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:

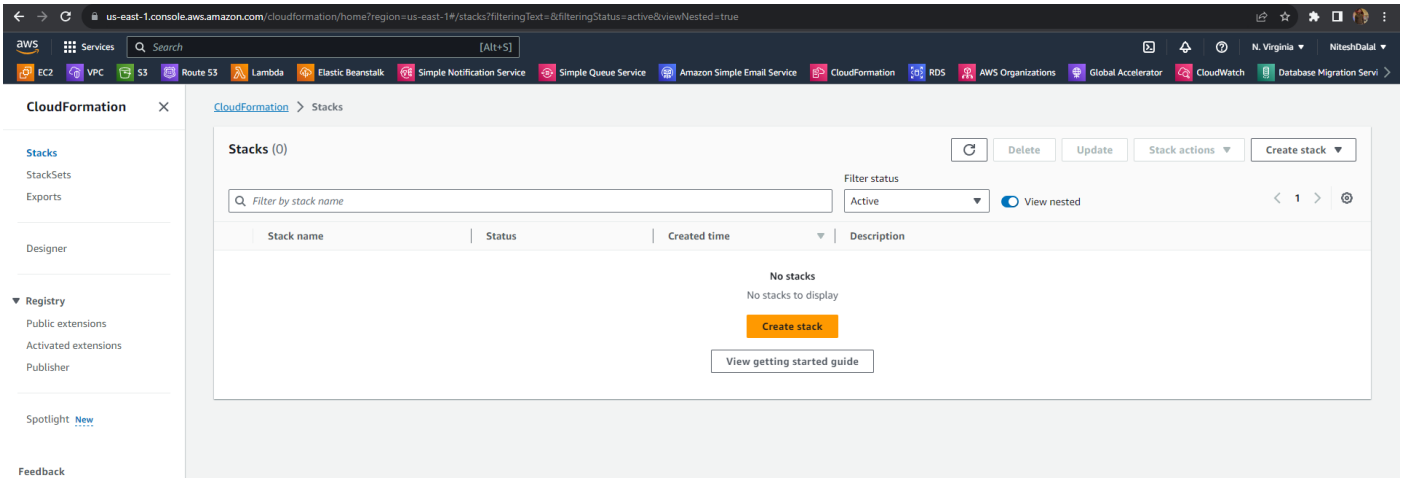
1. AWS CloudFormation to create a VPC

2. Connect VPC with AWS SNS

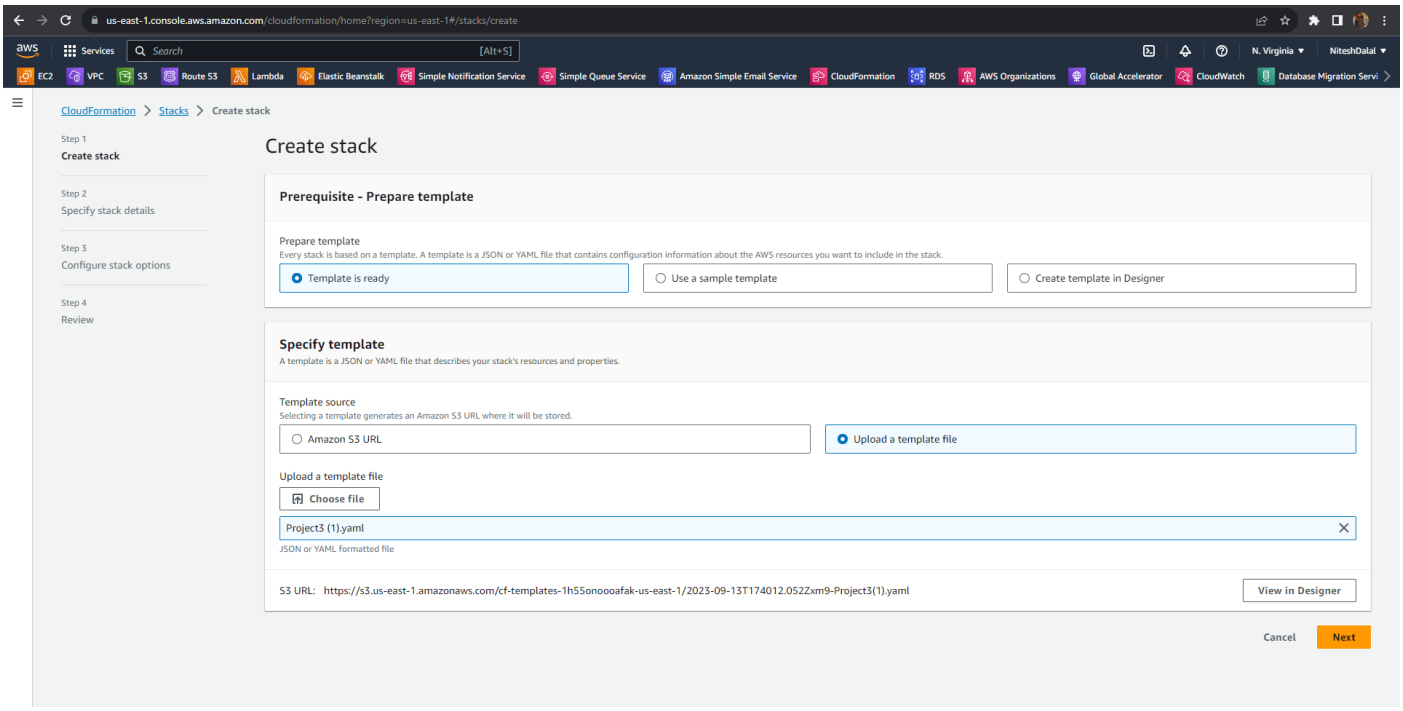
3. Publish message privately with SNS.

Solution:

1. Login into AWS and navigate to CloudFormation management console.



2. Click create stack and upload the template you have for creating the stack. Eother from S3 or from your local PC.



3. I will upload the template code here in this step.

```
AWSTemplateFormatVersion: 2010-09-09
Description: CloudFormation Template for SNS VPC Endpoints Tutorial
Parameters:
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instance
    Type: 'AWS::EC2::KeyPair::KeyName'
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  SSHLocation:
    Description: The IP address range that can be used to SSH to the EC2 instance
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: '(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
Mappings:
  RegionMap:
    us-east-1:
      AMI: ami-0c293f3f676ec4f90
    us-east-2:
      AMI: ami-08b6f2a5c291246a0
    us-west-1:
      AMI: ami-051317f1184dd6e92
    us-west-2:
      AMI: ami-0b9f27b05e1de14e9
Resources:
  VPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
      Tags:
```

```
      - Key: Name
        Value: VPCE-Tutorial-VPC
  Subnet:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref VPC
      CidrBlock: 10.0.1.0/24
      Tags:
       - Key: Name
         Value: VPCE-Tutorial-Subnet
  InternetGateway:
    Type: 'AWS::EC2::InternetGateway'
    Properties:
      Tags:
       - Key: Name
         Value: VPCE-Tutorial-InternetGateway
  VPCGatewayAttachment:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
  RouteTable:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref VPC
      Tags:
       - Key: Name
         Value: VPCE-Tutorial-RouteTable
  SubnetRouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref Subnet
  InternetGatewayRoute:
```

```
  Type: 'AWS::EC2::Route'
  Properties:
    RouteTableId: !Ref RouteTable
    GatewayId: !Ref InternetGateway
    DestinationCidrBlock: 0.0.0.0/0
SecurityGroup:
  Type: 'AWS::EC2::SecurityGroup'
  Properties:
    GroupName: Tutorial Security Group
    GroupDescription: Security group for SNS VPC endpoing tutorial
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: '-1'
        CidrIp: 10.0.0.0/16
      - IpProtocol: tcp
        FromPort: '22'
        ToPort: '22'
        CidrIp: !Ref SSHLocation
    SecurityGroupEgress:
      - IpProtocol: '-1'
        CidrIp: 10.0.0.0/16
    Tags:
      - Key: Name
        Value: VPCE-Tutorial-SecurityGroup
EC2Instance:
  Type: 'AWS::EC2::Instance'
  Properties:
    KeyName: !Ref KeyName
    InstanceType: t2.micro
    ImageId: !FindInMap
      - RegionMap
      - !Ref 'AWS::Region'
      - AMI
    NetworkInterfaces:
```

```
      - AssociatePublicIpAddress: 'true'
        DeviceIndex: '0'
        GroupSet:
          - !Ref SecurityGroup
        SubnetId: !Ref Subnet
    IamInstanceProfile: !Ref EC2InstanceProfile
