Access Amazon S3 from a VPC

By Haroon Zaman | December 2025

Introduction

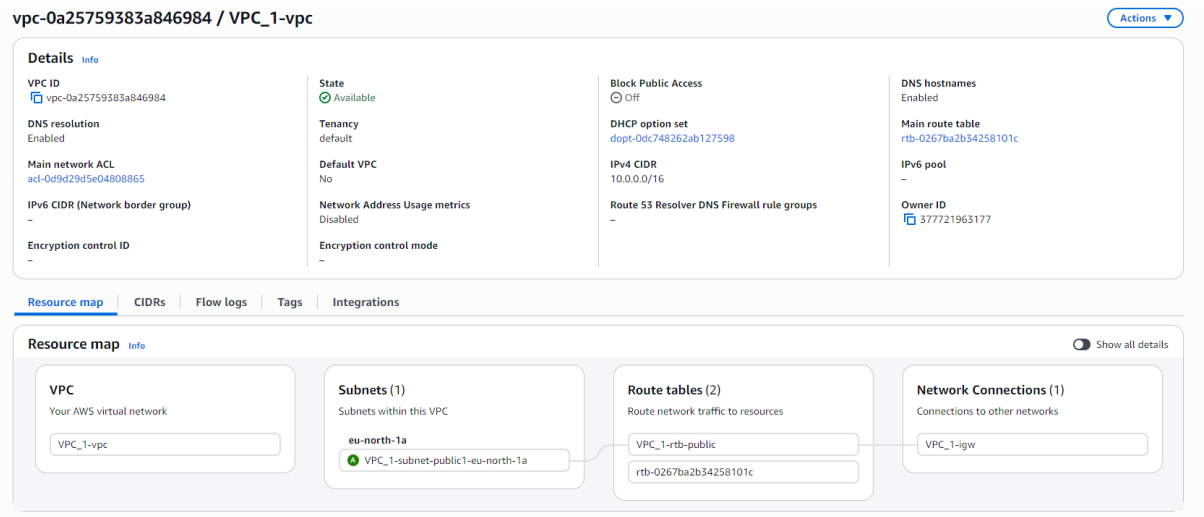
In this project, I learned how to allow resources inside a **Virtual Private Cloud (VPC)** to securely interact with **Amazon S3**. By default, resources in a VPC cannot access AWS services unless proper permissions and network configurations are in place. This project focuses on enabling that access in a controlled and secure way.

The goal of this project was to understand how EC2 instances communicate with AWS services like S3, how access permissions are granted using AWS credentials, and how traffic flows from a VPC to external AWS services.

In this project, I performed the following tasks:

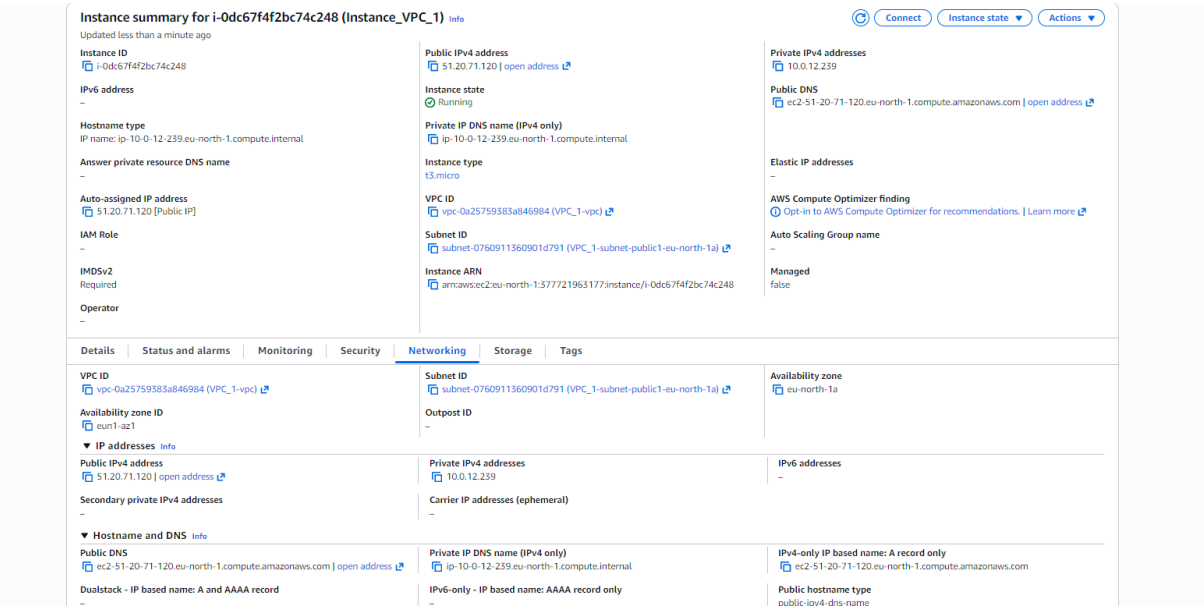
- Set up a **VPC with an EC2 instance**
- Used **AWS access keys** to give the EC2 instance permission to interact with AWS services
- Accessed and interacted with **Amazon S3 directly from the EC2 instance**
- Observed how network and security configurations allow private resources to reach AWS-managed services

