AWS

Documentation for AWS-Based Web Application Architecture

DESIGNED BY: Malak Hussein Sayed Ali

CONTENTS

OVERVIEW

This document outlines the architecture and components used to build a highly available, scalable web application on AWS.
The project aims to improve the performance of a student records web application during peak admissions periods by leveraging AWS services.

OBJECTIVES

- Create an architectural diagram: Illustrate the interactions between AWS services.
- Estimate costs: Use the AWS Pricing Calculator for cost estimation.
- Deploy a functional web application: Host on a virtual machine backed by a relational database.
- Ensure high availability and scalability: Implement load balancing and auto-scaling.
- Secure the application: Configure network security and manage access permissions.

ARCHITECTURE COMPONENTS

This section provides an overview of the key AWS components used to build a highly available, scalable, and secure web application. Each component plays a crucial role in ensuring the application meets performance and security requirements, especially during peak usage periods. The architecture is designed following AWS best practices to optimize cost, scalability, and high availability.

Key Components

- **Virtual Private Cloud (VPC)**: Establishes a secure network environment with isolated resources, including public and private subnets across multiple Availability Zones for redundancy.
- **Amazon EC2 Instances**: Hosts web applications, providing compute power with flexibility.
- **Amazon RDS**: Manages the relational database independently from the application servers. Configured with MySQL engine in a multi-AZ deployment for high availability.
- **AWS Secrets Manager:** Secures sensitive information like database credentials, enabling secure access by the web application without hardcoding credentials.
- **Application Load Balancer (ALB):** Distributes incoming traffic across multiple EC2 instances to ensure load balancing and high availability.
- **Auto Scaling Group**: Automatically adjusts the number of EC2 instances based on demand to maintain performance.
- **AWS Cloud9**: Provides a cloud-based development environment for managing and deploying code.
- **AWS WAF (Web Application Firewall):** Protects the web application from common web exploits and bot traffic by filtering harmful requests before they reach the application.
- **Amazon CloudWatch:** Monitors application performance and resource utilization, providing insights and alerts to maintain operational health and efficiency.
- **AWS Backup:** Centralizes backup management for AWS resources, ensuring data protection and compliance through automated scheduling of backups for RDS and EC2 volumes.

Each component has been carefully selected and configured to ensure that the application remains responsive and secure, even under heavy load. The following sections will delve into the purpose and configuration of each component in detail.

1. VIRTUAL PRIVATE CLOUD (VPC) AND AZ

We selected the Virtual Private Cloud (VPC) to isolate the application within a secure network environment. This ensures that the web application is protected from unauthorized access and can only be reached through controlled entry points.

Components

- **Public Subnets:** These are used to host resources that need to be accessible from the internet, such as load balancers or bastion hosts. Public subnets are configured with a route to an internet gateway, allowing inbound and outbound traffic.
- **Private Subnets:** These host resources that do not require direct access from the internet, such as databases and application servers. Private subnets are configured without direct internet access, enhancing security by limiting exposure.

Availability Zones

Deploying subnets across multiple Availability Zones (AZs) ensures redundancy and high availability. If one AZ experiences an outage, the resources in other AZs can continue to operate, minimizing downtime.

Security Features

- **Network Access Control Lists (NACLs):** These act as a firewall for controlling traffic in and out of one or more subnets. NACLs provide an additional layer of security at the subnet level.
- **Security Groups:** These are used to control inbound and outbound traffic to AWS resources within the VPC. Security groups are stateful, meaning they automatically allow return traffic regardless of any outbound rules.

Benefits

- **Isolation:** The VPC provides a logically isolated network environment for the web application.
- **Security:** By using private subnets and security groups, sensitive components like databases are protected from direct internet access.
- **Scalability:** The VPC can be easily scaled by adding more subnets or modifying existing configurations to accommodate growth.

Conclusion

The use of a VPC with both public and private subnets across multiple Availability Zones provides a robust foundation for building secure, scalable, and highly available applications on AWS. This setup aligns with the best practices outlined in the AWS Well-Architected Framework, ensuring optimal performance and security for the student records web application.

2. AMAZON EC2 INSTANCES

We utilized Amazon EC2 instances to host the web server and application code, providing the necessary compute power to run the application efficiently. These instances ensure that the application can handle varying loads by scaling up or down as needed.

Configuration

- **Operating System:** The EC2 instances use the latest Ubuntu Amazon Machine Image (AMI). Ubuntu is chosen for its stability, security features, and wide support for various applications and development environments.
- **Software Installation:** The setup script provided in JavaScript automates the installation of the required software and dependencies on the EC2 instances. This script ensures consistency across all instances, reducing manual configuration errors and speeding up deployment.

Key Features

- **Scalability**: EC2 instances can be easily scaled to accommodate increased traffic, ensuring that the application remains responsive even during peak periods.
- **Flexibility**: A variety of instance types and sizes are available, allowing you to choose configurations that best meet your performance and cost requirements.
- **Integration with AWS Services:** EC2 integrates seamlessly with other AWS services such as Auto Scaling, which help enhance the application's scalability and availability.

Security

- **IAM Roles**: Assign IAM roles to EC2 instances to securely access other AWS services without embedding credentials in your application code. This enhances security by managing permissions centrally.
- **Security Groups:** Security groups Configured to control inbound and outbound traffic. Only allow necessary ports for web traffic (HTTPS) and database access (3306), minimizing exposure to potential threats.

Best Practices

- Regularly update the AMI to ensure that all instances have the latest security patches and updates.
- Use the monitoring tool ‘Amazon CloudWatch’ to keep track of instance performance and health.
- Implement backup strategies for data stored on EC2 instances using ‘AWS Backup’.

Conclusion

Amazon EC2 provides a robust platform for hosting web applications, offering flexibility, scalability, and integration with other AWS services. By leveraging these capabilities, the student records web application can efficiently handle high traffic volumes while maintaining security and performance standards.

3. AMAZON RDS (RELATIONAL DATABASE SERVICE)

Amazon RDS is used to manage the database independently from the web server, providing a managed database service that simplifies setup, operation, and scaling. By decoupling the database from the application server, it enhances the overall architecture's flexibility and reliability.

Configuration

- **Database Engine:** The MySQL engine is chosen for its popularity, ease of use, and compatibility with many applications, especially the university's admission platform that we are currently optimizing. It also offers a balance of performance and cost-effectiveness.
- **Multi-AZ Deployment:** The database is configured to be hosted in a Multi-AZ deployment. This setup provides automatic failover to a standby instance in another Availability Zone if the primary instance fails. This configuration ensures high availability and data durability, which are critical for maintaining application uptime during peak admissions periods.

Key Features

- **Managed Service:** RDS automates routine tasks such as backups, patching, and scaling, allowing developers to focus on application development rather than database maintenance.
- **Security:** RDS provides options to encrypt data at rest and in transit. Additionally, access can be restricted using security groups and IAM roles to ensure that only authorized applications can connect to the database.
- **Performance Monitoring:** RDS includes tools for monitoring database performance metrics, which help in optimizing queries and managing workloads effectively.

Security Considerations

- **Network Isolation:** The RDS instance is placed in a private subnet within the VPC, preventing direct access from the internet. This setup enhances security by limiting exposure to potential threats.
- **Access Control:** Only the web application is allowed to access the database through specific security group rules. This minimizes the risk of unauthorized access.
- **Secrets Management:** AWS Secrets Manager is used to store and manage database credentials securely. This approach avoids hardcoding sensitive information in application code.

Best Practices

- Regularly update the MySQL engine version to benefit from performance improvements and security patches.
- We implement automated backups using 'Amazon Backup' to ensure data recovery in case of failure.
- We used 'read replicas' to improve read performance and offload traffic from the primary database instance.

