Slip3

Create a custom AMI from your configured EC2 instance. What steps would you take to delete an AMI?

A) Steps to Create a Custom AMI

1. In **AWS Console**, go to **EC2 → Instances**.
2. Select the **configured instance** (e.g., with apps/settings installed).
3. From **Actions → Image and templates → Create image**.
4. Fill details:
   - **Image name:** e.g., `MyCustomServerAMI`.
   - **Description:** (optional, like "Exam custom AMI").
   - **No reboot:** (by default, AWS reboots instance → ensures clean image).
5. Click **Create image**.
6. Go to **EC2 → AMIs** → wait until **status = Available**.
   (Now this AMI can be used to launch new instances with the same configuration.)

B) Steps to Delete an AMI

1. Go to **EC2 → AMIs**.
2. Select the custom AMI (e.g., `MyCustomServerAMI`).
3. Click **Deregister AMI** → Confirm.
   (This removes the AMI entry; no new instances can be launched from it.)
4. Important: The AMI is backed by **snapshots in EBS**.
   - Go to **EC2 → Snapshots**.
   - Find the snapshot linked to the AMI.
   - Select → **Delete snapshot** → Confirm.
     (This actually frees storage and stops billing.)

Exam Notes:

- **AMI (Amazon Machine Image):** Template containing **OS + software + configuration** of an EC2 instance.
- **Custom AMI:** Useful for quickly launching identical environments.
- **Deleting AMI:** Must **deregister AMI + delete snapshots** to fully remove.

Q2) Create a two S3 bucket in AWS and perform the following operation.

- Upload files to an S3 bucket
- Download a bucket item.
- Copy a bucket item to another bucket.

Create two S3 buckets + Upload, Download, Copy files

A) Create Two Buckets

1. Go to **AWS Console → S3 → Create bucket**.
2. Enter **bucket name** (must be globally unique), e.g.,
   - `exam-bucket-1`
   - `exam-bucket-2`
3. Choose **Region** → e.g., `Asia Pacific (Mumbai).`
4. Leave defaults (Block Public Access ON).
5. Click **Create bucket**.

B) Upload File to Bucket

1. Open **exam-bucket-1**.
2. Click **Upload** → Add files (e.g., `test.txt`).
3. Click **Upload** → File is stored in bucket.

C) Download a Bucket Item

1. Open **exam-bucket-1** → Click on `test.txt`.
2. Select **Download**.
3. File will be saved locally.

D) Copy an Item to Another Bucket

Method 1 – Console:

1. In **exam-bucket-1**, select `test.txt`.
2. Click **Actions → Copy**.
3. Choose destination bucket = `exam-bucket-2`.
4. Paste → File is copied.

Exam Notes:

- **S3 (Simple Storage Service):** Object storage for files.
- Buckets are **containers** for objects.
- Operations used: **Upload → Download → Copy**.
- Permissions (IAM policy + bucket policy) must allow these actions.

Slip4

1)How do you create and manage AWS IAM users and groups?

Create and Manage AWS IAM Users and Groups

A) Steps to Create IAM User

1. Go to **AWS Console → IAM → Users → Add users**.
2. Enter **username** (e.g., `student1`).
3. Select **Access type**:
    - **AWS Management Console access** → for logging in to console.
    - **Programmatic access** → for CLI/SDK.
4. Set a **password** (or auto-generate).
5. Click **Next**.

B) Steps to Create IAM Group

1. Go to **IAM → User groups → Create group**.
2. Enter **group name** (e.g., `Developers`).
3. Attach a **policy** (e.g., `AmazonEC2ReadOnlyAccess`).
4. Click **Create group**.

C) Add User to Group

1. In **IAM → Users**, select `student1`.
2. Go to **Groups → Add user to group**.
3. Choose **Developers** group.
4. User now inherits group permissions.

D) Manage Users & Groups

- **Change permissions:** Attach/detach policies from group.
- **Password management:** Reset user passwords.
- **Delete user/group:** Remove when not needed.
- **Least privilege principle:** Give only required permissions.

Exam Notes:

- **IAM User** = An identity for people/apps to access AWS.
- **IAM Group** = A collection of users (permissions assigned once to group).
- **Best practice:**
    - Use **groups** for easier management.
    - Apply **policies** at group level.
    - **Don't share root account**, always use IAM users.

Q2) You want to ensure that your EC2 instance's data is backed up regularly. What methods would you use to back up data from your EC2 instance? Discuss options such as creating snapshots of EBS volumes and using AWS Backup.

A) EBS Snapshots

- **What it is:** A backup of your **EBS volume** (hard disk of EC2).

- **Steps to create manually:**
    1. Go to **EC2 → Volumes**.
    2. Select your EBS volume.
    3. Click **Actions → Create snapshot**.
    4. Give **name + description**.
    5. Snapshot is stored in **Amazon S3 (internally)**.
- **Automation:**
    - Use **Amazon Data Lifecycle Manager (DLM)** to schedule daily/weekly snapshots.

Benefits:

- Fast, incremental backups.
- Can restore by creating new volume from snapshot.

B) AWS Backup Service

- **What it is:** A fully managed backup service.
- **Steps:**
    1. Go to **AWS Backup → Create backup plan**.
    2. Define **backup rules** (frequency: daily/weekly, retention period).
    3. Assign **resources** (EC2 volumes, RDS, DynamoDB, etc.).
    4. Backups are created automatically as per plan.
- **Benefits:**
    - Centralized → one place to manage all backups.
    - Supports multiple AWS services.
    - Compliance and retention management built-in.

Slip6

How can you create an IAM policy that allows only read access to S3 buckets?

