BUILDING A HIGHLY AVAILABLE AND SCALABLE WEB APPLICATION FOR EXAMPLE UNIVERSITY ON AWS

Abstract

This project develops a scalable and highly available web application for Example University on AWS, utilizing Amazon EC2 for hosting and Amazon RDS with MySQL for data storage. It features load balancing with an Application Load Balancer, automatic scaling via an Auto Scaling Group, and high availability through Multi-AZ deployments. Security is enhanced using Amazon VPC, Security Groups, and AWS Secrets Manager for credential management, ensuring a robust and secure solution for managing student records.

Ali Ahmed mohamed

alielnashar64@gmail.com

Contents

2. Cost Analysis

- Estimation of costs using the AWS Pricing Calculator for 12 months
- Breakdown of service costs (EC2, RDS, ALB, etc.)

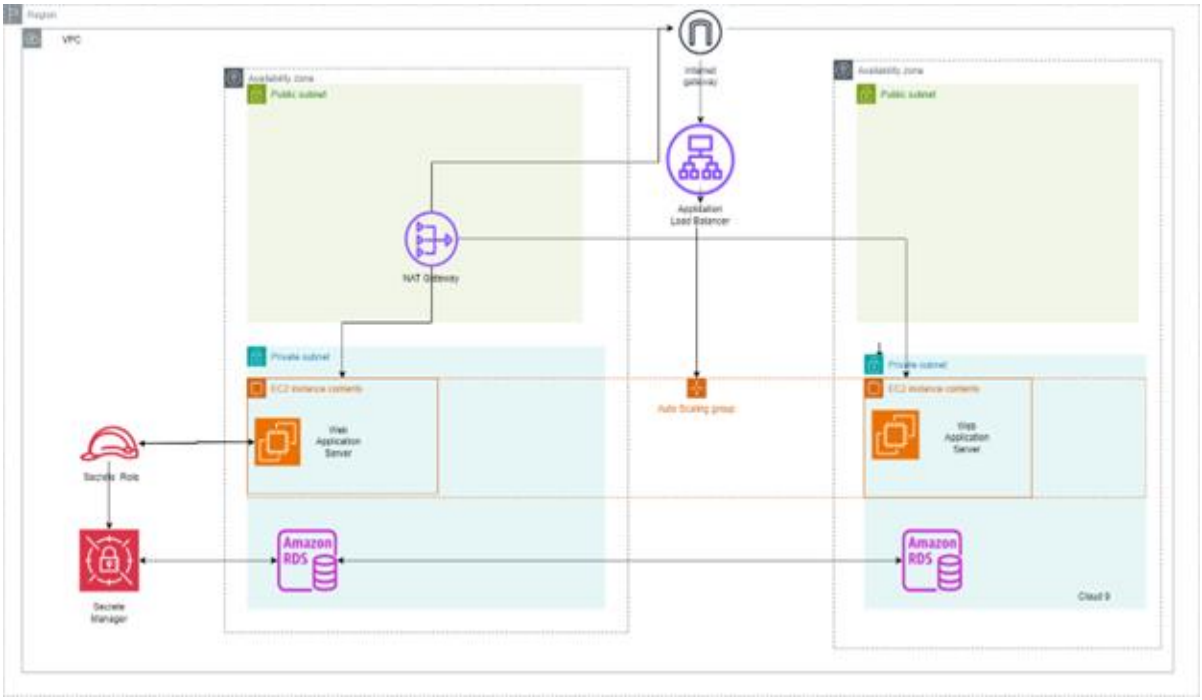
3. Implementation Steps

- Step 1: VPC creation
- Step 2: Security Group setup
- Step 3: EC2 instance deployment
- Step 4: Amazon RDS setup
- Step 5: Secrets Manager configuration
- Step 6: Application Load Balancer setup
- Step 7: Auto Scaling Group creation

4. Conclusion

- Summary of how the solution meets high availability, scalability, load balancing, and security requirements
- Cost optimization and resilience of the application to high traffic

1.Architecture diagram of the solution



2.1 Functional Requirement:

Requirement: The solution must allow users to view, add, delete, and modify student records without delay.

Solution:

- **Amazon EC2**: We used EC2 instances to host the web application. The web application was deployed on a Linux (Ubuntu) server, which is part of the public subnet within a Virtual Private Cloud (VPC).
- **Amazon RDS (MySQL)**: The database used to store and manage student records was deployed on Amazon RDS with a MySQL engine. This managed database service ensures fast and reliable performance.

AWS Services Explanation:

- **Amazon EC2**: Provides resizable compute capacity, allowing us to run virtual machines to host our web application.
- **Amazon RDS**: A fully managed relational database service that automates backups, patching, and scaling.

