Project- 2: Publishing Amazon SNS Messages Privately

Problem Statement: How to secure patient records online and send it privately to the intended party Topics: In this project, you will be working on a hospital project to send reports online and develop a platform so the patients can access the reports via mobile and push notifications. You will publish the report to an Amazon SNS keeping it secure and private. Your message will be hosted on an EC2 instance within your Amazon VPC. By publishing the messages privately, you can improve the message delivery and receipt through Amazon SNS.

Highlights: 1. AWSCloudFormation to create a VPC

2. Connect VPC with AWS SNS

3. Publish message privately with SNS

Solution:

To create a VPC using AWS CloudFormation, connect it with AWS SNS, and publish a message privately with SNS, you can use the following steps:

Step 1: Create a VPC

Here's the CloudFormation template to create a VPC:

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  MyVPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: '10.0.0.0/16'
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: MyVPC
  PublicSubnet:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref MyVPC
      CidrBlock: '10.0.1.0/24'
      MapPublicIpOnLaunch: true
      AvailabilityZone: !Select [ 0, !GetAZs !Ref "AWS::Region" ]
      Tags:
        - Key: Name
          Value: PublicSubnet
  InternetGateway:
    Type: 'AWS::EC2::InternetGateway'
  AttachGateway:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      VpcId: !Ref MyVPC
      InternetGatewayId: !Ref InternetGateway
  RouteTable:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref MyVPC
  PublicRoute:
    Type: 'AWS::EC2::Route'
    Properties:
      RouteTableId: !Ref RouteTable
      DestinationCidrBlock: '0.0.0.0/0'
      GatewayId: !Ref InternetGateway
  SubnetRouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref RouteTable
```

Step 2: Connect VPC with AWS SNS

Next, we set up an SNS topic and subscription:

```
Resources:
  MySNSTopic:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MySNSTopic'
  MySubscription:
    Type: 'AWS::SNS::Subscription'
    Properties:
      Protocol: email  # Choose the protocol (email, lambda, sqs, etc.)
      Endpoint: 'your-email@example.com'  # Replace with your email
      TopicArn: !Ref MySNSTopic
```

Step 3: Publish Message Privately with SNS

To publish a message, you can use AWS Lambda or other services. Here's how to use Lambda to publish a message:

```
Resources:
  # Lambda Execution Role
  LambdaExecutionRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: sts:AssumeRole
      Policies:
        - PolicyName: PublishSNSPolicy
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
```

```
        - Effect: Allow
         Action:
          - sns:Publish
         Resource: '*'

  # Lambda Function
  PublishMessageFunction:
   Type: 'AWS::Lambda::Function'
   Properties:
    Handler: index.handler
    Role: !GetAtt LambdaExecutionRole.Arn
    Code:
     ZipFile: |
      import boto3
      def handler(event, context):
        sns = boto3.client('sns')
        response = sns.publish(
          TopicArn='arn:aws:sns:region:account-id:MySNSTopic',
          Message='This is a test message',
          Subject='Test'
        )
        return response
    Runtime: python3.8
    Timeout: 30
```

Combine Everything

Combine the templates into a single CloudFormation template to create the entire setup:

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
 MyVPC:
  Type: 'AWS::EC2::VPC'
  Properties:
    CidrBlock: '10.0.0.0/16'
    EnableDnsSupport: true
    EnableDnsHostnames: true
    Tags:
      - Key: Name
        Value: MyVPC
PublicSubnet:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref MyVPC
    CidrBlock: '10.0.1.0/24'
    MapPublicIpOnLaunch: true
InternetGateway:
  Type: 'AWS::EC2::InternetGateway'
AttachGateway:
  Type: 'AWS::EC2::VPCGatewayAttachment'
  Properties:
    VpcId: !Ref MyVPC
    InternetGatewayId: !Ref InternetGateway
RouteTable:
  Type: 'AWS::EC2::RouteTable'
  Properties:
    VpcId: !Ref MyVPC
PublicRoute:
  Type: 'AWS::EC2::Route'
  Properties:
    RouteTableId: !Ref RouteTable
    DestinationCidrBlock: '0.0.0.0/0'
    GatewayId: !Ref InternetGateway
SubnetRouteTableAssociation:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    SubnetId: !Ref PublicSubnet
    RouteTableId: !Ref RouteTable

# SNS Topic and Subscription
MySNSTopic:
  Type: 'AWS::SNS::Topic'
  Properties:
    TopicName: 'MySNSTopic'
MySubscription:
  Type: 'AWS::SNS::Subscription'
  Properties:
    Protocol: email
    Endpoint: 'your-email@example.com'
    TopicArn: !Ref MySNSTopic

# Lambda Execution Role
LambdaExecutionRole:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: lambda.amazonaws.com
          Action: sts:AssumeRole
    Policies:
      - PolicyName: PublishSNSPolicy
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Effect: Allow
              Action:
                - sns:Publish
              Resource: '*'

# Lambda Function to Publish Message
PublishMessageFunction:
  Type: 'AWS::Lambda::Function'
  Properties:
    Handler: index.handler
    Role: !GetAtt LambdaExecutionRole.Arn
```

```
   Code:
     ZipFile: |
       import boto3
       def handler(event, context):
           sns = boto3.client('sns')
           response = sns.publish(
               TopicArn='arn:aws:sns:region:account-id:MySNSTopic',
               Message='This is a test message',
               Subject='Test'
           )
           return response
   Runtime: python3.8
   Timeout: 30
```

Note: Replace `'region'` with your AWS region and `'account-id'` with your AWS account ID.

Deploy the Stack

1. Save the template into a file, e.g., `template.yaml`.
2. Use AWS CLI or the AWS Management Console to create a CloudFormation stack with this template.

Using AWS CLI:

```
aws cloudformation create-stack --stack-name my-stack --template-body file://template.yaml
--capabilities CAPABILITY_IAM
```

This setup will create a VPC, an SNS topic, and a Lambda function to publish messages to the SNS topic