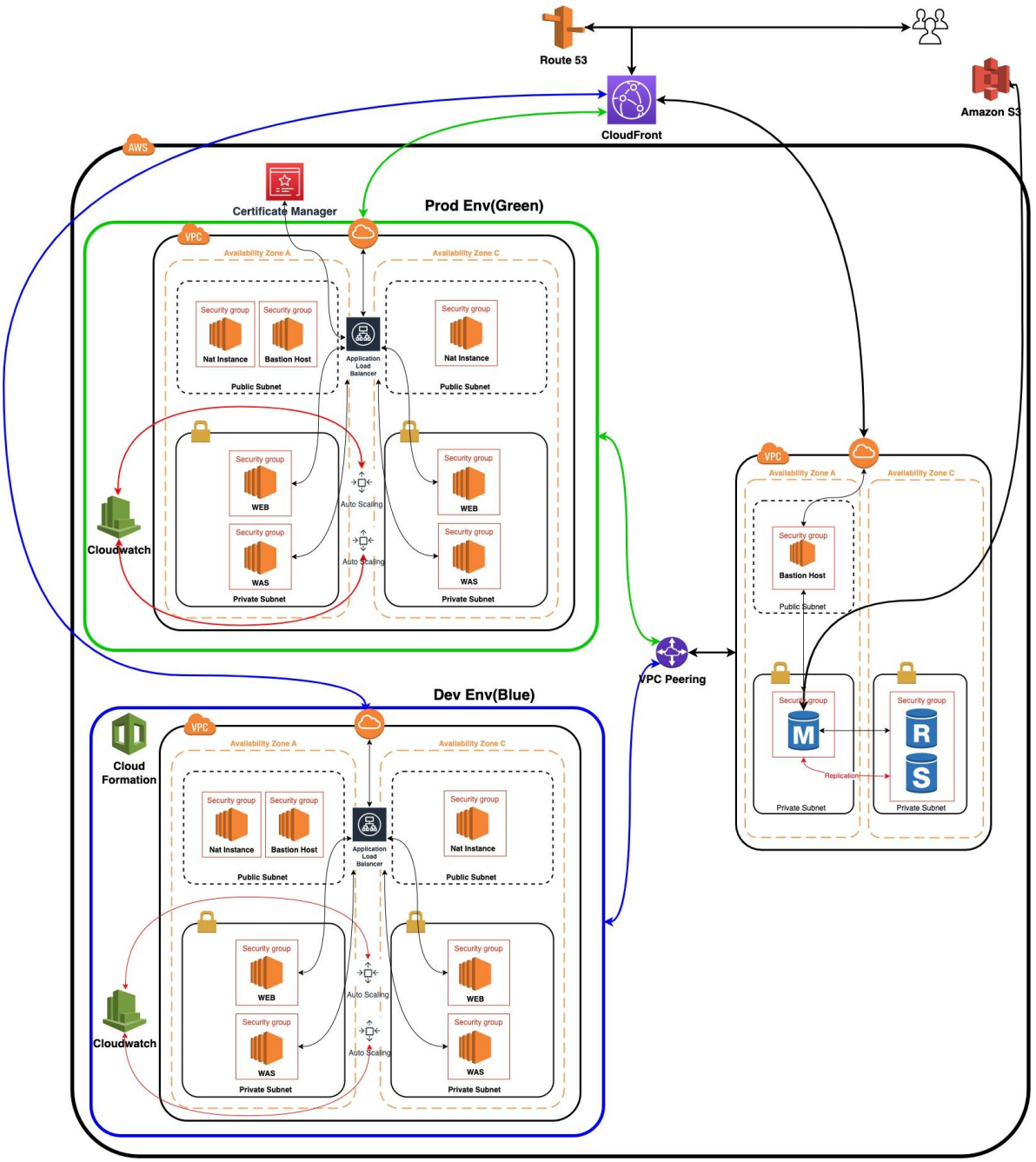
Project: Deploying 3 tier APM stacked Web Application via Cloudformation
*(Cloudformation*을 활용한 *APM* 기반 *3*티어 웹앱 구축*)*

<Agenda>

- ➔ *-Cloudfront*를 사용해 정적 컨텐츠 트래픽 완화
- ➔ *-ELB*에 *SSM*을 적용시켜 *HTTPS* 프로토콜 사용
- ➔ *-*정적 컨텐츠를 담당하는 *web*서버 및 *was* 서버를 분리함으로 각 서버에 대한 불필요한 트래픽 과부하 방지
- ➔ *-VPC Peering*을 사용해 *Blue/Green Deployment* 구축*(Cloudformation*으로 해당 인프라 *IaC*화*)* *Route53*의 *weighted routing* 정책을 사용해 *Canary Deployment*도 적용
- ➔ 별도의 *Bastion Host*를 생성해 *was* 서버나 *db* 서버에 접속할 때 보안 향상
- ➔ *RDS*는 *Read-only DB*를 별도로 생성 후 *Read* 트래픽은 *Read-only DB*로 보낸 후 *Master DB*의 부하 분산, 별도의 *Standby DB*도 생성해 *Replication* 진행 후 해당 데이터는 *S3*로 전달




```
Parameters:
 3tierAPM:
   Description: Name of an existing EC2 KeyPair to enable SSH access to the
instances. Linked to AWS Parameter
   Type: AWS::EC2::KeyPair::KeyName
   ConstraintDescription: must be the name of an existing EC2 KeyPair.
 DBNameP:
   Type: String
 MUser:
   Type: String
 MPass:
   Type: String
```

```yaml
Resources:
 VPC:
   Type: AWS::EC2::VPC
   Properties:
    CidrBlock: 10.0.0.0/16
    EnableDnsSupport: true
    EnableDnsHostnames: true
    Tags:
       - Key: Name
         Value: VPC

 VPCIGW:
   Type: AWS::EC2::InternetGateway
   Properties:
     Tags:
       - Key: Name
         Value: VPC-IGW

 VPCIGWAttachment:
   Type: AWS::EC2::VPCGatewayAttachment
   Properties:
     InternetGatewayId: !Ref VPCIGW
     VpcId: !Ref VPC

 VPCPublicRT:
   Type: AWS::EC2::RouteTable
   Properties:
     VpcId: !Ref VPC
     Tags:
       - Key: Name
         Value: VPC-Public-RT

 DefaultPublicRoute:
   Type: AWS::EC2::Route
   DependsOn: VPCIGWAttachment
   Properties:
     RouteTableId: !Ref VPCPublicRT
     DestinationCidrBlock: 0.0.0.0/0
     GatewayId: !Ref VPCIGW

 VPCPrivateRT:
   Type: AWS::EC2::RouteTable
   Properties:
     VpcId: !Ref VPC
```

```
    Tags:
      - Key: Name
        Value: VPC-Private-RT

VPCPrivateRT2:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: VPC-Private-RT2

DefaultPrivateRoute:
  Type: AWS::EC2::Route
  DependsOn: VPCPublicEC2A
  Properties:
    RouteTableId: !Ref VPCPrivateRT
    DestinationCidrBlock: 0.0.0.0/0
    InstanceId: !Ref VPCPublicEC2A

DefaultPrivateRouteC:
  Type: AWS::EC2::Route
  DependsOn: VPCPublicEC2C
  Properties:
    RouteTableId: !Ref VPCPrivateRT2
    DestinationCidrBlock: 0.0.0.0/0
    InstanceId: !Ref VPCPublicEC2C

DefaultPrivateRoutePeering:
  Type: AWS::EC2::Route
  DependsOn: PeeringConnection
  Properties:
    RouteTableId: !Ref VPCPrivateRT
    DestinationCidrBlock: 30.0.0.0/16
    VpcPeeringConnectionId: !Ref PeeringConnection

DefaultPrivateRoutePeeringC:
  Type: AWS::EC2::Route
  DependsOn: PeeringConnection
  Properties:
    RouteTableId: !Ref VPCPrivateRT2
    DestinationCidrBlock: 30.0.0.0/16
    VpcPeeringConnectionId: !Ref PeeringConnection

