

# **Cloud Computing Project**

## **Team Members:**

<b>23012082</b>	<b>سيف الدين سامي مذكور</b>
<b>23011569</b>	<b>مهند محمد السعيد</b>
<b>23011571</b>	<b>مهند مصطفى محمد</b>
<b>23011258</b>	<b>خالد بدر المغافوري</b>
<b>23011495</b>	<b>محمد مغازي غنيم</b>
<b>23011386</b>	<b>عمر محمد عبد العال</b>

## ● Creating AWS Account: مهند مصطفى محمد

## ● VPC and Testing:

مهند مصطفى محمد , خالد بدر المغافوري

### ○ 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.

Your VPCs (1/2) Info

Name	VPC ID	State	Block Public...	IPv4 CIDR
vpc-08b0b103ed28bf0cf	vpc-08b0b103ed28bf0cf	Available	Off	172.31.0.0/16
file-sharing-vpc	vpc-0cb7a6936fc63352	Available	Off	10.0.0.0/16

**Details**

VPC ID vpc-0cb7a6936fc63352	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

**Subnets (1/4) Info**

Name	Subnet ID	State	VPC	Block Pub
-	subnet-013fc217ea9f0f9b7	Available	vpc-08b0b103ed28bf0cf	Off
-	subnet-0a12db7449d29ce62	Available	vpc-08b0b103ed28bf0cf	Off
<b>public-subnet</b>	<b>subnet-05f5192a0d991fe3b</b>	<b>Available</b>	<b>vpc-0cbc7a6936fc63352   file-s...</b>	<b>Off</b>

**DETAILS**

Subnet ID subnet-05f5192a0d991fe3b	Subnet ARN arn:aws:ec2:eu-north-1:278654638260:subnet/subnet-05f5192a0d991fe3b	State Available	Block Public Access Off
IPv4 CIDR 10.0.1.0/24	IPv6 CIDR -	Network border group eu-north-1	IPv6 CIDR association ID -
Availability Zone eu-north-1a	Available IPv4 addresses 250	Default subnet No	VPC vpc-0cbc7a6936fc63352   file-sharing-vpc
Route table rtb-090c49c1f068a6b2e   public-route-table	Availability Zone ID eun1-az1	Customer-owned IPv4 pool Auto-assign public IPv4 address No	Go to Settings to activate Windows.
Network ACL acl-0d7020855b4200417	Network ACL -		

- **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 مغاري 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.

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, and AMI Catalog. The main content area has a title 'Instances (1/1) Info' with a note 'Last updated 1 minute ago'. It includes a search bar, 'Connect' button, 'Instance state' dropdown, 'Actions' dropdown, and a large orange 'Launch instances' button. Below this is a table with columns: Name (with a filter icon), Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability. A single row is selected for 'Cloud Project' (i-0c0df6a39ce0203b7), which is 'Running' (t3.micro). At the bottom of the table, it says '3/3 checks passed' and 'View alarms + eu-north-1a'. The instance details panel below shows fields like Instance ID (i-0c0df6a39ce0203b7), Public IPv4 address (51.21.222.93), Private IPv4 addresses (10.0.1.200), and Public IP DNS name (ip-10-0-1-200.eu-north-1.compute.internal). Other sections include Hostname type (IP name: ip-10-0-1-200.eu-north-1.compute.internal), Instance state (Running), and Private IP4 addresses (10.0.1.200). There's also a note to 'Activate Windows'.

This screenshot is identical to the one above, showing the same EC2 instance 'Cloud Project' (i-0c0df6a39ce0203b7). However, the instance details panel at the bottom is displayed differently. It now lists fields such as Answer private resource DNS name (empty), Instance type (t3.micro), Elastic IP addresses (empty), VPC ID (vpc-0bcb7a6936fc63352 (file-sharing-vpc)), AWS Compute Optimizer finding (Opt-in to AWS Compute Optimizer for recommendations), IAM Role (EC2S3AccessRole), Subnet ID (subnet-05f192a0d991fe3b (public-)), and Auto Scaling Group (Windows). The rest of the interface is the same, including the sidebar and the top navigation bar.