1. Steps to Create Policy (AWS Console)

1. Open **IAM → Policies → Create policy**.
2. Go to **JSON tab**.
3. Paste the following JSON (read-only to all buckets):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
```

```
      "Resource": [
        "arn:aws:s3:::*",
        "arn:aws:s3:::*/*"
      ]
    }
  ]
}
```

4. Review → Name the policy `S3ReadOnlyPolicy`.
5. Create policy.
6. Attach it to a **user, group, or role**.

---

2. Explanation of Policy

- `"s3:ListBucket"` → Lets user list bucket contents (see objects).
- `"s3:GetObject"` → Lets user download/read objects.
- `"arn:aws:s3:::*"` → Applies to all buckets.
- **No Put/Delete actions** → user cannot upload or remove objects.

---

3. Best Practice

- Apply to **specific buckets** instead of `*` for tighter security:

```
"Resource": [
  "arn:aws:s3:::my-example-bucket",
  "arn:aws:s3:::my-example-bucket/*"
]
```

- Always follow **principle of least privilege**.

---

4. Verification

- User can **list and download files** from bucket.
- Upload or delete attempt → **Access Denied**.

Create multiple key-pair and use same key-pair for multiple instances.

A) Creating Multiple Key Pairs

1. Open **AWS Console → EC2 → Key Pairs → Create key pair**.
2. Enter name, e.g.,
    - `exam-key1`
    - `exam-key2`
3. Choose **RSA** and `.pem` format.
4. Download the `.pem` files (e.g., `exam-key1.pem`, `exam-key2.pem`).
5. Set permissions on Linux/Mac:
6. `chmod 400 exam-key1.pem`
7. `chmod 400 exam-key2.pem`

B) Using the Same Key Pair for Multiple Instances

- When launching EC2 instances, in the **Key pair (login)** step:
  - Select the same key pair, e.g., `exam-key1`.
- Example:
  - Instance A → Key pair: `exam-key1`
  - Instance B → Key pair: `exam-key1`
  - Instance C → Key pair: `exam-key1`

 This means you can SSH into all these instances using the **same private key file (`exam-key1.pem`)**.

c)connect to instance

Instance A

ssh -i exam-key1.pem ec2-user@<Public-IP-A>

Instance B

ssh -i exam-key1.pem ec2-user@<Public-IP-B>

D) Important Notes (write in exam)

- **Multiple key pairs can be created**, but one EC2 instance can be associated with **only one key pair** at launch.
- The **same key pair can be reused** across multiple instances (easier management).
- If you lose the private key, you **cannot access** the instance → always back up securely.
- Best practice:
  - Use separate keys for different environments (e.g., dev vs. prod).
  - Share a key pair across instances only if **operationally safe**.

Slip7

How do you configure Network Access to an instance using Security groups?

To configure network access using Security Groups in AWS:

1. Open EC2 → Security Groups.
2. Create/Edit a Security Group and add **inbound rules** (e.g., SSH on port 22, HTTP on port 80).
3. Add **outbound rules** if needed (default allows all).
4. Attach the Security Group to the EC2 instance.

 Security Groups act like a virtual firewall controlling **who can enter (inbound)** and **where instance can go (outbound)**.

Q2 You have a web server EC2 instance that needs to be secured from unauthorized access while allowing necessary traffic. How would you configure security groups to secure your EC2 instance? Describe the process for creating and applying security group rules to allow only HTTP and SSH traffic while restricting all other access.

To secure a web server EC2 instance using Security Groups, we configure rules that allow only the required traffic (HTTP and SSH) and block everything else.

Steps:

1. **Login to AWS Console** → Go to **EC2 service**.
2. In the left panel, select **Security Groups** → Click **Create Security Group**.
3. Give a **name** (e.g., `WebServer-SG`) and **description**.
4. Add **Inbound Rules**:
    - **SSH (Port 22)** → Source: *Your IP* (for admin access).
    - **HTTP (Port 80)** → Source: `0.0.0.0/0` (to allow website access from anywhere).
5. Do not add any other inbound rules (this blocks all other traffic automatically).
6. Keep **Outbound Rules** as default (all traffic allowed) so the server can update packages.
7. Save the Security Group and **attach it to the EC2 instance**.

- Security Groups act as a **virtual firewall**.
- By adding only **SSH and HTTP**, we restrict access to administrators and website visitors while blocking all other ports.
- This ensures the EC2 web server is protected from unauthorized access.

Slip 8

1)If you have a custom AMI that you want to use to launch new EC2 instances. What are the steps to launch an EC2 instance using your custom AMI? Include details about choosing the AMI, selecting instance types, and configuring the instance settings.

Steps to Launch an EC2 Instance from a Custom AMI

1. **Log in to AWS Management Console**
   - Go to the **EC2 Dashboard**.
2. **Choose your Custom AMI**
   - In the left panel, click **AMIs**.
   - Select your **custom AMI** (the one you created earlier, either from a snapshot or an existing instance).
   - Click **Launch Instance from Image**.
3. **Select an Instance Type**
   - Pick the **instance type** (e.g., `t2.micro` for free tier, or something bigger if required).
   - This decides the **CPU, memory, and network capacity** for your instance.
4. **Configure Instance Settings**
   - **Number of instances**: Choose how many instances to launch.
   - **Network & Subnet**: Select the VPC and subnet where the instance should run.
   - **Auto-assign Public IP**: Enable if you want the instance reachable from the internet.
   - **IAM Role**: Attach a role if the instance needs AWS permissions.
   - **Shutdown behavior**: Choose stop or terminate on shutdown.
   - **Advanced details**: Add **user data scripts** if you want to run setup commands at launch.
5. **Add Storage (Volumes)**
   - By default, the AMI comes with its **root volume**.
   - You can **increase size** or **add extra EBS volumes** if needed.
6. **Add Tags**
   - Add tags like **Name=WebServer1** to easily identify the instance later.
7. **Configure Security Group**
   - Choose an **existing security group** or **create a new one**.
   - Example rules:
     - Allow **SSH (22)** for admin access.
     - Allow **HTTP (80)** if it's a web server.
     - Allow **HTTPS (443)** if using SSL.
8. **Review and Launch**
   - Check all the settings.
   - Click **Launch**.
   - Choose an existing **key pair** or create a new one (important for SSH login).
9. **Access Your Instance**
   - Once the instance state is **running**, copy its **Public IP / DNS**.
   - Connect using SSH or the service you configured.

✅**Shortly**
To launch an EC2 instance from a custom AMI, go to EC2 Dashboard → select your AMI → choose instance type → configure instance settings (network, IAM, shutdown behavior) → add storage → add tags → configure security group → review and launch → connect to the instance.

Q2 How do you configure AWS S3 versioning and lifecycle policies?