VPCPublicSNA:
```

```
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: 10.0.0.0/24
    Tags:
      - Key: Name
        Value: VPC-Public-SN-A

VPCPublicSNC:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 2, !GetAZs '' ]
    CidrBlock: 10.0.2.0/24
    Tags:
      - Key: Name
        Value: VPC-Public-SN-C

VPCPrivateSNA:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: 10.0.1.0/24
    Tags:
      - Key: Name
        Value: VPC-Private-SN-A
 VPCPrivateSNC:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 2, !GetAZs '' ]
    CidrBlock: 10.0.3.0/24
    Tags:
      - Key: Name
        Value: VPC-Private-SN-C

VPCPublicSNARouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref VPCPublicRT
    SubnetId: !Ref VPCPublicSNA

VPCPublicSNCRouteTableAssociation:
```

```
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref VPCPublicRT
    SubnetId: !Ref VPCPublicSNC

VPCPrivateSNARouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref VPCPrivateRT
    SubnetId: !Ref VPCPrivateSNA

VPCPrivateSNCRouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref VPCPrivateRT2
    SubnetId: !Ref VPCPrivateSNC

VPCSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable HTTP access via port 80 and SSH access via port 22
and ICMP
    VpcId: !Ref VPC
    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
    - IpProtocol: icmp
      FromPort: -1
      ToPort: -1
      CidrIp: 0.0.0.0/0

VPCPublicEC2A:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t2.micro
    ImageId: ami-0e4a9ad2eb120e054
    KeyName: !Ref 3tierAPM
    Tags:
      - Key: Name
```

```
        Value: NAT-Instance-A
    NetworkInterfaces:
      - DeviceIndex: 0
        SubnetId: !Ref VPCPublicSNA
        GroupSet:
        - !Ref VPCSecurityGroup
        AssociatePublicIpAddress: true
    UserData:
      Fn::Base64:
        !Sub |
          #!/bin/bash
          yum -y update
          echo 1 > /proc/sys/net/ipv4/ip_forward
          echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
          /sbin/iptables -t nat -A POSTROUTING -o eth0 -s 0.0.0.0/0 -j MASQUERADE
          /sbin/iptables-save > /etc/sysconfig/iptables
          mkdir -p /etc/sysctl.d/
          cat <<EOF > /etc/sysctl.d/nat.conf
          net.ipv4.ip_forward = 1
          net.ipv4.conf.eth0.send_redirects = 0
          EOF

VPCPublicEC2B:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t2.micro
    ImageId: ami-0e4a9ad2eb120e054
    KeyName: !Ref 3tierAPM
    Tags:
      - Key: Name
        Value: Bastion-A
    NetworkInterfaces:
      - DeviceIndex: 0
        SubnetId: !Ref VPCPublicSNA
        GroupSet:
        - !Ref VPCSecurityGroup
        AssociatePublicIpAddress: true
    UserData:
      Fn::Base64:
        !Sub |
          #!/bin/bash
          yum -y update

VPCPublicEC2C:
  Type: AWS::EC2::Instance
```

```
