FACULTY OF COMPUTER SCIENCE

Term Project

In
The Class of

**CSCI 5411: ADVANCED CLOUD ARCHITECTING**

*by*

Shreya Kapoor (B00957587)

# Table of Contents

# INTRODUCTION ABOUT THE APPLICATION

I have chosen an established open-source banking application having over 595 commits and 1.4k stars on its GitHub repository. The aim is simply to design the Cloud Architecture to host the application on AWS using the required AWS services and satisfying maximum number of pillars within the AWS Well-Architected Framework.

## ABOUT THE PROJECT

This project is a Full Stack Web Application that works like financial software used in banks. It calculates the account balance using a method called Double-entry bookkeeping. The app follows industry standards to ensure it is robust and reliable.

**Key Features:**

- **Language Support:** The app is available in three languages: English, German, and Polish.
- **Multiple Currencies:** It supports different currencies, with real-time exchange rates fetched from an external server.
- **Design Principles:** The app is built following best practices like SOLID, DRY, and KISS to make it easy to maintain and scale.
- **Progressive Web App (PWA):** The app works on all modern browsers and mobile devices, ensuring a smooth user experience.

**It uses the following technology stack:**

- **Frontend**: ReactJS
- **Backend**: Node.JS
- **Database**: PostgreSQL

## ARCHITECTURE DIAGRAM

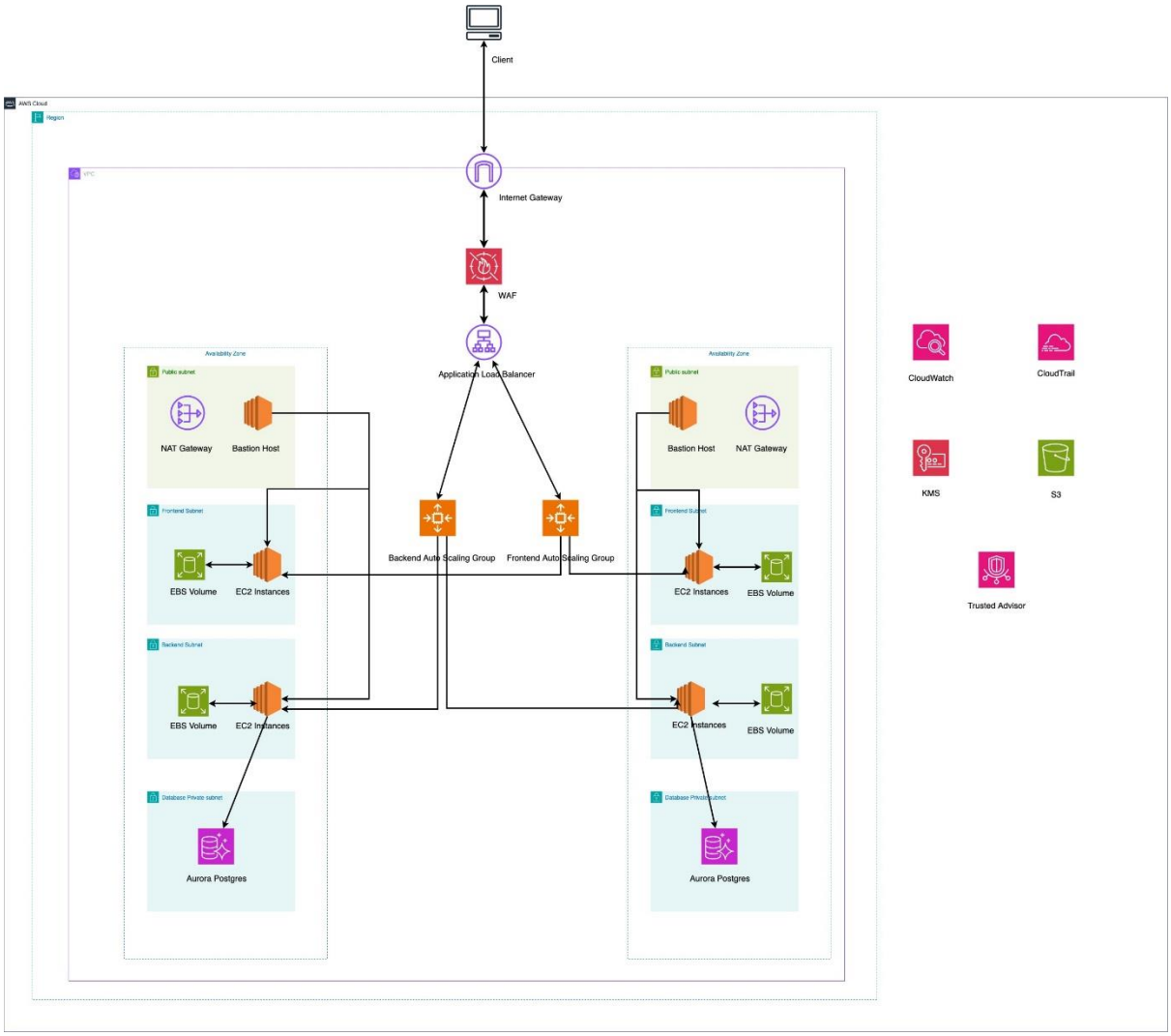


*Figure 1: Architecture Diagram*

Deployed Link: https://bank-loadbalancer-1215003019.us-east-1.elb.amazonaws.com
Github Link: https://github.com/pietrzakadrian/bank.git

## AWS SERVICES USED

To host this open-source bank application, I have leveraged the following AWS Services.

### COMPUTE

In the compute category, the AWS services are utilized to ensure the smooth and efficient operation of our bank's application. These services include EC2 (Elastic Compute Cloud and Auto Scaling Groups. Both services play a crucial role in managing and optimizing the computational resources required by our application.

#### Amazon EC2 Instances

EC2 (Elastic Compute Cloud) is a core part of our infrastructure, serving as the primary platform to run our application servers.

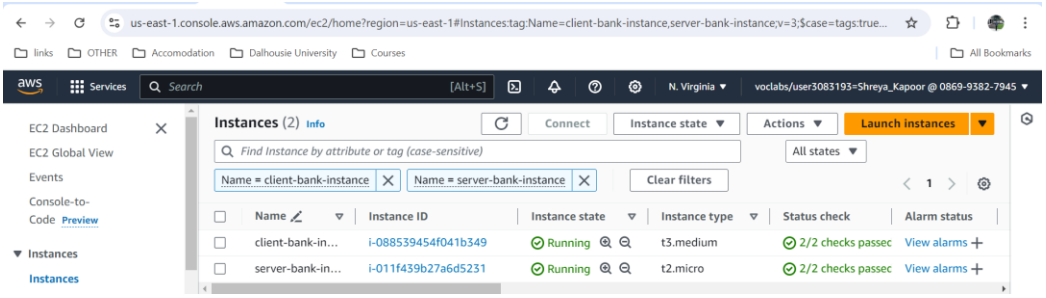


*Figure 2: EC2 Instances*

Following are the reasons why EC2 has been chosen:

- **Versatile Configuration and Customization**
  EC2 instances provide the ability to choose specific instance types, operating systems, storage options, and network setups. This versatility is vital for accommodating the unique demands of a banking application.
  In this project, the following instance types were selected:
  - **Frontend Application:** t3.medium
  - **Backend Application:** t2.micro

  Initially, I used a t2.micro instance for the frontend. However, it resulted in a "JavaScript: Heap Out Of Memory" error. To address this, I upgraded to a t3.medium instance, which has 4 GB of memory and 2 CPUs [1], meeting the application's requirements. Future adjustments to instance types can be made based on AWS Compute Optimizer's recommendations.

- **Comprehensive Control Over the Environment**
  EC2 instances provide complete control over the environment, allowing for the installation of necessary software, libraries, and configurations. This is crucial for managing a specialized application like mine.

