The journey of order in Italian Restaurant

## **Project** Scenario**:**

1. Network Setup:

- There are two VPCs:

- VPC 1: This contains a Public Subnet with an EC2 instance that collects web application traffic it express to order management.

- VPC 2: This has a Private Subnet with another EC2 instance dedicated to processing and storing traffic data in an RDS database express to prepare management.

- Both VPCs are connected through a VPC Peering Connection.

2. Traffic Collection and Storage:

- In VPC 1, the EC2 instance in the public subnet captures traffic data from the web application.

- This EC2 instance then uploads the collected traffic data to Amazon S3, which is located in the same region as VPC 1.

3. Data Processing with Lambda and SQS:

- Amazon S3 acts as the source for an AWS Lambda function. When new data is uploaded, it triggers the Lambda function.

- The Lambda function retrieves the data from the S3 bucket and sends it to an Amazon SQS queue.

- The SQS queue is set up to be accessed by the EC2 instance located in VPC 2's private subnet.

4. Traffic Data Handling in VPC 2:

- Through the VPC Peering Connection, the EC2 instance in VPC 2’s private subnet receives the traffic data from the SQS queue.

- This EC2 instance then processes the data and sends it to an RDS instance for storage, which is located on the same private subnet within VPC 2.

5. Final Storage:

- The RDS database in VPC 2 securely stores the traffic data, where it can be accessed and analyzed later and there are three levels in security on (subnet ) and (instance) and (data Monitoring)

**Steps for all project :**

1. **Network :**

Step 1: Create VPCs in Two Regions

1. Create First VPC :

- Go to VPC Dashboard, click "Create VPC."

- CIDR to \*10.0.0.0/16\*.

2. Create Second VPC :

- Create VPC2

- CIDR to \*10.1.0.0/16\*.

3. Create Internet Gateways:

- Create and attach IGW to each VPC, naming them appropriately (e.g., IGW-Region1).

Step 2: Create Public Subnets

1. Public Subnet in VPC 1:

- Name PublicSubnet-1, its CIDR to 10.0.1.0/24.

Step 3: Create Private Subnets

1. Private Subnet in VPC 2:

- Name PrivateSubnet-2 , CIDR 10.1.4.0/24.

Step 4: Create Public Route Tables

1. Public Route Table for VPC 1:

- Name PublicRouteTable-1 , associate with VPC-Region1, add route 0.0.0.0/0 to IGW-Region1

Step 5: Create Private Route Tables

1. Private Route Table for Region 2:

- Create PrivateRouteTable-2 for VPC-2

1. **Launching services:**

VPC Peering

1. Create Peering Connection between the two VPCs.

2. Accept Peering Request in the console.

3. Modify Route Tables for each VPC to route traffic through the peering connection.

EC2 Instances

1. Launch EC2 Instances:

- Use Amazon Linux 2, instance type in usage .

- Assign to VPC1 (public subnet) and VPC2 for the other.

- Configure security rules for HTTP, HTTPS, and SSH access.

2. Create AMI from this instance for Auto Scaling .

Auto Scaling

1. Create Launch Template :

- Based on the custom AMI and assign necessary subnets and security settings.

2. Create Auto Scaling Group:

- Use the launch template, specify VPC, subnets, and multiple availability zones.

- Set scaling limits: minimum, desired, and maximum instances.

3. Configure Scaling Policies:

- Set a target tracking policy based on CPU utilization.

- Optionally, add a step scaling policy for more control.

Load Balancer (ALB)

1. Create Application Load Balancer:

- Set it to \*Internet-facing\*, assign to the VPC and public subnets.

- Configure for HTTP/HTTPS and associate it with the EC2 instances.

S3 Bucket for Customer Data

1. Create S3 Bucket with versioning and server-side encryption enabled.

RDS for Orders

1. Launch RDS Instance:

- Use MySQL, set instance type, enable Multi-AZ, and place in a private subnet within the VPC.

