Allianz Test ( VPC Solution )

# 1. Overview

This document describes the AWS infrastructure created using the provided CloudFormation template. The stack deploys a serverless VPC management solution that includes a Lambda function, API Gateway, DynamoDB for persistence, and Cognito for authentication.

# 2. Resources Created

* AWS::DynamoDB::Table – Stores VPC information (VpcInfoTable).
* AWS::IAM::Role – Execution role for Lambda function with necessary policies (LambdaExecutionRole).
* AWS::Lambda::Function – The main logic to create and retrieve VPC information (MyLambdaFunction).
* AWS::Cognito::UserPool – User authentication pool (VpcUserPool).
* AWS::Cognito::UserPoolClient – Client for user pool (VpcUserClient).
* AWS::ApiGateway::RestApi – REST API exposed to clients (NoAuthVpcAPI).
* AWS::ApiGateway::Resource – Resource path /invoke under the REST API.
* AWS::ApiGateway::Method – GET and POST methods integrated with the Lambda function.
* AWS::ApiGateway::Authorizer – Cognito authorizer for API Gateway.

# 3. Lambda Function Flow

The Lambda function handles HTTP POST and GET requests:  
- POST: Creates a VPC, subnets, IGW, route table, and stores metadata in DynamoDB.  
- GET: Retrieves VPC information by name or returns all records.

# 4. IAM Role and Permissions

The LambdaExecutionRole grants the Lambda function permissions to interact with EC2, DynamoDB, CloudWatch Logs, and assumes the role via sts:AssumeRole. The role includes policies for managing network resources and storing/retrieving data from DynamoDB.

# 5. API Gateway Integration

API Gateway exposes two endpoints (GET and POST) under the path '/invoke'. Both methods use the AWS\_PROXY integration type and are authorized using Cognito User Pools.

# 6. Cognito User Pool Authentication

Cognito User Pool (VpcUserPool) is used for authenticating users via email. A user pool client is created without a client secret. API Gateway uses this pool to authorize requests.