  Properties:
    InstanceType: t2.micro
    ImageId: ami-0e4a9ad2eb120e054
    KeyName: !Ref 3tierAPM
    Tags:
      - Key: Name
        Value: NAT-Instance-C
    NetworkInterfaces:
      - DeviceIndex: 0
        SubnetId: !Ref VPCPublicSNC
        GroupSet:
        - !Ref VPCSecurityGroup
        AssociatePublicIpAddress: true
    UserData:
      Fn::Base64:
        !Sub |
          #!/bin/bash
          yum -y update
          echo 1 > /proc/sys/net/ipv4/ip_forward
          echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
          /sbin/iptables -t nat -A POSTROUTING -o eth0 -s 0.0.0.0/0 -j MASQUERADE
          /sbin/iptables-save > /etc/sysconfig/iptables
          mkdir -p /etc/sysctl.d/
          cat <<EOF > /etc/sysctl.d/nat.conf
          net.ipv4.ip_forward = 1
          net.ipv4.conf.eth0.send_redirects = 0
          EOF

VPCPublicEC2D:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t2.micro
    ImageId: ami-0e4a9ad2eb120e054
    KeyName: !Ref 3tierAPM
    Tags:
      - Key: Name
        Value: Bastion-C
    NetworkInterfaces:
      - DeviceIndex: 0
        SubnetId: !Ref VPCPublicSNC
        GroupSet:
        - !Ref VPCSecurityGroup
        AssociatePublicIpAddress: true
    UserData:
      Fn::Base64:
```

```
        !Sub |
          #!/bin/bash
          yum -y update

LaunchConfigurationWEB:
  Type: AWS::AutoScaling::LaunchConfiguration
  DependsOn: SecurityGroupWEB
  Properties:
    ImageId: ami-03a395aeeec067989
    InstanceType: t2.micro
    KeyName: !Ref 3tierAPM
    LaunchConfigurationName: WEBLC
    SecurityGroups:
      - !Ref SecurityGroupWEB
 SecurityGroupWEB:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: WEBSG
    GroupName: WEBSG
    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
      - IpProtocol: tcp
        FromPort: 443
        ToPort: 443
        CidrIp: 0.0.0.0/0
    VpcId: !Ref VPC

LaunchConfigurationWAS:
  Type: AWS::AutoScaling::LaunchConfiguration
  DependsOn: SecurityGroupWAS
  Properties:
    ImageId: ami-04b525f7909267825
    InstanceType: t2.micro
    KeyName: !Ref 3tierAPM
    LaunchConfigurationName: WASLC
    SecurityGroups:
      - !Ref SecurityGroupWAS
 SecurityGroupWAS:
```

```
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: WASSG
    GroupName: WASSG
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 8080
        ToPort: 8080
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 443
        ToPort: 443
        CidrIp: 0.0.0.0/0
    VpcId: !Ref VPC

ApplicationLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Name: ALB
    Type: application
    SecurityGroups:
    - !Ref SecurityGroupWAS
    Subnets:
    - !Ref VPCPublicSNA
    - !Ref VPCPublicSNC
 TargetGroupWEB:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    VpcId: !Ref VPC
    Name: WEBTG
    Port: 80
    Protocol: HTTP
    VpcId: !Ref VPC