Configuring AWS S3 Versioning and Lifecycle Policies

1. Enable Versioning on a Bucket

Versioning helps keep multiple versions of an object in case of accidental overwrite or delete.

Steps:

1. Open **AWS Management Console** → Go to **S3 service**.
2. Select the bucket you want.
3. Go to the **Properties** tab.
4. Under **Bucket Versioning**, click **Edit**.
5. Choose **Enable** and save changes.

➡ Now, every time you upload a new file with the same name, S3 keeps both the old and the new versions.

---

2. Configure Lifecycle Policies

Lifecycle policies help you **automatically move data** to cheaper storage or **delete old versions** to save cost.

Steps:

1. Open the same bucket in **S3 console**.
2. Go to the **Management** tab → **Lifecycle rules**.
3. Click **Create lifecycle rule**.
    - Give a name (e.g., `Expire-Old-Versions`).
4. Choose what the rule applies to:
    - Whole bucket or specific prefix (e.g., `/logs/`).
5. Add **Actions**:
    - Transition objects to **S3 Glacier/Glacier Deep Archive** after X days.
    - Expire (delete) objects after Y days.
    - Permanently delete **previous versions** after Z days (useful with versioning).
6. Review and **Save rule**.

✅**Shortly**

- **S3 Versioning:** Enable it in bucket properties to keep multiple versions of objects.
- **Lifecycle Policies:** Define rules under Management tab to automatically transition objects to cheaper storage (like Glacier) or delete old/unused versions after a set time.

Slip9

Q1. If your EC2 instance is not responding to requests and appears to be down. What steps would you take to troubleshoot and diagnose the issue? Include common checks and diagnostic tools you might use to resolve the problem.

Troubleshooting an EC2 Instance That's Not Responding

When an EC2 instance seems down or unresponsive, you should check step by step:

---

1. Check Instance State in AWS Console

- Make sure the instance is in **Running** state.
- If it's **stopped** or **terminated**, start or launch a new one.

---

2. Check System Status Checks

- AWS provides **2 health checks**:
  - **System Status Check** → hardware/network issues on AWS side.
  - **Instance Status Check** → issues inside the OS (e.g., crash, kernel panic).
- If **system check fails**, open an AWS support ticket.
- If **instance check fails**, log in and fix OS-level issues.

---

3. Verify Networking

- Check the **Security Group rules**:
  - Allow inbound SSH (22) or RDP (3389) for admin access.
  - Allow required ports (e.g., 80 for HTTP, 443 for HTTPS).
- Check the **Network ACLs** – they must not block traffic.
- Ensure the instance has a **public IP/Elastic IP** if you want external access.
- Ping or run `telnet <public-ip> <port>` from your system to test connectivity.

---

4. Check SSH or RDP Access

- Try connecting using:
  - **SSH (Linux)**: `ssh -i key.pem ec2-user@public-ip`
  - **RDP (Windows)**: Remote Desktop client
- If blocked: check firewall rules inside instance (iptables, Windows firewall).

---

5. Inspect Instance Logs

- Use **EC2 Console → Actions → Monitor and troubleshoot → Get system log**.
- Look for boot errors, kernel panic, or application crashes.

---

6. Use AWS Session Manager (if configured)

- If SSH/RDP fails but **SSM Agent** is installed, connect via **Session Manager** in AWS Console.
- Run commands inside instance without needing network access.

---

7. Check Disk Space and CPU/Memory Usage

- If the instance boots but is very slow:
  - Use **CloudWatch metrics** to check CPU, memory, and disk I/O.
  - Full disk or high CPU usage can make the system unresponsive.

---

8. Recover with Recovery Options

- If instance OS is corrupted:
  - Stop the instance.
  - Detach its root volume.
  - Attach the volume to another healthy instance and repair files/config.
  - Reattach to original instance and start again.

---

shortly
To troubleshoot a non-responding EC2 instance:

1. Check instance state and AWS status checks.
2. Verify networking (security groups, NACLs, public IP).
3. Test SSH/RDP connectivity.
4. Review system/application logs.
5. Use Session Manager if configured.
6. Monitor with CloudWatch for CPU/memory/disk issues.
7. If needed, detach the volume and repair from another instance.

Q2) How do you add an inbound rule to a custom security group to allow HTTP traffic on port 80?

Adding an Inbound Rule for HTTP (Port 80) in a Security Group

Steps:

1. **Log in to AWS Console** → Go to **EC2 Dashboard**.
2. **Select Security Groups** (from left-hand menu).
3. Find your **custom security group** and click on it.
4. Go to the **Inbound rules** tab.
5. Click **Edit inbound rules** → then **Add rule**.

6. In the rule settings:
    - **Type**: Choose **HTTP**.
    - **Protocol**: Auto-fills as **TCP**.
    - **Port range**: Auto-fills as **80**.
    - **Source**: Select **0.0.0.0/0** (for all IPv4) and/or `::/0` (for all IPv6) if you want it public.
7. Click **Save rules**.

Slip10

Q1) How do you add tags to an existing EC2 instance?

Adding Tags to an Existing EC2 Instance

Steps:

1. **Log in to AWS Console** → Open **EC2 Dashboard**.
2. In the left panel, click **Instances**.
3. Select the **instance** you want to tag.
4. Scroll down to the **Tags** tab.
5. Click **Manage tags** (or **Add/Edit tags**).
6. Click **Add tag** → Enter:
    - **Key** (e.g., `Name`)
    - **Value** (e.g., `WebServer1`).
7. Click **Save**.

To add tags to an existing EC2 instance: go to **EC2 Dashboard → Instances → select instance → Tags tab → Manage tags → Add key-value pairs → Save**.

Q2 How do you configure AWS IAM policies for data access control?

Configuring AWS IAM Policies for Data Access Control

1. Understand What an IAM Policy Is

- An IAM Policy is a **JSON document** that defines **who can do what** on which AWS resources.
- Example: Allowing a user to only read objects from a specific S3 bucket.

---

2. Steps to Configure IAM Policy

1. **Open IAM Console**
    - Go to **AWS Management Console → IAM**.
2. **Create a Policy**
    - In IAM, click **Policies → Create policy**.
    - Choose **Visual editor** or **JSON**.
    - Define:
        - **Service** (e.g., S3, EC2).
        - **Actions** (e.g., `GetObject`, `PutObject`).
        - **Resources** (e.g., a specific bucket or all buckets).
3. **Review and Save**
    - Give the policy a **name** and **description**.
    - Click **Create policy**.
4. **Attach the Policy**

- Go to **Users, Groups, or Roles**.
- Select the target (e.g., a user).
- Click **Permissions → Attach policies**.
- Select the newly created policy → Save.

---

3. Example: S3 Read-Only Policy

A JSON snippet for read-only access to one bucket:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": ["arn:aws:s3:::my-bucket/*"]
    }
  ]
}
```

Slip11

Q1. How do you add an SSH public key to an existing EC2 instance using the AWS Management Console?

