AWS CloudFormation Project

XYZ Technologies is a golden partner of AWS and utilizes most of the AWS services. Now, **CTO** is planning to **set up database infrastructure on the AWS cloud** and **expecting that infrastructure would expand in the future**. There can be multiple changes to infrastructure depending on the future requirements. Engineers should keep track of **future changes and easily roll back** to the previous version **if new infrastructure changes fail**.

Ample uses **MySQL** as a **database engine** and would like to create a database in the **private subnets**. You may use the below configurations to set up the complete infrastructure.

Database Configuration

Storage Space	20 GB
Database Instance Class	Db.t2.micro
Database name	Ampledb
Database Engine	MySQL
Engine Version	8.0.23
PubliclyAccessible	False
Username	awsuser
Password	Aws123456789
Port	3306
AllowUpgradeVersion	False
BackUp Retention Period	7 Days

VPC Configuration

CIDR Block	192.168.0.0/16
EnableDNSHostnames	False
EnableDNSSupport	False
No. of Private Subnets	2
Private Subnet 1 CIDR	192.168.1.0/24
Private Subnet 1 AZ	us-east-1a
Private Subnet 2 CIDR	192.168.2.0/24
Private Subnet 2 AZ	us-east-1b

Security Group Configuration

SG Ingress CIDR	0.0.0.0/0
SG Ingress FromPort	3306
SG Ingress ToPort	3306
SG Ingress IPProtocol	TCP
SG Egress CIDR	0.0.0.0/0
SG Egress FromPort	80
SG Egress ToPort	80
SG Egress IPProtocol	TCP

To-Dos:

- List down AWS services to be used? – CloudFormation, VPC, Database - MySQL
- Suggest your approach.
- Design your plan
- Implement your plan
- Verify if the complete infrastructure is running

Deliverables:

- Setup Complete Infrastructure

KEY POINTS

CloudFormation: CloudFormation stack to deploy an EC2 instance. CloudFormation is a powerful service that allows you to manage your AWS Infrastructure as a Code (IaC), making it easier to provision and manage your resources.

CloudFormation is a service that allows you to create and manage AWS resources using code rather than provisioning them manually on AWS Management Console. This offers several benefits such as consistent and repeatable deployments Version Control and change Management, **automatic roll backs and error handling, and the ability to create reusable templates for different environments**.

By using CloudFormation, you can define your entire AWS Infrastructure including resources like EC2 instances, security groups, and more in a template file. This template can then be used to provision and manage those resources, ensuring that your infrastructure is deployed consistently every time.

CTO: Chief Technical Officer

Database: A structured collection of data, organized for efficient retrieval and management. It's essentially a way to store and access information, whether it's electronic or physical

VPC: It stands for Virtual Private Cloud. A VPC is a virtual network that closely resembles a traditional network that you would operate in your own data center. After you create a VPC, you can add subnets.

Subnet: A subnet is a range of IP addresses in your VPC. A subnet must reside in a single Availability Zone. After you add subnets, you can deploy AWS resources in your VPC.

Private Subnet: A private subnet is a subnet within a Virtual Private Cloud (VPC) that does not have a direct route to the internet. Resources within a private subnet can't be directly accessed from the public internet, enhancing security by limiting exposure. They typically rely on Network Address Translation (NAT) gateways or instances in public subnets to access the internet for outbound traffic.

Public Subnet: A public subnet in a Virtual Private Cloud (VPC) is a subnet that has a route to an Internet Gateway, allowing resources within the subnet to directly communicate with the public internet. This direct access is enabled by a route table that contains a route to the Internet

Gateway. Essentially, it is a subnet where resources can be accessed from and send traffic to the internet.

CIDR: It stands for Classless Inter-Domain Routing. It is an IP address allocation method that improves data routing efficiency on the internet. Every machine, server, and end-user device that connects to the internet has a unique number, called an IP address, associated with it. Devices find and communicate with one another by using these IP addresses. Organizations use CIDR to allocate IP addresses flexibly and efficiently in their networks

Ingress rule: A configuration that controls incoming network traffic to a system or application.

Egress rule: An egress rule defines the permitted outbound traffic from a network or system

PART 1: AWS SERVICES USED

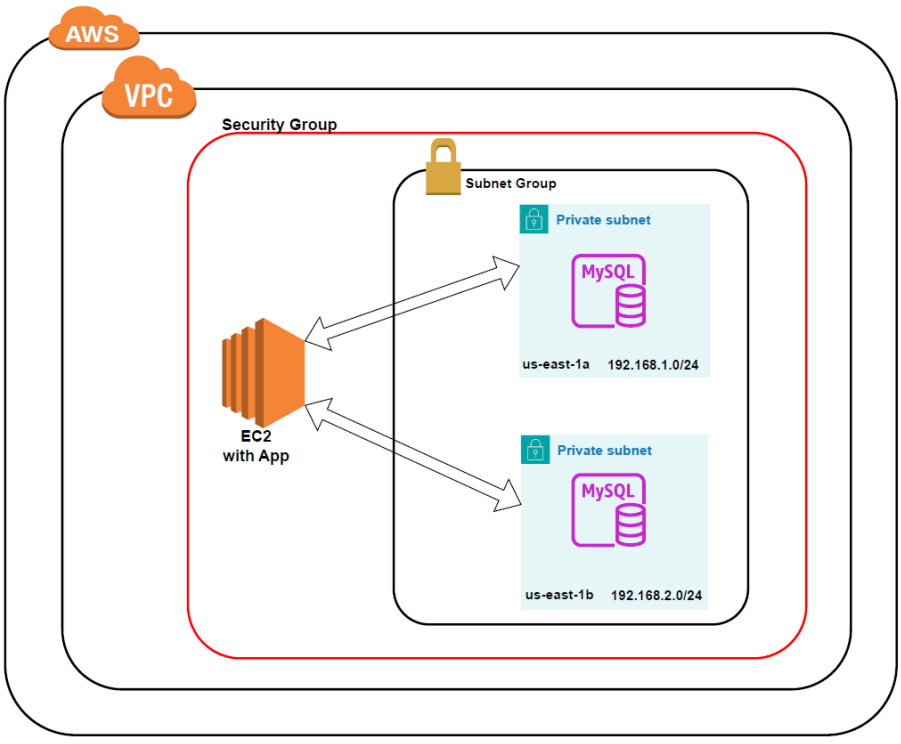
In this project, I will do the following:

1. Designed and implemented infrastructure-as-code using AWS CloudFormation.
2. Created templates for rapid deployment of complex applications.
3. Enforced best practices for reliable and scalable infrastructure provisioning.

We will use the following AWS Services

- CloudFormation
- VPC: This is the father of security groups
- Amazon RDS: RDS is running on EC2, since a database always need a server.
- Security Group that is a child to VPC
- Two Private Subnets
- Subnet Group (Used when dealing with RDS): It is advisable to create one Subnet Group and add the two subnets in it.

Architecture

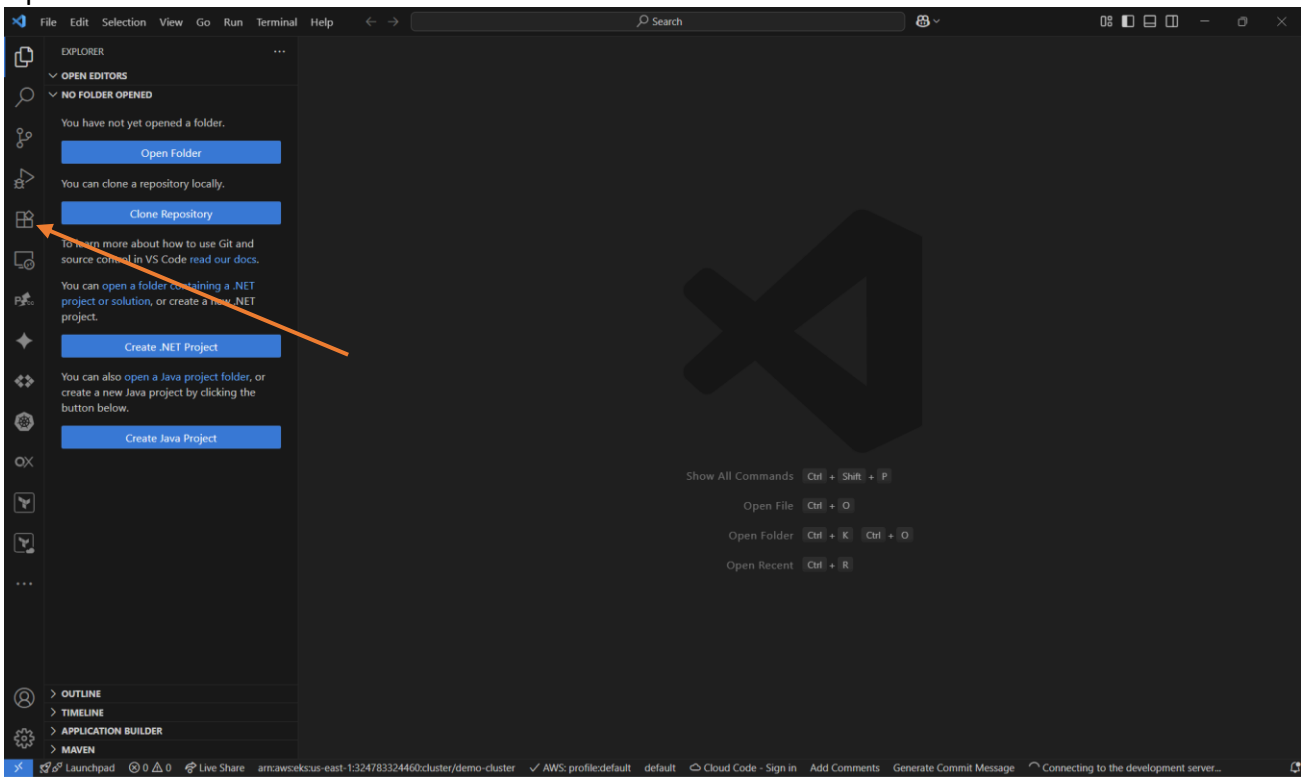


Follow these steps to create a DB instance in a VPC:

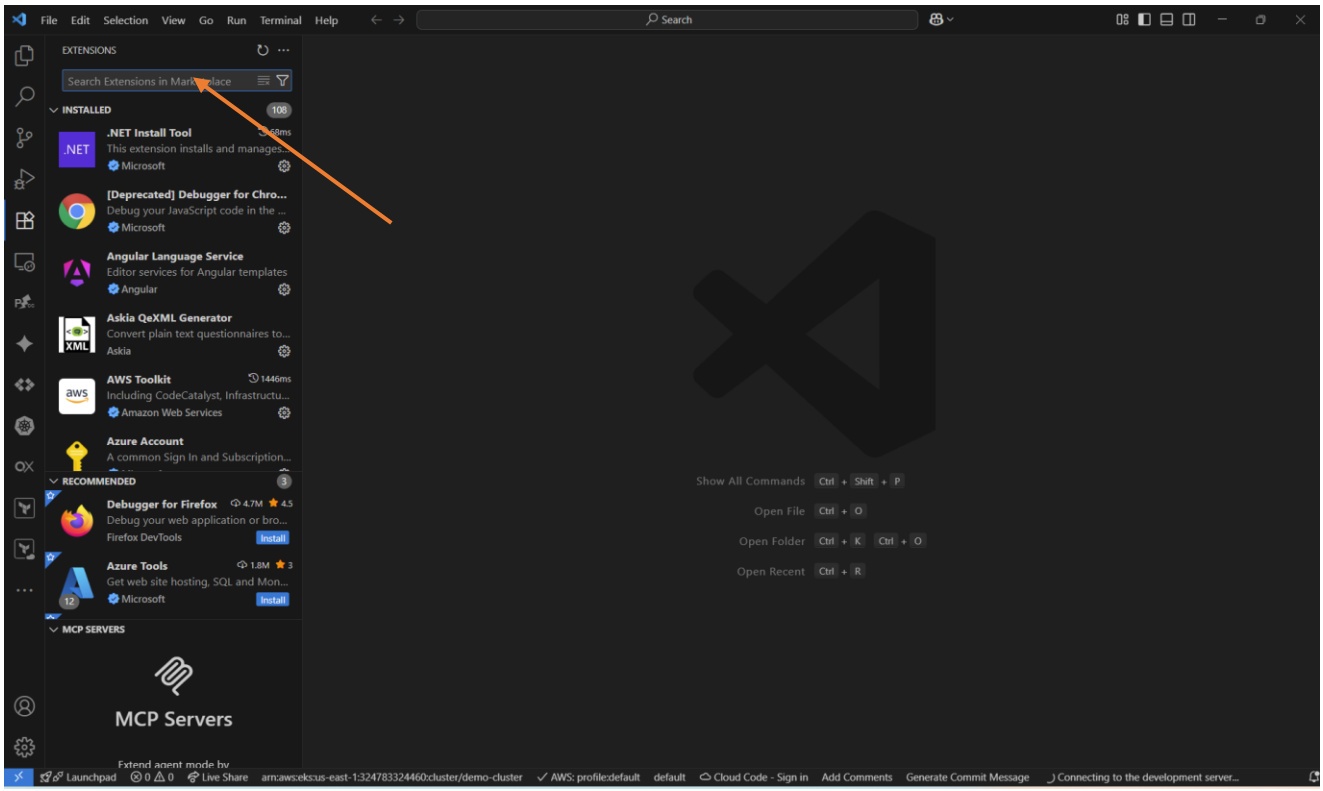
- Step 1: Create a VPC
- Step 2: Create a DB subnet group for the two private subnets
- Step 3: Create a VPC security group
- Step 4: Create a DB instance in the VPC

PART 4: IMPLEMENTATION

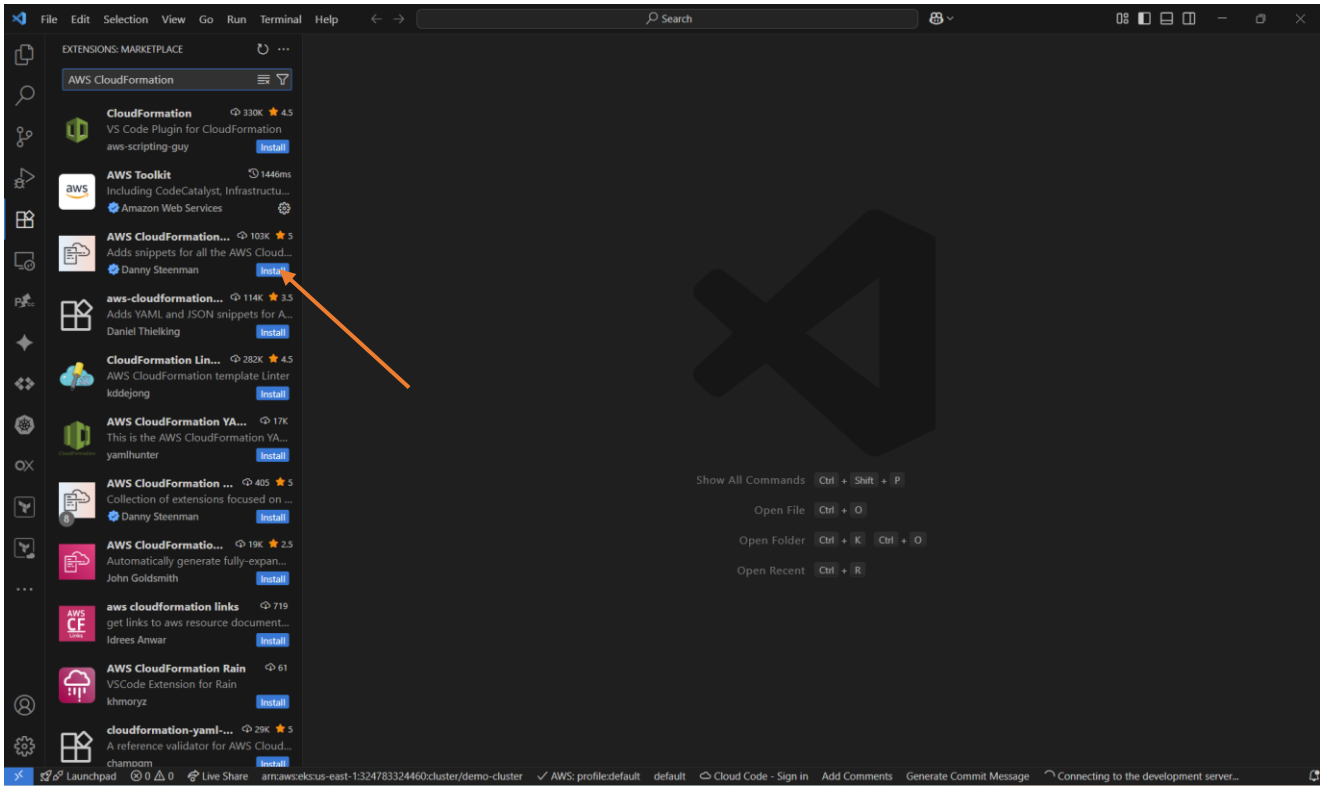
Open VSCode and install the extensions for CloudFormation



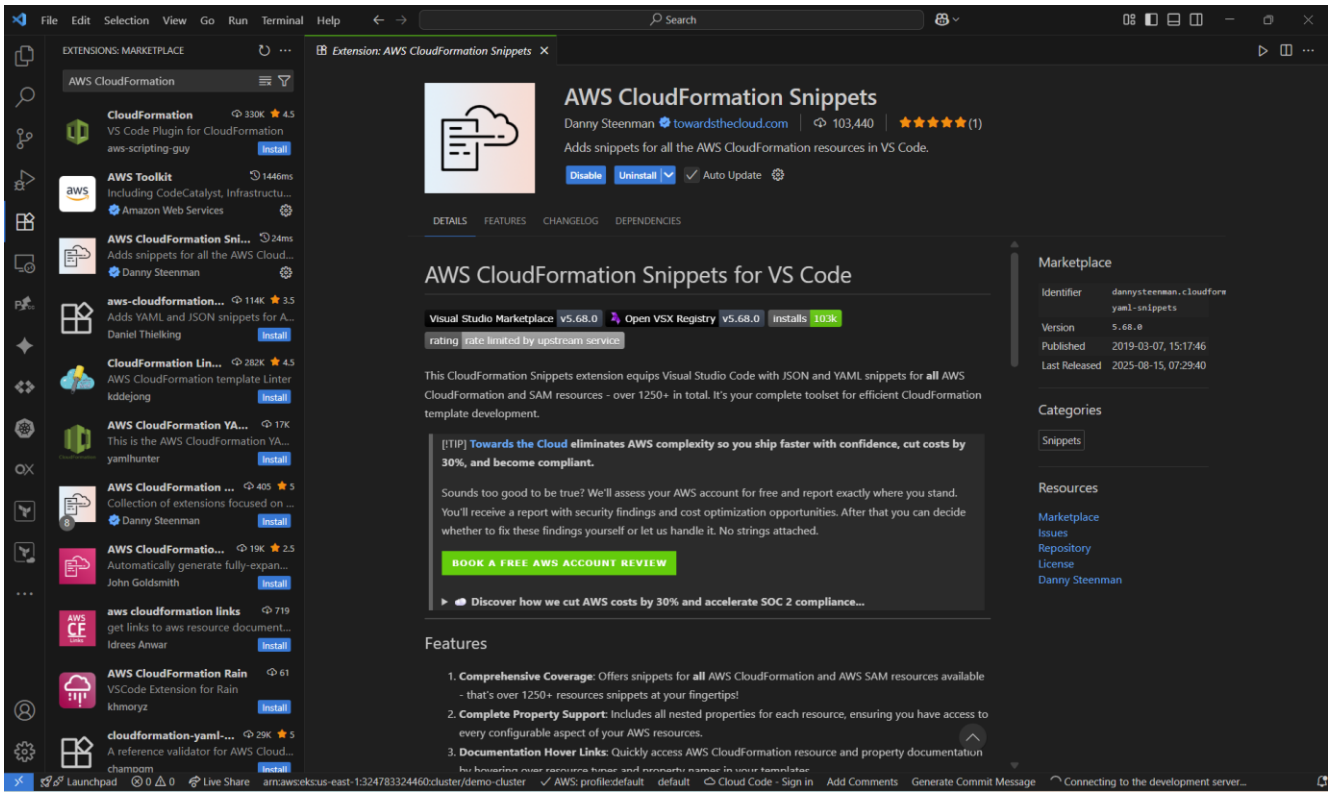
Click on "**Extensions**"



Search for "**AWS CloudFormation**"

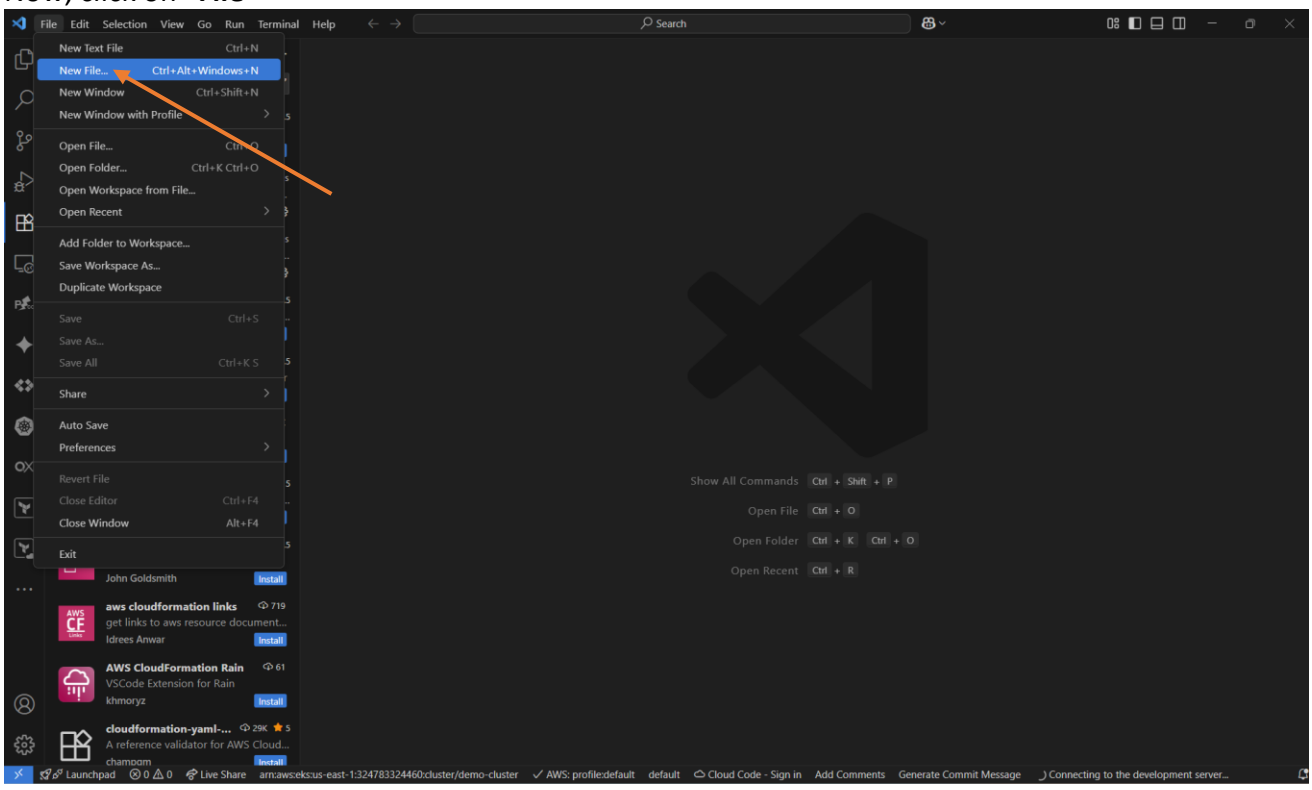


Click on install on "**AWS CloudFormation Snippet**"

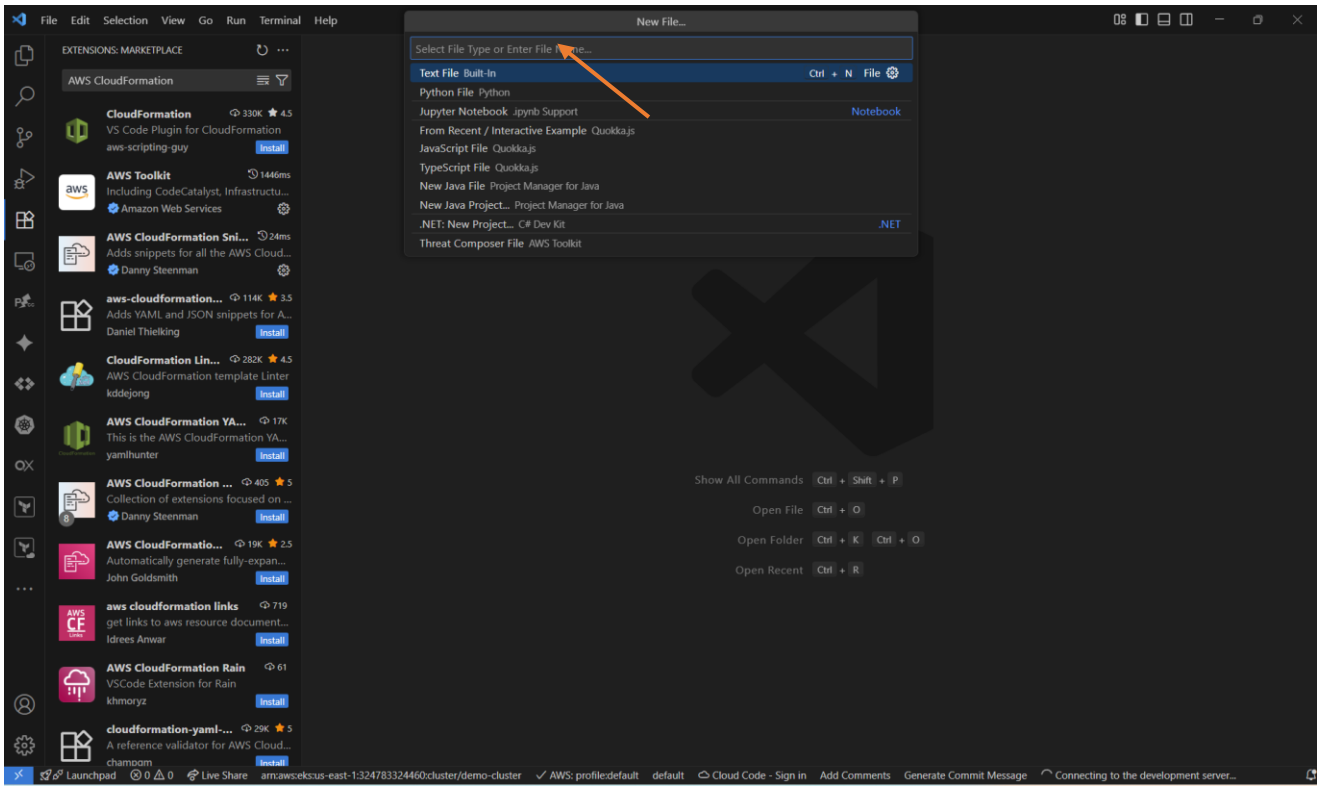


You can see that the "**AWS CloudFormation Snippet**" has been installed.

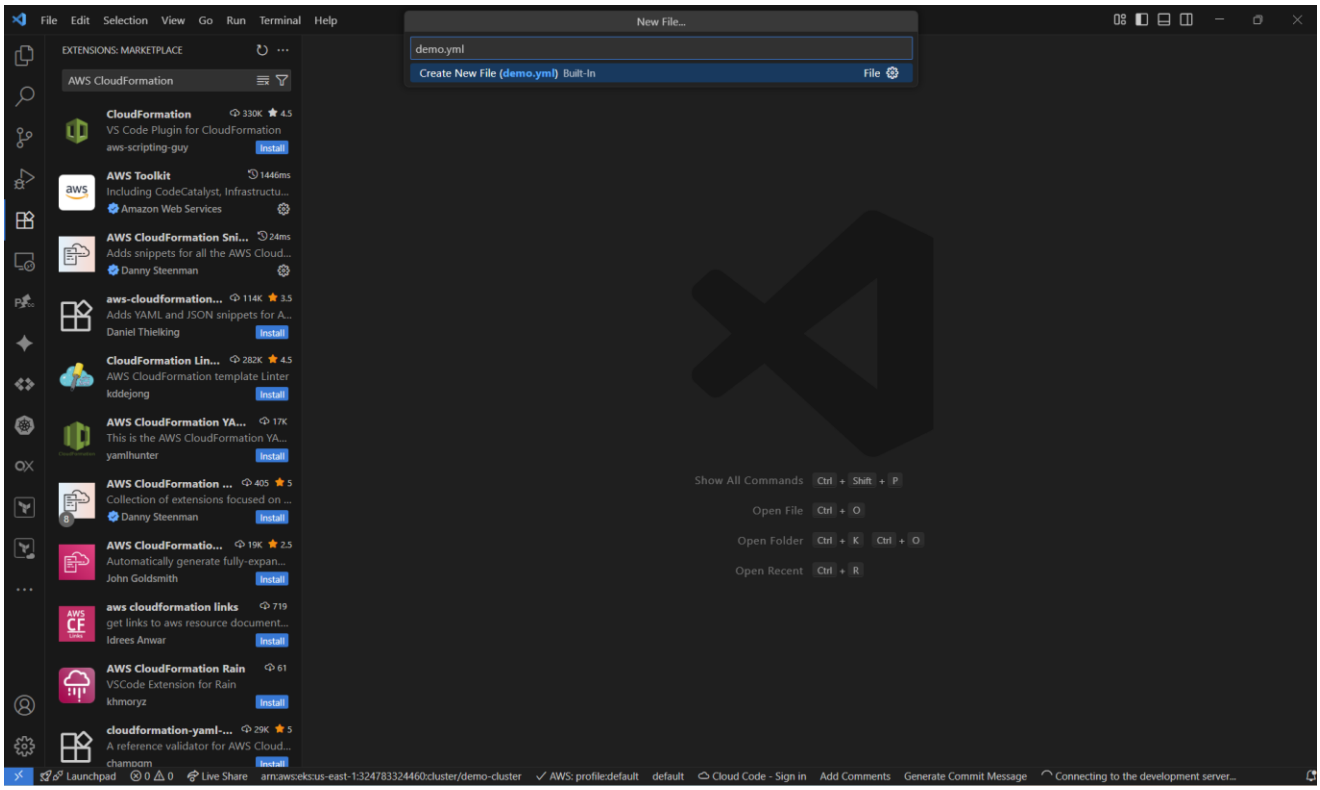
Now, click on "**File**"



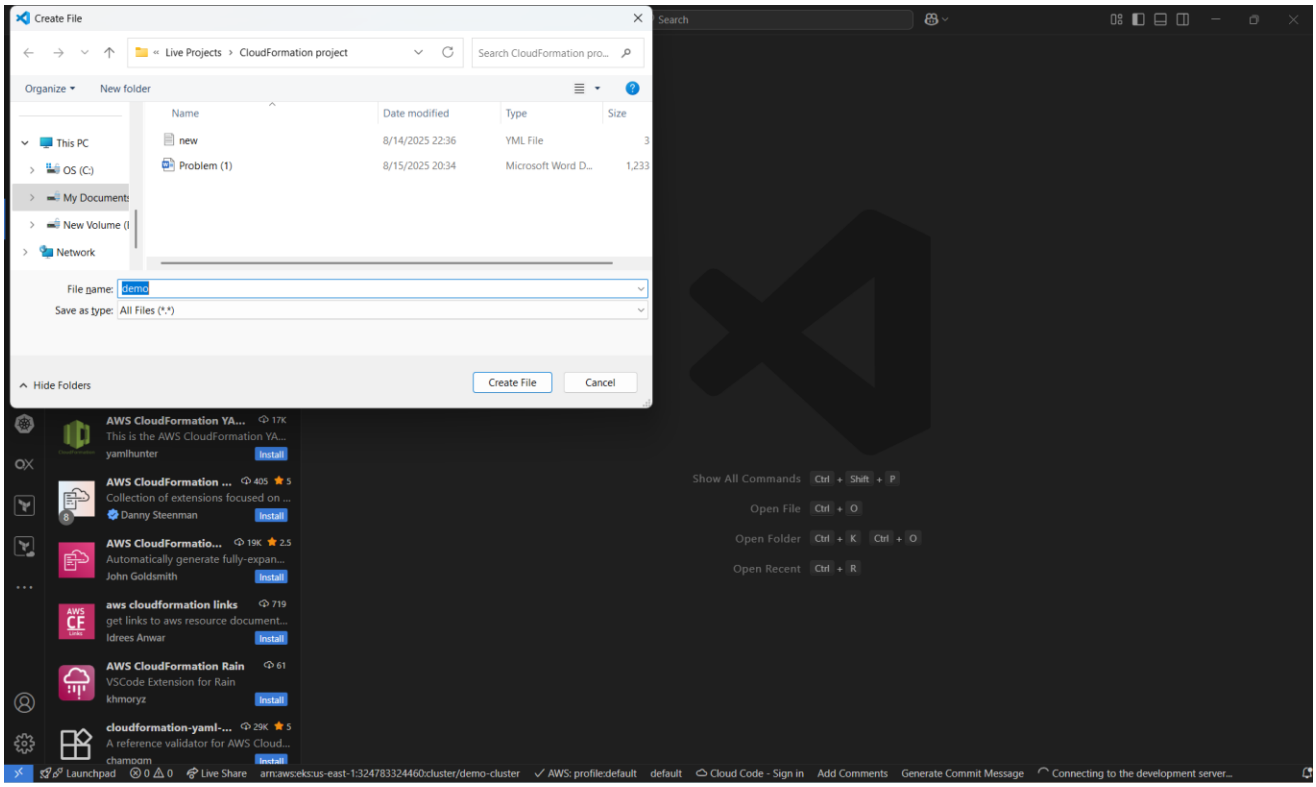
Select "**New File**"



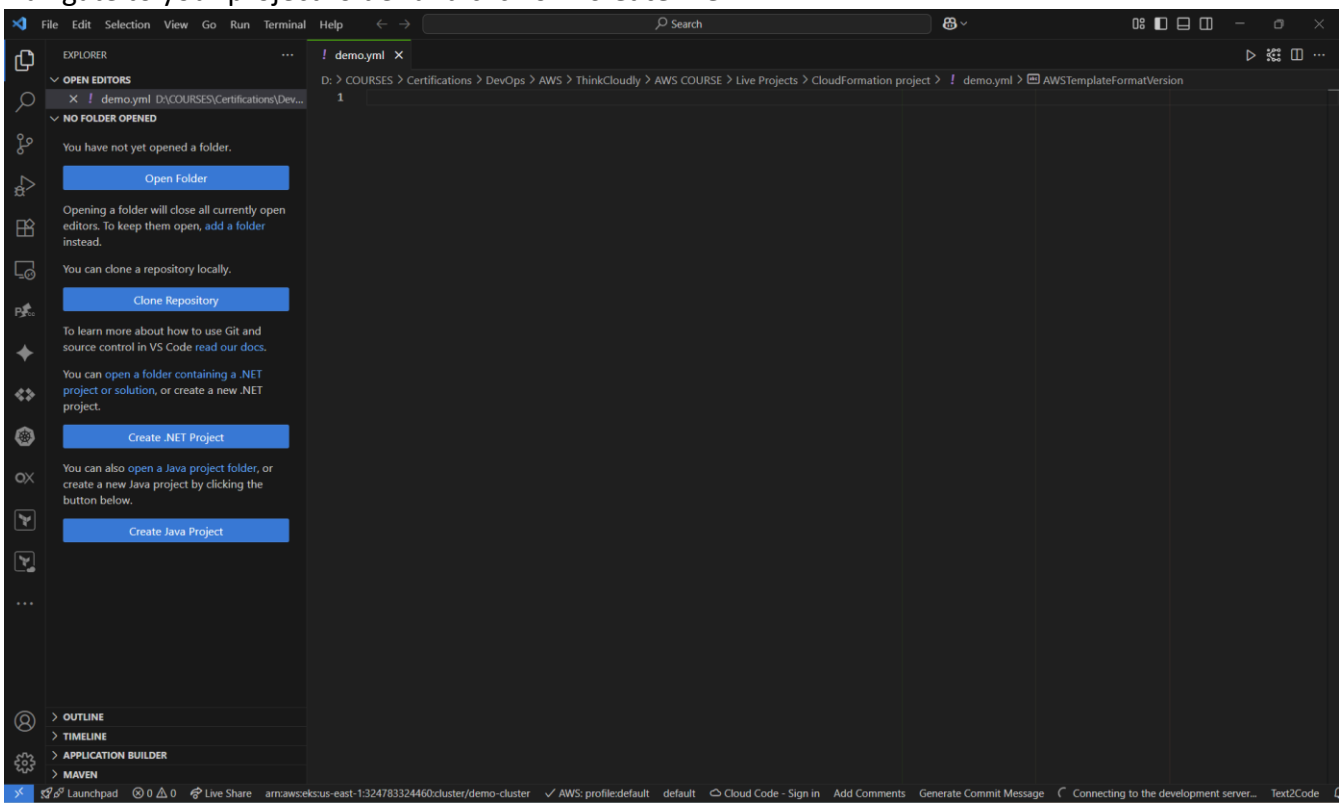
Enter "**demo.yml**". "**demo**" is the name of our CloudFormation configuration file.



