Cloud Computing Final Project

Team mEMbers:

Farah Walid (23010036)

Nour Essam (23010035)

Basmala Hossam El-Din (23011052)

Rodina Mohamed (23010014)

Mariam Ahmed (23012058)

Cyber Security Department

INTRODUCTION:

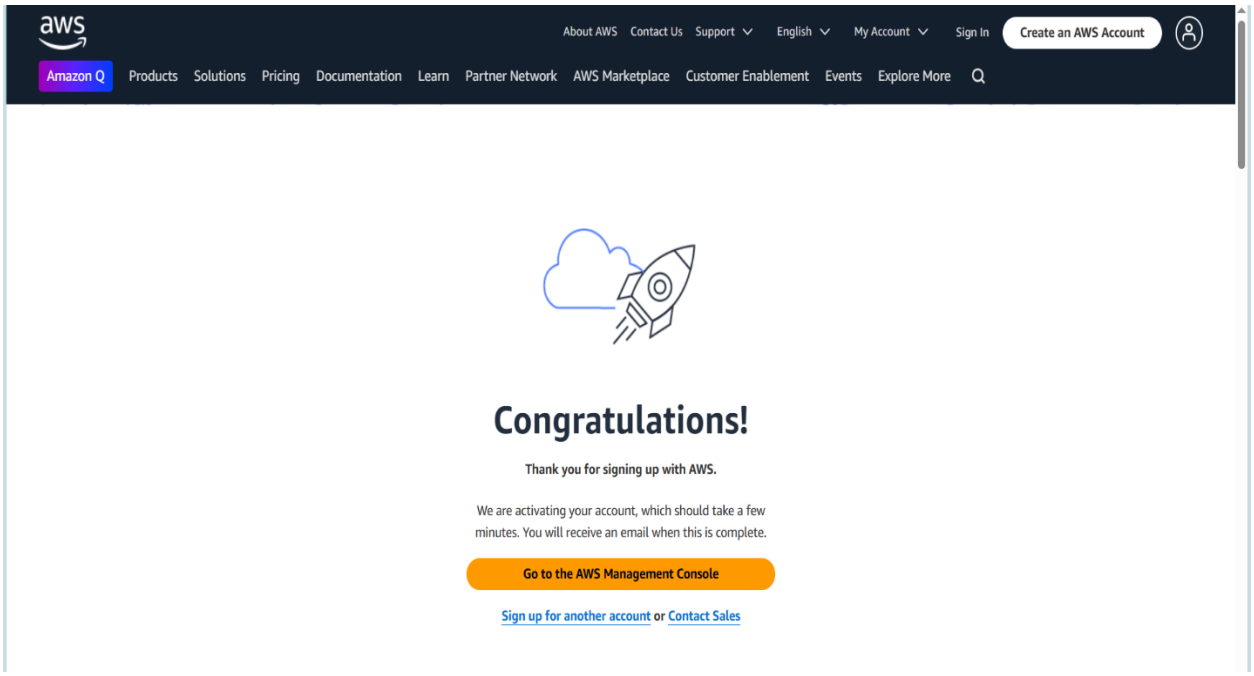
Cloud computing has transformed the way modern applications are developed, deployed, ana managed by offering scalable, cost-effective and reliable infrastructure on demand.

This project leverages the power of Amazon Web Services (AWS) to build a simplified file-sharing web application that demonstrates real-world use of core cloud services.

This report outlines the step-by-step setup of each AWS service, explains the implementation of the file-sharing application and reflects on the challenges encountered and lessons learned throughout the process.

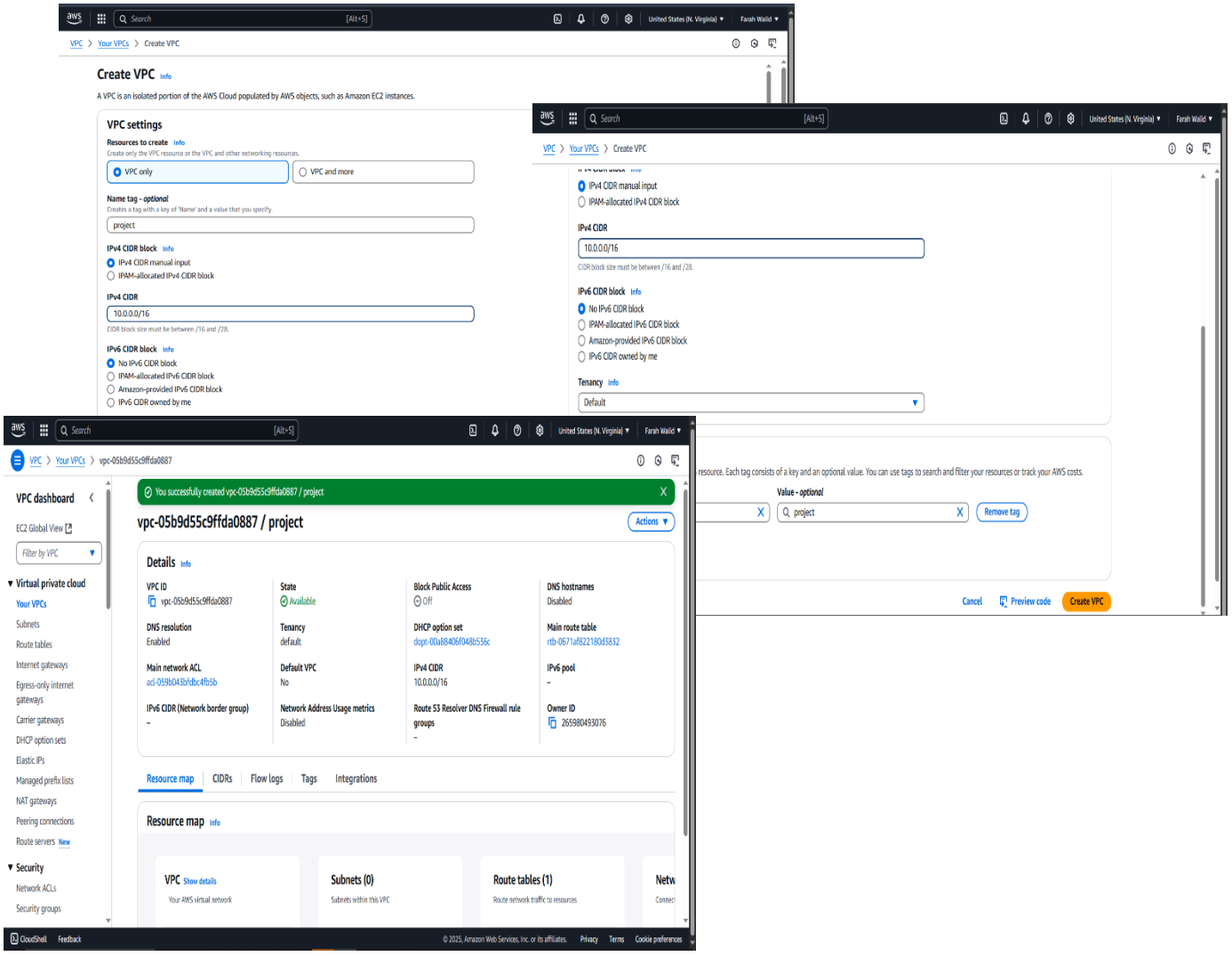
Here we go...

Congrats we have made our AWS account.



Creating a VPC in AWS:

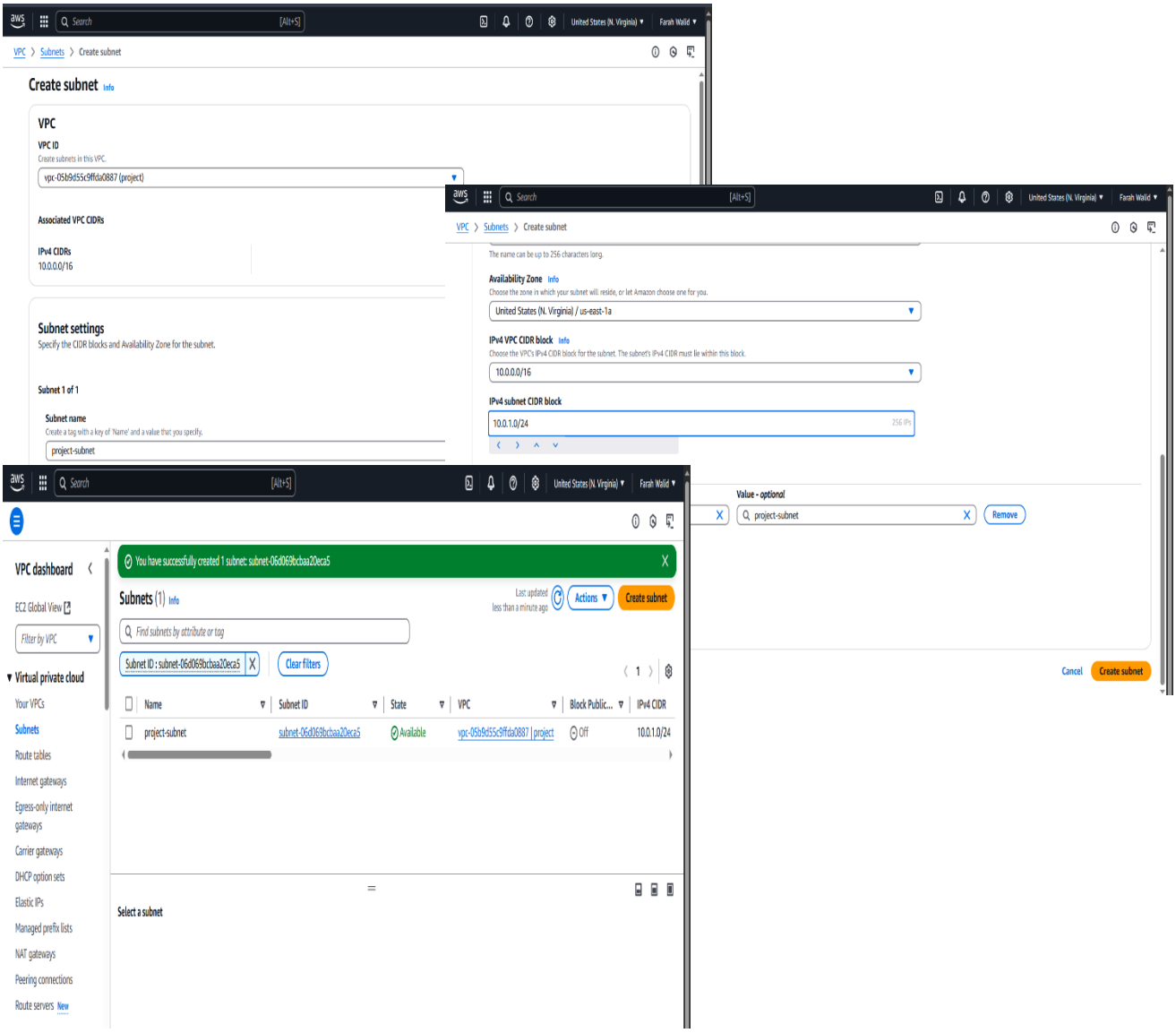
A Virtual Private Cloud in AWS is a logically isolated section of the AWS Cloud where we can launch AWS resources in a virtual network. VPC creation is essential for setting up a secure and organized network environment.



