Name :Roshan Saral Kumar(CLOUD-COMPUTING-INTERN)
DATE:1-11-2025

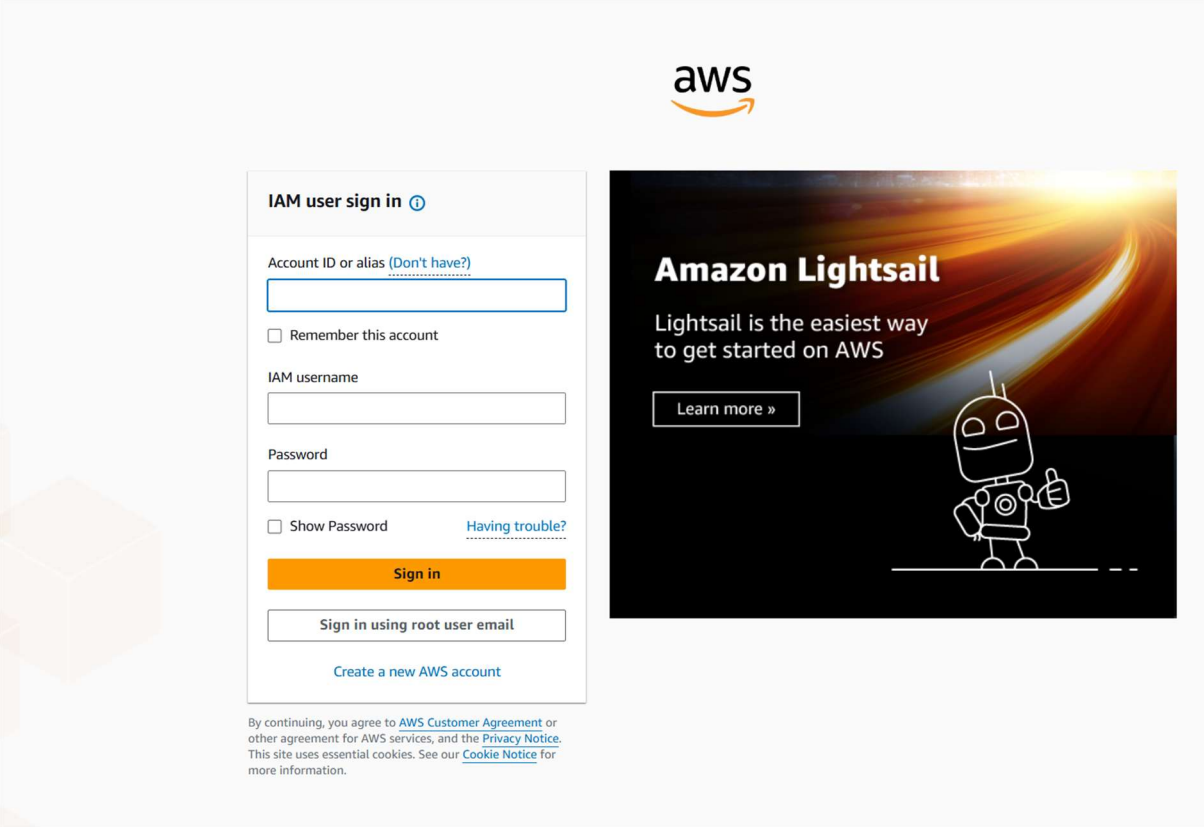
TASK 8

- **Task 8: Host and Deploy a Web Application on the Cloud**

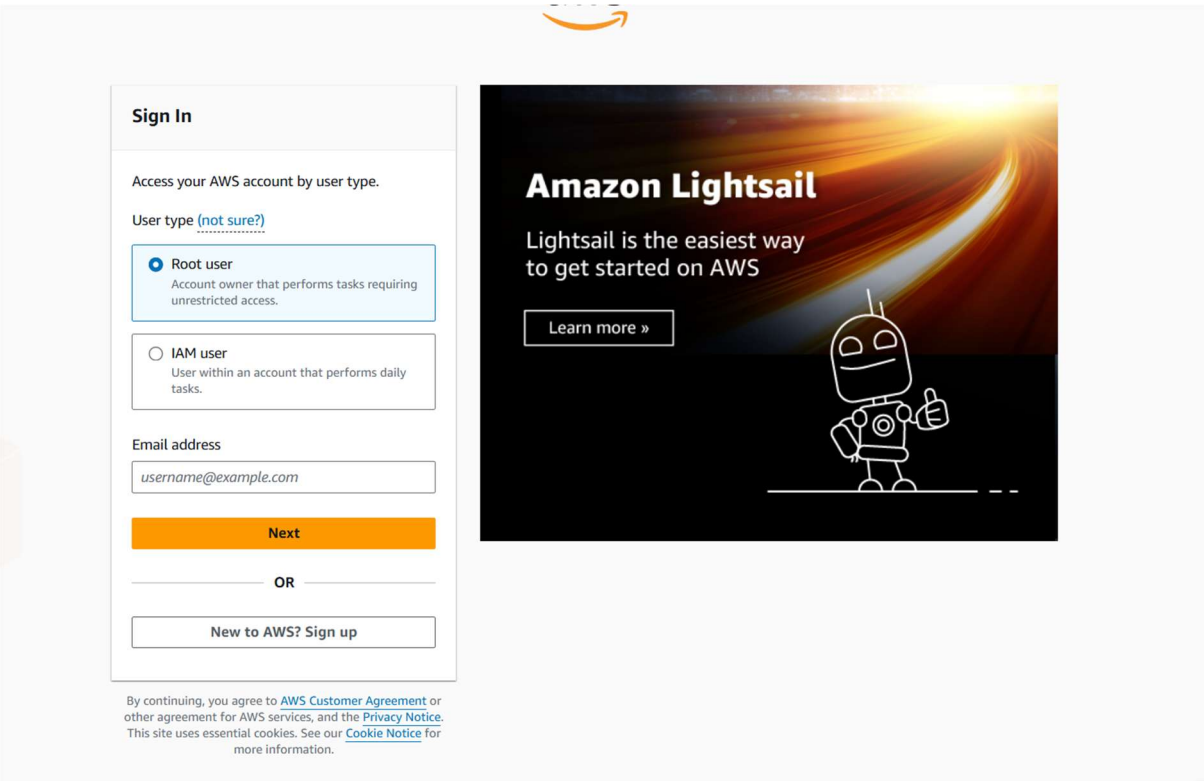
- **Objective:-**
- **To understand containerization and cloud-native deployment by creating a Docker image of a simple web application and deploying it to a cloud platform (Google Cloud Run / AWS ECS / Azure Container Instances).**
- **This teaches how modern applications are made portable, scalable, and easy to deploy.**

- **TOOLS USED FOR AWS (Free Tier):**
- **Docker Desktop (or Docker CLI)**
- **Cloud Platform:**
- **AWS Elastic Container Service (ECS)**
- **Programming Language: Python**
- **Code Editor: VS Code**
- **Terminal**