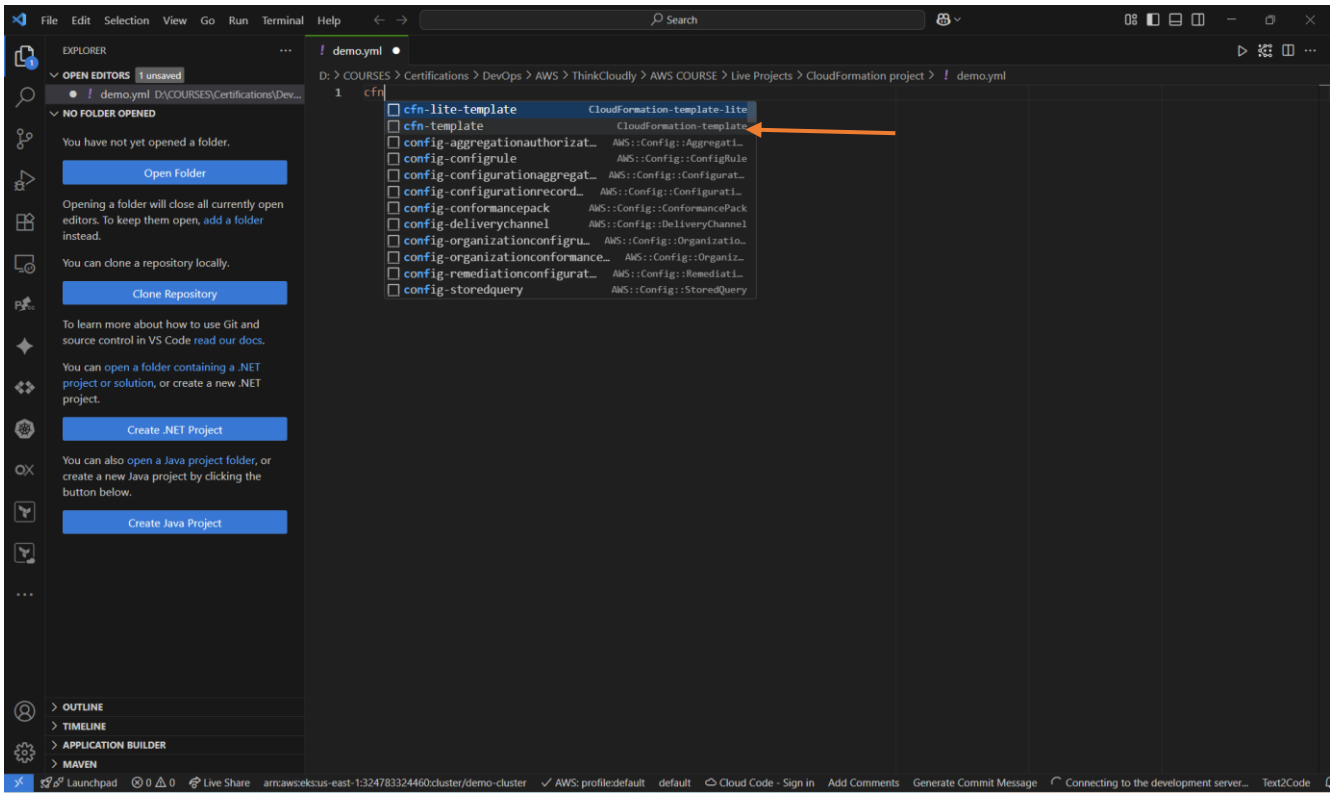
Press ENTER



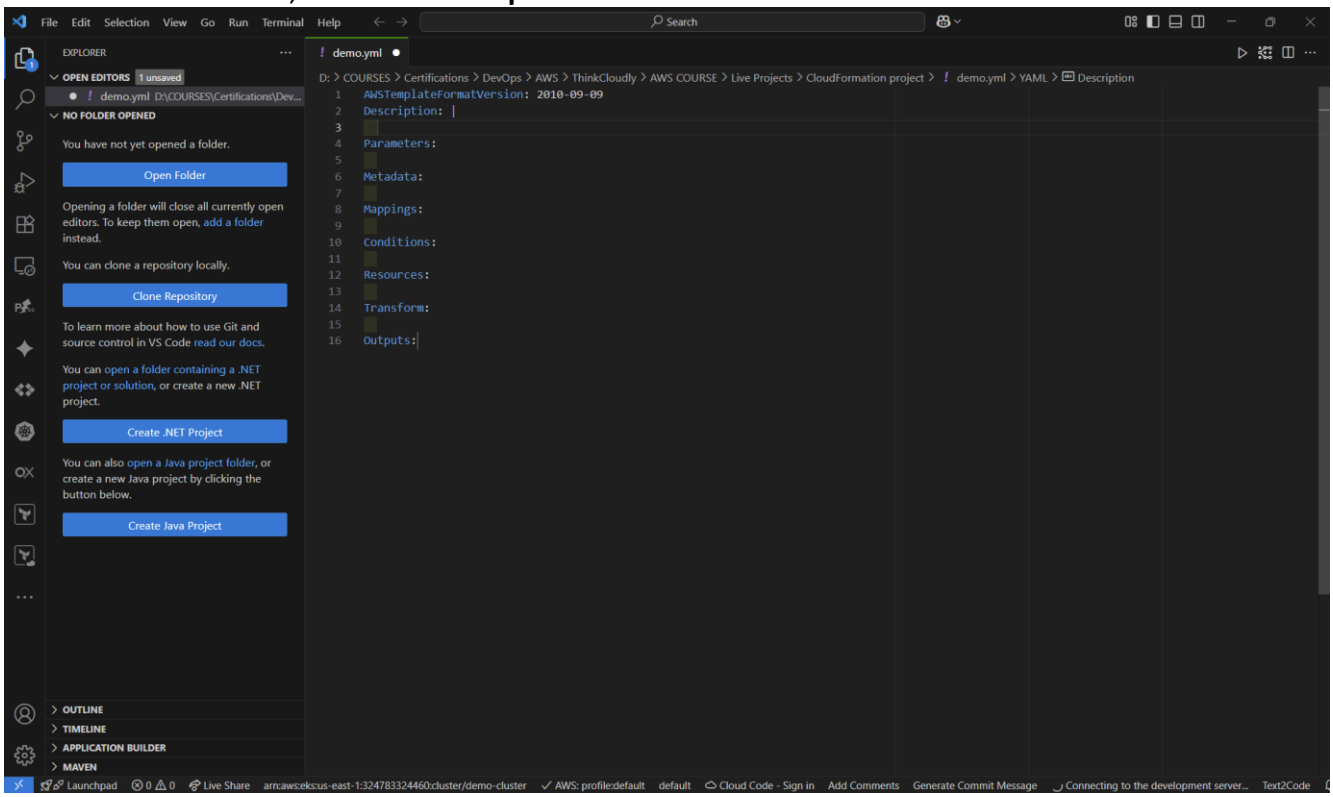
Navigate to your project folder and click on "**create file**"



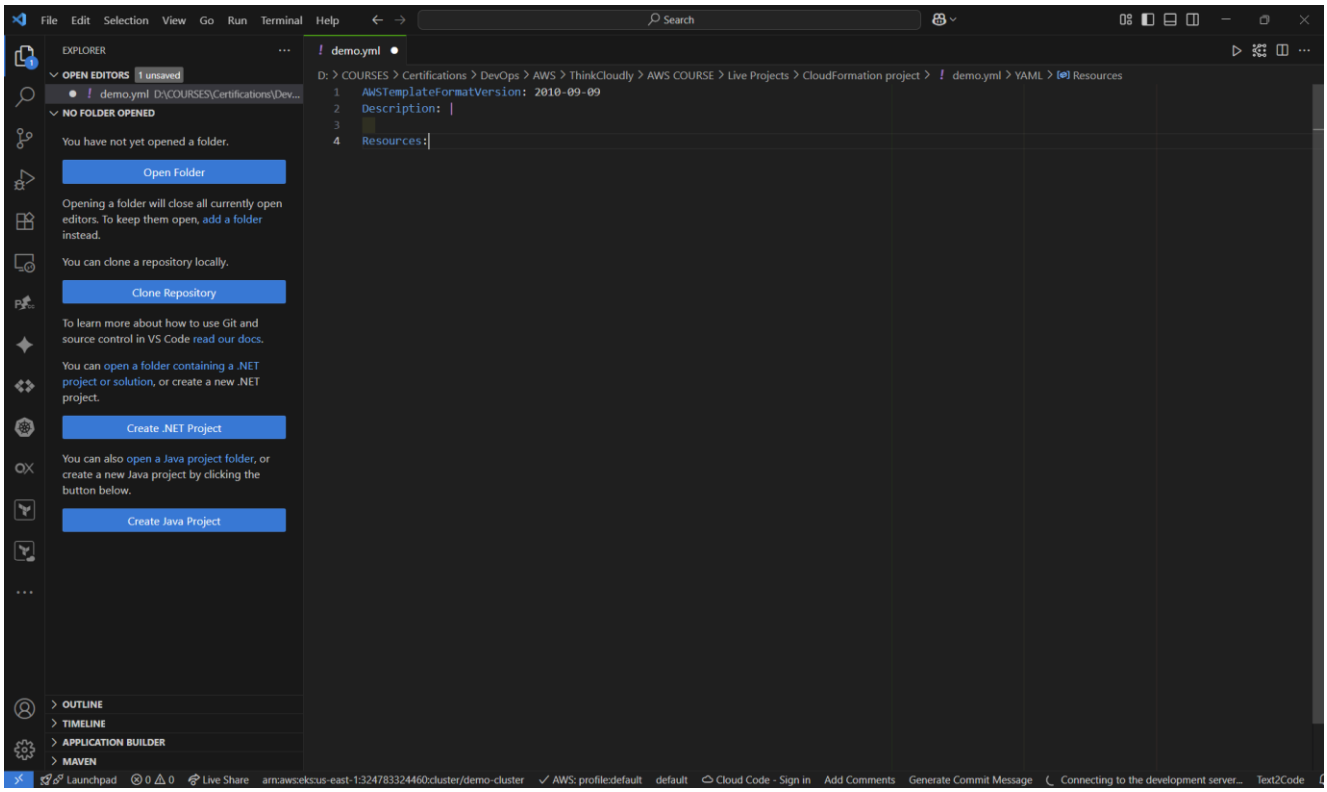
Now type "**cfn**"



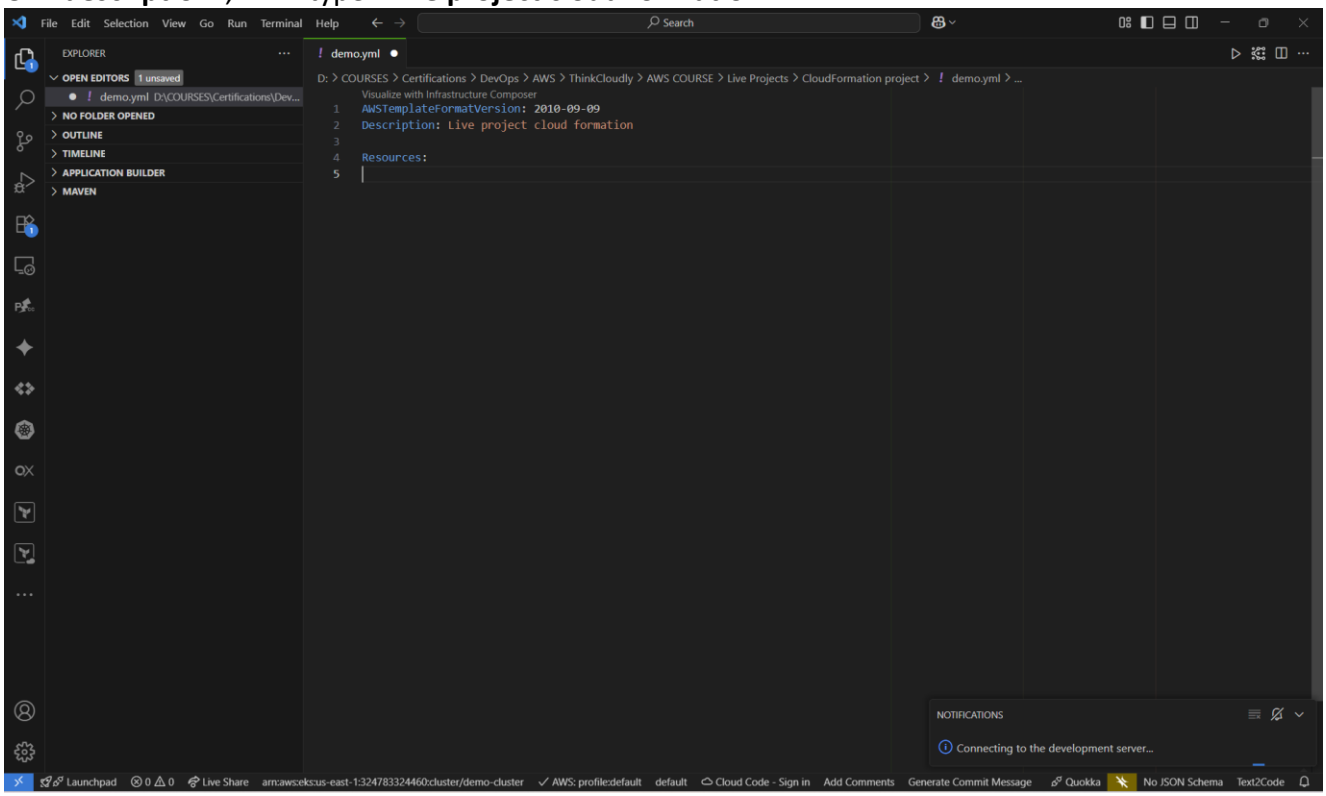
Select the second one, that is "**cfn template**"



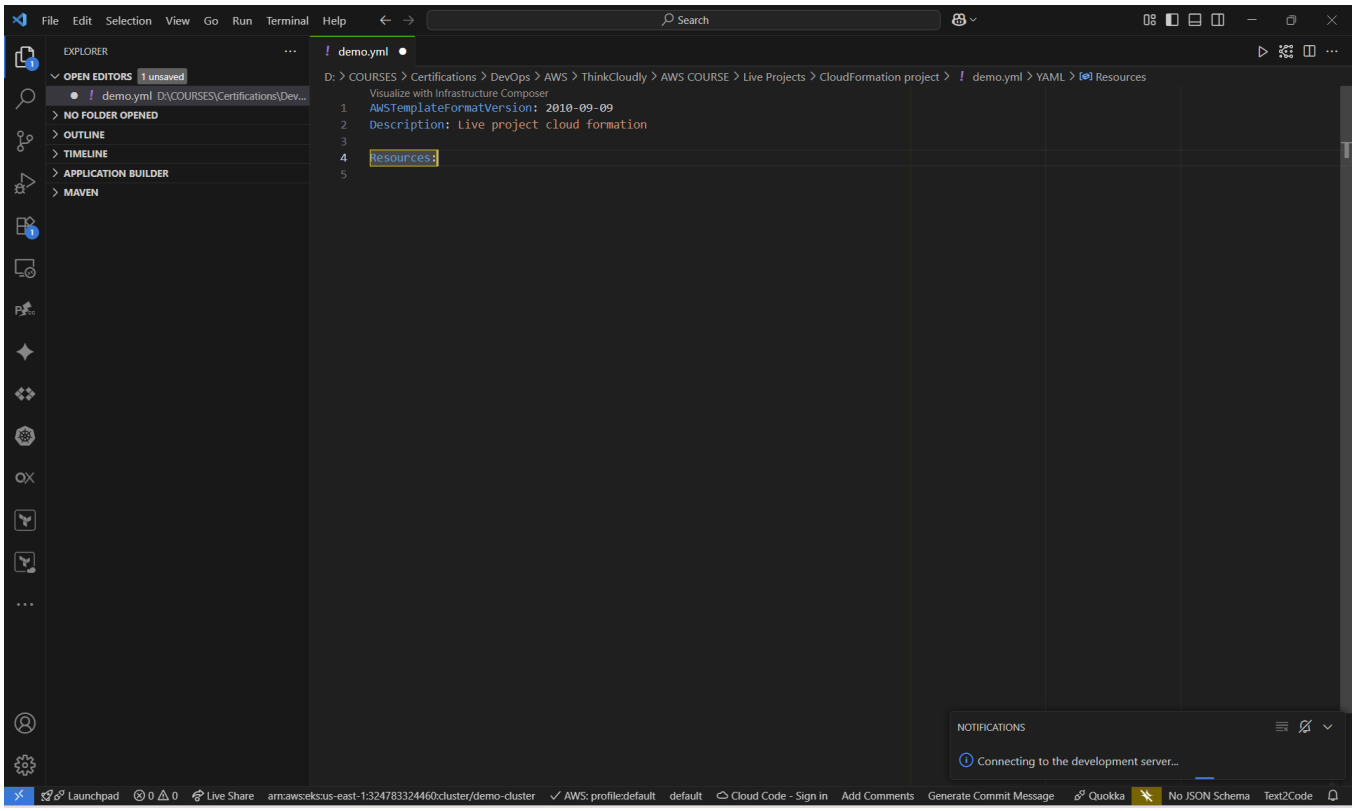
Delete all the lines you do not need. I will leave only "**Description**" and "**Resources**"



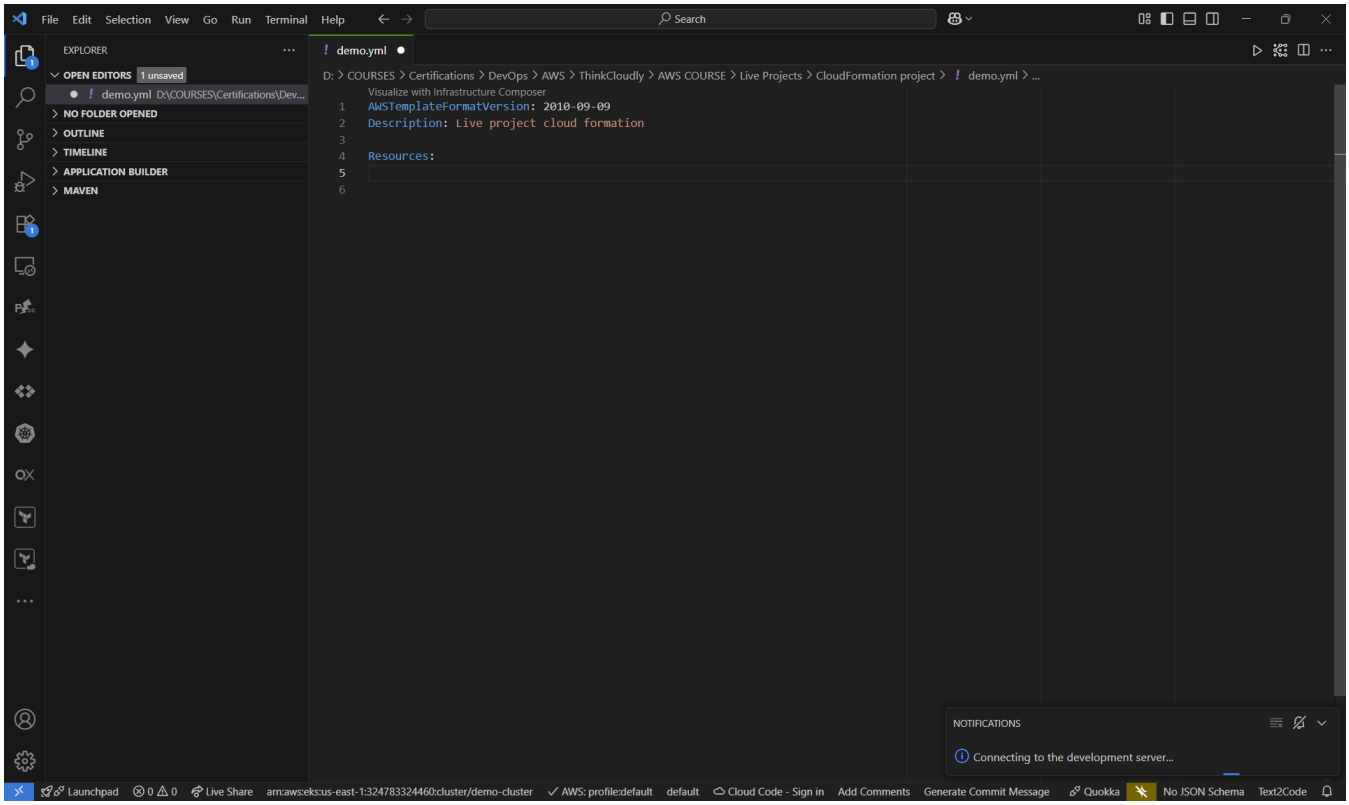
On "**description**", I will type "**Live project cloud formation**"



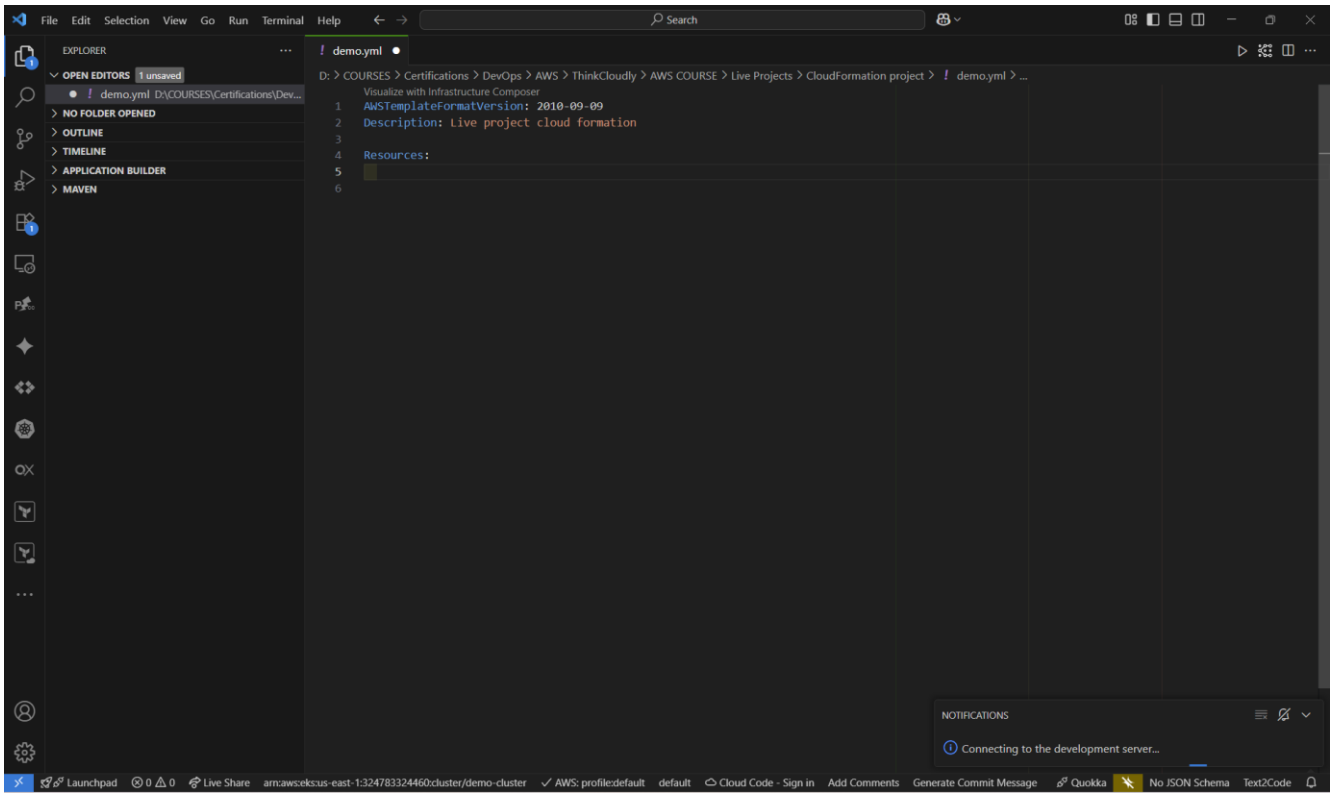
Put the cursor on the "**Resources**" line



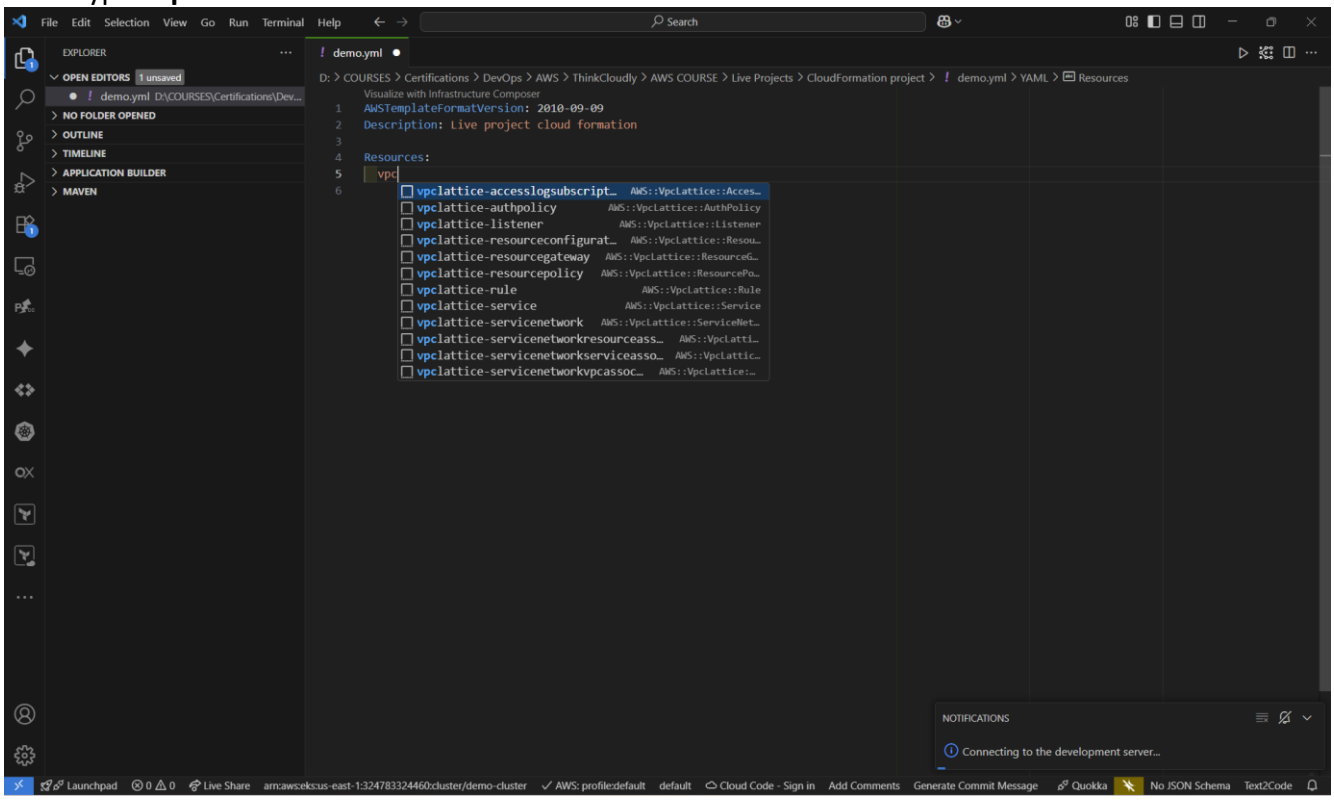
Press ENTER



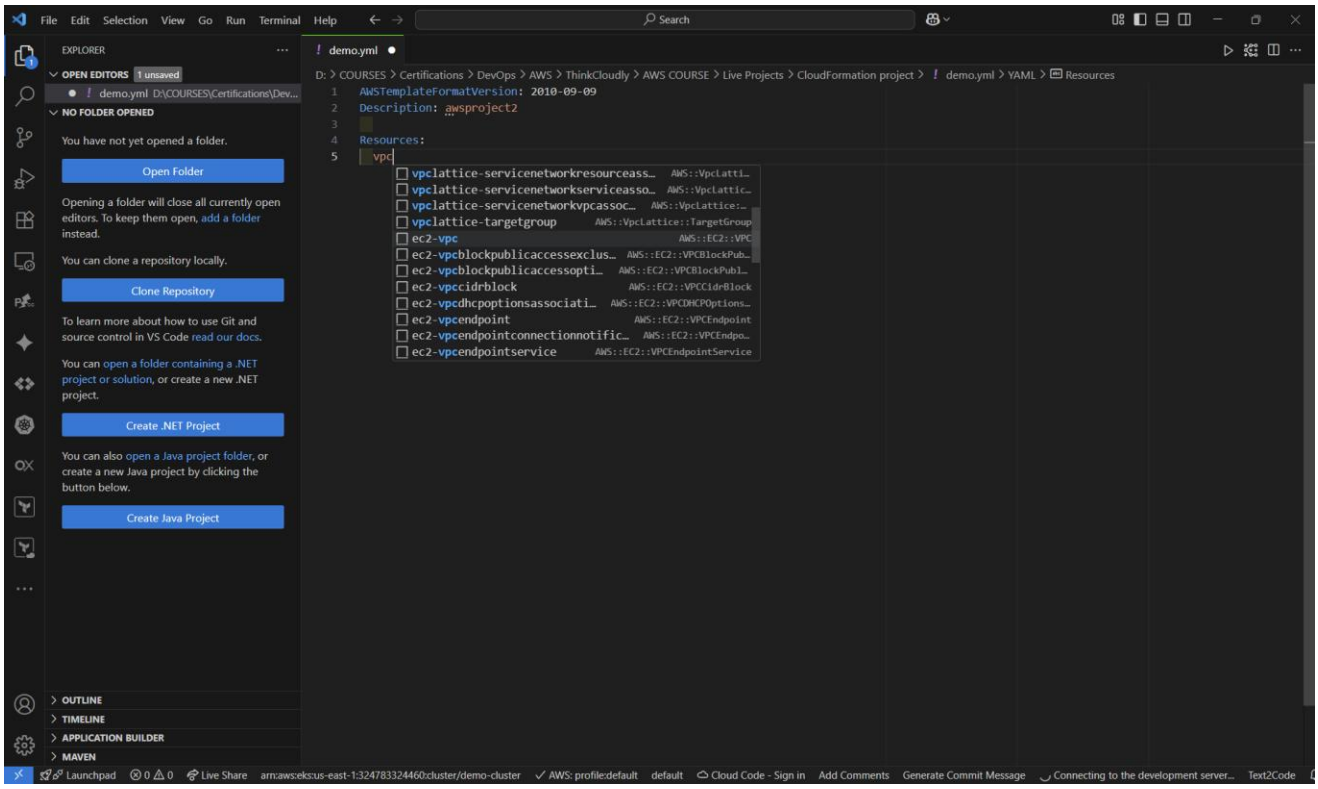
Then press “**Tab**” key once



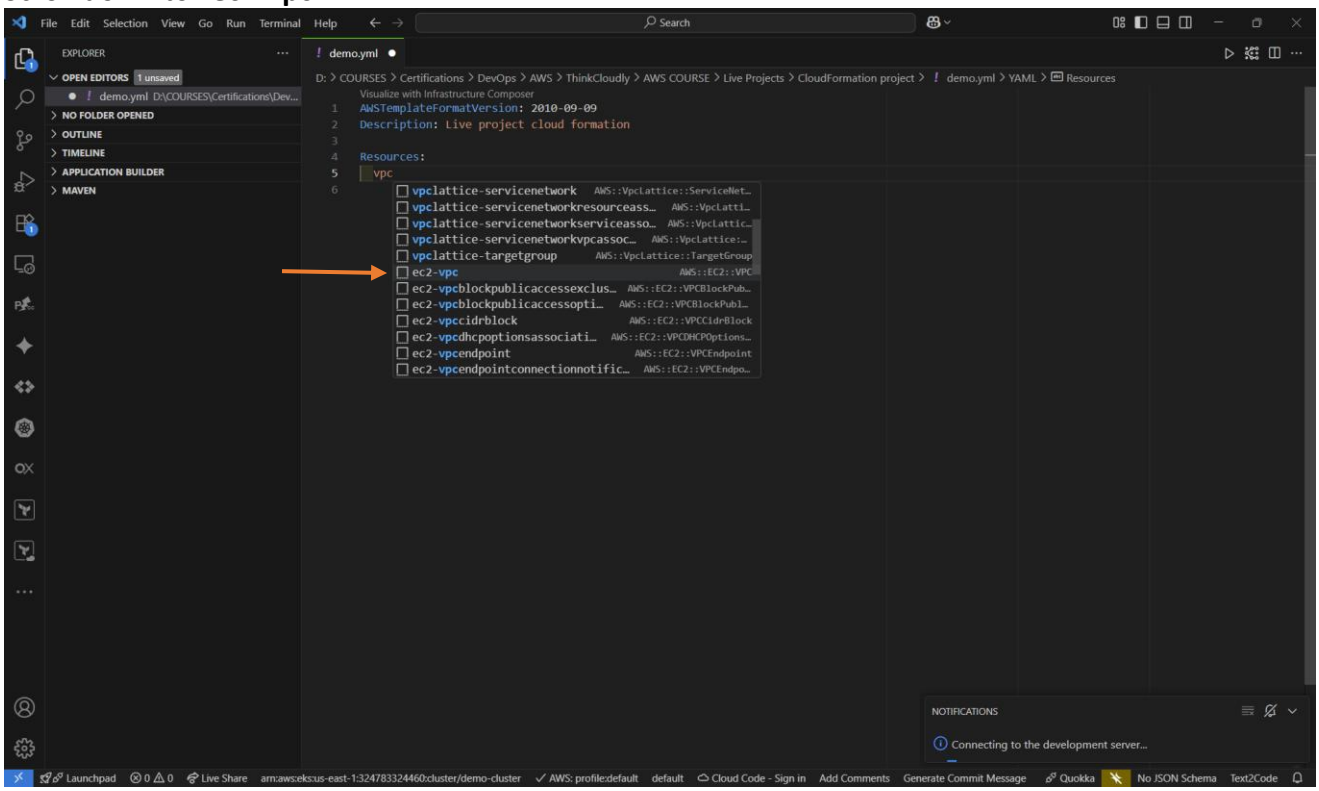
Then type "**vpc**"



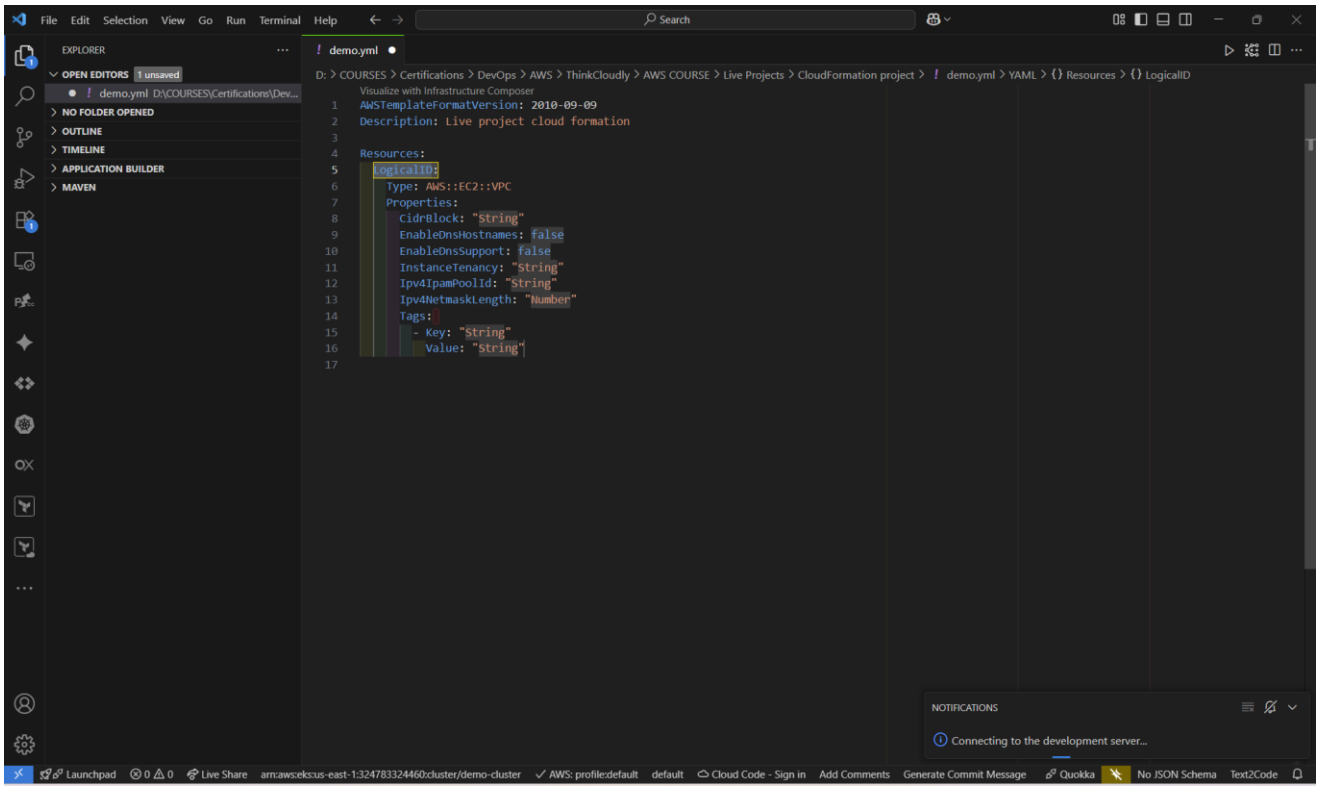
Scroll down



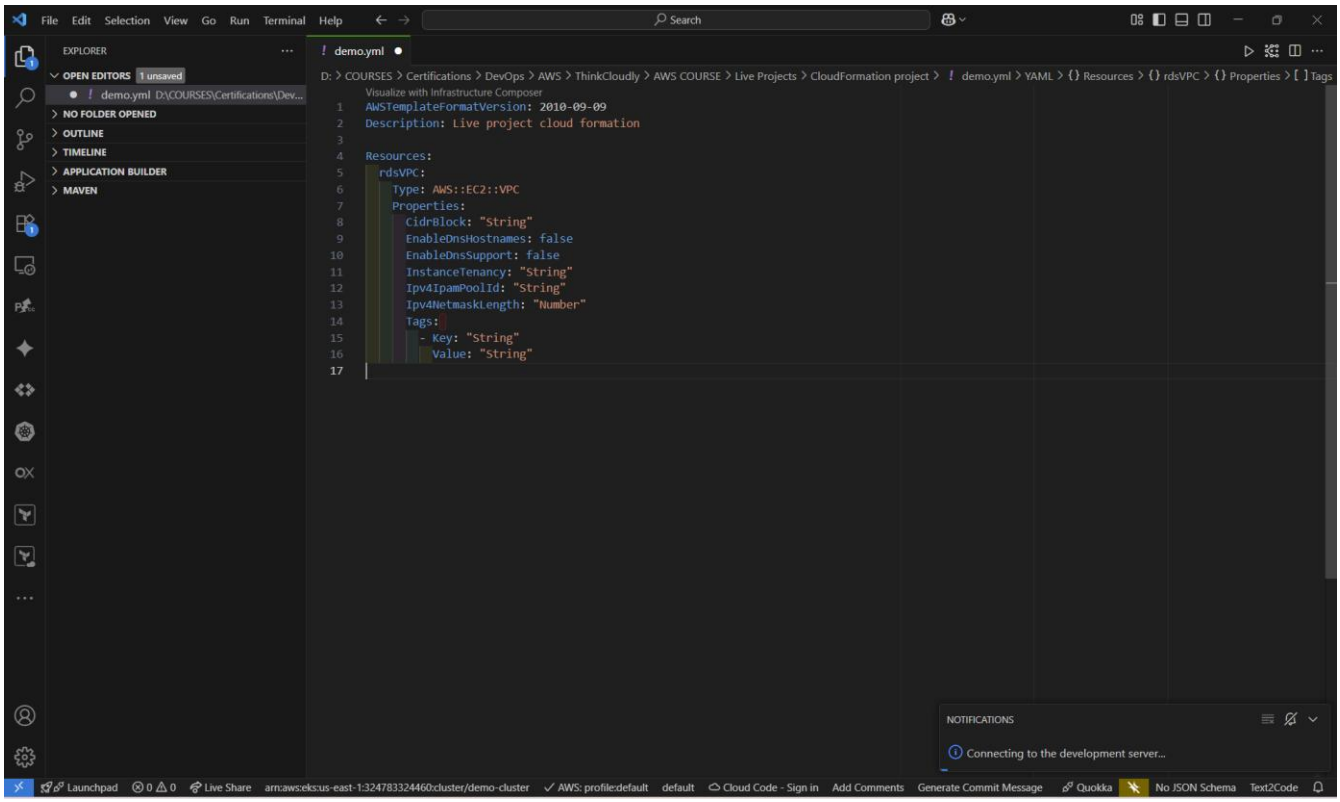
Scroll down to "**ec2-vpc**"



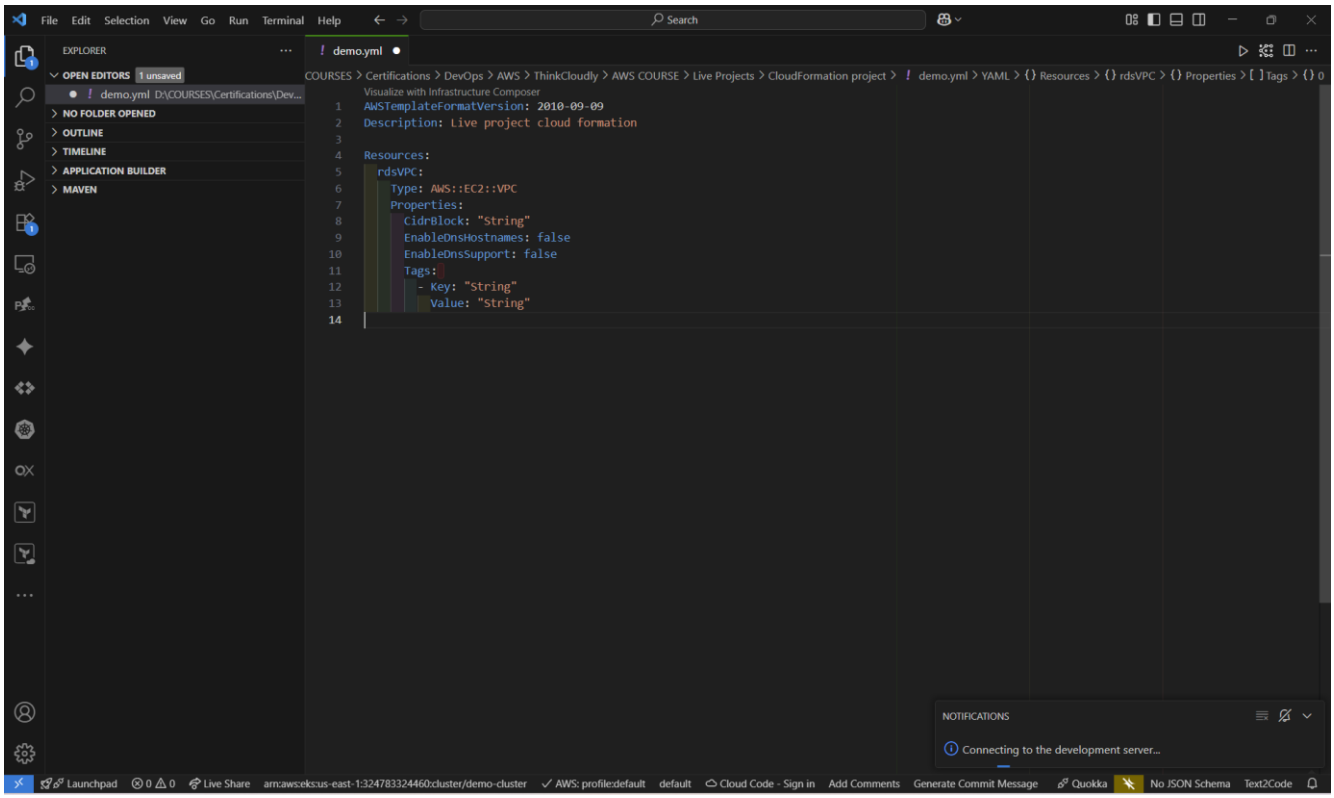
Select "**ec2-vpc**"



