3-TIER ARCHITECTURE WITH SNS NOTIFICATION

Name: Vikram

Problem Statement

You work for **XYZ Corporation**.

Your team is asked to deploy similar architecture multiple times for testing, development, and production purposes.

Implement CloudFormation for the tasks assigned to you below.

Tasks To Be Performed:

1. Use the template from CloudFormation task 1.
2. Add **Notification to the CloudFormation stack** using **SNS**, so that you get a notification via mail for every step of the stack creation process.

Step-by-Step Implementation

Step 1: Open CloudFormation

1. Go to the **AWS Management Console → CloudFormation → Create Stack (with new resources)**.
2. Choose **"Template is ready"** and **"Upload a template file"**.
3. Upload the updated CloudFormation YAML template file from task 1 (with SNS added).

```
AWSTemplateFormatVersion: '2010-09-09'
Description: >
  XYZ Corporation - 3 Tier Architecture (Web, App, DB) with Route53 DNS and SNS Notification.
  Web tier in public subnet, App tier in private subnet, DB tier (RDS MySQL) in private subnet.
  RDS instance retained after stack deletion. Sends SNS notifications for stack events.

Parameters:
  DomainName:
    Type: String
    Description: "Domain name for hosted zone (e.g., example.com)"
  KeyName:
    Type: AWS::EC2::KeyPair::KeyName
```

```
    Description: "Existing EC2 key pair for SSH login"
  DBUsername:
    Type: String
    Default: admin
    Description: "Database master username"
  DBPassword:
    Type: String
    NoEcho: true
    MinLength: 8
    Description: "Database master password"
  NotificationEmail:
    Type: String
    Description: "Email address to receive CloudFormation notifications"

Resources:

  # ------------------ SNS TOPIC & SUBSCRIPTION ------------------
  SNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      DisplayName: "XYZ CloudFormation Notifications"
      Subscription:
        - Endpoint: !Ref NotificationEmail
          Protocol: email
      TopicName: "xyz-cloudformation-sns"

  # ------------------ VPC & NETWORKING ------------------
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
```

```
      Value: xyz-vpc

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: xyz-igw

  VPCGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway

  PublicSubnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: us-east-1a
      CidrBlock: 10.0.1.0/24
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: xyz-public-subnet

  AppPrivateSubnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: us-east-1a
      CidrBlock: 10.0.2.0/24
      Tags:
        - Key: Name
          Value: xyz-app-private-subnet
```

```
DBPrivateSubnet:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: us-east-1b
    CidrBlock: 10.0.3.0/24
    Tags:
      - Key: Name
        Value: xyz-db-private-subnet

PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: xyz-public-rt

PublicRoute:
  Type: AWS::EC2::Route
  DependsOn: VPCGatewayAttachment
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PublicSubnetAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PublicSubnet
    RouteTableId: !Ref PublicRouteTable

# ------------------ SECURITY GROUPS ------------------
WebSecurityGroup:
```

```
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow HTTP and SSH from internet
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
    Tags:
      - Key: Name
        Value: xyz-web-sg

AppSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow SSH only from Web Tier
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        SourceSecurityGroupId: !Ref WebSecurityGroup
    Tags:
      - Key: Name
        Value: xyz-app-sg

DBSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow MySQL only from App Tier
```