Conclusion

We used Amazon RDS to provide a robust solution for managing databases independently from application servers. By leveraging RDS's managed services, the web application can achieve better performance, security, and operational efficiency.

This configuration aligns with AWS' best practices for building scalable and secure applications.

4. AWS SECRETS MANAGER AND IAM ROLES

AWS Secrets Manager is used to securely store and manage sensitive information such as database credentials. This service helps eliminate the need to hardcode sensitive data in application code, enhancing security by reducing the risk of credential exposure.

Integration

- **Access by Web Application:** The web application retrieves credentials from Secrets Manager at runtime. This approach ensures that credentials are not stored in the application code or configuration files, minimizing security risks.
- **IAM Roles:** The EC2 instances running the web application are assigned IAM roles with permissions to access Secrets Manager. This setup allows the application to fetch secrets securely without embedding access keys in the code.

Key Features

- **Automatic Rotation:** Secrets Manager will automatically rotate credentials according to a defined schedule, ensuring that they remain secure over time without manual intervention.
- **Audit and Monitoring:** Integration with AWS CloudTrail allows for monitoring access to secrets, providing an audit trail for security compliance.
- **Encryption:** Secrets are encrypted at rest using AWS Key Management Service (KMS), ensuring that sensitive data is protected.

Security Considerations

- **Least Privilege Access:** IAM policies have been configured to grant only the necessary permissions for accessing specific secrets, adhering to the principle of least privilege.
- **Network Isolation:** Ensure that access to Secrets Manager is restricted to authorized network paths within your VPC.

Best Practices

- We are intending to regularly review and update IAM policies associated with accessing secrets.
- Use automatic rotation features to keep credentials fresh and reduce the risk of compromise.
- Monitor access logs through AWS CloudTrail for any unauthorized attempts to access secrets.

Conclusion

AWS Secrets Manager provides a robust solution for managing sensitive information securely. By integrating it with the web application, you ensure that database credentials are handled securely, aligning with best practices for application security and compliance.

5. APPLICATION LOAD BALANCER

We used The Application Load Balancer (ALB) to distribute incoming traffic across multiple Amazon EC2 instances. This ensures that no single instance is overwhelmed with requests, which enhances the application's availability and reliability. By balancing the load, the ALB helps maintain optimal performance, even during peak traffic periods.

Configuration

- **Listener Configuration:** The ALB uses listeners to check for connection requests from clients. We configured listeners to use HTTPS protocol, allowing for secure communication between clients and the application.
- **Target Groups**: EC2 instances are registered in target groups, which the ALB uses to route requests. This setup allows for flexible scaling and management of instances without affecting the load balancer's configuration.

Key Features

- **Health Checks:** The ALB performs health checks on registered instances to ensure that traffic is only routed to healthy instances. This feature helps maintain application uptime by automatically rerouting traffic away from unhealthy instances.
- **Security Features:** We integrated with the AWS Certificate Manager (ACM) allows for easy management of SSL/TLS certificates, enabling secure communication over HTTPS.

Security Considerations

- **Security Groups:** Configure security groups for the ALB to allow only necessary inbound traffic from clients and restrict outbound traffic to specific ports used by EC2 instances.
- **IAM Policies:** Use IAM policies to control access to the ALB configuration and ensure that only authorized users can make changes.

Best Practices

- We regularly monitor the load balancer metrics using Amazon CloudWatch to ensure optimal performance and quickly identify any issues.
- Use AWS WAF (Web Application Firewall) with the ALB to protect against common web exploits and attacks.
- Implement SSL/TLS termination at the load balancer level for secure communication without burdening backend instances with encryption tasks.

Conclusion

The Application Load Balancer plays a critical role in ensuring that the web application remains highly available and responsive under varying loads. By distributing traffic efficiently and providing advanced routing capabilities, the ALB enhances both performance and security in accordance with AWS best practices.

6. AUTO SCALING GROUP

The Auto Scaling Group (ASG) automatically adjusts the number of Amazon EC2 instances based on demand. This ensures that the application maintains optimal performance by scaling out during high traffic periods and scaling in during low traffic periods, thereby optimizing resource usage and costs.

Policy

- **Target Tracking Policy:** This policy automatically adjusts the number of instances to maintain a specified metric, such as average CPU utilization. By setting a target value, the ASG ensures that the application performs efficiently under varying loads.

Configuration

- **Launch Template:** Defines the configuration for EC2 instances, including AMI, instance type, and security groups. This template is used by the ASG to launch and terminate instances as needed.
- **Availability Zones:** The ASG is configured to deploy instances across multiple Availability Zones to enhance fault tolerance and availability.

Key Features

- **Dynamic Scaling:** Automatically adjusts capacity in response to changes in demand, ensuring that the application remains responsive and cost-effective.
- **Health Checks:** Regularly monitors instance health and replaces any unhealthy instances to maintain application reliability.

Best Practices

- Regularly review and adjust scaling policies based on application performance metrics.
- Use Amazon CloudWatch alarms to trigger scaling actions based on custom metrics.
- Monitor costs and adjust instance types or sizes as necessary to optimize spending.

Conclusion

The Auto Scaling Group ensures that the web application can handle fluctuating traffic efficiently while maintaining high availability and performance. By leveraging AWS's automatic scaling capabilities, you can optimize resource usage and costs effectively.

7. INTERNET GATEWAY

An Internet Gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between instances in your VPC and the internet. It serves as a bridge to connect the VPC to the internet, enabling resources in public subnets to receive inbound traffic from the internet and send outbound traffic to the internet.

Integration with Project Components

- **Public Subnets:** The Internet Gateway is attached to the VPC and used to route traffic from public subnets. This setup allows resources such as the Application Load Balancer (ALB) and bastion hosts within these subnets to be accessible from the internet.
- **Application Load Balancer:** The ALB, which is deployed in public subnets, uses the Internet Gateway to handle incoming requests from users accessing the web application over the internet.

Benefits

- **Internet Access:** Provides necessary internet access for resources that require it, such as updating software packages on EC2 instances or allowing users to access web applications.
- **Scalability and Redundancy:** The Internet Gateway is designed to be highly available and scalable, ensuring consistent performance even under high traffic loads.

Security Considerations

- **Route Tables:** Ensure that route tables are configured correctly to direct internet-bound traffic through the Internet Gateway for public subnets while keeping private subnets isolated.
- **Security Groups and NACLs:** Use security groups and Network Access Control Lists (NACLs) to control inbound and outbound traffic, ensuring only authorized traffic is allowed.

Conclusion

Integrating an Internet Gateway is essential for enabling internet connectivity for public-facing components of your architecture. It plays a critical role in ensuring that your web application can be accessed by users over the internet while maintaining security through proper configuration of network components.

8. AWS CLOUD9

We used AWS Cloud9 to act as a cloud-based integrated development environment (IDE) that provides a convenient platform for running AWS Command Line Interface (CLI) commands and scripts. This environment is particularly useful for developing, testing, and deploying applications in the AWS ecosystem, as it comes pre-configured with essential tools and SDKs.

Instance Type

- **t3.micro:** This instance type is chosen for cost efficiency. It offers a balance of compute power and memory suitable for development tasks without incurring high costs. The t3.micro instance is part of the AWS Free Tier, making it an economical choice for development purposes.

Key Features

- **Pre-configured Environment:** AWS Cloud9 includes a wide range of development tools, including support for multiple programming languages, a built-in terminal, and integration with AWS services.
- **Collaboration**: It allows multiple developers to collaborate in real-time on the same codebase, enhancing team productivity.
- **Direct Access to AWS Resources**: With AWS Cloud9, we can directly access and manage AWS resources using the integrated terminal and CLI, streamlining the development workflow.

