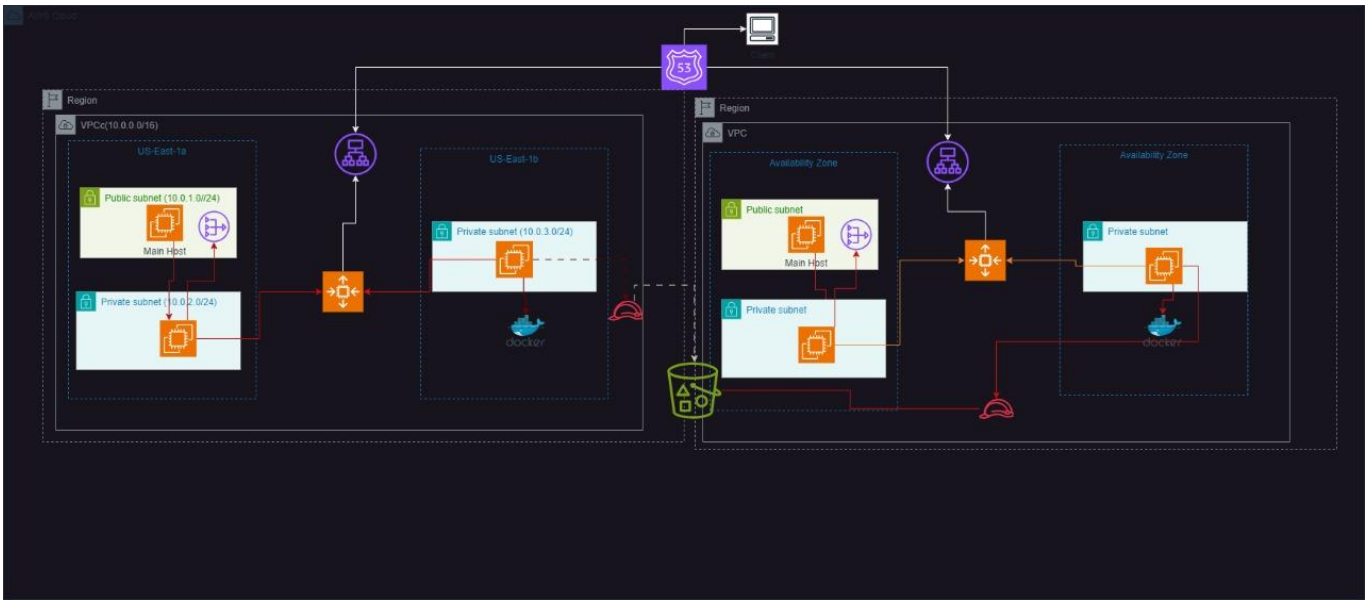
Overview

Our AWS architecture is designed to host a web application with high availability and scalability. It includes a VPC with two subnets, an ALB, an auto-scaling group with EC2 instances running Docker containers, CloudWatch for monitoring and alerts, and S3 for storing static content.

Architecture Diagram



Components Description

- **VPC**: A single VPC with a CIDR block of `10.0.0.0/16`.
- **Subnets**: Two subnets in different availability zones (`10.0.1.0/24` and `10.0.2.0/24`).
- **Route Tables**: Custom route tables associated with each subnet.
- **ALB**: An Application Load Balancer distributing traffic to the EC2 instances.
- **Auto Scaling Group**: Configured to maintain a minimum of 2 instances across different availability zones.
- **EC2 Instances**: Two t2.micro instances with Docker installed, running web application containers.
- **S3**: Static content is stored in an S3 bucket and mounted on the EC2 instances using S3FS.
- **CloudWatch**: Monitors CPU usage and triggers scaling policies when usage exceeds 60%. Sends email notifications for scaling events.

Security

- **Security Groups**: Configured to allow HTTP/HTTPS traffic to the ALB and SSH access to the EC2 instances.
- **IAM Roles**: Roles assigned to EC2 instances for accessing S3 and other AWS services.

Monitoring and Alerts

- **CloudWatch Alarms**: Set to monitor CPU usage and trigger scaling policies.
- **Notifications**: Email notifications sent via SNS when scaling events occur.

