Table of Contents

Project GitHub Link:
https://github.com/abdelkarimhajji/waiting_coder

Why this project:

The project at the provided GitHub link is a unique learning platform. Users can register, create profiles, access lessons, tools, and projects, engage in events and competitions, and there's an admin page for smooth management.

Deploying this project on AWS using the AWS Well-Architected Framework can provide several benefits:

1. **Operational Excellence**: The project has various components such as user registration, profile creation, lesson access, tool access, project access, event participation, and admin management. AWS provides services like AWS CloudFormation that can help automate these operational processes, reducing the risk of errors and increasing efficiency.
2. **Security**: User data security is crucial in this project as it involves handling user profiles, projects, events and competitions. AWS provides services like VPC, which can control the traffic into the application and secure the user data.
3. **Reliability**: The project needs to be reliable as users need to access lessons, tools, and projects anytime. AWS provides services like Amazon RDS, which are designed for high availability and durability.
4. **Performance Efficiency**: The project might need to handle a large number of users accessing various resources simultaneously. AWS provides services like AWS Auto Scaling and Load Balancers, which can help meet system requirements and maintain efficiency as demand changes.
5. **Cost Optimization**: As the project scales, cost optimization becomes important. AWS provides cost-optimized resources and you only pay for the services you use, which helps in reducing operating costs.
6. **Sustainability**: AWS is committed to running its business in the most environmentally friendly way possible. By deploying this project on AWS, I will be contributing to environmental sustainability.

Menu Requirements with AWS Service Categories:

Compute:

1. **EC2:** Hosts the Express.js backend application.
2. **EC2:** Hosts the React frontend application.
3. **Lambda:** Automates the creation of AMI images upon EC2 instance creation events.

Database:

1. **RDS:** Manages the MySQL database for the application.

Networking & Content Delivery:

1. **VPC:** Secures data at each layer.
2. **Elastic Load Balancing:**
    - **Public:** Distributes load for the frontend application.
    - **Internal:** Distributes load for the backend application.

Management & Governance:

1. **CloudFormation:** Implements Infrastructure as Code (IaC) for the entire infrastructure.
2. **Auto Scaling:** Manages resource provisioning according to changes in need and demand.
3. **CloudWatch:** For monitoring and logging on different services.
4. **AWS CloudTrail:** For monitoring and logging activities on infrastructure.

Security:

1. **IAM:**
    - Instance profiles for frontend and backend applications.
    - LabRole role associated with resources as required.

Justification for best fit:

1. Compute:
EC2 Instances:

- **Backend (Express.js) and Frontend (React):**
  - **Choice Justification:**
    - **Cost:** EC2 instances allow for flexible pricing models (on-demand, reserved, spot instances) which help manage costs based on usage [1].
    - **Performance:** EC2 provides scalable compute capacity, enabling the backend and frontend applications to handle varying loads efficiently.
    - **Security:** Integration with VPC and security groups ensures secure access and data protection.
    - **Scalability:** Auto Scaling groups manage the number of instances based on demand, ensuring high availability and performance.

Lambda Functions:

- **Automation of AMI Creation:**
  - **Choice Justification:**
    - **Cost:** Lambda functions are billed based on the number of executions and compute time, providing a cost-effective solution for automated tasks [2].
    - **Performance:** Executes tasks quickly and efficiently without the need for managing servers.
    - **Security:** IAM roles and policies ensure secure access to resources.
    - **Scalability:** Automatically scales based on the number of incoming requests.

2. Database:
Amazon RDS (MySQL):

- **Choice Justification:**
  - **Cost:** Amazon RDS offers various instance types and storage options, allowing cost optimization based on requirements [3].
  - **Performance:** Provides high availability and read replicas for improved performance and fault tolerance in different AZs.
  - **Security:** Offers built-in security features such as encryption at rest and in transit, and automated backups.
  - **Scalability:** Easily scales storage and compute resources based on demand.

3. Networking & Content Delivery:
VPC:

- **Choice Justification:**
  - **Cost:** No additional cost for using VPC, but it provides significant value in terms of security and network management [4].
  - **Performance:** Enables control over network traffic and ensures low latency communication between resources.
  - **Security:** Segregates resources into private and public subnets, providing a secure environment for sensitive data and applications.
  - **Scalability:** Supports adding more resources and subnets as needed.

Elastic Load Balancing:

- **Public and Internal Load Balancers:**
  - **Choice Justification:**
    - **Cost:** Pay-as-you-go pricing model based on the amount of data processed and the number of load balancer hours [5].
    - **Performance:** Distributes traffic efficiently across multiple instances, ensuring high availability and fault tolerance.
    - **Security:** Supports SSL termination, enhancing security for data in transit.
    - **Scalability:** Automatically scales to handle varying levels of incoming traffic.

4. Management & Governance:

AWS CloudFormation:

- **Choice Justification:**
  - **Cost:** No additional cost for using CloudFormation; it simplifies infrastructure management and deployment [6].
  - **Performance:** Automates the provisioning and updating of AWS resources, reducing manual efforts and errors.
  - **Security:** Templates can be version-controlled and reviewed for compliance with security best practices.
  - **Scalability:** Easily manages complex infrastructures and scales with the project's needs.

Auto Scaling:

- **Choice Justification:**
  - **Cost:** Adjusts the number of running instances based on demand, optimizing costs by scaling in when demand is low [7].
  - **Performance:** Ensures applications run smoothly by maintaining the desired performance levels during varying loads.
  - **Security:** Works seamlessly with IAM roles and policies to maintain secure operations.
  - **Scalability:** Automatically scales resources in and out based on defined policies and metrics.

CloudWatch:

- **Choice Justification:**
  - **Cost:** Pay-as-you-go pricing model based on the metrics and logs collected, providing cost-effective monitoring [8].
  - **Performance:** Provides comprehensive monitoring and logging capabilities, ensuring high visibility into system performance and operations.
  - **Security:** Offers secure access and storage for logs and metrics, with integration into AWS security services.
  - **Scalability:** Capable of monitoring resources at any scale, providing insights and alerting based on defined thresholds.

CloudTrail:

- **Choice Justification:**
  - **Cost:** Pay-as-you-go pricing model based on the number of events recorded, providing cost-effective auditing [9].
  - **Performance:** Captures detailed API call information across the AWS infrastructure, ensuring comprehensive logging.
  - **Security:** Enhances security by providing visibility into user activity, supporting compliance and auditing requirements.
  - **Scalability:** Scales with the infrastructure, providing detailed logs and insights regardless of the size of the deployment.

5. Security:

IAM (Instance Profiles and Roles):

- **Choice Justification:**
  - **Cost:** No additional cost for creating IAM roles and policies.
  - **Performance:** Ensures secure access to AWS resources with fine-grained permissions.
  - **Security:** Implements the principle of least privilege, reducing the risk of unauthorized access.
  - **Scalability:** Easily manage permissions for an increasing number of resources and users.

Adherence to AWS Well-Architected Framework Pillars:

1. Operational Excellence
Infrastructure as Code:

- **AWS CloudFormation:**
  - **Purpose:** My architecture leverages AWS CloudFormation for defining and deploying infrastructure as code (IaC).
  - **Benefits:** CloudFormation ensures consistent, repeatable, and automated provisioning of AWS resources. Templates define the architecture's configuration, including VPC settings, database, EC2 instances, load balancers, and more. This approach reduces manual intervention, minimizes human error, and facilitates version control of infrastructure changes [10].

Monitoring and Logging:

- **CloudWatch:**
  - **Integrated Monitoring:** CloudWatch monitors key performance metrics, application logs, and system-level metrics. It detects operational issues, tracks performance trends, and triggers alarms based on predefined thresholds.

How It Enhances Operational Excellence:

- **Automated Deployments:** Infrastructure changes and updates are managed through CloudFormation templates, ensuring consistency across deployments. This approach streamlines operations, improves resource management, and supports agile development practices.
- **Proactive Monitoring:** CloudWatch provides real-time insights into application and infrastructure health. Alarms notify administrators of performance deviations or operational issues, enabling prompt resolution and minimizing downtime.

2. Security
Logging and Monitoring:

- **CloudWatch and CloudTrail:**
  - **Purpose:** These services play crucial roles in monitoring and auditing activities within your AWS environment.
  - **CloudWatch:**
    - **Metrics Monitoring:** Collects and tracks metrics from various AWS services and resources, such as EC2 instances and load balancers. This visibility allows to monitor performance metrics and detect anomalies that may indicate security breaches or performance issues.
    - **Log Monitoring:** Aggregates log files from EC2 instances, including NGINX and Node.js servers, capturing application-level logs and system-level activities. This logging is critical for identifying unauthorized access attempts, system errors, or unusual activities that may pose security risks.
  - **CloudTrail:**
    - **API Call Logging:** Records API calls made on AWS account, providing a comprehensive trail of actions taken by users, applications, or AWS services.

- **Security Analysis:** By monitoring CloudTrail logs, I gain insights into who accessed the resources, from where, and when. This information is invaluable for auditing and compliance purposes, as well as for detecting and investigating security incidents [11].

Infrastructure Protection:

- **VPC and Security Groups:**
  - **VPC Segmentation:** Resources are segregated into private and public subnets within a custom VPC. This segmentation enhances security by isolating components based on their security requirements [12].
  - **Subnet Isolation:** Public-facing components like the load balancer reside in public subnets, accessible from the internet. Private components, such as EC2 instances, are located in private subnets, inaccessible from the internet directly.
  - **Security Groups:** Act as virtual firewalls, controlling inbound and outbound traffic to EC2 instances [12]. Security group rules are configured to allow only necessary traffic (e.g., HTTP, HTTPS) from designated sources (e.g., load balancers, specific IP ranges), minimizing exposure to potential threats.

3. Reliability

High Availability:

- **Multiple Availability Zones (AZs):**
  - **Deployment Strategy:** Resources, such as EC2 instances and databases, are distributed across multiple AZs to ensure fault tolerance and high availability. This architecture minimizes single points of failure and maintains application performance during AZ-level outages.

Health Checks and Auto-Healing:

- **Elastic Load Balancing and Auto Scaling:**
  - **Configuration:** ELB distributes incoming traffic across multiple EC2 instances, ensuring load distribution and redundancy. Auto Scaling dynamically adjusts instance capacity based on demand, scaling out during traffic spikes and scaling in during low demand periods [13].

Automated Backups:

- **RDS Backups:** Configured RDS auto backup option, Amazon RDS offers automated backups and snapshots. These features ensure data durability and facilitate quick recovery in case of data loss or corruption.

How It Ensures Reliability:

- **Fault Tolerance:** By spreading resources across AZs and leveraging ELB and Auto Scaling, my architecture maintains service availability and performance, even in the event of component failures or sudden increases in workload.
- **Continuous Availability:** Auto Scaling adjusts resource capacity based on traffic patterns, ensuring that my application can handle varying loads without manual intervention. This capability enhances reliability by maintaining consistent performance levels and responsiveness.

4. Performance Efficiency
Dynamic Scaling:

- **Elastic Load Balancing and Auto Scaling:**
  - **Performance Optimization:** ELB distributes traffic efficiently across instances, optimizing resource utilization and ensuring responsive application performance [13]. Auto Scaling adjusts instance capacity in response to changing demand, preventing over-provisioning and optimizing cost efficiency [14].

Efficient Resource Use:

- **Right-Sized EC2 Instances:**
  - **Performance Optimization:** Instances are selected based on workload requirements and performance metrics, avoiding over-provisioning and optimizing resource utilization. This approach ensures that resources are allocated efficiently, minimizing costs while meeting application performance needs.

How It Enhances Performance Efficiency:

- **Scalable Architecture:** By dynamically scaling resources and right-sizing instances, your architecture maximizes performance efficiency. This approach supports varying workload demands, maintains application responsiveness, and optimizes operational costs.

5. Cost Optimization
Monitoring Resource Usage:

- **AWS Cost Explorer:**
  - **Cost Management:** AWS Cost Explorer provides insights into resource usage and cost trends. It helps analyze spending patterns, identify cost drivers, and optimize resource allocation to reduce operational expenses [15].

Right-Sizing:

- **EC2 Instances:** Instances are selected based on workload requirements and performance metrics, ensuring optimal performance without overpaying for unused capacity.

Pay-As-You-Go:

- **Lambda and CloudFormation:** Services like Lambda and CloudFormation follow a pay-as-you-go pricing model [16], aligning costs with actual usage and demand. This flexible pricing structure enables cost-effective management of resources, particularly for fluctuating workloads.

How It Enhances Cost Optimization:

- **Cost Transparency:** AWS Cost Explorer provides visibility into resource spending, enabling informed decision-making and proactive cost management [15]. By right-sizing

instances and leveraging pay-as-you-go services, your architecture optimizes operational expenses while maintaining scalability and performance.

All in all, my architecture adheres to the AWS Well-Architected Framework's pillars by implementing robust security measures, leveraging automation for operational excellence, ensuring high reliability through fault-tolerant design, optimizing performance efficiency with dynamic scaling, and prioritizing cost optimization through efficient resource management and pay-as-you-go pricing models. These principles collectively support the scalability, security, and efficiency of my cloud infrastructure.