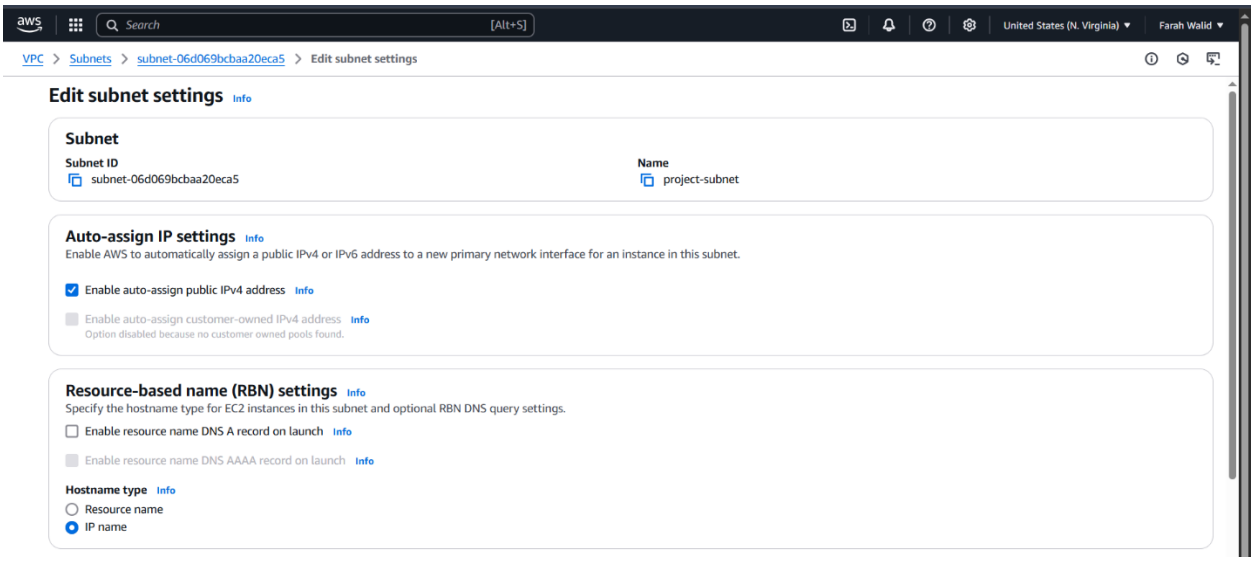
- ✓ Click on **"Create VPC"**.
- ✓ Choose the **"VPC only"** option as we wanted to manually configure subnets.
- ✓ VPC Name tag: "project".
- ✓ IPv4 CIDR block: "10.0.0.0/16" to define the IP address range for the network (Standard private range).
- ✓ Click "Create VPC".

Creating Subnets:

Subnets divide the VPC into smaller, manageable sections.



- ✓ Go to "Subnets", Click on "Create subnet".
- ✓ VPC ID: Select the "project" VPC we just created.
- ✓ Subnet Name: "project-subnet".
- ✓ Availability Zone: Choose "us-east-1a".
- ✓ IPv4 CIDR block: "10.0.1.0/24".
- ✓ Click "Create subnet".

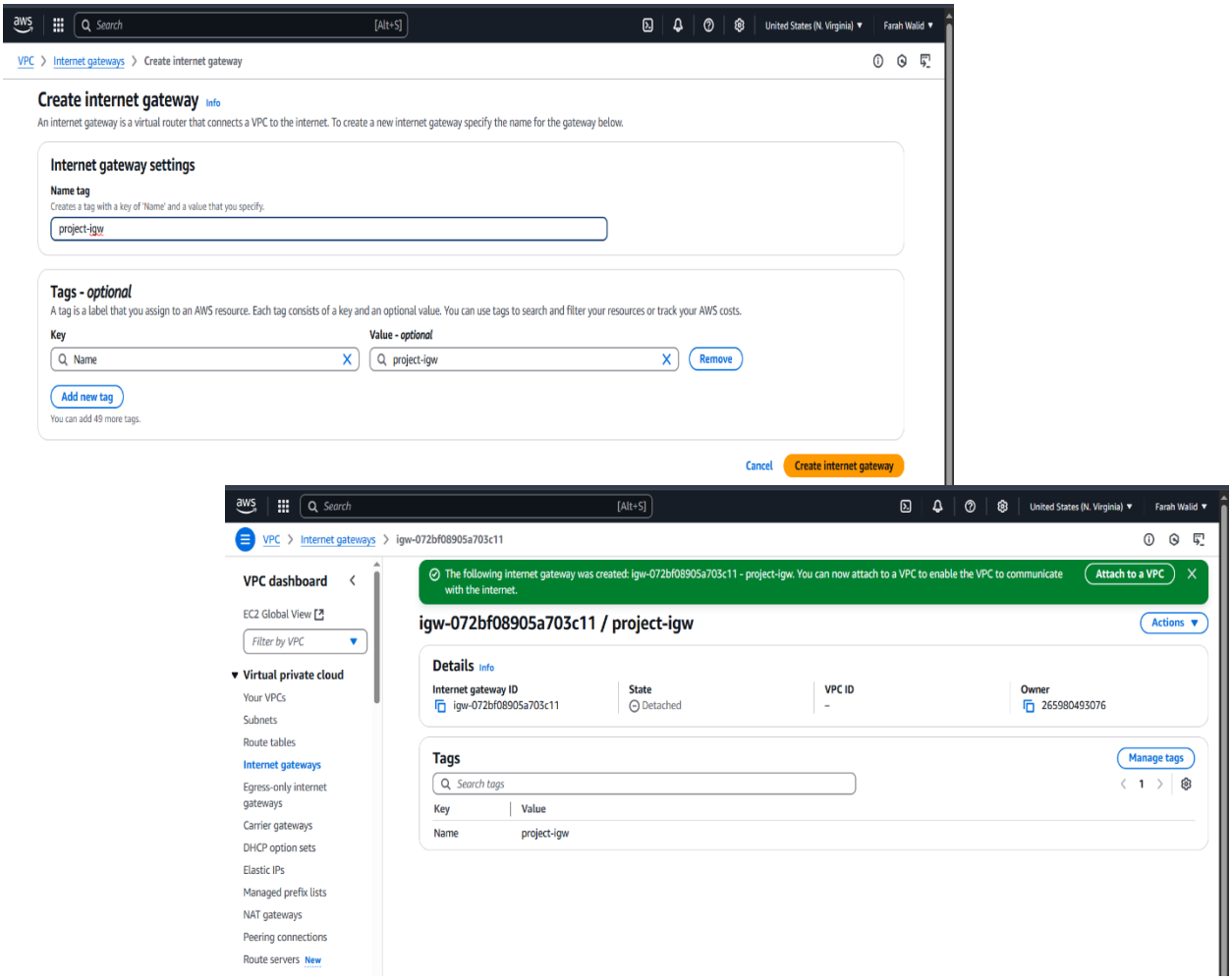


✓ Enable Auto-assign public IPv4 address: Select "project-subnet" then click on "Actions", then "Edit subnet settings", check on "Enable auto-assign public IPv4 address", then save.

- Subnets define where the EC2 instance runs within the VPC.
- We can control access to and from our instance.
- Subnets help us to organize our cloud network for better performance, scalability and security.

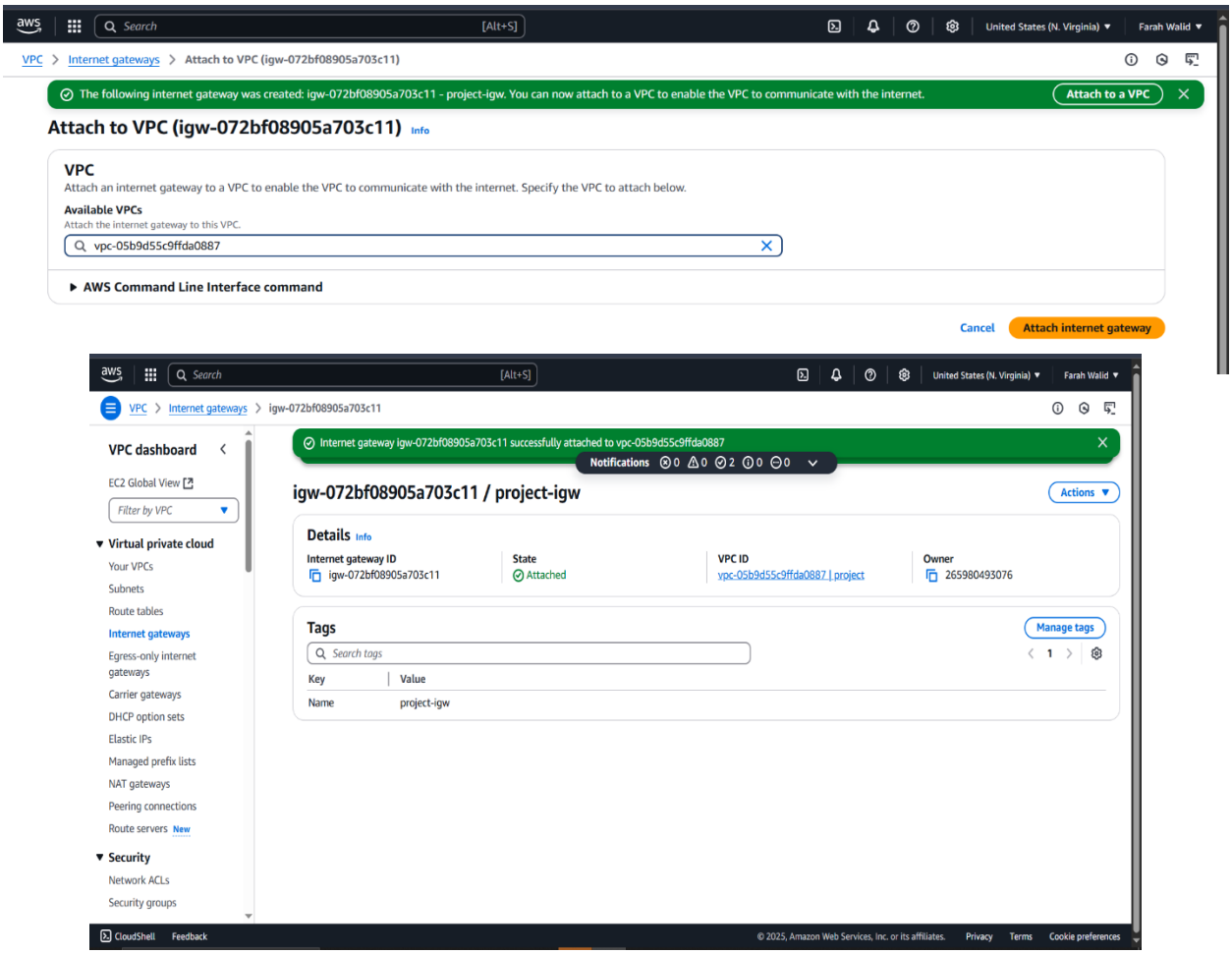
Creating Internet Gateway (IGW):

An IGW allows communication between the VPC and the Internet.



An Internet Gateway (IGW): a horizontally scaled, redundant, and highly available AWS-managed component that allows communication between resources in a VPC and the internet. It is a gateway attached to your VPC that enables public access to and from instances that have public IP addresses.

- ✓ Go to "Internet Gateways", Click "Create internet gateway".
- ✓ Name tag: "project-igw".
- ✓ Click "Create internet gateway".