This project helped me understand how AWS services work together, how identity and access management controls permissions, and how cloud resources securely communicate with services outside the VPC while staying within AWS's infrastructure.

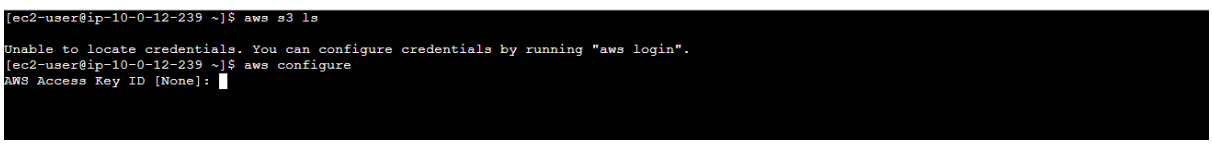


Create VPC and EC2 Instance

- I created a new VPC and named it **VPC_1**.
- I assigned the IPv4 CIDR block **10.0.0.0/16** to provide a large private IP range.
- I selected **one Availability Zone** for a simple setup.
- I created **one public subnet** and **no private subnet**, because this project only requires a public EC2 instance to access Amazon S3.
- I did not configure a **NAT Gateway**, since NAT is only needed when private subnets require outbound internet access.
- After reviewing the settings, I created the VPC.



- Next, I created an EC2 instance and named it **Instance_1**.
- I selected **Amazon Linux** as the operating system.
- I chose the **t3 instance type**, which is suitable for lightweight testing and learning.
- I did **not create a key pair**, as SSH access was not required for this task.
- In the network settings, I selected **VPC_1** and its public subnet.
- I enabled **Auto-assign Public IP** so the instance could reach AWS services.
- I created a new **Security Group** and allowed **SSH**.
- Finally, I launched the EC2 instance successfully.



AWS Access Keys and Credentials

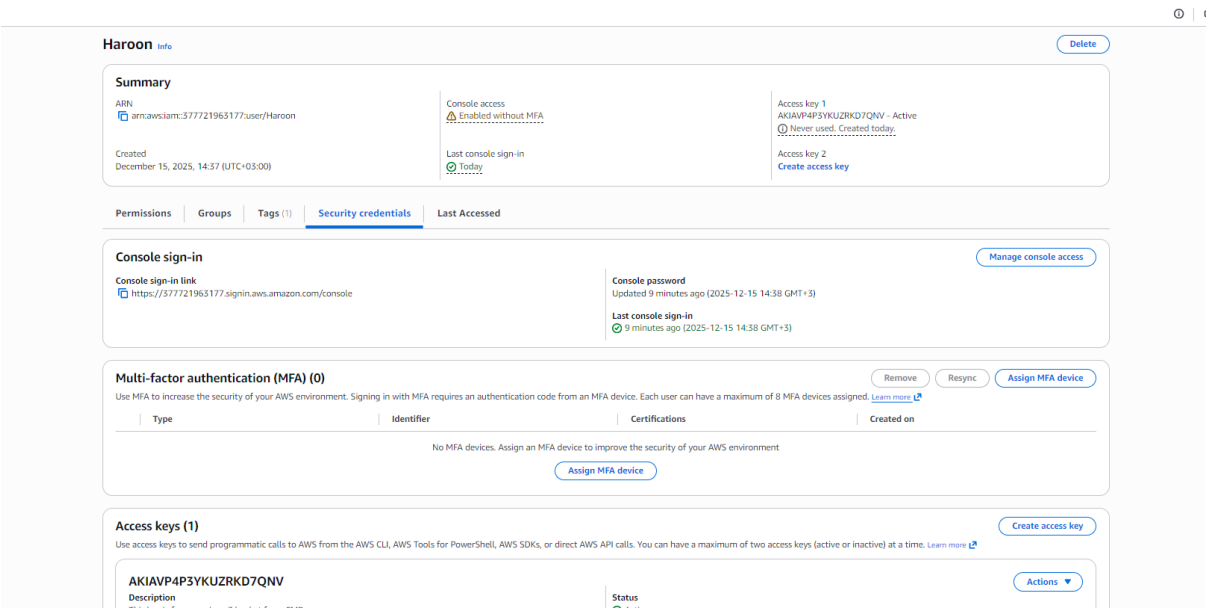
- After creating the instance, I connected to it and ran the command:
  `aws s3 ls`
- This returned an error saying **"Unable to locate credentials"**, which means the instance did not yet have permission to access AWS services.
- To fix this, I ran `aws configure`, which requires an **AWS Access Key ID**.

