### ****Step 1: EC2 Instance and Volume Creation****

#### ****1.1 Generate Key Pair (if you don't have one)****

1. Go to the [AWS Management Console](https://aws.amazon.com/console/).
2. Navigate to the **EC2 Dashboard**.
3. In the left menu, under **Network & Security**, select **Key Pairs**.
4. Click **Create Key Pair** and choose:
   * **Name**: Give a name (e.g., my-key-pair).
   * **Key Pair Type**: Select RSA.
   * **Private Key Format**: Select PPK (for using in PuTTY).
5. Download the key pair.

#### ****1.2 Launch an EC2 Instance****

1. In the **EC2 Dashboard**, click **Launch Instance**.
2. Configure the instance:
   * **AMI**: Choose **Amazon Linux 2** or **Ubuntu**.
   * **Instance Type**: Choose t2.micro (eligible for the free tier).
   * **Key Pair**: Choose the key pair you created earlier.
   * **Network Settings**: Select the default VPC and make sure the security group allows SSH (port 22) and HTTP (port 80).
3. Click **Launch** to start the instance.

#### ****1.4 Connect to the EC2 Instance Using PuTTY****

1. Open **PuTTY** and enter the **Public DNS (IPv4)** of your EC2 instance as the **Host Name**. You can find this in the EC2 instance details.
2. In the left menu, go to **Connection → SSH → Auth**.
3. Click **Browse** and select your .ppk file.
4. Go back to **Session**, give a name under **Saved Sessions**, and click **Save**.
5. Click **Open** to SSH into the instance. The default username is:
   * For **Amazon Linux**: ec2-user.
   * For **Ubuntu**: ubuntu.

#### ****1.5 Attach an EBS Volume to the EC2 Instance****

1. Go to **EC2 Dashboard**, click **Volumes** in the **Elastic Block Store** section.
2. Click **Create Volume** and configure:
   * **Volume Type**: Choose gp2 or gp3 for general-purpose SSD.
   * **Size**: 20 GB or based on your need.
   * **Availability Zone**: It must match your EC2 instance's zone (e.g., us-east-1a).
3. Once created, go to **Actions → Attach Volume**.
4. Select the EC2 instance and click **Attach**.

#### ****1.6 Mount the Volume in the EC2 Instance****

1. SSH into the instance using PuTTY.
2. Run lsblk to list available disks.
3. Format and mount the volume:

bash

Copy code

sudo mkfs -t ext4 /dev/xvdf # Replace /dev/xvdf with your volume name

sudo mkdir /mnt/my-volume

sudo mount /dev/xvdf /mnt/my-volume

1. Add the mount to /etc/fstab for persistence:

bash

Copy code

sudo nano /etc/fstab

Add:

bash

Copy code

/dev/xvdf /mnt/my-volume ext4 defaults,nofail 0 2

### ****Step 2: S3 Bucket Creation****

#### ****2.1 Create an S3 Bucket****

1. Go to the [S3 Console](https://s3.console.aws.amazon.com/s3/home).
2. Click **Create Bucket**.
3. Configure the bucket:
   * **Bucket Name**: Use a unique name (e.g., my-app-bucket).
   * **Region**: Match the region of your EC2 instance.
   * **Block Public Access**: Keep public access blocked unless you need it.
4. Click **Create Bucket**.

#### ****2.2 Upload Objects to the S3 Bucket****

1. After creating the bucket, click on its name.
2. Click **Upload** and choose files to upload (e.g., static files like images).

#### ****2.3 Setting Up Access to S3 from EC2****

##### **Create an IAM Role for S3 Access**

1. Go to the **IAM Console** and click **Roles**.
2. Click **Create Role**, select **EC2** as the trusted entity.
3. Attach the **AmazonS3FullAccess** policy (or create a custom policy with restricted permissions).
4. Name the role (e.g., EC2-S3-Access) and click **Create Role**.

##### **Attach IAM Role to EC2 Instance**

1. Go to the **EC2 Dashboard**, select your instance, click **Actions → Security → Modify IAM Role**.
2. Select the IAM role (EC2-S3-Access) and click **Update IAM Role**.