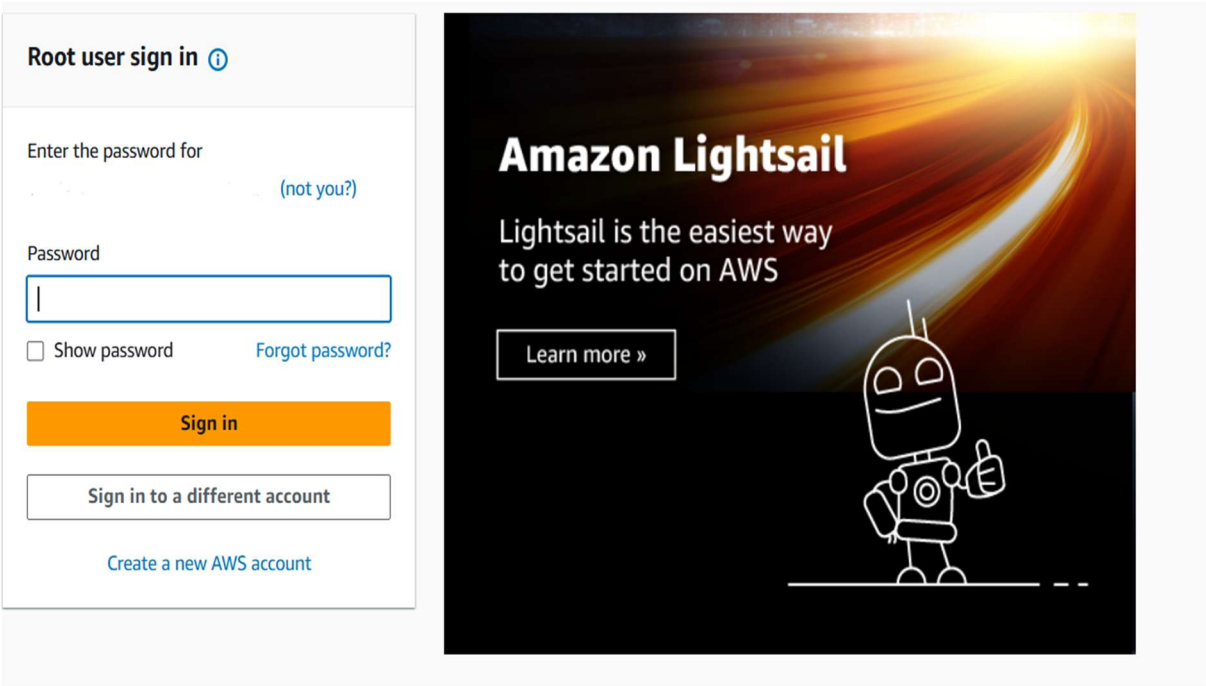
- **THIS IS THE SIGN IN CONSOLE PAGE AND IAM USING SIGN USING ROOT USER EMAIL (ROOT USER MEANS I HAVE ACCESS TO ALL THE SERVICES OF AWS IF IT IS AN IAM (IDENTITY AND ACCESS MANAGEMENT USER THEN THE USER WILL HAVE ACCESS TO ONLY A LIMITED NUMBER OF SERVICES THAT AWS OFFERS AND THE IAM USERS WILL HAVE IAM POLICIES ATTACHED TO THEM SO ONLY THOSE SERVICES THEY WILL BE ABLE TO USE) CURRENTLY LOGGING IN AS ROOT USER EMAIL.**



• AFTER CLICKING ON SIGN IN USING ROOT USER EMAIL I WLL GET THIS PAGE WHERE IHAVE TO ENTER MY EMAIL-ID. AND AS WE CAN SEE ROOT USER IS SELECTED BY DEFAULT.WE CAN ALSO SELECT IAM USER IN THIS PAGE IN CASE IF THERE IS A CHANGE.



- **AFTER ENTERING OUR EMAIL WE NEED TO GIVE OUR PASSWORD**



- **SINCE I HAVE AN AWS ACCOUNT ALREADY CREATED I ACTUALLY MADE AN MFA FOR MY ROOT USER . MFA STANDS FOR (MULTI-FACTOR AUTHENTICATION CODE) IT ACTS AS A DOUBLE LAYER PROTECTION FOR OUR ROOT USER OTHERWISE WE CAN GET HACKED SO ALREADY PROVIDE AN EXTRA LAYER OF PROTECTION TO BE ON THE SAFER SIDE.**

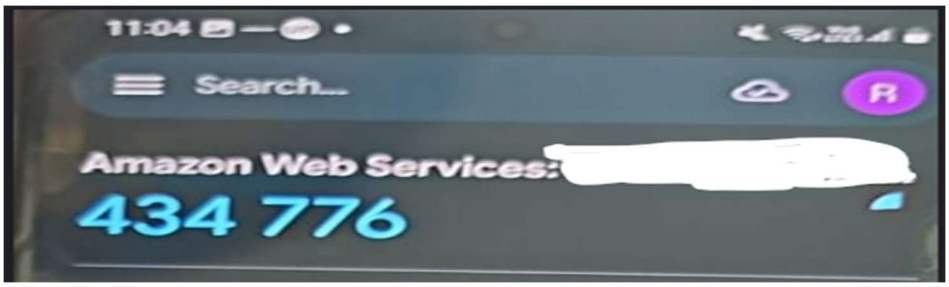
- MFA CODE IS GIVEN BY TOTEM TOKENS OR U CAN ALSO USE THE “GOOGLE AUTHENTICATOR APP” IN YOUR MOBILE WHICH IS MOSTLY PREFFERED.

- THIS APP IS IN MY MOBILE AND I HAVE TO GIVE THE MFA CODE WHICH IS ASKED FOR THE VERIFICATION OF ROOT USER.

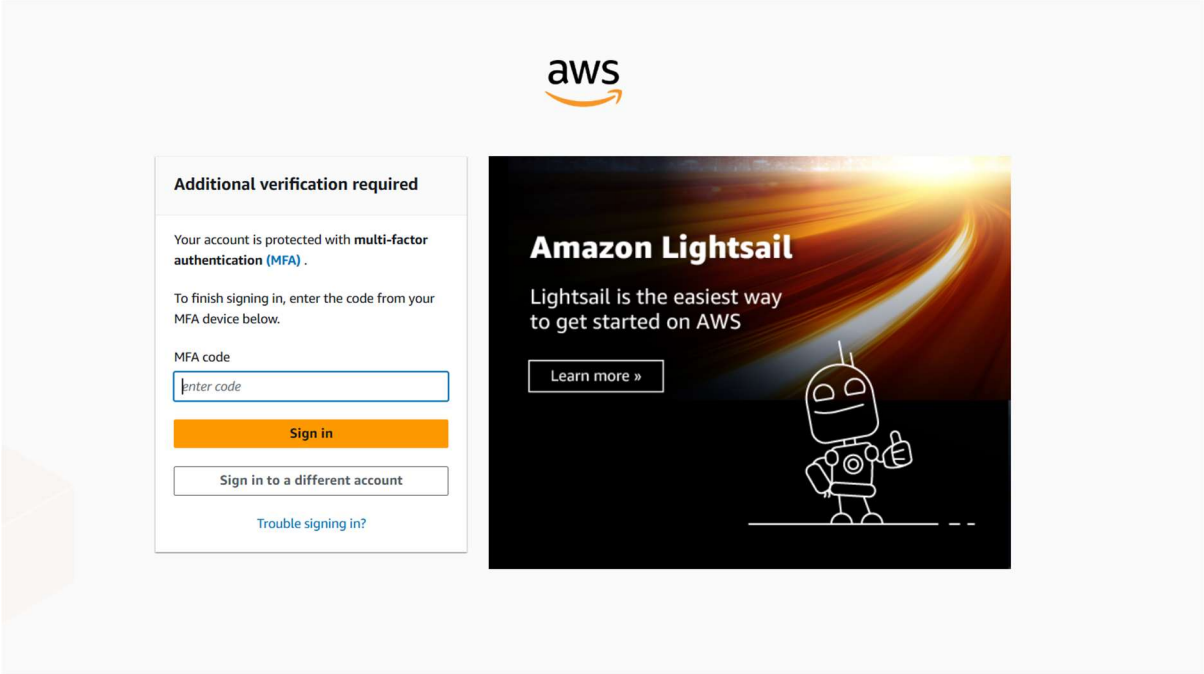
- ONCE I HAVE CLICKED ON THIS APP I WILL SEE THE BELOW SCREEN IN MY MOBILE.

