My First AWS Cloud Infrastructure Project Overview

Tarek Adel

June 01, 2025

1. Project Overview

This document provides an overview of my first AWS cloud infrastructure project, built from scratch following an architectural diagram. The project implements a scalable and highly available web application infrastructure on AWS, focusing on core services like networking, compute, database, and monitoring. Below is a summary of the implemented components and those that were not implemented.

1.1 Implemented Features

- **VPC and Subnets**: Created a VPC (MyVPC) with 3 Public Subnets and 3 Private Subnets across 3 Availability Zones (us-east-1a, us-east-1b, us-east-1c) for high availability.
- **Internet Gateway**: Attached an Internet Gateway (MyIGW) to the VPC and configured routing for Public Subnets.
- **NAT Gateway**: Deployed a NAT Gateway (MyNATGateway) in a Public Subnet to allow Private Subnets to access the internet securely.
- **Launch Template**: Created a Launch Template (MyWebTemplate) to define EC2 instance configurations, including an Apache web server installation via user data script.
- **Auto Scaling Group**: Set up an Auto Scaling Group (MyAutoScalingGroup) to launch 3 EC2 instances across 3 Availability Zones, ensuring high availability.
- **Application Load Balancer (ALB)**: Deployed an ALB (MyAppLB) with a Target Group (MyTargetGroup) to distribute traffic across EC2 instances.
- **Amazon RDS**: Deployed a MySQL RDS instance (mydb) in Private Subnets with Multi-AZ configuration for redundancy.
- **CloudWatch and SNS**: Configured CloudWatch Alarms to monitor EC2 CPU usage and send email notifications via SNS (EC2-Alerts) when thresholds are exceeded.

1.2 Features Not Implemented

- **IAM, Policies, Users, and Developer Roles**: I chose not to implement IAM configurations, including creating users, policies, or developer roles.
- **Dynamic Naming for EC2 Instances**: I did not implement dynamic naming (e.g., myweb1, myweb2) for EC2 instances using Tags or other methods.

- **Additional Customizations**: I did not proceed with features like Auto Scaling Policies, backend integration with RDS, or automation using Terraform/GitLab CI/CD.

2. Detailed Steps: Building the Project from Scratch

This section outlines the step-by-step process I followed to build the AWS infrastructure, as shown in the architecture diagram.

2.1 Step 1: Create VPC and Subnets

- Navigated to the VPC Dashboard in AWS Console.
- Created a VPC named `MyVPC` with CIDR block `10.0.0.0/16`.
- Created 3 Public Subnets:
  - `PublicSubnet1` in us-east-1a with CIDR `10.0.1.0/24`.
  - `PublicSubnet2` in us-east-1b with CIDR `10.0.2.0/24`.
  - `PublicSubnet3` in us-east-1c with CIDR `10.0.3.0/24`.
- Created 3 Private Subnets:
  - `PrivateSubnet1` in us-east-1a with CIDR `10.0.11.0/24`.
  - `PrivateSubnet2` in us-east-1b with CIDR `10.0.12.0/24`.
  - `PrivateSubnet3` in us-east-1c with CIDR `10.0.13.0/24`.

2.2 Step 2: Set Up Internet Gateway

- Created an Internet Gateway named `MyIGW` and attached it to `MyVPC`.
- Created a Route Table named `PublicRouteTable` for Public Subnets.
- Added a route for `0.0.0.0/0` pointing to `MyIGW`.
- Associated `PublicSubnet1`, `PublicSubnet2`, and `PublicSubnet3` with `PublicRouteTable`.

2.3 Step 3: Deploy NAT Gateway

- Created a NAT Gateway named `MyNATGateway` in `PublicSubnet1` with an Elastic IP.
- Created a Route Table named `PrivateRouteTable` for Private Subnets.
- Added a route for `0.0.0.0/0` pointing to `MyNATGateway`.
- Associated `PrivateSubnet1`, `PrivateSubnet2`, and `PrivateSubnet3` with `PrivateRouteTable`.

2.4 Step 4: Create Launch Template

- Navigated to EC2 > Launch Templates in AWS Console.
- Created a Launch Template named `MyWebTemplate` with:
  - AMI: Amazon Linux 2023 (`ami-03d8b47244d950bbb`).
  - Instance type: `t2.micro`.
  - Key pair: `MyKeyPair`.
  - Security Group: `WebSG` (allowing HTTP on port 80 and SSH on port 22).
  - User data script:

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "<h1>Hello from Auto Scaling EC2 - $(hostname)</h1>" > /var/www/html/in
```

2.5 Step 5: Create Auto Scaling Group

- Navigated to EC2 > Auto Scaling Groups.
- Created an Auto Scaling Group named `MyAutoScalingGroup`.
- Used `MyWebTemplate`.
- Configured network settings:
  - VPC: `MyVPC`.
  - Subnets: `PublicSubnet1`, `PublicSubnet2`, `PublicSubnet3` (across us-east-1a, us-east-1b, us-east-1c).
- Set desired, minimum, and maximum capacity to 3 to launch exactly 3 EC2 instances.
- Attached to an existing Application Load Balancer (ALB) via a Target Group.

2.6 Step 6: Set Up Application Load Balancer (ALB)

- Navigated to EC2 > Load Balancers.
- Created an Application Load Balancer named `MyAppLB` (Internet-facing).
- Created a Target Group named `MyTargetGroup` with HTTP protocol on port 80.
- Associated the ALB with `PublicSubnet1`, `PublicSubnet2`, and `PublicSubnet3`.
- Attached the Target Group to the Auto Scaling Group to distribute traffic across the 3 EC2 instances.

2.7 Step 7: Deploy Amazon RDS

- Navigated to RDS > Create database.
- Created a MySQL RDS instance named `mydb` (Free Tier, `db.t3.micro`).
- Configured with:
  - DB Subnet Group: `MyDBSubnetGroup` (spanning `PrivateSubnet1`, `PrivateSubnet2`, `PrivateSubnet3`).
  - Public access: No.
  - Security Group: `RDSSG` (allowing MySQL traffic on port 3306 from `WebSG`).

2.8 Step 8: Configure CloudWatch and SNS for Monitoring

- Navigated to SNS > Topics.
- Created an SNS Topic named `EC2-Alerts`.
- Subscribed my email to the SNS Topic and confirmed the subscription.
- Navigated to CloudWatch > Alarms.
- Created an alarm to monitor CPU utilization of EC2 instances:
  - Metric: `CPUUtilization`.
  - Threshold: Greater than 80% for 2 consecutive periods (10 minutes).
  - Notification: Send to `EC2-Alerts` SNS Topic.