2.2 Load Balanced Requirement:

Requirement: The solution must balance user traffic to avoid overloaded or underutilized resources.

Solution:

- **Application Load Balancer (ALB)**: We used an ALB to distribute incoming web traffic across multiple EC2 instances. This ensures that user traffic is balanced and no single EC2 instance becomes overloaded during peak periods.

AWS Services Explanation:

- **Application Load Balancer (ALB)**: Distributes incoming traffic across multiple targets (EC2 instances) in different Availability Zones, enhancing the application's fault tolerance and scalability.

2.3 Scalability Requirement:

Requirement: The solution must automatically scale to meet the demands of peak periods.

Solution:

- **Auto Scaling Group (ASG)**: We configured an Auto Scaling group to dynamically adjust the number of EC2 instances based on user traffic. The ASG automatically adds or removes instances depending on the load.

AWS Services Explanation:

- **Auto Scaling**: Ensures that you have the right number of EC2 instances running to handle the traffic load. It adds more instances during high traffic periods and reduces instances during lower traffic periods.

2.4 High Availability Requirement:

Requirement: The solution must have limited downtime and remain available when a web server becomes unavailable.

Solution: To achieve high availability, we implemented the following:

- **Application Load Balancer (ALB):** We used an ALB to distribute incoming web traffic across multiple EC2 instances, each deployed in different Availability Zones. This ensures that even if one instance or Availability Zone goes down, traffic is automatically routed to healthy instances in other zones.
- **Auto Scaling Group (ASG):** The Auto Scaling Group dynamically adjusts the number of EC2 instances based on traffic demand. When user traffic increases, the ASG automatically launches additional instances to handle the load. When traffic decreases, the ASG scales down, reducing unnecessary costs.

AWS Services Explanation:

- **Application Load Balancer (ALB):** Ensures even traffic distribution across multiple EC2 instances, improving the fault tolerance of the application.
- **Auto Scaling Group (ASG):** Automatically adjusts the number of EC2 instances, scaling out during high traffic periods and scaling in during low traffic to optimize cost.

2.5 Security Requirement:

Requirement: The database must not be accessible directly from public networks, and the web application must be publicly accessible over the internet without hardcoding credentials.

Solution:

- **Amazon VPC with Security Groups**: The web application is hosted in a public subnet, while the RDS database is placed in a private subnet. Security groups were used to restrict access to necessary ports (80 for web traffic, 3306 for database access, but limited only to the web servers).
- **AWS Secrets Manager**: Database credentials are stored in AWS Secrets Manager, which the application accesses securely instead of hardcoding credentials.

AWS Services Explanation:

- **Security Groups**: Firewall rules that control inbound and outbound traffic for EC2 instances and RDS
- **AWS Secrets Manager**: Securely manages and retrieves credentials, ensuring the application never hardcodes sensitive information.

3. Cost Analysis:

We used the **AWS Pricing Calculator** to estimate the cost of running this architecture in **us-east-1** for 12 months. Here's a summary of the projected costs:

Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Amazon Virtual Private Cloud (VPC) **Status:** - **Description:** vpc network **Config summary:** Number of NAT Gateways (1)	No group applied	US East (N. Virginia)	0.00 USD	33.75 USD
Elastic Load Balancing **Status:** - **Description:** **Config summary:** Number of Application Load Balancers (1)	No group applied	US East (N. Virginia)	0.00 USD	28.11 USD
Amazon EC2 **Status:** - **Description:** **Config summary:** Tenancy (Shared Instances), Operating system (Ubuntu Pro), Workload (Consistent, Number of instances: 1), Advance EC2 instance (t2.micro), Pricing strategy (1yr No Upfront), 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)	No group applied	US East (N. Virginia)	0.00 USD	7.08 USD