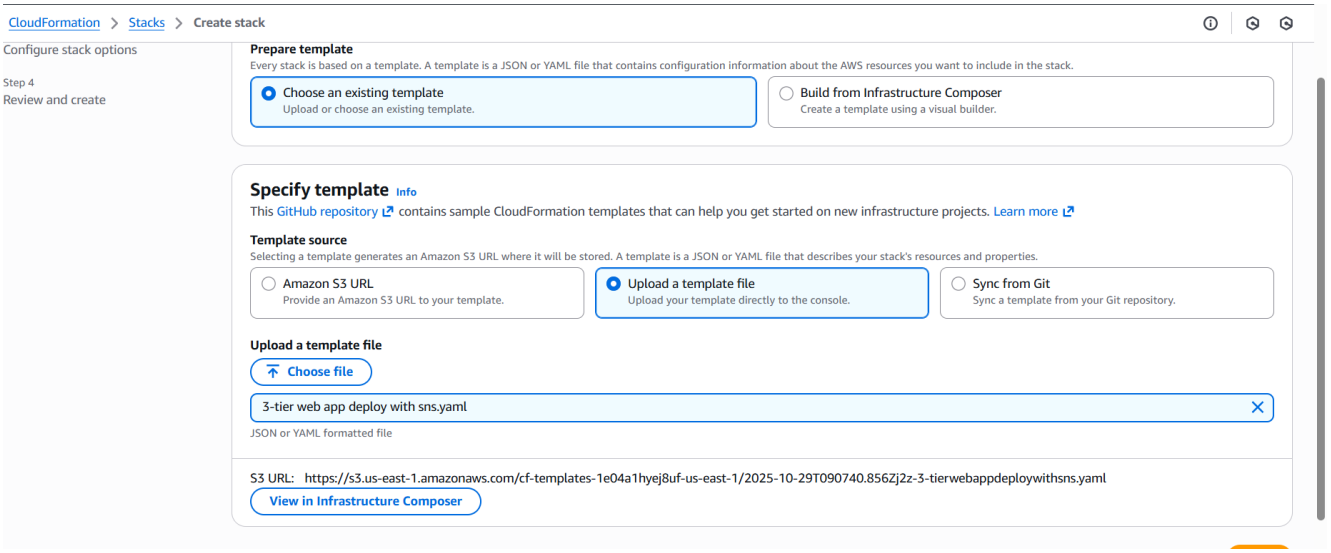
```
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 3306
        ToPort: 3306
        SourceSecurityGroupId: !Ref AppSecurityGroup
    Tags:
      - Key: Name
        Value: xyz-db-sg

# ------------------ EC2 INSTANCES ------------------
WebInstance:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t3.micro
    KeyName: !Ref KeyName
    ImageId: ami-07860a2d7eb515d9a  # ✅ Updated AMI
    SubnetId: !Ref PublicSubnet
    SecurityGroupIds:
      - !Ref WebSecurityGroup
    Tags:
      - Key: Name
        Value: xyz-web-instance
    UserData:
      Fn::Base64: !Sub |
        #!/bin/bash
        yum update -y
        yum install -y httpd
        systemctl enable httpd
        systemctl start httpd
        echo "<h1>Welcome to XYZ Web Server (Web Tier)</h1>" > /var/www/html/index.html

AppInstance:
  Type: AWS::EC2::Instance
  Properties:
```

```
    InstanceType: t3.micro
    KeyName: !Ref KeyName
    ImageId: ami-07860a2d7eb515d9a  # ✅ Updated AMI
    SubnetId: !Ref AppPrivateSubnet
    SecurityGroupIds:
      - !Ref AppSecurityGroup
    Tags:
      - Key: Name
        Value: xyz-app-instance

# ------------------ RDS DATABASE ------------------
DBSubnetGroup:
  Type: AWS::RDS::DBSubnetGroup
  Properties:
    DBSubnetGroupDescription: Subnet group for xyz DB
    SubnetIds:
      - !Ref AppPrivateSubnet
      - !Ref DBPrivateSubnet
    Tags:
      - Key: Name
        Value: xyz-db-subnet-group

MySQLDB:
  Type: AWS::RDS::DBInstance
  DeletionPolicy: Retain
  Properties:
    DBInstanceIdentifier: xyz-mysql-db
    AllocatedStorage: 20
    DBInstanceClass: db.t3.micro
    Engine: mysql
    MasterUsername: !Ref DBUsername
    MasterUserPassword: !Ref DBPassword
    DBSubnetGroupName: !Ref DBSubnetGroup
    VPCSecurityGroups:
      - !Ref DBSecurityGroup
```

```
      PubliclyAccessible: false
      MultiAZ: false

  # ------------------ ROUTE 53 ------------------
  HostedZone:
    Type: AWS::Route53::HostedZone
    Properties:
      Name: !Ref DomainName
      HostedZoneConfig:
        Comment: "XYZ Hosted Zone"

  DNSRecord:
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref HostedZone
      Name: !Sub "${DomainName}."
      Type: A
      TTL: '300'
      ResourceRecords:
        - !GetAtt WebInstance.PublicIp

Outputs:
  WebInstancePublicIP:
    Description: "Public IP of Web Instance"
    Value: !GetAtt WebInstance.PublicIp
  AppInstanceID:
    Description: "Application Instance ID"
    Value: !Ref AppInstance
  DBEndpoint:
    Description: "RDS MySQL Endpoint"
    Value: !GetAtt MySQLDB.Endpoint.Address
  HostedZoneID:
    Description: "Route53 Hosted Zone ID"
    Value: !Ref HostedZone
  SNSTopicARN:
```

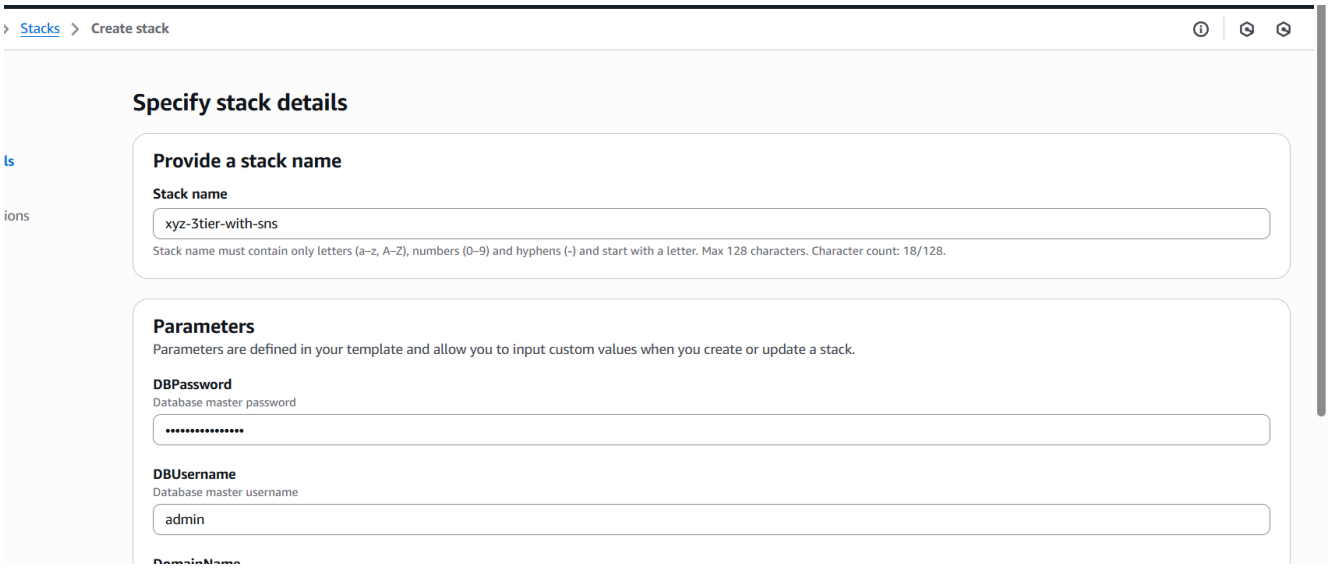
Description: "SNS Topic for CloudFormation Notifications"

Value: !Ref SNSTopic



Step 2: Provide Parameters

- **DomainName** = xyztest.local
- **KeyName** = *(your existing EC2 key pair)*
- **DBUsername** = admin
- **DBPassword** = *(choose a strong password)*
- **NotificationEmail** = Provide Email



DomainName

Domain name for hosted zone (e.g., example.com)

xyztest.local

KeyName

Existing EC2 key pair for SSH login

vik-87

NotificationEmail

Email address to receive CloudFormation notifications

gangadharavikram2002@gmail.com

Cancel | Previous | Next

CloudFormation > Stacks > xyz-3tier-with-sns

tacks (0)

Filter status

Search by stack name | Active

View nested

1

Stacks

No stacks

No stacks to display

Create stack

View getting started guide

xyz-3tier-with-sns

Delete | Update stack | Stack actions | Create stack

Stack info | Events | Resources | Outputs | Parameters | Template | Changesets | Git sync