    HealthCheckEnabled: true
    HealthCheckIntervalSeconds: 100
    HealthCheckPath: /
    HealthyThresholdCount: 5
```

```
    UnhealthyThresholdCount: 2

 TargetGroupWAS:
   Type: AWS::ElasticLoadBalancingV2::TargetGroup
   Properties:
     VpcId: !Ref VPC
     Name: WASTG
     Port: 80
     Protocol: HTTP
     VpcId: !Ref VPC
     HealthCheckEnabled: true
     HealthCheckIntervalSeconds: 100
     HealthCheckPath: /
     HealthyThresholdCount: 5
     UnhealthyThresholdCount: 2

 ALBListener:
   Type: AWS::ElasticLoadBalancingV2::Listener
   DependsOn: ApplicationLoadBalancer
   Properties:
     DefaultActions:
       - RedirectConfig:
           Protocol: HTTPS
           Host: '#{host}'
           Port: 443
           Path: '/#{path}'
           Query: '/#{query}'
           StatusCode: HTTP_301
         Type: redirect
     LoadBalancerArn: !Ref ApplicationLoadBalancer
     Port: 80
     Protocol: HTTP

 ALBListenerSSLWEB:
   Type: AWS::ElasticLoadBalancingV2::Listener
   Properties:
     Certificates:
       - CertificateArn:
arn:aws:acm:ap-northeast-2:367888156959:certificate/924fbddd-2a5f-463b-a44f-dadb91d
3413d
     DefaultActions:
       - Type: forward
         TargetGroupArn: !Ref TargetGroupWEB
     LoadBalancerArn: !Ref ApplicationLoadBalancer
     Port: 443
```

```
    Protocol: HTTPS
    SslPolicy: ELBSecurityPolicy-TLS-1-0-2015-04

 ALBListenerSSLWAS:
   Type: AWS::ElasticLoadBalancingV2::Listener
   Properties:
     Certificates:
       - CertificateArn:
arn:aws:acm:ap-northeast-2:367888156959:certificate/924fbddd-2a5f-463b-a44f-dadb91d
3413d
     DefaultActions:
       - Type: forward
         TargetGroupArn: !Ref TargetGroupWAS
     LoadBalancerArn: !Ref ApplicationLoadBalancer
     Port: 8080
     Protocol: HTTPS
     SslPolicy: ELBSecurityPolicy-TLS-1-0-2015-04

 WEBASG:
   Type: AWS::AutoScaling::AutoScalingGroup
   Properties:
     AutoScalingGroupName: WEBASG
     VPCZoneIdentifier:
       - !Ref VPCPrivateSNA
       - !Ref VPCPrivateSNC
     Cooldown: 100
     LaunchConfigurationName: !Ref LaunchConfigurationWEB
     MaxSize: 4
     MinSize: 1
     DesiredCapacity: 1
     TargetGroupARNs:
       - !Ref TargetGroupWEB
     Tags:
       - Key: Name
         Value: WEBASG
         PropagateAtLaunch: true
  WEBASGPolicy:
   Type: AWS::AutoScaling::ScalingPolicy
   Properties:
     AutoScalingGroupName: !Ref WEBASG
     PolicyType: TargetTrackingScaling
     TargetTrackingConfiguration:
       PredefinedMetricSpecification:
         PredefinedMetricType: ASGAverageCPUUtilization
       TargetValue: 60
```

```
WASASG:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    AutoScalingGroupName: WASASG
    VPCZoneIdentifier:
      - !Ref VPCPrivateSNA
      - !Ref VPCPrivateSNC
    Cooldown: 100
    LaunchConfigurationName: !Ref LaunchConfigurationWAS
    MaxSize: 4