What is an Access Key ID?

- An **Access Key ID** is part of an AWS credential.
- AWS credentials are similar to a **username and password**.
- The **Access Key ID** acts like the **username**, and the **Secret Access Key** acts like the password.
- You don't get an access key automatically — it must be created through **AWS IAM**.

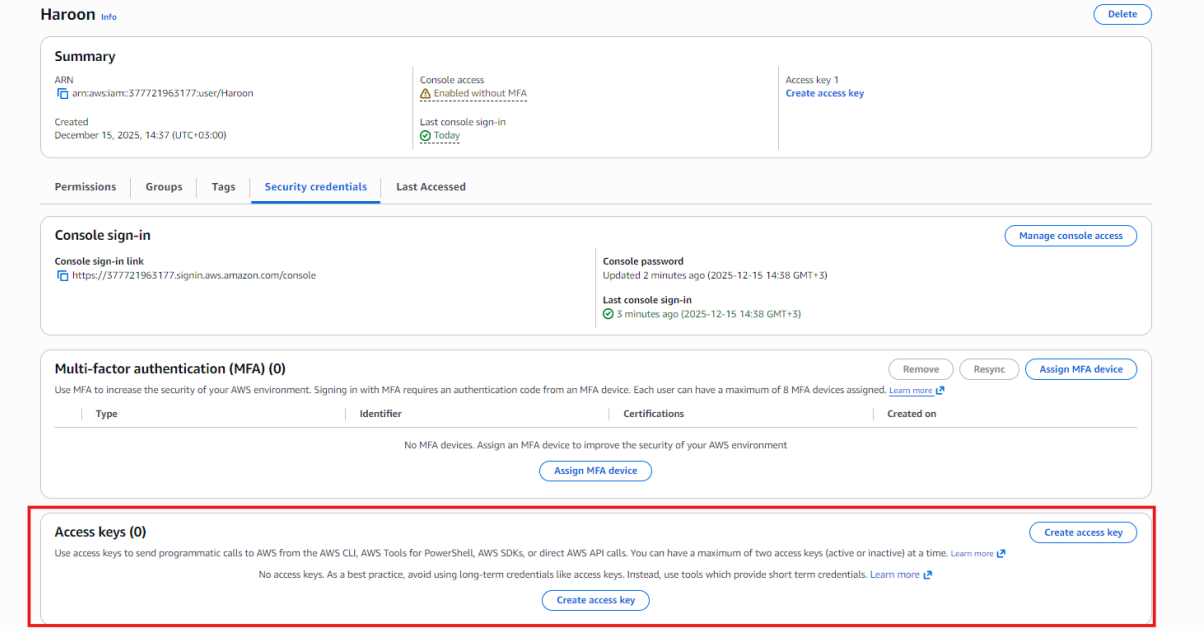
Difference Between Access Keys and Key Pairs

- **Key Pairs** are used for **logging into EC2 instances via SSH**.
- **Access Keys** are used by **applications and services** (like AWS CLI) to authenticate and interact with AWS resources such as S3.
- In simple terms:
  - **Key Pair → Login to EC2**
  - **Access Key → Access AWS services**