1. **Make a new key on your computer**
    - Run this command:
    - `ssh-keygen`
    - This gives you two files:
        - `new-key` (private key, keep safe)
        - `new-key.pub` (public key, to share with the server).
2. **Go to AWS Console → EC2 → Connect to your instance**
    - Use your **old key** to get inside first.
3. **Tell the server about your new key**
    - Once inside, open the file called:
    - `~/.ssh/authorized_keys`
    - Paste the content of `new-key.pub` (your public key) into this file.
4. **Save and Exit**
    - Now the server knows your new key.
5. **Use your new key to login**
    - Next time, connect with:
    - `ssh -i new-key ec2-user@<your-instance-ip>`

✅**Shortly:**
To add a new SSH key:

1. Generate a new key pair on your computer.
2. Connect to the EC2 instance with the old key.
3. Add the new public key into `~/.ssh/authorized_keys`.
4. Save it and set permissions.
5. Use the new private key to log in.

Q2)How does an IAM user goes towards enabling Multi-Factor Authentication (MFA)?

Steps:

1. **Log in to AWS Console**
    - Use your **IAM user credentials** (username + password).
2. **Go to IAM Dashboard**
    - Click your **username** (top-right corner → *My Security Credentials*).
3. **Select Multi-Factor Authentication (MFA)**
    - Under **MFA**, click **Activate MFA**.
4. **Choose MFA Device**
    - Options:
        - **Virtual MFA app** (like Google Authenticator, Authy).
        - **Hardware token** (physical device).
5. **Configure the Device**
    - If using a phone app:
        - Scan the **QR code** shown on screen.

        - The app will start generating **6-digit codes**.
6. **Verify**
    - Enter **two consecutive codes** from the app into AWS Console.
    - This proves the device is linked correctly.
7. **Finish**
    - Click **Assign MFA**.
    - From now, logging in requires **password + MFA code**.

✅**Shortly:**
An IAM user enables MFA by logging in → going to **My Security Credentials** → selecting **Activate MFA** → choosing a device (virtual app or hardware) → scanning QR code → entering two generated codes → saving. Now login requires both password and MFA code.

Slip12:

Q1 How can you create an IAM policy that allows only read access to S3 buckets?

Creating an IAM Policy for Read-Only Access to S3

Steps:

1. **Log in to AWS Console → IAM service**
2. Go to **Policies → Create policy**.
3. In the **Visual editor**:
   - **Service** → Choose **S3**.
   - **Actions** → Select **Read** (like `GetObject`, `ListBucket`).
   - **Resources** → Choose **All buckets** or select a specific bucket.
4. **Review and Create**
   - Give the policy a **name** (e.g., `S3ReadOnlyPolicy`).
   - Save it.
5. **Attach the Policy**
   - Go to **Users / Groups / Roles**.
   - Attach this policy to whoever needs read-only access.

