**Job-Ready Project:**

**Kafka-Lambda Data Streaming**

***with***

**VPC and NAT Gateway in AWS**

**Overview**

This project demonstrates how to set up a scalable and resilient event-streaming architecture using MSK for message brokering, AWS Lambda for processing, and SQS for load balancing. The setup involves creating a VPC with public and private subnets, launching an MSK cluster in private subnets, and deploying a Lambda function to consume messages from Kafka topics.

**Project Steps**

1. **Create the VPC and Subnets**
2. **Set up Route Tables and a NAT Gateway**
3. **Launch the MSK Cluster in Private Subnets**
4. **Configure AWS Lambda to Process Kafka Messages**
5. **Add SQS for Load Balancing**

**Step 1: Create the VPC and Subnets**

1. **Create a VPC** with a CIDR block of 11.0.0.0/16.

aws ec2 create-vpc --cidr-block 11.0.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=Lambda-MSK-VPC}]'

1. **Create Subnets**: Create two public and two private subnets for high availability.

aws ec2 create-subnet --vpc-id <VPC\_ID> --cidr-block 11.0.1.0/24 --availability-zone us-east-1a --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-A}]'

aws ec2 create-subnet --vpc-id <VPC\_ID> --cidr-block 11.0.2.0/24 --availability-zone us-east-1b --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-B}]'

aws ec2 create-subnet --vpc-id <VPC\_ID> --cidr-block 11.0.3.0/24 --availability-zone us-east-1a --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-A}]'

aws ec2 create-subnet --vpc-id <VPC\_ID> --cidr-block 11.0.4.0/24 --availability-zone us-east-1b --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-B}]'

**Step 2: Set up Route Tables and NAT Gateway**

1. **Create and Attach an Internet Gateway**.

aws ec2 create-internet-gateway --tag-specifications 'ResourceType=internet-gateway,Tags=[{Key=Name,Value=Lambda-MSK-IGW}]'

aws ec2 attach-internet-gateway --vpc-id <VPC\_ID> --internet-gateway-id <IGW\_ID>

1. **Create Route Tables** for the public and private subnets.

aws ec2 create-route-table --vpc-id <VPC\_ID> --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Public-Route-Table}]'

aws ec2 create-route-table --vpc-id <VPC\_ID> --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Private-Route-Table}]'

1. **Add Routes**: Associate the public route table with the subnets and configure the route for the NAT gateway in the private route table.

aws ec2 create-route --route-table-id <Public-Route-Table\_ID> --destination-cidr-block 0.0.0.0/0 --gateway-id <IGW\_ID>

1. **Create a NAT Gateway** in one of the public subnets.

aws ec2 allocate-address

aws ec2 create-nat-gateway --subnet-id <Public-Subnet-A\_ID> --allocation-id <EIP\_ALLOCATION\_ID>

1. **Update the Route in the Private Route Table** to use the NAT Gateway.

aws ec2 create-route --route-table-id <Private-Route-Table\_ID> --destination-cidr-block 0.0.0.0/0 --nat-gateway-id <NAT\_Gateway\_ID>

**Step 3: Launch the MSK Cluster in Private Subnets**

1. **Create the MSK Cluster** in the private subnets.

aws kafka create-cluster --cluster-name Lambda-MSK-Cluster --kafka-version 2.8.0 \

--number-of-broker-nodes 2 \

--broker-node-group-info file://broker-node-group.json

**broker-node-group.json**:

{

"brokerAZDistribution": "DEFAULT",

"clientSubnets": ["<Private-Subnet-A\_ID>", "<Private-Subnet-B\_ID>"],

"instanceType": "kafka.m5.large",

"securityGroups": ["<SECURITY\_GROUP\_ID>"]

}

**Step 4: Configure AWS Lambda to Process Kafka Messages**

1. **Create an IAM Role** for Lambda with the necessary permissions for SQS, MSK, and VPC access.

aws iam create-role --role-name LambdaMSKRole --assume-role-policy-document file://trust-policy.json

**trust-policy.json**:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "lambda.amazonaws.com"

},

"Action": "sts:AssumeRole"

}

]

}

1. **Attach Policies** for VPC, SQS, and MSK access.

aws iam attach-role-policy --role-name LambdaMSKRole --policy-arn arn:aws:iam::aws:policy/AmazonMSKFullAccess

aws iam attach-role-policy --role-name LambdaMSKRole --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole

aws iam attach-role-policy --role-name LambdaMSKRole --policy-arn arn:aws:iam::aws:policy/AmazonSQSFullAccess

1. **Create the Lambda Function** to read from SQS and write to MSK.

aws lambda create-function --function-name MSKProducerLambda \

--runtime python3.8 --role <LambdaMSKRole\_ARN> \

--handler lambda\_function.lambda\_handler \

--zip-file fileb://producer\_lambda.zip \

--environment Variables="{MSK\_BROKER=<Broker\_URL>,TOPIC=<Topic\_Name>,SQS\_QUEUE\_URL=<SQS\_Queue\_URL>}" \

--vpc-config SubnetIds=<Private-Subnet-A\_ID>,<Private-Subnet-B\_ID>,SecurityGroupIds=<SECURITY\_GROUP\_ID>

**lambda\_function.py**:

import json

import boto3

from kafka import KafkaProducer

# Setup Kafka producer

producer = KafkaProducer(bootstrap\_servers=[os.environ['MSK\_BROKER']],

value\_serializer=lambda x: json.dumps(x).encode('utf-8'))

def lambda\_handler(event, context):

sqs\_client = boto3.client('sqs')

for record in event['Records']:

message = json.loads(record['body'])

producer.send(os.environ['TOPIC'], value=message)

producer.flush()

return {'statusCode': 200, 'body': json.dumps('Message processed')}

1. **Add a Layer to the Lambda Function**: This layer will contain the kafka-python package.

aws lambda publish-layer-version --layer-name KafkaPythonLayer --zip-file fileb://kafka-python.zip --compatible-runtimes python3.8

aws lambda update-function-configuration --function-name MSKProducerLambda --layers <LAYER\_ARN>

1. **Set Lambda Timeout to Handle Multiple Messages**.

aws lambda update-function-configuration --function-name MSKProducerLambda --timeout 120

**Step 5: Set Up SQS Queue for Load Balancing**

1. **Create an SQS Queue** to buffer requests.

aws sqs create-queue --queue-name MSKLoadBalancerQueue

1. **Trigger the Lambda Function from SQS**: Configure the Lambda to process messages from this SQS queue.

aws lambda create-event-source-mapping --function-name MSKProducerLambda --event-source-arn <SQS\_QUEUE\_ARN> --batch-size 10

This end-to-end setup provides a scalable streaming architecture with detailed commands and code. Adjust parameters like instance types and Kafka settings based on the environment and expected load. The solution can be expanded with additional error handling, monitoring, and alerting as needed.