Creating an IAM User for Access Keys

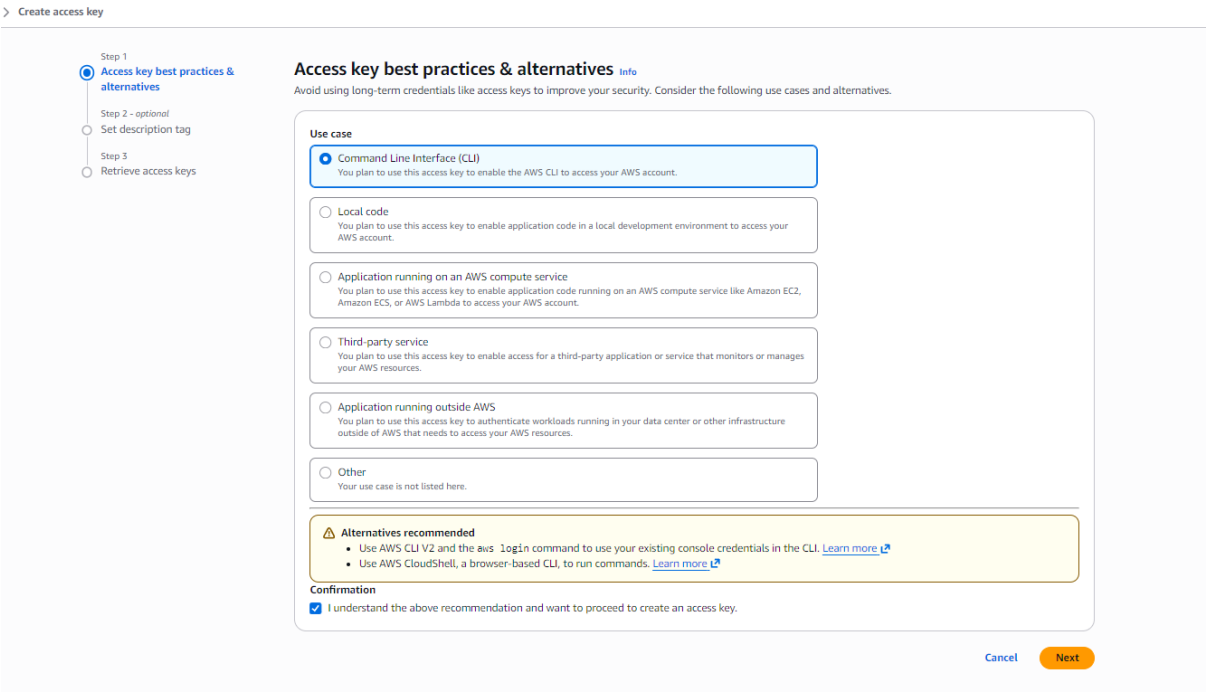
- To create an **Access Key**, I first needed an **IAM user**, because access keys are managed at the user level.
- Since I did not have any IAM user already created, I created one myself.
- I created a new IAM user and named it **Haroon**.
- I granted this user **Administrator permissions** so it would have full access to AWS services for this project.
- After creating the user, I was able to manage and generate **Access Keys** for it.



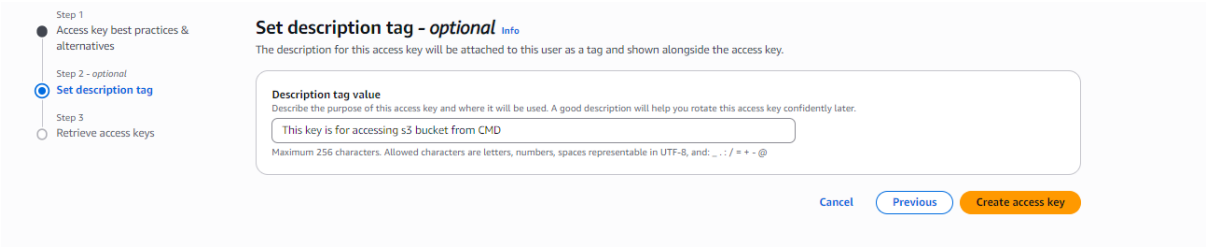
- After creating the IAM user, I opened the user **Haroon** in the IAM console.
- I scrolled down to the **Access keys** section.
- From there, I clicked on **Create access key** to generate new credentials for AWS CLI access.
- After clicking **Create access key**, a new window opened.

In **Step 1**, I had to choose the **use case** for the access key.

- The available use case options were:
- **Command Line Interface (CLI)**
  Used when you want to access AWS services using the AWS CLI commands.
- **Local code**
  Used when application code running on your local machine needs to access AWS resources.
- **Application running on an AWS compute service**
  Used when applications running on EC2, ECS, or Lambda need AWS access.
- **Third-party service**
  Used when an external tool or service needs permission to manage or monitor AWS resources.