Architecture Diagram:

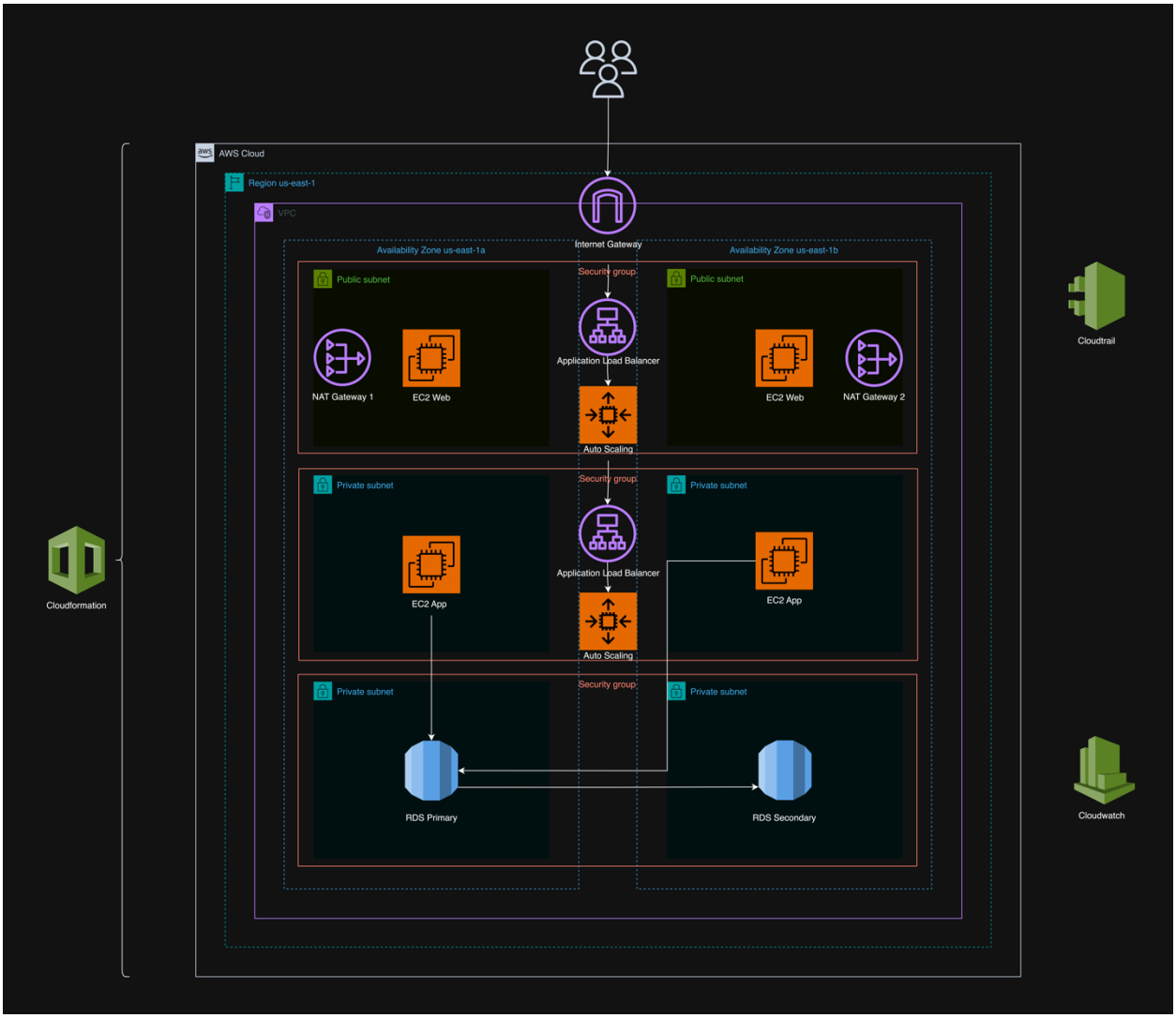

Figure 1: Architecture Diagram

Public URL and Evidence:

URL: http://term-end-web-public-alb-621261604.us-east-1.elb.amazonaws.com/

App Evidence:

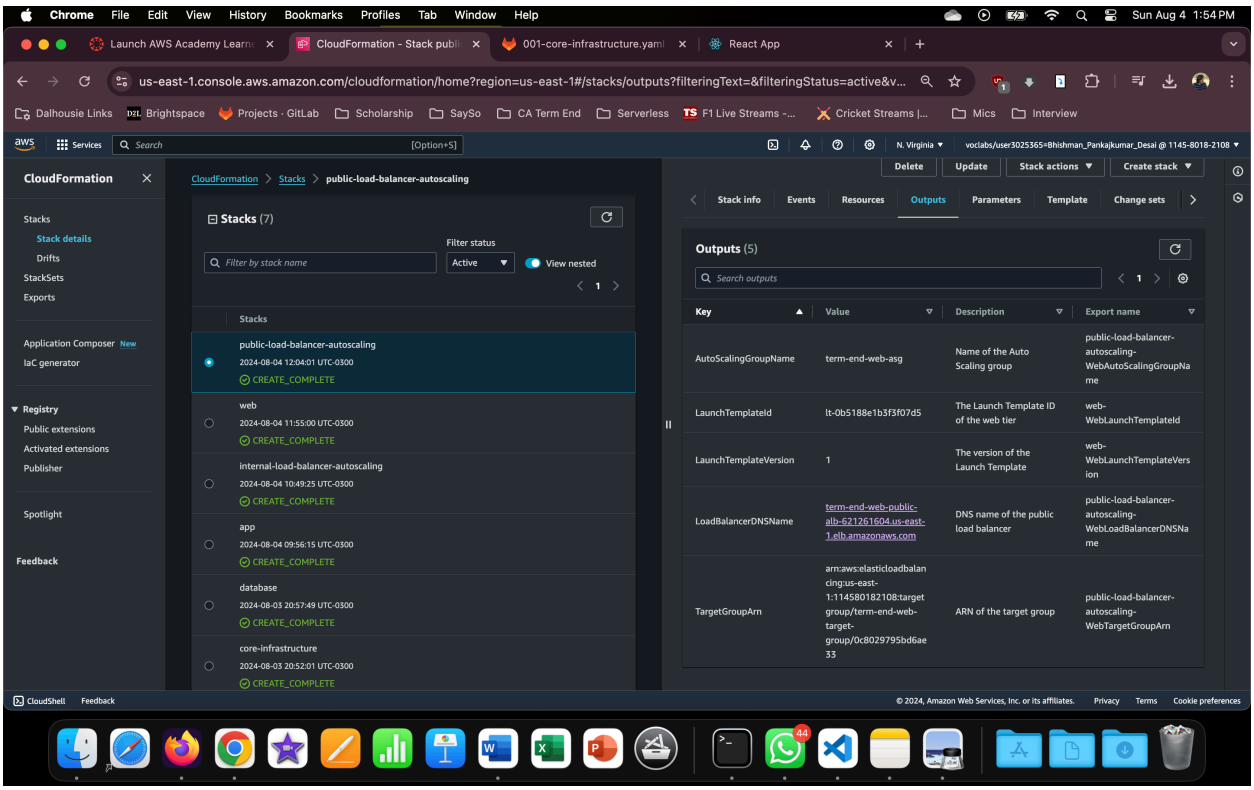

Figure 2: Stack completion

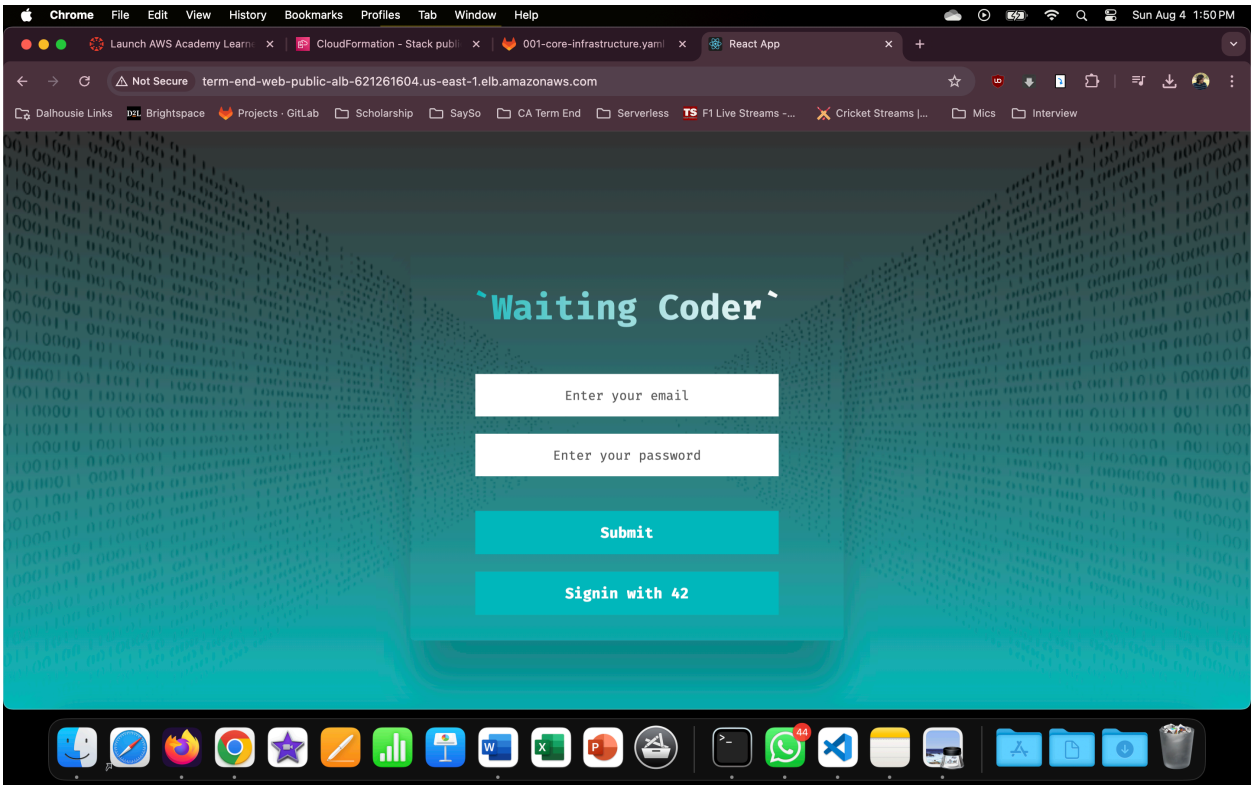

Figure 3: Login Page

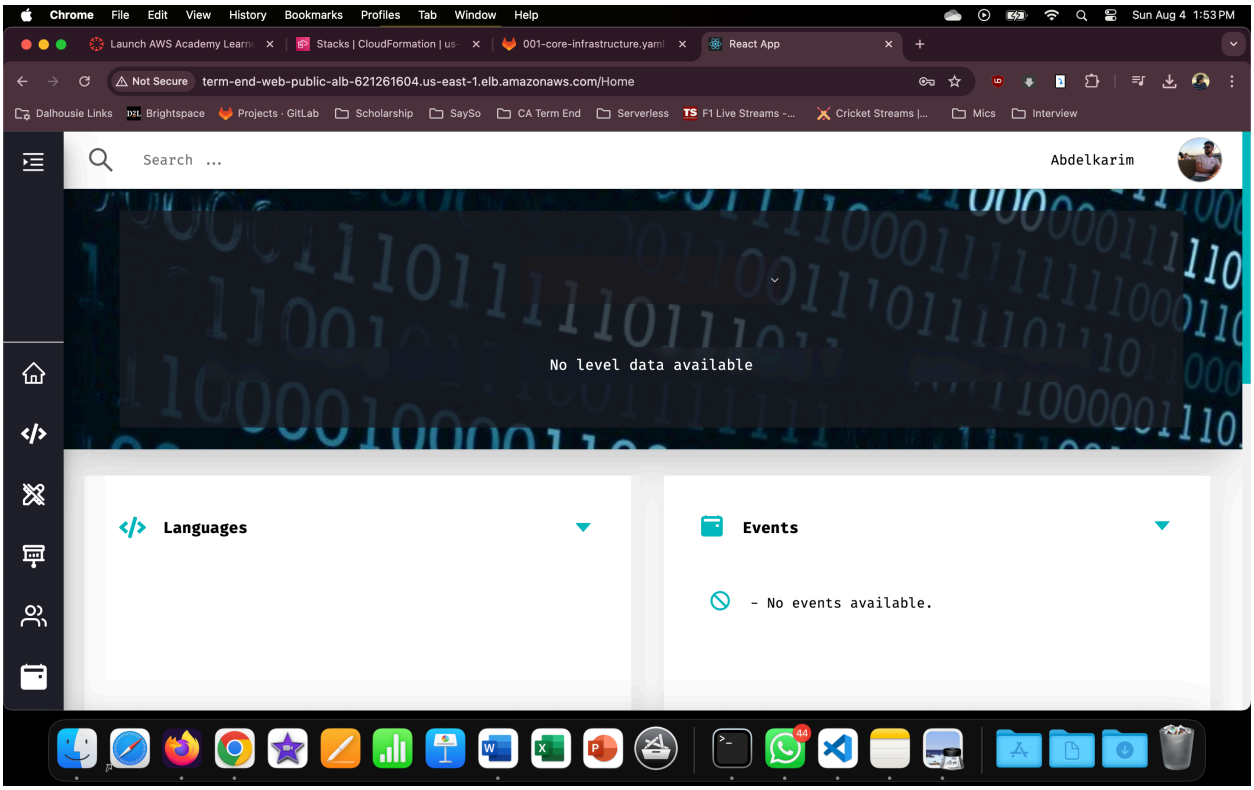

Figure 4: Homepage