Security Considerations

- **IAM Roles:** We assigned IAM roles to the Cloud9 environment to control access to AWS resources. This ensures that only authorized actions can be performed from within the environment.
- **Network Security:** Cloud9 environment is configured within a secure VPC setup to limit exposure to potential threats.

Best Practices

- Cloud9 environment is regularly updated to ensure it has the latest security patches and software updates.
- We used the version control systems ‘Git’ within Cloud9 to manage code changes effectively.
- Monitor usage and performance metrics to optimize resource allocation and cost management.

Conclusion

AWS Cloud9 provides a robust and flexible development environment that integrates seamlessly with AWS services. By using a t3.micro instance, we developed the application efficiently while keeping costs low. The platform's features support collaborative development and streamline workflows for deploying applications in the cloud.

9. AMAZON CLOUDWATCH

We implemented Amazon CloudWatch as a monitoring and management service built for developers, system operators, site reliability engineers (SRE), and IT managers. It provides data and actionable insights to monitor applications, respond to system-wide performance changes, optimize resource utilization, and get a unified view of operational health.

Integration with Project Components

- **EC2 Instances:** CloudWatch monitor EC2 instances for metrics such as CPU utilization, disk reads/writes, and network traffic. This helps in ensuring that the instances are performing optimally and can trigger scaling actions if needed.
- **RDS:** It monitors database performance metrics such as CPU load, memory usage, and I/O operations. This is crucial for maintaining database health and performance.
- **Application Load Balancer:** CloudWatch provides metrics on request counts, latency, and error rates for the load balancer, helping to ensure that traffic is distributed efficiently.
- **Auto Scaling:** It has been used to monitor scaling activities and ensure that the Auto Scaling policies are working as intended.
- **AWS WAF:** Monitor WAF logs and set up alerts for suspicious activities or rule violations.

Key Features

- **Alarms:** Alarms have been set up to automatically notify us of performance issues or threshold breaches, enabling proactive management.
- **Logs:** CloudWatch collect and monitor log files from EC2 instances and other AWS services. This helps in troubleshooting application issues.
- **Dashboards:** Custom dashboards have been created to visualize key metrics from different AWS services in one place for easy monitoring.

Benefits

- **Real-Time Monitoring:** Provides real-time insights into application performance and resource utilization.
- **Automated Responses:** Integrates with AWS Lambda to automate responses to specific conditions.
- **Cost Management:** Helps in optimizing resource usage by providing insights into underutilized resources.

Conclusion

Amazon CloudWatch is a comprehensive monitoring tool that integrates seamlessly with the AWS services used in this project. It provides the necessary insights and tools to ensure that the web application remains highly available, scalable, and high-performing.

10. AWS BACKUP INTEGRATION

AWS Backup provides centralized backup management for AWS services, allowed us to automate and schedule backups. It ensures data protection and compliance by creating regular backups of your resources.

Integration with Project Components

- **Amazon RDS:** We used AWS Backup to schedule automated backups of the RDS database, ensuring that you can recover data in case of accidental deletion or corruption.
- **EC2 Instances:** While the application code should ideally be stored in a version control system, AWS Backup has been used to back up EC2 instance volumes, preserving the state of the application server.

Benefits

- **Centralized Management:** Manage all backups from a single dashboard, simplifying operations and monitoring.
- **Automated Scheduling:** Set up regular backup schedules to ensure data is consistently backed up without manual intervention.
- **Compliance and Retention:** Define retention policies to meet compliance requirements and manage storage costs effectively.

Conclusion

Integrating AWS Backup into the project enhances data protection and operational resilience. It provides peace of mind by ensuring that critical data and configurations are securely backed up and easily recoverable.

11. AWS WAF INTEGRATION

We implemented 'AWS WAF' which is a web application firewall that helps protect the web applications from common web exploits and bot traffic that can affect availability, compromise security, or consume excessive resources. It provides an additional layer of security by allowing you to control the traffic that reaches your application.

Integration with Project Components

- **Application Load Balancer:** AWS WAF are integrated with the Application Load Balancer (ALB) to inspect incoming traffic and block or allow requests based on the rules we specified. This setup enhances the security of the web application by filtering potentially harmful traffic before it reaches the EC2 instances.

Benefits

- **Protection Against Common Attacks**: AWS WAF helps protect against SQL injections, cross-site scripting (XSS), and other common attacks by defining custom rules that filter out malicious requests.
- **Bot Traffic Management:** It can detect and block unwanted bot traffic, which can help in reducing unnecessary load on your application and preserving resources.
- **Customizable Rules:** Custom rules have been created to tailor the protection to the application needs. It also offers managed rule sets that are regularly updated to protect against new threats.

Security Considerations

- **Rule Management:** Regularly review and update WAF rules to ensure they remain effective against evolving threats.
- **Monitoring and Logging:** Use AWS CloudWatch to monitor WAF logs and set up alerts for suspicious activities or rule violations.

Conclusion

Integrating AWS WAF into the project provides enhanced security for the web application by protecting against a variety of web-based attacks. This integration aligns with the project's goal of creating a secure, high-performing application environment in line with AWS' best practices.