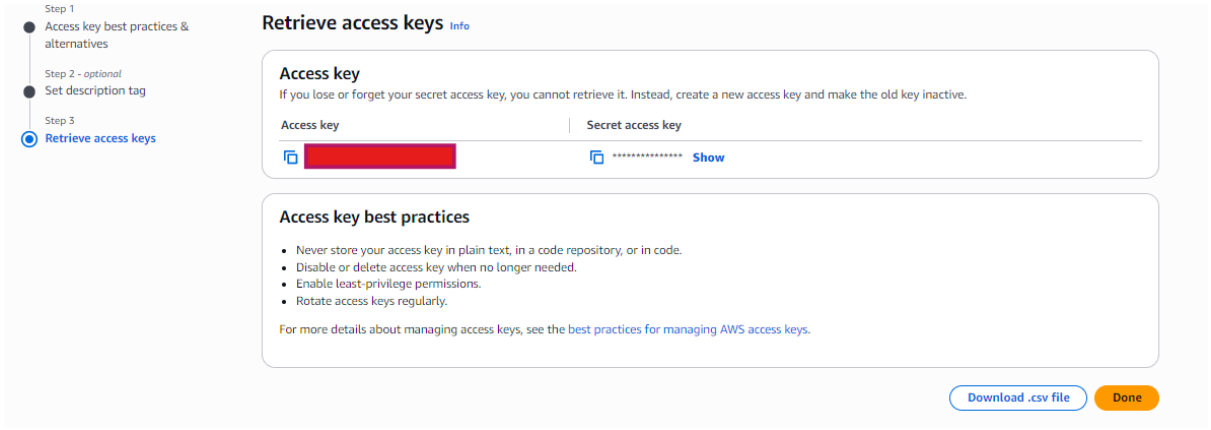


- **Application running outside AWS**
  Used when workloads running outside AWS (on-prem or another cloud) need AWS access.
- **Other**
  Used if your use case does not fit any of the listed options.
- I selected **Command Line Interface (CLI)** because I planned to use the AWS CLI from my EC2 instance.
- I checked the **confirmation box** and clicked **Next**.

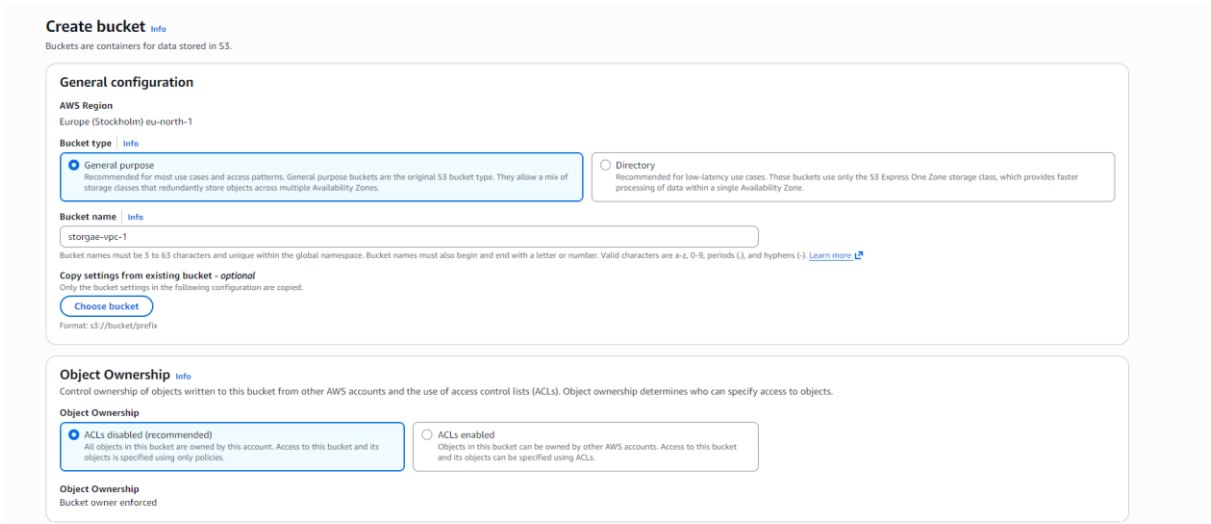


- In the next step, I added a **description** to help identify the access key later.
- After setting the description, I clicked **Create access key** to generate the credentials.

- After creating the access key, AWS generated an **Access Key ID** and a **Secret Access Key**.
- These credentials are used together to authenticate with AWS services.
- I was given the option to **download the keys as a `.csv` file** for safe storage.
- This is the only time the **Secret Access Key** is visible, so it must be saved securely.



