**Project 2: Cloud Automation using Python (with Boto3)**

**✅ Project Objective:**

To automate various cloud infrastructure operations (like **launching EC2**, **managing S3**, **monitoring with CloudWatch**) using Python and AWS SDK (**Boto3**), making infrastructure management programmable and efficient.

**🧰 Tools & Technologies Used:**

* **Python**
* **Boto3** (AWS SDK for Python)
* **AWS IAM Role/Access Key**
* **Cloud Services:** EC2, S3, CloudWatch, Lambda (optional)

**⚙️ Key Automation Tasks Covered:**

1. **Launching an EC2 Instance:**
   * Use Boto3 to launch a virtual machine.
   * Choose AMI, instance type, key pair, security group.
2. **Creating and Managing S3 Buckets:**
   * Automate S3 bucket creation.
   * Upload/download files.
   * Set object lifecycle policies.
3. **Reading Logs from CloudWatch:**
   * Connect to CloudWatch using Boto3.
   * Retrieve logs of EC2 or Lambda for monitoring.
4. **Auto-Terminate EC2 Instances:**
   * Write logic to find running instances and shut them down based on time or cost.

**🔄 Example Python Snippets:**

**▶️ Launch EC2:**

python

Copy code

import boto3

ec2 = boto3.resource('ec2')

instance = ec2.create\_instances(

ImageId='ami-0c02fb55956c7d316', # Amazon Linux

InstanceType='t2.micro',

MinCount=1,

MaxCount=1,

KeyName='your-key-name'

)

print("Launched instance:", instance[0].id)

**▶️ Read Logs from CloudWatch:**

python

Copy code

import boto3

client = boto3.client('logs')

response = client.get\_log\_events(

logGroupName='your-log-group',

logStreamName='your-log-stream',

startFromHead=True

)

for event in response['events']:

print(event['message'])

**💡 Use Cases:**

* Developers automating test environments
* Auto-scaling or cleaning unused resources
* Scheduled data uploads or backups

**✅ Outcome/Benefits:**

* Complete DevOps-ready Python scripts for AWS
* Saves cloud costs by auto-managing resources
* Strong real-world automation project to showcase