Cloud Computing Project Report

File-sharing web application

🌟Upload your files; let them soar in the cloud🌟

Team Members:

NAME	Roles
Nada Fayez	VPC and Networking
Nouran Hasan	IAM account setup & Billing alerts
Mariam Ramadan	EC2 Server
Omnia mohamed	S3 storage
Shahd Nezar	S3 storage
Salma Ahmed	Web App development

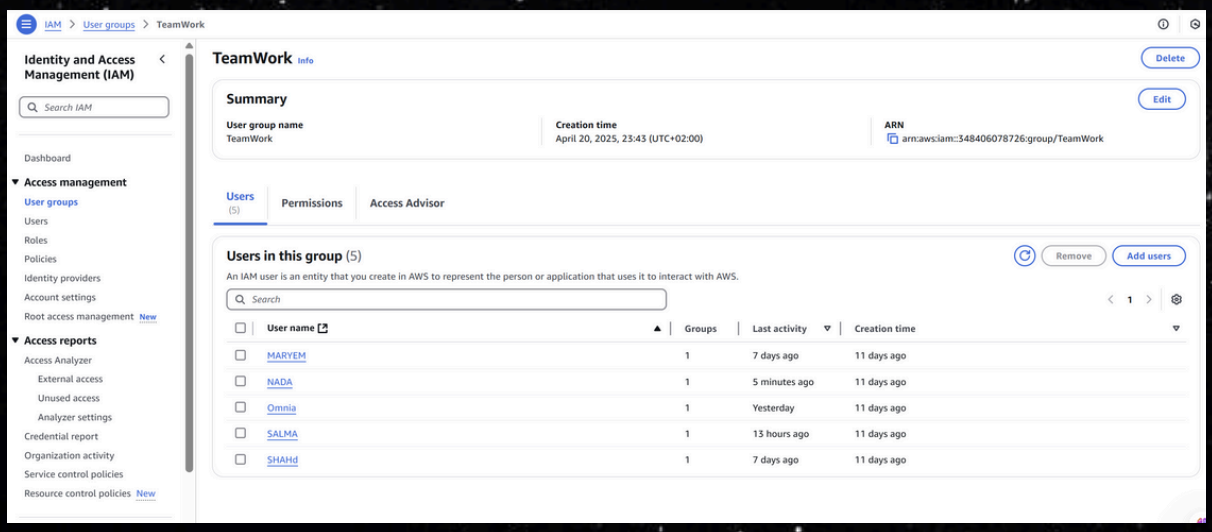
Project Overview:

We developed a cloud-based web application seamlessly integrated with Amazon EC2 to ensure 24/7 accessibility. Through this platform, users can effortlessly upload their files, which are then securely stored in Amazon S3. The entire system operates within a dedicated network, providing a safe, reliable, and protected .environment for data storage and access

---

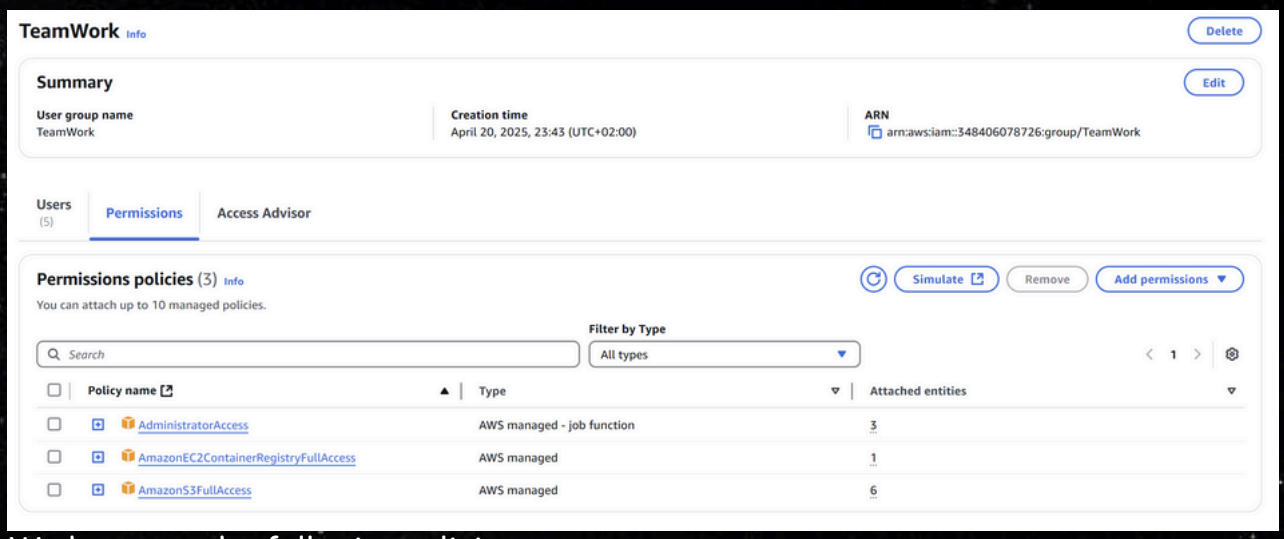
1- IAM Account & Billing alerts:

We used IAM accounts instead of the root account to enhance security and provide complete control over access privileges. With IAM, we can ,define precise permissions for each user or application individually allowing them to only have the permissions they need without granting excessive permissions that could compromise the system



