**Tasks To Be Performed:**

1. Web tier: Launch an instance in a public subnet and that instance should

allow HTTP and SSH from the internet.

2. Application tier: Launch an instance in a private subnet of the web tier and

it should allow only SSH from the public subnet of Web Tier-3.

3. DB tier: Launch an RDS MYSQL instance in a private subnet and it should

allow connection on port 3306 only from the private subnet of Application

Tier-4.

4. Setup a Route 53 hosted zone and direct traffic to the EC2 instance.

You have been also asked to propose a solution so that:

1. Development team can test their code without having to involve the system

admins and can invest their time in testing the code rather than

provisioning, configuring and updating the resources needed to test the

code.

2. Make sure when the development team deletes the stack, RDS DB

instances should not be deleted.

Soln🡪

So lets first understand the architecture.

a)we need a vpc.

b)in vpc we need too subnets one should be private and one public.

c) for public subnet we need a internet gateway.

d)we need to add internet gateway in route table for public subnet.

e) so we need two route table also one for private , one for public.

f)

so let’s go 🡪

first we will create a vpc.

So yaml for vpc.

Resources:

  VPC:

    Type: AWS::EC2::VPC

    Properties:

      CidrBlock: 10.10.0.0/16

      EnableDnsSupport: true

      EnableDnsHostnames: true

      InstanceTenancy: default

      Tags:

        - Key: Name

          Value: VPCAssessment

What does enabling DNS do?

DNS translates domain names to IP addresses so browsers can load Internet resources. Each device connected to the Internet has a unique IP address which other machines use to find the device. DNS servers eliminate the need for humans to memorize IP addresses such as 192.168.So it will provide dns to public instances.

What is instance tenancy?

* **Instance tenancy default**

The default tenancy is shared, which means that multiple AWS accounts can share the same physical hardware.

* **Dedicated server**

A dedicated server is a high-performance computer system that is leased to a single customer or organization. The customer has full control of the physical server's resources.

So now next we need a internet gateway so our public instance can connect to internet.

 InternetGateway:

    Type: AWS::EC2::InternetGateway

    Properties:

      Tags:

        - Key: Name

          Value: InternetGatewayAssessment

  VPCGatewayAttachment:

    Type: AWS::EC2::VPCGatewayAttachment

    Properties:

      VpcId: !Ref VPC

      InternetGatewayId: !Ref InternetGateway

So we have created a internet gateway and a attachment to vpc.

Now we will create the subnets.

So first we will go with public subnet , so what we need a subnet to be public.

We need internet gateway to be connect in route table and in route records.

So our yaml will be like this.

1. Public subnet

SubnetA:

    Type: AWS::EC2::Subnet

    Properties:

      AvailabilityZone: !Select [ 0, !GetAZs ]

      VpcId: !Ref VPC

      CidrBlock: 10.10.1.0/24

      MapPublicIpOnLaunch: true

      Tags:

        - Key: Name

          Value: PublicSubnetAssessment

1. Public route table

PublicRouteTable:

    Type: AWS::EC2::RouteTable

    Properties:

      VpcId: !Ref VPC

      Tags:

        - Key: Name

          Value: RouteTablePublicSubnet

1. Now we will create a route.

PublicInternetRoute:

    Type: AWS::EC2::Route

    DependsOn: VPCGatewayAttachment

    Properties:

      DestinationCidrBlock: 0.0.0.0/0

      GatewayId: !Ref InternetGateway

      RouteTableId: !Ref PublicRouteTable

1. Now we will attach the route table to our subnet.

SubnetARouteTableAssociation:

    Type: AWS::EC2::SubnetRouteTableAssociation

    Properties:

      RouteTableId: !Ref PublicRouteTable

      SubnetId: !Ref SubnetA

Now we need a private subnet and according to the case study we want our private subnet to be connect via internet.

a)so we will create a subnet.

b) we will create a NAT gateway , attach it to public subnet.

c) create a route table , route and in the route we will attach NAT gateway.

d) than associate that table to private subnet.

So yam file look something like this.