- **Standardized Deployments with AMI**
  Amazon Machine Images (AMIs) allow for the creation of consistent, pre-configured instances. This ensures a reliable and secure deployment process. For hosting both my client and server applications, I used the Amazon Linux 2023 AMI.

### Auto Scaling Group

The Auto Scaling Group (ASG) is a crucial component for managing the scalability and efficiency of the application.
In my application, I have created separate Auto Scaling Group for the client and server.

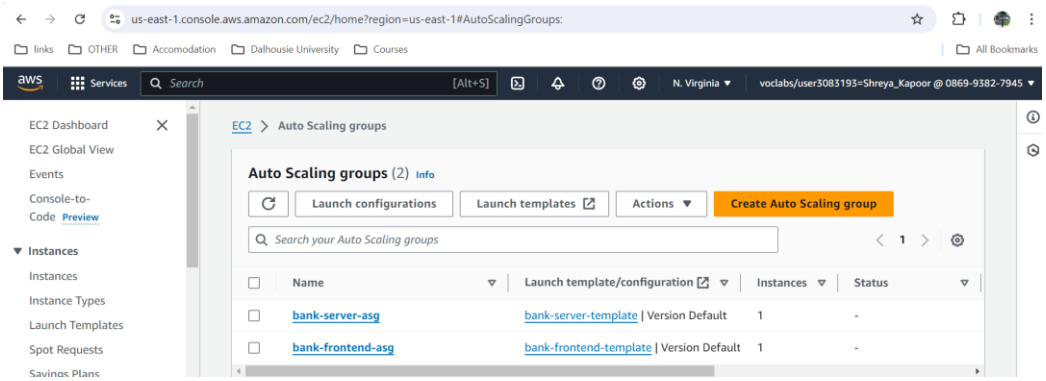


*Figure 3: Auto Scaling Groups*

Following are the reasons why ASG has been chosen:

- **Scalability**
  The ASG ensures that the application can handle varying workloads by automatically adjusting the number of EC2 instances based on current demand. During peak periods, the ASG will scale out by adding more instances, and during off-peak times, it will scale in by reducing the number of instances. For example, if the CPU utilization exceeds 70% on an instance, the ASG will launch a new instance to balance the load.

- **High Availability**
  To maintain continuous operation and minimize downtime, the ASG distributes instances across multiple Availability Zones (AZs). In the event of a failure in one AZ, traffic is redirected to healthy instances in other AZs. For this application, the ASGs are set up in AZs us-east-1a and us-east-1b, ensuring that the system remains available even if one AZ encounters issues.

- **Cost Optimization**
  The ASG helps control costs by provisioning instances only when needed. During low-demand periods, excess instances are terminated to save on expenses. In my banking application, if the CPU Utilization is under 20%, then ASG will automatically reduce the instance by 1 unit as depicted below:

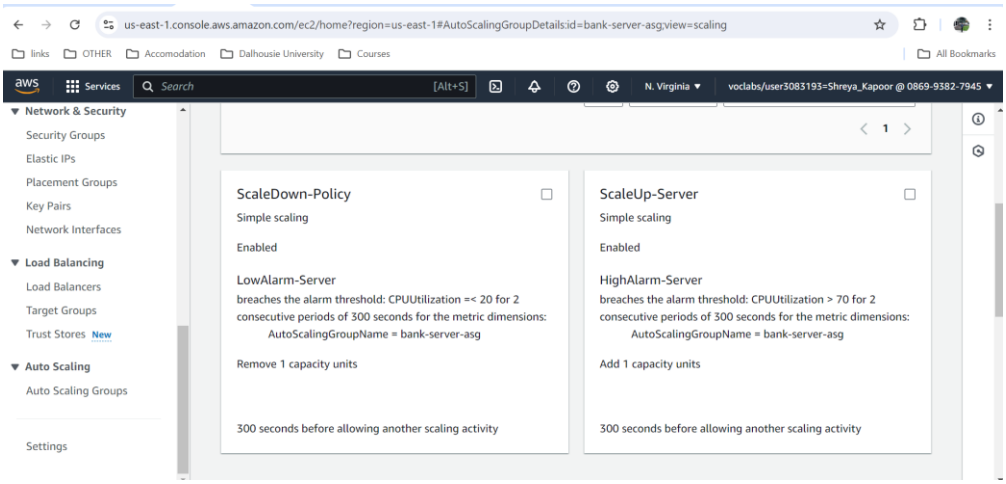


*Figure 4: Automatic Scaling*

## NETWORK AND CONTENT DELIVERY

To ensure efficient and secure delivery of content for the bank application, several AWS services are utilized under the Networking and Content Delivery category. These services play a critical role in optimizing the network infrastructure, ensuring low latency, and enhancing security.

### Elastic Load Balancer

Elastic Load Balancing (ELB) is employed to distribute incoming application traffic across multiple EC2 instances, ensuring high availability and reliability. It automatically scales its handling capacity to meet incoming traffic demands.

Following are the reasons why ASG has been chosen:

- **Traffic Distribution:** ELB evenly distributes traffic to EC2 instances, preventing any single instance from being overwhelmed. This load balancing ensures optimal application performance.

- **Health Checks:** ELB continuously monitors the health of registered instances. If an instance becomes unhealthy, ELB routes traffic to healthy instances, maintaining the application's availability.

- **High Availability and Fault Tolerance:** ELB enhances the availability and fault tolerance of the application by distributing traffic across healthy instances. It automatically routes traffic away from unhealthy instances, ensuring a reliable user experience.

- **Ease of Use:** ELB is a fully managed service that requires minimal configuration. It automatically scales with the traffic load, simplifying the process of managing incoming requests.

- **Integration with Auto Scaling:** ELB seamlessly integrates with Auto Scaling, ensuring that new instances launched by Auto Scaling are automatically registered with the load balancer. This simplifies the scaling process and maintains a balanced distribution of traffic [2].

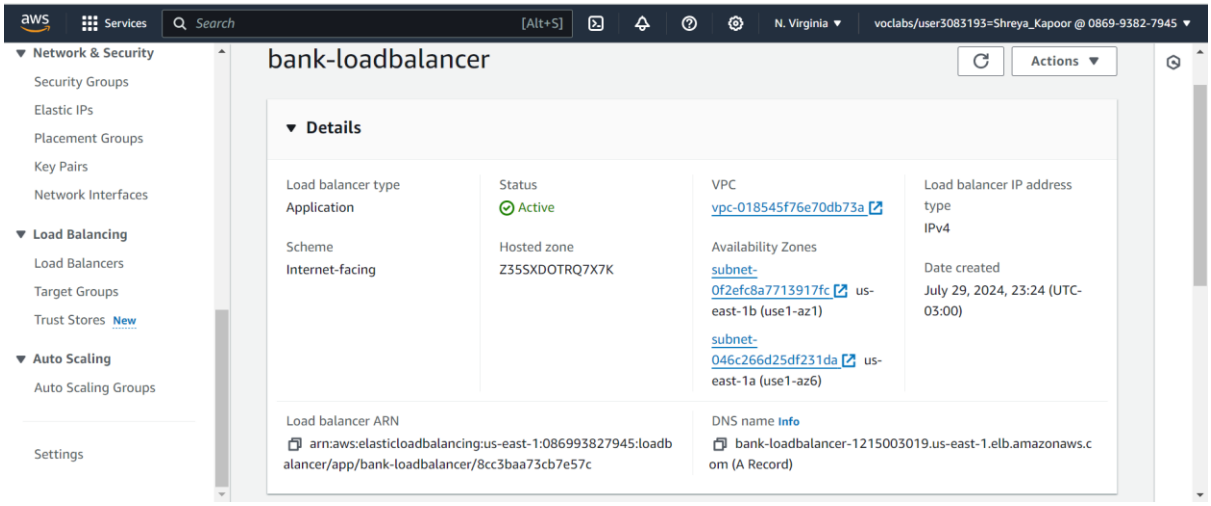


*Figure 5: Load Balancer for Application*

Above is the load balancer for my application, in which I have configured the rules for both client and the server application. This load balancer is present in the public subnet of the VPC and is internet facing. I have created 2 load balancer listeners that listens on port 80 and 4000. I have created 2 target groups for my client and server application. My frontend application runs on 3000 port. I have mapped listener of port 80 to my frontend target group. My backend application runs on 4000 port which is mapped to listener on port 4000. Following are the target groups for my application.

