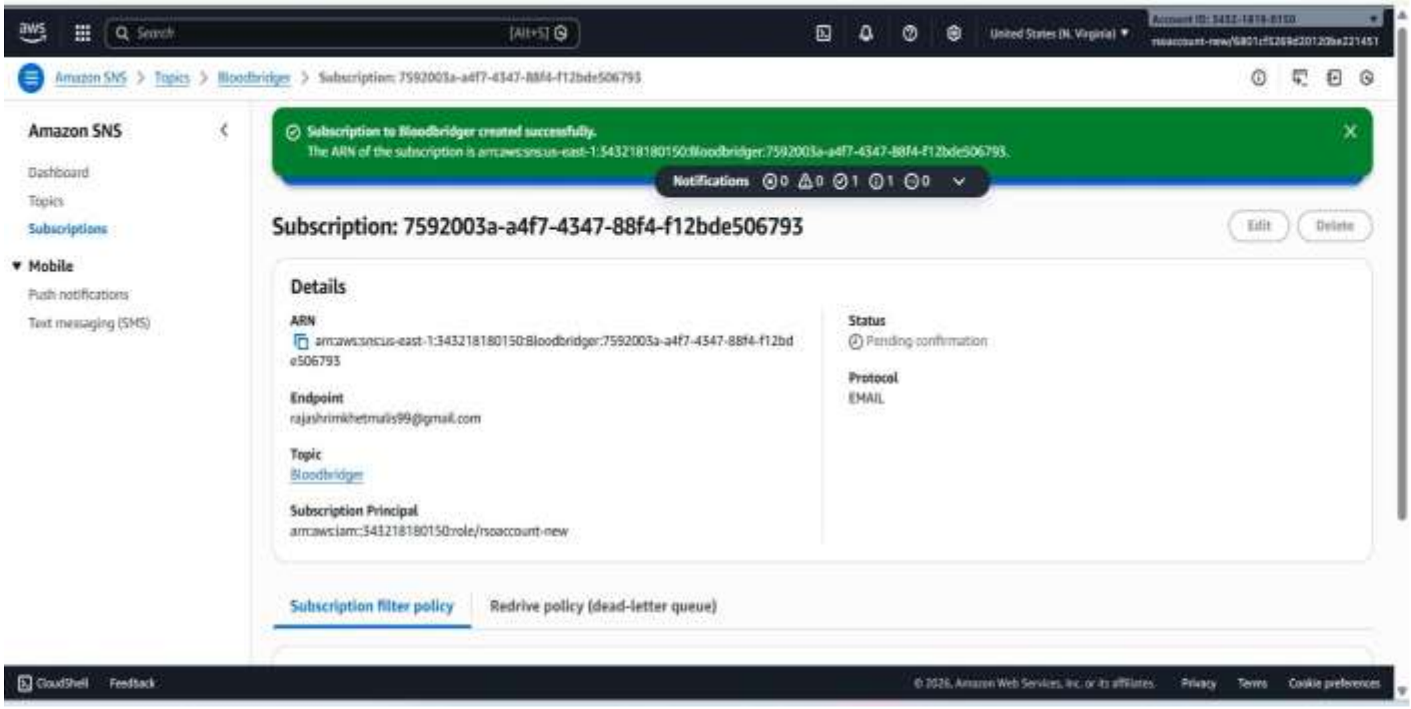
Topic ARN
arn:aws:sns:us-east-1:543218180150:Bloodbridger

Protocol
The type of endpoint to subscribe
Email

Endpoint
An email address that can receive notifications from Amazon SNS.
rajashrimkhetmalis99@gmail.com

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

AWS CAPSTONE 2025
Teja Sai: terminal commands:


Subscription to Bloodbridger created successfully.
The ARN of the subscription is arn:aws:sns:us-east-1:343218180150:Bloodbridger:7592003a-a4f7-4347-88f4-f12bde506793.

Subscription: 7592003a-a4f7-4347-88f4-f12bde506793

Details

ARN
arn:aws:sns:us-east-1:343218180150:Bloodbridger:7592003a-a4f7-4347-88f4-f12bde506793

Endpoint
rajashrimkhetmalis99@gmail.com

Topic
Bloodbridger

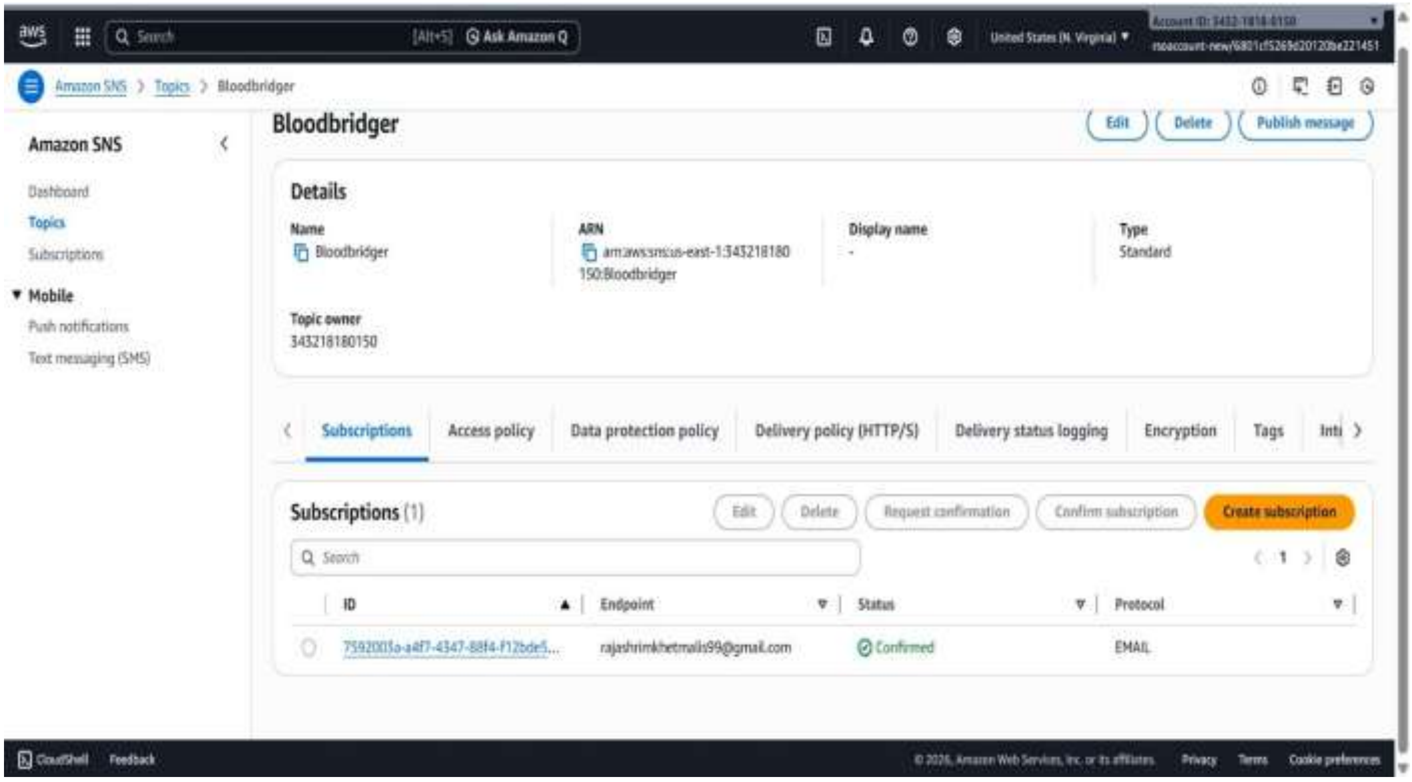
Subscription Principal
arn:aws:iam::343218180150:role/rsoaccount-new

Status
Pending confirmation

Protocol
EMAIL

Amazon SNS > Topics > Bloodbridger > Subscription: 7592003a-a4f7-4347-88f4-f12bde506793
Amazon SNS
Dashboard
Topics
Subscriptions
Mobile
Push notifications
Text messaging (SMS)
Notifications
Edit
Delete
Subscription filter policy
Redrive policy (dead-letter queue)
CloudShell
Feedback

○ After subscription request for the mail confirmation


Bloodbridger

Details

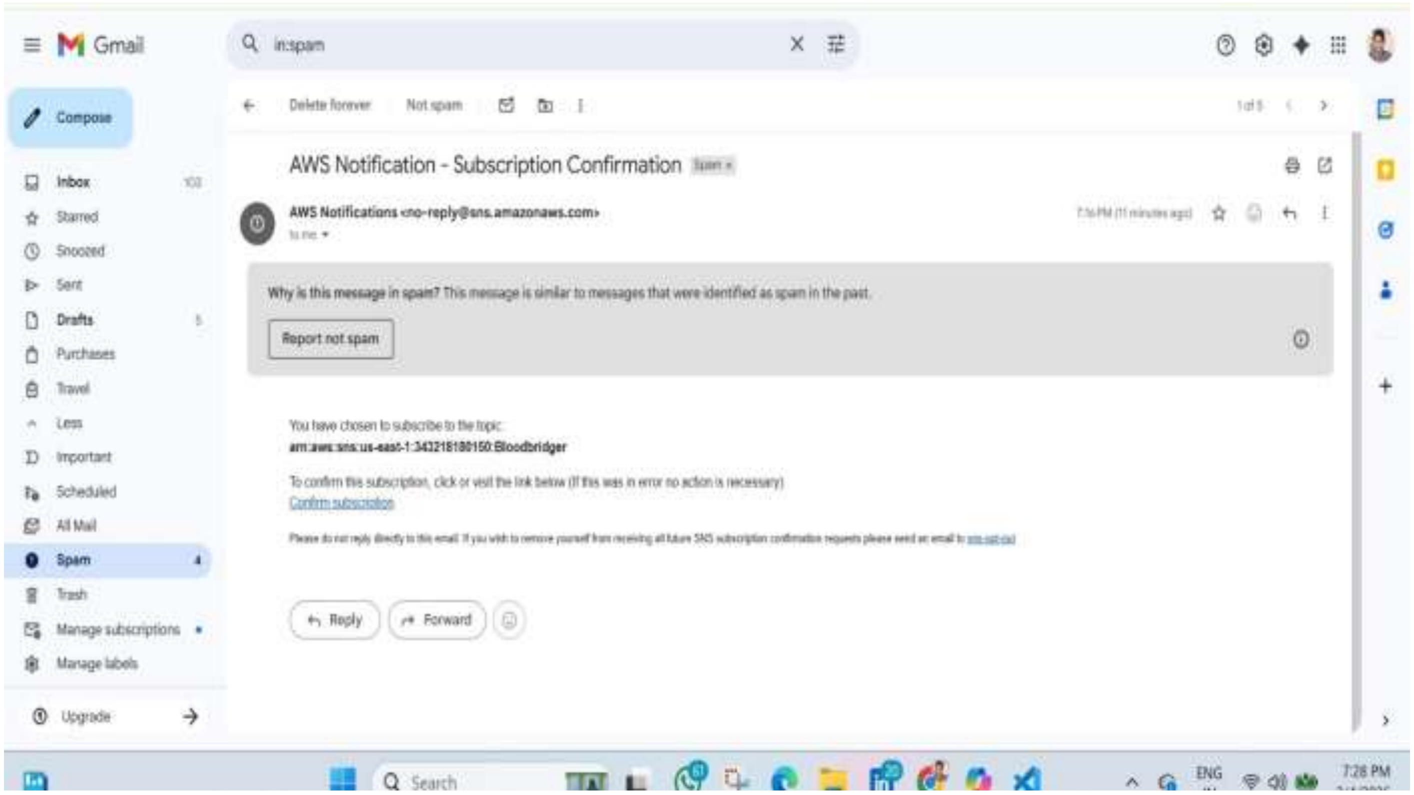
Name	ARN	Display name	Type
Bloodbridger	arn:aws:sns:us-east-1:343218180150:Bloodbridger	-	Standard

Topic owner
343218180150

Subscriptions (1)

ID	Endpoint	Status	Protocol
7592003a-a4f7-4347-88f4-f12bde5...	rajashrimkhetmalis99@gmail.com	Confirmed	EMAIL

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



AWS Notification - Subscription Confirmation

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

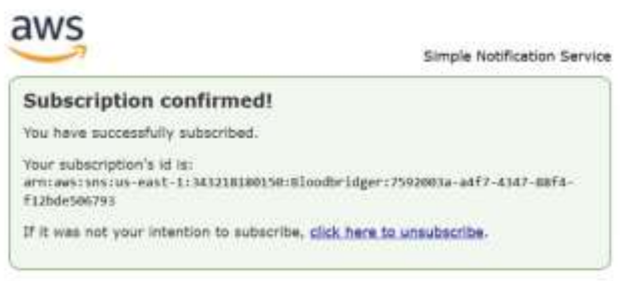
to me

Why is this message in spam? This message is similar to messages that were identified as spam in the past.

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:343218180150:Bloodbridger

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.

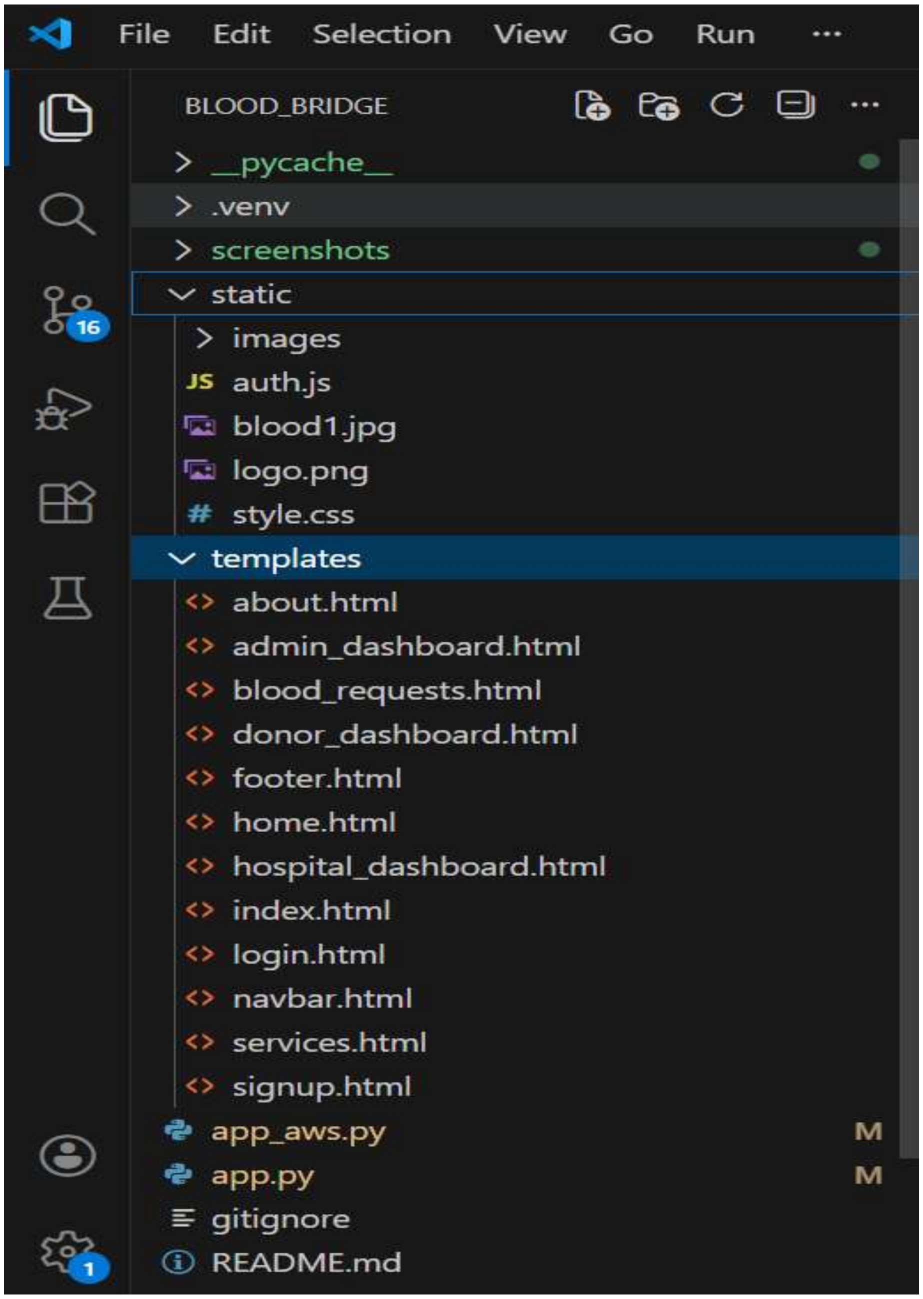
Your subscription's id is:
arn:aws:sns:us-east-1:343218180150:Bloodbridger:7592003a-a4f7-4347-88f4-f12bde506793

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

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

Milestone 4:Backend Development and Application Setup

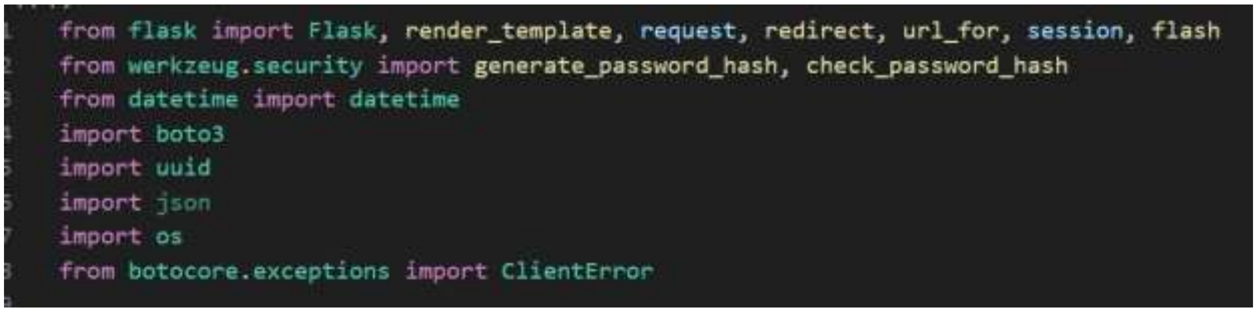
- **Activity 4.1: Develop the backend using Flask**
    - File Explorer Structure


```
BLOOD_BRIDGE
> __pycache__
> .venv
> screenshots
v static
  > images
  auth.js
  blood1.jpg
  logo.png
  style.css
v templates
  about.html
  admin_dashboard.html
  blood_requests.html
  donor_dashboard.html
  footer.html
  home.html
  hospital_dashboard.html
  index.html
  login.html
  navbar.html
  services.html
  signup.html
app_aws.py        M
app.py            M
gitignore
README.md
```

Description of the code :

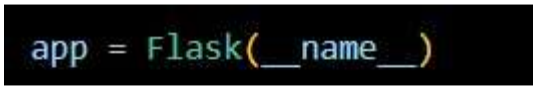
- **Flask App Initialization**

Imports and Configuration:



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

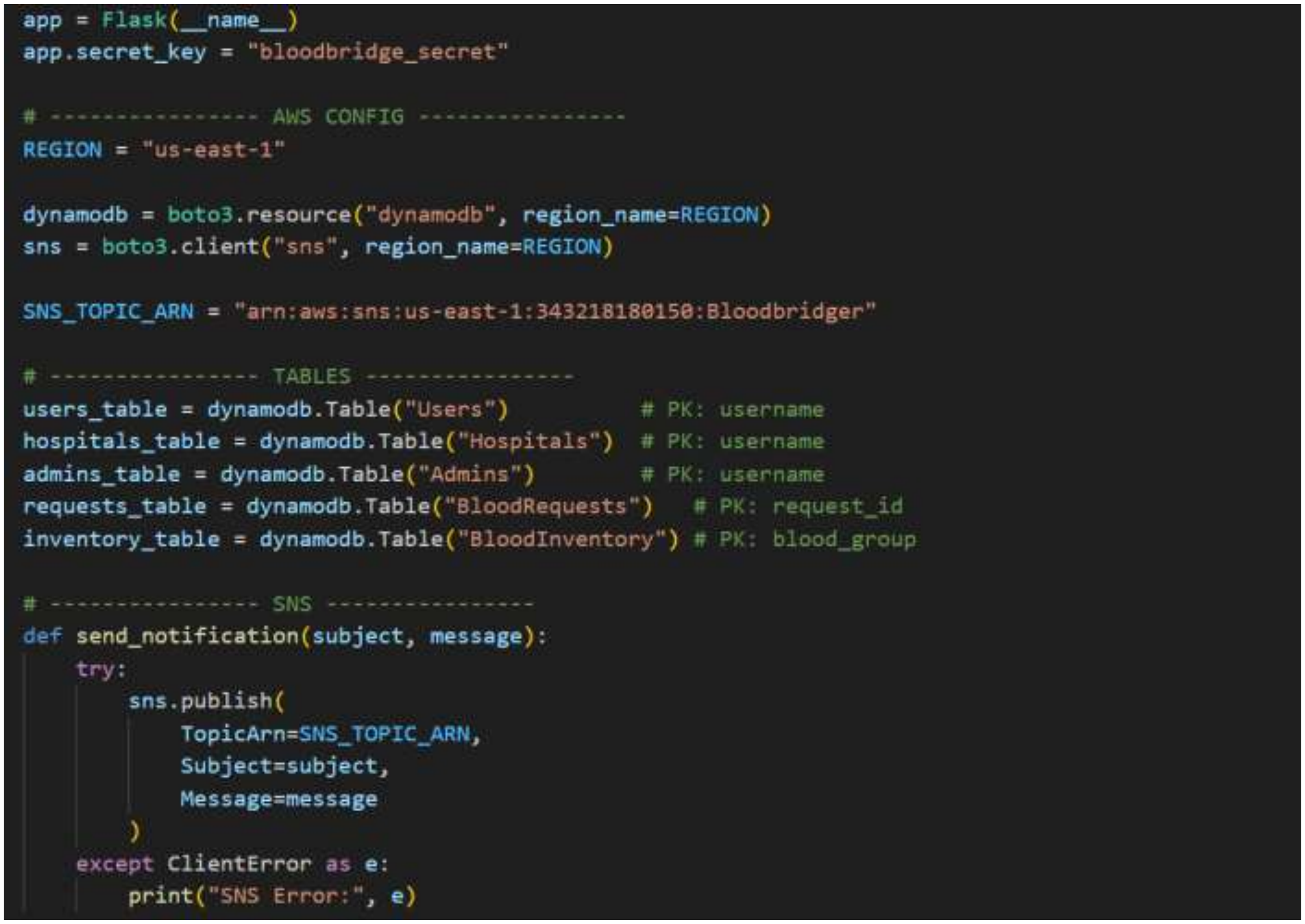
Description:This project uses Flask for routing, session management, and user authentication with secure password hashing. It integrates AWS services via Boto3 for handling data storage, notifications, and unique user operations.