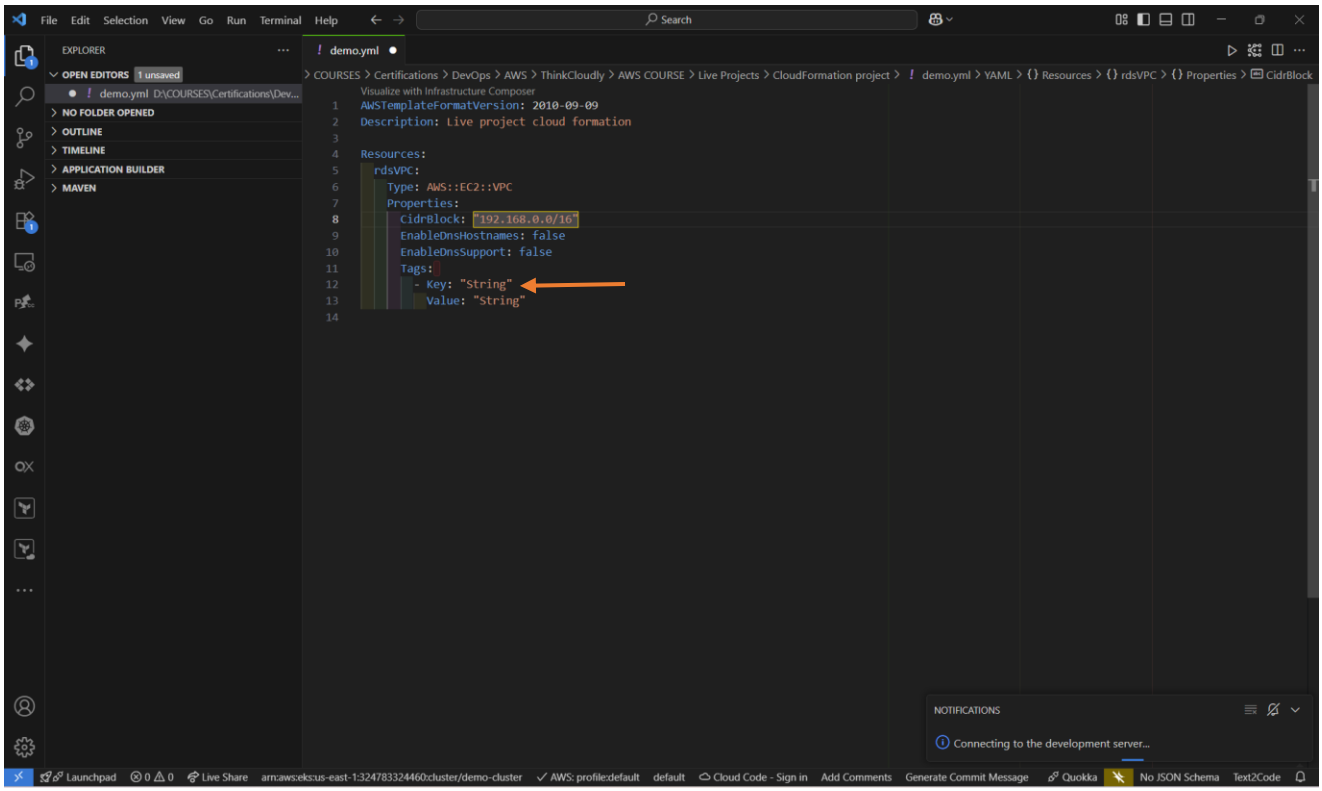
Replace "**logicalID**" with "**rdsvpc**"



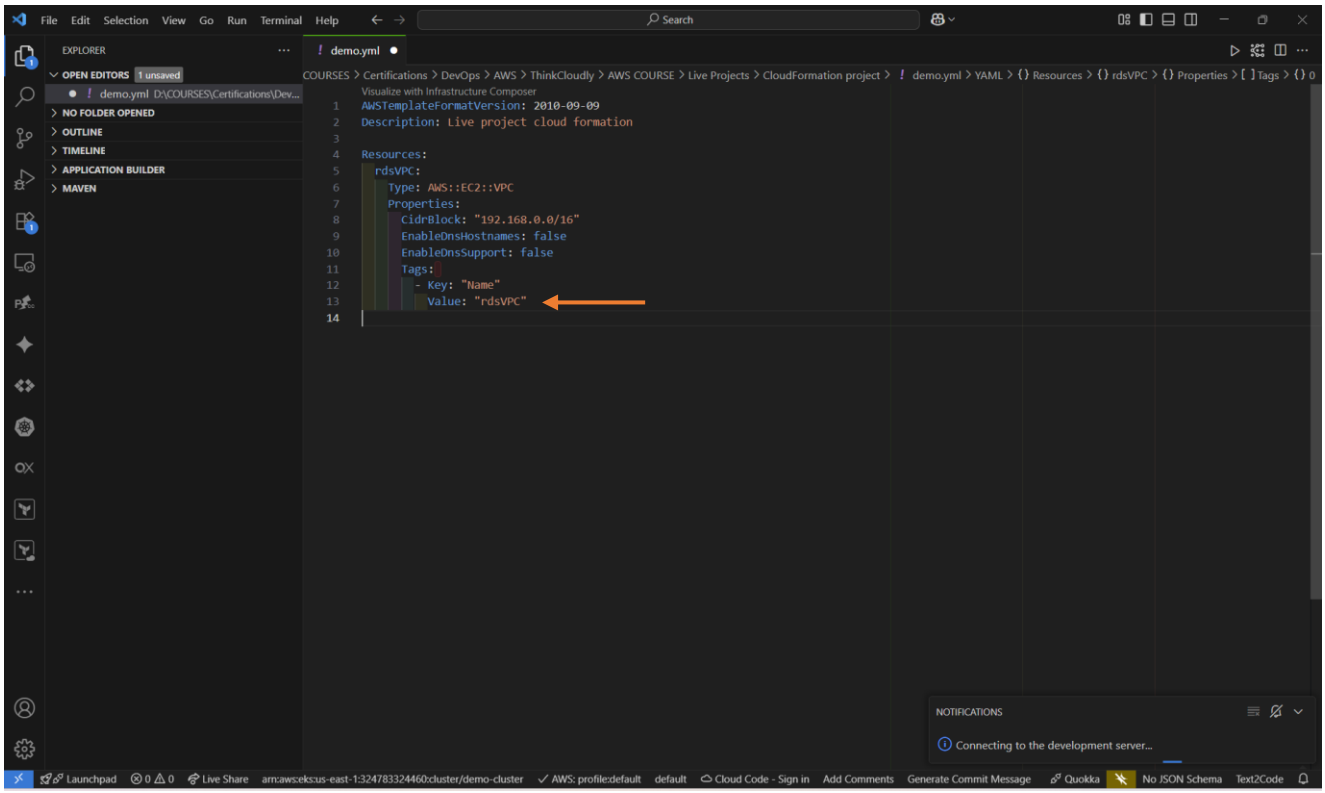
Remove all the lines we do not need. Per our project, we need just "**CIDR Block**", "**EnableDNSHostnames**", and "**EnableDNSSupport**".



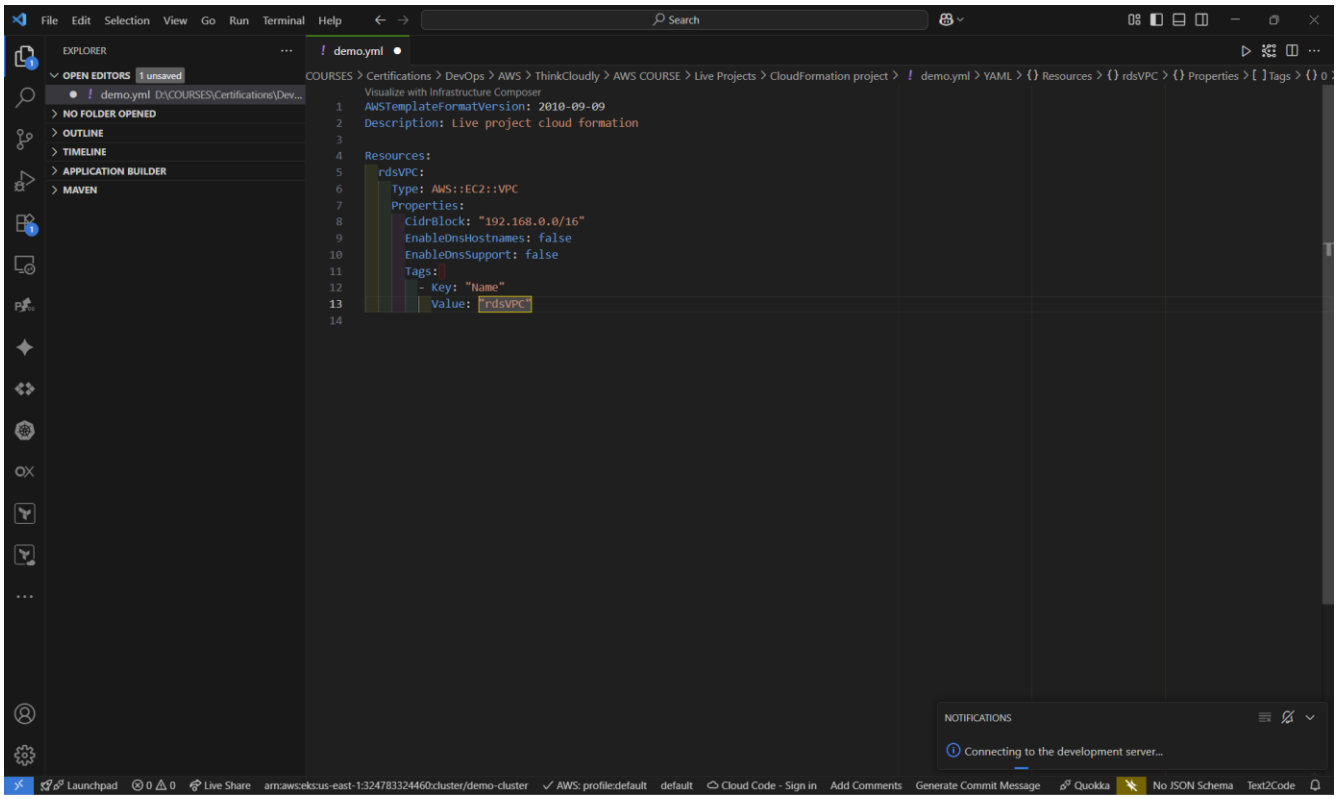
Then enter the details:
CIDR Block: 192.168.0.0/16
EnableDNSHostnames: False
EnableDNSHostnames: False



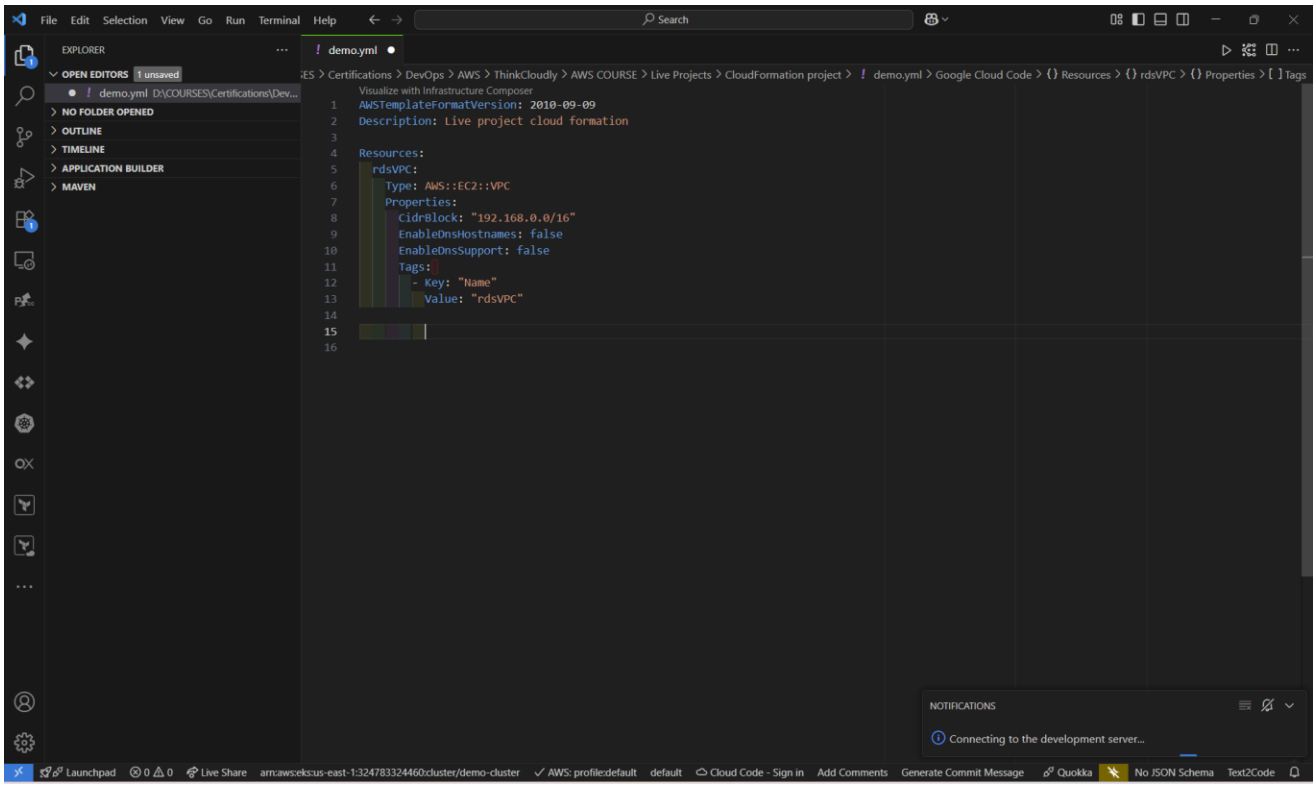
On "**key**", enter "**Name**" and on "**value**", enter the value "**rdsVPC**"



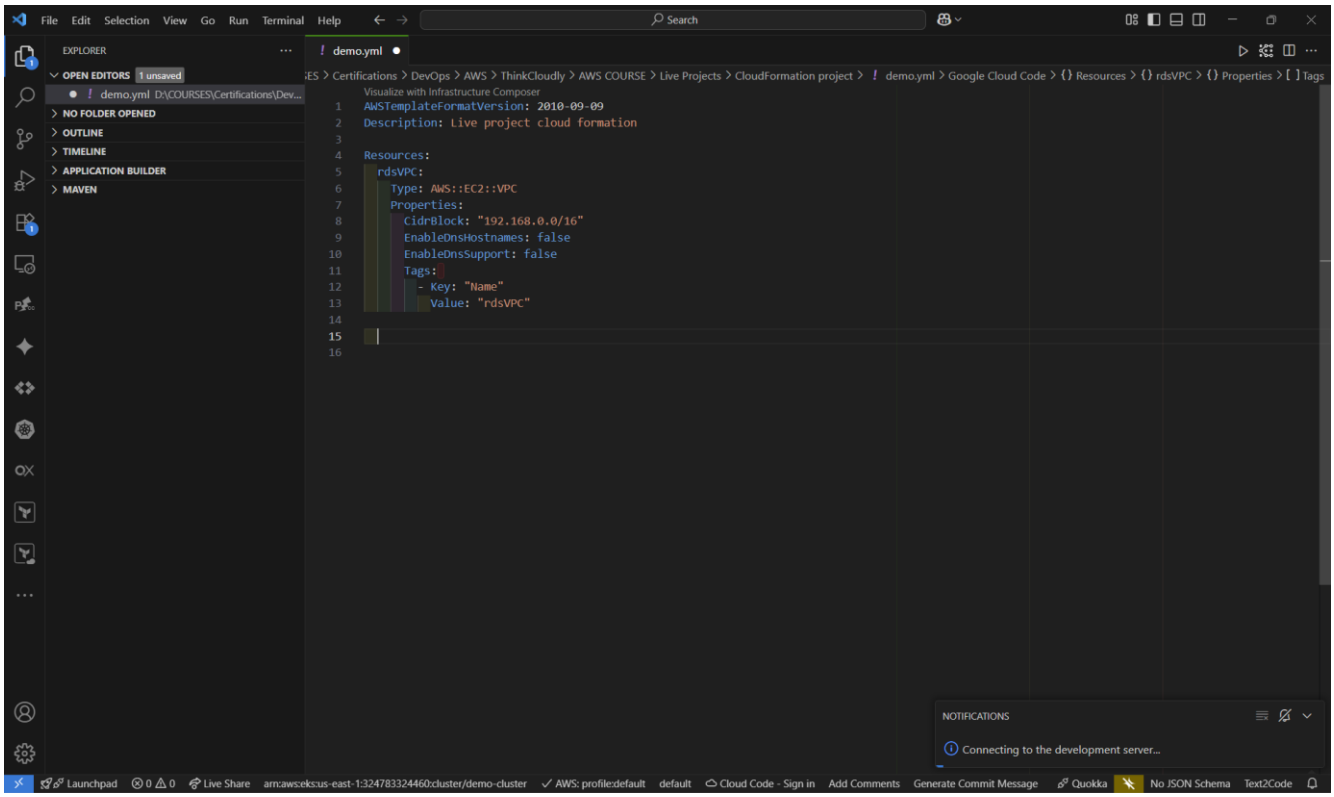
Put the cursor at the end of the value line



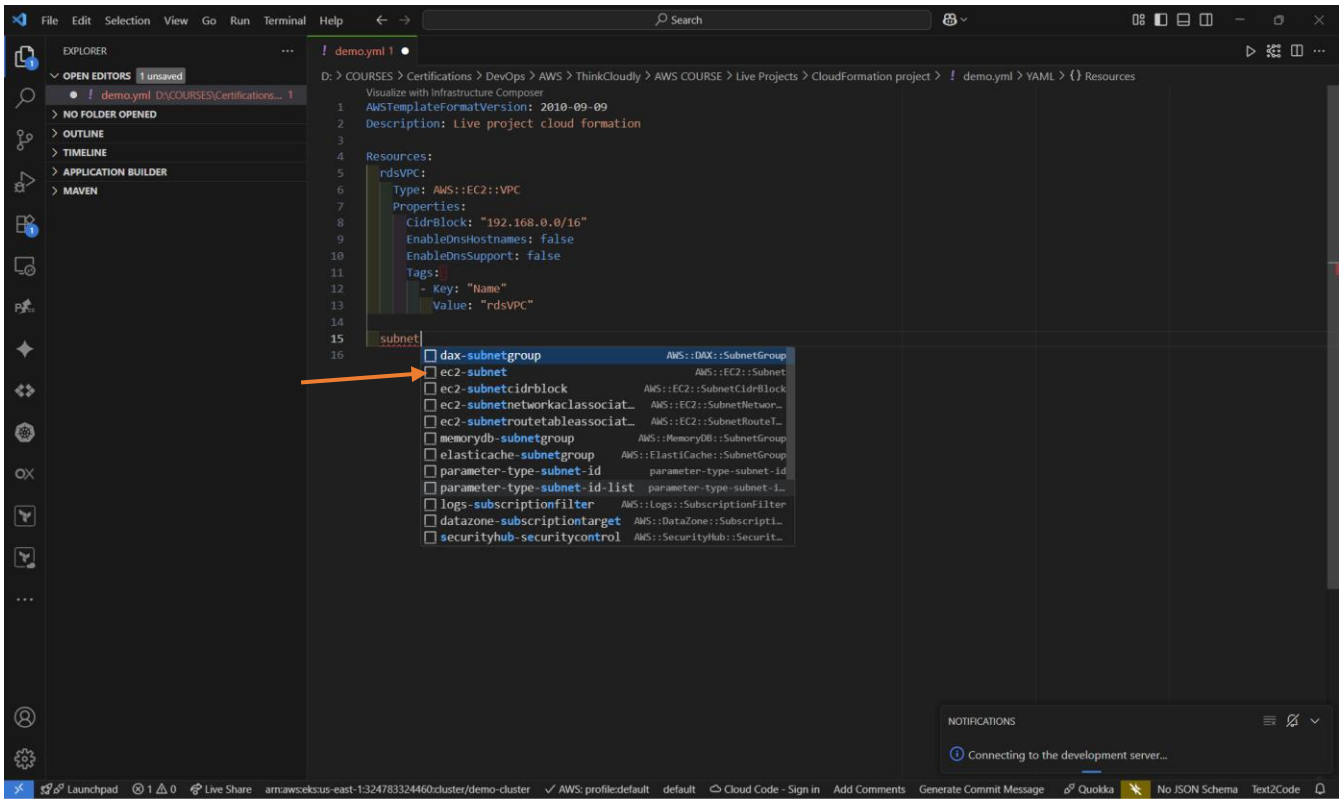
Press Enter twice



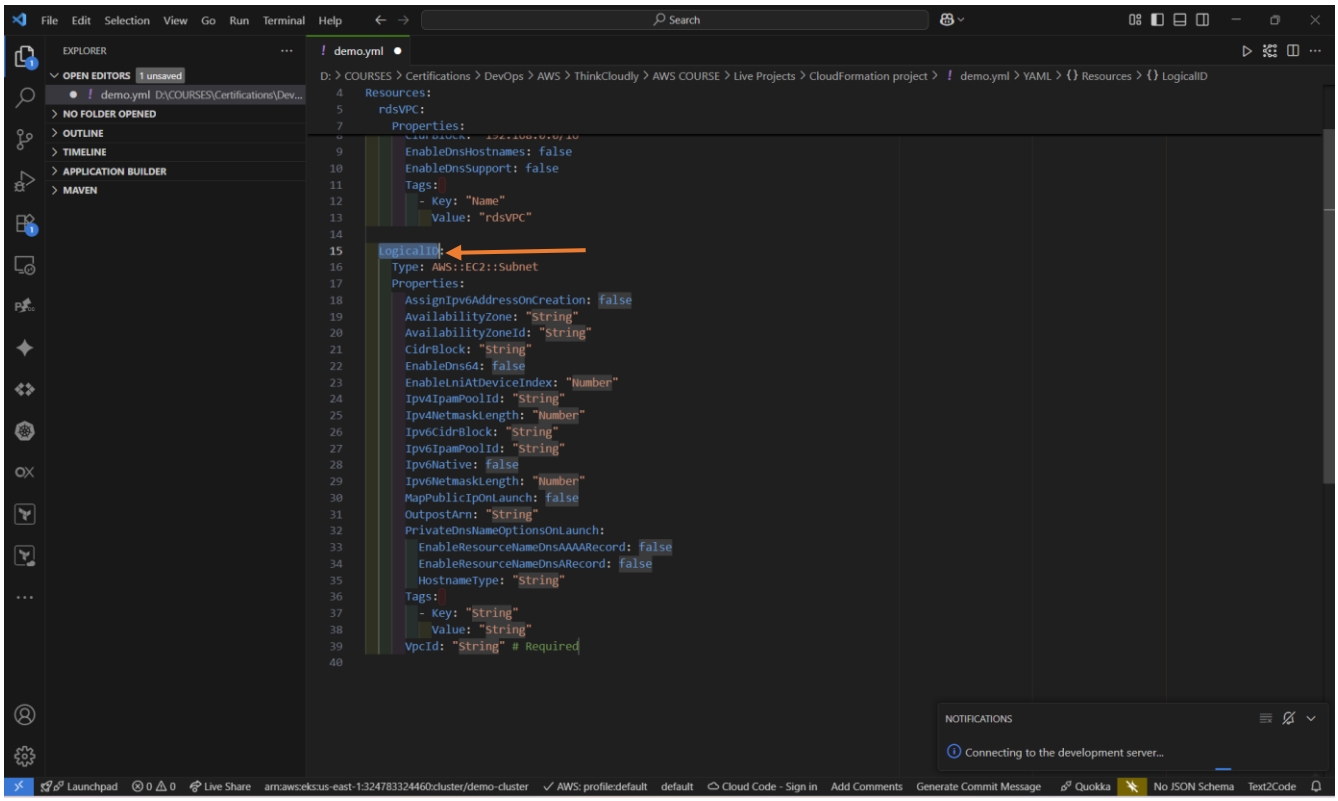
Move the cursor backward under the line "**rds_VPC**" using the "**Backspace**" key



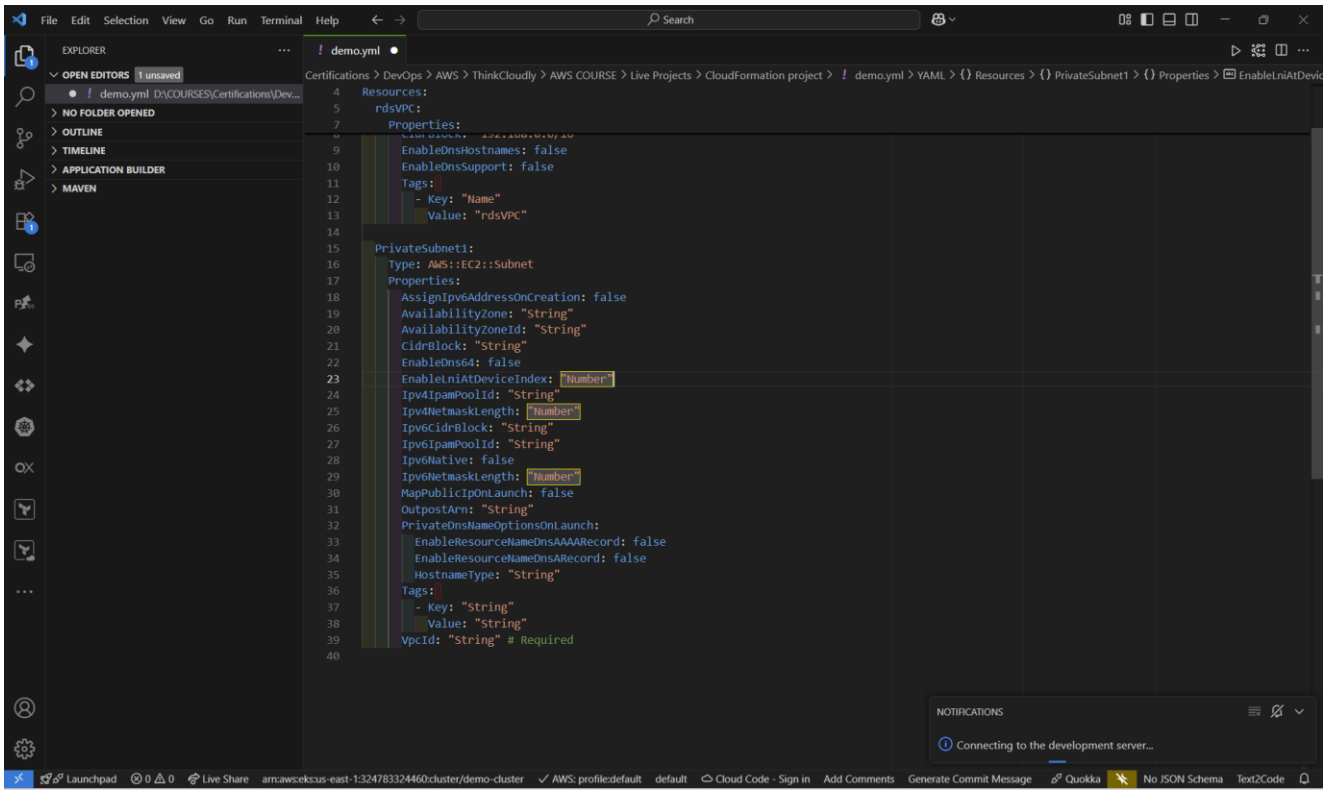
Type "**subnet**"



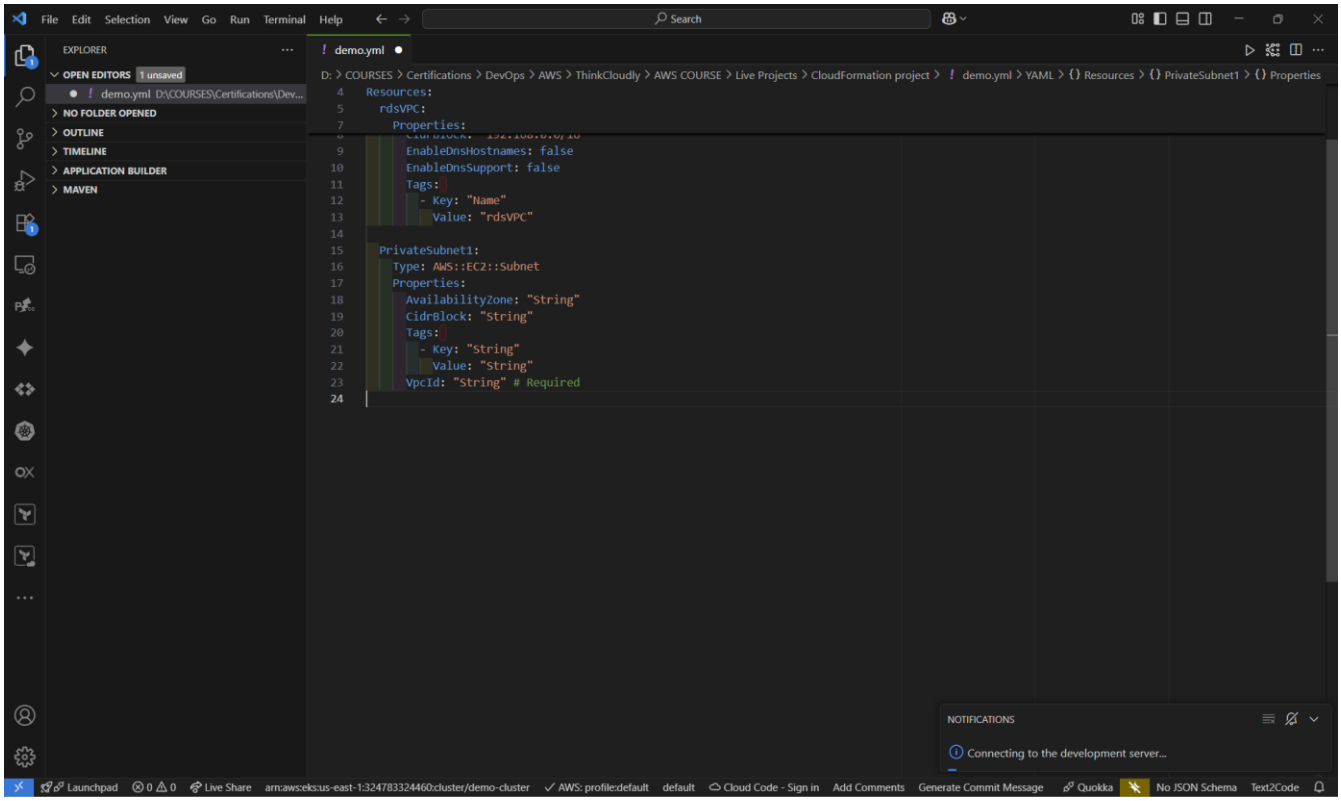
Select "**ec2-subnet**"



On "**logicalID**", enter "**PrivateSubnet1**"



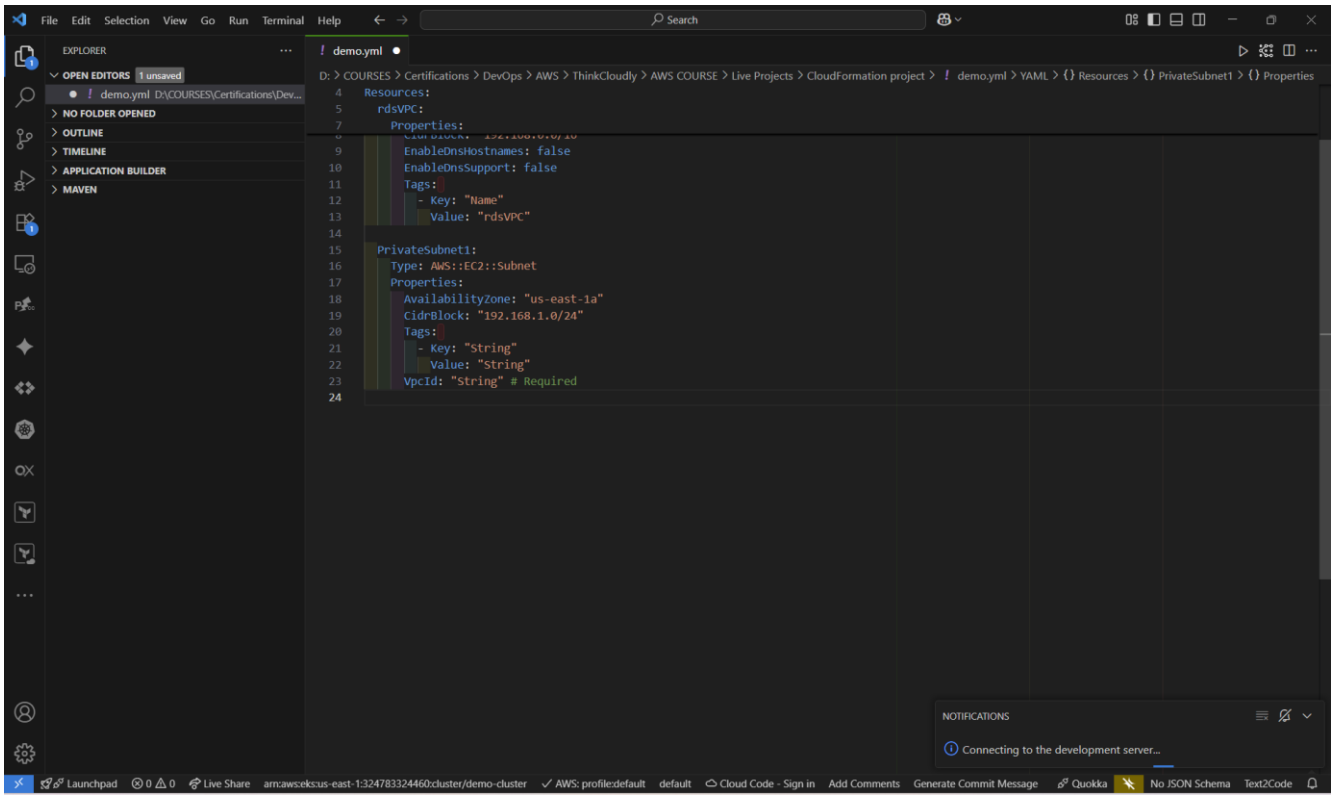
Remove the lines that are not needed.



