Problem Statement:

You work for XYZ Corporation. Your corporation wants to launch a new web-based application. The development team has prepared the code but it is
not tested yet. The development team needs the system admins to build a web server to test the code but the system admins are not available.

Tasks To Be Performed:

1. Web tier: Launch an instance in a public subnet and that instance should allow HTTP and SSH from the internet.

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.

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.

4. Setup a Route 53 hosted zone and direct traffic to the EC2 instance.

Step-By-Step Procedure:

Step1: Using **AWS CloudFormation**, we create a **multi-tier EC2 architecture**. First, write the template in any **code editor** and save the file. The code below represents the **CloudFormation template used for this assignment**.

Casestudy_template.yml

```
AWSTemplateFormatVersion: '2010-09-09'

Parameters:
  AmiId:
    Type: String
    Description: "AMI ID for the EC2 instance"

  KeyName:
    Type: 'AWS::EC2::KeyPair::KeyName'
    Description: "Name of an existing EC2 KeyPair to enable SSH access."
```

```
  DBName:
    Type: String
    Default: "testdb"
    Description: "The name of the database to be created in the RDS instance."

  DBUser:
    Type: String
    Default: "dbadmin"
    Description: "The database admin account username."

  DBPassword:
    Type: String
    Description: "The database admin account password (must be at least 8
chars)."
    MinLength: 8

Resources:
  MyVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags: [{Key: Name, Value: Dev-VPC}]

  InternetGateway:
    Type: AWS::EC2::InternetGateway

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


  PublicSubnetWeb:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref MyVPC
```

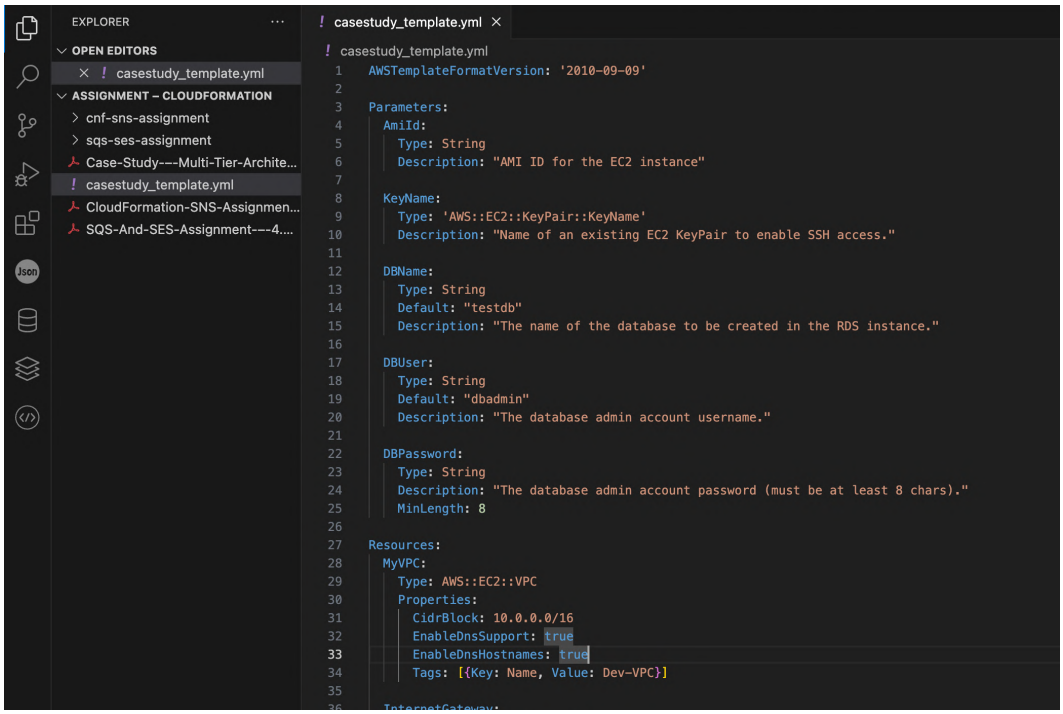
```
      CidrBlock: 10.0.1.0/24
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      MapPublicIpOnLaunch: true
      Tags: [{Key: Name, Value: Web-Tier-Public}]

  PrivateSubnetApp:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref MyVPC
      CidrBlock: 10.0.2.0/24
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      Tags: [{Key: Name, Value: App-Tier-Private}]

  PrivateSubnetDB:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref MyVPC
      CidrBlock: 10.0.3.0/24
      AvailabilityZone: !Select [ 1, !GetAZs '' ] # Different AZ for RDS High
Availability
      Tags: [{Key: Name, Value: DB-Tier-Private}]


  MyDBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: Subnets for RDS Instance
      SubnetIds:
        - !Ref PrivateSubnetApp
        - !Ref PrivateSubnetDB


  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref MyVPC

  DefaultPublicRoute:
    Type: AWS::EC2::Route
```

```
  DependsOn: VPCGatewayAttachment
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PublicSubnetRouteAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PublicSubnetWeb
    RouteTableId: !Ref PublicRouteTable


WebSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow HTTP and SSH from Internet
    VpcId: !Ref MyVPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0

AppSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow SSH only from Web Tier
    VpcId: !Ref MyVPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        SourceSecurityGroupId: !Ref WebSecurityGroup
```