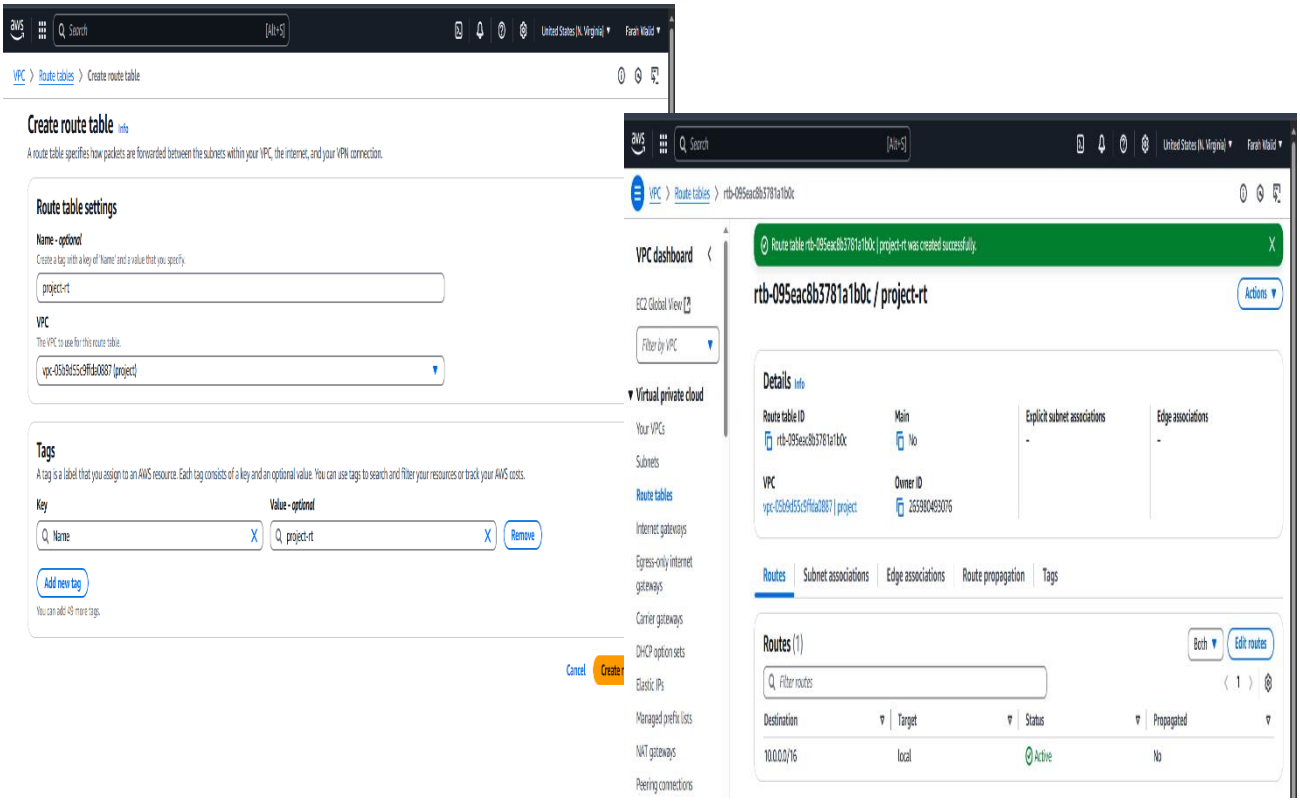


- ✓ Select the created IGW "project-igw", then click "Actions".
- ✓ Click "Attach to VPC".
- ✓ Select "project", then click "Attach internet gateway".
- ✓ The IGW successfully attached to our VPC.

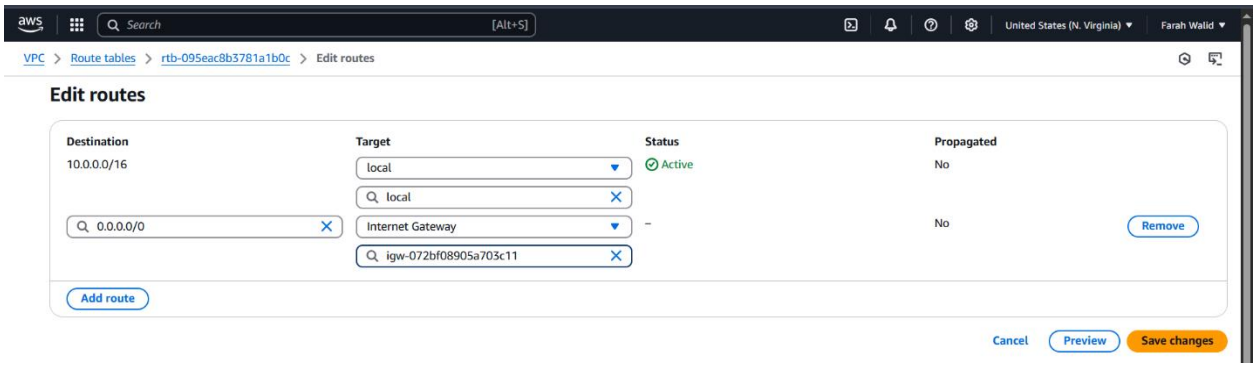
An **Internet Gateway (IGW)** is a **horizontally scaled, redundant, and highly available AWS-managed component** that allows communication between resources in a VPC and the internet. It is a **gateway attached to your VPC** that enables **public access** to and from instances that have public IP addresses.

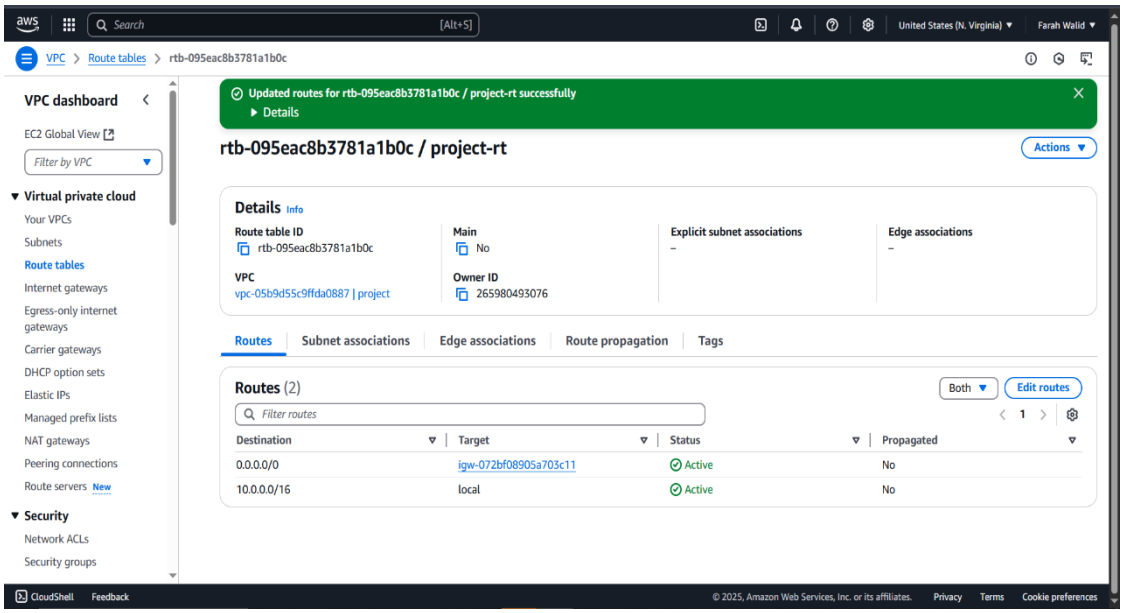
Creating Route Table and Associations:

In AWS, a Route Table is a set of rules—called routes—that determine how network traffic is directed within a Virtual Private Cloud (VPC). Each subnet in a VPC must be associated with a route table, which defines how traffic leaving that subnet is routed.

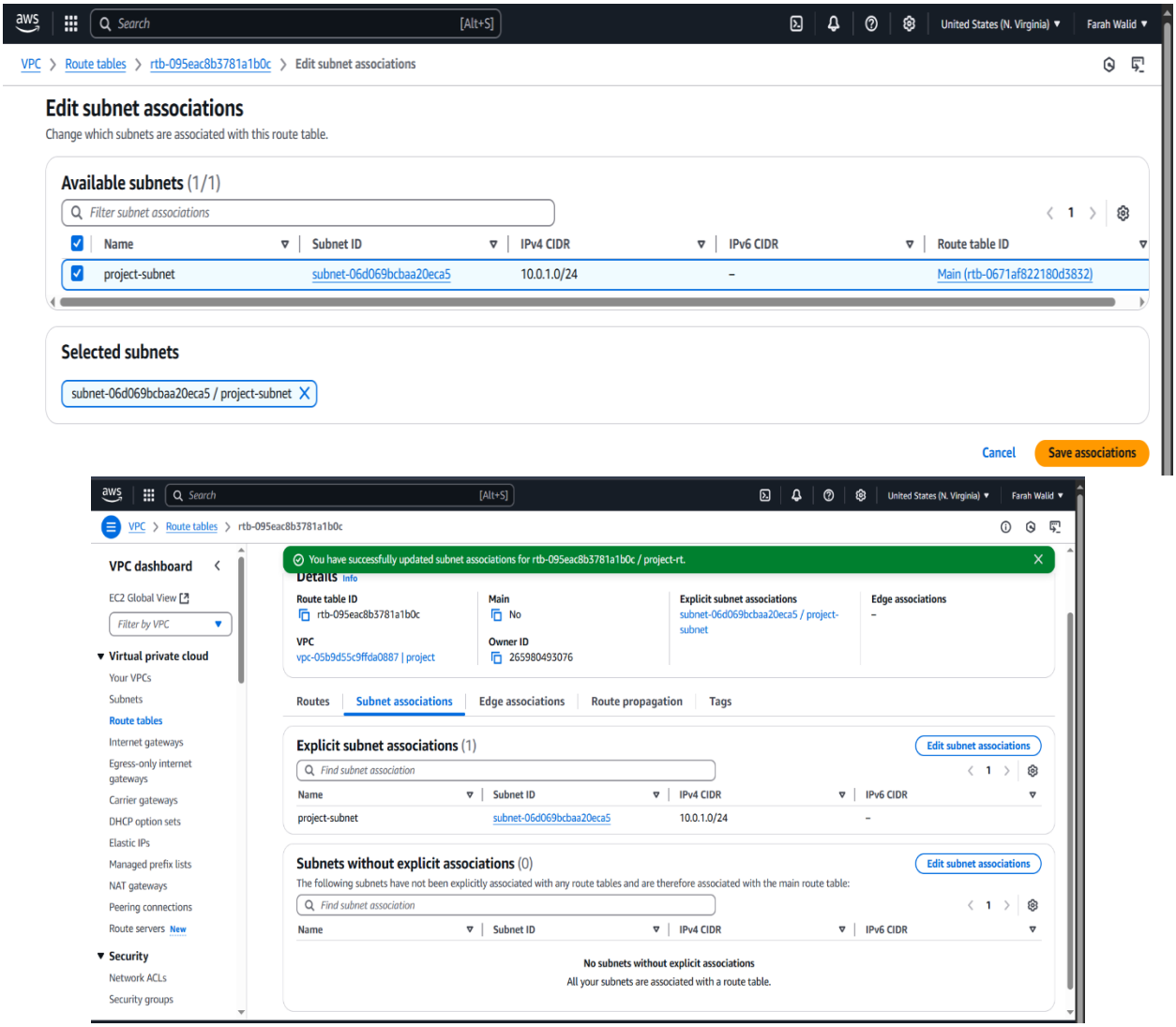


- ✓ Go to "Route Tables", then click "Create route table".
- ✓ Name tag: "project-rt".
- ✓ VPC: Select "project".
- ✓ Click "Create route table".





- ✓ Select the new route table "project-rt".
- ✓ Go below to the Routes tab, Click "Edit routes".
- ✓ Click "Add route".
- ✓ Destination: 0.0.0.0/0 (All Traffic).
- ✓ Target: Select "Internet Gateway", then "project-igw".
- ✓ Click "Save changes".