- **THE SCREEN IN MY MOBLIE LOOKS LIKE THIS:-THE CONTENT WHICH IAM HIDING USING A WHITE COLOR IS MY PRIVATE INFORMATION OF MY ACCOUNT WHICH CANNOT BE SHARED BUT HIS IS HOW A SIMPLE MFA CODE**

LOOKS LIKE AND THE MFA CODE KEEPS ON CHANGING ON A DAILY BASIS.ALWAYS USE MFA TO ENSURE ENHANCED SECURITY.



- **THIS IS THE CODE THAT I HAVE TO ENTER IN MY AWS WEB BROWSER IN THE BELOW IMAGE IT IS SHOWN.**



- **AFTER SIGINING IN WE ENTER THE AWS MANAGEMENT CONSOLE AND THE DFAULT REGION IS NORTH VIRGINIA THAT IS "us-east-1" WE CAN SELECT DIFFERENT REIGIONS. IN THE IMAGE IT IS CLEARLY SHOWN IN THE RIGHT HAND SIDE WHICH REGION IAM CURRENTLY WORKING IN. THE BELOW IMAGE SHOWS THE SERVICES WHICH I VISITED AND THE CURRENT REGION WHICH IAM WORKING IN.**

- **TERMINOLOGIES:-**

- **Region:- It is a geothermal location which consists of a group of AZs (AVAILABILITY ZONES) and the network of a region is called a VPC(VIRTUAL PRIVATE CLOUD).**

- **AZ(Availability Zone):- they are a group of data centers.the network of a Az is a subnet.**

- **Edge Location:- it is a location in between various Azs or it is found at the boundary of an az In order to cache the incoming data acts like a storage for easier retrieval**.

PLS NOTE CHANGE STARTS FROM HERE BELOW:-

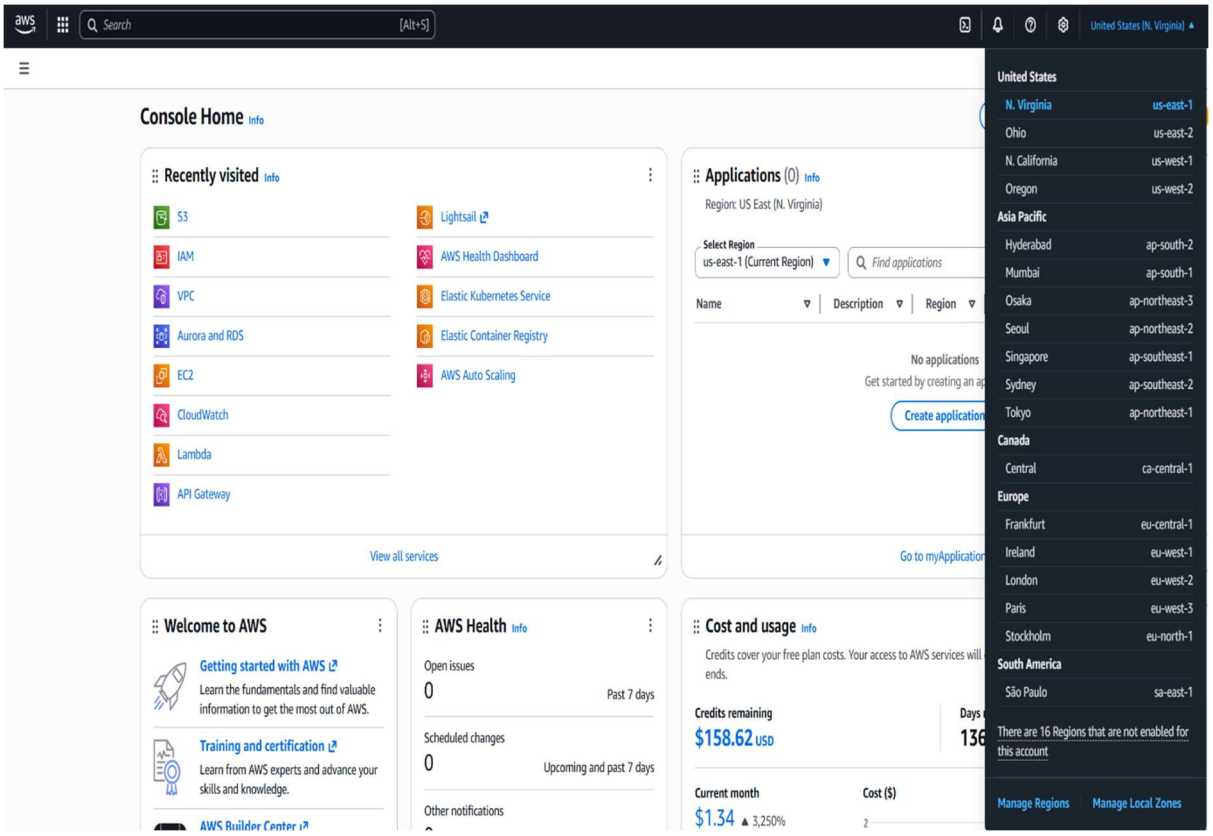
- NOTE:-**WE ARE WORKING IN THE PUBLIC SUBNET THAT IS "us-east-1"** if we are working in any azs it will show us—east-1a,us-east-1b and so on upt to us-east-1f. We can check the azs in our aws management console itself that is given in below figure.

SO NOW FOR THE deployment of web application SETTINGS IAM CURRENTLY WORIKING IN THE NORTH VIRGINIA REGION SO IAM USING "us-east-1".

IF WE CLICK ON VPC:-
HERE WE WILL BE ABLE TO SEE THE SUBNETS IN THE VPC DASHBOARD IF WE CLICK ON SUBNETS THEN TOTALLY THERE ARE 6 SUBNETS THAT IS PRIVATE SUBNETS FROM "us-east-1a" to" us-east-1f" totally we have 6 private subnets or we can also say we have 6 azs under NORTH VIRGINIA REGION THAT IS "US-EAST-1" REGION. This is provided by aws itself by default we cannot create any Availability Zones.