Table view | Timeline view

Events (1)

View root cause

Search events

Timestamp	Logical ID	Status	Detailed status	Status reason
2025-10-29 14:40:14 UTC+0530	xyz-3tier-with-sns	CREATE_IN_PROGRESS	-	User Initiated

---

Events (49)

View root cause

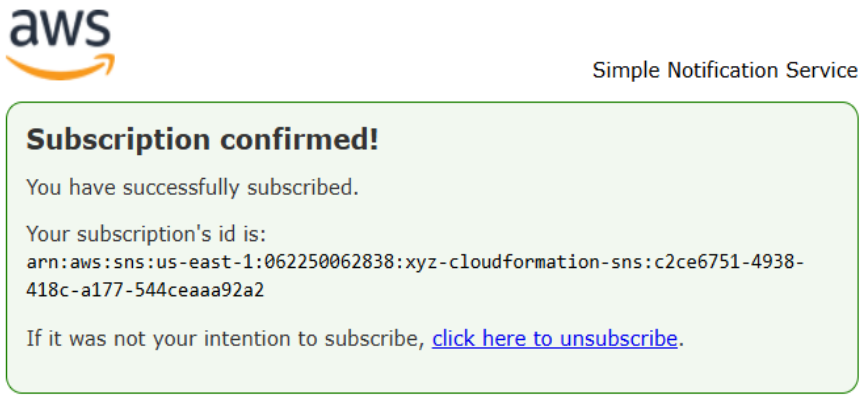
Search events

Timestamp	Logical ID	Status	Detailed status	Status reason
2025-10-29 15:22:15 UTC+0530	PublicRoute	CREATE_COMPLETE	-	-
2025-10-29 15:22:14 UTC+0530	PublicRoute	CREATE_IN_PROGRESS	-	Resource creation Initiated
2025-10-29 15:22:13 UTC+0530	WebInstance	CREATE_IN_PROGRESS	CONFIGURATION_COMPLETE	Eventual consistency check initiated
2025-10-29 15:22:13 UTC+0530	PublicRoute	CREATE_IN_PROGRESS	-	-
2025-10-29 15:22:12 UTC+0530	PublicRouteTable	CREATE_COMPLETE	-	-
2025-10-29 15:22:12 UTC+0530	AppSecurityGroup	CREATE_IN_PROGRESS	-	Resource creation Initiated
2025-10-29 15:22:11		CREATE_IN_PROGRES		Resource creation

Step 3: Confirm SNS Subscription

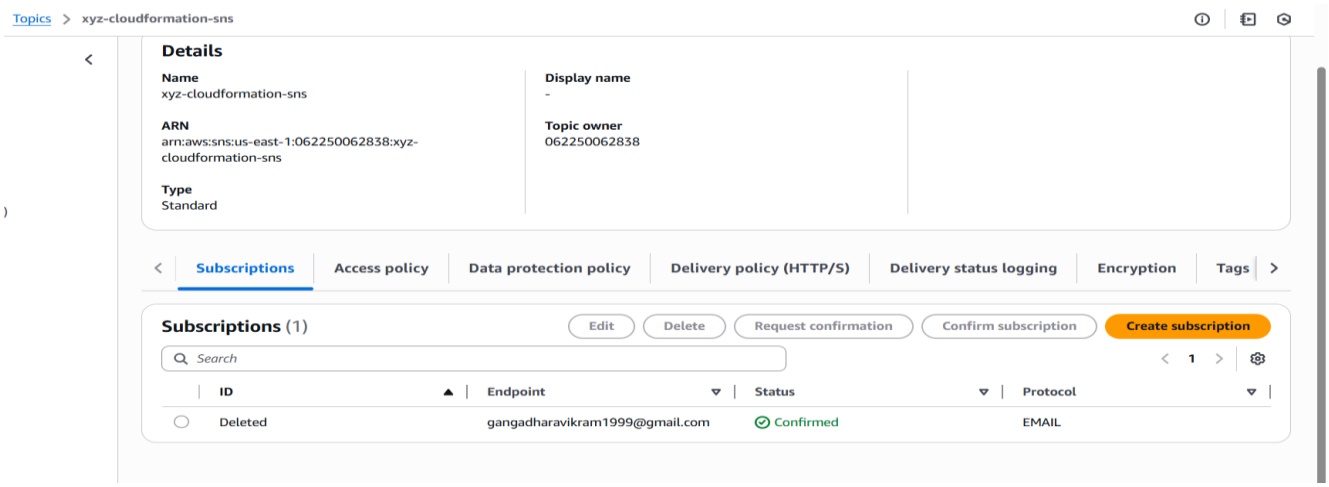
1. After launching the stack, check your email inbox (gangadharavikram1999@gmail.com).
2. You'll receive a **"AWS Notification Subscription"** mail.
3. Click **"Confirm subscription"** to activate it.