CloudFront

1. Set up CloudFront Distribution using the S3 bucket as the origin, enabling SSL for security.

### **Lambda**

#### Step 1: Create a global Lambda Function

1. Navigate to AWS Lambda
2. Create the Lambda Function
   * **Function name**:
   * **Runtime**: python 3.11
3. Permissions
   * AmazonS3ReadOnlyAccess
   * AmazonSQSFullAccess
4. Create function
5. Add Code to the Lambda Function
6. **Trigger**: S3 PutObject event
7. **Destination**: SQS queue via asynchronous invocation.

### **SQS**

1. Navigate to SQS
2. Create a New Queue
   * standard type
3. Configure Queue Settings
   * Name:
   * Enable Server-Side Encrption : to protect data in transit and at rest

### **Route 53**

#### Step 1: Register a Domain

1. Navigate to Route 53 : Get started
2. Register a domain
   * Domain Name: Italianoresturant.com (check availability.)
   * Contact information and choosing duration

#### Step 2: Create a Hosted Zone

1. Under DNS **management** section, click on **Hosted Zones**.
2. Enter the domain name for which you want to create the hosted zone
3. Type : Public hosted zone

#### Step 3: Add DNS Records to the Hosted Zone

1. click on the **Create record** button .
2. **Record type**: A
3. **Value**: ip of Ec2
4. **TTL**: 300
5. **Security levels :**

1. Create Network ACLs

- Create NACLs for each VPC: Set up NACLs with rules to allow necessary traffic.

2. Set Up Security Groups

- For EC2 in Public Subnet : Create a Security Group allowing inbound access on ports 3000

- For EC2 in Private Subnet: Create a Security Group allowing access from port 5000.

3. Create IAM Users

- Create an IAM User with access to S3 and RDS.

- Create another IAM User with permissions for all services.

4. Monitoring and Security

- Enable AWS CloudTrail and AWS Config for activity tracking and security monitoring.

5. Encrypted data in reset at RDS

*Step 1: Create a KMS Key*

1. go to KMS.
2. Click Create key. Choose Symmetric and click Next.
3. Provide a key alias and description, then click Next.
4. Select IAM users/roles for key management permissions, then click Next.
5. Choose IAM users/roles for key usage permissions, then click Next.
6. Review the settings and click Finish to create the key.

*Step 2: Enable Encryption for RDS*

1. Go to RDS in the AWS Console and click Create database.
2. Choose a database engine and configure basic settings.
3. In Additional configuration, enable Encryption and select your KMS key from the dropdown.
4. Complete other settings and click Create database.
5. **Connect the services with each other and with Web APP :**

* Connect between EC2 and S3

Built in the code

* Connect between s3 and lambda and between lambda and SQS

Built in launching lambda

* Connect between SQS and EC2

Built in the code

* Connect between EC2 and RDS

### Step 1: Collect Traffic Data on EC2

1. **Install necessary software** on the EC2 instance to monitor and collect traffic. Tools like **tcpdump**, **ntop**, or **Wireshark** can help collect raw traffic data, while **nginx** or **Apache logs** can provide application-level traffic information.
2. **Write a script** to process and format the collected data.

### Step 2: Configure the RDS Database for Storing Traffic Data

1. **Set up the database schema**: In RDS database, create a table to store traffic data

**Step 3: Write Data to RDS from EC2**

1. **Install database client libraries**: On the EC2 instance, install the client libraries for the database engine you're using (e.g., mysql-connector-python for MySQL, psycopg2 for PostgreSQL).
2. **Script to insert data**: Write a script that connects to the RDS instance and inserts the collected traffic data. Here’s a Python example for MySQL:

import mysql.connector

from datetime import datetime

# Connect to the RDS database

connection = mysql.connector.connect(

host=" rds-endpoint",

user=" username",

password=" password",

database=" database-name"

)

# Close the connection

cursor.close()

connection.close()