Logging into aws amanagement console and searching for VPC(VIRTUAL PRIVATE CLOUD)

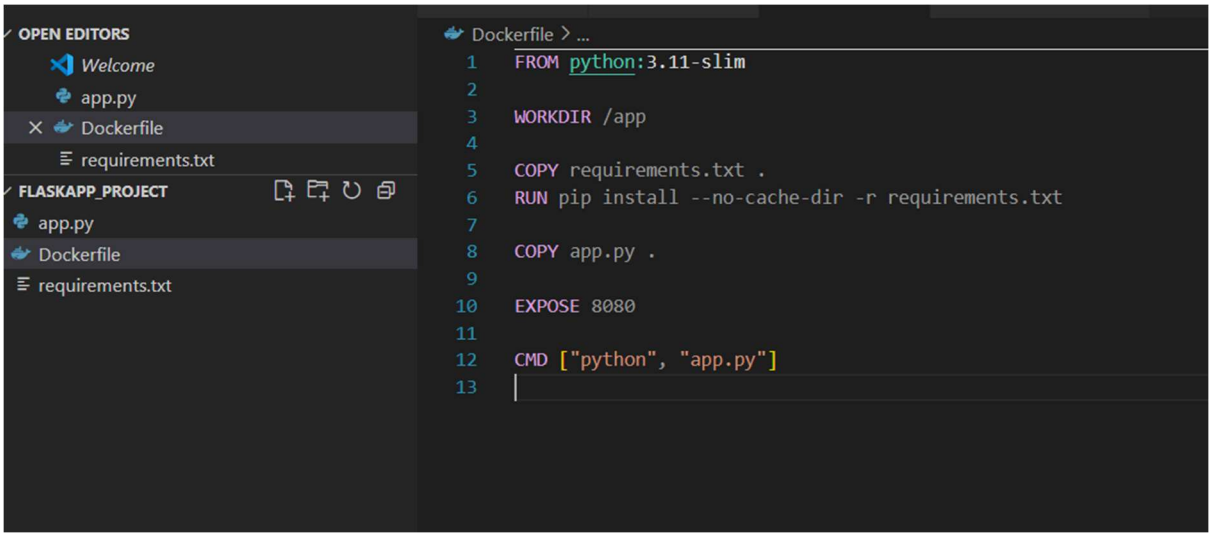
THIS IS THE AWS MANAGEMENT CONSOLE AFTER LOGGING IN CURRENTLY WORKING IN NORTH VIRGINIA REGION(US-EAST-1) the below image shows that



Step 1 :- write all the dockerfile,flask app , requirements.txt files in the vs code

Code:-

Docker file:-



FROM python:3.11-slim

WORKDIR /app

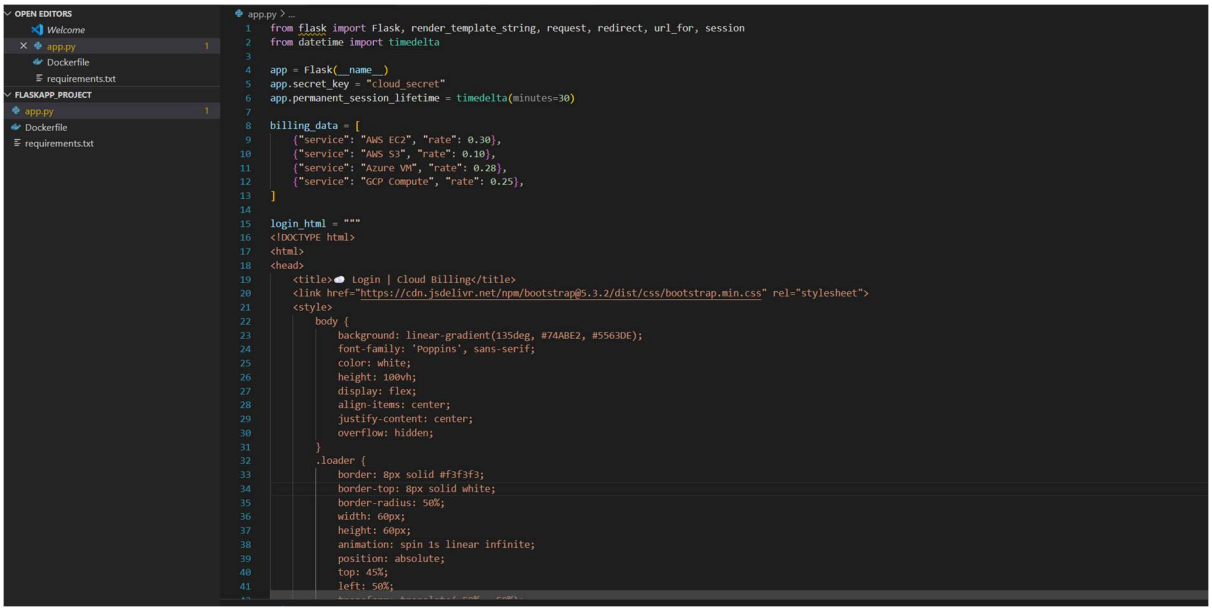
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY app.py .

EXPOSE 8080

CMD ["python", "app.py"]

PYTHON FILE WHICH CONTAINS THE FLASK APPLICATION:-




```
from flask import Flask, render_template_string, request, redirect, url_for, session
from datetime import timedelta

app = Flask(__name__)
app.secret_key = "cloud_secret"
app.permanent_session_lifetime = timedelta(minutes=30)

billing_data = [
    {"service": "AWS EC2", "rate": 0.30},
    {"service": "AWS S3", "rate": 0.10},
    {"service": "Azure VM", "rate": 0.28},
    {"service": "GCP Compute", "rate": 0.25},
]

login_html = """
<!DOCTYPE html>
<html>
<head>
    <title> Login | Cloud Billing</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            background: linear-gradient(135deg, #74ABE2, #5563DE);
            font-family: 'Poppins', sans-serif;
```