    MinSize: 1
    DesiredCapacity: 1
    TargetGroupARNs:
      - !Ref TargetGroupWAS
    Tags:
      - Key: Name
        Value: WASASG
        PropagateAtLaunch: true
 WASASGPolicy:
  Type: AWS::AutoScaling::ScalingPolicy
  Properties:
    AutoScalingGroupName: !Ref WASASG
    PolicyType: TargetTrackingScaling
    TargetTrackingConfiguration:
      PredefinedMetricSpecification:
        PredefinedMetricType: ASGAverageCPUUtilization
      TargetValue: 60

VPCPeering:
  Type: AWS::EC2::VPC
  Properties:
   CidrBlock: 30.0.0.0/16
   EnableDnsSupport: true
   EnableDnsHostnames: true
   Tags:
      - Key: Name
        Value: VPCPeering

VPCPeeringIGW:
  Type: AWS::EC2::InternetGateway
  Properties:
    Tags:
      - Key: Name
        Value: VPCPeering-IGW
```

```
VPCPeeringIGWAttachment:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    InternetGatewayId: !Ref VPCPeeringIGW
    VpcId: !Ref VPCPeering

VPCPeeringPublicRT:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPCPeering
    Tags:
      - Key: Name
        Value: VPCPeering-Public-RT

PeeringDefaultPublicRoute:
  Type: AWS::EC2::Route
  DependsOn: VPCPeeringIGWAttachment
  Properties:
    RouteTableId: !Ref VPCPeeringPublicRT
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref VPCPeeringIGW

VPCPeeringPrivateRT:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPCPeering
    Tags:
      - Key: Name
        Value: VPCPeering-Private-RT

VPCPeeringPrivateRT2:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPCPeering
    Tags:
      - Key: Name
        Value: VPCPeering-Private-RT2

VPCPeeringPublicSNA:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPCPeering
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: 30.0.0.0/24
```

```
    Tags:
      - Key: Name
        Value: VPCPeering-Public-SN-A

VPCPeeringPrivateSNA:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPCPeering
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: 30.0.1.0/24
    Tags:
      - Key: Name
        Value: VPCPeering-Private-SN-A
 VPCPeeringPrivateSNC:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPCPeering
    AvailabilityZone: !Select [ 2, !GetAZs '' ]
    CidrBlock: 30.0.3.0/24
    Tags:
      - Key: Name
        Value: VPCPeering-Private-SN-C

VPCPeeringPublicSNARouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref VPCPeeringPublicRT
    SubnetId: !Ref VPCPeeringPublicSNA

VPCPeeringPrivateSNARouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref VPCPeeringPrivateRT
    SubnetId: !Ref VPCPeeringPrivateSNA

VPCPeeringPrivateSNCRouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref VPCPeeringPrivateRT2
    SubnetId: !Ref VPCPeeringPrivateSNC

VPCPeeringSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: All traffic
```

```
    VpcId: !Ref VPCPeering
    SecurityGroupIngress:
    - IpProtocol: '-1'