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

Description:A new Flask application instance is initialized, and a secret key is set to securely manage user sessions and protect against cookie tampering.

- **Dynamodb and SNS Setup:**



```
app = Flask(__name__)
app.secret_key = "bloodbridge_secret"

# --------------- AWS CONFIG ---------------
REGION = "us-east-1"

dynamodb = boto3.resource("dynamodb", region_name=REGION)
sns = boto3.client("sns", region_name=REGION)

SNS_TOPIC_ARN = "arn:aws:sns:us-east-1:343218180150:Bloodbridger"

# --------------- TABLES ---------------
users_table = dynamodb.Table("Users")          # PK: username
hospitals_table = dynamodb.Table("Hospitals")  # PK: username
admins_table = dynamodb.Table("Admins")        # PK: username
requests_table = dynamodb.Table("BloodRequests")   # PK: request_id
inventory_table = dynamodb.Table("BloodInventory") # PK: blood_group

# --------------- SNS ---------------
def send_notification(subject, message):
    try:
        sns.publish(
            TopicArn=SNS_TOPIC_ARN,
            Subject=subject,
            Message=message
        )
    except ClientError as e:
        print("SNS Error:", e)
```

The Blood Bridge application uses boto3 to connect with Amazon DynamoDB for managing user registrations (donors, hospitals, admins), blood requests, and real-time blood inventory. DynamoDB tables are created in the us-east-1 region. The application performs CRUD operations such as storing user details, creating blood requests, updating request status, and maintaining blood stock availabilit

- **SNS Connection**
    - **Description:**Configure **SNS** to send notifications when a movie ticket is booked. 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.

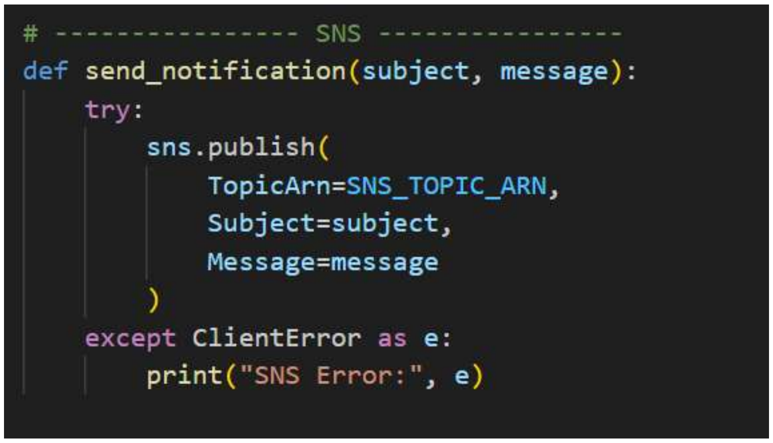
- **Function to send the Notifications:**

Description:

This function is responsible for sending real-time notifications using **AWS Simple Notification Service (SNS)**. Whenever a critical event occurs in the system—such as **new user registration, blood request creation, donor acceptance, or inventory update**—the function formats the relevant details into a message and publishes it to a predefined SNS topic.

The notification is then delivered to subscribed users (via email), ensuring timely communication between donors, hospitals, and administrators. This feature enhances emergency responsiveness and improves coordination in life-saving blood donation processes.

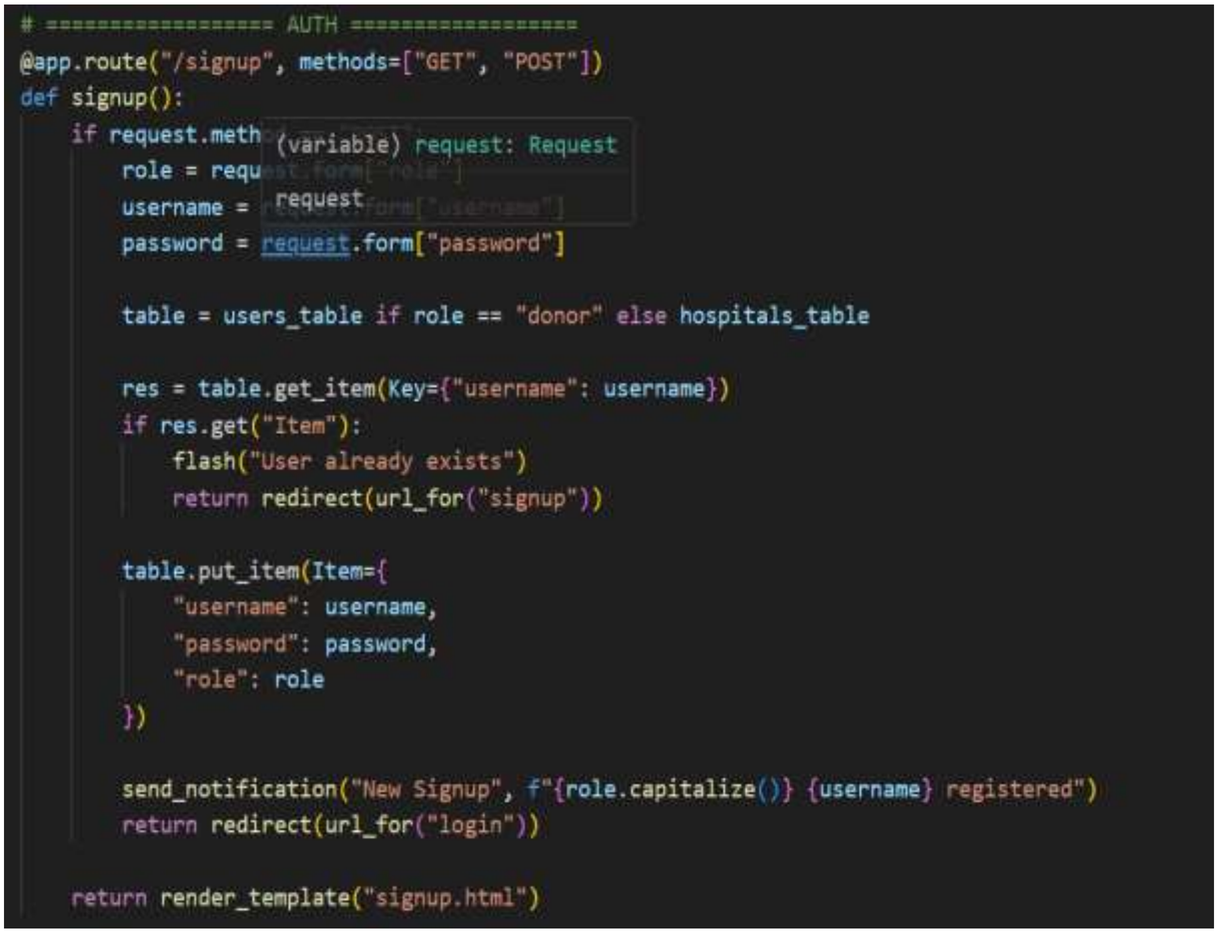
.



```python
# ---------------- SNS ----------------
def send_notification(subject, message):
    try:
        sns.publish(
            TopicArn=SNS_TOPIC_ARN,
            Subject=subject,
            Message=message
        )
    except ClientError as e:
        print("SNS Error:", e)
```

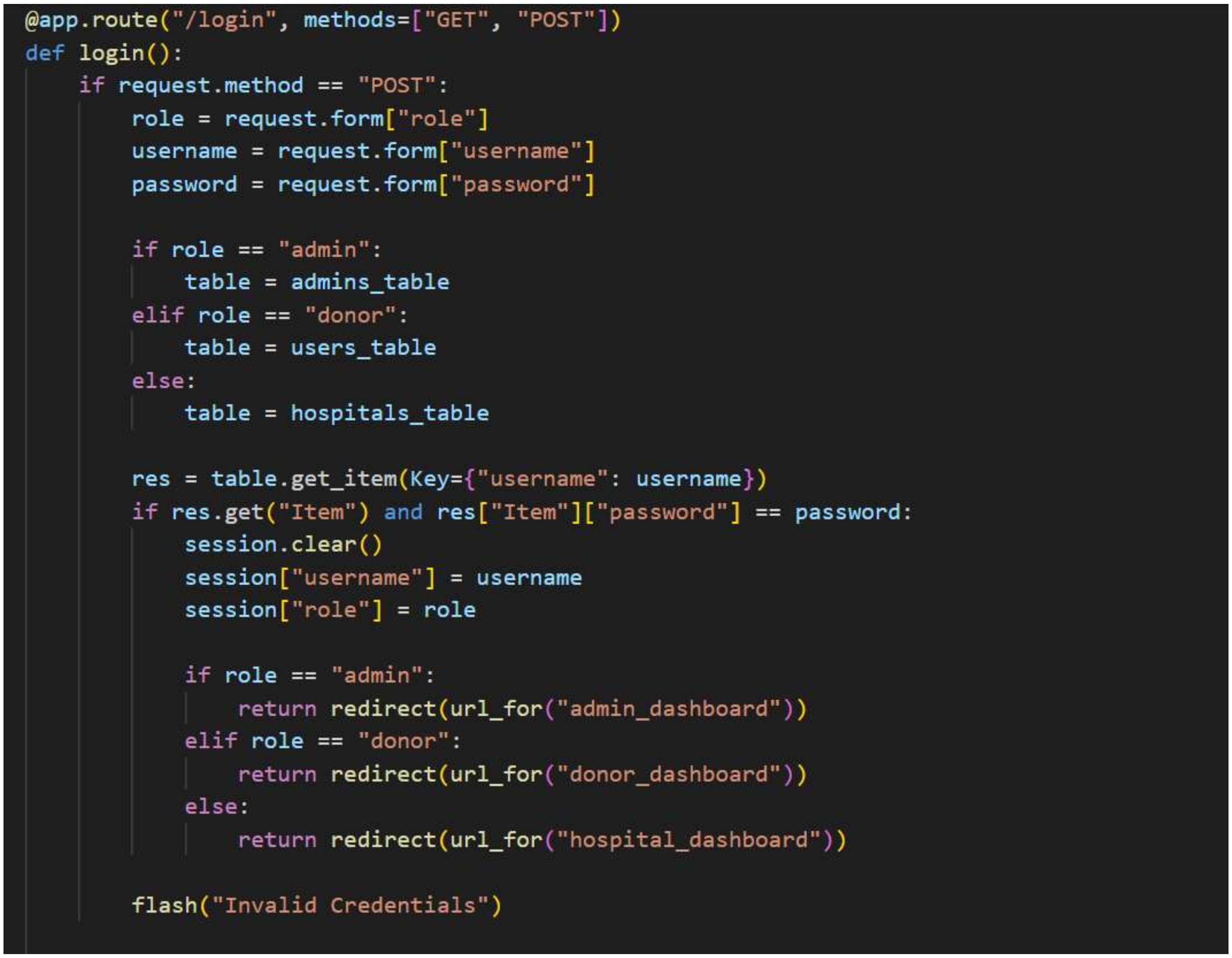
- **Routes for Web Pages**

- **Register User:** Collecting registration data, hashes the password, and stores user details in the database.



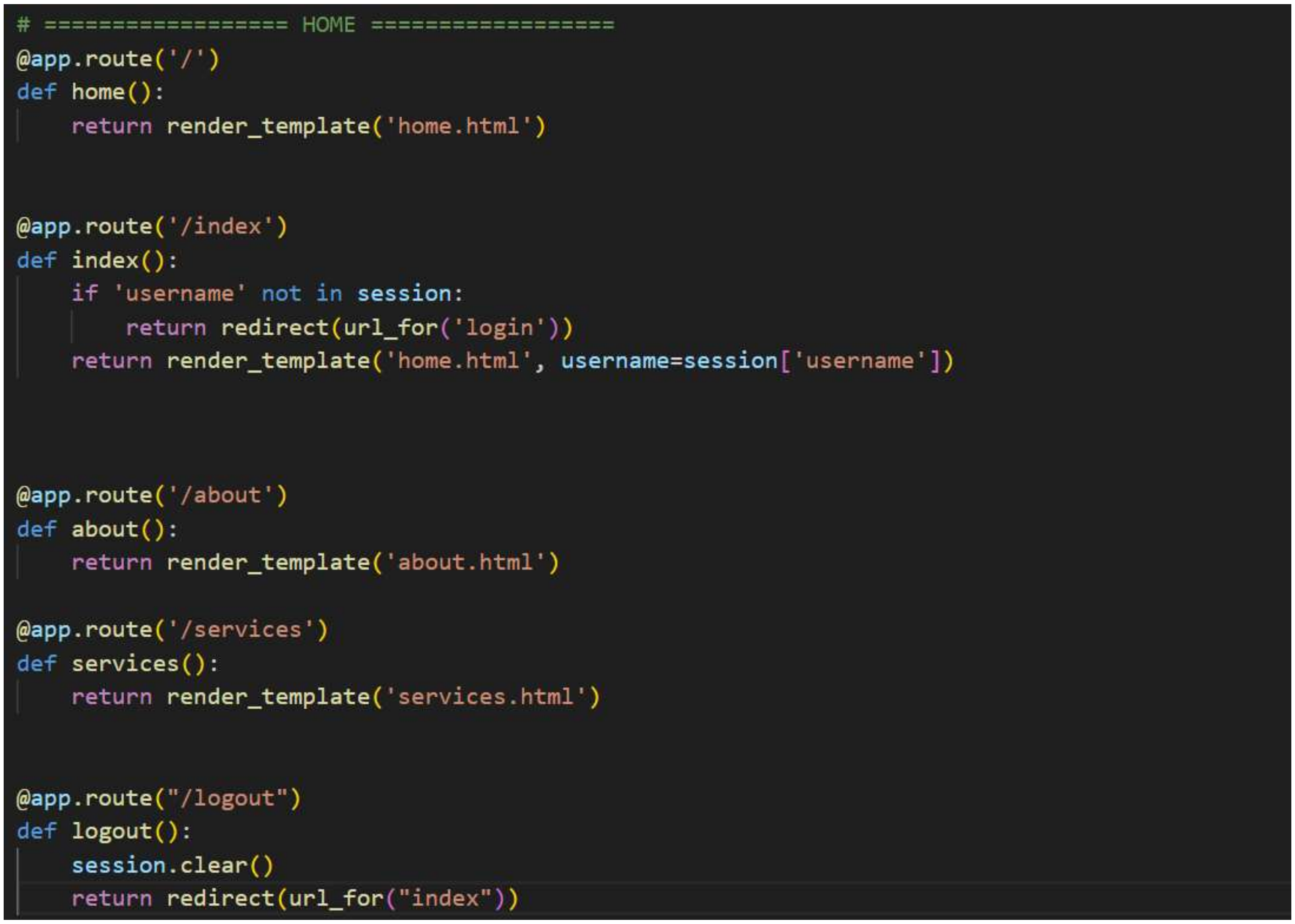
```python
# ================== AUTH ===================
@app.route("/signup", methods=["GET", "POST"])
def signup():
    if request.method == "POST":
        role = request.form["role"]
        username = request.form["username"]
        password = request.form["password"]

        table = users_table if role == "donor" else hospitals_table

        res = table.get_item(Key={"username": username})
        if res.get("Item"):
            flash("User already exists")
            return redirect(url_for("signup"))

        table.put_item(Item={
            "username": username,
            "password": password,
            "role": role
        })

        send_notification("New Signup", f"{role.capitalize()} {username} registered")
        return redirect(url_for("login"))

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

- **login Route (GET/POST)**: Verifies user credentials, increments login count, and redirects to the dashboard on success.



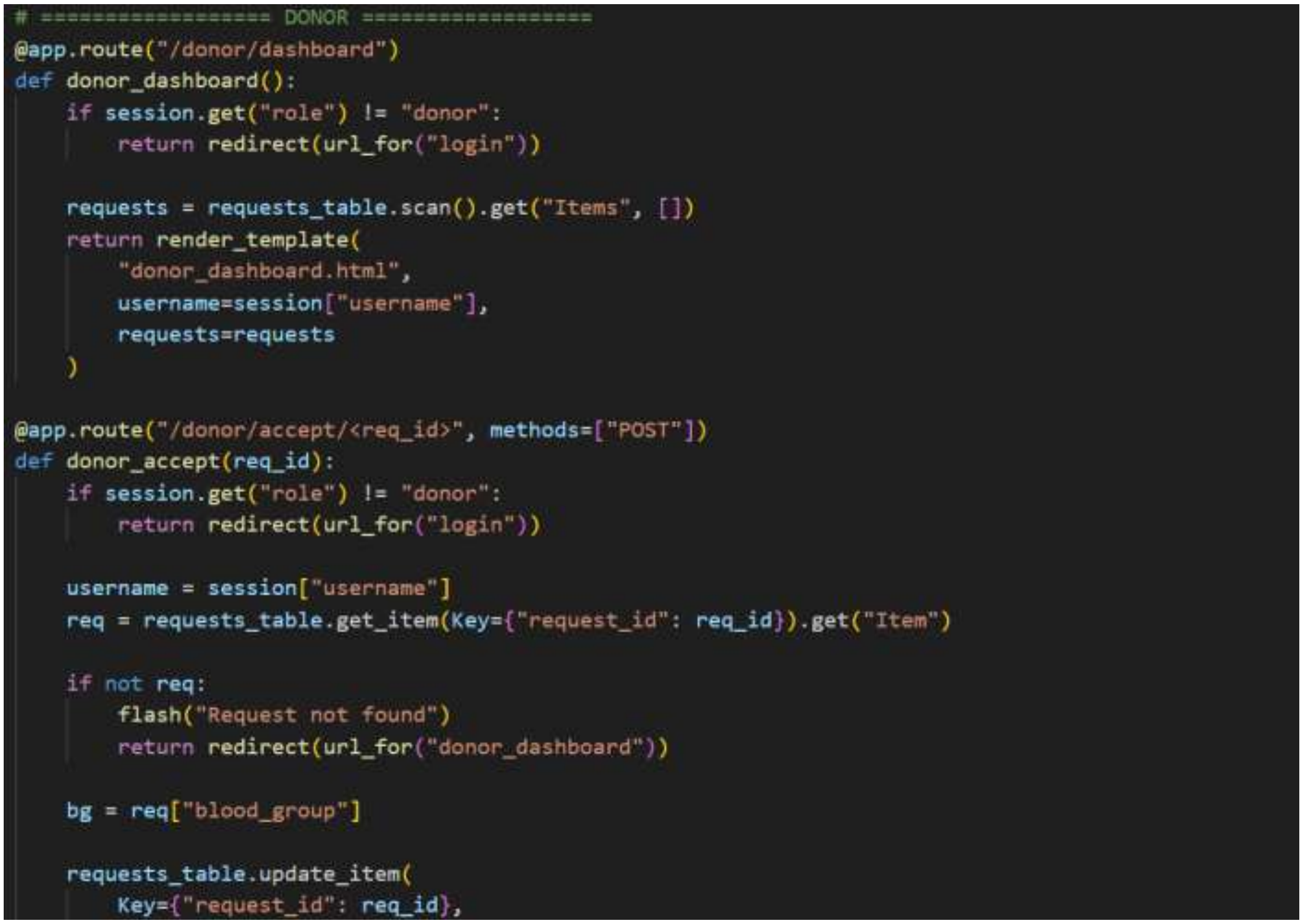
```python
@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        role = request.form["role"]
        username = request.form["username"]
        password = request.form["password"]

        if role == "admin":
            table = admins_table
        elif role == "donor":
            table = users_table
        else:
            table = hospitals_table

        res = table.get_item(Key={"username": username})
        if res.get("Item") and res["Item"]["password"] == password:
            session.clear()
            session["username"] = username
            session["role"] = role

            if role == "admin":
                return redirect(url_for("admin_dashboard"))
            elif role == "donor":
                return redirect(url_for("donor_dashboard"))
            else:
                return redirect(url_for("hospital_dashboard"))

        flash("Invalid Credentials")
```

- These Flask routes handle key navigation in the app: /logout logs out the user by clearing the session and showing a flash message; /home1 is a protected route accessible only to logged-in users; /about , /services and /index render static pages with information about the app and ways to get in touch.

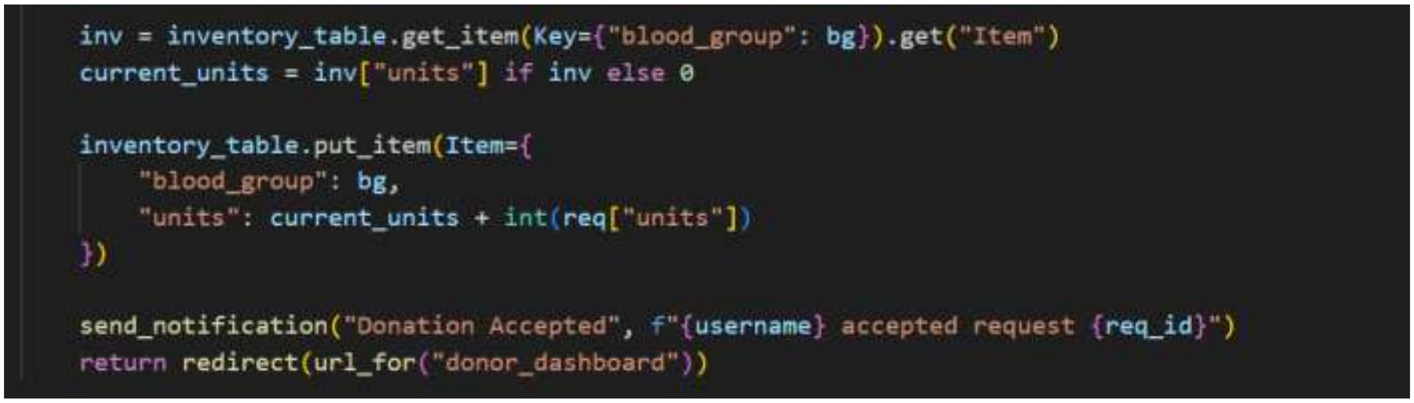


```
# ================== HOME ===================
@app.route('/')
def home():
    return render_template('home.html')


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



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

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


@app.route("/logout")
def logout():
    session.clear()
    return redirect(url_for("index"))
```

- **Donor Dashboard Page Route:**
  - View blood requests
  - Accept donation requests
  - Update inventory automatically



