CSCI 5411
Advanced Cloud Architecting

“Architecting Applications on AWS”

Term Project Report

Submitted By:

Name: Rushil Borad

Banner Id: B00977837

Email: rs519505@dal.ca

Table of Contents

1. Executive Summary 4
 1.1 Key Features of the Architecture 7
2. Detailed Architecture Design 8
 2.1 Application Selection 8
 2.2 Cloud Architecture Overview 9
  2.2.1 Architecture Diagram 10
  2.2.2 Summary of Data Flow 12
 2.3 Justification of AWS Services by Category 13
 3. Adherence to AWS Well-Architected Framework 14
4. Infrastructure of Code 16
 4.1 CloudFormation template and the architecture 16
 4.2 Successful Launch of cloud Formation Template 20
5. References 23

Table of Figures

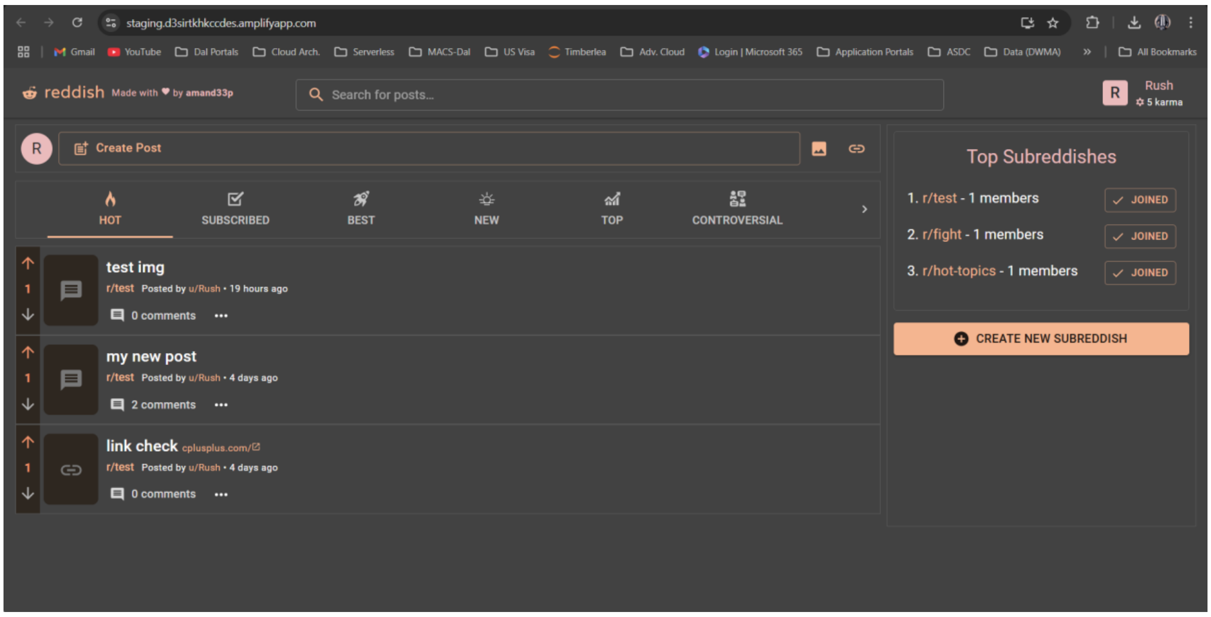
Figure 1: Hosted Application Frontend UI connected with backend. 4
Figure 2 : Application home page 5
Figure 3 : Add a new post page 5
Figure 4 : Search Functionality for posts on subreddits 6
Figure 5 : Individual post page with comments 6
Figure 6 : Final architecture diagram with complete implementation of services 10
Figure 7 : Successful launch of CloudFormation template (IAC) 20
Figure 8 : Services creation Amplify app and Application Load Balancer 20
Figure 9 : Creation complete for AttachGateway, Backend AutoScalingGroup, BackendLaunchTemplate, ImageBucket, InternetGateway, Listner for ALB 21
Figure 10 : Creation complete for NotificaitonsTopic, PrivateSecurityGroup, PrivateSubnet1, PrivateSubnet2, PublicRuote and Route table and Route table association 21
Figure 11 : Creation complete for target Group 22

1. Executive Summary

This project demonstrates the deployment of a scalable, secure, and cost-efficient Reddit clone application—**Reddish**—using the MERN stack, hosted entirely on AWS. The purpose of this project is to design and implement a robust cloud architecture designed to meet the application's requirements while also following to AWS best practices, particularly the principles outlined in the AWS Well-Architected Framework.

The deployed application allows users to interact seamlessly by creating posts, commenting, and uploading images, with the backend infrastructure ensuring high availability and fault tolerance. The application is accessible via the following public

Frontend Deployed URL: https://staging.d3sirtkhkccdes.amplifyapp.com .



test img
r/test Posted by u/Rush • 19 hours ago
0 comments

my new post
r/test Posted by u/Rush • 4 days ago
2 comments

link check cplusplus.com/
r/test Posted by u/Rush • 4 days ago
0 comments

1. r/test - 1 members JOINED
2. r/fight - 1 members JOINED
3. r/hot-topics - 1 members JOINED

Figure 1: Hosted Application Frontend UI connected with backend.

