Highlights

```
# **High-Level AWS VPC API Solution Steps**

## **Solution Overview**
This solution automates the creation of a Virtual Private Cloud (VPC) with
multiple subnets using an AWS Lambda-based API, secured with AWS Cognito
authentication. The infrastructure is deployed using Terraform, and the API is
accessible via API Gateway.

---

## **Steps to Implement the Solution**

### **Step 1: Provision AWS Infrastructure with Terraform**
- Set up AWS provider configuration.
- Create a **DynamoDB table** (`VPCRecords`) to store VPC data.
- Deploy **API Gateway** to expose the API.
- Configure API Gateway with an **authenticated route** (`/vpc`) using
Cognito.
- Define **Lambda function** to handle API requests.
- Set up **IAM roles** to allow Lambda access to DynamoDB.
- Deploy **Cognito User Pool** to manage authentication.
- Configure **API Gateway Authorizer** to enforce Cognito-based
authentication.

### **Step 2: Implement AWS Lambda Function**
- Implement a Python-based Lambda function to:
 - Create a VPC with a default CIDR block.
 - Create multiple subnets dynamically.
 - Store created VPC and subnet details in **DynamoDB**.
 - Handle API authentication using Cognito.
- Package and deploy the Lambda function.

### **Step 3: Secure the API with Cognito**
- Set up **Cognito User Pool** to manage authentication.
- Configure an API Gateway **Cognito Authorizer**.
- Users must authenticate with Cognito before accessing the API.

### **Step 4: Deploy & Validate the API**
- Initialize and apply **Terraform scripts** to provision AWS resources.
- Retrieve the API Gateway **endpoint URL**.
- Create a test user in Cognito and obtain an authentication token.
- Send API requests using the Cognito token.
- Verify that VPCs and subnets are created and stored in **DynamoDB**.

### **Step 5: Execute your terraform**
- Code goes here
### **Step 6: Cloudwatch logging can be enabled in code**
```

Installed below -
Terraform
AWS Command Line
Python version latest

STEP1) aws configure:

```
export AWS_ACCESS_KEY_ID="XYZ"
export AWS_SECRET_ACCESS_KEY="ABC"
export AWS_REGION="us-east-1" ( As per need )
```

STEP2) Go to your terraform script directory to run:(Attached in email)

```
terraform init                  ( Initialize)
terraform plan                  ( Preview what is going to run )
terraform apply -auto-approve   ( Apply changes)
```

STEP3) Package and upload the code of lambda: (Attached in the email)
zip **lambda_function**.zip **lambda_function**.py
aws s3 cp **lambda_function**.zip s3://your-s3-bucket/   (Choose bucket name as per standard)
terraform apply -auto-approve

STEP4) Retrieve API gateway

Once deployment is complete, run:terraform output

Copy the **API Gateway URL** and later used it for testing

STEP5) Create a congnito user (via email account)

```
aws cognito-idp sign-up \
  --client-id your-client-id \
  --username bkhade \
  --password Bkhade@1234 \
  --user-attributes Name=email,Value=Bkhade@gmail.com
```

STEP6) Authentication token to test the API:

```
curl -X POST "https://your-api-id.execute-api.us-east-1.amazonaws.com/prod/create-vpc" \

   -H "Authorization: Bearer your-id-token" \

   -H "Content-Type: application/json" \
     -d '{"cidr_block": "10.0.0.0/16", "subnet_count": 3}'
```

STEP7) To remove all AWS resources, run:

```
terraform destroy -auto-approve
```