    Tags:
      - Key: Name
        Value: VPCE-Tutorial-EC2Instance
EC2InstanceProfile:
  Type: 'AWS::IAM::InstanceProfile'
  Properties:
    Roles:
      - !Ref EC2InstanceRole
    InstanceProfileName: myEC2InstanceProfile-Sudip
EC2InstanceRole:
  Type: 'AWS::IAM::Role'
  Properties:
    RoleName: VPCE-Tutorial-EC2InstanceRole
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Principal:
            Service: ec2.amazonaws.com
          Action: 'sts:AssumeRole'
    ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/AmazonSNSFullAccess'
LambdaExecutionRole:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
```

```
      - Effect: Allow
        Principal:
          Service: lambda.amazonaws.com
        Action: 'sts:AssumeRole'
    ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
  LambdaFunction1:
    Type: 'AWS::Lambda::Function'
    Properties:
      Code:
        ZipFile: |
          from __future__ import print_function
          print('Loading function')
          def lambda_handler(event, context):
            message = event['Records'][0]['Sns']['Message']
            print("From SNS: " + message)
            return message
      Description: SNS VPC endpoint tutorial lambda function 1
      FunctionName: VPCE-Tutorial-Lambda-1
      Handler: index.lambda_handler
      Role: !GetAtt
        - LambdaExecutionRole
        - Arn
      Runtime: python3.9
      Timeout: '3'
  LambdaPermission1:
    Type: 'AWS::Lambda::Permission'
    Properties:
      Action: 'lambda:InvokeFunction'
      FunctionName: !Ref LambdaFunction1
      Principal: sns.amazonaws.com
      SourceArn: !Ref SNSTopic
  LambdaLogGroup1:
    Type: 'AWS::Logs::LogGroup'
```

```
  Properties:
    LogGroupName: !Sub "/aws/lambda/${LambdaFunction1}"
    RetentionInDays: '7'
LambdaFunction2:
  Type: 'AWS::Lambda::Function'
  Properties:
    Code:
      ZipFile: |
        from __future__ import print_function
        print('Loading function')
        def lambda_handler(event, context):
          message = event['Records'][0]['Sns']['Message']
          print("From SNS: " + message)
          return message
    Description: SNS VPC endpoint tutorial lambda function 2
    FunctionName: VPCE-Tutorial-Lambda-2
    Handler: index.lambda_handler
    Role: !GetAtt
      - LambdaExecutionRole
      - Arn
    Runtime: python3.9
    Timeout: '3'
LambdaPermission2:
  Type: 'AWS::Lambda::Permission'
  Properties:
    Action: 'lambda:InvokeFunction'
    FunctionName: !Ref LambdaFunction2
    Principal: sns.amazonaws.com
    SourceArn: !Ref SNSTopic
LambdaLogGroup2:
  Type: 'AWS::Logs::LogGroup'
  Properties:
    LogGroupName: !Sub "/aws/lambda/${LambdaFunction2}"
    RetentionInDays: '7'
```

SNSTopic:

Type: 'AWS::SNS::Topic'

Properties:

DisplayName: VPCE-Tutorial-Topic

TopicName: VPCE-Tutorial-Topic

Subscription:

- Endpoint: !GetAtt

  - LambdaFunction1

  - Arn

  Protocol: lambda

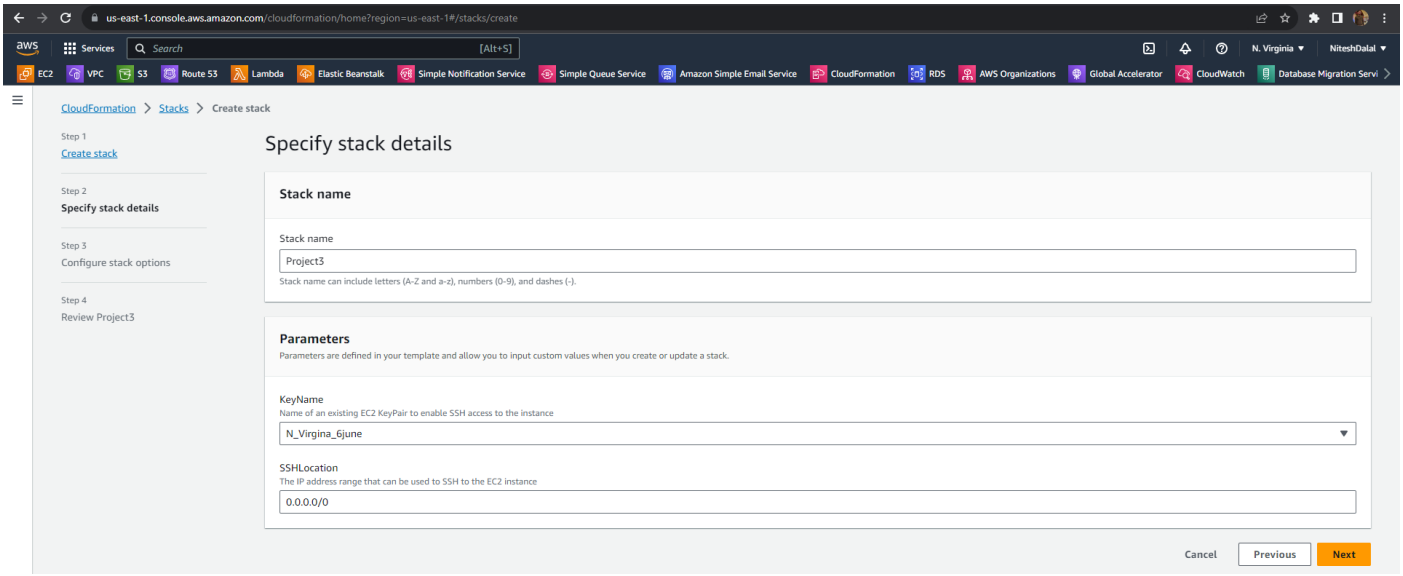
- Endpoint: !GetAtt

  - LambdaFunction2