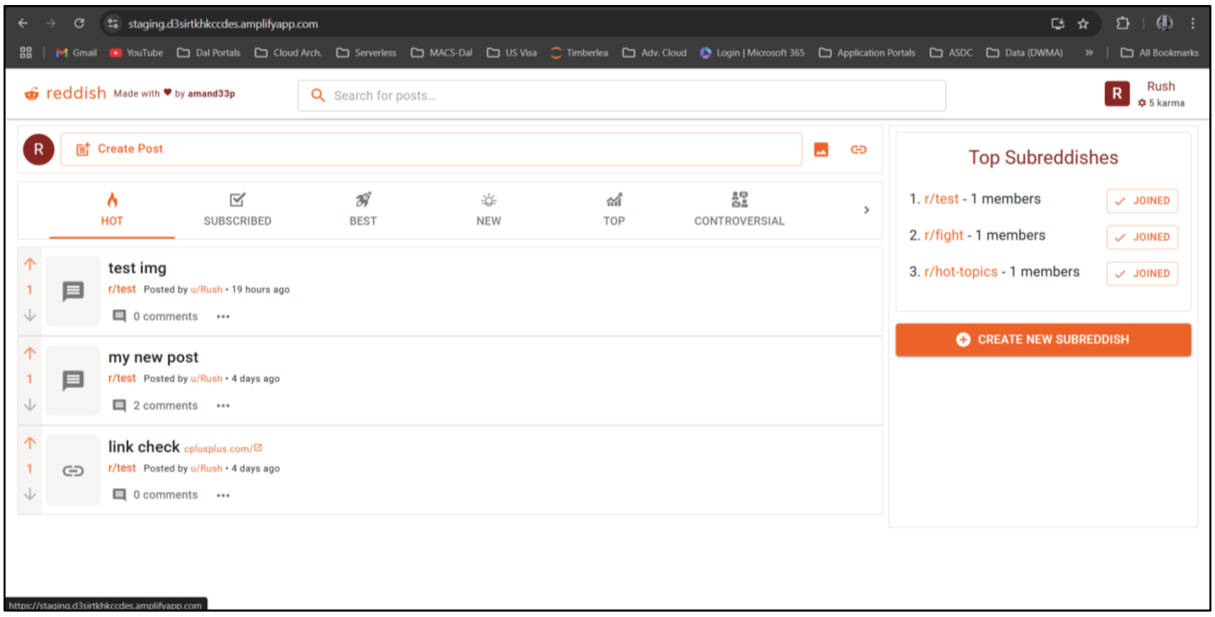

staging.d3sirtkhkccdes.amplifyapp.com
reddish Made with ♥ by amand33p
Search for posts...
Rush
5 karma
Create Post
HOT
SUBSCRIBED
BEST
NEW
TOP
CONTROVERSIAL
test img
r/test Posted by u/Rush • 19 hours ago
0 comments
my new post
r/test Posted by u/Rush • 4 days ago
2 comments
link check cplusplus.com/
r/test Posted by u/Rush • 4 days ago
0 comments
Top Subreddishes
1. r/test - 1 members
2. r/fight - 1 members
3. r/hot-topics - 1 members
JOINED
CREATE NEW SUBREDDISH

Figure 2 : Application home page

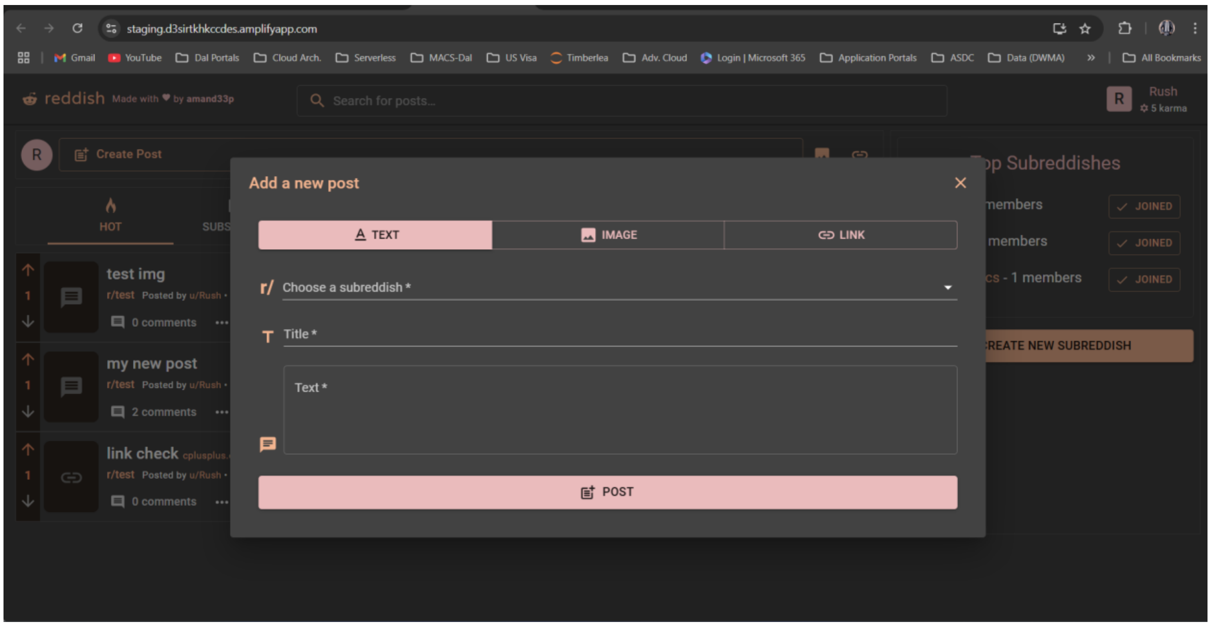
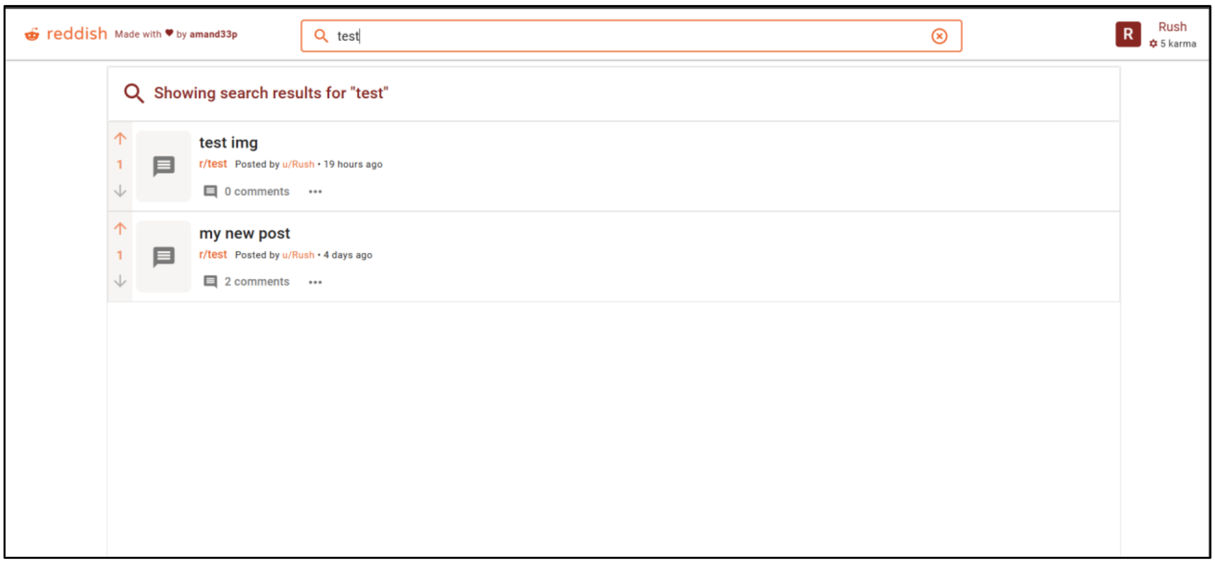


