

# BUILDING A HIGHLY AVAILABLE AND SCALABLE WEB APPLICATION

Ahmed Hossam El-Dien Rabie

[Document  
subtitle]

# Contents

## 1.Introduction

- Overview of the project
- Objectives of the solution

### 1.1 Architecture Diagram:

- Visual representation of the architecture components

### 1.2 Functional Requirements:

- Amazon EC2 for web application hosting
- Amazon RDS for managing student records

### 1.3 Load Balancing:

- Use of Application Load Balancer (ALB) for traffic distribution
- Fault tolerance and scalability

### 1.4 Scalability:

- Auto Scaling Group (ASG) for dynamic scaling based on user traffic
- Automatic adjustment of instances during peak and low periods

### 1.5 High Availability:

- Ensuring uptime through ALB and ASG across multiple Availability Zones (AZs)
- Traffic routing during server or AZ failures

### 1.6 Security:

- Use of Amazon VPC for network isolation
- Security groups to restrict access
- AWS Secrets Manager for managing database credentials securely

## **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**

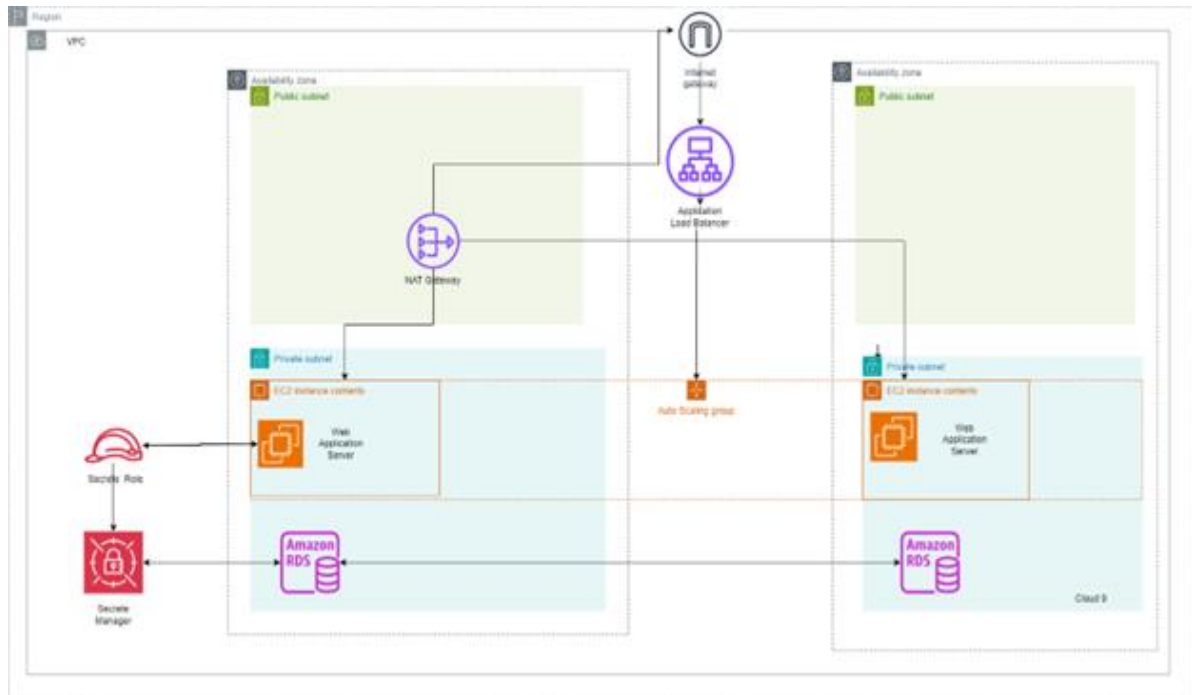
- Step 1: VPC creation
- Step 2: Security Group setup
- Step 3: EC2 instance deployment
- Step 4: Amazon RDS setup
- Step 5: Secrets Manager configuration
- Step 6: Application Load Balancer setup
- Step 7: Auto Scaling Group creation

