Creating a Scalable and Highly Available Environment for the Café

A business request for the café: Implementing a scalable and highly available environment

Step 1: Inspecting the environment

In this task, you will evaluate the current state of the environment.

1. Explore the environment, including how the network is set up.

Step 2: Creating a NAT gateway for the second Availability Zone

2. Create a NAT gateway in the *Public Subnet* in the second Availability Zone.
3. Configure the network to send internet-bound traffic from instances in *Private Subnet 2* to the NAT gateway you just created.

Step 3: Creating a bastion host instance in a public subnet

4. From the **Amazon EC2 console**, create an EC2 instance in one of the public subnets of the *Lab VPC*. It must meet the following criteria:
   - Name**: Bastion Host**
   - **Amazon Machine Image (AMI)**: *Amazon Linux 2023 AMI*
   - **Instance type**: *t2.micro*
   - Uses the **vockey** key pair
   - **Auto-assign Public IP**: This setting should be enabled
   - Only allows the following traffic:
     - **Type**: *SSH*
     - **Port**: 22
     - **Source**: Your IP address

Step 4: Creating a launch template

5. Create a launch template by using the AMI. It must meet the following criteria.
   - **AMI**: Cafe WebServer Image
   - **Instance type**: *t2.micro*

Key pair (login): Uses a *new key pair*

   - **Security groups**: CafeSG
   - **Resource tags**:
     - **Key**: Name
     - **Value**: webserver
     - **Resource types**: *Instances*
   - **IAM Instance Profile**: CafeRole

Step 5: Creating an Auto Scaling group

6. Create a new Auto Scaling Group that meets the following criteria:
   - **Launch template**: Uses the launch template that you created in the previous task

   - **VPC**: Uses the VPC that was configured
   - **Subnets**: Uses Private Subnet 1 and Private Subnet 2
   - Skips *all* the advanced options
   - Has a **Group size** configured as:
     - **Desired capacity**: 2
     - **Minimum capacity**: 2
     - **Maximum capacity**: 6
   - Enables the **Target tracking scaling policy** configured as:
     - **Metric type**: *Average CPU utilization*
     - **Target Value**: 25
     - **Instances need**: 60

7. To verify that you created the Auto Scaling group correctly, go to the **Amazon EC2 console**. You should have two instances, both with the name that you configured as *resource tags* in the previous task.

Step 6: Creating a load balancer

8. Create an HTTP Application Load Balancer that meets the following criteria:
   - **VPC**: Uses the VPC configured
   - **Subnets**: Uses the two *public subnets*
   - Skips the HTTPS security configuration settings
   - **Security group**: Creates a *new security group* that allows HTTP traffic from anywhere
   - **Target group**: Creates a *new target group*
   - Skips registering targets
9. Modify the Auto Scaling group that you created in the previous task by adding this new load balancer.

Step 7: Testing the web application

10. To test the café web application, visit the Domain Name System (DNS) name of your load balancer and append /cafe to the URL.

The café application should load.

Step 8: Testing automatic scaling under load

In this task, you will test whether the café application *scales out* automatically.

11. By using *Secure Shell (SSH) passthrough through the bastion host instance*, use SSH to connect to one of the running web server instances.
12. From the web server instance, use the following commands to start a stress test. This test increases the load on the web server CPU:

```
sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
sudo yum install stress -y
stress --cpu 1 --timeout 600
```

13. Verify that the Auto Scaling group deploys new instances.
    - Continue to observe the Amazon EC2 console.
    - During the test, you should observe that more web server instances are deployed.