Figure 3 : Add a new post page

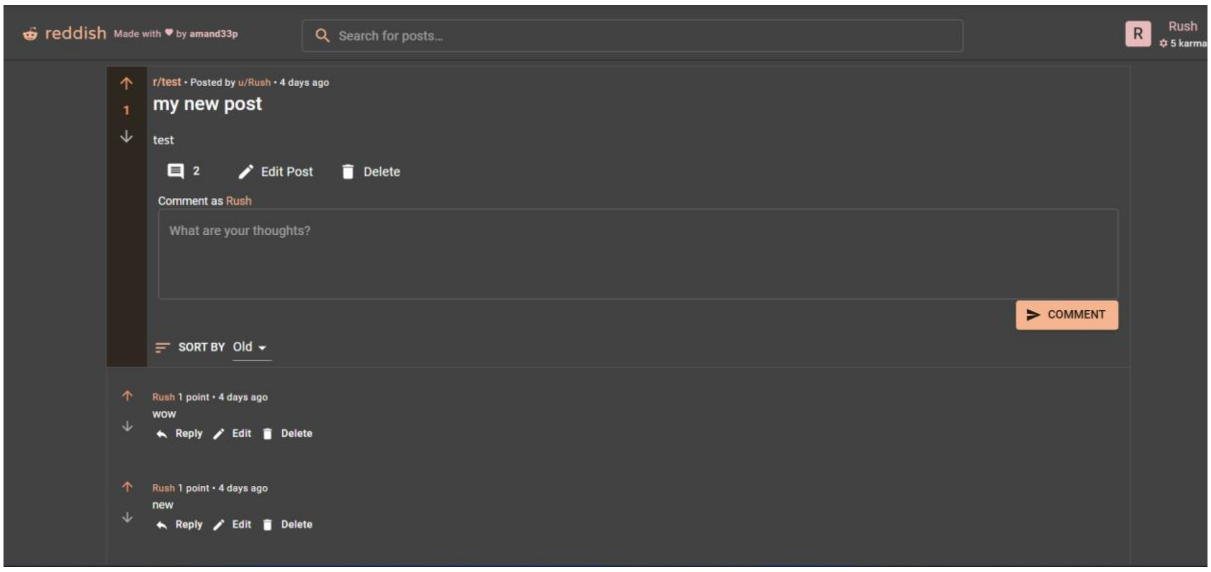


Showing search results for "test"

test img
r/test Posted by u/Rush • 19 hours ago
0 comments

my new post
r/test Posted by u/Rush • 4 days ago
2 comments

Figure 4 : Search Functionality for posts on subreddits



r/test • Posted by u/Rush • 4 days ago

my new post

test

2 Edit Post Delete

Comment as Rush

What are your thoughts?

COMMENT

SORT BY Old

Rush 1 point • 4 days ago
wow
Reply Edit Delete

Rush 1 point • 4 days ago
new
Reply Edit Delete

Figure 5 : Individual post page with comments

1.1 Key Features of the Architecture

1. **Scalability**:
   - The architecture is designed to handle varying levels of user traffic with an Auto Scaling Group for backend EC2 instances and Amplify-managed hosting for the frontend.
   - The use of Amazon SNS for notifications allows for reliable scaling of user engagement mechanisms.
2. **Cost-Effectiveness**:
   - Managed services like Amplify and S3 are chosen to minimize operational overhead and reduce costs for hosting static content and storing user-generated images.
3. **Performance**:
   - Amplify provides a global Content Delivery Network (CDN), ensuring low-latency access to frontend resources for users worldwide.
   - Backend and database components are deployed across multiple Availability Zones, ensuring minimal downtime and high responsiveness.
4. **Security**:
   - The architecture incorporates secure VPC design, segregating public-facing and private resources to mitigate risks.
   - IAM policies and roles are applied for least-privilege access to AWS resources, and HTTPS ensures secure communication.
5. **Adherence to AWS Best Practices**:
   - The architecture aligns with the six pillars of the AWS Well-Architected Framework, focusing on operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability.

2. Detailed Architecture Design

2.1 Application Selection

The chosen open-source application is **Reddish**, a Reddit clone developed with the MERN stack. This application aligns perfectly with the project's objectives due to its structured architecture and good reputation (100+ commits and 50+ stars).
The choice was guided by the following factors:

- **Complexity and Relevance**: Reddish demonstrates a modern web application architecture using frontend, backend, and database layers, making it ideal for deploying in a cloud environment.
- **Scalability and User Base**: The application can be scaled to support a growing user base, reflecting real-world scenarios for cloud solutions.
- **Customization**: It allows modifications to integrate AWS services for hosting, scaling, and notifications.

2.2 Cloud Architecture Overview

Category	**Chosen Service**
Compute	EC2 with Auto Scaling, Application Load Balancer (ALB) ECR: Stores containerized backend Docker images for seamless deployment.
Storage	Amazon S3
Networking and Content Delivery	Virtual Private Cloud (VPC) with public and private subnets, ALB
Application Integration	Simple Notification Service (SNS)
Management and Governance	CloudFormation, CloudWatch Logs
Developer Tools	AWS Amplify