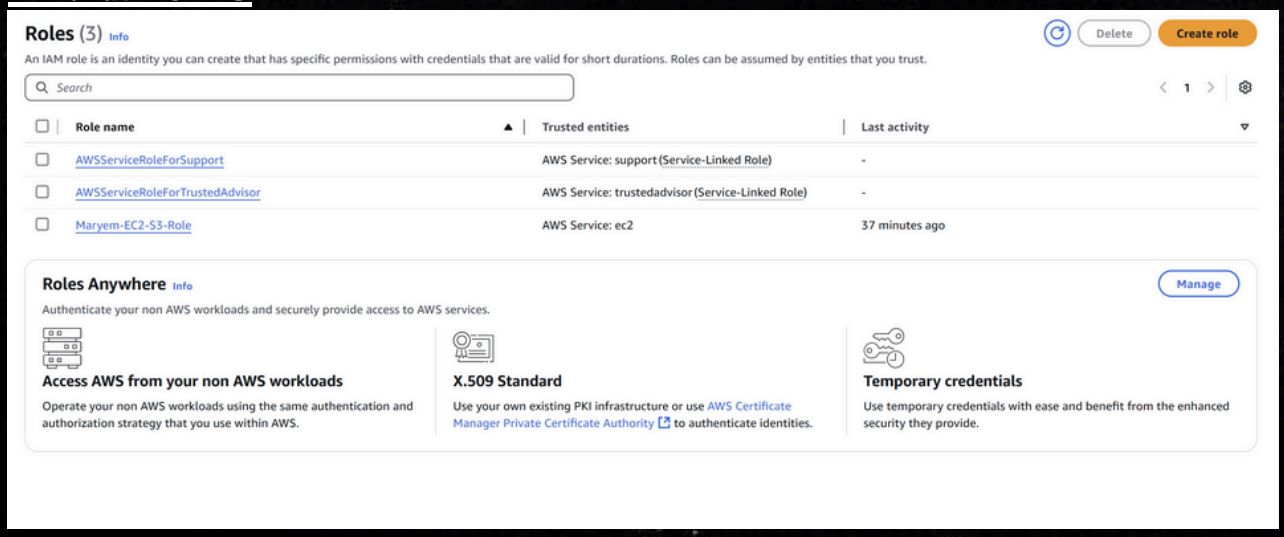
First, we created a group called "Team Work," then created accounts for all users working on the project. This way, we can manage the permissions of each team member separately and precisely, enhancing security and facilitating monitoring of activities

IAM Policies



We have set the following policies:
AdministratorAccess & AmazonEC2ContainerRegistryFullAccess, and AmazonS3FullAccess
These are AWS-managed policies, similar to the policies AWS provides to users by default
Administrator Access: This policy grants full administrative privileges to all AWS resources
AmazonEC2ContainerRegistryFullAccess: This policy allows full access to Amazon ECR (a (container storage service
AmazonS3FullAccess: This policy allows full access to Amazon S3

IAM Roles



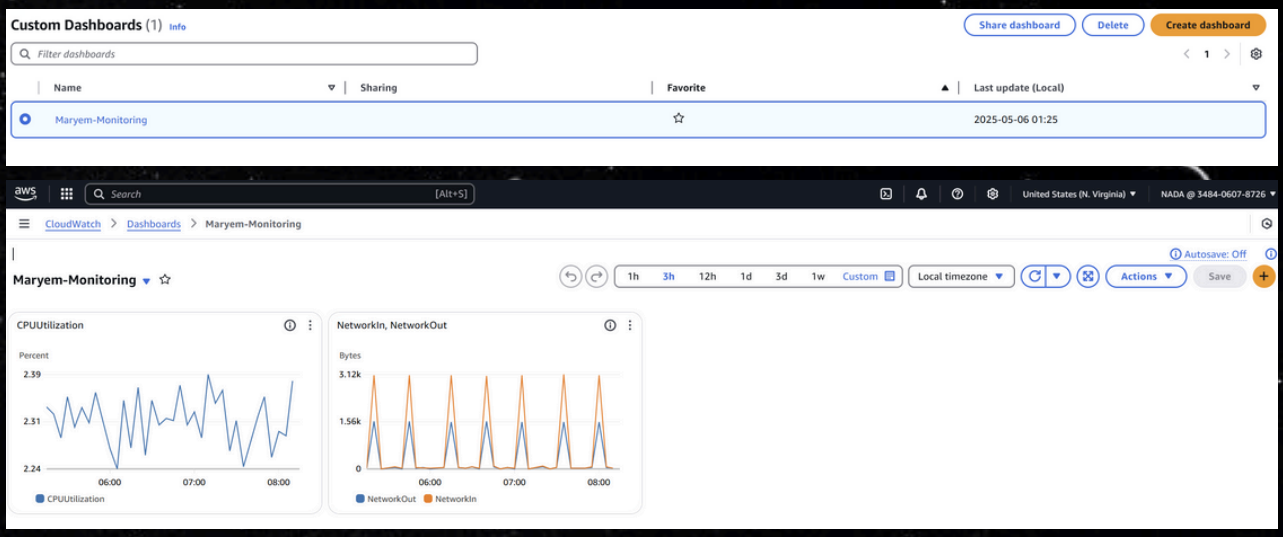
We created and used IAM Roles to define the permissions granted to different AWS services