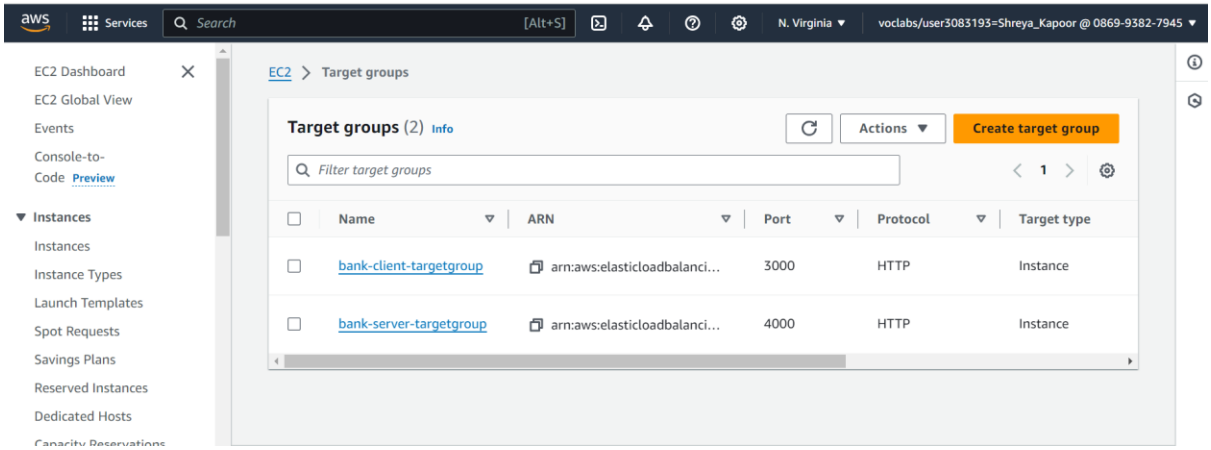


*Figure 6: Target groups for load balancer*

## VPC (Virtual Private Cloud)

Amazon VPC is a fundamental component in the AWS infrastructure, providing network isolation and control over the networking environment. It allows me to launch AWS resources in a virtual network that I define, giving me complete control over my virtual networking environment, including resource placement, connectivity, and security.

The VPC is set up in the us-east-1 (N. Virginia) region, which is popular for its wide range of AWS services and low-latency connectivity. The VPC uses a 10.0.0.0/16 CIDR block, providing plenty of IP addresses for the subnets and resources [3].

Two Availability Zones, us-east-1a and us-east-1b, are used to ensure high availability and fault tolerance. If one AZ experiences issues, the other can keep running, making the application more reliable.

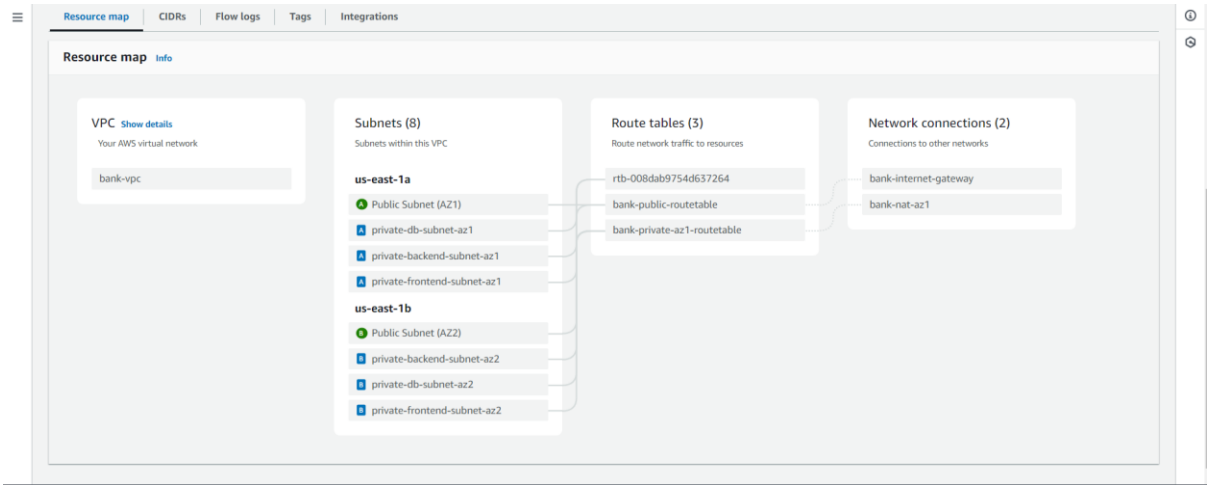


*Figure 7: Resource Map in VPC*

Each AZ has four subnets, each serving different parts of the application:

- **Frontend Application:** The client facing frontend application is present in the private subnet, ensuring they are not directly accessible from the internet.
- **Backend Subnet:** The backend server is present in the private subnet, which do not have direct internet access for added security.
- **RDS Subnet:** The database tier is present in the private subnet, also without direct internet access to enhance security.
- **Load Balancer Subnet:** The load balancer is present in the public subnet, allowing it to distribute incoming traffic to the EC2 instances in the private subnets [4].

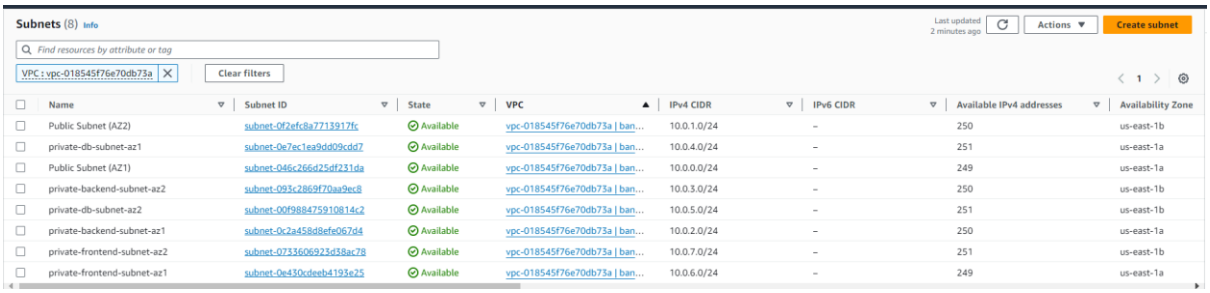


*Figure 8: Subnets in VPC*

**Internet Gateway:** An Internet Gateway is attached to the VPC to allow instances in the load balancer subnet to access the internet, enabling users to reach the banking application securely [5].


*Figure 9: Internet Gateway*

Following are the reasons why VPC has been chosen:

- **Security and Compliance:** VPCs offer the necessary isolation and control to meet security and compliance needs for sensitive data and applications, ensuring the bank's application runs in a secure environment.

- **Customizable Networking:** VPCs let us define custom IP ranges, subnets, and route tables to fit the bank's specific networking requirements, allowing for optimal resource placement and traffic management.

- **Scalability and Flexibility:** VPCs allow easy scaling of resources within the network. As the application grows, we can add more resources without disrupting the existing setup.

- **Network Security:** VPCs provide strong network security through security groups and network ACLs, controlling traffic to individual instances and subnets [3].