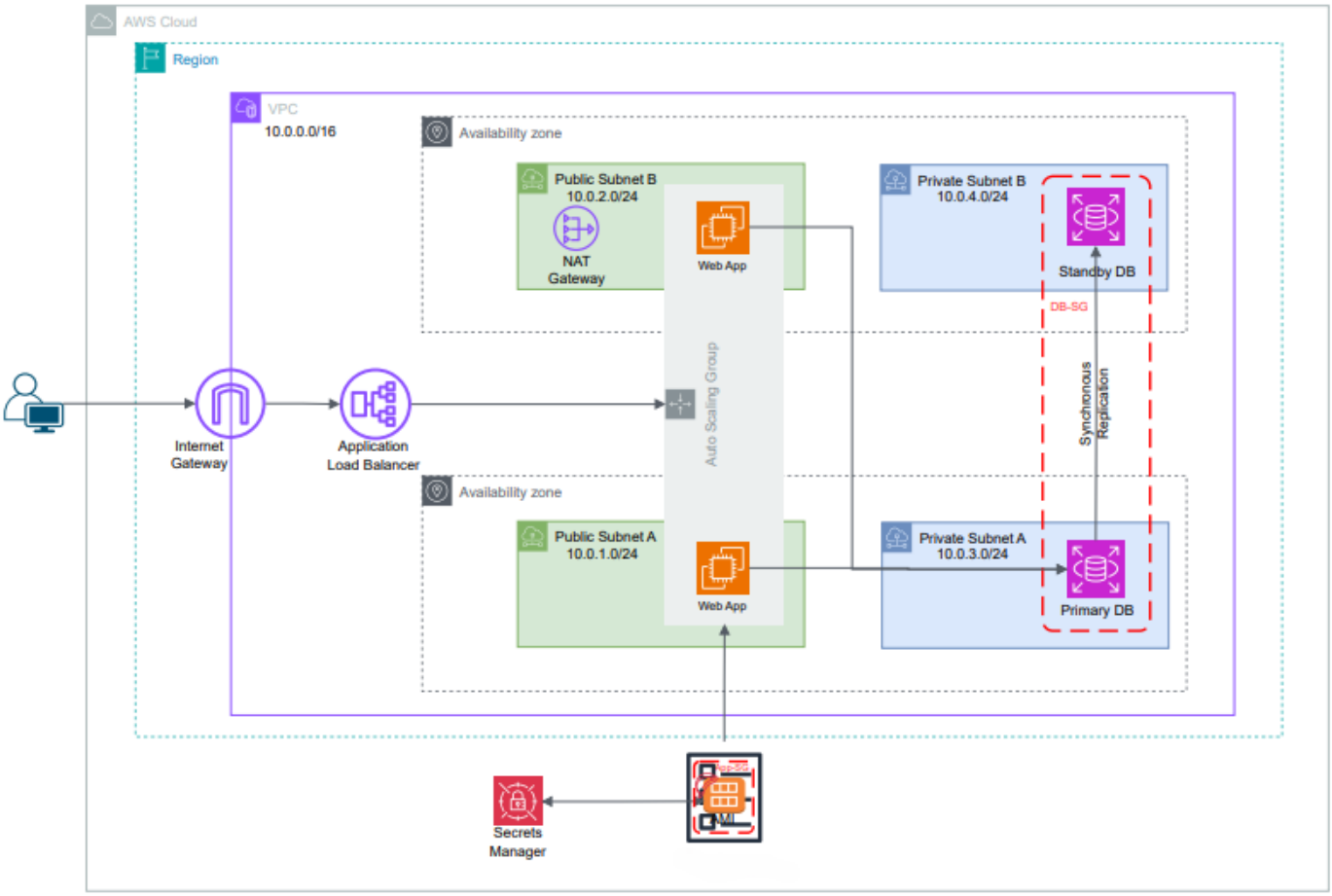
IMPLEMENTATION PHASE

The implementation phase outlines the steps necessary to build and deploy a highly available, scalable, and secure web application using AWS services. This phase is divided into several tasks that should guide through setting up the infrastructure, deploying the application, and ensuring it meets the project requirements.

PHASE 1: PLANNING THE DESIGN AND ESTIMATING COST

1. **Creating an Architectural Diagram**
   - Use AWS Architecture Icons to create a diagram that illustrates the interaction between AWS services like EC2, RDS, VPC, ALB, Auto Scaling, and Secrets Manager.
   - Ensure the diagram reflects high availability and security considerations.

Architecture for University Admission Web Application



2. **Developing a Cost Estimate**
   - Use the AWS Pricing Calculator to estimate the cost of running the solution in the us-east-1 Region for 12 months.
   - Consider costs for EC2 instances, RDS, ALB, Auto Scaling, and any other relevant services.

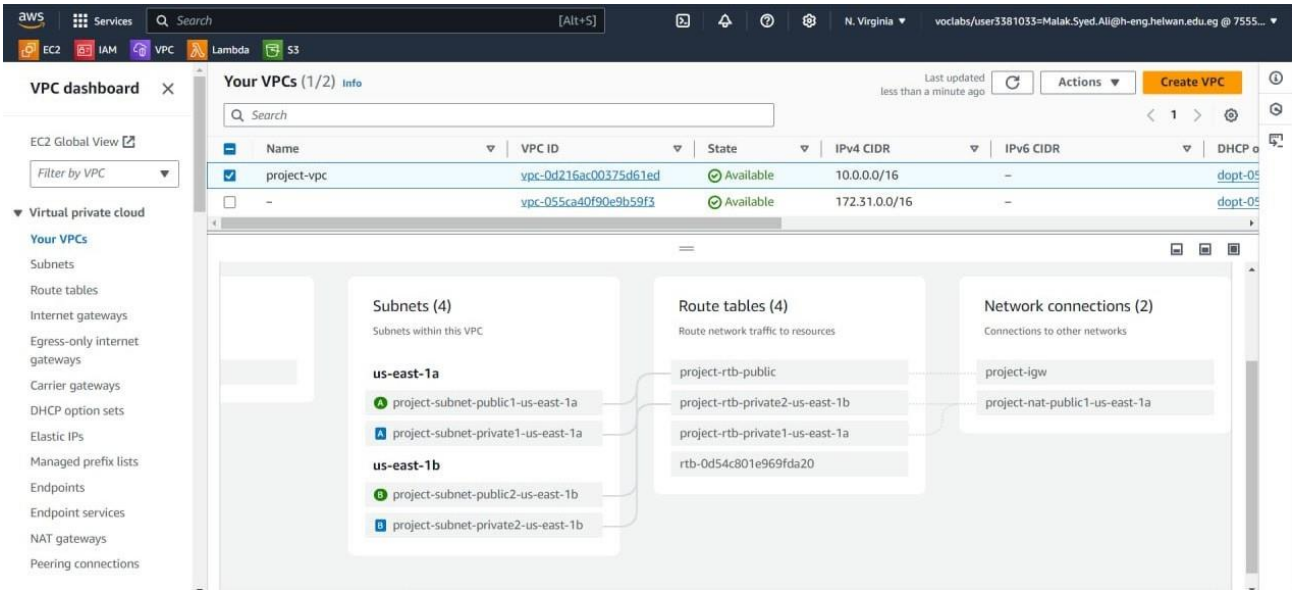
PHASE 2: CREATING BASIC FUNCTIONAL WEB APPLICATION

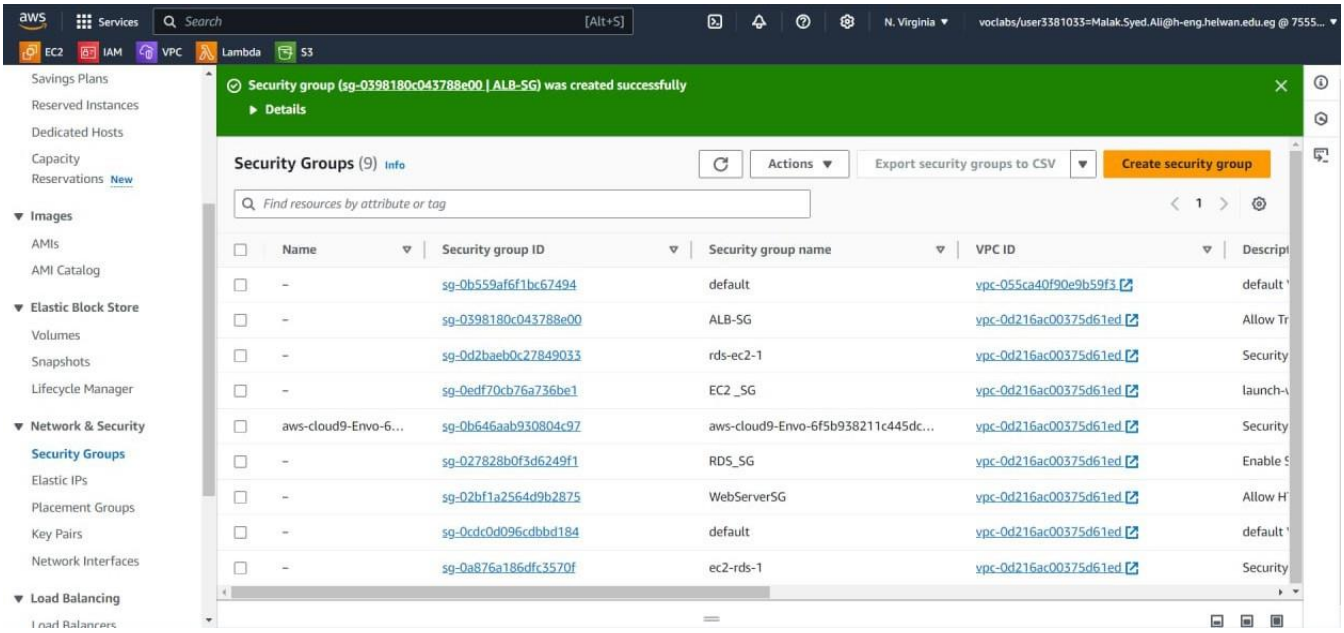
1. Creating a Virtual Network

- Set up a VPC '10.0.0.0/16' with public and private subnets across two Availability Zones.