AWSServiceRoleForSupport
This is a Service-Linked Role automatically created by AWS and used to support management services

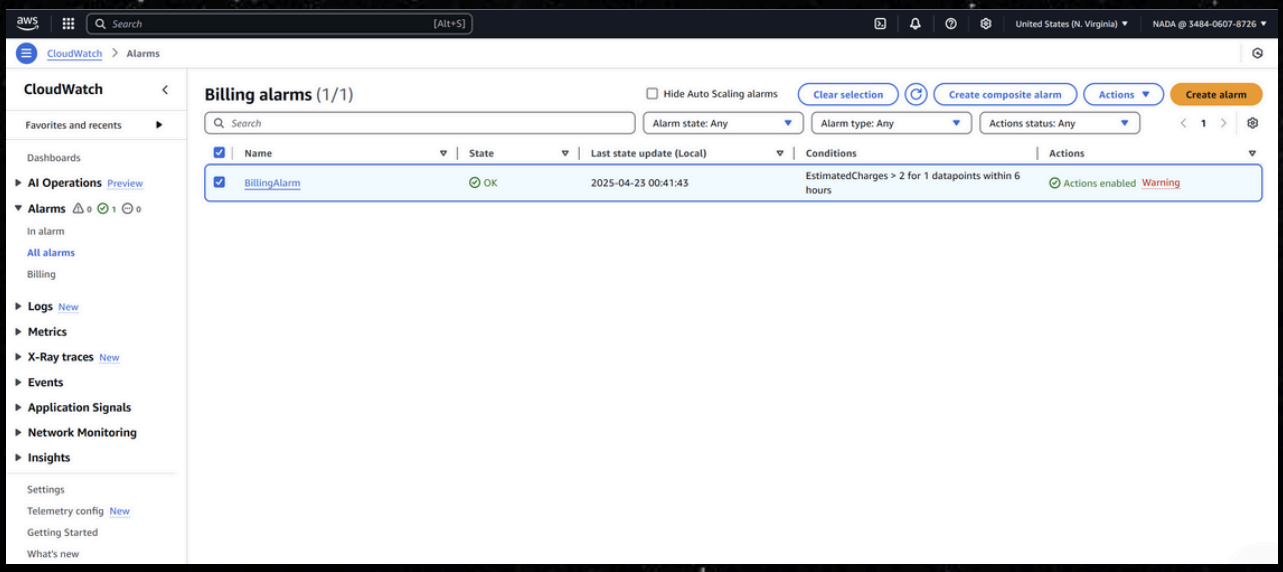
AWSServiceRoleForTrustedAdvisor
This is also a Service-Linked Role used by Trusted Advisor to improve performance and enhance security

Maryem-EC2-S3-Role
This is a custom Role we created to allow EC2 to access the S3 service.This means that only the EC2 instance can use this role