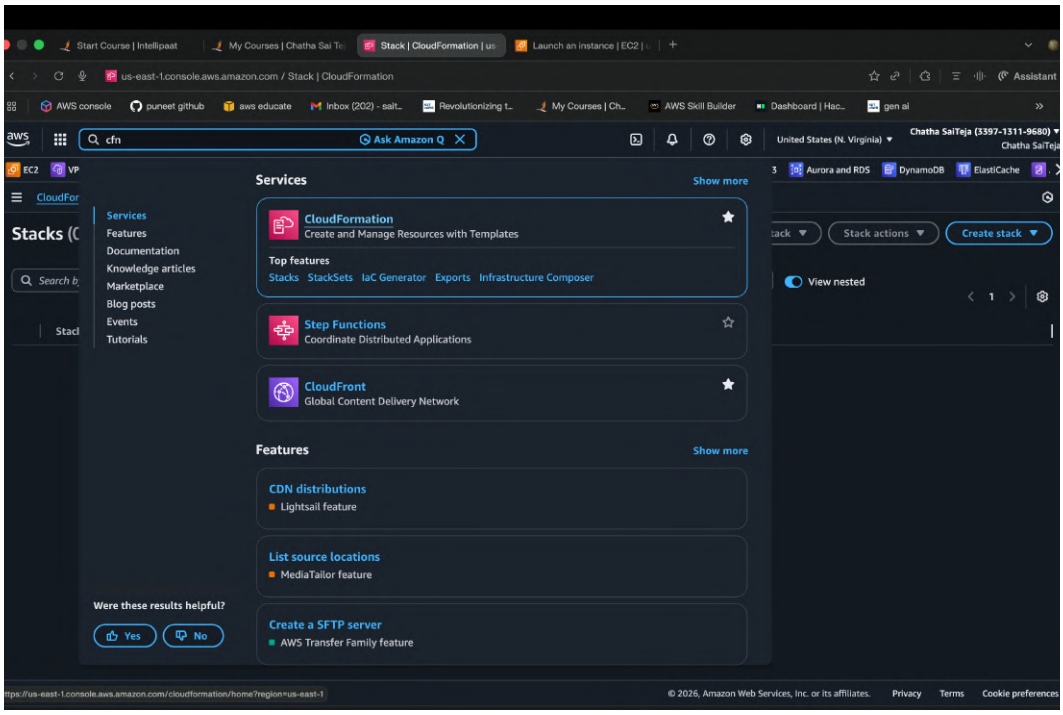
```
DBSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow MySQL from App Tier
    VpcId: !Ref MyVPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 3306
        ToPort: 3306
        SourceSecurityGroupId: !Ref AppSecurityGroup
WebInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref AmiId
    KeyName: !Ref KeyName
    InstanceType: t2.micro
    SubnetId: !Ref PublicSubnetWeb
    SecurityGroupIds: [!Ref WebSecurityGroup]

AppInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref AmiId
    KeyName: !Ref KeyName
    InstanceType: t2.micro
    SubnetId: !Ref PrivateSubnetApp
    SecurityGroupIds: [!Ref AppSecurityGroup]

RDSInstance:
  Type: AWS::RDS::DBInstance
  DeletionPolicy: Retain
  Properties:
    DBName: !Ref DBName
    MasterUsername: !Ref DBUser
    MasterUserPassword: !Ref DBPassword
    DBSubnetGroupName: !Ref MyDBSubnetGroup
    AllocatedStorage: '20'
```

```
    DBInstanceClass: db.t3.micro
    Engine: mysql
    VPCSecurityGroups: [!Ref DBSecurityGroup]

Outputs:
  WebPublicIP:
    Description: "Public IP address of the Web Instance"
    Value: !GetAtt WebInstance.PublicIp

  AppPrivateIP:
    Description: "Private IP address of the App Instance
   Value: !GetAtt AppInstance.PrivateIp

  RDSEndpoint:
    Description: "The connection endpoint for the RDS Database"
    Value: !GetAtt RDSInstance.Endpoint.Addres
```

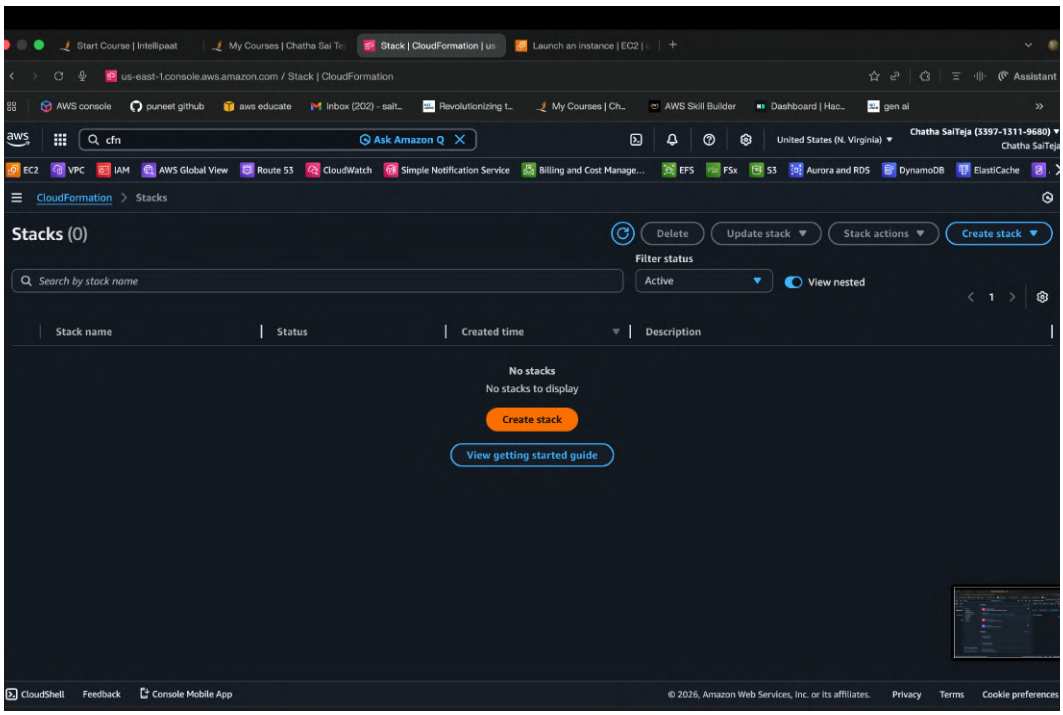
Step 2:- Save the file, open the **CloudFormation console**, and click **Create stack**. Upload the template, click **Next**, specify stack name and give inputs in parameters section ,ignore the optional steps, and select **Create stack**. It takes a few minutes to provision the stack. Once completed, **verify the created resources** in the stack.