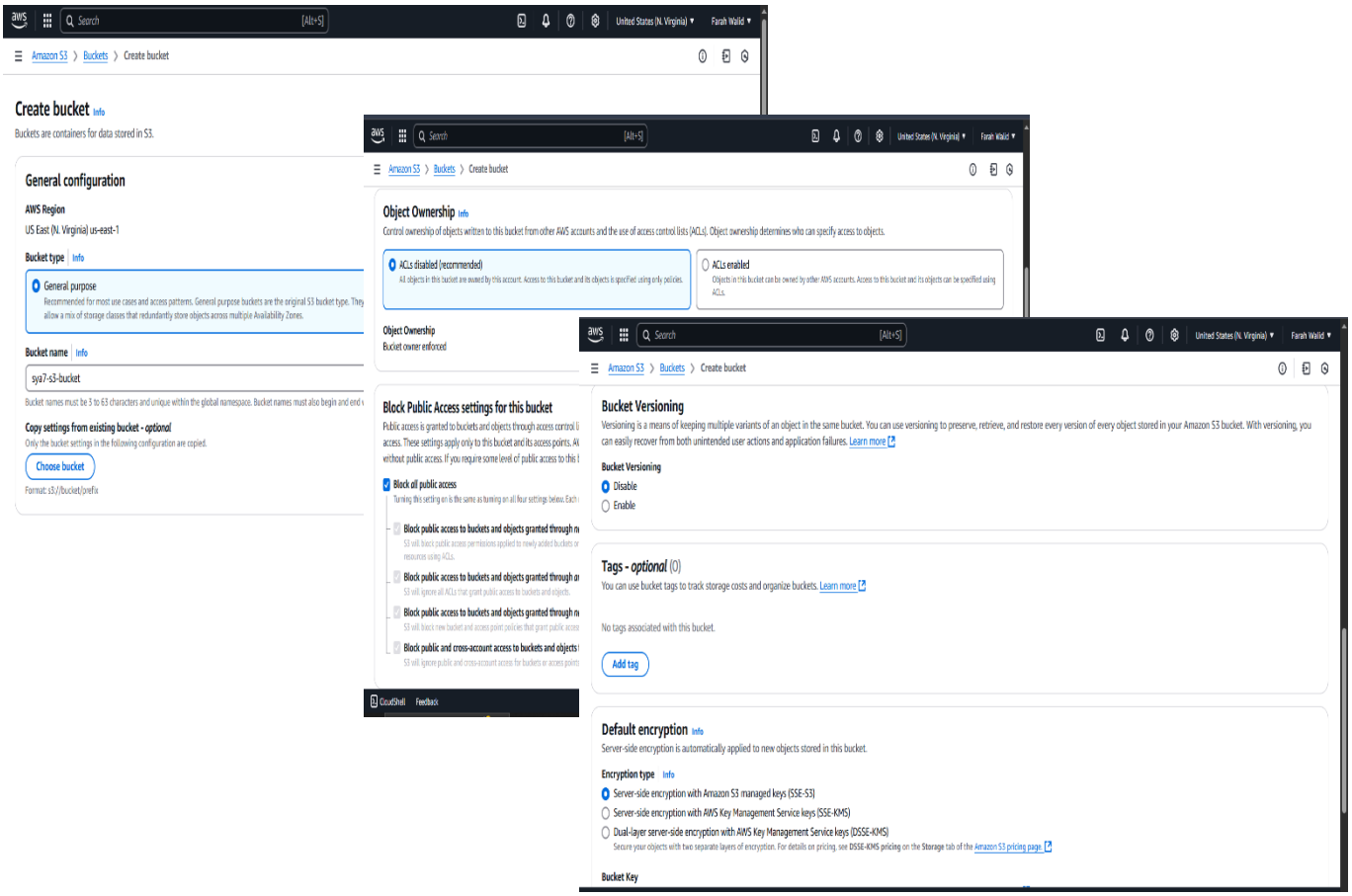


- ✓ Go to "Subnet Associations" tab, Click "Edit subnet associations".
- ✓ Select "project-subnet".
- ✓ Click "Save associations".

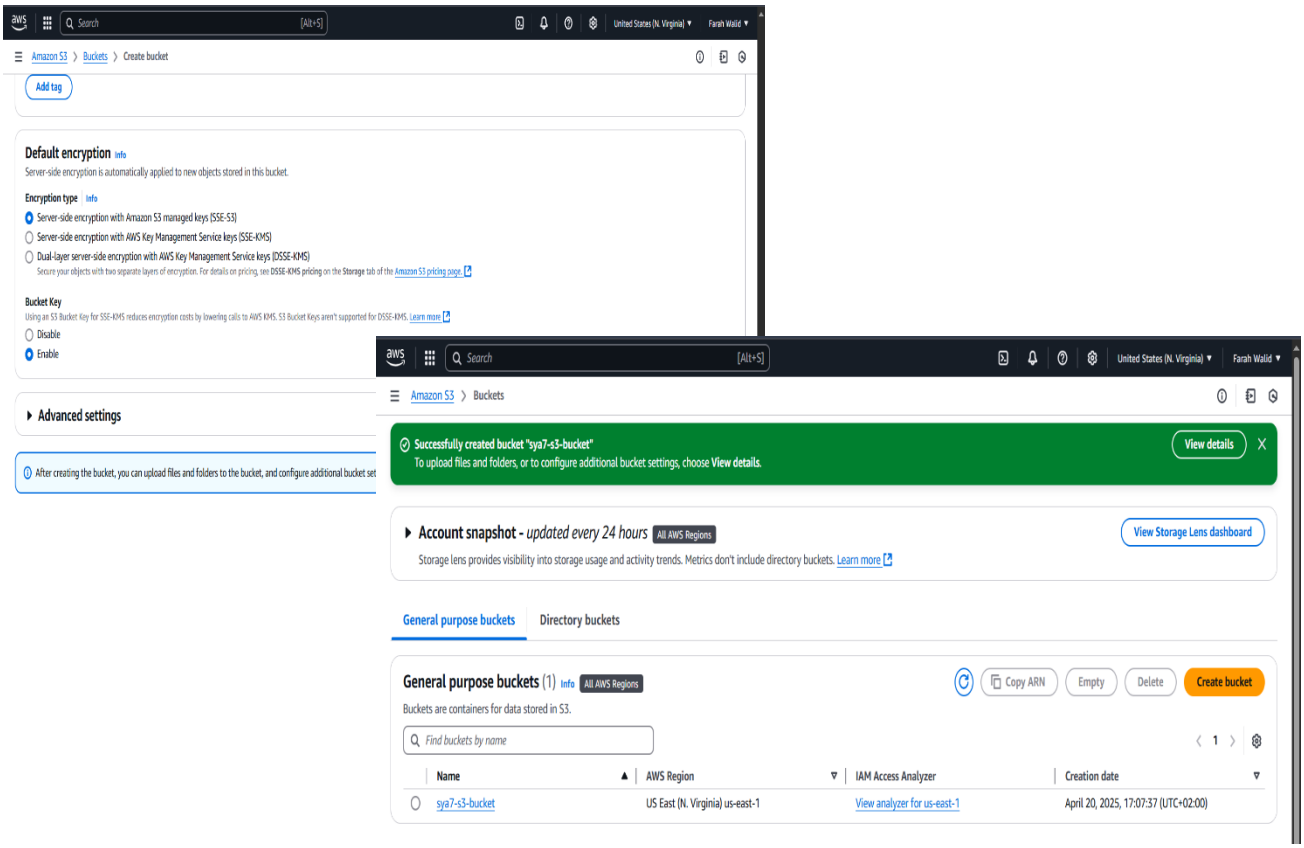
In AWS, **associations** link a **route table** to a **subnet**. This tells AWS which routing rules apply to that subnet. Each **subnet must be associated with one route table.**

Creating Amazon S3 Bucket:

Amazon S3 (Simple Storage Service) is a scalable object storage service provided by AWS that is used to store and retrieve any amount of data at any time, from anywhere on the web. Designed for high durability, availability and performance. Making it ideal for use cases like backup, data retrieving, cloud-native applications.



- ✓ Search for S3.
- ✓ Click "Create Bucket".
- ✓ Bucket name: We have chosen a unique name "sya7-s3-bucket".
- ✓ AWS Region: "us-east-1".
- ✓ Object Ownership: Keep "ACLs disabled (recommended)".
- ✓ Block Public Access settings for this bucket: Keep all boxes checked "Blocked". Access will be granted via IAM role.



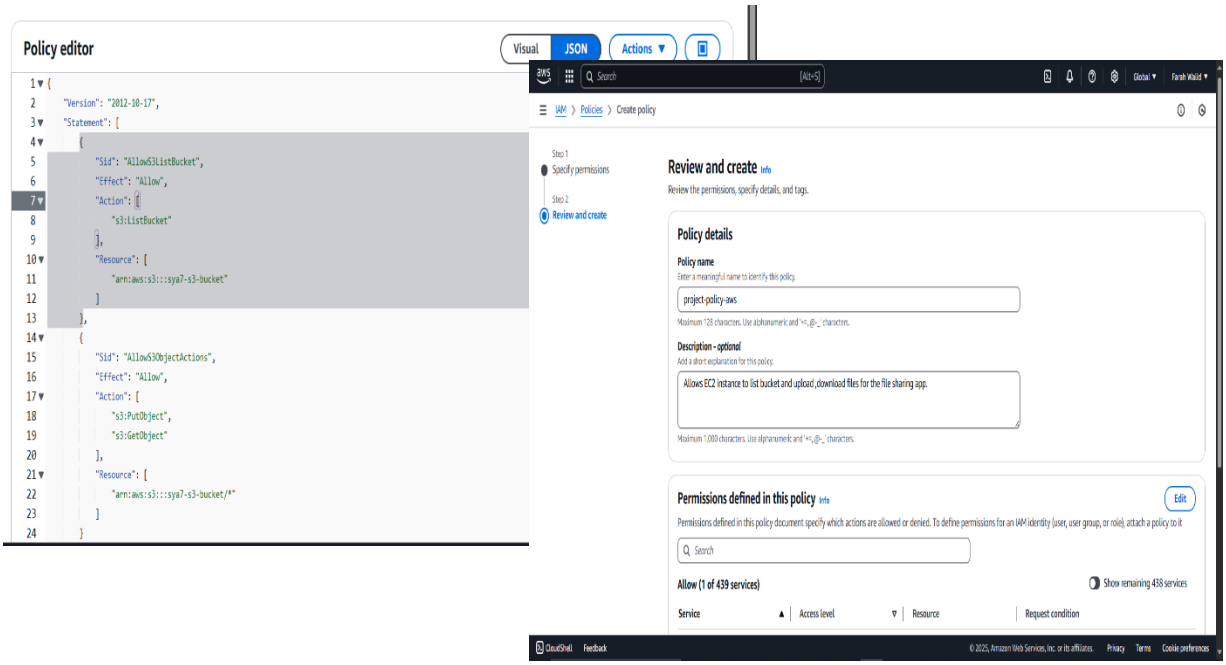
- ✓ Then leave other settings default.
- ✓ Click "Create Bucket".
- Successfully we created our S3-Bucket.

Features of Amazon S3:

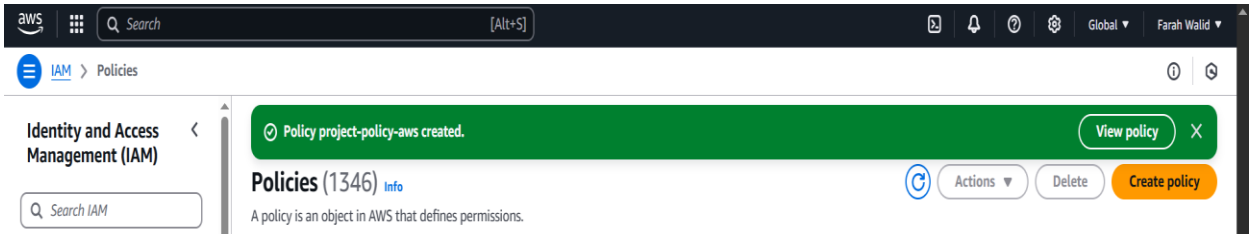
- **Durability**: 99.999999999% (11 9s) durability by redundantly storing data across multiple facilities.
- **Scalability**: Virtually unlimited storage capacity.
- **Security**: Supports encryption (at rest and in transit), IAM integration, and access logging.
- **Versioning**: Keeps multiple versions of an object to recover from accidental overwrites or deletions.
- **Lifecycle Management**: Automatically transitions objects to different storage classes or deletes them based on rules.

Creating a policy:

Policies are JSON documents that define permissions, they specify what actions are allowed or denied on which resources.

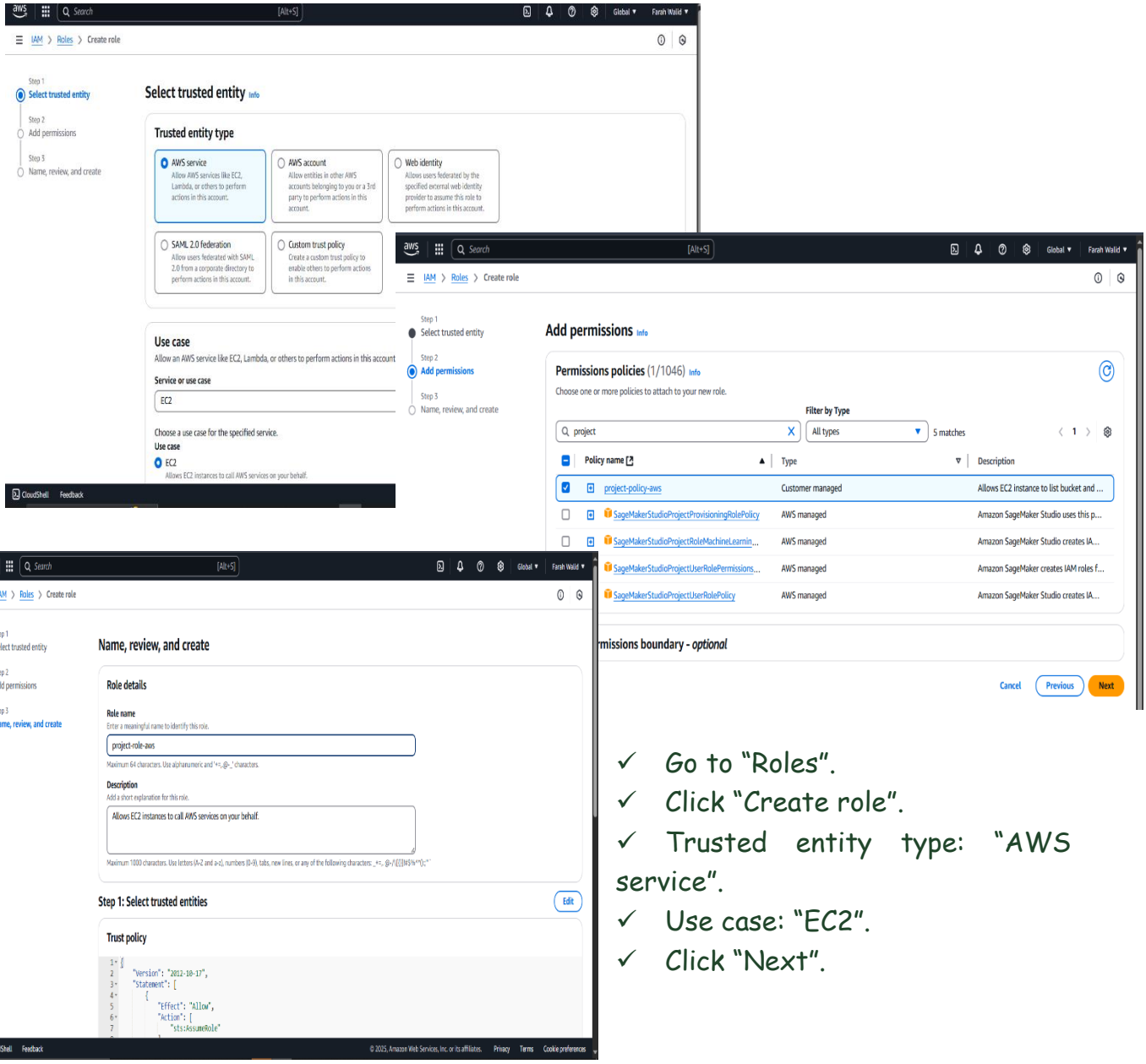


- ✓ Click "Create policy".
- ✓ Select "JSON " tab.
- ✓ Then write your policy code concerning our bucket name "sya7-s3-bucket".
- ✓ Click "Next: Tags", then "Next: Review".
- ✓ Policy Name: "project-policy-aws".
- ✓ Description: "Allows EC2 insrance to list bucket and upload/download files for the file sharing app."
- ✓ Click "Create policy".
- ✓ Then go now to create the IAM Role.

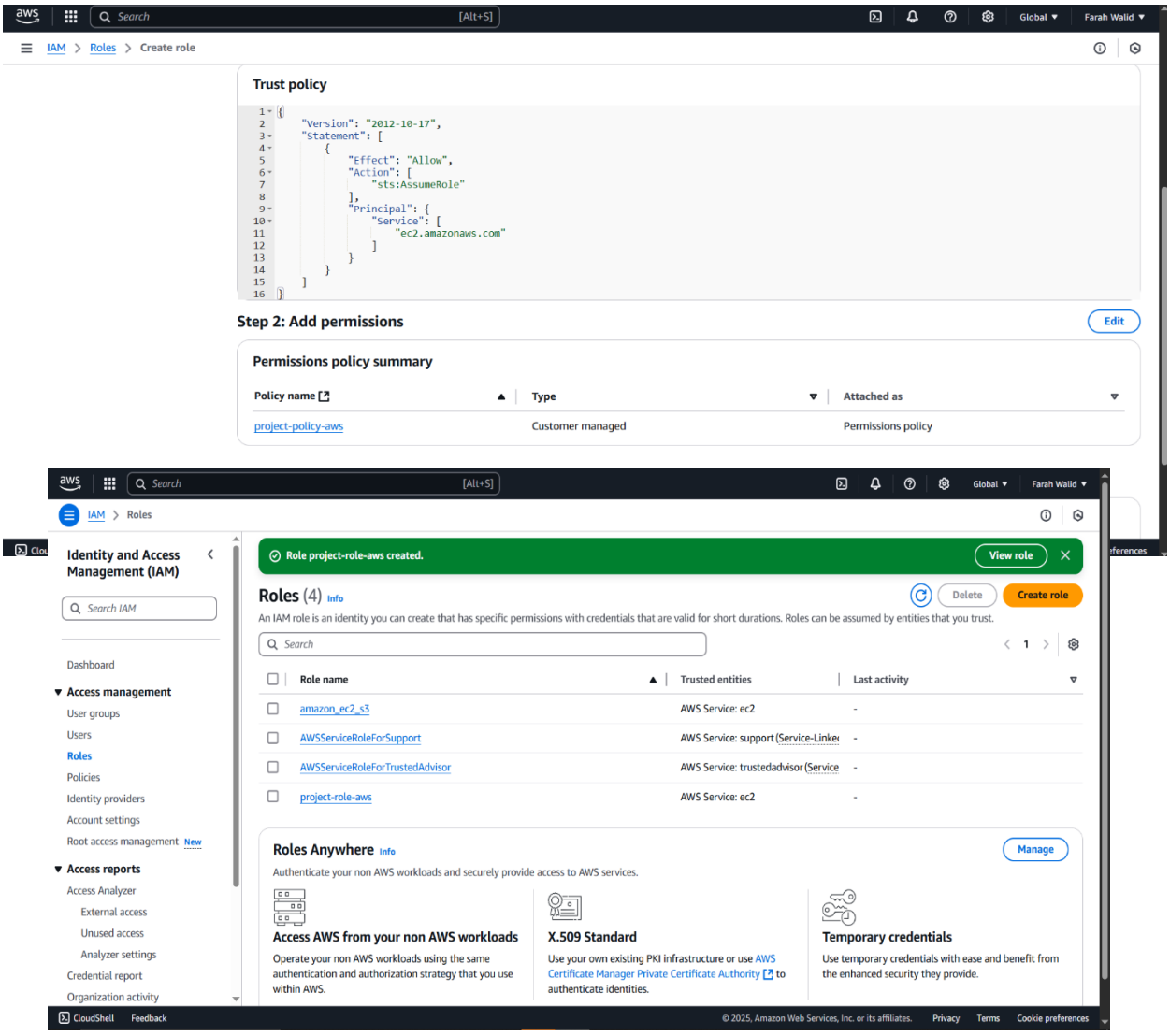


Creating IAM Role:

AWS IAM (Identity and Access Management) is a web service that helps us securely control access to AWS services and resources. It enables us to create and manage AWS users & groups, and define permissions that allow or deny access to specific AWS resources.



- ✓ Go to "Roles".
- ✓ Click "Create role".
- ✓ Trusted entity type: "AWS service".
- ✓ Use case: "EC2".
- ✓ Click "Next".

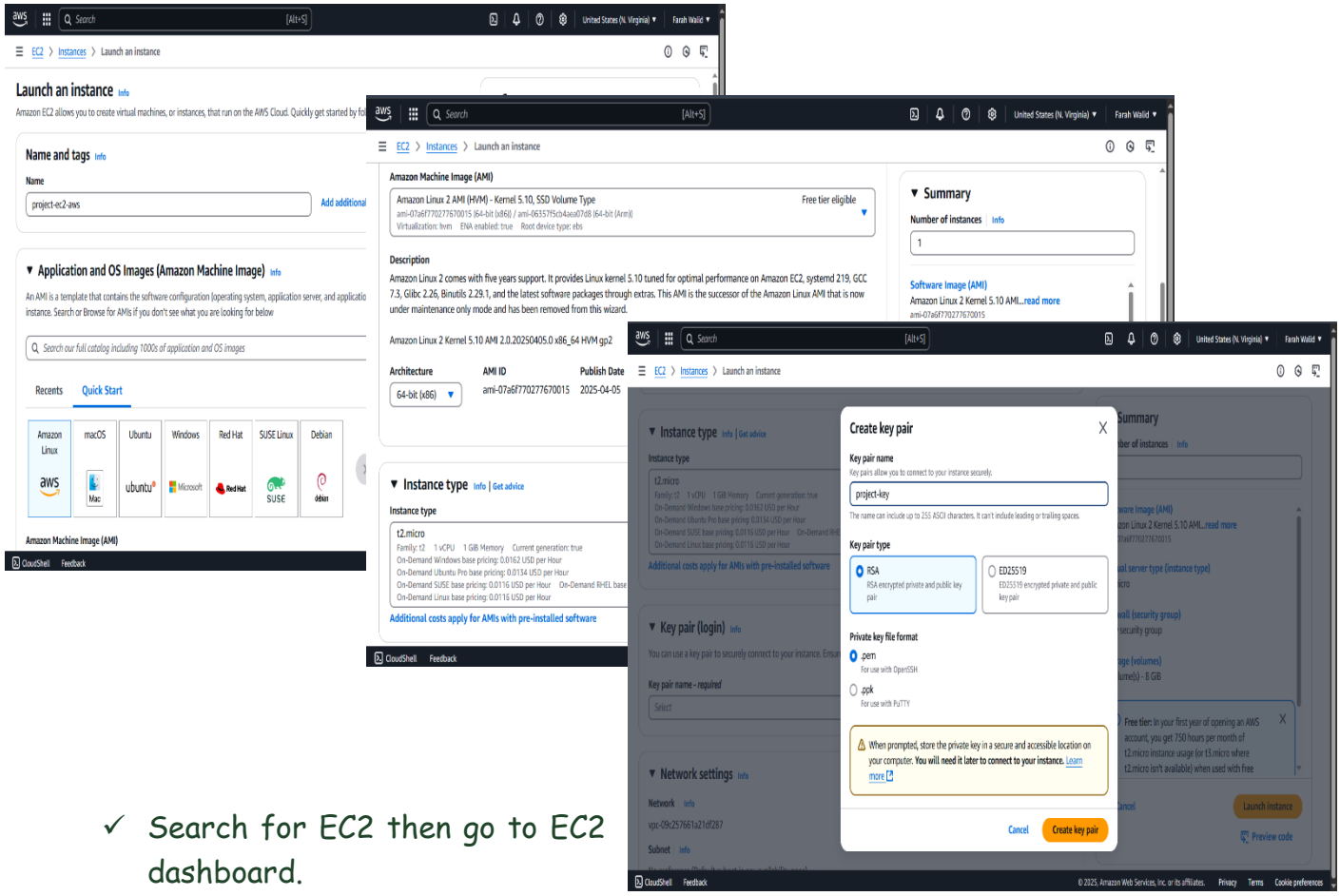


- ✓ Add Permissions: Select the policy we have just made before "project-policy-aws".
- ✓ Click "Next".
- ✓ Then Role Name: "project-role-aws".
- ✓ Description: "IAM role for EC2 instance to access S3 bucket for file sharing app."
- ✓ Verify the trusted entity is EC2 and our policy is attached then Click "Create role".

