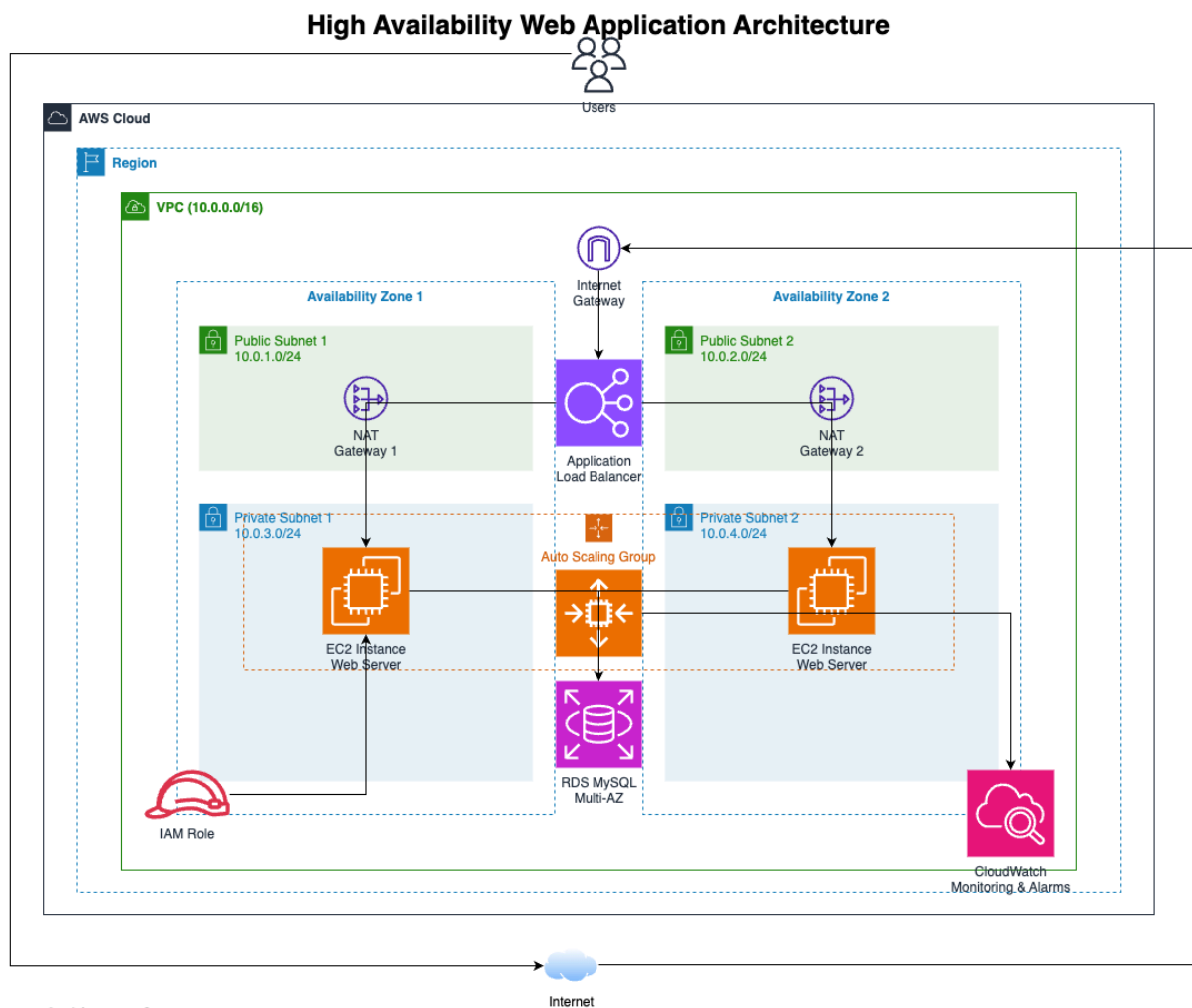


Day2 High Availability Web Application with Monitoring Real Flow



Architecture Components:

1. Multi-AZ VPC with public and private subnets
2. Internet Gateway and NAT Gateways for secure outbound connectivity
3. Application Load Balancer distributing traffic across Availability Zones
4. Auto Scaling Group maintaining EC2 instances with scale up/down policies
5. EC2 instances running web servers with CloudWatch monitoring
6. Multi-AZ RDS MySQL database for high availability
7. CloudWatch for monitoring and triggering Auto Scaling events
8. IAM roles for secure service permissions

1. IAM policy and Role for this project.

CloudFormation-HA-WebApp-Policy

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:GetRole",
        "iam:PutRolePolicy",
        "iam:DeleteRolePolicy",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:CreateInstanceProfile",
        "iam:DeleteInstanceProfile",
        "iam:AddRoleToInstanceProfile",
        "iam:RemoveRoleFromInstanceProfile",
        "iam:GetInstanceProfile",
        "iam:TagRole",
        "iam:TagInstanceProfile"
      ],
      "Resource": [
        "arn:aws:iam::*:role/ha-web-app-*",
        "arn:aws:iam::*:instance-profile/ha-web-app-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
        "cloudwatch:*",
        "rds:*",
        "logs:*",
        "ssm:*"
      ],
      "Resource": "*"
    }
  ]
}

```