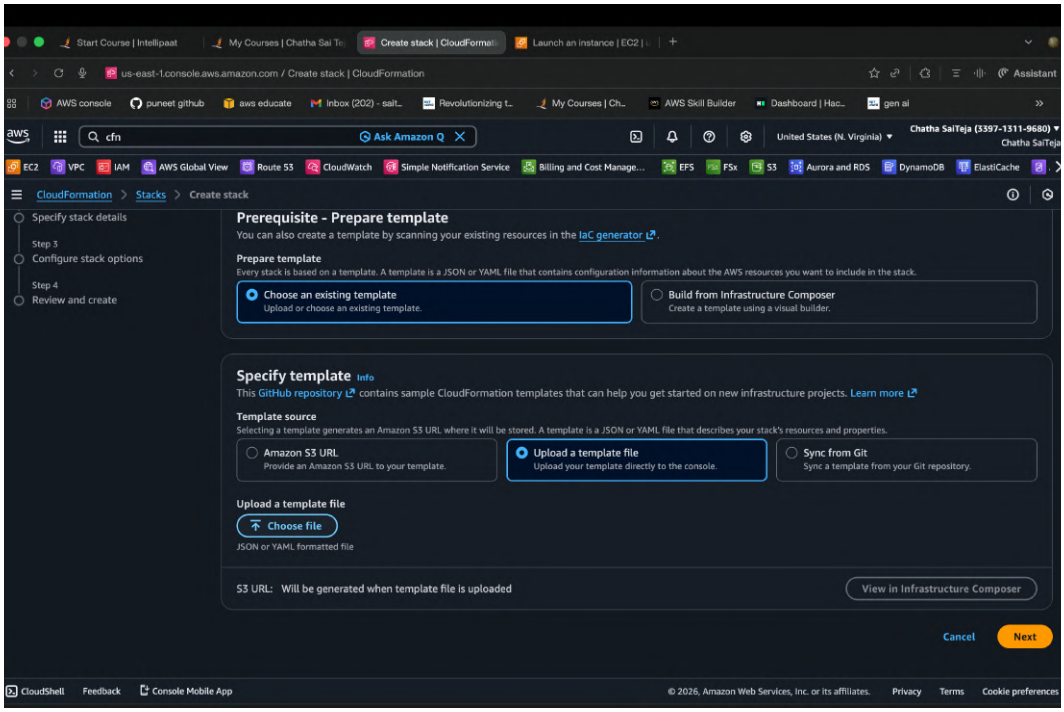
Save the file



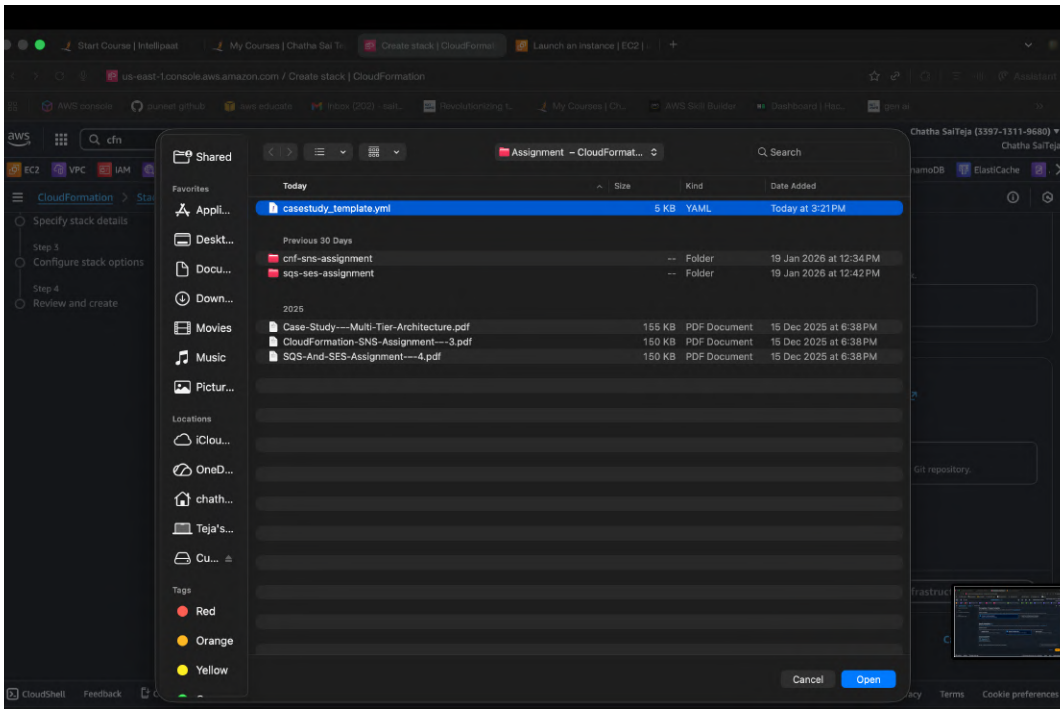
Open cloud formation in console



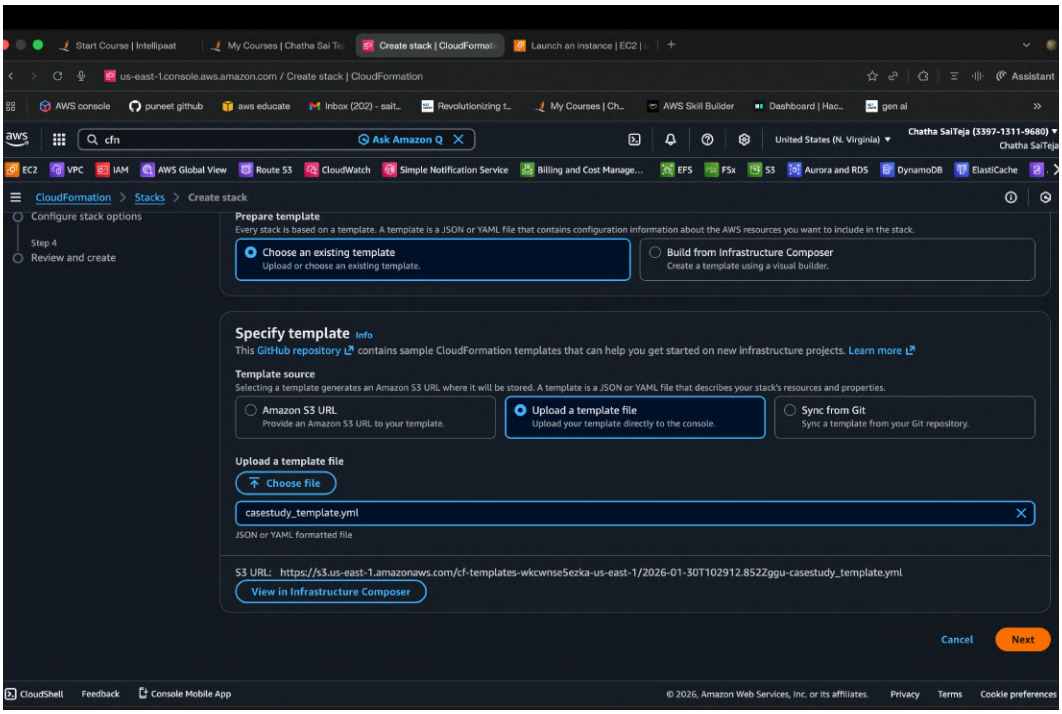
