

# 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

CloudFormation > Stacks > Create stack

Configure stack options

Step 4  
Review and create

**Prepare template**  
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

Choose an existing template  
Upload or choose an existing template.

Build from Infrastructure Composer  
Create a template using a visual builder.

**Specify template** Info  
This GitHub repository [contains sample CloudFormation templates that can help you get started on new infrastructure projects. Learn more](#)

**Template source**  
Selecting a template generates an Amazon S3 URL where it will be stored. A template is a JSON or YAML file that describes your stack's resources and properties.

Amazon S3 URL  
Provide an Amazon S3 URL to your template.

Upload a template file  
Upload your template directly to the console.

Sync from Git  
Sync a template from your Git repository.

**Upload a template file**  
[Choose file](#)

3-tier web app deploy with sns.yaml X  
JSON or YAML formatted file

S3 URL: <https://s3.us-east-1.amazonaws.com/cf-templates-1e04a1hyej8uf-us-east-1/2025-10-29T090740.856Zj2z-3-tierwebappdeploywithsns.yaml>

[View in Infrastructure Composer](#)

## Step 2: Provide Parameters

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

Stacks > Create stack

**Specify stack details**

**Provide a stack name**

**Stack name**  
xyz-3tier-with-sns

Stack name must contain only letters (a-z, A-Z), numbers (0-9) and hyphens (-) and start with a letter. Max 128 characters. Character count: 18/128.

**Parameters**  
Parameters are defined in your template and allow you to input custom values when you create or update a stack.

**DBPassword**  
Database master password  
\*\*\*\*\*

**DBUsername**  
Database master username  
admin

**DomainName**

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

**KeyName**  
Existing EC2 key pair for SSH login

**NotificationEmail**  
Email address to receive CloudFormation notifications

[Cancel](#) [Previous](#) [Next](#)

[CloudFormation](#) > [Stacks](#) > xyz-3tier-with-sns

Stacks (0)	X	Filter status
<input type="text" value="Search by stack name"/> <a href="#">Active</a>		<a href="#">Delete</a> <a href="#">Update stack</a> <a href="#">Stack actions</a> <a href="#">Create stack</a>
<a href="#">View nested</a>		
< 1 >		
<a href="#">Stacks</a> No stacks No stacks to display <a href="#">Create stack</a> <a href="#">View getting started guide</a>		

**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)**

Timestamp	Logical ID	Status	Detailed status	Status reason
2025-10-29 14:40:14 UTC+0530	<a href="#">xyz-3tier-with-sns</a>	CREATE_IN_PROGRESS	-	User Initiated

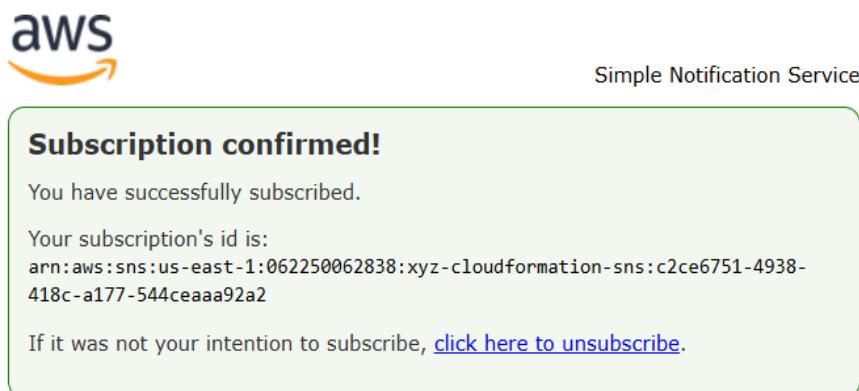
**Events (49)**

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	<a href="#">WebInstance</a>	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	<a href="#">AppSecurityGroup</a>	CREATE_IN_PROGRESS	-	Resource creation initiated
2025-10-29 15:22:11 UTC+0530	<a href="#">WebInstance</a>	CREATE_IN_PROGRESS	-	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.

The screenshot shows the AWS SNS console. At the top, there is a breadcrumb navigation: Topics > xyz-cloudformation-sns. Below this, a "Details" panel shows the topic's name (xyz-cloudformation-sns), ARN (arn:aws:sns:us-east-1:062250062838:xyz-cloudformation-sns), and type (Standard). To the right of the details panel is a "Subscriptions" tab, which is currently selected. Below the tabs, there is a "Subscriptions (1)" section. It contains a table with one row, showing a single subscription. The subscription details are: ID (Deleted), Endpoint (gangadharavikram1999@gmail.com), Status (Confirmed), and Protocol (EMAIL). There are also buttons for "Edit", "Delete", "Request confirmation", "Confirm subscription", and "Create subscription".

---

## 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.

AWS Notifications <no-reply@sns.amazonaws.com>

to me ▾

Your subscription to the topic below has been deactivated:

**arn:aws:sns:us-east-1:062250062838:xyz-stack-notifications**



If this was in error or you wish to resubscribe, click or visit the link below:

[Resubscribe](#)

Please do not reply directly to this email. If you have any questions or comments regarding this email, please visit [AWS Support](#).

---

### Final Output:

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