**Cloud Formation template to create infrastructure**

AWSTemplateFormatVersion: '2010-09-09'

Description: 'cloud formation to create a vpc with 2 public and 2 private subnet with 1 EC2 into private and 1 into public subnet'

Parameters:

  InstanceType:

    Type: String

    Description: t2 micro

    Default: t2.micro

  KeyName:

    Description: SSH Keypair to login to the instance

    Type: AWS::EC2::KeyPair::KeyName

    Default: LinuxKey

Resources:

      chatVPC:

        Type: 'AWS::EC2::VPC'

        Properties:

         CidrBlock: [172.31.0.0/16](http://172.31.0.0/16)

         Tags:

            - Key: Name

              Value: CF-newVpc

      PublicSubnet1:

        Type: 'AWS::EC2::Subnet'

        Properties:

          Tags:

            - Key: Name

              Value: CF-PublicSubnet1

          VpcId: !Ref chatVPC

          AvailabilityZone: us-east-2a

          CidrBlock: [172.31.1.0/24](http://172.31.1.0/24)

          MapPublicIpOnLaunch: true

      PublicSubnet2:

        Type: 'AWS::EC2::Subnet'

        Properties:

          Tags:

            - Key: Name

              Value: CF-PublicSubnet2

          VpcId: !Ref chatVPC

          AvailabilityZone: us-east-2b

          CidrBlock: [172.31.2.0/24](http://172.31.2.0/24)

          MapPublicIpOnLaunch: true

      PrivateSubnet1:

        Type: 'AWS::EC2::Subnet'

        Properties:

          Tags:

            - Key: Name

              Value: CF-PrivateSubnet1

          VpcId: !Ref chatVPC

          AvailabilityZone: us-east-2a

          CidrBlock: [172.31.3.0/24](http://172.31.3.0/24)

          MapPublicIpOnLaunch: false

      PrivateSubnet2:

        Type: 'AWS::EC2::Subnet'

        Properties:

          Tags:

            - Key: Name

              Value: CF-PrivateSubnet2

          VpcId: !Ref chatVPC

          AvailabilityZone: us-east-2b

          CidrBlock: [172.31.4.0/24](http://172.31.4.0/24)

          MapPublicIpOnLaunch: false

      InternetGateway:

        Type: 'AWS::EC2::InternetGateway'

        Properties:

          Tags:

            - Key: Name

              Value: CloudFormation\_IG

            - Key: Network

              Value: Public

      GatewayToInternet:

        Type: 'AWS::EC2::VPCGatewayAttachment'

        Properties:

          VpcId: !Ref chatVPC

          InternetGatewayId: !Ref InternetGateway

      PublicRouteTable:

        Type: 'AWS::EC2::RouteTable'

        Properties:

          VpcId: !Ref chatVPC

          Tags:

            - Key: Network

              Value: Public

      PublicRoute:

        Type: 'AWS::EC2::Route'

        DependsOn: GatewayToInternet

        Properties:

          RouteTableId: !Ref PublicRouteTable

          DestinationCidrBlock: [0.0.0.0/0](http://0.0.0.0/0)

          GatewayId: !Ref InternetGateway

      PublicSubnet1RouteTableAssociation:

        Type: 'AWS::EC2::SubnetRouteTableAssociation'

        Properties:

          SubnetId: !Ref PublicSubnet1

          RouteTableId: !Ref PublicRouteTable

      PublicSubnet2RouteTableAssociation:

        Type: 'AWS::EC2::SubnetRouteTableAssociation'

        Properties:

          SubnetId: !Ref PublicSubnet2

          RouteTableId: !Ref PublicRouteTable

      NatGateway:

        Type: "AWS::EC2::NatGateway"

        DependsOn: NatPublicIP

        Properties:

          AllocationId: !GetAtt NatPublicIP.AllocationId

          SubnetId: !Ref PublicSubnet1

      NatPublicIP:

        Type: "AWS::EC2::EIP"

        DependsOn: chatVPC

        Properties:

          Domain: vpc

      PrivateRouteTable:

        Type: 'AWS::EC2::RouteTable'

        Properties:

          VpcId: !Ref chatVPC

          Tags:

            - Key: Network

              Value: Private

      PrivateRoute:

        Type: 'AWS::EC2::Route'

        Properties:

          RouteTableId: !Ref PrivateRouteTable

          DestinationCidrBlock: [0.0.0.0/0](http://0.0.0.0/0)

          NatGatewayId: !Ref NatGateway

      PrivateSubnet1RouteTableAssociation:

        Type: 'AWS::EC2::SubnetRouteTableAssociation'

        Properties:

          SubnetId: !Ref PrivateSubnet1

          RouteTableId: !Ref PrivateRouteTable

      PrivateSubnet2RouteTableAssociation:

        Type: 'AWS::EC2::SubnetRouteTableAssociation'

        Properties:

          SubnetId: !Ref PrivateSubnet2

          RouteTableId: !Ref PrivateRouteTable

      FrontEnd:

        Type: 'AWS::EC2::Instance'

        Properties:

           ImageId: ami-097dfc609cdd75f29

           InstanceType: !Ref InstanceType

           KeyName: !Ref KeyName

           Tags:

             - Key: Name

               Value: FrontEnd\_CF

           NetworkInterfaces:

           - DeviceIndex: 0

             AssociatePublicIpAddress: true

             SubnetId: !Ref PublicSubnet1

             GroupSet:

             - !Ref FrontEndSecurityGroup

      BackEnd:

        Type: 'AWS::EC2::Instance'

        Properties:

           ImageId: ami-0b4698249ff860cd8

           InstanceType: !Ref InstanceType

           KeyName: !Ref KeyName

           Tags:

             - Key: Name

               Value: BackEnd\_CF

           NetworkInterfaces:

           - DeviceIndex: 0

             AssociatePublicIpAddress: false

             SubnetId: !Ref PrivateSubnet1

             GroupSet:

             - !Ref BackendSecurityGroup

      BackendSecurityGroup:

        Type: 'AWS::EC2::SecurityGroup'

        Properties:

           VpcId: !Ref chatVPC

           GroupDescription: SG to allow SSH access via port 22,80,8000

           SecurityGroupIngress:

           - IpProtocol: tcp

             FromPort: '22'

             ToPort: '22'

             CidrIp: '[0.0.0.0/0](http://0.0.0.0/0)'

           - IpProtocol: tcp

             FromPort: '80'

             ToPort: '80'

             CidrIp: '[0.0.0.0/0](http://0.0.0.0/0)'

           Tags:

           - Key: Name

             Value: BackEnd\_SG

      FrontEndSecurityGroup:

        Type: 'AWS::EC2::SecurityGroup'

        Properties:

          VpcId: !Ref chatVPC

          GroupDescription: SG to allow SSH access via port 22,80

          SecurityGroupIngress:

          - IpProtocol: tcp

            FromPort: '22'

            ToPort: '22'

            CidrIp: '[0.0.0.0/0](http://0.0.0.0/0)'

          - IpProtocol: tcp

            FromPort: '80'

            ToPort: '80'

            CidrIp: '[0.0.0.0/0](http://0.0.0.0/0)'

          - IpProtocol: tcp

            FromPort: '8000'

            ToPort: '8000'

            CidrIp: '[0.0.0.0/0](http://0.0.0.0/0)'

          - IpProtocol: tcp

            FromPort: '3306'

            ToPort: '3306'

            CidrIp: '[0.0.0.0/0](http://0.0.0.0/0)'

          Tags:

           - Key: Name

             Value: FrontEnd\_SG

      BackEndLaunchConfig:

        Type: AWS::AutoScaling::LaunchConfiguration

        Properties:

          ImageId: ami-0b4698249ff860cd8

          InstanceType: !Ref InstanceType

          KeyName: !Ref KeyName

          SecurityGroups:

            - !Ref BackendSecurityGroup

      BackEndASG:

        Type: AWS::AutoScaling::AutoScalingGroup

        Properties:

          VPCZoneIdentifier:

            - !Ref PrivateSubnet1

            - !Ref PrivateSubnet2

          DesiredCapacity: 1

          LaunchConfigurationName: !Ref BackEndLaunchConfig

          TargetGroupARNs:

            - !Ref BackTargetGroup

          MaxSize: 2

          MinSize: 1

          Tags:

            - Key: Name

              Value: BackASG

              PropagateAtLaunch: True

        UpdatePolicy:

          AutoScalingReplacingUpdate:

            WillReplace: True

      AppScaleUpPolicy:

        Type: AWS::AutoScaling::ScalingPolicy

        Properties:

          AdjustmentType: ChangeInCapacity

          AutoScalingGroupName: !Ref BackEndASG

          Cooldown: '60'

          ScalingAdjustment: 1

      AppScaleDownPolicy:

        Type: AWS::AutoScaling::ScalingPolicy

        Properties:

          AdjustmentType: ChangeInCapacity

          AutoScalingGroupName: !Ref BackEndASG

          Cooldown: '300'

          ScalingAdjustment: -1

      BackALBSG:

        Type: 'AWS::EC2::SecurityGroup'

        Properties:

          VpcId: !Ref chatVPC

          GroupDescription: Enable HTTP access

          SecurityGroupIngress:

            - IpProtocol: tcp

              FromPort: '22'

              ToPort: '22'

              CidrIp: '[0.0.0.0/0](http://0.0.0.0/0)'

            - IpProtocol: tcp

              FromPort: '80'

              ToPort: '80'