Click on create stack



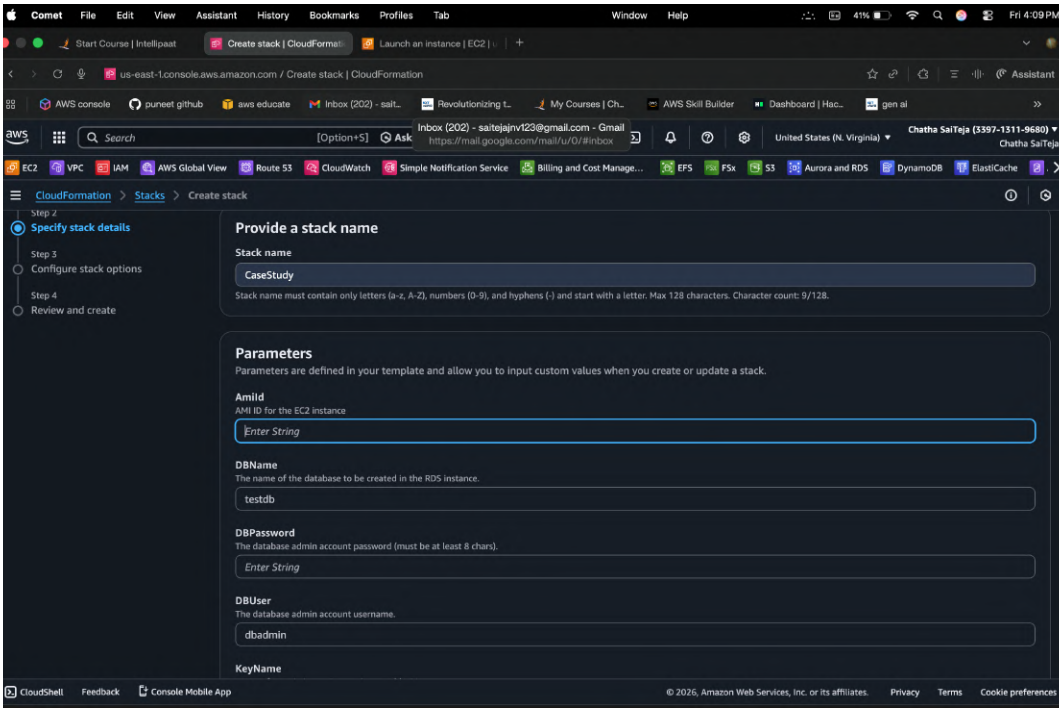
Click on upload template file



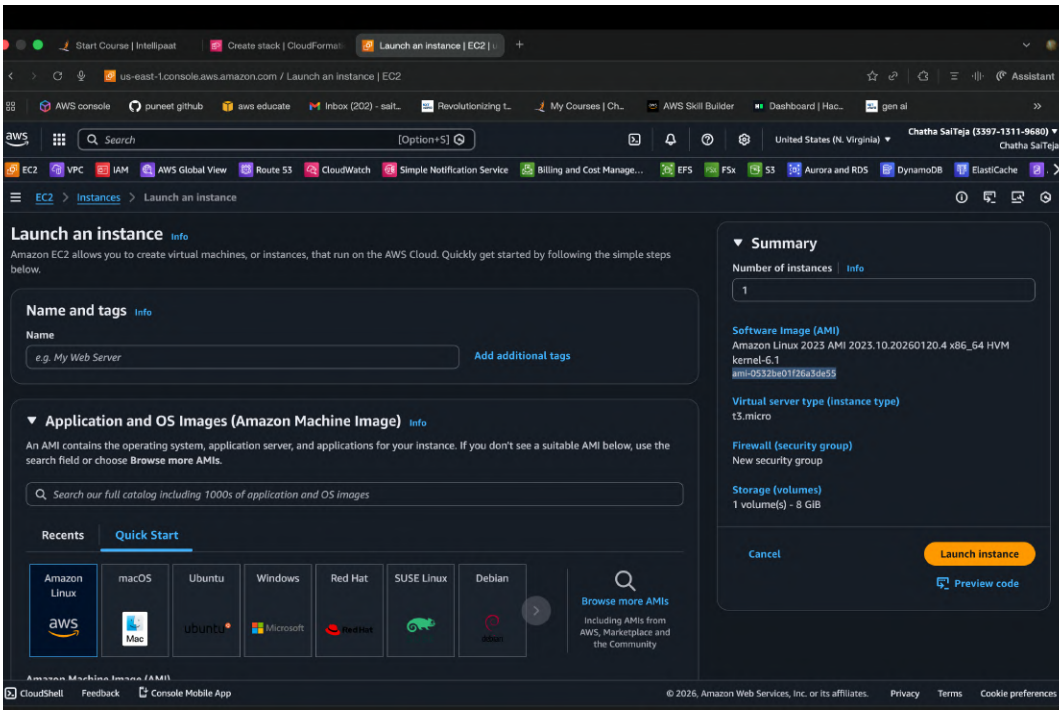
Choose the file