## **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

## **5. contact info**

## 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
<b>Amazon Virtual Private Cloud (VPC)</b> <b>Status:</b> - <b>Description:</b> vpc network <b>Config summary:</b> Number of NAT Gateways (1)	No group applied	US East (N. Virginia)	0.00 USD	33.75 USD
<b>Elastic Load Balancing</b> <b>Status:</b> - <b>Description:</b> <b>Config summary:</b> Number of Application Load Balancers (1)	No group applied	US East (N. Virginia)	0.00 USD	28.11 USD
<b>Amazon EC2</b> <b>Status:</b> - <b>Description:</b> <b>Config summary:</b> 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

**Your VPCs (2)** [Info](#)

Last updated 1 minute ago [Refresh](#) [Actions](#) [Create VPC](#)

<input type="checkbox"/>	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP opt
<input type="checkbox"/>	lab_vpc-vpc	<a href="#">vpc-Od269eeb60a30aeff</a>	Available	10.0.0.0/16	-	<a href="#">dopt-046...</a>

**VPC** [Show details](#)  
Your AWS virtual network

lab\_vpc-vpc

**Subnets (4)**  
Subnets within this VPC

**us-east-1a**

- lab\_vpc-subnet-public1-us-east-1a
- lab\_vpc-subnet-private1-us-east-1a

**us-east-1b**

- lab\_vpc-subnet-public2-us-east-1b
- lab\_vpc-subnet-private2-us-east-1b

**Route tables (4)**  
Route network traffic to resources

- rtb-0840bb8c7f0284980
- lab\_vpc-rtb-private2-us-east-1b
- lab\_vpc-rtb-public
- lab\_vpc-rtb-private1-us-east-1a

**Network connections**  
Connections to other resources

- lab\_vpc-igw
- lab\_vpc-nat-public

- Step 2: Create SG

**Security Groups (6)** [Info](#)

[Refresh](#) [Actions](#) [Export security groups to CSV](#) [Create security group](#)

Security group name	VPC ID	Description	Owner	Instance count
lab_security	<a href="#">vpc-Od269eeb60a30aeff</a>	ec2 allow traffic	616988447740	3
ALB -lab	<a href="#">vpc-Od269eeb60a30aeff</a>	dd	616988447740	1
aws-cloud9-lab-cloud9-b0e68a45891...	<a href="#">vpc-Od269eeb60a30aeff</a>	Security group for AWS Cloud9 enviro...	616988447740	2
default	<a href="#">vpc-Od269eeb60a30aeff</a>	default VPC security group	616988447740	1
lab_securityDB	<a href="#">vpc-Od269eeb60a30aeff</a>	DB allow traffic	616988447740	1
default	<a href="#">vpc-060d724c24b3422a9</a>	default VPC security group	616988447740	1

#### Step 3: EC2 Instance Deployment

**Instances (4)** [Info](#)

Last updated less than a minute ago [Refresh](#) [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

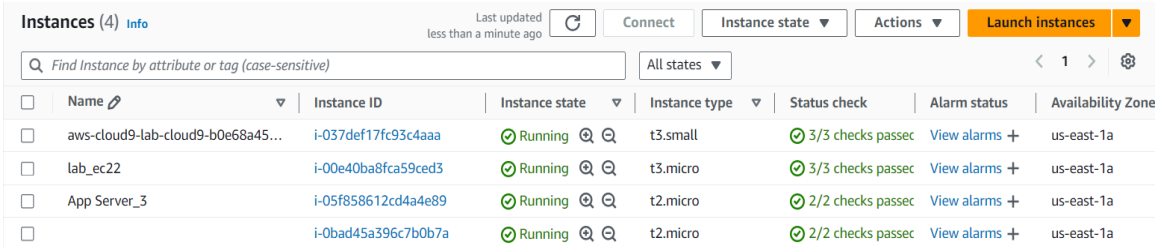
[All states](#)

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	aws-cloud9-lab-cloud9-b0e68a45...	<a href="#">i-037def17fc93c4aaa</a>	Running	t3.small	3/3 checks passed	<a href="#">View alarms</a>	us-east-1a
<input type="checkbox"/>	lab_ec22	<a href="#">i-00e40ba8fca59ced3</a>	Running	t3.micro	3/3 checks passed	<a href="#">View alarms</a>	us-east-1a
<input type="checkbox"/>	App Server_3	<a href="#">i-05f858612cd4a4e89</a>	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a>	us-east-1a
<input type="checkbox"/>		<a href="#">i-0bad45a396c7b0b7a</a>	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a>	us-east-1a



- **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.