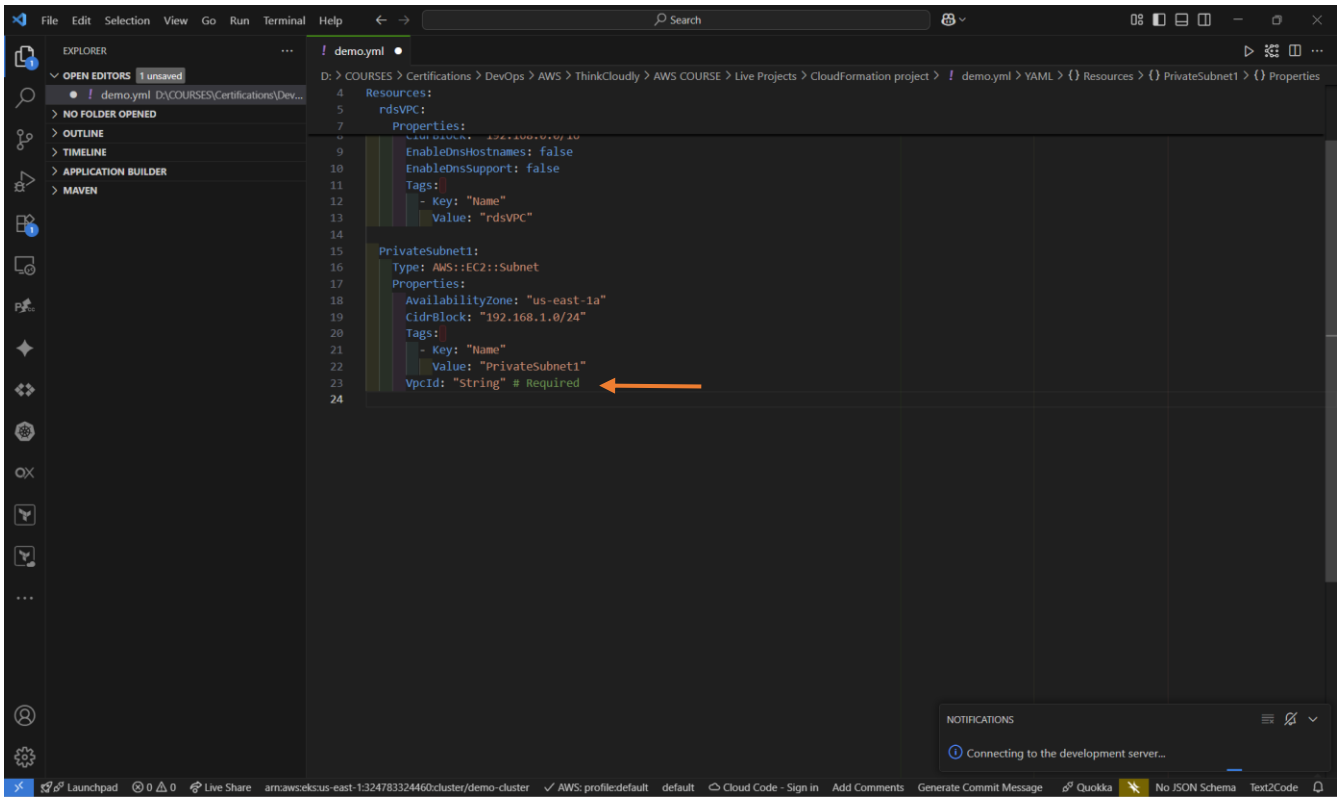
Enter these details:

AvailabilityZone: us-east-1a

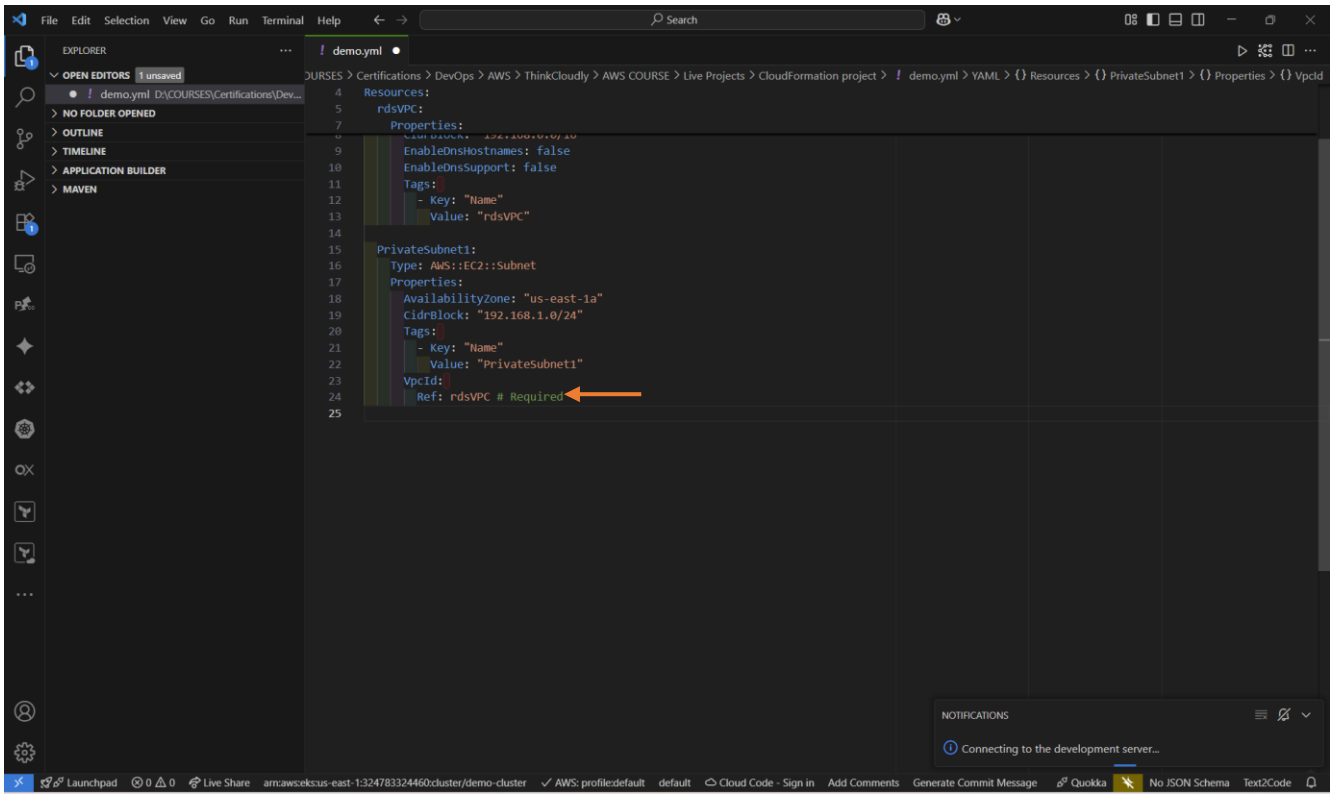
CidrBlock: 192.168.1.0/24



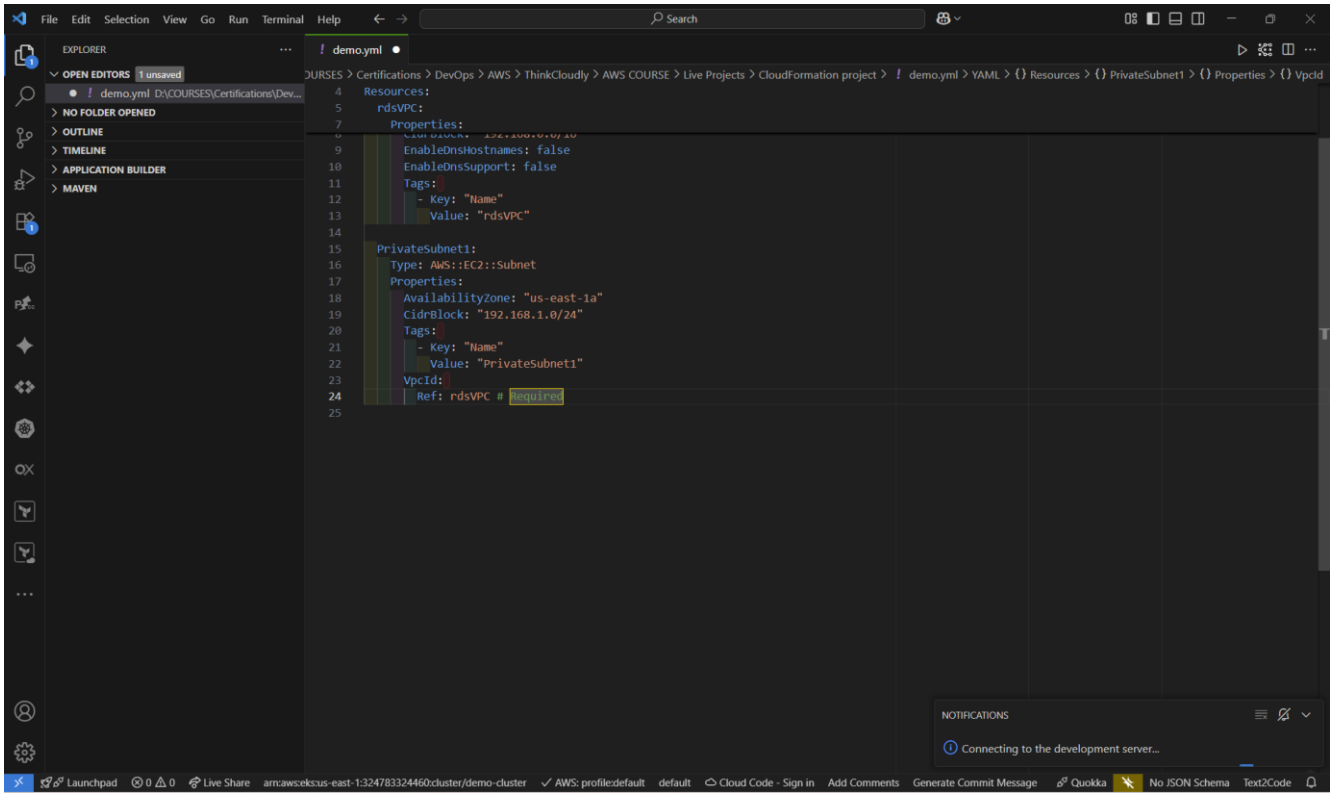
Under "**Tag**", in "**key**" enter "**Name**" and in "**value**" enter "**PrivateSubnet1**"



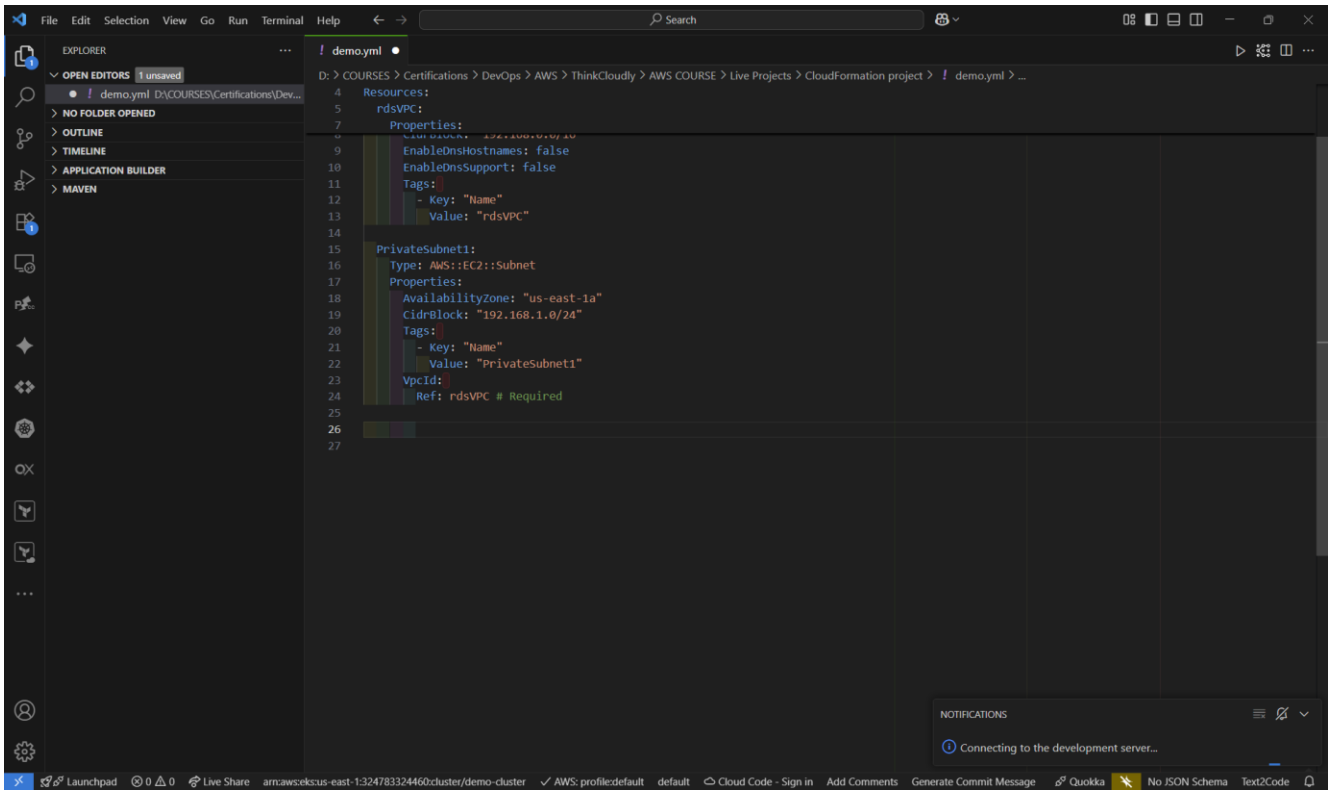
On "**vpcId**" enter "**Ref: rdsvpc**". The Ref value (**rdsVPC**) should match with the name of the VPC (**rdsVPC**)



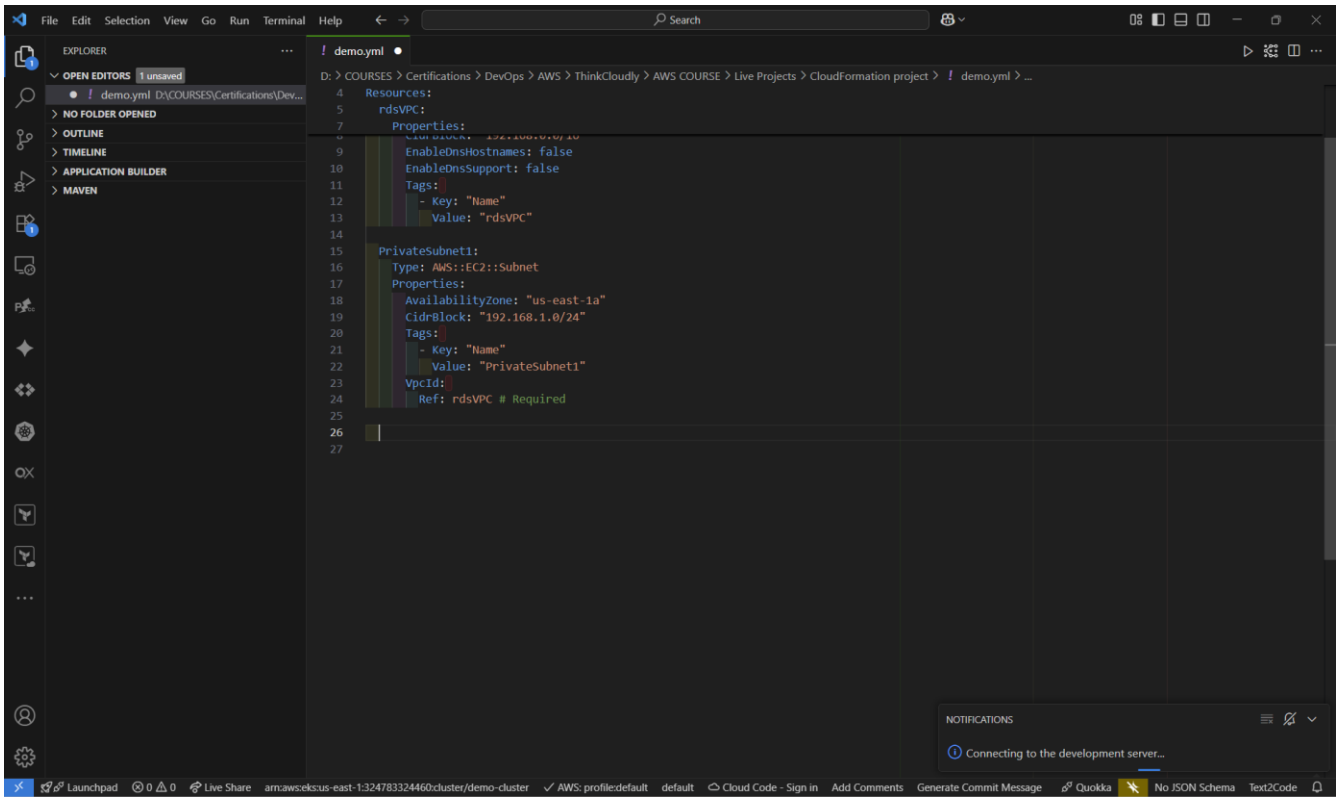
Put your cursor at the end of the "**vpcId**" line



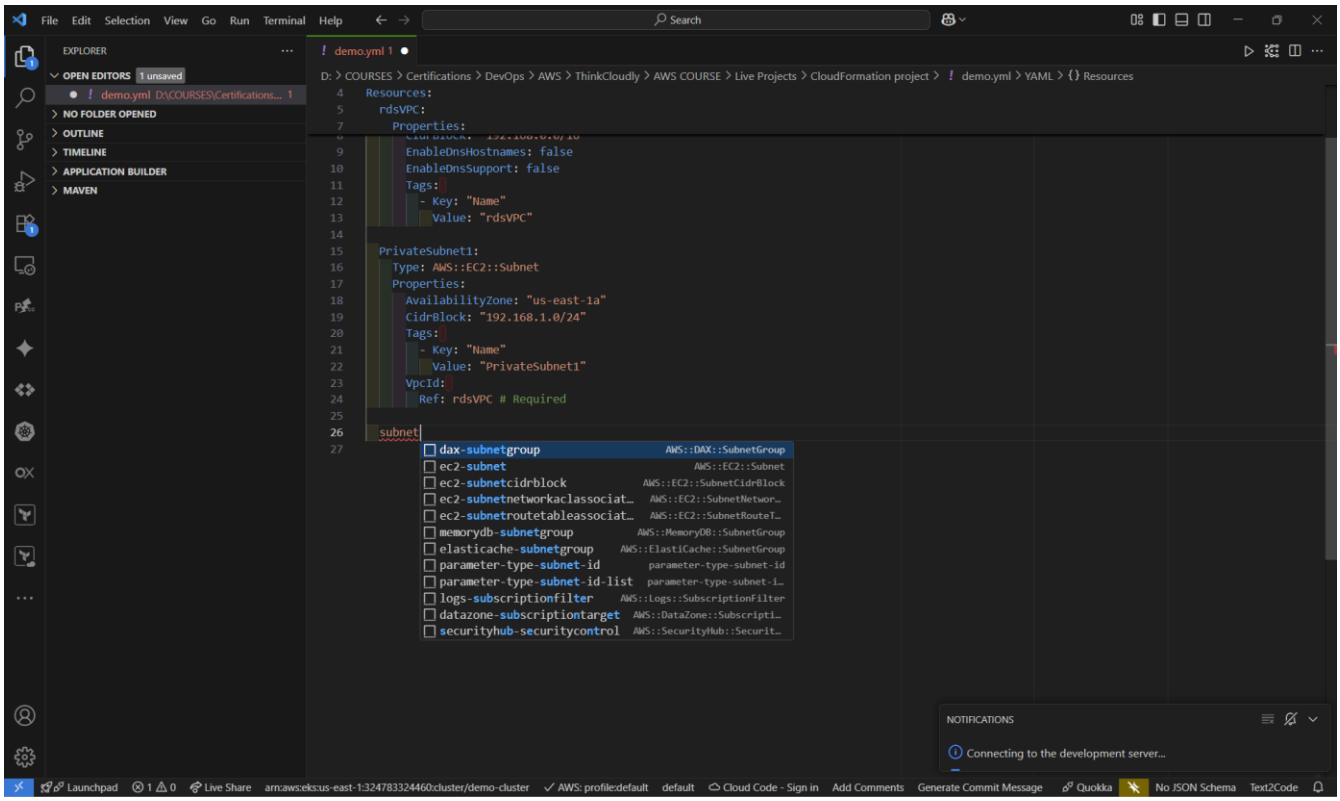
Press ENTER twice



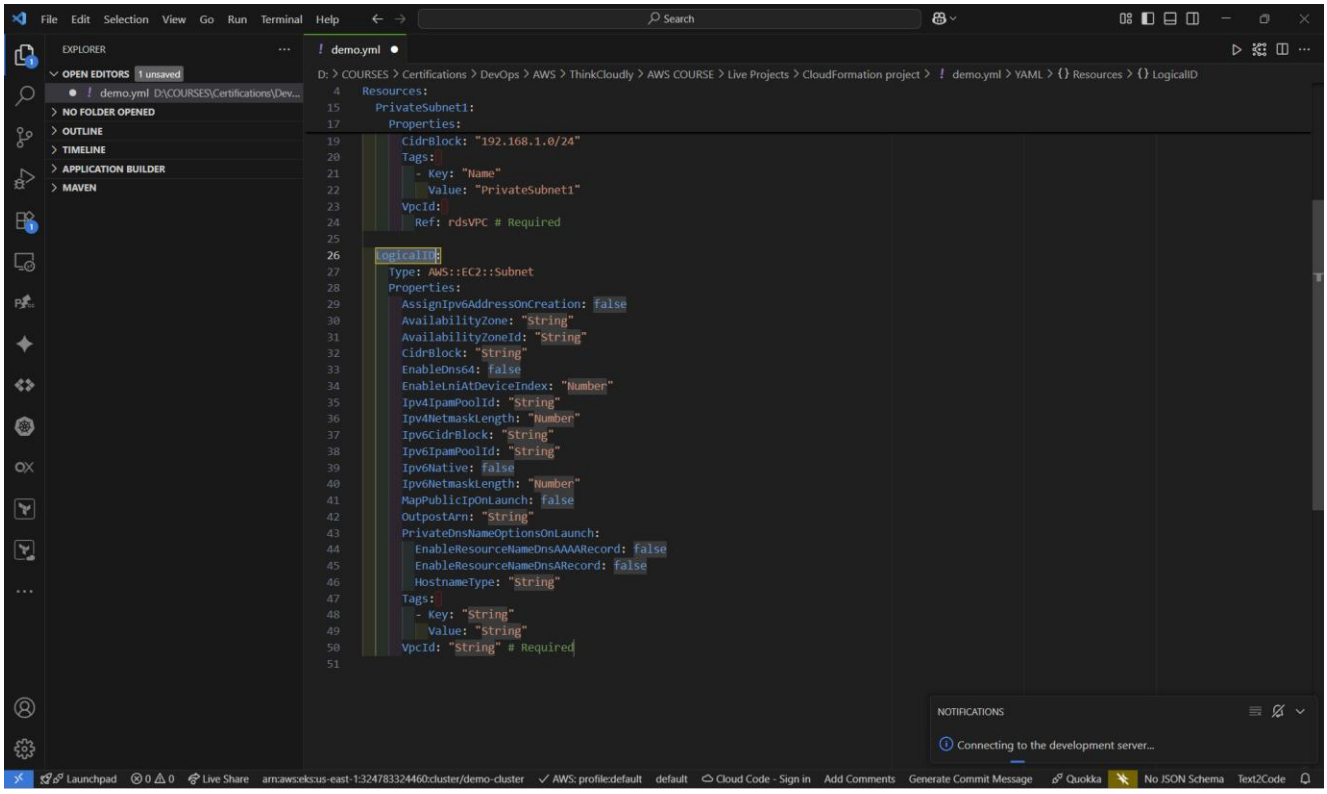
Then move the cursor backward to the line "**PrivateSubnet1**"



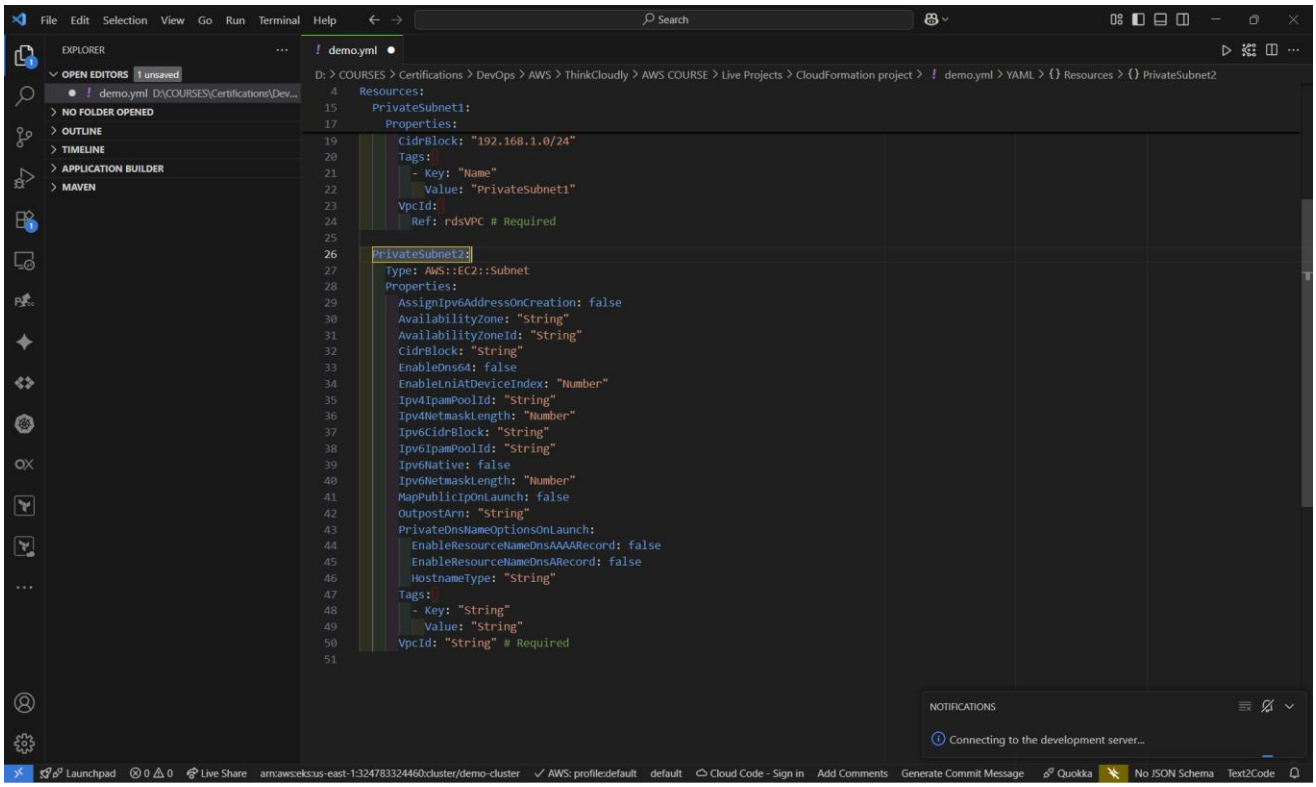
Type "**subnet**"



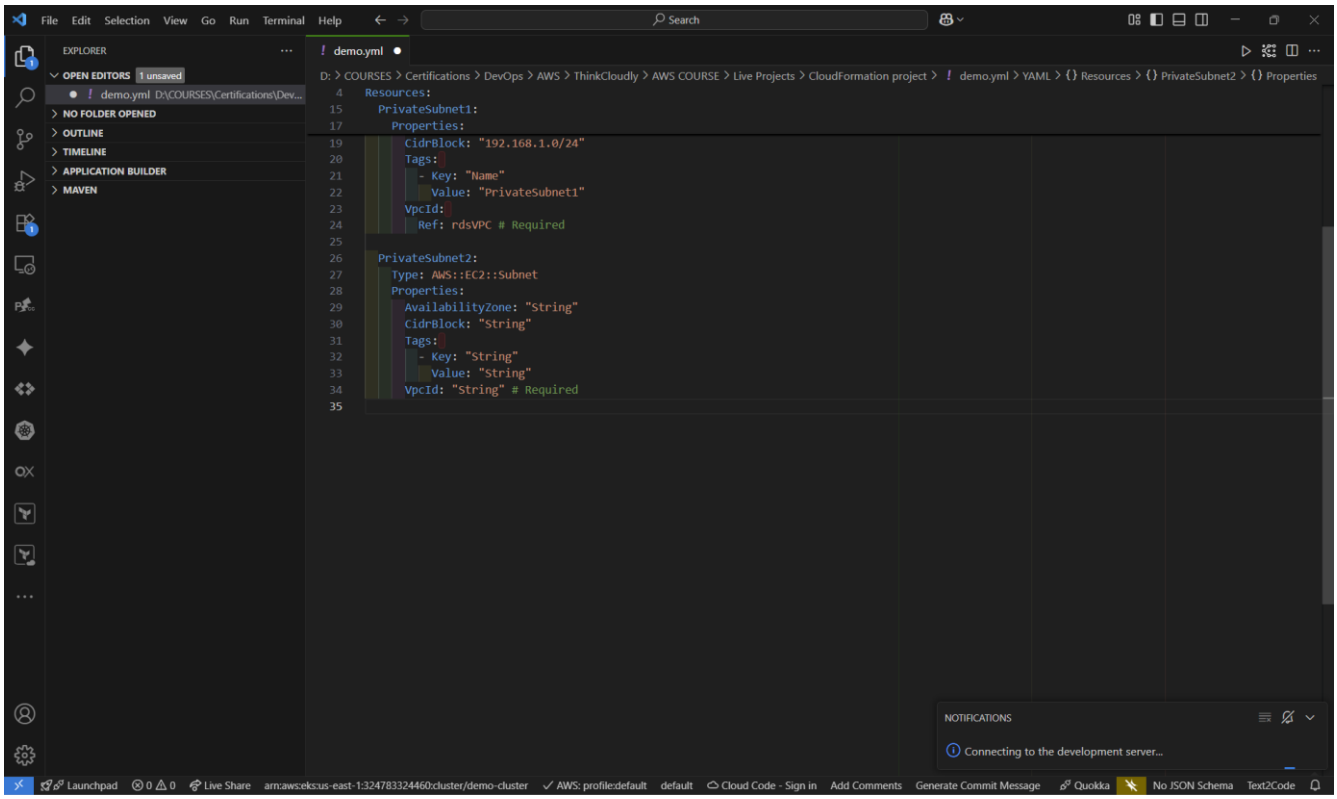
Select "**ec2-subnet**"



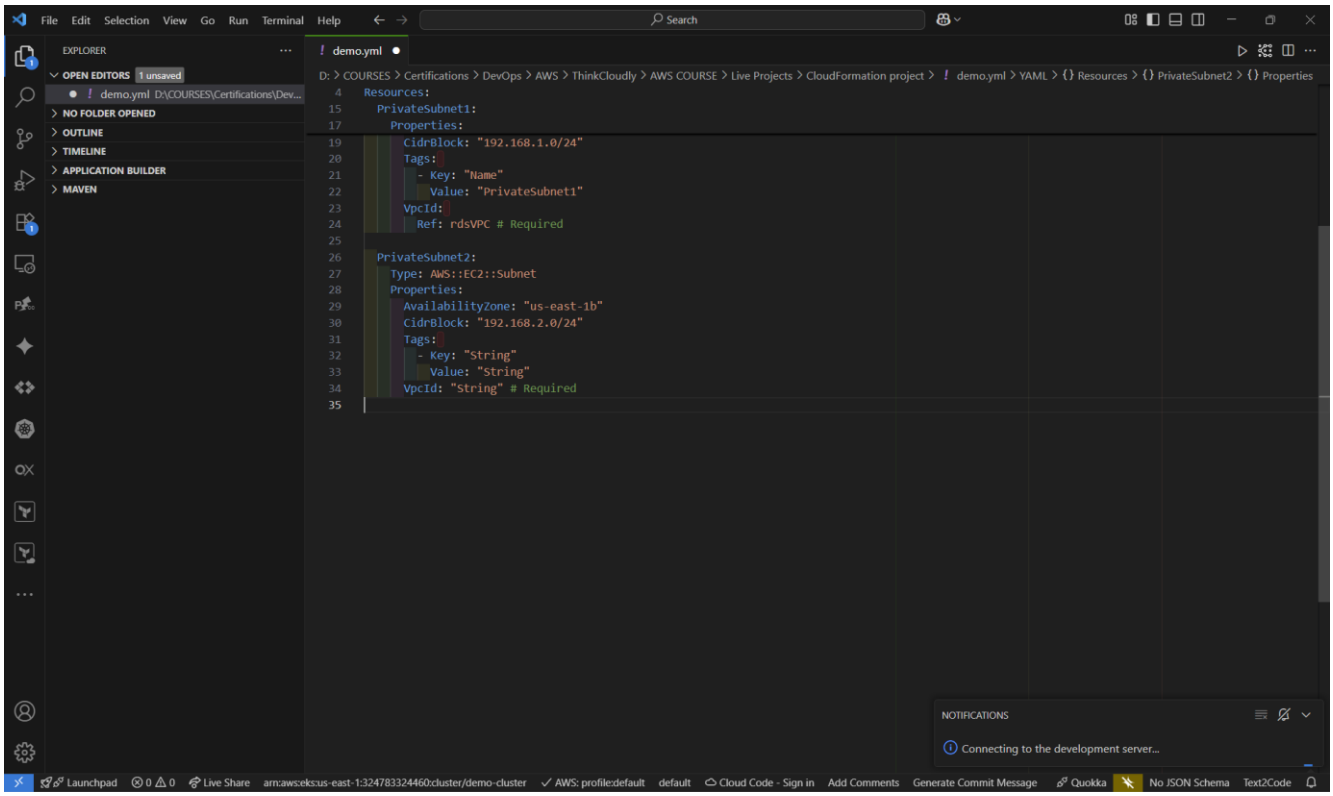
Replace "**logicalID**" with the name "**PrivateSubnet2**"



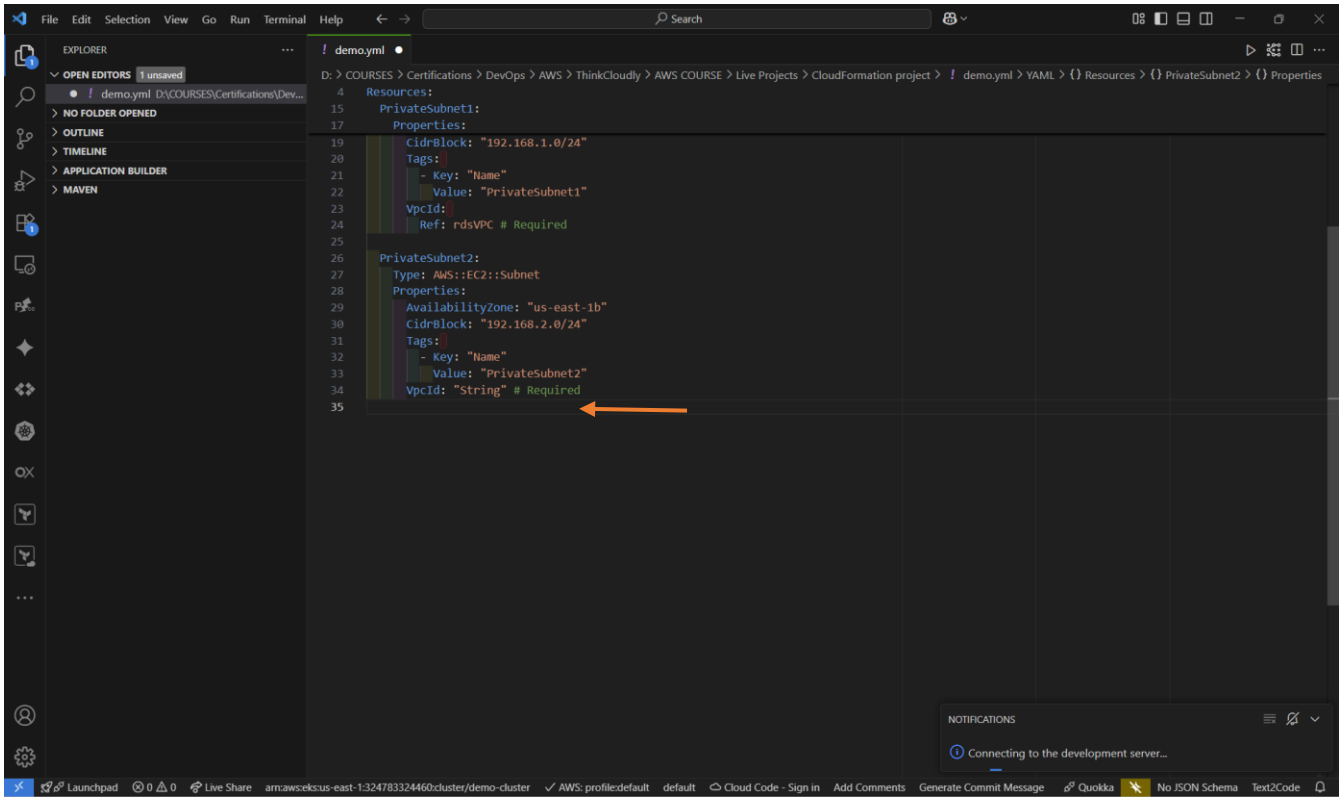
Remove the lines that we do not need



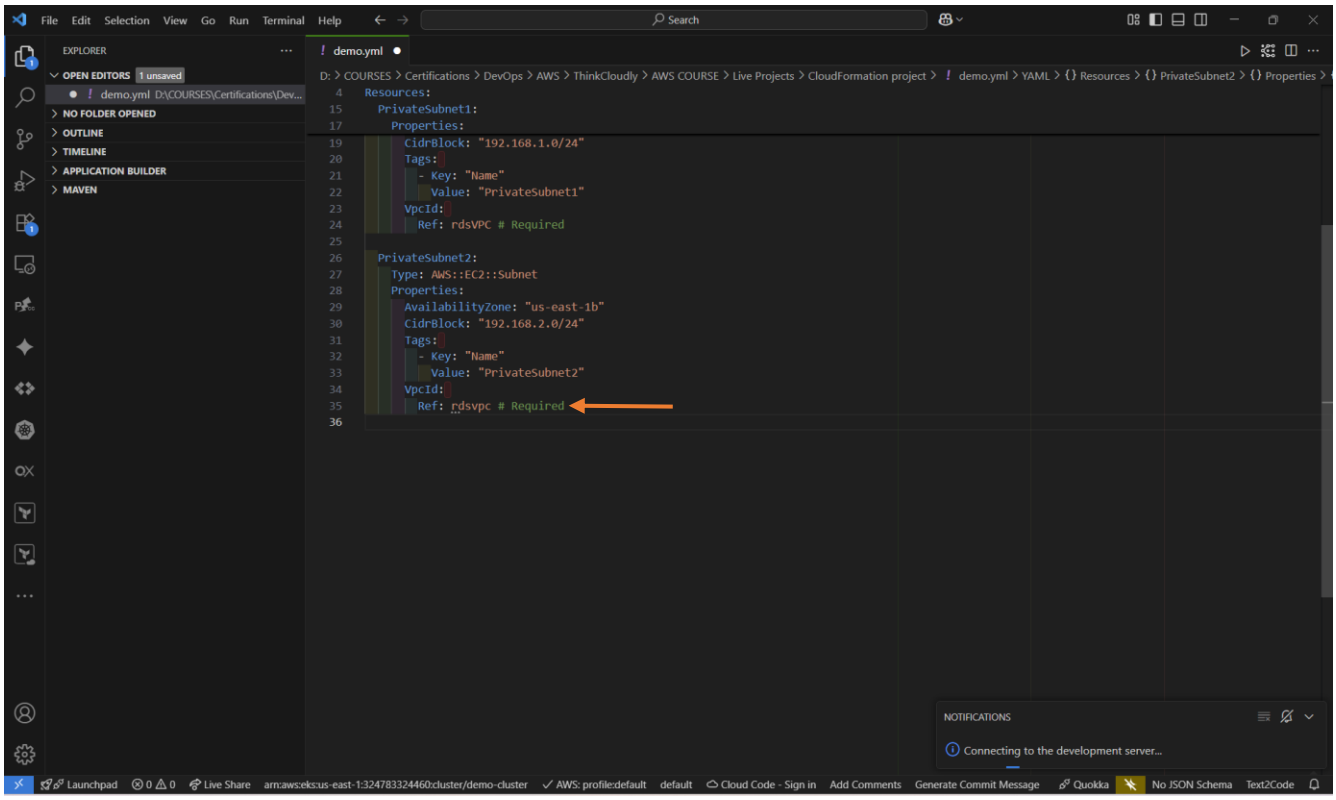
Enter these details:
CidrBlock: us-east-1b
AvailabilityZone: 192.168.2.0/24



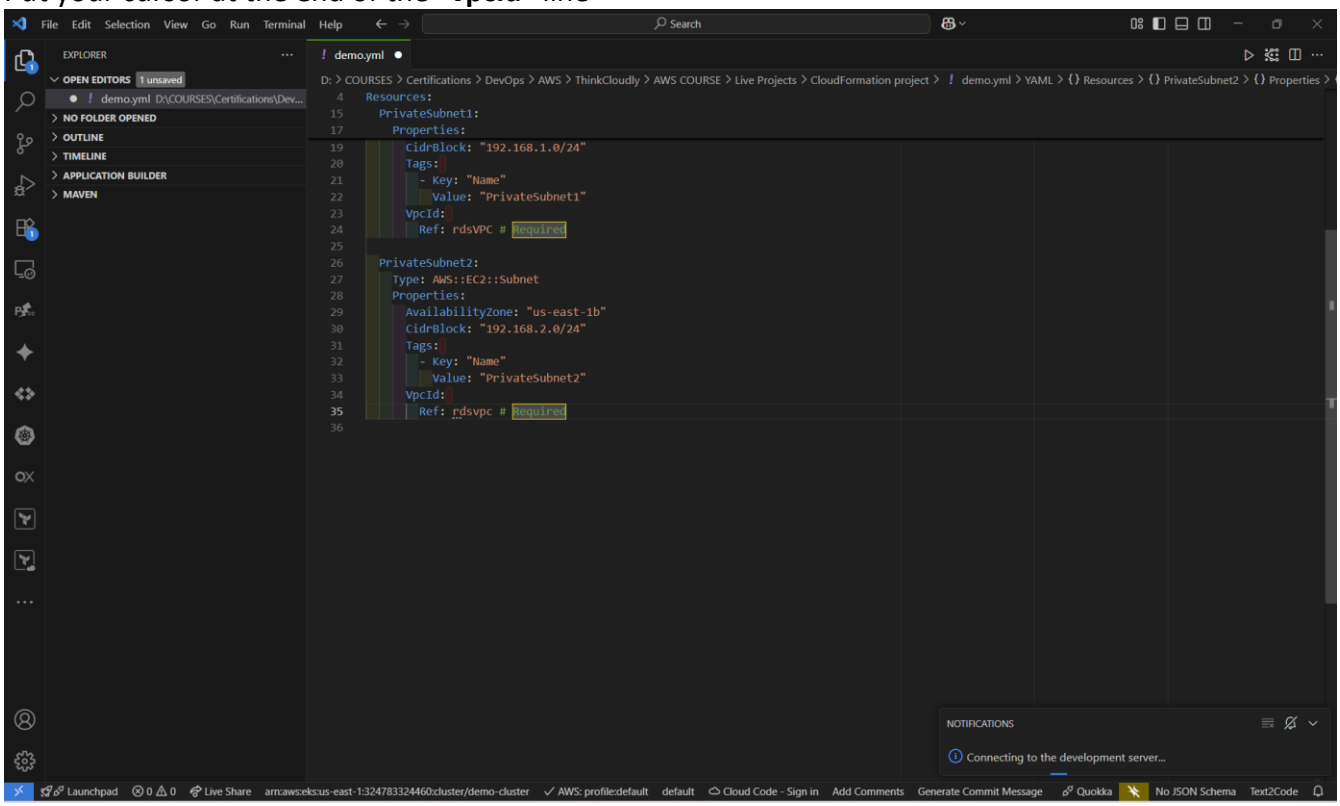
Under "**Tag**", in "**key**" enter "**Name**" and in "**value**" enter "**PrivateSubnet2**"