## NAT Gateway

A NAT (Network Address Translation) Gateway is a crucial part of our Amazon VPC setup, allowing instances in private subnets to connect to the internet or other AWS services while preventing inbound internet connections to those instances.

The NAT Gateway is set up in the us-east-1 (N. Virginia) region and is placed in a public subnet within the same Availability Zones used for the VPC.

The NAT Gateway is linked to a public subnet, enabling it to route traffic from instances in private subnets to the internet. Each private subnet directs outbound traffic to the NAT Gateway, ensuring instances remain isolated yet connected.

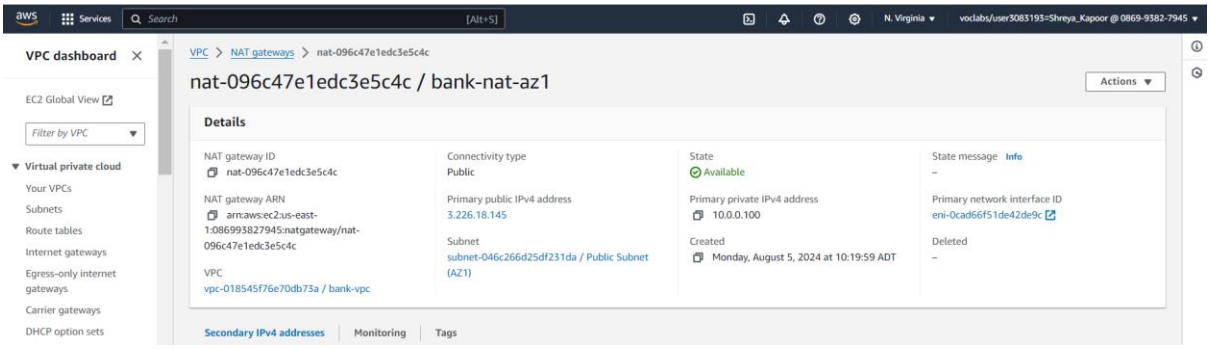


*Figure 10: NAT Gateway*

**Why the NAT Gateway Was Chosen:**

- **Enhanced Security:** Using a NAT Gateway allows instances in private subnets to securely access the internet without exposing them to potential threats.

- **Simplified Configuration:** The NAT Gateway provides an easy method for private instances to initiate outbound communication.

- **Cost Efficiency:** One NAT Gateway can serve multiple private subnets, making it a cost-effective solution for providing internet access to private instances [6].

## SECURITY

In our bank's AWS infrastructure, maintaining robust security is paramount. To ensure a secure environment, we utilize Security Groups and a Web Application Firewall (WAF) to protect our application from various threats and vulnerabilities.

### Security Groups

Security Groups act as virtual firewalls for our EC2 instances, controlling inbound and outbound traffic. They are essential for defining the rules that determine the traffic allowed to reach our instances.

**Configuration for the Banking Application:**

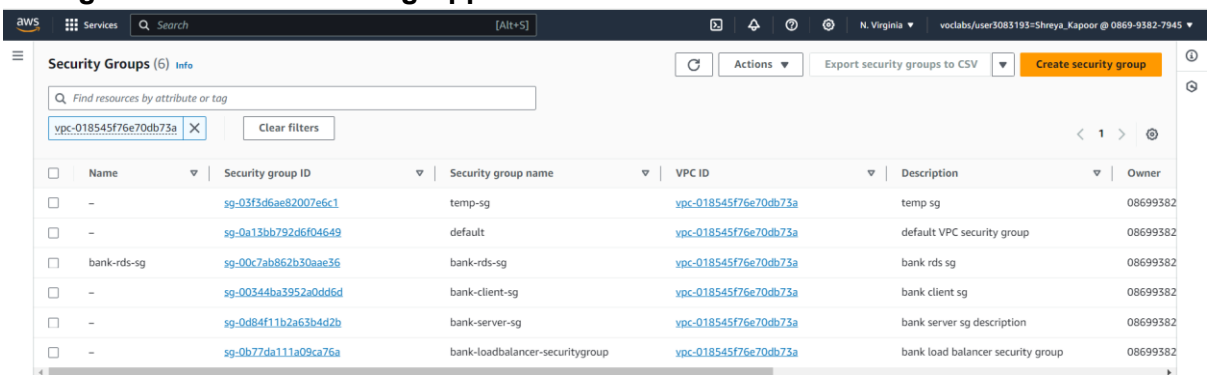


*Figure 11: Security Groups*

1. **Frontend Security Group:**
    a. **Inbound Rules:**

        i. Allows traffic from the application load balancer on port 3000.
        ii. Allows SSH (port 22) traffic for administrative purposes.
    b. **Outbound Rules:**
        i. Allows all outbound traffic to ensure instances can initiate communications as needed.

2. **Backend Security Group:**
    a. **Inbound Rules:**
        i. Allows traffic from the application load balancer on the ports 4000.
    b. **Outbound Rules:**
        i. Allows all outbound traffic to ensure instances can initiate communications as needed.

3. **Database Security Group:**
    a. **Inbound Rules:**
        i. Allows traffic from the backend security group on the database port 3306 for MySQL.
    b. **Outbound Rules:**
        i. Allows all outbound traffic to ensure the database instances can connect to necessary services for updates and maintenance.

4. **Load Balancer Security Group:**
    a. **Inbound Rules:**
        i. Allows HTTP (port 80) and backend port 4000 traffic from anywhere.
    b. **Outbound Rules:**
        i. Allows all outbound traffic to ensure the load balancer can communicate with backend instances and other services.

**Benefits of Using Security Groups:**

- **Fine-Grained Control:** Security Groups provide precise control over the traffic that can reach our instances, enhancing security by allowing only specific types of traffic.

- **Stateful Filtering:** Security Groups automatically allow response traffic for inbound requests, simplifying rule management.

- **Isolation and Segmentation:** By creating separate Security Groups for different tiers of the application, we can isolate and segment traffic, reducing the attack surface and limiting the potential impact of a security breach [7].

## Web Application Firewall (WAF)

AWS WAF is a managed security service that protects web applications from common exploits like SQL injection and cross-site scripting (XSS). Positioned in front of an Application Load Balancer (ALB), it inspects and filters incoming web traffic, adding an extra layer of security.

In my application, I have added the Web ACL with the following rules

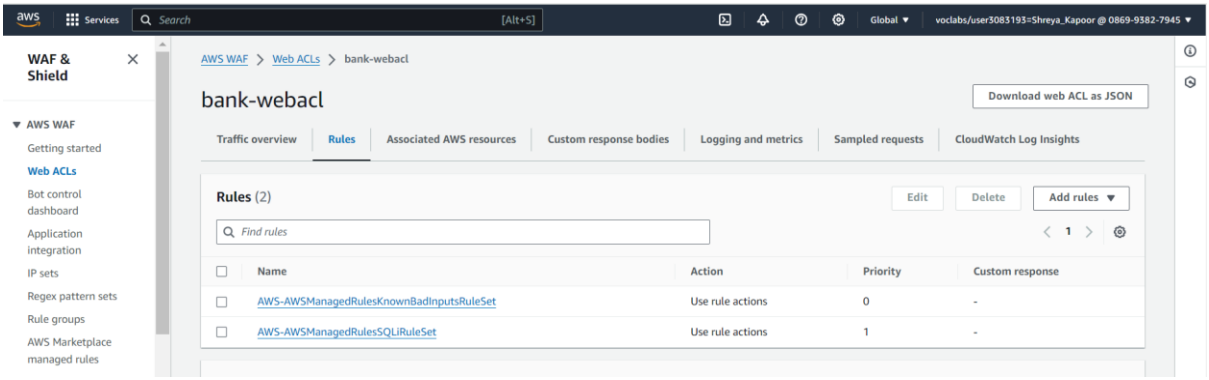


*Figure 12: Rules of Web ACL*

The following traffic rule has been captured after added the WAF in front of the load balancer