```python
# =================== DONOR ===================
@app.route("/donor/dashboard")
def donor_dashboard():
    if session.get("role") != "donor":
        return redirect(url_for("login"))

    requests = requests_table.scan().get("Items", [])
    return render_template(
        "donor_dashboard.html",
        username=session["username"],
        requests=requests
    )

@app.route("/donor/accept/<req_id>", methods=["POST"])
def donor_accept(req_id):
    if session.get("role") != "donor":
        return redirect(url_for("login"))

    username = session["username"]
    req = requests_table.get_item(Key={"request_id": req_id}).get("Item")

    if not req:
        flash("Request not found")
        return redirect(url_for("donor_dashboard"))

    bg = req["blood_group"]

    requests_table.update_item(
        Key={"request_id": req_id},
```

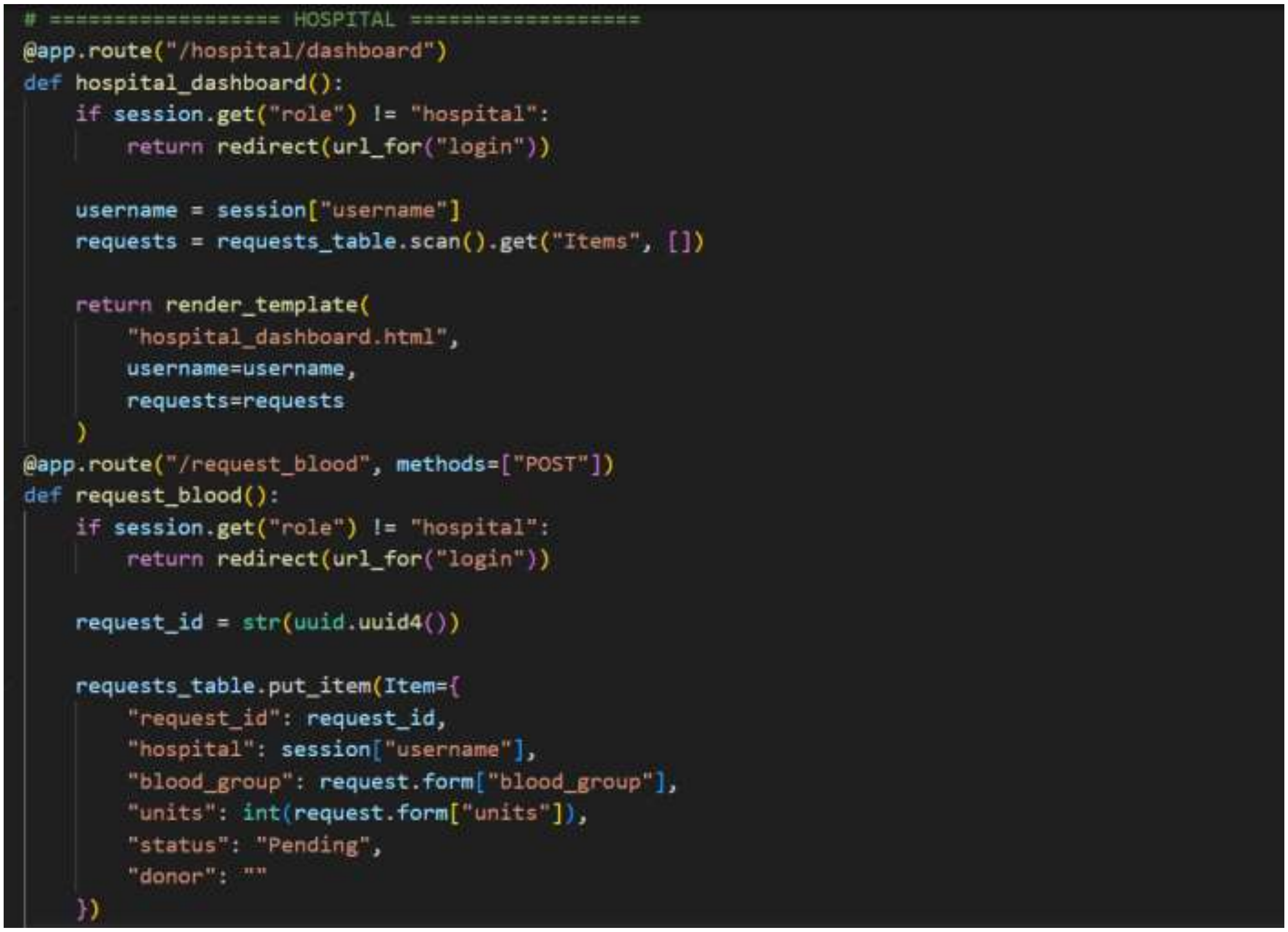


```
inv = inventory_table.get_item(Key={"blood_group": bg}).get("Item")
current_units = inv["units"] if inv else 0

inventory_table.put_item(Item={
    "blood_group": bg,
    "units": current_units + int(req["units"])
})

send_notification("Donation Accepted", f"{username} accepted request {req_id}")
return redirect(url_for("donor_dashboard"))
```

Hospital Dashboard Page Route:

Description:

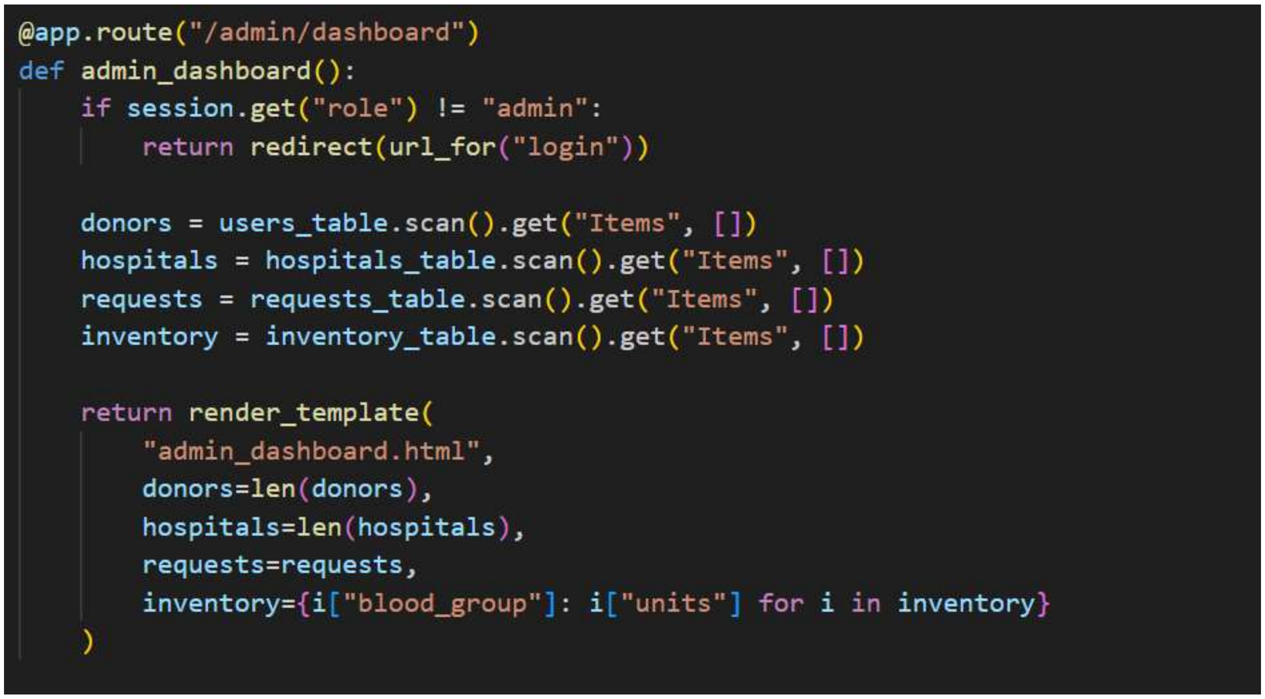
- Create blood requests
- Track request status
- Monitor inventory levels



```
# ================== HOSPITAL ==================
@app.route("/hospital/dashboard")
def hospital_dashboard():
    if session.get("role") != "hospital":
        return redirect(url_for("login"))

    username = session["username"]
    requests = requests_table.scan().get("Items", [])

    return render_template(
        "hospital_dashboard.html",
        username=username,
        requests=requests
    )
@app.route("/request_blood", methods=["POST"])
def request_blood():
    if session.get("role") != "hospital":
        return redirect(url_for("login"))

    request_id = str(uuid.uuid4())

    requests_table.put_item(Item={
        "request_id": request_id,
        "hospital": session["username"],
        "blood_group": request.form["blood_group"],
        "units": int(request.form["units"]),
        "status": "Pending",
        "donor": ""
    })
```

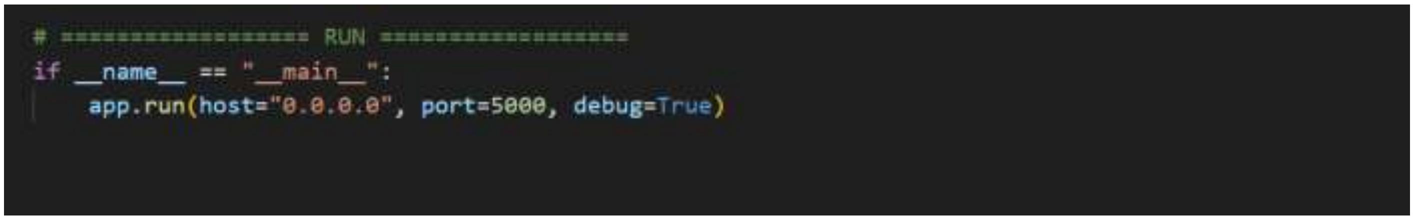
Admin Dashboarad Route page:

- View system analytics
- Monitor donors & hospitals
- Track accepted and pending requests
- Manage blood stock data



```python
@app.route("/admin/dashboard")
def admin_dashboard():
    if session.get("role") != "admin":
        return redirect(url_for("login"))

    donors = users_table.scan().get("Items", [])
    hospitals = hospitals_table.scan().get("Items", [])
    requests = requests_table.scan().get("Items", [])
    inventory = inventory_table.scan().get("Items", [])