      CidrIp: 0.0.0.0/0

 VPCPeeringDBSecurityGroup:
   Type: AWS::EC2::SecurityGroup
   Properties:
     GroupDescription: Enable MYSQL/Aurora access via port 3306 and SSH access via
port 22 and ICMP
     VpcId: !Ref VPCPeering
     SecurityGroupIngress:
     - IpProtocol: tcp
       FromPort: '3306'
       ToPort: '3306'
       CidrIp: 0.0.0.0/0
     - IpProtocol: tcp
       FromPort: '22'
       ToPort: '22'
       CidrIp: 0.0.0.0/0

 VPCPeeringPublicEC2A:
   Type: AWS::EC2::Instance
   Properties:
     InstanceType: t2.micro
     ImageId: ami-0e4a9ad2eb120e054
     KeyName: !Ref 3tierAPM
     Tags:
       - Key: Name
         Value: BastionPeering-A
     NetworkInterfaces:
       - DeviceIndex: 0
         SubnetId: !Ref VPCPeeringPublicSNA
         GroupSet:
         - !Ref VPCPeeringSecurityGroup
         AssociatePublicIpAddress: true
     UserData:
       Fn::Base64:
         !Sub |
           #!/bin/bash
           yum -y update
  PeeringConnection:
   Type: AWS::EC2::VPCPeeringConnection
   Properties:
     PeerVpcId: !Ref VPCPeering
```

```
    Tags:
      - Key: Name
        Value: VPCPeeringV1
    VpcId: !Ref VPC

PeeringDefaultPrivateRoute:
  Type: AWS::EC2::Route
  DependsOn: PeeringConnection
  Properties:
    RouteTableId: !Ref VPCPeeringPrivateRT
    DestinationCidrBlock: 10.0.0.0/16
    VpcPeeringConnectionId: !Ref PeeringConnection

PeeringDefaultPrivateRouteC:
  Type: AWS::EC2::Route
  DependsOn: PeeringConnection
  Properties:
    RouteTableId: !Ref VPCPeeringPrivateRT2
    DestinationCidrBlock: 10.0.0.0/16
    VpcPeeringConnectionId: !Ref PeeringConnection

VPCPeeringSecurityGroupDB:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable Mariadb access via port 3306
    VpcId: !Ref VPCPeering
    SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: '3306'
      ToPort: '3306'
      CidrIp: 0.0.0.0/0
 DBSubnet:
  Type: AWS::RDS::DBSubnetGroup
  Properties:
    DBSubnetGroupDescription: PriA,PriC
    DBSubnetGroupName: RDS_TEST_SubG
    SubnetIds:
      - !Ref VPCPeeringPrivateSNA
      - !Ref VPCPeeringPrivateSNC

RDS:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: !Ref DBNameP
    MasterUsername: !Ref MUser
```

```
    MasterUserPassword: !Ref MPass
    Engine: MariaDB
    DBInstanceClass: db.t2.micro
    StorageType: gp2
    PubliclyAccessible: false
    AllocatedStorage: "10"
    DBInstanceIdentifier: !Join ["-",["TierThreeAPM", !Ref "AWS::Region"]]
    MultiAZ: true
    VPCSecurityGroups:
      - !Ref VPCPeeringSecurityGroupDB
    DBSubnetGroupName: !Ref DBSubnet

RDSReplica:
  Type: AWS::RDS::DBInstance
  DependsOn: RDS
  Properties:
    SourceDBInstanceIdentifier: !Ref RDS
    DBInstanceIdentifier: TierThreeAPMRR
    DBInstanceClass: db.t2.micro
    StorageType: gp2
    PubliclyAccessible: false
    VPCSecurityGroups:
      - !Ref VPCPeeringSecurityGroupDB
```