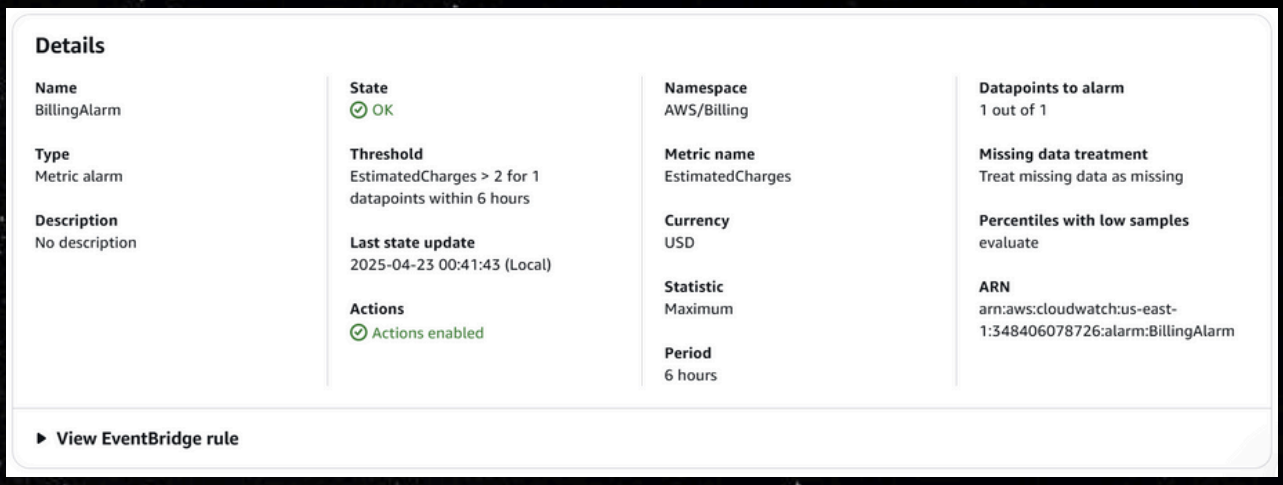
Billing Alarm

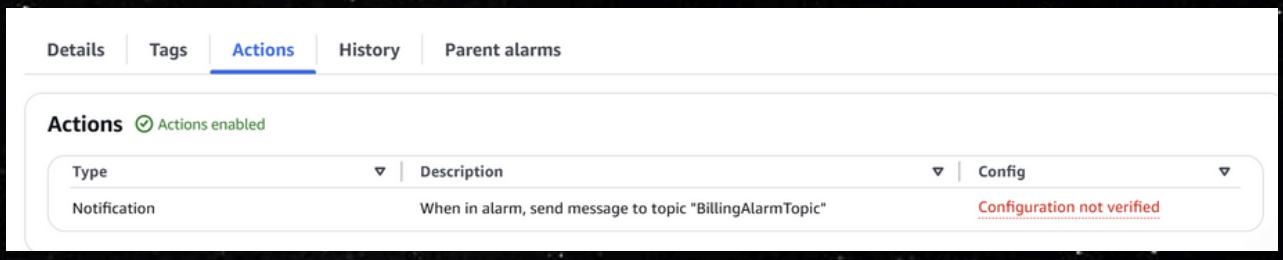


First, we created a dashboard to monitor server performance in real-time displaying vital indicators such as CPU usage and network in/out. This tool helps quickly detect any issues, such as abnormal data traffic. Thanks to this dashboard, actions can be taken to ensure server stability



Then we created a billing alarm using CloudWatch to track the estimated cost of our account on a regular basis



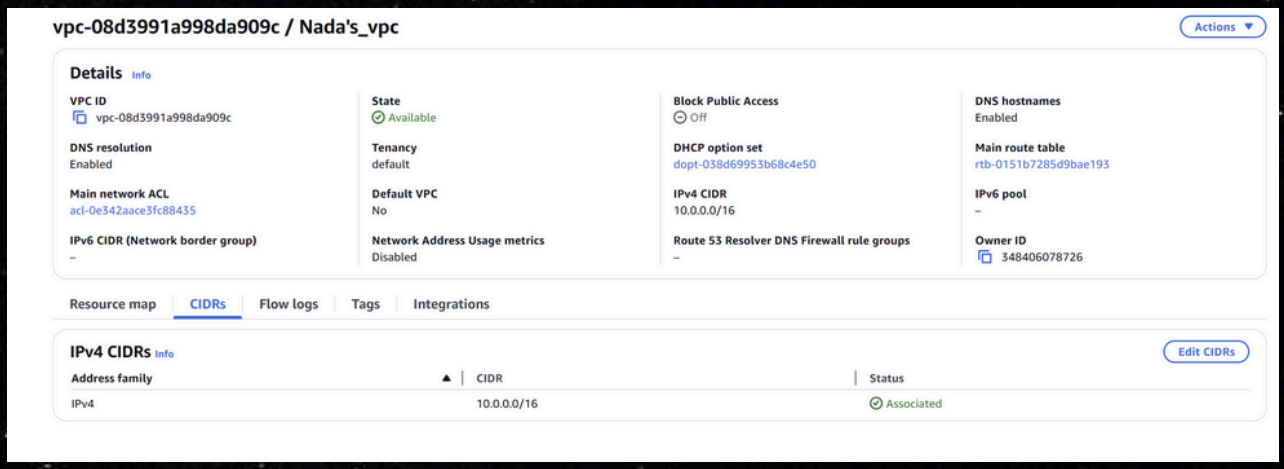


The alarm is set to trigger if the estimated cost exceeds $2 within a 6 hour period, and an automatic notification is sent to the SNS Topic if this occurs. This step helps us maintain budget and control our project's cloud usage safely and effectively

2- VPC and Networking

VPC

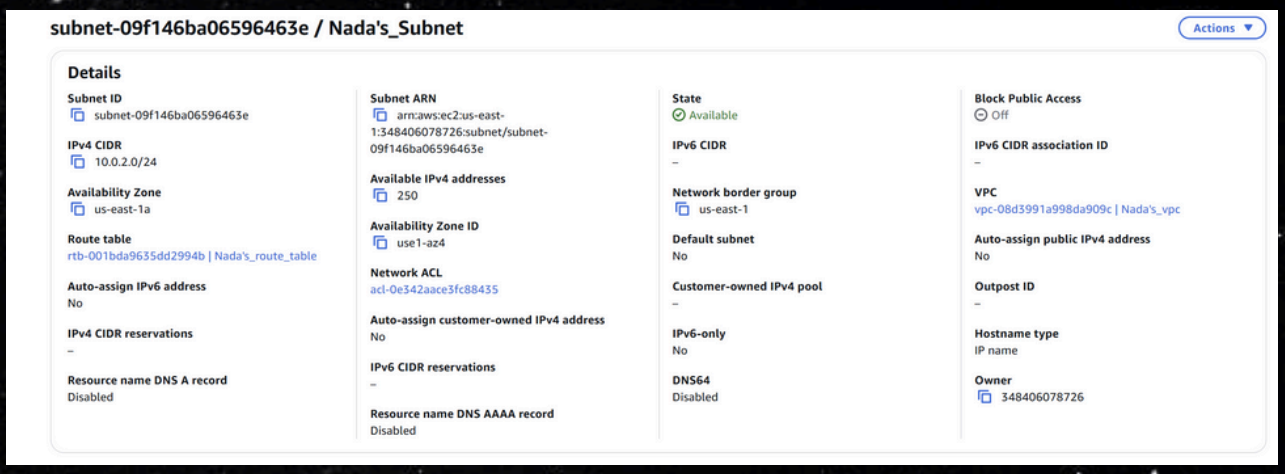
A VPC (Virtual Private Cloud) is an isolated virtual network within the AWS environment that serves as a private and secure network environment, allowing us to have complete control over the network configuration for applications, including IP addresses, network segmentation into subnets, firewall settings, and traffic routing.