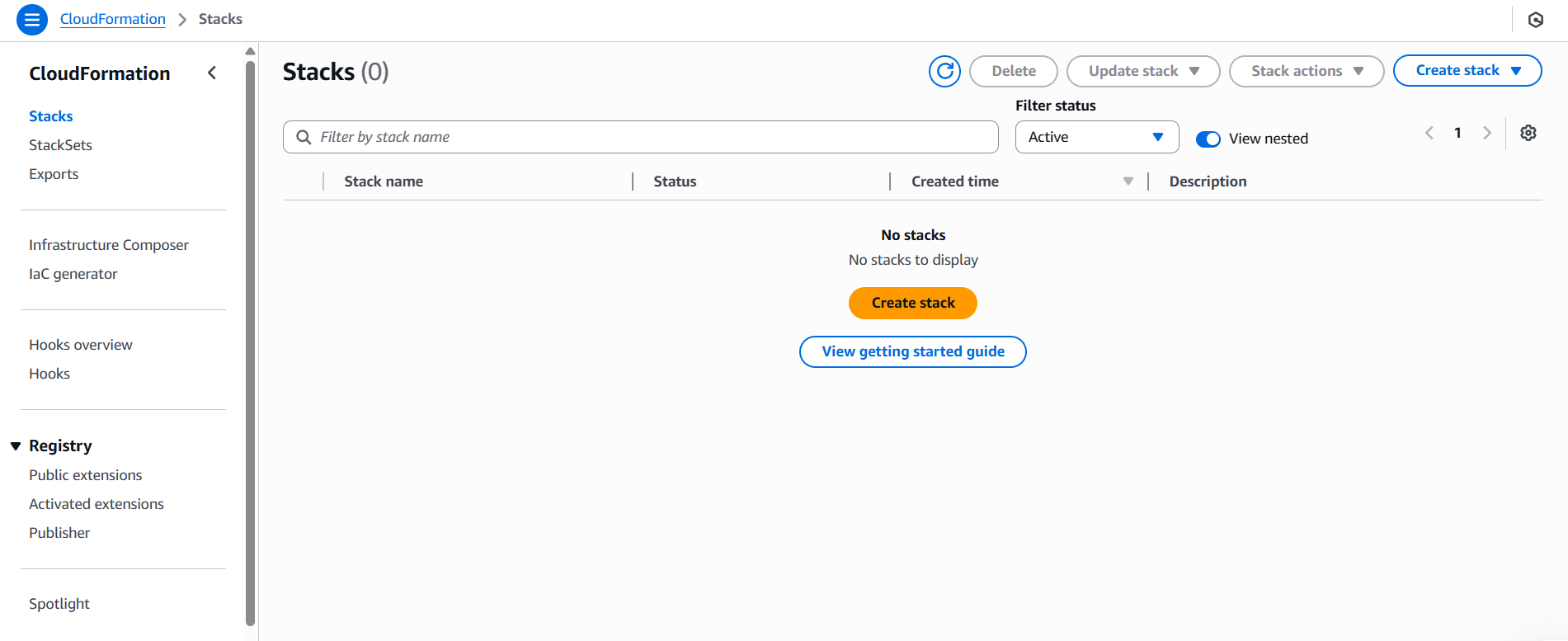
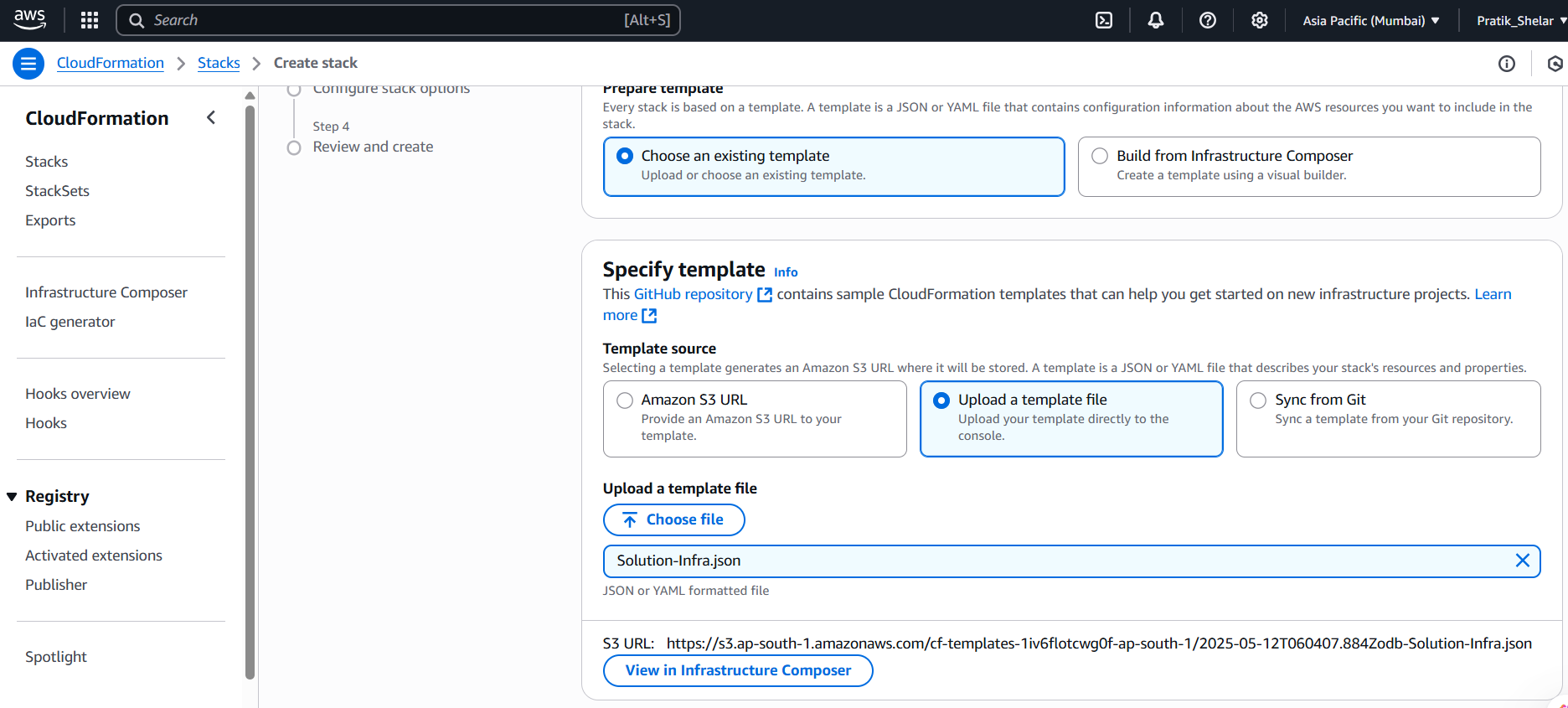