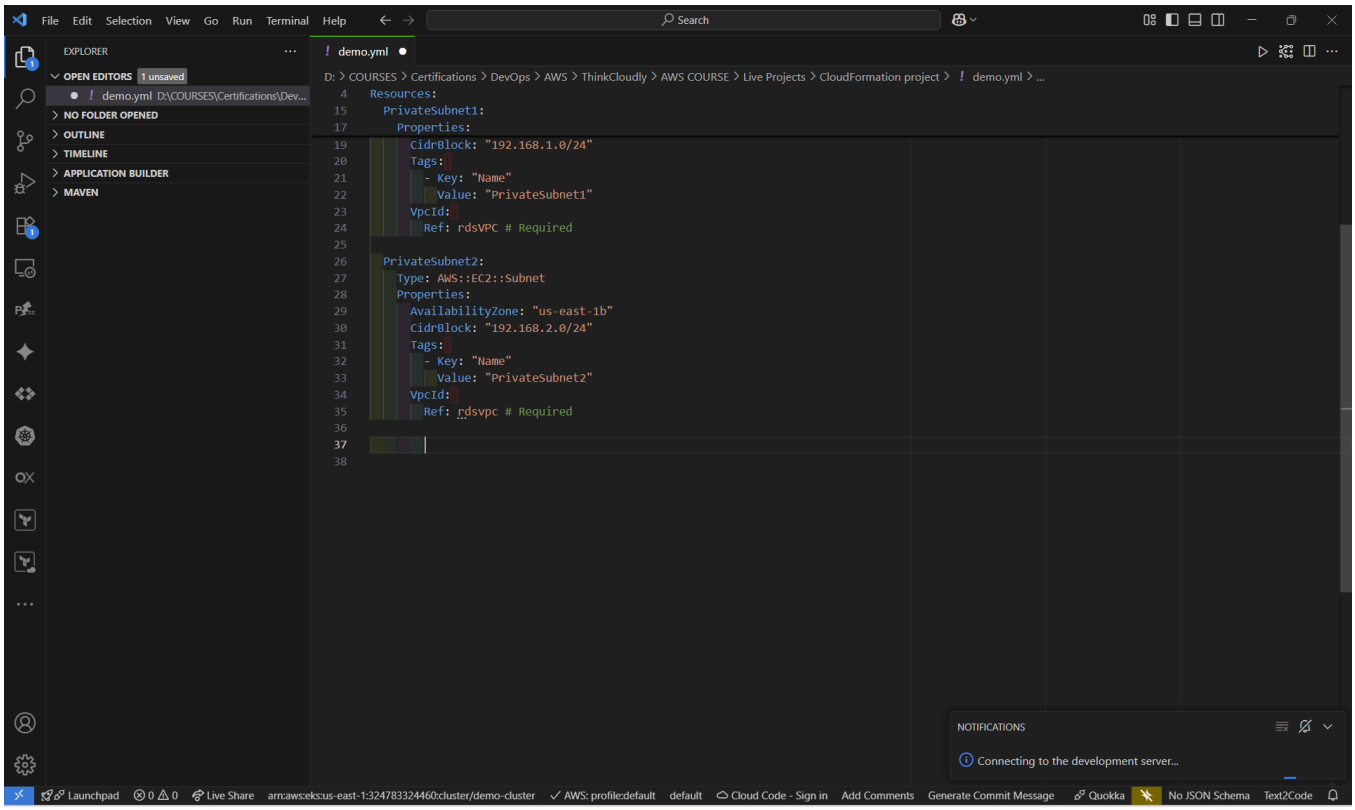
On "**vpcId**" enter "**Ref: rdsvpc**"



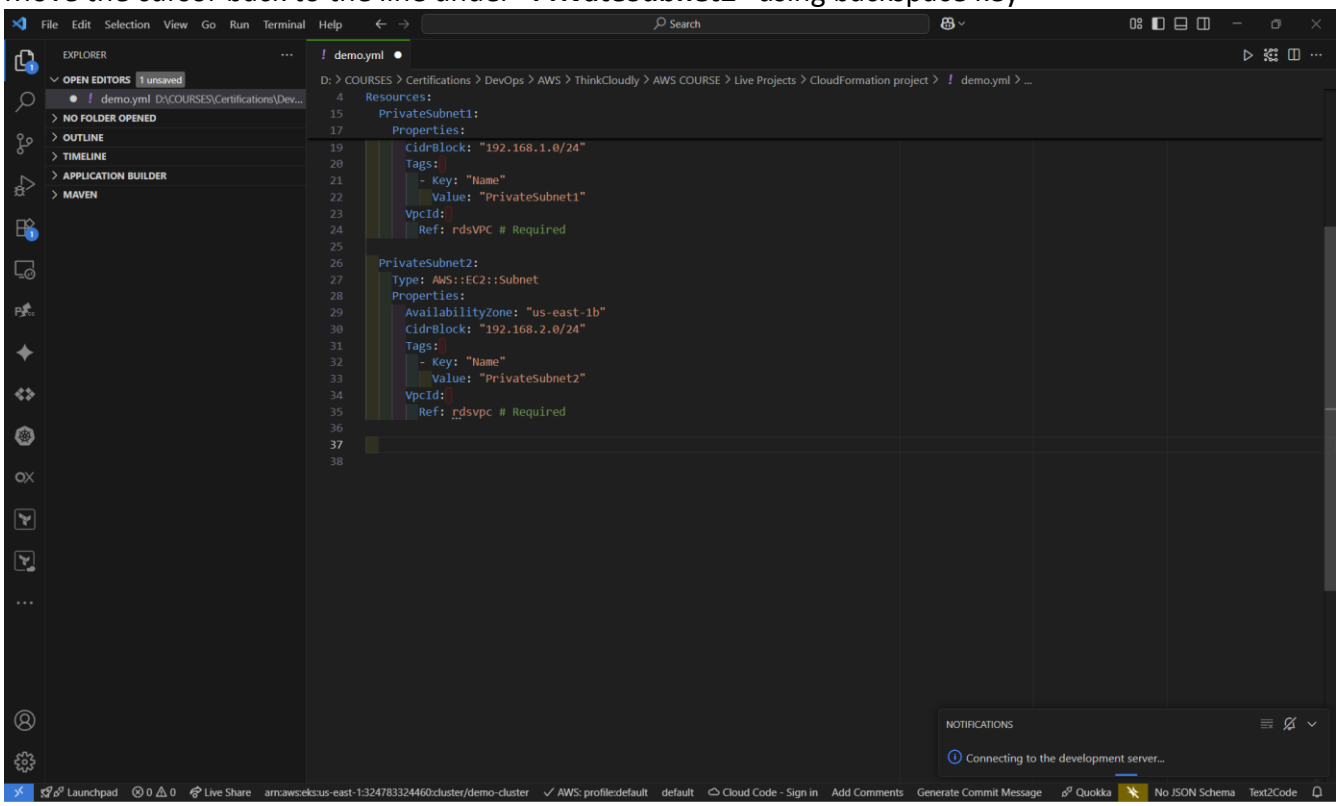
Put your cursor at the end of the "**vpcId**" line



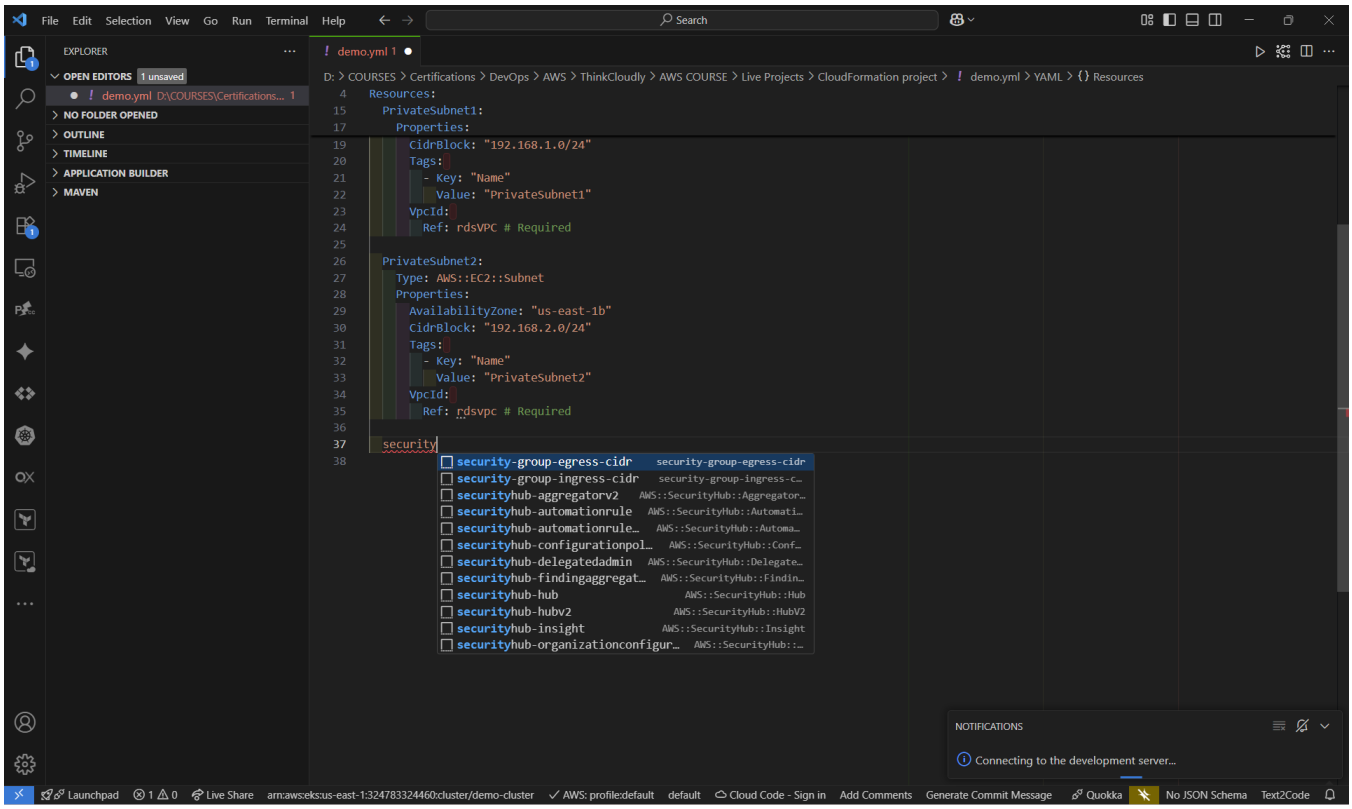
Press Enter twice



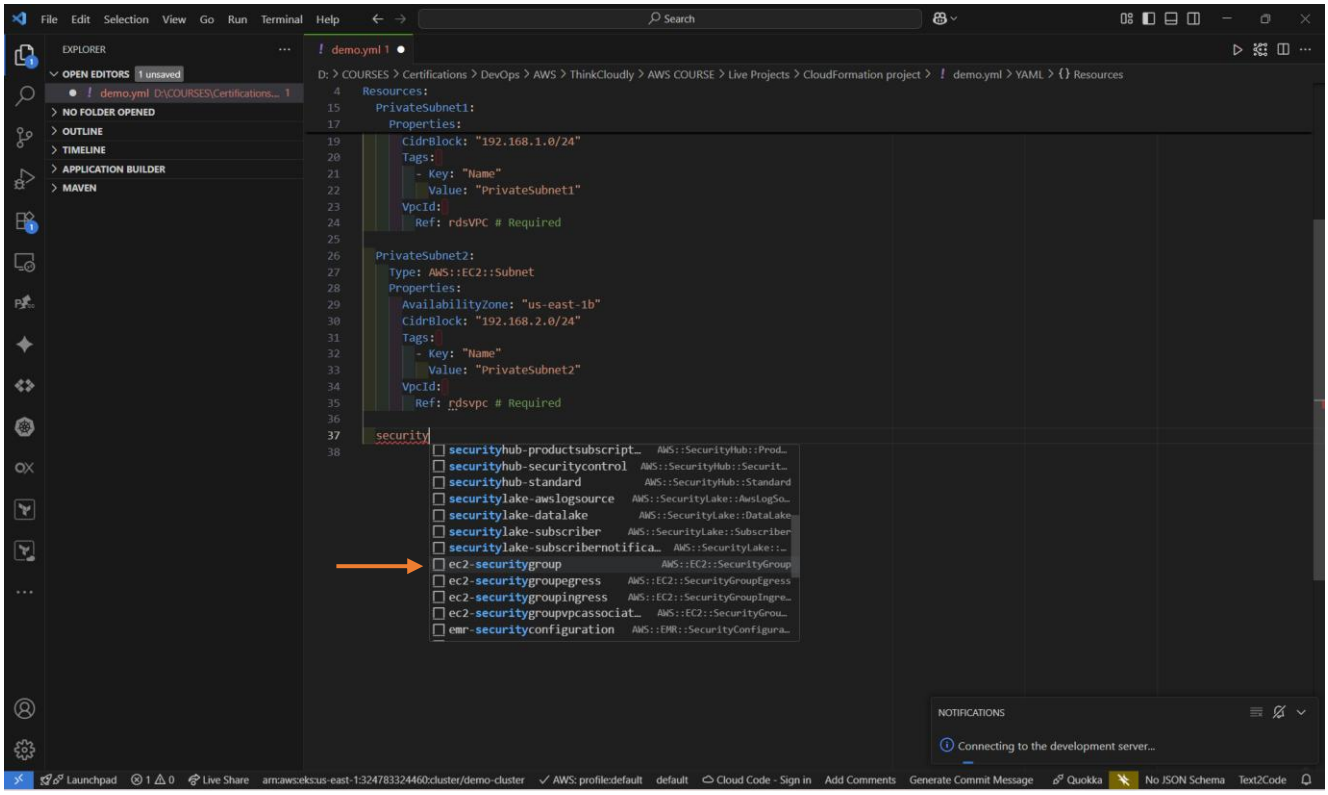
Move the cursor back to the line under "**PrivateSubnet2**" using backspace key



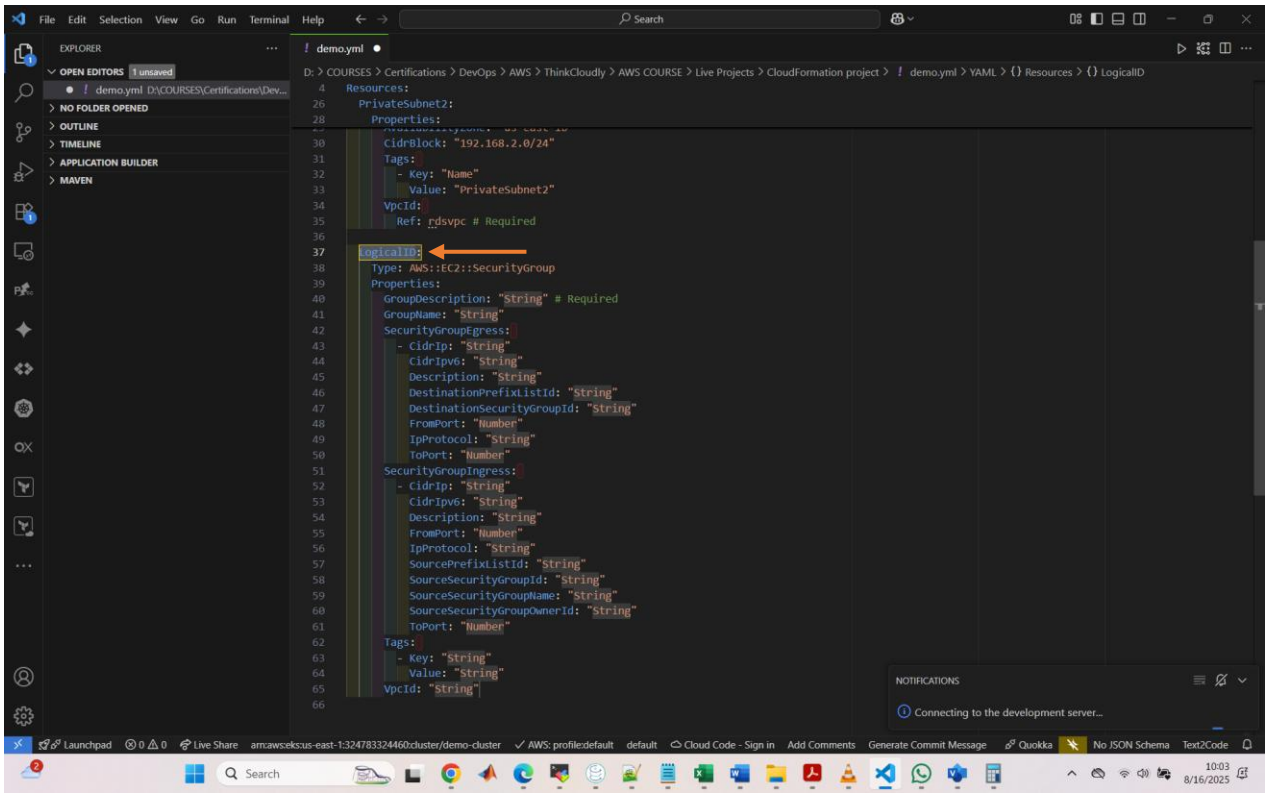
We now have to create the "Security Group", Type "**security**"



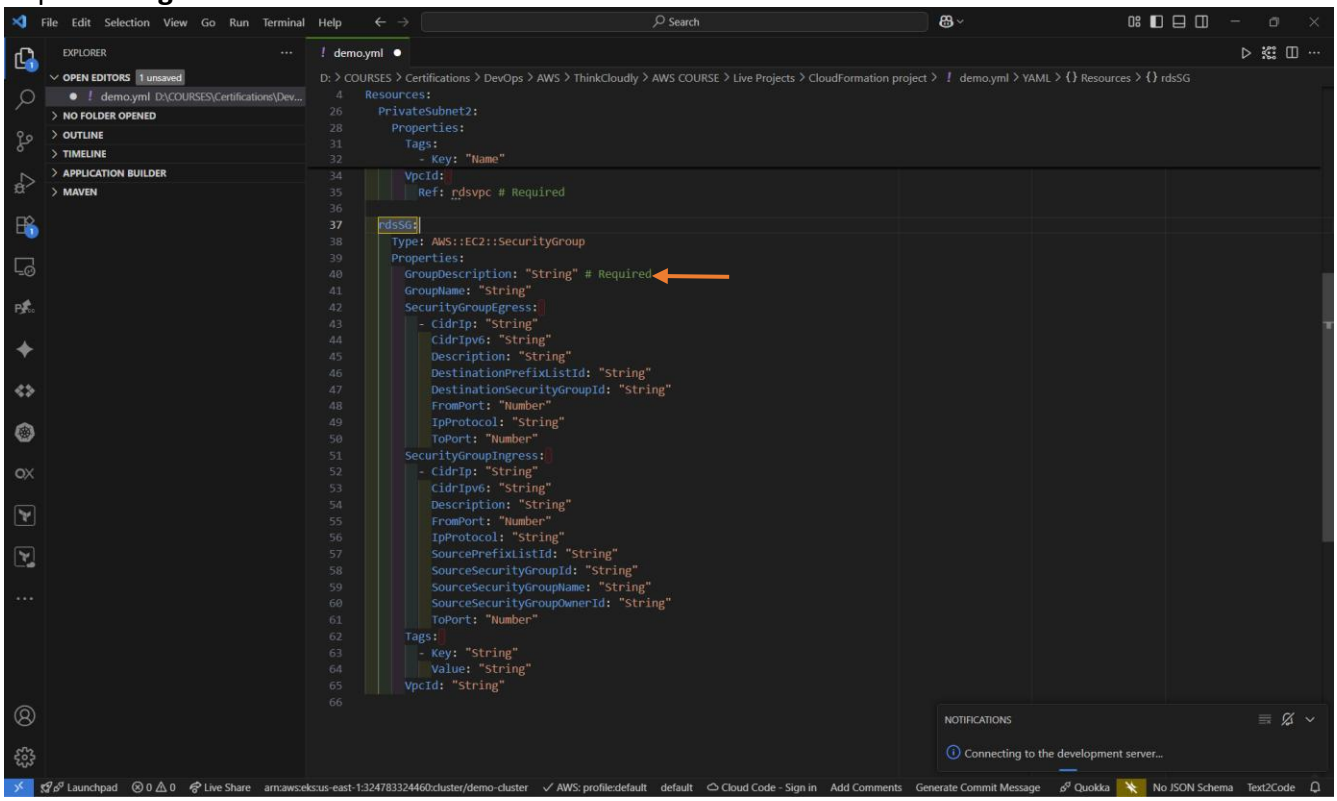
Scroll down



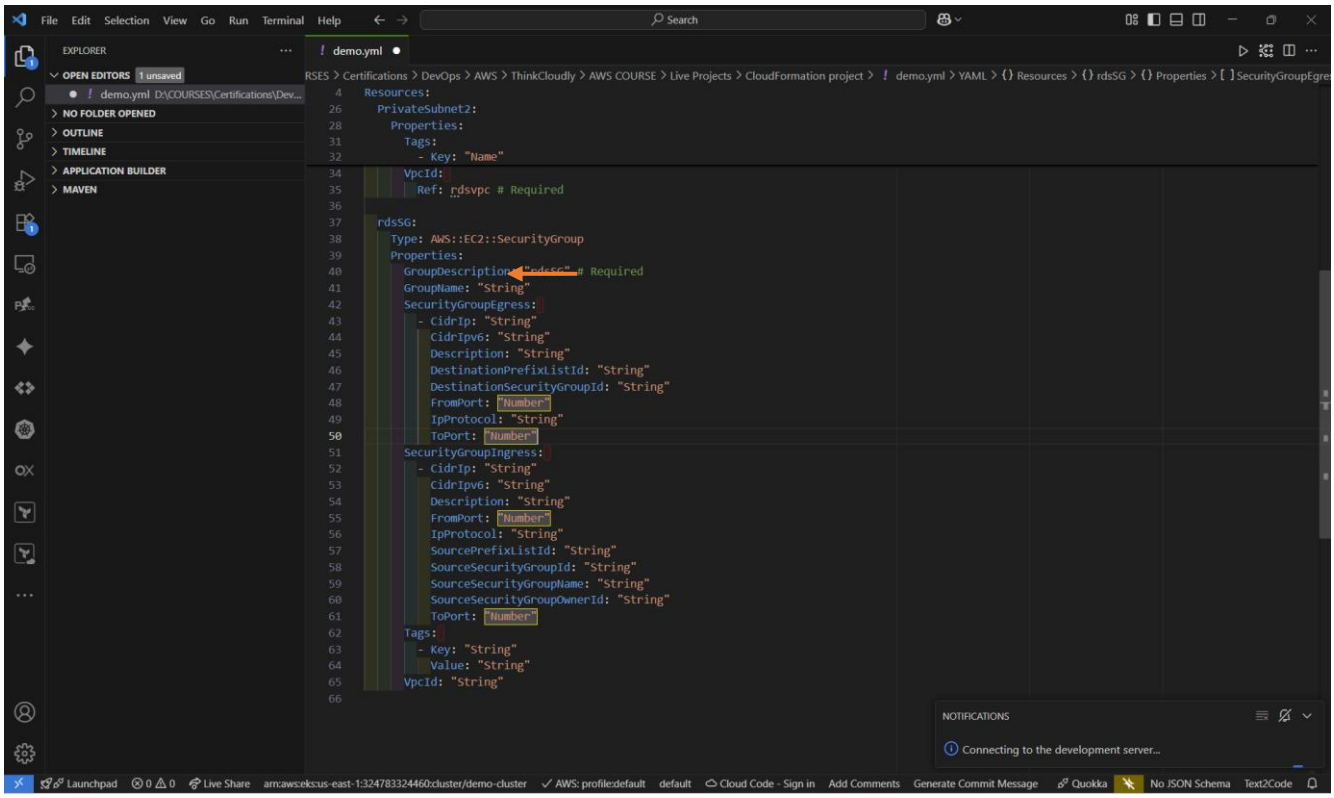
Select "**ec2-securitygroup**"



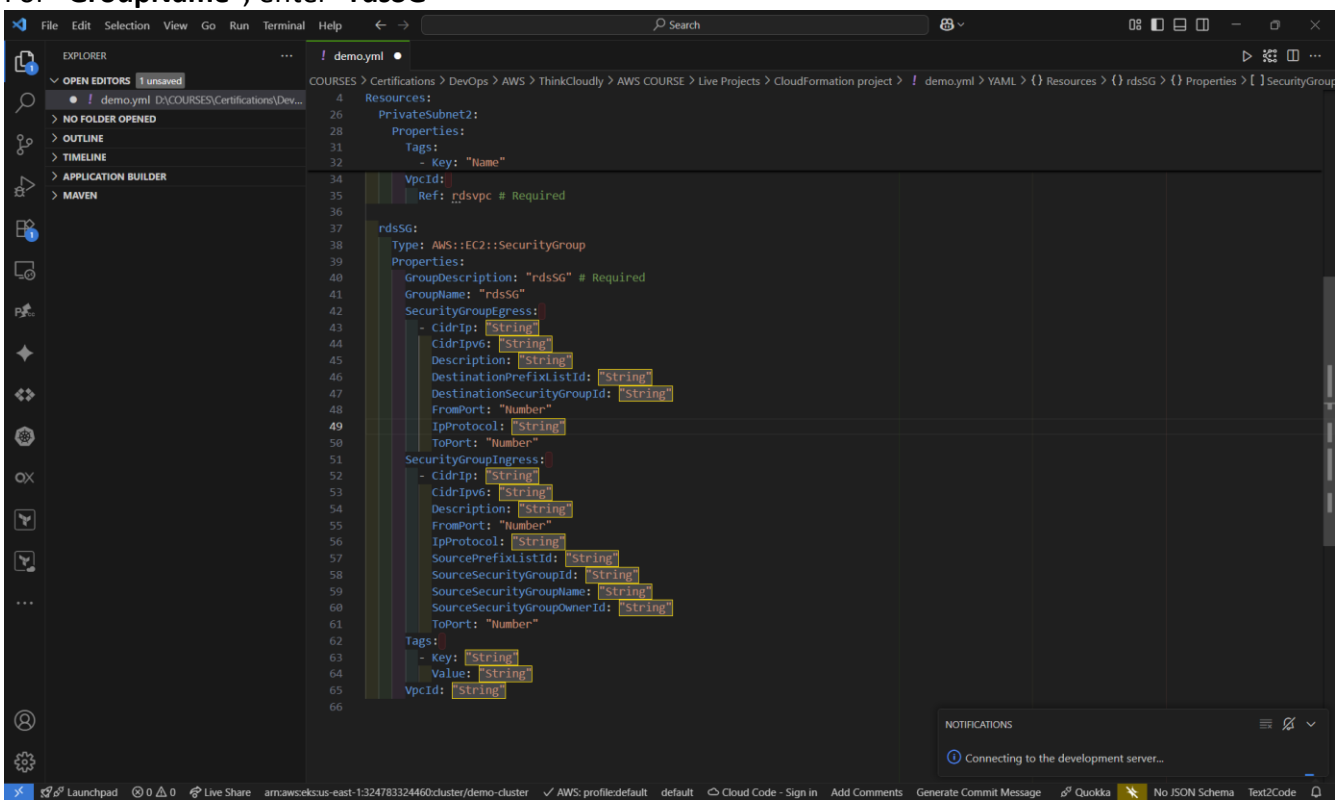
Replace "**LogicalID**" with "**rdsSG**"



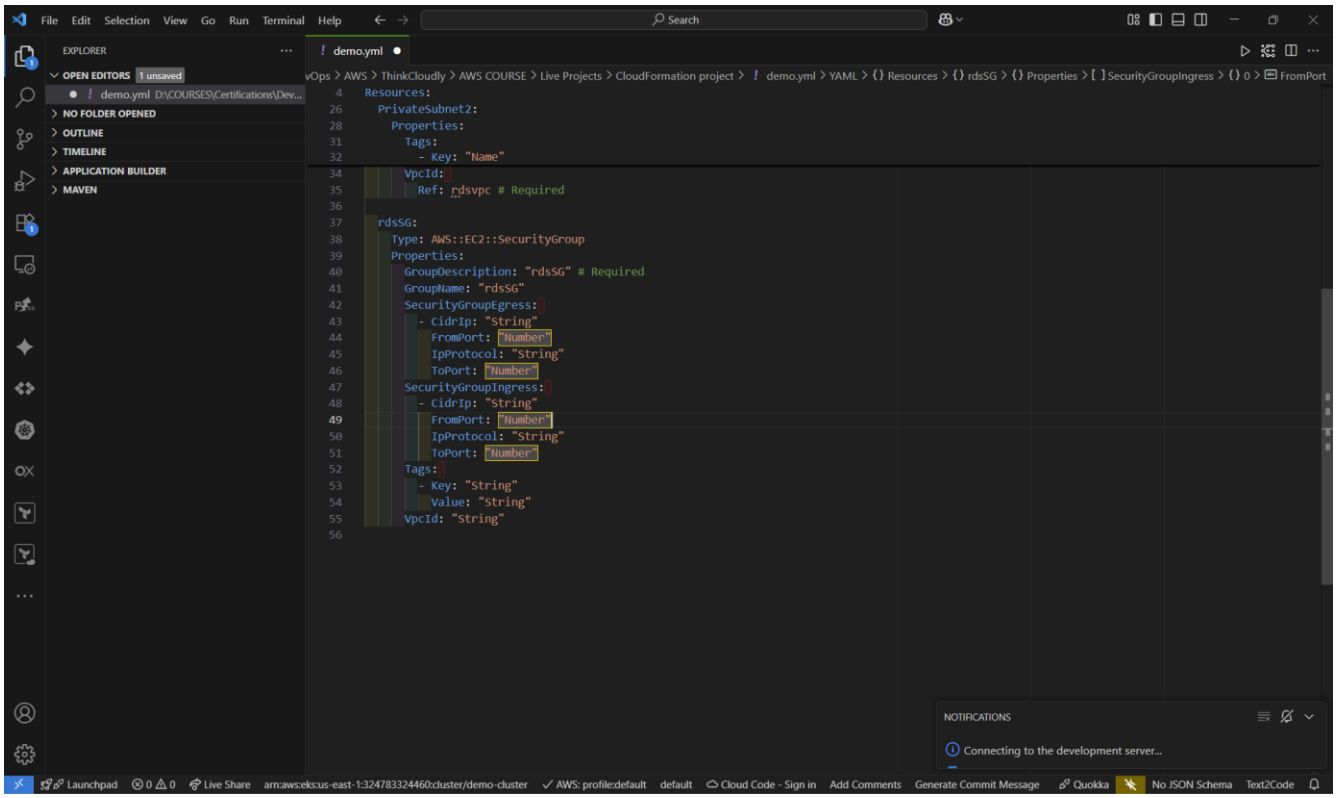
On "**GroupDescription**" enter "**rdsSG**"



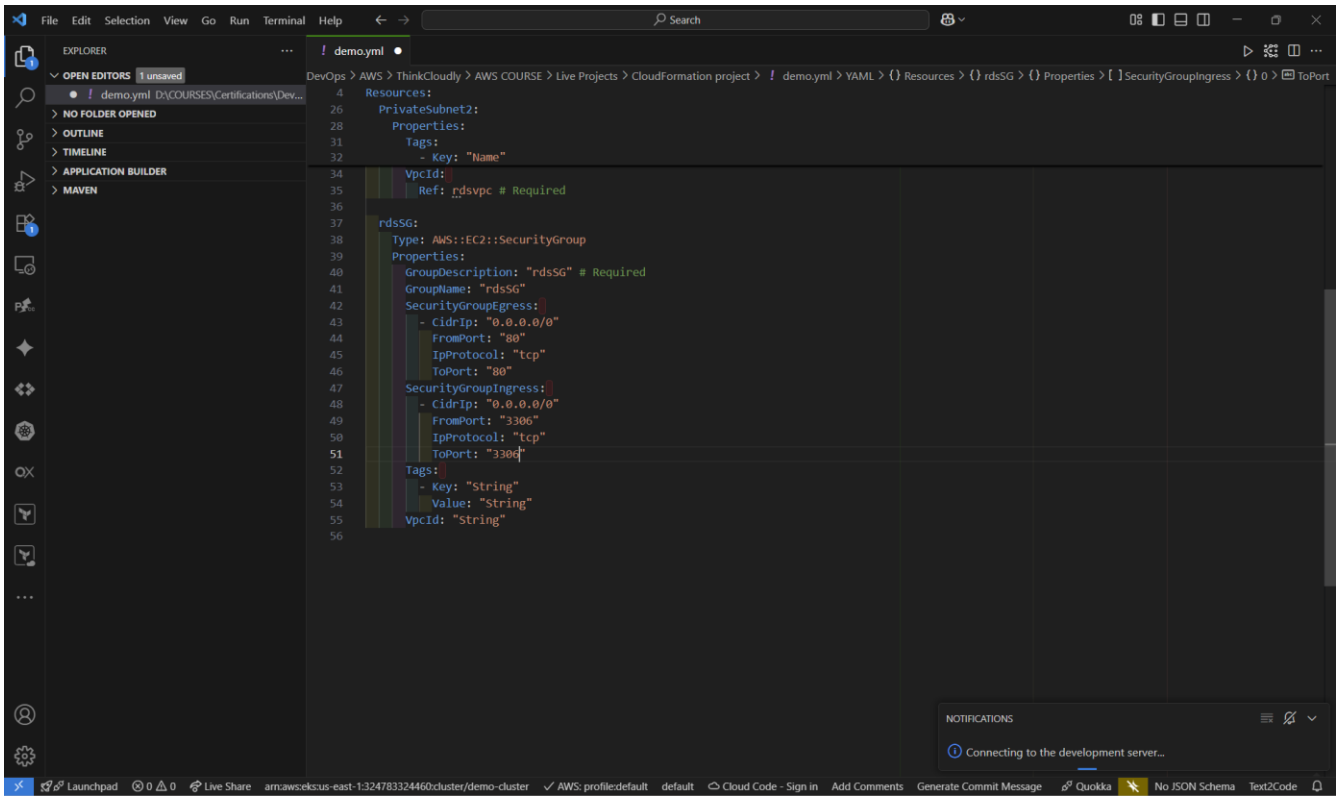
For "**GroupName**", enter "**rdsSG**"



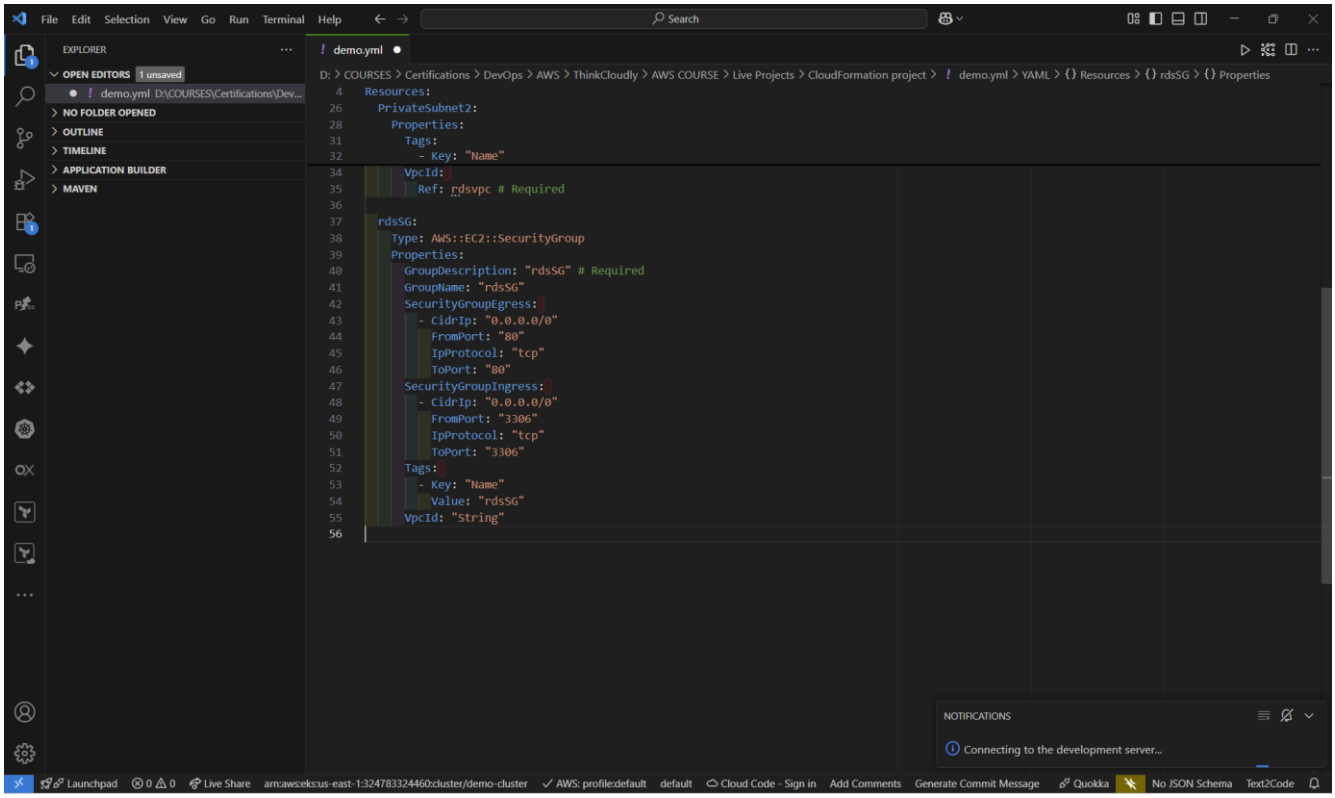
Remove the lines that are not needed



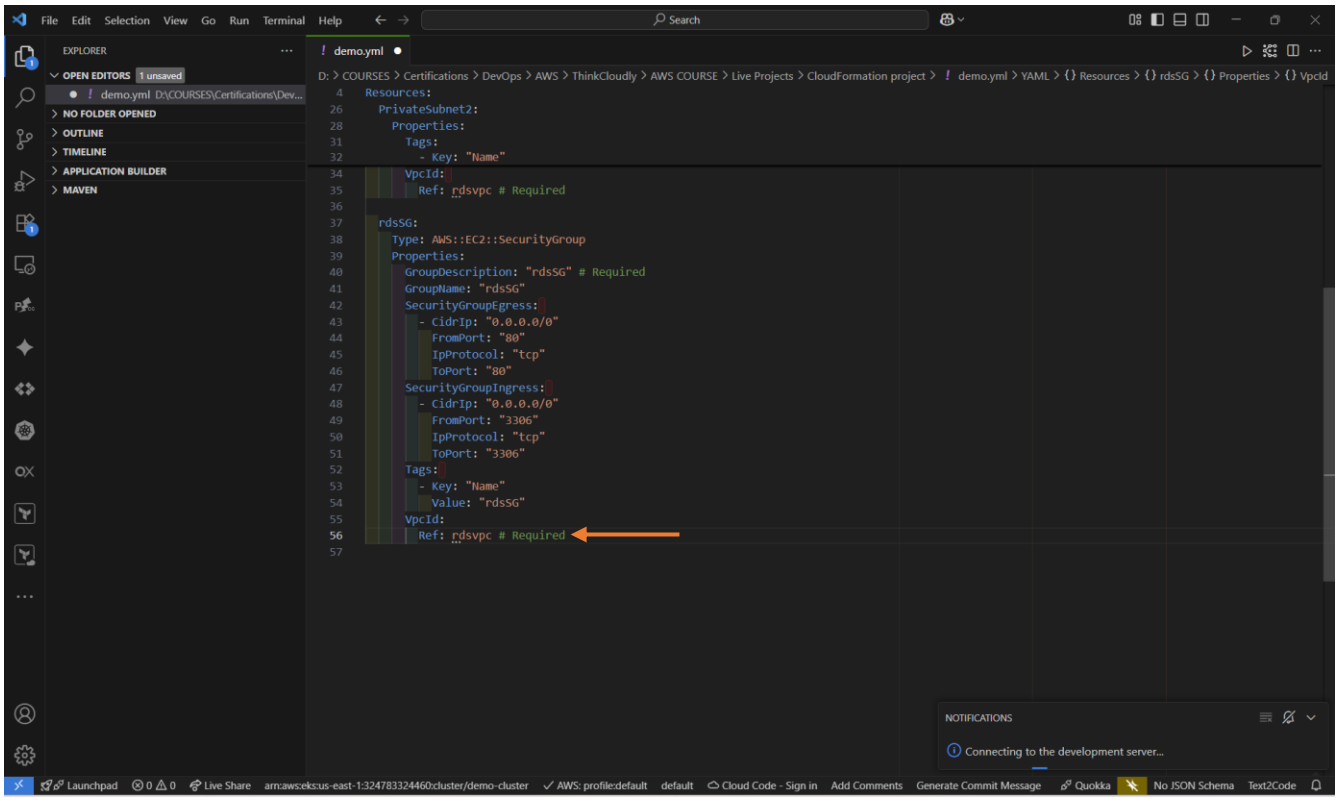
Enter these details:
SG Egress CIDR: 0.0.0.0/0
SG Egress FromPort: 80
SG Egress ToPort: 80
SG Egress IPProtocol: TCP
SG Ingress CIDR: 0.0.0.0/0
SG Ingress FromPort: 3306
SG Ingress ToPort: 3306
SG Ingress IPProtocol: TCP