2.2.1 Architecture Diagram

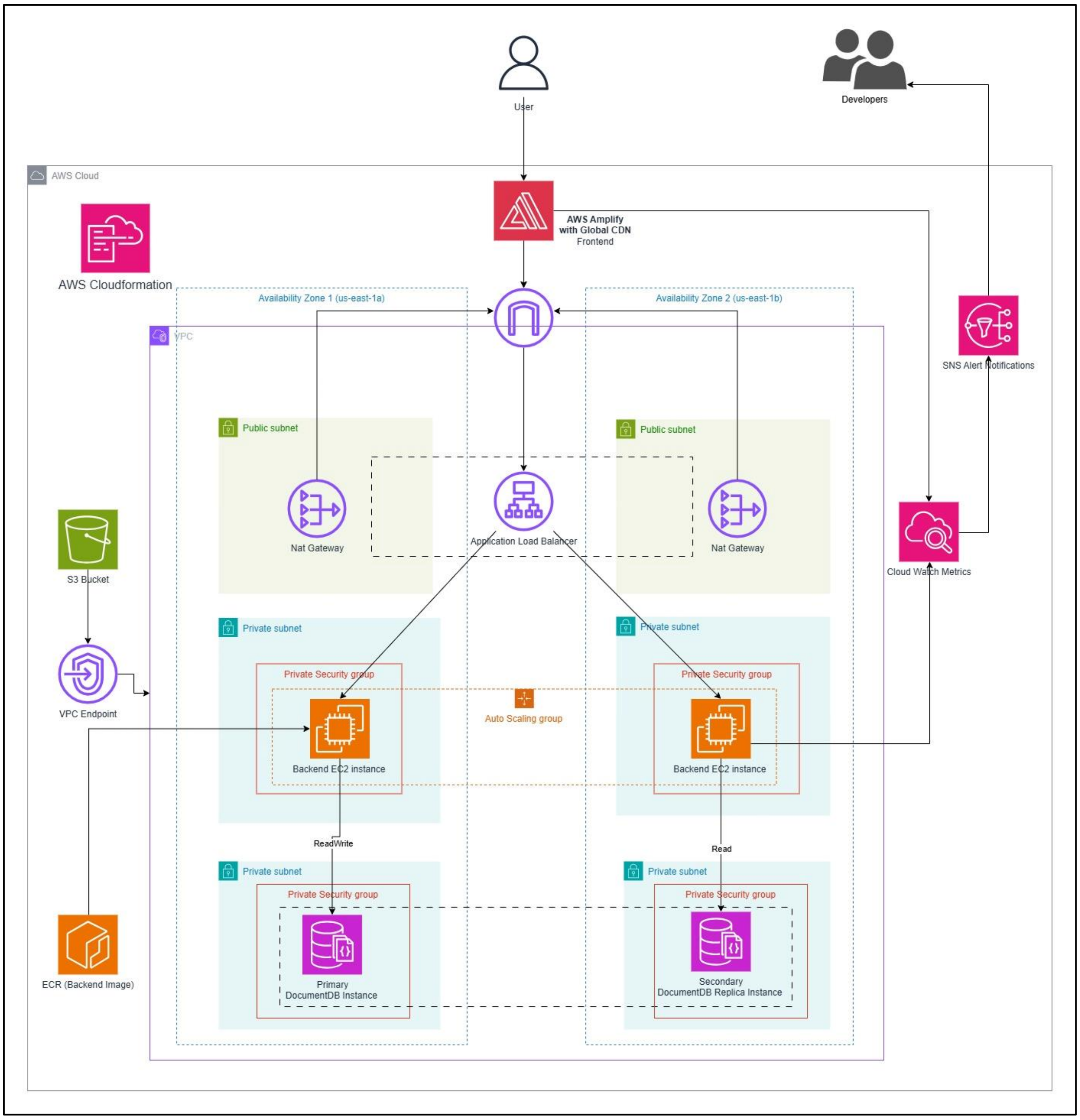


Figure 6 : Final architecture diagram with complete implementation of services

An overview of the architecture and its components:

1. **Frontend Layer**:
    - Hosted on **AWS Amplify**, which simplifies CI/CD for the React-based frontend and provides **global CDN** for low-latency delivery.
    - Amplify ensures the frontend is securely deployed with automatic **HTTPS** and scalability to handle user traffic spikes.
2. **Backend Layer**:
    - Deployed as **containerized instances** on EC2, with Docker images stored in **Amazon Elastic Container Registry (ECR)**.
    - Instances are managed by an **Auto Scaling Group** for dynamic scaling based on load.
    - **Application Load Balancer (ALB)** handles traffic distribution across EC2 instances, ensuring reliability and fault tolerance.
3. **Storage**:
    - **S3** Stores user-uploaded images (posts, avatars) **securely** and **cost-effectively**.
    - Appropriate **bucket policy** is attached to S3 to access the objects of the bucket
4. **Application Integration**:
    - **AWS Simple Notification Service (SNS)** is integrated for notifications to users, such as updates on posts or community activity.
5. **Networking and Content Delivery**:
    - A **VPC** with two public and two private subnets is designed for networking.
        - Public subnets host the ALB.
        - Private subnets host EC2 instances, securing backend and API communications.
    - The ALB ensures HTTP communication and balances traffic across private instances.
6. **Management and Governance**:
    - **CloudFormation** is used for Infrastructure as Code (IaC), enabling consistent and repeatable resource provisioning.
    - **CloudWatch Logs** monitor application performance and provide insights for troubleshooting and optimization.

7. **Developer Tools**:
    - **AWS Amplify** offers a seamless development experience, automating frontend deployment with integrated version control and build pipelines

Note on Database:

- **MongoDB Atlas**: Used for the project due to AWS Academy Lab limitations. It offers a managed database with high availability, security, and compatibility with Reddish.
- **DocumentDB (MongoDB Compatibility)**: Would replace MongoDB Atlas in a professional AWS environment for a fully AWS-integrated solution.

2.2.2 Summary of Data Flow

1. **Frontend** → Amplify-hosted React app handles user interactions and sends requests.
2. **API Gateway** → ALB routes requests to backend EC2 instances.
3. **Backend** → Processes requests, interacts with S3 for image storage, and MongoDB Atlas for database operations.
4. **Storage** → S3 stores image files, while MongoDB manages application data.
5. **Notifications** → SNS notifies stakeholders of key events.
6. **Scaling** → Auto Scaling ensures backend availability during traffic surges.

2.3 Justification of AWS Services by Category

2.3.1 Compute (EC2 with Auto Scaling and ALB)

- EC2 provides flexible and scalable compute capacity for the containerized backend.
- Auto Scaling dynamically adjusts the number of instances, optimizing costs and performance.
- ALB ensures even traffic distribution, improving fault tolerance and application reliability.

2.3.2 Storage (S3)

- Docker images for the backend are stored in Amazon ECR, ensuring high availability and integration with EC2 Auto Scaling for seamless deployments.

2.3.3 Networking and Content Delivery (VPC, ALB)

- The VPC isolates public and private resources, ensuring security while maintaining accessibility.
- The ALB provides secure HTTPS traffic termination and optimizes request routing across backend instances.

2.3.4 Application Integration (SNS)

- SNS simplifies notification delivery to users for updates, ensuring reliability and scalability.

2.3.5 Management and Governance (CloudFormation, CloudWatch)

- CloudFormation automates resource creation, enforcing consistent infrastructure provisioning.
- CloudWatch collects logs and metrics, enabling proactive monitoring and operational excellence.