Scalability and Fault Tolerance

- **Auto Scaling**: Ensures the application scales based on demand.
- **Multi-AZ Deployment**: Ensures high availability by distributing instances across multiple availability zones.

S3 Mount

Step 1: Install Required Packages

1. **Install `fuse`**:

- sudo yum install -y fuse

2. **Install `s3fs-fuse`**:

- sudo yum install -y s3fs-fuse

Step 2: Configure IAM Role

1. **Create an IAM Role** with S3 permissions (e.g., AmazonS3FullAccess).
2. **Attach the Role** to your EC2 instance.

Step 3: Create a Mount Point

- sudo mkdir -p /mnt/s3bucket

Step 4: Set Permissions on Mount Point

1. **Change Ownership and Permissions**:

- sudo chown ec2-user:ec2-user /mnt/s3bucket
- sudo chmod 755 /mnt/s3bucket

Step 5: Test Mounting the S3 Bucket

1. **Mount the Bucket Temporarily**:

- s3fs s3bucket /mnt/s3bucket -o iam_role=auto,allow_other

2. **Verify the Mount**:

- ls /mnt/s3bucket

Step 6: Make the Mount Persistent Across Reboots

1. **Edit `/etc/fstab`**:

- sudo nano /etc/fstab

2. **Add the Following Line**:

- s3fs#s3bucket /mnt/s3bucket fuse _netdev,iam_role=auto,allow_other 0 0

The cloudformation code for all the environment:

For the primary resources:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: 'VPC with Private Subnets, Public Subnet, NAT Gateway, AutoScaling Group, and ALB'

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
```

```
      - Key: Name
        Value: MyVPC

PublicSubnet:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.1.0/24
    AvailabilityZone: us-west-2a
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: PublicSubnet


PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.2.0/24
    AvailabilityZone: us-west-2a
    Tags:
      - Key: Name
        Value: PrivateSubnet1

PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
```

```
    CidrBlock: 10.0.3.0/24
    AvailabilityZone: us-west-2b
    Tags:
     - Key: Name
       Value: PrivateSubnet2

InternetGateway:
  Type: AWS::EC2::InternetGateway

AttachGateway:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    VpcId: !Ref VPC
    InternetGatewayId: !Ref InternetGateway

PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC

PublicRoute:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PublicSubnetRouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
```

```
  Properties:
    SubnetId: !Ref PublicSubnet
    RouteTableId: !Ref PublicRouteTable

PublicSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PublicSubnet2
    RouteTableId: !Ref PublicRouteTable

NatGateway:
  Type: AWS::EC2::NatGateway
  Properties:
    SubnetId: !Ref PublicSubnet
    AllocationId: !GetAtt EIP.AllocationId

EIP:
  Type: AWS::EC2::EIP
  Properties:
    Domain: vpc

PrivateRouteTable1:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC

PrivateRouteTable2:
  Type: AWS::EC2::RouteTable
  Properties:
```

```
    VpcId: !Ref VPC

PrivateRoute1:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway

PrivateRoute2:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway

PrivateSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PrivateSubnet1
    RouteTableId: !Ref PrivateRouteTable1

PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable2

InstanceSecurityGroup:
```

```
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow HTTP traffic
    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  # Allow SSH for Bastian access

LaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateName: MyLaunchTemplate
    LaunchTemplateData:
      ImageId: ami-0fcbe951d32437119
      InstanceType: t2.micro
      SecurityGroupIds:
        - !Ref InstanceSecurityGroup  # Ensure security group is linked to the same VPC
      KeyName: webserver  # Use the correct key pair
      UserData:
        Fn::Base64: |
          #!/bin/bash
          sudo docker run -p 80:80 -v /home/ec2-user/app/app:/usr/share/nginx/html -d webserver
```

```
AutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - !Ref PrivateSubnet1
      - !Ref PrivateSubnet2
    LaunchTemplate:
      LaunchTemplateId: !Ref LaunchTemplate
      Version: "1"  # Version must match your actual Launch Template
    MinSize: 2
    MaxSize: 4
    TargetGroupARNs:
      - !Ref ALBTargetGroup

ALB:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Name: MyALB
    Subnets:
      - !Ref PublicSubnet
      - !Ref PublicSubnet2
    SecurityGroups:
      - !Ref InstanceSecurityGroup  # Attach security group to ALB
    Scheme: internet-facing
    Type: application

ALBTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
```

```
    VpcId: !Ref VPC
    Port: 80
    Protocol: HTTP
    TargetType: instance
    HealthCheckProtocol: HTTP
    HealthCheckPort: 80
    HealthCheckPath: /
    Matcher:
      HttpCode: 200
# Scaling Policies
  ScaleUpPolicy:
    Type: AWS::AutoScaling::ScalingPolicy
    Properties:
     AutoScalingGroupName: !Ref AutoScalingGroup
     PolicyType: TargetTrackingScaling
     TargetTrackingConfiguration:
      PredefinedMetricSpecification:
       PredefinedMetricType: ASGAverageCPUUtilization
      TargetValue: 60.0

  ScaleDownPolicy:
    Type: AWS::AutoScaling::ScalingPolicy
    Properties:
     AutoScalingGroupName: !Ref AutoScalingGroup
     PolicyType: TargetTrackingScaling
     TargetTrackingConfiguration:
      PredefinedMetricSpecification:
       PredefinedMetricType: ASGAverageCPUUtilization
      TargetValue: 30.0
```

```
# CloudWatch Alarms for CPU Utilization
HighCPUAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription: "Alarm when CPU exceeds 60%"
    Namespace: AWS/EC2
    MetricName: CPUUtilization
    Dimensions:
      - Name: AutoScalingGroupName
        Value: !Ref AutoScalingGroup
    Statistic: Average
    Period: 60
    EvaluationPeriods: 1
    Threshold: 60
    ComparisonOperator: GreaterThanOrEqualToThreshold
    AlarmActions:
      - !Ref ScaleUpPolicy

LowCPUAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription: "Alarm when CPU is below 30%"
    Namespace: AWS/EC2
    MetricName: CPUUtilization
    Dimensions:
      - Name: AutoScalingGroupName
        Value: !Ref AutoScalingGroup
    Statistic: Average
```

```
    Period: 60
    EvaluationPeriods: 1
    Threshold: 30
    ComparisonOperator: LessThanOrEqualToThreshold
    AlarmActions:
     - !Ref ScaleDownPolicy


  BastianHost:
   Type: AWS::EC2::Instance
   Properties:
    InstanceType: t2.micro
    SubnetId: !Ref PublicSubnet
    KeyName: webserver
    SecurityGroupIds:
     - !Ref InstanceSecurityGroup
    ImageId: ami-033067239f2d2bfbc
    UserData:
     Fn::Base64: |
      #!/bin/bash
      sudo yum update -y

Outputs:
  VPCId:
   Description: VPC Id
   Value: !Ref VPC
```

For the DR resources:

AWSTemplateFormatVersion: '2010-09-09'

Description: 'VPC with Private Subnets, Public Subnet, NAT Gateway, AutoScaling Group, and ALB'

Resources:

VPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: 10.0.0.0/16

EnableDnsSupport: true

EnableDnsHostnames: true

Tags:

- Key: Name

Value: MyVPC-us-east

PublicSubnet:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

CidrBlock: 10.0.1.0/24

AvailabilityZone: us-east-1a

MapPublicIpOnLaunch: true

Tags:

```
      - Key: Name
        Value: PublicSubnet

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      CidrBlock: 10.0.2.0/24
      AvailabilityZone: us-east-1a
      Tags:
        - Key: Name
          Value: PrivateSubnet1

