Name: Mrs. Rupali N Hosmani.

Course: Executive Post Graduate Certification in Cloud Computing

Contact No. 7720003531

**Assignment : CloudFormation**

1. **Task to Be Performed:**
2. Web tier: Launch an instance in a public subnet and that instance should allow HTTP and SSH from the internet.
3. 2. Application tier: Launch an instance in a private subnet of the web tier and it should allow only SSH from the public subnet of Web Tier-3.
4. 3. DB tier: Launch an RDS MYSQL instance in a private subnet and it should allow connection on port 3306 only from the private subnet of Application Tier-4.
5. Setup a Route 53 hosted zone and direct the traffic to the EC2 instance.

**To propose a solution so that:**

1. Development team can test their code without having to involve the system admins and can invest

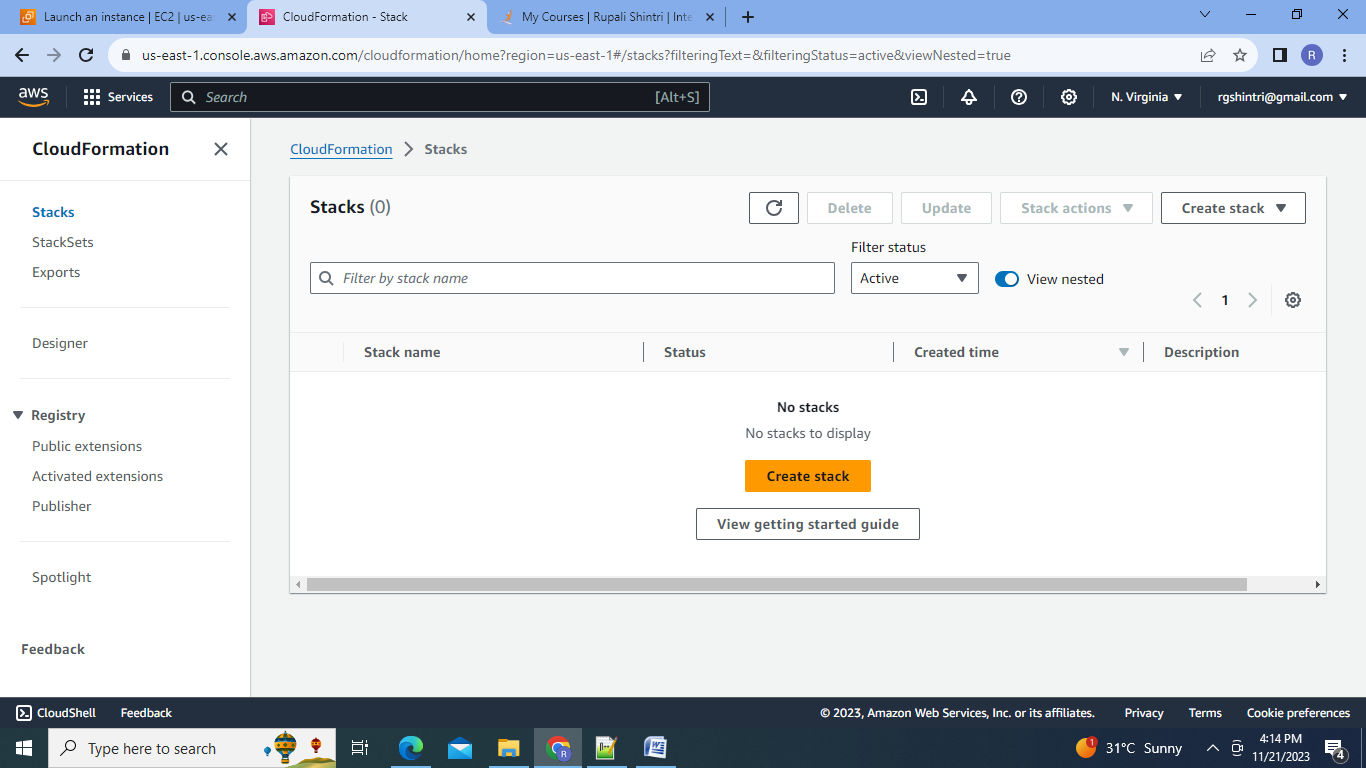
their time in testing the code rather than provisioning, configuring and updating the resources needed to test the code.

1. Make sure when the development team deletes the stack, RDS DBinstances should not be deleted.
2. **Tasks To Be Performed**:
3. Use the template from CloudFormation task 1.
4. Add Notification to the CloudFormation stack using SNS so that you get a notification via mail for every step of the stack creation process.
5. **Tasks To Be Performed**:
6. Create a FIFO SQS queue and test by sending messages.
7. Register your mail in SES and send a test mail to yourself.
8. **Steps to be performed:**
9. Prepare the yaml template specifying the resources and their properties that to be uploaded on to the stack.

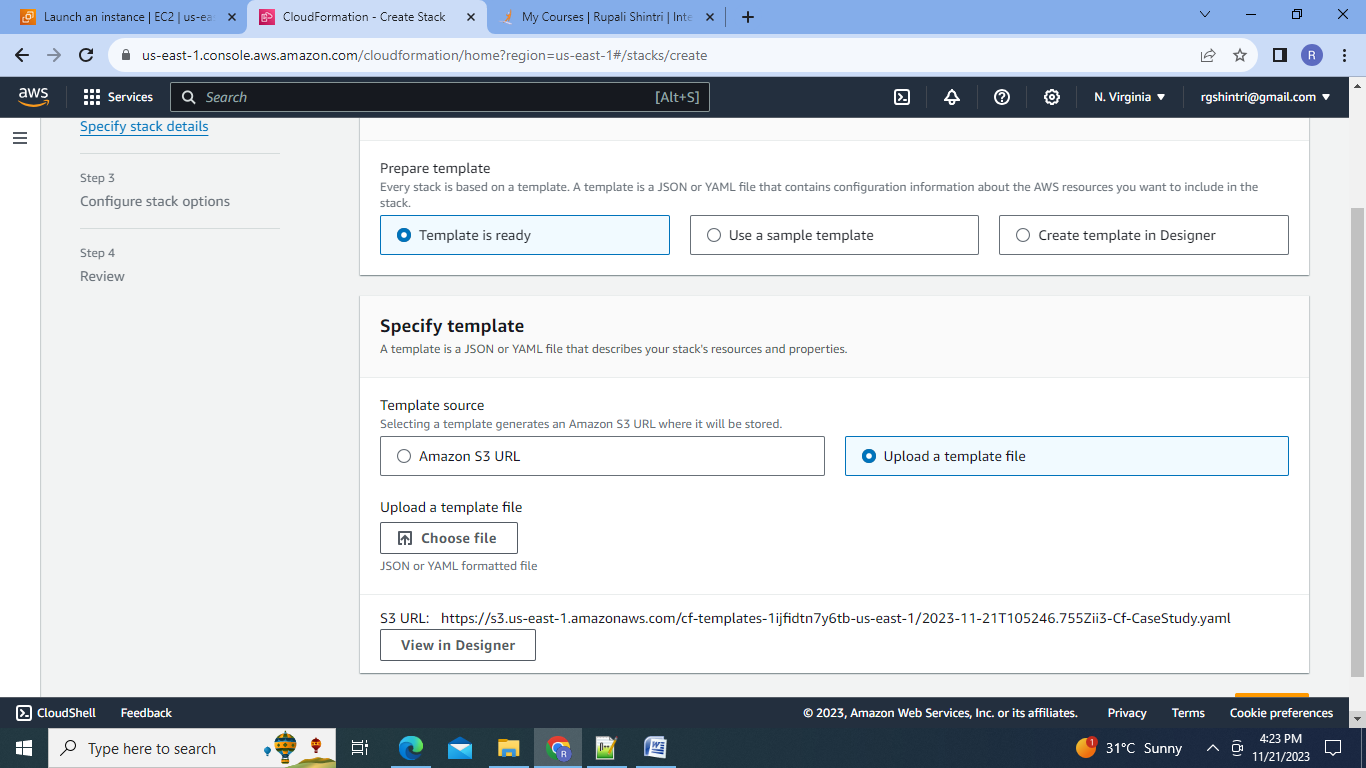
Cloud Template:

|  |
| --- |
| AWSTemplateFormatVersion: 2010-09-09  Parameters:  InstanceTypeParameter:  Type: String  Default: t2.micro  Description: Enter instance size. Default is t2.micro.  AMI:  Type: String  Default: ami-0fc5d935ebf8bc3bc  Description: The Ubuntu AMI to use.  Key:  Type: AWS::EC2::KeyPair::KeyName  Description: Select from Existing Keys.  MasterUsername:  Type: String  Default: null  Description: The username for the database.    MasterUserPassword:  Type: String  Default: null  Description: The password for the database.  "NoEcho": true    Resources:  VPC:  Type: AWS::EC2::VPC  Properties:  CidrBlock: 10.10.0.0/16  EnableDnsSupport: true  EnableDnsHostnames: true  InstanceTenancy: default  Tags:  - Key: Name  Value: VPCAssessment  InternetGateway:  Type: AWS::EC2::InternetGateway  Properties:  Tags:  - Key: Name  Value: InternetGatewayAssessment  VPCGatewayAttachment:  Type: AWS::EC2::VPCGatewayAttachment  Properties:  VpcId: !Ref VPC  InternetGatewayId: !Ref InternetGateway  #Public Subnet  SubnetA:  Type: AWS::EC2::Subnet  Properties:  AvailabilityZone: !Select [ 0, !GetAZs ]  VpcId: !Ref VPC  CidrBlock: 10.10.1.0/24  MapPublicIpOnLaunch: true  Tags:  - Key: Name  Value: PublicSubnetAssessment  PublicRouteTable:  Type: AWS::EC2::RouteTable  Properties:  VpcId: !Ref VPC  Tags:  - Key: Name  Value: RouteTablePublicSubnet  PublicInternetRoute:  Type: AWS::EC2::Route  DependsOn: VPCGatewayAttachment  Properties:  DestinationCidrBlock: 0.0.0.0/0  GatewayId: !Ref InternetGateway  RouteTableId: !Ref PublicRouteTable  SubnetARouteTableAssociation:  Type: AWS::EC2::SubnetRouteTableAssociation  Properties:  RouteTableId: !Ref PublicRouteTable  SubnetId: !Ref SubnetA    #Private Subnet  SubnetB:  Type: AWS::EC2::Subnet  Properties:  AvailabilityZone: !Select [ 1, !GetAZs ]  VpcId: !Ref VPC  CidrBlock: 10.10.2.0/24  MapPublicIpOnLaunch: false  Tags:  - Key: Name  Value: PrivateSubnetAssessment    # A NAT Gateway:  NATGateway:  Type: AWS::EC2::NatGateway  Properties:  AllocationId: !GetAtt ElasticIPAddress.AllocationId  SubnetId: !Ref SubnetA  Tags:  - Key: Name  Value: NatGetwayAssessment  ElasticIPAddress:  Type: AWS::EC2::EIP  Properties:  Domain: VPC  RouteTablePrivate:  Type: AWS::EC2::RouteTable  Properties:  VpcId: !Ref VPC  Tags:  - Key: Name  Value: RouteTablePrivateSubnet  NATRoute:  DependsOn: NATGateway  Type: AWS::EC2::Route  Properties:  RouteTableId: !Ref RouteTablePrivate  DestinationCidrBlock: 0.0.0.0/0  NatGatewayId: !Ref NATGateway  SubnetBRouteTableAssociationPrivate:  Type: AWS::EC2::SubnetRouteTableAssociation  Properties:  RouteTableId: !Ref RouteTablePrivate  SubnetId: !Ref SubnetB  InstanceSecurityGroup:  Type: AWS::EC2::SecurityGroup  Properties:  GroupName: "Internet Group"  GroupDescription: "SSH and web traffic in, all traffic out."  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  SecurityGroupEgress:  - IpProtocol: -1  CidrIp: 0.0.0.0/0  InstanceSecurityGroupPrivate:  Type: AWS::EC2::SecurityGroup  Properties:  GroupName: "Security Group Private"  GroupDescription: "SSH from the Public Subnet"  VpcId: !Ref VPC  SecurityGroupIngress:  - IpProtocol: tcp  FromPort: '22'  ToPort: '22'  CidrIp: 10.10.1.0/24  SecurityGroupEgress:  - IpProtocol: -1  CidrIp: 0.0.0.0/0  InstanceSecurityGroupDataBase:  Type: "AWS::EC2::SecurityGroup"  Properties:  GroupDescription: "Database instances security group"  VpcId: !Ref VPC  SecurityGroupIngress:  - IpProtocol: tcp  CidrIp: 10.10.2.0/24  FromPort: 3306  ToPort: 3306  SecurityGroupEgress:  - IpProtocol: -1  CidrIp: 0.0.0.0/0  RDSDBSubnetGroup:  Type: "AWS::RDS::DBSubnetGroup"  Properties:  DBSubnetGroupDescription: "Subnet Group for mySQL database"  DBSubnetGroupName: !Sub "${AWS::Region}-aws-database-subnet-group14"  SubnetIds:  - !Ref SubnetA  - !Ref SubnetB  Tags:  - Key: Name  Value: DBSubnetGroup    RDSDBInstance:  Type: AWS::RDS::DBInstance  Properties:  DBInstanceIdentifier: DBAssessment12  AllocatedStorage: 20  DBInstanceClass: db.t2.micro  Engine: "MYSQL"  MasterUsername: !Ref MasterUsername  MasterUserPassword: !Ref MasterUserPassword  MultiAZ: false  EngineVersion: 8.0.28  AutoMinorVersionUpgrade: true  PubliclyAccessible: false  StorageType: gp2  Port: 3306  StorageEncrypted: false  CopyTagsToSnapshot: true  EnableIAMDatabaseAuthentication: false  DeletionProtection: true  DBSubnetGroupName: !Ref RDSDBSubnetGroup  VPCSecurityGroups:  - !Ref InstanceSecurityGroupDataBase  MaxAllocatedStorage: 1000  Tags:  - Key: Name  Value: DBAssessment  - Key: createdBy  Value: Igor Silva  - Key: Project  Value: AssessmentModule7  - Key: Environment  Value: Prod  LinuxPublic:  Type: 'AWS::EC2::Instance'  Properties:  SubnetId: !Ref SubnetA  ImageId: !Ref AMI  InstanceType: !Ref InstanceTypeParameter  KeyName: !Ref Key  SecurityGroupIds:  - Ref: InstanceSecurityGroup  Tags:  - Key: Name  Value: LinuxPublic  LinuxPrivate:  Type: 'AWS::EC2::Instance'  Properties:  SubnetId: !Ref SubnetB  ImageId: !Ref AMI  InstanceType: !Ref InstanceTypeParameter  KeyName: !Ref Key  SecurityGroupIds:  - Ref: InstanceSecurityGroupPrivate  Tags:  - Key: Name  Value: LinuxPrivate    HostedZone:  Type: AWS::Route53::HostedZone  Properties:  HostedZoneConfig:  Comment: ''  Name: newpracticedomain.ml  MyDNSRecord:  Type: AWS::Route53::RecordSet  Properties:  HostedZoneId: !Ref HostedZone  Name: www.newpracticedomain.ml.  Type: A  TTL: 300  ResourceRecords:  - !GetAtt LinuxPublic.PublicIp    Outputs:  PublicIp:  Description: Server's PublicIp Address  Value:  Fn::GetAtt:  - LinuxPublic  - PublicIp  HostedZoneID:  Description: The ID of the Hosted Zone.  Value:  Ref: HostedZone |

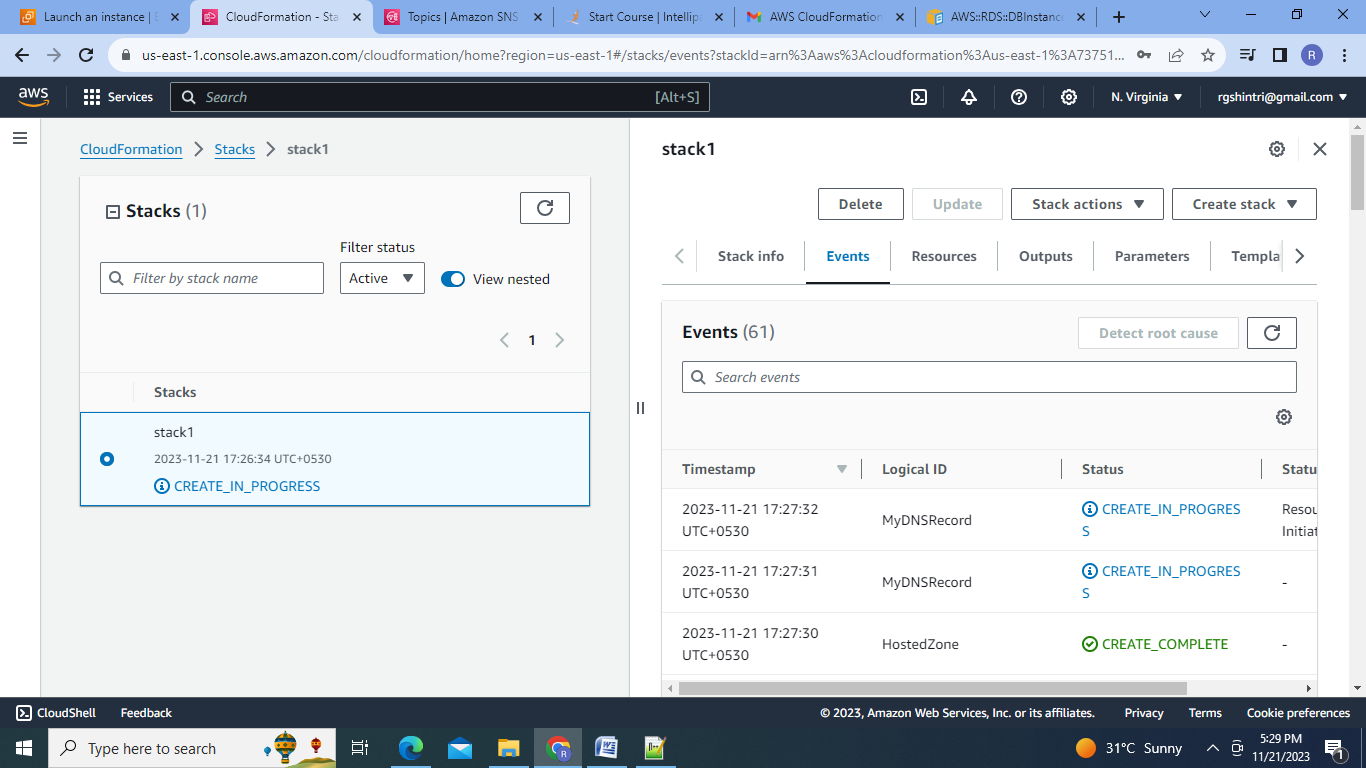
1. Go to CloudFormation Console & then go to create stack option and upload this template to crate the stack.