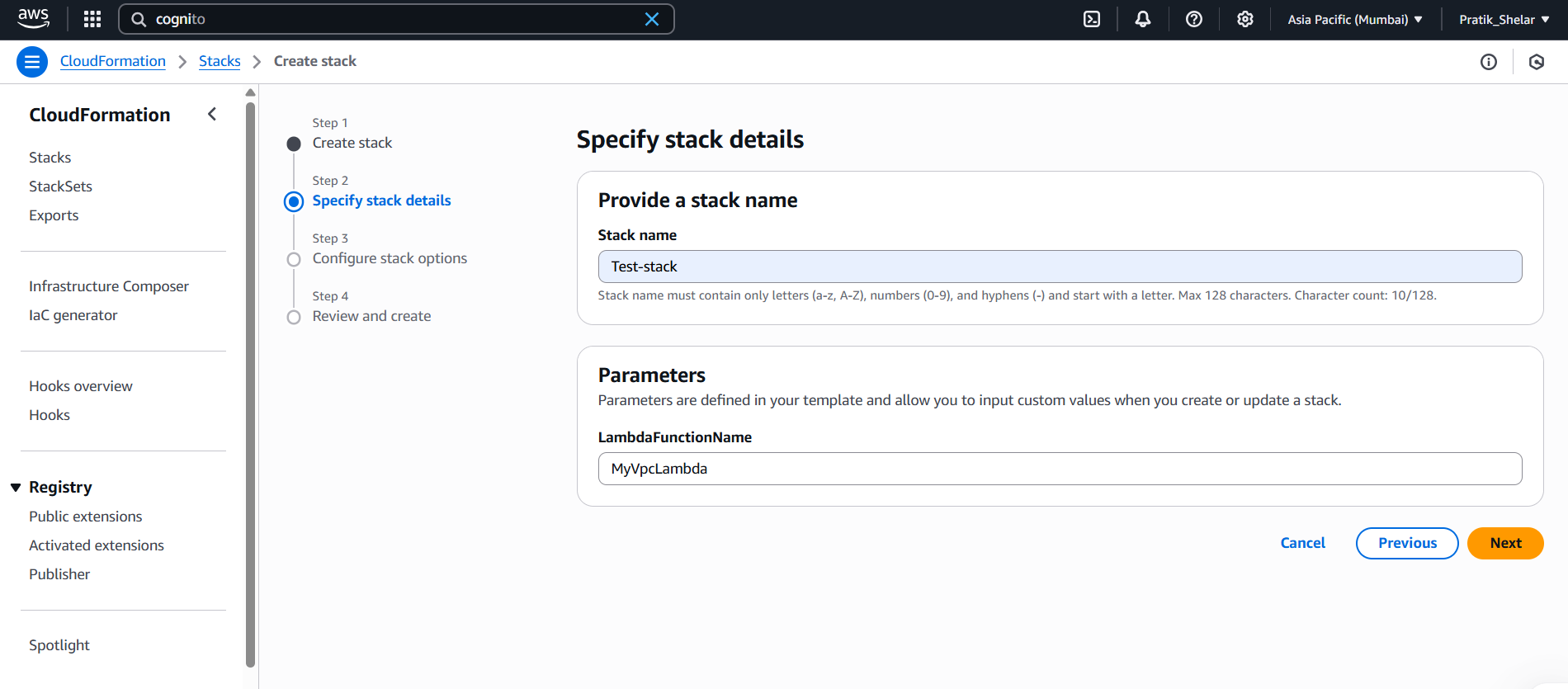
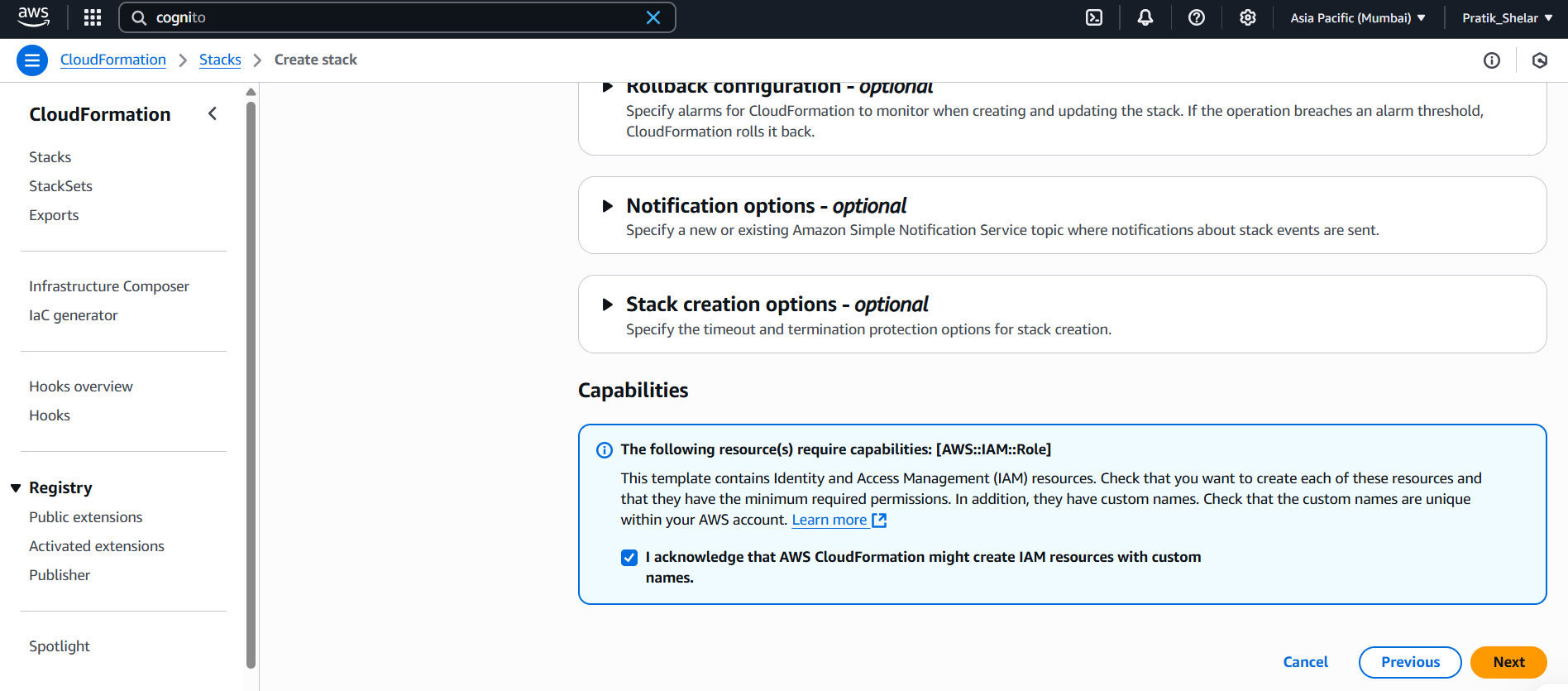