Contact your AWS representative: Contact Sales

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

Language: **English**

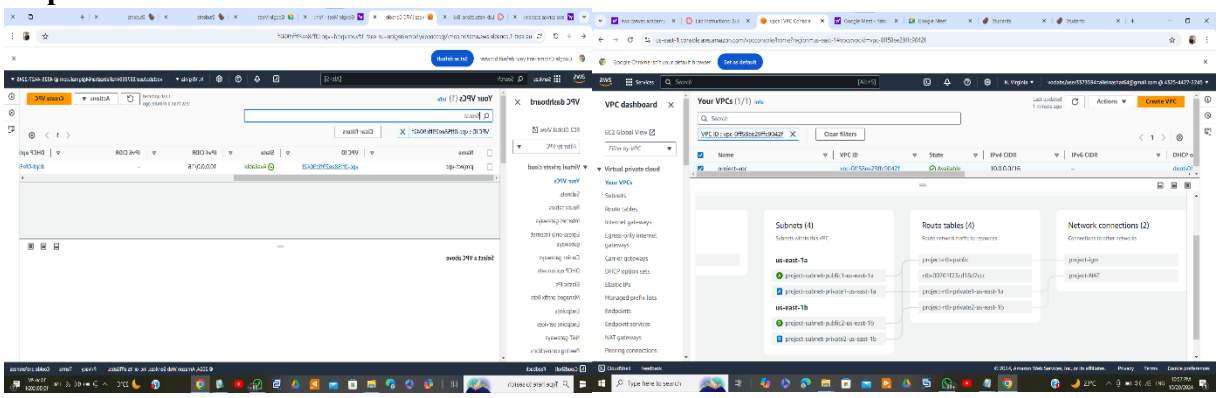
Estimate URL: **https://calculator.aws/#/estimate?id=dd71affb2c713c2304dedc03074bcb783551d76d**

Estimate summary

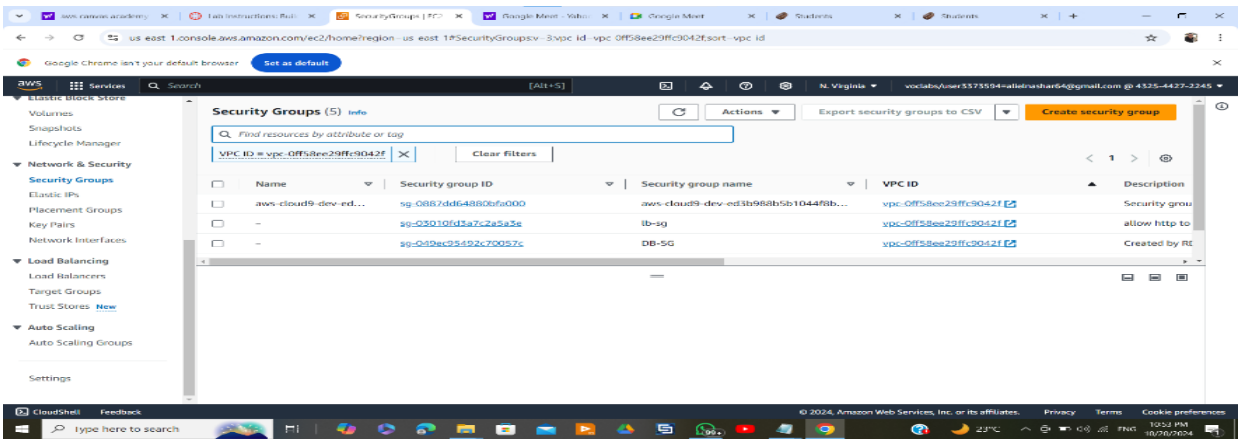
Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	128.80 USD	1,545.60 USD Includes upfront cost

