🚀High Availability Web Application on AWS

A **production-grade, highly available, self-healing web application** deployed on AWS using **EC2, Docker, Application Load Balancer (ALB), and Auto Scaling Group (ASG)**.

This project demonstrates **real DevOps practices**, including fault tolerance, auto-scaling, containerization, and AWS networking.

Project Objective

To deploy a web application that: - Remains **available even if EC2 instances or containers fail** - Automatically **heals itself without manual intervention** - Scales across **multiple Availability Zones** - Follows **production best practices**

🏗️Architecture Overview

```
User
 ↓
Application Load Balancer (HTTP :80)
 ↓
Target Group (Health Check :80)
 ↓
Auto Scaling Group
 ↓
EC2 Instances (Multiple AZs)
 ↓
Docker Container (Web App :80)
```

Key Characteristics

- Multi-AZ deployment
- Zero downtime
- Automatic recovery
- Container-based application

🛖AWS Services Used

- **Amazon EC2** – Compute instances
- **Auto Scaling Group (ASG)** – Instance scaling & self-healing
- **Application Load Balancer (ALB)** – Traffic distribution & health checks
- **Target Group** – Backend instance registration
- **Security Groups** – Network access control
- **Docker** – Application containerization

🐳Docker Setup

Dockerfile (Example)

```
FROM nginx:alpine
COPY index.html /usr/share/nginx/html/index.html
```

Run Container (Production)

```
docker run -d --restart always -p 80:80 username/my-web-app
```

🔹 `--restart always` ensures container auto-recovery even after manual stop or reboot.

EC2 User Data Script (Final & Stable)

This script runs automatically when a new EC2 instance is launched by ASG.

```
#!/bin/bash
set -xe

apt update -y
apt install -y docker.io

systemctl start docker
systemctl enable docker

sleep 20

docker pull username/my-web-app
docker run -d --restart always -p 80:80 username/my-web-app
```

Why this works

- Prevents Docker startup race conditions
- Fully automated container startup
- Enables instance-level self-healing

🚚Base64 Encoding (macOS)

AWS CLI requires User Data to be Base64 encoded.

```
base64 -i userdata.sh
```

Launch Template Strategy

- Launch Template **versions are immutable**
- Every change creates a **new version**

Example evolution: - v1 – Initial setup - v2 – Port fix - v3 – Docker restart policy - v4 – User Data stability fix

ASG always launches instances using the **configured version**.

🚡 Auto Scaling & Self-Healing Flow

1. ASG launches EC2 using Launch Template
2. User Data installs Docker & runs container
3. EC2 registers with Target Group
4. ALB health checks pass
5. Traffic is routed automatically

Failure Handling

Failure	Recovery
EC2 terminated	ASG launches new EC2
AZ failure	ASG launches in another AZ
Container crash	Docker restarts container

Common Errors Faced & Fixes

EC2 Not Launching

Issue: Instance type not Free Tier eligible
Fix: Use `t2.micro` or `t3.micro`

Target Group Unhealthy

Issue: Port mismatch (85 vs 80)
Fix: Align ALB, Target Group, EC2, and container on port **80**

Target State = `unused`

Issue: ALB listener not forwarding traffic
Fix: Update ALB listener to forward to correct Target Group

Website Works on EC2 but Not via ALB

Issue: EC2 SG not allowing traffic from ALB SG
Fix: Allow inbound port 80 from **ALB Security Group**

Container Not Restarting

Issue: Used `--restart unless-stopped`
Fix: Use `--restart always`

Container Not Created on New EC2

Issue: Docker daemon not ready
Fix: Add `sleep 20`, create new Launch Template version, replace EC2

AWS CLI Errors (macOS)

- Smart quotes instead of ASCII quotes
- Using Load Balancer ARN instead of Listener ARN
- Wrong API for Target Group updates

Validation Checklist

After deployment, confirm: - EC2 instances are running - Target Group shows **healthy** - ALB DNS opens the website - `docker ps` shows running container - `curl http://localhost` works on EC2

How to Test Self-Healing

Test Container Recovery

```
docker stop <container_id>
```

Container restarts automatically.

Test Instance Recovery

```
aws ec2 terminate-instances --instance-ids <instance-id>
```

ASG launches a new EC2 automatically.

🚢Interview-Ready Summary

> "We deployed a Dockerized web application on EC2 instances managed by an Auto Scaling Group behind an Application Load Balancer. The system is fully self-healing, highly available, and production-ready."

🚀Future Enhancements

- HTTPS using ACM (443)
- CloudWatch alarms & scaling policies
- Blue-Green deployments
- CI/CD with Jenkins or GitHub Actions
- Migration to Amazon ECS / EKS

Author

Mayank Mathur
DevOps Engineer | AWS | Docker | CI/CD

🔲If this project helped you, consider starring the repository!