

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:

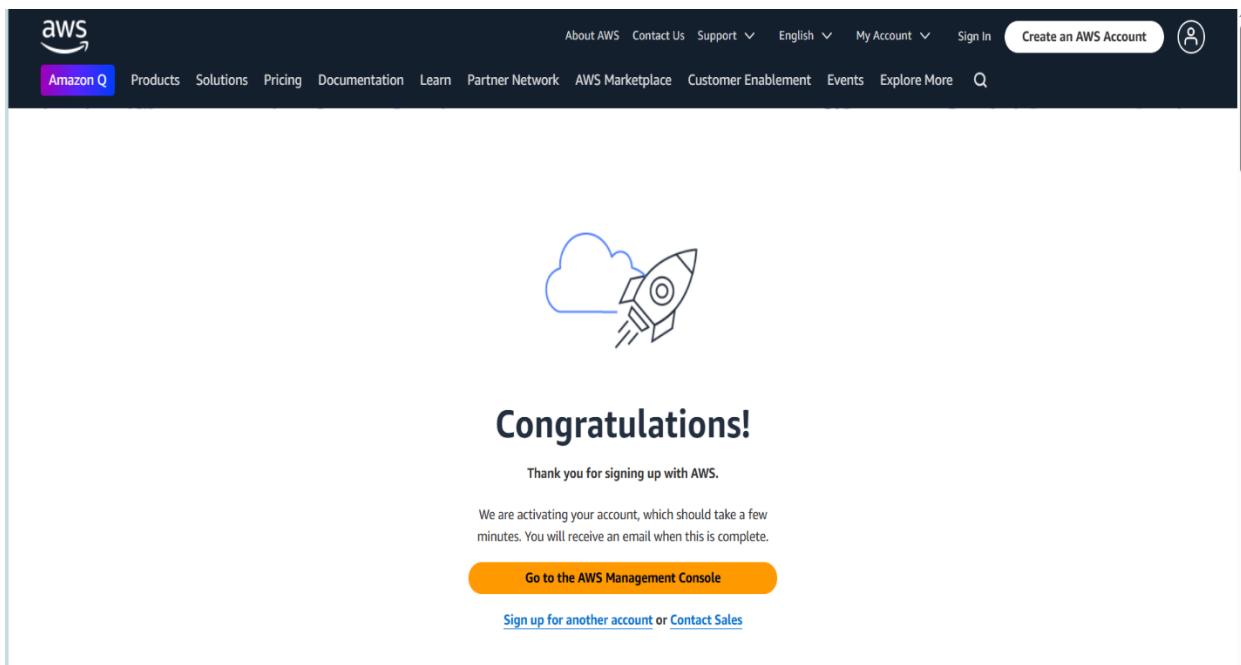
Cloud computing has transformed the way modern applications are developed, deployed, and managed by offering scalable, cost-effective and reliable infrastructure on demand.

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.

The image contains three screenshots of the AWS VPC service interface. The top-left screenshot shows the 'Create VPC' wizard step 1, 'VPC settings', where the 'VPC only' option is selected. The top-right screenshot shows step 2, 'IPv4 CIDR', with the CIDR '10.0.0.0/16' entered. The bottom screenshot shows the 'VPC dashboard' for the newly created VPC 'vpc-05b9d55c9ffda0887'. It displays details like VPC ID, State (Available), and Block Public Access (Off). The 'Resource map' tab is selected, showing a map of the VPC structure.

- ✓ 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.

The image consists of three vertically stacked screenshots from the AWS VPC console. The top two screenshots show the 'Create subnet' wizard, while the bottom one shows the 'Subnets' dashboard.

Screenshot 1: Create subnet - Step 1
Shows the 'VPC' section with 'VPC ID' set to 'vpc-05b9d5c9ffda0887 (project)'. The 'Associated VPC CIDRs' section shows 'IPv4 CIDRs' as '10.0.0.0/16'. The 'Subnet settings' section shows 'Subnet 1 of 1' with 'Subnet name' set to 'project-subnet'.

Screenshot 2: Create subnet - Step 2
Shows the 'Availability Zone' set to 'United States (N. Virginia) / us-east-1a'. The 'IPv4 VPC CIDR block' is set to '10.0.0.0/16'. The 'IPv4 subnet CIDR block' is set to '10.0.1.0/24'.

Screenshot 3: Subnets dashboard
Shows a green success message: 'You have successfully created 1 subnet: subnet-06d069bcbaa20eca5'. The 'Subnets (1) info' table lists the subnet: Name: project-subnet, Subnet ID: subnet-06d069bcbaa20eca5, State: Available, VPC: vpc-05b9d5c9ffda0887 [project], Block Public: Off, IPv4 CIDR: 10.0.1.0/24. A search bar and 'Actions' dropdown are also visible.

- ✓ 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".

Screenshot of the AWS VPC Subnets settings page for subnet-06d069bcbaa20eca5. The 'Edit subnet settings' section is open, showing configuration for auto-assign IP settings and resource-based name (RBN) settings.

Subnet

- Subnet ID: subnet-06d069bcbaa20eca5
- Name: project-subnet

Auto-assign IP settings (Info)

Enable AWS to automatically assign a public IPv4 or IPv6 address to a new primary network interface for an instance in this subnet.

- Enable auto-assign public IPv4 address (Info)
- Enable auto-assign customer-owned IPv4 address (Info)
Option disabled because no customer owned pools found.

Resource-based name (RBN) settings (Info)

Specify the hostname type for EC2 instances in this subnet and optional RBN DNS query settings.

- Enable resource name DNS A record on launch (Info)
- Enable resource name DNS AAAA record on launch (Info)

Hostname type (Info)

- Resource name
- IP name

- ✓ **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.

The screenshot shows two consecutive screenshots of the AWS VPC console. The first screenshot is titled 'Create internet gateway' and shows the configuration for a new Internet Gateway. It includes fields for 'Name tag' (set to 'project-igw') and 'Tags - optional' (one tag named 'Name' with value 'project-igw'). The second screenshot shows the 'Internet gateways' list after the gateway has been created, with the ID 'igw-072bf08905a703c11' and name 'project-igw'. A green message bar indicates the gateway was created successfully and provides a link to 'Attach to a VPC'.

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".

The screenshot shows the AWS VPC Internet Gateways console. At the top, a green banner indicates that an internet gateway has been created and can now be attached to a VPC. Below this, the 'Attach to VPC' page for the internet gateway 'igw-072bf08905a703c11' is displayed. A search bar shows the VPC ID 'vpc-05b9d55c9ffda0887'. A button labeled 'Attach to a VPC' is visible. The main area shows the 'Available VPCs' section with a dropdown menu. The 'AWS Command Line Interface command' section contains a command to attach the gateway. Below this, the 'VPC dashboard' shows the attached internet gateway 'igw-072bf08905a703c11 / project-igw'. The 'Details' tab shows the gateway's ID, state (Attached), VPC ID, and owner. The 'Tags' tab shows a single tag 'Name: project-igw'. A success message at the top right of the dashboard indicates the gateway was successfully attached.

- ✓ 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.

The screenshot shows two side-by-side AWS VPC Route Tables interfaces. The left interface is the 'Create route table' wizard, and the right is the 'Route table details' page.

Create route table (Left):

- Route table settings:**
 - Name: project-rt
 - VPC: vpc-0358d55c9ffab0007 [project]
- Tags:** A tag named 'Name' with value 'project-rt' is selected.
- Buttons:** 'Cancel' and 'Create' (highlighted in orange).

Route table details (Right):

- Details:** Route table ID: rtb-095eac8b3781a1b0c, Main, Owner ID: 35590493015.
- Routes:** One route entry: Destination: 10.0.0.1/16, Target: local, Status: Active, Propagated: No.

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

The screenshot shows two screenshots of the AWS VPC Route Tables interface. The top screenshot is titled 'Edit routes' and shows a table with two rows. The first row has a 'Destination' of '10.0.0.0/16' and a 'Target' of 'local'. The second row has a 'Destination' of '0.0.0.0/0' and a 'Target' of 'Internet Gateway'. Both rows have a status of 'Active' and are 'Propagated' to 'No'. There is a 'Remove' button next to the second row. The bottom screenshot shows a success message: 'Updated routes for rtb-095eac8b3781a1b0c / project-rt successfully' followed by a 'Details' link. Below this, the 'Details' tab is selected for the route table 'rtb-095eac8b3781a1b0c / project-rt'. It shows the 'Route table ID' as 'rtb-095eac8b3781a1b0c', 'Main' as 'No', 'Owner ID' as '265980493076', and 'VPC' as 'vpc-05b9d55c9ffda0887 | project'. The 'Routes' tab is selected, showing two routes: one for '0.0.0.0/0' targetting 'igw-072bf08905a703c11' and another for '10.0.0.0/16' targetting 'local', both in an 'Active' state and not propagated.

- ✓ 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".

The screenshot shows two consecutive screenshots of the AWS VPC Route Tables interface.

Screenshot 1: Edit subnet associations

- Available subnets (1/1):**

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
project-subnet	subnet-06d069bcbaa20eca5	10.0.1.0/24	-	Main (rtb-0671af822180d3832)
- Selected subnets:** A single subnet is selected: `subnet-06d069bcbaa20eca5 / project-subnet`.
- Buttons:** Cancel and Save associations.

Screenshot 2: Subnet associations updated

- Details:** Shows the successful update message: "You have successfully updated subnet associations for rtb-095eac8b3781a1b0c / project-rt." and the updated route table settings.
- Explicit subnet associations (1):**

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
project-subnet	subnet-06d069bcbaa20eca5	10.0.1.0/24	-
- Subnets without explicit associations (0):** No subnets are listed.
- Buttons:** Edit subnet associations.

- ✓ 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.

The image displays three sequential screenshots of the AWS S3 'Create bucket' wizard:

- General configuration:** Shows the 'Bucket name' field set to 'sya7-s3-bucket'. Other fields include 'AWS Region' (us-east-1), 'Bucket type' (General purpose), and 'Object ownership' (Bucket owner enforced). A note states: "Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or digit." A 'Choose bucket' button is present.
- Object Ownership:** Shows the 'Object Ownership' section with 'ACLs disabled (recommended)' selected. A note says: "All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies." An alternative 'ACLs enabled' option is shown.
- Block Public Access settings for this bucket:** Shows the 'Block public access' section with all four checkboxes checked:
 - Block public access to buckets and objects granted through IAM roles and policies
 - Block public access to buckets and objects granted through VPC endpoints
 - Block public access to buckets and objects granted through AWS Lambda functions
 - Block public cross-account access to buckets and objectsA note states: "Turning the setting on is the same as turning on all four settings below. Each setting applies to the same objects as the others." A 'CloudShell' button is at the bottom.

- ✓ 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.

The screenshot shows two consecutive screenshots of the AWS S3 console.

Screenshot 1: Create Bucket Wizard

- Header: AWS, Search, United States (N. Virginia), Farah Walid.
- Breadcrumbs: Amazon S3 > Buckets > Create bucket.
- Buttons: Add tag.
- Section: Default encryption (Info)
 - Server-side encryption is automatically applied to new objects stored in this bucket.
 - Encryption type: Server-side encryption with Amazon S3 managed keys (SSE-S3)
 - Server-side encryption with AWS Key Management Service keys (SSE-KMS)
 - Dual-layer server-side encryption with AWS Key Management Service keys (SSE-KMS)
 - Secure your objects with two separate layers of encryption. For details on pricing, see SSE-KMS pricing on the Storage tab of the [Amazon S3 pricing page](#).
- Section: Bucket Key
 - Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for SSE-KMS. [Learn more](#).
 - Disable
 - Enable
- Section: Advanced settings
- Note: After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Screenshot 2: Bucket Listing

- Header: AWS, Search, United States (N. Virginia), Farah Walid.
- Breadcrumbs: Amazon S3 > Buckets.
- Message: Successfully created bucket "syat-s3-bucket". To upload files and folders, or to configure additional bucket settings, choose View details.
- Section: Account snapshot - updated every 24 hours (All AWS Regions)
 - Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets. [Learn more](#).
 - [View Storage Lens dashboard](#)
- Tab: General purpose buckets (selected) | Directory buckets
- Table: General purpose buckets (1)
 | Name | AWS Region | IAM Access Analyzer | Creation date |
| --- | --- | --- | --- |
| syat-s3-bucket | US East (N. Virginia) us-east-1 | [View analyzer for us-east-1](#) | April 20, 2025, 17:07:37 (UTC+02:00) |
- Actions: Copy ARN, Empty, Delete, Create bucket.

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

Features of Amazon S3:

- **Durability:** 99.99999999% (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.

The screenshot shows the AWS IAM Policy Editor interface. On the left, there is a code editor window titled "Policy editor" containing a JSON policy document. The policy defines two statements: one allowing list bucket operations on a specific bucket, and another allowing object actions like put and get on all objects in the same bucket. On the right, there is a "Review and create" step-by-step wizard. The first step, "Specify permissions", is completed. The second step, "Review and create", is currently selected. It includes fields for "Policy name" (set to "project-policy-aws"), "Description" (set to "Allows EC2 instance to list bucket and upload/download files for the file sharing app."), and a "Permissions defined in this policy" section. The permissions listed are "Allow (1 of 439 services)" with "Service" set to "Allow".

- ✓ 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 instance to list bucket and upload/download files for the file sharing app."
- ✓ Click "Create policy".
- ✓ Then go now to create the IAM Role.

The screenshot shows the AWS IAM Policies page. A green success message at the top states "Policy project-policy-aws created." Below it, the "Policies (1346)" section is visible, with a sub-header "A policy is an object in AWS that defines permissions." There are buttons for "View policy", "Actions", "Delete", and "Create policy". The navigation bar on the left shows "Identity and Access Management (IAM)".

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.

The image consists of three vertically stacked screenshots of the AWS IAM 'Create role' wizard, showing the progression through three steps: Step 1 (Select trusted entity), Step 2 (Add permissions), and Step 3 (Name, review, and create).

Step 1: Select trusted entity

This step shows the 'Select trusted entity' screen. The 'Trusted entity type' section is expanded, showing four options: 'AWS service' (selected), 'AWS account', 'Web identity', and 'SAML 2.0 federation'. Below this, the 'Use case' section shows 'Service or use case' set to 'EC2'. A note says 'Choose a use case for the specified service.' Under 'Use case', it lists 'EC2' with the description 'Allows EC2 instances to call AWS services on your behalf.'