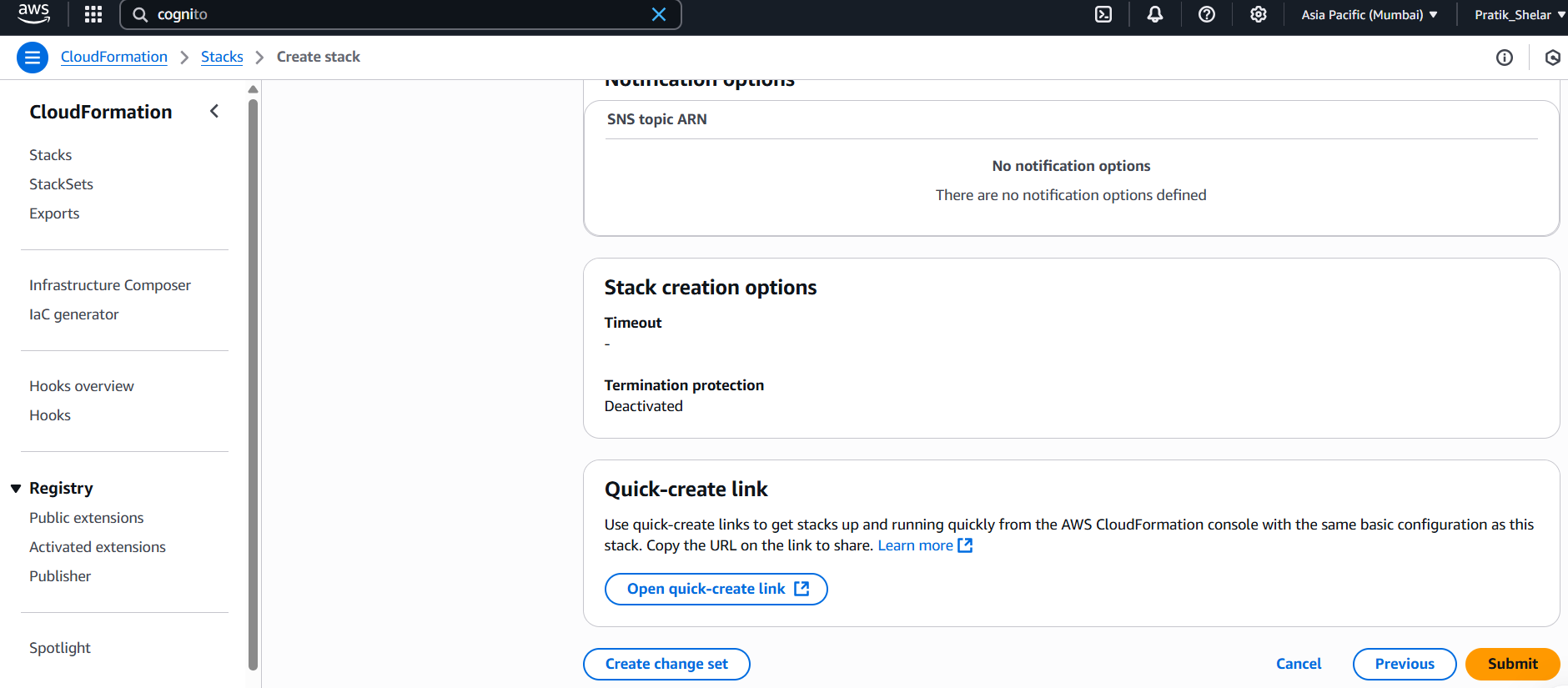
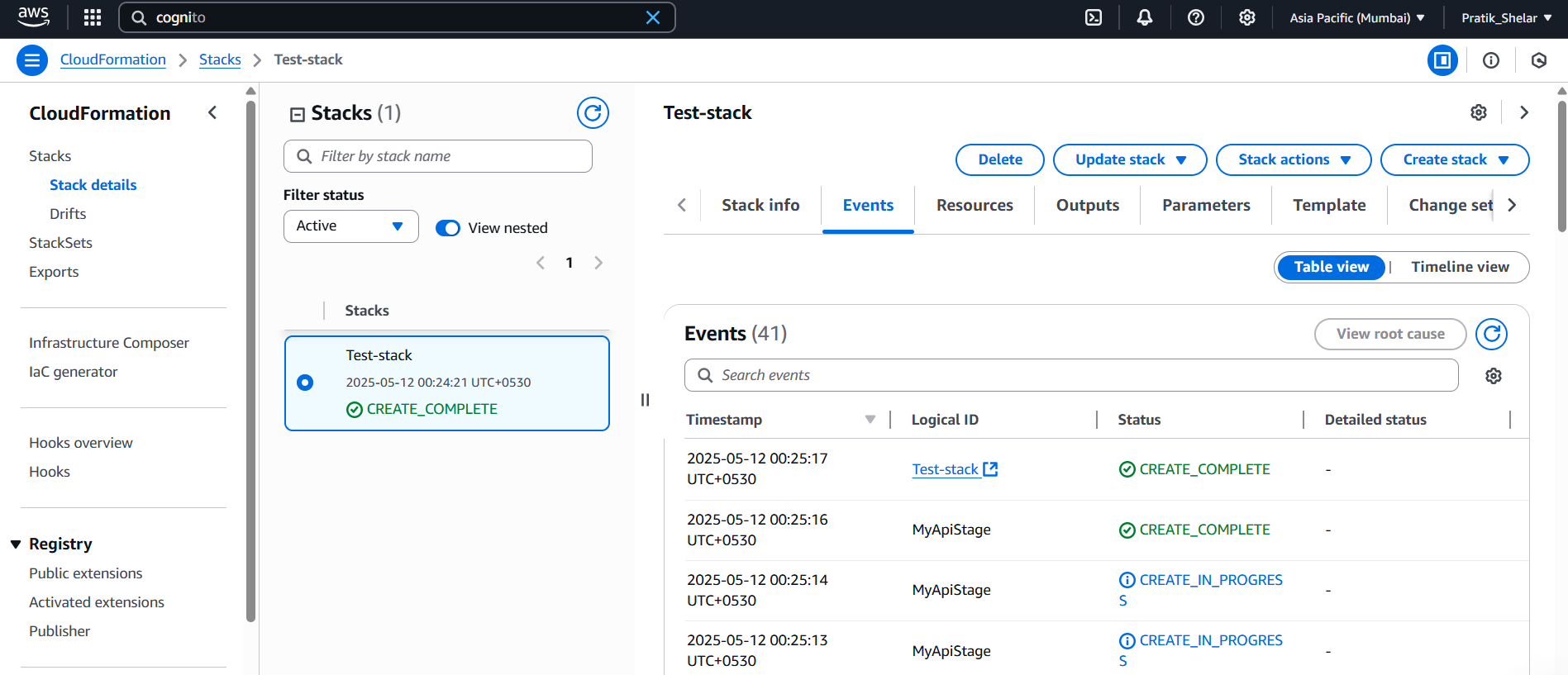
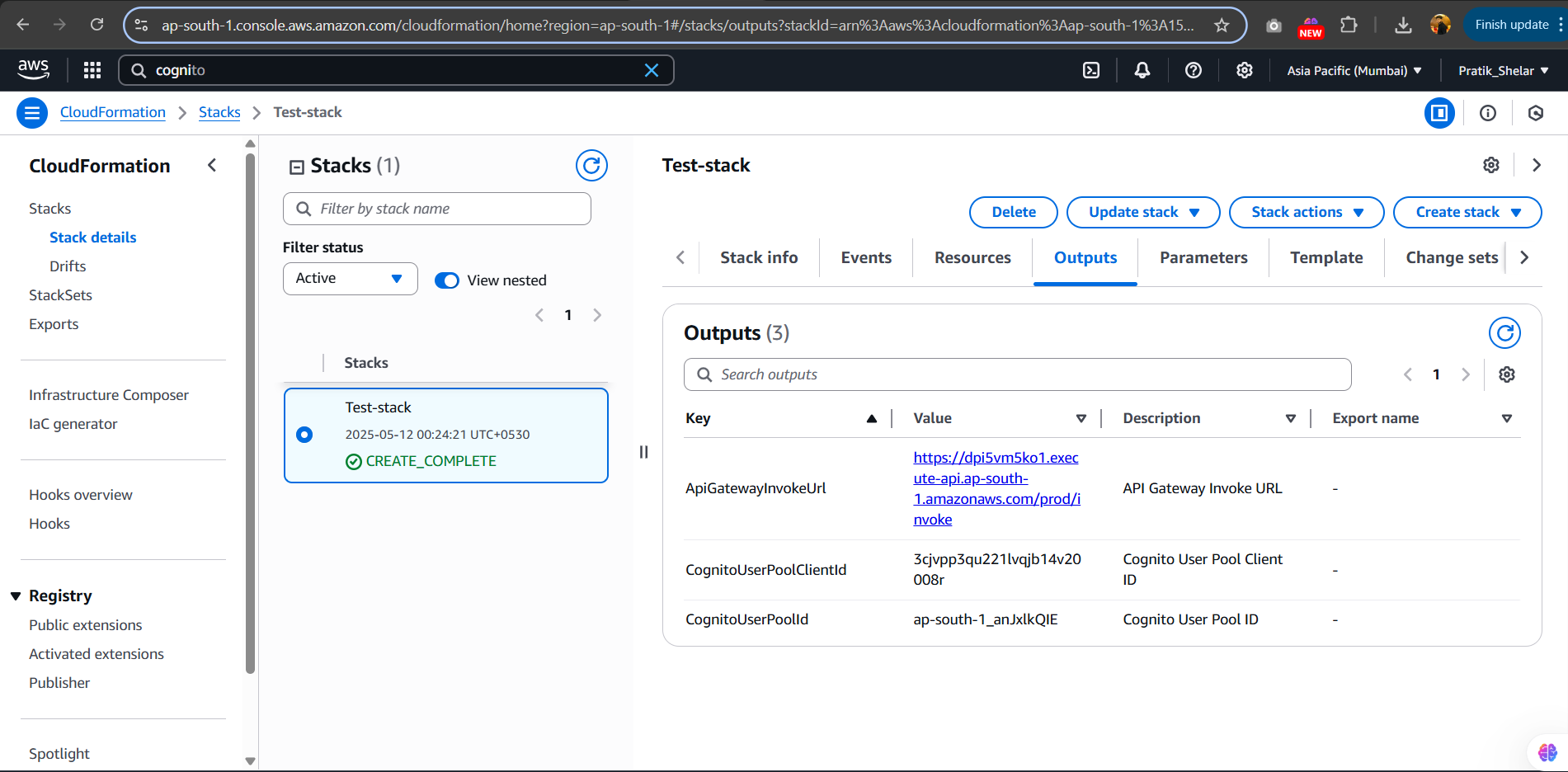
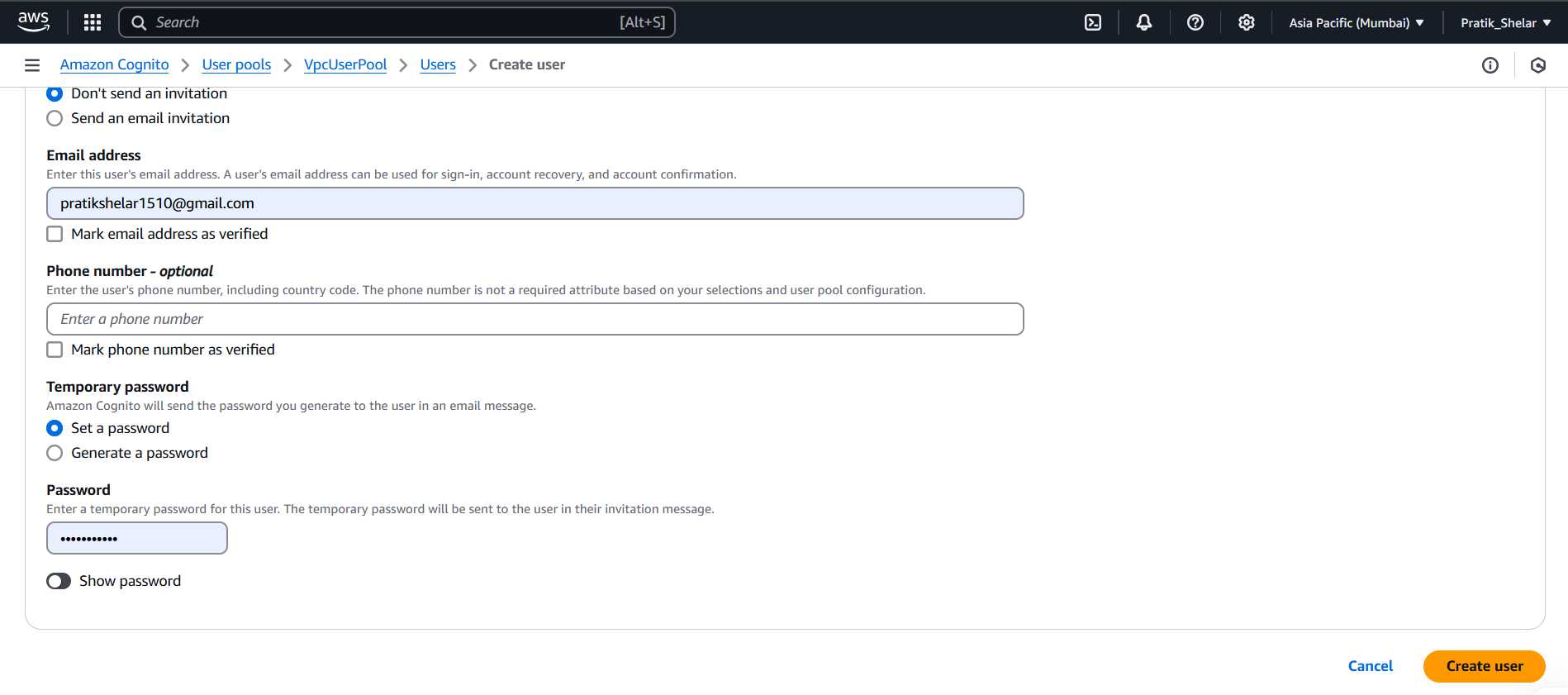
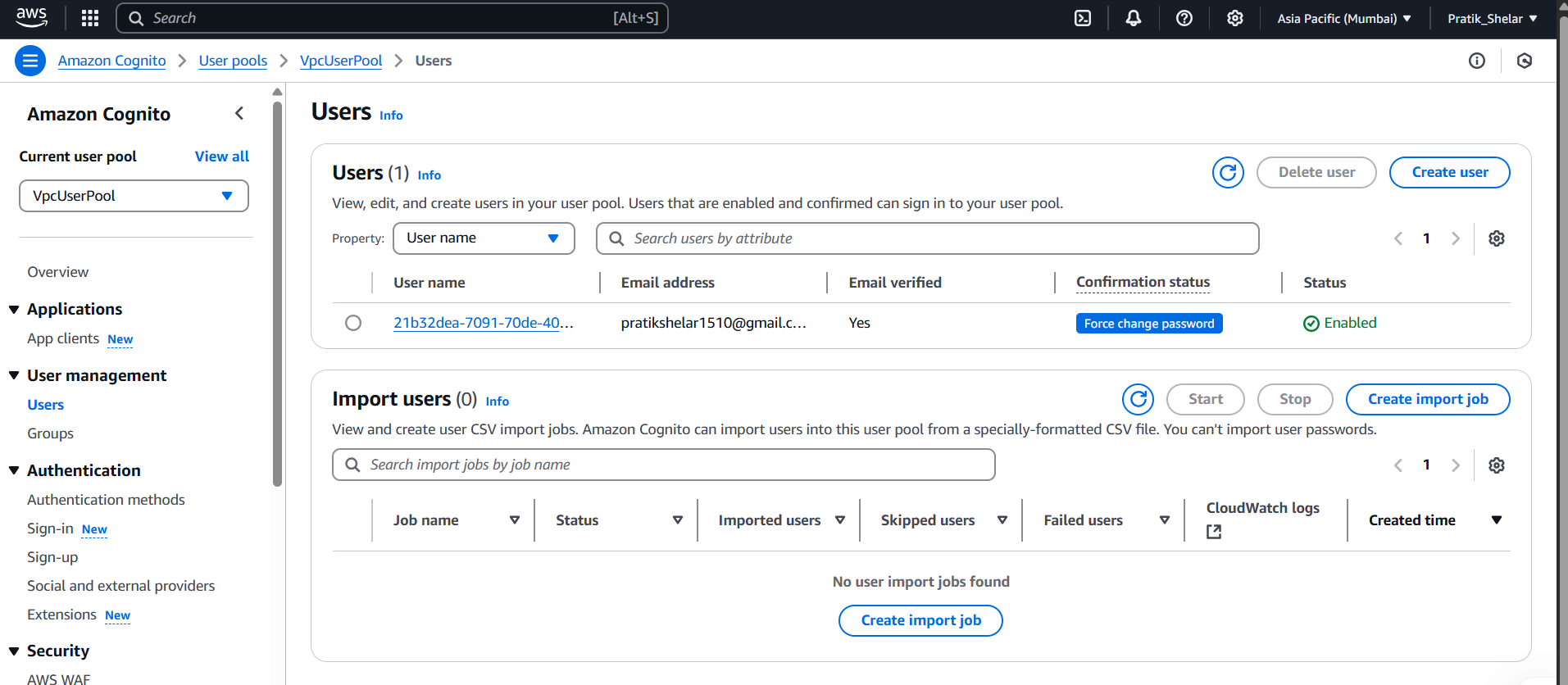
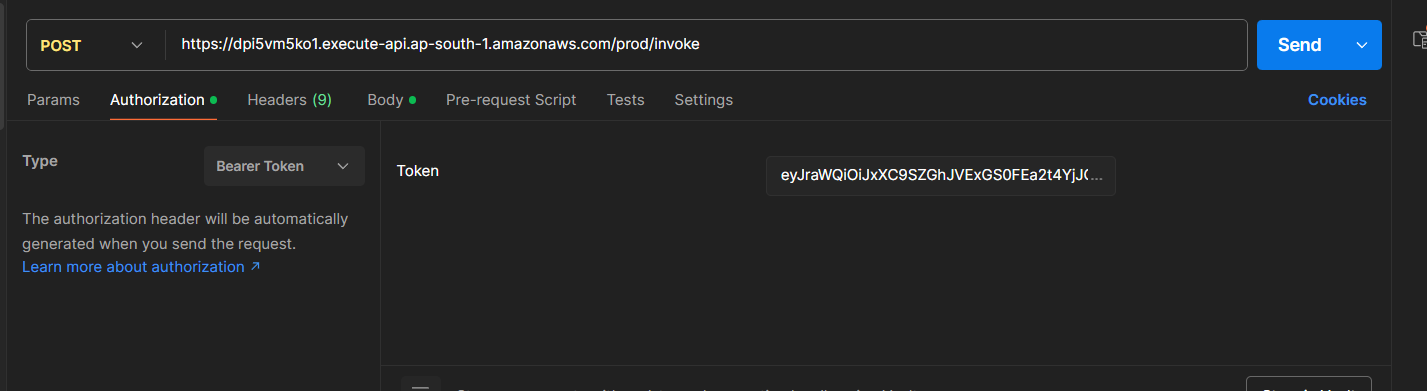