```

    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::*:role/ha-web-app-*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "ec2.amazonaws.com",
            "rds.amazonaws.com"
          ]
        }
      }
    }
  ]
}

```



CloudFormation-HA-WebApp-Role

Permissions policies (2) [Info](#)

You can attach up to 10 managed policies.

Filter by Type

Search All types < 1 > ⚙

<input type="checkbox"/>	Policy name ↗	Type	Attached entities
<input type="checkbox"/>	 AWSCloudFormationFullAccess	AWS managed	2
<input type="checkbox"/>	 CloudFormation-HA-WebApp-Policy	Customer managed	1

2.Create CloudFormation Stack

AWSTemplateFormatVersion: '2010-09-09'

Description: 'Day 2 - High Availability Web Application with Monitoring'

Parameters:

EnvironmentName:

Description: Environment name for resource prefixes

Type: String

Default: ha-web-app

VpcCIDR:

Description: CIDR block for VPC

Type: String

Default: 10.0.0.0/16

PublicSubnet1CIDR:

Type: String

Default: 10.0.1.0/24

PublicSubnet2CIDR:

Type: String

Default: 10.0.2.0/24

PrivateSubnet1CIDR:

Type: String

Default: 10.0.3.0/24

PrivateSubnet2CIDR:

Type: String

Default: 10.0.4.0/24

InstanceType:

Description: EC2 instance type

Type: String

Default: t2.micro

DBInstanceClass:

Description: RDS instance class

Type: String

Default: db.t3.micro

DBName:

Description: Database name

Type: String

Default: appdb

DBUsername:

Description: Database admin username

Type: String

Default: admin

DBPassword:

Description: Database admin password

Type: String

NoEcho: true

MinLength: 8

Resources:

IAM Roles and Policies

EC2Role:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Version: '2012-10-17'

Statement:

- Effect: Allow

Principal:

Service: ec2.amazonaws.com

Action: sts:AssumeRole

ManagedPolicyArns:

- arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
- arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy

Policies:

- PolicyName: EC2CustomPolicy

PolicyDocument:

Version: '2012-10-17'

Statement:

- Effect: Allow

Action:

- rds:DescribeDBInstances
- elasticloadbalancing:DescribeLoadBalancers

Resource: '*'

EC2InstanceProfile:

Type: AWS::IAM::InstanceProfile

Properties:

Path: /

Roles:

- !Ref EC2Role

VPC and Network Configuration

VPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: !Ref VpcCIDR

EnableDnsHostnames: true

EnableDnsSupport: true

Tags:

- Key: Name
Value: !Sub \${EnvironmentName}-VPC

InternetGateway:

Type: AWS::EC2::InternetGateway

Properties:

Tags:

- Key: Name
Value: !Sub \${EnvironmentName}-IGW

InternetGatewayAttachment:

Type: AWS::EC2::VPCEGatewayAttachment

Properties:

InternetGatewayId: !Ref InternetGateway

VpcId: !Ref VPC

PublicSubnet1:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

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

CidrBlock: !Ref PublicSubnet1CIDR

MapPublicIpOnLaunch: true

Tags:

- Key: Name

Value: !Sub \${EnvironmentName}-PublicSubnet1

PublicSubnet2:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

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

CidrBlock: !Ref PublicSubnet2CIDR

MapPublicIpOnLaunch: true

Tags:

- Key: Name

Value: !Sub \${EnvironmentName}-PublicSubnet2

PrivateSubnet1:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

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

CidrBlock: !Ref PrivateSubnet1CIDR

MapPublicIpOnLaunch: false

Tags:

- Key: Name

Value: !Sub \${EnvironmentName}-PrivateSubnet1

PrivateSubnet2:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

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

CidrBlock: !Ref PrivateSubnet2CIDR

MapPublicIpOnLaunch: false

Tags:

- Key: Name

Value: !Sub \${EnvironmentName}-PrivateSubnet2

NAT Gateways

NatGateway1EIP:

Type: AWS::EC2::EIP

DependsOn: InternetGatewayAttachment

Properties:

Domain: vpc

NatGateway2EIP:

Type: AWS::EC2::EIP

DependsOn: InternetGatewayAttachment

Properties:

Domain: vpc

NatGateway1:

Type: AWS::EC2::NatGateway

Properties:

AllocationId: !GetAtt NatGateway1EIP.AllocationId

SubnetId: !Ref PublicSubnet1

NatGateway2:

Type: AWS::EC2::NatGateway

Properties:

AllocationId: !GetAtt NatGateway2EIP.AllocationId

SubnetId: !Ref PublicSubnet2

Route Tables

PublicRouteTable:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

Tags:

- Key: Name

Value: !Sub \${EnvironmentName}-PublicRoutes

PrivateRouteTable1:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

Tags:

- Key: Name

Value: !Sub \${EnvironmentName}-PrivateRoutes1

PrivateRouteTable2:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

Tags:

- Key: Name

Value: !Sub \${EnvironmentName}-PrivateRoutes2

DefaultPublicRoute:

Type: AWS::EC2::Route

DependsOn: InternetGatewayAttachment

Properties:

RouteTableId: !Ref PublicRouteTable

DestinationCidrBlock: 0.0.0.0/0

GatewayId: !Ref InternetGateway

DefaultPrivateRoute1:

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PrivateRouteTable1

DestinationCidrBlock: 0.0.0.0/0

NatGatewayId: !Ref NatGateway1

DefaultPrivateRoute2:

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PrivateRouteTable2

DestinationCidrBlock: 0.0.0.0/0

NatGatewayId: !Ref NatGateway2

PublicSubnet1RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref PublicRouteTable

SubnetId: !Ref PublicSubnet1

PublicSubnet2RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref PublicRouteTable

SubnetId: !Ref PublicSubnet2

PrivateSubnet1RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref PrivateRouteTable1

SubnetId: !Ref PrivateSubnet1

PrivateSubnet2RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref PrivateRouteTable2

SubnetId: !Ref PrivateSubnet2

Security Groups

ALBSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: ALB Security Group

VpcId: !Ref VPC

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: 80

ToPort: 80

CidrIp: 0.0.0.0/0

Tags:

- Key: Name

Value: !Sub \${EnvironmentName}-ALB-SG

WebServerSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Web Server Security Group

VpcId: !Ref VPC

SecurityGroupIngress:

- IpProtocol: tcp
- FromPort: 80
- ToPort: 80
- SourceSecurityGroupId: !Ref ALBSecurityGroup

Tags:

- Key: Name
- Value: !Sub \${EnvironmentName}-WebServer-SG

RDSSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: RDS Security Group

VpcId: !Ref VPC

SecurityGroupIngress:

- IpProtocol: tcp
- FromPort: 3306
- ToPort: 3306
- SourceSecurityGroupId: !Ref WebServerSecurityGroup

Tags:

- Key: Name
- Value: !Sub \${EnvironmentName}-RDS-SG

Application Load Balancer

ApplicationLoadBalancer:

Type: AWS::ElasticLoadBalancingV2::LoadBalancer

Properties:

Subnets:

- !Ref PublicSubnet1
- !Ref PublicSubnet2

SecurityGroups:

- !Ref ALBSecurityGroup

Tags:

- Key: Name
- Value: !Sub \${EnvironmentName}-ALB

ALBListener:

Type: AWS::ElasticLoadBalancingV2::Listener

Properties:

DefaultActions:

- Type: forward

TargetGroupArn: !Ref ALBTargetGroup

LoadBalancerArn: !Ref ApplicationLoadBalancer

Port: 80

Protocol: HTTP

ALBTargetGroup:

Type: AWS::ElasticLoadBalancingV2::TargetGroup

Properties:

HealthCheckIntervalSeconds: 30

HealthCheckPath: /health

HealthCheckTimeoutSeconds: 5

HealthyThresholdCount: 2

Port: 80

Protocol: HTTP

UnhealthyThresholdCount: 5

VpcId: !Ref VPC

TargetGroupAttributes:

- Key: deregistration_delay.timeout_seconds
Value: '30'

RDS Instance

DBSubnetGroup:

Type: AWS::RDS::DBSubnetGroup

Properties:

DBSubnetGroupDescription: Subnet group for RDS

SubnetIds:

- !Ref PrivateSubnet1
- !Ref PrivateSubnet2

RDSInstance:

Type: AWS::RDS::DBInstance

Properties:

DBName: !Ref DBName

Engine: mysql

MasterUsername: !Ref DBUsername

MasterUserPassword: !Ref DBPassword

```
DBInstanceClass: !Ref DBInstanceClass
DBSubnetGroupName: !Ref DBSubnetGroup
VPCSecurityGroups:
  - !Ref RDSSecurityGroup
AllocatedStorage: '20'
MultiAZ: true
PubliclyAccessible: false
```

Launch Template

WebServerLaunchTemplate:

```
Type: AWS::EC2::LaunchTemplate
```

Properties:

LaunchTemplateData:

```
ImageId: !Sub '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-linux-x86_64-gp2}}
```

```
InstanceType: !Ref InstanceType
```

SecurityGroupIds:

```
- !Ref WebServerSecurityGroup
```

IamInstanceProfile:

```
Name: !Ref EC2InstanceProfile
```

UserData:

```
Fn::Base64: !Sub |
```

```
#!/bin/bash
```

```
yum update -y
```

```
yum install -y httpd mysql
```

```
systemctl start httpd
```

```
systemctl enable httpd
```

```
echo "Healthy" > /var/www/html/health
```

```
echo "<h1>Hello from ${EnvironmentName}</h1>" > /var/www/html/index.html
```

Install CloudWatch agent

```
yum install -y amazon-cloudwatch-agent
```

```
cat > /opt/aws/amazon-cloudwatch-agent/bin/config.json << 'EOF'
```

```
{
```

```
  "metrics": {
```

```
    "metrics_collected": {
```

```
      "cpu": {
```

```
        "measurement": ["cpu_usage_idle", "cpu_usage_user", "cpu_usage_system",
```

```
      ],
```

```

    "mem": {
      "measurement": ["mem_used_percent"]
    },
    "disk": {
      "measurement": ["disk_used_percent"],
      "resources": ["/"]
    }
  }
}
}
EOF

```

```

/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-c
systemctl start amazon-cloudwatch-agent
systemctl enable amazon-cloudwatch-agent

```

Auto Scaling Group

WebServerASG:

Type: AWS::AutoScaling::AutoScalingGroup

Properties:

VPCZoneIdentifier:

- !Ref PrivateSubnet1
- !Ref PrivateSubnet2

LaunchTemplate:

LaunchTemplateId: !Ref WebServerLaunchTemplate

Version: !GetAtt WebServerLaunchTemplate.LatestVersionNumber

TargetGroupARNs:

- !Ref ALBTargetGroup

HealthCheckType: ELB

HealthCheckGracePeriod: 300

MinSize: 2

MaxSize: 4

DesiredCapacity: 2

Tags:

- Key: Name
 - Value: !Sub \${EnvironmentName}-WebServer
 - PropagateAtLaunch: true

CloudWatch Alarms

CPUAlarmHigh:

Type: AWS::CloudWatch::Alarm

Properties:

AlarmDescription: Scale up if CPU > 70% for 5 minutes

MetricName: CPUUtilization

Namespace: AWS/EC2

Statistic: Average

Period: 300

EvaluationPeriods: 2

ComparisonOperator: GreaterThanThreshold

Threshold: 70

AlarmActions:

- !Ref WebServerScaleUpPolicy

Dimensions:

- Name: AutoScalingGroupName
Value: !Ref WebServerASG

CPUAlarmLow:

Type: AWS::CloudWatch::Alarm

Properties:

AlarmDescription: Scale down if CPU < 30% for 5 minutes

MetricName: CPUUtilization

Namespace: AWS/EC2

Statistic: Average

Period: 300

EvaluationPeriods: 2

ComparisonOperator: LessThanThreshold

Threshold: 30

AlarmActions:

- !Ref WebServerScaleDownPolicy

Dimensions:

- Name: AutoScalingGroupName
Value: !Ref WebServerASG

Auto Scaling Policies

WebServerScaleUpPolicy:

Type: AWS::AutoScaling::ScalingPolicy

Properties:

AdjustmentType: ChangeInCapacity
AutoScalingGroupName: !Ref WebServerASG
Cooldown: 300
ScalingAdjustment: 1

WebServerScaleDownPolicy:

Type: AWS::AutoScaling::ScalingPolicy
Properties:
AdjustmentType: ChangeInCapacity
AutoScalingGroupName: !Ref WebServerASG
Cooldown: 300
ScalingAdjustment: -1

Outputs:

VPC:

Description: VPC ID
Value: !Ref VPC

PublicSubnets:

Description: Public subnet IDs
Value: !Join [",", [!Ref PublicSubnet1, !Ref PublicSubnet2]]

PrivateSubnets:

Description: Private subnet IDs
Value: !Join [",", [!Ref PrivateSubnet1, !Ref PrivateSubnet2]]

ALBDNSName:

Description: Application Load Balancer DNS Name
Value: !GetAtt ApplicationLoadBalancer.DNSName

RDS Endpoint:

Description: RDS Instance Endpoint
Value: !GetAtt RDSInstance.Endpoint.Address

AutoScalingGroupName:

Description: Auto Scaling Group Name
Value: !Ref WebServerASG

3. Testing from ALB



4. Testing terminating ec2 to see if ASG works

Activity history (4)

Filter activity history

Status	Description	Cause	Start time	End time
Successful	Launching a new EC2 instance: i-03245ab334c8fb43c	At 2025-03-11T00:38:08Z an instance was launched in response to an unhealthy instance needing to be replaced.	2025 March 11, 09:38:10 AM +09:00	2025 March 11, 09:38:16 AM +09:00
Successful	Terminating EC2 instance: i-047ac8f6940565d5b	At 2025-03-11T00:38:08Z an instance was taken out of service in response to an EC2 health check indicating it has been terminated or stopped.	2025 March 11, 09:38:08 AM +09:00	2025 March 11, 09:38:22 AM +09:00
Successful	Launching a new EC2 instance: i-00592341bcfb2a354	At 2025-03-11T00:13:52Z a user request update of AutoScalingGroup constraints to min: 2, max: 4, desired: 2 changing the desired capacity from 0 to 2. At 2025-03-11T00:14:02Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2025 March 11, 09:14:04 AM +09:00	2025 March 11, 09:14:11 AM +09:00
Successful	Launching a new EC2 instance: i-047ac8f6940565d5b	At 2025-03-11T00:13:52Z a user request update of AutoScalingGroup constraints to min: 2, max: 4, desired: 2 changing the desired capacity from 0 to 2. At 2025-03-11T00:14:02Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2025 March 11, 09:14:04 AM +09:00	2025 March 11, 09:14:11 AM +09:00

5. cloudwatch alarm

Alarms (2)

Hide Auto Scaling alarms

Clear selection

Create composite alarm

Actions

Create

Search

Alarm state: Any

Alarm type: Any

Actions status: Any

1

Name	State	Last state update (Local)	Conditions	Actions
ha-web-app-stack-CPUAlarmLow-fEnBJUToshKb	In alarm	2025-03-11 09:18:21	CPUUtilization < 30 for 2 datapoints within 10 minutes	Actions enabled
ha-web-app-stack-CPUAlarmHigh-CA7A0DAePQ3W	OK	2025-03-11 09:15:26	CPUUtilization > 70 for 2 datapoints within 10 minutes	Actions enabled

All done.

Appendix

Manual implementation

Manual Implementation Guide - High Availability Web Application

Phase 1: VPC and Network Setup

```
graph TD
  A[VPC Creation] → B[Subnet Creation]
  B → C[Internet Gateway]
  C → D[NAT Gateway]
  D → E[Route Tables]
```

1. Create VPC

1. Go to VPC Console
2. Click "Create VPC"
3. Configure:

Name: ha-web-app-vpc
IPv4 CIDR: 10.0.0.0/16
Tenancy: Default
Tags:
- Name: ha-web-app-vpc

2. Create Subnets

Create 4 subnets:

Public Subnet 1:
Name: ha-web-app-public-1
AZ: ap-northeast-1a
CIDR: 10.0.1.0/24
Auto-assign public IPv4: Yes

Public Subnet 2:

Name: ha-web-app-public-2

AZ: ap-northeast-1c

CIDR: 10.0.2.0/24

Auto-assign public IPv4: Yes

Private Subnet 1:

Name: ha-web-app-private-1

AZ: ap-northeast-1a

CIDR: 10.0.3.0/24

Auto-assign public IPv4: No

Private Subnet 2:

Name: ha-web-app-private-2

AZ: ap-northeast-1c

CIDR: 10.0.4.0/24

Auto-assign public IPv4: No

3. Internet Gateway

1. Create Internet Gateway:

Name: ha-web-app-igw

2. Attach to VPC

4. NAT Gateways

Create two NAT Gateways:

NAT Gateway 1:

Name: ha-web-app-nat-1

Subnet: Public Subnet 1

Elastic IP: Create New

NAT Gateway 2:

Name: ha-web-app-nat-2

Subnet: Public Subnet 2

Elastic IP: Create New

5. Route Tables

Create and configure route tables:

Public Route Table:

Name: ha-web-app-public-rt

Routes:

- 0.0.0.0/0 → Internet Gateway

Associations:

- Public Subnet 1
- Public Subnet 2

Private Route Table 1:

Name: ha-web-app-private-rt-1

Routes:

- 0.0.0.0/0 → NAT Gateway 1

Associations:

- Private Subnet 1

Private Route Table 2:

Name: ha-web-app-private-rt-2

Routes:

- 0.0.0.0/0 → NAT Gateway 2

Associations:

- Private Subnet 2

Phase 2: Security Groups Setup

graph TD

A[Create Security Groups] → B[ALB Security Group]

B → C[Web Server Security Group]

C → D[RDS Security Group]

D → E[Configure Rules]

1. Create Security Groups

Go to EC2 Console > Security Groups > Create Security Group

ALB Security Group:

Name: ha-web-app-alb-sg

Description: Security group for Application Load Balancer

VPC: ha-web-app-vpc

Inbound rules:

- Type: HTTP (80)

Source: 0.0.0.0/0

Tags:

- Name: ha-web-app-alb-sg

Web Server Security Group:

Name: ha-web-app-web-sg

Description: Security group for web servers

VPC: ha-web-app-vpc

Inbound rules:

- Type: HTTP (80)

Source: ALB Security Group

Tags:

- Name: ha-web-app-web-sg

RDS Security Group:

Name: ha-web-app-rds-sg

Description: Security group for RDS

VPC: ha-web-app-vpc

Inbound rules:

- Type: MySQL/Aurora (3306)

Source: Web Server Security Group

Tags:

- Name: ha-web-app-rds-sg

Phase 3: RDS Setup

graph TD

A[Create Subnet Group] → B[Create Parameter Group]

B → C[Create RDS Instance]

1. Create DB Subnet Group

1. Go to RDS Console
2. Subnet groups > Create DB Subnet Group

Name: ha-web-app-db-subnet-group
Description: Subnet group for HA web app
VPC: ha-web-app-vpc
Availability Zones:

- ap-northeast-1a
- ap-northeast-1c

Subnets:

- Private Subnet 1
- Private Subnet 2

2. Create RDS Instance

1. Go to RDS Console > Create database

Creation method: Standard
Engine: MySQL
Version: 8.0.28
Templates: Production
Settings:

- DB instance identifier: ha-web-app-db
- Master username: admin
- Master password: [Create secure password]

Instance configuration:

- Instance class: db.t3.micro

Storage:

- Storage type: General Purpose SSD (gp2)
- Allocated storage: 20 GB

Availability & durability:

- Multi-AZ deployment: Yes

Connectivity:

- VPC: ha-web-app-vpc
- Subnet group: ha-web-app-db-subnet-group
- Security group: ha-web-app-rds-sg

Public access: No
Database authentication:
Password authentication
Additional configuration:
Initial database name: appdb
Backup retention: 7 days
Enable encryption: Yes

Updated: We can not use db.t3.micro in Production environment, choose Dev/Test and multi-AZ (2 instances Primary/standby) instead

Revised RDS Setup

graph TD
A[Create Subnet Group] → B[Create RDS Instance]
B → C[Configure Multi-AZ]
C → D[Configure Monitoring]

1. Create DB Subnet Group (Same as before)

1. Go to RDS Console
2. Subnet groups > Create DB Subnet Group

Name: ha-web-app-db-subnet-group
Description: Subnet group for HA web app
VPC: ha-web-app-vpc
Availability Zones:

- ap-northeast-1a
- ap-northeast-1c

Subnets:

- Private Subnet 1
- Private Subnet 2

2. Create RDS Instance (Revised for Dev/Test)

1. Go to RDS Console > Create database

Creation method: Standard create

Engine options:

Engine type: MySQL

Version: MySQL 8.0.28

Templates:

Dev/Test

Settings:

DB instance identifier: ha-web-app-db

Master username: admin

Master password: [Create secure password]

Instance configuration:

Instance class: Burstable classes (includes t classes)

Select: db.t3.micro

Storage:

Storage type: General Purpose SSD (gp2)

Allocated storage: 20 GB

Enable storage autoscaling: Yes

Maximum storage threshold: 100 GB

Availability & durability:

Multi-AZ deployment: Enable Multi-AZ DB instance

Note: This option is available even with db.t3.micro for MySQL

Connectivity:

Compute resource: Don't connect to EC2

Network type: IPv4

VPC: ha-web-app-vpc

DB Subnet group: ha-web-app-db-subnet-group

Public access: No

VPC security group: Choose existing

- Select: ha-web-app-rds-sg

Availability Zone: No preference

Database port: 3306

Authentication:

Database authentication: Password authentication

Monitoring:

Enhanced monitoring: Disabled (to save costs)

Enable Performance Insights: No (to save costs)

Additional configuration:

Initial database name: appdb

Backup:

Retention period: 7 days

Backup window: No preference

Maintenance:

Auto minor version upgrade: Enable

Maintenance window: No preference

Deletion protection: Enable

3. Cost-Saving Considerations

Cost-optimization features:

- Using db.t3.micro instance class
- General Purpose SSD (gp2) storage
- Disabled Enhanced Monitoring
- Disabled Performance Insights

Still maintaining HA features:

- Multi-AZ deployment
- Automated backups
- Storage autoscaling

4. Post-Creation Verification

Check:

1. Instance status
2. Multi-AZ configuration
3. Security group attachments
4. Subnet group configuration

5. Connectivity from EC2 instances

Test:

1. Connect from EC2 instance:

```
mysql -h <rds-endpoint> -u admin -p
```

2. Verify database creation:

```
SHOW DATABASES;
```

```
USE appdb;
```

Phase 4: Load Balancer Setup

graph TD

A[Create Target Group] → B[Create ALB]

B → C[Configure Listener]

1. Create Target Group

1. Go to EC2 Console > Target Groups > Create target group

Target type: Instances

Target group name: ha-web-app-tg

Protocol: HTTP

Port: 80

VPC: ha-web-app-vpc

Health check settings:

Protocol: HTTP

Path: /health

Port: traffic-port

Healthy threshold: 2

Unhealthy threshold: 5

Timeout: 5

Interval: 30

2. Create Application Load Balancer

1. Go to EC2 Console > Load Balancers > Create load balancer

Type: Application Load Balancer
Name: ha-web-app-alb
Scheme: internet-facing
IP address type: ipv4
Network mapping:
VPC: ha-web-app-vpc
Mappings:
- ap-northeast-1a: Public Subnet 1
- ap-northeast-1c: Public Subnet 2
Security groups:
- ha-web-app-alb-sg
Listeners:
- Protocol: HTTP
Port: 80
Default action: Forward to ha-web-app-tg

Phase 5: Launch Template & Auto Scaling Group

graph TD
A[Create IAM Role] → B[Create Launch Template]
B → C[Create Auto Scaling Group]

1. Create IAM Role for EC2

1. Go to IAM Console > Roles > Create role

Trusted entity: EC2
Permissions:
- AmazonSSMManagedInstanceCore
- CloudWatchAgentServerPolicy
Name: ha-web-app-ec2-role

2. Create Launch Template

1. Go to EC2 Console > Launch Templates > Create launch template

Name: ha-web-app-launch-template

Description: Launch template for HA web app

Auto Scaling guidance: Checked

Template content:

AMI: Amazon Linux 2 AMI

Instance type: t2.micro

Network settings:

- Security groups: ha-web-app-web-sg

IAM role: ha-web-app-ec2-role

Advanced details:

User data:

```
#!/bin/bash
yum update -y
yum install -y httpd mysql
systemctl start httpd
systemctl enable httpd
echo "Healthy" > /var/www/html/health
echo "<h1>Hello from HA Web App</h1>" > /var/www/html/index.html
```

```
# Install CloudWatch agent
```

```
yum install -y amazon-cloudwatch-agent
```

```
cat > /opt/aws/amazon-cloudwatch-agent/bin/config.json << 'EOF'
```

```
{
  "metrics": {
    "metrics_collected": {
      "cpu": {
        "measurement": ["cpu_usage_idle", "cpu_usage_user", "cpu_usage_
system"]
      },
      "mem": {
        "measurement": ["mem_used_percent"]
      },
      "disk": {
        "measurement": ["disk_used_percent"],
        "resources": ["/"]
      }
    }
  }
}
```

```
}  
EOF  
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl  
-a fetch-config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/c  
onfig.json  
systemctl start amazon-cloudwatch-agent  
systemctl enable amazon-cloudwatch-agent
```

Phase 6: Auto Scaling Group Setup

```
graph TD  
  A[Create ASG] --> B[Configure Scaling Policies]  
  B --> C[Setup CloudWatch Alarms]  
  C --> D[Configure Notifications]
```

1. Create Auto Scaling Group

1. Go to EC2 Console > Auto Scaling Groups > Create Auto Scaling group

Step 1 - Choose launch template:

Name: ha-web-app-asg

Launch template: ha-web-app-launch-template

Version: Latest

Step 2 - Choose instance launch options:

VPC: ha-web-app-vpc

Availability Zones and subnets:

- Private Subnet 1
- Private Subnet 2

Step 3 - Configure advanced options:

Load balancing:

- Attach to an existing load balancer
- Choose from your target groups
- Select: ha-web-app-tg

Health checks:

- ELB health check
- Grace period: 300 seconds

Step 4 - Configure group size:

Desired capacity: 2

Minimum capacity: 2

Maximum capacity: 4

Step 5 - Configure scaling policies:

Enable dynamic scaling policies: Yes

2. Create Scaling Policies

1. Select your Auto Scaling group
2. Go to Automatic scaling tab > Create dynamic scaling policy

Note:

Correct Order for Setting Up Auto Scaling and CloudWatch Alarms

1. First: Create Scaling Policies in Auto Scaling Group

graph TD

A[Go to EC2 Console] → B[Auto Scaling Groups]

B → C[Select your ASG]

C → D[Automatic Scaling tab]

D → E[Create dynamic scaling policy]

1. Go to EC2 Console > Auto Scaling Groups
2. Select your Auto Scaling Group
3. Go to "Automatic Scaling" tab
4. Click "Create dynamic scaling policy"

Create Scale Out Policy:

Policy type: Simple scaling

Name: Scale-Out-Policy

Take the action:

Add: 1 capacity units

And wait: 300 seconds

Click: Create

Create Scale In Policy:

Policy type: Simple scaling

Name: Scale-In-Policy

Take the action:

Remove: 1 capacity units

And wait: 300 seconds

Click: Create

2. Then: Create CloudWatch Alarms

After creating the scaling policies, now you can create the alarms:

1. Go to CloudWatch > Alarms > Create Alarm

Select Metric:

AWS Namespaces: AWS/EC2

By Auto Scaling Group

Select: CPUUtilization

Select your ASG name

Specify metric and conditions:

Statistic: Average

Period: 5 minutes

Threshold type: Static

Condition: Greater/Less than

Define threshold value: 70/30

Configure actions:

Select an Auto Scaling action

- You should now see your ASG and its scaling policies

Select appropriate scaling policy:

- Scale-Out-Policy for high CPU
- Scale-In-Policy for low CPU

The key point is:

- Scaling policies must exist before creating CloudWatch alarms
- The Auto Scaling actions become available in CloudWatch only after policies are created

Scale Out Policy:

Policy type: Simple scaling

Name: Scale-Out-CPU-High

Execute policy when:

Create new alarm:

Metric: CPUUtilization

Statistic: Average

Time period: 5 minutes

Threshold type: Static

Threshold value: 70

Datapoints to alarm: 2

Evaluation periods: 2

Take action:

Add: 1 capacity unit

Wait: 300 seconds between scaling activities

Scale In Policy:

Policy type: Simple scaling

Name: Scale-In-CPU-Low

Execute policy when:

Create new alarm:

Metric: CPUUtilization

Statistic: Average

Time period: 5 minutes

Threshold type: Static

Threshold value: 30

Datapoints to alarm: 2

Evaluation periods: 2

Take action:

Remove: 1 capacity unit
Wait: 300 seconds between scaling activities

Phase 7: CloudWatch Configuration

graph TD
A[Create Dashboard] → B[Setup Metrics]
B → C[Configure Alarms]
C → D[Setup Logs]

1. Create CloudWatch Dashboard

1. Go to CloudWatch Console > Dashboards > Create dashboard

Dashboard name: ha-web-app-dashboard

Add widgets:

1. EC2 Metrics:

- CPUUtilization
- NetworkIn/Out
- DiskReadOps/WriteOps

2. Load Balancer Metrics:

- RequestCount
- TargetResponseTime
- HTTPCode_Target_2XX_Count
- HTTPCode_Target_5XX_Count

3. RDS Metrics:

- CPUUtilization
- FreeableMemory
- DatabaseConnections

2. Create Additional CloudWatch Alarms

ALB 5XX Error Alarm:

Metric: HTTPCode_Target_5XX_Count

Threshold: > 10 in 5 minutes
Action: Create SNS notification

RDS CPU Alarm:

Metric: CPUUtilization
Threshold: > 80% for 5 minutes
Action: Create SNS notification

Target Group Health Alarm:

Metric: HealthyHostCount
Threshold: < 2 for 1 minute
Action: Create SNS notification

Phase 8: Testing and Validation

1. Load Balancer Testing:

- Access ALB DNS name
- Verify "Hello from HA Web App" page
- Test health check endpoint (/health)

2. Auto Scaling Testing:

- Terminate one EC2 instance
- Verify new instance is launched
- Check load balancer distribution

3. Database Connectivity:

- SSH into EC2 instance
- Test MySQL connection
- Verify Multi-AZ setup

4. Monitoring Validation:

- Check CloudWatch metrics
- Verify alarm configurations
- Test notification delivery

Phase 9: Final Checks

Security:

- Verify security group configurations
- Check network access paths
- Validate IAM roles and permissions

High Availability:

- Confirm resources across AZs
- Verify Auto Scaling functionality
- Test failover scenarios

Monitoring:

- Confirm CloudWatch metrics collection
- Verify alarm configurations
- Check dashboard visibility

Additional step for cleanup:

Cleanup Process for High Availability Web Application

graph TD