Example JSON Policy

This gives read-only access to **all S3 buckets**:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": "*"
    }
  ]
}
```

Q2 How would you create a security group to ensure your EC2 instance is accessible only through necessary ports (e.g., HTTP and SSH)? What inbound and outbound rules would you configure, and why?

Q2: Creating a Security Group for EC2 (Only HTTP & SSH Access)

1. Create Security Group

- Go to **AWS EC2 Dashboard → Security Groups → Create security group**.
- Give it a **name** (e.g., `WebServer-SG`) and description.

- Select the right **VPC**.

---

2. Configure Inbound Rules (Allow only necessary traffic)

- **SSH (Port 22)**: Allow access so admins can connect and manage the server.
  - Source: `My IP` (recommended for security) or `0.0.0.0/0` if needed from anywhere.
- **HTTP (Port 80)**: Allow access for web traffic.
  - Source: `0.0.0.0/0` (IPv4) and `::/0` (IPv6) so anyone can access the website.

✅Why?

- SSH → for admin control.
- HTTP → for users to access the website.
- No other ports open → reduces attack surface.

---

3. Configure Outbound Rules (Default: allow all traffic)

- By default, outbound is set to **All traffic (0.0.0.0/0 and ::/0)**.
- This allows the instance to download updates, connect to external APIs, etc.
- Usually, you keep this default unless you need stricter control.

---

4. Attach the Security Group

- While launching an EC2 instance → select this **custom security group**.

✅**Shortly:**
To secure an EC2 instance, create a security group with:

- **Inbound rules:** Allow SSH (22) from admin IP and HTTP (80) from all.
- **Outbound rules:** Allow all traffic for updates and communication.
  This ensures the instance is only accessible on required ports, reducing security risks.

Slip13:

Q1 Your EC2 instance needs to access other AWS services, such as S3 buckets, securely. How would you set up an IAM role for your EC2 instance to grant it access to specific AWS services? Explain how you would attach the role to your instance and configure permissions.

Setting up an IAM Role for EC2 to Access AWS Services (like S3)

1. Why Use an IAM Role?

- Instead of storing AWS **access keys** inside your EC2 instance (which is risky), you attach an **IAM role**.
- The role gives **temporary secure permissions** to the instance.

---

2. Steps to Create the Role

1. **Go to IAM Console → Roles → Create role**.
2. **Choose Trusted Entity**: Select **AWS Service**.
3. **Choose Service**: Pick **EC2** (because you want the role for an EC2 instance).
4. **Attach Policies**:
    - Example: `AmazonS3ReadOnlyAccess` if you only want S3 read access.
    - Or create a **custom policy** to allow specific actions.
5. **Name the Role**: e.g., `EC2-S3-Access-Role`.
6. **Create Role**.

---

3. Attach Role to EC2 Instance

- Go to **EC2 Dashboard → Instances**.
- Select your instance → **Actions → Security → Modify IAM role**.
- Choose the role (`EC2-S3-Access-Role`) → Save.

---

4. How It Works

- Now, your EC2 instance automatically gets **temporary credentials** for that role.
- Inside the instance, you can run commands like:
- `aws s3 ls`

  and it will access S3 **without storing any keys**.

✅**Shortly:**
To allow EC2 to securely access AWS services like S3:

1. Create an IAM role with **EC2 as the trusted service**.

2. Attach required policies (e.g., S3 read-only).
3. Attach the role to the EC2 instance via **Modify IAM Role**.
4. EC2 now uses temporary credentials to access services without needing keys.

Q2. How can you access the system logs of an EC2 instance from the AWS Management Console?

Slip14

Q1) Launch a new EC2 instance to run a basic web server application. Describe the steps you would take to launch an EC2 instance. Include details about choosing an AMI, selecting an instance type, configuring instance details, and assigning a security group.

Steps to Launch EC2 Instance

1. **Login to AWS Console**
   - Go to **AWS Management Console → EC2 Service → Launch Instance**.
2. **Choose an AMI (Amazon Machine Image)**
   - Select an OS image for your server.
   - Example: **Amazon Linux 2** or **Ubuntu**.
   - This image provides the base operating system + preinstalled packages.
3. **Choose an Instance Type**
   - Select hardware size (CPU, RAM).
   - Example: **t2.micro** (free-tier eligible, good for small web servers).
4. **Configure Instance Details**
   - Set number of instances (usually 1).
   - Choose the **VPC and subnet** (default is fine for beginners).
   - Assign **IAM role** if the server needs AWS access (e.g., S3).
5. **Add Storage**
   - Define disk size (default: 8GB).
   - Increase if your application needs more space.
6. **Add Tags (Optional but Recommended)**
   - Example: `Name = WebServer`.
   - Helps in identifying the instance easily.
7. **Configure Security Group (Firewall Rules)**
   - Create a new security group or use an existing one.
   - Allow **SSH (port 22)** for remote access.
   - Allow **HTTP (port 80)** for web traffic.
   - Allow **HTTPS (port 443)** if secure web access is needed.
8. **Review and Launch**
   - Double-check all settings.
   - Click **Launch**.
   - Select or create a **key pair** (used for SSH login).
9. **Connect to the Instance**
   - After it launches, copy the **Public IP**.
   - Connect using:
     - SSH (Linux/Mac) → `ssh -i key.pem ec2-user@<public-ip>`
     - Putty (Windows).
10. **Install Web Server Application**

- Example on Amazon Linux:
- `sudo yum update -y`
- `sudo yum install -y httpd`
- `sudo systemctl start httpd`
- `sudo systemctl enable httpd`
- `echo "Hello from my EC2 Web Server" | sudo tee /var/www/html/index.html`
- Now, visit `http://<public-ip>` in a browser.

Q2) Q2. **How can you launch an EC2 instance with a specific IAM role?**

Step-by-Step Explanation

1. **Create or Verify an IAM Role**
    - Go to **IAM → Roles → Create Role**.
    - Select **EC2** as the trusted entity.
    - Attach the required policies (e.g., S3 ReadOnlyAccess).
    - Name the role (e.g., `EC2-AccessRole`).
2. **Go to EC2 Console → Launch Instance**
    - Click **Launch Instance**.
3. **Choose an AMI (Amazon Machine Image)**
    - Select an OS (e.g., Amazon Linux 2 or Ubuntu).
4. **Choose Instance Type**
    - Example: **t2.micro** for small workloads.
5. **Configure Instance Details**
    - In this step, there is a field called **IAM Role**.
    - From the dropdown, select the role you created (`EC2-AccessRole`).
    - This links the role to your instance.
6. **Add Storage**
    - Set default (8 GB) or more as needed.
7. **Add Tags**
    - Example: `Name = IAMRoleInstance`.
8. **Configure Security Group**
    - Add inbound rules (e.g., SSH port 22, HTTP port 80).
9. **Review and Launch**
    - Confirm all settings.
    - Select a **key pair** for SSH access.
    - Launch the instance.
10. **Verify Access**

- Once running, connect via SSH.
- Test permissions (e.g., run `aws s3 ls`).
- The instance will securely use the IAM role without storing keys.

Slip16:

Q1. Your EC2 instance needs to send outbound traffic to an external API but should not allow any other outbound traffic. How would you configure the outbound rules of your security group to permit only the necessary traffic? What considerations would you take into account for setting these rules?

Q1. Outbound Traffic Control for EC2 Instance

Step 1: Requirement

- EC2 should **send traffic only to an external API**.
- No other outbound traffic should be allowed.

Step 2: Configure Security Group Outbound Rules

1. Go to **EC2 Console → Security Groups → Select Security Group → Outbound rules → Edit**.
2. **Remove the default rule** (which usually allows all outbound traffic).
3. **Add a specific outbound rule**:
    - **Protocol:** TCP (or as required by API).
    - **Port Range:** API port (e.g., 443 for HTTPS, 80 for HTTP).
    - **Destination:**
        - Either the **IP address** of the external API.
        - Or its **CIDR range**, if provided.

This ensures EC2 can only talk to the API, and nothing else.

Step 3: Considerations

- ✅**Know the API endpoint and port** (most APIs use HTTPS on port 443).
- ✅**Use least privilege principle** → allow only what is needed.
- ✅**Check for dependencies** → sometimes your EC2 might also need DNS (port 53) or NTP (port 123) for time sync.
- ✅**Application Failures** → If you block everything except the API, ensure your app won't break (e.g., it may need OS updates).
- ✅**Monitoring** → Use **VPC Flow Logs** or CloudWatch to confirm only expected traffic flows.

✅shortly Answer

To allow an EC2 instance to send outbound traffic only to an external API:

1. Edit the **outbound rules** of its security group.
2. Remove the default "allow all outbound traffic" rule.

3. Add a specific outbound rule permitting only the required **protocol (TCP)**, **port (443/80)**, and **destination (API IP or CIDR)**.