Instances (4) <a href="#">Info</a>							
Find Instance by attribute or tag (case-sensitive)				All states		< 1 > ⚙	
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	aws-cloud9-lab-cloud9-b0e68a45...	i-037def17fc93c4aaa	Running	t3.small	3/3 checks passed	<a href="#">View alarms</a> +	us-east-1a
<input type="checkbox"/>	lab_ec22	i-00e40ba8fca59ced3	Running	t3.micro	3/3 checks passed	<a href="#">View alarms</a> +	us-east-1a
<input type="checkbox"/>	App Server_3	i-05f858612cd4a4e89	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a> +	us-east-1a
<input type="checkbox"/>		i-0bad45a396c7b0b7a	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a> +	us-east-1a

- **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.

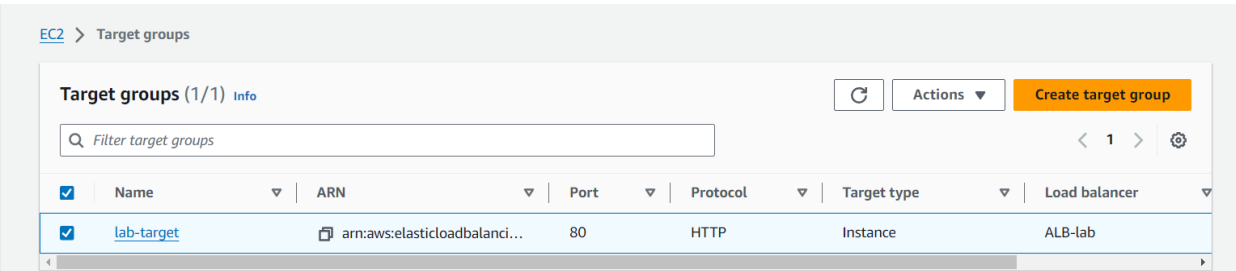
- **Step 6: Application Load Balancer Setup**

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



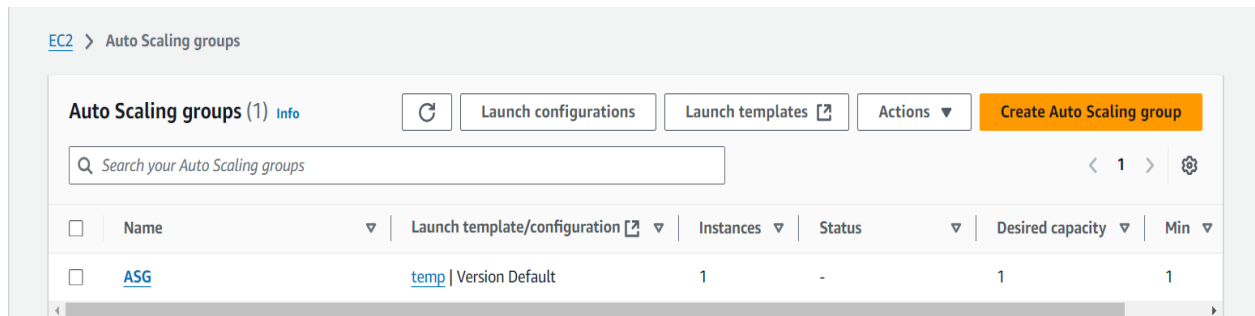
ALB-lab			
Details			
Load balancer type Application	Status Active	VPC vpc-Od269eeb60a30aeff	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone Z355XDOTRQ7X7K	Availability Zones subnet-0f22a1b278de3be04 us-east-1a (use1-az1) subnet-09485f7ec9f6869bd us-east-1b (use1-az2)	Date created October 21, 2024, 22:48 (UTC+03:00)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:616988447740:loadbalancer/app/ALB-lab/0a4dbb6791df7839		DNS name ALB-lab-307958598.us-east-1.elb.amazonaws.com (A Record)	