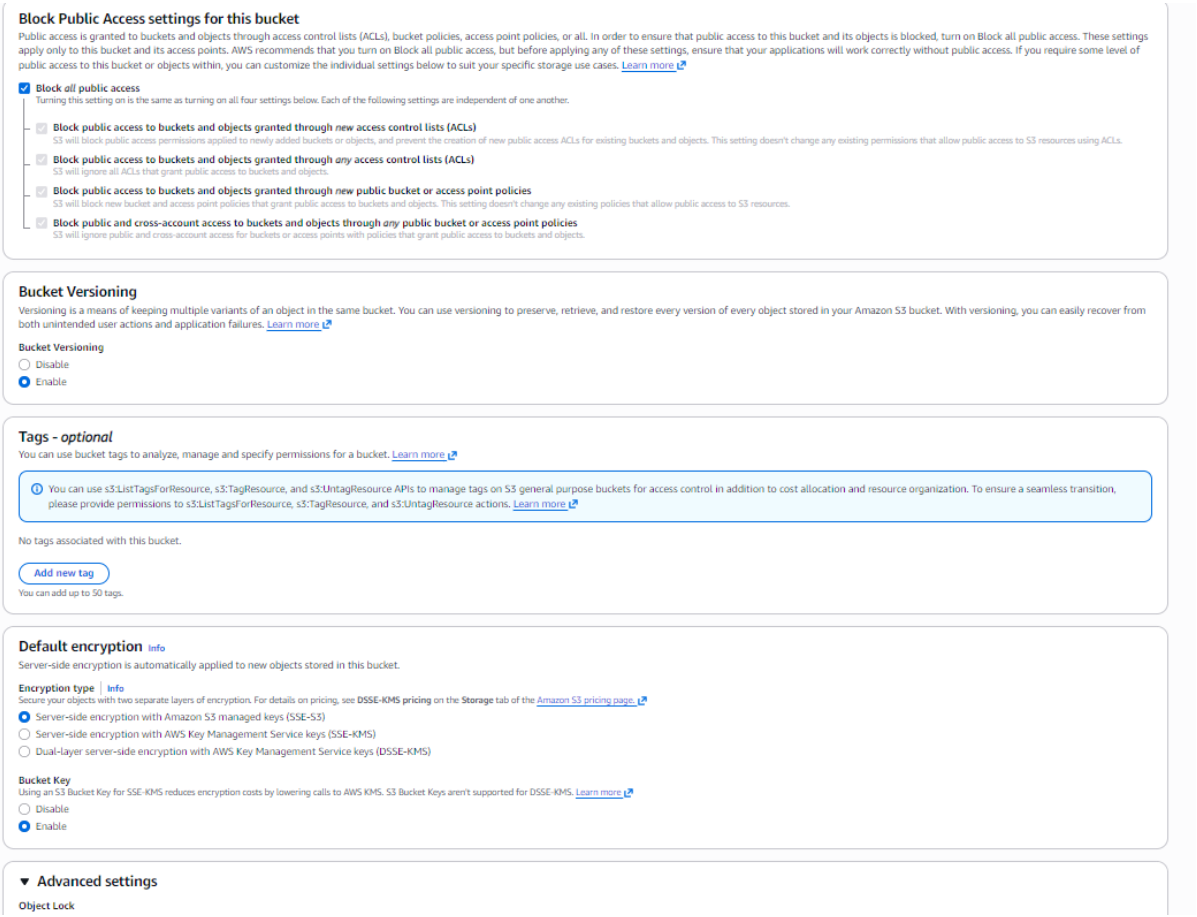
- After that, I went back to the EC2 instance I had connected to earlier.
- I entered the **Access Key ID** and then the **Secret Access Key** when prompted.
- After providing the credentials, the login and configuration were **successful**.



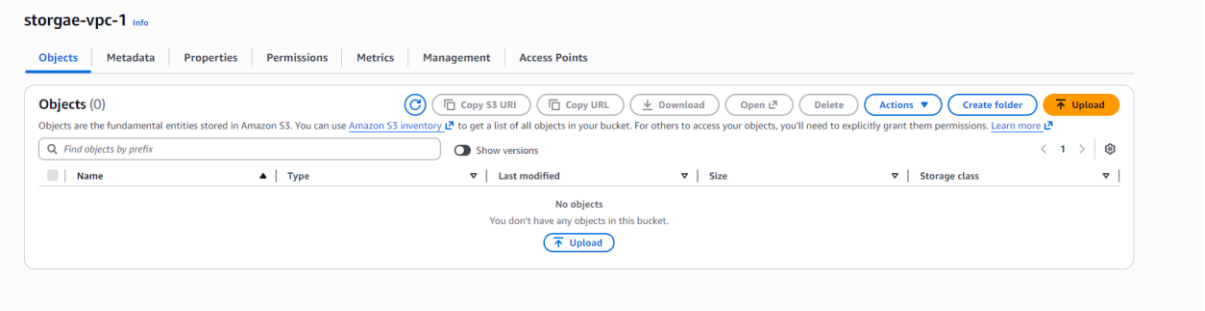
Creating the S3 Bucket

- Next, I created an **Amazon S3 bucket**.
- I searched for **S3** in the AWS console and clicked **Create bucket**.
- I selected the **General purpose** bucket type, which is used for standard object storage like files, backups, and logs.
- I gave the bucket the name **storage_vpc_1**.
- For **Object Ownership**, I kept **ACLs disabled**.