4. This ensures the EC2 can communicate with the API but no other destinations.

Considerations:

- Know the API endpoint and required port.
- Apply the **principle of least privilege**.
- Allow supporting traffic if needed (DNS, NTP).
- Monitor with **logs** to verify correct behavior.

- ✅ This way, the EC2 instance is secure and restricted to only the required outbound API traffic.
- Would you like me to also make a **diagram (EC2 → Security Group → API only)** that you can quickly sketch in your exam for extra marks?

Q2 How does AWS handle connections when an EC2 instance is stopped and started again?

Q2. AWS Connections When an EC2 Instance is Stopped and Started

When you **stop and start** an EC2 instance, AWS handles networking and instance state in a specific way:

---

1. Instance State Changes

- **Stop** → The instance **shuts down**, and the virtual server stops running.
- **Start** → AWS provisions it on physical hardware again (might be different host).

---

2. Public IP Address

- If the instance has an **auto-assigned public IP**:
  - The public IP **changes** after stopping and starting.
  - Old public IP is released.
- If the instance has an **Elastic IP** (EIP) attached:
  - The IP **remains the same**.

✅*Tip:* Use Elastic IP if you need a **fixed public IP** across stop/start cycles.

---

3. Private IP Address

- **Private IP in the VPC** usually **remains the same** unless you change the subnet or manually modify it.
- Internal communications in the VPC continue using the private IP.

---

4. Connections and Sessions

- Any **active connections (SSH, HTTP, TCP)** are **terminated** when the instance is stopped.
- You need to **reconnect** once the instance is started again.

---

5. Storage and Data

- **EBS volumes**: Data **persists** across stop/start.
- **Instance Store volumes**: Data is **lost** when instance stops.

✅shortly

1. Stopping an EC2 shuts down the instance; starting it provisions it again.
2. Auto-assigned public IP **changes**; Elastic IP remains fixed.
3. Private IP usually **stays the same**.
4. All active connections are **terminated**; you must reconnect.
5. EBS-backed storage **persists**; instance store **loses data**.
6. Networking and security group rules **remain unchanged**.

Slip17:

Q1 How do you restrict HTTP traffic to only a specific IP address or range in a Security Group?

Q1. How to Restrict HTTP Traffic to a Specific IP or Range in a Security Group

Step 1: Understand the Requirement

- EC2 instance is running a web server (HTTP port 80).
- Only **specific IP(s) or range of IPs** should access it.
- All other traffic should be denied.

✅*Why:* Restricting access reduces the attack surface and prevents unauthorized users from accessing the web server.

Step 2: Go to the Security Group

1. Open **AWS Management Console → EC2 → Security Groups**.
2. Select the security group attached to your EC2 instance.
3. Click **Inbound rules → Edit inbound rules**.

Step 3: Add HTTP Rule with Restricted Source

1. Click **Add Rule**.
2. **Type:** HTTP (Port 80)
3. **Protocol:** TCP (default for HTTP)
4. **Port Range:** 80
5. **Source:**
   - Single IP → `203.0.113.25/32` (example)
   - IP range → `203.0.113.0/24` (example, allows all addresses from 203.0.113.0 to 203.0.113.255)
6. Remove any other rules allowing HTTP from `0.0.0.0/0` (all IPs) if present.

✅*Why:* CIDR notation lets you define precise IP(s) or range of addresses for controlled access.

Step 4: Save the Rule

- Click **Save rules**.
- The security group now allows **HTTP access only from the specified IP(s)/range**.

Step 5: Considerations

1. **Least Privilege:** Only allow IPs that absolutely need access.

2. **Dynamic IPs:** If the allowed users have dynamic IPs, consider VPN or bastion host instead of hardcoding IPs.
3. **Other Ports:** Ensure other services (SSH, HTTPS) also follow access restrictions.
4. **Testing:** Verify the restriction by trying to access the web server from allowed and disallowed IPs.
5. **Logging:** Enable **VPC Flow Logs** to monitor allowed/denied traffic.

Step 6: Optional Enhancements

- Combine with **Network ACLs** for an additional layer of filtering.
- Use **WAF (Web Application Firewall)** if you need application-level IP filtering.

✅shortly:

To restrict HTTP traffic to a specific IP or range in a Security Group:

1. Open **AWS Console → EC2 → Security Groups**.
2. Select the security group attached to your EC2 instance.
3. Click **Inbound rules → Edit inbound rules**.
4. Add a rule:
    - Type: HTTP (Port 80)
    - Protocol: TCP
    - Port: 80
    - Source: Enter specific IP (`x.x.x.x/32`) or IP range (`x.x.x.x/24`).
5. Remove any existing rules that allow HTTP from all IPs (`0.0.0.0/0`).
6. Save the rules.
7. Considerations:
    - Apply **least privilege**, only allow necessary IPs.
    - For dynamic IPs, consider **VPN or bastion hosts**.
    - Test access from allowed/disallowed IPs.
    - Enable **VPC Flow Logs** to monitor traffic.
8. Optional: Use **Network ACLs** or **WAF** for additional filtering.

This configuration ensures that only the authorized IP(s) can access the HTTP port, reducing the risk of unauthorized access.

Q2)How do you attach a policy to an IAM user using the AWS Management Console?

Attaching a Policy to an IAM User Using AWS Console

Step 1: Log in to AWS Console

- Open **AWS Management Console → IAM service**.

Step 2: Go to Users

- Click **Users** in the left menu.
- Select the **IAM user** you want to attach the policy to.

Step 3: Attach Policy

1. Go to the **Permissions** tab of the user.
2. Click **Add permissions → Attach policies directly**.
3. You will see a list of AWS managed and custom policies.
4. Check the **policy** you want to attach (e.g., `AmazonS3ReadOnlyAccess`).

Step 4: Review and Add Permissions

- Click **Next: Review → Add permissions**.
- The policy is now attached, and the user can perform the actions allowed by that policy.

✅shortly:

To attach a policy to an IAM user:

1. Go to **AWS Console → IAM → Users**.
2. Select the user → **Permissions → Add permissions → Attach policies directly**.
3. Choose the desired **AWS managed or custom policy**.
4. Click **Add permissions** to attach the policy.

Slip18:

Q1. How do you add an outbound rule to a Network ACL to block all traffic to the internet?

Step 1: Understand the Requirement

- You want to **block all outbound traffic** from a subnet to the Internet.
- Network ACLs (NACLs) are **stateless firewalls** for subnets.
- Outbound rules control traffic leaving the subnet.

Step 2: Open the NACL in AWS Console

1. Go to **AWS Management Console → VPC → Network ACLs**.
2. Select the **NACL** associated with your subnet.
3. Click on **Outbound Rules → Edit outbound rules**.

Step 3: Add a Deny Rule

1. Click **Add Rule**.
2. Set **Rule Number** (lower number has higher priority, e.g., 100).
3. **Type:** All Traffic (or specific protocols if desired).
4. **Protocol:** All (-1)
5. **Port Range:** All
6. **Destination:** 0.0.0.0/0 (IPv4) and ::/0 (IPv6 if needed)
7. **Action:** Deny

✅*Why:* This explicitly blocks all outbound traffic to any destination on the Internet.

Step 4: Save the Rule

- Click **Save rules**.
- The NACL now blocks all outbound traffic leaving the subnet.

Step 5: Considerations

1. NACLs are **stateless**, so you may also need corresponding **inbound rules** if needed.
2. Make sure you are **not blocking traffic required for essential services**, e.g., DNS (port 53), NTP (port 123), or AWS service endpoints.
3. Rule numbers matter: NACLs process rules in **ascending order**, and the first matching rule is applied.

✅shortly:

To block all outbound traffic to the Internet using a Network ACL:

1. Open **VPC → Network ACLs** in AWS Console.
2. Select the NACL linked to your subnet → **Outbound Rules → Edit**.
3. Add a rule:
    - Rule Number: 100
    - Type: All Traffic
    - Protocol: All (-1)
    - Port Range: All
    - Destination: 0.0.0.0/0 (and ::/0 for IPv6)
    - Action: Deny
4. Save the rules.

Considerations:

- NACLs are **stateless**, so inbound/outbound rules are independent.
- Check required traffic (DNS, NTP, AWS services) before blocking all traffic.
- Rule order determines priority; lower numbers are applied first.

Q2. How do you create an IAM policy that denies all access to a specific S3 bucket?

Step 1: Understand the Requirement

- You want to **deny all actions** on a particular S3 bucket.
- IAM policies can **allow or deny** actions.
- Deny always **overrides allow**, so it's a safe way to restrict access.

Step 2: Go to IAM Console

1. Open **AWS Management Console → IAM → Policies → Create Policy**.
2. Choose **JSON** tab for more control.

Step 3: Write the Policy JSON

Example JSON to deny all access to a bucket named `my-sensitive-bucket`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::my-sensitive-bucket",
                "arn:aws:s3:::my-sensitive-bucket/*"
            ]
        }
    ]
}
```