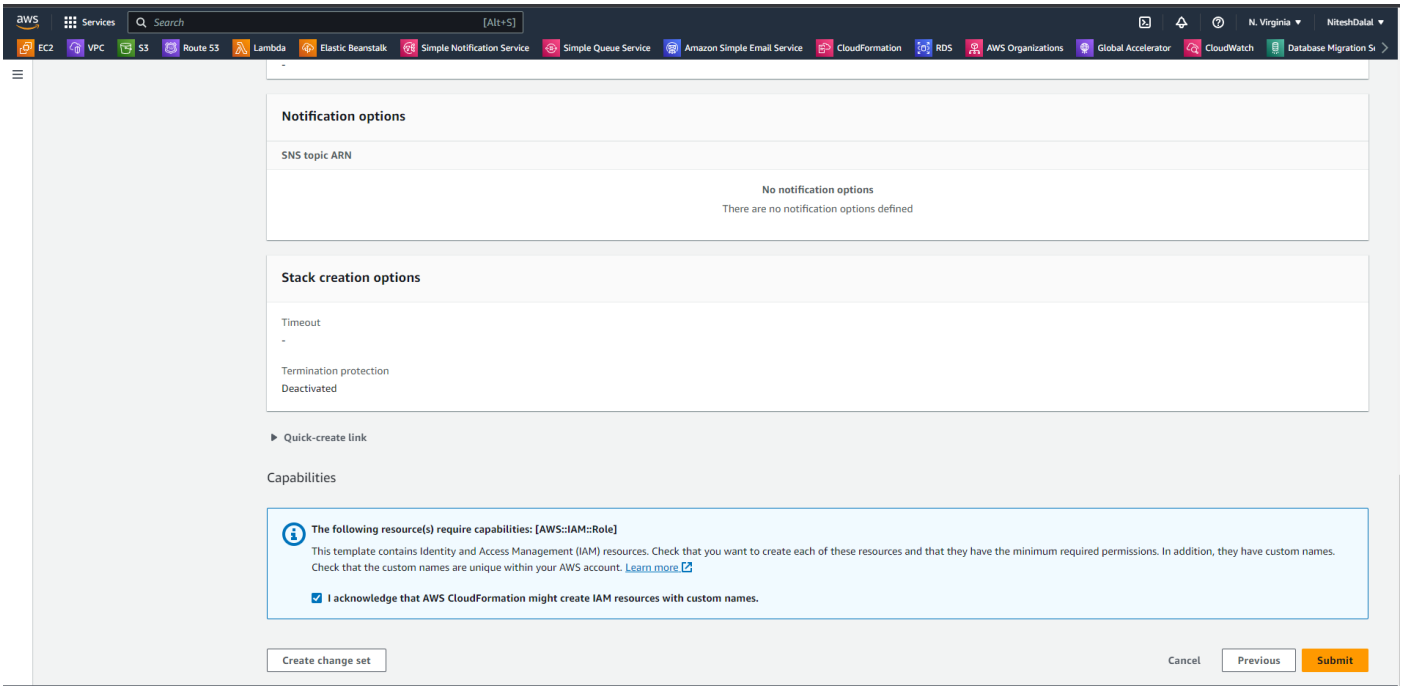
  - Arn

  Protocol: lambda

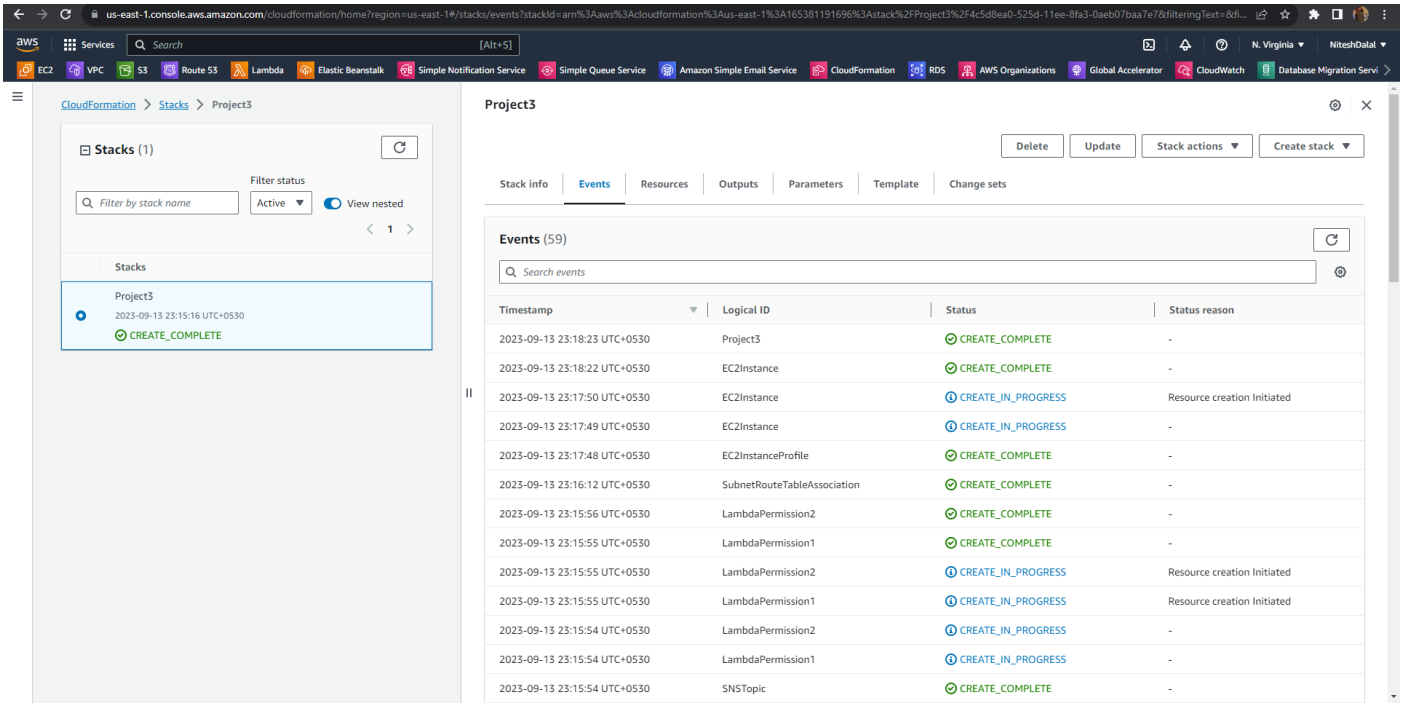
4. Now you need to pass the parameters value that is required. *Note a key is required to be present in your