3. Steps Taken:

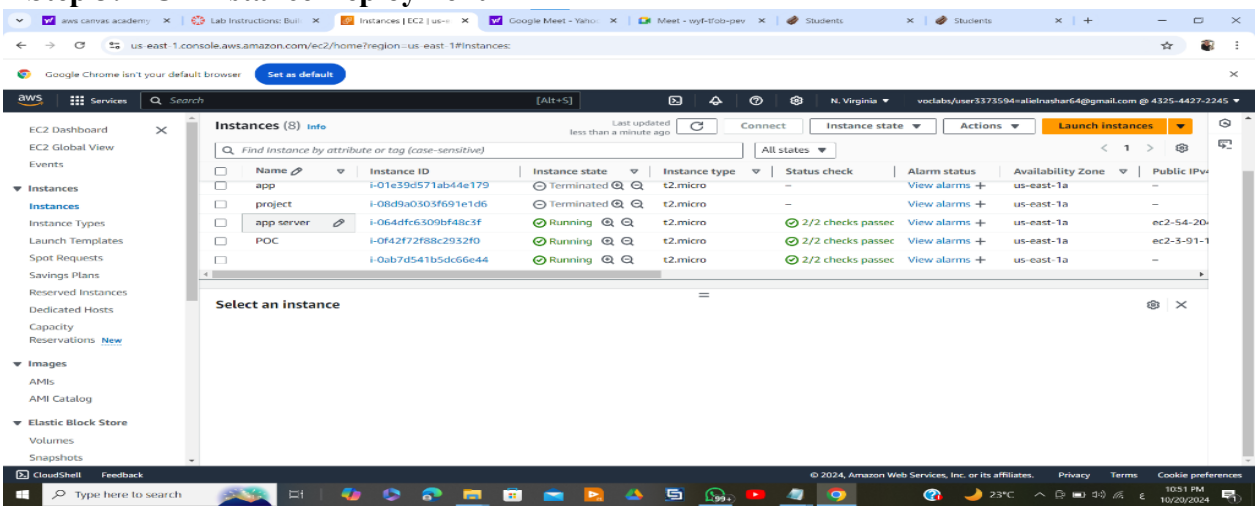
Step 1: VPC Creation



- **Step 2: Create SG**

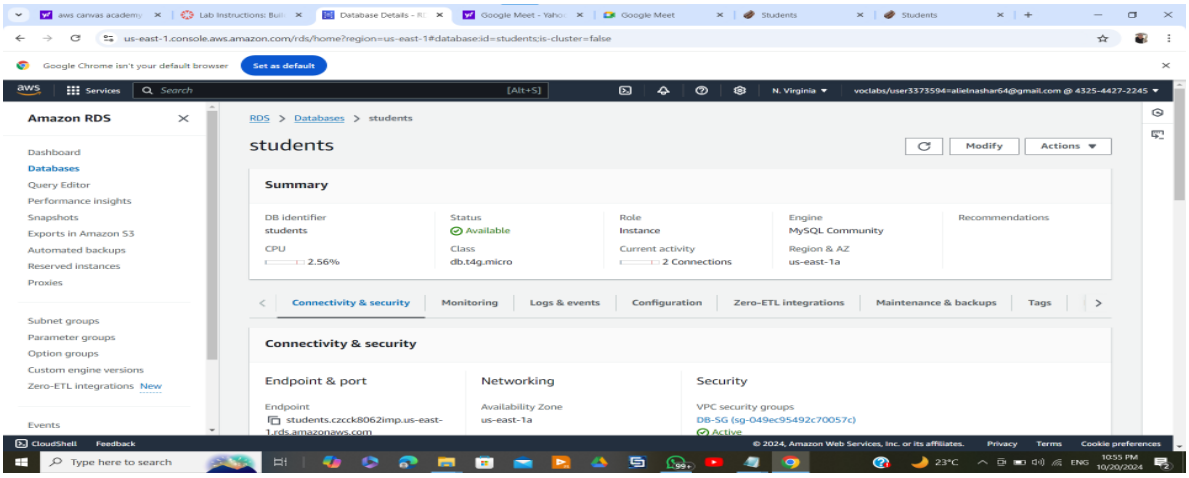


Step 3: EC2 Instance Deployment



- **Step 4: Amazon RDS Database Setup**

  An Amazon RDS instance was created using MySQL in the private subnet, with connections limited only to the web servers.



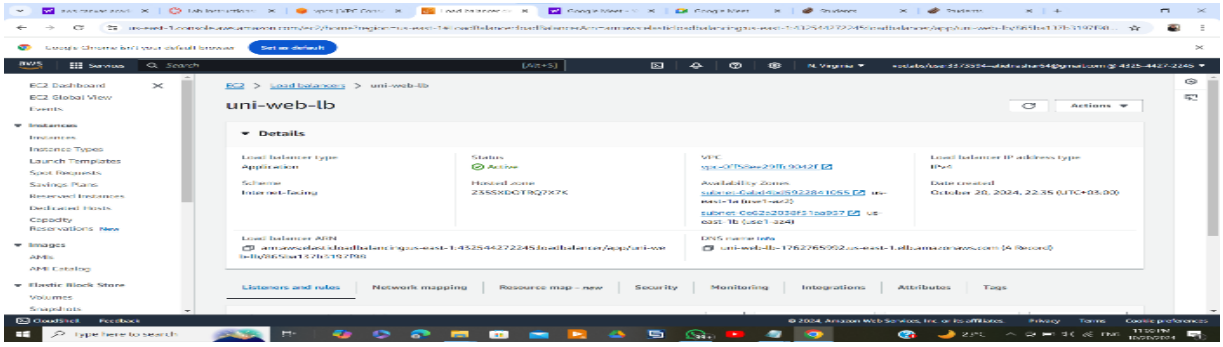
- **Step 5: Secrets Manager Configuration**

  AWS Secrets Manager was used to store the database credentials securely, and the web application was configured to retrieve these credentials.