Step 2: Add permissions

This step shows the 'Add permissions' screen. It displays a list of 'Permissions policies (1/1046)' with one item selected: 'project-policy-aws'. The 'Filter by Type' dropdown is set to 'Policy name' with 'project' entered. Other policies listed include 'SageMakerStudioProjectProvisioningRolePolicy', 'SageMakerStudioProjectRoleMachineLearn...', 'SageMakerStudioProjectUserPermissions...', and 'SageMakerStudioProjectUserRolePolicy'. A note says 'Choose one or more policies to attach to your new role.'

Step 3: Name, review, and create

This step shows the 'Name, review, and create' screen. It includes 'Role details' (Role name: 'project-role-aws', Description: 'Allows EC2 instances to call AWS services on your behalf.'), 'Step 1: Select trusted entities' (Trust policy showing JSON code), and 'Permissions boundary - optional' (with 'Cancel', 'Previous', and 'Next' buttons). A note at the bottom says 'Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+-.').

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

Trust policy

```

1+ [ {
2 "Version": "2012-10-17",
3 "Statement": [
4 {
5 "Effect": "Allow",
6 "Action": [
7 "sts:AssumeRole"
8 ],
9 "Principal": {
10 "Service": [
11 "ec2.amazonaws.com"
12 ]
13 }
14 }
15 ]
16 ]

```

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
project-policy-aws	Customer managed	Permissions policy

Role project-role-aws created.

Roles (4)

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
amazon_ec2_s3	AWS Service: ec2	-
AWSRoleForSupport	AWS Service: support (Service-Linker)	-
AWSRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service)	-
project-role-aws	AWS Service: ec2	-

- ✓ 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.

The screenshot shows the AWS EC2 'Launch an instance' wizard. Step 1: 'Amazon Machine Image (AMI)' selection. Step 2: 'Instance type' selection (t2.micro). Step 3: 'Key pair (login)' creation ('project-key'). Step 4: 'Network settings' configuration (VPC and subnet).

- ✓ 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 eligible".
- ✓ 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.

The screenshots illustrate the AWS EC2 'Launch an instance' process. The first step shows basic storage configuration with an 8 GB gp2 root volume. The second step shows advanced details, including selecting an IAM instance profile ('project-role-aws') and specifying a DNS hostname. The third step shows the user data section, which contains a Python script for file sharing between the instance and an S3 bucket.

- ✓ **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".



The screenshot shows the AWS EC2 Instances "Launch an instance" page. At the top, there's a blue progress bar with the text "Launching instance" and "Creating security group rules". Below it, a message says "Please wait while we launch your instance. Do not close your browser while this is loading." A green success message at the bottom states "Successfully initiated launch of instance i-054e2799f0d3ea53".

✓ Click "Launch Instance".

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, and Elastic Block Store. The main area shows a table titled "Instances (1/1) info" with one row for "project-ec2-aws" (Instance ID: i-054e2799f0d3ea53, State: Running, Type: t2.micro). Below the table is a detailed view for the instance "i-054e2799f0d3ea53 (project-ec2-aws)".