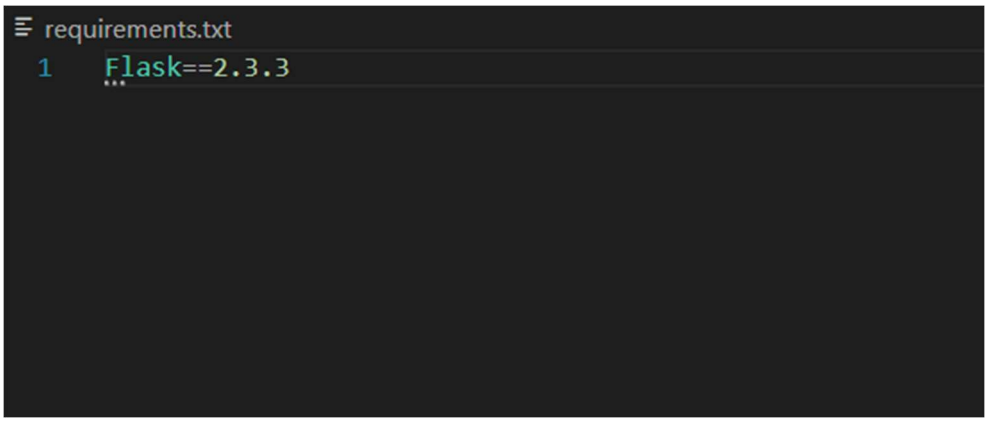
```
            color: white;
            height: 100vh;
            display: flex;
            align-items: center;
            justify-content: center;
            overflow: hidden;
        }
        .loader {
            border: 8px solid #f3f3f3;
            border-top: 8px solid white;
            border-radius: 50%;
            width: 60px;
            height: 60px;
            animation: spin 1s linear infinite;
            position: absolute;
            top: 45%;
            left: 50%;
            transform: translate(-50%, -50%);
        }
        @keyframes spin {
            0% { transform: rotate(0deg); }
            100% { transform: rotate(360deg); }
        }
        .card {
            display: none;
            animation: fadeIn 1s ease-in;
        }
        @keyframes fadeIn {
            from { opacity: 0; transform: translateY(-20px); }
            to { opacity: 1; transform: translateY(0); }
        }
    </style>
</head>
<body>
    <div class="loader"></div>
    <div class="card bg-light text-dark p-4 rounded shadow" style="width: 22rem;">
        <h4 class="mb-3 text-center">☁️ Cloud Billing Login</h4>
        {% if error %}
        <div class="alert alert-danger">{{ error }}</div>
        {% endif %}
        <form method="POST">
            <input type="text" name="username" class="form-control mb-2" placeholder="Username"
required>
            <input type="password" name="password" class="form-control mb-3"
placeholder="Password" required>
            <button type="submit" class="btn btn-primary w-100">Login</button>
        </form>
    </div>
    <script>
        setTimeout(() => {
            document.querySelector('.loader').style.display = 'none';
            document.querySelector('.card').style.display = 'block';
        }, 2000);
```

```
    </script>
</body>
</html>
"""

dashboard_html = """
<!DOCTYPE html>
<html>
<head>
    <title>☁️ Dashboard | Cloud Billing</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            background: linear-gradient(135deg, #e0ecff, #ffffff);
            font-family: 'Poppins', sans-serif;
        }
        .table-container {
            margin-top: 40px;
        }
        .total {
            font-size: 1.2rem;
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-dark bg-primary px-4">
        <span class="navbar-brand mb-0 h1">☁️ Cloud Billing Dashboard</span>
        <form action="/logout" method="get">
            <button class="btn btn-light">Logout</button>
        </form>
    </nav>

    <div class="container table-container">
        <table class="table table-bordered table-hover">
            <thead class="table-primary">
                <tr>
                    <th>Service</th>
                    <th>Rate ($/hr)</th>
                    <th>Quantity</th>
                </tr>
            </thead>
            <tbody>
                {% for row in billing_data %}
                <tr>
                    <td>{{ row.service }}</td>
                    <td>{{ row.rate }}</td>
                    <td><input type="number" min="0" value="0" class="form-control"
onchange="updateTotal()"></td>
                </tr>
                {% endfor %}
            </tbody>
```

```
    </table>
    <div class="total text-center">Total Cost: $<span id="total">0.00</span></div>
  </div>

  <script>
    const rates = {{ billing_data | tojson }};
    function updateTotal() {
      let total = 0;
      const inputs = document.querySelectorAll("input[type='number']");
      inputs.forEach((input, i) => {
        const qty = parseFloat(input.value) || 0;
        total += qty * rates[i].rate;
      });
      document.getElementById("total").innerText = total.toFixed(2);
    }
  </script>
</body>
</html>
"""

logout_html = """
<!DOCTYPE html>
<html>
<head>
  <title>☁️ Goodbye</title>
  <style>
    body {
      background: linear-gradient(135deg, #74ABE2, #5563DE);
      font-family: 'Poppins', sans-serif;
      color: white;
      height: 100vh;
      margin: 0;
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
    }
    .cloud {
      font-size: 2rem;
      background: white;
      color: #5563DE;
      padding: 30px;
      border-radius: 50px;
      box-shadow: 0 0 20px rgba(0,0,0,0.2);
      animation: float 3s ease-in-out infinite;
      margin-bottom: 20px;
    }
    @keyframes float {
      0% { transform: translateY(0px); }
      50% { transform: translateY(-20px); }
      100% { transform: translateY(0px); }
    }
    iframe {
```

```
            border-radius: 15px;
            box-shadow: 0 0 15px rgba(0,0,0,0.3);
        }
    </style>
</head>
<body>
    <div class="cloud">☁️ Thanks for using my Cloud Billing Dashboard!</div>
    <iframe width="560" height="315"
src="https://www.youtube.com/embed/bbJUcwvKCsA?autoplay=1&loop=1&playlist=bbJUcwvKCs
A&mute=1" frameborder="0" allow="autoplay; encrypted-media" allowfullscreen></iframe>
</body>
</html>
"""

@app.route("/", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        user = request.form["username"]
        password = request.form["password"]
        if user == "admin" and password == "cloud123":
            session.permanent = True
            session["user"] = user
            return redirect(url_for("dashboard"))
        return render_template_string(login_html, error="Invalid credentials")
    return render_template_string(login_html)

@app.route("/dashboard")
def dashboard():
    if "user" not in session:
        return redirect(url_for("login"))
    return render_template_string(dashboard_html, billing_data=billing_data)

@app.route("/logout")
def logout():
    session.pop("user", None)
    return render_template_string(logout_html)

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=8080)
```

RQUIREMENTS.TXT FILE:-



Flask==2.3.3

Step 2:- create a ecr repository before going for deployment.

TO CREATE THE REPOSITORY IN ECR (ELASTIC CONTAINER REGISTERY:-
I CANNOT SHARETHE OUTPUT BCZ IT CONSISTS OF PERSONAL DETAILS.

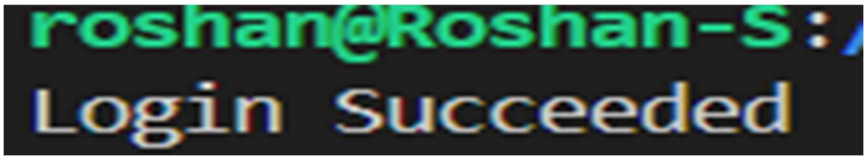
THE BELOW SHOWS THE COMMAND USED:-

aws ecr create-repository --repository-name flaskproductionapp --region us-east-1

AUTHENTICATING DOCKER TO AWS ECR COMMAND:-

aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin <account id>.dkr.ecr.us-east-1.amazonaws.com/<repo-name>

OUTPUT:-



NOW AFTER THIS WE HAVE TO BUILD OUR DOCKER IMAGE(COMMAND USED IS):-

docker build -t flaskproductionapp .

output:-

```
roshan@Roshan-S:/mnt/c/Users/rosha/flaskapp_project$ docker build -t flaskproductionapp .
[+] Building 2.6s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 217B
 => [internal] load metadata for docker.io/library/python:3.11-slim
 => [auth] library/python:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/python:3.11-slim@sha256:8eb5fc663972b871c528fef04be4eaa9ab8ab4539a5316c4b8c133771214a617
 => => resolve docker.io/library/python:3.11-slim@sha256:8eb5fc663972b871c528fef04be4eaa9ab8ab4539a5316c4b8c133771214a617
 => [internal] load build context
 => => transferring context: 7.24kB
 => CACHED [2/5] WORKDIR /app
 => CACHED [3/5] COPY requirements.txt .
 => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt
 => CACHED [5/5] COPY app.py .
 => exporting to image
 => => exporting layers
 => => exporting manifest sha256:e0e085a62264f3e901164fa40caaf5ac62972665120fb4c59a620eeb3a86c0f7
 => => exporting config sha256:069aeeb355bf0a56aa4cf986e012930d774cb7e86fc2d0d8f2e3c7824cb6d82e
 => => exporting attestation manifest sha256:8c3539a9e46fbaa946907dc7a59ff9bef16c6e661cf6794ebcef591813d9b2ed
 => => exporting manifest list sha256:4275e4a71fa50572a9f36ab1e80addcc114572a8448e64128918ddb0e8752f89
 => => naming to docker.io/library/flaskproductionapp:latest
 => => unpacking to docker.io/library/flaskproductionapp:latest
```

NEXT I WILL TAG THE IMAGE FOR ECR:-

docker tag flaskproductionapp:latest <your-account-id>.dkr.ecr.us-east-1.amazonaws.com/flaskproductionapp:latest

next I will push the image to ecr:-

docker push <your-account-id>.dkr.ecr.us-east-1.amazonaws.com/flaskproductionapp:latest