First, I created an isolated virtual private network (VPC). I defined the network scope using IPv4 CIDR 10.0.0.0/16, providing up to 65,536 internal IP addresses. I also enabled DNS Resolution to translate domain names to IP addresses, enabling resources within the network to connect to the internet, and enabled DNS Hostnames to automatically provide internal DNS names for each instance within the network

Subnet

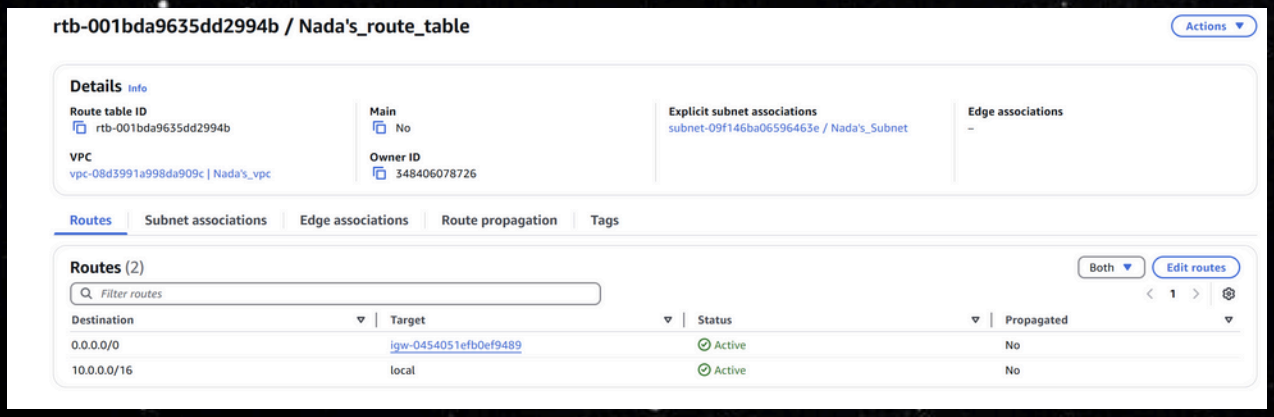
A subnet is a part of a VPC that we use to organize the distribution of resources within the network, and separate sensitive resources (such as databases) from public resources (such as servers that receive users), to improve security and communication efficiency



For our subnet, we chose IPv4 CIDR 10.0.2.0/24, which allows approximately 256 IP addresses to be distributed to resources such as EC2 instances. The subnet is available for use and is linked to the created VPC (Nada's_vpc). It is not available for public access by default, as "Block Public Access" is disabled in the network settings

---

Route Table



It's a table that defines the paths for data traffic in and out of the VPC network. I linked it to our subnet, and it contains two rules: one for internal communication between network resources (10.0.0.0/16), and the other for routing internet connections through the Internet Gateway (0.0.0.0/0). This rule not only allows the server to go online, but also allows users to log in to it

Internet Getaway

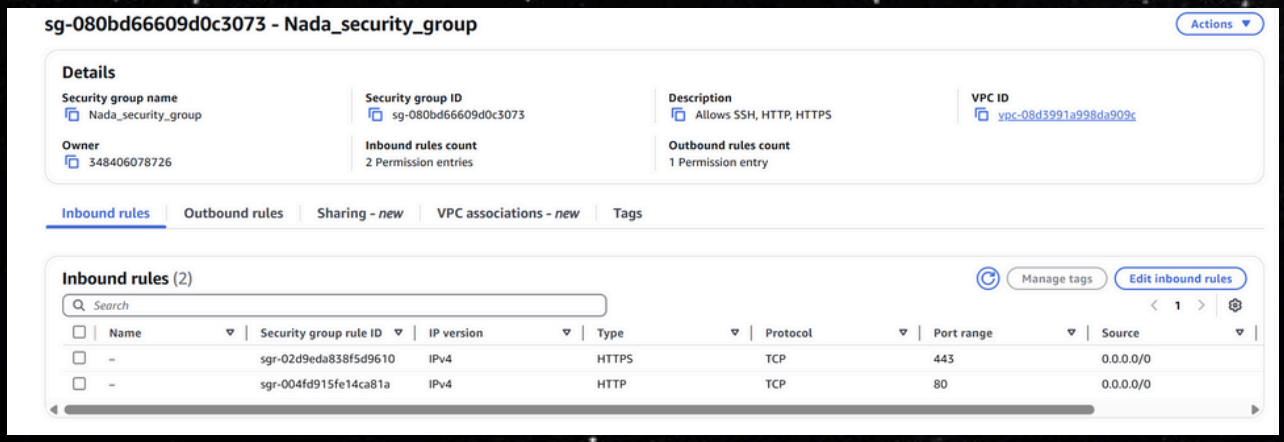
The Internet Gateway (IGW) is a core component of AWS that allows resources within a VPC to communicate with the internet

The IGW which I've named Nada's_IGW is connected to our VPC Nada's_vpc It allows all resources within the VPC (such as EC2 instances) to access and exit the internet based on rules defined in the Route Table.So its primary function is to connect the VPC network to the internet