    return render_template(
        "admin_dashboard.html",
        donors=len(donors),
        hospitals=len(hospitals),
        requests=requests,
        inventory={i["blood_group"]: i["units"] for i in inventory}
    )
```

Application Entry point:



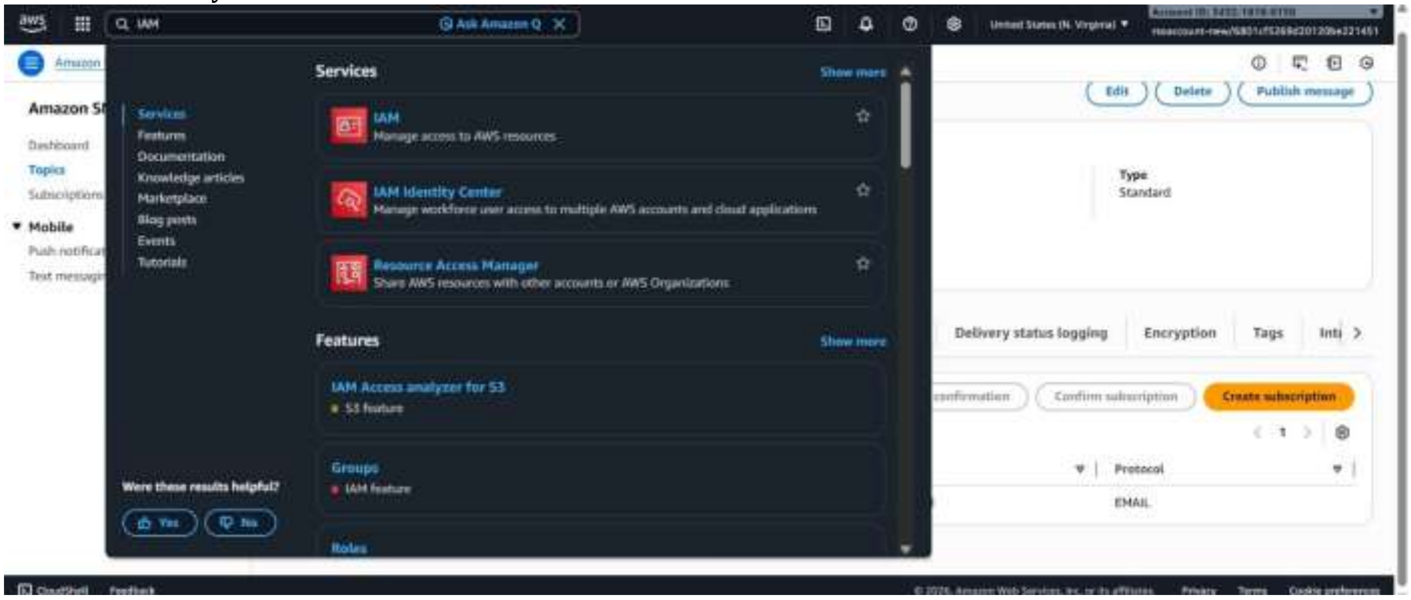
```python
# ================== RUN ==================
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)
```

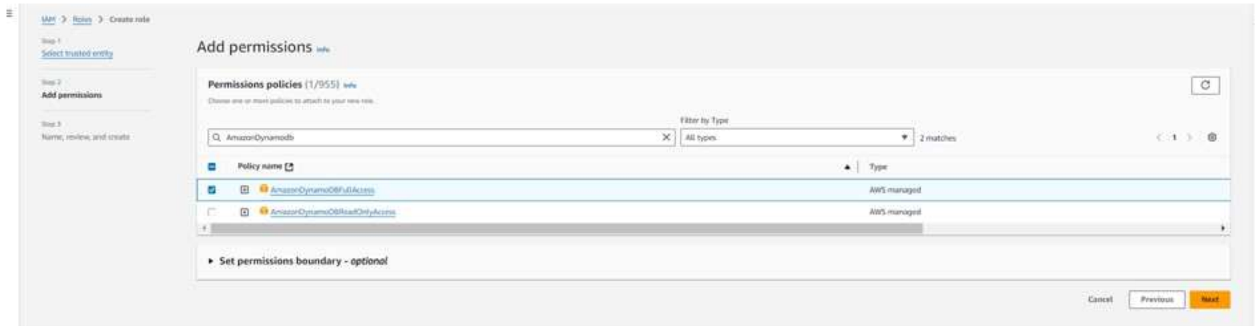
- **Description**: This block starts the Flask application using the built-in development server, setting the host, port, and enabling debug mode for easier development and testing.

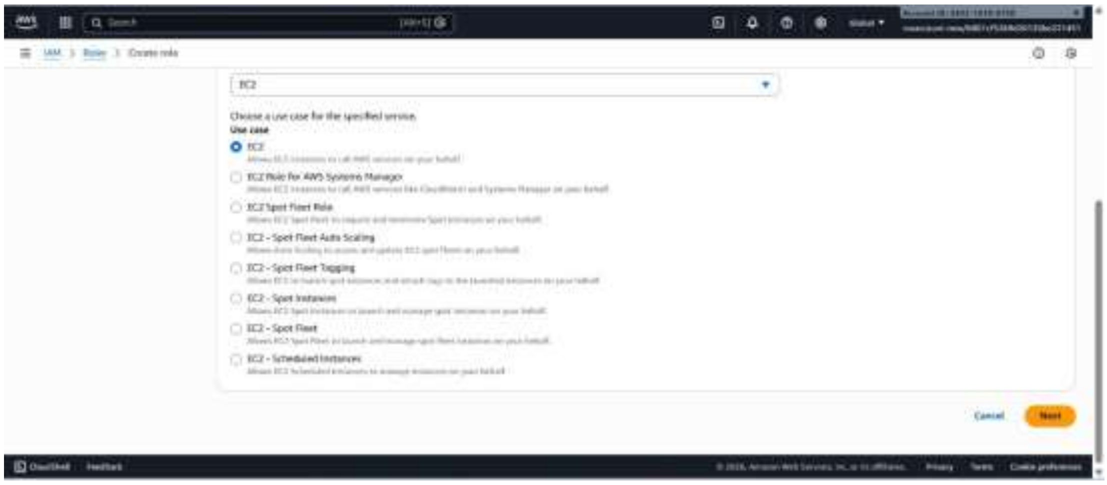
Milestone 5: IAM Role Setup

- **Activity 5.1:Create IAM Role.**

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






IAM > Roles > Create role

EC2

Choose a use case for the specified service.

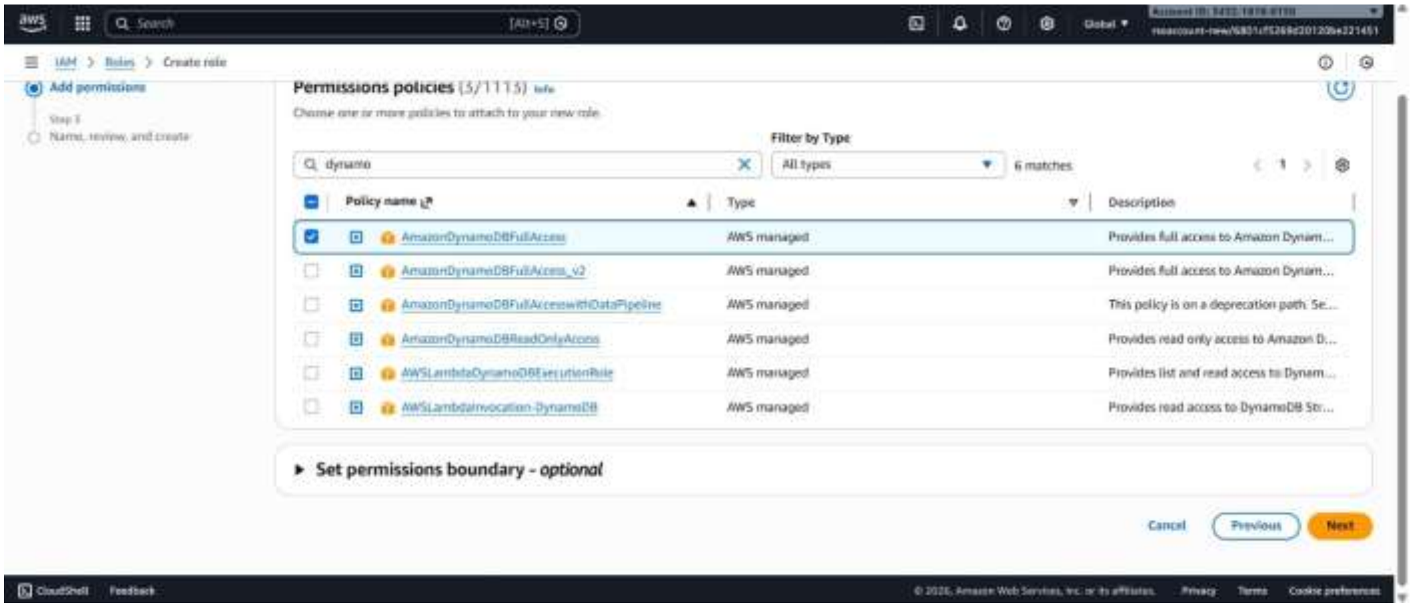
Use case

- (●) EC2 — Allows EC2 instances to call AWS services on your behalf.
- ( ) EC2 Role for AWS Systems Manager — Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- ( ) EC2 Spot Fleet Role — Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.
- ( ) EC2 - Spot Fleet Auto Scaling — Allows Auto Scaling to access and update EC2 spot fleets on your behalf.
- ( ) EC2 - Spot Fleet Tagging — Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.
- ( ) EC2 - Spot Instances — Allows EC2 Spot Instances to launch and manage spot instances on your behalf.
- ( ) EC2 - Spot Fleet — Allows EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.
- ( ) EC2 - Scheduled Instances — Allows EC2 Scheduled Instances to manage instances on your behalf.

- **Activity 5.2: Attach Policies.**

  Attach the following policies to the role:

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

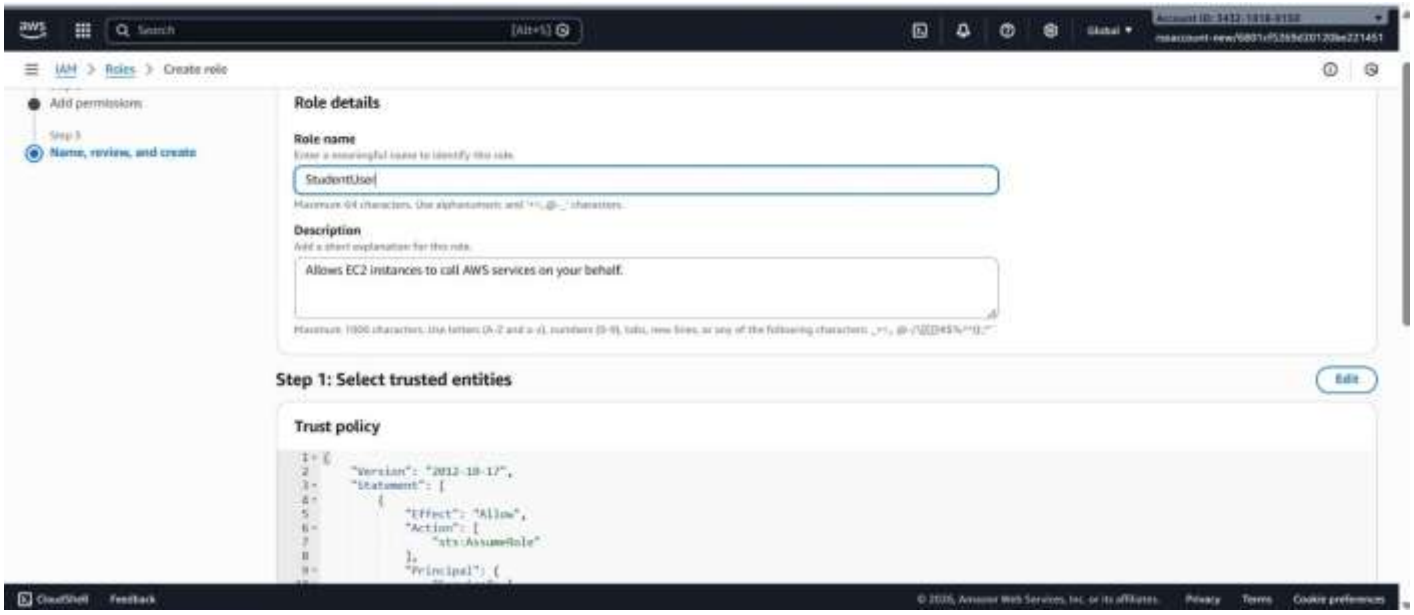

IAM > Roles > Create role

Permissions policies (3/1113)

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

Search: dynamo — Filter by Type: All types — 6 matches

	Policy name	Type	Description
✓	AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon Dynam...
	AmazonDynamoDBFullAccess_v2	AWS managed	Provides full access to Amazon Dynam...
	AmazonDynamoDBFullAccesswithDataPipeline	AWS managed	This policy is on a deprecation path. Se...
	AmazonDynamoDBReadOnlyAccess	AWS managed	Provides read only access to Amazon D...
	AWSLambdaDynamoDBExecutionRole	AWS managed	Provides list and read access to Dynam...
	AWSLambdaInvocation-DynamoDB	AWS managed	Provides read access to DynamoDB Str...



IAM > Roles > Create role

Add permissions

Step 3
Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
StudentUser
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

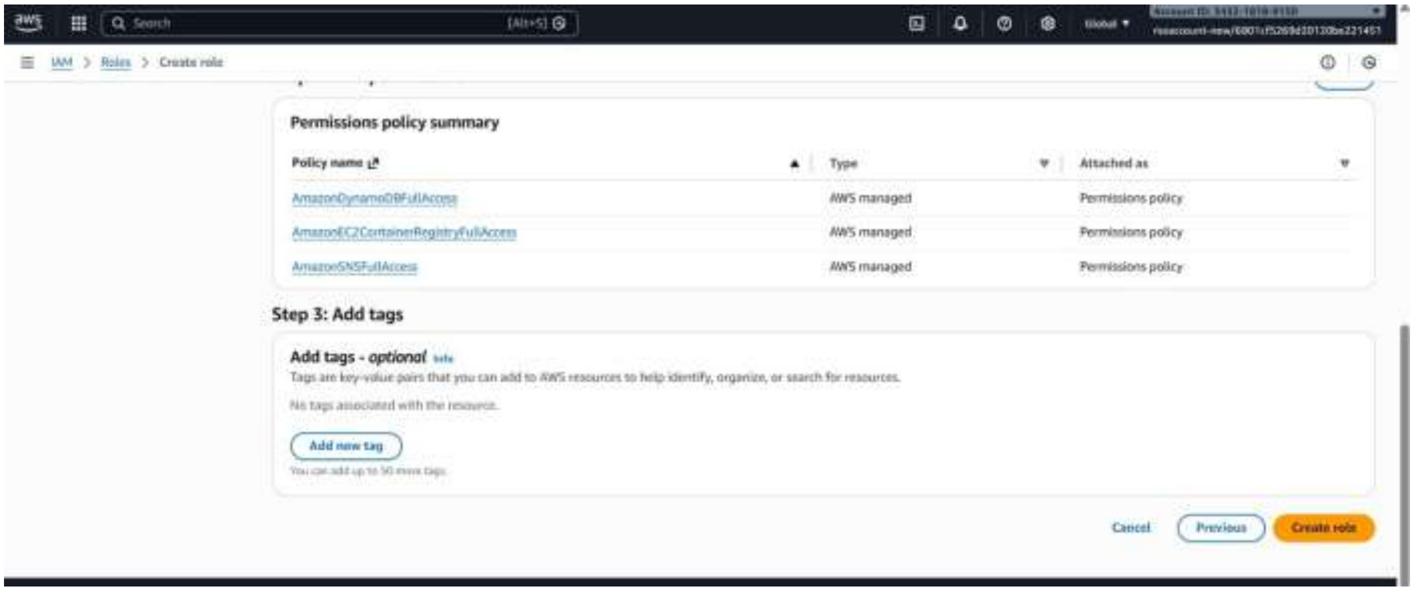
Description
Add a short explanation for this role.
Allows EC2 instances to call AWS services on your behalf.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,.@-/\[]{}!#$%^*();:'"

Step 1: Select trusted entities

Trust policy

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

© 2026, Amazon Web Services, Inc. or its affiliates.



IAM > Roles > Create role

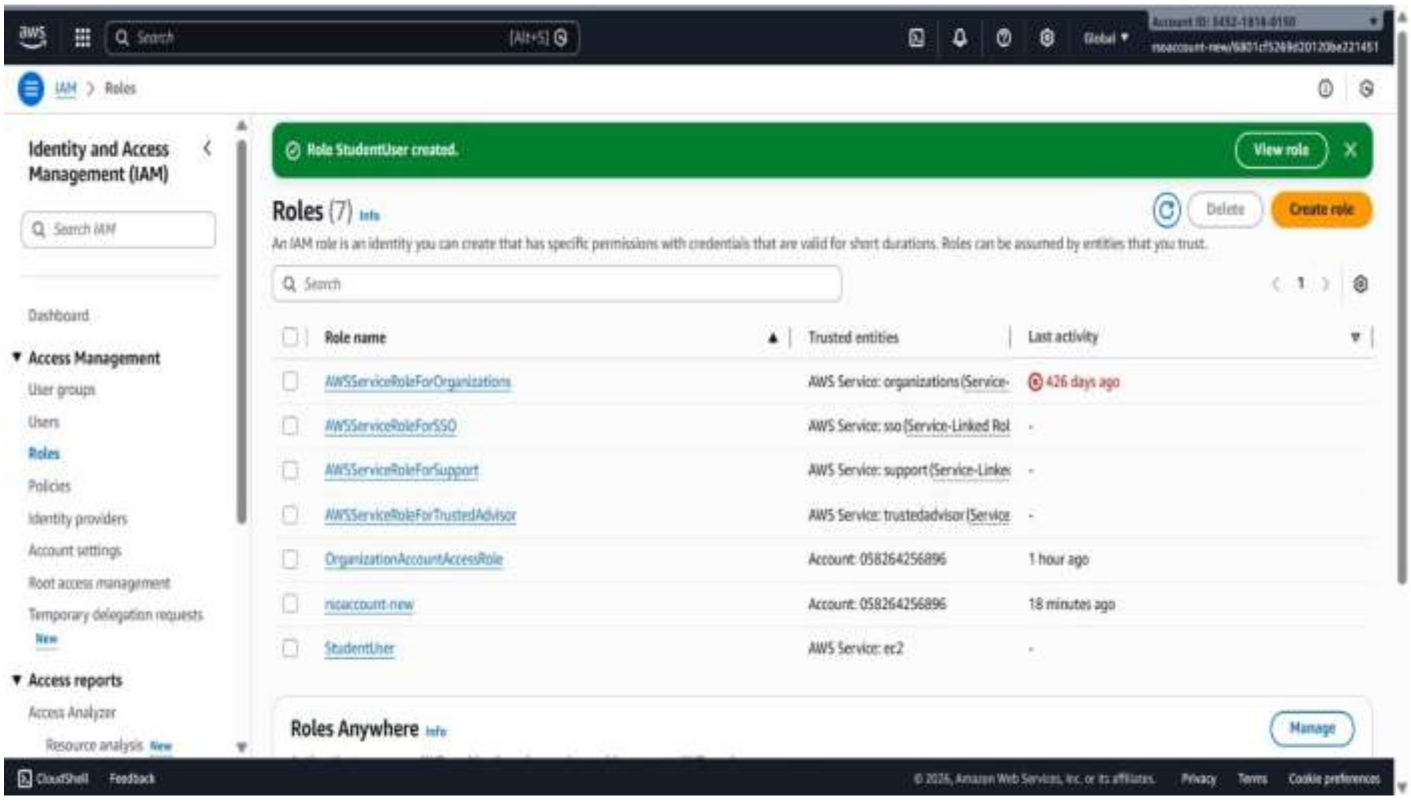
Permissions policy summary

Policy name	Type	Attached as
AmazonDynamoDBFullAccess	AWS managed	Permissions policy
AmazonEC2ContainerRegistryFullAccess	AWS managed	Permissions policy
AmazonSNSFullAccess	AWS managed	Permissions policy

Step 3: Add tags

Add tags - *optional* Info
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.
No tags associated with the resource.
Add new tag
You can add up to 50 more tags.

Cancel Previous Create role



Role StudentUser created.

Roles (7) Info

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

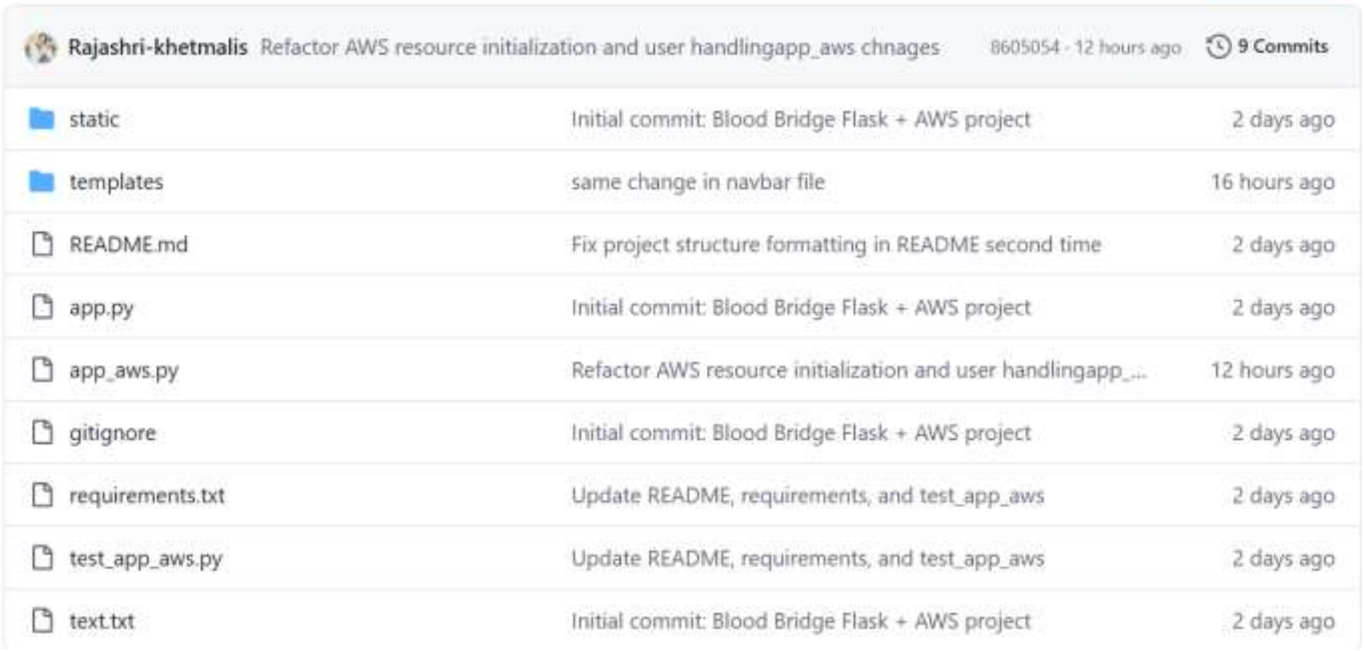
Role name	Trusted entities	Last activity
AWSServiceRoleForOrganizations	AWS Service: organizations (Service-	426 days ago
AWSServiceRoleForSSO	AWS Service: sso (Service-Linked Rol	-
AWSServiceRoleForSupport	AWS Service: support (Service-Linke	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service	-
OrganizationAccountAccessRole	Account: 058264256896	1 hour ago
nsaccount-new	Account: 058264256896	18 minutes ago
StudentUser	AWS Service: ec2	-

Roles Anywhere Info

© 2026, Amazon Web Services, Inc. or its affiliates.

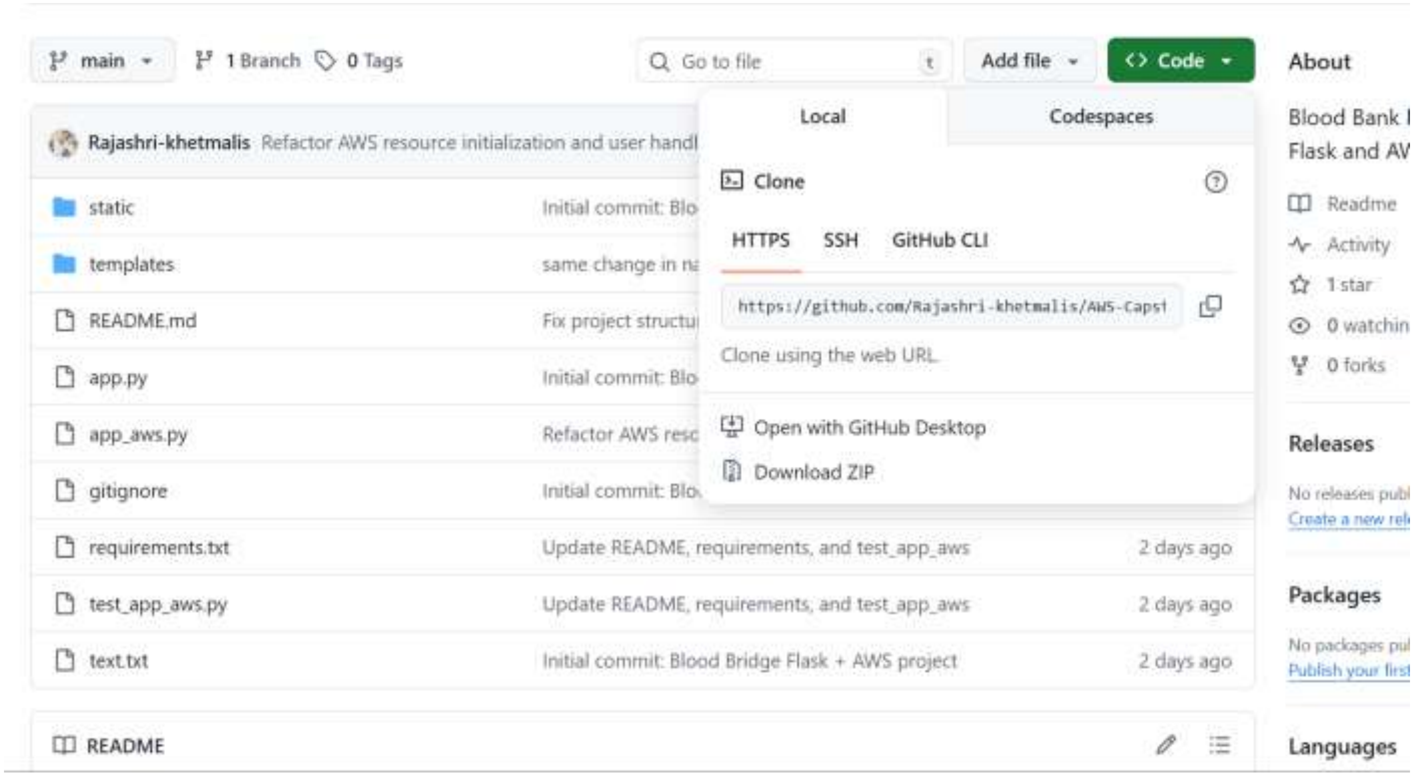
Milestone 6: EC2 Instance Setup

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



Rajashri-khetmalis Refactor AWS resource initialization and user handlingapp_aws chnages 8605054 · 12 hours ago 9 Commits

static	Initial commit: Blood Bridge Flask + AWS project	2 days ago
templates	same change in navbar file	16 hours ago
README.md	Fix project structure formatting in README second time	2 days ago
app.py	Initial commit: Blood Bridge Flask + AWS project	2 days ago
app_aws.py	Refactor AWS resource initialization and user handlingapp_...	12 hours ago
gitignore	Initial commit: Blood Bridge Flask + AWS project	2 days ago
requirements.txt	Update README, requirements, and test_app_aws	2 days ago
test_app_aws.py	Update README, requirements, and test_app_aws	2 days ago
text.txt	Initial commit: Blood Bridge Flask + AWS project	2 days ago



main · 1 Branch · 0 Tags · Go to file · Add file · Code

Rajashri-khetmalis Refactor AWS resource initialization and user handl

static	Initial commit: Blo	
templates	same change in na	
README.md	Fix project structu	
app.py	Initial commit: Blo	
app_aws.py	Refactor AWS reso	
gitignore	Initial commit: Blo	
requirements.txt	Update README, requirements, and test_app_aws	2 days ago
test_app_aws.py	Update README, requirements, and test_app_aws	2 days ago
text.txt	Initial commit: Blood Bridge Flask + AWS project	2 days ago

Local | Codespaces

Clone

HTTPS SSH GitHub CLI