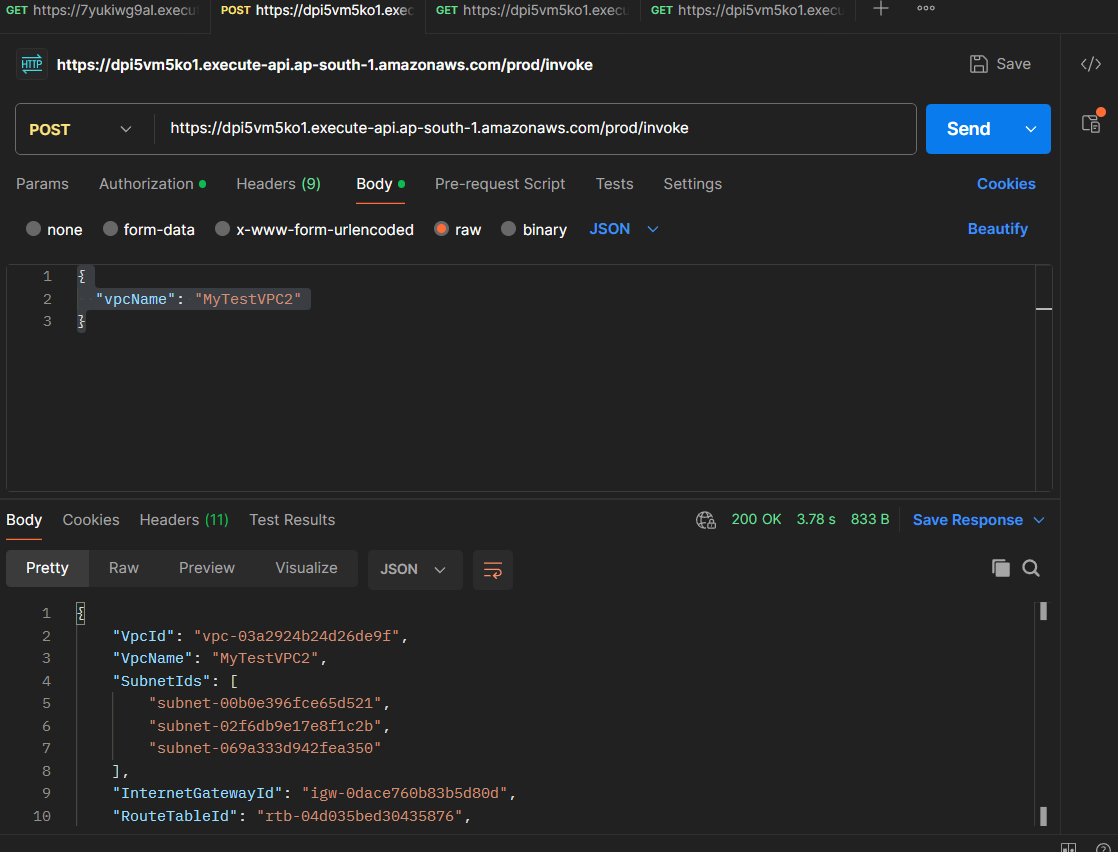
# 7. DynamoDB Table

