**Cloud Engineering Assignment: Real-World Scenario**

**Objective:** The goal of this assignment is to simulate tasks that a cloud engineer would typically perform in a real-world environment. You'll be working with cloud resources like compute instances, networking, storage, and security practices, as well as ensuring scalability, reliability, and security for an application.  
Link to the web application: https://github.com/CarineRam/User-Profile-Application

**Assignment Overview:**

You are tasked with setting up a cloud environment for a fictitious e-commerce company, **TechStore**, that plans to launch a web application to handle user traffic. Your objective is to build a scalable, secure, and highly available cloud infrastructure.

**Part 1: Cloud Infrastructure Setup**

1. **Create a Virtual Private Cloud (VPC):**
   * Design a VPC with private and public subnets for different layers of the application (e.g., web layer, database layer).
   * Configure route tables and a NAT Gateway for internet access in private subnets.
   * Set up security groups to control inbound and outbound traffic.
2. **Set up Compute Resources:**
   * Launch EC2 instances (or equivalent in another cloud provider) in the public subnets for web servers (Nginx, Apache).
   * Ensure auto-scaling for the web server instances to handle traffic spikes.
   * Set up a Load Balancer (e.g., AWS ELB, Azure Load Balancer) to distribute traffic across the web servers.
   * Implement health checks for the instances to ensure availability.
3. **Storage Setup:**
   * Create and configure S3 (AWS), Blob Storage (Azure), or Cloud Storage (GCP) to store user-uploaded files (product images, etc.).
   * Enable object versioning and lifecycle policies to manage data retention and delete obsolete files.
4. **Database Setup:**
   * Set up a managed database service like Amazon RDS, Azure SQL Database, or Cloud SQL (GCP).
   * Implement database backups and multi-AZ deployment for high availability and disaster recovery.
   * Enable automated scaling of the database instance to handle varying loads.
5. **Networking and Security:**
   * Set up a Virtual Private Network (VPN) for secure access between your local machine and the cloud environment.
   * Implement Identity and Access Management (IAM) roles to restrict access based on the principle of least privilege. Use roles for EC2, RDS, and other services.
   * Set up an AWS WAF, Azure Web Application Firewall, or equivalent to protect the application from common attacks (SQL injection, XSS).
   * Enable CloudWatch (AWS), Azure Monitor, or Stackdriver (GCP) for logging and monitoring the application and resources.

**Part 2: Automation and CI/CD Pipeline**

1. **Automate Infrastructure Deployment:**
   * Use **Infrastructure as Code (IaC)** tools like Terraform or AWS CloudFormation to define your infrastructure.
   * Automate the creation of VPC, subnets, EC2 instances, Load Balancer, database, and S3 buckets.
2. **Create CI/CD Pipeline:**
   * Set up a Jenkins pipeline or GitLab CI/CD pipeline to automatically deploy code to the EC2 instances.
   * Integrate with version control (GitHub, GitLab) to trigger builds and deployments upon new code commits.
   * Ensure that every deployment triggers tests (unit tests, integration tests) before it is pushed to production.
3. **Monitoring and Alerts:**
   * Configure CloudWatch Alarms (or the equivalent in other providers) to trigger notifications when resource limits (CPU, memory, disk space) are exceeded.
   * Set up automated scaling policies for your EC2 instances to scale up or down based on traffic.
   * Enable application-level monitoring with a service like AWS X-Ray, Datadog, or Azure Monitor.

**Part 3: Security and Cost Optimization**

1. **Implement Security Best Practices:**
   * Set up **AWS IAM** roles with permissions for each service based on least privilege.
   * Use AWS KMS (or similar services) to manage encryption keys for sensitive data stored in S3 and RDS.
   * Configure VPC Peering or Transit Gateway for secure connections between VPCs.
   * Use CloudTrail (AWS), Activity Logs (Azure), or Cloud Audit Logs (GCP) to track and log user actions and API calls.
2. **Cost Management and Optimization:**
   * Set up AWS Cost Explorer, Azure Cost Management, or Google Cloud’s billing dashboard to monitor spending.
   * Identify and eliminate any unused or underused resources, such as orphaned Elastic IPs, unattached volumes, or unnecessary EC2 instances.
   * Implement Reserved Instances for long-running resources like databases and web servers to save on costs.

**Part 4: Documentation**

1. **Write a comprehensive report** detailing:  
     
   * The design of your cloud infrastructure.
   * How security and cost optimization best practices were implemented.
   * A step-by-step guide on how to deploy and configure each service.
   * Challenges faced during the implementation and how they were resolved.
   * Future improvements or optimizations that could be made.

**Submission Instructions:**

* Submit your Terraform/CloudFormation scripts and configuration files.
* Share your Jenkins pipeline configuration file or GitLab CI configuration.
* Include a link to a Git repository that contains the scripts for automation.
* Provide access to a shared document (Google Docs, Confluence, or Word document) for your report and any diagrams you created for the infrastructure.