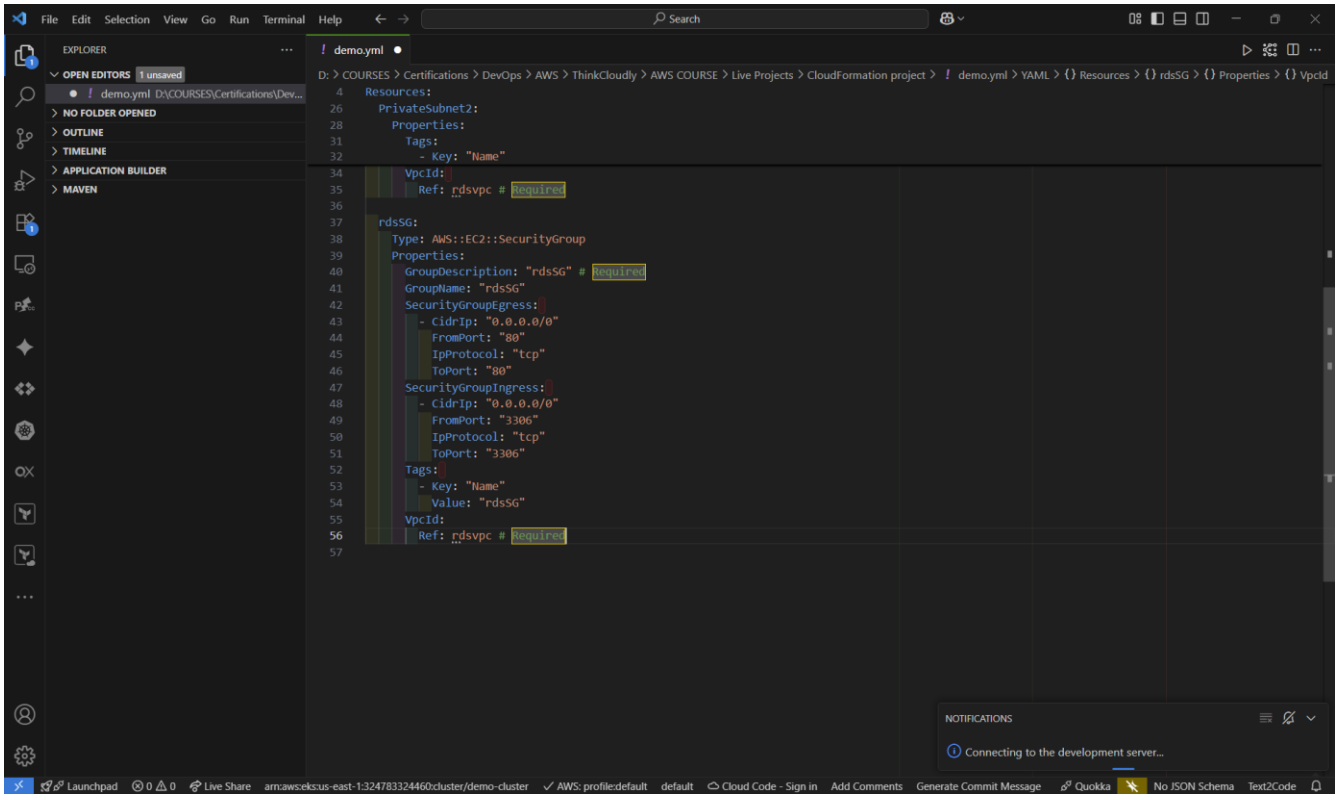
Under “**Tag**”, in “**key**” enter “**Name**” and in “**value**” enter “**rdsSG**”



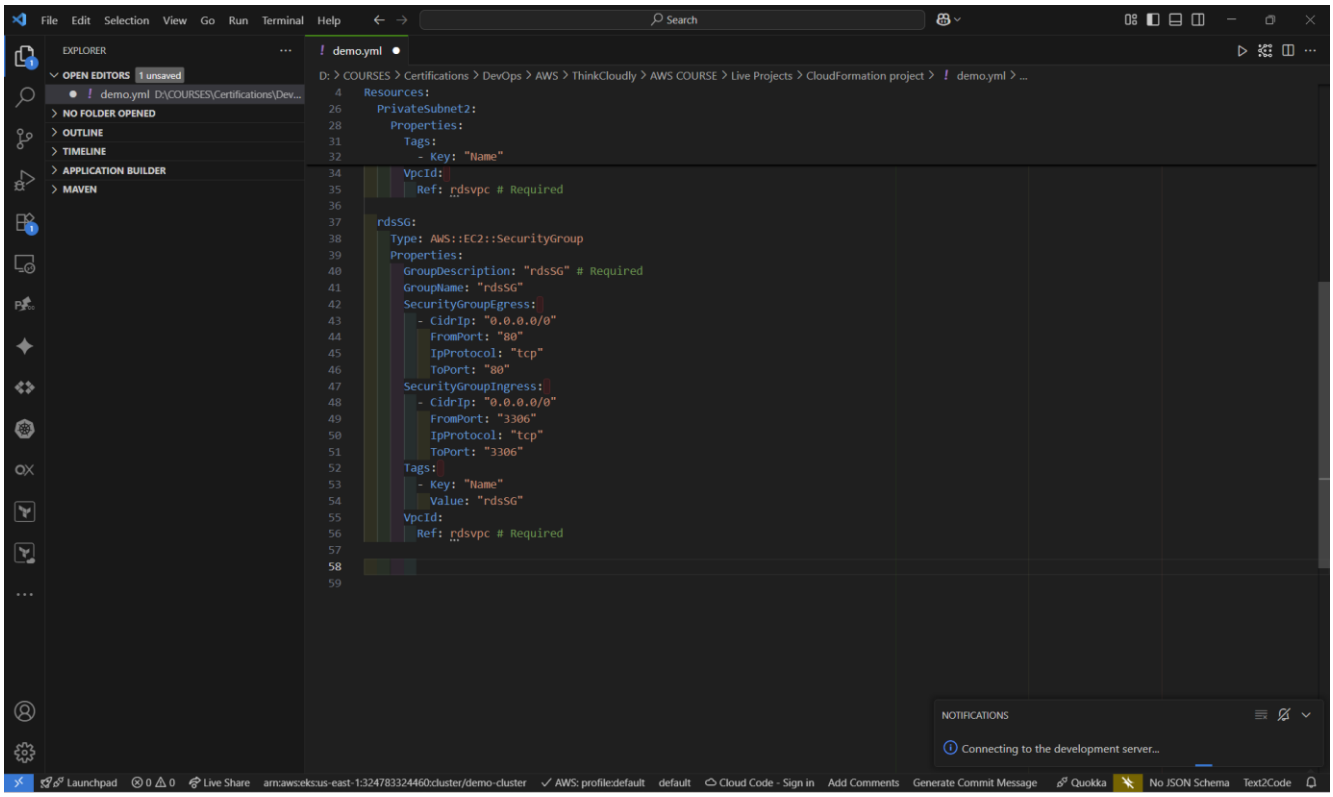
On “**vpcId**” enter “**Ref: rdsvpc**”



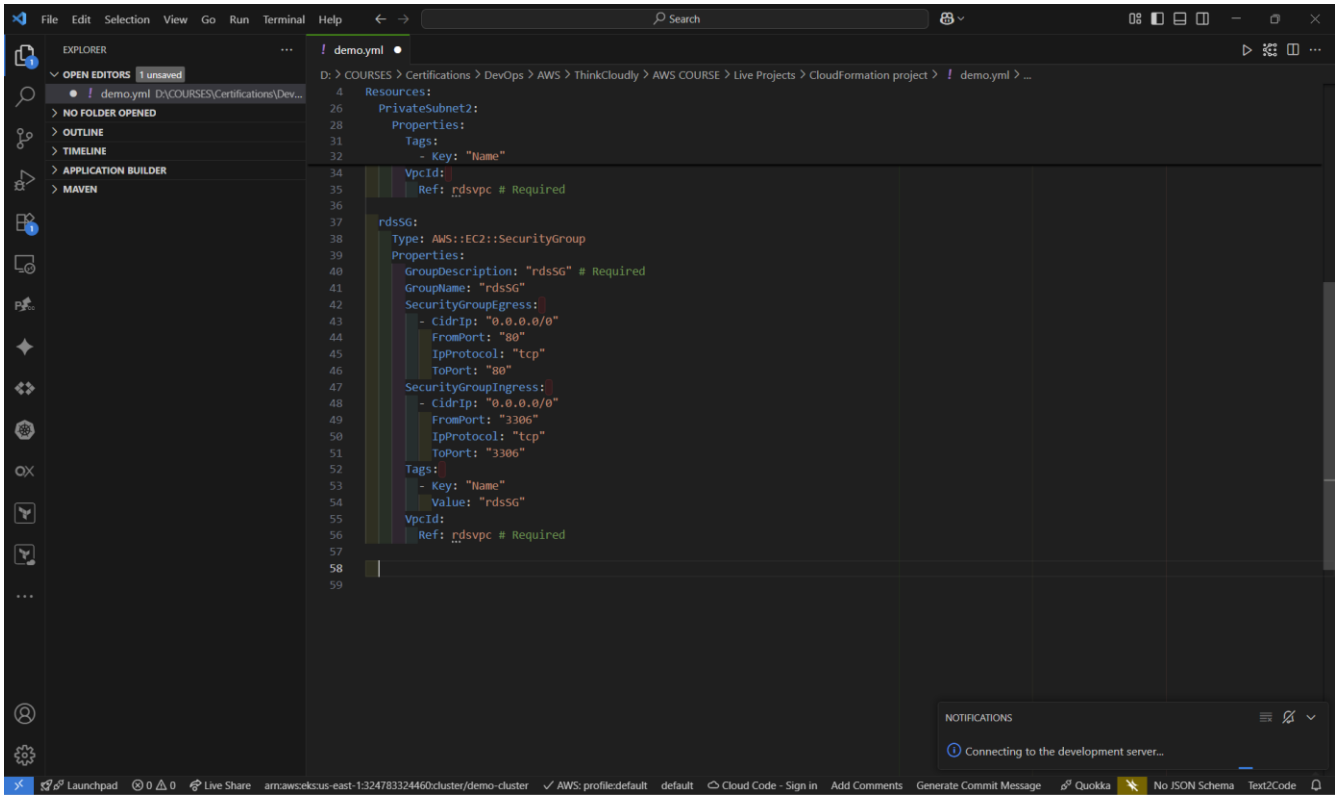
Put your cursor at the end of the "**vpcId**" line



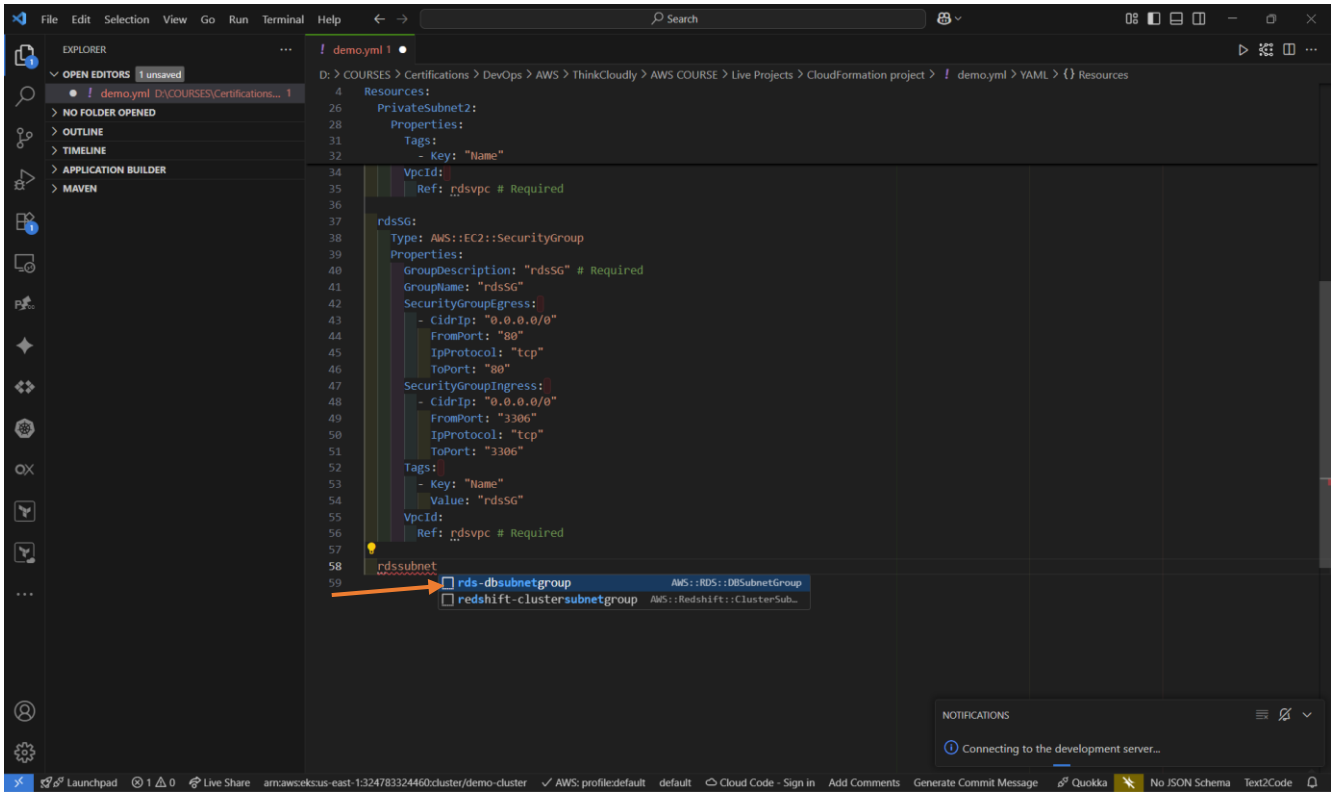
Press Enter twice



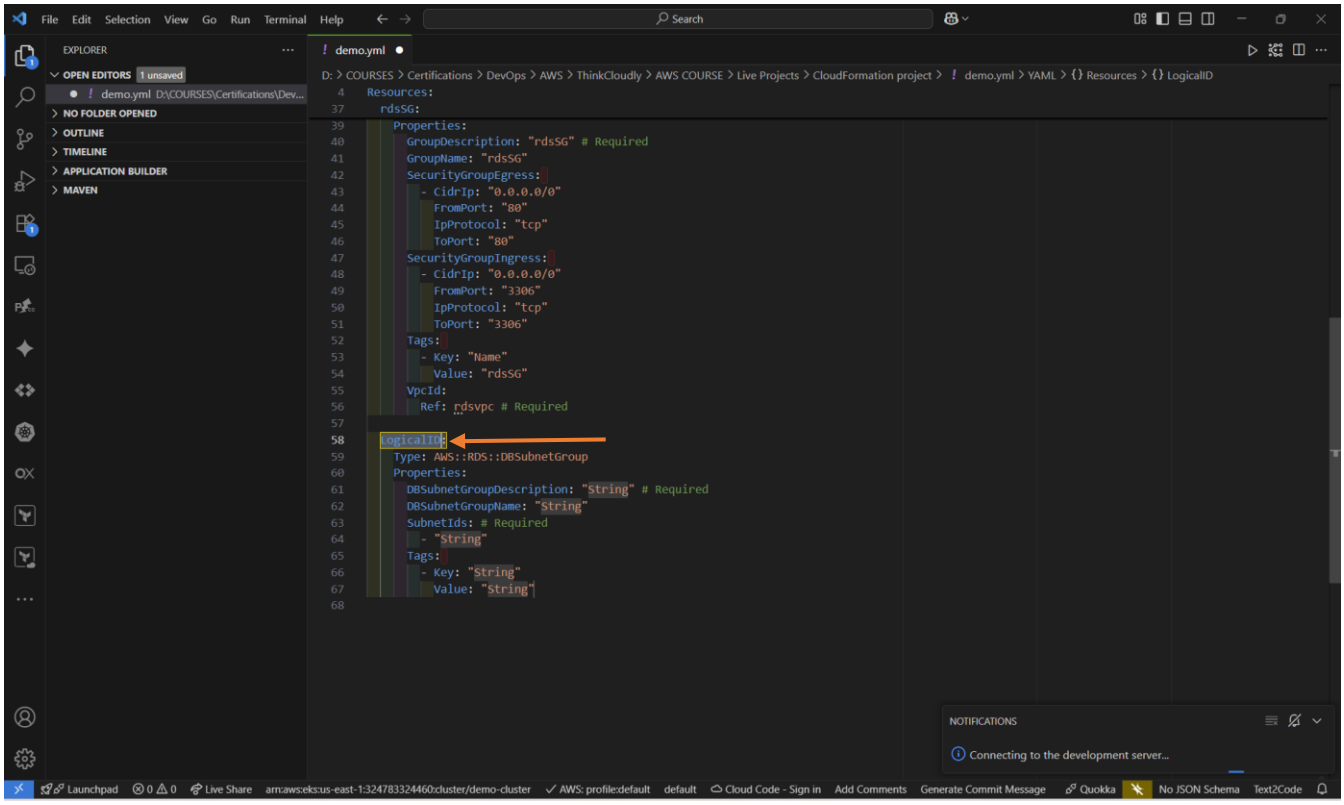
Move the cursor back to the line under "**rdsSG**" using backspace key



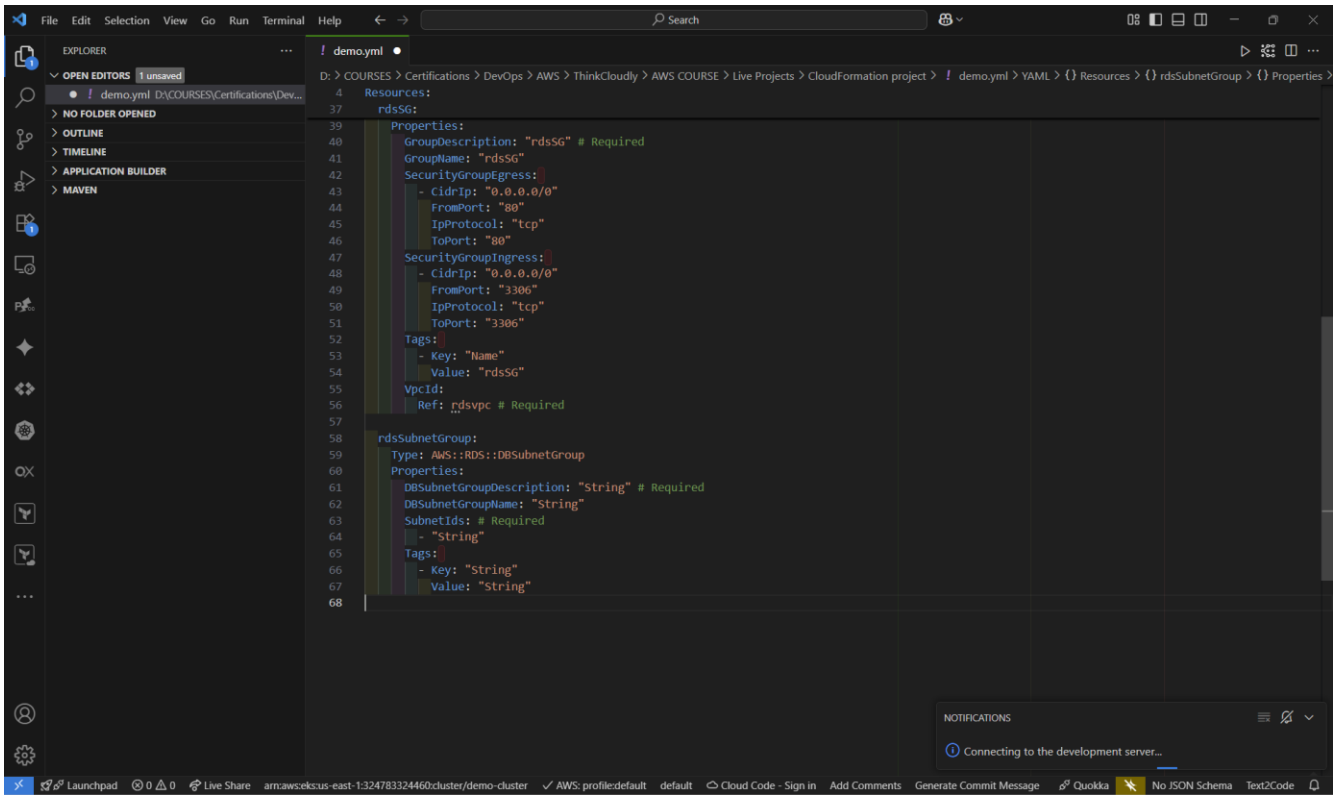
Now, we have to create the subnet group. Type "**rdssubnet**"



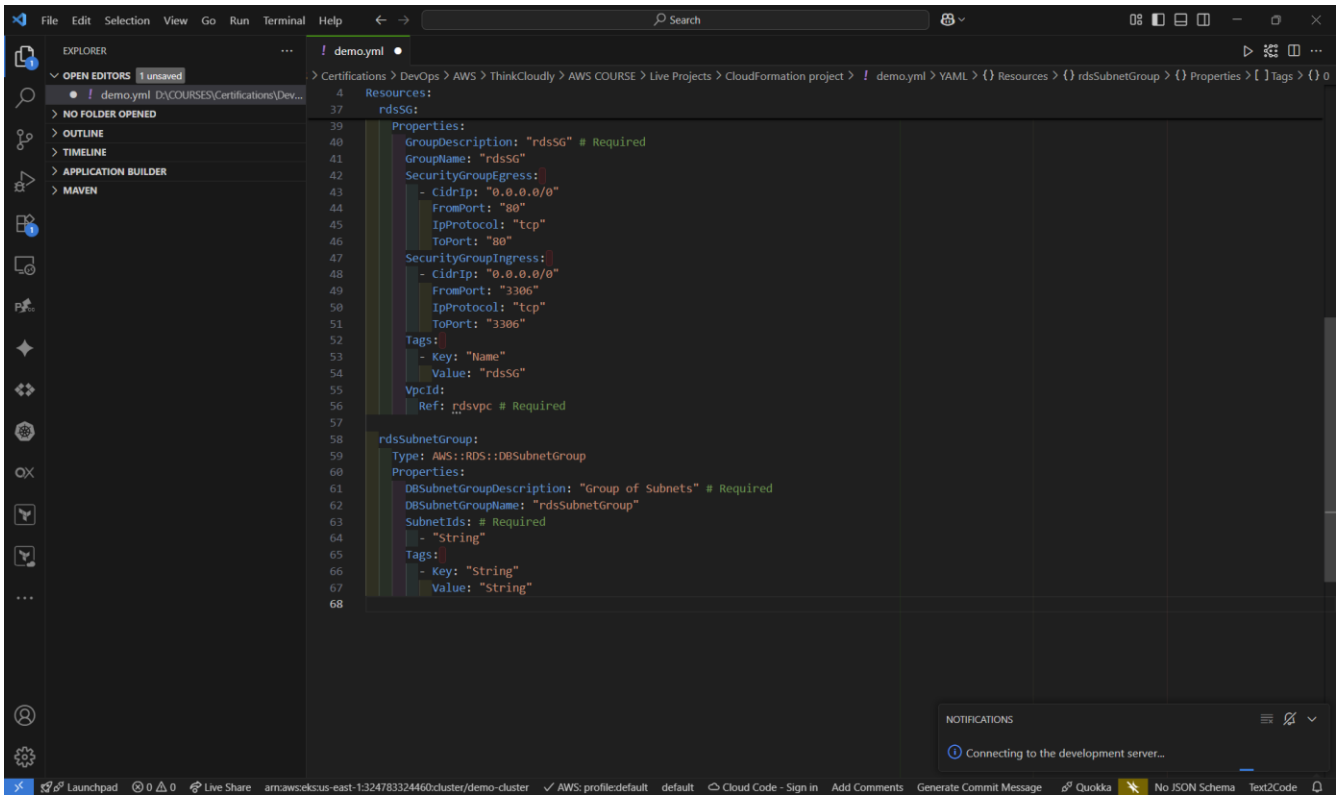
Select "**rds-dbsubnetgroup**"



Replace "**LogicalID**" with "**rdsSubnetGroup**"

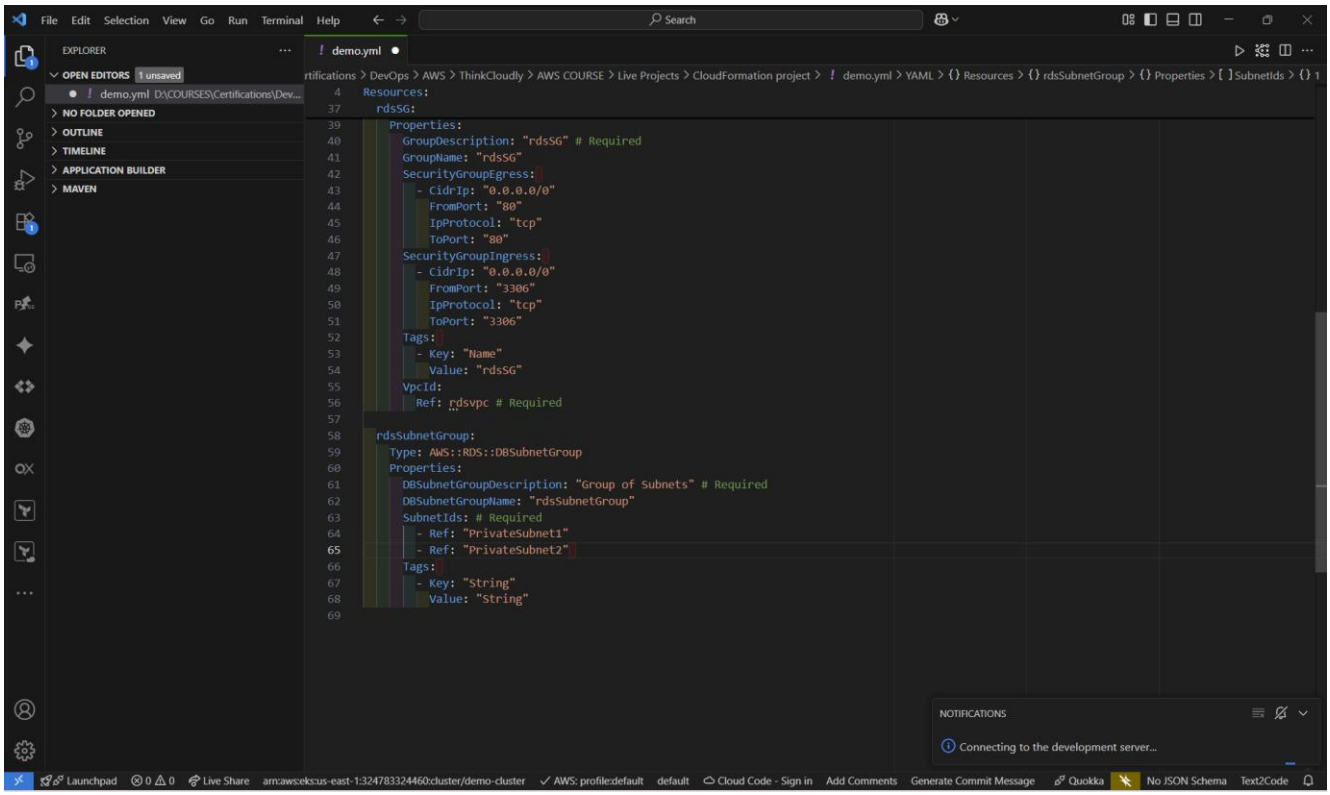


For "**DBSubnetGroupDescription**" enter "**Group of Subnets**" and for "**DBSubnetGroupName**" enter "**rdsSubnetGroup**"

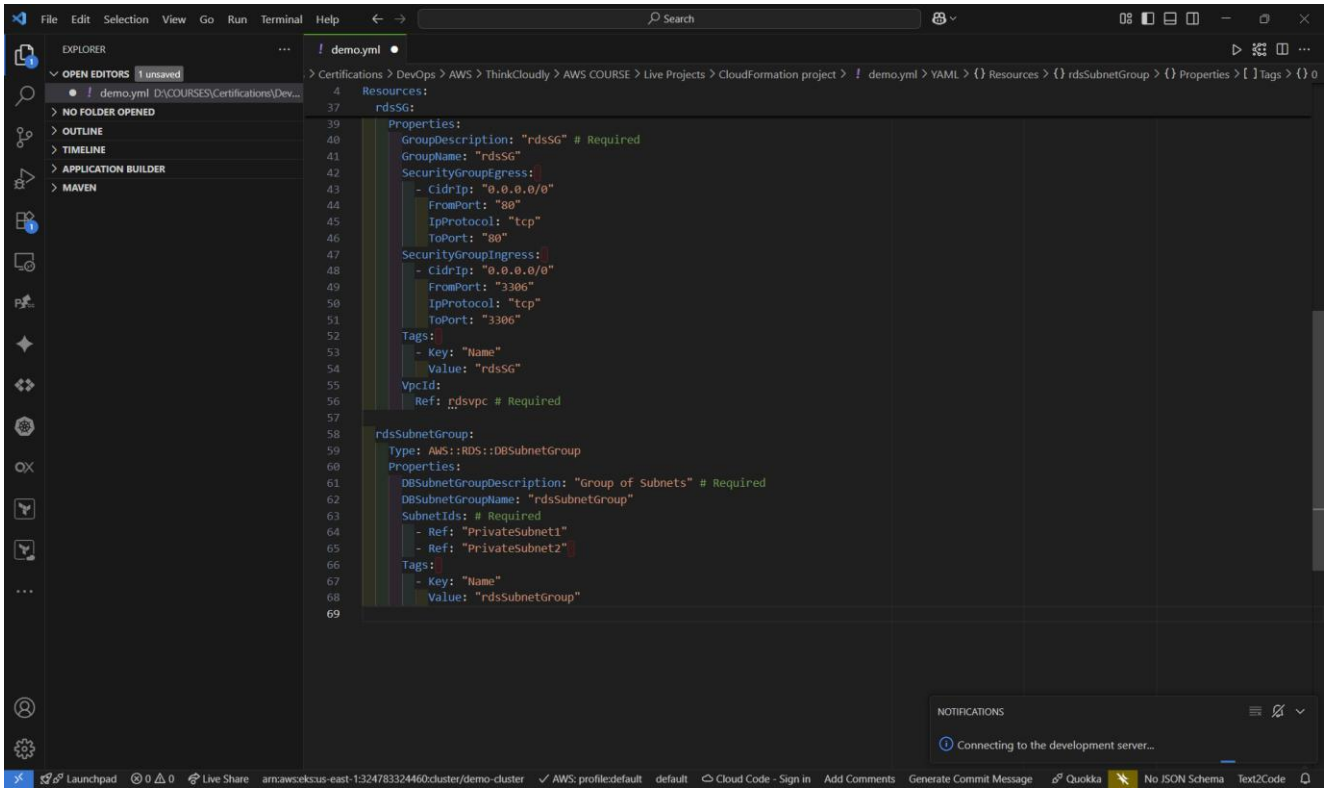


For "subnetIds" enter these details:

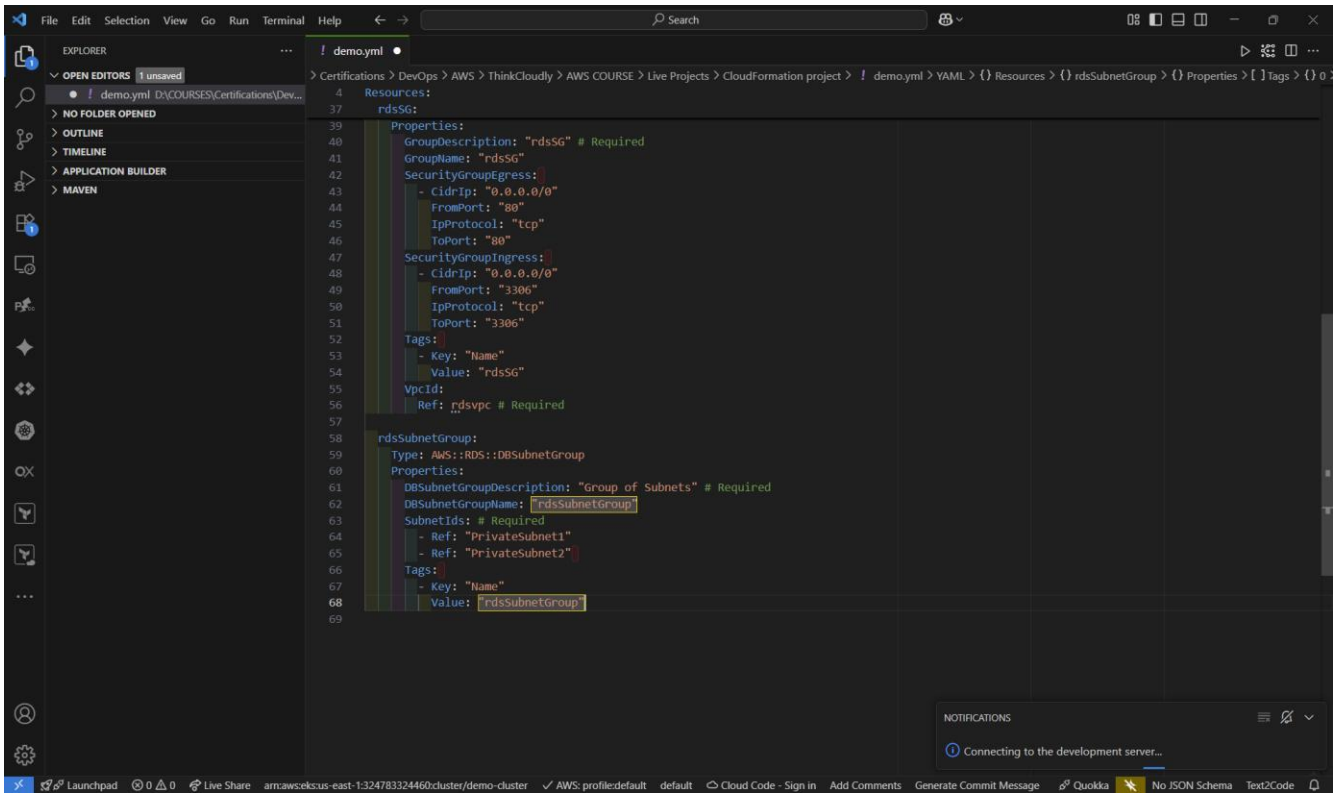
- Ref: "PrivateSubnet1"
- Ref: "PrivateSubnet2"



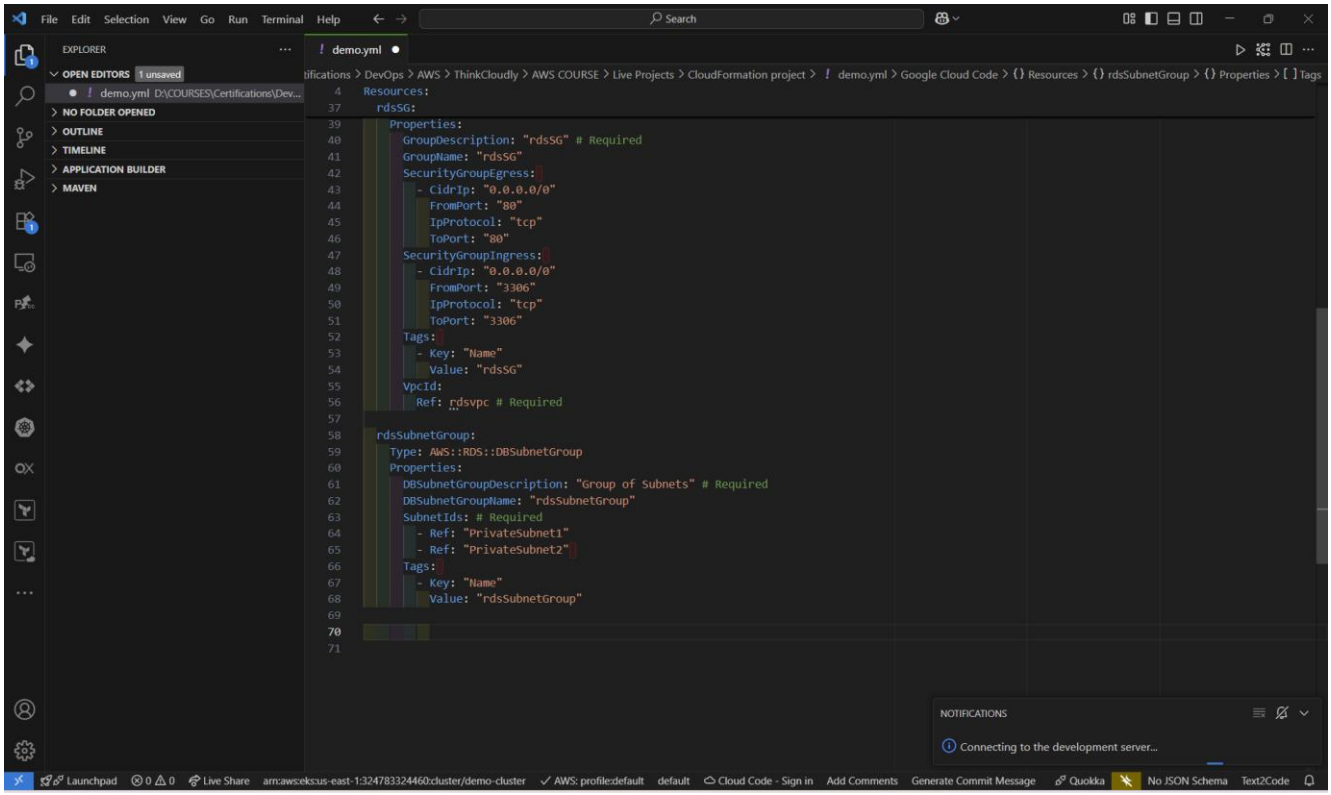
Under "**Tag**", in "**key**" enter "**Name**" and in "**value**" enter "**rdsSubnetGroup**"



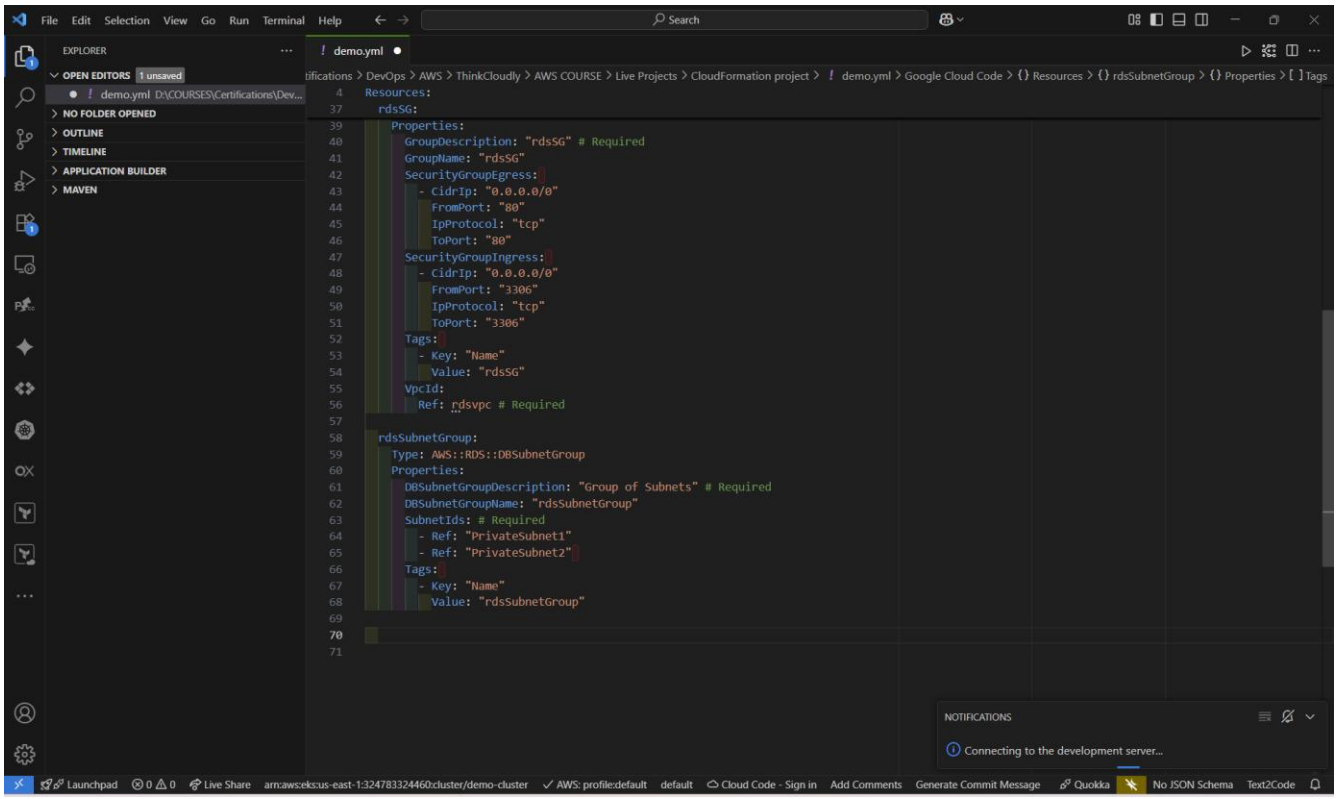
Then the next resource is our database instance. Put your cursor at the end of Value: "**rdsSubnetGroup**" line