✅*Explanation:*

- `"Effect": "Deny"` → explicitly denies access.
- `"Action": "s3:*"` → all S3 actions (read/write/delete/list).
- `"Resource"` → applies to the bucket itself **and all objects inside**.

Step 4: Review and Create

- Click **Next → Review**.
- Give the policy a **name**, e.g., `DenyAllMySensitiveBucket`.
- Click **Create Policy**.

Step 5: Attach Policy

- Go to **IAM Users, Groups, or Roles → Attach Policy**.
- Select the newly created policy to enforce the deny.

Step 6: Considerations

1. Deny policies **override all allows**, so even admin users will be blocked for this bucket.
2. Use **least privilege principle** for other buckets.
3. Test by trying to access the bucket with a user/role that has this policy.

✅Shortly

To create an IAM policy that denies all access to a specific S3 bucket:

1. Open **IAM → Policies → Create Policy → JSON**.
2. Write a JSON policy:
   - `"Effect": "Deny"`
   - `"Action": "s3:*"`
   - `"Resource": "arn:aws:s3:::<bucket-name>"` and `"arn:aws:s3:::<bucket-name>/*"`
3. Review, name the policy, and **Create Policy**.
4. Attach the policy to **users, groups, or roles** that should be denied access.

Considerations:

- Deny overrides any allows.
- Ensure the deny policy is applied only to the intended bucket.
- Test to verify that access is correctly blocked.

Slip19:

Q1. How do you implement a Network ACL that allows only HTTPS traffic on port 443 and denies all other traffic?

Step 1: Understand the Requirement

- Allow **only HTTPS (port 443)**.
- Deny all other inbound and outbound traffic.
- Network ACLs (NACLs) are **stateless**, meaning both inbound and outbound rules must be explicitly defined.

Step 2: Open the NACL in AWS Console

1. Go to **AWS Console → VPC → Network ACLs**.
2. Select the **NACL** associated with your subnet.
3. Click **Inbound Rules → Edit inbound rules**.

Step 3: Configure Inbound Rules

1. **Add Allow Rule for HTTPS:**
    - **Rule Number:** 100 (lower numbers are higher priority)
    - **Type:** HTTPS
    - **Protocol:** TCP
    - **Port Range:** 443
    - **Source:** 0.0.0.0/0 (or restrict to a specific IP/range if needed)
    - **Action:** Allow
2. **Add Deny Rule for All Other Traffic:**
    - **Rule Number:** 200
    - **Type:** All Traffic
    - **Protocol:** All
    - **Port Range:** All
    - **Source:** 0.0.0.0/0
    - **Action:** Deny

Step 4: Configure Outbound Rules

- Follow the same approach:

1. **Allow HTTPS traffic only:**
    - Rule Number: 100
    - Type: HTTPS
    - Protocol: TCP
    - Port: 443
    - Destination: 0.0.0.0/0
    - Action: Allow
2. **Deny all other outbound traffic:**

- Rule Number: 200
- Type: All Traffic
- Protocol: All
- Port Range: All
- Destination: 0.0.0.0/0
- Action: Deny

Step 5: Save Rules

- Click **Save rules** for both inbound and outbound rules.
- The NACL now only allows HTTPS traffic and blocks all other traffic.

Step 6: Considerations

1. **Stateless Nature:** NACLs are stateless → both inbound and outbound rules are required.
2. **Rule Order:** Lower numbers are processed first; first match is applied.
3. **Essential Services:** If the instance requires DNS (port 53) or NTP (port 123), allow them explicitly.
4. **Testing:** Verify by trying to access HTTP, SSH, or other ports—they should be blocked; HTTPS should work.

✅Shortly

To implement a NACL that allows only HTTPS traffic:

1. Open **VPC → Network ACLs → Select NACL**.
2. **Inbound Rules:**
   - Allow HTTPS (TCP port 443) from required source.
   - Deny all other traffic (All Traffic, All Protocols, All Ports).
3. **Outbound Rules:**
   - Allow HTTPS (TCP port 443) to required destination.
   - Deny all other traffic.
4. Save the rules.
5. Considerations: NACLs are stateless, rule numbers matter, essential services like DNS may need explicit allow.

Q2. How do you grant read access to a specific S3 bucket for an IAM user?