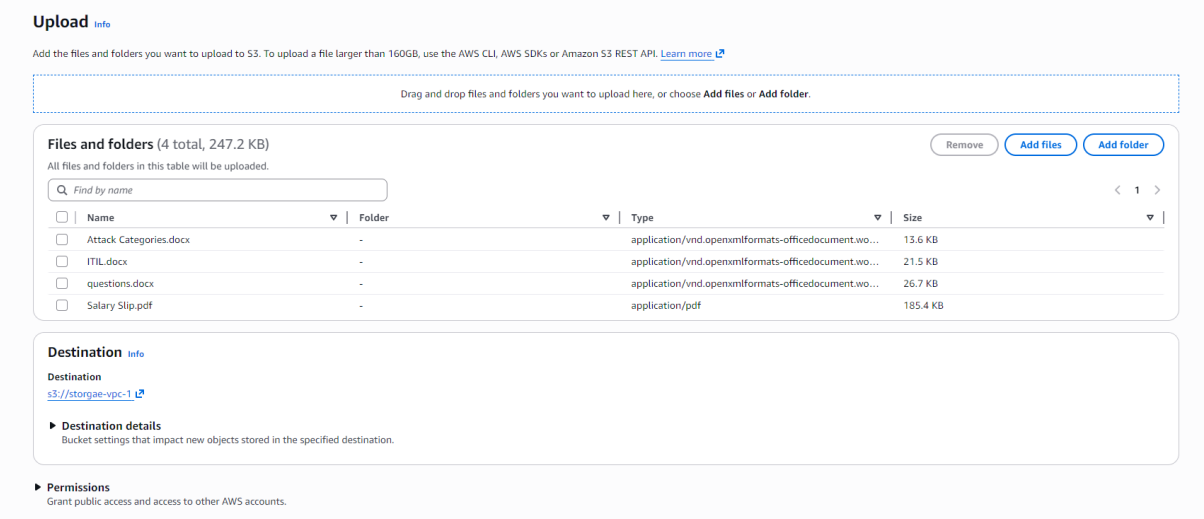
- This is recommended by AWS because access should be managed using **IAM policies** instead of ACLs.
- Disabling ACLs improves security and simplifies permission management.



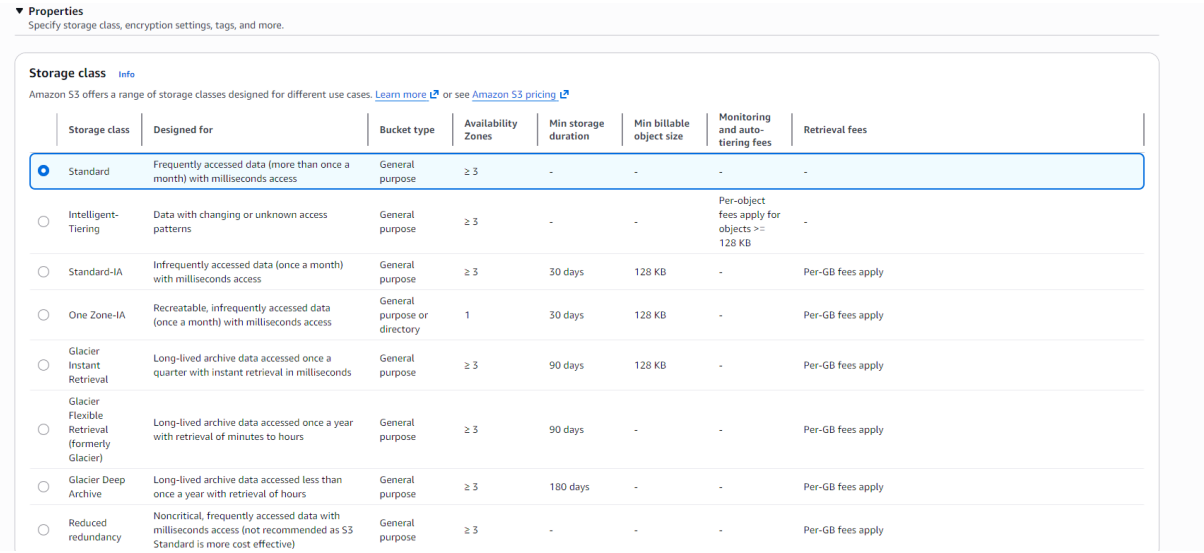
- I left all other settings on **default**.
- Finally, I clicked **Create bucket**, and the S3 bucket was created successfully.



