

Cloud Computing Project

Team Members:

23012082	سيف الدين سامي مدكور
23011569	مهند محمد السعيد
23011571	مهند مصطفى محمد
23011258	خالد بدر المغاوري
23011495	محمد مغازي غنيم
23011386	عمر محمد عبد العال

- Creating AWS Account: مهند مصطفى محمد

- VPC and Testing:

مهند مصطفى محمد , خالد بدر المغاوري

- VPC Configuration

- VPC CIDR: 10.0.0.0/16

- Subnet CIDR: 10.0.1.0/24

- Internet Gateway attached.

- Routing table modified to route all 0.0.0.0/0 traffic via IGW.

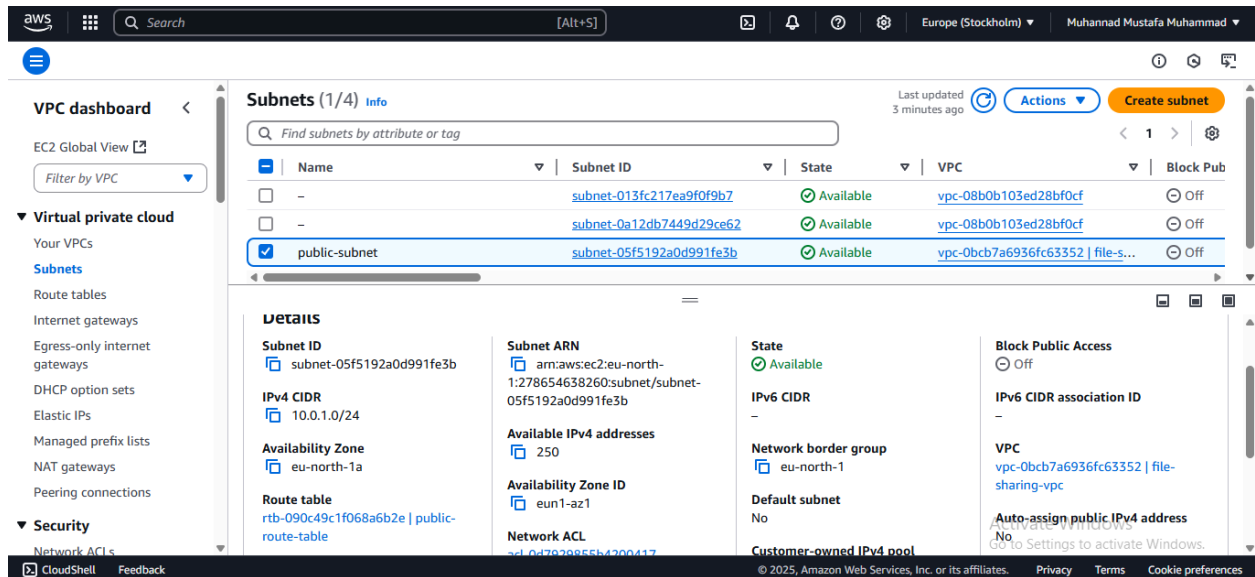
The screenshot displays the AWS Management Console interface for the 'file-sharing-vpc'. The left sidebar shows the 'VPC dashboard' with a search bar and a list of resources including Subnets, Route tables, Internet gateways, and more. The main content area shows the 'Your VPCs (1/2)' list with a table containing columns for Name, VPC ID, State, Block Public Access, and IPv4 CIDR. The 'file-sharing-vpc' is selected, and its details are shown below. The details include VPC ID, State (Available), Block Public Access (Off), DNS hostnames (Disabled), DNS resolution (Enabled), Tenancy (default), Default VPC (No), Network Address Usage metrics (Disabled), DHCP option set, IPv4 CIDR (10.0.0.0/16), Route 53 Resolver DNS Firewall rule groups, Main network ACL, and IPv6 CIDR (Network border group). The bottom of the console shows the footer with copyright information and links to Privacy, Terms, and Cookie preferences.