```
https://github.com/Rajashri-khetmalis/AWS-Capst
```

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

About

Blood Bank I
Flask and AV

Readme
Activity
1 star
0 watchin
0 forks

Releases
No releases publ
Create a new rel

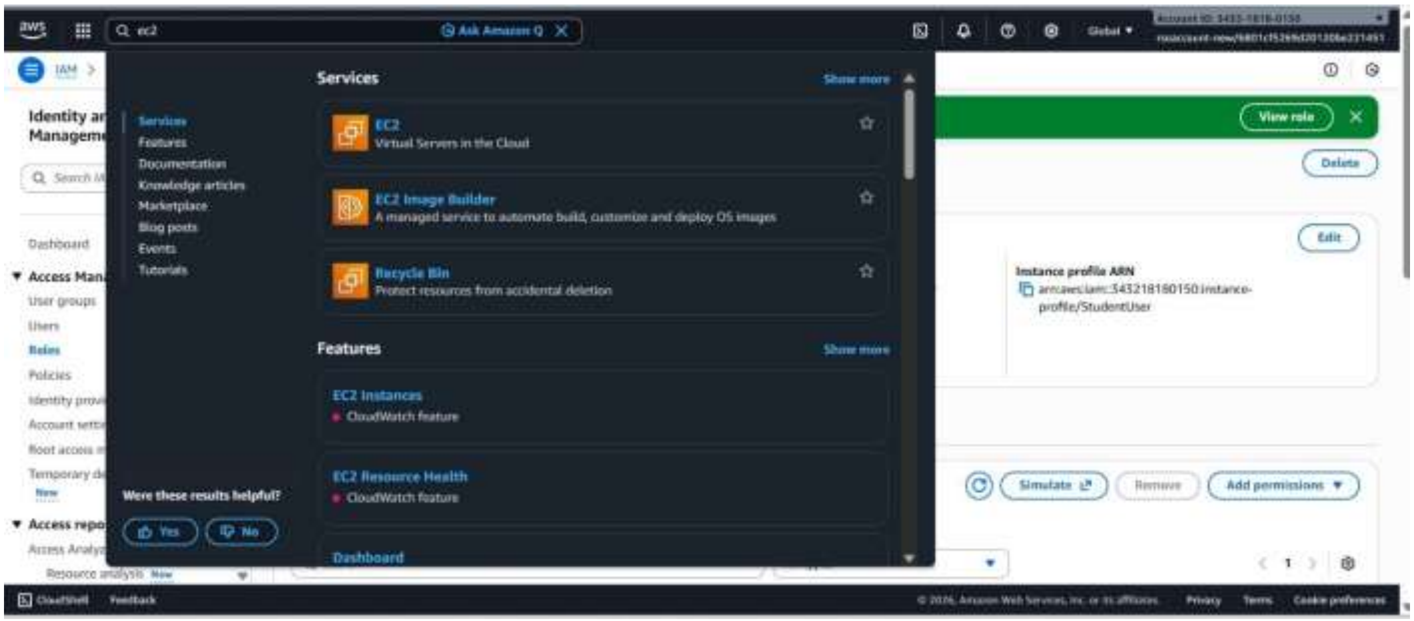
Packages
No packages pu
Publish your first

README

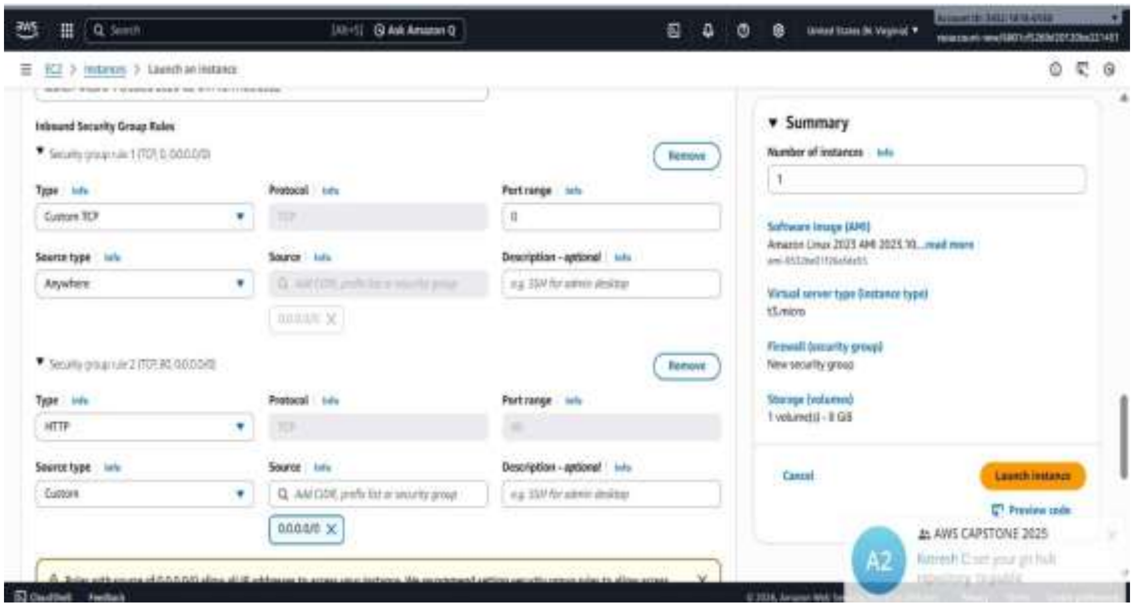
Languages

- **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.



- Click on Launch instance to launch EC2 instance

- 

Inbound Security Group Rules

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

Type: Custom TCP | Protocol: TCP | Port range: 0

Source type: Anywhere | Source: 0.0.0.0/0 | Description - optional: e.g. SSH for admin desktop

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

Type: HTTP | Protocol: TCP | Port range: 80

Source type: Custom | Source: 0.0.0.0/0 | Description - optional: e.g. SSH for admin desktop

Summary

Number of instances: 1

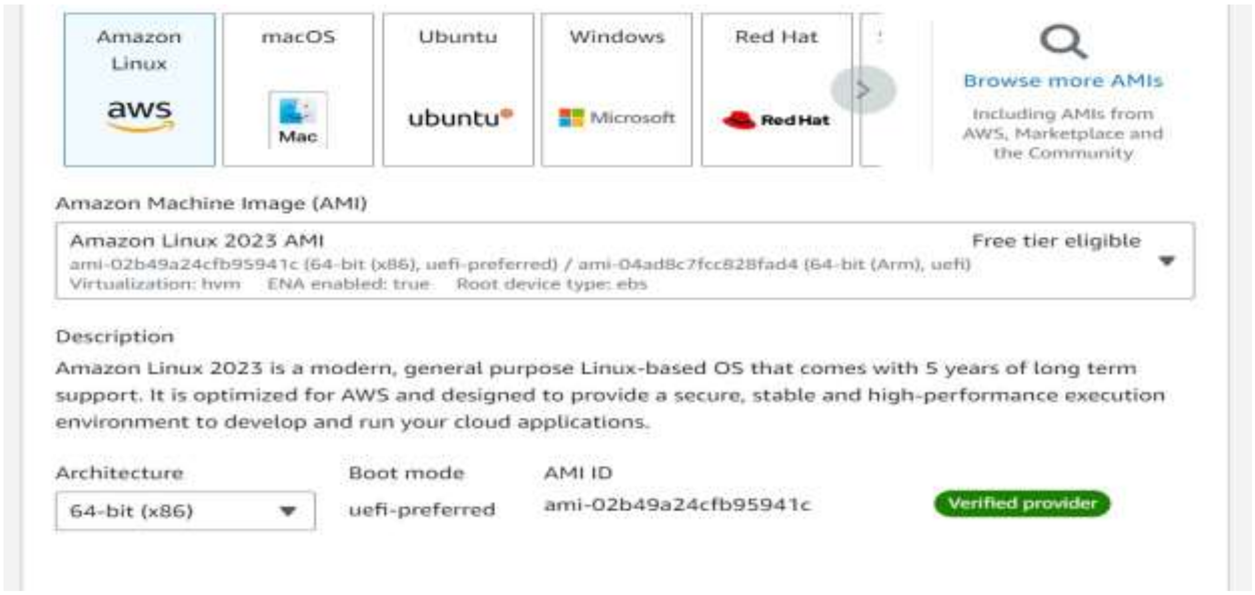
Software Image (AMI): Amazon Linux 2023 AMI 2025.10...read more

Virtual server type (instance type): t3.micro

Firewall (security group): New security group

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

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



Amazon Machine Image (AMI)

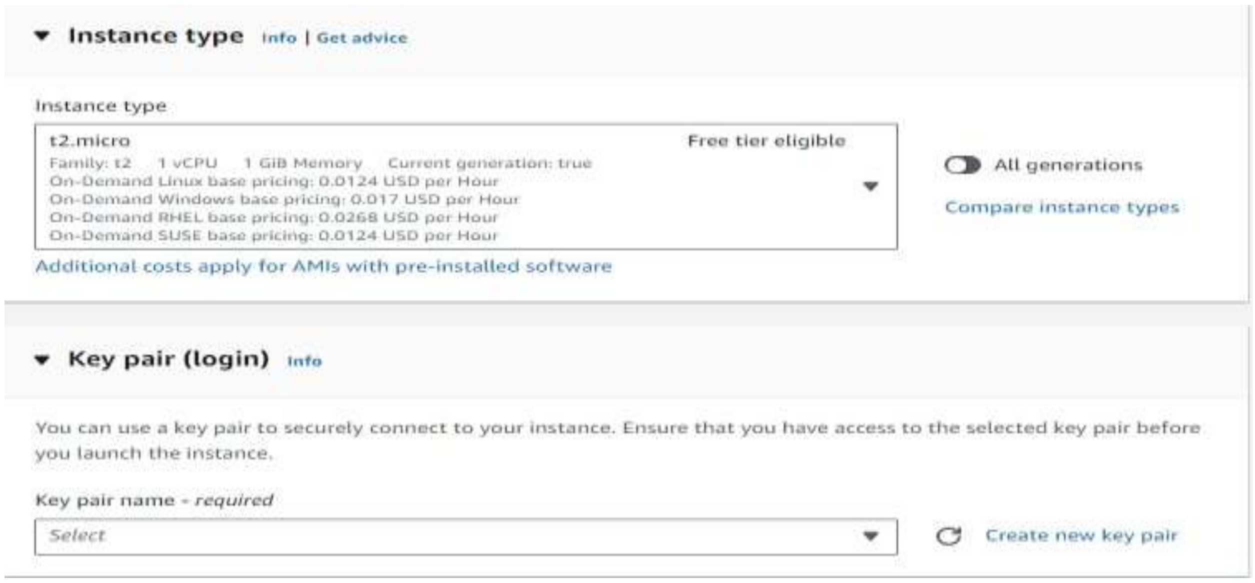
Amazon Linux 2023 AMI — Free tier eligible
ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

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

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

- Create and download the key pair for Server access.



▼ Instance type Info | Get advice

Instance type

t2.micro Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0268 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

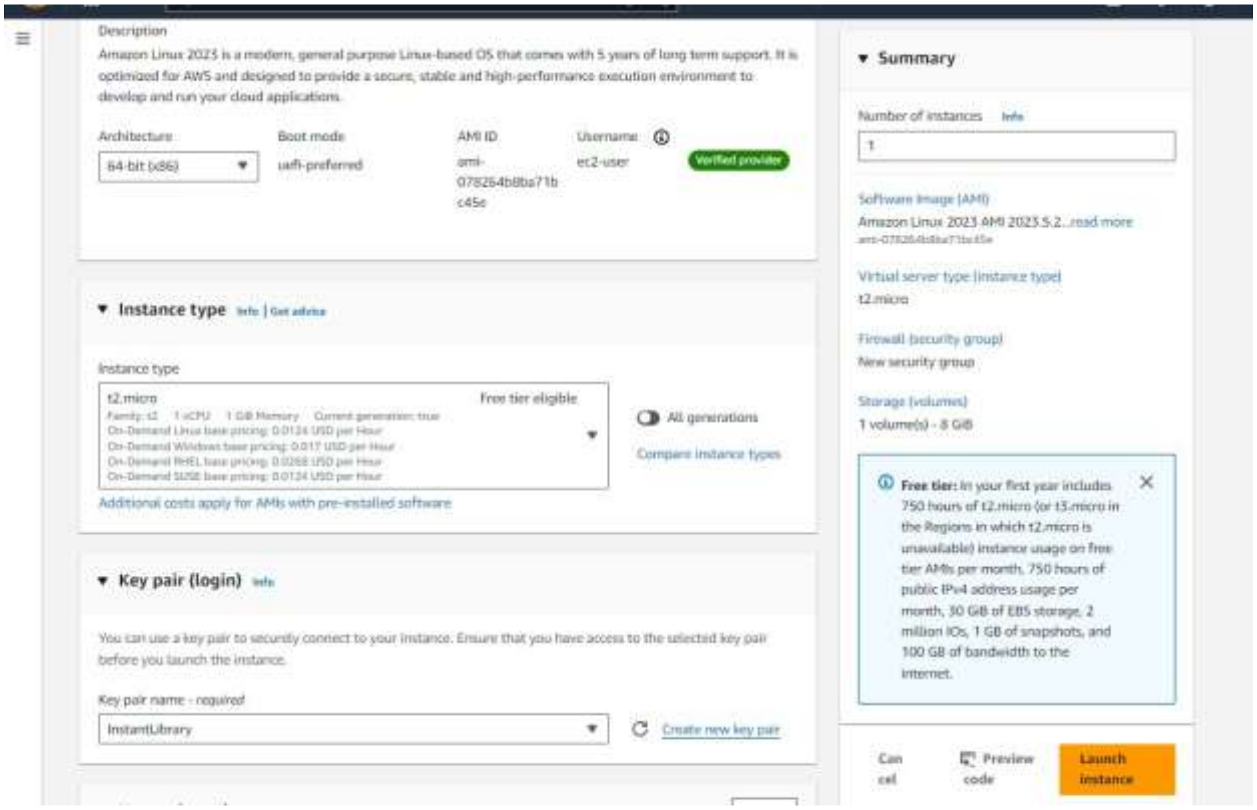
▼ Key pair (login) Info

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

Key pair name - *required*

Select

Create new key pair


Description

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

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

Verified provider

Instance type Info | Get advice

Instance type

t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0268 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

Key pair (login) Info

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

Key pair name - *required*

InstantLibrary

Create new key pair

Summary

Number of instances Info

1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.5.2...read more
ami-078264b8ba71bc45e

Virtual server type (instance type)
t2.micro

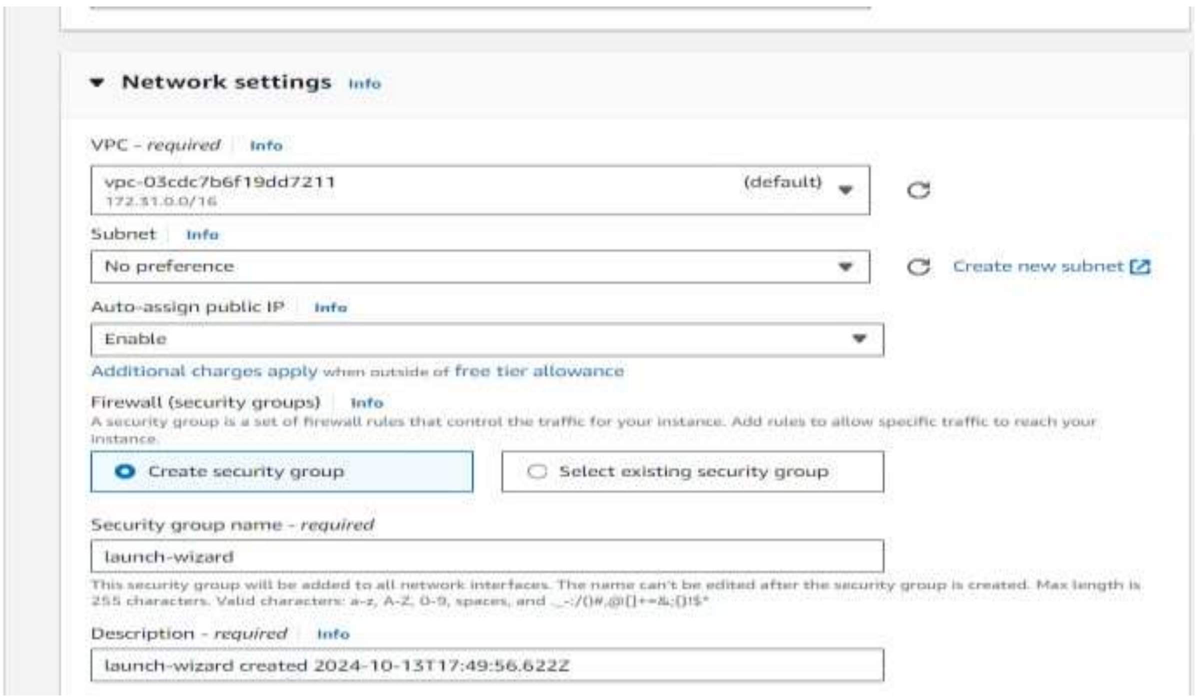
Firewall (security group)
New security group

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

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

Cancel Preview code Launch instance

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



Network settings Info

VPC - *required* Info

vpc-03cdc7b6f19dd7211 (default)
172.31.0.0/16

Subnet Info

No preference

Create new subnet

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

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

[✓ Create security group] Select existing security group

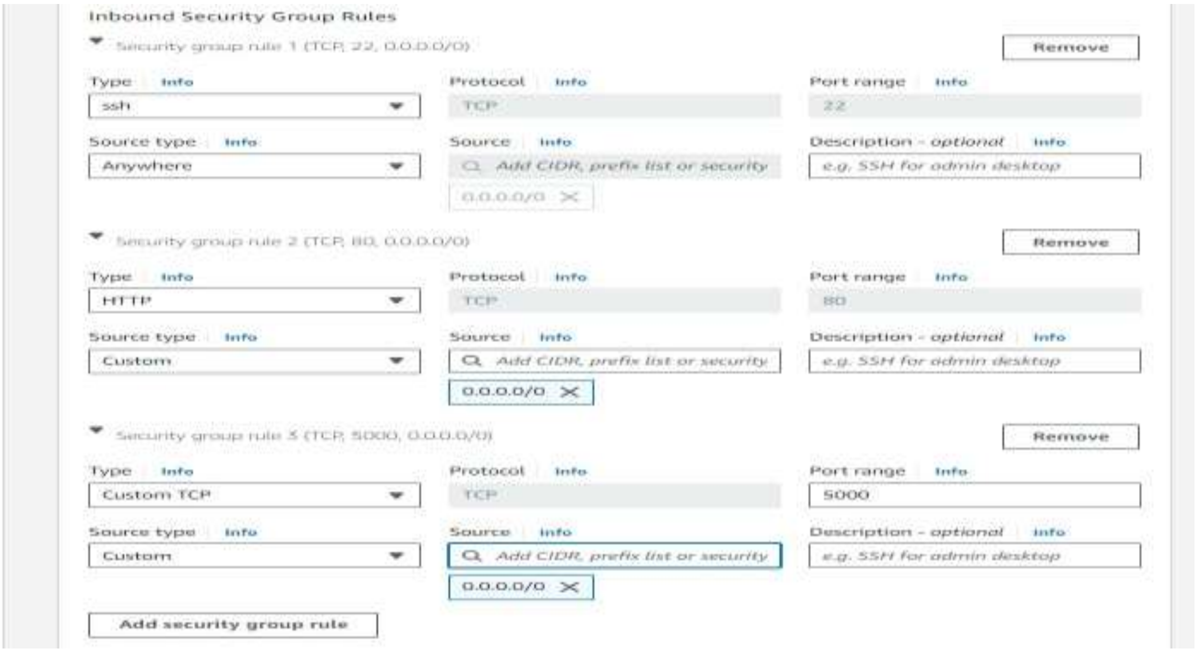
Security group name - *required*

launch-wizard

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

Description - *required* Info

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



Inbound Security Group Rules

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

Type	Protocol	Port range
ssh	TCP	22
Source type	**Source**	**Description - optional**
Anywhere	0.0.0.0/0	e.g. SSH for admin desktop

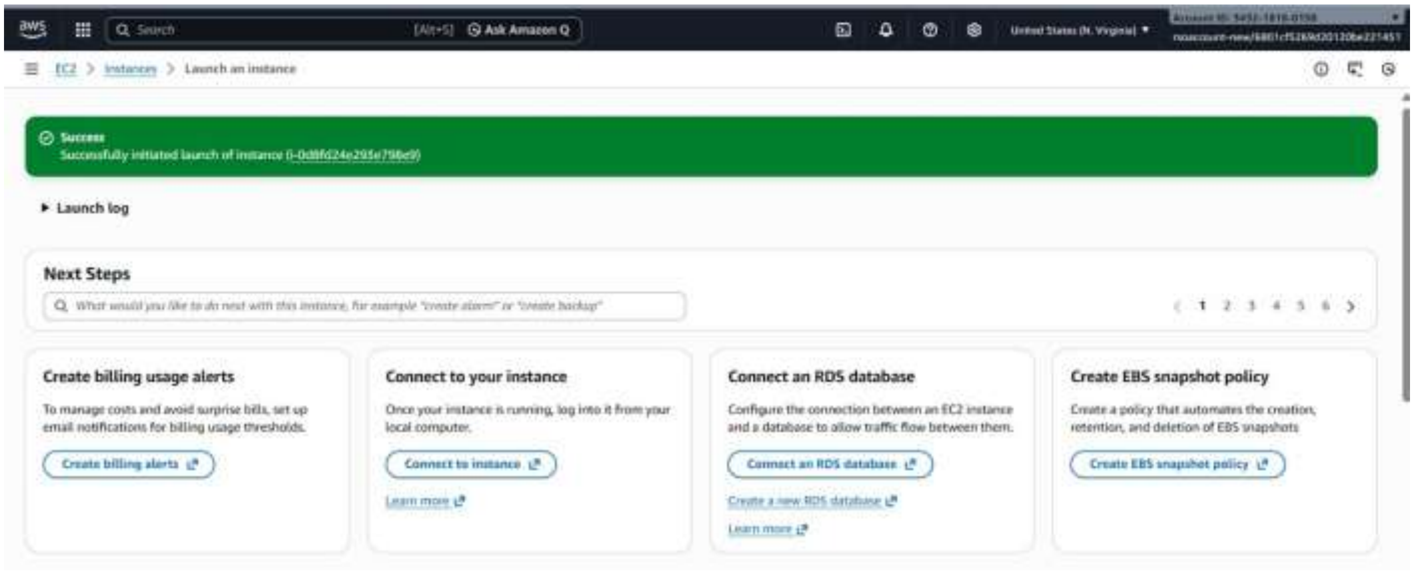
Security group rule 2 (TCP, 80, 0.0.0.0/0)

Type	Protocol	Port range
HTTP	TCP	80
Source type	**Source**	**Description - optional**
Custom	0.0.0.0/0	e.g. SSH for admin desktop

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

Type	Protocol	Port range
Custom TCP	TCP	5000
Source type	**Source**	**Description - optional**
Custom	0.0.0.0/0	e.g. SSH for admin desktop

Add security group rule



Success
Successfully initiated launch of instance (i-0d8fd24e293e798e9)

▸ Launch log

Next Steps

What would you like to do next with this instance, for example "create alarm" or "create backup"

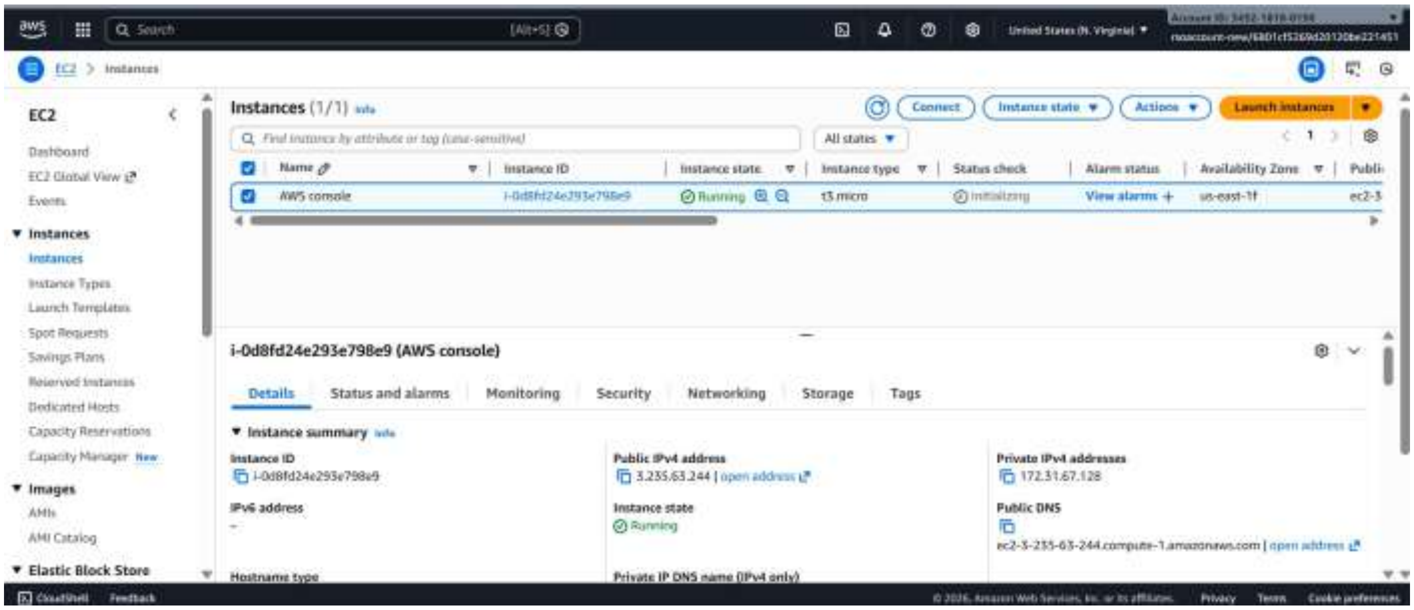
Create billing usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing usage thresholds.
Create billing alerts

Connect to your instance
Once your instance is running, log into it from your local computer.
Connect to instance
Learn more

Connect an RDS database
Configure the connection between an EC2 instance and a database to allow traffic flow between them.
Connect an RDS database
Create a new RDS database
Learn more

Create EBS snapshot policy
Create a policy that automates the creation, retention, and deletion of EBS snapshots
Create EBS snapshot policy

- 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.



Instances (1/1)

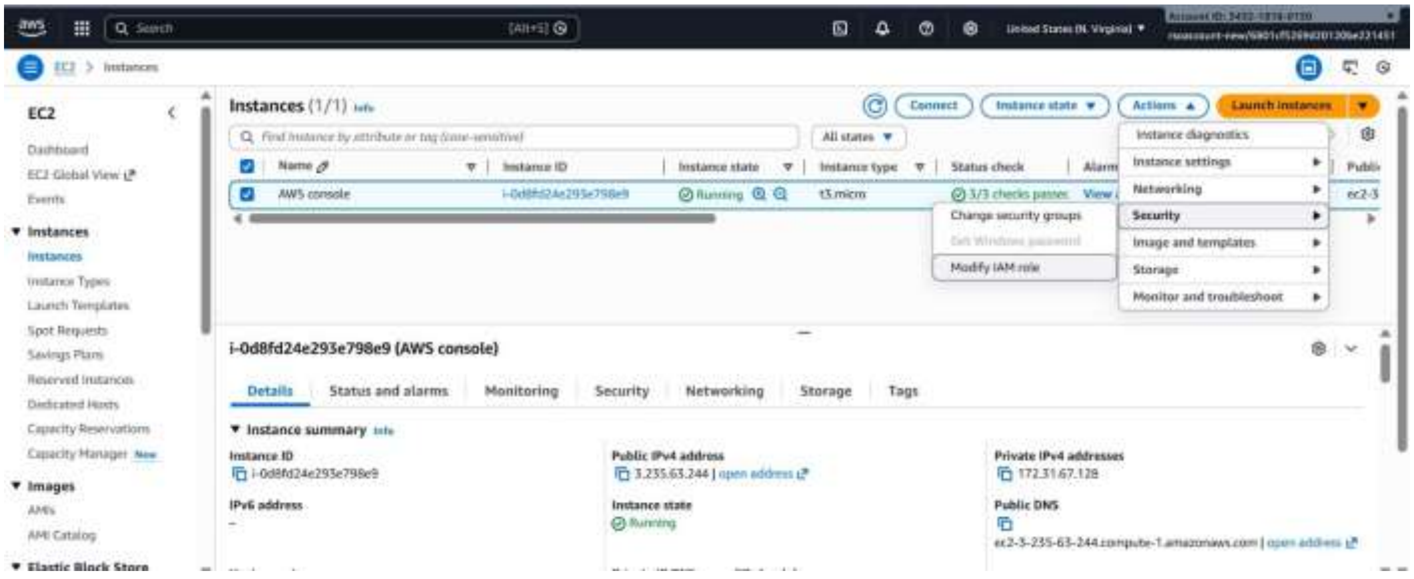
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Publi
AWS console	i-0d8fd24e293e798e9	Running	t3.micro	Initializing	View alarms +	us-east-1f	ec2-3

i-0d8fd24e293e798e9 (AWS console)

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

▾ Instance summary

Instance ID: i-0d8fd24e293e798e9
Public IPv4 address: 3.235.63.244 | open address
Private IPv4 addresses: 172.31.67.128
IPv6 address: –
Instance state: Running
Public DNS: ec2-3-235-63-244.compute-1.amazonaws.com | open address
Hostname type
Private IP DNS name (IPv4 only)

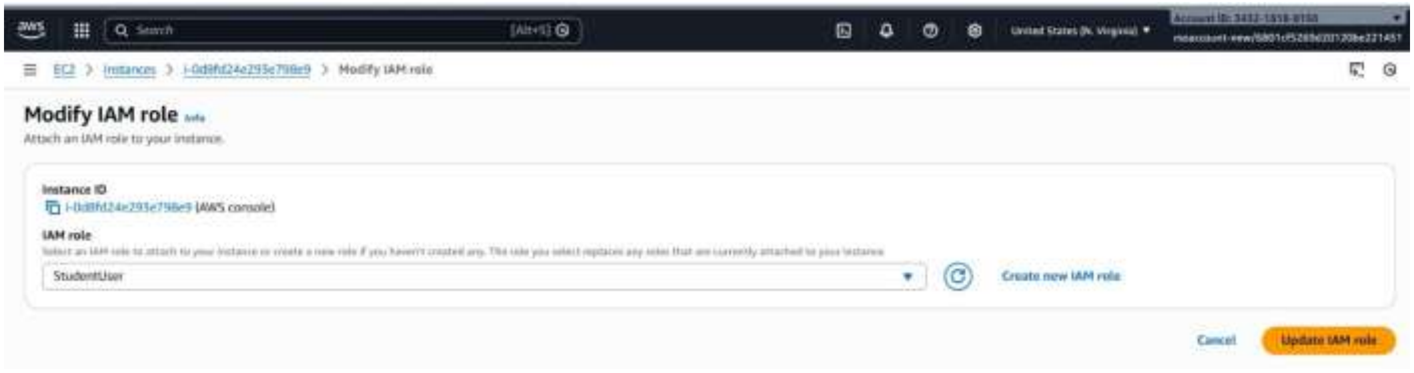


i-0d8fd24e293e798e9 (AWS console)

Name	Instance ID	Instance state	Instance type	Status check
AWS console	i-0d8fd24e293e798e9	Running	t3.micro	3/3 checks passed

Instance summary

Instance ID: i-0d8fd24e293e798e9
Public IPv4 address: 3.235.63.244 | open address
Private IPv4 addresses: 172.31.67.128
IPv6 address: –
Instance state: Running
Public DNS: ec2-3-235-63-244.compute-1.amazonaws.com | open address



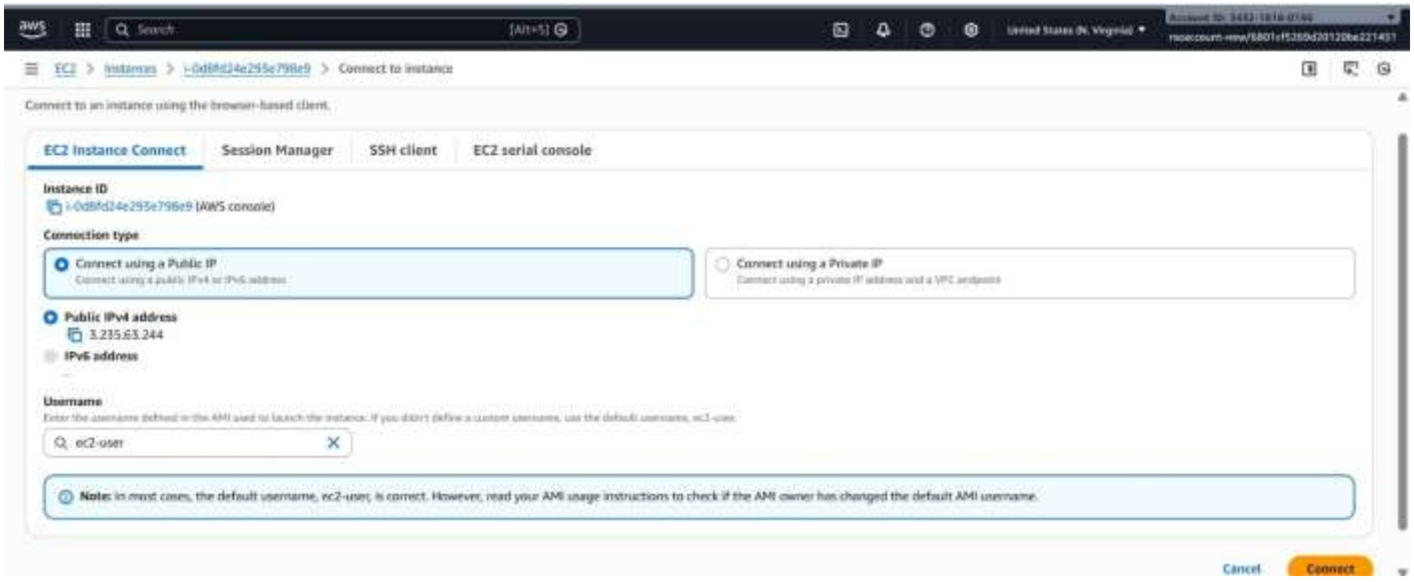
Modify IAM role

Attach an IAM role to your instance.

Instance ID: i-0d8fd24e293e798e9 (AWS console)

IAM role: StudentUser

- Now connect the EC2 with the files


EC2 > Instances > i-0d8fd24e293e798e9 > Connect to instance

Connect to an instance using the browser-based client.

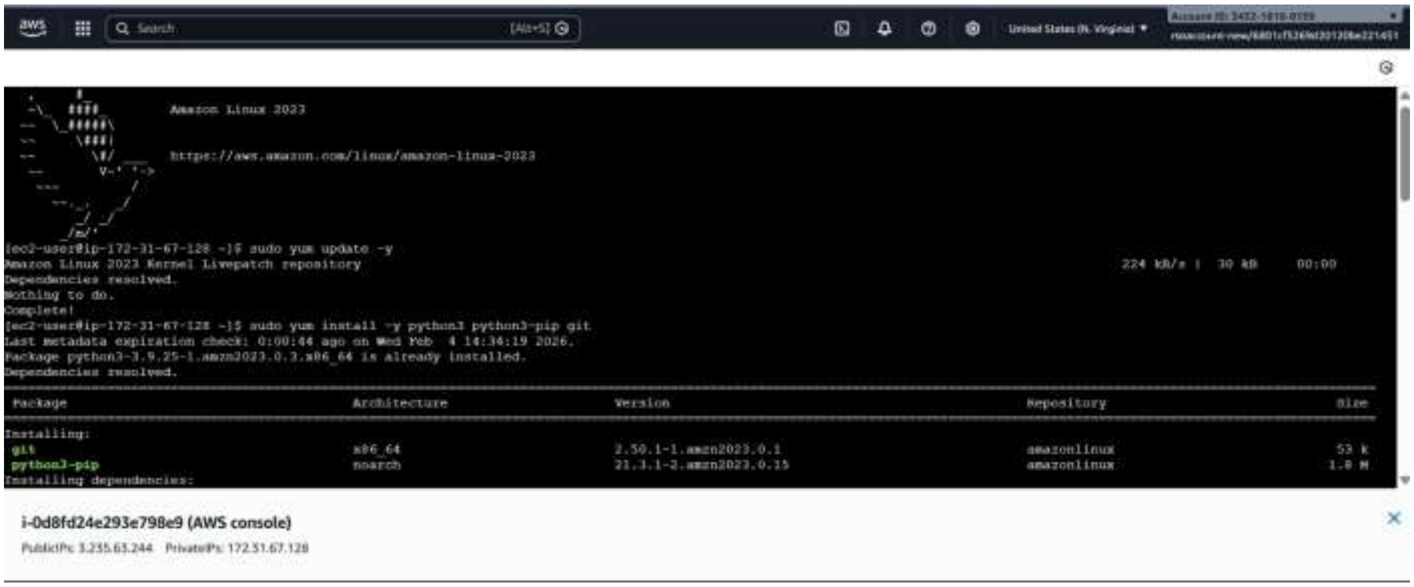
Instance ID
i-0d8fd24e293e798e9 (AWS console)

Connection type
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
3.235.63.244
IPv6 address

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

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