Step 1: Understand the Requirement

- Read access means the user can **list objects** and **get/download objects** from the bucket.
- We will create or attach an IAM policy to allow only these actions.

Step 2: Go to IAM Console

1. Open **AWS Management Console → IAM → Policies → Create Policy**.
2. Choose the **JSON** tab for precise control.

Step 3: Write the Policy JSON

Example JSON for read-only access to bucket `my-example-bucket`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::my-example-bucket",
                "arn:aws:s3:::my-example-bucket/*"
            ]
        }
    ]
}
```

✅*Explanation:*

- `"Effect": "Allow"` → Grants permission.
- `"Action": ["s3:GetObject", "s3:ListBucket"]` → Allows reading the bucket and objects.
- `"Resource"` → Bucket ARN for the bucket itself + `/*` for all objects inside.

Step 4: Review and Create Policy

- Click **Next → Review**.
- Give a name, e.g., `ReadMyExampleBucket`.
- Click **Create Policy**.

Step 5: Attach Policy to IAM User

1. Go to **IAM → Users → Select User → Permissions → Add Permissions**.
2. Choose **Attach policies directly**.
3. Select the policy you just created (`ReadMyExampleBucket`).
4. Click **Add permissions**.

Step 6: Test Access

- Log in as the IAM user and try accessing the bucket.
- User should be able to **list and read objects** but **cannot upload, delete, or modify** anything.

✅shortly

To grant read access to a specific S3 bucket for an IAM user:

1. Go to **IAM → Policies → Create Policy → JSON**.
2. Write a policy allowing:
    - `"Action": ["s3:GetObject", "s3:ListBucket"]`
    - `"Resource": ["arn:aws:s3:::<bucket-name>", "arn:aws:s3:::<bucket-name>/*"]`
3. Review and create the policy.
4. Attach the policy to the IAM user via **Permissions → Add permissions → Attach policy**.
5. Test to ensure the user can read objects but cannot modify or delete them.

Slip20:

Q1. How do you allow only traffic from other EC2 instances in the same security group?

Allow Only Traffic from Other EC2 Instances in the Same Security Group

Step 1: Understand the Requirement

- You want your EC2 instances to communicate **only with other instances in the same security group**.
- This is useful for private communication between backend servers while blocking all other traffic.

Step 2: Edit Security Group Inbound Rules

1. Go to **AWS Console → EC2 → Security Groups**.
2. Select the **security group** attached to your EC2 instances.
3. Click **Inbound Rules → Edit inbound rules → Add Rule**.
4. Configure the rule:
    - **Type:** Choose the protocol/port you want (e.g., SSH, HTTP, or custom).
    - **Protocol:** TCP (default for most services).
    - **Port Range:** Specify the port (e.g., 22 for SSH).
    - **Source:** Select **Custom** → Enter the **security group ID** itself (e.g., `sg-123456`).
5. Save the rule.

✅*Why:* By specifying the **security group itself as the source**, only instances with the same security group can access this port.

Step 3: Edit Outbound Rules (Optional)

- Outbound rules are usually **allow all by default**, but you can restrict them similarly:
    - **Destination:** The same security group ID.
    - Only instances in this group can receive outbound traffic.

Step 4: Considerations

1. **Least Privilege:** Only open required ports between instances.
2. **Security Group ID:** Always use the actual **security group ID** (`sg-xxxxxx`) rather than CIDR.
3. **Testing:** Verify by trying to access the instance from inside vs. outside the security group.

✅shortly:

To allow traffic only from other EC2 instances in the same security group:

1. Open **EC2 → Security Groups → Select Security Group → Inbound Rules → Edit**.
2. Add a rule:
    - Type: Select protocol/port (e.g., SSH/HTTP/custom)
    - Protocol: TCP
    - Port: Required port (e.g., 22, 80)
    - Source: **Security Group ID itself**
3. Save the rule.
4. (Optional) Restrict outbound traffic to the same security group.
5. Test to ensure only instances in the same group can communicate.

✅*Result:* Only instances within the same security group can access each other; all external traffic is blocked.

Q2. How can you set up an S3 bucket to be publicly accessible?

How to Set Up an S3 Bucket to Be Publicly Accessible

Step 1: Create or Select the Bucket

1. Open **AWS Management Console → S3 → Buckets**.
2. Click **Create Bucket** or select an existing bucket.
3. Give it a name and choose the region.

Step 2: Disable Block Public Access

By default, S3 **blocks public access**. To make the bucket public:

1. Go to **Permissions → Block public access (bucket settings)**.
2. Click **Edit**.
3. **Uncheck** the boxes for blocking public access:
    - Block all public access
    - Block public ACLs
    - Block public bucket policies
4. Click **Save** and confirm.

✅*Why:* S3 blocks public access by default for security. You need to allow public access explicitly.

Step 3: Add a Bucket Policy for Public Access

1. Go to **Permissions → Bucket Policy → Edit**.
2. Add a JSON policy like this to allow public read access:

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "PublicReadAccess",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::your-bucket-name/*"
        }
    ]
}
```

✅*Explanation:*

- `"Principal": "*"` → Anyone (public) can access.
- `"Action": "s3:GetObject"` → Allows read/download only.
- `"Resource": "arn:aws:s3:::your-bucket-name/*"` → Applies to all objects in the bucket.

Step 4: Set Object Permissions (Optional)

- If you upload new files, ensure **public read access** is enabled for them, either via **ACLs** or the bucket policy.

Step 5: Test Public Access

1. Copy the **object URL** (e.g., `https://your-bucket-name.s3.amazonaws.com/file.txt`).
2. Open it in a browser → it should be accessible without AWS credentials.

Step 6: Considerations

1. Making a bucket public can **expose data to anyone**, so only do this for non-sensitive files.
2. Prefer using **CloudFront** or **signed URLs** for controlled public access.
3. Always test public access and monitor bucket usage.

✅Shortly

To make an S3 bucket publicly accessible:

1. Open **S3 → Select/Create Bucket**.
2. Go to **Permissions → Block Public Access** and disable all blocks.
3. Add a **bucket policy** allowing `s3:GetObject` for `Principal: "*"`.
4. Ensure uploaded objects inherit public read permissions.
5. Test by accessing the object URL in a browser.

Considerations: Public access exposes data → use only for non-sensitive files or static websites.