- **Step 7: Create Target Group**



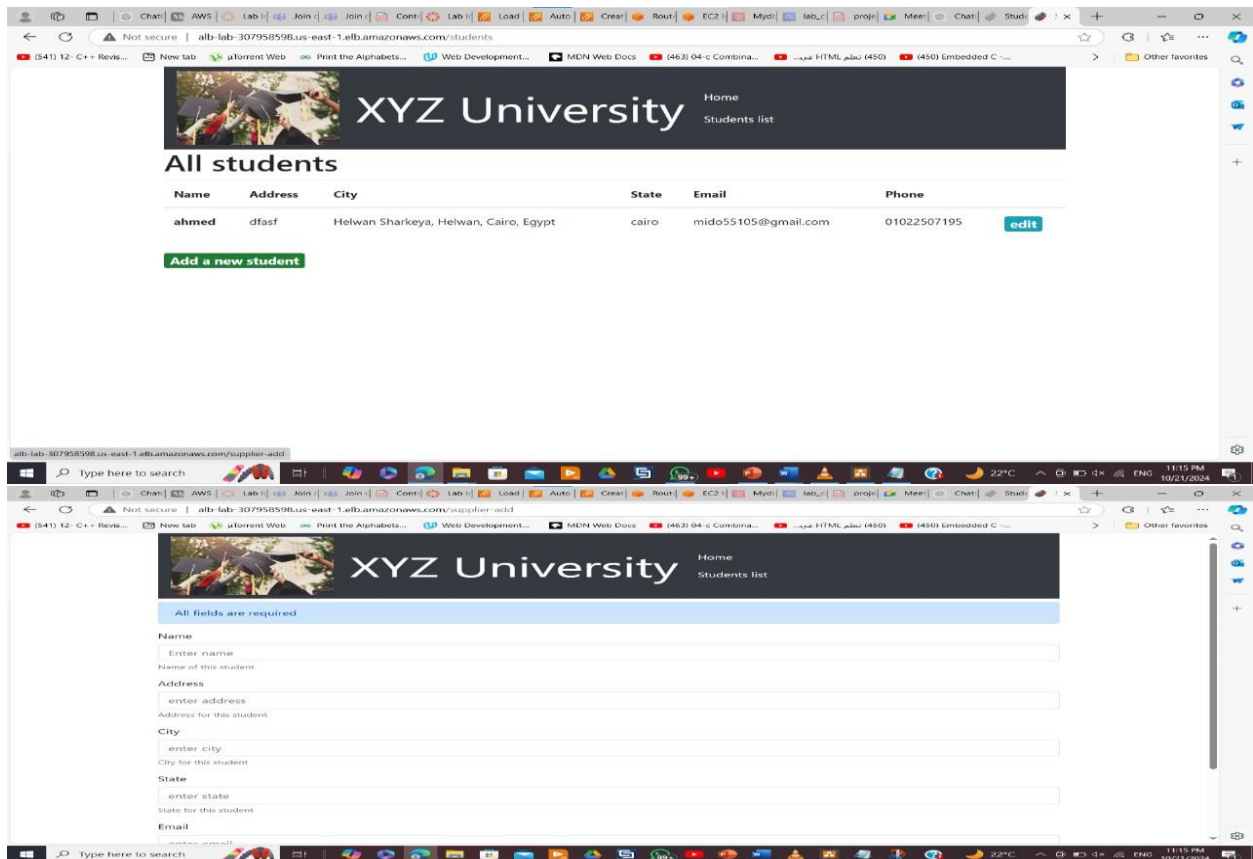
Target groups (1/1) <a href="#">Info</a>						
Filter target groups				< 1 > ⚙		
<input checked="" type="checkbox"/>	Name	ARN	Port	Protocol	Target type	Load balancer
<input checked="" type="checkbox"/>	lab-target	arn:aws:elasticloadbalanci...	80	HTTP	Instance	ALB-lab

- **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.



## **5.Contact info:**

**Email :** [mido55105@gmail.com](mailto:mido55105@gmail.com)

**Phone number:** 01150136230

**Linkedin :** [https://www.linkedin.com/in/ahmed-hossam-256aa0226?utm\\_source=share&utm\\_campaign=share\\_via&utm\\_content=profile&utm\\_medium=android\\_app](https://www.linkedin.com/in/ahmed-hossam-256aa0226?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=android_app)