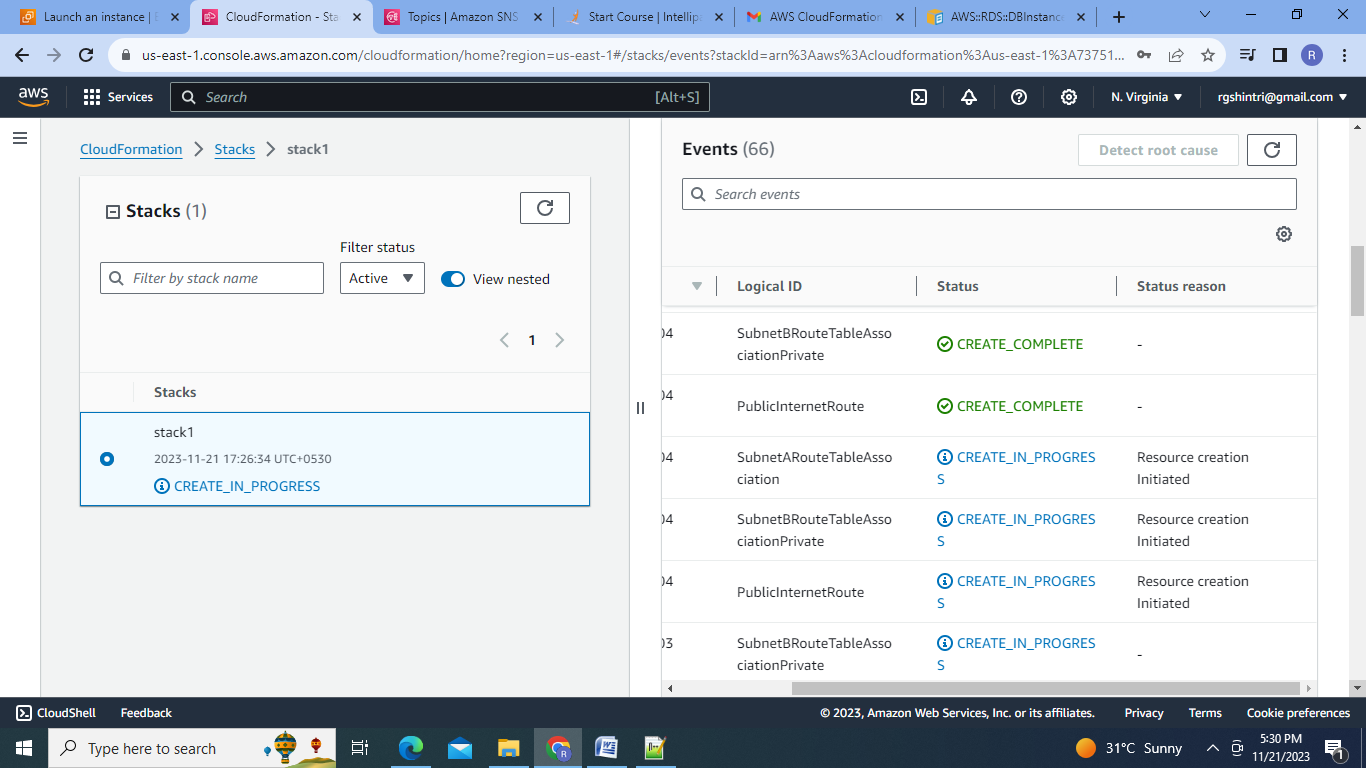


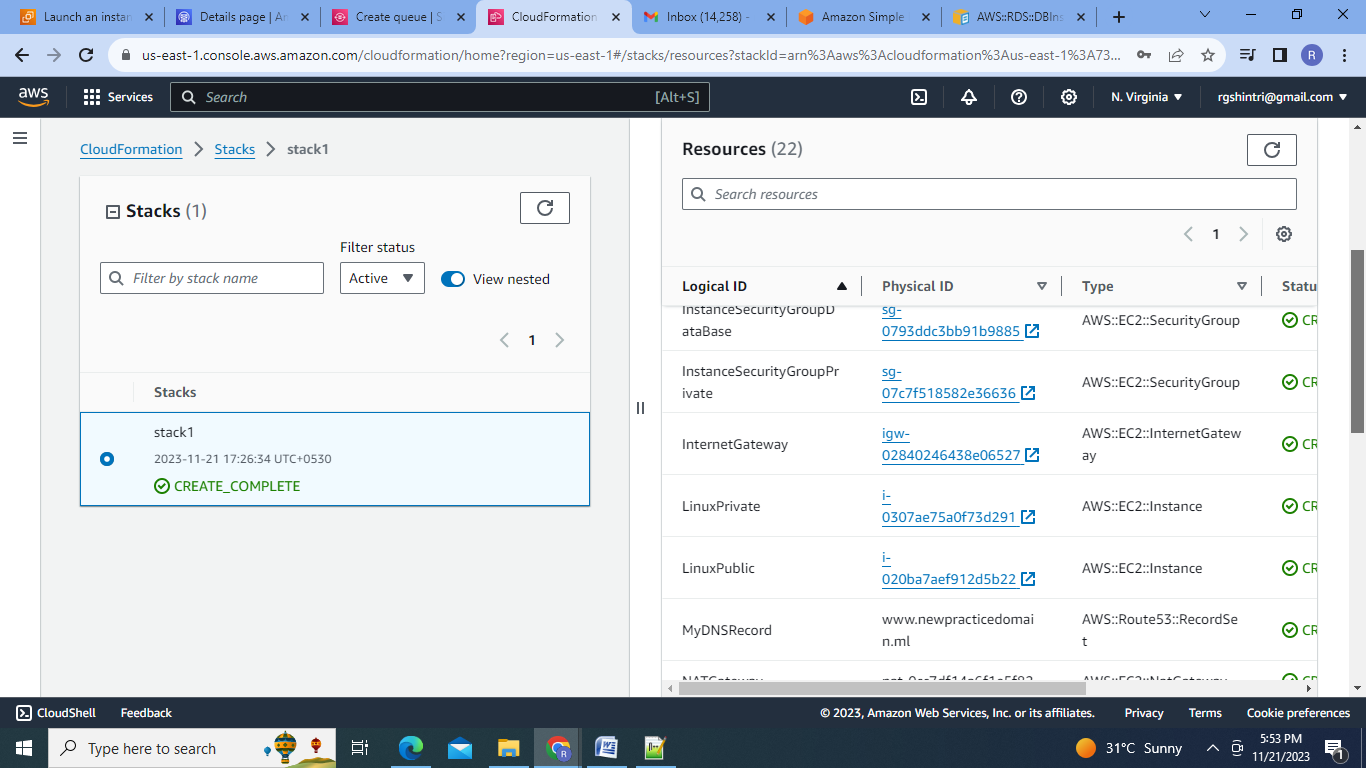
3.Upload the Template:

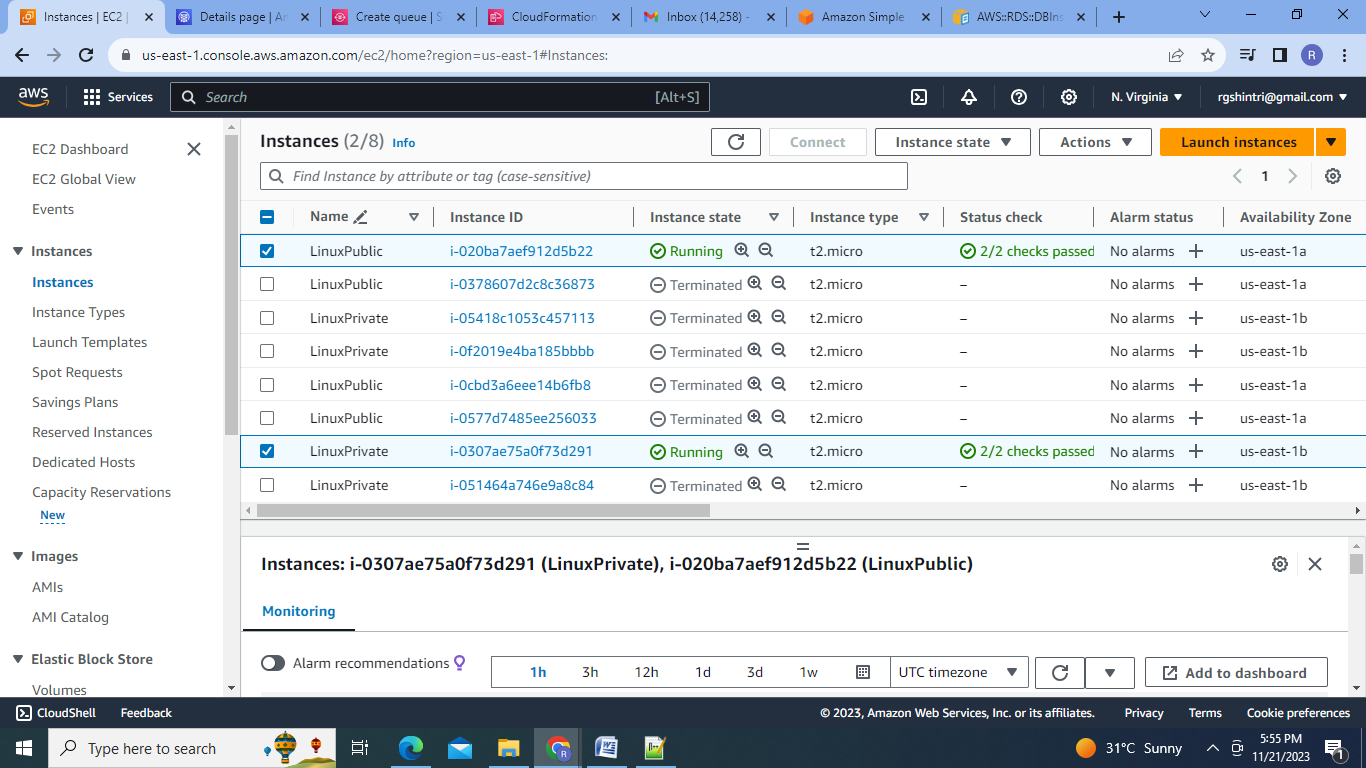


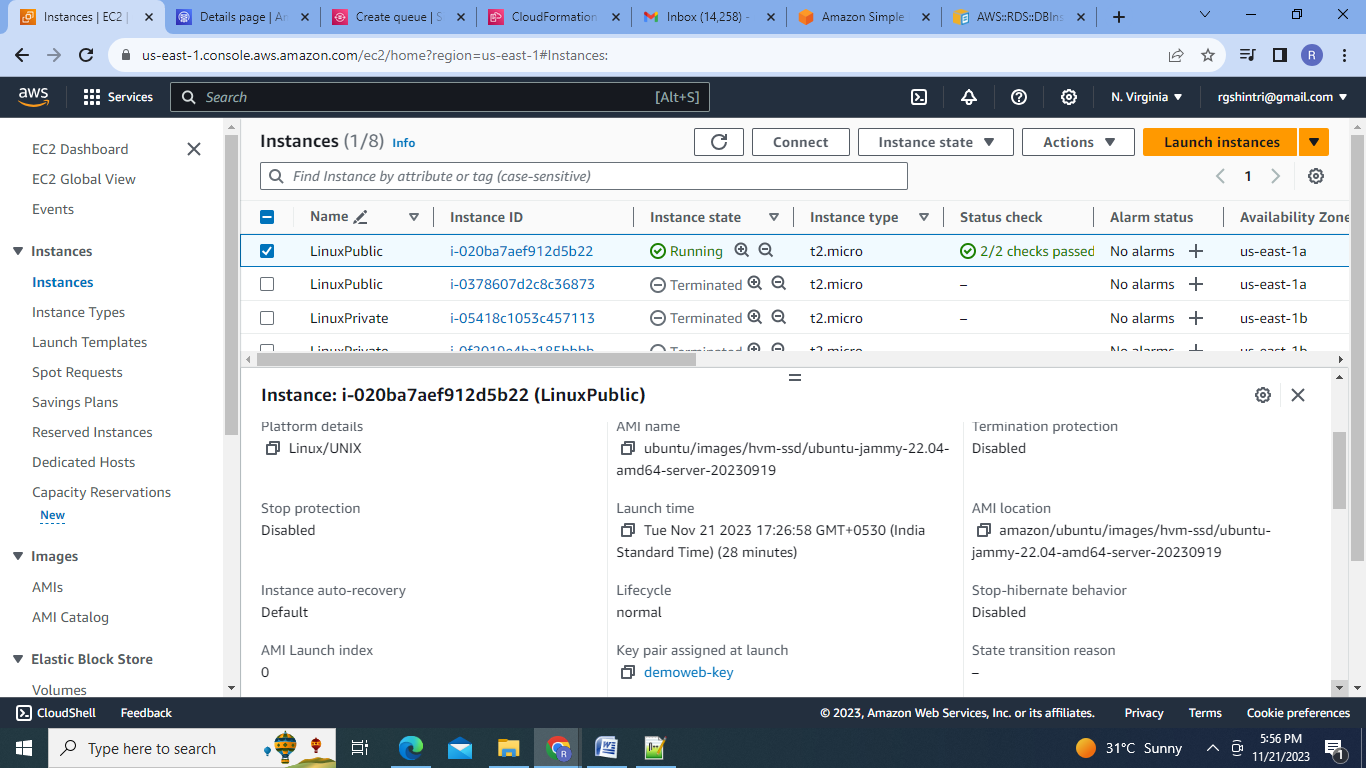
4.Name the stack, select the keypair & Select the notification Topic (SNS) Create the stack.