Once confirmed, you'll start receiving notifications for each CloudFormation event — such as *CREATE_IN_PROGRESS, CREATE_COMPLETE, ROLLBACK_COMPLETE*, etc.



Step 4: Stack Creation Complete

- Once stack creation finishes, check your **SNS Topic** in the **AWS SNS Console → Subscriptions** tab — the status should show **"Confirmed"**.
- You will receive an email whenever the stack is created, updated, or deleted.



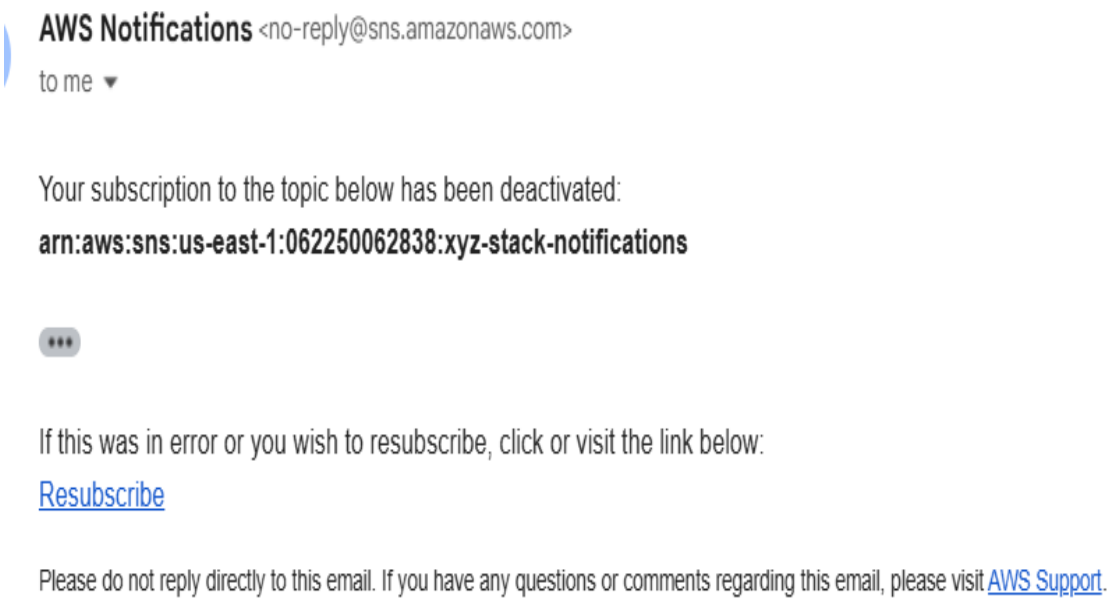
Step 5: Validation

Note: SNS Email Subscription Behavior

During the implementation of the SNS notification setup, the topic and email subscription were successfully created using both CloudFormation and the AWS Management Console. However, upon confirming the subscription via email, it was automatically **deactivated within a few seconds**.

This behavior occurs due to **SNS sandbox restrictions** in certain AWS accounts, which temporarily limit or disable email notifications.

✅ The SNS setup is technically correct — in a fully verified AWS (production) account, notifications will work as expected.



✅ Final Output:

- successfully deployed a **3-tier architecture** using **CloudFormation** and configured **SNS Email Notification** for every stack event.