2.3.6 Developer Tools (Amplify)

- AWS Amplify accelerates frontend development, integrating CI/CD pipelines and deployment automation for React applications.

3. Adherence to AWS Well-Architected Framework

3.1 Operational Excellence

- **Automation with CloudFormation**: The use of CloudFormation enables consistent infrastructure deployments, reducing manual configuration errors. IaC scripts ensure repeatability across environments.
- **Monitoring and Alerting with CloudWatch**: CloudWatch monitors resource usage and application logs, allowing proactive issue resolution and real-time performance insights.

3.2 Security

- **Network Isolation with VPC**: Backend EC2 instances are deployed in private subnets, inaccessible from the internet. The ALB in public subnets handles all traffic routing, ensuring secure communication with backend resources.

3.3 Reliability

- **Auto Scaling**: Ensures that the system dynamically adapts to changing demands, maintaining application availability during traffic spikes.
- **Multi-AZ Deployment**: Resources are distributed across multiple Availability Zones, minimizing downtime risks and improving fault tolerance.
- **Health Checks via ALB**: ALB performs regular health checks to route traffic only to operational backend instances, maintaining service reliability.

3.4 Performance Efficiency

- **Dynamic Scaling**: Auto Scaling adjusts the number of EC2 instances based on real-time traffic, optimizing performance and resource usage.
- **Global CDN with Amplify**: Amplify uses a global content delivery network to serve frontend assets with low latency, improving user experience.
- **Load Balancing**: ALB efficiently distributes traffic across backend instances, reducing latency and improving response times.

3.5 Cost Optimization

- **Elastic Resource Allocation**: Auto Scaling minimizes unnecessary resource usage by scaling down during low demand, reducing operational costs.

- **Amplify’s Managed Service**: By offloading CI/CD and hosting tasks to Amplify, the project avoids additional resource management overhead and costs.

3.6 Sustainability

- **Elastic Resource Usage**: Resources like EC2 and ALB are scaled dynamically to match demand, reducing waste and energy consumption.
- **Efficient Storage Management**: S3 lifecycle policies ensure optimal storage use, contributing to sustainable cloud resource usage.

4. Infrastructure of Code

4.1 CloudFormation template and the architecture

1. VPC and Networking

The foundation of our architecture is the **Virtual Private Cloud (VPC)**, which defines the isolated network environment where all our resources will reside.

- **VPC:**
    - Created with a CIDR block of 10.0.0.0/16 to provide ample private IP addresses.
    - DNS support and hostnames are enabled for resolving internal IPs.
- **Subnets:**
    - **Public Subnets:** Two subnets (10.0.1.0/24 and 10.0.2.0/24) span two Availability Zones for redundancy. These allow resources to communicate with the internet.
    - **Private Subnets:** Two subnets (10.0.3.0/24 and 10.0.4.0/24) also span two AZs. These are used for backend resources that don't require direct internet access.
- **Internet Gateway (IGW):**
    - Enables public internet connectivity for the VPC.
- **Route Tables:**
    - Public Route Table is associated with public subnets, allowing traffic to flow through the IGW.

2. Elastic Load Balancer (ALB)

This is the entry point for traffic, distributing incoming requests to backend EC2 instances.

- **ALB Configuration:**
    - Internet-facing, accessible to public traffic.
    - Subnets: Attached to both public subnets for redundancy.
    - Security Group: Allows HTTP (port 80) and API traffic (port 3005).
- **Target Group:**
    - Configured to route traffic to EC2 instances on port 3005, which runs the backend service.

    - Health checks ensure the instances are running and healthy. A custom health check path /api/health-check is used.
- **Listener:**
    - Listens on port 80 (HTTP) and forwards traffic to the target group.

3. Security Groups

Security groups control network traffic to and from the resources.

- **PublicSecurityGroup (ALB):**
    - Allows HTTP (port 80) and SSH (port 22) traffic from anywhere.
    - Ensures that the ALB can receive traffic from external users.
- **PrivateSecurityGroup (Backend):**
    - Restricts traffic to internal communication only (e.g., within the VPC or from specific sources).
    - Allows SSH (port 22) and backend service (port 3005) communication.

4. Backend EC2 Instances

The backend is hosted on Amazon EC2 instances running Docker containers.

- **Launch Template:**
    - EC2 AMI: Amazon Linux 2 is used.
    - Instance Type: Defaults to t3.micro, suitable for cost-effective deployments.
    - User Data: Automates setup with the following:
        - Updates packages, installs Docker, and logs into Amazon ECR.
        - Pulls the Docker image (BackendECRImageUri) for the backend service.
        - Runs the container with environment variables for MongoDB, S3 bucket, and AWS credentials.
- **Auto Scaling Group (ASG):**
    - Ensures high availability by maintaining 1-2 EC2 instances.
    - Automatically scales up or down based on demand.
    - Integrates with the ALB target group for health monitoring.

5. MongoDB Connection

The backend connects to MongoDB Atlas using the MONGODB_URI environment variable.

- **MongoDB Setup:**
    - Hosted externally (MongoDB Atlas).
    - Credentials are securely passed via environment variables.

6. Amplify Frontend

AWS Amplify is used for hosting and managing the frontend React application.

- **Amplify App:**
    - Custom rules redirect all frontend traffic to index.html to ensure proper SPA behavior.
    - An environment variable (REACT_APP_API_ENDPOINT) points to the backend API.
- **Amplify Branch:**
    - The production branch is linked to the Amplify app.

7. S3 for Images

A dedicated S3 bucket is used for storing user-uploaded images.

- **Bucket Configuration:**
    - Name: ReddishImageBucketName.
    - Public access is allowed for hosted images while ensuring other configurations (e.g., policies) are in place.
- **VPC Endpoint:**
    - Allows private communication between resources in the VPC and the S3 bucket.

8. Notifications via SNS

Simple Notification Service (SNS) is configured to send notifications (e.g., alerts).

- **Topic:**

    - Created with the name ReddishNotifications for subscription.