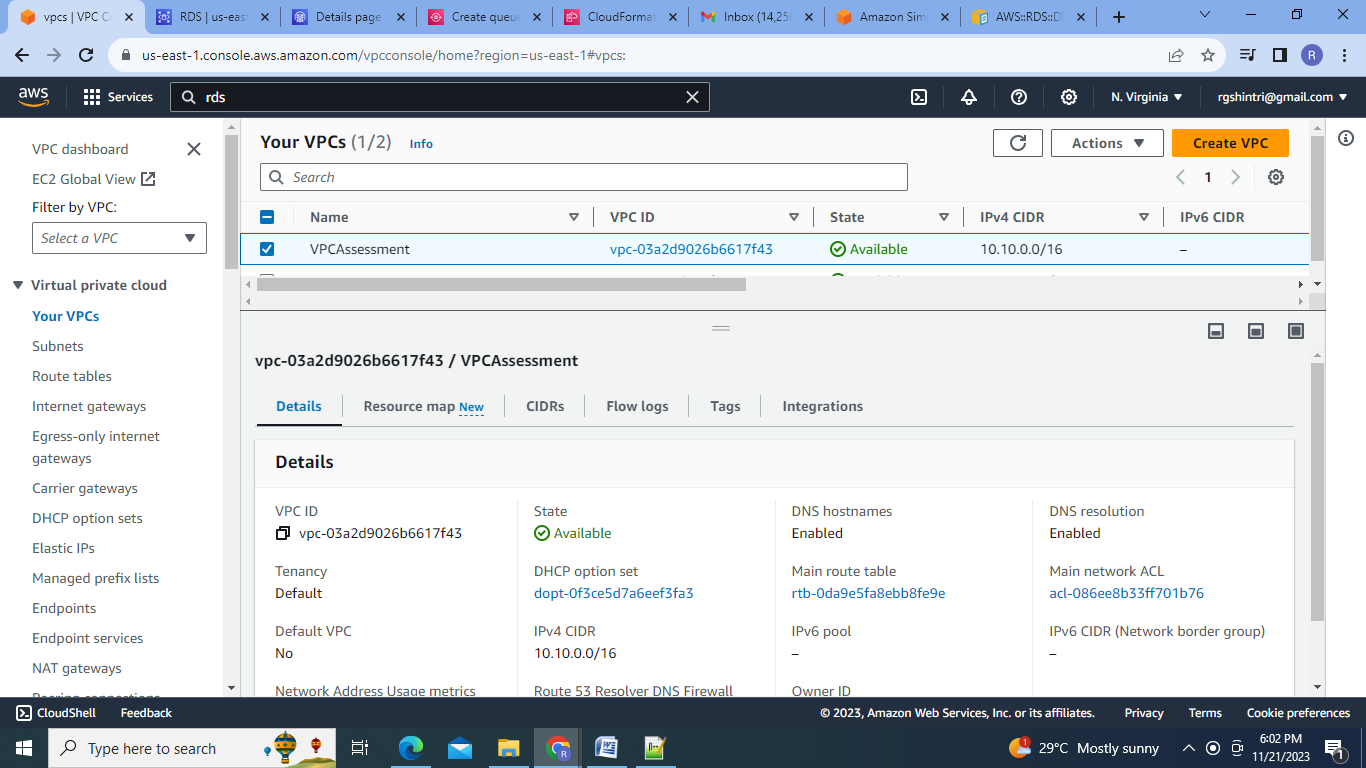


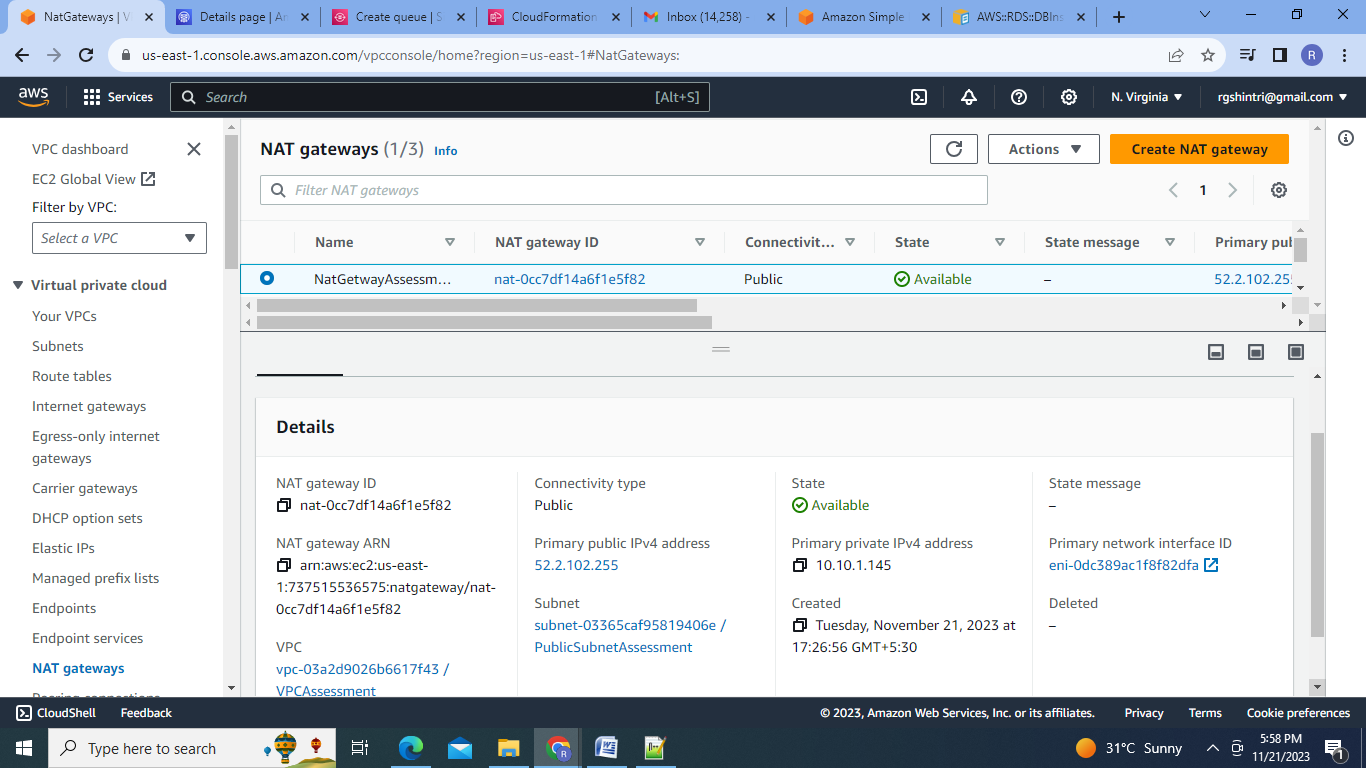


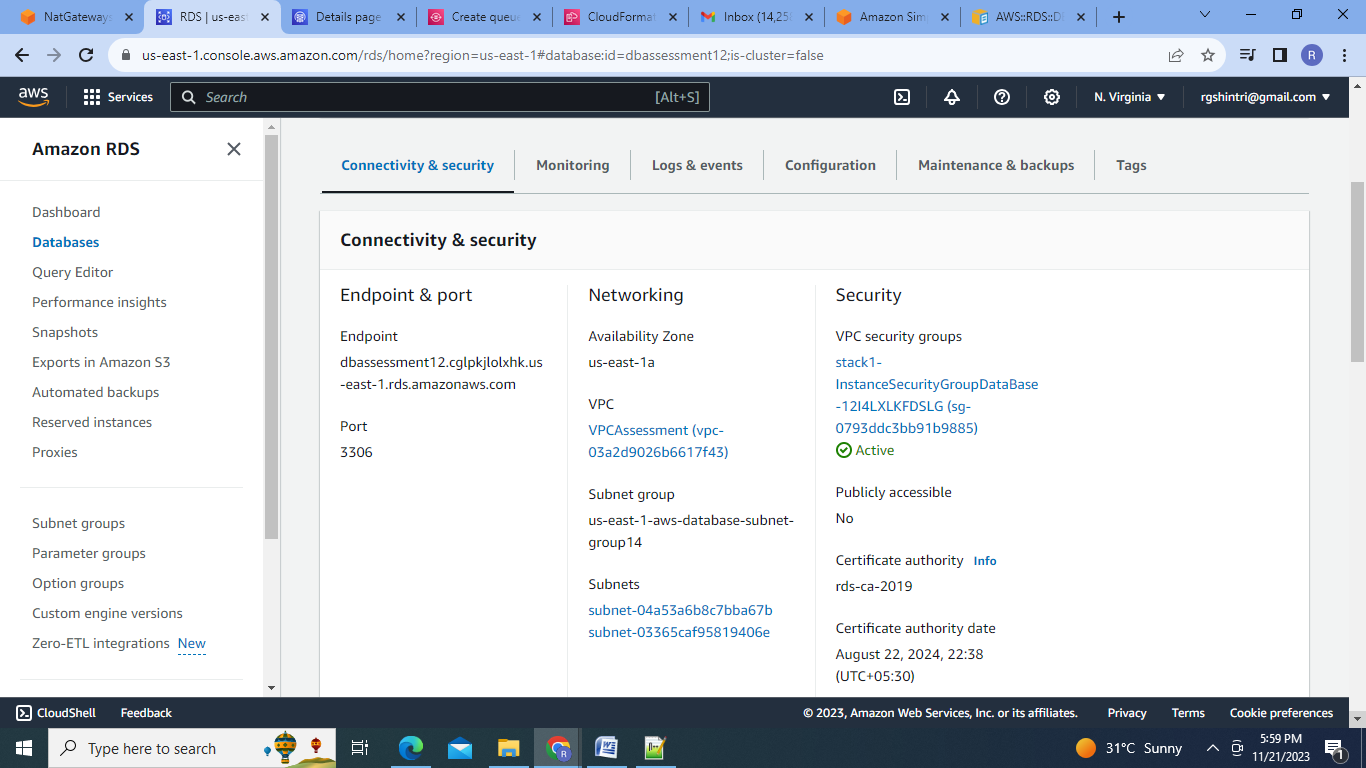




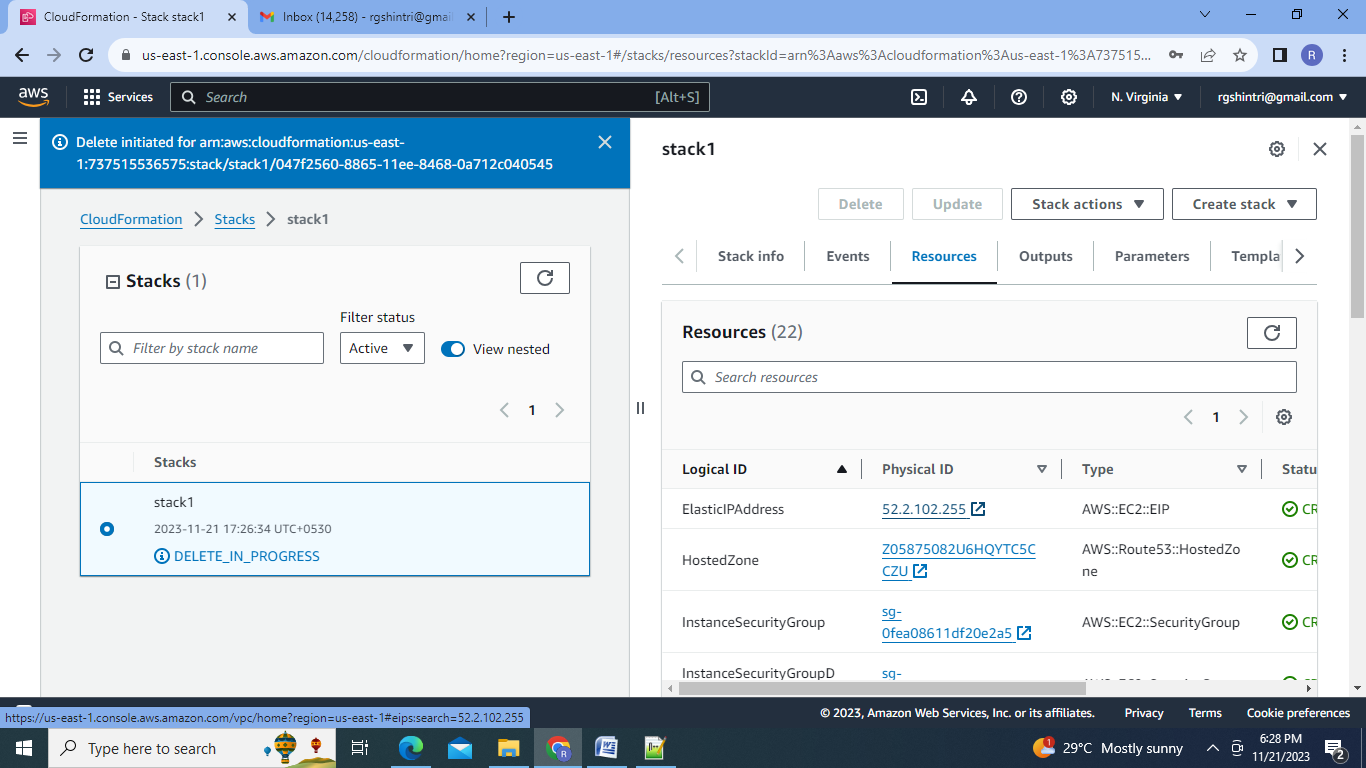


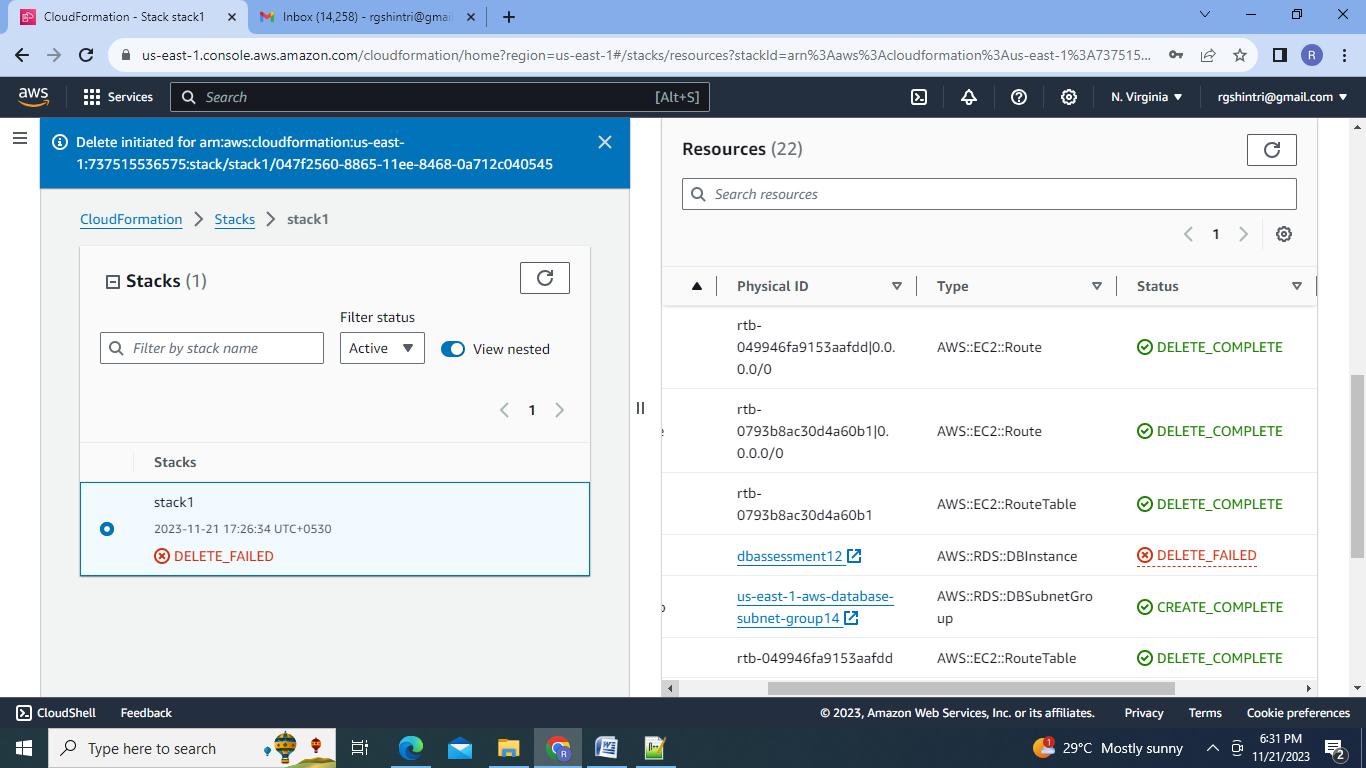






5.Deleteing the stack



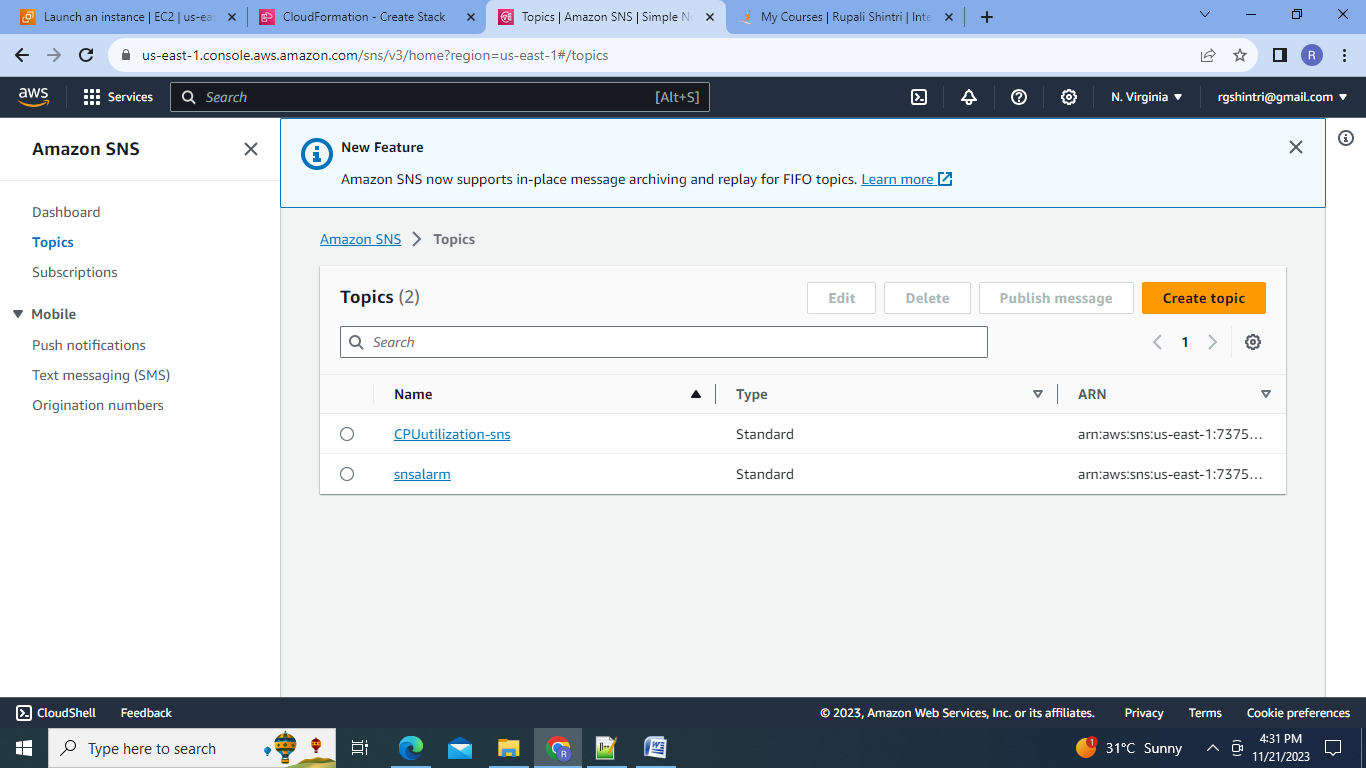


1. **Tasks To Be Performed**:

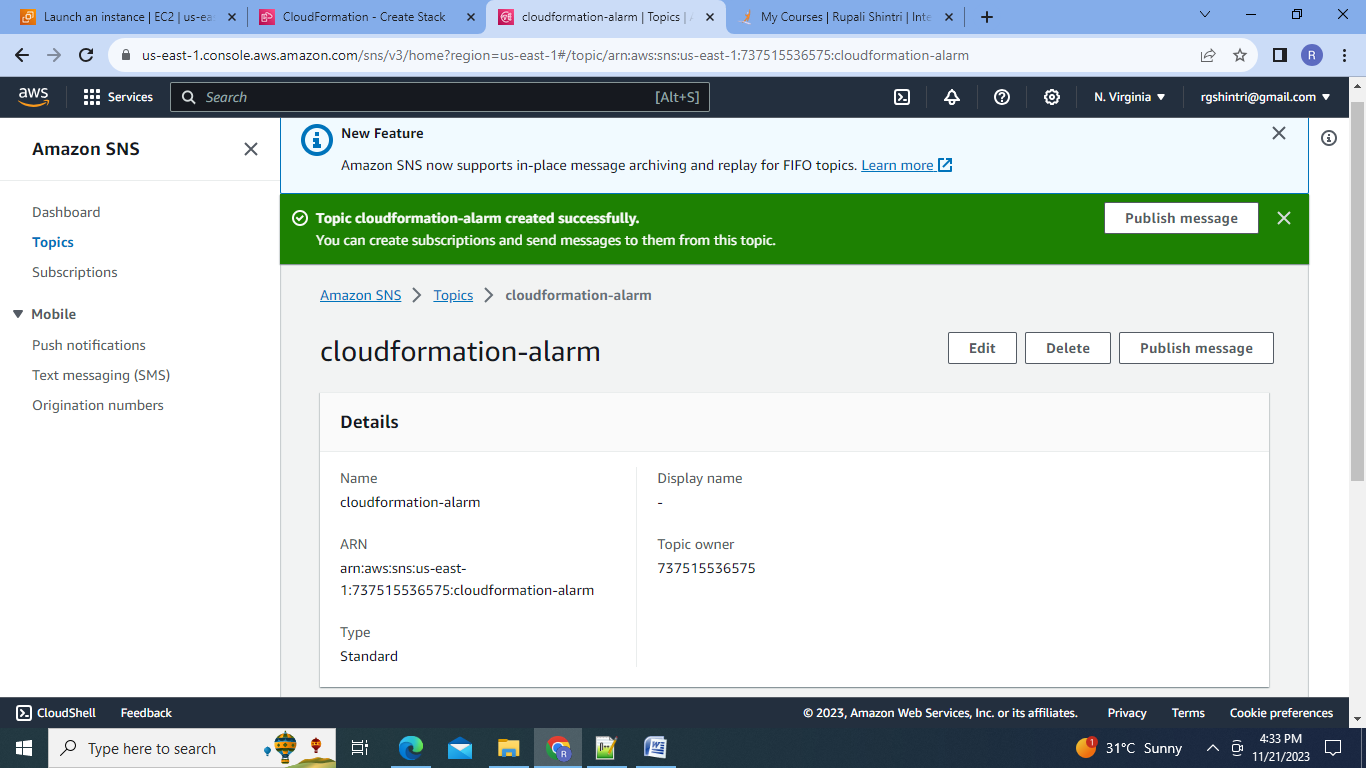
1. Use the template from CloudFormation task 1.

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.

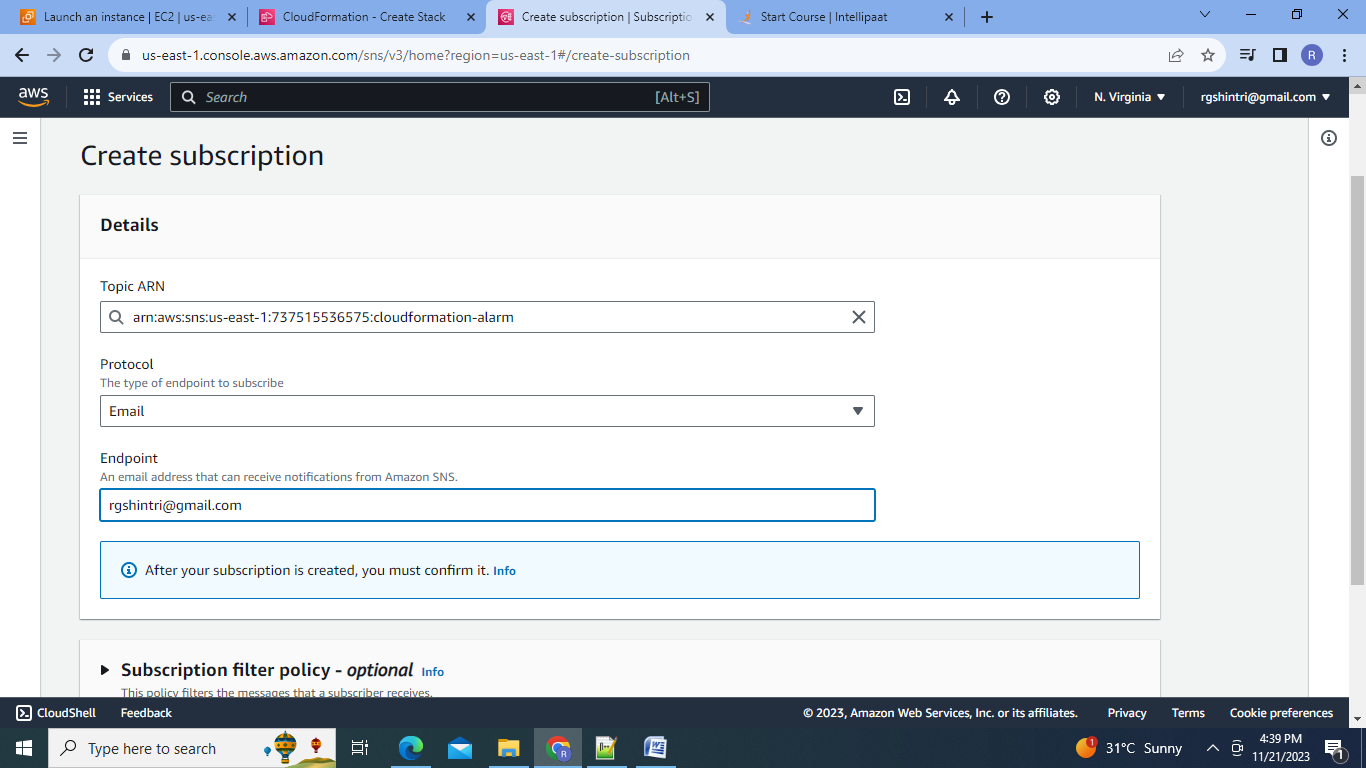
Steps:

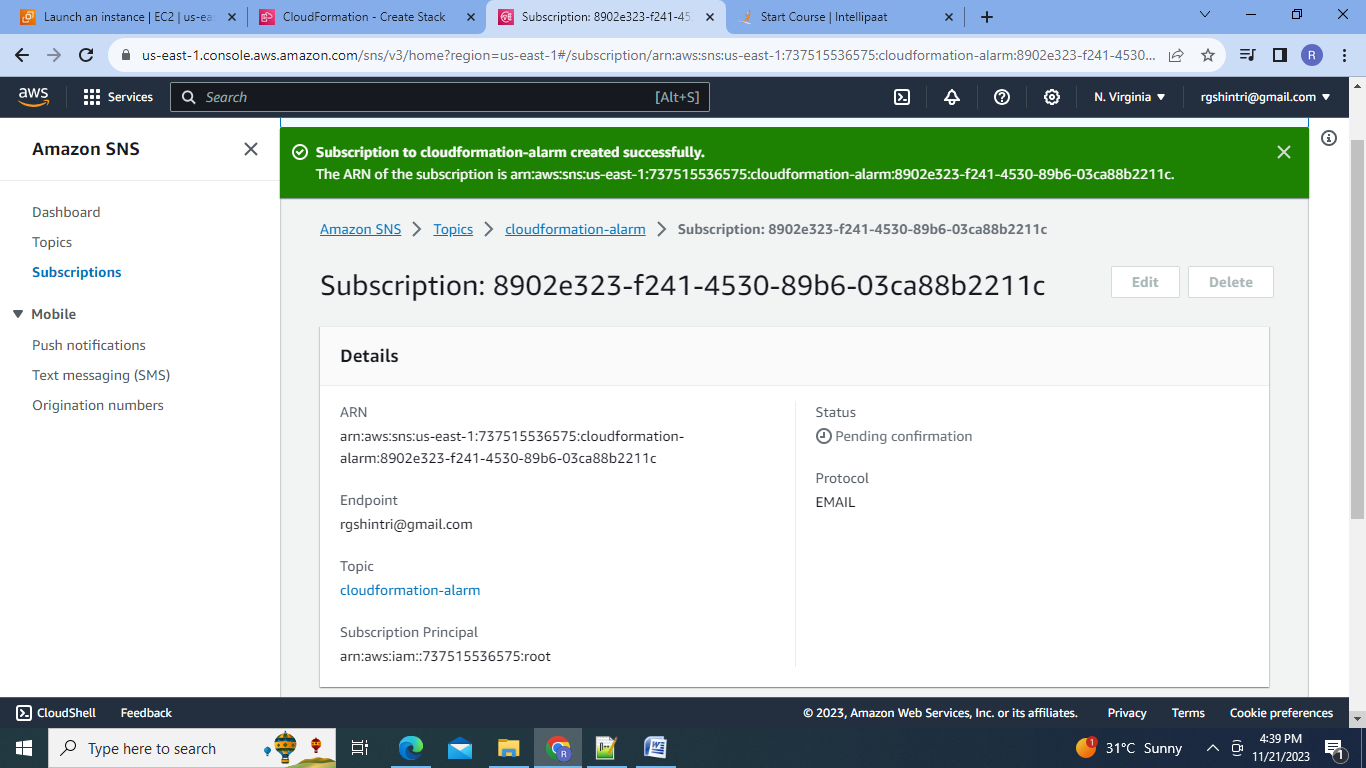


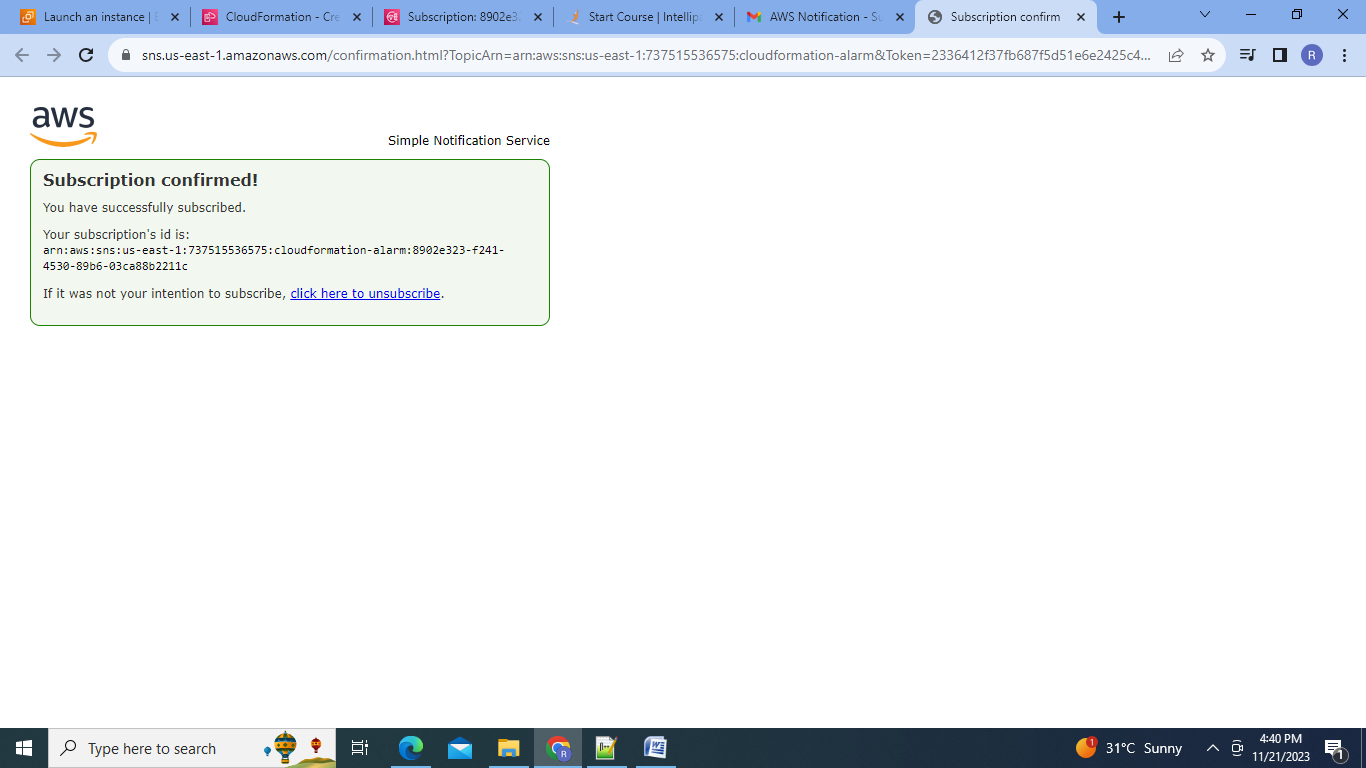
1.Go to SNS service and create Standard Topic:

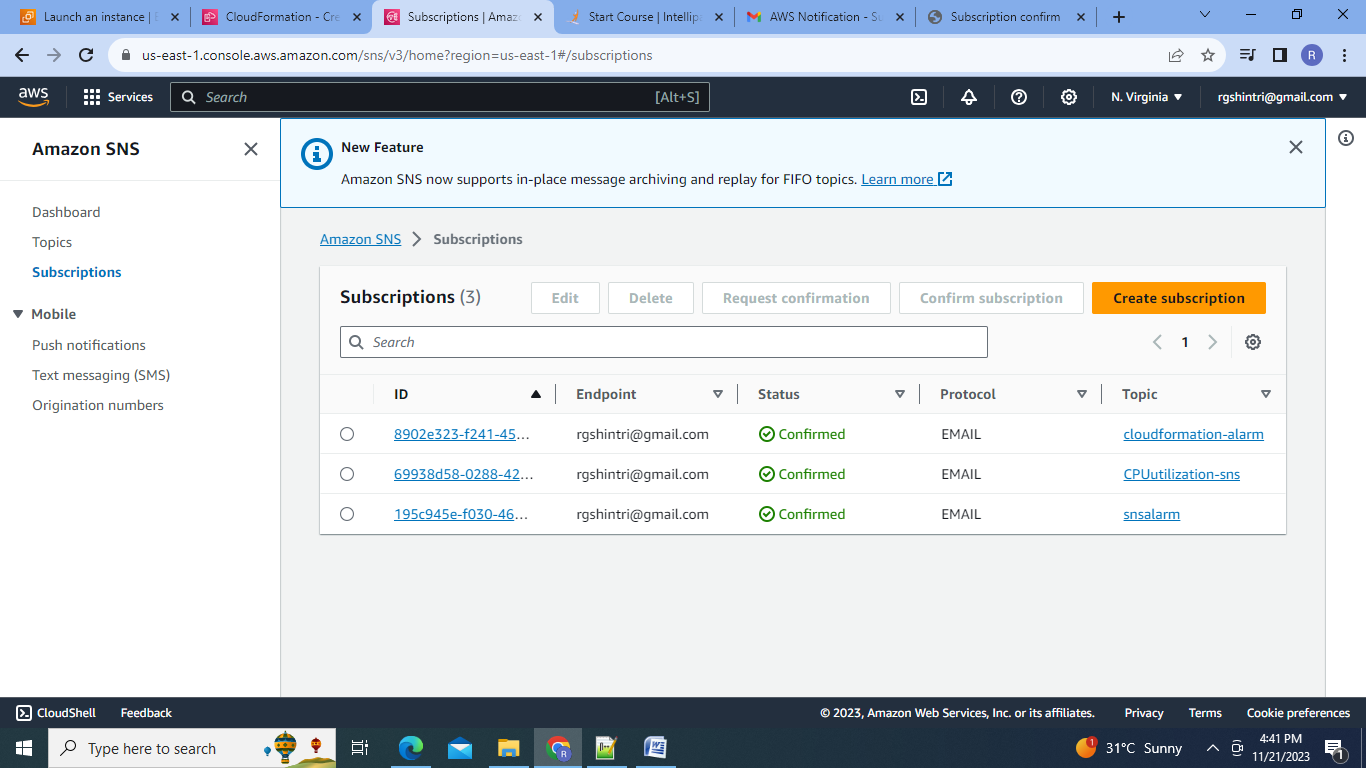


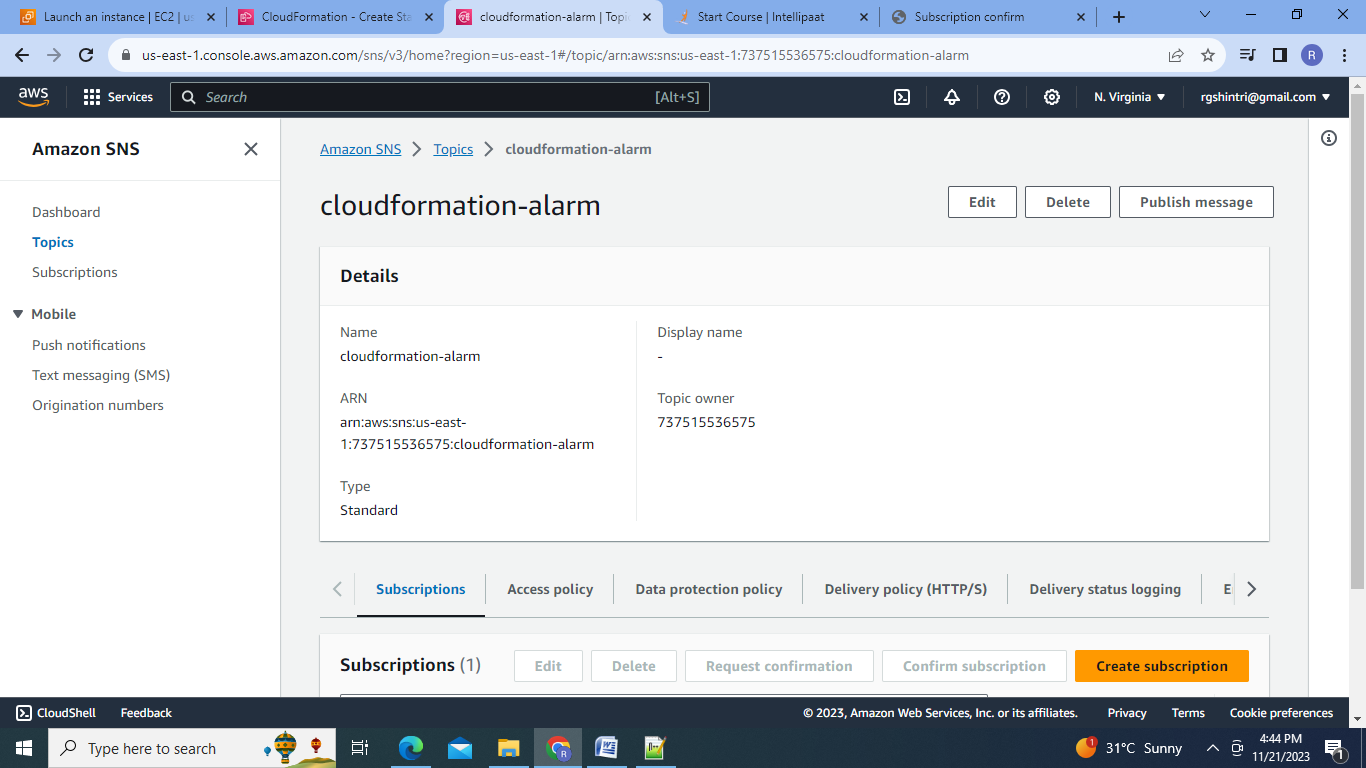
2.Create subscription:

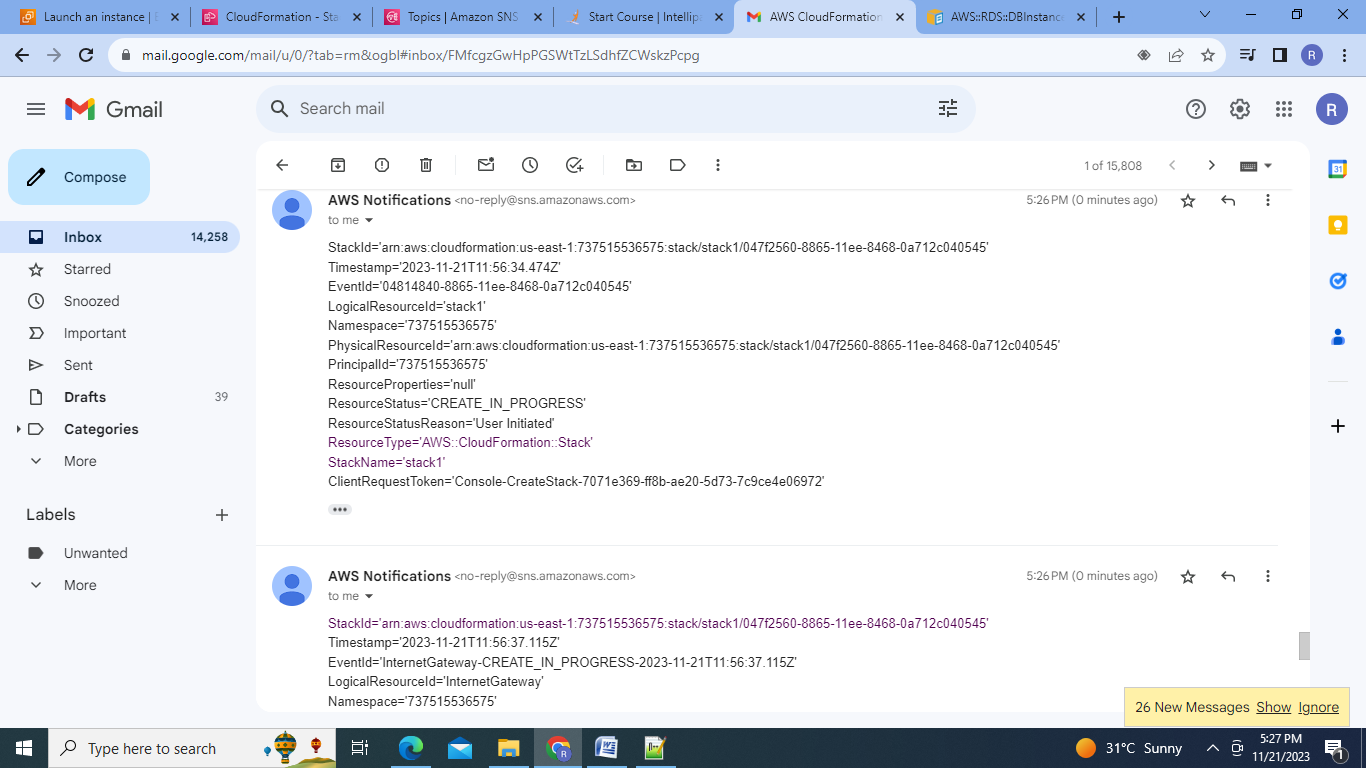


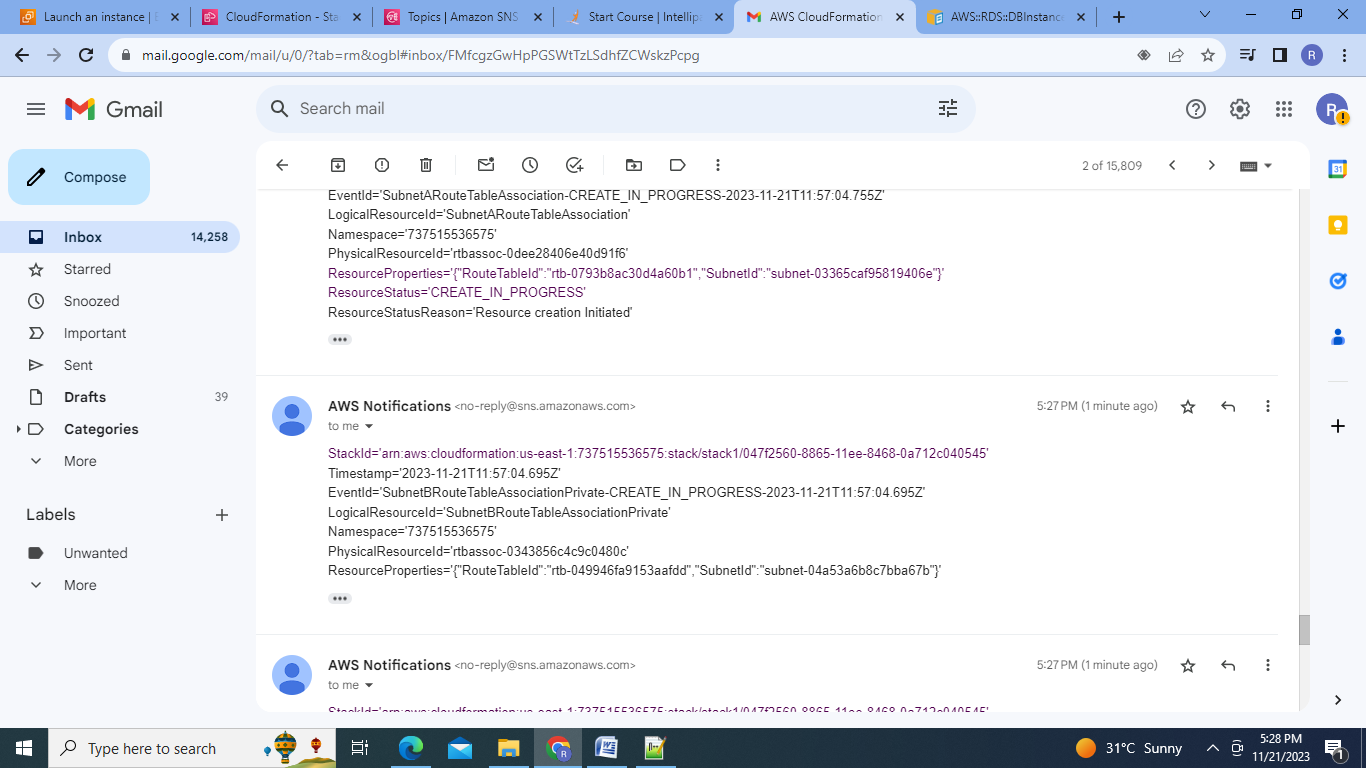


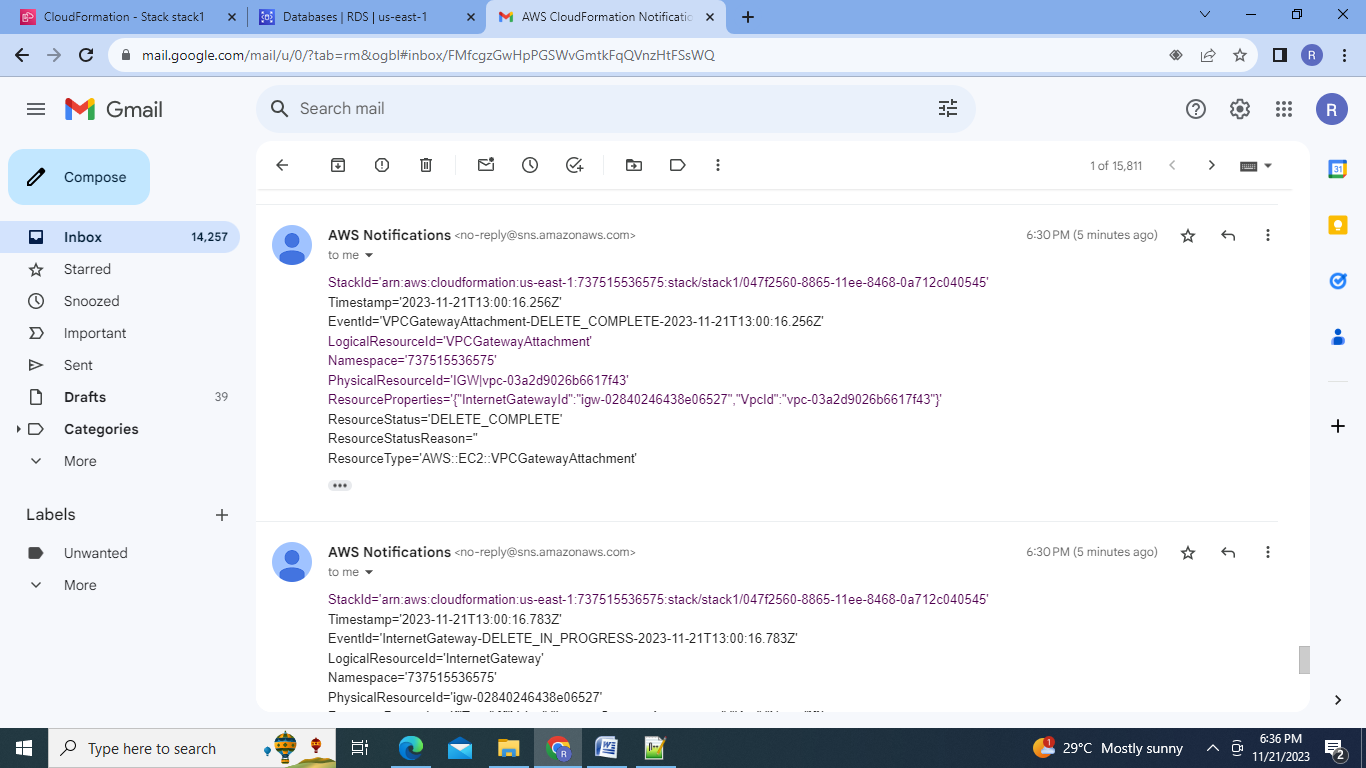










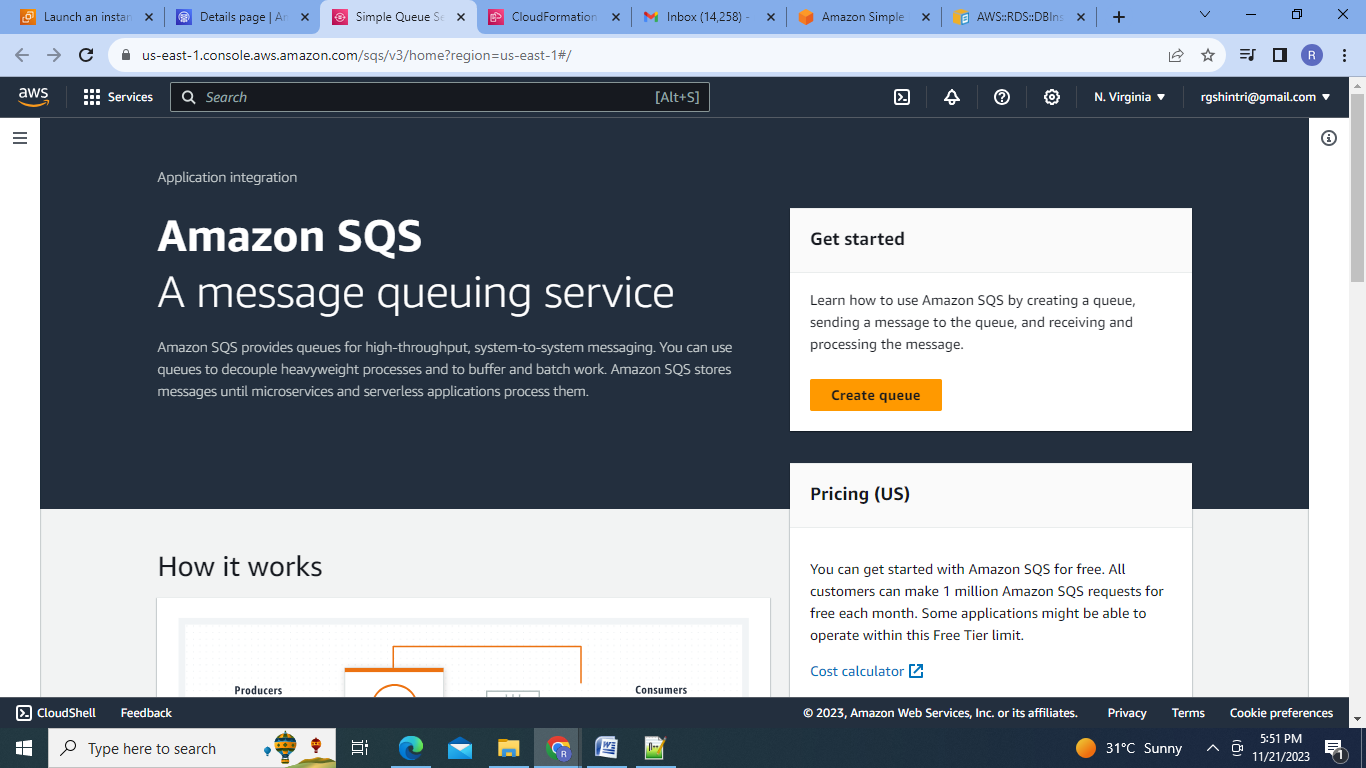


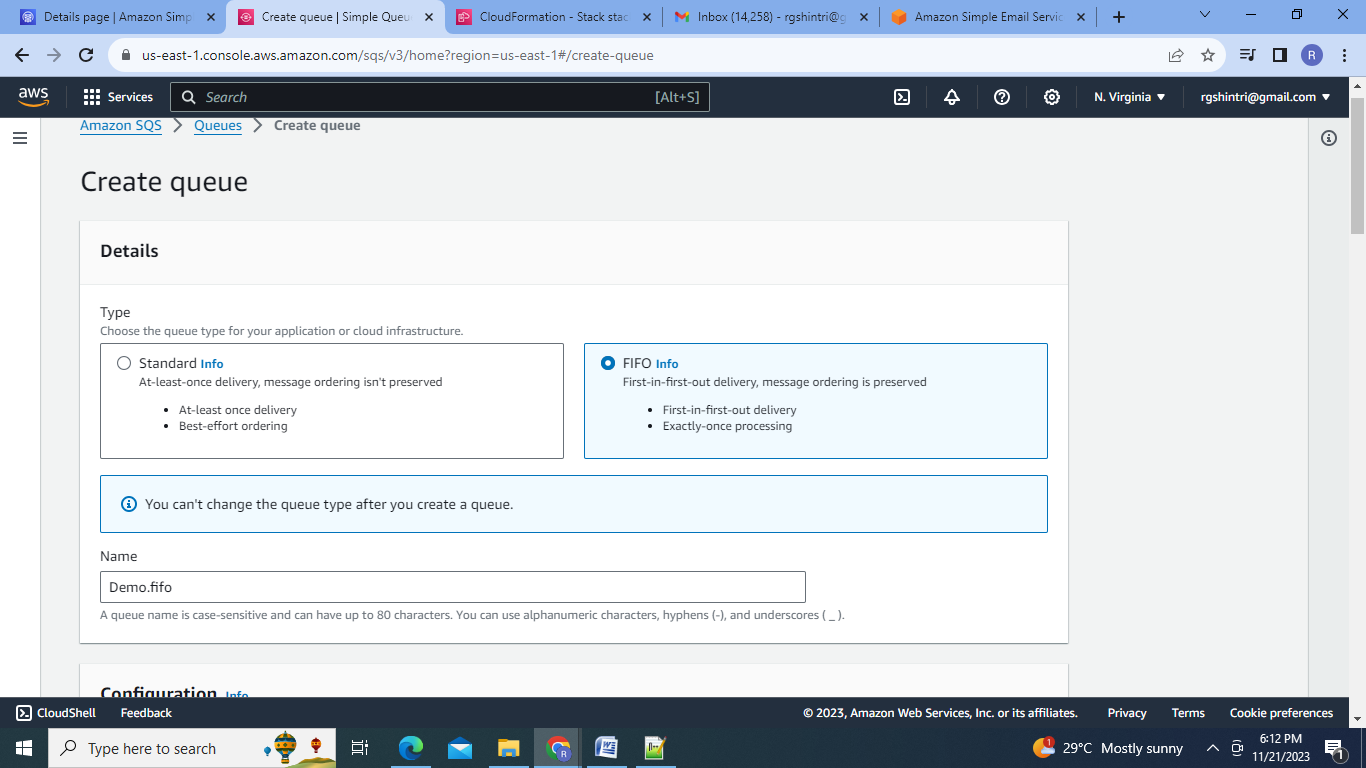
**C.Tasks To Be Performed**:

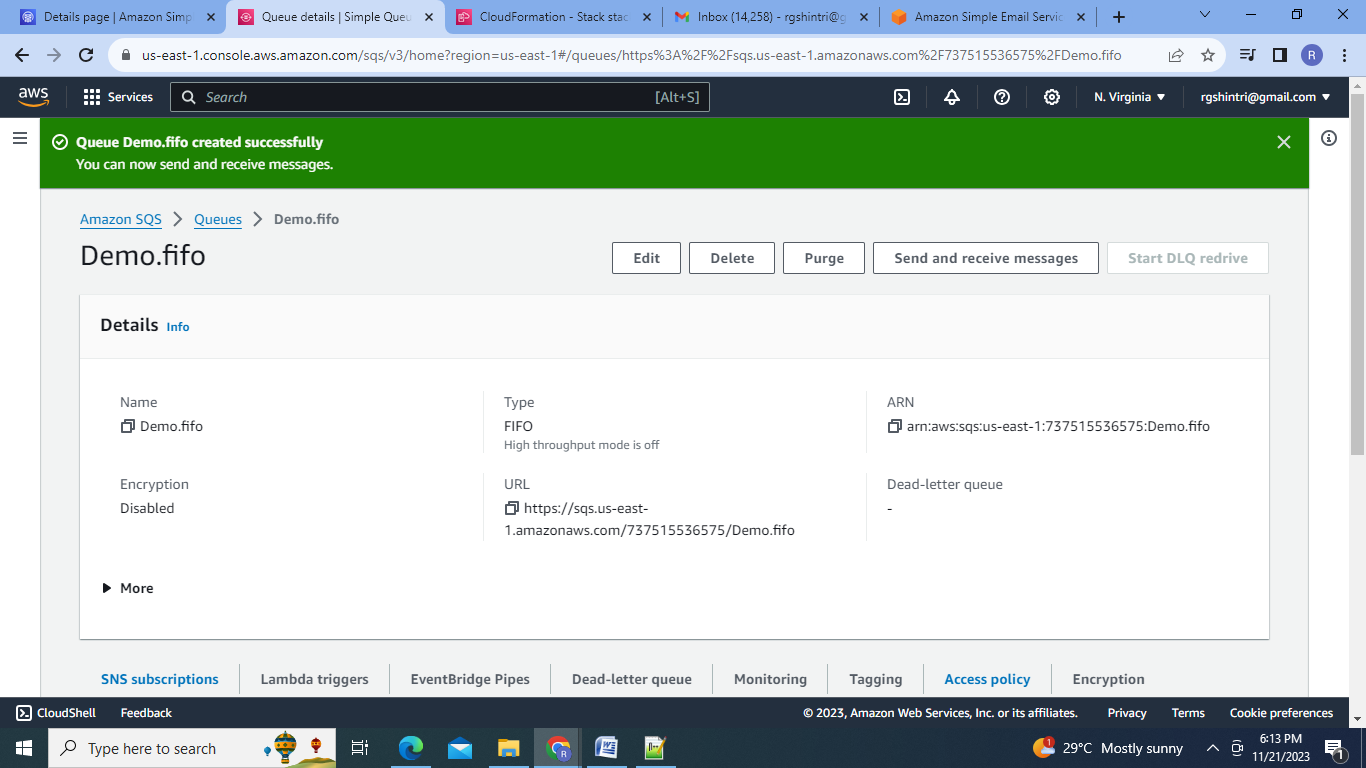
1.Create a FIFO SQS queue and test by sending messages.

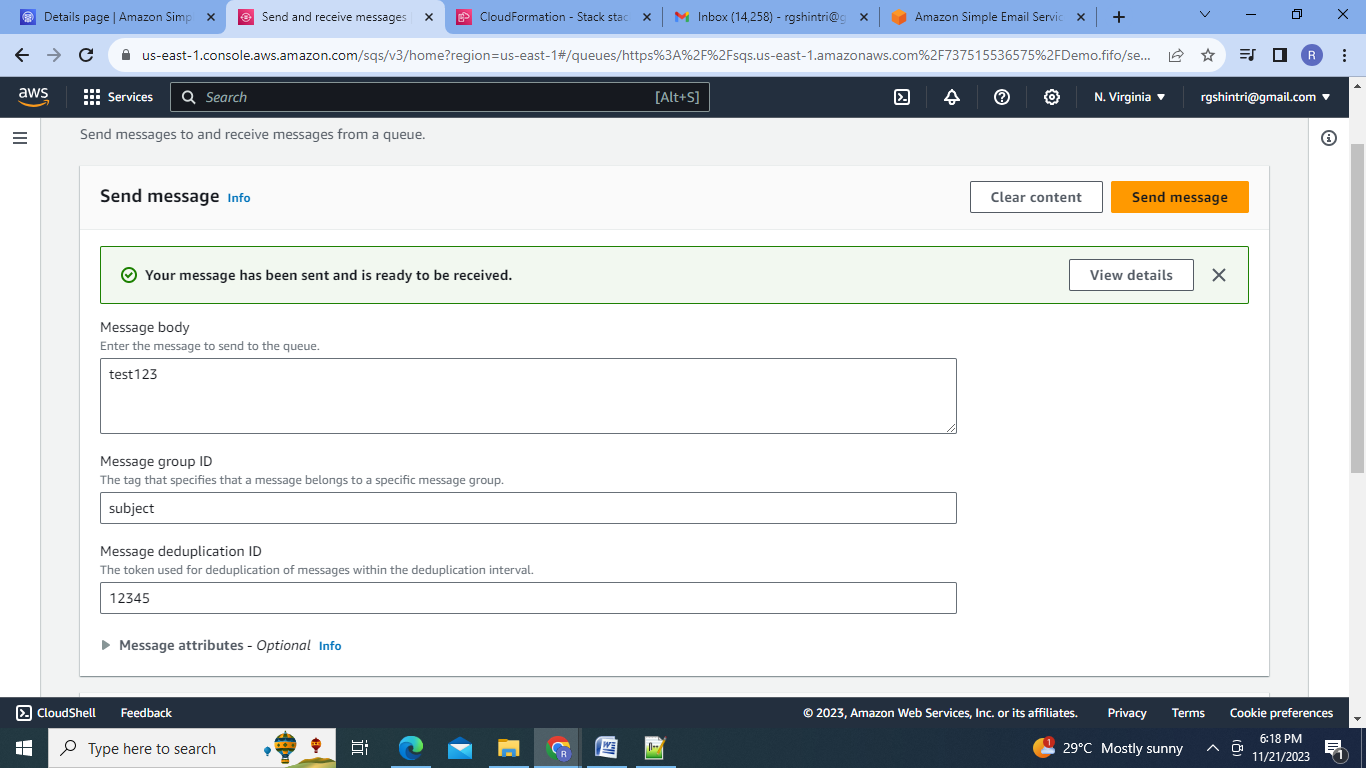
Steps:

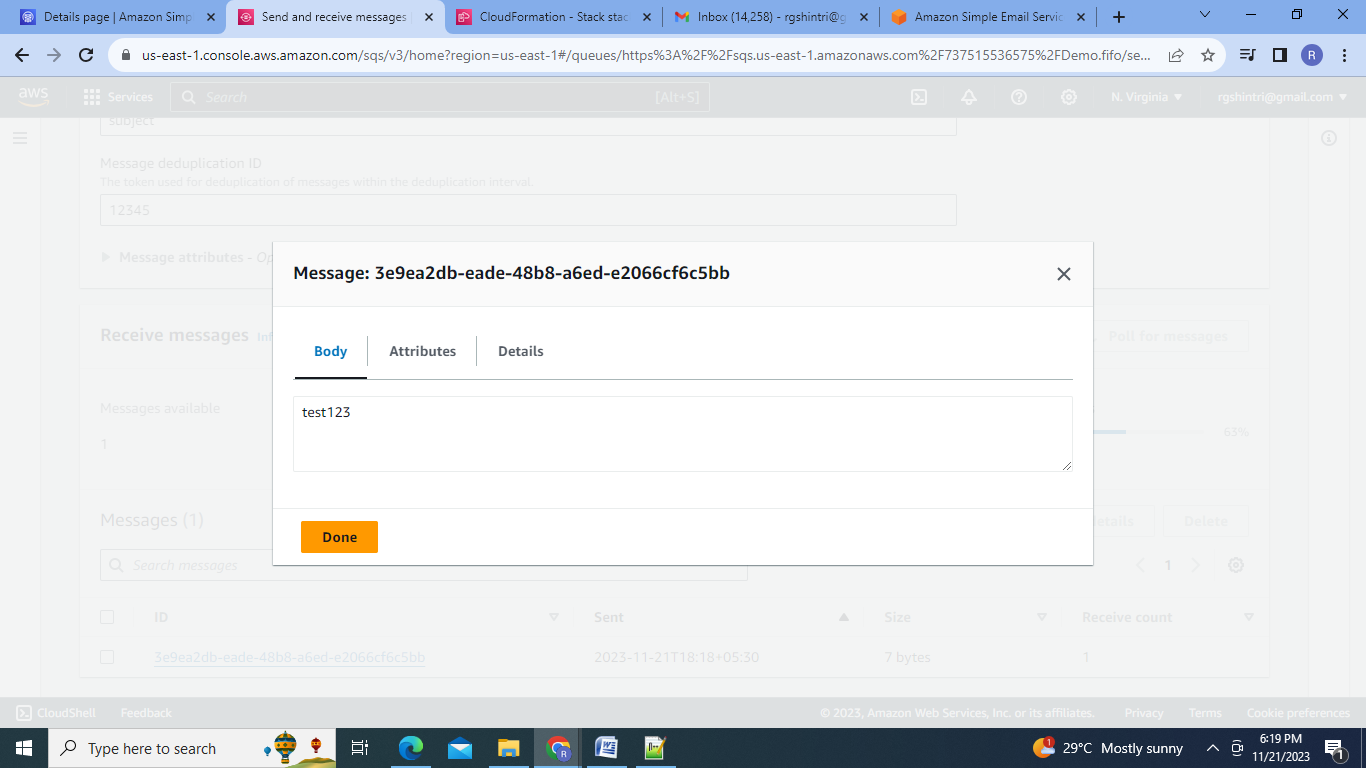
1.Go to SQS service:



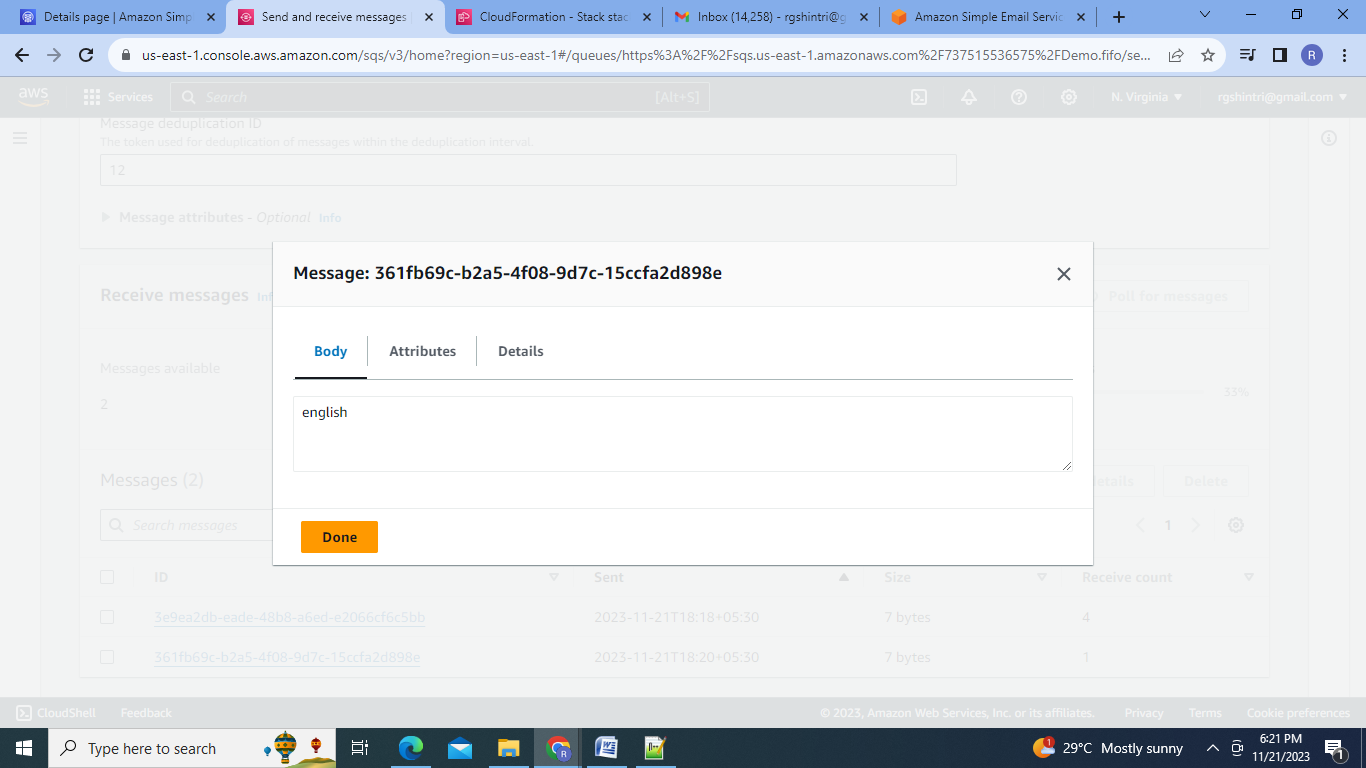








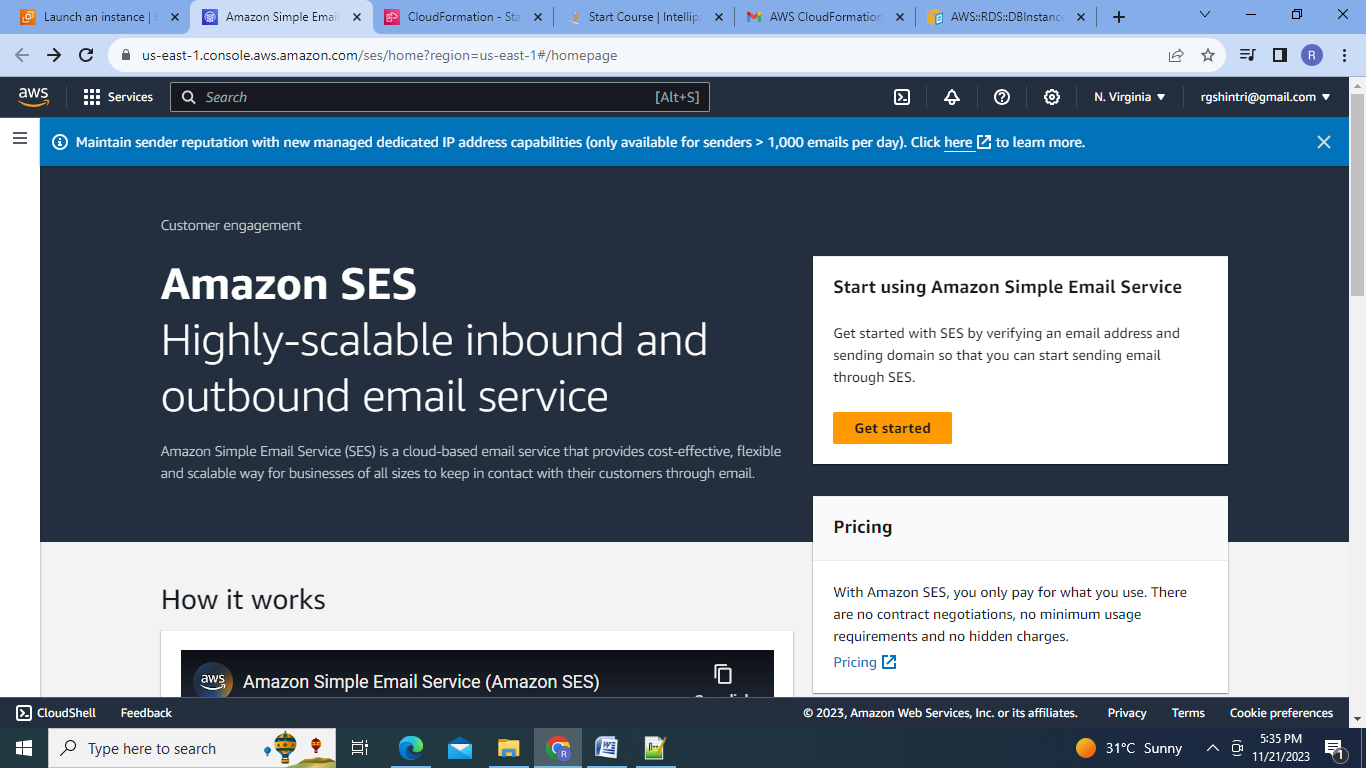


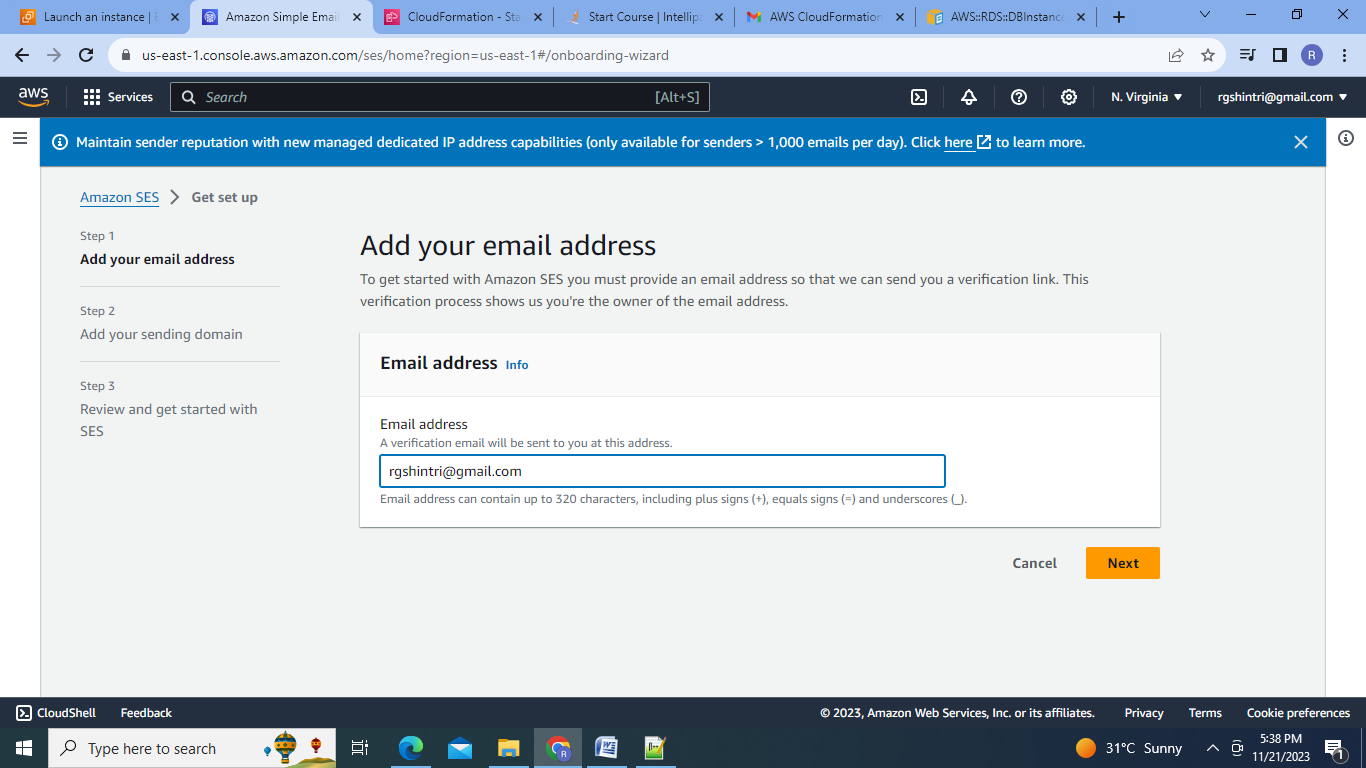


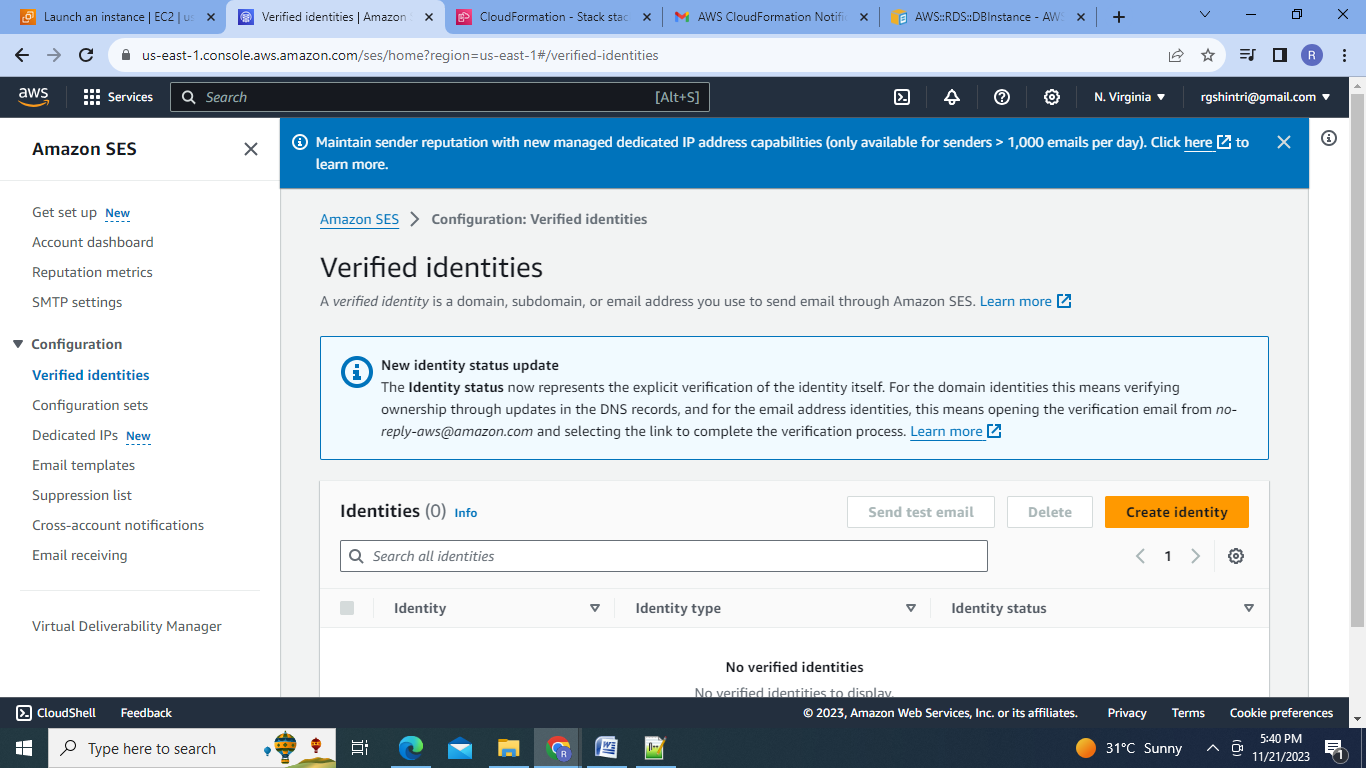
2.Register your mail in SES and send a test mail to yourself

Steps:

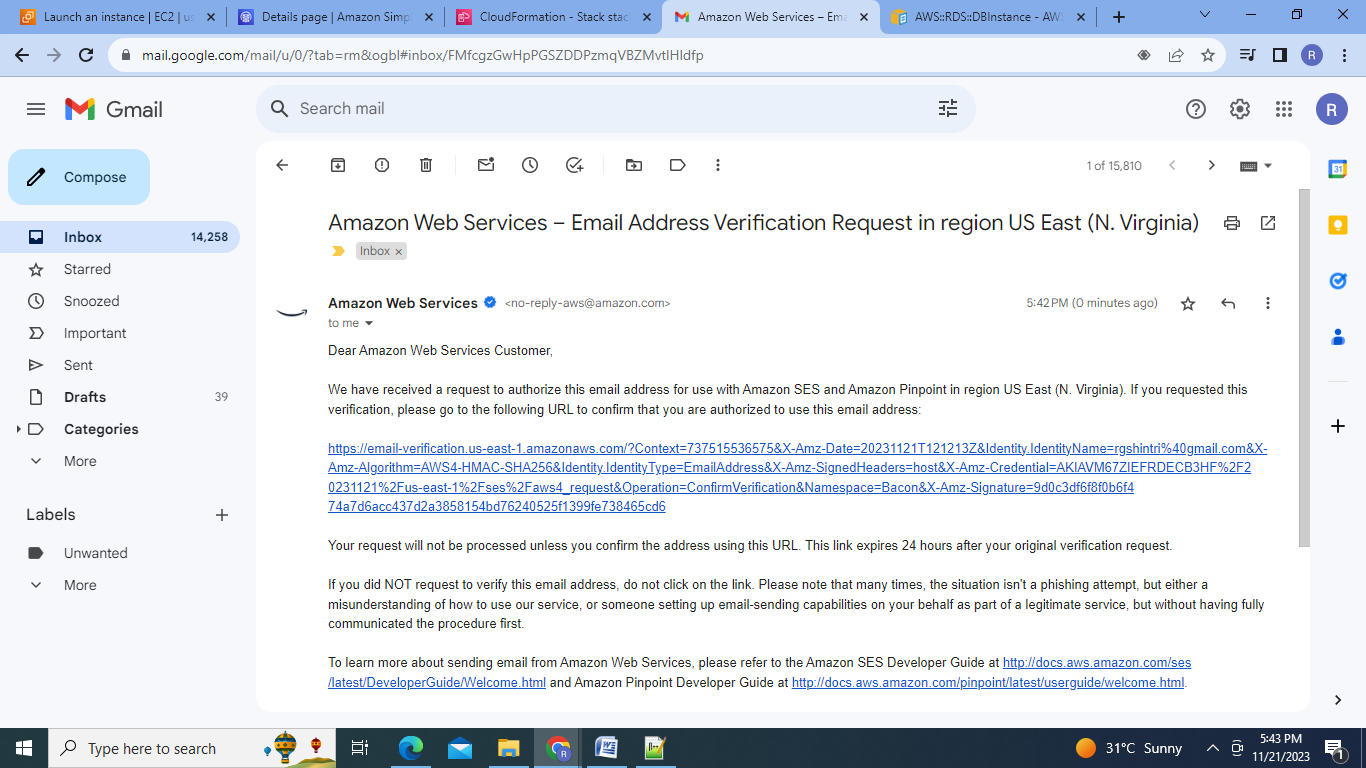
1. Go to SES service:

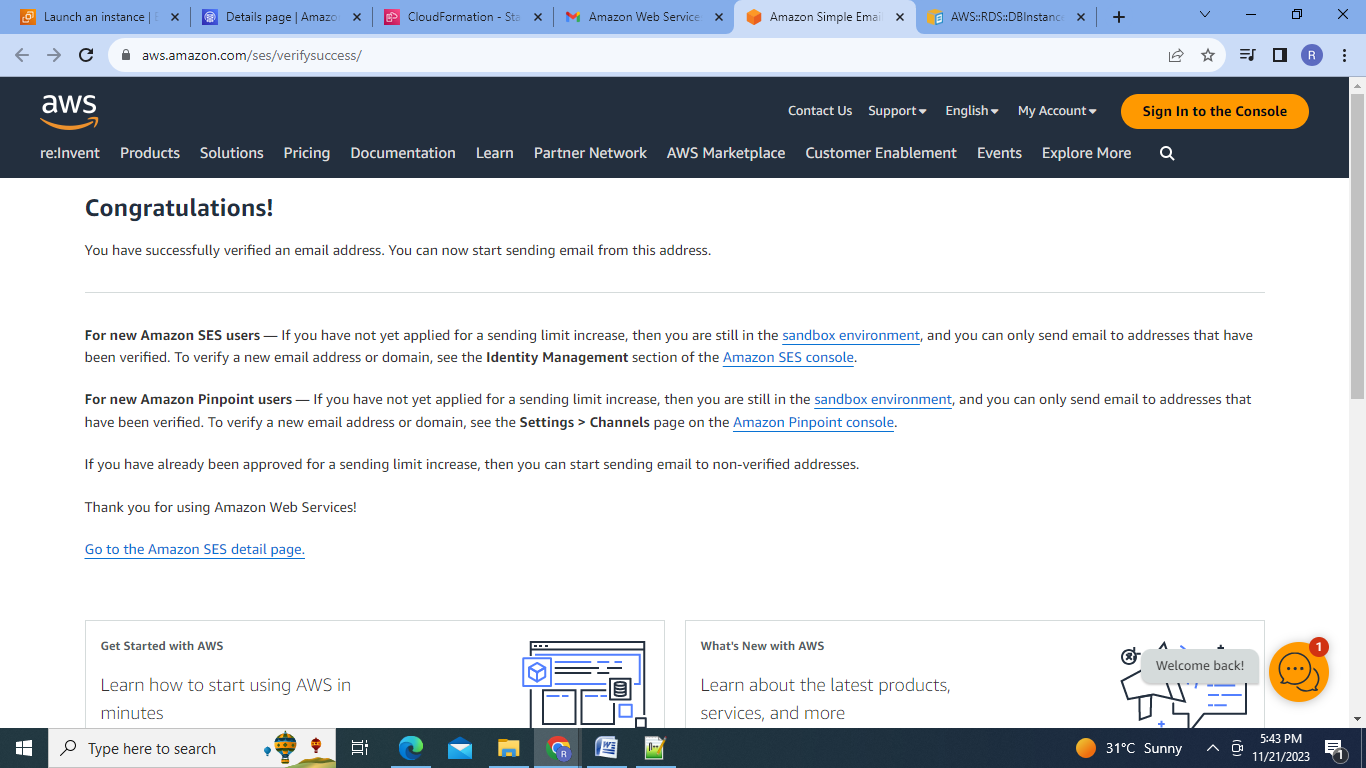


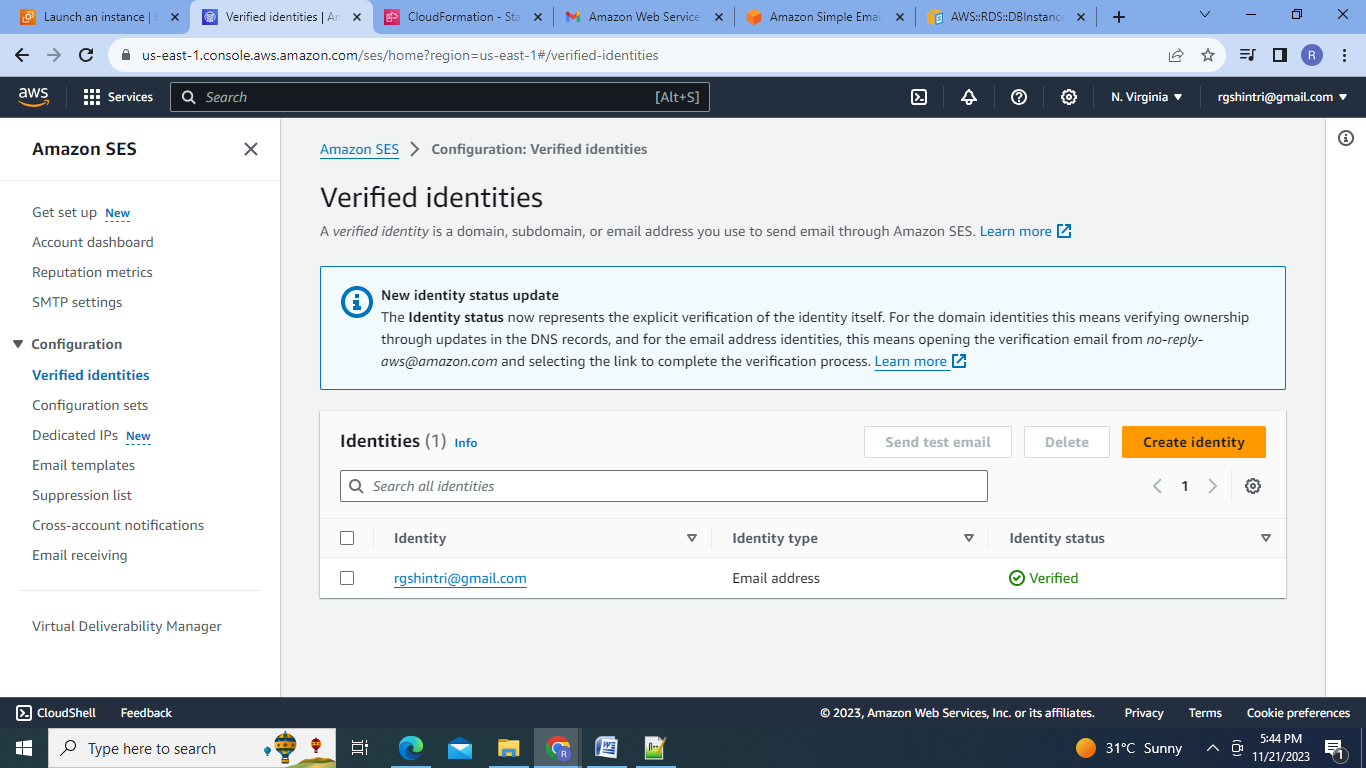


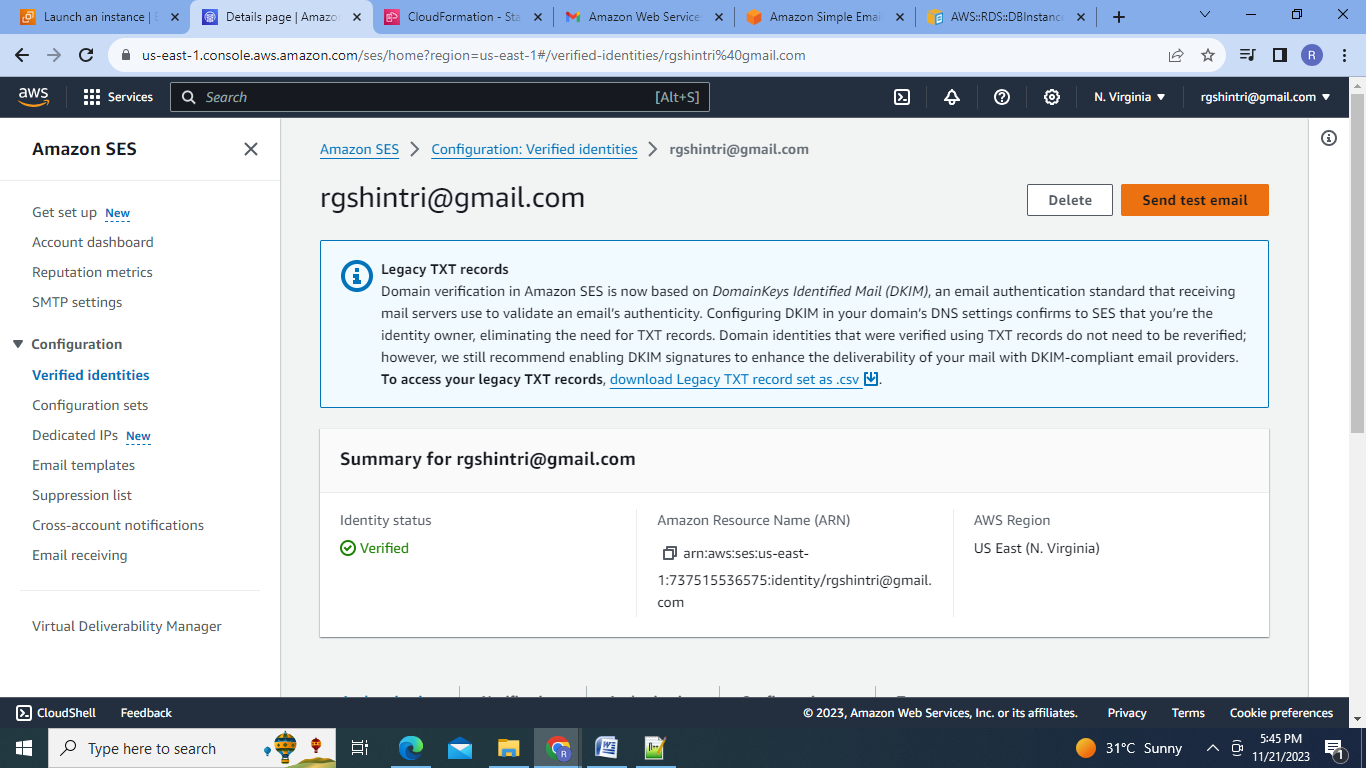


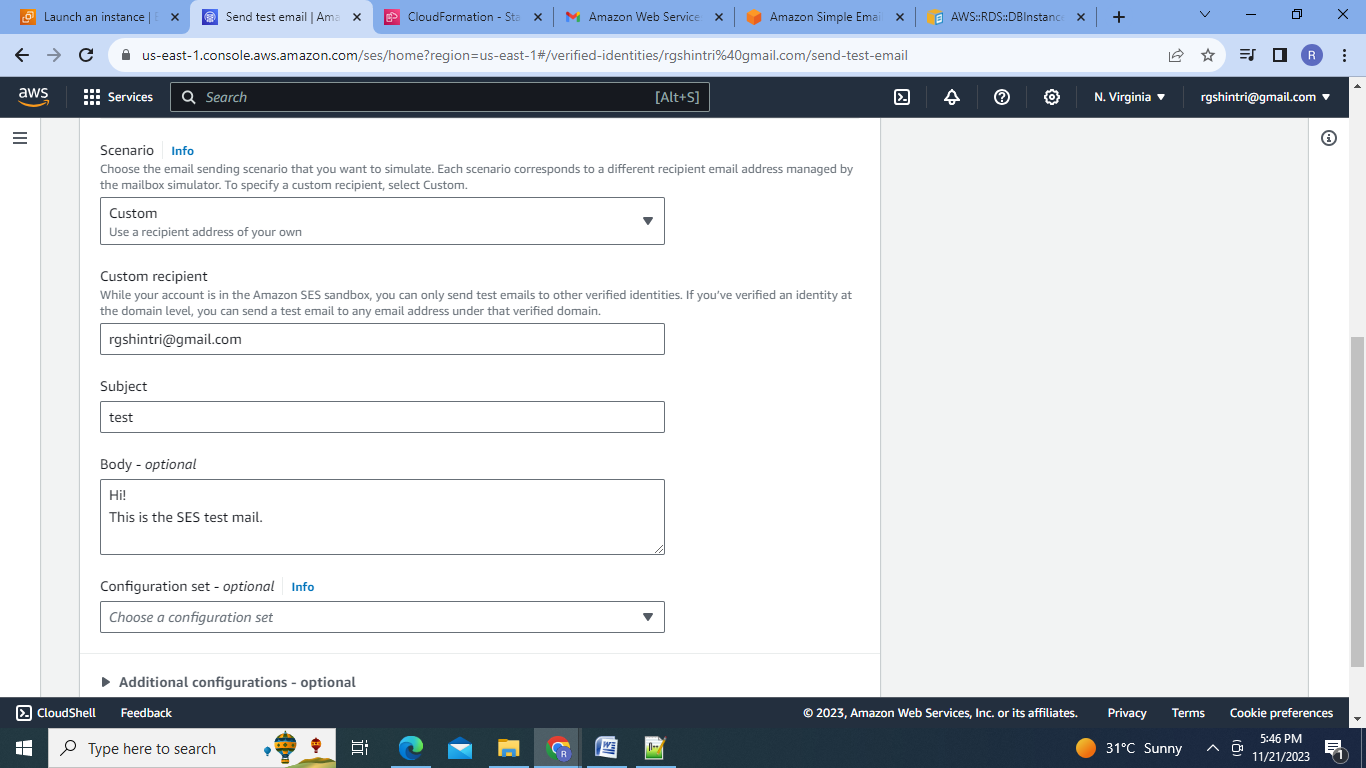


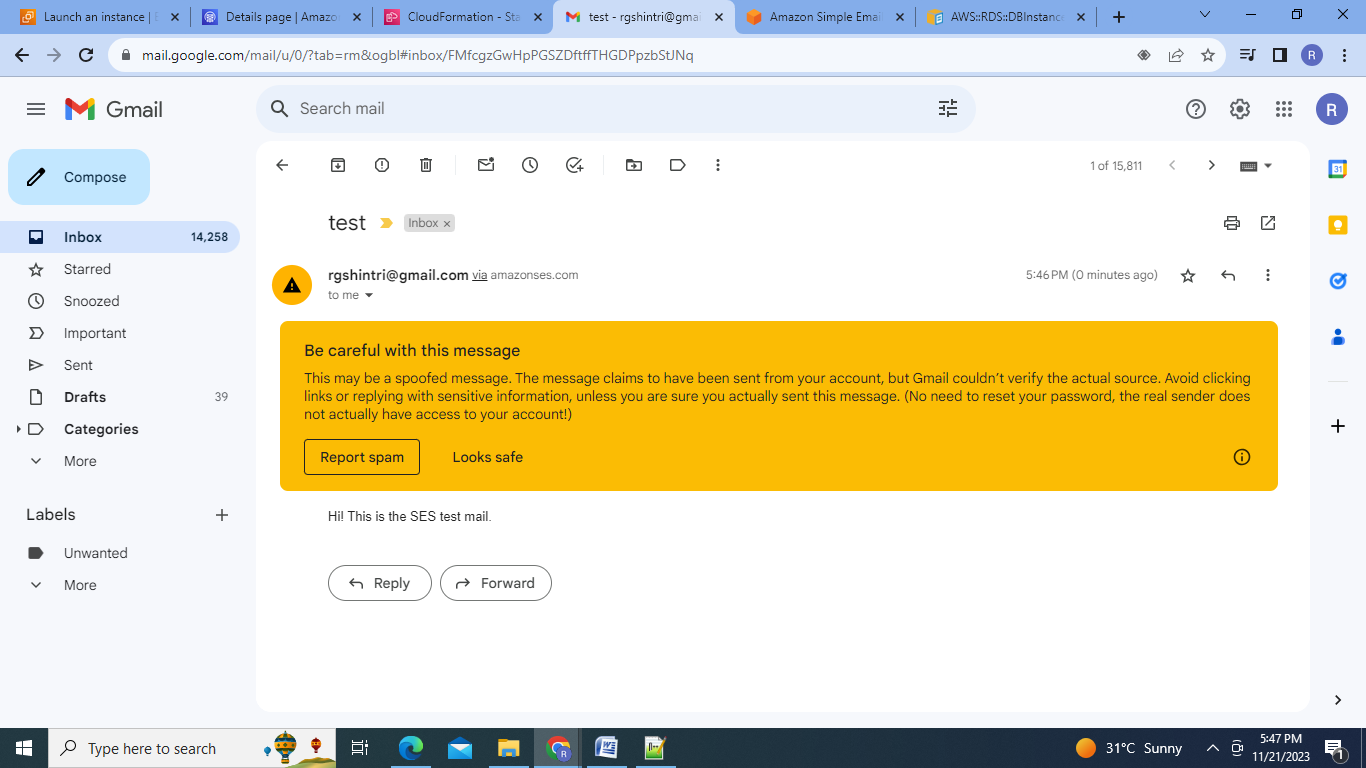












Conclusion : 1.Using Cloudformation we can create the multiple resources in one go and we can automate the repeated processes.2.SNS gives the notifications regarding execution of each step in cloudformation .3.SQS allows us to send and receive multiple messages in FIFO manner.4. SES is the cloud based reliable and cost effective email service .