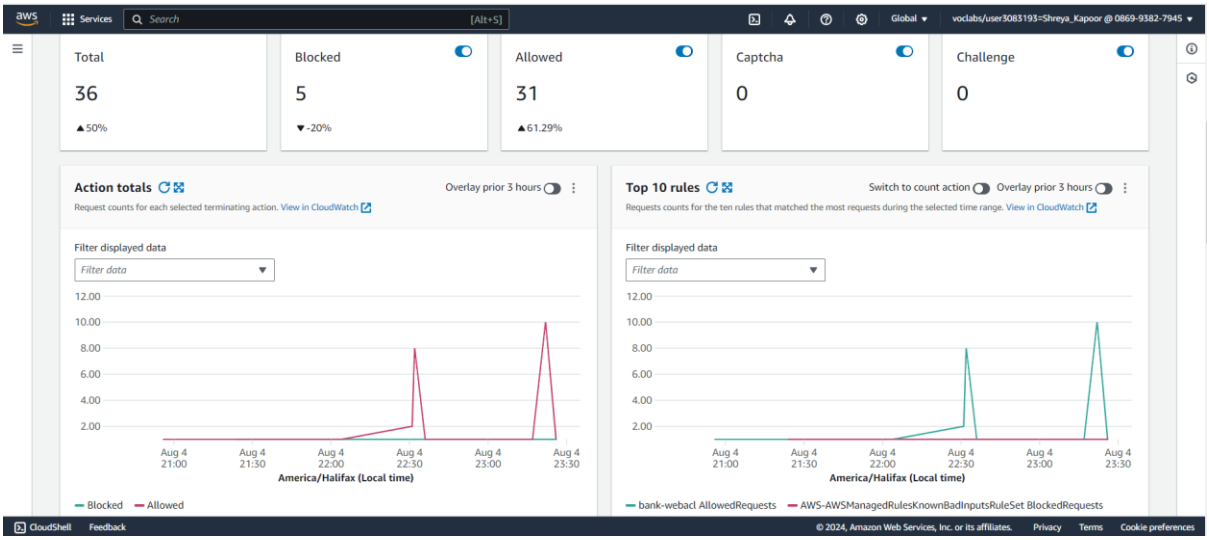


*Figure 13: Traffic Overview*

**Benefits of Using WAF:**

- **Protection Against Web Exploits:** AWS WAF defends against common web exploits by inspecting and filtering web traffic before it reaches the ALB, mitigating threats early on.

- **Granular Control with Rules and Conditions:** AWS WAF lets us define specific rules and conditions to control web traffic, enabling tailored security policies.

- **Managed Rule Sets for Quick Deployment:** AWS WAF offers managed rule sets that provide swift and effective protection against known threats. AWS regularly updates these sets to ensure our defenses remain current.

- **Logging and Monitoring for Incident Response:** AWS WAF's logging and monitoring features give us visibility into web traffic and security events. This is vital for incident response, allowing our team to analyze and counteract threats.

- **Seamless Integration with ALB:** AWS WAF integrates smoothly with the Application Load Balancer (ALB), ensuring all incoming traffic is inspected and filtered before reaching our servers. This centralizes security policy implementation and simplifies management [8].

## DATABASE

### Amazon RDS

In the database category, Amazon RDS (Relational Database Service) is used to manage and store the banking application's data. RDS ensures high availability, scalability, and security for our database operations.

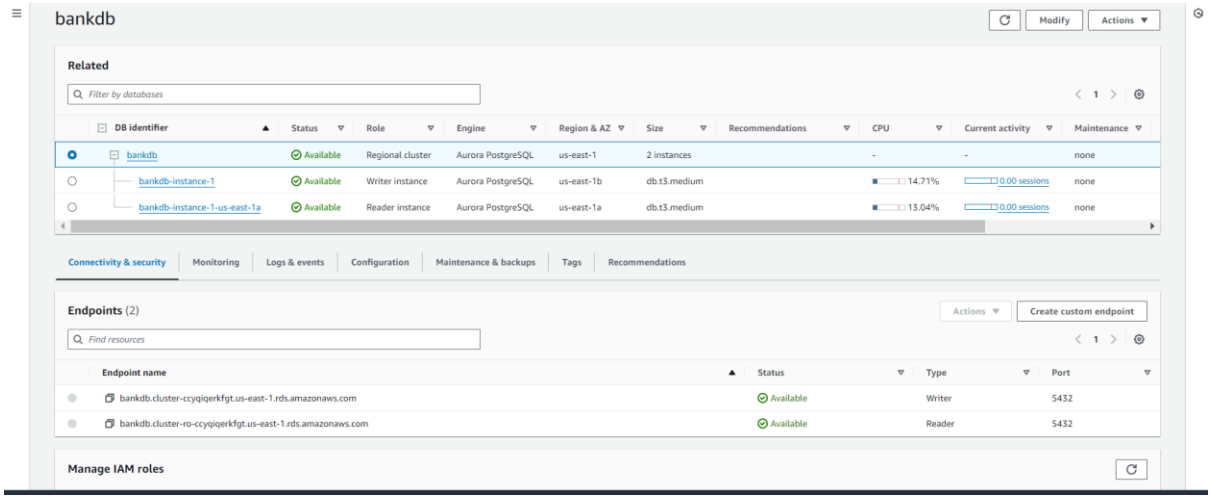


*Figure 14: RDS*

**Configuration and Features:**

1. **Instance Configuration:**
    - **Instance Class:**
        - I have used a db.t3.medium instance class, which balances compute, memory, and network resources efficiently.
    - **Multi-AZ Deployment:**
        - To enhance availability and fault tolerance, I used Multi-AZ deployments. This setup includes a standby replica in another Availability Zone (AZ) that automatically takes over in case of primary instance failure.

2. **Storage:**
    - **Backup and Retention:**
        - Automated backups are configured with a 7-day retention period. Regular snapshot backups are also performed to ensure data durability.

3. **Security:**
    - **Encryption:**
        - Data at rest is encrypted using AWS Key Management Service (KMS). Data in transit is secured with SSL/TLS.
    - **Access Control:**
        - RDS instances are placed in private subnets within the VPC, ensuring they are not directly accessible from the internet. Security groups manage inbound and outbound traffic.

4. **Monitoring and Maintenance:**
    - **Amazon CloudWatch:**
        - CloudWatch monitors key metrics like CPU utilization, memory usage, and disk I/O, helping maintain the database's health and performance.

**Benefits:**

- **Automated Database Management:** RDS automates routine tasks such as backups, patching, and software updates, allowing us to focus on optimizing performance and application development.

- **High Availability:** Multi-AZ deployments ensure continuous availability, with automatic failover to a standby replica in case of an AZ-level failure.

- **Scalability:** RDS allows for easy scaling of compute and memory resources to handle varying workloads efficiently.

- **Read Replicas:** Read replicas can be created to offload read-intensive queries, improving overall performance and handling high read traffic.

- **Security:** Robust security features include network isolation through Amazon VPC, encryption at rest and in transit, and integration with AWS Identity and Access Management (IAM) for access control. Placing RDS in private subnets further protects it from cyber-attacks.

- **Monitoring and Metrics:** Comprehensive monitoring through Amazon CloudWatch provides real-time performance insights, enabling proactive issue identification and resolution [9].

## MANAGEMENT AND GOVERNANCE

In our banking application, management and governance are essential to maintain operational efficiency, security, and compliance. AWS offers various services to help

manage and monitor resources effectively. Key among these services is Amazon CloudWatch.

## Cloud Watch

Amazon CloudWatch is a monitoring and observability service that provides real-time insights into the performance and health of AWS resources and applications.

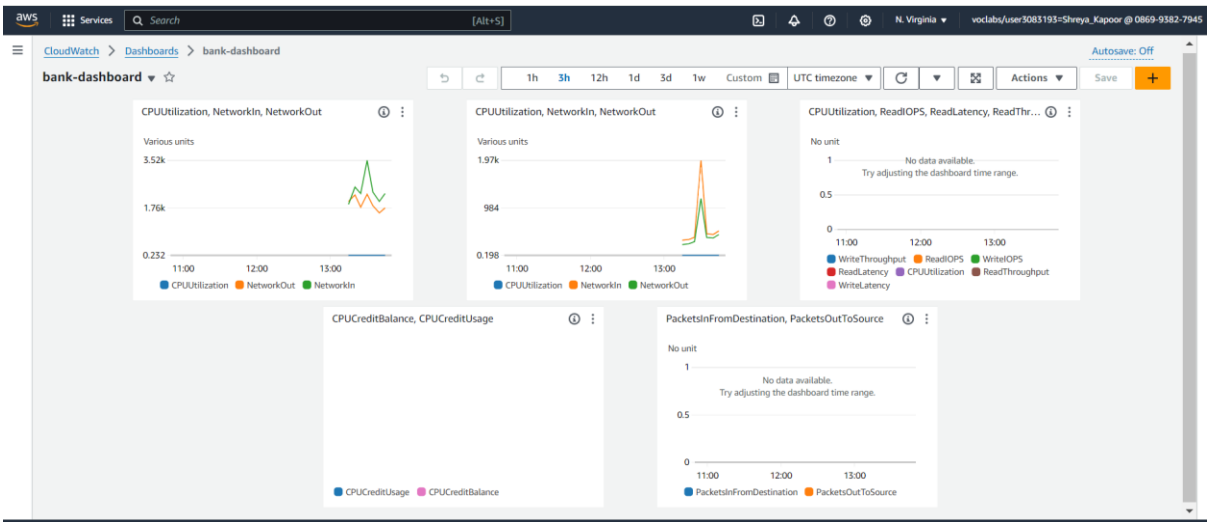


*Figure 15: Cloud Watch Dashboard*

Here's how CloudWatch is utilized in our banking application:

a. **Monitoring Metrics:**
   - **EC2 Instances:**
     CloudWatch collects detailed metrics for EC2 instances, such as CPU utilization, and network activity. These metrics help in monitoring the performance and identifying potential issues.
   - **RDS Instances:**
     For RDS, CloudWatch tracks key metrics like CPU usage, memory, disk I/O, and database connections, allowing us to ensure the database is performing optimally.
   - **Auto Scaling:**
     CloudWatch monitors the scaling activities of Auto Scaling groups, providing insights into when and why instances are added or removed.

b. **Alarms:**
   CloudWatch allows us to set alarms on specific metrics. For example, if CPU utilization on an EC2 instance exceeds a certain threshold, an alarm can trigger an action such as auto-scaling or sending notifications to the operations team.

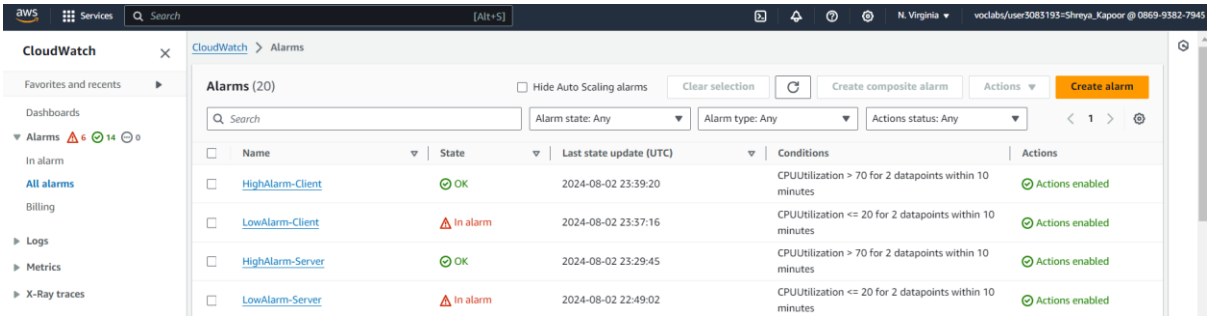


*Figure 16: Cloud Watch Alarms*

c. **Logs:**
Application logs from EC2 instances can be collected and monitored using CloudWatch Logs. This helps in troubleshooting and identifying issues in the application code.

d. **Dashboards:**
CloudWatch Dashboards provide a unified view of metrics and logs from various AWS services. Custom dashboards can be created to visualize key performance indicators (KPIs) and operational data, enabling quick and informed decision-making.

**Benefits:**

- **Comprehensive Monitoring:** CloudWatch offers detailed and real-time monitoring of AWS resources, enabling proactive management and quick issue resolution.

- **Automated Responses:** Through alarms and events, CloudWatch facilitates automated actions in response to operational thresholds or changes, enhancing operational efficiency.

- **Centralized Logging:** Centralized log management helps in maintaining a consolidated view of application and system logs, simplifying troubleshooting and ensuring compliance.

- **Enhanced Visibility:** Custom dashboards and detailed metrics provide enhanced visibility into the performance and health of the application, supporting better decision-making [10].

## AWS CloudTrail

AWS CloudTrail is a service that enables governance, compliance, and operational and risk auditing of my AWS account. It continuously logs and retains account activity related to actions across my AWS infrastructure, providing a detailed event history for all management events.

**Configuration and Features:**

1. **Trail Creation:**

- A trail named "bank-trail-events" has been created to capture all management events in the AWS account.
- The trail is configured to log data events, ensuring comprehensive coverage of all critical resources.

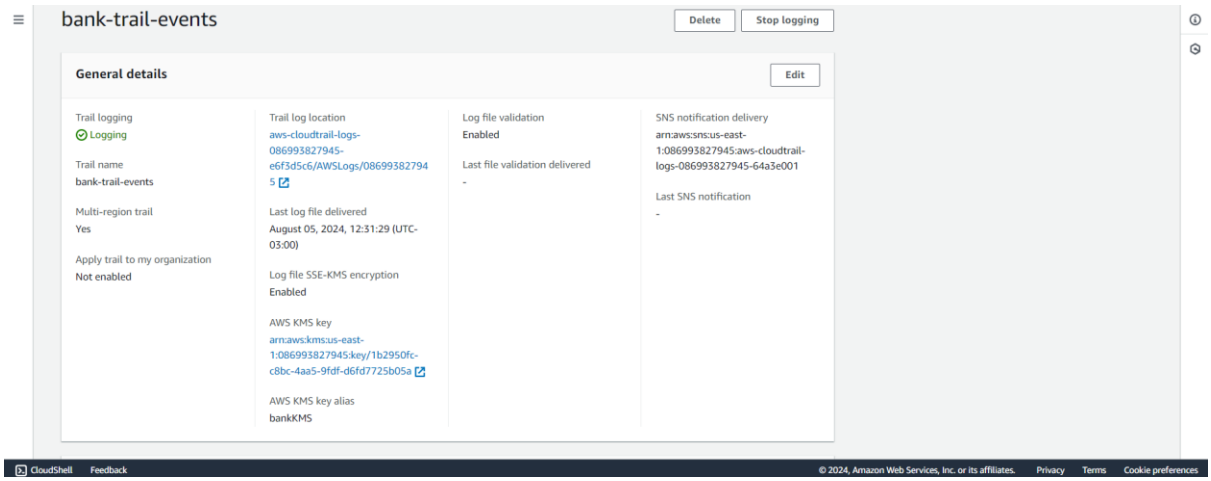


*Figure 17: Cloud Trail Events*

2. **S3 Bucket for Log Storage:**
   - All CloudTrail logs are stored in an encrypted S3 bucket for security and compliance purposes.
   - The S3 bucket has lifecycle policies to manage log retention and deletion, optimizing storage costs.