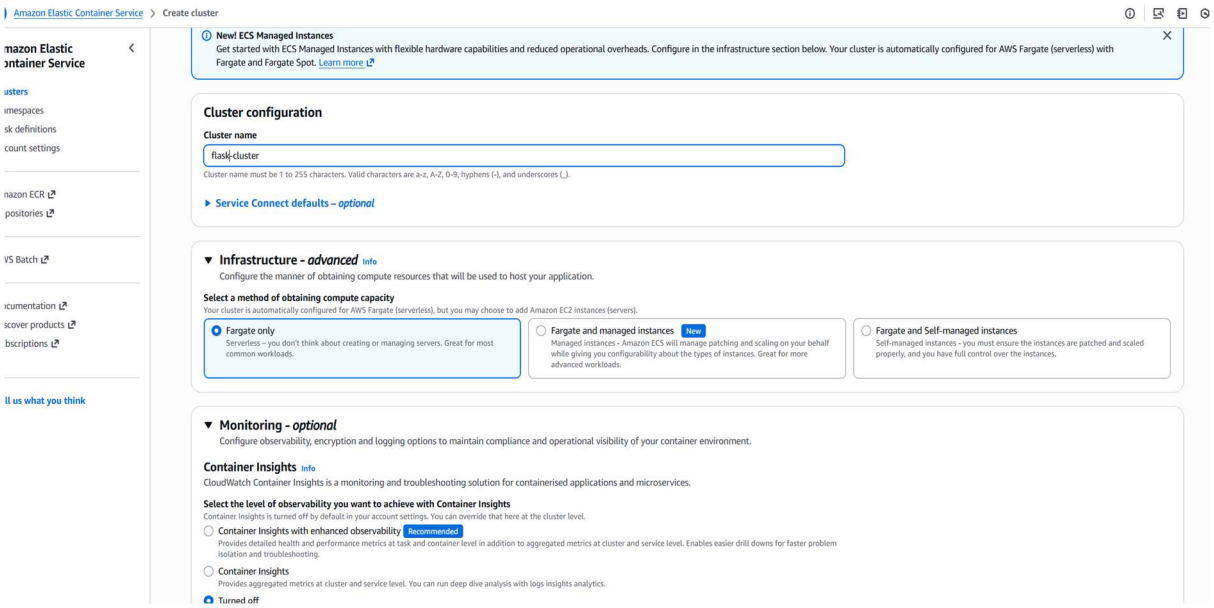
OUTPUT:-

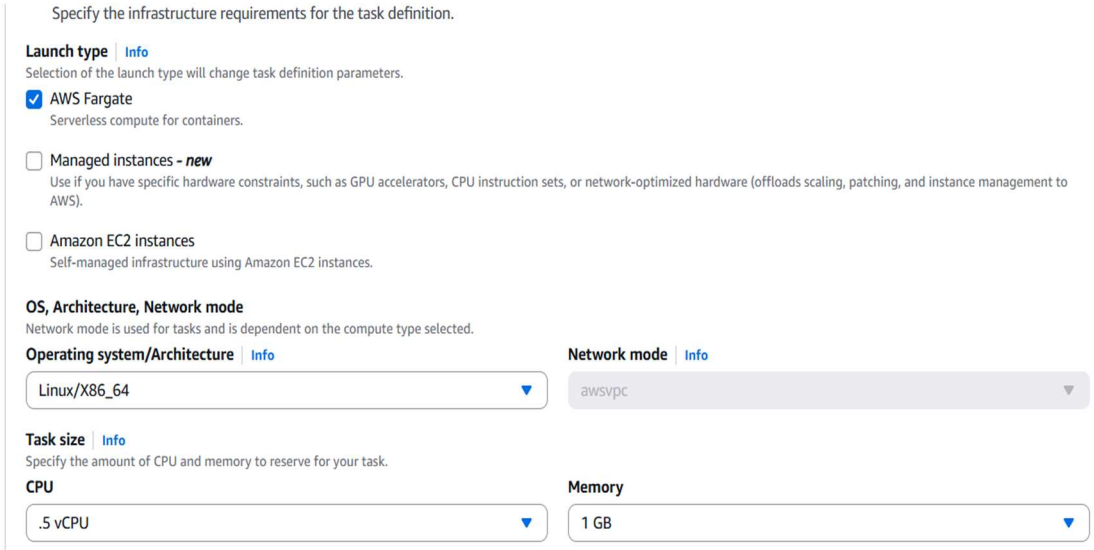
```
oshan@Roshan-S:/mnt/c/Users/r
he push refers to repository
eca3b45c2a0: Pushed
daa4f9aedb5: Pushed
352dfa3f357: Pushed
c645de2d9ac: Pushed
db477cf87de: Pushed
73850a50582: Pushed
9fb8589da02: Pushed
9ffe18d7fdb: Pushed
8513bd72563: Pushed
atest: digest: sha256:4275e4a
```

It has pushed all the layers of the image to the ecr or the repo which I created for ecr

Step 3:- AFTER THIS WE HAVE TO SEARCH FOR ECS(ELASTIC CONTAINER SERVICE) AND WE HAVE TO CREATE A CLUSTER FOR FLASKAPPPRODUCTION.

Cluster name is :- flask-cluster and iam using fargate only.next iam going to create it.





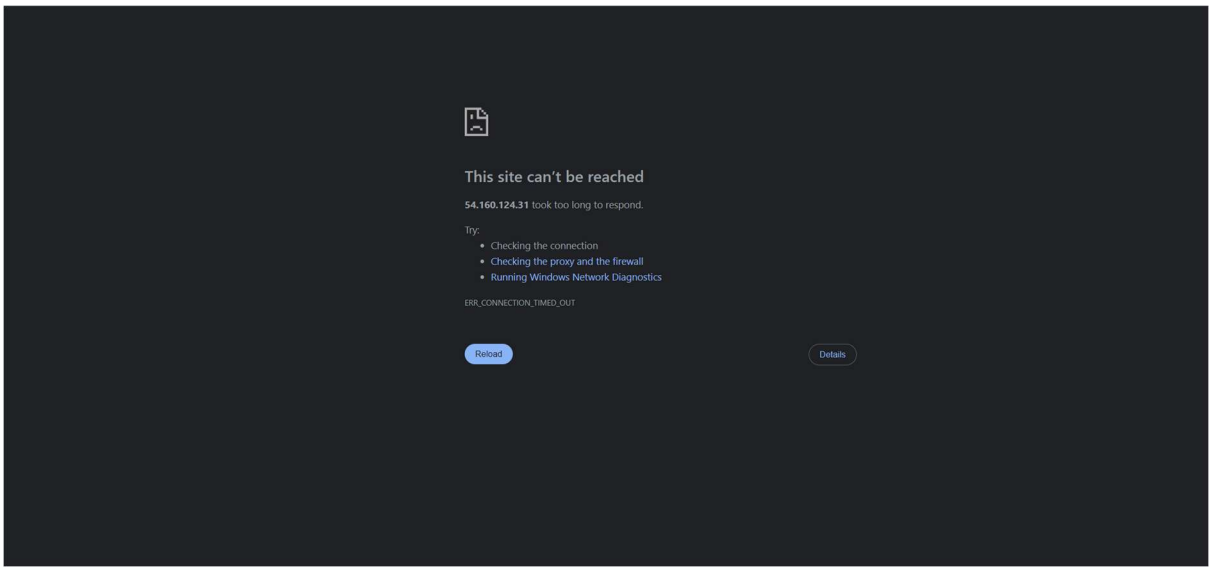
The memory and cpu gives a huge cost so always take the minimum that is 0.5vcpu and igb memory. To reduce the cost.

Here some few things to consider one is the security group handling was very important whci hiwas facing as a challenge but I was bale to resolve it and was able to allow port 8080 traffic initillay the application ws not running bcz there was nothing open to port 80 instead by entire application nedded to run in port 8080.