Name	VPC ID	State	Block Public...	IPv4 CIDR
-	vpc-08b0b103ed28bf0cf	Available	Off	172.31.0.0/16
file-sharing-vpc	vpc-0bcb7a6936fc63352	Available	Off	10.0.0.0/16

Details

VPC ID vpc-0bcb7a6936fc63352	State Available	Block Public Access Off	DNS hostnames Disabled
DNS resolution Enabled	Tenancy default	DHCP option set dopt-014c9a1a4f5ac56ff	Main route table rtb-093acb86412678d38
Main network ACL acl-0d7929855b4200417	Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool -
IPv6 CIDR (Network border group) -	Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups -	Owner ID 278654638260

Go to Settings to activate Windows.



○ Testing

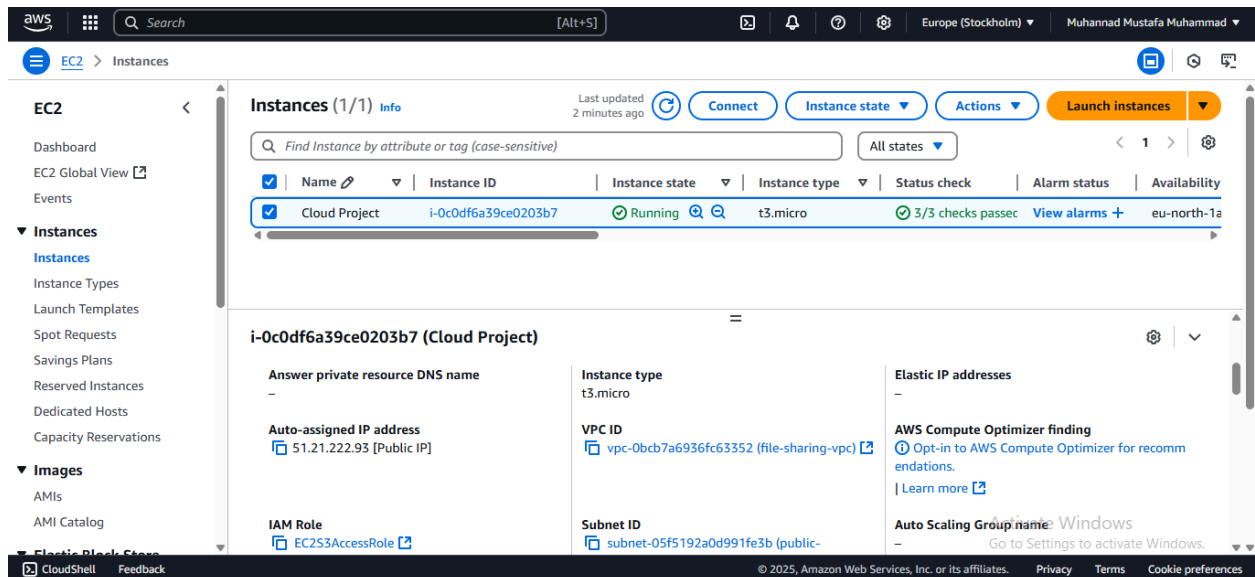
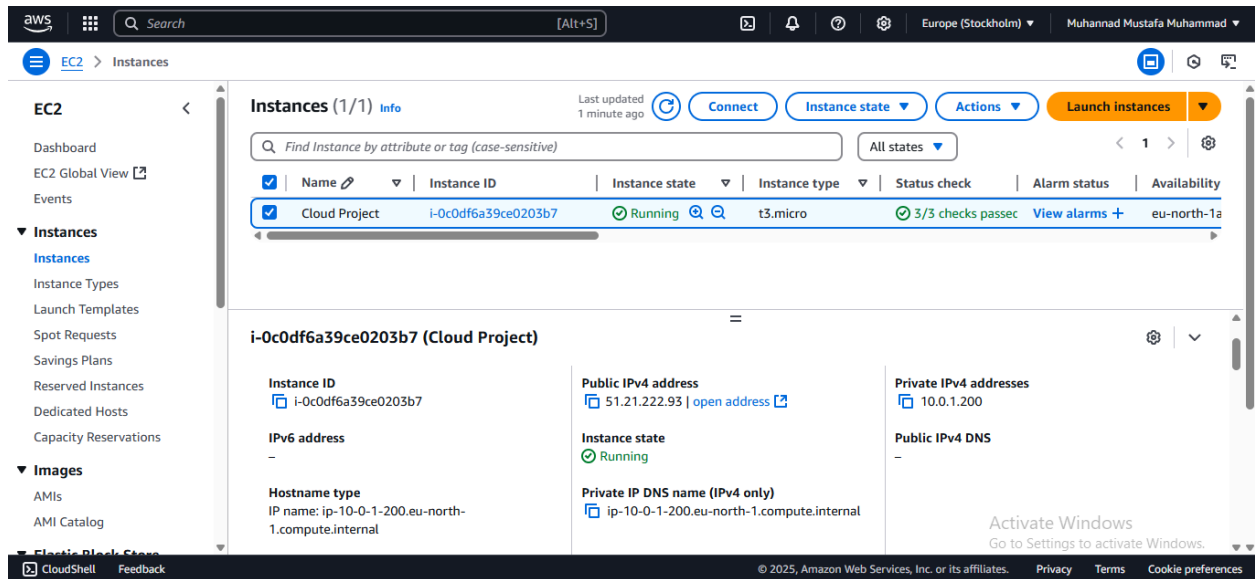
- The EC2 instance successfully connected to the S3 bucket using the IAM role.
- Files could be uploaded from the EC2 terminal to S3 via CLI, proving connectivity and correct permissions.
- Download links were generated using S3's pre-signed URL feature.