```
A[Start Cleanup] → B[Auto Scaling]
B → C[Load Balancer]
C → D[EC2 Resources]
D → E[RDS]
E → F[Network Resources]
F → G[IAM Resources]
```

1. Auto Scaling Resources

1. Auto Scaling Group:

- Go to EC2 > Auto Scaling Groups
- Select your ASG: ha-web-app-asg
- Actions > Delete
- Wait for instances to terminate

2. Launch Template:

- Go to EC2 > Launch Templates
- Select: ha-web-app-launch-template
- Actions > Delete template

2. Load Balancer Resources

1. Load Balancer:

- Go to EC2 > Load Balancers
- Select: ha-web-app-alb
- Actions > Delete

2. Target Group:

- Go to EC2 > Target Groups
- Select: ha-web-app-tg
- Actions > Delete

3. RDS Cleanup

1. RDS Instance:

- Go to RDS > Databases
- Select: ha-web-app-db
- Actions > Delete
- Uncheck "Create final snapshot"
- Check "I acknowledge..."
- Type "delete me" for confirmation
- Delete

2. DB Subnet Group:

- Go to RDS > Subnet groups

- Select: ha-web-app-db-subnet-group
- Delete

4. CloudWatch Resources

1. Alarms:

- Go to CloudWatch > Alarms
- Select all related alarms
- Actions > Delete

2. Dashboard:

- Go to CloudWatch > Dashboards
- Select: ha-web-app-dashboard
- Actions > Delete

5. Security Groups

Delete in this order:

1. RDS Security Group (ha-web-app-rds-sg)
2. Web Server Security Group (ha-web-app-web-sg)
3. ALB Security Group (ha-web-app-alb-sg)

Go to EC2 > Security Groups:

- Select each security group
- Actions > Delete security group

6. Network Resources

Delete in this order:

1. NAT Gateways:

- Go to VPC > NAT Gateways
- Delete both NAT gateways
- Wait for deletion to complete

2. Elastic IPs:

- Go to EC2 > Elastic IPs

- Release associated EIPs

3. Subnet Associations:

- Remove route table associations

4. Route Tables:

- Delete custom route tables:
 - Private route tables
 - Public route table

5. Subnets:

- Delete all 4 subnets

6. Internet Gateway:

- Detach from VPC
- Delete IGW

7. VPC:

- Finally delete the VPC

7. IAM Resources

1. IAM Role:

- Go to IAM > Roles
- Search for 'ha-web-app'
- Delete EC2 role and any related roles

2. Instance Profile:

- Will be deleted with role

Verification Checklist

Verify deletion of:

- ☐ Auto Scaling Group and instances
- ☐ Launch Template
- ☐ Load Balancer and Target Group
- ☐ RDS instance and subnet group

- ☐ CloudWatch alarms and dashboard
- ☐ Security Groups
- ☐ NAT Gateways and EIPs
- ☐ Route Tables
- ☐ Subnets
- ☐ Internet Gateway
- ☐ VPC
- ☐ IAM Roles