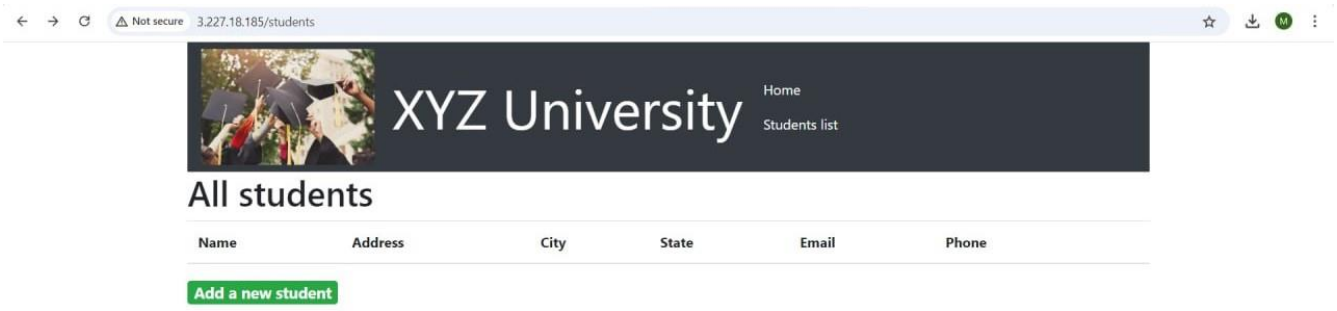
	AZ-A	*AZ-B*
VPC	10.0.0.0/16	
Public	10.0.1.0/24	10.0.2.0/24
Private	10.0.3.0/24	10.0.4.0/24

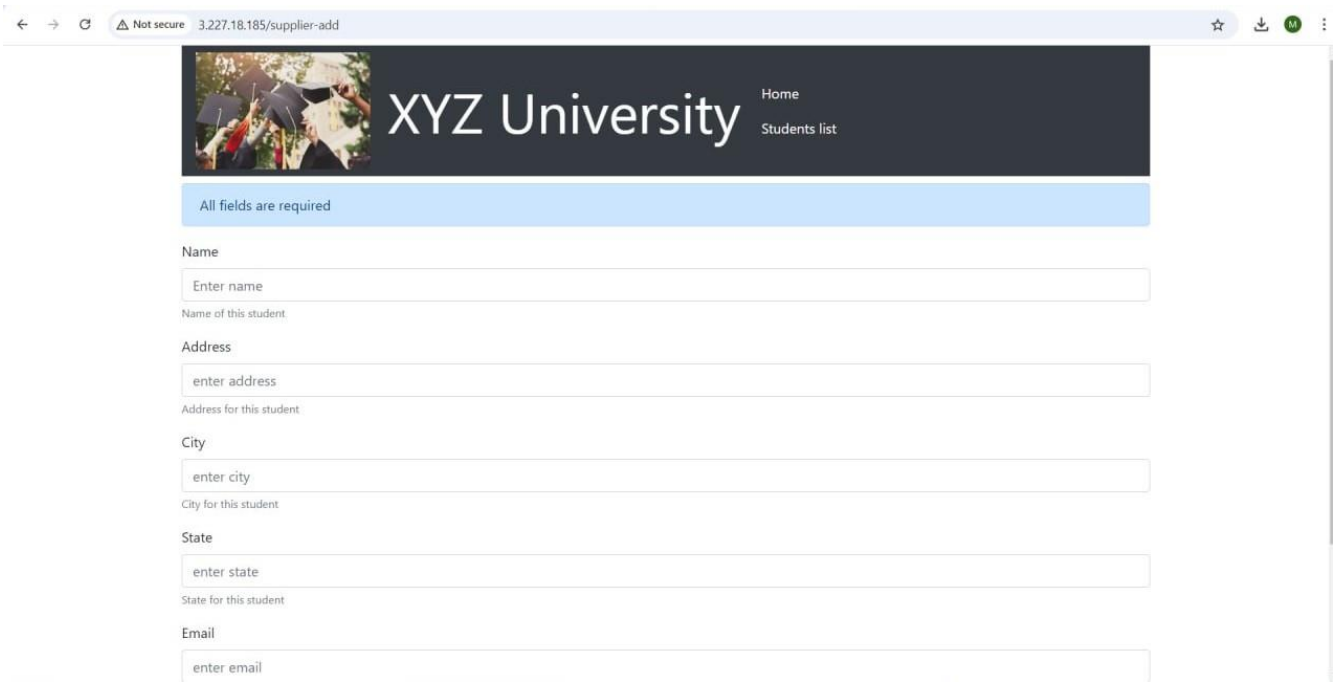
- Configure necessary networking resources like internet gateways and route tables.

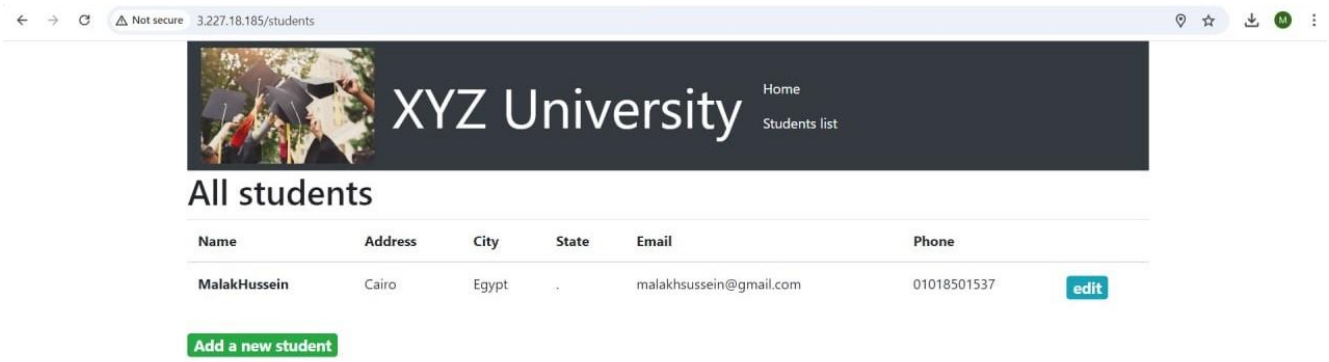


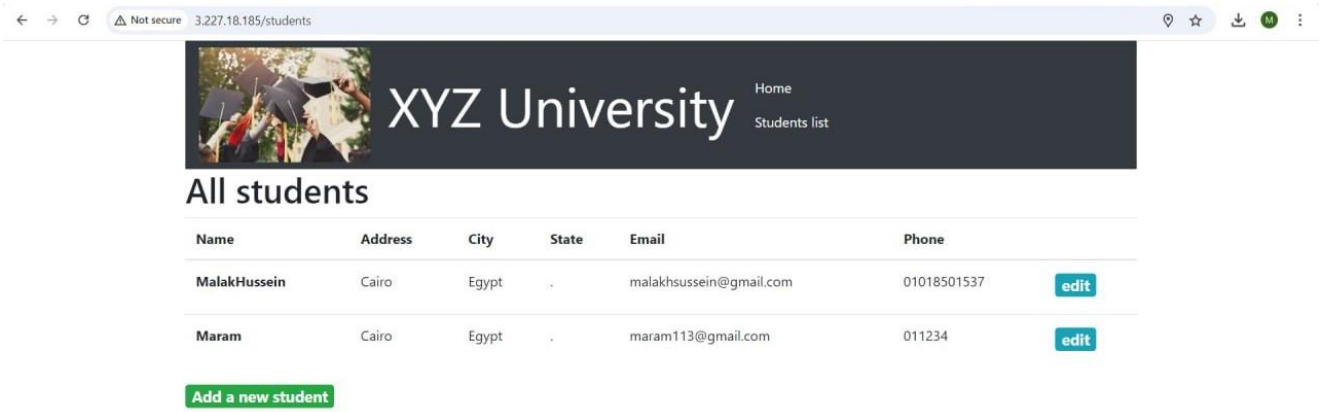


2. **Creating a Virtual Machine**
   - Launch an EC2 instance using the latest Ubuntu AMI.
   - Use provided JavaScript code to install the web application and database on this instance.
3. **Testing the Deployment**
   - Access the web application via the EC2 instance's IPv4 address.
   - Perform basic operations like viewing, adding, deleting, or modifying student records to ensure functionality.