3. **CloudTrail Insights:**
   - CloudTrail Insights is enabled to detect unusual operational activity, providing valuable insights into potential security and operational issues.
   - Insights are used to identify anomalies, such as sudden spikes in API calls or changes in usage patterns.

4. **Integration with CloudWatch:**
   - CloudTrail can be integrated with CloudWatch to enable real-time monitoring and alerting for specific events, I cannot enable it due to IAM issues.
   - CloudWatch Alarms can be set up to notify the operations team of any critical actions, such as changes to security groups or unauthorized API calls.

**Benefits:**

- **Audit and Compliance:**
  - CloudTrail provides a comprehensive record of AWS account activity, supporting audit requirements and ensuring compliance with internal and external regulations.
  - Detailed logs of API calls help track changes and access patterns, facilitating security audits and forensic investigations.

- **Operational Monitoring:**
  - Continuous logging of account activity helps in monitoring the health and performance of the AWS environment.

- Integration with CloudWatch allows for real-time alerting and automated responses to critical events.

- **Security:**
  - CloudTrail enhances security by enabling detailed tracking of user activity and changes to the AWS environment.
  - Detecting unusual activities through CloudTrail Insights helps in identifying and mitigating potential security threats.

- **Governance:**
  - CloudTrail supports governance by providing visibility into the actions taken in the AWS environment.
  - It helps enforce policies and detect deviations from established practices, ensuring adherence to organizational standards [11].

# PILLARS WITHIN THE AWS WELL-ARCHITECTED FRAMEWORK

## Security

- **Security**
  - **AWS Shared Responsibility Model**
    The AWS Shared Responsibility Model splits security responsibilities between AWS and me. For my banking application, AWS handles the security of the cloud (like infrastructure and hardware), while I manage security in the cloud. This includes configuring security groups, and encrypting data, ensuring a secure environment.
  - **AWS Trusted Advisor**
    Although restricted in AWS Academy but AWS Trusted Advisor provides real-time guidance to help optimize my AWS environment according to AWS best practices. For my banking application if I could use this service, Trusted Advisor offers security checks that identify potential security risks, such as underutilized resources, open access permissions, and unencrypted data, helping me to implement best practices and maintain a secure AWS environment. By leveraging Trusted Advisor, I could continuously improve my security posture and ensure compliance with industry standards.

- **Identity and Access Management**
  - **IAM**
    AWS Identity and Access Management (IAM) lets us manage access to AWS services and resources securely. In my banking application, I was unable to create the IAM rules due to restricted access of AWS academy, however if I had the valid permissions, I would have created and managed users and groups, and sets permissions to allow or deny their access to specific resources, ensuring only authorized personnel can access sensitive data.
  - **Multi-Factor Authentication (MFA)**

Although restricted due to Learners Lab, Multi-Factor Authentication (MFA) adds an extra layer of security. For my banking application, even if a user's password is compromised, the attacker would still need the second factor to gain access, reducing the risk of unauthorized access.

- **Detective Controls**
  - **AWS CloudTrail**
    AWS CloudTrail provides logs of all API activities within my AWS account. For my banking application, I have created the trail which ensures a complete audit trail of user and resource activity, aiding in security analysis, resource change tracking, and meeting compliance requirements. It improves transparency and accountability by logging every action taken on my AWS resources.

- **Infrastructure Protection**
  - **Amazon VPC**
    Amazon Virtual Private Cloud (VPC) provides an isolated network environment for my banking application. By using VPC, I controlled the inbound and outbound traffic rules, isolated sensitive data, and protected my resources from external threats, maintaining a secure and compliant environment.
  - **AWS WAF**
    AWS Web Application Firewall (WAF) protects my web applications from common web exploits and vulnerabilities. In my banking application, AWS WAF filters and monitors HTTP and HTTPS requests based on rules that block malicious traffic, ensuring application security.

- **Data Protection**
  - **Elastic Load Balancer (ELB)**
    Elastic Load Balancer (ELB) distributes incoming application traffic across multiple targets. For my banking application, ELB enhances security by encrypting traffic between the load balancer and clients, ensuring secure communication.
  - **AWS KMS**
    AWS Key Management Service (KMS) manages cryptographic keys. In my banking application, KMS manages keys for encrypting data in services like EBS, RDS, and S3, ensuring data at rest is securely encrypted and access to keys is controlled.
  - **Amazon S3**
    Amazon S3 provides scalable object storage. For my banking application, S3 stores the cloudtrail logs. Data stored in S3 is encrypted at rest using server-side encryption with AWS KMS. S3 bucket policies and IAM policies control access, ensuring sensitive data is securely stored and accessible only to authorized users.
  - **RDS**
    Amazon Relational Database Service (RDS) allows me to set up and scale a relational database in the cloud. In my banking application, enabling encryption for RDS instances using AWS KMS ensures data at rest is

encrypted, and automated backups and multi-AZ deployments enhance data protection and availability.

- **Incident Response**
  - **IAM**
    Although restricted due to Learners Lab, IAM plays a crucial role in incident response by creating roles and policies that grant temporary access to AWS resources during an incident, ensuring only authorized personnel can perform critical tasks during a security event.
  - **CloudWatch**
    Amazon CloudWatch monitors and observes AWS resources and applications. For my banking application, CloudWatch sets up alarms and dashboards to monitor resource usage and performance. In case of an incident, these alarms trigger and send notifications, helping detect and respond to potential security threats quickly.

- **Application Security**
  - **AWS WAF**
    AWS Web Application Firewall (WAF) protects my banking application's web interfaces from threats like SQL injection and cross-site scripting (XSS). By creating custom rules for my application's traffic patterns, AWS WAF ensures that only legitimate traffic reaches my application, reducing the risk of exploitation and maintaining the integrity of my services [12].

## Reliability

- **Foundations**
  - **IAM**
    AWS Identity and Access Management (IAM) ensures reliable access control by managing user permissions and policies. For my banking application, I could not use IAM due to access issues. IAM helps establish a secure and reliable foundation by ensuring that only authorized users have access to critical resources, minimizing the risk of unauthorized changes that could affect reliability.
  - **Amazon VPC**
    Amazon Virtual Private Cloud (VPC) provides a secure and isolated environment for my resources. By setting up subnets, route tables, and security groups, I controlled the network traffic and ensured that my banking application's infrastructure is reliable and resilient to failures. VPC helps in segmenting and protecting my application's components, reducing the impact of potential issues.
  - **AWS Trusted Advisor**
    AWS Trusted Advisor provides real-time guidance to help improve the reliability of my AWS environment. For my banking application, Trusted Advisor could offer checks on underutilized resources, security groups, and IAM permissions, helping me identify potential issues that could impact

reliability. By leveraging these insights, I could make informed decisions to enhance the stability and reliability of my application.

- **Change Management**
  - **AWS CloudTrail**
    AWS CloudTrail logs all API activities, providing a comprehensive audit trail of changes. For my banking application, I have setup the CloudTrail which helps in tracking and monitoring changes to my AWS environment, ensuring that any unauthorized or unexpected changes can be quickly identified and addressed, maintaining the reliability of my system.
  - **Auto Scaling**
    Auto Scaling adjusts the number of EC2 instances based on demand, ensuring that my application can handle varying loads. In my banking application, Auto Scaling ensures that the application remains reliable during traffic spikes and maintains performance by automatically scaling in during low demand periods, optimizing resource usage.
  - **Amazon CloudWatch**
    Amazon CloudWatch monitors the AWS resources and applications, providing real-time insights and alerts. For my banking application, CloudWatch helps in detecting performance issues and triggering automated responses to maintain reliability. By setting up alarms and dashboards, I proactively managed and responded to potential problems, ensuring continuous availability.