The VpcInfoTable stores information about each created VPC. The primary key is VpcId (string). Lambda stores VpcId, VpcName, InternetGatewayId, SubnetIds, and RouteTableId for each entry.

# 8. Outputs

* CognitoUserPoolId – ID of the created Cognito User Pool.
* CognitoUserPoolClientId – ID of the created Cognito User Pool Client.
* ApiGatewayInvokeUrl – Invoke URL for the deployed API.

# Step By Step Instructions to run the solution

1. Login to Github and clone the repository in your local machine
   1. **Repo Link:** [**https://github.com/pratik1415/Allianz.git**](https://github.com/pratik1415/Allianz.git)
2. Go to AWS Console and Login
3. Open Cloudformation Service from the console
   1. 
4. Click on Create Stack and upload **“Solution-Infra.json”** file.
   1. 
5. Click on next and Provide stack name and Lambda Function Name(Default name will be MyVpcLambda, you can change it)
   1. 
6. Click next and select “**I acknowledge that AWS CloudFormation might create IAM resources with custom names.”** and click on next.
   1. 
7. Click Submit
   1. 
8. Wait until stack creation will not get complete.
9. Once stack creation status is “CREATE\_COMPLETE”, your service got created.
   1. 
10. Goto ***Outputs*** tab, you find the ApiGatewayInvokeUri, CognitoUserPoolId, CognitoUserPoolClientId.
    1. 
11. Goto User Pools in Amazon Cognito service and in Left side you will get the Users tab under User Management.
12. Under Users, you will find the option Create user. Click on it and fill the details. Make sure you will check the ***“Mark email address as verified”***
    1. 
13. Once user create, keep the user name and password
    1. 
14. Open TokenGen.py file and update the below details:
    1. **Username:** Congito User Name
    2. **Password:** Cognito User Password
    3. **ClientID:** Cognito User Pool Client ID ( you will get it either in Userpool or in output tab in CloudFormation created stack )
    4. **UserPoolID:** User Pool ID ( you will get it either in Userpool or in output tab in CloudFormation created stack )
15. Once Done, run the script using below command and get the ID Token
    1. python TokenGen.py
16. Goto Postman
    1. Select Post method, and ApiGatewayInvokeUri ( You will get it in output tab in CloudFormation created stack )
    2. Goto Authorization Tab, Select Bearer Token in type and insert your ID token.
    3. 
    4. Goto Body -> Select raw -> select JSON-> add vpcName ( Mandatory ) -> Click on send.
    5. 
    6. Your VPC Will Get Created.
    7. Got Get, Do the same process like POST but Method Will be GET
    8. Using Curl Command
       1. POST (Create VPC):  
          ```  
          curl -X POST https://<api-id>.execute-api.<region>.amazonaws.com/prod/invoke \  
           -H "Authorization: <JWT\_TOKEN>" \  
           -H "Content-Type: application/json" \  
           -d '{"vpcName": "MyCustomVPC"}'  
          ```
       2. GET (Retrieve VPCs):  
          ```  
          curl -X GET https://<api-id>.execute-api.<region>.amazonaws.com/prod/invoke?vpcName=MyCustomVPC \  
           -H "Authorization: <JWT\_TOKEN>"  
          ```
    9. Also you will get your data in browser also using below link
       1. https://{api-id}.execute-api.{region}.amazonaws.com/prod/invoke?vpcName={your vpcNAme}
17. ID Token Authorization Validation:
    1. Goto API Gateway -> API -> Select your API -> Goto Authorizer -> Select your cognito authorizer -> Test Authorizer -> Put your ID Token and click on Test authorizer
       1. 