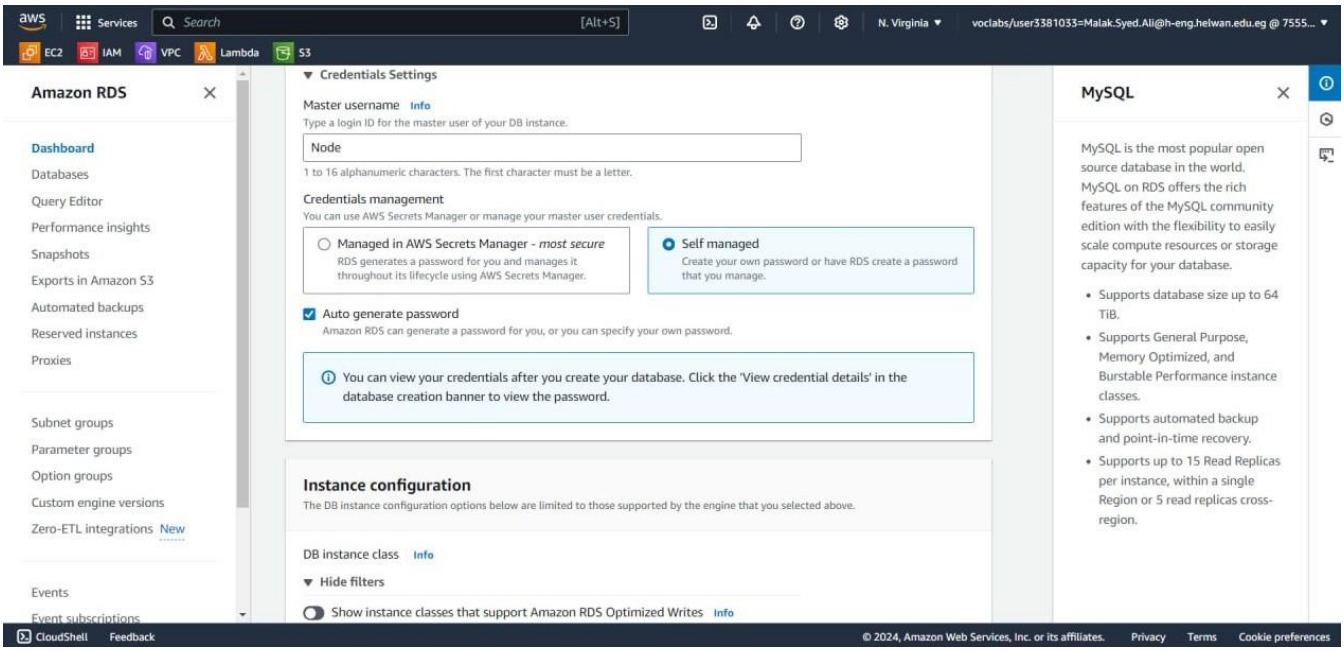




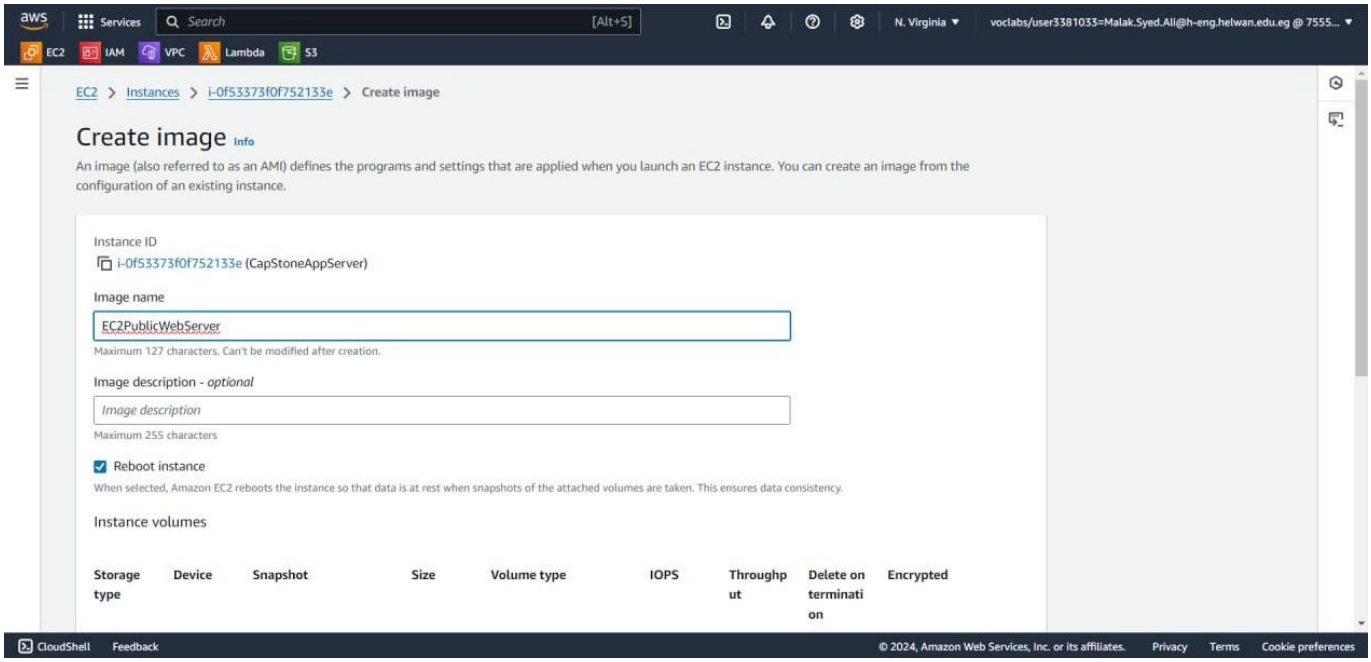


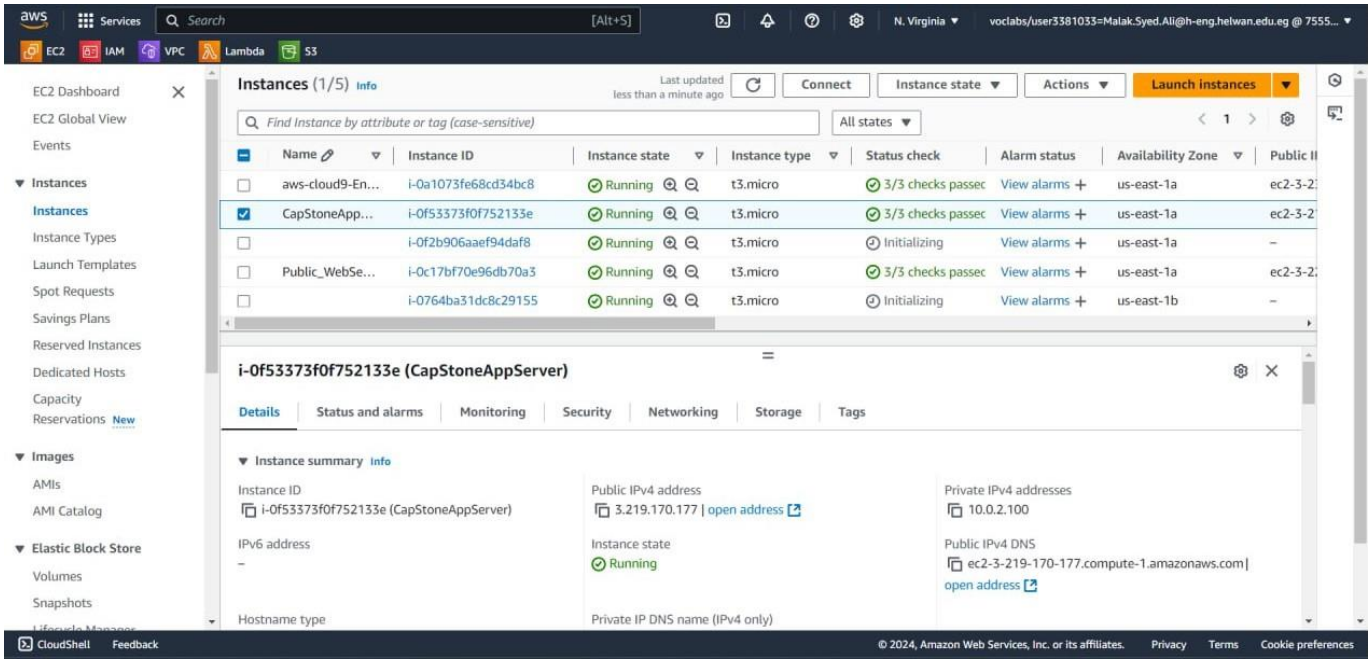
PHASE 3: DECOUPLING APPLICATION COMPONENTS

1. **Changing VPC Configuration**
   - Update VPC settings to support separate hosting of the database.
   - Ensure private subnets are configured in at least two Availability Zones.
2. **Creating and Configuring Amazon RDS Database**
   - Set up an RDS instance running MySQL in two Availability Zone.
   - Restrict database access to only the web application.



3. **Configuring Development Environment**
   - Provision an AWS Cloud9 environment using a t3.micro instance for running CLI commands.
   - Connect via SSH for secure access.
4. **Provisioning Secrets Manager**