              CidrIp: '[0.0.0.0/0](http://0.0.0.0/0)'

            - IpProtocol: tcp

              FromPort: '8000'

              ToPort: '8000'

              CidrIp: '[0.0.0.0/0](http://0.0.0.0/0)'

            - IpProtocol: tcp

              FromPort: '3306'

              ToPort: '3306'

              CidrIp: '[0.0.0.0/0](http://0.0.0.0/0)'

          Tags:

            - Key: Name

              Value: BackALBSG

      BackEndALB:

        Type: AWS::ElasticLoadBalancingV2::LoadBalancer

        Properties:

          IpAddressType: ipv4

          Scheme: internal

          SecurityGroups:

            - !Ref BackALBSG

          Subnets:

            - !Ref PrivateSubnet1

            - !Ref PrivateSubnet2

          Tags:

            - Key: Name

              Value: BackALB

          Type: application

      Listener:

        Type: AWS::ElasticLoadBalancingV2::Listener

        Properties:

          DefaultActions:

            - Type: forward

              TargetGroupArn:

                Ref: BackTargetGroup

          LoadBalancerArn:

            Ref: BackEndALB

          Port: '8000'

          Protocol: HTTP

      BackTargetGroup:

        Type: AWS::ElasticLoadBalancingV2::TargetGroup

        Properties:

          Port: '8000'

          Protocol: 'HTTP'

          VpcId: !Ref chatVPC

          HealthCheckIntervalSeconds: 30

          HealthCheckPath: /

          HealthCheckPort: 8000

          HealthCheckProtocol: HTTP

          HealthCheckTimeoutSeconds: 5

          HealthyThresholdCount: 2

          Tags:

            - Key: Name

              Value: BackendTG

      FrontEndLaunchConfig:

        Type: AWS::AutoScaling::LaunchConfiguration

        Properties:

          ImageId: ami-097dfc609cdd75f29

          InstanceType: !Ref InstanceType

          KeyName: !Ref KeyName

          SecurityGroups:

            - !Ref FrontEndSecurityGroup

      FrontEndASG:

        Type: AWS::AutoScaling::AutoScalingGroup

        Properties:

          VPCZoneIdentifier:

            - !Ref PublicSubnet1

            - !Ref PublicSubnet2

          DesiredCapacity: 1

          LaunchConfigurationName: !Ref FrontEndLaunchConfig

          TargetGroupARNs:

            - !Ref FrontEndTargetGroup

          MaxSize: 2

          MinSize: 1

          Tags:

            - Key: Name

              Value: FrontEnd-ASG

              PropagateAtLaunch: True

      FrontEndCPUPolicy:

        Type: AWS::AutoScaling::ScalingPolicy

        Properties:

          AutoScalingGroupName: !Ref FrontEndASG

          PolicyType: TargetTrackingScaling

          TargetTrackingConfiguration:

            PredefinedMetricSpecification:

              PredefinedMetricType: ASGAverageCPUUtilization

            TargetValue: 80

      FrontEndLBSG:

        Type: 'AWS::EC2::SecurityGroup'

        Properties:

          VpcId: !Ref chatVPC

          GroupDescription: Enable HTTP access

          SecurityGroupIngress:

            - IpProtocol: 'tcp'

              FromPort: '80'

              ToPort: '80'

              CidrIp: [0.0.0.0/0](http://0.0.0.0/0)

            - IpProtocol: tcp

              FromPort: 8000

              ToPort: 8000

              CidrIp: "[0.0.0.0/0](http://0.0.0.0/0)"

            - IpProtocol: tcp

              FromPort: '22'

              ToPort: '22'

              CidrIp: "[0.0.0.0/0](http://0.0.0.0/0)"

          Tags:

            - Key: Name

              Value: FrontEndLBSG

      FrontEndLoadBalancer:

        Type: AWS::ElasticLoadBalancingV2::LoadBalancer

        Properties:

          IpAddressType: ipv4

          Scheme: internet-facing

          SecurityGroups:

            - !Ref FrontEndLBSG

          Subnets:

            - !Ref PublicSubnet1

            - !Ref PublicSubnet2

          Tags:

            - Key: Name

              Value: FrontEnd-LB

          Type: application

      FrontEndLoadBalancerListener:

        Type: "AWS::ElasticLoadBalancingV2::Listener"

        Properties:

          DefaultActions:

            - TargetGroupArn: !Ref FrontEndTargetGroup

              Type: forward

          LoadBalancerArn: !Ref FrontEndLoadBalancer

          Port: 80

          Protocol: HTTP

      FrontEndTargetGroup:

        Type: "AWS::ElasticLoadBalancingV2::TargetGroup"

        Properties:

          Name: FrontEndTargetGroup

          Port: 80

          Protocol: HTTP

          VpcId: !Ref chatVPC