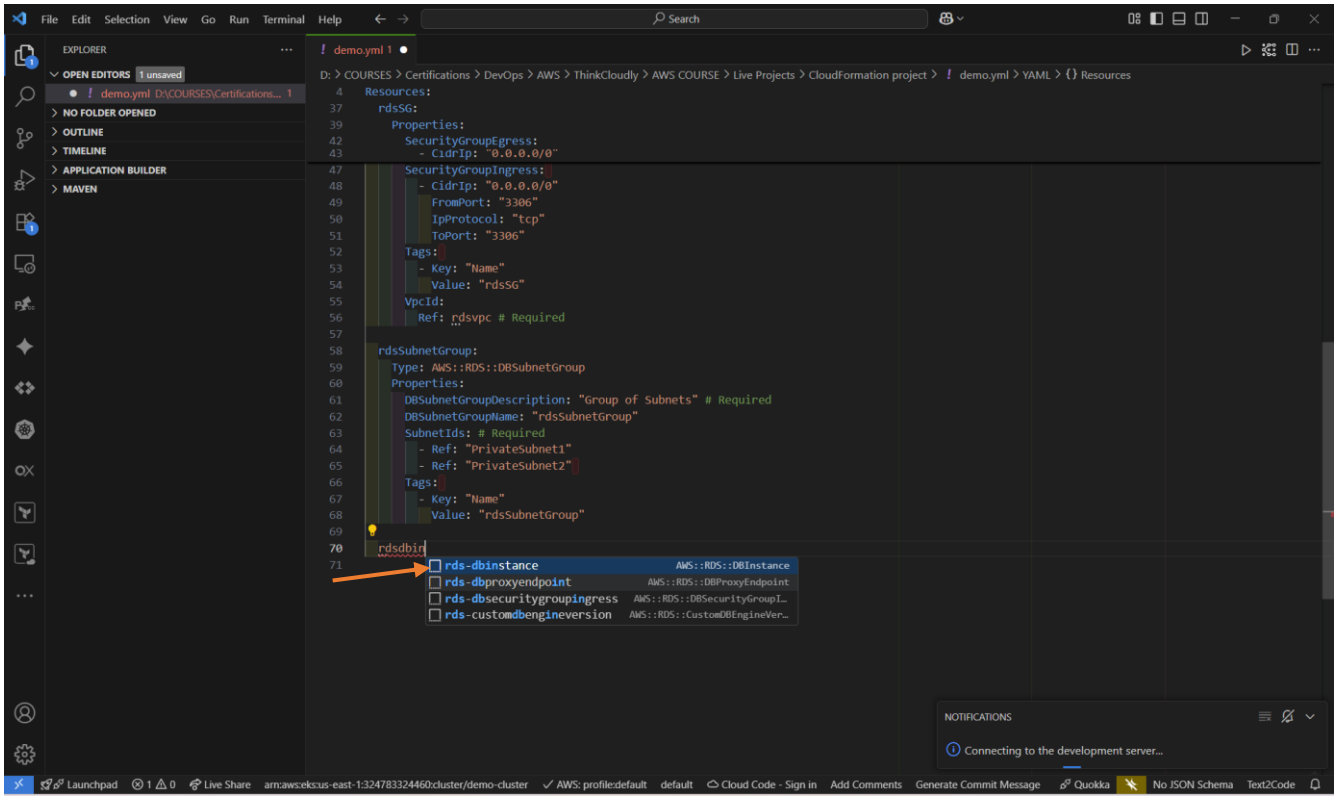
Press Enter twice



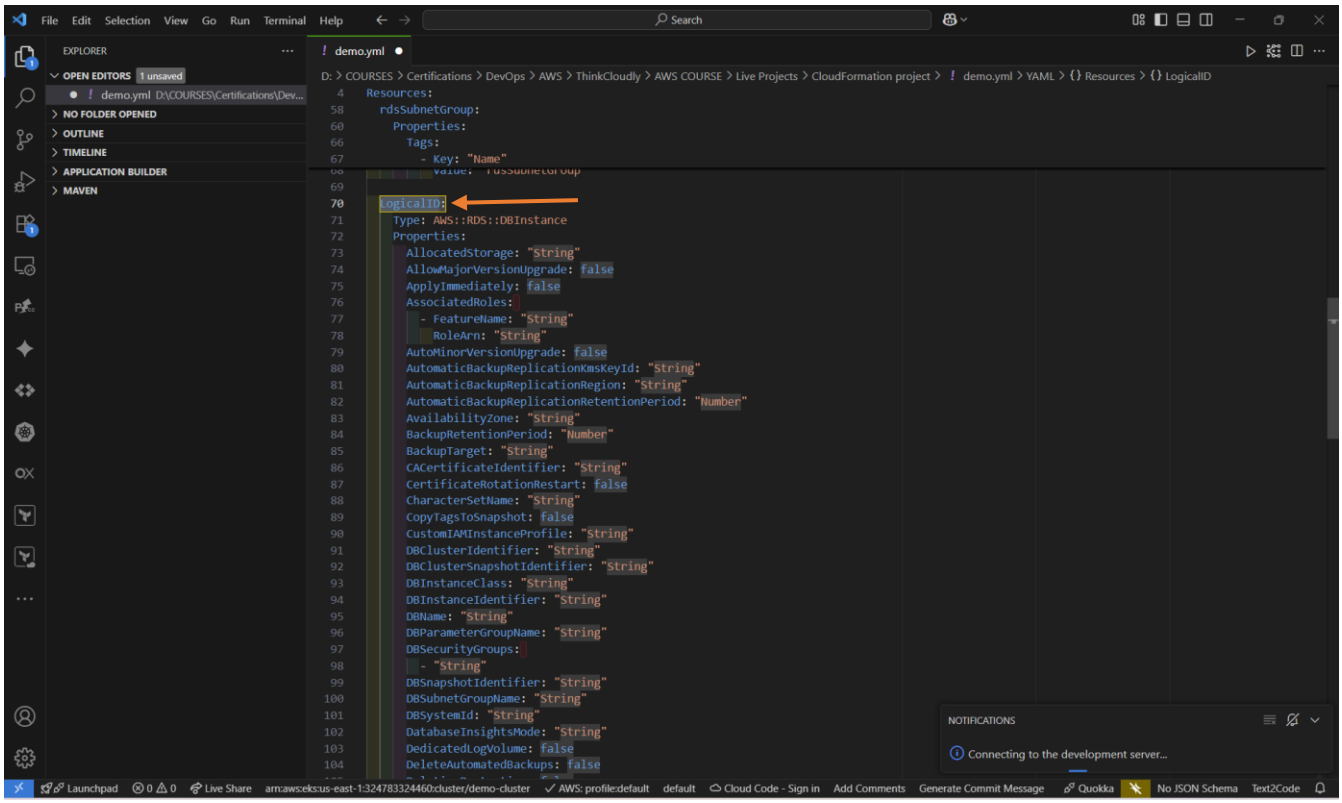
Move the cursor back to the line under "**rdsSubnetGroup**" using backspace key



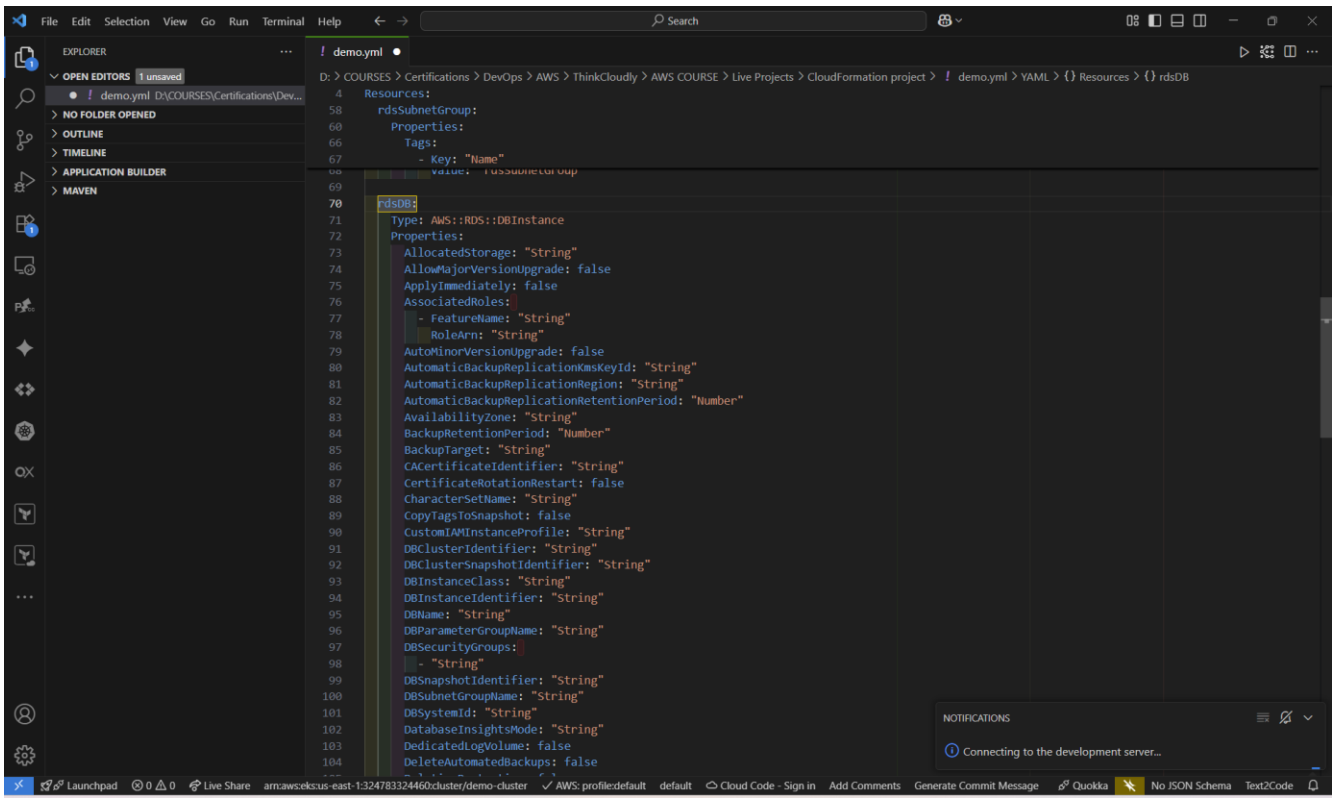
Type "**rdsdbinstance**"



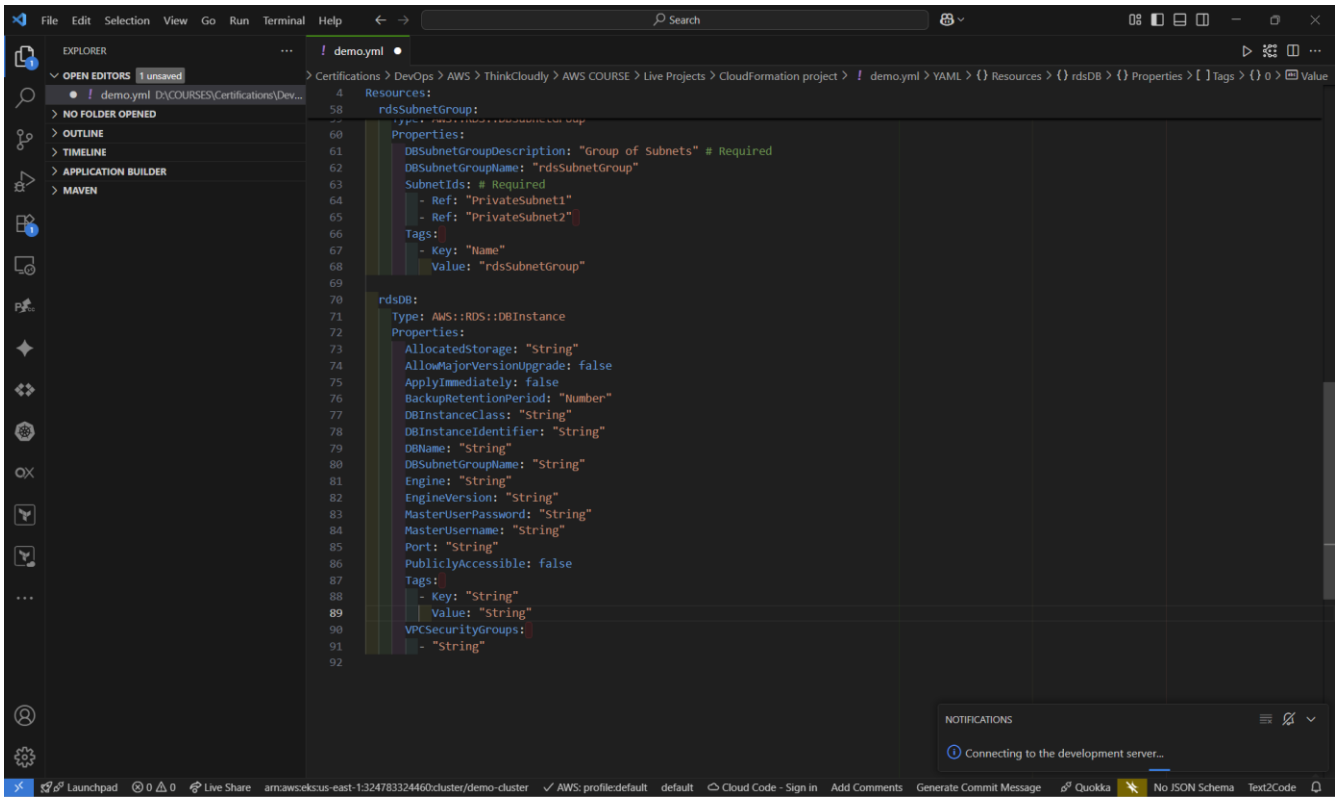
Select "**rds-dbinstance**"



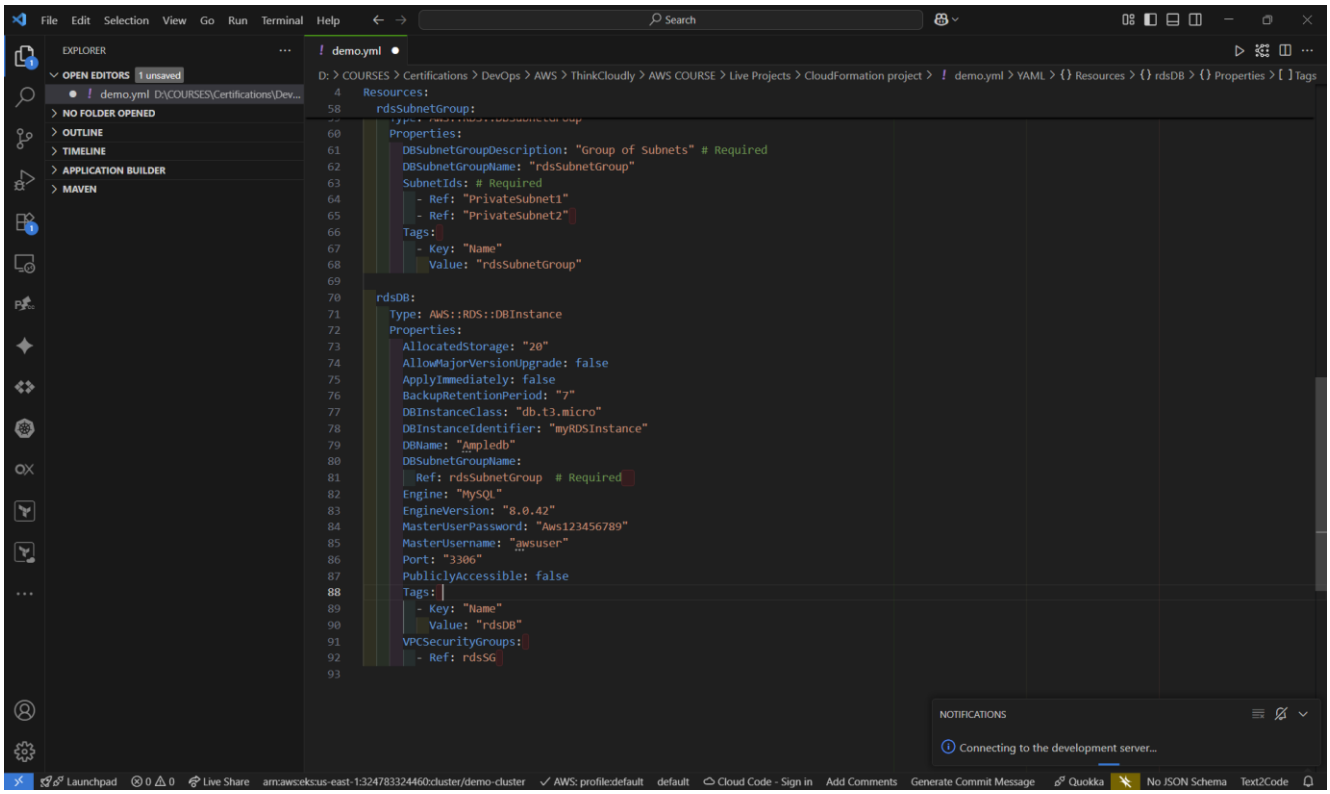
Replace "**LogicalID**" with "**rdsDB**"



Remove the line we do not need



Then enter these details:
Storage Space: 20 GB
Database Instance Class: Db.t2.micro
Database name: Ampledb
Database Engine: MySQL
Engine Version: 8.0.42
PubliclyAccessible: False
Username: awsuser
Password: Aws123456789
Port: 3306
AllowUpgradeVersion: False
BackUp Retention Period: 7 Days



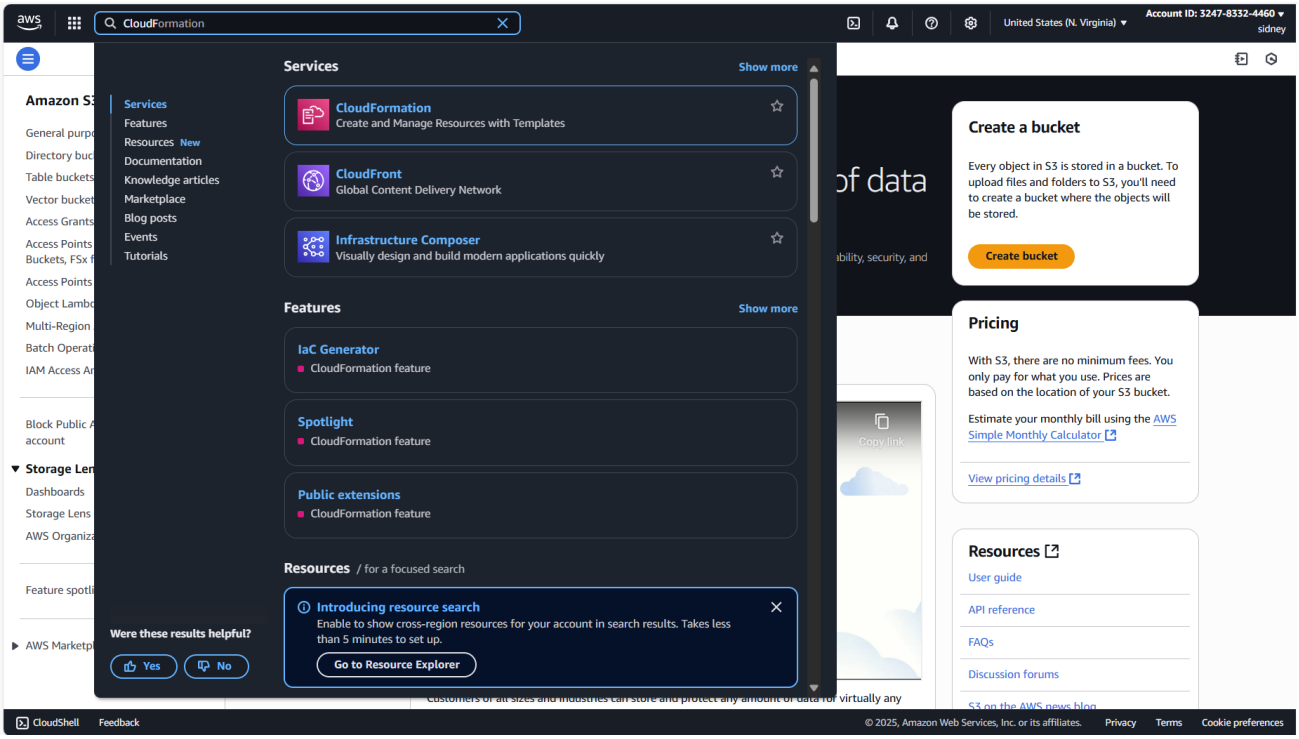
Save the file by clicking on File -> Save

```
AWSTemplateFormatVersion: 2010-09-09
Description: Live project cloud formation

Resources:
  rdsVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: "192.168.0.0/16"
      EnableDnsHostnames: false
      EnableDnsSupport: false
      Tags:
        - Key: "Name"
          Value: "rds_VPC"

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: "us-east-1a"
      CidrBlock: "192.168.1.0/24"
      Tags:
        - Key: "Name"
          Value: "PrivateSubnet1"
      VpcId:
        Ref: rdsVPC #The Ref value (rdsVPC) should match with the name of the VPC resource (rdsVPC)

  PrivateSubnet2:
```

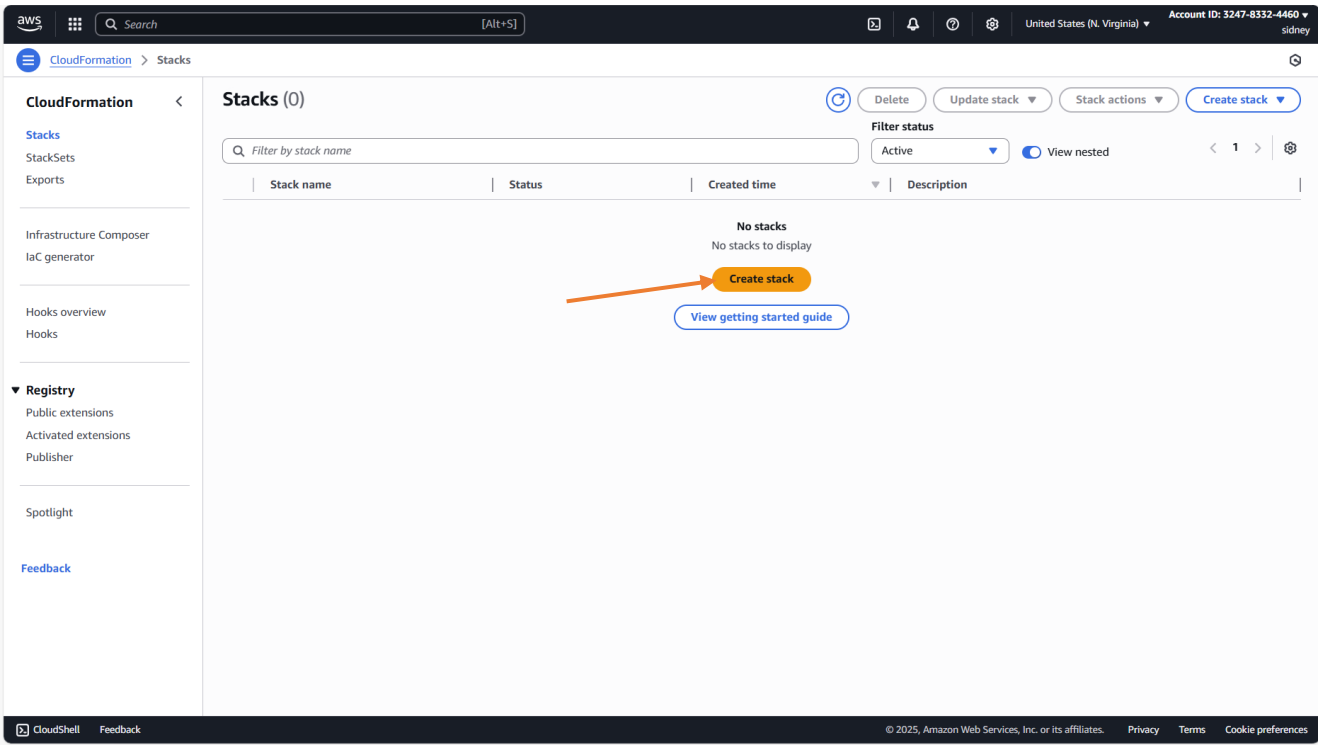
```
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: "us-east-1b"
      CidrBlock: "192.168.2.0/24"
      Tags:
        - Key: "Name"
          Value: "PrivateSubnet2"
      VpcId:
        Ref: rdsVPC # Required

  rdsSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: "rdsSG" # Required
      GroupName: "rdsSG"
      SecurityGroupEgress:
        - CidrIp: "0.0.0.0/0"
          FromPort: "80"
          IpProtocol: "tcp"
          ToPort: "80"
      SecurityGroupIngress:
        - CidrIp: "0.0.0.0/0"
          FromPort: "3306"
          IpProtocol: "tcp"
          ToPort: "3306"
      Tags:
        - Key: "Name"
          Value: "rdsSG"
      VpcId:
        Ref: rdsVPC # Required

  rdsSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: "Group of Subnets" # Required
      DBSubnetGroupName: "rdsSubnetGroup"
      SubnetIds: # Required
        - Ref: "PrivateSubnet1"
        - Ref: "PrivateSubnet2"
      Tags:
        - Key: "Name"
          Value: "rdsSubnetGroup"

  rdsDB:
    Type: AWS::RDS::DBInstance
    Properties:
      AllocatedStorage: "20"
      AllowMajorVersionUpgrade: false
      ApplyImmediately: false
      BackupRetentionPeriod: "7"
```

```
DBInstanceClass: "db.t3.micro"
DBInstanceIdentifier: "myRDSInstance"
DBName: "Ampledb"
DBSubnetGroupName:
  Ref: rdsSubnetGroup  # Required
Engine: "MySQL"
EngineVersion: "8.0.42"
MasterUserPassword: "Aws123456789"
MasterUsername: "awsuser"
Port: "3306"
PubliclyAccessible: false
Tags:
  - Key: "Name"
    Value: "rdsDB"
VPCSecurityGroups:
  - Ref: rdsSG
```

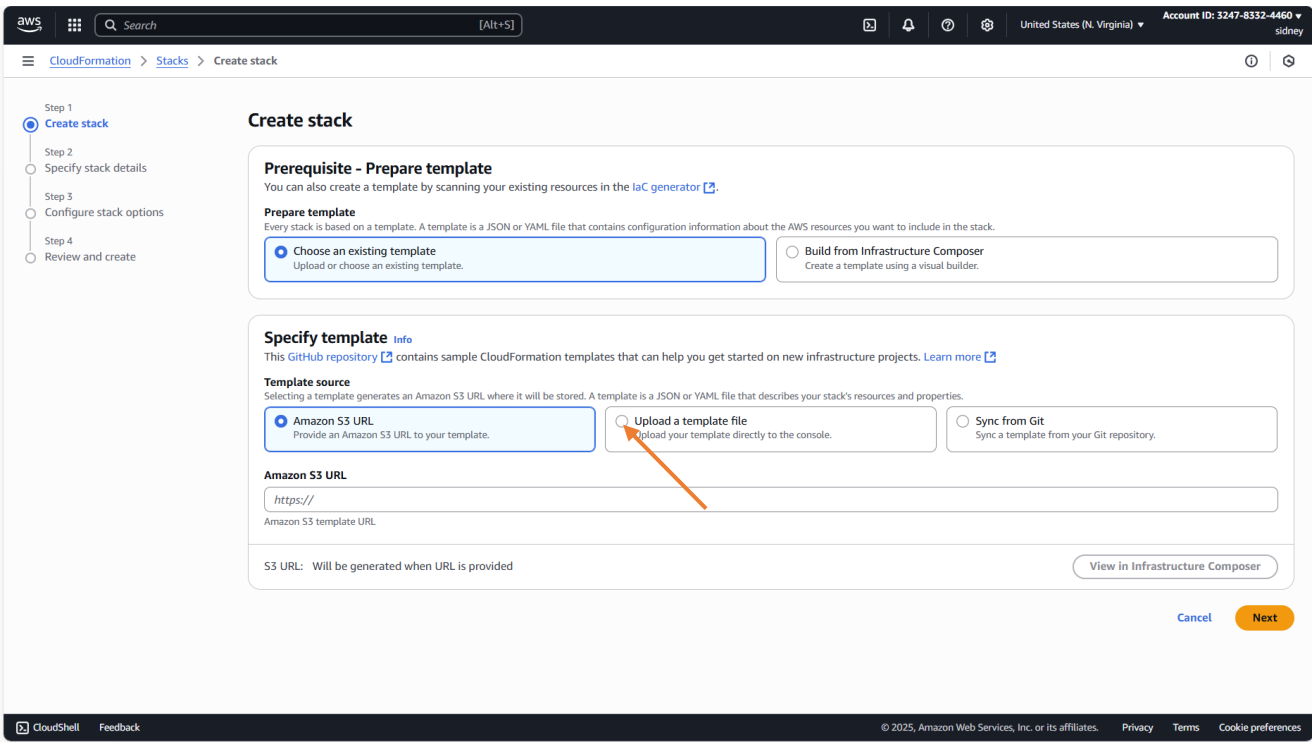
Then go to AWS Management console and search for "**CloudFormation**"



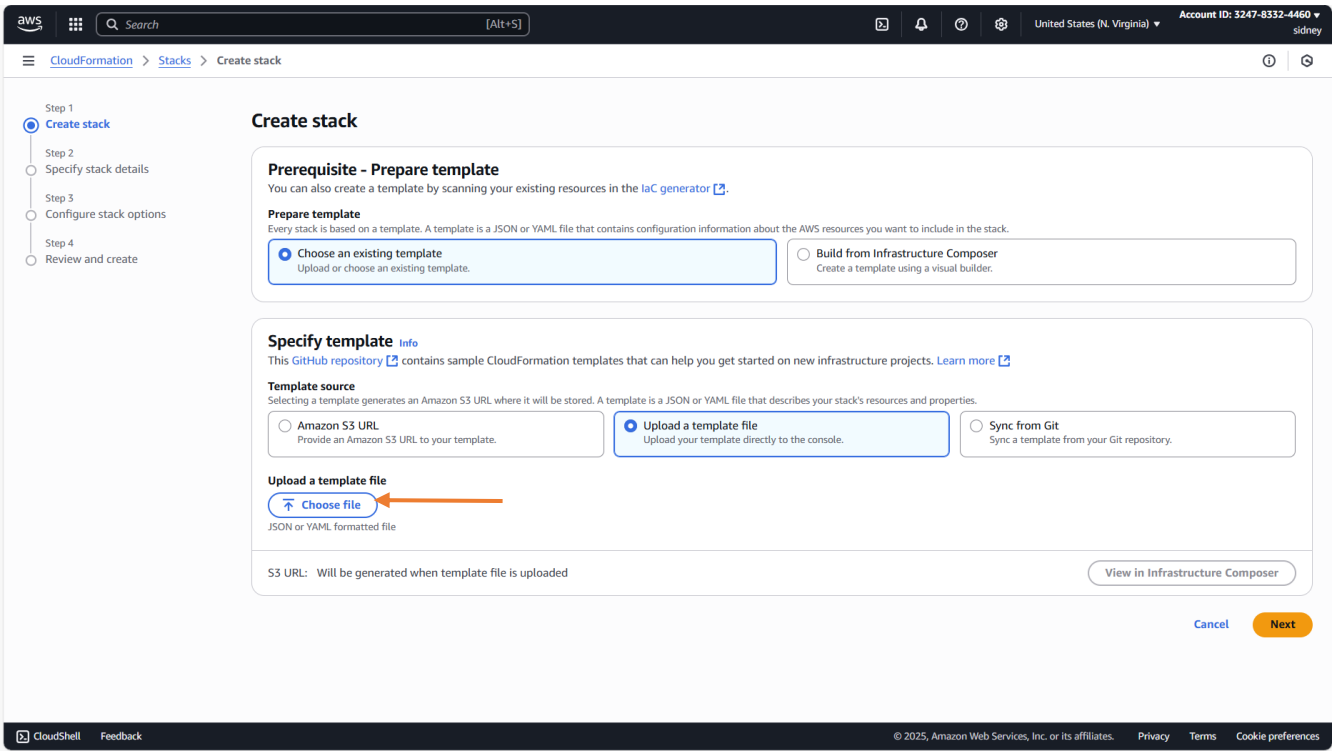
Click on "**CloudFormation**" under services



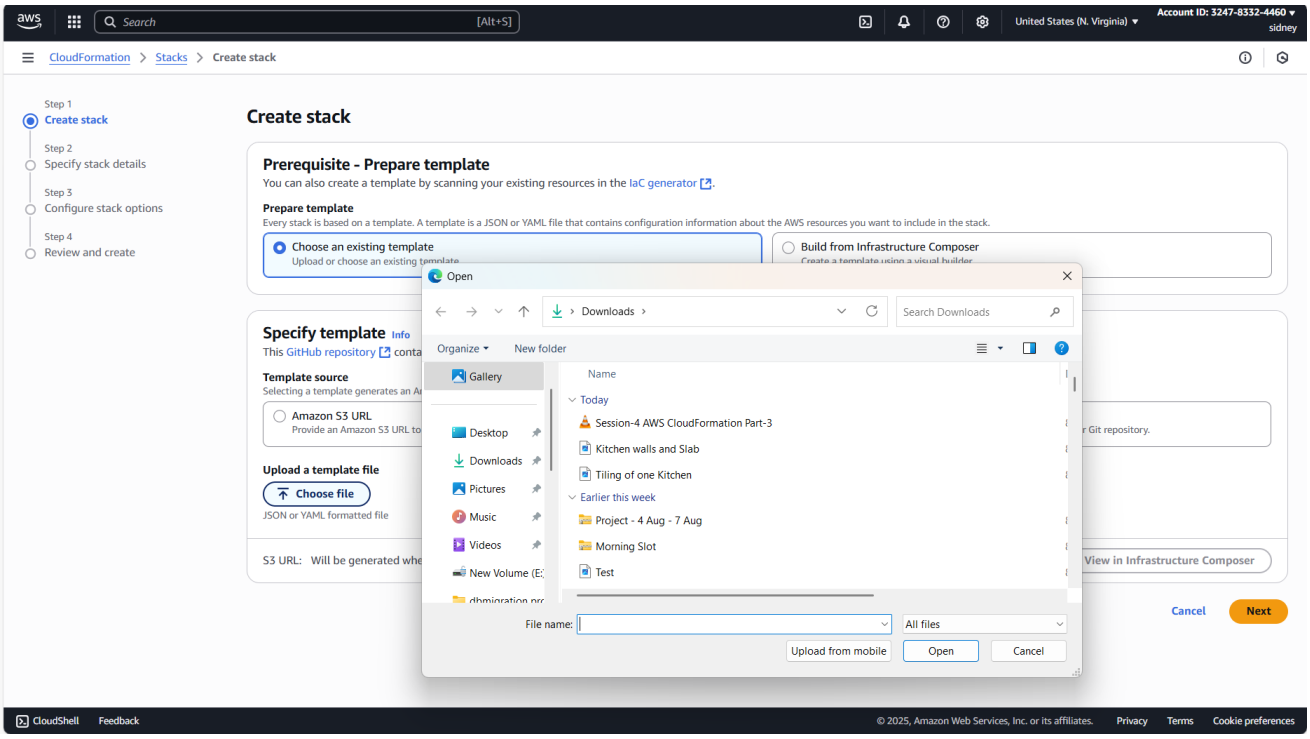
Click on "**Create Stack**"



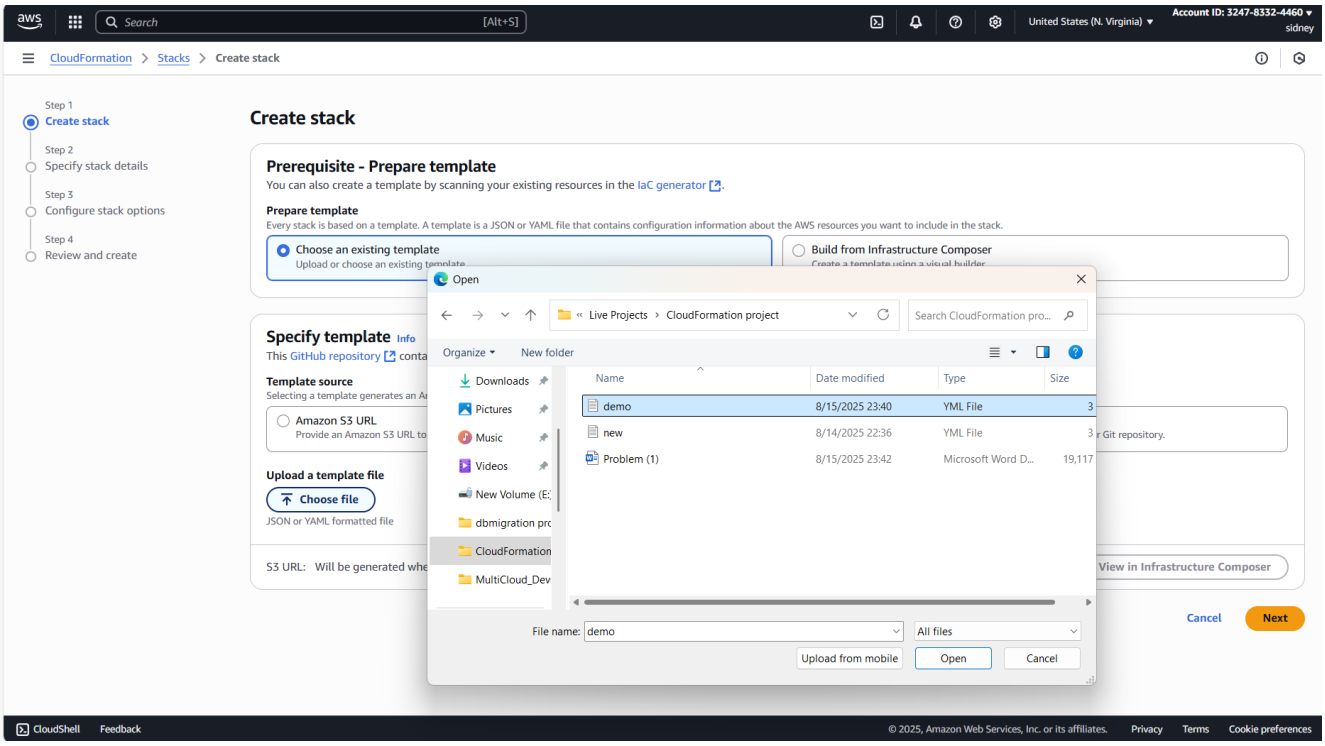
On "**Template Source**", select "**Upload a template file**"



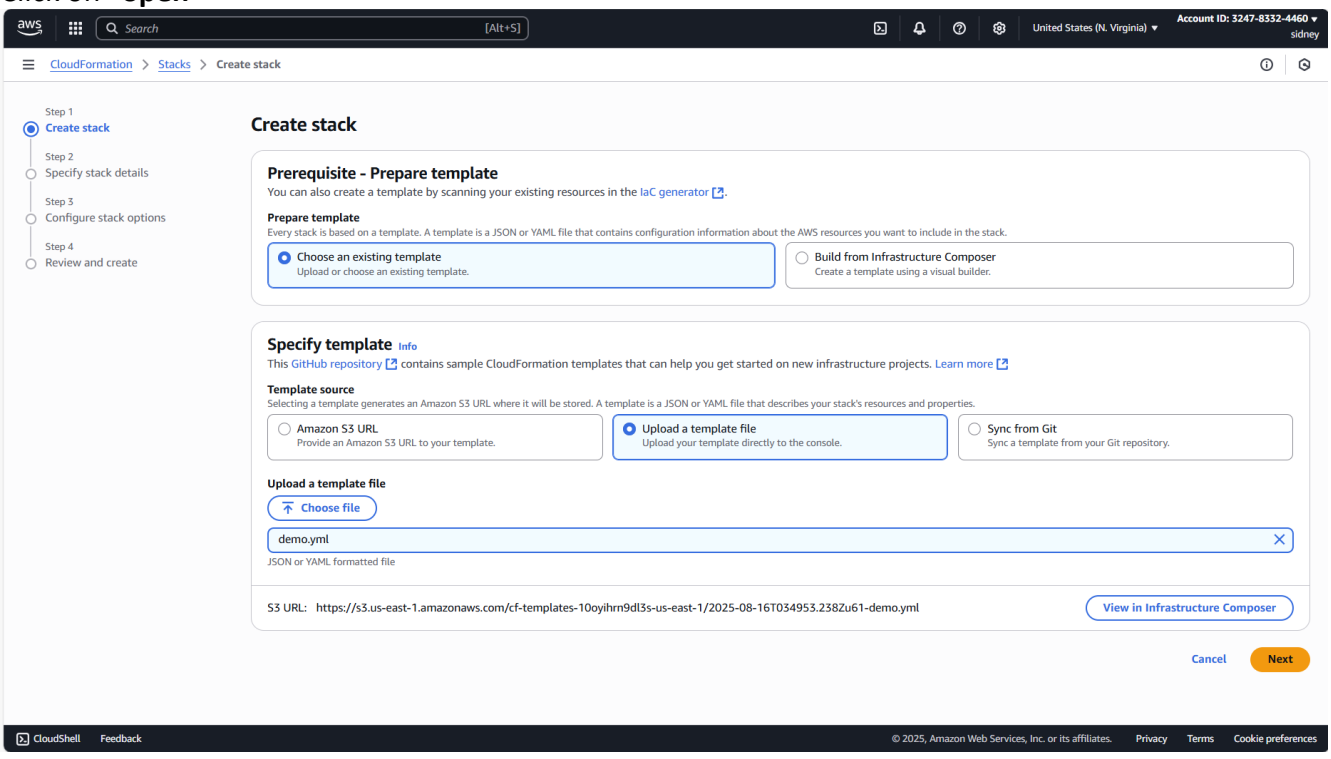
Click on "**Choose file**"



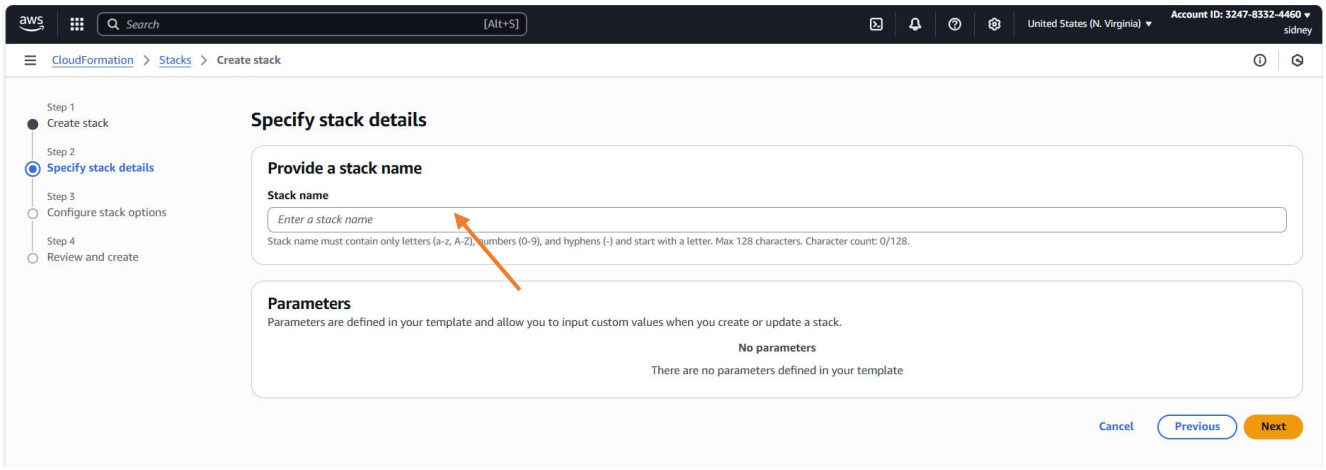
Navigate to where our configuration file is stored



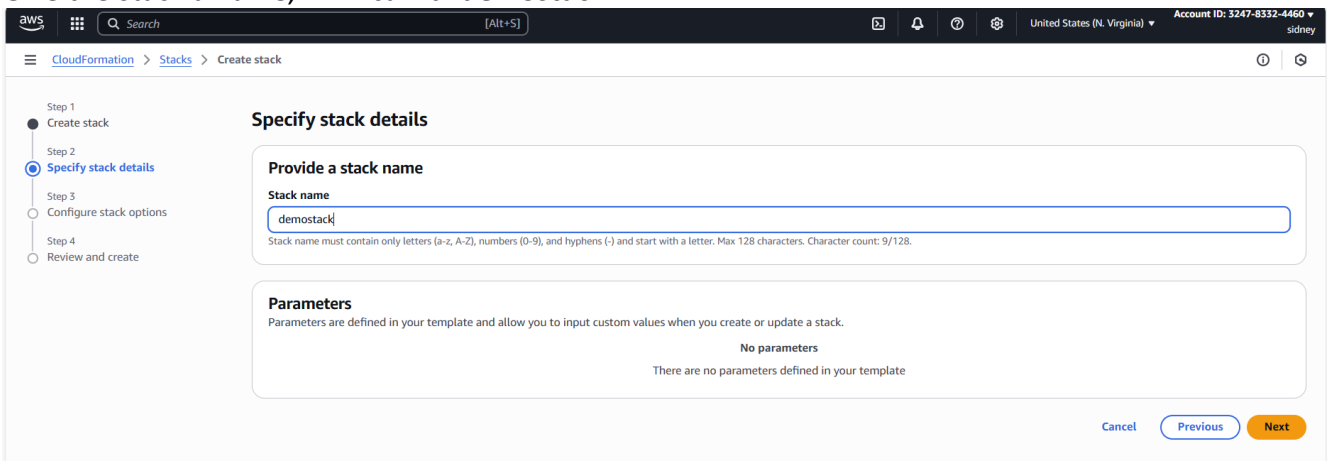
Click on “**open**”



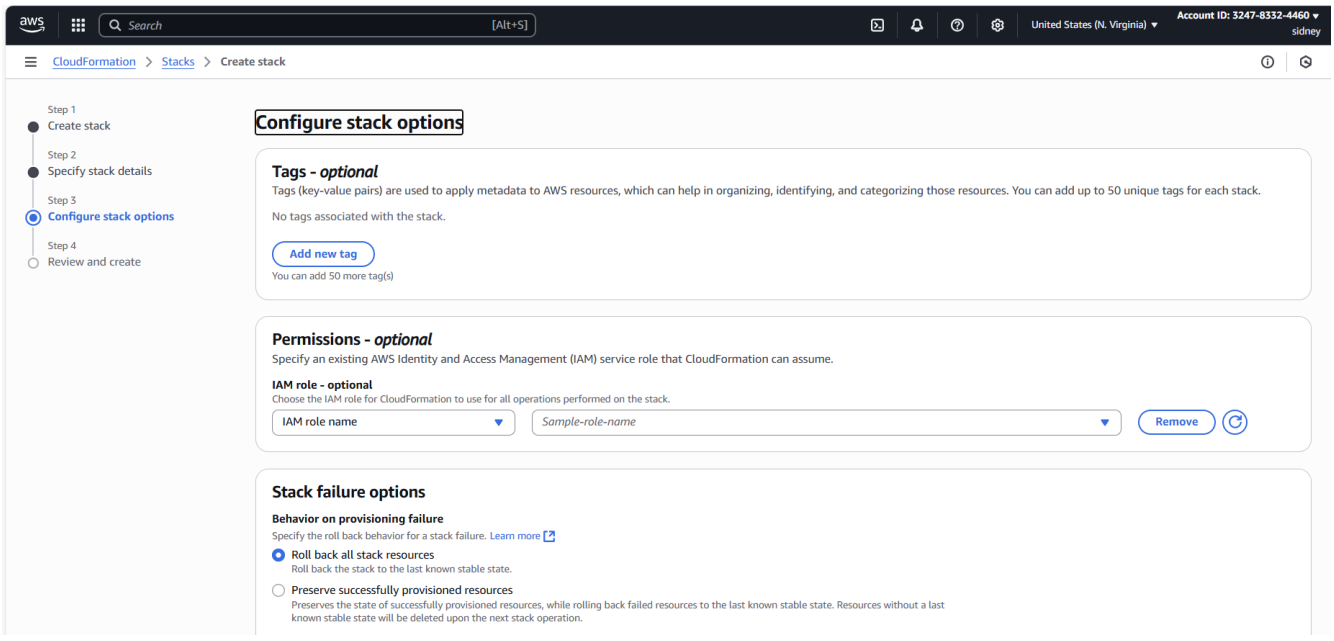
Click on “**Next**”

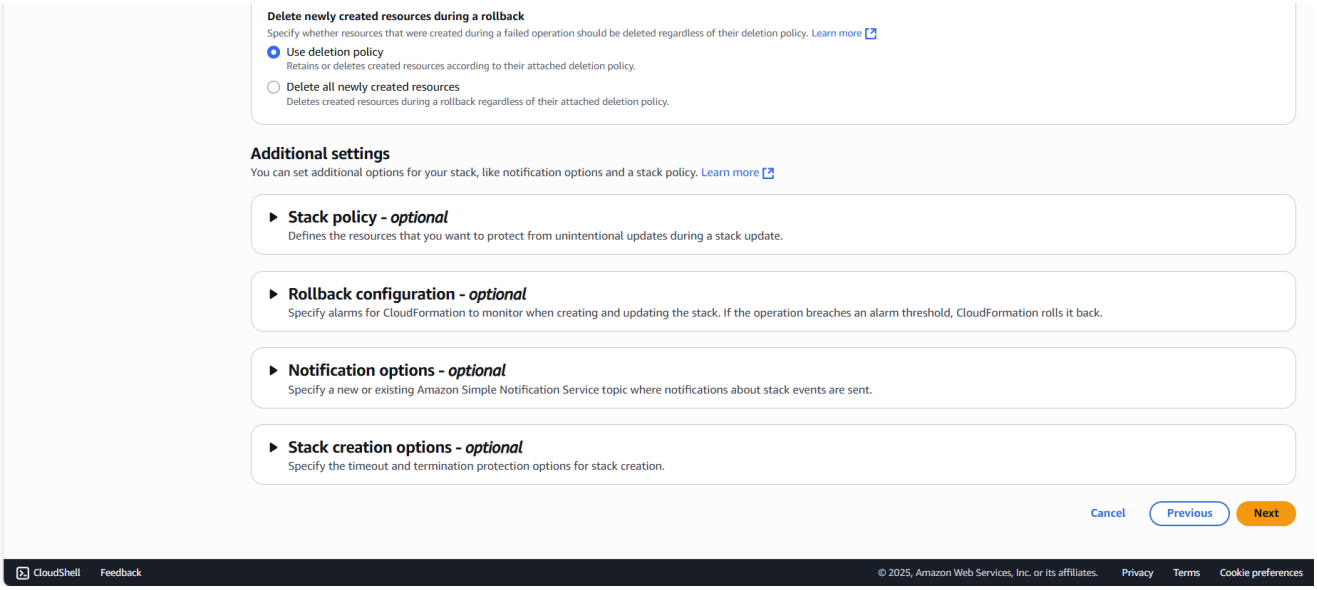


Give the stack a name, I will call it "**demostack**"

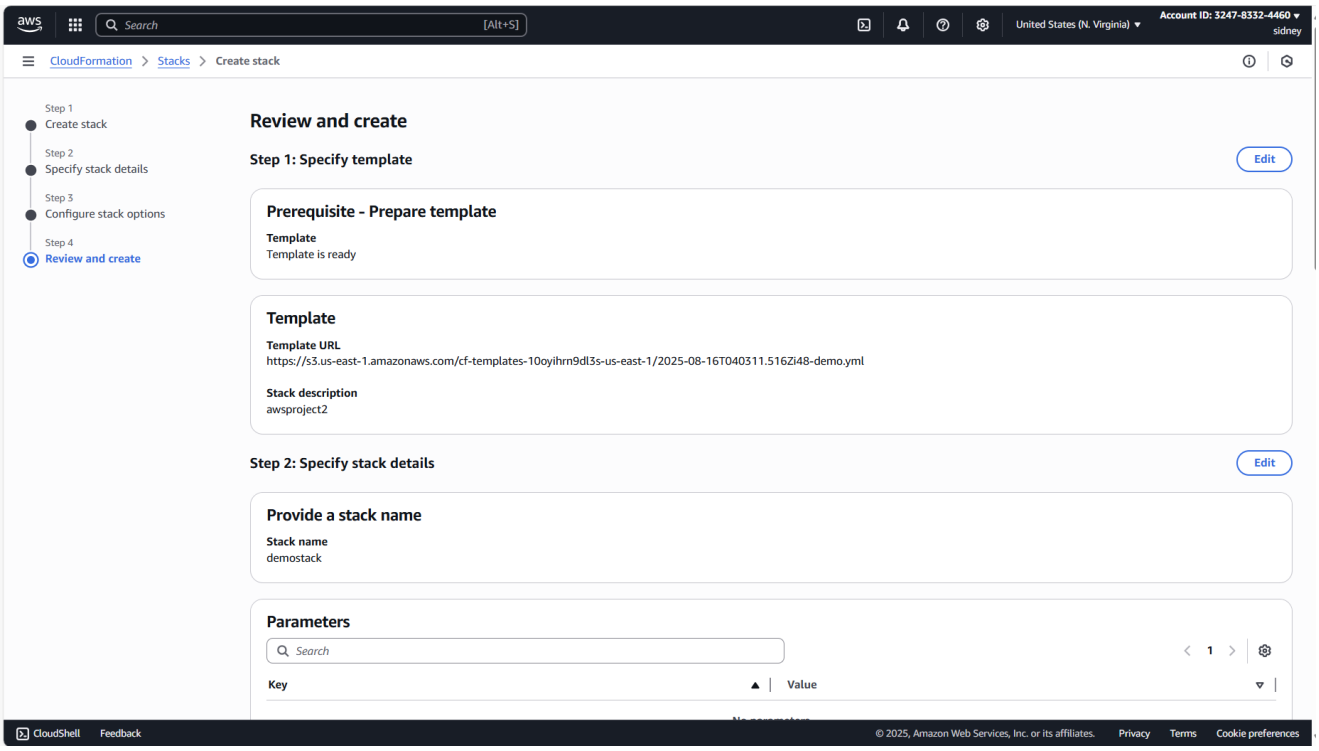


Click on "**Next**"

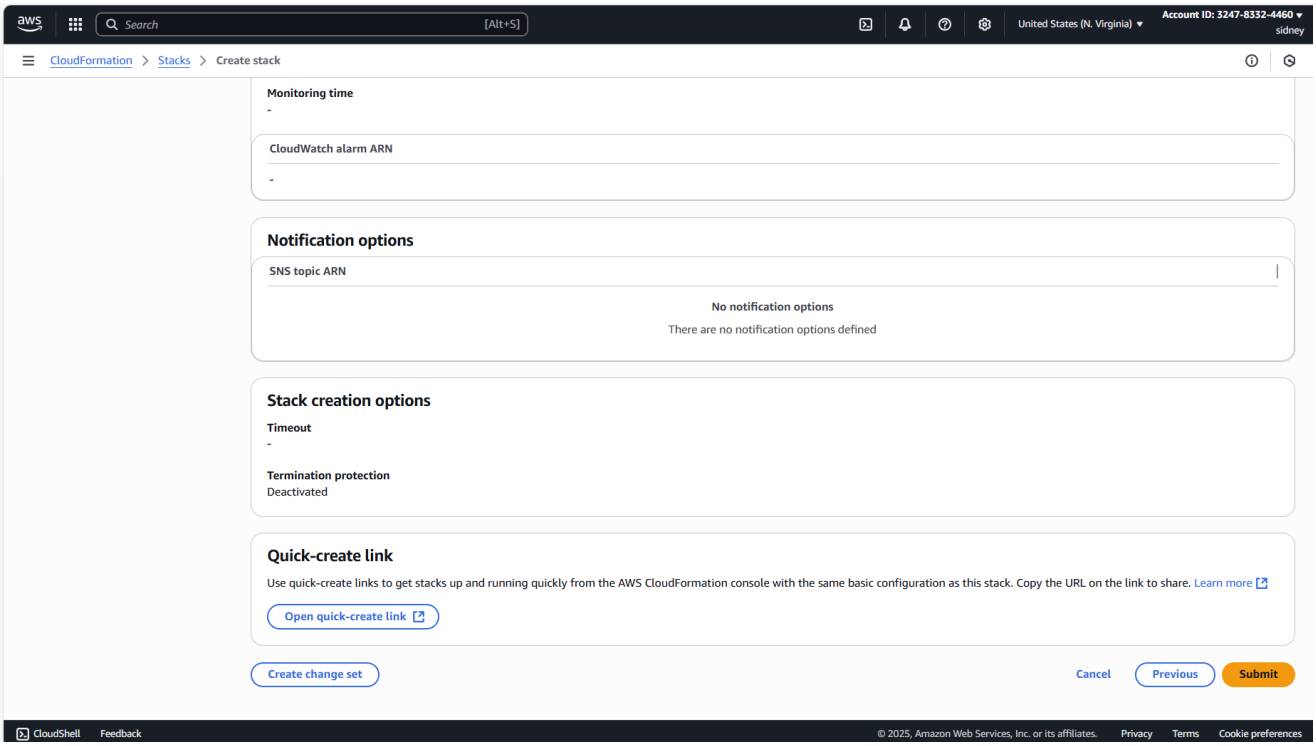




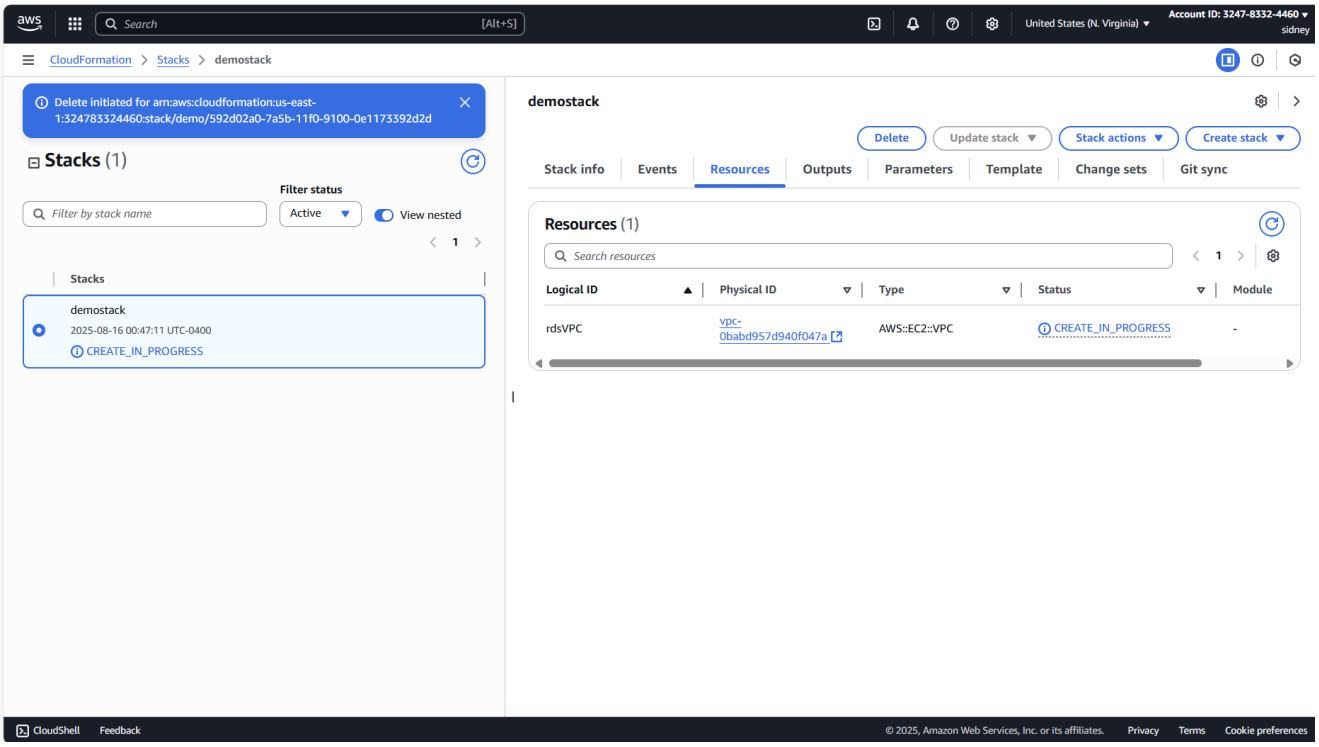
Click on “Next”



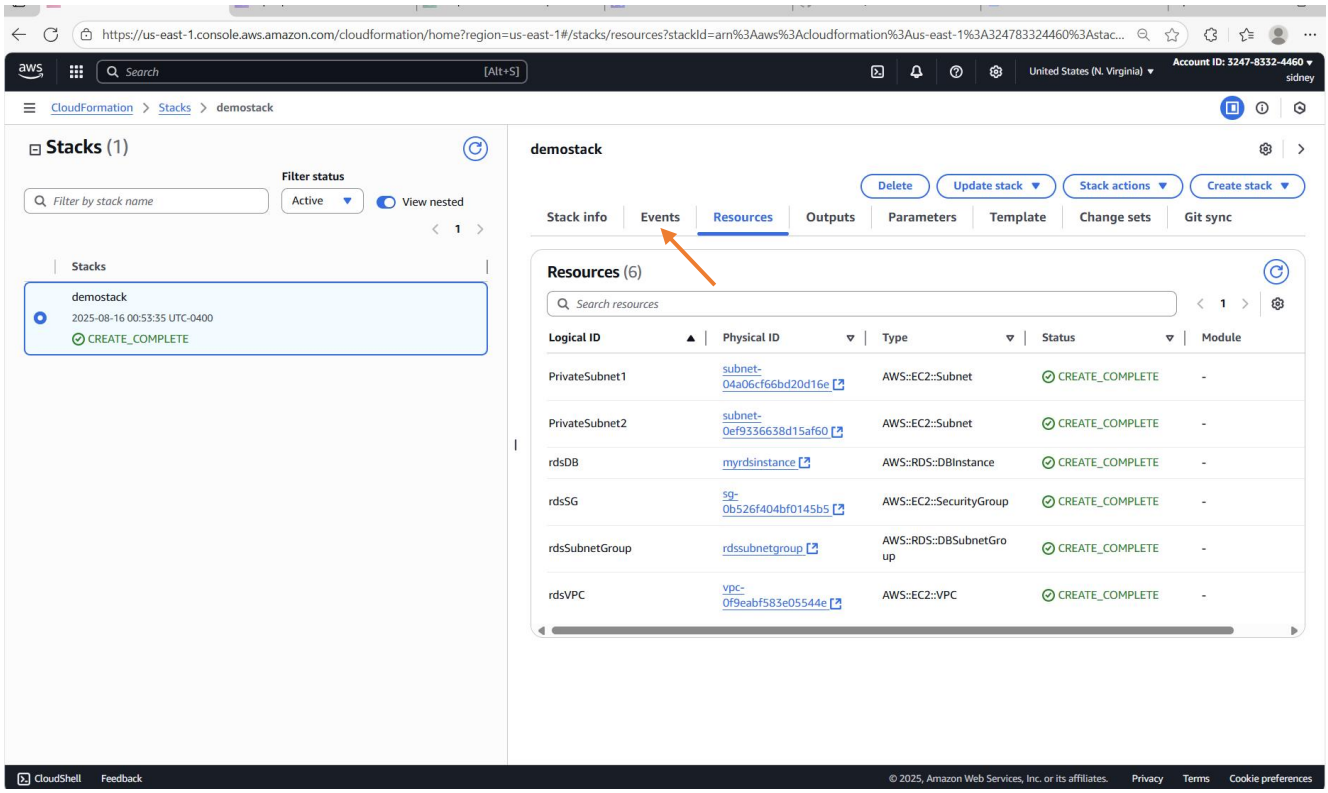
Review



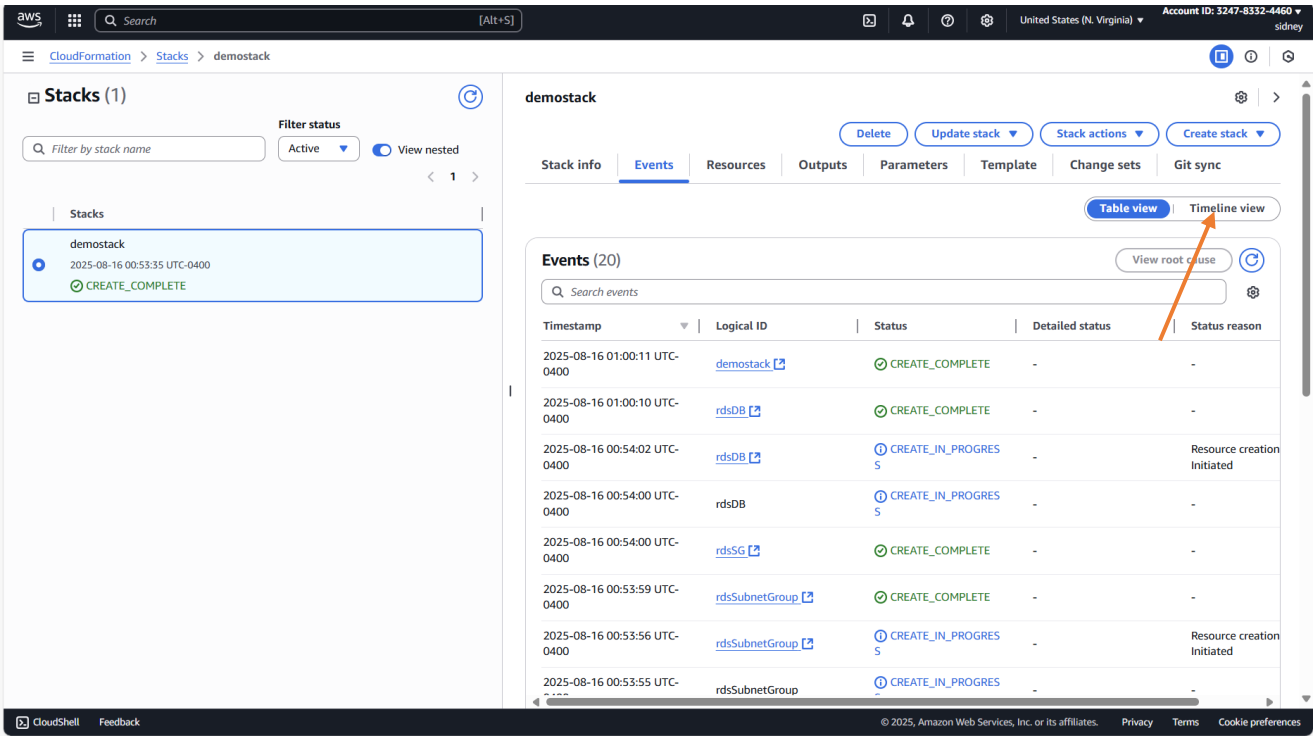
Click on “Submit”



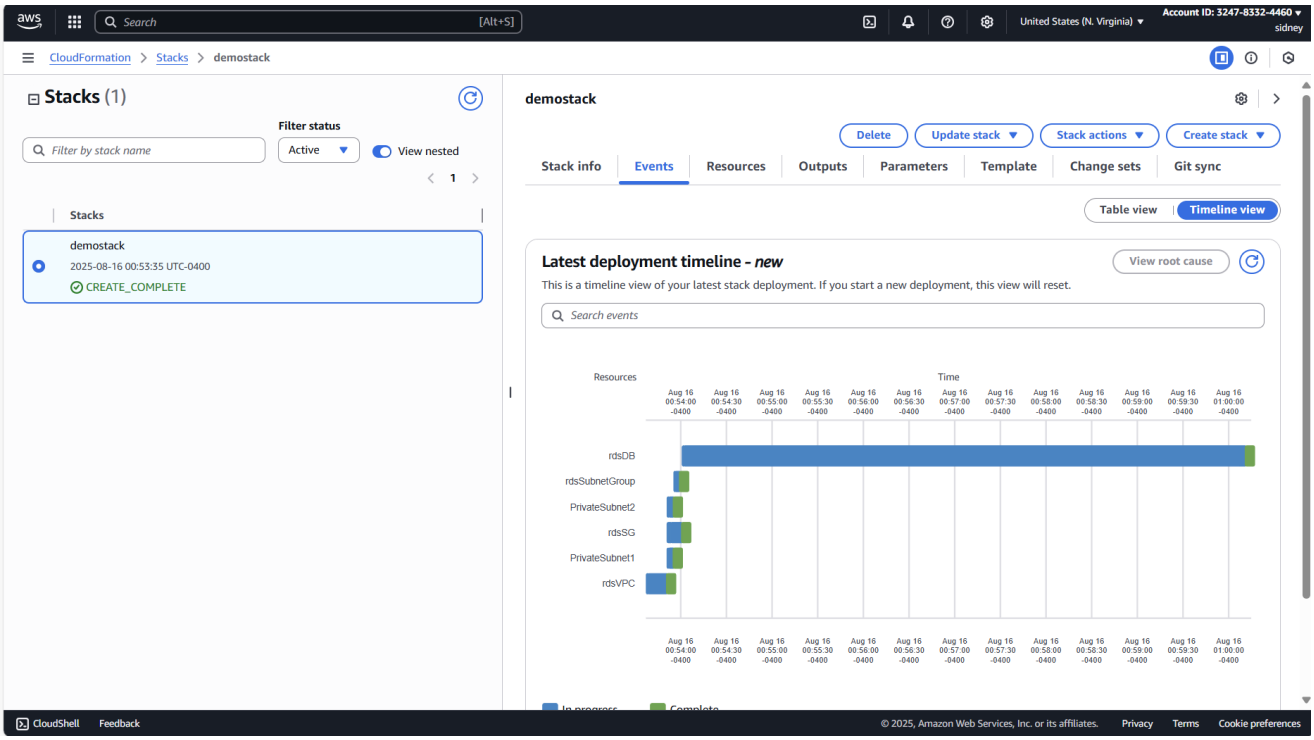
The stack is being created



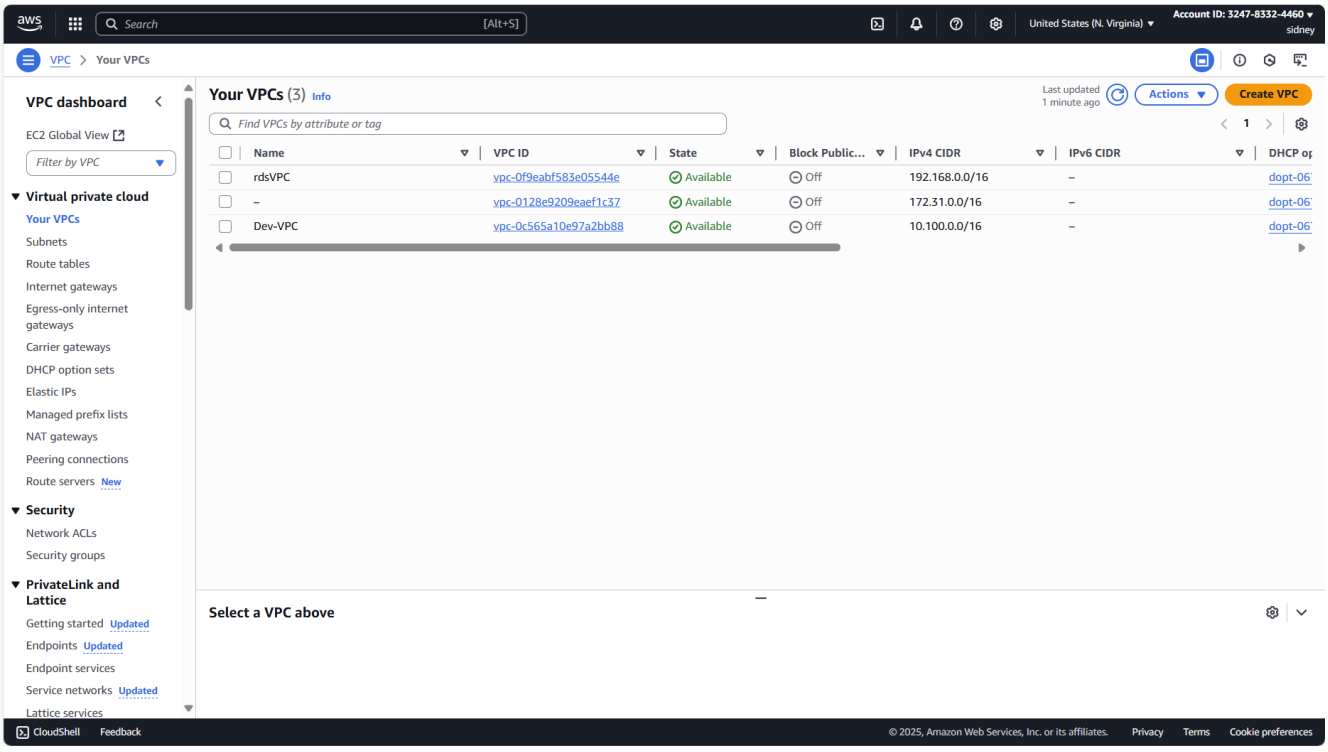
The stack has been created. Click on “Events” tab



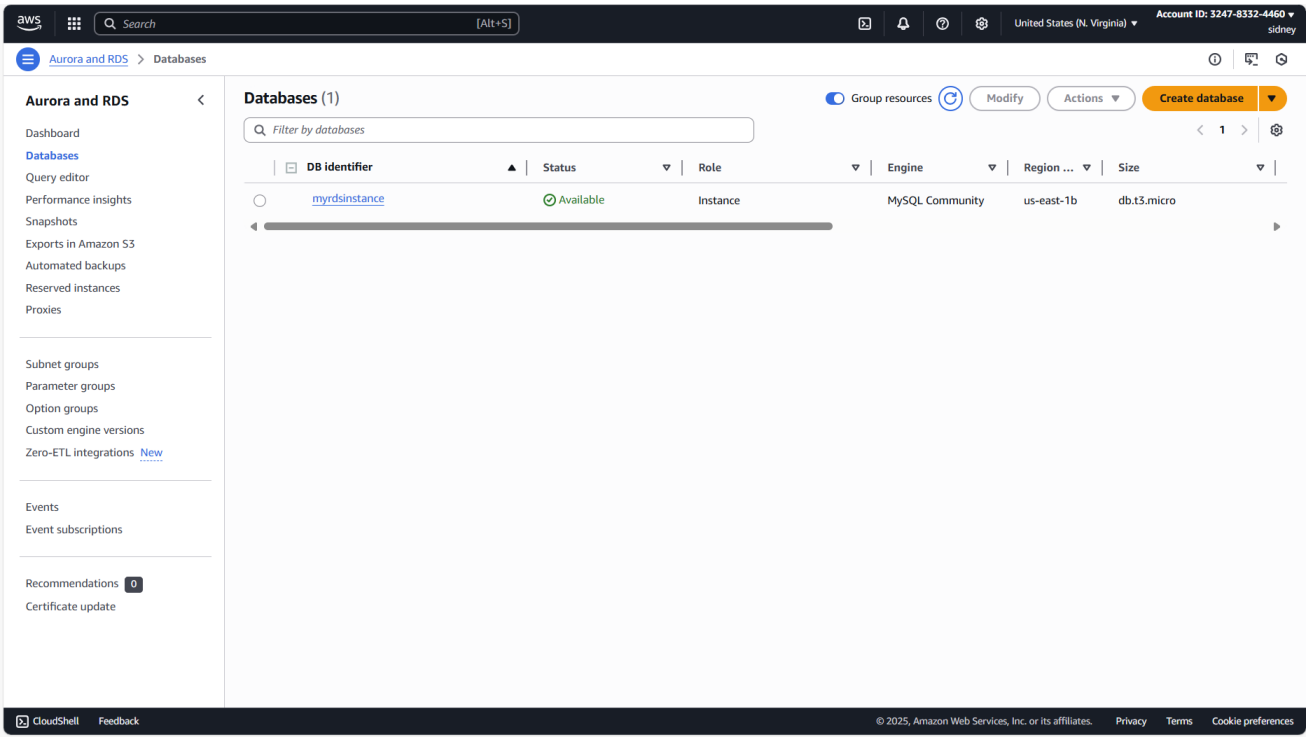
Click on “Timeline View”



Let us check if our resources have been created. Go to VPC dashboard on AWS Console

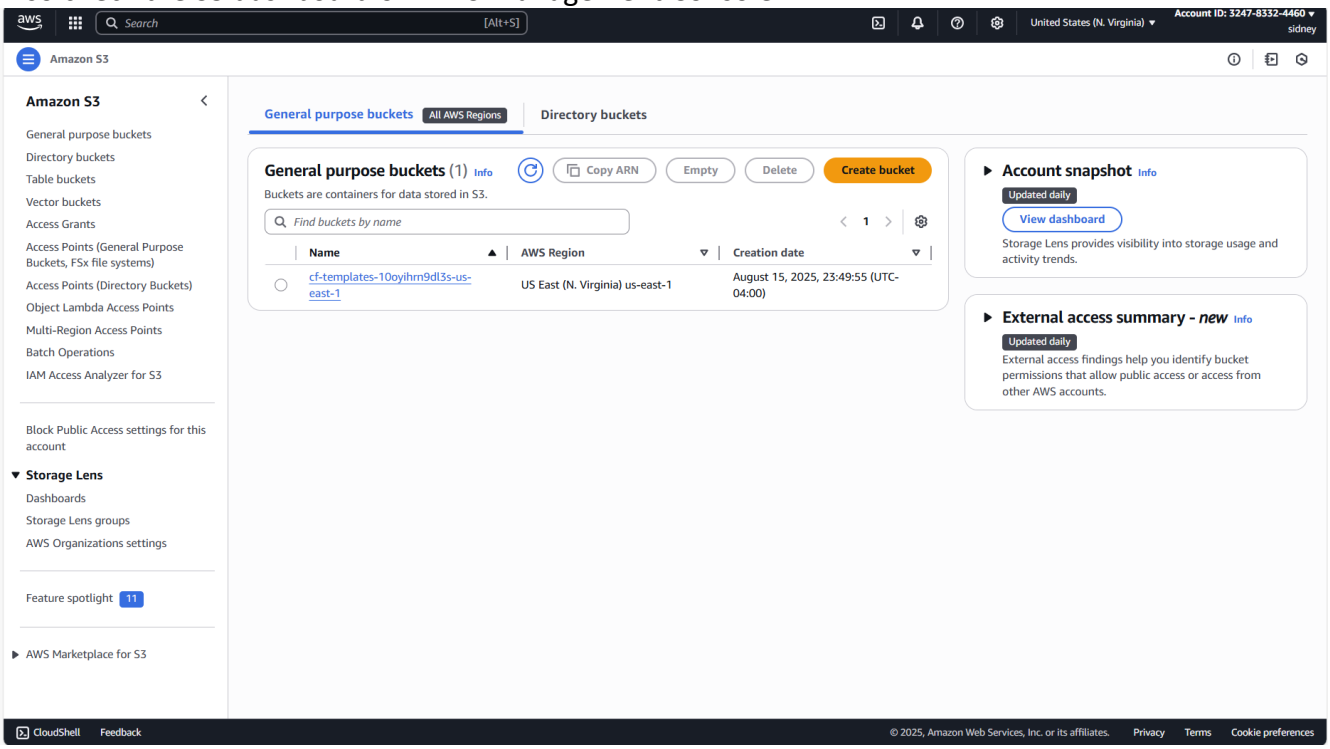


You can see that the VPC has been created. Also go to the Database dashboard on AWS Management console



You can see that our Database has been created

Also check the S3 dashboard on AWS Management Console



You can see that an S3 bucket has also been created.