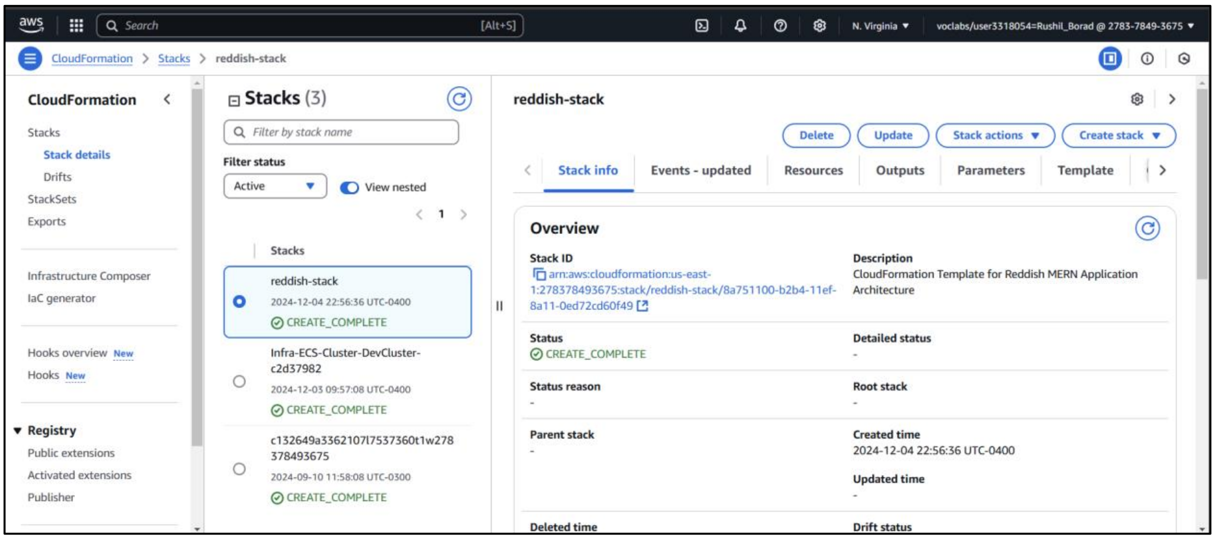
Outputs

Finally, outputs provide key information about the deployed infrastructure:

- **LoadBalancerDNSName:** DNS of the ALB to access the backend API.
- **AmplifyAppId:** ID of the Amplify app for reference.
- **AmplifyAppUrl:** Public URL of the frontend application.

4.2 Successful Launch of cloud Formation Template

Screenshots:



Stacks (3)

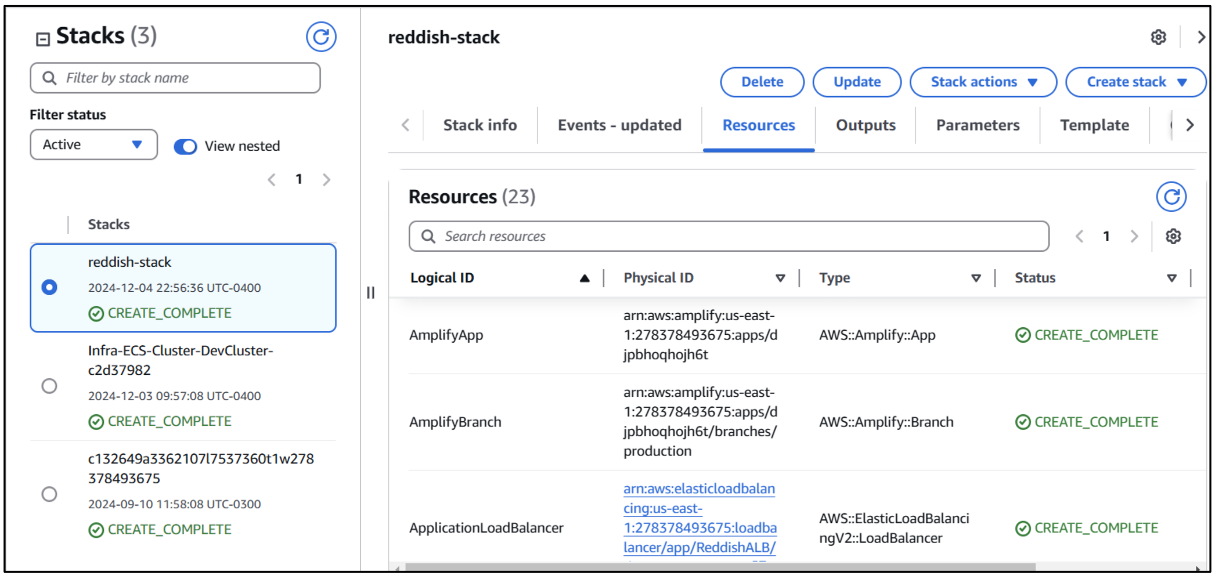
Stacks
reddish-stack 2024-12-04 22:56:36 UTC-0400 CREATE_COMPLETE
Infra-ECS-Cluster-DevCluster-c2d37982 2024-12-03 09:57:08 UTC-0400 CREATE_COMPLETE
c132649a3362107l7537360t1w278378493675 2024-09-10 11:58:08 UTC-0300 CREATE_COMPLETE

reddish-stack

Overview

Stack ID arn:aws:cloudformation:us-east-1:278378493675:stack/reddish-stack/8a751100-b2b4-11ef-8a11-0ed72cd60f49	**Description** CloudFormation Template for Reddish MERN Application Architecture
Status CREATE_COMPLETE	**Detailed status** -
Status reason -	**Root stack** -
Parent stack -	**Created time** 2024-12-04 22:56:36 UTC-0400 **Updated time** -
Deleted time	**Drift status**

Figure 7 : Successful launch of CloudFormation template (IAC)



Stacks (3)