   - Use AWS Secrets Manager to store database credentials securely.
   - Integrate Secrets Manager with the web application for credential retrieval.
5. **Provisioning a New Instance for Web Server**
   - Launch a new EC2 instance for hosting the web application.



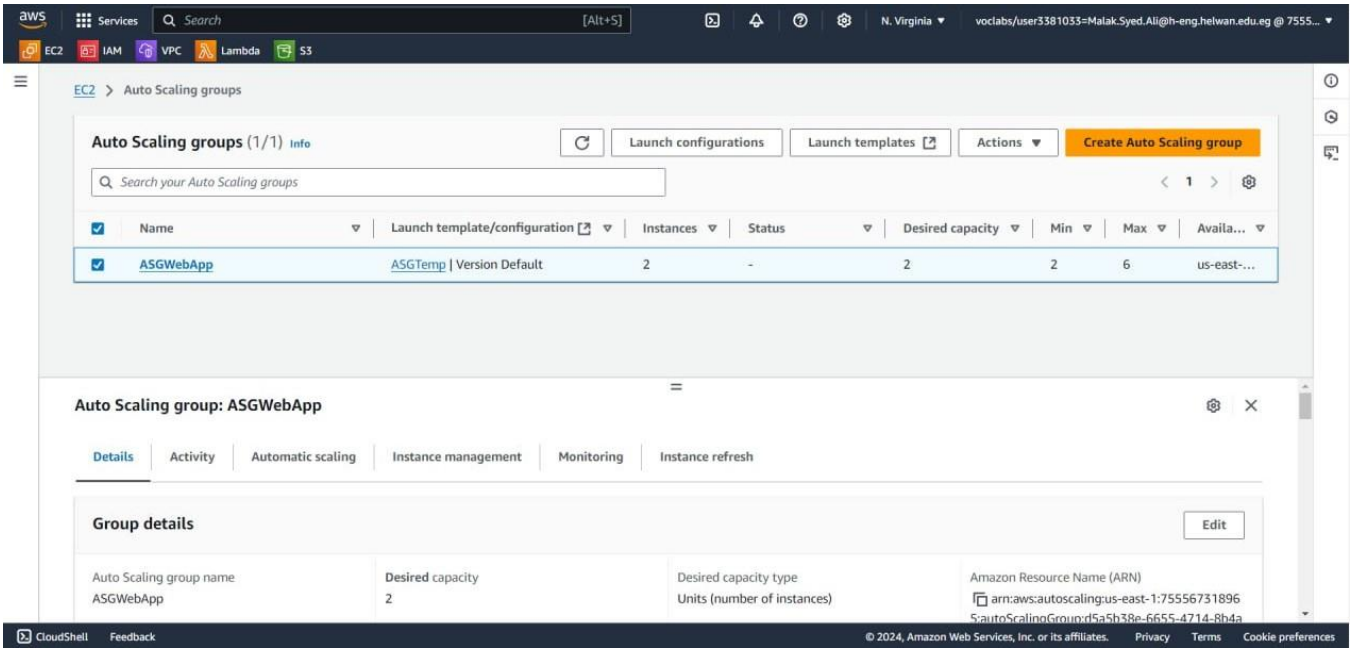


- Attach IAM role for secure access to Secrets Manager.

6. **Testing the Application**
   - Verify that all functionalities work as expected with the new database setup.

PHASE 4: IMPLEMENTING HIGH AVAILABILITY AND SCALABILITY

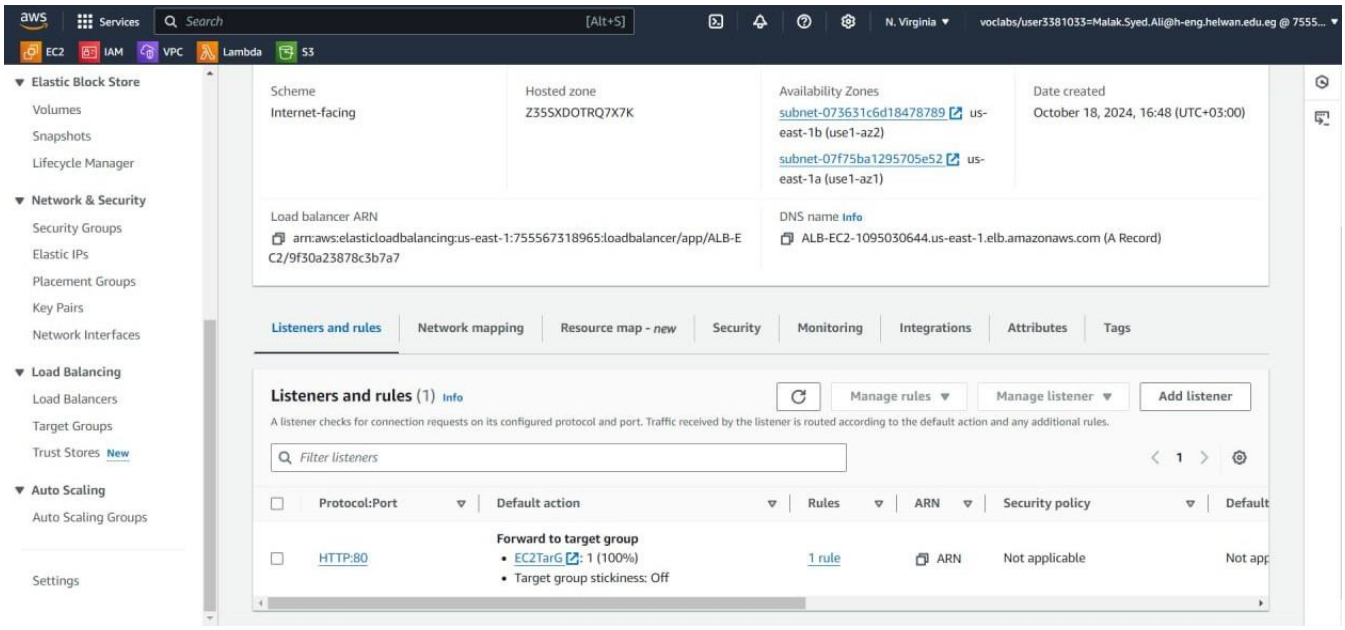
1. **Creating an Application Load Balancer**
   - Deploy an ALB across multiple Availability Zones.
2. **Implementing Amazon EC2 Auto Scaling**
   - Create a launch template and configure an Auto Scaling group.