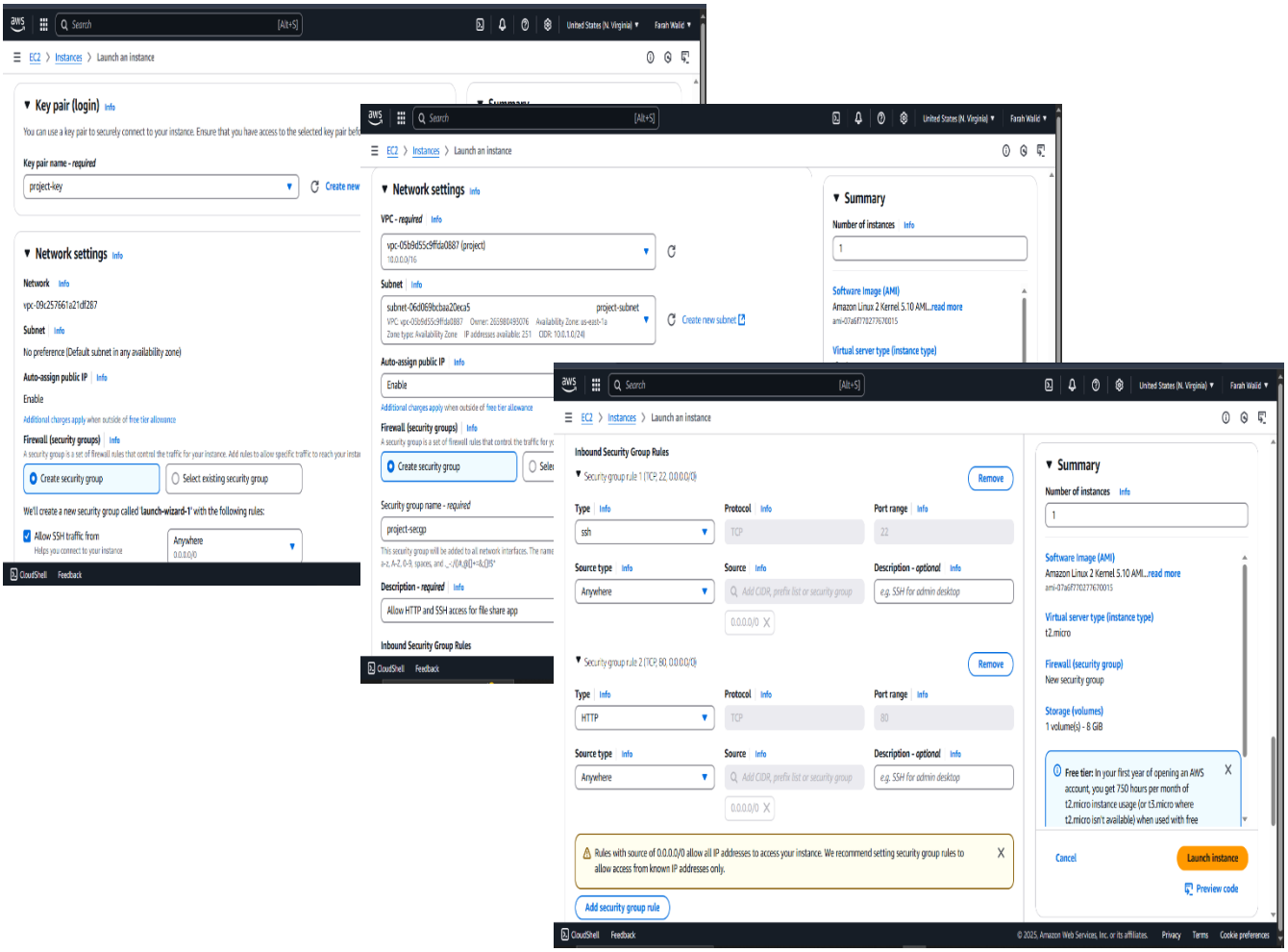
IAM is essential for applying the **principle of least privilege**, where users and applications are granted only the permissions they need to perform their tasks.

Creating EC2 Instance:

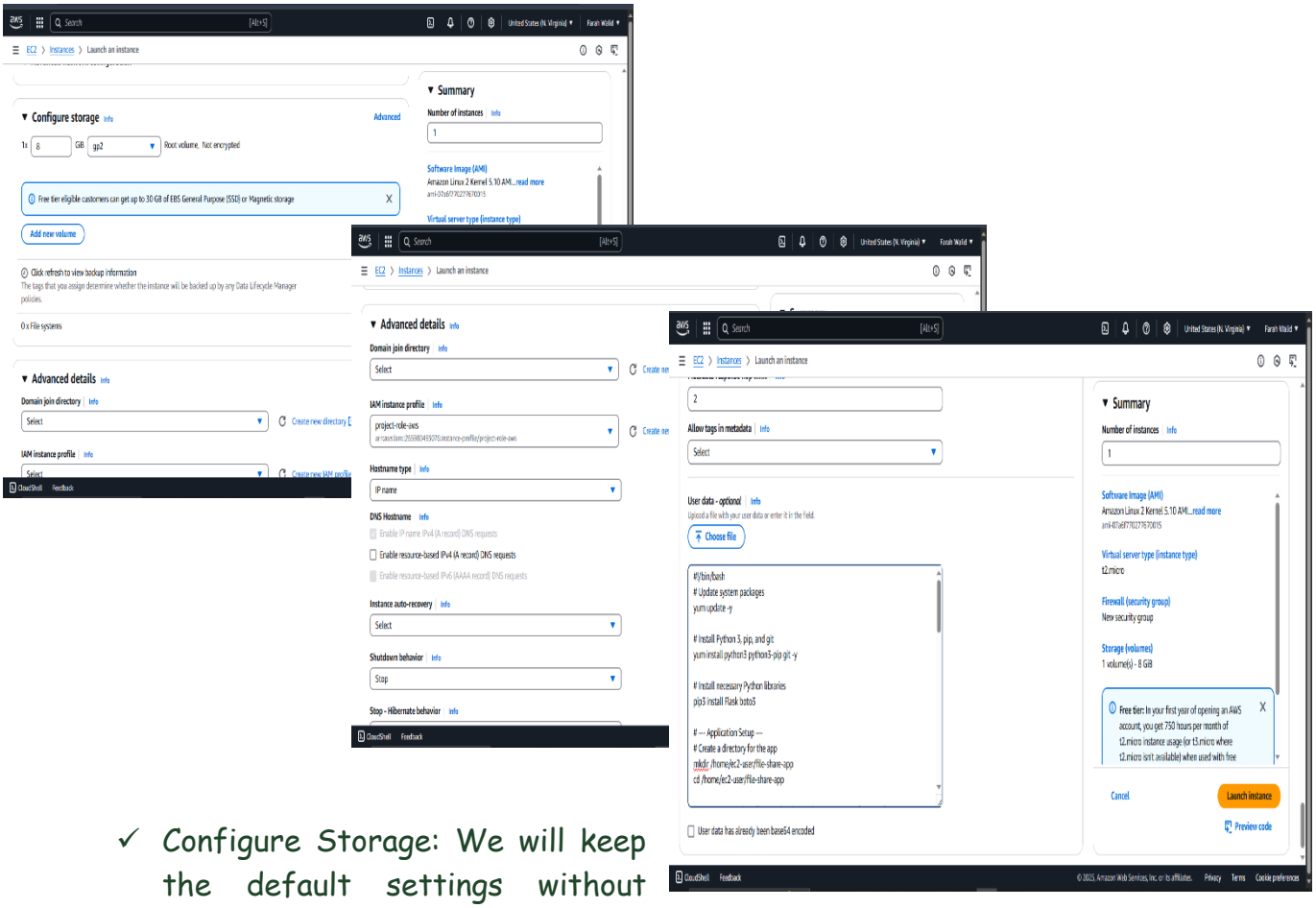
Amazon EC2 (Elastic Compute Cloud) is a web service provided by AWS that offers secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers by allowing them to launch and manage virtual servers called instances…on demand.



- ✓ Search for EC2 then go to EC2 dashboard.
- ✓ Click "Launch instances".
- ✓ Name: "project-ec2-aws".
- ✓ Application and OS Images (AMI): Select "Amazon Linux" then Amazon Linux 2 AMI(HVM)- Kernel 5.10,SSD Volume Type "Free tier eligble".
- ✓ Instance Type: "t2.micro" (Free tier eligible).
- ✓ Key pair (login): Create new key pair ("project-key")
- ✓ Download the .pem file then keep it secure.
- ✓ Select "project-key".

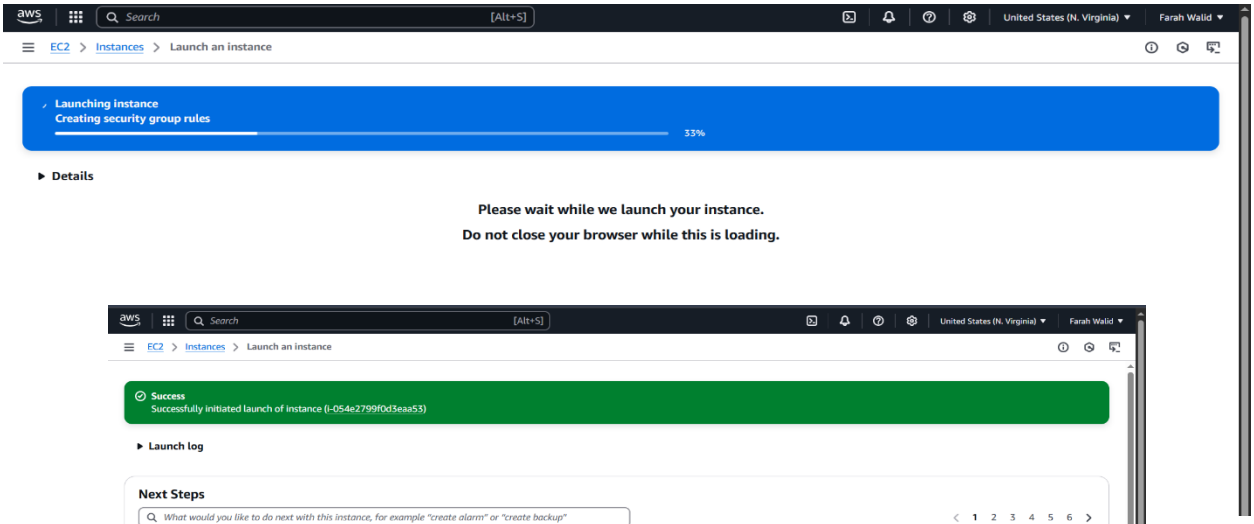


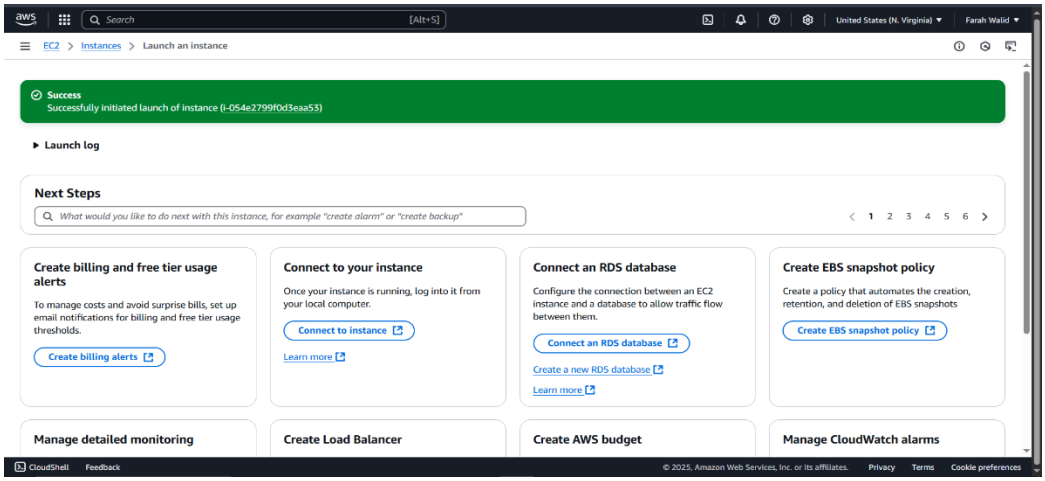
- ✓ Click “Edit” in Network settings .
- ✓ VPC: Select “project” that we recently created.
- ✓ Subnet: Select “project-subnet”.
- ✓ Auto-assign public IP: “Enable”.
- ✓ Firewall (Security Groups): Click “Create security group”.
- ✓ Security group name: “project-secgp”.
- ✓ Description: “Allow HTTP and SSH access for file share app”.
- ✓ Inbound security group rules: we made 2 rules (HTTP & SSH).
- ✓ SSH rule allows connection for troubleshooting.