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

The Web Application Development

```
[ec2-user@ip-10-0-1-210 ~]
[ec2-user@ip-10-0-1-210 ~]$ chmod 400 C:/users/dell/Downloads/project-key.pem
[ec2-user@ip-10-0-1-210 ~]$ ssh -i C:/users/dell/Downloads/project-key.pem ec2-user@34.228.158.194
Amazon Linux
  #_
  ~\##_
  ~\##_ Amazon Linux 2
  ~\##_ AL2 End of Life is 2026-06-30.
  ~\##_ A newer version of Amazon Linux is available!
  ~\##_ Amazon Linux 2023, GA and supported until 2028-03-15.
  ~\##_ https://aws.amazon.com/linux/amazon-linux-2023/
[ec2-user@ip-10-0-1-210 ~]$ sudo yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
 0 packages marked as update
[ec2-user@ip-10-0-1-210 ~]$ sudo yum install python3 python3-pip git -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Package python3-3.7.16-1.amzn2.0.17.x86_64 already installed and latest version
Package python3-pip-20.2.2-1.amzn2.0.10.noarch already installed and latest version
Package python3-setuptools-57.1-1.amzn2.0.2.x86_64 already installed and latest version
Nothing to do
[ec2-user@ip-10-0-1-210 ~]$ pip install flask boto3 --user
[bash]: pip: command not found
[ec2-user@ip-10-0-1-210 ~]$ pip3 install Flask boto3 --user
Requirement already satisfied: Flask in /usr/local/lib/python3.7/site-packages (2.2.3)
Requirement already satisfied: werkzeug<2.2.1 in /usr/local/lib/python3.7/site-packages (2.2.3)
Requirement already satisfied: importlib-metadata<3.6.0; python_version < '3.10' in /usr/local/lib/python3.7/site-packages (from flask) (6.7.0)
Requirement already satisfied: Jinja2>=3.0.0 in /usr/local/lib/python3.7/site-packages (from flask) (3.1.6)
Requirement already satisfied: click<8.0.0 in /usr/local/lib/python3.7/site-packages (from flask) (8.1.8)
Requirement already satisfied: itsdangerous<2.1.0; python_version < '3.10' in /usr/local/lib/python3.7/site-packages (from flask) (2.1.2)
Requirement already satisfied: importlib-resources<2.0.0; python_version < '3.10' in /usr/local/lib/python3.7/site-packages (from boto3) (1.0.1)
Requirement already satisfied: six!=1.5 in /usr/local/lib/python3.7/site-packages (from boto3) (0.8.2)
Requirement already satisfied: botocore<1.30.0,>=1.33.13 in /usr/local/lib/python3.7/site-packages (from boto3) (1.33.13)
Requirement already satisfied: MarkupSafe==2.1.1 in /usr/local/lib64/python3.7/site-packages (from werkzeug<2.2.2->flask) (2.1.5)
Requirement already satisfied: typing_extensions<3.1.0; python_version < '3.8' in /usr/local/lib/python3.7/site-packages (from importlib-metadata<3.6.0; python_version < '3.10'>flask) (3.15.0)
Requirement already satisfied: urllib3<1.27.1,>=1.25.4; python_version < "3.10" in /usr/local/lib/python3.7/site-packages (from importlib-resources<2.0.0; python_version < '3.10'>flask) (1.26.20)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.7/site-packages (from botocore<1.34.0,>=1.33.13>boto3) (2.9.0.post0)
Requirement already satisfied: six!=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.34.0,>=1.33.13>boto3) (1.17.0)
[ec2-user@ip-10-0-1-210 ~]$
```

This code illustrates the connection of our EC2 via SSH.

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

```
[ec2-user@ip-10-0-1-210 ~]$ nano app.py
[ec2-user@ip-10-0-1-210 ~]$ app.py
----- Start of app.py -----
import os
import boto3
from botocore.exceptions import ClientError
from flask import Flask, request, render_template, redirect, url_for, flash
# --- Configuration ---
# AWS values required at requested ***
S3_BUCKET = "your-s3-bucket"
AWS_REGION = "us-east-1" # Using the region identifier
# -----#
# If not S3_BUCKET:
#     print("ERROR: S3_BUCKET variable is empty!")
#     # Handle error appropriately in a real app
# -----#
# Flask App Initialization ---
app = Flask(__name__)
# Replace with a persistent secret key in a production environment, maybe from env vars
app.secret_key = os.urandom(24)
# -----#
# AWS S3 Client Initialization ---
# If no credentials are present in the EC2 instance,
# boto3 will automatically use its credentials.
# If no IAM role is specified, try to enable IAM role is specified correctly.
s3_client = None
try:
    print(f"Initializing S3 client for region {AWS_REGION} and bucket {S3_BUCKET}...")
    s3_client = boto3.client(s3, region_name=AWS_REGION)
    # If no IAM role is specified, try to enable IAM role is specified correctly.
    # boto3 will automatically use its credentials.
    # Try to enable IAM role is specified correctly.
    s3_client.head_bucket(Bucket=S3_BUCKET) # Checks if bucket exists and you have permission
except ClientError as e:
    if e.response['Error']['Code'] == 'NoSuchBucket':
        print(f"ERROR: S3 Bucket {S3_BUCKET} does not exist in region: {AWS_REGION}.")
    else:
        print(f"ERROR: Access Denied. Check IAM role permissions for bucket: {S3_BUCKET}. Error: {e}")
else:
    print(f"INFO: Bucket initialized S3 client or access bucket: {e}")
# Keep s3_client as None if initialization failed
# Keep s3_client as None if initialization failed
print(f"INFO: Unexpected error occurred during s3 client initialization: {e}")
# Keep s3_client as None
# -----#
def generate_presigned_url(bucket_name, object_name, expiration=3600):
    """Generate a presigned URL to share an S3 object (valid for 1 hour)."""
    if not s3_client:
        print(f"Skipping presigned url generation: s3_client={s3_client}, bucket={bucket_name}, object={object_name}")
    return None
try:
    Get Help      Write Out      Where Is      Cut Text      Justify      Cur Pos      Undo      Mark Text      To Bracket      Previous      Back Forward      Prev Word      Next Word
    Exit      Read File      Replace      Uncut Text      To Linter      Go To Line      Redo      whereIs Next      Next

```

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

```

ec2-user@ip-10-0-1-210:~/projectAWS/templates
GNU nano 2.9.8                               index.html