3. **Accessing the Application**
    - Test application functionality through the load balancer's URL.
4. **Load Testing the Application**
    - Conduct load testing using provided scripts to ensure scaling capabilities are effective under stress conditions.

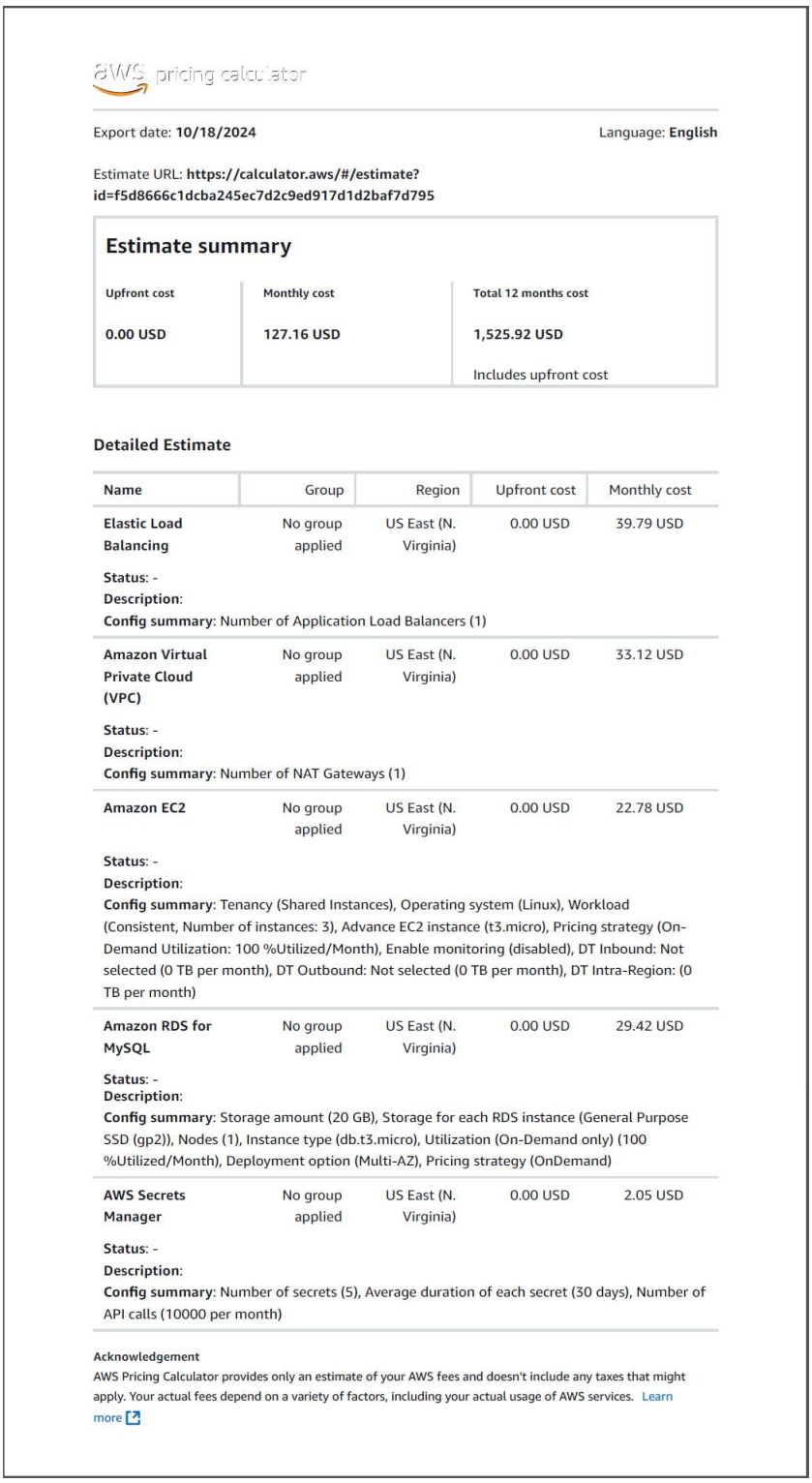
Security Considerations

- Ensure that only necessary ports are open on security groups.
- Use IAM roles to manage permissions between services securely.
- Store sensitive information like database credentials in Secrets Manager.



Cost Optimization

- Use t3.micro instances where applicable to minimize costs.
- Monitor usage with AWS Budgets to prevent exceeding budget limits.



aws pricing calculator

Export date: **10/18/2024** Language: **English**

Estimate URL: **https://calculator.aws/#/estimate?id=f5d8666c1dcba245ec7d2c9ed917d1d2baf7d795**

Estimate summary

Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	**127.16 USD**	**1,525.92 USD** Includes upfront cost

Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Elastic Load Balancing	No group applied	US East (N. Virginia)	0.00 USD	39.79 USD
Status: - **Description**: **Config summary**: Number of Application Load Balancers (1)				
Amazon Virtual Private Cloud (VPC)	No group applied	US East (N. Virginia)	0.00 USD	33.12 USD
Status: - **Description**: **Config summary**: Number of NAT Gateways (1)				
Amazon EC2	No group applied	US East (N. Virginia)	0.00 USD	22.78 USD
Status: - **Description**: **Config summary**: Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 3), Advance EC2 instance (t3.micro), Pricing strategy (On-Demand Utilization: 100 %Utilized/Month), Enable monitoring (disabled), DT Inbound: Not selected (0 TB per month), DT Outbound: Not selected (0 TB per month), DT Intra-Region: (0 TB per month)				
Amazon RDS for MySQL	No group applied	US East (N. Virginia)	0.00 USD	29.42 USD
Status: - **Description**: **Config summary**: Storage amount (20 GB), Storage for each RDS instance (General Purpose SSD (gp2)), Nodes (1), Instance type (db.t3.micro), Utilization (On-Demand only) (100 %Utilized/Month), Deployment option (Multi-AZ), Pricing strategy (OnDemand)				
AWS Secrets Manager	No group applied	US East (N. Virginia)	0.00 USD	2.05 USD
Status: - **Description**: **Config summary**: Number of secrets (5), Average duration of each secret (30 days), Number of API calls (10000 per month)				

Acknowledgement

AWS Pricing Calculator provides only an estimate of your AWS fees and doesn't include any taxes that might apply. Your actual fees depend on a variety of factors, including your actual usage of AWS services. Learn more

Conclusion

This implementation plan provides a structured approach to deploying a robust web application on AWS, ensuring it meets all functional, security, scalability, and providing an improved user experience during peak periods.