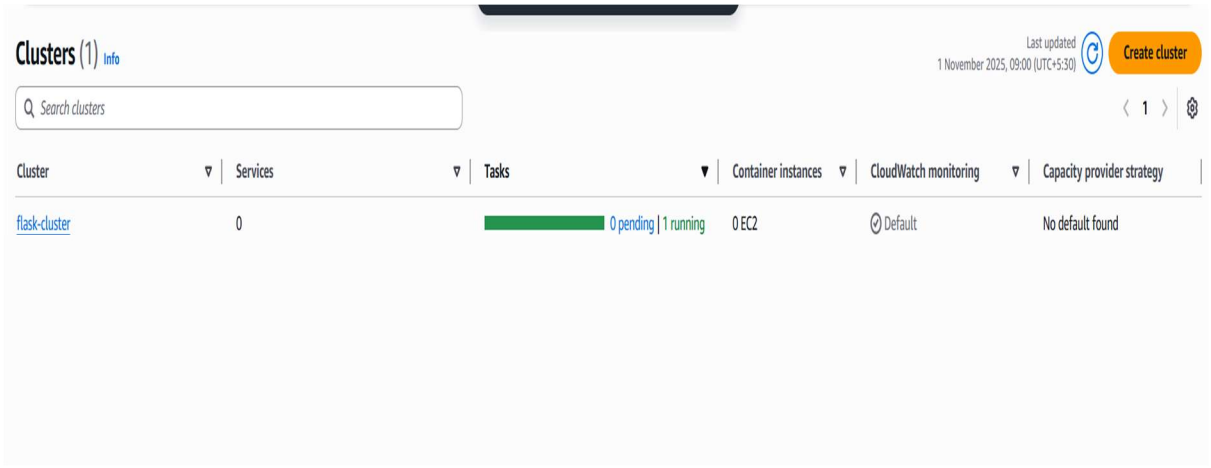
So always choose the right container and application port otherwise the application wil not work and this scenario happened to me yesterday.today I fixed it.

The below figure was the issue and I fixed it using security groups and allowing inbound rule access

for port 8080.

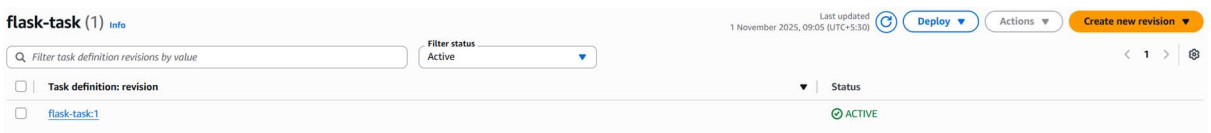


This is the ecs cluster which is created.



The security group was not mentioned in my tasks that is why the issue persisted.

This is the flask-task which was created for the ecs.



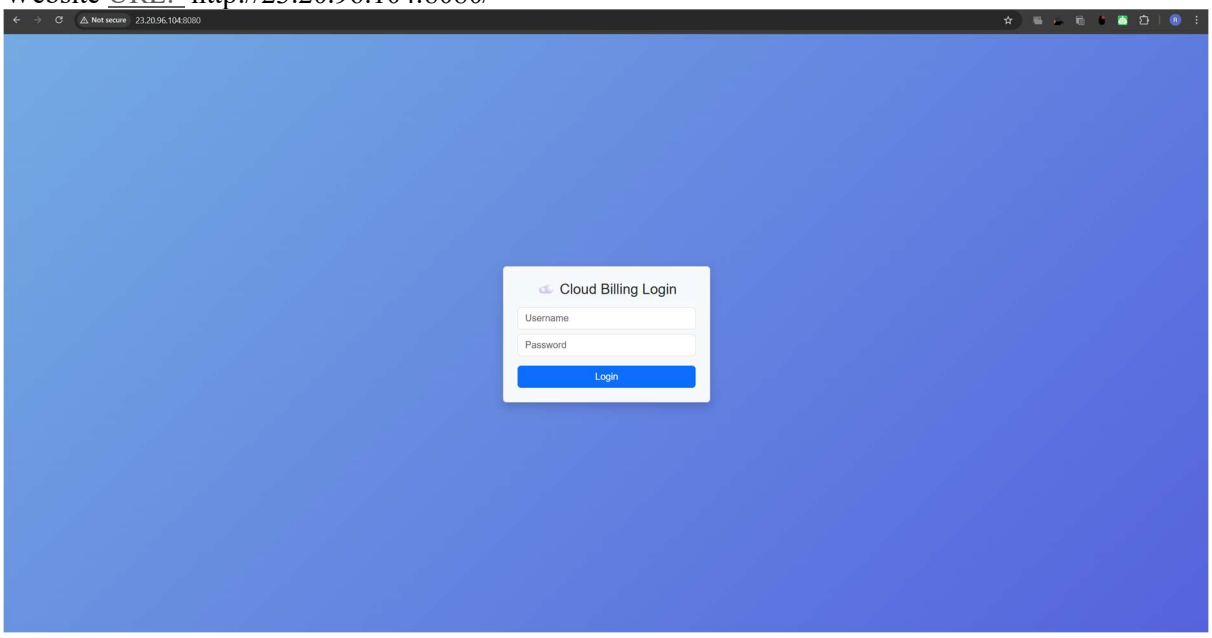
Task details:-

Configuration

Operating system/Architecture Linux/X86_64	**Capacity provider** Fargate	**ENI ID** eni-002f0d0adc7a3e570	**Public IP** 23.20.96.104 | open address
CPU | Memory .5 vCPU | 1 GB	**Launch type** Fargate	**Network mode** awsvpc	**Private IP** 172.31.41.99
Platform version 1.4.0	**Task definition: revision** flask-task:1	**Subnet** subnet-0d5cabf48603fed34	**MAC address** 0e:82:64:20:5a:47
Task execution role ecsTaskExecutionRole	**Task group** family:flask-task		
Task role -			
Fault injection Turned off			
ECS Exec | Info			