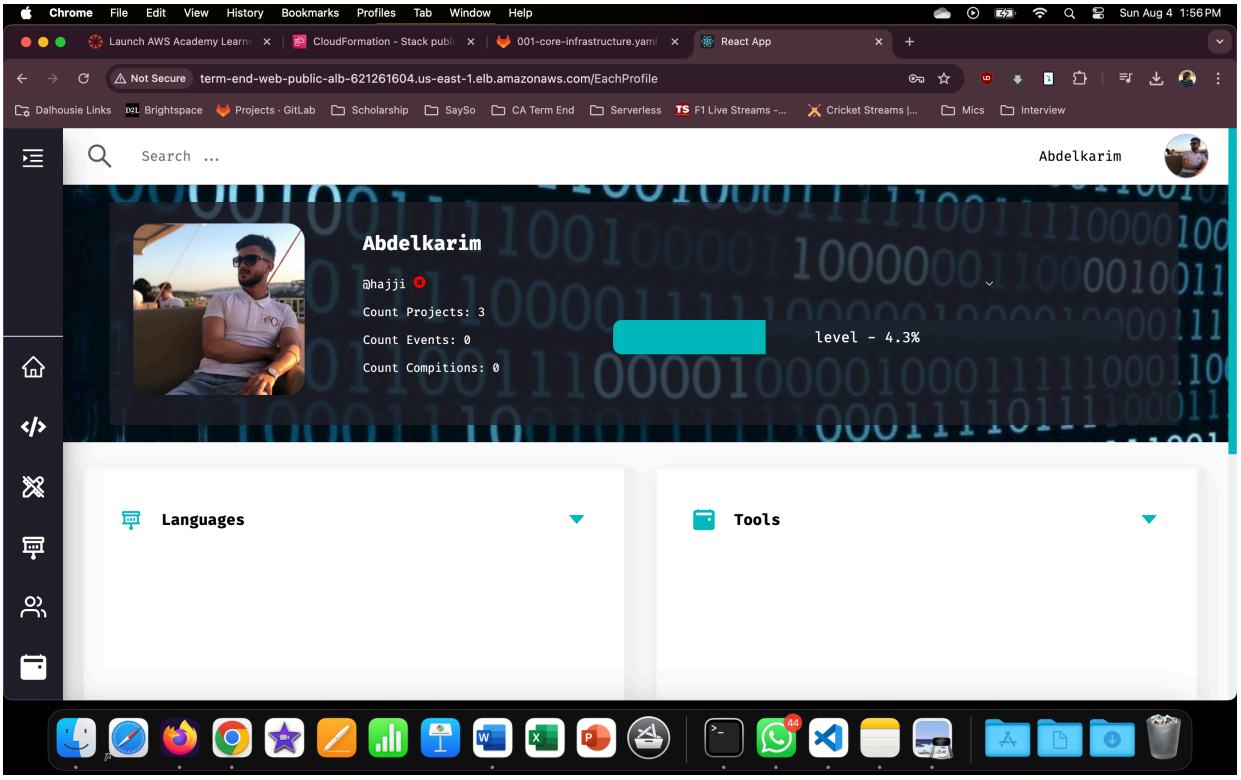

Figure 5: Profile Page

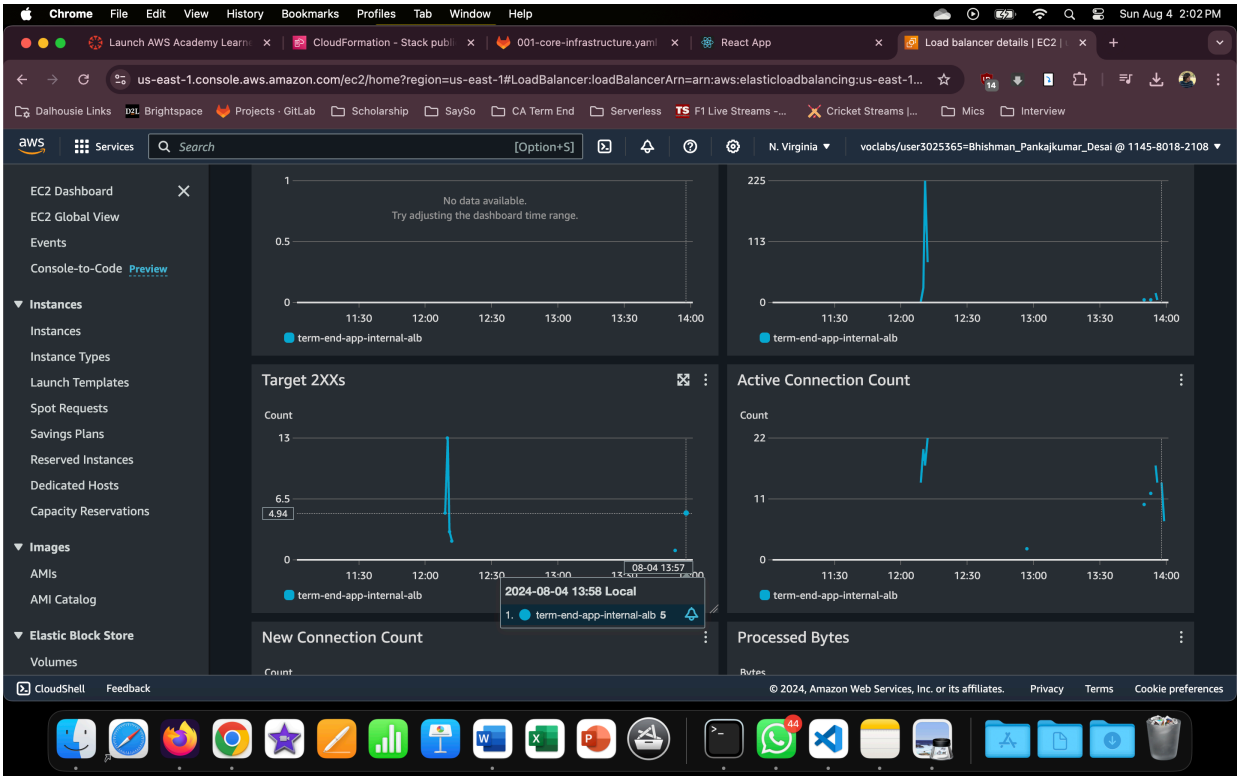

Figure 6: Frontend making a call at internal load balancer

Resources Evidence:

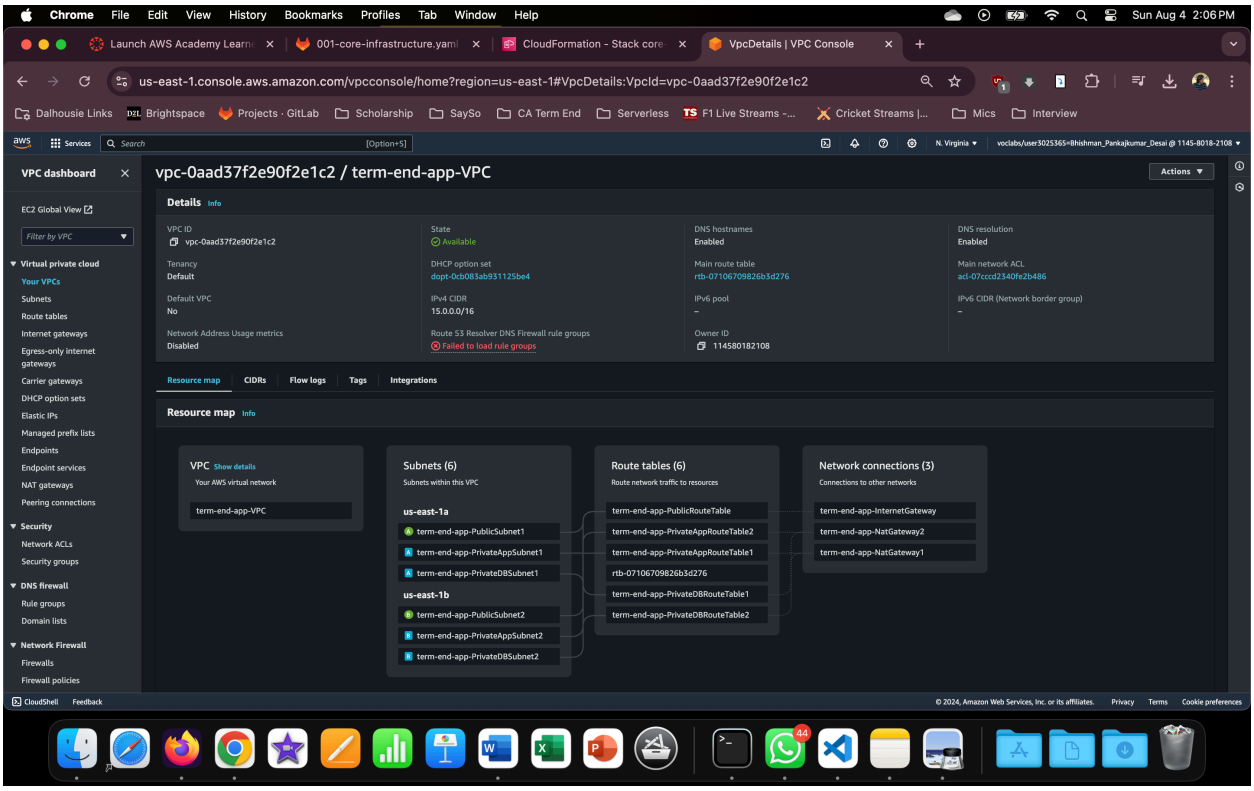

Figure 7: VPC

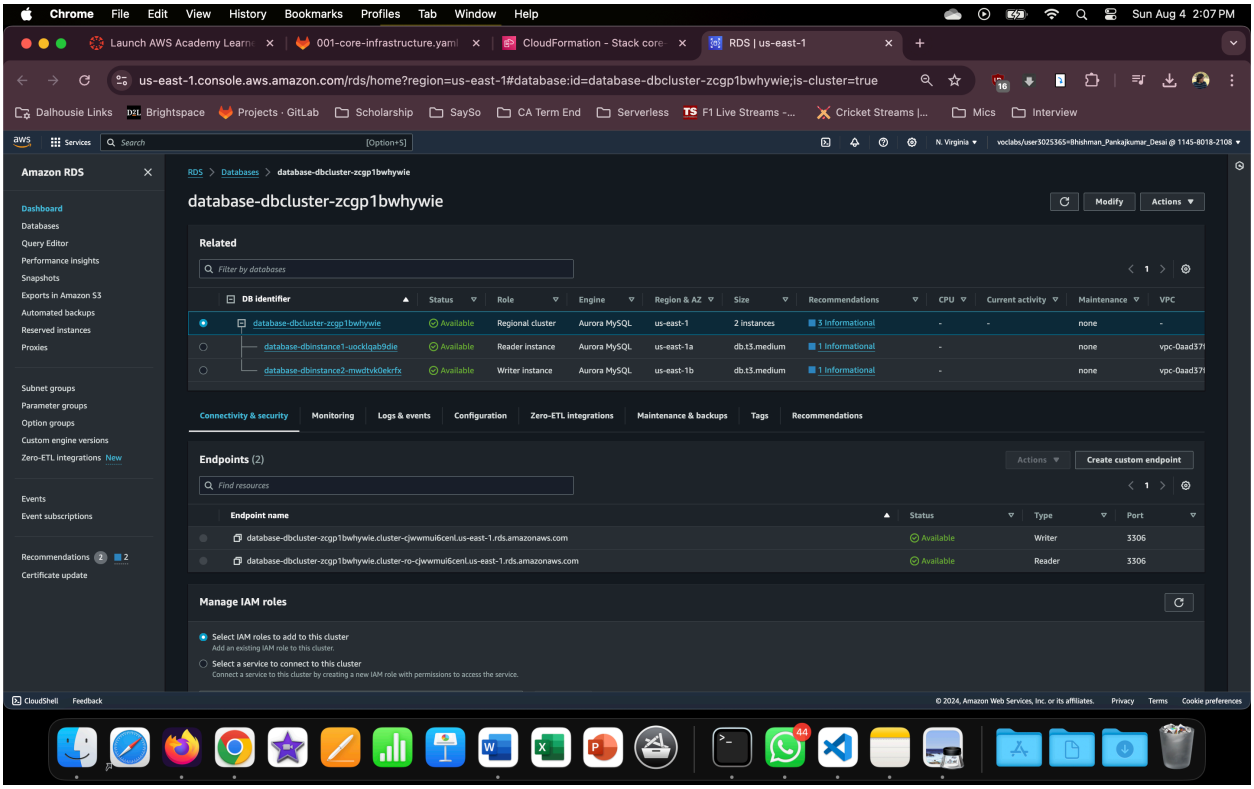

Figure 8: Database Cluster