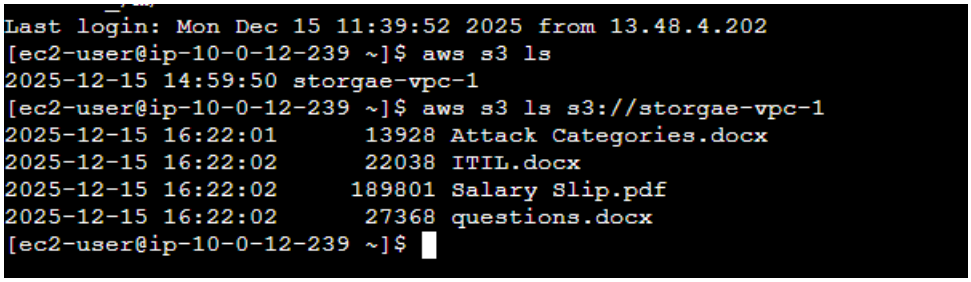
- After creating the S3 bucket, it was time to upload files.
- I opened the bucket **storage_vpc_1**.
- I clicked **Upload** and it opened a new window.



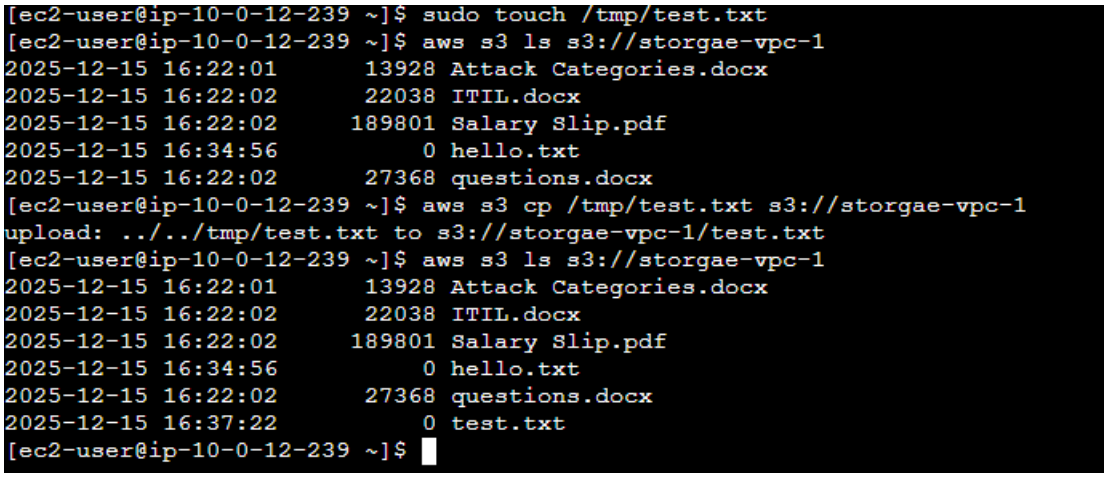
- In the upload window, I clicked **Add files** and selected some random files from my PC.
- I also had the option to **create a folder** inside the bucket.
- I could choose the **destination location** for the files and **set permissions** if needed.



- After scrolling down, I reached the **Storage class** section.
- I selected **Standard**, which is used for frequently accessed data.
- Other storage class options available were:
- **Intelligent-Tiering**
- **Standard-IA (Infrequent Access)**
- **One Zone-IA**
- **Glacier Instant Retrieval**
- **Glacier Flexible Retrieval**
- **Glacier Deep Archive**
- After selecting the storage class, I clicked **Upload**.
- The files were uploaded **successfully** to the S3 bucket.



- From inside the EC2 instance in the VPC, after configuring the AWS credentials, I accessed the S3 bucket.
- I ran the command:
  `aws s3 ls s3://storage_vpc_1`
- This command listed **all files and folders** stored inside the S3 bucket.
- The successful output confirmed that the EC2 instance inside the VPC could **access Amazon S3** correctly.



- I also created a file directly from the EC2 instance using the CLI.
- I ran the command:
  `sudo touch /tmp/test.txt`
- This command created a file named **test.txt** inside the `/tmp` directory.
- After creating **test.txt**, I copied it to the S3 bucket.
- I ran the command:
  `aws s3 cp /tmp/test.txt s3://storage_vpc_1`
- After copying the file, I checked the bucket contents again using:
  `aws s3 ls s3://storage_vpc_1`
- The **test.txt** file was listed in the output, confirming that the file was successfully uploaded from the EC2 instance to the S3 bucket.

Conclusion

In this project, I successfully configured a VPC and EC2 instance to securely access **Amazon S3**. I learned how AWS credentials work, how access keys differ from key pairs, and how IAM users control permissions for AWS services.

By creating an S3 bucket, uploading files through the AWS Console, and interacting with the bucket directly from an EC2 instance using the AWS CLI, I verified that resources inside a VPC can communicate with AWS services when proper identity and access configurations are in place.

This project strengthened my understanding of **IAM, S3, and AWS networking**, and showed how permissions and credentials enable secure access to cloud services without exposing unnecessary public access.