#Private Subnet

  SubnetB:

    Type: AWS::EC2::Subnet

    Properties:

      AvailabilityZone:  !Select [ 1, !GetAZs ]

      VpcId: !Ref VPC

      CidrBlock: 10.10.2.0/24

      MapPublicIpOnLaunch: false

      Tags:

        - Key: Name

          Value: PrivateSubnetAssessment

  # A NAT Gateway:

  NATGateway:

   Type: AWS::EC2::NatGateway

   Properties:

     AllocationId: !GetAtt ElasticIPAddress.AllocationId

     SubnetId: !Ref SubnetA

     Tags:

     - Key: Name

       Value: NatGetwayAssessment

  ElasticIPAddress:

    Type: AWS::EC2::EIP

    Properties:

      Domain: VPC

  RouteTablePrivate:

    Type: AWS::EC2::RouteTable

    Properties:

      VpcId: !Ref VPC

      Tags:

        - Key: Name

          Value: RouteTablePrivateSubnet

  NATRoute:

    DependsOn: NATGateway

    Type: AWS::EC2::Route

    Properties:

      RouteTableId: !Ref RouteTablePrivate

      DestinationCidrBlock: 0.0.0.0/0

      NatGatewayId: !Ref NATGateway

  SubnetBRouteTableAssociationPrivate:

    Type: AWS::EC2::SubnetRouteTableAssociation

    Properties:

      RouteTableId: !Ref RouteTablePrivate

      SubnetId: !Ref SubnetB

Here we have attach a elastic ip to NAT gateway so we have created it as well.

Now we will create security group .

We need three groups .

a)public instance.

InstanceSecurityGroup:

    Type: AWS::EC2::SecurityGroup

    Properties:

      GroupName: "Internet Group"

      GroupDescription: "SSH and web traffic in, all traffic out."

      VpcId: !Ref VPC

      SecurityGroupIngress:

        - IpProtocol: tcp

          FromPort: '22'

          ToPort: '22'

          CidrIp:  0.0.0.0/0

        - IpProtocol: tcp

          FromPort: '80'

          ToPort: '80'

          CidrIp:  0.0.0.0/0

      SecurityGroupEgress:

        - IpProtocol: -1 // -1 is for all

          CidrIp: 0.0.0.0/0

b) private instance security group.

InstanceSecurityGroupPrivate:

    Type: AWS::EC2::SecurityGroup

    Properties:

      GroupName: "Security Group Private"

      GroupDescription: "SSH from the Public Subnet"

      VpcId: !Ref VPC

      SecurityGroupIngress:

        - IpProtocol: tcp

          FromPort: '22'

          ToPort: '22'

          CidrIp:  10.10.1.0/24

      SecurityGroupEgress:

        - IpProtocol: -1

          CidrIp: 0.0.0.0/0

Here we have only allow ssh .

c) database security group.

InstanceSecurityGroupDataBase:

    Type: "AWS::EC2::SecurityGroup"

    Properties:

      GroupDescription: "Database instances security group"

      VpcId: !Ref VPC

      SecurityGroupIngress:

          - IpProtocol: tcp

            CidrIp: 10.10.2.0/24

            FromPort: 3306

            ToPort: 3306

      SecurityGroupEgress:

          - IpProtocol: -1

            CidrIp: 0.0.0.0/0

// in this we have given range of only private subnet cidr.

Now we will create our db instance.

So what we will require first for our db instance.

Security group we have already created , next is subnet group.

For rds we need subnetgroup to launch our db instance.

RDSDBSubnetGroup:

    Type: "AWS::RDS::DBSubnetGroup"

    Properties:

        DBSubnetGroupDescription: "Subnet Group for mySQL database"

        DBSubnetGroupName: !Sub "${AWS::Region}-aws-database-subnet-group14"

        SubnetIds:

          - !Ref SubnetA

          - !Ref SubnetB

        Tags:

          - Key: Name

            Value: DBSubnetGroup

We have select both of our subnets here.

Now we will create our db instance.