---

Security Group

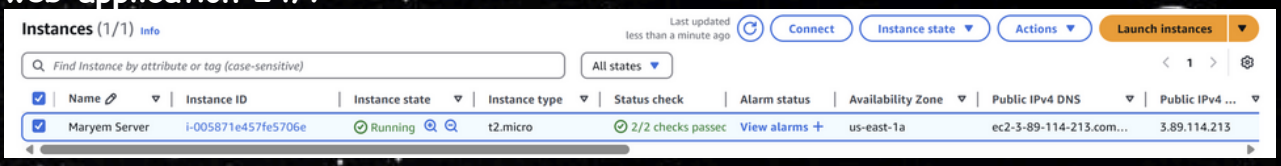
A Security Group is a set of security rules that defines the type of communication allowed between EC2 instances and other resources in a VPC. Nada_security_group is the Security Group we are using in this project

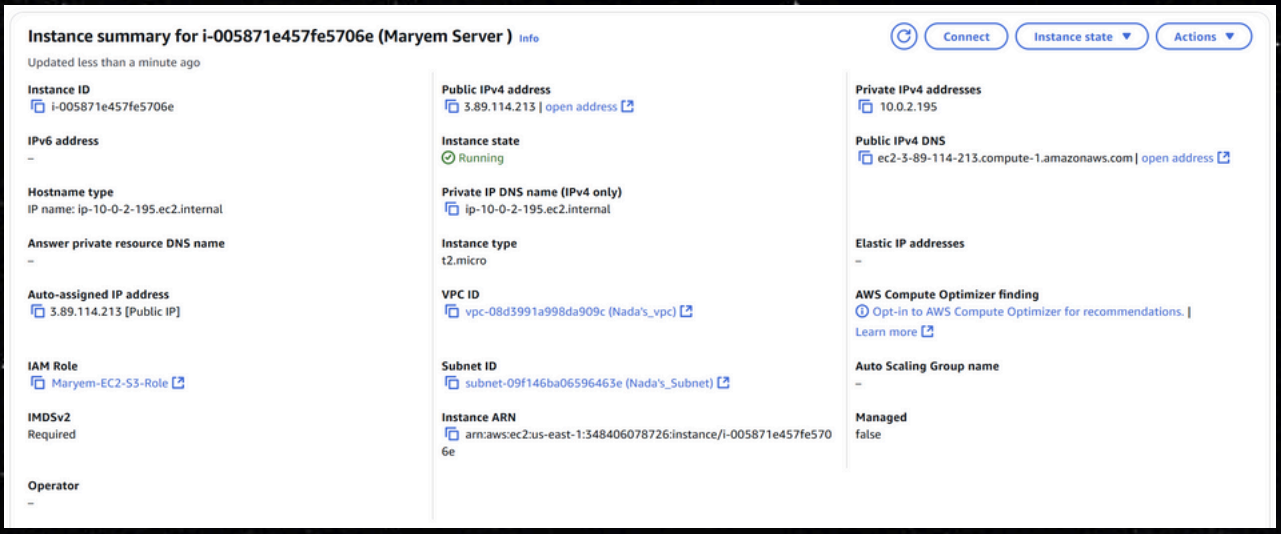


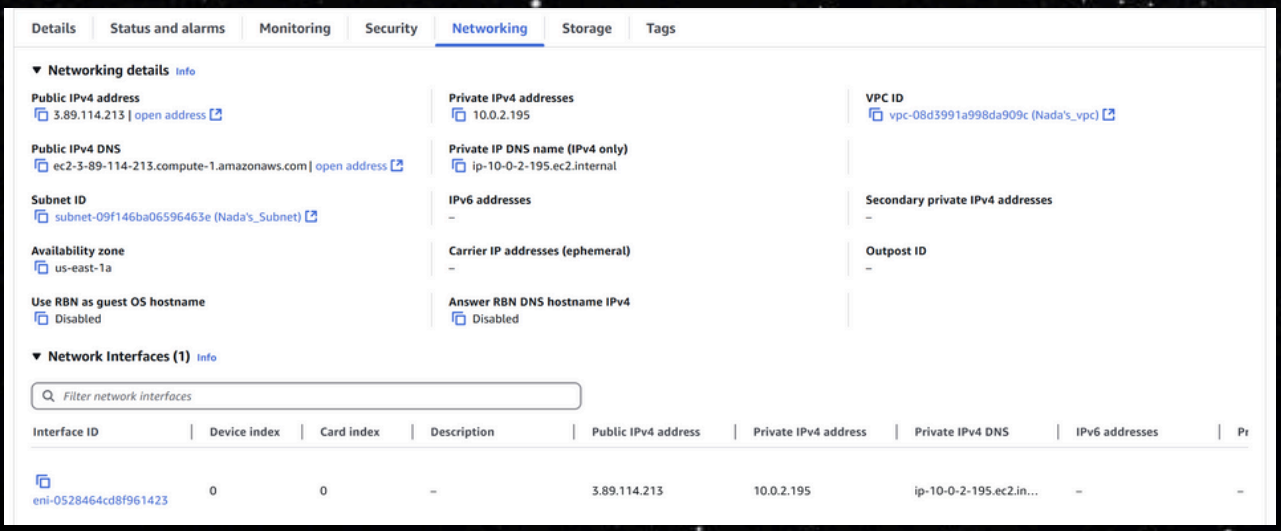
The two existing rules allow access to the server from anywhere in the world (0.0.0.0/0) via HTTP (port 80). HTTP is used for browsing websites and is an unencrypted protocol, meaning that data exchanged between the client and the server is sent in plain text. HTTPS is used for browsing secure websites and relies on the same HTTP protocol with an additional layer of encryption using SSL/TLS to ensure the protection of the exchanged data