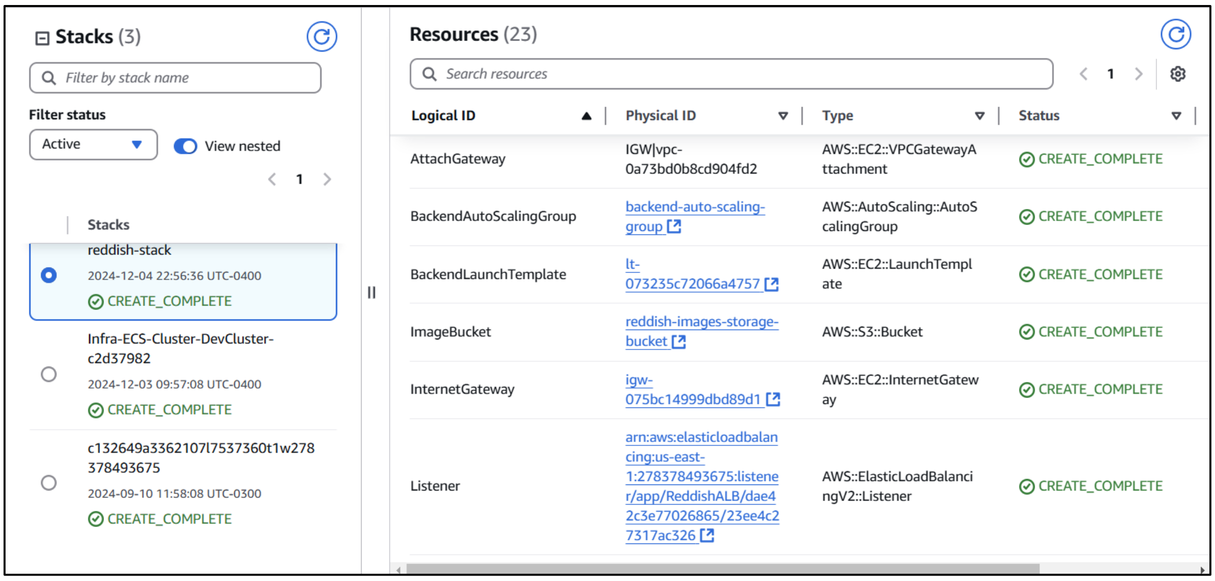
Stacks
reddish-stack 2024-12-04 22:56:36 UTC-0400 CREATE_COMPLETE
Infra-ECS-Cluster-DevCluster-c2d37982 2024-12-03 09:57:08 UTC-0400 CREATE_COMPLETE
c132649a3362107l7537360t1w278378493675 2024-09-10 11:58:08 UTC-0300 CREATE_COMPLETE

reddish-stack

Resources (23)

Logical ID	Physical ID	Type	Status
AmplifyApp	arn:aws:amplify:us-east-1:278378493675:apps/djpbhoqhojh6t	AWS::Amplify::App	CREATE_COMPLETE
AmplifyBranch	arn:aws:amplify:us-east-1:278378493675:apps/djpbhoqhojh6t/branches/production	AWS::Amplify::Branch	CREATE_COMPLETE
ApplicationLoadBalancer	arn:aws:elasticloadbalancing:us-east-1:278378493675:loadbalancer/app/ReddishALB/	AWS::ElasticLoadBalancingV2::LoadBalancer	CREATE_COMPLETE

Figure 8 : Services creation Amplify app and Application Load Balancer


Stacks (3)

Filter by stack name

Filter status: Active — View nested

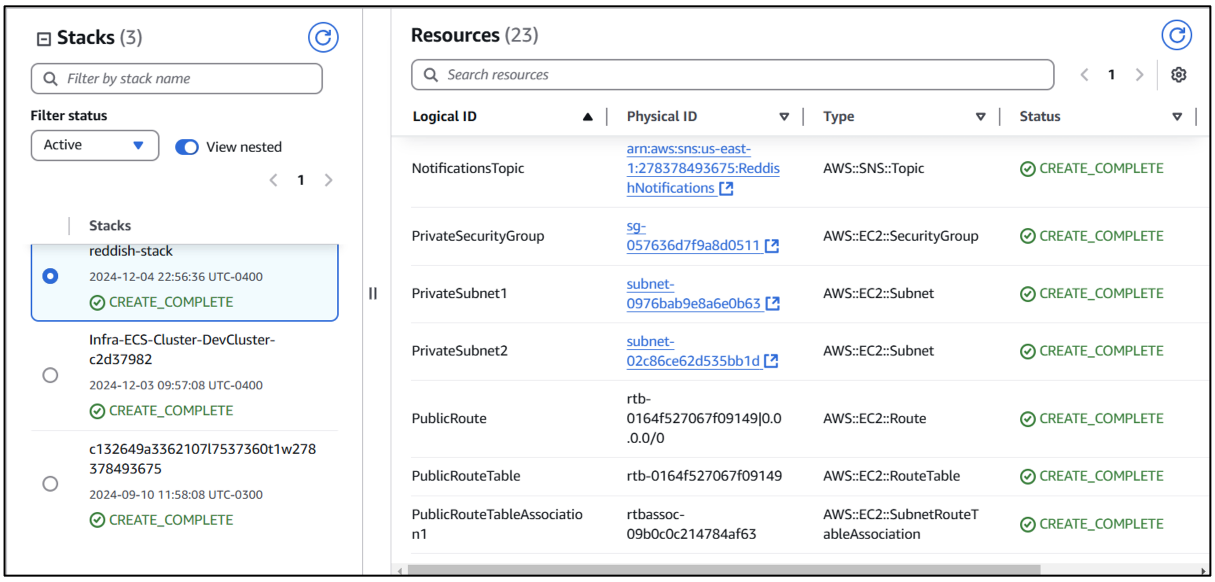
Stacks
reddish-stack 2024-12-04 22:56:36 UTC-0400 CREATE_COMPLETE
Infra-ECS-Cluster-DevCluster-c2d37982 2024-12-03 09:57:08 UTC-0400 CREATE_COMPLETE
c132649a3362107l7537360t1w278378493675 2024-09-10 11:58:08 UTC-0300 CREATE_COMPLETE

Resources (23)

Search resources

Logical ID	Physical ID	Type	Status
AttachGateway	IGW|vpc-0a73bd0b8cd904fd2	AWS::EC2::VPCGatewayAttachment	CREATE_COMPLETE
BackendAutoScalingGroup	backend-auto-scaling-group	AWS::AutoScaling::AutoScalingGroup	CREATE_COMPLETE
BackendLaunchTemplate	lt-073235c72066a4757	AWS::EC2::LaunchTemplate	CREATE_COMPLETE
ImageBucket	reddish-images-storage-bucket	AWS::S3::Bucket	CREATE_COMPLETE
InternetGateway	igw-075bc14999dbd89d1	AWS::EC2::InternetGateway	CREATE_COMPLETE
Listener	arn:aws:elasticloadbalancing:us-east-1:278378493675:listener/app/ReddishALB/dae42c3e77026865/23ee4c27317ac326	AWS::ElasticLoadBalancingV2::Listener	CREATE_COMPLETE

Figure 9 : Creation complete for AttachGateway, Backend AutoScalingGroup, BackendLaunchTemplate, ImageBucket, InternetGateway, Listner for ALB


Stacks (3)