```

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple AWS File Share</title>
  <link href="CSS/style.css" rel="stylesheet" />
</head>
<body>
  <h1>Simple AWS S3 File Share</h1>
  <!-- Flash Messages -->
  <% with messages = get_flashed_messages(with_categories=true) %>
    <% if messages %>
      <div class="flash-messages">
        <% for category, message in messages %>
          <div class="flash {{ category }}">{{ message }}


Terminal status bar: Get Help, Write Out, Where Is, Cut Text, Justify, Cur Pos, Undo, Mark Text, To Bracket, Previous, Back, Copy Text, Where's Next, Next, Forward, Next Word, Exit, Read File, Replace, Uncut Text, Go To Line, Undo, Redo.


```

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

```

ec2-user@ip-10-0-1-210:~/projectAWS
GNU nano 2.9.8                               static/style.css

```

```

body {
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
  line-height: 1.6;
  margin: 20px auto /* Center content */;
  max-width: 800px /* Limit width */;
  background-color: #f8f9fa;
  color: #212128;
  padding: 15px;
}

h1, h2 {
  color: #0056b3 /* A shade of blue */;
  border-bottom: 2px solid #e6eaf2;
  padding-bottom: 8px;
  margin-top: 25px;
  margin-bottom: 15px;
}

h1 {
  text-align: center;
  margin-bottom: 30px;
}

.upload-section, .file-list-section {
  background-color: #f1f1f1;
  padding: 25px;
  margin-bottom: 25px;
  border-radius: 8px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.05);
  border: 1px solid #e6eaf2;
}

form {
  display: flex;
  flex-wrap: wrap /* Allow wrapping on smaller screens */;
  align-items: center;
}

form input[type="file"] {
  flex-grow: 1 /* Take up available space */;
  margin-right: 10px;
  margin-bottom: 10px /* Add space below the wrap */;
  padding: 10px;
  border: 1px solid #e6eaf2;
  border-radius: 4px;
  background-color: #f8f9fa;
}

form button {
  padding: 10px 20px;
  background-color: #ff9900 /* AWS orange */;
  color: white;
  border: none;
}

```

Terminal status bar: Get Help, Write Out, Where Is, Cut Text, Justify, Cur Pos, Undo, Mark Text, To Bracket, Previous, Back, Copy Text, Where's Next, Next, Forward, Next Word, Exit, Read File, Replace, Uncut Text, Go To Line, Undo, Redo.

```

ec2-user@ip-10-0-1-210:~/projectAWS
GNU nano 2.9.8                               static/style.css

```

```

cursor: pointer;
font-weight: bold;
transition: background-color 0.2s ease-in-out;
margin-bottom: 10px /* Watch input spacing */;

form button:hover {
  background-color: #e68a00; /* Darker orange */;
}

ul {
  list-style: none;
  padding: 0;
}

li {
  padding: 12px 15px;
  border-bottom: 1px solid #e6eaf2;
  display: flex;
  justify-content: space-between;
  align-items: center;
  flex-wrap: wrap; /* Allow wrapping for long names/links */;
}

li:last-child {
  border-bottom: none;
}

.filename {
  margin-right: 15px; /* Space between filename and link */;
  word-break: break-all; /* Break long filenames */;
  padding-bottom: 5px /* Space if wrapping */;
}

.download-link {
  text-decoration: none;
  color: #007bff /* Standard link blue */;
  font-size: 0.9em;
  white-space: nowrap;
  padding-bottom: 5px /* Space if wrapping */;
}