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      CidrBlock: 10.0.3.0/24
      AvailabilityZone: us-east-1b
      Tags:
        - Key: Name
          Value: PrivateSubnet2
```

InternetGateway:

Type: AWS::EC2::InternetGateway

AttachGateway:

Type: AWS::EC2::VPCGatewayAttachment

Properties:

VpcId: !Ref VPC

InternetGatewayId: !Ref InternetGateway

PublicRouteTable:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

PublicRoute:

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PublicRouteTable

DestinationCidrBlock: 0.0.0.0/0

GatewayId: !Ref InternetGateway

PublicSubnetRouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

SubnetId: !Ref PublicSubnet

RouteTableId: !Ref PublicRouteTable

NatGateway:

Type: AWS::EC2::NatGateway

Properties:

SubnetId: !Ref PublicSubnet

AllocationId: !GetAtt EIP.AllocationId

EIP:

Type: AWS::EC2::EIP

Properties:

Domain: vpc

PrivateRouteTable1:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

PrivateRouteTable2:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

PrivateRoute1:

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PrivateRouteTable1

DestinationCidrBlock: 0.0.0.0/0

NatGatewayId: !Ref NatGateway

PrivateRoute2:

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PrivateRouteTable2

DestinationCidrBlock: 0.0.0.0/0

NatGatewayId: !Ref NatGateway

PrivateSubnet1RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

SubnetId: !Ref PrivateSubnet1

RouteTableId: !Ref PrivateRouteTable1

PrivateSubnet2RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

SubnetId: !Ref PrivateSubnet2

RouteTableId: !Ref PrivateRouteTable2

InstanceSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Allow HTTP traffic

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  # Allow SSH for Bastian access

AutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - !Ref PrivateSubnet1
      - !Ref PrivateSubnet2
    LaunchTemplate:
      LaunchTemplateId: lt-0a8e5eb68a4e0202c
      Version: 1
    MinSize: 2
    MaxSize: 4
    HealthCheckGracePeriod: 300
    TargetGroupARNs:
      - !Ref ALBTargetGroup
    MetricsCollection:
      - Granularity: "1Minute"
    Tags:
      - Key: Name
        Value: "MyASG"
```

PropagateAtLaunch: true

ALB:

Type: AWS::ElasticLoadBalancingV2::LoadBalancer

Properties:

Name: MyALB-us-east

Subnets:

- !Ref PublicSubnet

SecurityGroups:

- !Ref InstanceSecurityGroup

Scheme: internet-facing

Type: application

ALBTargetGroup:

Type: AWS::ElasticLoadBalancingV2::TargetGroup

Properties:

VpcId: !Ref VPC

Port: 80

Protocol: HTTP

TargetType: instance

HealthCheckProtocol: HTTP

HealthCheckPort: 80

```
    HealthCheckPath: /
    Matcher:
      HttpCode: 200

# Scaling Policies
ScaleUpPolicy:
  Type: AWS::AutoScaling::ScalingPolicy
  Properties:
    AutoScalingGroupName: !Ref AutoScalingGroup
    PolicyType: TargetTrackingScaling
    TargetTrackingConfiguration:
      PredefinedMetricSpecification:
        PredefinedMetricType: ASGAverageCPUUtilization
      TargetValue: 60.0

ScaleDownPolicy:
  Type: AWS::AutoScaling::ScalingPolicy
  Properties:
    AutoScalingGroupName: !Ref AutoScalingGroup
    PolicyType: TargetTrackingScaling
    TargetTrackingConfiguration:
      PredefinedMetricSpecification:
```

```
      PredefinedMetricType: ASGAverageCPUUtilization
    TargetValue: 30.0

# CloudWatch Alarms for CPU Utilization
HighCPUAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription: "Alarm when CPU exceeds 60%"
    Namespace: AWS/EC2
    MetricName: CPUUtilization
    Dimensions:
      - Name: AutoScalingGroupName
        Value: !Ref AutoScalingGroup
    Statistic: Average
    Period: 60
    EvaluationPeriods: 1
    Threshold: 60
    ComparisonOperator: GreaterThanOrEqualToThreshold
    AlarmActions:
      - !Ref ScaleUpPolicy

LowCPUAlarm:
```

```
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: "Alarm when CPU is below 30%"
      Namespace: AWS/EC2
      MetricName: CPUUtilization
      Dimensions:
        - Name: AutoScalingGroupName
          Value: !Ref AutoScalingGroup
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 30
      ComparisonOperator: LessThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleDownPolicy

  BastianHost:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t2.micro
      SubnetId: !Ref PublicSubnet
      KeyName: s3-keypair
```

```
      SecurityGroupIds:
        - !Ref InstanceSecurityGroup
      ImageId: ami-0e54eba7c51c234f6
      UserData:
        Fn::Base64: |
          #!/bin/bash
          sudo yum update -y

Outputs:
  VPCId:
    Description: VPC Id
    Value: !Ref VPC
```