Filter by stack name

Filter status: Active — View nested

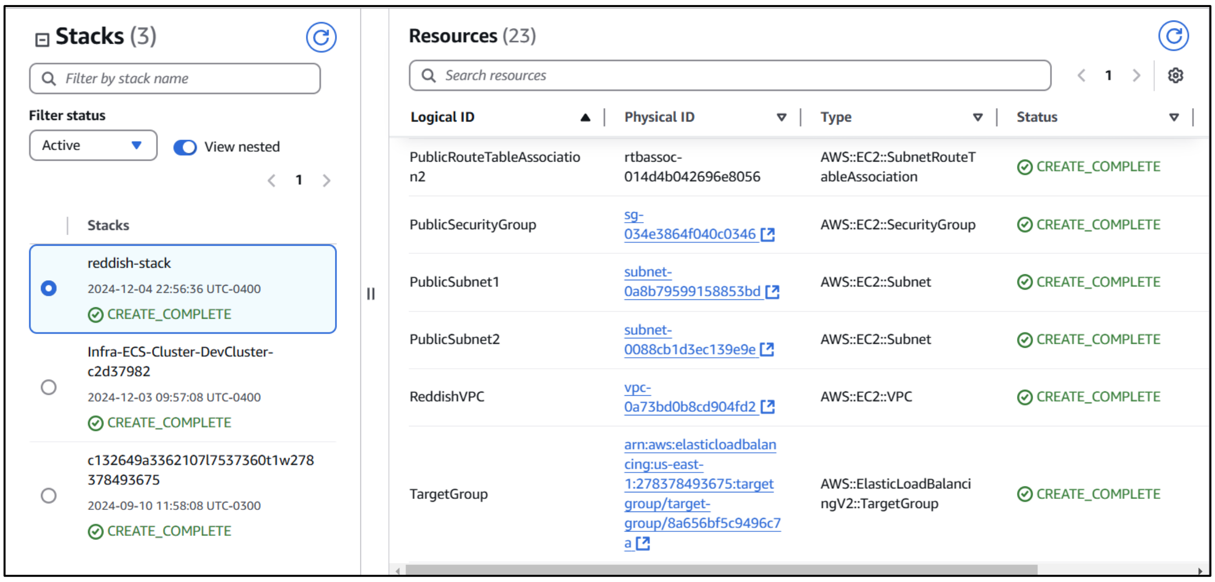
Stacks
reddish-stack 2024-12-04 22:56:36 UTC-0400 CREATE_COMPLETE
Infra-ECS-Cluster-DevCluster-c2d37982 2024-12-03 09:57:08 UTC-0400 CREATE_COMPLETE
c132649a3362107l7537360t1w278378493675 2024-09-10 11:58:08 UTC-0300 CREATE_COMPLETE

Resources (23)

Search resources

Logical ID	Physical ID	Type	Status
NotificationsTopic	arn:aws:sns:us-east-1:278378493675:ReddishNotifications	AWS::SNS::Topic	CREATE_COMPLETE
PrivateSecurityGroup	sg-057636d7f9a8d0511	AWS::EC2::SecurityGroup	CREATE_COMPLETE
PrivateSubnet1	subnet-0976bab9e8a6e0b63	AWS::EC2::Subnet	CREATE_COMPLETE
PrivateSubnet2	subnet-02c86ce62d535bb1d	AWS::EC2::Subnet	CREATE_COMPLETE
PublicRoute	rtb-0164f527067f09149|0.0.0.0/0	AWS::EC2::Route	CREATE_COMPLETE
PublicRouteTable	rtb-0164f527067f09149	AWS::EC2::RouteTable	CREATE_COMPLETE
PublicRouteTableAssociation1	rtbassoc-09b0c0c214784af63	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE

Figure 10 : Creation complete for NotificaitonsTopic, PrivateSecurityGroup, PrivateSubnet1, PrivateSubnet2, PublicRuote and Route table and Route table association



Stacks (3)

Filter status: Active

View nested

Stacks		
reddish-stack	2024-12-04 22:56:36 UTC-0400	CREATE_COMPLETE
Infra-ECS-Cluster-DevCluster-c2d37982	2024-12-03 09:57:08 UTC-0400	CREATE_COMPLETE
c132649a3362107l7537360t1w278378493675	2024-09-10 11:58:08 UTC-0300	CREATE_COMPLETE

Resources (23)

Logical ID	Physical ID	Type	Status
PublicRouteTableAssociation2	rtbassoc-014d4b042696e8056	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE
PublicSecurityGroup	sg-034e3864f040c0346	AWS::EC2::SecurityGroup	CREATE_COMPLETE
PublicSubnet1	subnet-0a8b79599158853bd	AWS::EC2::Subnet	CREATE_COMPLETE
PublicSubnet2	subnet-0088cb1d3ec139e9e	AWS::EC2::Subnet	CREATE_COMPLETE
ReddishVPC	vpc-0a73bd0b8cd904fd2	AWS::EC2::VPC	CREATE_COMPLETE
TargetGroup	arn:aws:elasticloadbalancing:us-east-1:278378493675:targetgroup/target-group/8a656bf5c9496c7a	AWS::ElasticLoadBalancingV2::TargetGroup	CREATE_COMPLETE

Figure 11 : Creation complete for target Group

5. References

[1] Amazon Web Services, "AWS Amplify," [Online]. Available: https://aws.amazon.com/amplify/ . [Accessed: Dec. 2 2024].

[2] Amazon Web Services, "Amazon EC2," [Online]. Available: https://aws.amazon.com/ec2/ . [Accessed: Dec. 3, 2024].

[3] Amazon Web Services, "Amazon S3," [Online]. Available: https://aws.amazon.com/s3/ . [Accessed: Dec. 3, 2024].

[4] Amazon Web Services, "Elastic Load Balancing," [Online]. Available: https://aws.amazon.com/elasticloadbalancing/ . [Accessed: Dec. 3, 2024].

[5] Amazon Web Services, "Amazon SNS," [Online]. Available: https://aws.amazon.com/sns/ . [Accessed: Dec. 4, 2024].

[6] Amazon Web Services, "Amazon CloudWatch," [Online]. Available: https://aws.amazon.com/cloudwatch/ . [Accessed: Dec. 3, 2024].

[7] Amazon Web Services, "AWS CloudFormation," [Online]. Available: https://aws.amazon.com/cloudformation/ . [Accessed: Dec. 4, 2024].

[8] Amazon Web Services, "AWS Well-Architected Framework," [Online]. Available: https://aws.amazon.com/architecture/well-architected/ . [Accessed: Dec. 4, 2024].

[9] MongoDB, Inc., "MongoDB Atlas," [Online]. Available: https://www.mongodb.com/cloud/atlas . [Accessed: Dec. 2, 2024].