SECTION 1 - Do Every Task in the section.

***1.*. Lambda Function Triggered by S3 Events**

Task: Write a Lambda function (using Python or Node.js) that is triggered by S3 “object created” events. The function should log key metadata of the new object.

Deliverable: Share your Lambda function code along with evidence (e.g., sample logs or test results) showing that the function was triggered.

2. AWS Budgets Alarm Setup

Task: Set up an AWS Budgets alarm using the AWS Console or CLI to notify you when your estimated monthly cost exceeds a set threshold (e.g., $50).

Deliverable: Provide a screenshot or CLI output confirming the budget alarm configuration along with a brief explanation of your setup.

3 .TechBlogs Inc., a new content platform, needs to host their static website and ensure basic security/compliance. As their cloud engineer, you must:

Assignment: Basic Web Hosting & Security

Task 1: Static Website Hosting

- Create an S3 bucket named "techblogs-website-[your-initials]"
- Enable static website hosting with index.html
- Upload a sample HTML file (can be 1 line: "<h1>Coming Soon</h1>")

Task 2: Secure Content Delivery

- Create a CloudFront distribution for the S3 bucket
- Force HTTPS-only connections
- Restrict S3 bucket access to only CloudFront (via OAI)

Task 3: Monitoring & Access

- Launch a t2.micro EC2 instance (Amazon Linux)
- Configure a Security Group allowing SSH access **only from your current IP**
- Enable detailed monitoring on the EC2 instance

Task 4: Cost Awareness

- Set a $10/month budget alarm named "BlogHostingBudget"

Deliverables

1. **Technical Outputs**:
    - S3 bucket URL (e.g., http://techblogs-website-abc.s3-website-us-east-1.amazonaws.com)
    - CloudFront distribution domain (e.g., d123.cloudfront.net)
    - EC2 Instance ID (e.g., i-0123456789abcdef0)
    - Budget alarm ARN (via aws budgets describe-budget --budget-name BlogHostingBudget)
2. **Brief Reflection (50-100 words)**:
    - What was the hardest part?
    - How did you verify S3 bucket was only accessible via CloudFront?
    - If you had 2 more hours, what improvement would you prioritize?

SECTION 2 : Complete any 1 of below 2 assignments:

1.A healthcare startup needs to migrate their patient record system to AWS while maintaining HIPAA compliance. You're tasked with building a secure, scalable foundation.

Assignment: Secure Multi-Tier Architecture Deployment

Task Requirements

1. **Infrastructure Setup**
    - Deploy a VPC with:
        - 1 public subnet (web tier)
        - 1 private subnet (database tier)
        - NAT Gateway for private subnet internet access
    - Launch an EC2 instance in the public subnet (Amazon Linux 2023 AMI)
    - Configure an RDS MySQL instance in the private subnet
2. **Security Implementation**
    - Create IAM roles/policies restricting RDS access to only the EC2 instance
    - Encrypt RDS storage using AWS KMS
    - Set up Security Groups to allow:
        - HTTPS (443) to EC2 from anywhere
        - MySQL (3306) only from EC2's private IP

3. **Monitoring & Cost Control**
    - Create CloudWatch alarms for:
        - RDS CPU utilization > 70%
        - EC2 network out > 5 MB/sec
    - Configure AWS Budgets to alert when monthly costs exceed $300

Deliverables

- Terraform code for infrastructure (VPC, EC2, RDS)
- CLI outputs showing:

```
aws rds describe-db-instances --db-instance-identifier patient-db
aws cloudwatch describe-alarms --alarm-names "RDS-HighCPU"
```

- 1-page explanation of:
    - Any configuration challenges faced (e.g., NAT Gateway routing)
    - Cost optimization measures implemented

---

2. StreamFlow, a video streaming startup, needs a scalable and fault-tolerant infrastructure to host their web app. You must deploy a secure, high-availability environment using EC2.

Assignment: Multi-AZ Web Server Deployment

Infrastructure Requirements

1. **VPC & Networking**
    - Deploy a VPC (10.0.0.0/16) with:
        - 2 public subnets (web tier) in different AZs (e.g., us-east-1a, us-east-1b)
        - 1 private subnet (database tier) in us-east-1c
    - Configure a NAT Gateway for private subnet outbound traffic.
2. **Compute & Scaling**
    - Launch an **EC2 Auto Scaling Group** in public subnets:
        - AMI: *Amazon Linux 2023*
        - Instance Type: t3.large

        - Min/Max Size: 2/4 instances
    - Attach an **Application Load Balancer (ALB)** to distribute traffic.
3. **Database & Storage**
    - Deploy an **RDS MySQL** instance in the private subnet.
    - Attach a **100 GB EBS volume** (gp3) to each EC2 instance for media caching.

Technical Tasks

1. **Terraform Setup**
    - Write Terraform code to provision:
        - VPC, subnets, NAT Gateway
        - ALB + Auto Scaling Group with health checks
        - RDS MySQL with daily automated backups
2. **Security Hardening**
    - Create IAM roles for EC2 instances with read-only S3 access (for media files).
    - Configure Security Groups:
        - ALB: Allow HTTP/80 and HTTPS/443 from 0.0.0.0/0
        - EC2: Allow traffic **only from the ALB** on port 8080
        - RDS: Allow MySQL/3306 **only from EC2 security group**
3. **Monitoring & Automation**
    - Create CloudWatch alarms for:
        - ALB HTTP_5XX_Count > 10 in 5 minutes
        - EC2 average CPU > 75%
    - Use **AWS Systems Manager (SSM)** to automate NGINX installation via a user-data script.
4. **Disaster Recovery**
    - Write a script to create AMIs of the EC2 instances weekly.
    - Configure S3 bucket lifecycle policies to transition media files to Glacier after 90 days.

Deliverables

1. **Code & Outputs**
    - Terraform files for VPC, ALB, Auto Scaling, and RDS
    - User-data script for EC2 (NGINX setup)
    - CLI outputs:

```
# Verify Auto Scaling group
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name
StreamFlow-ASG

# Check ALB health
aws elbv2 describe-target-health --target-group-arn <your-tg-arn>
```

2. **1-Page Summary**
    - How you ensured high availability (e.g., multi-AZ subnets, ALB health checks)
    - Challenges faced (e.g., debugging Security Group dependencies)
    - One cost optimization implemented (e.g., Auto Scaling termination policies)