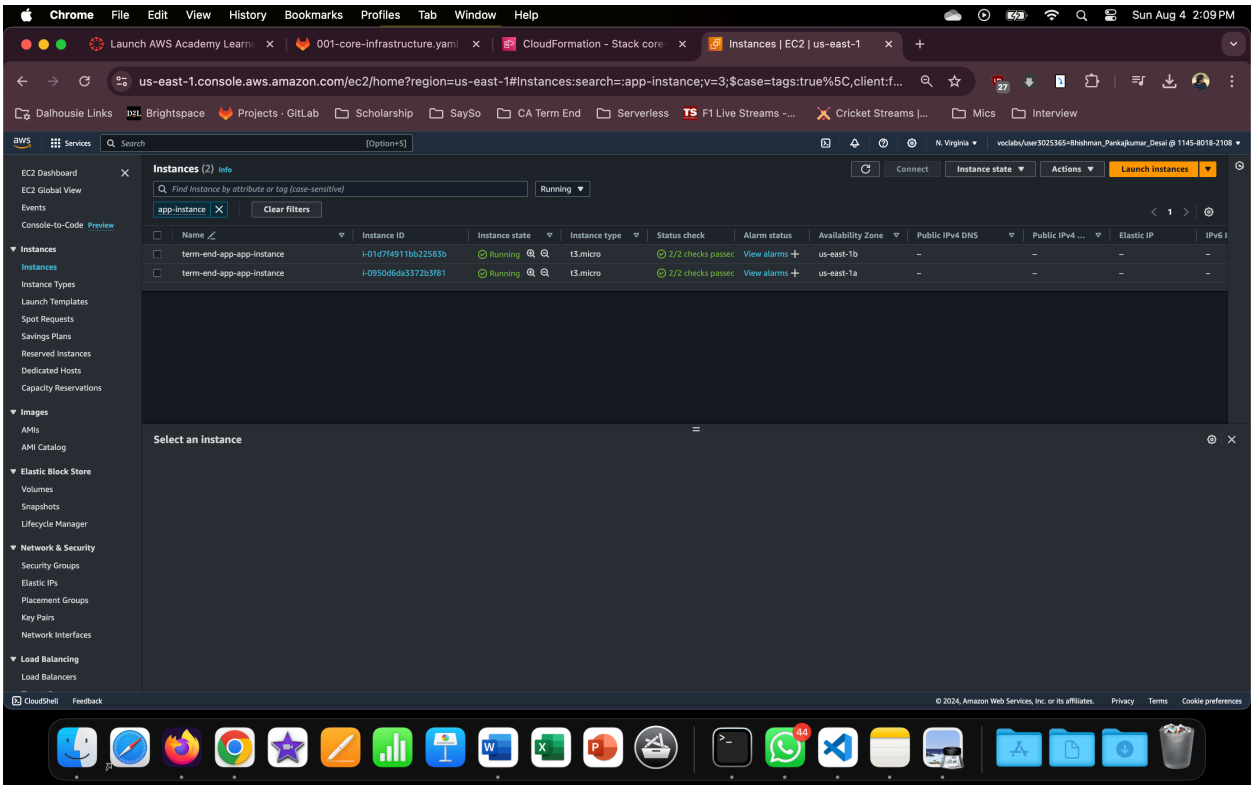

Figure 9: App Instances

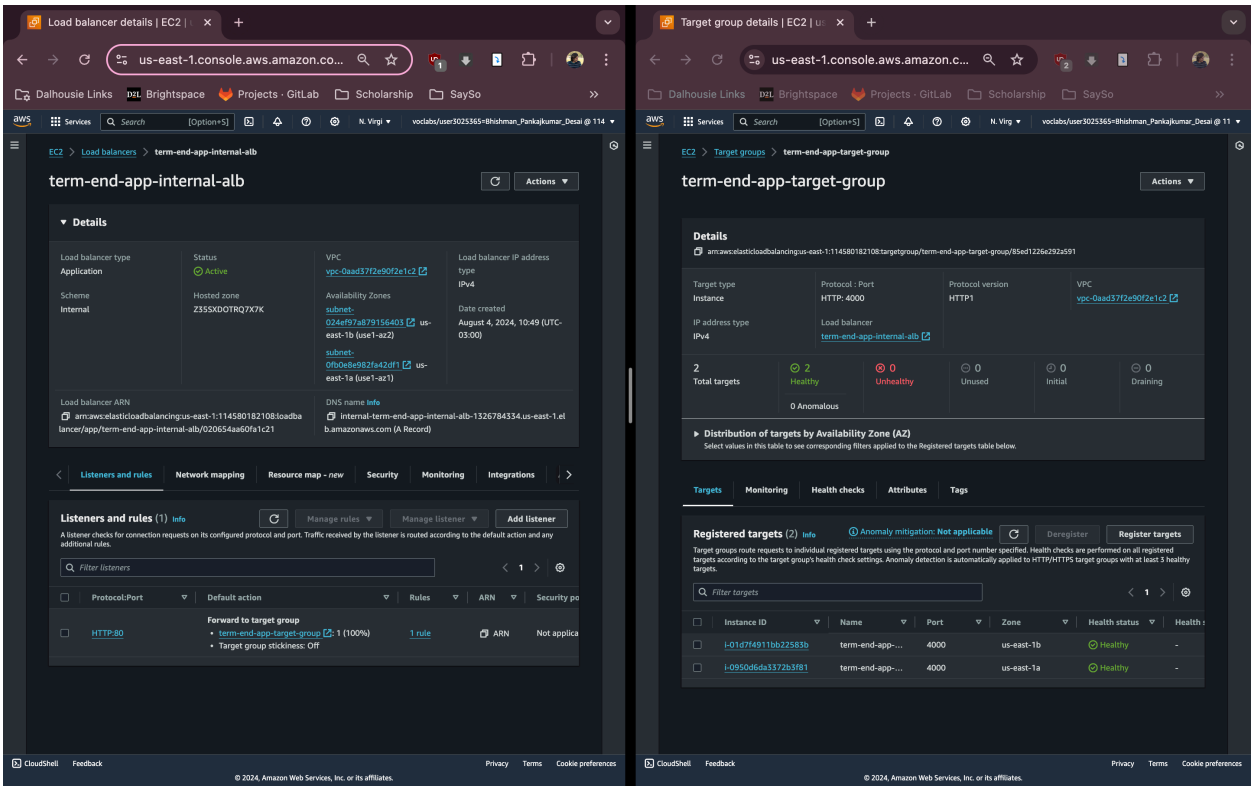

Figure 10: Internal Load balancer (app-tier)