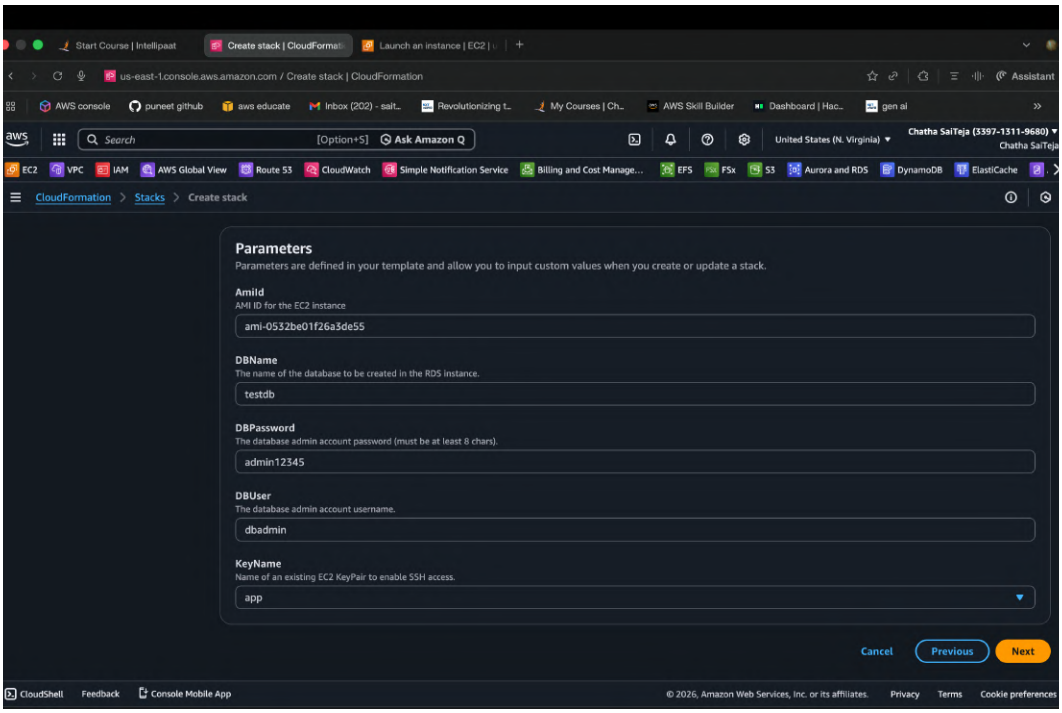
Template is uploaded ->click on next



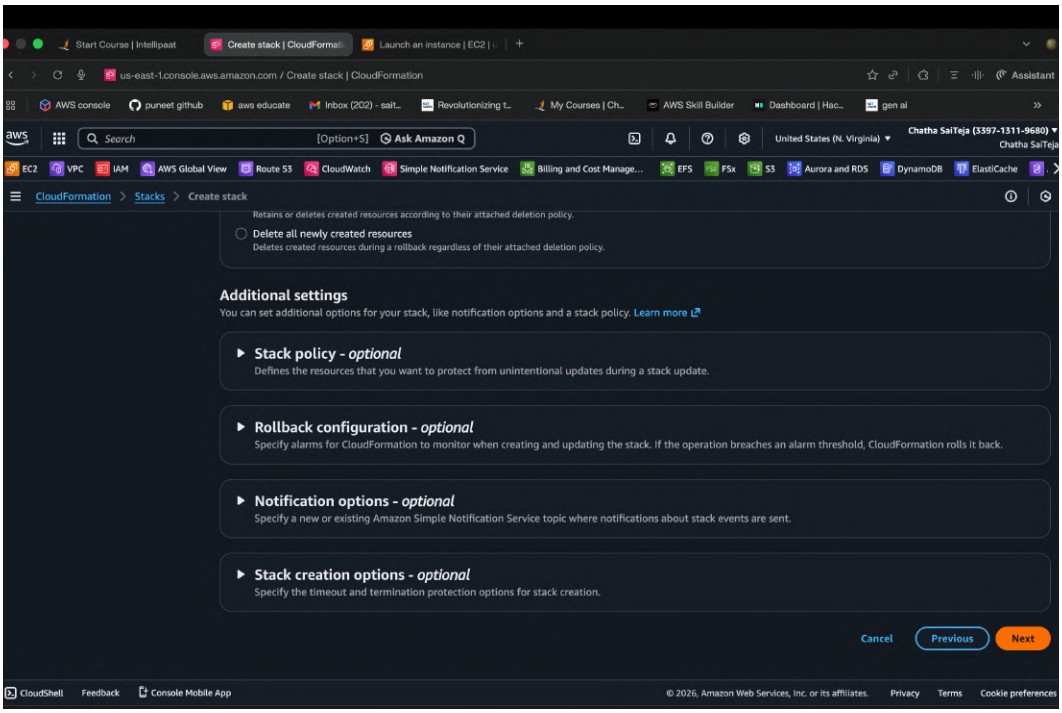
Enter stack name



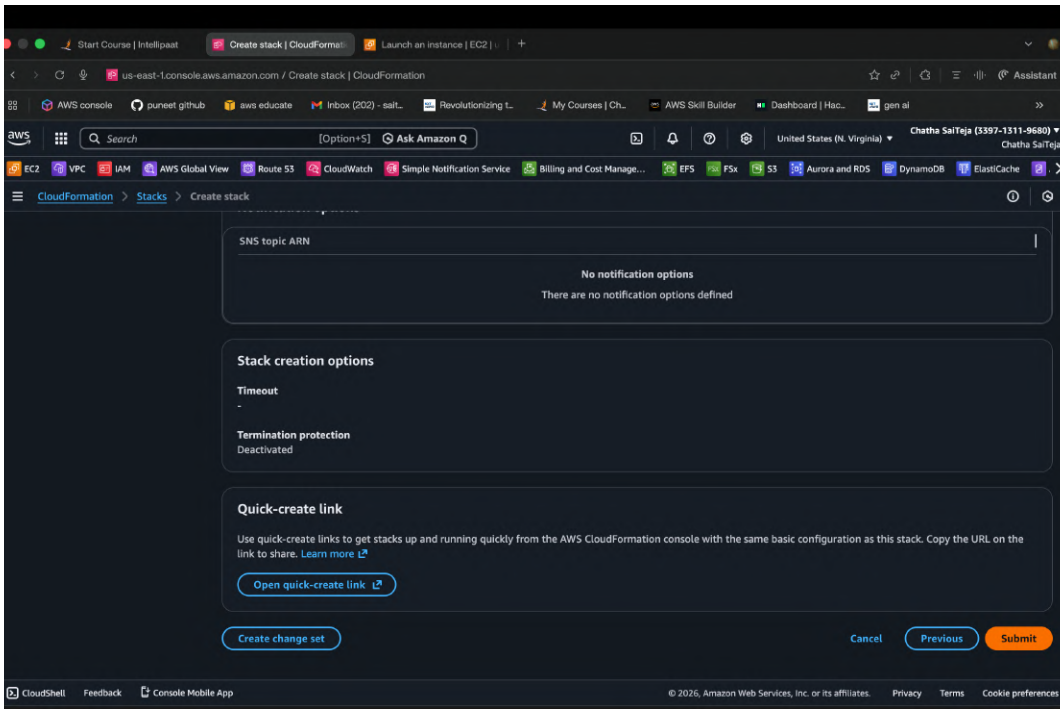