● EC2 and IAM:

محمد مغازي غنيم

○ Amazon EC2

- **A virtual server (t3.micro – Free Tier eligible) was launched.**
- **A user data script was attached to initialize server configuration and (optionally) deploy the application automatically.**
- **The EC2 instance was assigned to our subnet for internet access that we created.**
- **Security Group was configured to open HTTP (port 80) and SSH (port 22) access.**

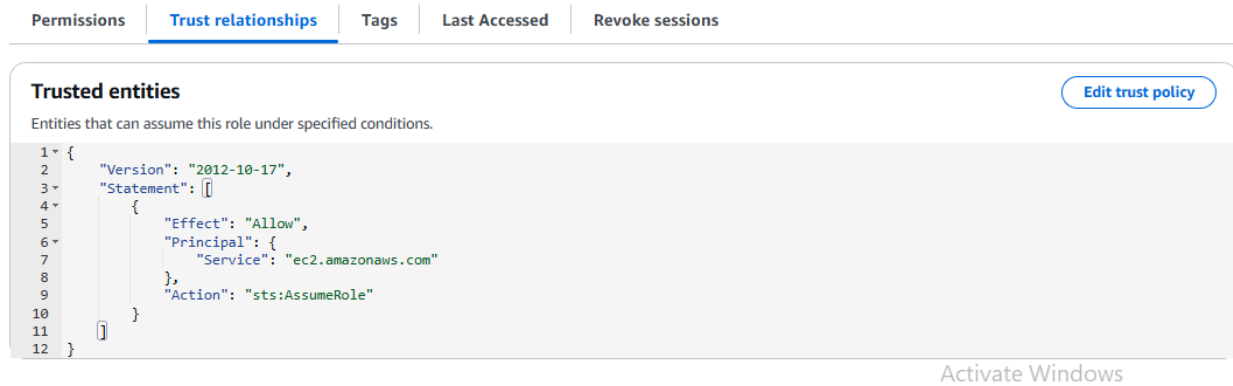
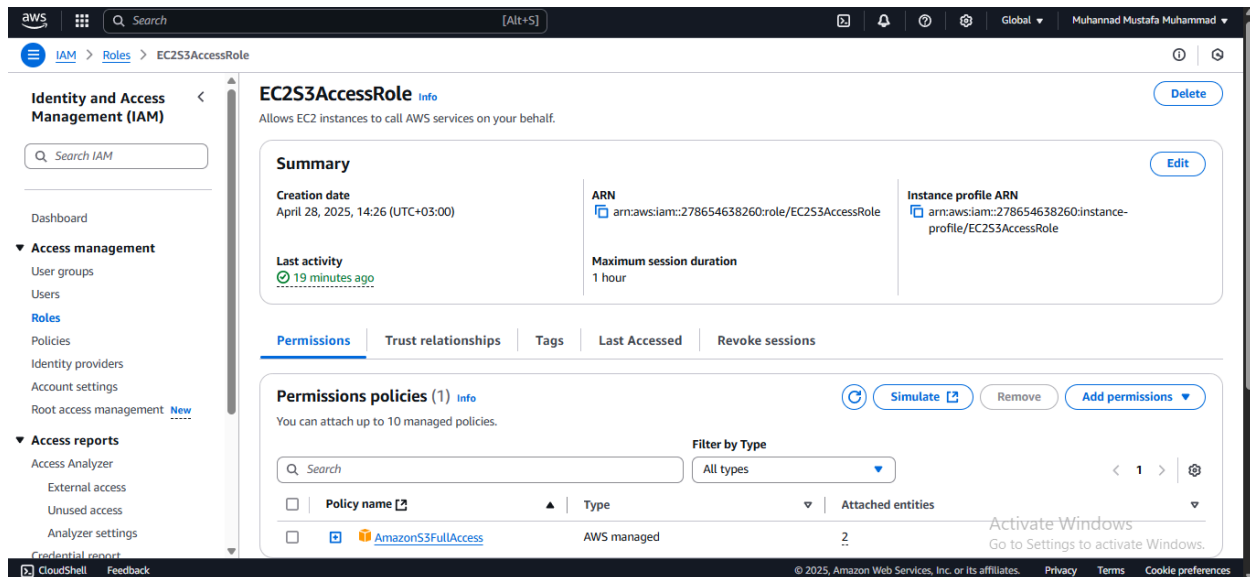


○ IAM Rule

- An IAM role was created with a policy allowing access to the S3 bucket.

■ This role was attached to the EC2 instance securely.

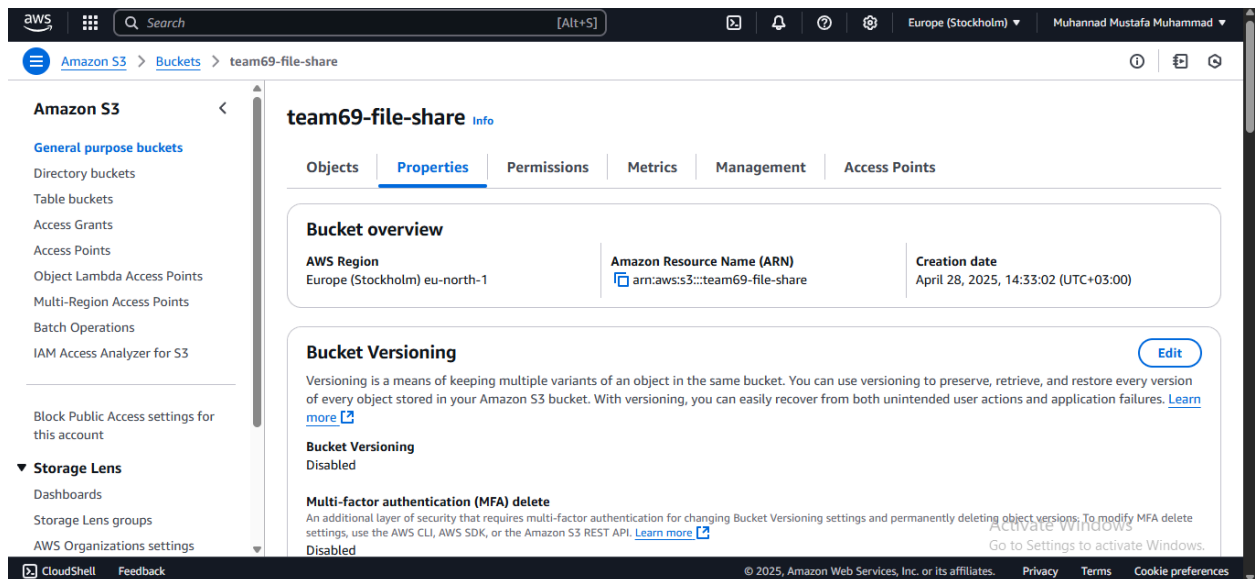
■ Principle of Least Privilege was applied – the role only allowed necessary S3 actions.