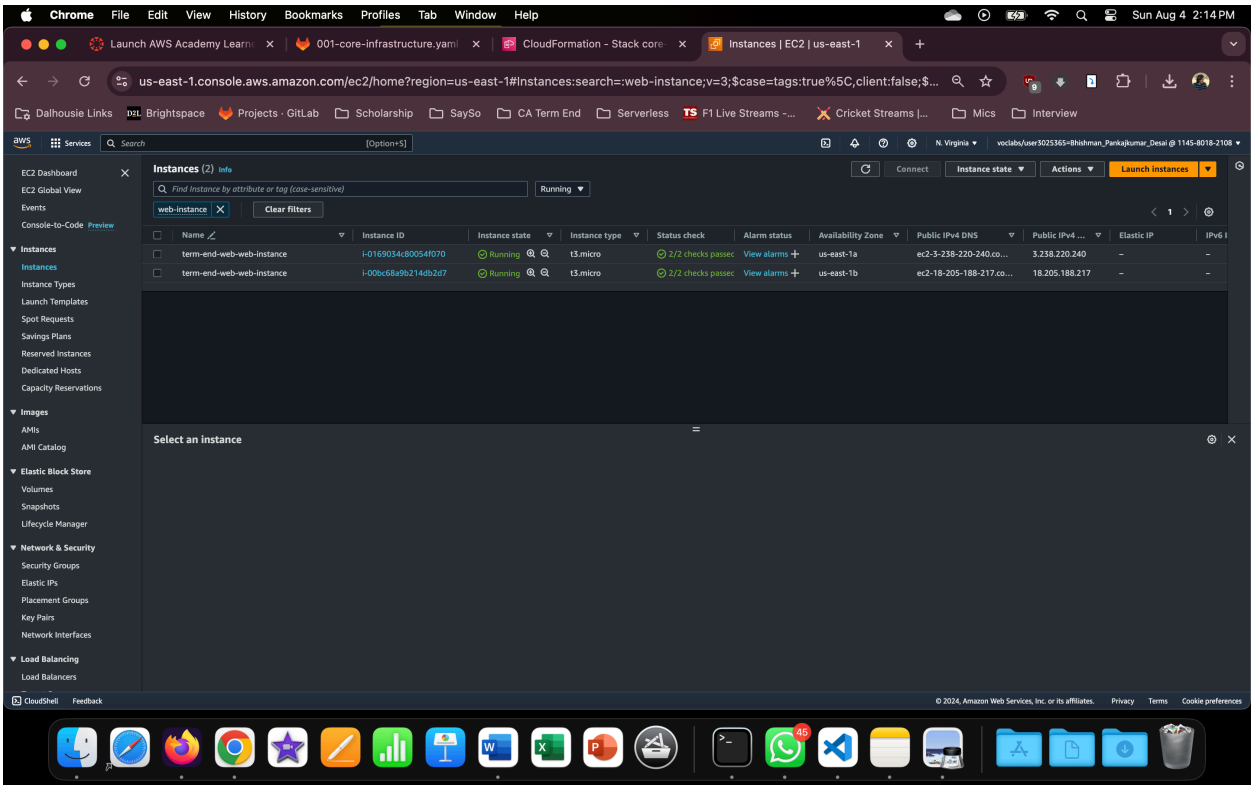

Figure 11: Web Instances

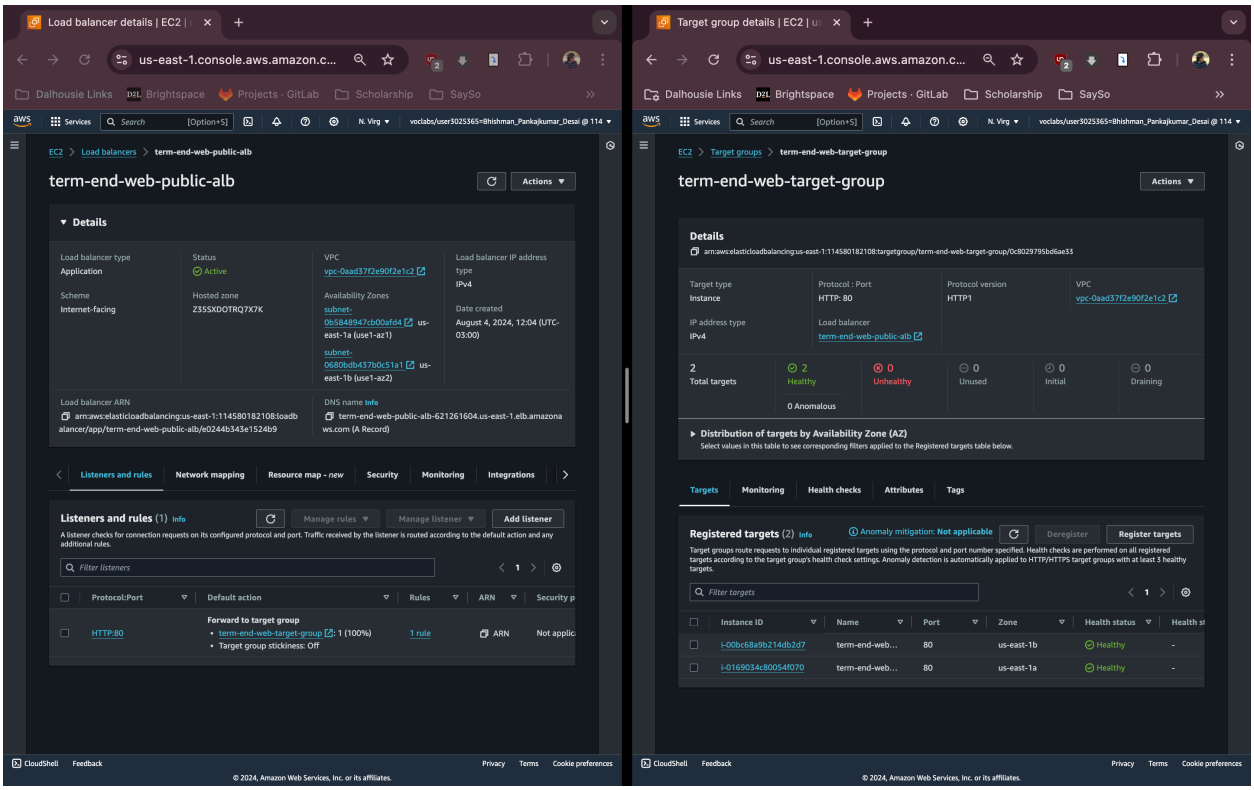

Figure 12: Public Load Balancer (web-tier)

References:

[1] "Amazon EC2 Pricing," Amazon Web Services, [Online]. Available: https://aws.amazon.com/ec2/pricing/. [Accessed: July 1, 2024].

[2] "AWS Lambda Pricing," Amazon Web Services, [Online]. Available: https://aws.amazon.com/lambda/pricing/. [Accessed: July 2, 2024].

[3] "AWS VPC Pricing," Amazon Web Services, [Online]. Available: https://aws.amazon.com/vpc/pricing/. [Accessed: July 3, 2024].

[4] "Elastic Load Balancing Pricing," Amazon Web Services, [Online]. Available: https://aws.amazon.com/elasticloadbalancing/pricing/. [Accessed: July 5, 2024].

[5] "AWS CloudFormation Pricing," Amazon Web Services, [Online]. Available: https://aws.amazon.com/cloudformation/pricing/. [Accessed: July 6, 2024].

[6] "AWS Auto Scaling Pricing," Amazon Web Services, [Online]. Available: https://aws.amazon.com/autoscaling/pricing/. [Accessed: July 8, 2024].

[7] "Amazon CloudWatch Pricing," Amazon Web Services, [Online]. Available: https://aws.amazon.com/cloudwatch/pricing/. [Accessed: July 9, 2024].

[8] "AWS CloudTrail Pricing," Amazon Web Services, [Online]. Available: https://aws.amazon.com/cloudtrail/pricing/. [Accessed: July 10, 2024].

[9] "AWS CloudFormation," The Knowledge Academy, [Online]. Available: https://www.theknowledgeacademy.com/blog/aws-cloudformation/. [Accessed: July 12, 2024].

[10] "CloudWatch vs CloudTrail: Understanding the Differences," AWS Fundamentals, [Online]. Available: https://blog.awsfundamentals.com/cloudwatch-vs-cloudtrail-understanding-the-differences. [Accessed: July 13, 2024].

[11] "Security Groups for Your VPC," Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-groups.html. [Accessed: July 14, 2024].

[12] "Auto Scaling Benefits," Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-benefits.html. [Accessed: July 15, 2024].

[13] "What is AWS Auto Scaling and Elastic Load Balancer (ELB)?," AWS Community, [Online]. Available: https://community.aws/content/2Yns4L77ddmwjvcST9YMPB2j05K/what-is-aws-autoscaling-and-elastic-load-balancer-elb. [Accessed: July 17, 2024].

[14] "AWS Cost Explorer," Amazon Web Services, [Online]. Available: https://aws.amazon.com/aws-cost-management/aws-cost-explorer/. [Accessed: July 18, 2024].

[15] "Using AWS Lambda with AWS CloudFormation," Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/services-cloudformation.html. [Accessed: July 19, 2024].

[16] "AWS Lambda Integration with CloudFormation," Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/services-cloudformation.html. [Accessed: July 19, 2024].