For AMI ID copy the AMI ID



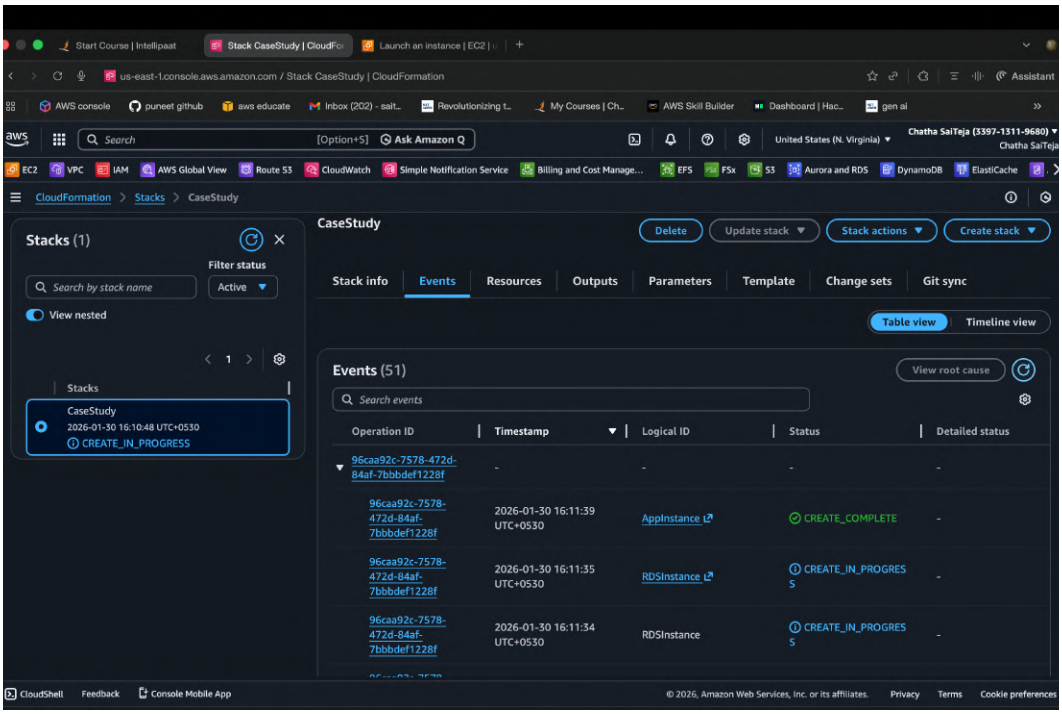
Specify all inputs then click next



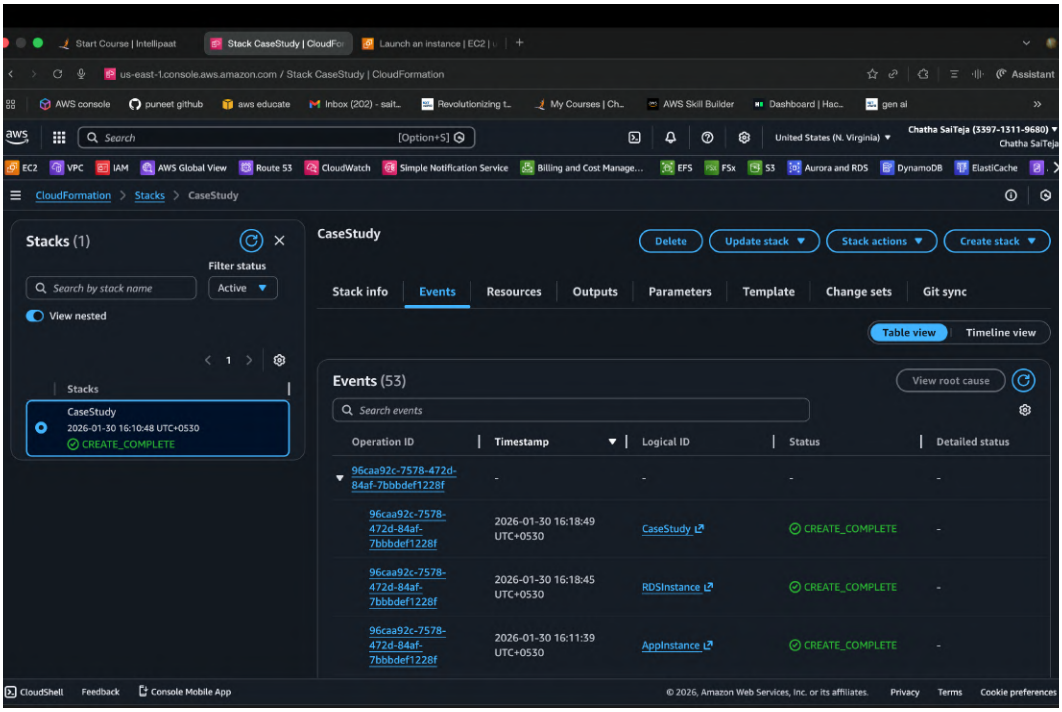