.download-link:hover {
  text-decoration: underline;
  color: #0056b3;
}

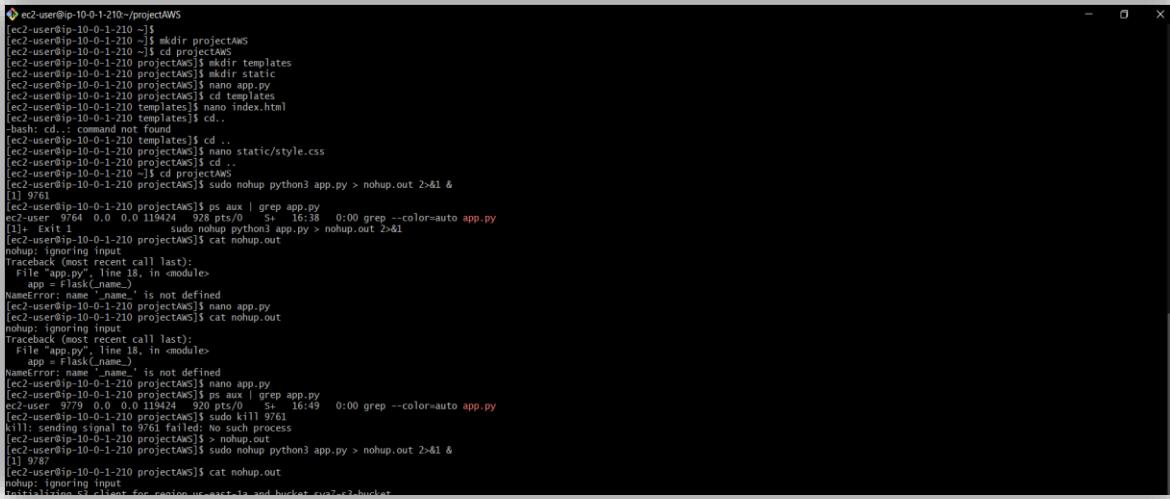
.error-link {
  color: #dc3545 /* Bootstrap danger red */;
  font-size: 0.9em;
  white-space: nowrap;
  padding-bottom: 5px /* Space if wrapping */;
}

```

Terminal status bar: Get Help, Write Out, Where Is, Cut Text, Justify, Cur Pos, Undo, Mark Text, To Bracket, Previous, Back, Copy Text, Where's Next, Next, Forward, Next Word, Exit, Read File, Replace, Uncut Text, Go To Line, Undo, Redo.

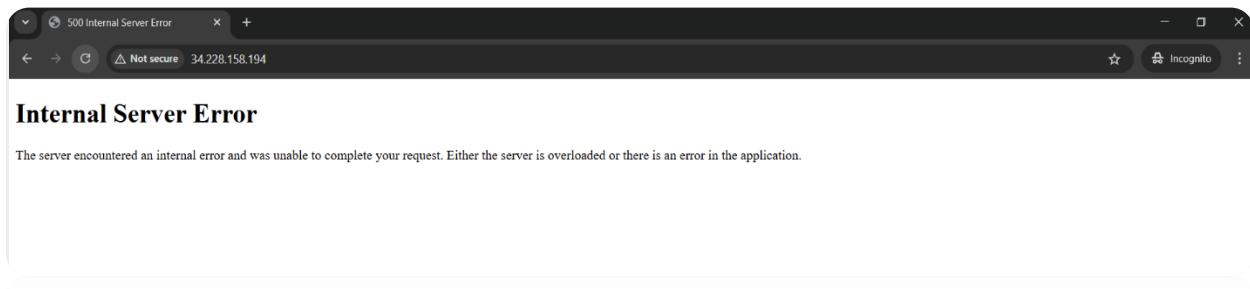
Let's start runnig our program:

```
[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]$ |
```



```
[ec2-user@ip-10-0-1-210 ~]$ cd projectAWS
[ec2-user@ip-10-0-1-210 projectAWS]$ ls
[ec2-user@ip-10-0-1-210 projectAWS]$ cd ..
[ec2-user@ip-10-0-1-210 ~]$ ls
[ec2-user@ip-10-0-1-210 ~]$ cd projectAWS
[ec2-user@ip-10-0-1-210 projectAWS]$ ls
[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]$ ps aux | grep app.py
[ec2-user@ip-10-0-1-210 projectAWS]$ 9764 0.0 0.0 119424 928 pts/0 S+ 16:38 0:00 grep --color=auto app.py
[1]+ Exit 1 sudo nohup python3 app.py > nohup.out 2>&1
[ec2-user@ip-10-0-1-210 projectAWS]$ cat nohup.out
nohup: ignoring input
nohup: ignoring input
Traceback (most recent call last):
  File "app.py", line 18, in module
    app = Flask(__name__)
NameError: name '__name__' is not defined
[ec2-user@ip-10-0-1-210 projectAWS]$ nano app.py
[ec2-user@ip-10-0-1-210 projectAWS]$ ps aux | grep app.py
[ec2-user@ip-10-0-1-210 projectAWS]$ 9764 0.0 0.0 119424 928 pts/0 S+ 16:49 0:00 grep --color=auto app.py
[ec2-user@ip-10-0-1-210 projectAWS]$ kill 9761
kill: sending signal to 9761: failed: No such process
[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]$ cat nohup.out
initialization S3 client for region us-east-1 and bucket suez-cs-bucket
```

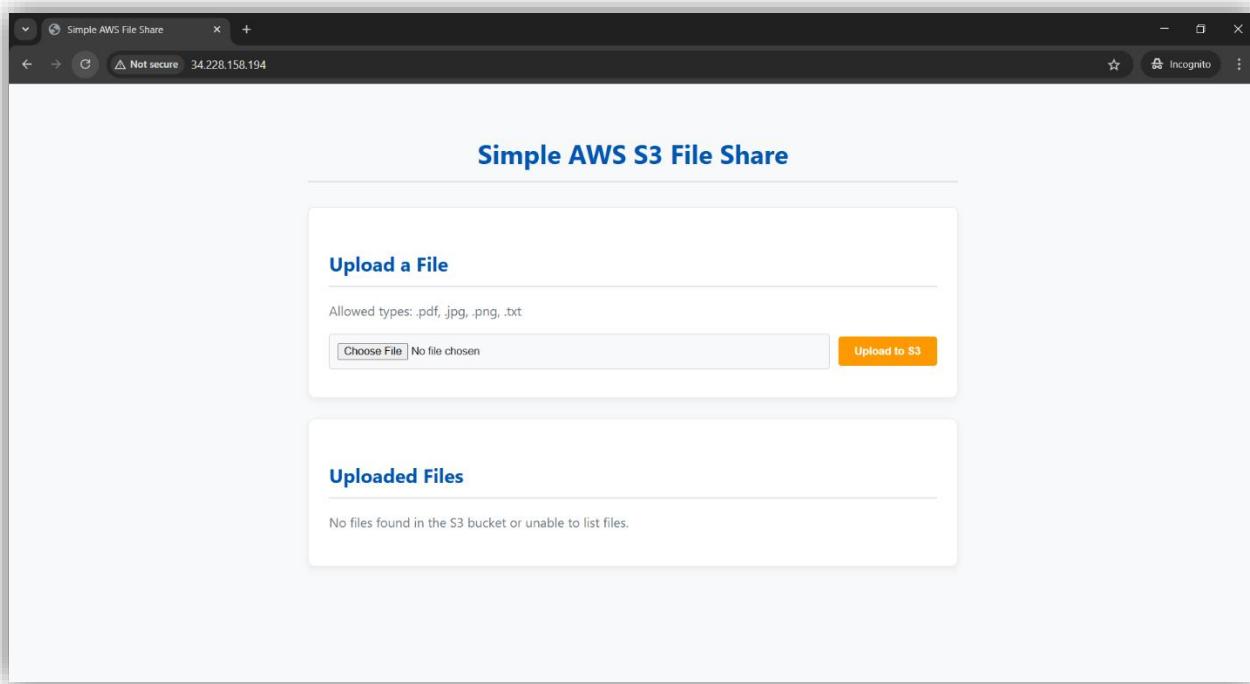
The runnig commands are all written as shown and then we got the IPv4 address of our “project-aws-ec2” instance to the a new tab in the browser, but... there was a problem due to some syntax errors in the flask code.



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