##### **Install AWS CLI in EC2 Instance**

1. SSH into the EC2 instance using PuTTY.
2. Install the AWS CLI:
   * For **Amazon Linux 2**:

bash

Copy code

sudo yum install aws-cli -y

* + For **Ubuntu**:

bash

Copy code

sudo apt update

sudo apt install awscli -y

##### **Test S3 Access from EC2 Instance**

1. In the EC2 instance, run:

bash

Copy code

aws s3 ls s3://my-app-bucket # Replace with your S3 bucket name

You should see a list of objects in the bucket if permissions are correct.

### ****Step 3: RDS Instance Creation with Private Access****

#### ****3.1 Launch an RDS Instance****

1. Go to the [RDS Console](https://console.aws.amazon.com/rds/home).
2. Click **Create Database**, choose:
   * **Database Creation Method**: Standard.
   * **Engine**: MySQL or PostgreSQL.
   * **Instance Class**: Use db.t3.micro for small workloads.
   * **Storage**: Adjust as needed (e.g., 20 GB).
   * **DB Instance Identifier**: Name it (e.g., mydb-instance).
   * **Master Username/Password**: Create a user and strong password.
3. **Network Settings**:
   * **VPC**: Choose your VPC.
   * **Public Access**: Set this to **No** (for private access).
4. Click **Create Database** and wait for the instance to launch.

#### ****3.2 Connect EC2 to RDS****

1. **Modify the EC2 Security Group**:
   * Go to **EC2 Dashboard** → **Security Groups**.
   * Edit the inbound rules for your EC2 security group to allow traffic on port 3306 (MySQL) or 5432 (PostgreSQL) from the RDS security group.
2. **Install MySQL/PostgreSQL Client** in EC2:
   * For MySQL:

bash

Copy code

sudo yum install mysql -y # Amazon Linux

sudo apt install mysql-client -y # Ubuntu

* + For PostgreSQL:

bash

Copy code

sudo yum install postgresql -y # Amazon Linux

sudo apt install postgresql-client -y # Ubuntu

1. **Connect to the RDS Instance** from EC2:
   * For MySQL:

bash

Copy code

mysql -h <rds-endpoint> -u <master-username> -p

* + For PostgreSQL:

bash

Copy code

psql -h <rds-endpoint> -U <master-username> -d <database-name>

### ****Step 4: Server Deployment Using EC2, S3, and RDS****

#### ****4.1 Use S3 for Static File Storage****

Your application can upload and retrieve files from S3 using the AWS SDK (e.g., aws-sdk for Node.js or PHP SDK).

1. **In Node.js**:

bash

Copy code

npm install aws-sdk

Example code to upload to S3:

javascript

Copy code

const AWS = require('aws-sdk');

const s3 = new AWS.S3();

const params = {

Bucket: 'my-app-bucket',

Key: 'file.txt',

Body: 'Hello World'

};

s3.upload(params, (err, data) => {

if (err) {

console.log("Error", err);

} else {

console.log("Upload Success", data.Location);

}

});

#### ****4.2 Configure Your Application to Use RDS****

If you’re deploying a web app, configure it to connect to the RDS instance.

Example for **Node.js** using **MySQL**:

1. Install the MySQL client for Node.js:

npm install mysql

1. In your Node.js application, configure the database connection like this:

const mysql = require('mysql');

const connection = mysql.createConnection({

host: 'your-rds-endpoint', // Replace with your RDS instance endpoint

user: 'your-username', // Replace with your RDS master username

password: 'your-password', // Replace with your RDS master password

database: 'your-database' // Replace with your RDS database name

});

connection.connect((err) => {

if (err) throw err;

console.log('Connected to RDS MySQL!');

});

This will connect your application to the RDS instance, allowing it to store and retrieve data from the database.

### ****Summary of Key Steps:****

1. **Launch EC2 instance** and attach an EBS volume.
2. **Create an S3 bucket** and upload files for object storage.
3. **Create an RDS instance** with private access for database services.
4. **Deploy your server** on EC2 and configure it to use S3 for file storage and RDS for the database backend.