3- EC2

Amazon EC2 is a service from AWS that provides virtual servers (instances) to run applications. It allows you to choose the type of server based on your processing, storage, and memory needs, with the flexibility to resize as needed. The service is pay-as-you-go and provides continuous hosting for your web application 24/7



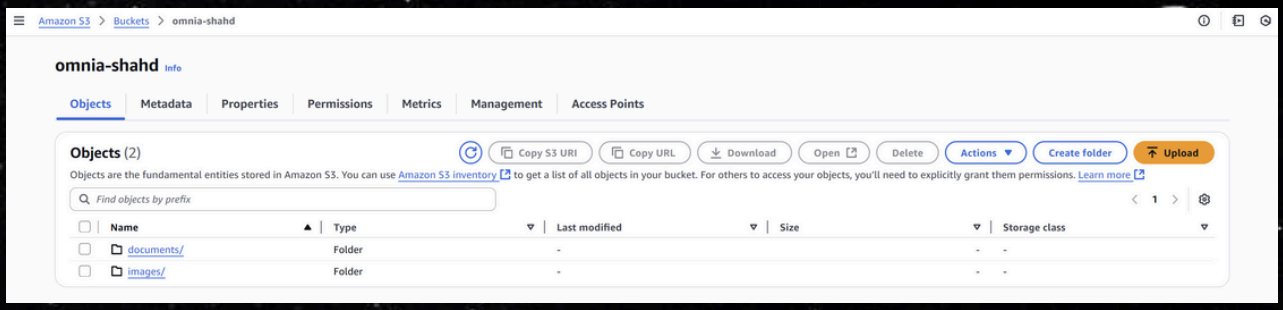




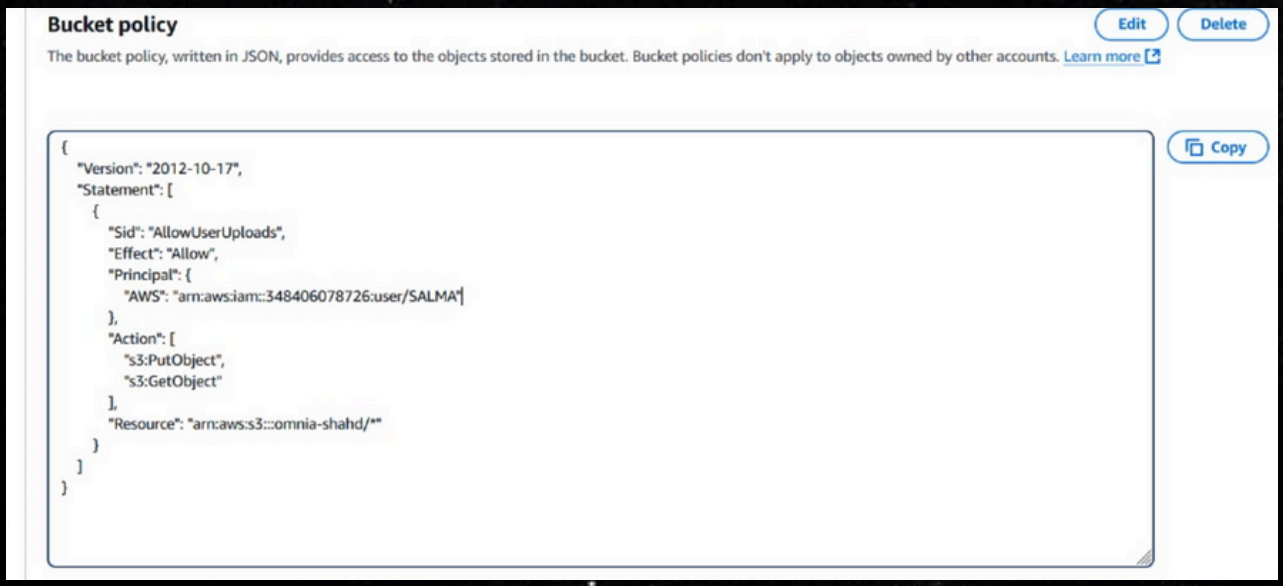
The EC2 instance "Maryem Server" is a t2.micro server with a public IP address (3.89.114.213) and a private IP address (10.0.2.195) inside the VPC "Nada's_vpc" and subnet "Nada's_Subnet". The server is associated with the IAM role "Maryem-EC2-S3-Role" which allows it to access the S3 service. In the security details, the security group Nada_security_group" is assigned to the server we create

4- S3 Storage

Amazon S3 is one of the most important cloud storage services provided by AWS, as it allows files and data to be stored securely and easily accessible over the Internet



An S3 Bucket named "omnia-shahd" was created in AWS to store the required files and data. The bucket's settings were configured, with the "Block all public .access" option disabled, to allow access via EC2 and its associated IAM role The bucket has internal file organization by creating folders such as Documents Photos, and Videos to facilitate data management