```
[ec2-user@ip-10-0-1-210 projectAWS]$ nohup.out
[ec2-user@ip-10-0-1-210 projectAWS]$ sudo nohup python3 app.py > nohup.out 2>&1 &
[1] 9787
[ec2-user@ip-10-0-1-210 projectAWS]$ cat nohup.out
nohup: ignoring input
Initializing S3 client for region us-east-1a and bucket sya7-s3-bucket...
ERROR: An unexpected error occurred during S3 client initialization: Could not connect to the endpoint URL: "https://sya7-s3-bucket.s3.us-east-1a.amazonaws.com/"
Starting Flask app on 0.0.0.0:80
Configured S3 Bucket: sya7-s3-bucket
Configured AWS Region: us-east-1a
  * Serving Flask app 'app'
  * Debug mode: off
/usr/local/lib/python3.7/site-packages/boto3/compat.py:82: PythonDeprecationWarning: Boto3 will no longer support Python 3.7 starting December 13, 2023. To continue receiving service updates, bug fixes, and security updates please upgrade to Python 3.8 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)
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://127.0.0.1:80
  * Running on http://10.0.1.210:80
Press CTRL+C to quit
[ec2-user@ip-10-0-1-210 projectAWS]$
```

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:

Simple AWS S3 File Share

File "Project Simplified File-Sharing Web Application on AWSdocument (2).pdf" uploaded successfully!

Upload a File

Allowed types: .pdf, .jpg, .png, .txt

Choose File No file chosen Upload to S3

Uploaded Files

Project Simplified File-Sharing Web Application on AWSdocument (2).pdf Download (Link expires soon)

aws Amazon S3 Buckets sya7-s3-bucket

Amazon S3

General purpose buckets

Directory buckets

Table buckets

Access Grants

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

Storage Lens groups

AWS Organizations settings

Feature spotlight

CloudShell Feedback

Objects (1) Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Name	Type	Last modified	Size	Storage class
Project Simplified File-Sharing Web Application on AWSdocument (2).pdf	pdf	April 20, 2025, 19:15:13 (UTC+02:00)	105.1 KB	Standard

The file is successfully uploaded ^_^\n

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

Simple AWS S3 File Share

File type '.csv' not allowed. Allowed types: .txt, .pdf, .jpg, .png, .jpeg, .pdf

Upload a File

Allowed types: .pdf, .jpg, .png, .txt

Choose File No file chosen Upload to S3

Uploaded Files

Project Simplified File-Sharing Web Application on AWSdocument (2).pdf Download (Link expires soon)

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

Drive Link: ^-^

<https://drive.google.com/file/d/1A5fK4JQgDUH7oINnupxYirtG9nonPiI/view?usp=sharing>

Demo Link: ^-^

<http://34.228.158.194>