## ○ 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.

The screenshot shows the AWS IAM Role Details page for the role 'EC2S3AccessRole'. The left sidebar shows navigation options like Identity and Access Management (IAM), Access management, and Access reports. The main panel displays the role's summary, including its ARN (arn:aws:iam::278654638260:role/EC2S3AccessRole) and an instance profile ARN (arn:aws:iam::278654638260:instance-profile/EC2S3AccessRole). Below this, the 'Permissions' tab is selected, showing one managed policy named 'AmazonS3FullAccess'. The 'Trust relationships' tab is also visible. At the bottom, the 'Trusted entities' section shows the JSON trust policy:

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

● S3:

مهند محمد السعيد , عمر محمد عبد العال

- 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.

The screenshot shows the AWS S3 Properties page for the 'team69-file-share' bucket. The top navigation bar includes the AWS logo, search bar, and user information (Muhammad Mustafa Muhammad). The left sidebar has sections for 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) and Storage Lens (Dashboards, Storage Lens groups, AWS Organizations settings). The main content area is titled 'team69-file-share' with an 'Info' link. It features tabs for Objects, Properties (which is selected), Permissions, Metrics, Management, and Access Points. The 'Properties' tab displays the 'Bucket overview' section with details: AWS Region (Europe (Stockholm) eu-north-1), Amazon Resource Name (ARN) (arn:aws:s3:::team69-file-share), and Creation date (April 28, 2025, 14:33:02 (UTC+03:00)). Below this is the 'Bucket Versioning' section, which is disabled. A note about Multi-factor authentication (MFA) delete is present, stating it is disabled. At the bottom, there are links for CloudShell, Feedback, and copyright information (© 2025, Amazon Web Services, Inc. or its affiliates.).

## 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 مذكر

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.

**Upload**

Choose a file:

Choose File No file chosen

Upload Now

**Download**

DS\_Tools\_FinalProject.pdf-1746128873896  
[Click here to download](#)

DS\_Tools\_FinalProject.pdf-1746129150406  
[Click here to download](#)

DS\_Tools\_FinalProject.pdf-1746131035124  
[Click here to download](#)

Lec 4 \_ Regression .pdf-1746035094960  
[Click here to download](#)

arcane\_style an adult robot wearing pink shades... (1).png-1746021962434  
[Click here to download](#)

books\_clean.csv-1746123512190  
[Click here to download](#)

books\_clean.json-1746131061803  
[Click here to download](#)

phase1.pdf-1746034917758  
[Click here to download](#)

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

**Upload**

Choose a file:

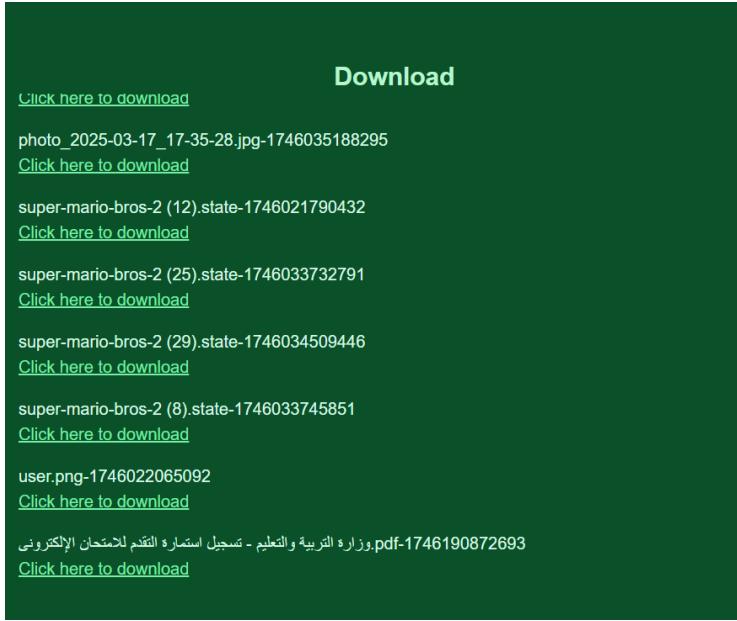
Choose File وزارة التربية والتعليم - ... اراء التقديم للامتحان الإلكتروني.pdf

Upload Now

Msg: File Uploaded Successfully!

Click here to download

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



## 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}`,
    }))
  } 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.**