1. Hosting Static Websites on AWS S3 and EC2

EC2: Apache:

- EC2-> Create instance-> name-> Ubuntu-> t3.micro-> key pair-> ssh, http, https
- Launch instance
- EC2 -> Select this instance-> Connect-> Public ip-> connect
- sudo apt update -y
- sudo apt upgrade -y
- sudo apt install apache2 -y
- sudo systemctl start apache2
- sudo systemctl enable apache2
- cd /var/www/html
- sudo rm index.html
- sudo nano index.html
- Paste this code in nano editor:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My EC2 Website</title>
  <style>
    body {
      background: linear-gradient(to right, #74ebd5, #ACB6E5);
      font-family: Arial, sans-serif;
      text-align: center;
      padding: 50px;
    }
    h1 {
      color: navy;
      font-size: 3em;
    }
    p {
      color: #333;
      font-size: 1.2em;
    }
    .box {
      background: white;
      border-radius: 10px;
      padding: 20px;
      display: inline-block;
      box-shadow: 0 4px 8px rgba(0,0,0,0.2);
    }
  </style>
</head>
<body>
  <div class="box">
```

```
        <h1>Welcome to Riya’s Website on EC2!</h1>
        <p>This is a static site hosted with Apache on an Ubuntu server.</p>
    </div>
</body>
</html>
```

- CTRL + O -> Enter -> CTRL + X (exit nano editor)
- sudo chown www-data:www-data /var/www/html/index.html
- Copy public ip and paste in browser: http://<your-public-ip>
- Done

S3:

- S3 -> Create bucket -> name: awsweb-riya
- All default -> uncheck block -> I acknowledge-> create bucket
- S3 bucket-> Properties -> buck versioning Edit-> enable -> save changes
- Objects -> upload -> Add 4 website files -> upload
- Properties -> Static website hosting -> Enable + index.html, error.html -> save
- Permissions -> Bucket policy-> Edit-> code with correct arn:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadForWebsite",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::mys3bkr/*"
        }
    ]
}
```

- Properties-> static website hosting-> URL/ endpoint in browser
- Done

2. EC2 Setup and MySQL Database Management, including Database with Triggers and Stored Procedures

- Make an EC2 instance with Ubuntu & Connect to the EC2 instance via EC2 console
- sudo apt update -y
- sudo apt install mysql-server -y
- sudo systemctl start mysql
- sudo systemctl enable mysql
- sudo mysql
- Now mysql code: create and use DB, create table:

```
CREATE DATABASE companyDB;
USE companyDB;
CREATE TABLE Employees (
   EmpID INT AUTO_INCREMENT PRIMARY KEY,
   Name VARCHAR(50),
   Department VARCHAR(50),
   Salary DECIMAL(10,2)
);
```

- <u>Insert Sample data</u>:

```
INSERT INTO Employees (Name, Department, Salary) VALUES
('Alice', 'HR', 50000),
('Bob', 'IT', 60000),
('Charlie', 'Finance', 70000);
SELECT * FROM Employees;
```

- <u>Create a Trigger</u>:

```
CREATE TABLE SalaryLog (
   LogID INT AUTO_INCREMENT PRIMARY KEY,
   EmpID INT,
   OldSalary DECIMAL(10,2),
   NewSalary DECIMAL(10,2),
   ChangeDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

DELIMITER //
CREATE TRIGGER before_salary_update
BEFORE UPDATE ON Employees
FOR EACH ROW
BEGIN
   IF OLD.Salary <> NEW.Salary THEN
      INSERT INTO SalaryLog (EmpID, OldSalary, NewSalary)
      VALUES (OLD.EmpID, OLD.Salary, NEW.Salary);
   END IF;
END//
DELIMITER ;
```

- Test it:

```
UPDATE Employees SET Salary = 65000 WHERE Name='Bob';
SELECT * FROM SalaryLog;
```

- Create a Stored Procedure:

```
DELIMITER //
CREATE PROCEDURE IncreaseSalaryByDept(IN deptName VARCHAR(50), IN
percentIncrease DECIMAL(5,2))
BEGIN
   UPDATE Employees
   SET Salary = Salary + (Salary * percentIncrease / 100)
   WHERE Department = deptName;
END//
DELIMITER ;
```

- Call the procedure:

```
CALL IncreaseSalaryByDept('IT', 10);
SELECT * FROM Employees;
```

- EXIT;

3. Web Application Deployment using AWS Elastic Beanstalk

- Elastic Beanstalk -> Create application/environment
- Web server environment -> name app -> name env -> Platform: Python
- Upload your code-> local-> myapp.zip (zip inside folder) -> version label-> next
- Configuration service access-> create Service & EC2 roles if needed -> next
- Next next next till Review -> create -> time
- Click URL in Domain -> Done

4. Serverless Computing – S3 and Lambda Integration

- S3 -> Create bucket -> name -> block all -> all default -> create bucket -> view details
- Duplicate tab: lambda
- Create function-> mys3lm -> python 3.13 -> x86_64
- Change default execution role -> create new role AWS Policy-> role name: mylmrole
- Policy templates: s3 -> s3 object read only permissions -> Create function
- Code editor: paste this code:

```
import json
import urllib.parse
import boto3
print('Loading function')
s3 = boto3.client('s3')

def lambda_handler(event, context):
    # print("Received event: " + json.dumps(event, indent=2))
    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')

    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in
the same region as this function.'.format(key, bucket))
        raise e
```

- Deploy
- Add Trigger -> trigger config -> S3 -> mys3lm bucket -> all default -> I acknow -> Add
- Go to S3 tab
- Properties -> Event notifications -> lambda function link -> lambda tab opens
- In lambda tab -> Configuration -> Triggers -> S3 trigger visible
- Configuration -> Permissions -> Resource-based policy statements -> lambda link
- Policy statement details open, check: effect:allow, action: invoke, stringequals & arn
- Go to S3 tab
- Objects -> upload (any file) -> select file -> upload (will trigger lm func to run)
- Go to lambda tab
- Monitor -> many graphs
- View CloudWatch logs -> cloudwatch tab opens
- Logs -> log groups (side pane)
- Log streams -> click on the latest one -> Log events show up
- Done -> Delete lambda function and S3 bucket

5. EC2 Auto Scaling using Launch Templates and Scaling Policies

- Go to EC2
- Note region
- Launch templates in left pane
- Create launch template
- Mytemp name
- Temp description
- Auto Scaling guidance: Select Provide guidance.
- AMI -> Quick Start -> Amazon Linux -> x86_64
- T2.micro
- No key pair
- Subnet: Choose Don't include in launch template
- Skip Security groups
- Advanced network configuration-> Add network interface:
- Auto-assign public IP: Choose Enable
- Security groups: default one
- Delete on termination: Choose Yes
- Create launch template
- View launch templates
- Click on your template link
- Actions -> Create Auto Scaling group
- Auto Scaling group name: myasg
- Launch template: mytemp
- Version: Choose Latest (1)
- Choose Next
- VPC: default one
- Subnets: us-east-1a
- Choose Skip to review
- Create Auto Scaling group
- Go to EC2 Dashboard
- Choose Instances (running)
- Done

6. S3 Bucket File Management and Public Access Configuration

- Create S3 bucket
- Upload a file
- Bucket -> File-> Object URL -> Browser -> public access denied
- Bucket-> Permissions tab -> Edit bucket policy

Code with correct arn:

```
{

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

  "Id": "Policy1",

  "Statement": [

    {

      "Sid": "Stmt1",

      "Effect": "Allow",

      "Principal": "*",

      "Action": "s3:GetObject",

      "Resource": "arn:aws:s3:::rs3bk/*"

    }

      ]

}
```

- Try the object URL again in browser
- Publicly accessible -> Done

Option 2: Enable Public Access via Object Settings

1. Click the file you want to make public.
2. Go to the **Object actions → Make public** option.
3. Confirm the warning about public access.