system to make SSH connection to your EC2*. I have chosen one that I already have.



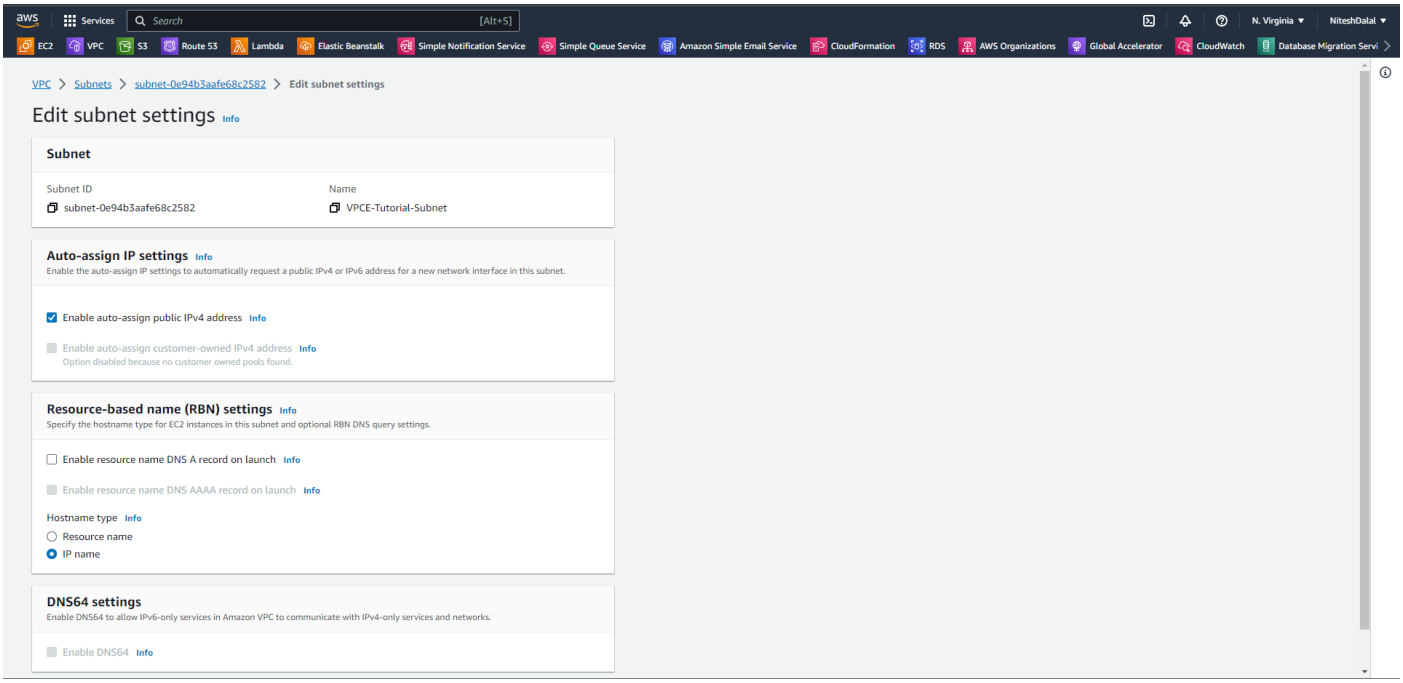
5. Rest all things are optional, just review & acknowledge it at last step. And click submit.



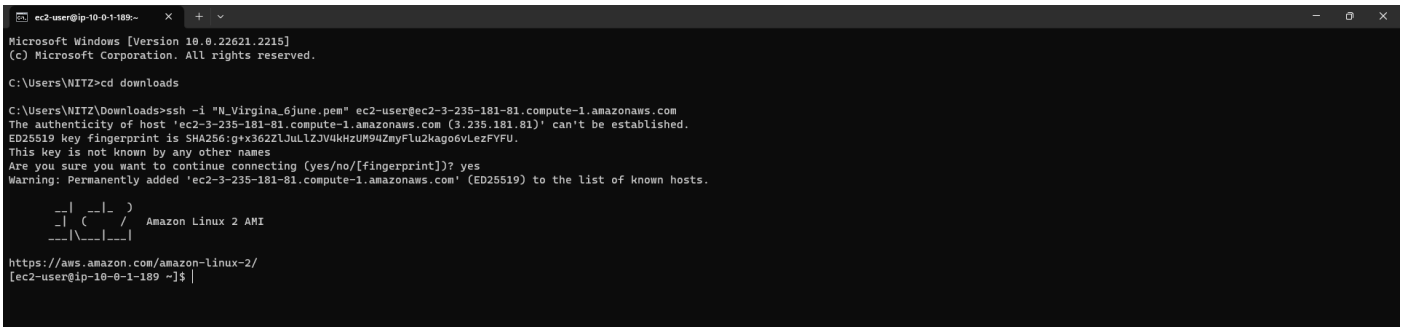
6. It will take up to 5-6 minutes. But once completed it will look like this.



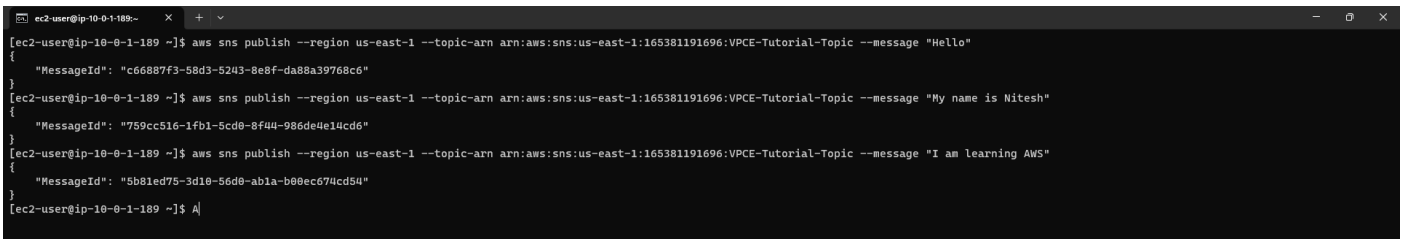
7. Now first thing to do is to Go to your VPC that is created by this cloud formation template and edit the subnet settings. Your subnet of VPC should allow auto assign public IPV4 address. Else you will fail to connect internet and public message to SNS.