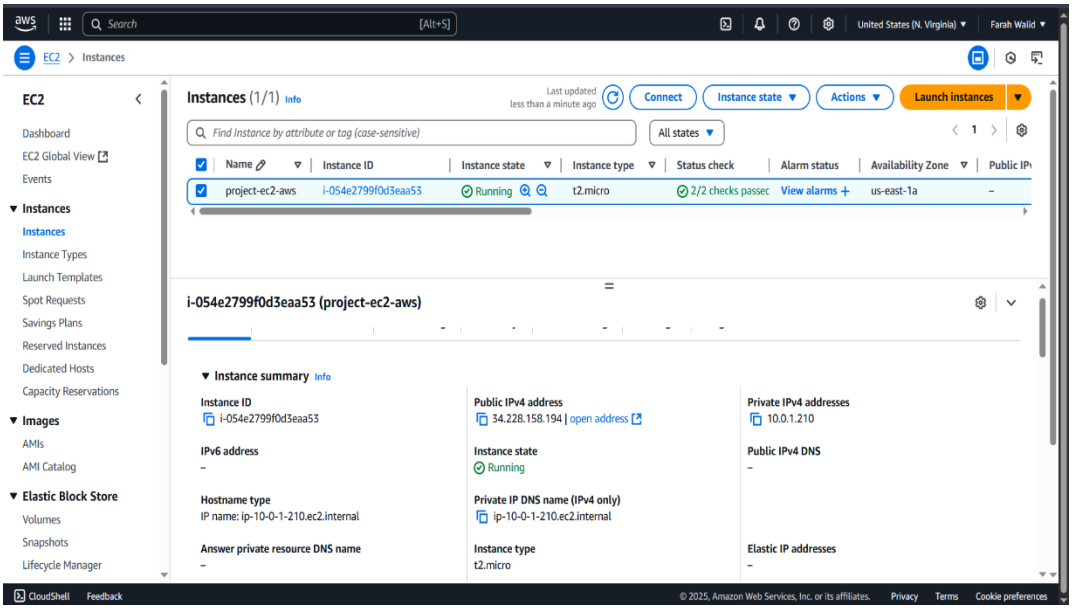
- ✓ Configure Storage: We will keep the default settings without change (Free Tier Eligible).

- ✓ Expand the Advanced details section Then:
  IAM instance profile: Select "project-role-aws" to allow the EC2 instance to interact with S3 without hardcoding credentials.
  User Data: We got the code with replacing the bucket name with ours "sya7-s3-bucket".



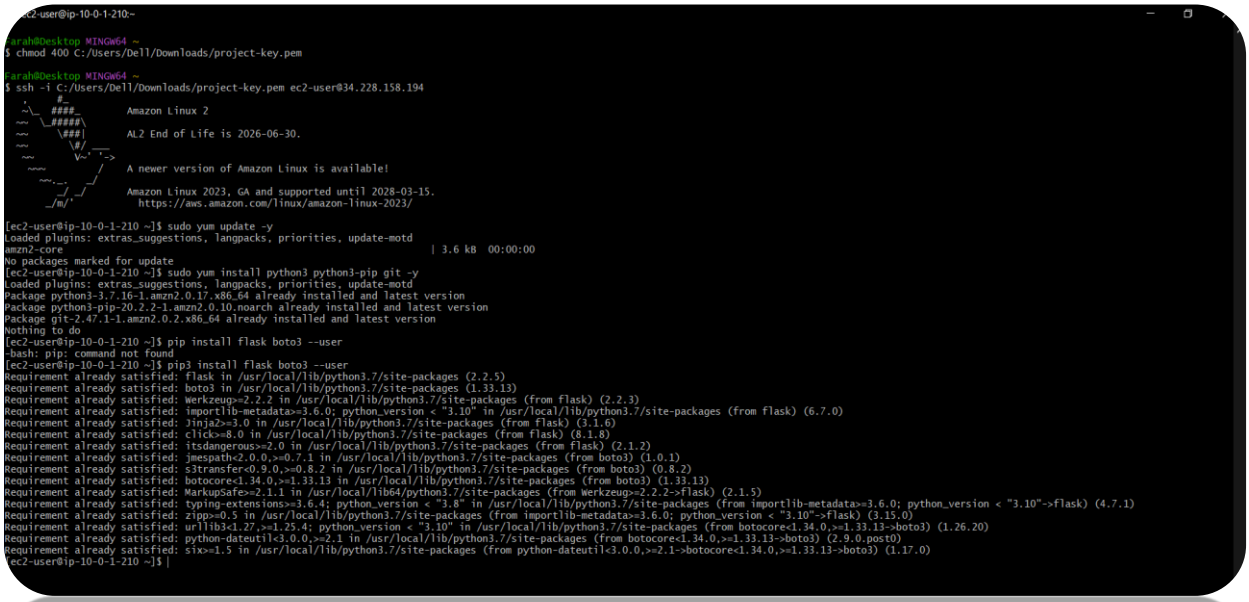


✓ Click "Launch Instance".



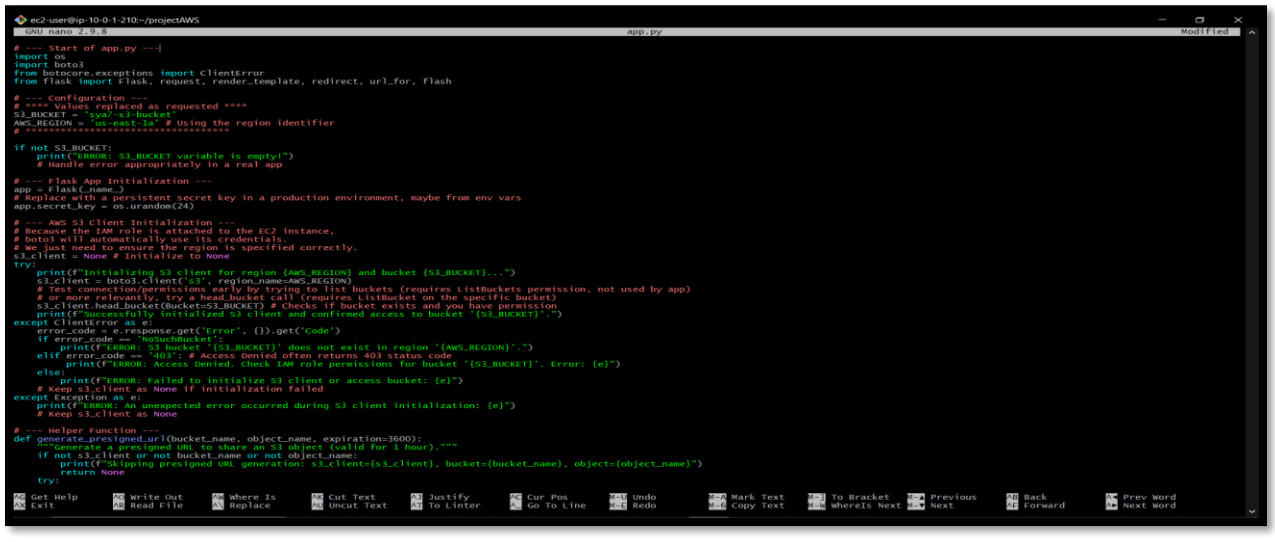
Successfully we created our "project-ec2-aws" instance ^_^

The Web Application Development

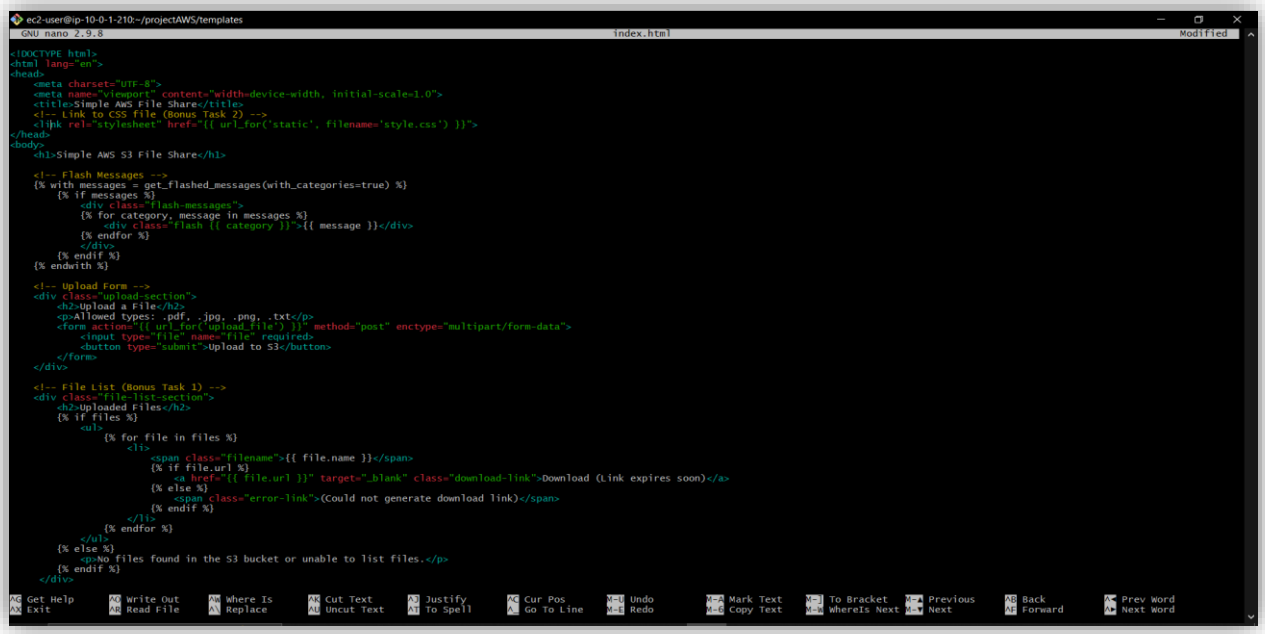


This code illustrates the connection of our EC2 via SSH.

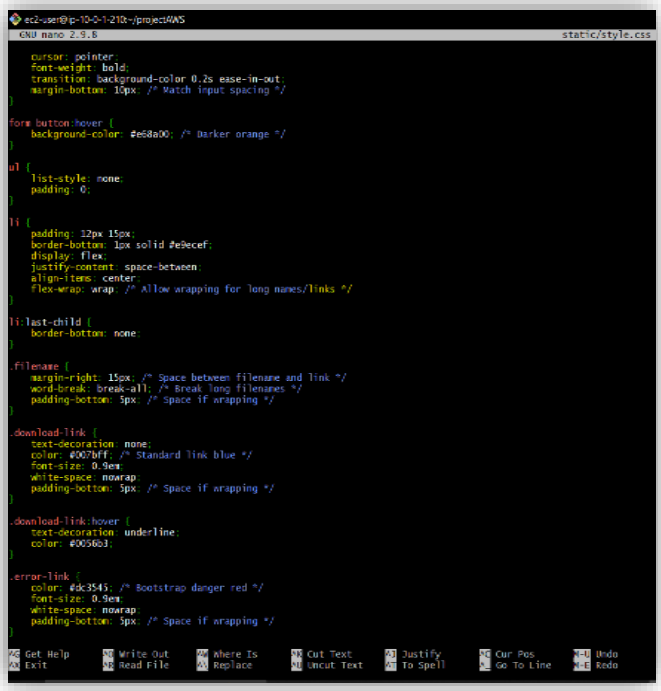
- Now use the command "nano app.py" to make a file and start wirting our flask code in it.



Then use the command "nano index.html" to create our HTML template.

