CLOUDZENIA INTERVIEW: HANDS-ON

Objective:

This task assesses your skills in **Infrastructure as Code (IaC)** using **Terraform** for cloud resource provisioning and your ability to configure and manage cloud resources post-deployment. You will create two different infrastructures on AWS, document your approach, and provide the running endpoints. This hands-on challenge will test your understanding of cloud infrastructure, automation, deployment best practices, and web server management.

Challenges:

1. **ECS with ALB, RDS and SecretsManager**
    a. **Infrastructure Requirements:**
        i. **ECS**
            1. Create an **ECS Cluster** with the **service** running in **Private Subnets**.
            2. **Services**
                a. WordPress Docker Image
                b. **Custom microservice** (a lightweight Node.js application responding with “Hello from Microservice”). *Share the Node.js code and DockerFile too.*
            3. Setup auto scaling based on CPU and Memory.
        ii. **RDS**
            1. Choose the **appropriate instance type** to be used by WordPress as a database.
            2. Create **custom user and password** in RDS to be used with Wordpress that do not auto rotate.
            3. Configure **automated backups.**
            4. **RDS instance** should be deployed in the **Private Subnets**.
        iii. **SecretsManager**

1. Store the **RDS database credentials** as **secrets**.

iv. **IAM and Security**

1. Configure the **ECS task definition** to use the secrets stored in **AWS SecretsManager**, ensuring the WordPress securely connects to the RDS instance.
2. Use **IAM roles** to grant the ECS service the necessary permissions to access secrets from AWS SecretsManager.
3. Security Groups should be the least privilege.

v. **Application Load Balancer (ALB) & Domain Mapping**

1. Set up an **Application Load Balancer (ALB)** in **Public Subnets** to handle incoming HTTP/HTTPS traffic.
2. Set up SSL certificate. The website should not open HTTP.
3. HTTP traffic should be redirected to HTTPS.
4. Configure the ALB to associate with a domain name.
   "wordpress.<domain-name>"
   "microservice.<domain-name>"

2. EC2 Instance with Domain Mapping and NGINX

a. Infrastructure Requirements:

i. EC2 Instance:

1. Deploy 2 **EC2 Instance** in a **Private Subnet**.
2. Attach an **Elastic IP** to the instance to ensure it has a static public IP address for domain mapping.

ii. Domain Mapping:

1. Associate with a domain name.
   "**ec2-docker1.<domain-name>**" and
   "**ec2-instance1.<domain-name>**"
   "**ec2-docker2.<domain-name>**" and
   "**ec2-instance2.<domain-name>**"

iii. Application Load Balancer (ALB) & Domain Mapping

1. Set up an **Application Load Balancer (ALB)** in **Public Subnets** to handle incoming HTTP/HTTPS traffic.
2. Set up SSL certificate. The website should not open HTTP.
3. HTTP traffic should be redirected to HTTPS.

4. Configure the ALB to associate with a domain name.
   "ec2-alb-docker.<domain-name>"
   "ec2-alb-instance.<domain-name>"

iv. Docker

1. Install **Docker**
2. Run a Docker container that responds with "**Namaste from Container**" on an **internal port** (e.g., **8080**)

v. NGINX Configuration:

1. Install **NGINX**
2. Domain-based Content Serving:
   a. **"ec2-instance.<domain-name>"** - Configure NGINX to serve the text "**Hello from Instance**"
   b. **"ec2-docker.<domain-name>"** - Set up NGINX to forward requests to a Docker container running on the same instance, which serves the text "**Namaste from Container**".

vi. SSL/TLS with Let's Encrypt:

1. Set up **Let's Encrypt** to obtain the subdomain's **SSL certificate**.
2. Configure **NGINX** to use this certificate for **HTTPS** access. Ensure that all **HTTP traffic is redirected to HTTPS** for the subdomain.

3. Observability

a. EC2

i. Configure metrics on **EC2 Instance** so **RAM utilization** can be seen in **CloudWatch**.
ii. Configure logs on **EC2 Instance** so we can see **NGINX access logs** on **CloudWatch**.

4. GitHub Actions

a. **Custom Microservice** should be stored in a GitHub Repository.

   b. **GitHub Actions** should build an Docker Image, Push to ECS and Deploy to ECR.

5. **S3 Static Website Hosting with CDN (**Optional)
   a. Host a static website on an S3 bucket with the URL "**static-s3.<domain-name>"**.
   b. Configure a **CloudFront Distribution** to serve the static website with low latency and caching.
   c. Enable **geo-restriction** to block access from specific countries.
   d. Use **Lambda@Edge** to modify HTTP headers for SEO purposes. (Optional)

Submission:

1. **Code**:
   a. Submit all Terraform scripts and relevant configuration files used for the deployments.
   b. Create reusable terraform modules.
2. Submit the GitHub Actions Workflow file and grant access to the GitHub repository, or make the repository public.
   a. Include a comprehensive document (.md or .pdf) detailing the infrastructure setup, configurations, and endpoint URLs.
3. **Running Endpoint:**
   a. Ensure the setup is live, accessible, and working as expected for testing.
4. **Video Demostration (Optional)**
   a. Provide a video demonstrating the entire challenge in less than 3 minutes.

Deadline:

1. **AWS Account:**
   a. You have 48 hours to complete this task from the moment you start.

2. **Post-Submission Availability:**
    a. Ensure that your deployed endpoints remain accessible for at least 48 hours after submission for evaluation.

Important Notes:

1. **AWS Account:**
    a. You will have to perform this task in your own AWS account.
2. **Leverage Free Tier:**
    a. Use free-tier resources to minimize costs. This task does not require high-performance or high-cost resources.
3. **Cost Management:**
    a. CloudZenia will not be responsible for any costs incurred in your AWS account during the completion of this task. Monitor your spending to avoid unexpected charges.
4. **Resource Cleanup:**
    a. Ensure you clean up all resources created once the task is completed and verified. This will help you avoid unnecessary charges.
5. **Domain Management:**
    a. If you don’t have an existing domain, use a free subdomain from a dynamic DNS provider.
6. **Support:**
    a. If you encounter issues related to this task, refer to AWS documentation or community forums. Direct support from CloudZenia will not be provided for this task.