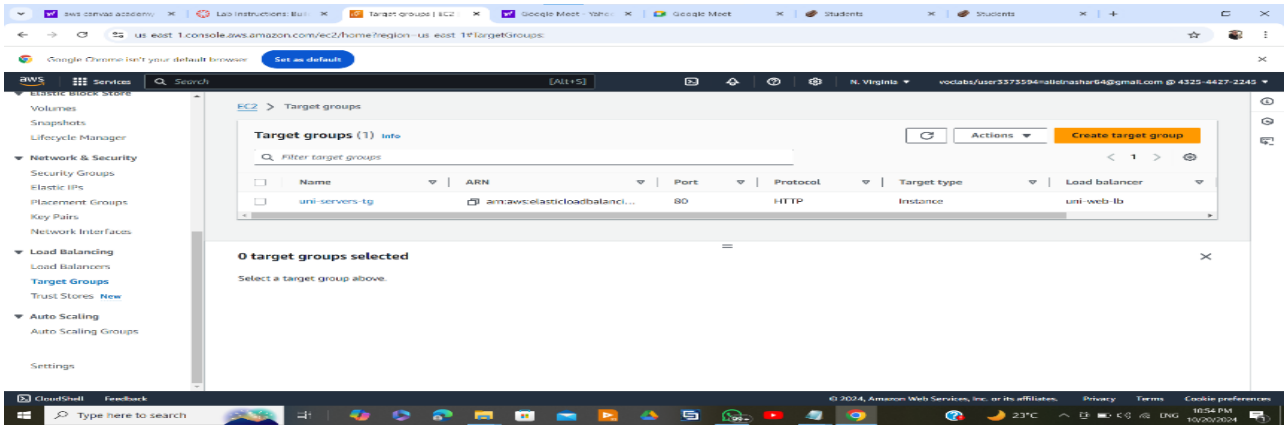
```
{
    "ARN": "arn:aws:secretsmanager:us-east-1:626828110804:secret:Mydbnewsecret-8YhdQ0",
    "Name": "Mydbnewsecret",
    "VersionId": "8ad0cca9-75f0-47f2-97cc-0d920c1efb1f"
}
```

- **Step 6: Application Load Balancer Setup**

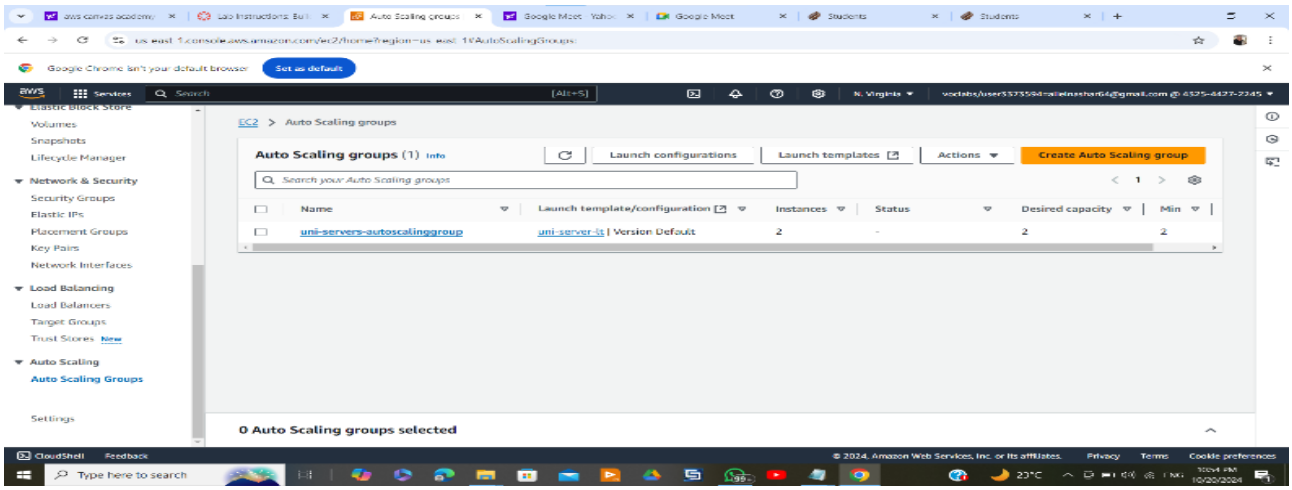
An ALB was configured to distribute traffic across multiple instances in the Auto Scaling group.



- **Step 7: Create Target Group**



- **Step 8: create auto scaling group**



4. Conclusion:

The solution designed and implemented on AWS meets the project requirements for high availability, scalability, load balancing, and security. By using a combination of EC2, RDS, ALB, Auto Scaling, and Secrets Manager, we successfully built a POC web application that supports high user traffic, is resilient to failures, and keeps costs optimized.