Ignore the optional steps ->click next



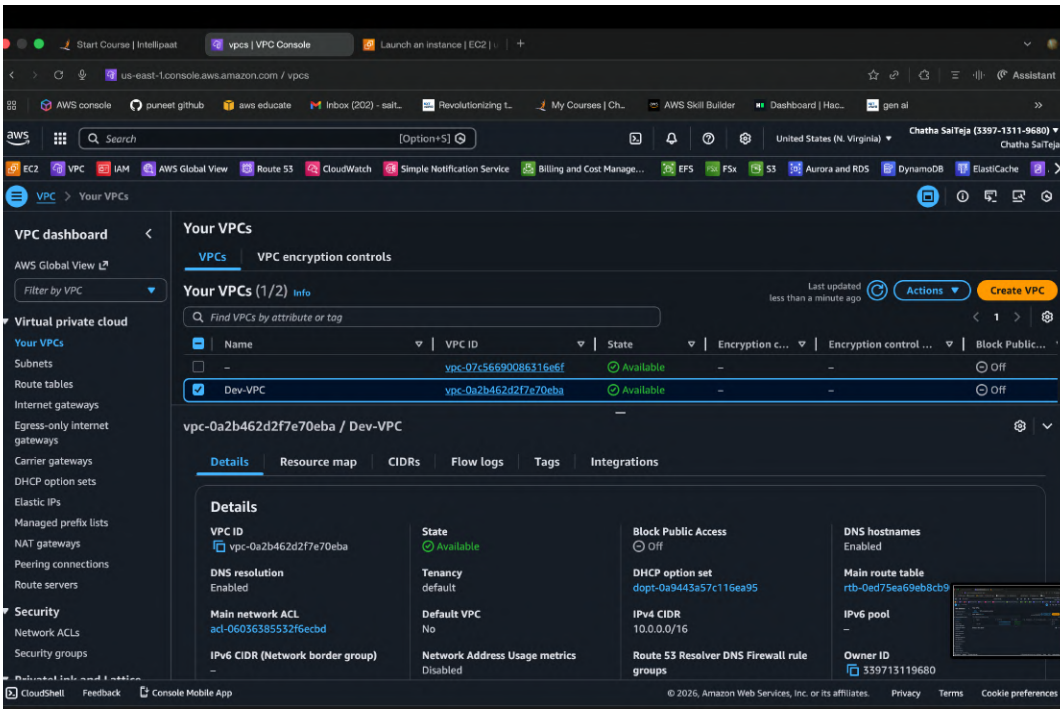
Click on submit



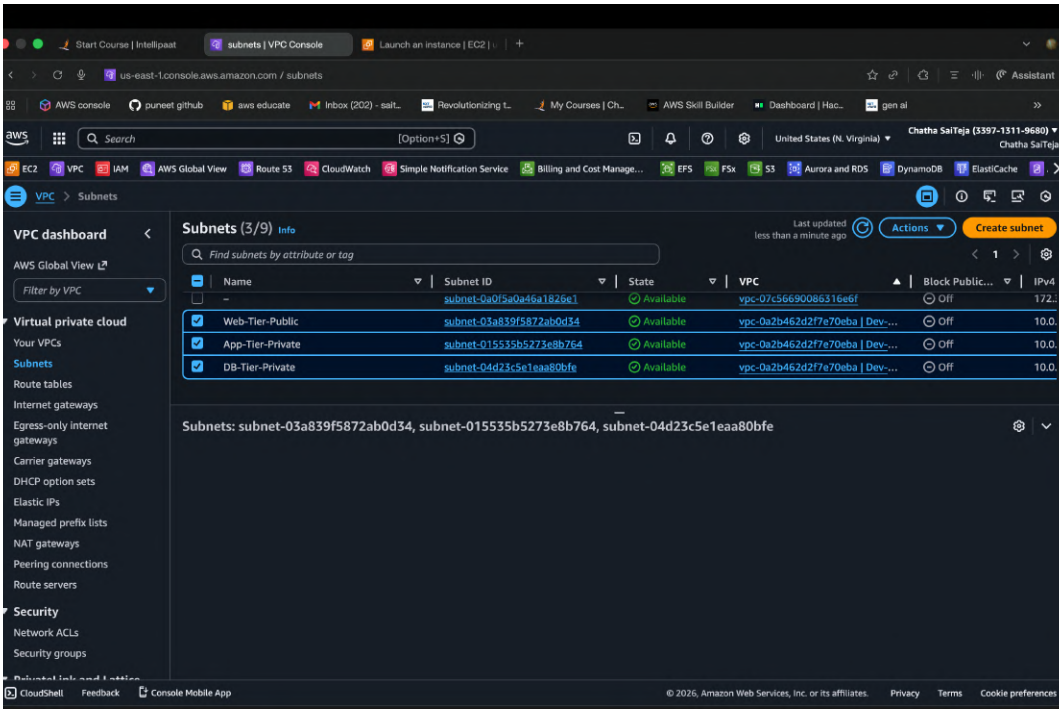
It takes few minutes to create the resources