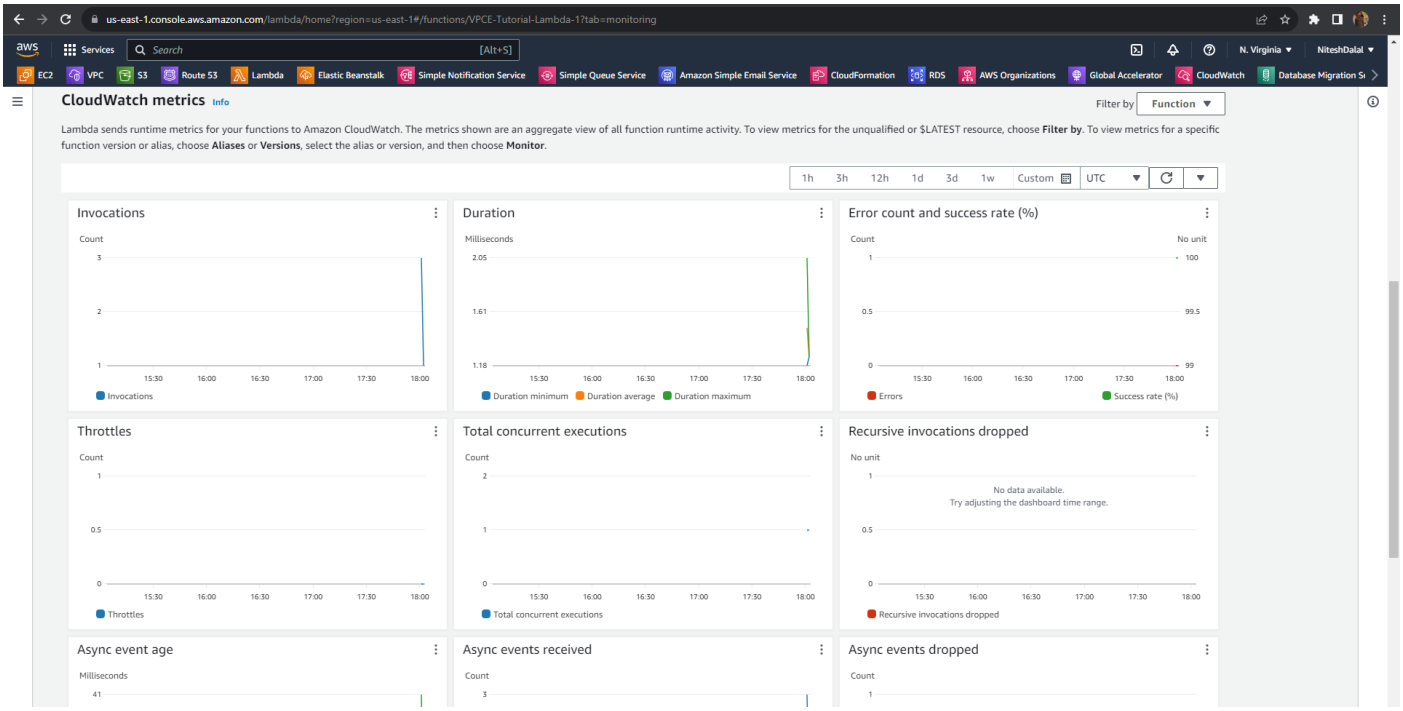
8. Now connect your EC2 instance through SSH or on web browser.



9. Try hitting the command to publish message i.e.
aws sns publish --region us-east-1 --topic-arn arn:aws:sns:us-east-1:165381191696:VPCE-Tutorial-Topic --message “Hello”



10. Now to monitor you can open lambda function and see from there. Also you can open cloudwatch logs groups to see the messages invocations.



11. Also from Logs.