● S3:

مهند محمد السعيد , عمر محمد عبدالعال

- **S3 Bucket**
 - **Name: team69-file-share**
 - **Block Public Access: Enabled**
 - **Permissions: Set via IAM Role and Bucket Policy, Bucket policy and permissions were adjusted to allow uploads and retrieval by the application.**



Bucket policy

[Edit](#)[Delete](#)

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::team69-file-share/*"
    }
  ]
}
```

[Copy](#)

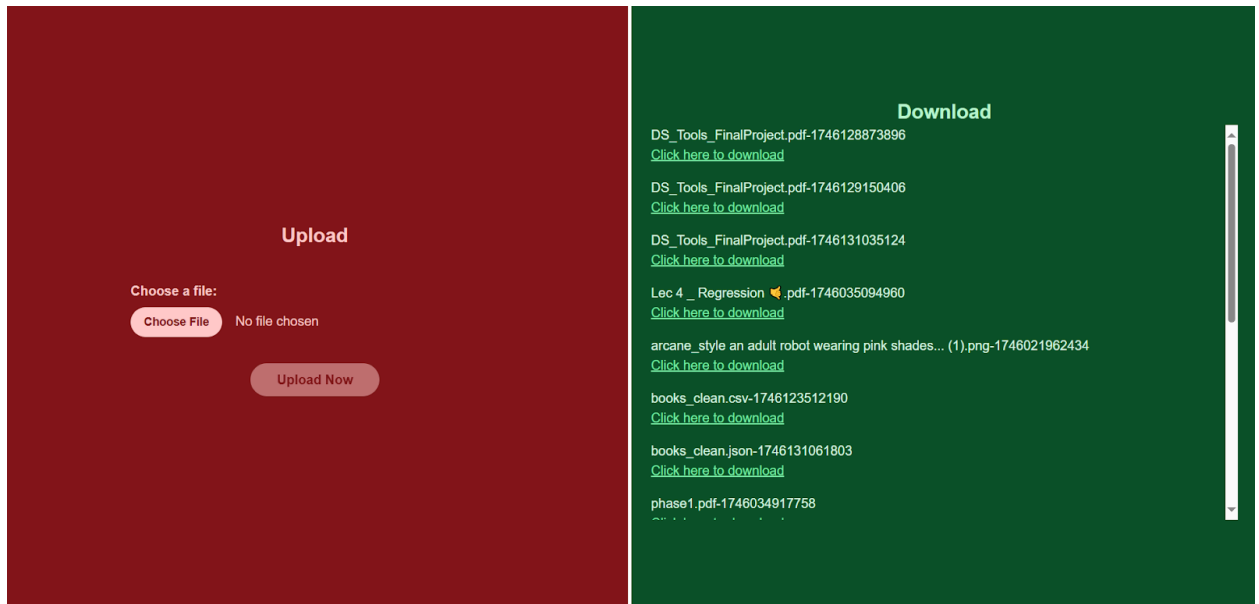
Activate Windows
Go to Settings to activate Windows.

● Web App:

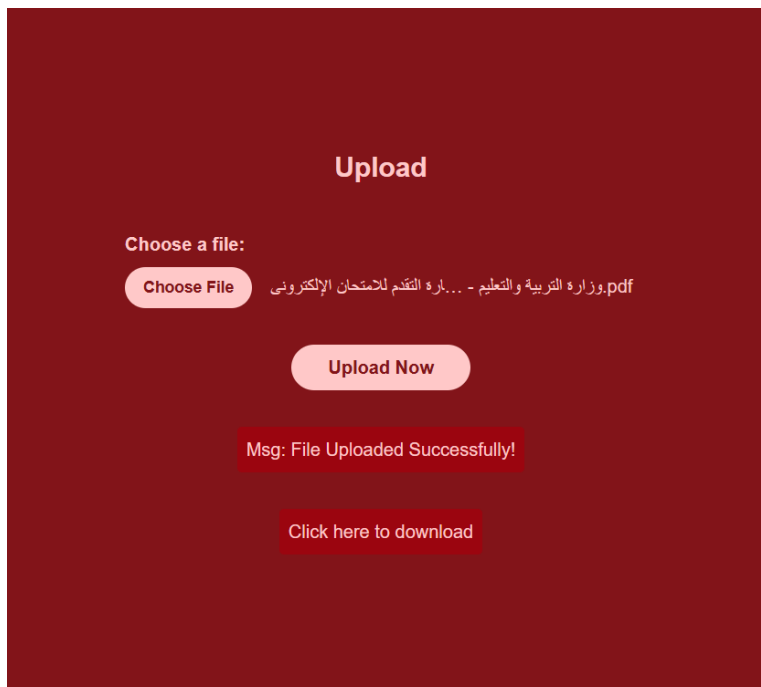
سيف الدين سامي مذكور

Using Next.JS created the web app, added UI styling using TailwindCSS, and used the API routing system in Next.JS.

Split the screen using a flexbox to two sections Upload, and Download.



The moment you choose a file then upload it, you will get a download link immediately.



Also a **list with all uploaded files** will contain the new uploaded file.

Download

[Click here to download](#)

photo_2025-03-17_17-35-28.jpg-1746035188295

[Click here to download](#)

super-mario-bros-2 (12).state-1746021790432

[Click here to download](#)

super-mario-bros-2 (25).state-1746033732791

[Click here to download](#)

super-mario-bros-2 (29).state-1746034509446

[Click here to download](#)

super-mario-bros-2 (8).state-1746033745851

[Click here to download](#)

user.png-1746022065092

[Click here to download](#)

وزارة التربية والتعليم - تسجيل استماره التقدم لامتحان الإلكتروني.pdf-1746190872693

[Click here to download](#)

A view of the API code:

```
export async function POST(req: NextRequest) {
  try {
    const formData = await req.formData();
    const file = formData.get('file');

    if (!file || !(file instanceof File)) {
      return NextResponse.json({ msg: 'Upload Error!!', err: 'Invalid or No File Found to Upload' }, { status: 400 });
    }

    const buffer = Buffer.from(await file.arrayBuffer());
    const fileUrl = await uploadFileToS3(buffer, file.name);

    return NextResponse.json({ msg: 'File Uploaded Successfully!!', fileUrl });
  } catch (e) {
    console.error('Error in POST /api/upload:', e);
    const errorMessage = e instanceof Error ? e.message : JSON.stringify(e);
    return NextResponse.json({ msg: 'Upload Error!!', err: errorMessage }, { status: 500 });
  }
}
```

```

export async function GET() {
  const bucketName = process.env.NEXT_PUBLIC_AWS_S3_BUCKET_NAME!

  try {
    const command = new ListObjectsV2Command({
      Bucket: bucketName,
    })

    const response = await s3.send(command)

    const files = response.Contents?.map((file) => ({
      name: file.Key!,
      url: `https://${bucketName}.s3.${process.env.NEXT_PUBLIC_AWS_REGION}.amazonaws.com/${file.Key}`,
    },)) || []

    // You, 2 days ago • project done for most ...
    return NextResponse.json(files)
  } catch (error) {
    console.error('Error fetching files from S3:', error)
    return NextResponse.json({ error: 'Failed to fetch files' }, { status: 500 })
  }
}

```

All Secret access keys and important info was saved using environment variables as shown in the code to save them securely.