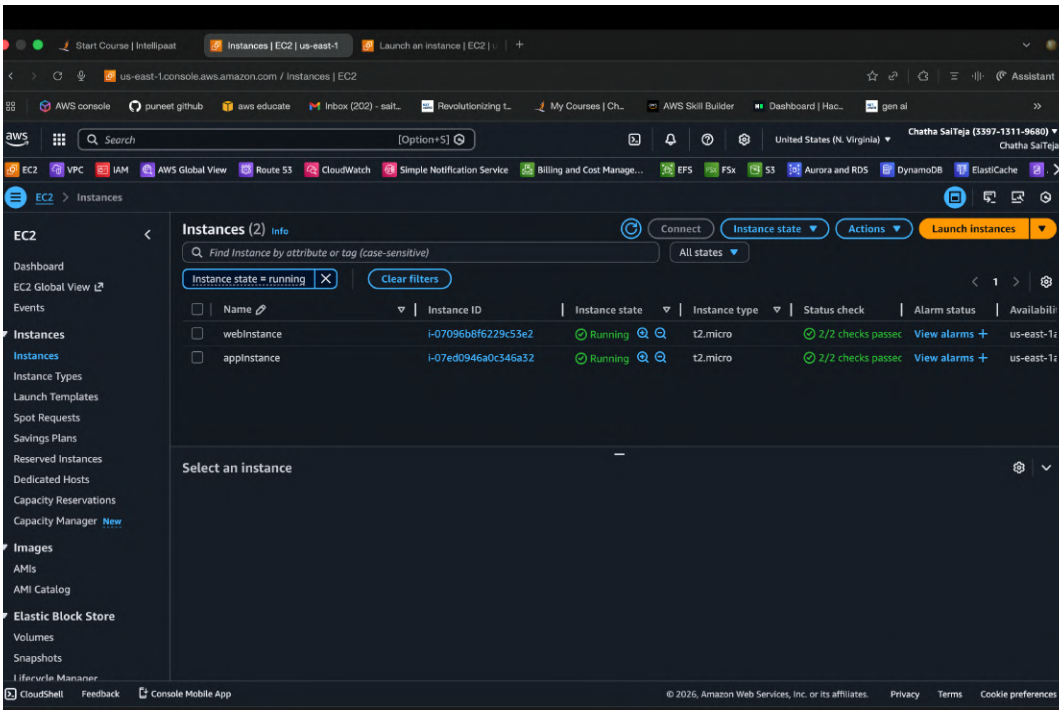
Stack creation is completed



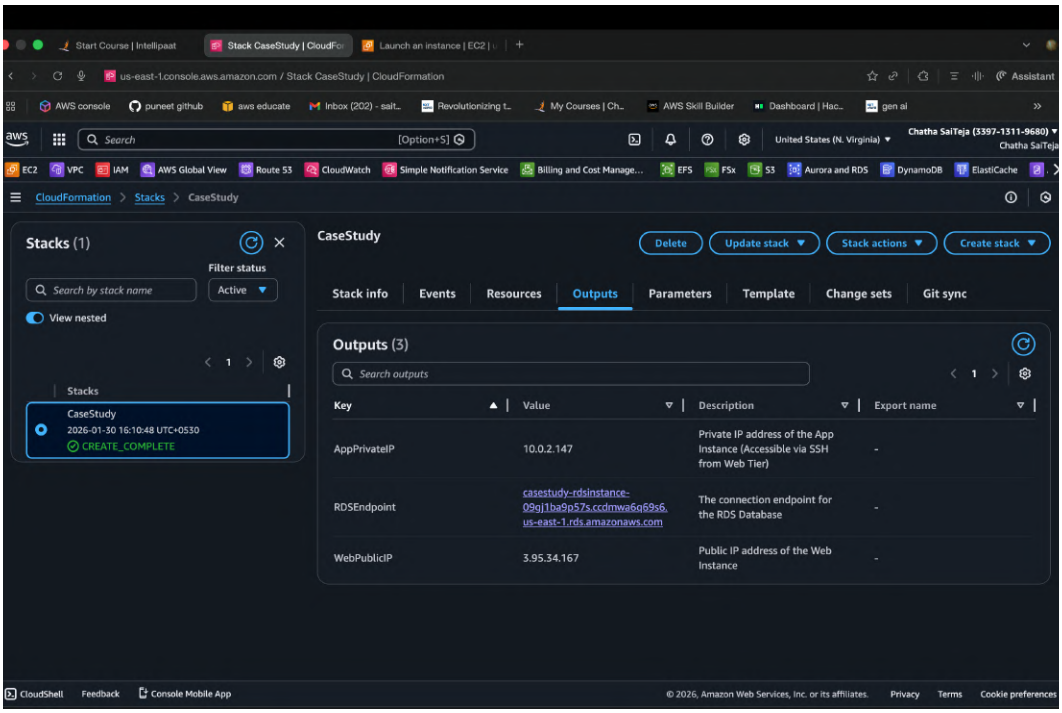
As stack created it creates a VPC for multitier



Three subnets are created(web tier, apptier, db tier)



Two instances is created



outputs section of the stack containing webInstance PublicIp, AppInstance privateIP, db endpoint.