- **Failure Management**
  - **Amazon S3**
    Amazon S3 provides durable and highly available object storage. In my banking application, S3 is used for storing the logs of cloudtrail.
  - **AWS KMS**
    AWS Key Management Service (KMS) manages encryption keys, ensuring secure data encryption. For my banking application, KMS protects sensitive data stored in S3 and cloudtrail by managing encryption keys securely. This ensures that even in the event of a failure, my data remains protected and can be reliably accessed and decrypted when needed.

- **Workload Architecture**
  - **AWS SDK**
    This does not fit into my application's architecture
  - **AWS Lambda**
    This does not fit into my application's architecture [13]

## Performance Efficiency

- **Architecture Selection**
  - **Auto Scaling for Compute**

Auto Scaling ensures optimal performance by automatically adjusting the number of EC2 instances based on demand. For my banking application, Auto Scaling allows the system to handle peak loads efficiently, ensuring users experience consistent performance even during traffic spikes, and reduces costs during low-demand periods by scaling down resources.

- **Amazon EBS**
  Amazon Elastic Block Store (EBS) provides high-performance block storage for EC2 instances. For my banking application, using EBS ensures low-latency access to data and high throughput, which is essential for transaction processing and maintaining database performance.
- **Amazon RDS**
  Amazon RDS manages database instances, optimizing performance through features like automated backups, scaling, and replication. In my banking application, RDS ensures efficient database performance, automated scaling to handle load variations, and read replicas to balance read-heavy workloads.
- **VPC**
  Amazon Virtual Private Cloud (VPC) offers a private, isolated environment for deploying resources. For my banking application, VPC helps optimize network performance by controlling traffic flow and ensuring low-latency communication between application components, which enhances overall system efficiency.

- **Compute and Hardware**
  - **EC2**
    Amazon EC2 provides scalable computing capacity in the cloud. For my banking application, EC2 allows me to choose appropriate instance types based on workload requirements, ensuring optimal performance and cost efficiency by right-sizing instances to match the compute needs of my application.

- **Data Management**
  - **Object (S3)**
    Amazon S3 offers scalable object storage with high availability and durability. For my banking application, S3 is used for storing large volumes of data, such as cloud trail logs, ensuring fast retrieval times and efficient data access, which enhances application performance.
  - **Block (EBS)**
    Amazon EBS provides persistent block storage for EC2 instances, delivering high IOPS and low latency. For my banking application, EBS ensures efficient data storage and retrieval for databases and critical applications, contributing to overall system performance and reliability.

- **Networking and Content Delivery**
  - **VPC**
    Amazon VPC optimizes network performance by isolating resources and controlling traffic flow. For my banking application, VPC ensures low-latency communication between components, which is critical for maintaining high performance in transaction processing and real-time data access.

- **Amazon RDS Read Replicas**
  Amazon RDS Read Replicas enhance database performance by distributing read traffic across multiple replicas. For my banking application, read replicas ensure that read-heavy workloads are efficiently managed, reducing the load on the primary database and improving response times for users.

- **Process and Culture**
  - **AWS CloudWatch**
    Amazon CloudWatch provides monitoring and observability for AWS resources and applications. For my banking application, CloudWatch enables real-time performance monitoring, alerting, and detailed insights into system performance, allowing me to proactively address issues and optimize resource usage for better performance efficiency [14].

## Sustainability

- **Region Selection**
  - **AWS Global Infrastructure**
    Selecting the appropriate AWS region for my banking application helps reduce the carbon footprint by utilizing AWS data centers with the most efficient energy usage and renewable energy sources. By deploying resources in regions that align with AWS's sustainability efforts, my application can contribute to lower energy consumption and improved sustainability.

- **Alignment to Demand**
  - **Auto Scaling**
    Auto Scaling ensures that my application only uses the necessary compute resources based on current demand. This reduces energy waste by scaling down resources during low traffic periods and scaling up during high demand, ensuring optimal use of infrastructure and contributing to more sustainable resource management.

- **User Behavior Patterns**
  - **Auto Scaling**
    Auto Scaling helps adjust resources in real-time based on user behavior patterns, optimizing resource usage by matching supply with demand. This minimizes idle resources, reducing energy consumption and contributing to a more sustainable infrastructure.
  - **Elastic Load Balancing**
    Elastic Load Balancing distributes incoming application traffic across multiple targets, ensuring efficient utilization of resources. By optimizing the load on servers, it reduces the need for excess capacity, leading to more efficient energy use and improved sustainability.

- **Software and Architecture**
  - **AWS Design Principles**

Applying AWS design principles ensures that my application is built with sustainability in mind. This includes designing for efficiency, optimizing resource utilization, and incorporating practices that reduce the environmental impact of my application's operations, such as leveraging serverless architectures and efficient coding practices.

- **Data**
  - **Amazon EBS**
    Amazon Elastic Block Store (EBS) provides persistent storage for EC2 instances, optimized for performance and sustainability. Using EBS for my banking application ensures that data storage is efficient, reducing unnecessary storage overhead and energy consumption, contributing to a more sustainable data management strategy.
  - **Amazon S3**
    Amazon S3 offers scalable object storage with high durability and availability. S3's efficient storage mechanisms, such as intelligent tiering and lifecycle policies, help optimize data storage, reducing energy usage by automatically moving data to the most cost-effective and energy-efficient storage classes.

- **Hardware and Services**
  - **Amazon EC2**
    Amazon EC2 provides scalable compute capacity, optimized for energy efficiency. By selecting the appropriate instance types and leveraging EC2's energy-efficient hardware, my banking application can minimize its carbon footprint, contributing to overall sustainability.

- **Process and Culture**
  - **AWS CloudFormation**
    Although not currently in use, implementing AWS CloudFormation in the future can enhance sustainability by automating infrastructure provisioning and management. This reduces manual intervention, ensuring consistent and efficient resource usage, which aligns with best practices for sustainable cloud operations [15].

# REFERENCES


[1] "Amazon EC2 Instance types," *Amazon.com*. [Online]. Available: https://aws.amazon.com/ec2/instance-types/. [Accessed: July 20, 2024].

[2] "Application Load Balancer," *Amazon.com*. [Online]. Available: https://aws.amazon.com/elasticloadbalancing/application-load-balancer/. [Accessed: July 22, 2024].

[3] "What is Amazon VPC?," *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html. [Accessed: July 28, 2024].

[4] "Subnets for your VPC," *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/configure-subnets.html. [Accessed: July 28, 2024].

[5] "Enable VPC internet access using internet gateways," *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html. [Accessed: July 28, 2024].

[6] "NAT gateways," *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html. [Accessed: Aug 1, 2024].

[7] "Control traffic to your AWS resources using security groups," *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-groups.html. [Accessed: Aug 1, 2024].

[8] "AWS WAF," *Amazon.com*. [Online]. Available: https://aws.amazon.com/waf/. [Accessed: Aug 1, 2024].

[9] "Working with Amazon Aurora PostgreSQL," *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.AuroraPostgreSQL.html. [Accessed: Aug 1, 2024].

[10] "Amazon CloudWatch," *Amazon.com*. [Online]. Available: https://aws.amazon.com/cloudwatch/. [Accessed: Aug 3, 2024].

[11] "What Is AWS CloudTrail?," *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html. [Accessed: Aug 3, 2024].

[12] “Security Pillar - AWS Well-Architected Framework,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/welcome.html. [Accessed: Aug 3, 2024].

[13] “Reliability Pillar - AWS Well-Architected Framework,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/welcome.html. [Accessed: Aug 5, 2024].

[14] “Performance Efficiency Pillar - AWS Well-Architected Framework,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/wellarchitected/latest/performance-efficiency-pillar/welcome.html. [Accessed: Aug 5, 2024].

[15] “Sustainability Efficiency Pillar - AWS Well-Architected Framework,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/wellarchitected/latest/sustainability-pillar/sustainability-pillar.html. [Accessed: Aug 5, 2024].