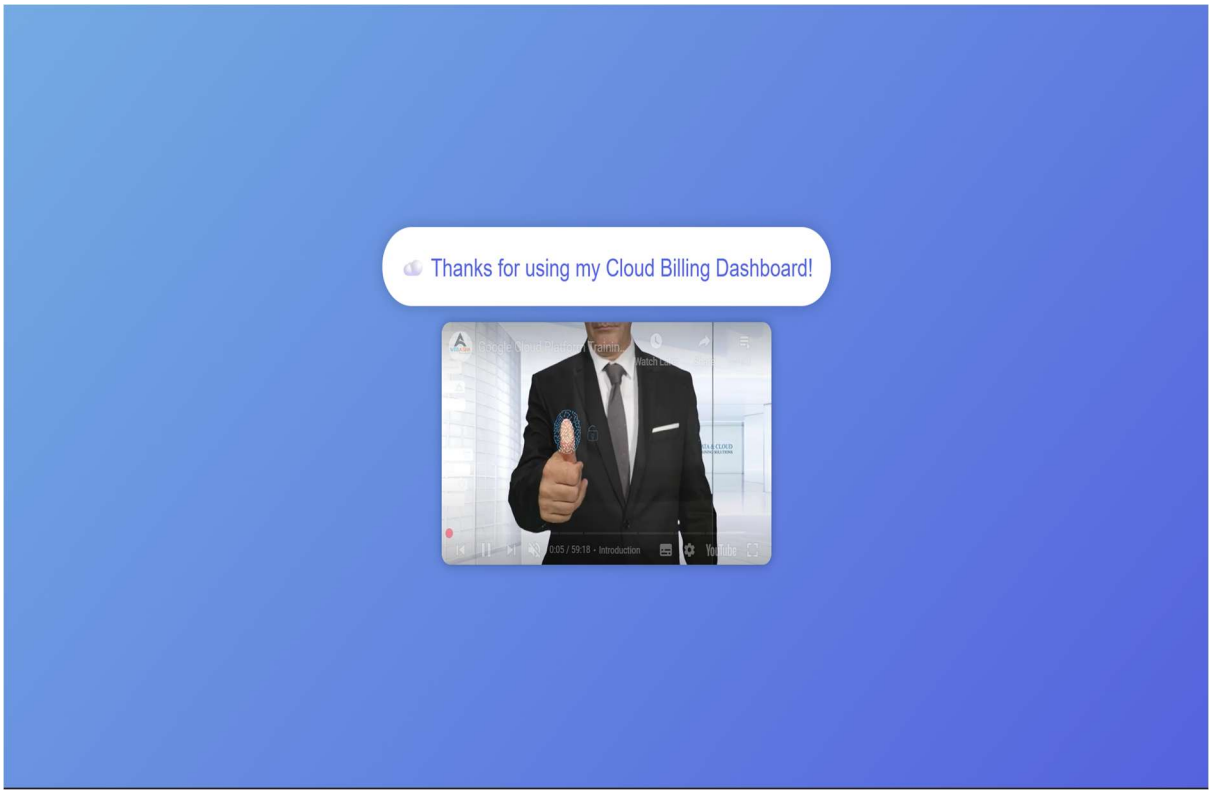
This is the application that is developed.A “CLOUD BILLING DASHBOARD”

Website URL:- http://23.20.96.104:8080/



Can calculate the sum of quantity which is purchased we can increase or decrease the quantity as needed.

Now when iam going to logout the youtube video will play for it that is showing how we can use use cloud billing dashboard in gcp.



⚠️
THESE WERE THE CHALLENGES WHICH I FACED.
Deployment Challenges Faced:-
🔐 AWS CLI Credential Errors
Encountered InvalidClientTokenId when running aws sts get-caller-identity

Root cause: expired or misconfigured credentials in ~/.aws/credentials

Solution: regenerated IAM access keys and reconfigured using aws configure

🐬 Docker Login Failure
Error: docker login requires at least one argument

Cause: missing ECR registry URL in the login command

Fix: added full registry URL using --password-stdin with account ID

📦 ECR Image Push Validation
Uncertainty whether image was successfully pushed to ECR

Verified using aws ecr list-images to confirm presence of latest tag

🌐 ECS Task Timeout
Browser showed ERR_CONNECTION_TIMED_OUT when accessing public IP

Cause: ECS task was running, but port access was blocked

🔒 Security Group Restrictions
Couldn't modify inbound rules to allow port 8080

Solution: created a new security group with custom TCP rule for port 8080 and attached it to ECS task

⚙️ Port Mismatch in Task Definition
App was running on port 8080, but ECS container was mapped to port 80

Fix: revised task definition to match container port with app port

Feature	**Docker**	**Amazon ECR**	**Amazon ECS**
Purpose	Build and run containers locally	Store container images in AWS	Deploy and manage containers in AWS
Type	Container engine	Container registry	Container orchestration service
Used for	Packaging apps into images	Hosting images for deployment	Running containers on AWS infrastructure
Where it runs	Developer machine or	AWS cloud	AWS cloud (Fargate or EC2)

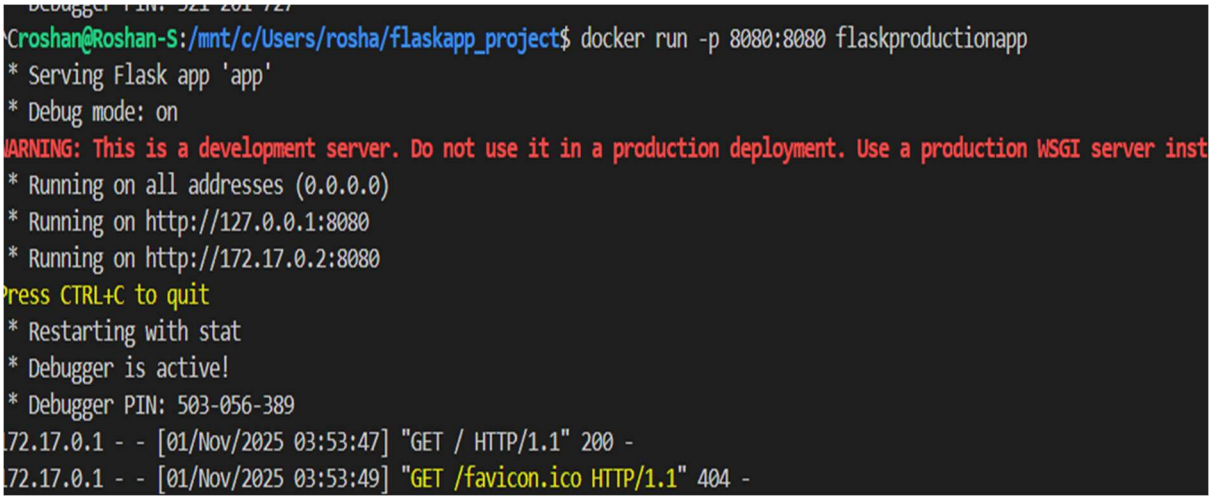
Feature	Docker	Amazon ECR	Amazon ECS
	server		
Key command	docker build, docker run	docker push, docker pull	aws ecs run-task, Console setup
Authentication	Local only	Requires AWS credentials	Requires AWS setup and networking
Storage	Local disk	Cloud-based image storage	No storage; pulls image from ECR
Integration	Works with ECR and ECS	Feeds images to ECS	Pulls images from ECR to run containers

RECENT ADVANCEMENTS IN 2025:-

Category	Use Case / Advancement	Description
AI Dashboards	Real-time ML model monitoring	Flask apps serve dashboards with live predictions, ECS auto-scales based on traffic.
Fintech	Transaction APIs & billing platforms	Flask handles secure endpoints; ECS ensures zero-downtime scaling and patching.
IoT & Edge Data	Sensor data ingestion and visualization	Flask apps stream and display real-time device data; ECS handles burst loads.
Healthcare	Patient vitals dashboard	Flask apps visualize live metrics; ECS ensures HIPAA-compliant isolation and scaling.
E-Learning	Interactive coding platforms & quizzes	Flask powers dynamic content; ECS scales with student load during exams or launches.
DevOps Tools	Internal dashboards for CI/CD pipelines	Flask apps track deployments and logs; ECS integrates with CloudWatch and IAM roles.
Retail & E-Commerce	Flash sale engines and inventory trackers	Flask apps update stock and pricing in real time; ECS handles traffic spikes.
Customer Support	Live ticketing and chatbot interfaces	Flask routes support queries; ECS ensures high availability during peak hours.
Media & Streaming	Personalized content feeds and recommendation engines	Flask serves user-specific content; ECS scales horizontally with viewer demand.

Category	Use Case / Advancement	Description
Smart Cities	Public dashboards for traffic, pollution, and utilities	Flask apps visualize civic data; ECS supports secure, multi-tenant deployments.

This is how I ran my container for the application to launch in vs code this locally from my system.



Conclusion:- could understand how to debug and fix the security groups and allow inbound traffic for port 8080 for theapplixcation developed in vs code. I have used ecr, ecs and docker desktop for running my application.And built a flask application using python language.