```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-67-128 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository                          224 kB/s |  30 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-67-128 ~]$ sudo yum install -y python3 python3-pip git
Last metadata expiration check: 0:00:44 ago on Wed Feb  4 14:34:19 2026.
Package python3-3.9.25-1.amzn2023.0.3.x86_64 is already installed.
Dependencies resolved.
==========================================================================================================
 Package              Architecture         Version                          Repository             Size
==========================================================================================================
Installing:
 git                  x86_64               2.50.1-1.amzn2023.0.1            amazonlinux            53 k
 python3-pip          noarch               21.3.1-2.amzn2023.0.15           amazonlinux           1.8 M
Installing dependencies:
```

i-0d8fd24e293e798e9 (AWS console)
PublicIPs: 3.235.63.244 PrivateIPs: 172.31.67.128

Milestone 7: Deployment on EC2

Activity 7.1: Install Software on the EC2 Instance

Install Python3, Flask, and Git:

On Amazon Linux 2:

```
sudo yum update -y
```

sudo yum install python3 git

sudo pip3 install flask boto3

Verify Installations:

flask --version

git --version

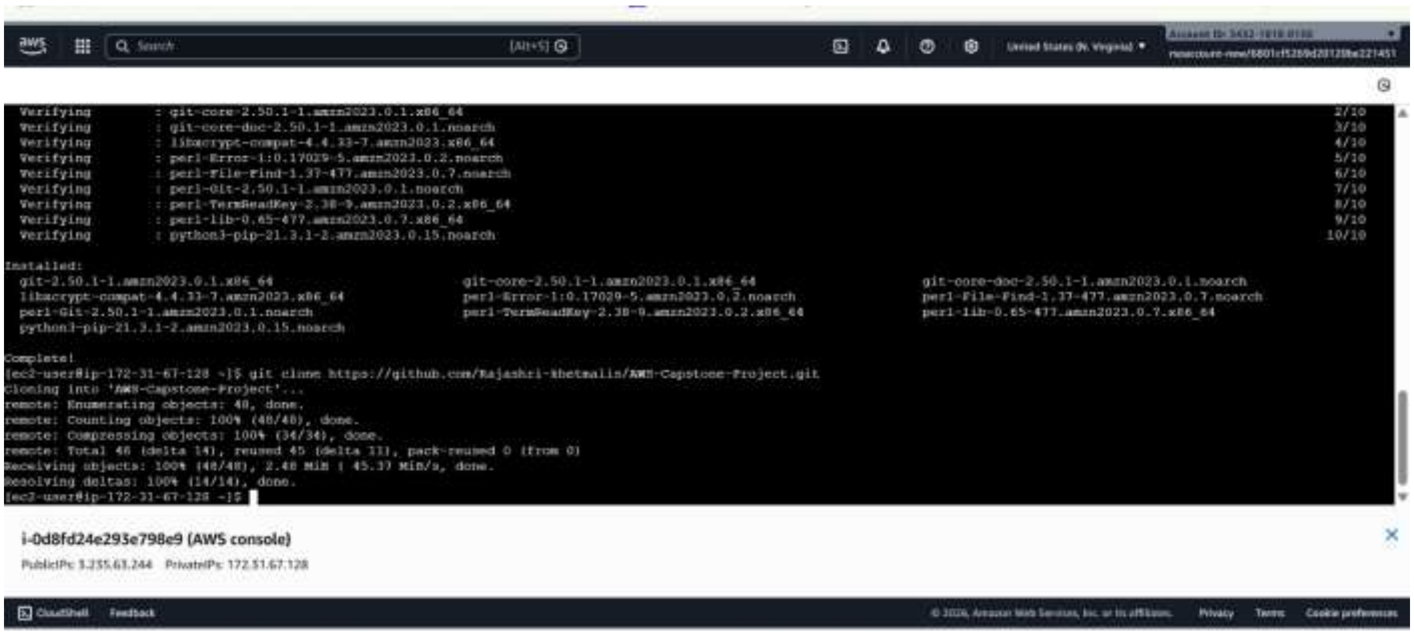
Activity 7.2:Clone Your Flask Project from GitHub

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

Run: https://github.com/Rajashri-khetmalis/AWS-Capstone-Project.git

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

here: ‘https://github.com/Rajashri-khetmalis/AWS-Capstone-Project.git



```
  Verifying        : git-core-2.50.1-1.amzn2023.0.1.x86_64                                    2/10
  Verifying        : git-core-doc-2.50.1-1.amzn2023.0.1.noarch                                3/10
  Verifying        : libxcrypt-compat-4.4.33-7.amzn2023.x86_64                                4/10
  Verifying        : perl-Error-1:0.17029-5.amzn2023.0.2.noarch                               5/10
  Verifying        : perl-File-Find-1.37-477.amzn2023.0.7.noarch                              6/10
  Verifying        : perl-Git-2.50.1-1.amzn2023.0.1.noarch                                    7/10
  Verifying        : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64                              8/10
  Verifying        : perl-lib-0.65-477.amzn2023.0.7.x86_64                                    9/10
  Verifying        : python3-pip-21.3.1-2.amzn2023.0.15.noarch                               10/10

Installed:
  git-2.50.1-1.amzn2023.0.1.x86_64               git-core-2.50.1-1.amzn2023.0.1.x86_64            git-core-doc-2.50.1-1.amzn2023.0.1.noarch
  libxcrypt-compat-4.4.33-7.amzn2023.x86_64      perl-Error-1:0.17029-5.amzn2023.0.2.noarch       perl-File-Find-1.37-477.amzn2023.0.7.noarch
  perl-Git-2.50.1-1.amzn2023.0.1.noarch          perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64      perl-lib-0.65-477.amzn2023.0.7.x86_64
  python3-pip-21.3.1-2.amzn2023.0.15.noarch

Complete!
[ec2-user@ip-172-31-67-128 ~]$ git clone https://github.com/Rajashri-khetmalis/AWS-Capstone-Project.git
Cloning into 'AWS-Capstone-Project'...
remote: Enumerating objects: 48, done.
remote: Counting objects: 100% (48/48), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 48 (delta 14), reused 45 (delta 11), pack-reused 0 (from 0)
Receiving objects: 100% (48/48), 2.48 MiB | 45.37 MiB/s, done.
Resolving deltas: 100% (14/14), done.
[ec2-user@ip-172-31-67-128 ~]$
```

i-0d8fd24e293e798e9 (AWS console)

PublicIPs: 3.235.63.244   PrivateIPs: 172.31.67.128

- This will download your project to the EC2 instance.

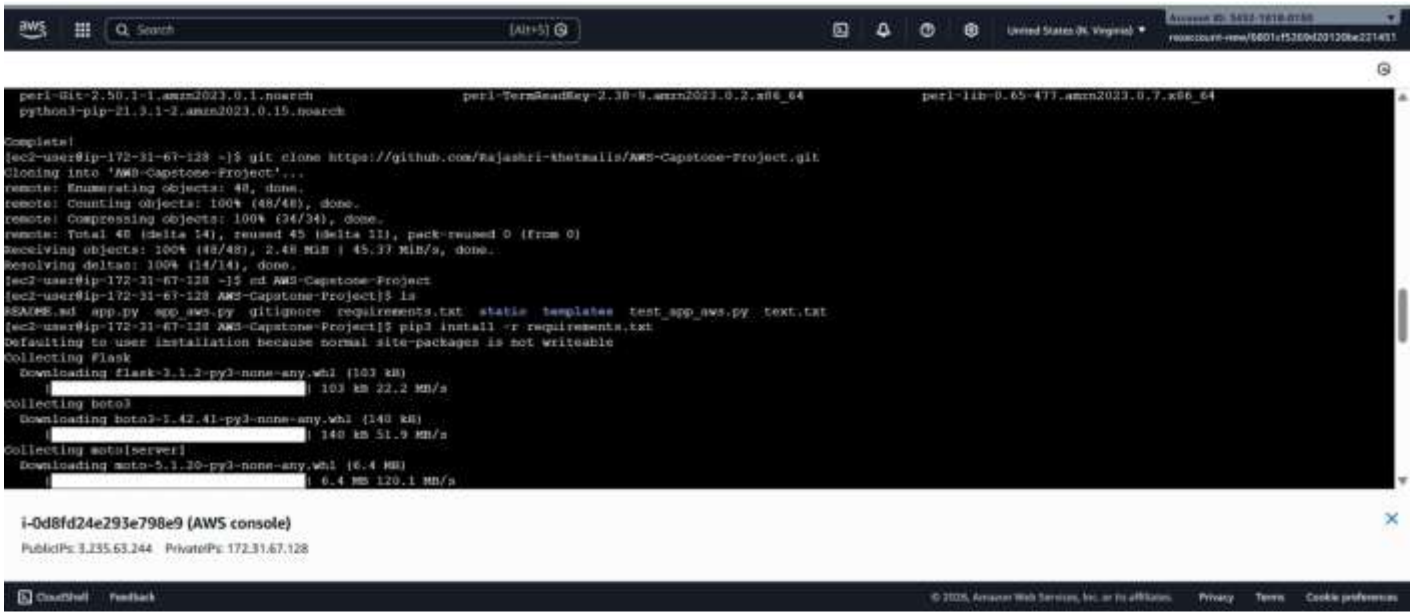
To navigate to the project directory, run the following command:

cd AWS-Capstone-Project
Cd app_aws.py

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



```
  perl-Git-2.50.1-1.amzn2023.0.1.noarch                perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64              perl-lib-0.65-477.amzn2023.0.7.x86_64
  python3-pip-21.3.1-2.amzn2023.0.15.noarch

Complete!
[ec2-user@ip-172-31-67-128 ~]$ git clone https://github.com/Rajashri-khetmalis/AWS-Capstone-Project.git
Cloning into 'AWS-Capstone-Project'...
remote: Enumerating objects: 48, done.
remote: Counting objects: 100% (48/48), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 48 (delta 14), reused 45 (delta 11), pack-reused 0 (from 0)
Receiving objects: 100% (48/48), 2.48 MiB | 45.37 MiB/s, done.
Resolving deltas: 100% (14/14), done.
[ec2-user@ip-172-31-67-128 ~]$ cd AWS-Capstone-Project
[ec2-user@ip-172-31-67-128 AWS-Capstone-Project]$ ls
README.md  app.py  app_aws.py  gitignore  requirements.txt  static  templates  test_app_aws.py  text.txt
[ec2-user@ip-172-31-67-128 AWS-Capstone-Project]$ pip3 install -r requirements.txt
Defaulting to user installation because normal site-packages is not writeable
Collecting Flask
  Downloading flask-3.1.2-py3-none-any.whl (103 kB)
     |████████████████████████████████| 103 kB 22.2 MB/s
Collecting boto3
  Downloading boto3-1.42.41-py3-none-any.whl (140 kB)
     |████████████████████████████████| 140 kB 51.9 MB/s
Collecting moto[server]
  Downloading moto-5.1.20-py3-none-any.whl (6.4 MB)
     |████████████████████████████████| 6.4 MB 120.1 MB/s
```

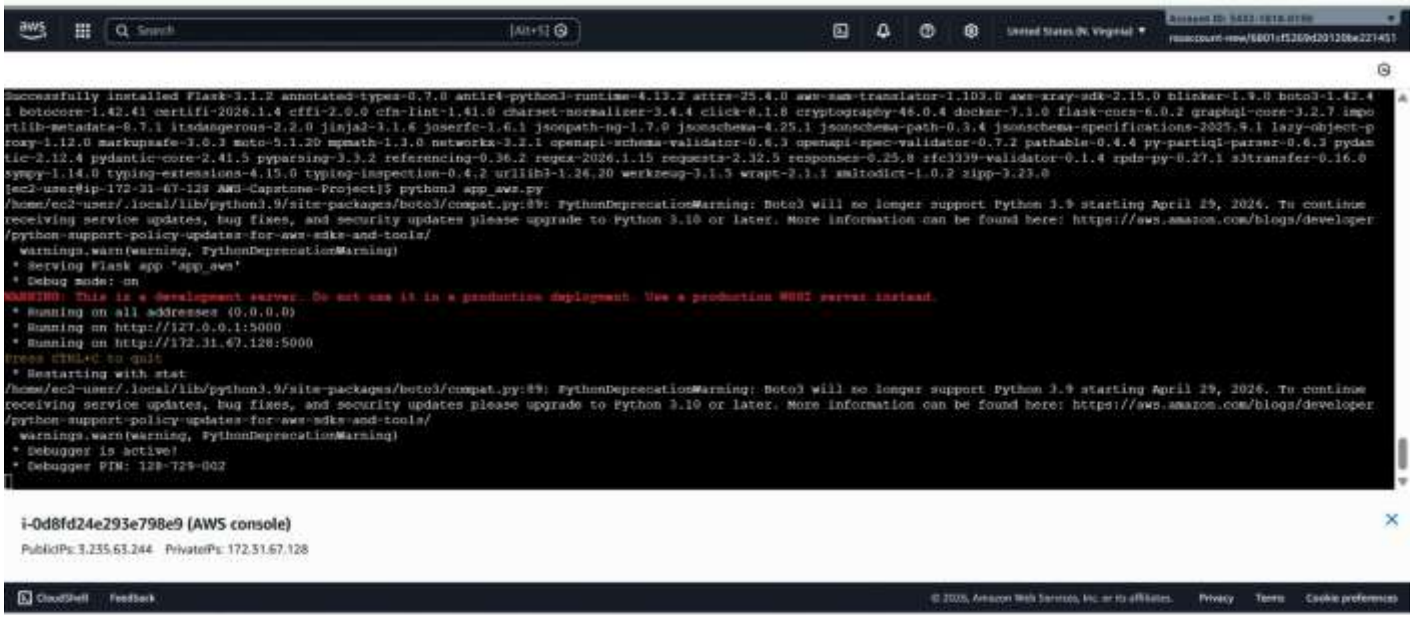
i-0d8fd24e293e798e9 (AWS console)
PublicIPs: 3.235.63.244   PrivateIPs: 172.31.67.128

Search
[Alt+S]
United States (N. Virginia)
CloudShell
Feedback
© 2026, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

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

ip

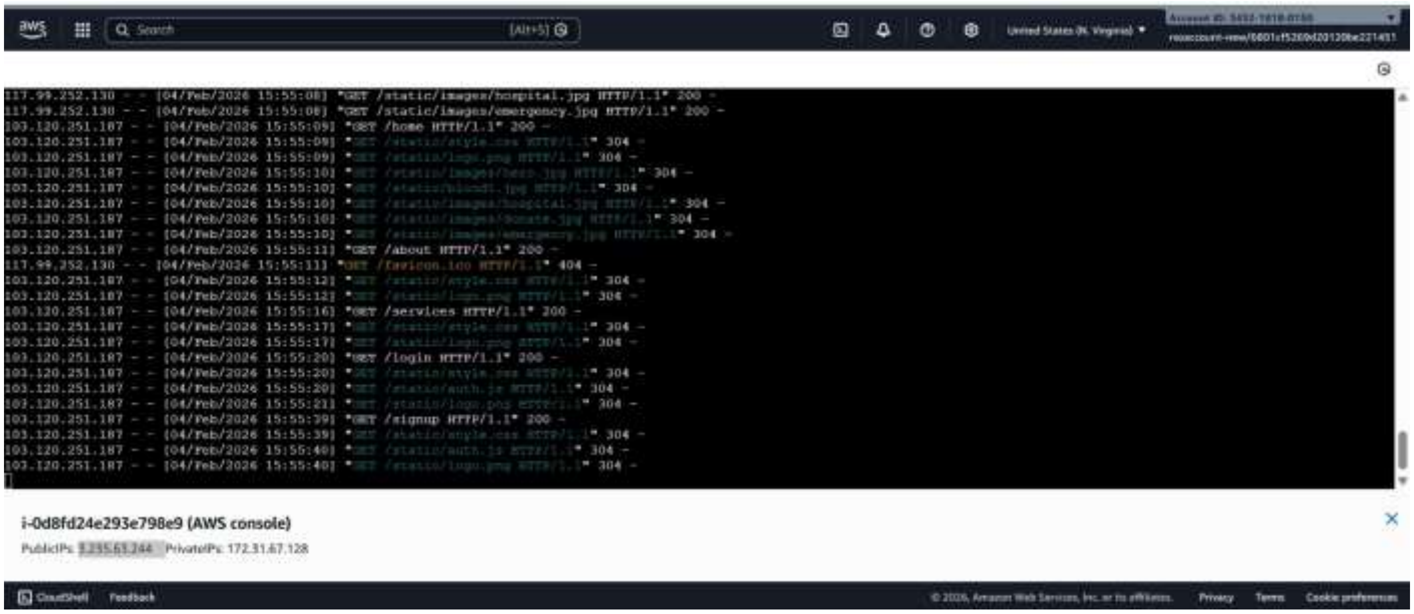
○ Run the Flask app on the EC2 instance



```
Successfully installed Flask-3.1.2 annotated-types-0.7.0 antlr4-python3-runtime-4.13.2 attrs-25.4.0 aws-sam-translator-1.103.0 aws-xray-sdk-2.15.0 blinker-1.9.0 boto3-1.42.4
1 botocore-1.42.41 certifi-2026.1.4 cffi-2.0.0 cfn-lint-1.41.0 charset-normalizer-3.4.4 click-8.1.8 cryptography-46.0.4 docker-7.1.0 flask-cors-6.0.2 graphql-core-3.2.7 impo
rtlib-metadata-8.7.1 itsdangerous-2.2.0 jinja2-3.1.6 joserfc-1.6.1 jsonpath-ng-1.7.0 jsonschema-4.25.1 jsonschema-path-0.3.4 jsonschema-specifications-2025.9.1 lazy-object-p
roxy-1.12.0 markupsafe-3.0.3 moto-5.1.20 mpmath-1.3.0 networkx-3.2.1 openapi-schema-validator-0.6.3 openapi-spec-validator-0.7.2 pathable-0.4.4 py-partiql-parser-0.6.3 pydan
tic-2.12.4 pydantic-core-2.41.5 pyparsing-3.3.2 referencing-0.36.2 regex-2026.1.15 requests-2.32.5 responses-0.25.8 rfc3339-validator-0.1.4 rpds-py-0.27.1 s3transfer-0.16.0
sympy-1.14.0 typing-extensions-4.15.0 typing-inspection-0.4.2 urllib3-1.26.20 werkzeug-3.1.5 wrapt-2.1.1 xmltodict-1.0.2 zipp-3.23.0
[ec2-user@ip-172-31-67-128 AWS-Capstone-Project]$ python3 app_aws.py
/home/ec2-user/.local/lib/python3.9/site-packages/boto3/compat.py:89: PythonDeprecationWarning: Boto3 will no longer support Python 3.9 starting April 29, 2026. To continue
receiving service updates, bug fixes, and security updates please upgrade to Python 3.10 or later. More information can be found here: https://aws.amazon.com/blogs/developer
/python-support-policy-updates-for-aws-sdks-and-tools/
  warnings.warn(warning, PythonDeprecationWarning)
 * Serving Flask app 'app_aws'
 * 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://172.31.67.128:5000
Press CTRL+C to quit
 * Restarting with stat
/home/ec2-user/.local/lib/python3.9/site-packages/boto3/compat.py:89: PythonDeprecationWarning: Boto3 will no longer support Python 3.9 starting April 29, 2026. To continue
receiving service updates, bug fixes, and security updates please upgrade to Python 3.10 or later. More information can be found here: https://aws.amazon.com/blogs/developer
/python-support-policy-updates-for-aws-sdks-and-tools/
  warnings.warn(warning, PythonDeprecationWarning)
 * Debugger is active!
 * Debugger PIN: 128-729-002
```

i-0d8fd24e293e798e9 (AWS console)

PublicIPs: 3.235.63.244 PrivateIPs: 172.31.67.128



```
117.99.252.130 - - [04/Feb/2026 15:55:08] "GET /static/images/hospital.jpg HTTP/1.1" 200 -
117.99.252.130 - - [04/Feb/2026 15:55:08] "GET /static/images/emergency.jpg HTTP/1.1" 200 -
103.120.251.187 - - [04/Feb/2026 15:55:09] "GET /home HTTP/1.1" 200 -
103.120.251.187 - - [04/Feb/2026 15:55:09] "GET /static/style.css HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:09] "GET /static/logo.png HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:10] "GET /static/images/hero.jpg HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:10] "GET /static/blood1.jpg HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:10] "GET /static/images/hospital.jpg HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:10] "GET /static/images/donate.jpg HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:10] "GET /static/images/emergency.jpg HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:11] "GET /about HTTP/1.1" 200 -
117.99.252.130 - - [04/Feb/2026 15:55:11] "GET /favicon.ico HTTP/1.1" 404 -
103.120.251.187 - - [04/Feb/2026 15:55:12] "GET /static/style.css HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:12] "GET /static/logo.png HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:16] "GET /services HTTP/1.1" 200 -
103.120.251.187 - - [04/Feb/2026 15:55:17] "GET /static/style.css HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:17] "GET /static/logo.png HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:20] "GET /login HTTP/1.1" 200 -
103.120.251.187 - - [04/Feb/2026 15:55:20] "GET /static/style.css HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:20] "GET /static/auth.js HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:21] "GET /static/logo.png HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:39] "GET /signup HTTP/1.1" 200 -
103.120.251.187 - - [04/Feb/2026 15:55:39] "GET /static/style.css HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:40] "GET /static/auth.js HTTP/1.1" 304 -
103.120.251.187 - - [04/Feb/2026 15:55:40] "GET /static/logo.png HTTP/1.1" 304 -
```

i-0d8fd24e293e798e9 (AWS console)

PublicIPs: 3.235.63.244 PrivateIPs: 172.31.67.128

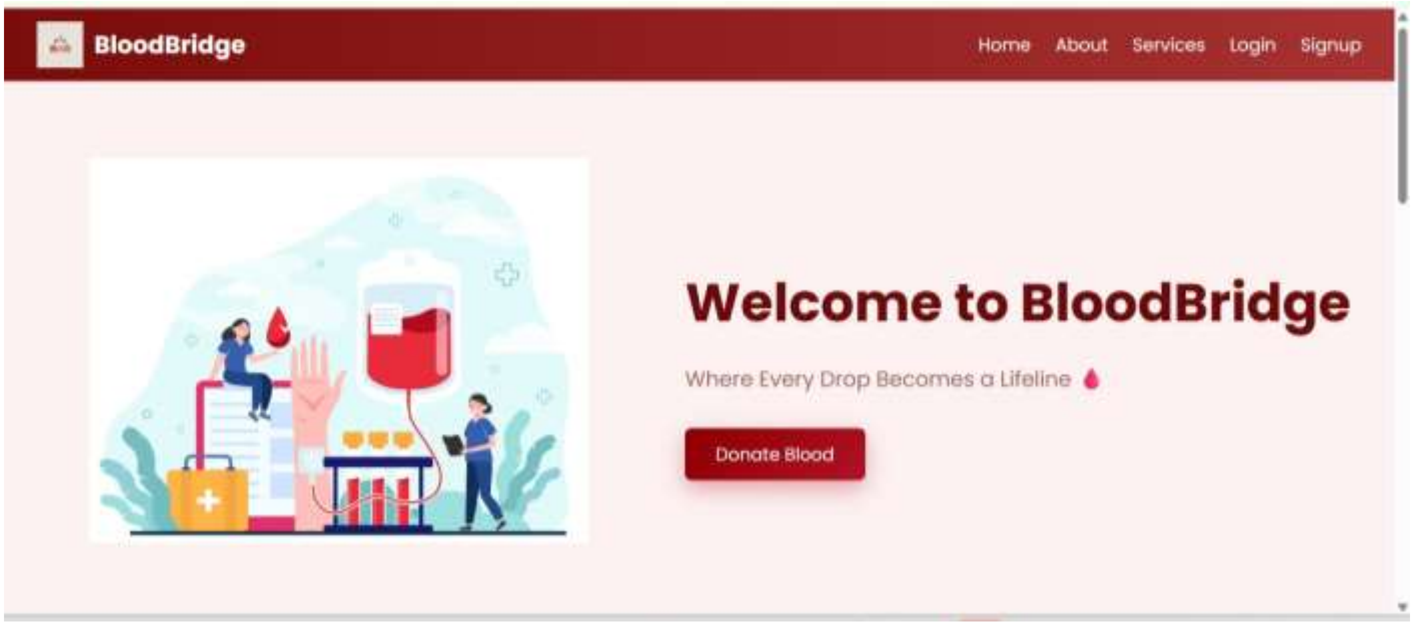
Access the website through:

PublicIPs: https://3.235.63.244:5000

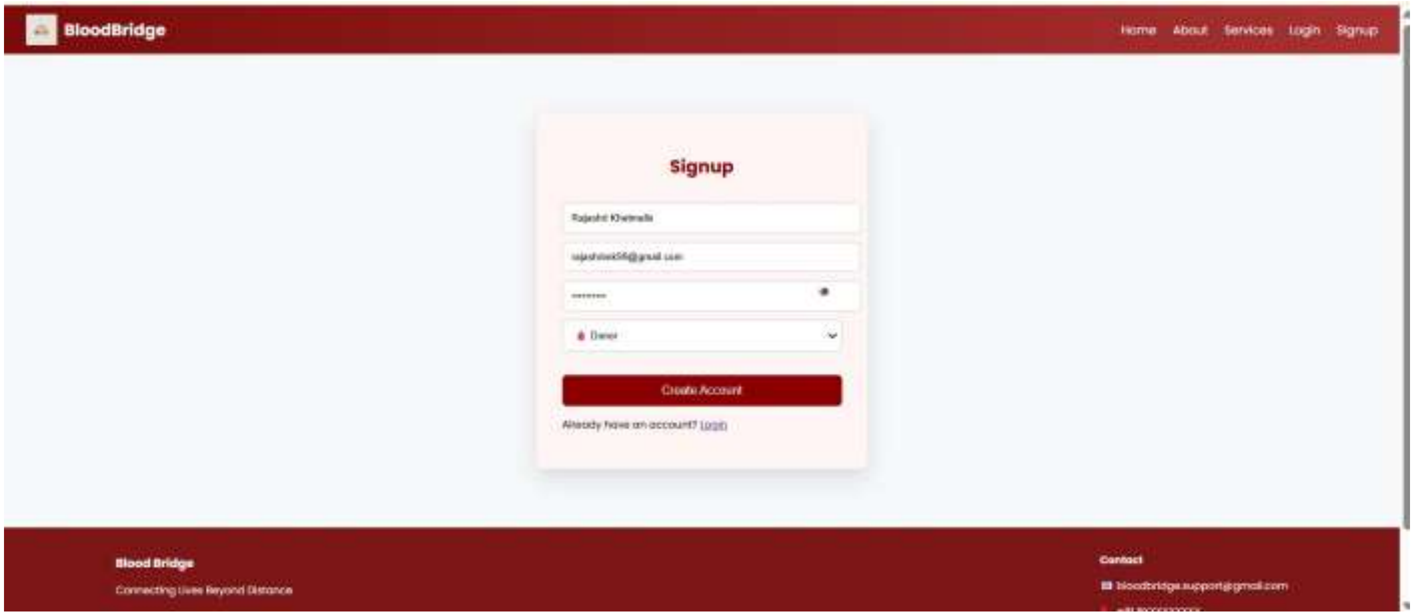
Milestone 8: Testing and Deployment

- **Activity 8.1: Conduct functional testing to verify user registration, login, search results, Booking_Form, and notifications.**

Home Page:



Register Page:



Signup

Rajashri Khatmalle

rajashree[illegible]@gmail.com

••••••••

Donor

Create Account

Already have an account? Login

Blood Bridge

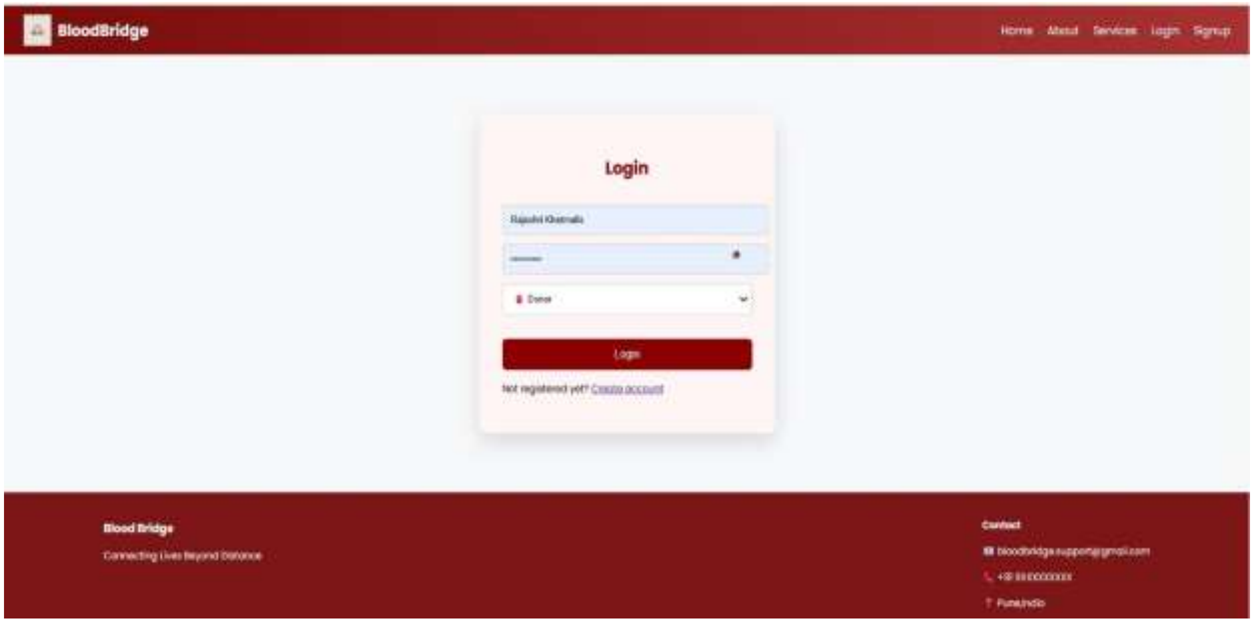
Connecting Lives Beyond Distance

Contact

bloodbridge.support@gmail.com

+91 [illegible]

Login Page:



Login

Rajashri Khatmalle

••••••••

Donor

Login

Not registered yet? Create account

Blood Bridge

Connecting Lives Beyond Distance

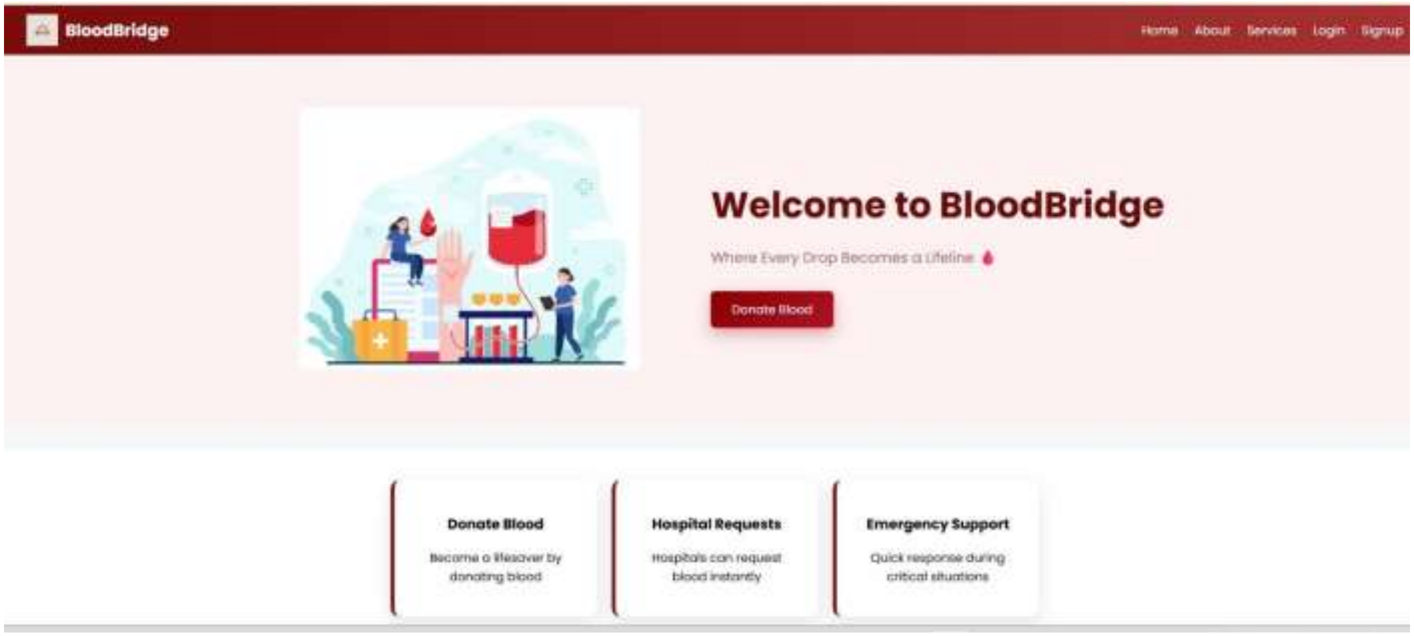
Contact

bloodbridge.support@gmail.com

+91 [illegible]

Pune,India

Home page:



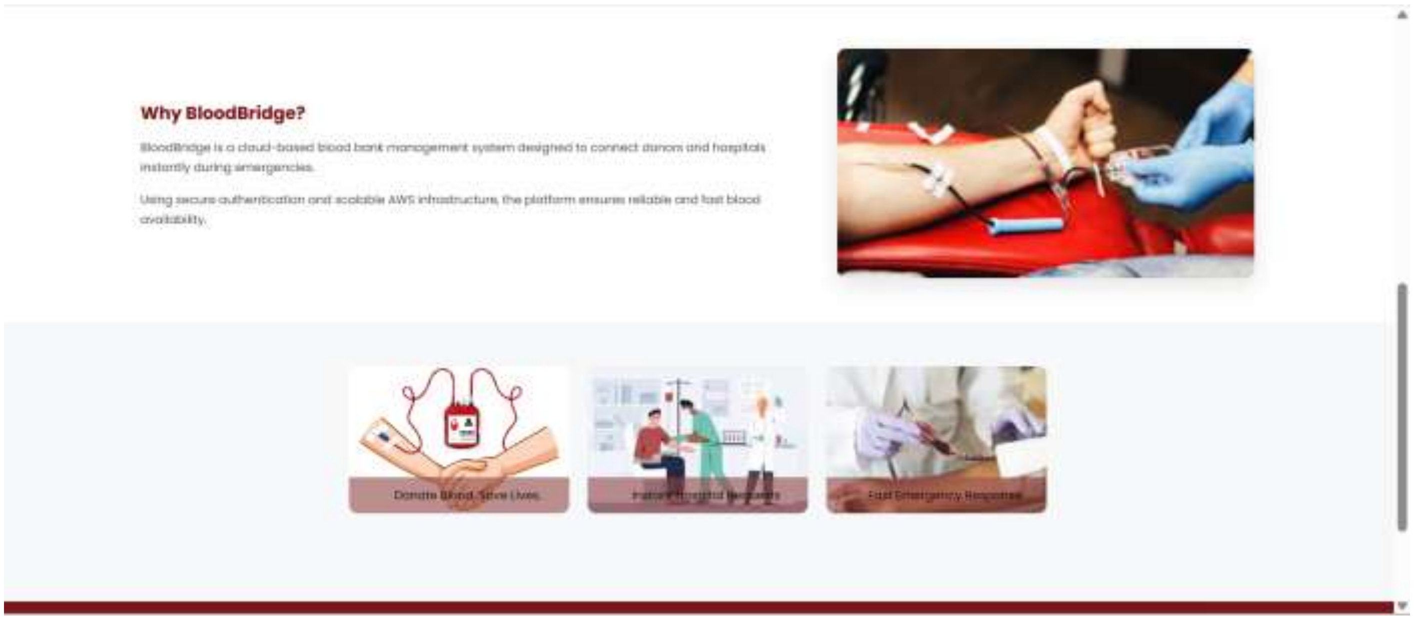
Welcome to BloodBridge

Where Every Drop Becomes a Lifeline 🩸

Donate Blood
Become a lifesaver by donating blood

Hospital Requests
Hospitals can request blood instantly

Emergency Support
Quick response during critical situations

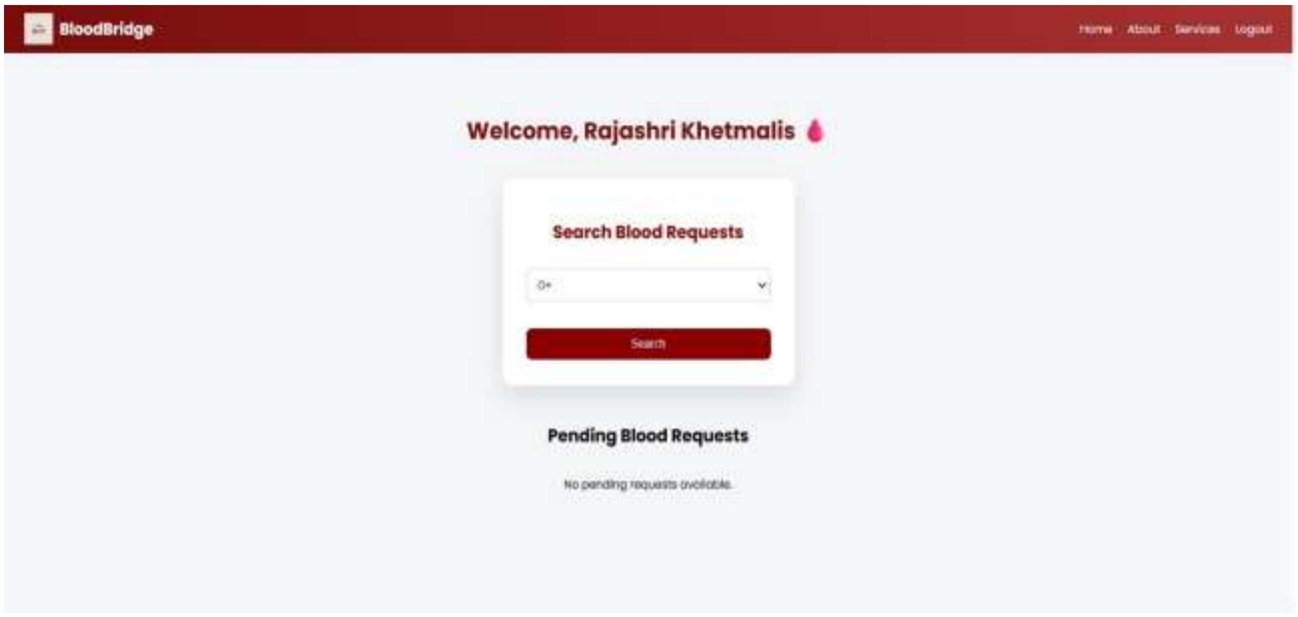


Why BloodBridge?

BloodBridge is a cloud-based blood bank management system designed to connect donors and hospitals instantly during emergencies.

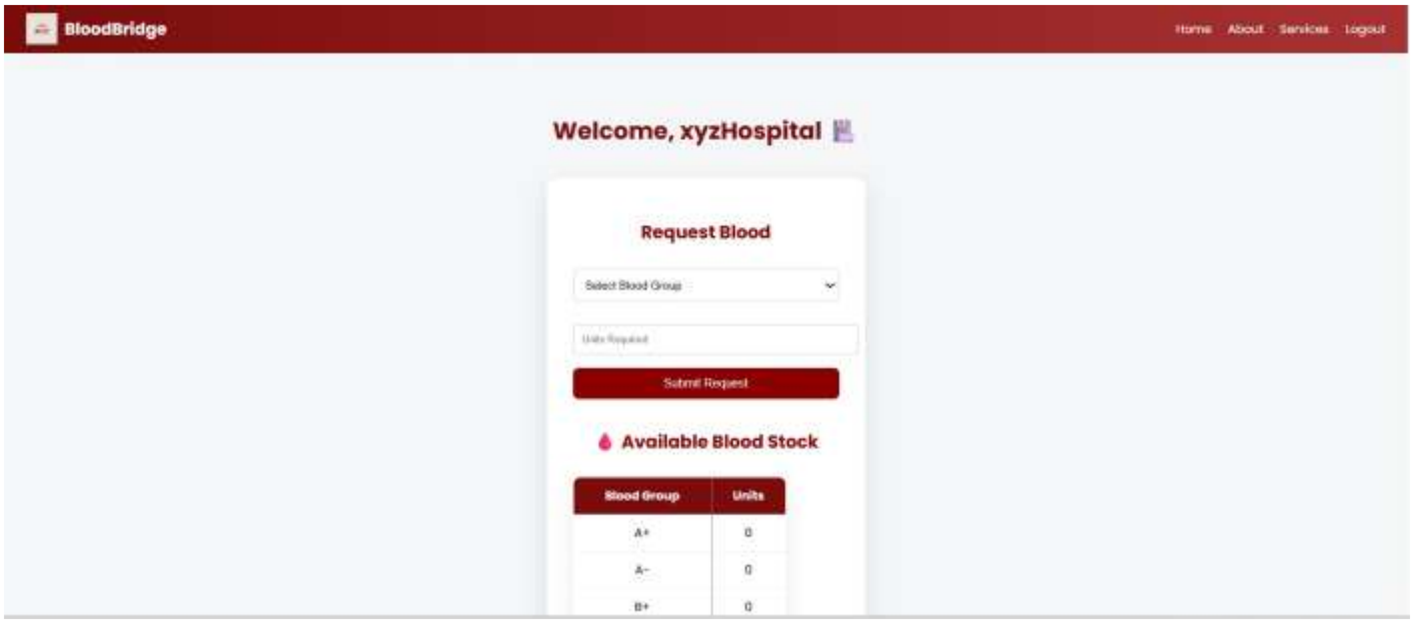
Using secure authentication and scalable AWS infrastructure, the platform ensures reliable and fast blood availability.

Donor Dashboard:



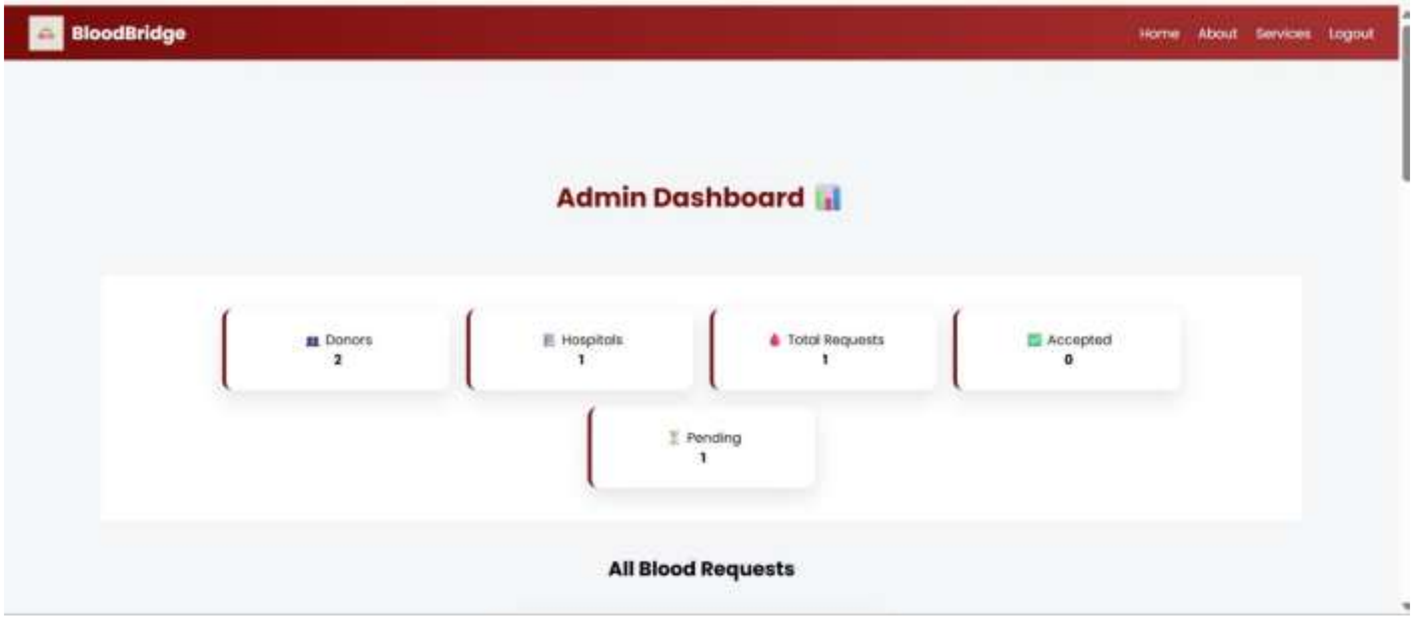
No pending requests available.

Hospital Dashboard:



Blood Group	Units
A+	0
A-	0
B+	0

Admin Dashboard:



BloodBridge

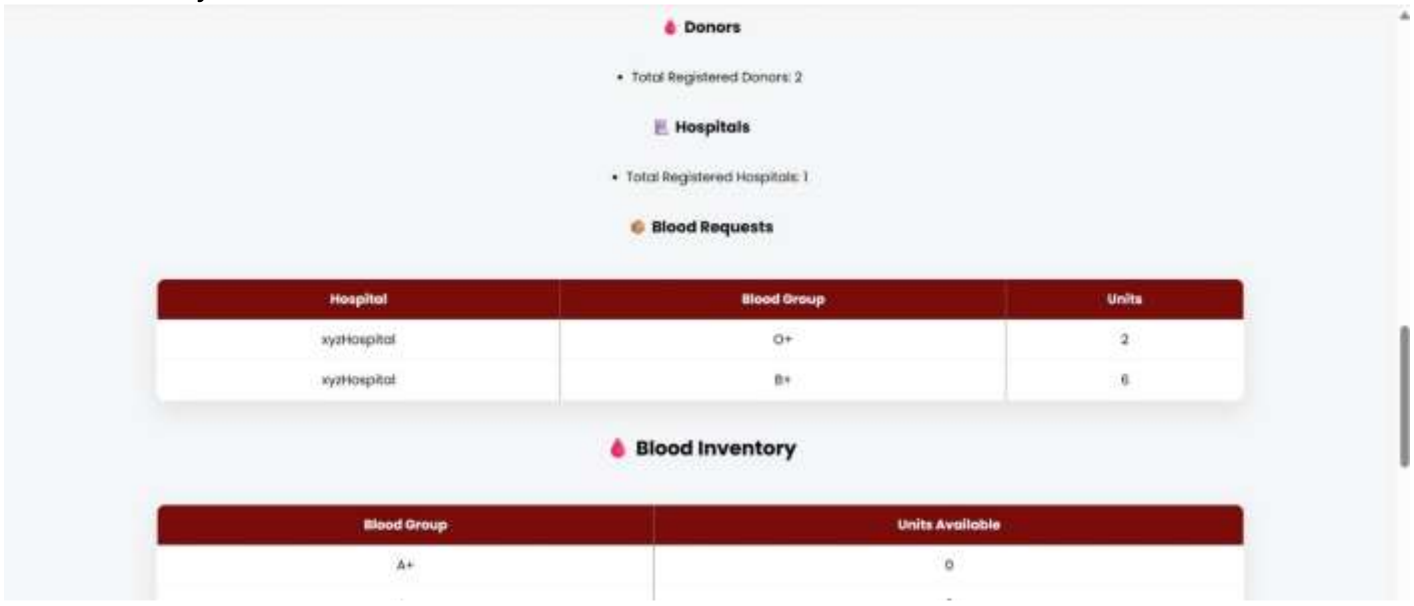
Home About Services Logout

Admin Dashboard 📊

Donors	Hospitals	Total Requests	Accepted	Pending
2	1	1	0	1

All Blood Requests

Blood Invntory:



Donors

- Total Registered Donors: 2

Hospitals

- Total Registered Hospitals: 1

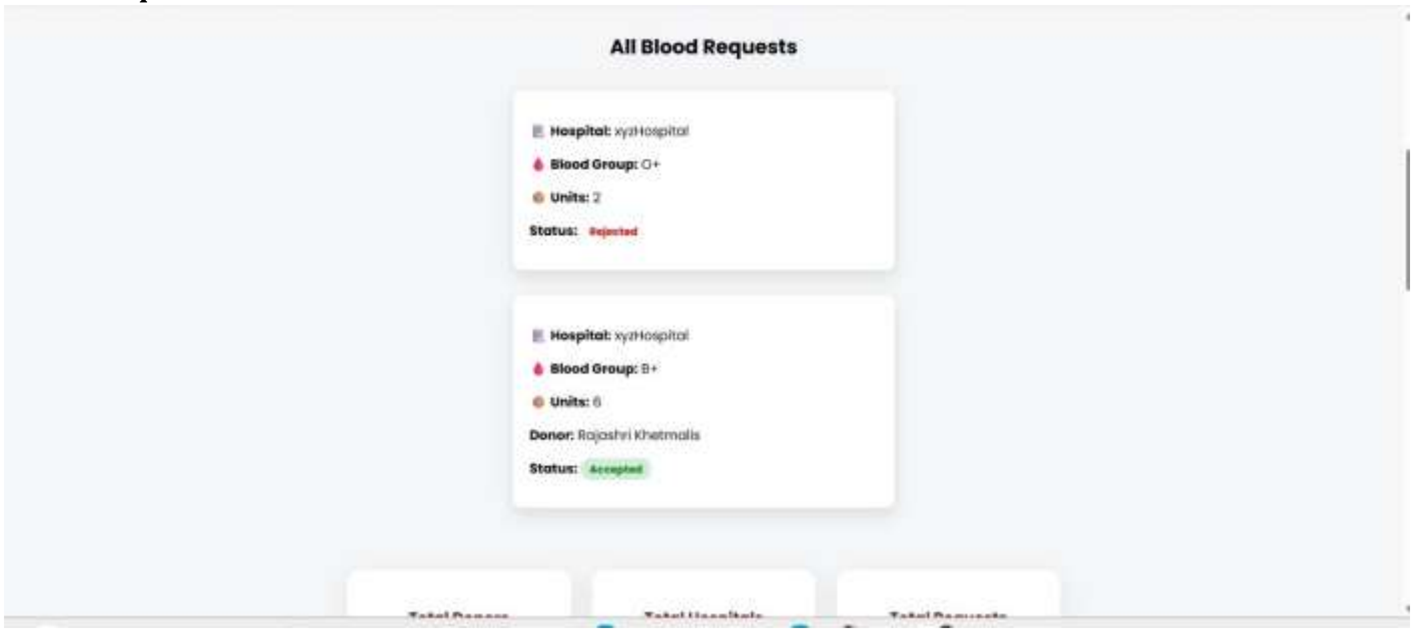
Blood Requests

Hospital	Blood Group	Units
xyzHospital	O+	2
xyzHospital	B+	6

Blood Inventory

Blood Group	Units Available
A+	0

Blood Request:


All Blood Requests

Hospital: xyzHospital

Blood Group: O+

Units: 2

Status: Rejected

Hospital: xyzHospital

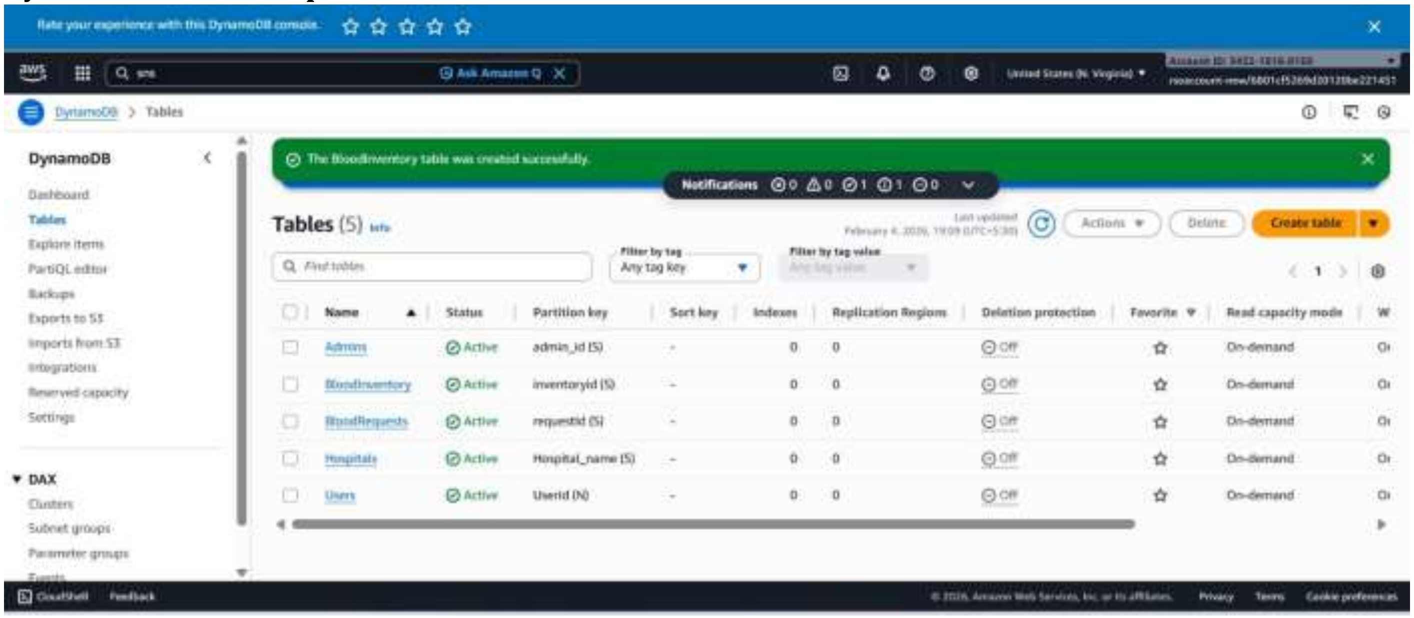
Blood Group: B+

Units: 6

Donor: Rajashri Khetmalis

Status: Accepted

Dynamodb Database updations :

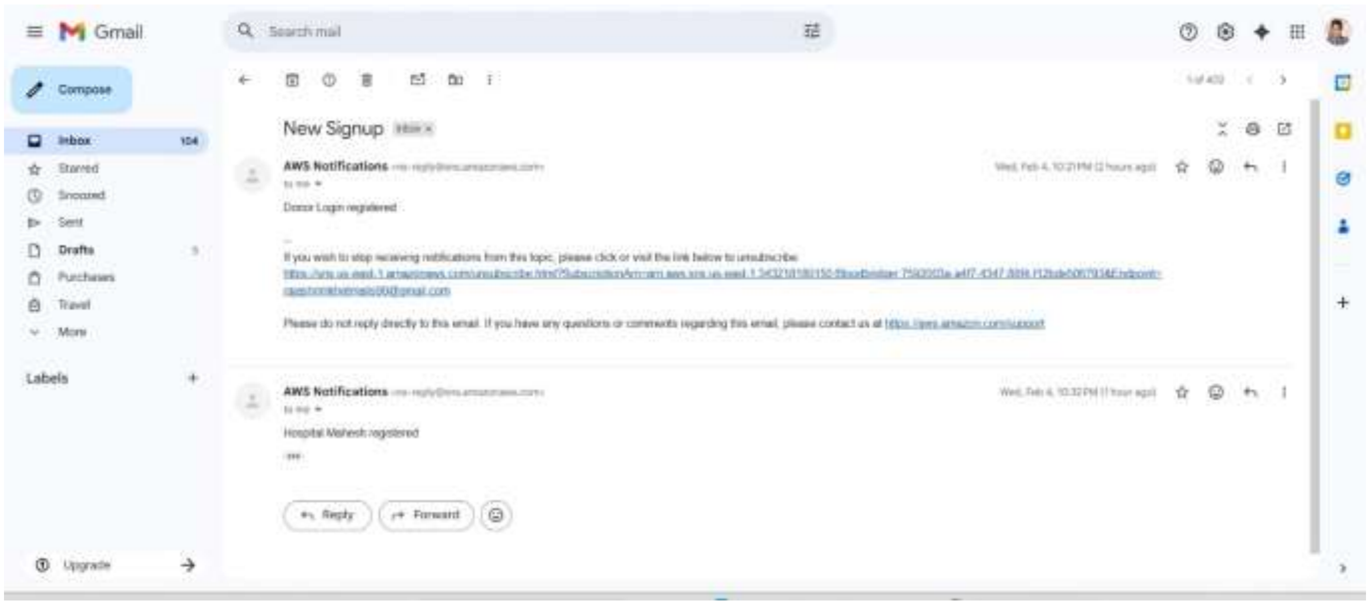


The BloodInventory table was created successfully.

Tables (5)

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode
Admins	Active	admin_id (S)	-	0	0	Off	☆	On-demand
BloodInventory	Active	inventoryid (S)	-	0	0	Off	☆	On-demand
BloodRequests	Active	requestid (S)	-	0	0	Off	☆	On-demand
Hospitals	Active	Hospital_name (S)	-	0	0	Off	☆	On-demand
Users	Active	Userid (N)	-	0	0	Off	☆	On-demand

1. **Mail to the User:**



New Signup

AWS Notifications <no-reply@sns.amazonaws.com>
to me
Wed, Feb 4, 10:21 PM (2 hours ago)

Doctor Login registered

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
[illegible]

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

AWS Notifications <no-reply@sns.amazonaws.com>
to me
Wed, Feb 4, 10:32 PM (1 hour ago)

Hospital Mahesh registered

Conclusion:

The **Blood Bridge** platform has been successfully designed, developed, and deployed using a robust, cloud-native architecture powered by AWS services. By leveraging **Amazon EC2** for application hosting, **Amazon DynamoDB** for real-time data storage, and **AWS SNS** for instant notifications, the system delivers a scalable, reliable, and user-friendly solution for blood bank and donation management.

This application effectively addresses critical challenges in traditional blood bank systems, such as lack of real-time inventory visibility, delayed donor coordination, and inefficient communication between hospitals and donors. Blood Bridge enables seamless donor registration, hospital blood requests, real-time blood stock tracking, and automated notification alerts—all through a centralized web platform.

The cloud-based infrastructure ensures high availability and scalability, allowing the system to handle multiple concurrent users during emergency situations without performance degradation. The integration of the **Flask framework** with AWS services ensures smooth backend operations, including secure authentication, role-based dashboards (Admin, Donor, Hospital), blood request processing, and inventory updates.

Extensive testing confirms that all core functionalities—from user registration and login to blood request approval and notification delivery—operate accurately and efficiently. The clean and responsive user interface further enhances usability, making the system intuitive and accessible for all stakeholders.

In conclusion, **Blood Bridge** demonstrates the effective use of cloud technologies to modernize life-saving healthcare services. It provides a scalable, secure, and efficient solution for managing blood donation and distribution, showcasing the real-world impact and potential of full-stack cloud applications in strengthening healthcare infrastructure and saving lives.