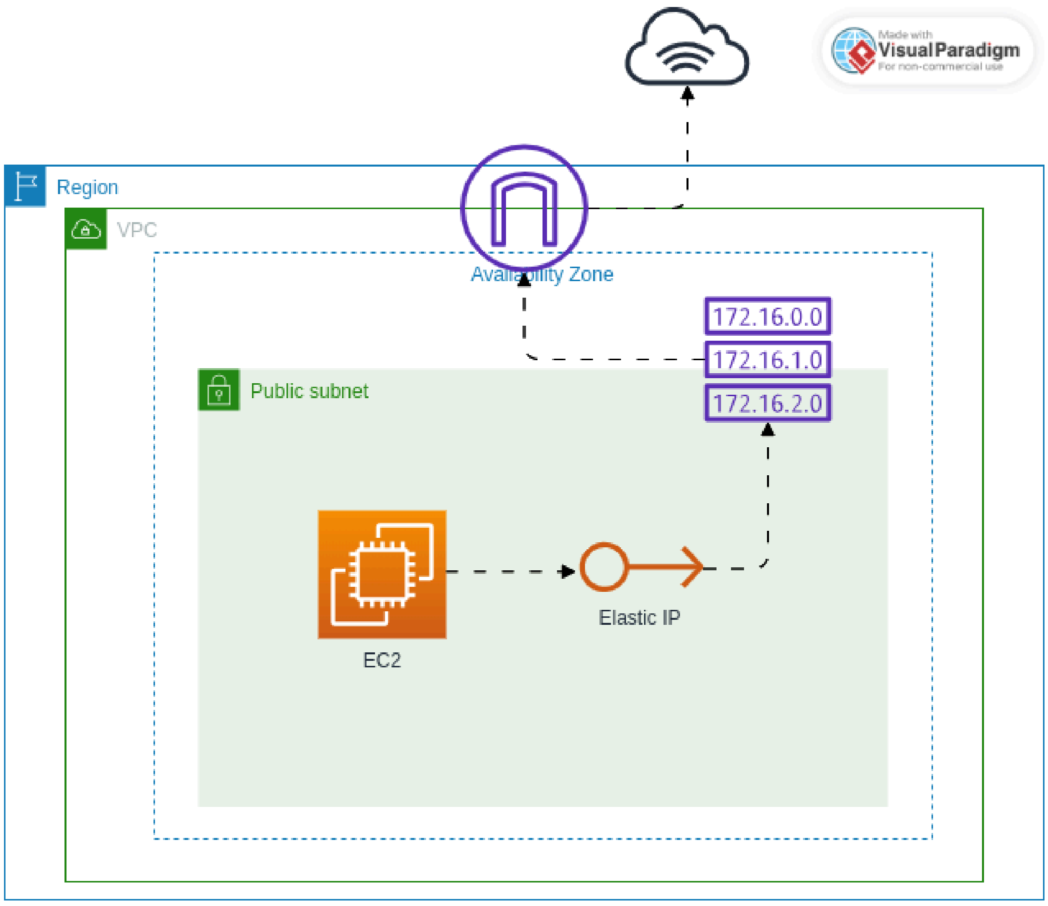
Task1

Task 1: Cloud Infrastructure & Deployment on AWS

Architecture Diagram



Steps to Deploy the Application

1. **Create an AWS Account**
2. **Create a CloudFormation Template**

   Refer to the structure from the AWS documentation: CloudFormation Template Formats.

