AWS Troubleshooting Project: Broken Web App

What happens when your web app is deployed but completely unreachable? In this project, I simulated that exact scenario by intentionally deploying a broken AWS environment, then strategically bringing it back to life.

I launched a CloudFormation stack in **us-east-1** that provisioned an Application Load Balancer (ALB), Target Group, Auto Scaling Group with EC2 instances, and Security Groups. Everything looked fine in the console, but the application was unreachable. This was the starting point of my troubleshooting exercise, much like a real-world situation where an application is live but inaccessible to users.

My Troubleshooting Approach

Instead of jumping into random fixes, I took a systematic approach:

- Start at the **load balancer** (entry point for users)
- Work through the **target group** and **health checks**
- Verify **security groups and network paths**
- Finally, confirm the **application layer** was serving traffic

This top-down process mirrors how I'd debug a real production issue: following the data path step by step until I isolate the break.

🟦 Issues Identified & Fixed

Issue 1: No Registered Targets

- The Target Group had 0 instances.
- Registered the EC2 instances on port 80 to ensure traffic could be routed.

Issue 2: Health Checks Failing

- The health check path was misconfigured as `/healthcheck.mhtl`.
- Updated it to /healthcheck.html, allowing the ALB to validate instance health.

Issue 3: Security Group Misconfiguration

- The EC2 Security Group only allowed SSH (22).
- Added inbound rule: HTTP (80) from the ALB Security Group.
- Result: ALB could successfully connect to the web servers.

Issue 4: Application Not Serving Content

- The ALB was mistakenly attached to private subnets.
- Updated ALB to use public subnets for internet access.
- Result: The ALB DNS now returned the application successfully.

✅ Results & Takeaway

- All targets in Target Group = **Healthy**
- The ALB DNS resolved successfully and served the application in the browser.

This project wasn't just about fixing configuration mistakes, it was about **thinking like a troubleshooter**. By simulating a broken environment, I practiced isolating issues layer by layer and applying the right fixes in the right order.

It reinforced that in cloud environments, issues are rarely isolated: sometimes the problem lies at the infrastructure level (wrong subnets), sometimes in networking (SG rules), and sometimes at the application layer (health checks, web server). Approaching problems systematically is key to restoring functionality quickly and with confidence.

I'm excited to keep building and sharing more AWS projects that highlight real-world problem solving! 🚀

#AWS #CloudComputing #DevOps #Troubleshooting #CloudFormation #EC2 #ElasticLoadBalancer