A container-specific policy was defined by navigating to the Permissions tab in the container settings, where a policy was added that allows access to upload and download files during the test period. This policy ensures flexible ways to interact with the data stored within the container

5- Web Application

Backend

The backend is the part responsible for processing data and performing operations in the background

```python
app = Flask(__name__)
CORS(app, resources={r"/": {"origins": "*"}}, supports_credentials=True)

s3 = boto3.client(
    's3',
    region_name='us-east-1',
    aws_access_key_id='AKIAVCHU63UDHOUANH46',
    aws_secret_access_key='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'
)

BUCKET_NAME = 'omnia-shahd'

def get_file_folder(filename):
    extension = os.path.splitext(filename)[1].lower()

    if extension in ['.jpg', '.jpeg', '.png', '.gif', '.bmp', '.webp']:
        return 'images/'
    elif extension in ['.pdf', '.doc', '.docx', '.xls', '.xlsx', '.ppt', '.pptx', '.txt']:
        return 'documents/'
    elif extension in ['.mp4', '.mov', '.avi', '.mkv', '.wmv']:
        return 'videos/'
    else:
        return 'others/'

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'}), 400

    file = request.files['file']
```

```python
def upload_file():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'}), 400

    file = request.files['file']

    if file.filename == '':
        return jsonify({'error': 'No selected file'}), 400

    try:
        folder = get_file_folder(file.filename)
        filename = f"{folder}{uuid.uuid4().hex}_{file.filename}"

        s3.upload_fileobj(file, BUCKET_NAME, filename)

        file_url = f"https://{BUCKET_NAME}.s3.amazonaws.com/{filename}"

        return jsonify({
            'message': 'File uploaded successfully',
            'url': file_url,
            'folder': folder[:-1]
        }), 200

    except NoCredentialsError:
        return jsonify({'error': 'Credentials not available'}), 500
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

The application was built using Flask to power the web application, while the boto3 library interacts with AWS and stores files in an S3 Bucket. Files are uploaded using the upload_fileobj function, which uploads the file to the Bucket using the specified file name and content type. After the file is successfully uploaded and stored, a download link is returned to the user to access the saved file

Pretty-print ☐

{"download_link":"https://omnia-shahd.s3.amazonaws.com/Salma-Ahmed-Mohamed-Tabana-FlowCV-Resume-20250428.pdf"}

Frontend

The front end is the part that interacts with the user and displays data visually

```
<DOCTYPE html!>
<"html lang="en>
<head>
</ "meta charset="UTF-8>
<title>Cloudy Files ✨</title>
<style>
} html, body
;height: 100%
;margin: 0
;font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif
{

} body
;background: linear-gradient(135deg, #667eea 0%, #764ba2 100%)
;display: flex
;justify-content: center
;align-items: center
;flex-direction: column
;overflow-x: hidden
;text-align: center
;position: relative
{

} body::before
;"" :content
;position: absolute
;top: 0
;left: 0
;width: 100%
;height: 100%
;background-image: url('https://images.unsplash.com/photo-1519681393784-d120267933ba?ixlib=rb-1.2.1&auto=format&fit=crop&w=1350&q=80')
;background-size: cover
;background-position: center
;opacity: 0.3
;z-index: -1
{

} h1
;color: #fff
;text-shadow: 2px 2px 5px #00000060
;white-space: nowrap
;font-size: 2.5rem
;margin-bottom: 2rem
{

} form
;margin: 20px auto
;padding: 30px
;background-color: #ffffffee
;border-radius: 20px
;width: 350px
;box-shadow: 0 10px 30px rgba(0,0,0,0.3)
;backdrop-filter: blur(5px)
;border: 1px solid rgba(255,255,255,0.2)
{

} input[type="file"]
;margin-bottom: 15px
;width: 100%
```

```
} button:hover
;transform: translateY(-3px)
;box-shadow: 0 7px 20px rgba(0,0,0,0.3)
{

} button:active
;transform: translateY(1px)
{

} link.
;margin-top: 20px
;color: #fff
;font-weight: bold
;font-size: 1.1rem
{

} keyframes float@
{ ;transform: translateY(0px) } %·
{ ;transform: translateY(-10px) } %٥·
{ ;transform: translateY(0px) } %١··
{

} h1
;animation: float 3s ease-in-out infinite
{

} media (max-width: 768px)@
} form
;width: 80%
;padding: 20px
{

} h1
;font-size: 1.8rem
;white-space: normal
{
{
<style/>
<head/>
<body>
<h1>🌟 Upload your files, let them soar in the cloud 🌟</h1>

<"form enctype="multipart/form-data" method="POST" action="http://127.0.0.1:5000/upload>
</ input type="file" name="file" accept=".pdf,.jpg,.png,.txt,.doc,.docx,.xls,.xlsx" required><br>
<button type="submit">Upload to Cloud</button>
<form/>

<div class="link" id="downloadLink"></div>

<script>
} document.querySelector('form').addEventListener('submit', function(e)
;alert('File is being uploaded to the cloud...')
;({
<script/>
```

The front-end is based on an HTML template that allows users to upload files of various types, such as .pdf, .jpg, .png, and .txt. Data is sent via a POST request to the server, and when the upload is complete, the server returns a download link to the file stored in S3, allowing the user to download the file directly