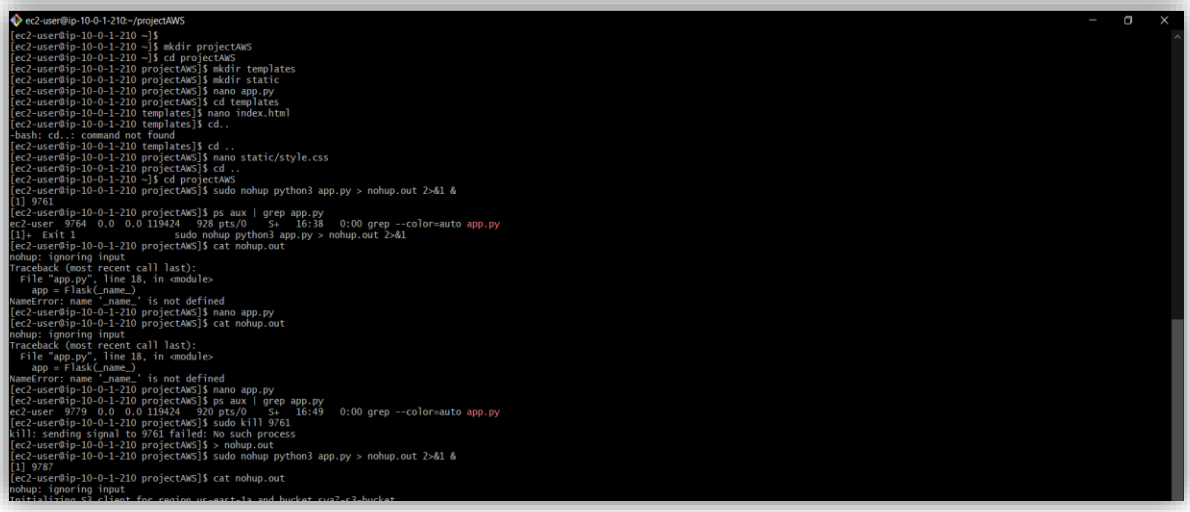


Then use the command "nano static/style.css" to create our CSS file.

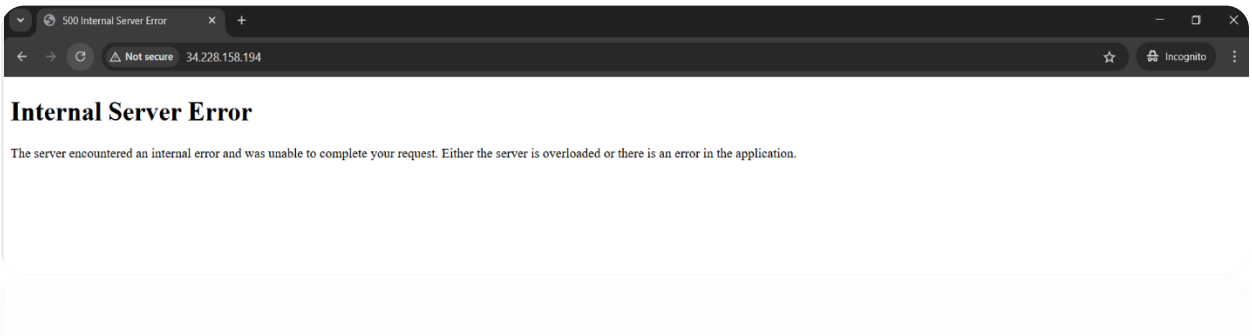


Let's start runnig our program:

```
[ec2-user@ip-10-0-1-210 ~]$
[ec2-user@ip-10-0-1-210 ~]$ mkdir projectAWS
[ec2-user@ip-10-0-1-210 ~]$ cd projectAWS
[ec2-user@ip-10-0-1-210 projectAWS]$ mkdir templates
[ec2-user@ip-10-0-1-210 projectAWS]$ mkdir static
[ec2-user@ip-10-0-1-210 projectAWS]$ nano app.py
[ec2-user@ip-10-0-1-210 projectAWS]$ cd templates
[ec2-user@ip-10-0-1-210 templates]$ nano index.html
[ec2-user@ip-10-0-1-210 templates]$ cd..
-bash: cd..: command not found
[ec2-user@ip-10-0-1-210 templates]$ cd ..
[ec2-user@ip-10-0-1-210 projectAWS]$ nano static/style.css
[ec2-user@ip-10-0-1-210 projectAWS]$ cd ..
[ec2-user@ip-10-0-1-210 ~]$ cd projectAWS
[ec2-user@ip-10-0-1-210 projectAWS]$ sudo nohup python3 app.py > nohup.out 2>&1 &
[1] 9761
[ec2-user@ip-10-0-1-210 projectAWS]$
```

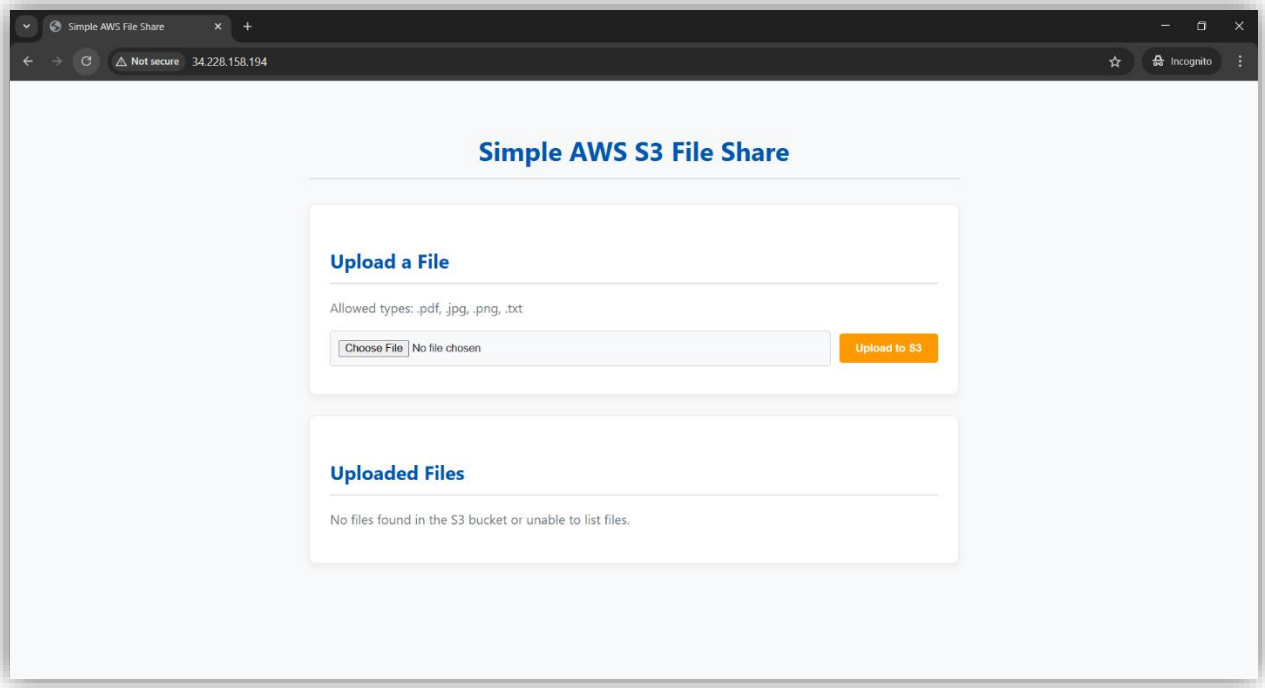


The runnig commands are all written as shown and then we got the IPv4 address of our "project-aws-ec2" instance to the a new tab in the browser, but… there was a problem due to some syntax errors in the flask code.



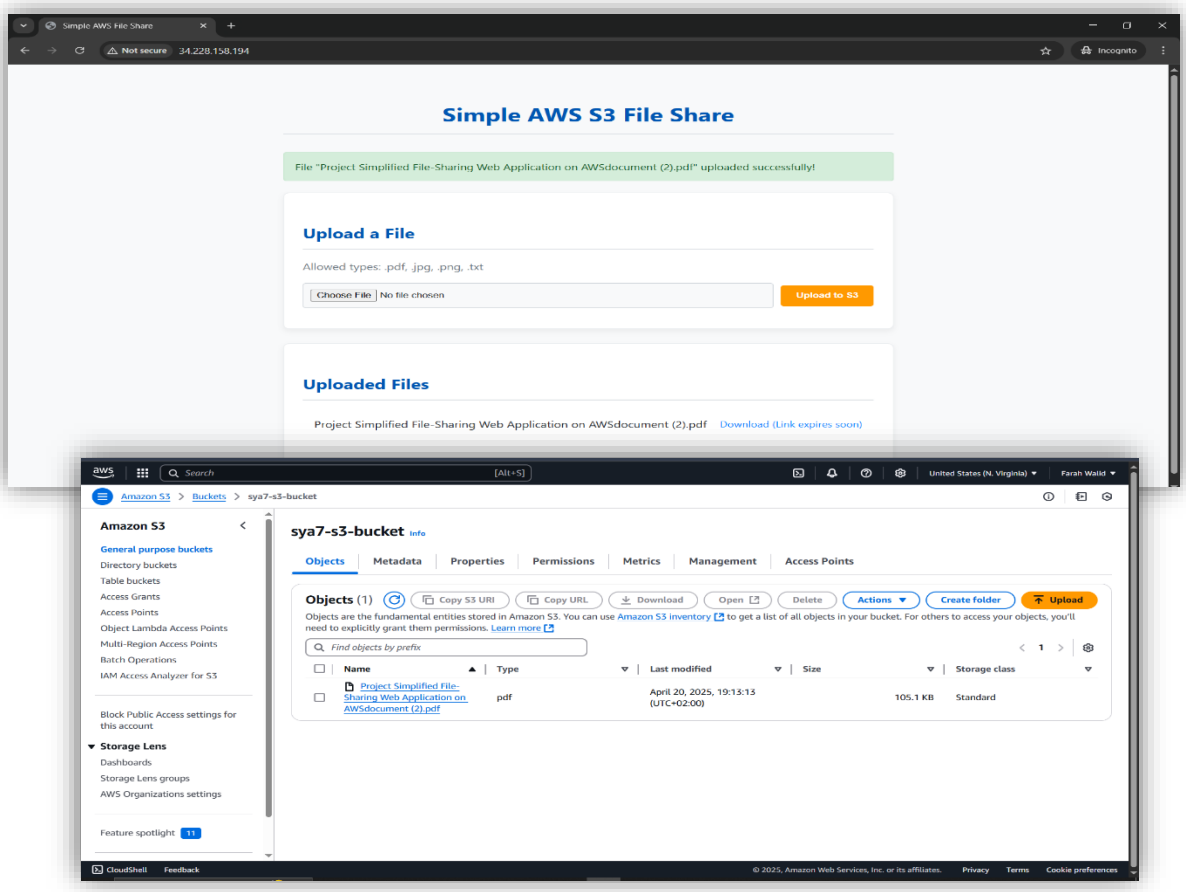
After solving the errors problem, Finally our web application worked.

Now let's show it and start testing.



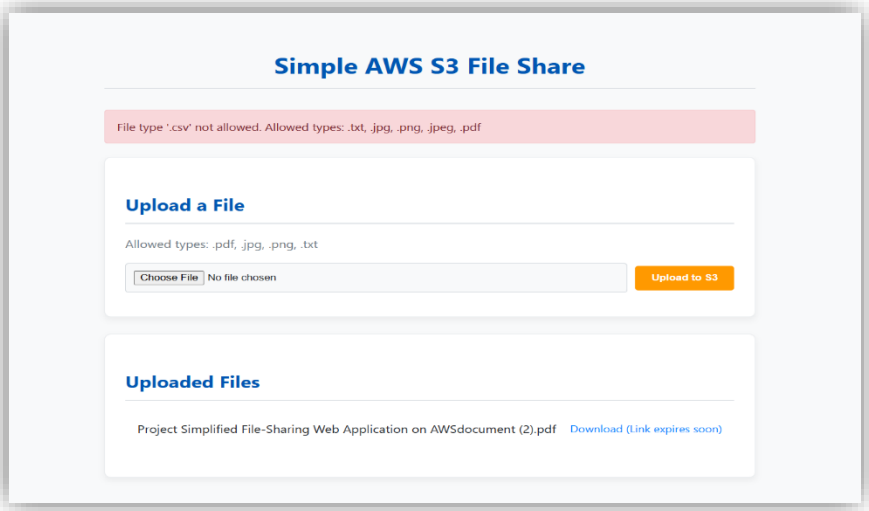
Finally Here is our web application interface.

NOW try to upload a file of an allowed type:



The file is successfully uploaded ^_^

If we tried to upload a file of a not allowed type it will deny uploading it.



Team Role	Team Member
AWS Account Setup	Farah
EC2 & IAM Setup	Basmala
S3 & File Handling	Rodina
Web App development	Nour
VPC & Testing	Mariam
Test and Documentation	Farah

Drive Link: ^-^

https://drive.google.com/file/d/1A5fK4JQgDUH7oINnupxYirtG9nonPiql/view?usp=sharing

Demo Link: ^-^

http://34.228.158.194