   ```
   AWSTemplateFormatVersion: 2010-09-09
   Description: Interview Test file
   ```

```yaml
Parameters:
  EC2InstanceSizeInput:
    Description: The supported instance sizes for EC2
    Type: String
    Default: t2.micro
    AllowedValues:
      - t3.micro
      - t2.micro

Resources:
  # VPC, subnet, IGW, route table, route table to IGW
rule, security group, security group rules, EIP, NIC, EC2,
user data, SSM role, SSM policy, role assumption

  # VPC
  TestVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      Tags:
        - Key: ProjectNumber
          Value: 4
        - Key: ProjectName
          Value: interviewData

  TestIGW:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: ProjectNumber
          Value: 4
        - Key: ProjectName
          Value: interviewData

  TestAttachGateway:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref TestVPC
      InternetGatewayId: !Ref TestIGW
  # Subnet
  TestPublicSubnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref TestVPC
      AvailabilityZone: "ap-south-1a"
      CidrBlock: 10.0.0.1/24
      Tags:
        - Key: ProjectNumber
          Value: 4
        - Key: ProjectName
          Value: interviewData

  TestRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref TestVPC
      Tags:
        - Key: ProjectNumber
          Value: 4
        - Key: ProjectName
          Value: interviewData

  TestInternetPublicRoute:
    Type: AWS::EC2::Route
    DependsOn: TestIGW
    Properties:
      RouteTableId: !Ref TestRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref TestIGW

  TestRouteTableToTestSubnetAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref TestRouteTable
      SubnetId: !Ref TestPublicSubnet

  TestInstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow EC2 traffic
      VpcId: !Ref TestVPC
      SecurityGroupIngress:
        - Description: Allow SSH
          IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
        - Description: Allow HTTP
          IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - Description: Allow HTTPS
          IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
        - Description: Allow all
          IpProtocol: -1
          CidrIp: 0.0.0.0/0
      SecurityGroupEgress:
        - Description: Allow SSH
          IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
        - Description: Allow HTTP
          IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - Description: Allow HTTPS
          IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
        - Description: Allow all
          IpProtocol: -1
          CidrIp: 0
          CidrIp: 0.0.0.0/0
      Tags:
        - Key: ProjectNumber
          Value: 4
        - Key: ProjectName
          Value: interviewData

  TestEIP:
    Type: AWS::EC2::EIP
    Properties:
      Tags:
        - Key: ProjectNumber
          Value: 4
        - Key: ProjectName
          Value: interviewData

  TestNetworkInterface:
    Type: AWS::EC2::NetworkInterface
    Properties:
      Description: A External Network Interface for the
EC2
      SubnetId: !Ref TestPublicSubnet
      GroupSet:
        - !Ref TestInstanceSecurityGroup
      Tags:
        - Key: ProjectNumber
          Value: 4
        - Key: ProjectName
          Value: interviewData

  TestEIPAssociation:
    Type: AWS::EC2::EIPAssociation
    Properties:
      AllocationId: !GetAtt TestEIP.AllocationId
      NetworkInterfaceId: !Ref TestNetworkInterface
    DependsOn:
      - TestNetworkInterface
      - TestEIP

  TestEC2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      ImageId: ami-002f6e91abff6eb96 # ami-
053b12d3152c0cc71 for Ubuntu
      InstanceType: !Ref EC2InstanceSizeInput
      IamInstanceProfile: !Ref InstanceProfileOfRoleToEC2
      NetworkInterfaces:
        - Description: A Network interface made externally
with AWS EIP attached at startup as primary
          DeviceIndex: 0
          NetworkInterfaceId: !Ref TestNetworkInterface
      KeyName: myEC2KeyForInterview
      Tags:
        - Key: ProjectNumber
          Value: 4
        - Key: ProjectName
          Value: interviewData
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash
          dnf update -y
          dnf install httpd git python pip -y
          yum install docker -y
          systemctl start docker
          systemctl enable docker
          usermod -aG docker $USER
          mkdir -p /usr/local/lib/docker/cli-plugins
          curl -SL
https://github.com/docker/compose/releases/latest/download
/docker-compose-linux-x86_64 -o /usr/local/lib/docker/cli-
plugins/docker-compose
          chmod +x /usr/local/lib/docker/cli-
plugins/docker-compose
          cd /home/ec2-user
          git clone https://github.com/AryanGitHub/a-very-
simple-webapp-for-assignment.git 2> error.log
          cd a-very-simple-webapp-for-assignment
          bash bash.sh 2> error_bash.log
    DependsOn: TestEIPAssociation
```

```yaml
  SSMEC2ControlRole:
    Type: AWS::IAM::Role
    Properties:
      Description: SSM Role for Test EC2
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      ManagedPolicyArns:
        -
arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
      MaxSessionDuration: 3600
      RoleName: Test_EC2_Role
      Policies: # Adding inline policy for CloudWatch Logs
        - PolicyName: CloudWatchLogsPolicy
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - logs:CreateLogGroup
                  - logs:CreateLogStream
                  - logs:PutLogEvents
                  - logs:DescribeLogStreams
                Resource: "*"
      Tags:
        - Key: ProjectNumber
          Value: 4
        - Key: ProjectName
          Value: interviewData

  InstanceProfileOfRoleToEC2:
    Type: AWS::IAM::InstanceProfile
    Properties:
      InstanceProfileName: SSMEC2Role
      Roles:
        - !Ref SSMEC2ControlRole
```

3. **Build the App and Push It on GitHub**

   main.py

```
from fastapi import FastAPI, Form, Request
from fastapi.responses import HTMLResponse,
RedirectResponse
from fastapi.templating import Jinja2Templates
from prometheus_fastapi_instrumentator import
Instrumentator

app = FastAPI()
templates = Jinja2Templates(directory="templates")

todos = []
Instrumentator().instrument(app).expose(app)

@app.get("/", response_class=HTMLResponse)
async def read_root(request: Request):
    return templates.TemplateResponse("index.html",
{"request": request, "todos": todos})

@app.post("/add", response_class=HTMLResponse)
async def add_todo(request: Request, task: str =
Form(...)):
    todos.append(task)
    return RedirectResponse(url="/", status_code=303)
```

   templates/index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>FastAPI ToDo App</title>
</head>
<body>
```

```
    <h1>📝 ToDo List</h1>

    <form action="/add" method="post">
        <input type="text" name="task" placeholder="Enter
a task" required>
        <button type="submit">Add</button>
    </form>

    <ul>
        {% for todo in todos %}
            <li>{{ todo }}</li>
        {% else %}
            <li>No tasks yet!</li>
        {% endfor %}
    </ul>
</body>
</html>
```

Bash Script to Host the Application

```
#!/bin/bash
python -m venv .venv
source .venv/bin/activate
pip install -r ./requirements.txt
pip install prometheus-fastapi-instrumentator
uvicorn main:app --host 0.0.0.0 --port 80 --reload
```

4. **Deploy CloudFormation Template Using AWS CLI Command**

This template contains USERDATA, so it will automatically pull the app from the GitHub repository.

```
#!/bin/bash

aws cloudformation deploy --region ap-south-1 \
  --template-file ./main.yaml \
  --stack-name ec2forinterview \
  --tags madeFromCLI=yeah
anotherTagForAllStackResources=okay \
  --capabilities CAPABILITY_NAMED_IAM

# --no-execute-changeset
```

5. **AWS Configurations Used (Resource Groups, Networking, etc.)**

CloudFormation Resources Summary

VPC and Networking Resources

- **VPC**
    - Logical ID: `TestVPC`
    - Type: `AWS::EC2::VPC`
    - Properties:
        - CidrBlock: `10.0.0.0/16`
- **Internet Gateway**
    - Logical ID: `TestIGW`
    - Type: `AWS::EC2::InternetGateway`
- **VPC Gateway Attachment**
    - Logical ID: `TestAttachGateway`
    - Type: `AWS::EC2::VPCGatewayAttachment`
    - Properties:
        - VpcId: `!Ref TestVPC`
        - InternetGatewayId: `!Ref TestIGW`
- **Subnet**
    - Logical ID: `TestPublicSubnet`
    - Type: `AWS::EC2::Subnet`
    - Properties:
        - VpcId: `!Ref TestVPC`
        - AvailabilityZone: `ap-south-1a`
        - CidrBlock: `10.0.0.1/24`
- **Route Table**
    - Logical ID: `TestRouteTable`

    - Type: `AWS::EC2::RouteTable`
    - Properties:
        - VpcId: `!Ref TestVPC`
- **Route**
    - Logical ID: `TestInternetPublicRoute`
    - Type: `AWS::EC2::Route`
    - Properties:
        - RouteTableId: `!Ref TestRouteTable`
        - DestinationCidrBlock: `0.0.0.0/0`
        - GatewayId: `!Ref TestIGW`
- **Subnet Route Table Association**
    - Logical ID: `TestRouteTableToTestSubnetAssociation`
    - Type: `AWS::EC2::SubnetRouteTableAssociation`
    - Properties:
        - RouteTableId: `!Ref TestRouteTable`
        - SubnetId: `!Ref TestPublicSubnet`

Security Resources

- **Security Group**
    - Logical ID: `TestInstanceSecurityGroup`
    - Type: `AWS::EC2::SecurityGroup`
    - Properties:
        - VpcId: `!Ref TestVPC`
        - Ingress and Egress rules defined for SSH, HTTP, HTTPS, and all traffic.

EC2 Resources

9. **Elastic IP**
    - **Logical ID:** `TestEIP`
    - **Type:** `AWS::EC2::EIP`
10. **Network Interface**
    - **Logical ID:** `TestNetworkInterface`

    - **Type:** `AWS::EC2::NetworkInterface`
    - **Properties:**
        - SubnetId: `!Ref TestPublicSubnet`
        - GroupSet: `!Ref TestInstanceSecurityGroup`

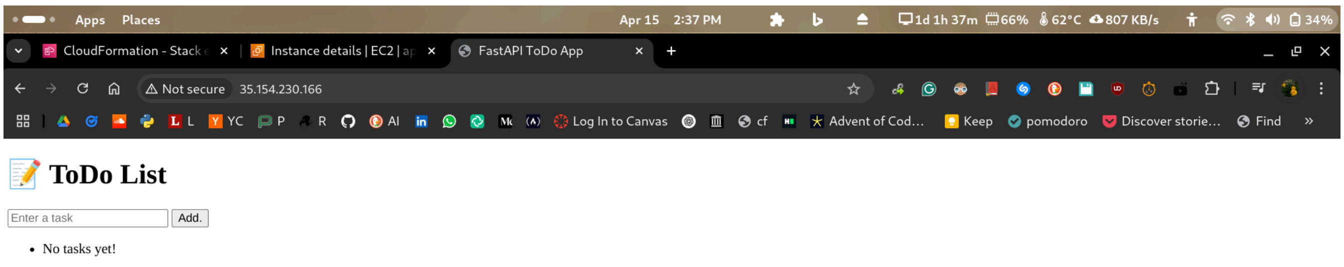
11. **EIP Association**
    - **Logical ID:** `TestEIPAssociation`
    - **Type:** `AWS::EC2::EIPAssociation`
    - **Properties:**
        - AllocationId: `!GetAtt TestEIP.AllocationId`
        - NetworkInterfaceId: `!Ref TestNetworkInterface`
12. **EC2 Instance**
    - **Logical ID:** `TestEC2Instance`
    - **Type:** `AWS::EC2::Instance`
    - **Properties:**
        - ImageId: `ami-002f6e91abff6eb96`
        - InstanceType: `!Ref EC2InstanceSizeInput`
        - NetworkInterfaces: `!Ref TestNetworkInterface`
        - UserData: Script for instance initialization.

IAM Resources

13. **IAM Role**
    - **Logical ID:** `SSMEC2ControlRole`
    - **Type:** `AWS::IAM::Role`
    - **Properties:**
        - AssumeRolePolicyDocument for EC2
        - ManagedPolicyArns: `arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore`
        - Inline policy for CloudWatch Logs.
14. **IAM Instance Profile**
    - **Logical ID:** `InstanceProfileOfRoleToEC2`
    - **Type:** `AWS::IAM::InstanceProfile`
    - **Properties:**

- Roles: `!Ref SSMEC2ControlRole`

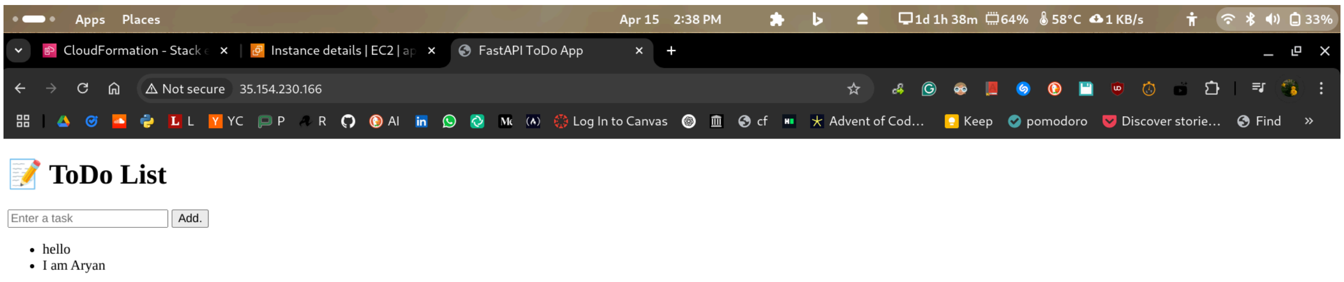
Screenshots



📝 ToDo List

Enter a task [Add.]

- No tasks yet!



📝 ToDo List

Enter a task [Add.]

- hello
- I am Aryan