RDSDBInstance:

        Type: AWS::RDS::DBInstance

        Properties:

            DBInstanceIdentifier: DBAssessment12

            AllocatedStorage: 20

            DBInstanceClass: db.t3.micro

            Engine: "MYSQL"

            MasterUsername: !Ref MasterUsername

            MasterUserPassword: !Ref MasterUserPassword

            MultiAZ: false

            EngineVersion: 8.0.28

            AutoMinorVersionUpgrade: true

            PubliclyAccessible: false

            StorageType: gp2

            Port: 3306

            StorageEncrypted: false

            CopyTagsToSnapshot: true

            EnableIAMDatabaseAuthentication: false

            DeletionProtection: true // we have make this so no one can delete it.

            DBSubnetGroupName: !Ref RDSDBSubnetGroup

            VPCSecurityGroups:

              - !Ref InstanceSecurityGroupDataBase

            MaxAllocatedStorage: 1000

            Tags:

              - Key: Name

                Value: DBAssessment

              - Key: createdBy

                Value: Igor Silva

              - Key: Project

                Value: AssessmentModule7

              - Key: Environment

                Value: Prod

Now we need our public and private instances.

LinuxPublic:

    Type: 'AWS::EC2::Instance'

    Properties:

      SubnetId: !Ref SubnetA

      ImageId: !Ref AMI

      InstanceType: !Ref InstanceTypeParameter

      KeyName: !Ref Key

      SecurityGroupIds:

        - Ref: InstanceSecurityGroup

      Tags:

        - Key: Name

          Value: LinuxPublic

  LinuxPrivate:

    Type: 'AWS::EC2::Instance'

    Properties:

      SubnetId: !Ref SubnetB

      ImageId: !Ref AMI

      InstanceType: !Ref InstanceTypeParameter

      KeyName: !Ref Key

      SecurityGroupIds:

        - Ref: InstanceSecurityGroupPrivate

      Tags:

      - Key: Name

        Value: LinuxPrivate

Now we will create our hosted zone for our route 53.

HostedZone:

    Type: AWS::Route53::HostedZone

    Properties:

      HostedZoneConfig:

        Comment: ''

      Name: newpracticedomain.ml

  MyDNSRecord:

    Type: AWS::Route53::RecordSet

    Properties:

      HostedZoneId: !Ref HostedZone

      Name: www.newpracticedomain.ml.

      Type: A

      TTL: 300

      ResourceRecords:

      - !GetAtt LinuxPublic.PublicIp

So now our cloud formation configuration is done

We are going to use some parameters also.

Parameters:

  InstanceTypeParameter:

    Type: String

    Default: t2.micro

    Description: Enter instance size. Default is t2.micro.

  AMI:

    Type: String

    Default: ami-0bcda2433f3dabc41

    Description: The Ubuntu AMI to use.

  Key:

    Type: AWS::EC2::KeyPair::KeyName

    Description: Select from Existing Keys.

  MasterUsername:

    Type: String

    Description: The username for the database.

  MasterUserPassword:

    Type: String

    Description: The password for the database.

    "NoEcho": true

Some values we will provide at run time.

Now for developers to be easy we will give some outputs also.

Outputs:

  PublicIp:

   Description: Server's PublicIp Address

   Value:

     Fn::GetAtt:

       - LinuxPublic

       - PublicIp

  HostedZoneID:

    Description: The ID of the Hosted Zone.

    Value:

      Ref: HostedZone

Now lets create it .

Search cloud formation 🡪 create stack.

A screenshot of a computer

Description automatically generated

Upload your template file here.

Click on next.

A screenshot of a computer

Description automatically generated

Give it a name.

Now put and check our parameter values.

A screenshot of a computer

Description automatically generated

We have select our existing ami id and key.

Click on next.

A screenshot of a computer

Description automatically generated

Put other configurations a s default an create it.

Now we can see our stack getting created.

A screenshot of a computer

Description automatically generated

We have to change version of mysql in our template.

A screenshot of a computer error message

Description automatically generated

Lets change it and launch again.

We can see all our events getting created.

A screenshot of a computer

Description automatically generated

After completion we can see outputs.

A screenshot of a computer

Description automatically generated

Now let’s try to delete and check will our deletion protection on rds works.

We can see this error.

A white background with red and blue text

Description automatically generated