**Assignment 3**

Name: Nidhish Kumar Adiyodi

Student Number: 100859177

1: Show evidence of a successfully built docker image for the application stored in private registry

Successful Build (from local):

A screenshot of a computer program

Description automatically generated

Running in local:

A screenshot of a computer

Description automatically generated

Successful code build:

A screenshot of a computer

Description automatically generated

Image pushed to Private Repo:

A screenshot of a computer

Description automatically generated

2:Show evidence of a successfully executed CICD pipeline

3: Show a screenshot of your docker file

A screenshot of a computer

Description automatically generated

4: IaC infrastructure script created and submitted

CloudFormation script for core infra:

AWSTemplateFormatVersion: "2010-09-09"

Description: CloudFormation template for FInal Test.

Parameters:

  LaunchTemplateId:

    Type: String

    Description: ID of the Launch Template

Resources:

  # VPC and Subnets

  adiyodi\_final\_test:

    Type: AWS::EC2::VPC

    Properties:

      CidrBlock: 10.0.0.0/16

  adiyodi\_final\_testPublicSubnetOne:

    Type: AWS::EC2::Subnet

    Properties:

      VpcId: !Ref adiyodi\_final\_test

      CidrBlock: 10.0.1.0/24

      AvailabilityZone: !Select [0, !GetAZs ""]

      MapPublicIpOnLaunch: true

  PublicSubnetTwo:

    Type: AWS::EC2::Subnet

    Properties:

      VpcId: !Ref adiyodi\_final\_test

      CidrBlock: 10.0.2.0/24

      AvailabilityZone: !Select [1, !GetAZs ""]

      MapPublicIpOnLaunch: true

  PrivateSubnetOne:

    Type: AWS::EC2::Subnet

    Properties:

      VpcId: !Ref adiyodi\_final\_test

      CidrBlock: 10.0.3.0/24

      AvailabilityZone: !Select [0, !GetAZs ""]

  PrivateSubnetTwo:

    Type: AWS::EC2::Subnet

    Properties:

      VpcId: !Ref adiyodi\_final\_test

      CidrBlock: 10.0.4.0/24

      AvailabilityZone: !Select [1, !GetAZs ""]

  # Internet Gateway and Route Table

  InternetGateway:

    Type: AWS::EC2::InternetGateway

  VPCGatewayAttachment:

    Type: AWS::EC2::VPCGatewayAttachment

    Properties:

      InternetGatewayId: !Ref InternetGateway

      VpcId: !Ref adiyodi\_final\_test

  PublicRouteTable:

    Type: AWS::EC2::RouteTable

    Properties:

      VpcId: !Ref adiyodi\_final\_test

  PublicRoute:

    Type: AWS::EC2::Route

    Properties:

      RouteTableId: !Ref PublicRouteTable

      DestinationCidrBlock: 0.0.0.0/0

      GatewayId: !Ref InternetGateway

  # Route Table Associations

  adiyodi\_final\_testPublicSubnetOneRouteTableAssociation:

    Type: AWS::EC2::SubnetRouteTableAssociation

    Properties:

      SubnetId: !Ref adiyodi\_final\_testPublicSubnetOne

      RouteTableId: !Ref PublicRouteTable

  PublicSubnetTwoRouteTableAssociation:

    Type: AWS::EC2::SubnetRouteTableAssociation

    Properties:

      SubnetId: !Ref PublicSubnetTwo

      RouteTableId: !Ref PublicRouteTable

  PrivateRouteTable:

    Type: AWS::EC2::RouteTable

    Properties:

      VpcId: !Ref adiyodi\_final\_test

  PrivateRoute:

    Type: AWS::EC2::Route

    Properties:

      RouteTableId: !Ref PrivateRouteTable

      DestinationCidrBlock: 0.0.0.0/0

      NatGatewayId: !Ref NatGateway

  # Nat Gateway

  NatGatewayEIP:

    Type: AWS::EC2::EIP

    Properties:

      Domain: vpc

  NatGateway:

    Type: AWS::EC2::NatGateway

    Properties:

      AllocationId: !GetAtt NatGatewayEIP.AllocationId

      SubnetId: !Ref adiyodi\_final\_testPublicSubnetOne

  # Security Groups

  adiyodi\_final\_testPublicSecurityGroup:

    Type: AWS::EC2::SecurityGroup

    Properties:

      GroupDescription: Allow HTTP traffic from the internet

      VpcId: !Ref adiyodi\_final\_test

      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 # Allowing SSH access from anywhere

  adiyodi\_final\_testPrivateSecurityGroup:

    Type: AWS::EC2::SecurityGroup

    Properties:

      GroupDescription: Allow HTTP traffic from ALB

      VpcId: !Ref adiyodi\_final\_test

      SecurityGroupIngress:

        - IpProtocol: tcp

          FromPort: 80

          ToPort: 80

          SourceSecurityGroupId: !Ref adiyodi\_final\_testPublicSecurityGroup

  # ALB, Target Group, and Listener

  adiyodi\_final\_testALB:

    Type: AWS::ElasticLoadBalancingV2::LoadBalancer

    Properties:

      Name: adiyodi\_final\_testALB

      Scheme: internet-facing

      Subnets:

        - !Ref adiyodi\_final\_testPublicSubnetOne

        - !Ref PublicSubnetTwo

      SecurityGroups:

        - !Ref adiyodi\_final\_testPublicSecurityGroup

      Type: application

  adiyodi\_final\_testTargetGroup:

    Type: AWS::ElasticLoadBalancingV2::TargetGroup

    Properties:

      Name: adiyodi\_final\_testTargetGroup

      Port: 80

      Protocol: HTTP

      VpcId: !Ref adiyodi\_final\_test

      TargetType: instance

  adiyodi\_final\_testListener:

    Type: AWS::ElasticLoadBalancingV2::Listener

    Properties:

      DefaultActions:

        - Type: forward

          TargetGroupArn: !Ref adiyodi\_final\_testTargetGroup

      LoadBalancerArn: !Ref adiyodi\_final\_testALB

      Port: 80

      Protocol: HTTP

  # EC2 Instance

  goldenEC2Instance:

    Type: AWS::EC2::Instance

    Properties:

      InstanceType: t2.micro

      ImageId: ami-019f9b3318b7155c5

      KeyName: vokey

      SecurityGroupIds:

        - !Ref adiyodi\_final\_testPublicSecurityGroup

      SubnetId: !Ref adiyodi\_final\_testPublicSubnetOne

      UserData:

        Fn::Base64: !Sub |

          #!/bin/bash

          sudo yum update -y

          sudo yum install -y httpd php

          sudo systemctl start httpd

          sudo systemctl enable httpd

          sudo echo "<?php phpinfo(); ?>" > /var/www/html/index.php

          # Allow SSH access

          sudo yum install -y openssh-server

          sudo systemctl start sshd

          sudo systemctl enable sshd

  # Launch Template and Auto Scaling Group

  adiyodi\_final\_testLaunchTemplate:

    Type: AWS::EC2::LaunchTemplate

    Properties:

      LaunchTemplateName: ApacheLaunchTemplate

      LaunchTemplateData:

        ImageId: ami-019f9b3318b7155c5

        InstanceType: t2.micro

        KeyName: vokey

        SecurityGroupIds:

          - !Ref adiyodi\_final\_testPrivateSecurityGroup

        UserData:

          Fn::Base64: !Sub |

            #!/bin/bash

            sudo yum update -y

            sudo yum install -y httpd php

            sudo systemctl start httpd

            sudo systemctl enable httpd

            sudo echo "<?php phpinfo(); ?>" > /var/www/html/index.php

            # Allow SSH access

            sudo yum install -y openssh-server

            sudo systemctl start sshd

            sudo systemctl enable sshd

  adiyodi\_final\_testASG:

    Type: AWS::AutoScaling::AutoScalingGroup

    Properties:

      AutoScalingGroupName: adiyodi\_final\_testASG

      MinSize: "1"

      MaxSize: "3"

      DesiredCapacity: "2"

      LaunchTemplate:

        LaunchTemplateId: !Ref adiyodi\_final\_testLaunchTemplate

        Version: !GetAtt adiyodi\_final\_testLaunchTemplate.LatestVersionNumber

      TargetGroupARNs:

        - !Ref adiyodi\_final\_testTargetGroup

      VPCZoneIdentifier:

        - !Ref PrivateSubnetOne

        - !Ref PrivateSubnetTwo

Outputs:

  adiyodi\_final\_testPublicSubnetOneId:

    Description: Public subnet one ID

    Value: !Ref adiyodi\_final\_testPublicSubnetOne

  PublicSubnetTwoId:

    Description: Public subnet two ID

    Value: !Ref PublicSubnetTwo

  PrivateSubnetOneId:

    Description: Private subnet one ID

    Value: !Ref PrivateSubnetOne

  ALBARN:

    Description: The ARN of the ALB

    Value: !Ref adiyodi\_final\_testALB

  TargetGroupARN:

    Description: The ARN of the Target Group

    Value: !Ref adiyodi\_final\_testTargetGroup

  AutoScalingGroupName:

    Description: The name of the Auto Scaling Group

    Value: !Ref adiyodi\_final\_testASG

5:Show evidence of application running behind ELB

Load Balancer:

A screenshot of a computer

Description automatically generated

Target Groups:

A screenshot of a computer

Description automatically generated

Application accessible through ALB:

A screenshot of a computer

Description automatically generated

6:Show evidence of each route is accessible for api

Main Route:

A screenshot of a computer

Description automatically generated

Host Route:

A white screen with black text

Description automatically generated

Ip route

A screenshot of a computer

Description automatically generated

7:show evidence of at least 2 containers running API

Service:

A screenshot of a computer

Description automatically generated

2 Tasks Running:

A screenshot of a computer

Description automatically generated

8: Show evidence of api application running in private subnet

Task without Public ip:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

9:Show evidence of proper security policy

ELB sg(allows 80 from internet):

A screenshot of a computer

Description automatically generated

ECS SG (allows only 5000 from elb sg)

A screenshot of a computer

Description automatically generated

Custom Resources:

VPC:

A screenshot of a computer

Description automatically generated

Subnets:

A screenshot of a computer

Description automatically generated

10:show evidence of autoscaling policies in place

Scale in and out based on cpu usage:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated