Building a Highly Available, Scalable Web Application Project

Name: George Adel Mousa

Group: CAI1_SWD8_M2d

Cell Phone: 01227951345

Email: georgeadelmousa@gmail.com

1. Overview:

This project involves creating a proof-of-concept (POC) to host a web application in AWS that is highly available, scalable, load-balanced, and secure. The scenario simulates a university admissions system, which experiences performance issues during peak times. The project follows the AWS Well-Architected Framework and utilizes several AWS services including EC2, RDS, VPC, Auto Scaling, and Load

2. Requirements:

- Create an architectural diagram to show the AWS services
- Estimate the cost by using the AWS Pricing Calculator.
- Deploy a functional web application on a virtual machine and is backed by a relational database.
- Architect a web application to separate layers of the application, such as the web server and database.
- Create a virtual network that is configured appropriately to host a web application that is publicly accessible and secure.
- Deploy a web application with the load distributed across multiple web servers.
- Configure the network security settings for the web servers and database.
- Implement high availability and scalability in the deployed solution.
- Configure access permissions between AWS services

3. Solution of the requirements:

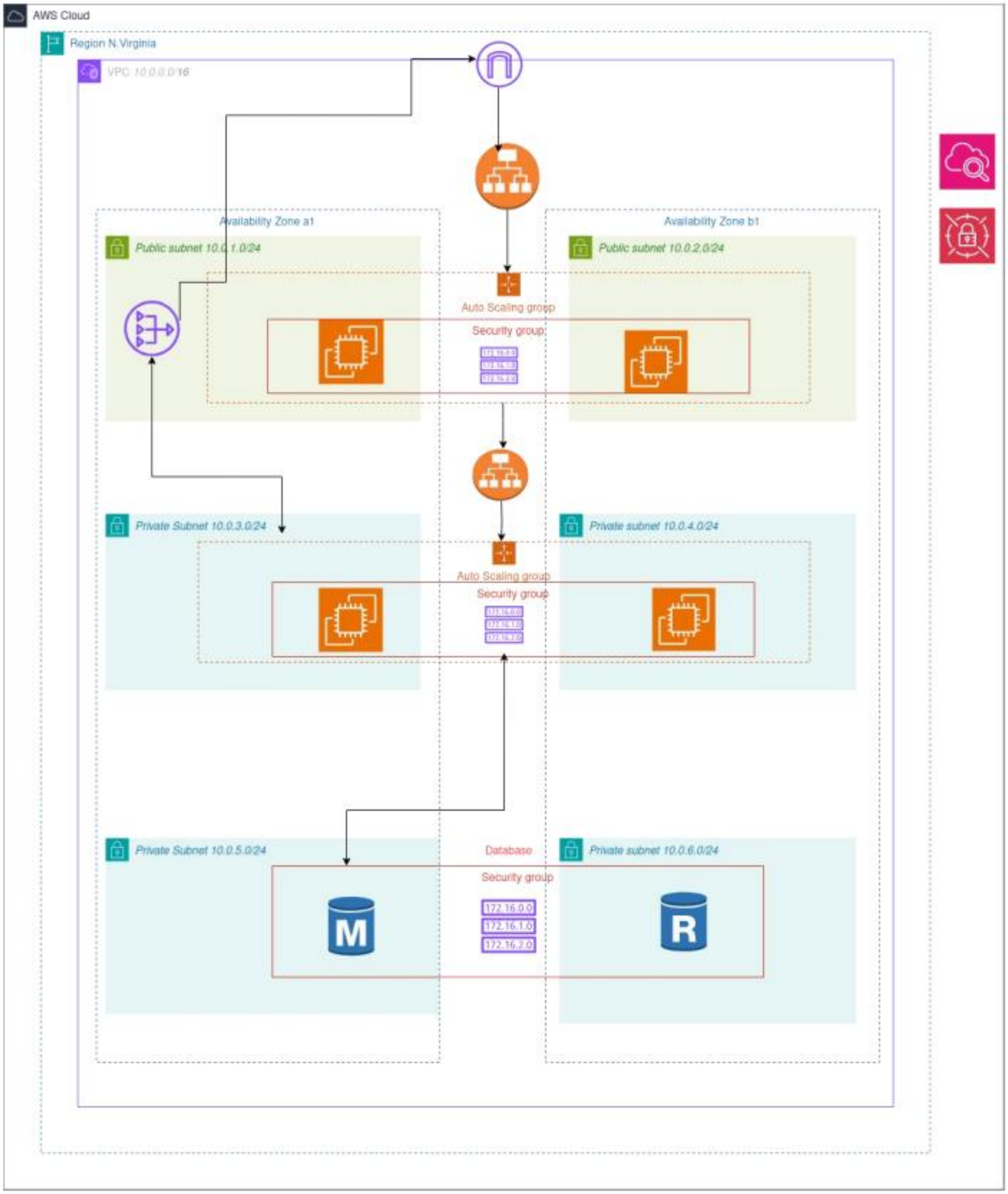
- **Functional:** Meets the functional requirements, such as the ability to view, add, delete, or modify the student records, without any delay.
- **Load balanced:** Balance traffic to avoid overloaded or underutilized resources.
- **Scalable:** Scale and meet the demands that are placed on the application.
- **Highly available:** Have limited downtime when a server becomes unavailable.
- **Secure:** The database is secured and can’t be accessed from public networks.
- **Cost-optimized:** Is designed to keep costs low.
- **High performing:** The routine operations (viewing, adding, deleting, or modifying records) are performed without a delay under normal, variable, and peak loads.

4. Project:

Phase 1: Planning and Cost Estimation**:**

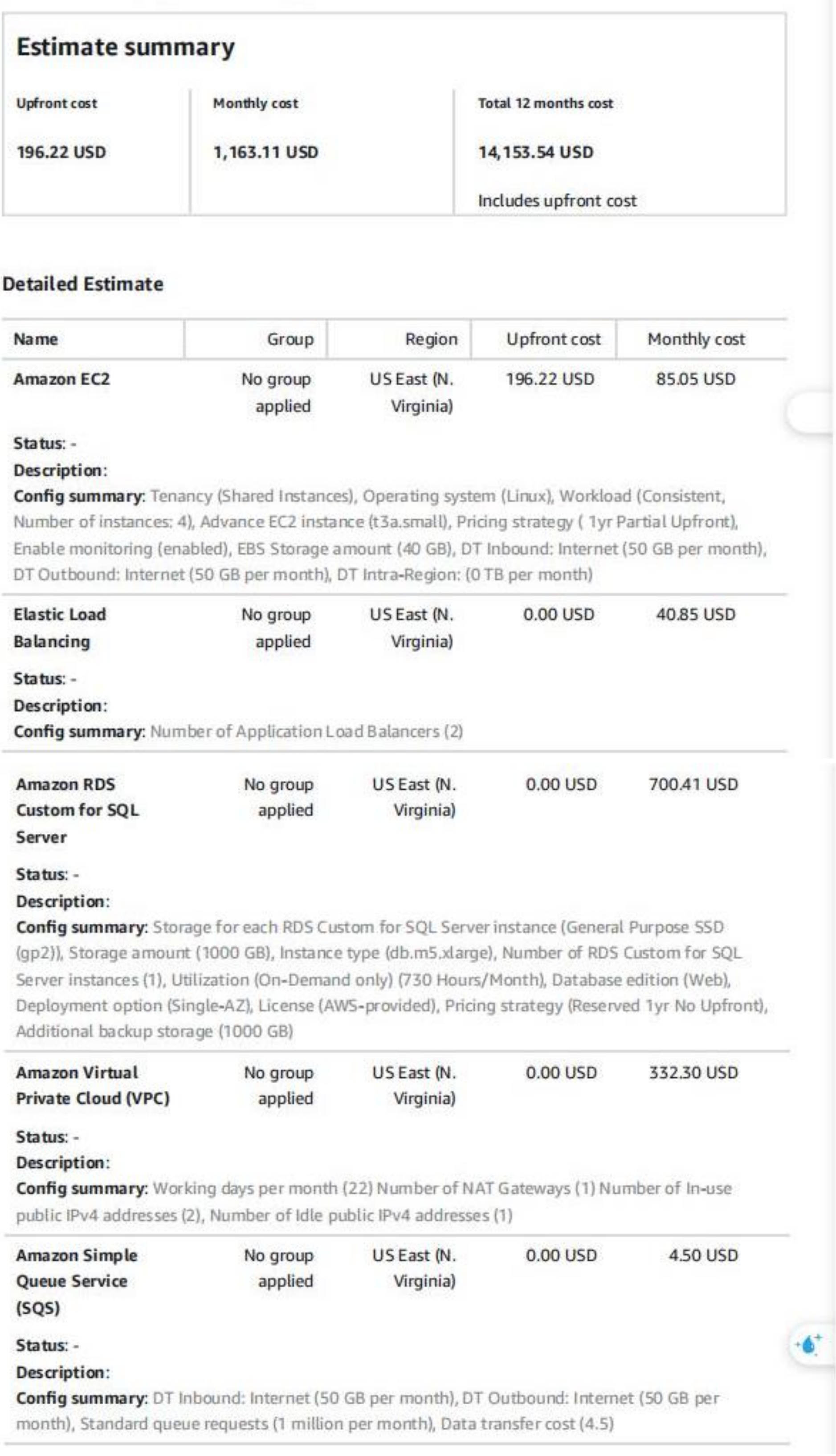
1- Architectural Design:

- Create an architecture diagram using AWS icons or tools.
- The architecture must include EC2 for the web server, Amazon RDS for the database, Load Balancers, Auto Scaling, and VPC with subnets.



2- Estimate Costs:

A cost projection for operating the architecture on AWS for a 12-month period is generated using the AWS Pricing Calculator to ensure accurate budgeting



Estimate summary

Upfront cost	Monthly cost	Total 12 months cost
196.22 USD	1,163.11 USD	14,153.54 USD Includes upfront cost

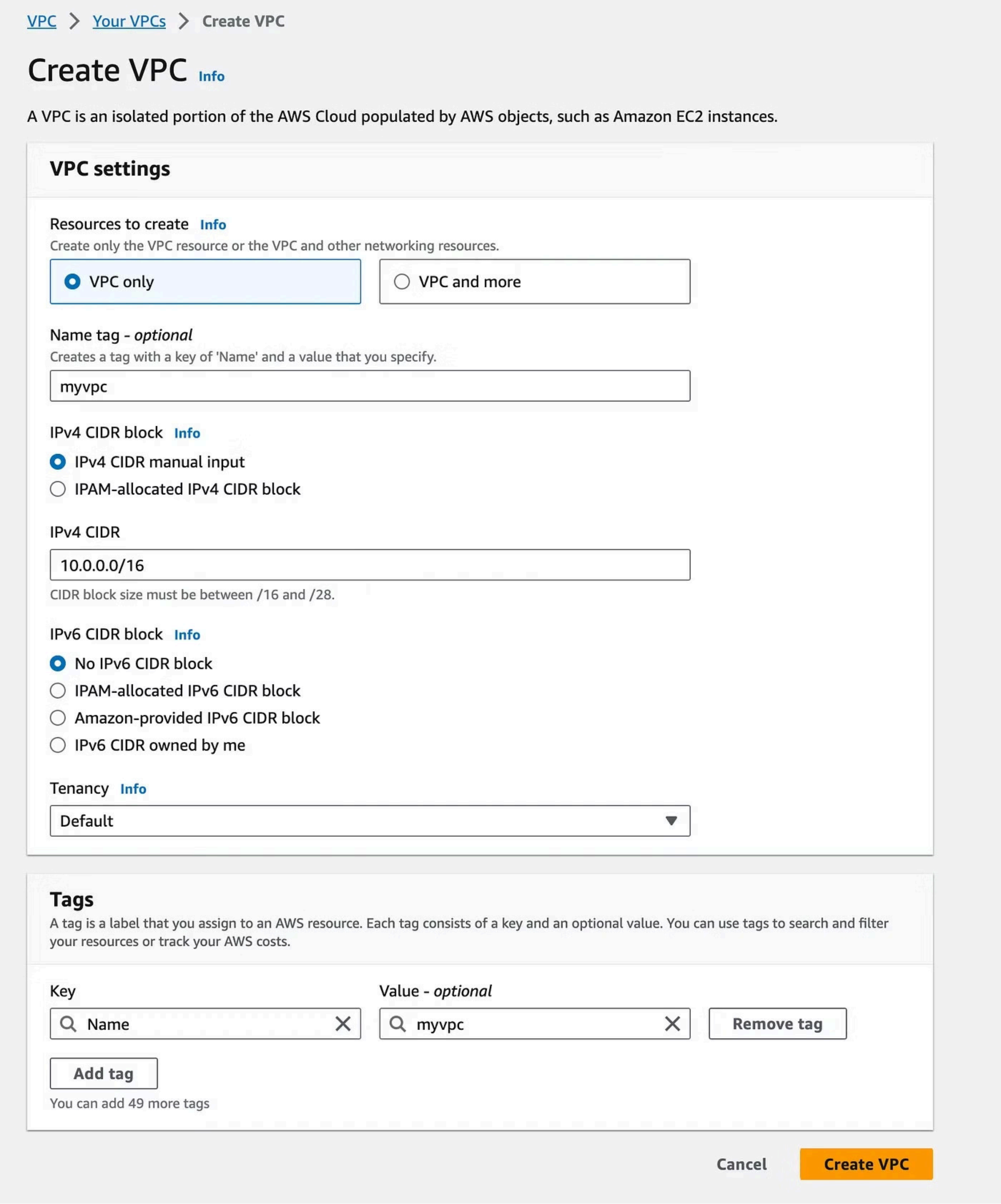
Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Amazon EC2	No group applied	US East (N. Virginia)	196.22 USD	85.05 USD
Status: - **Description:** **Config summary:** Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 4), Advance EC2 instance (t3a.small), Pricing strategy (1yr Partial Upfront), Enable monitoring (enabled), EBS Storage amount (40 GB), DT Inbound: Internet (50 GB per month), DT Outbound: Internet (50 GB per month), DT Intra-Region: (0 TB per month)				
Elastic Load Balancing	No group applied	US East (N. Virginia)	0.00 USD	40.85 USD
Status: - **Description:** **Config summary:** Number of Application Load Balancers (2)				
Amazon RDS Custom for SQL Server	No group applied	US East (N. Virginia)	0.00 USD	700.41 USD
Status: - **Description:** **Config summary:** Storage for each RDS Custom for SQL Server instance (General Purpose SSD (gp2)), Storage amount (1000 GB), Instance type (db.m5.xlarge), Number of RDS Custom for SQL Server instances (1), Utilization (On-Demand only) (730 Hours/Month), Database edition (Web), Deployment option (Single-AZ), License (AWS-provided), Pricing strategy (Reserved 1yr No Upfront), Additional backup storage (1000 GB)				
Amazon Virtual Private Cloud (VPC)	No group applied	US East (N. Virginia)	0.00 USD	332.30 USD
Status: - **Description:** **Config summary:** Working days per month (22) Number of NAT Gateways (1) Number of In-use public IPv4 addresses (2), Number of Idle public IPv4 addresses (1)				
Amazon Simple Queue Service (SQS)	No group applied	US East (N. Virginia)	0.00 USD	4.50 USD
Status: - **Description:** **Config summary:** DT Inbound: Internet (50 GB per month), DT Outbound: Internet (50 GB per month), Standard queue requests (1 million per month), Data transfer cost (4.5)				

Phase 2: Basic Functional Web Application Deployment:

1- Creating a Virtual Network (VPC):

- Set up a Virtual Private Cloud (VPC) with public and private subnets.
- Configure security groups to allow traffic only on required ports (HTTP/HTTPS for webserver, MySQL for RDS).



VPC > Your VPCs > Create VPC

Create VPC Info

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.

VPC settings

Resources to create Info

Create only the VPC resource or the VPC and other networking resources.

● VPC only ○ VPC and more

Name tag - *optional*

Creates a tag with a key of 'Name' and a value that you specify.

myvpc

IPv4 CIDR block Info

● IPv4 CIDR manual input

○ IPAM-allocated IPv4 CIDR block

IPv4 CIDR

10.0.0.0/16

CIDR block size must be between /16 and /28.

IPv6 CIDR block Info

● No IPv6 CIDR block

○ IPAM-allocated IPv6 CIDR block

○ Amazon-provided IPv6 CIDR block

○ IPv6 CIDR owned by me

Tenancy Info

Default ▼

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

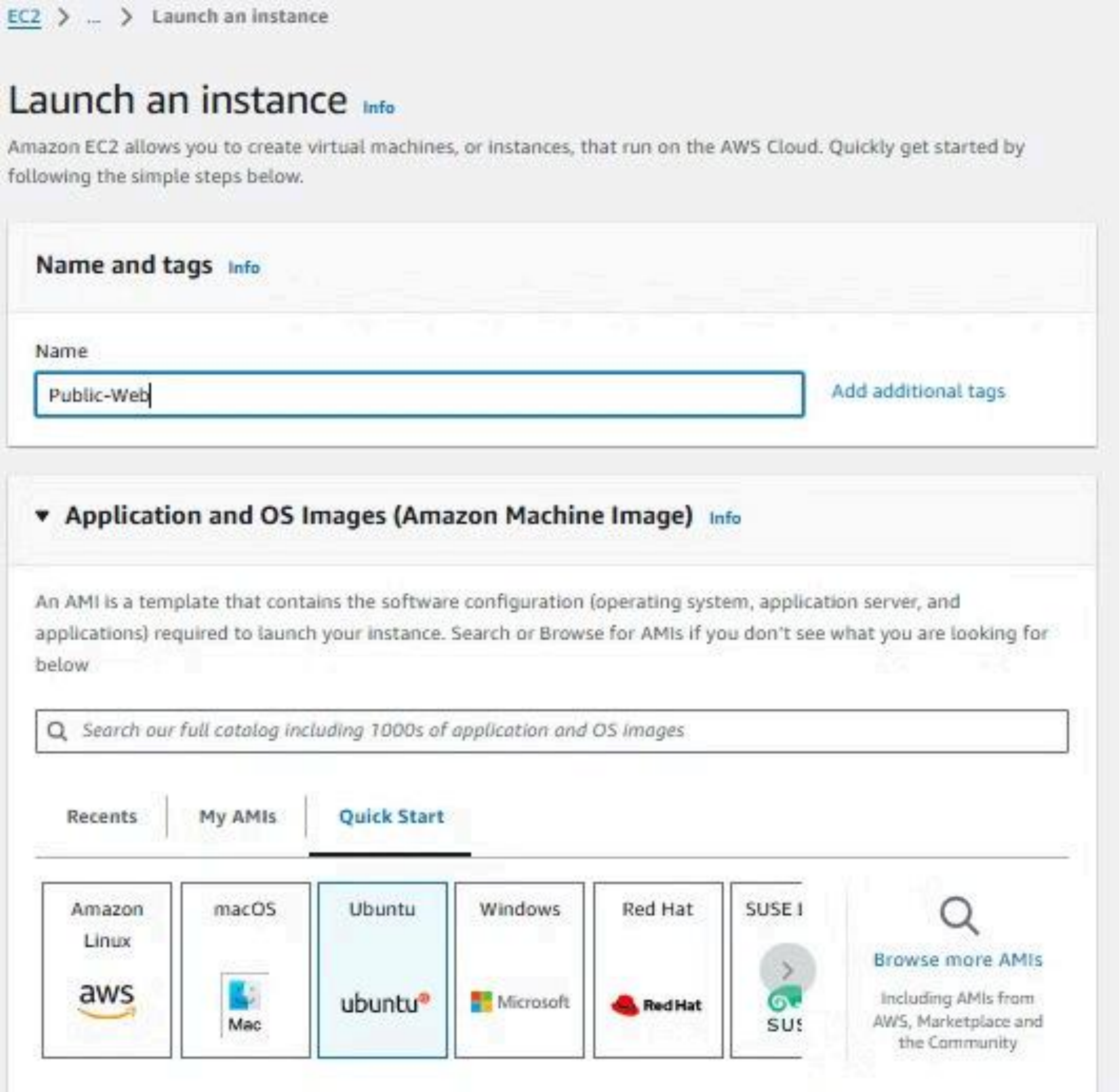
Key	Value - *optional*	
Name	myvpc	Remove tag

Add tag

You can add 49 more tags

2- Deploying a Virtual Machine (EC2)

Using Amazon EC2, we launch a virtual machine to host the web application. The instance runs the latest Ubuntu AMI, and the provided JavaScript code is deployed to serve as the frontend and backend of the student records application.



Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: Public-Web

▼ Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images
Recents
My AMIs
Quick Start
Amazon Linux
aws
macOS
Mac
Ubuntu
ubuntu
Windows
Microsoft
Red Hat
Red Hat
SUSE I
SUS
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

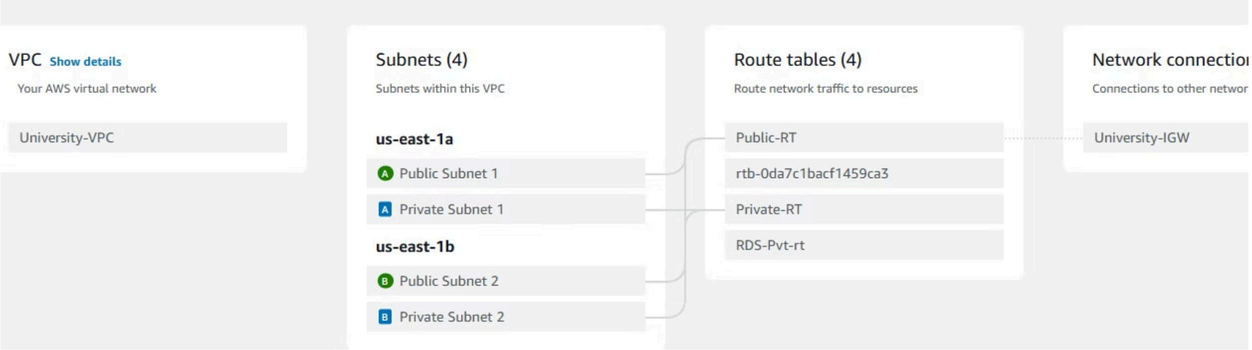
3- Testing Deployment:

Once the instance is running, the web application is accessible via its public IPv4 address. Basic operations such as viewing, adding, deleting, and modifying student records are tested to ensure functionality.

Phase 3: Decoupling Application Components:

1- Reconfiguring the VPC:

- Modify the VPC to include private subnets for hosting the database.
- Set up routing and security to ensure the database is not publicly accessible.



VPC Show details
Your AWS virtual network
University-VPC

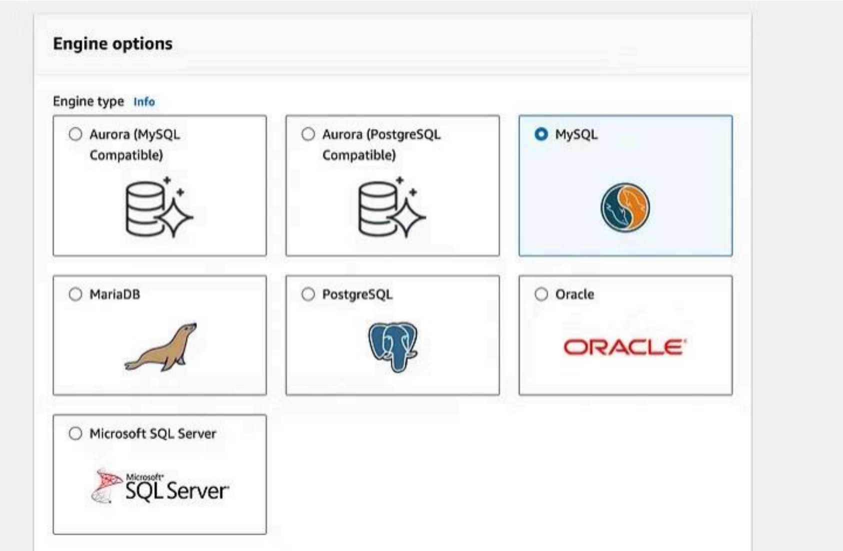
Subnets (4)
Subnets within this VPC
us-east-1a
Public Subnet 1
Private Subnet 1
us-east-1b
Public Subnet 2
Private Subnet 2

Route tables (4)
Route network traffic to resources
Public-RT
rtb-0da7c1bacf1459ca3
Private-RT
RDS-Pvt-rt

Network connectio
Connections to other networ
University-IGW

2- Creating the Amazon RDS Database:

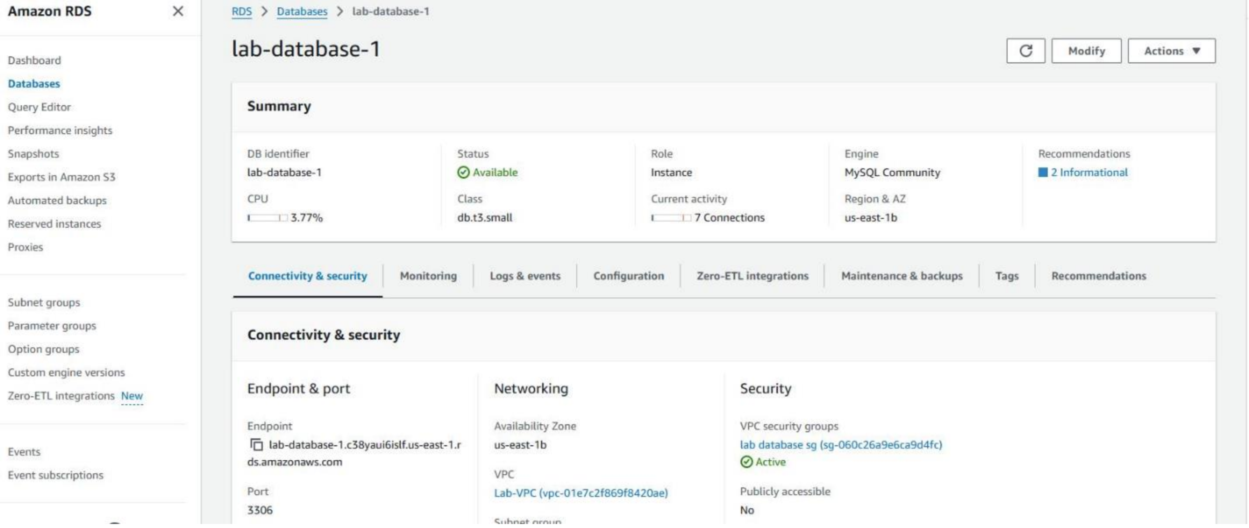
The first step is to create a security group that will be assosiated to the RDS to open the ports needed for connection to the EC2 web server.



Engine options

Engine type Info

○ Aurora (MySQL Compatible)
○ Aurora (PostgreSQL Compatible)
● MySQL
○ MariaDB
○ PostgreSQL
○ Oracle
○ Microsoft SQL Server



Amazon RDS

Dashboard
Databases
Query Editor
Performance insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies

Subnet groups
Parameter groups
Option groups
Custom engine versions
Zero-ETL integrations New

Events
Event subscriptions

RDS > Databases > lab-database-1

lab-database-1

Modify | Actions ▼

Summary

DB identifier: lab-database-1
Status: Available
Role: Instance
Engine: MySQL Community
Recommendations: 2 Informational
CPU: 3.77%
Class: db.t3.small
Current activity: 7 Connections
Region & AZ: us-east-1b

Connectivity & security | Monitoring | Logs & events | Configuration | Zero-ETL integrations | Maintenance & backups | Tags | Recommendations

Connectivity & security

Endpoint & port
Endpoint: lab-database-1.c38yaui6islf.us-east-1.rds.amazonaws.com
Port: 3306

Networking
Availability Zone: us-east-1b
VPC: Lab-VPC (vpc-01e7c2f869f8420ae)
Subnet group

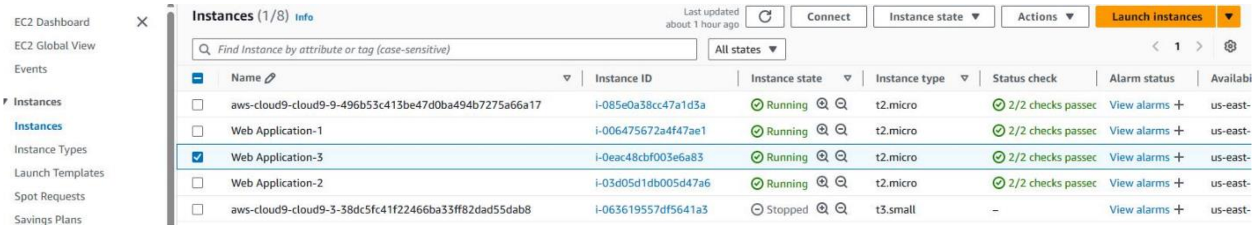
Security
VPC security groups: lab database sg (sg-060c26a9e6ca9d4fc)
Active
Publicly accessible: No

3- Configuring AWS Cloud9 and Secrets Manager:

Cloud9 is an IDE provided by AWS. It allows developers to write, run, and debug their code in the cloud. You can access your development environment from anywhere using just a web browser.

4- Deploying a New Web Server Instance:

- Launch a new EC2 instance for the server and configure it to use RDS database.
- Ensure the instance is configured with the IAM roles and security settings.



☐	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availabi
☐	aws-cloud9-cloud9-9-496b53c413be47d0ba494b7275a66a17	i-085e0a38cc47a1d3a	Running	t2.micro	2/2 checks passec	View alarms +	us-east-
☐	Web Application-1	i-006475672a4f47ae1	Running	t2.micro	2/2 checks passec	View alarms +	us-east-
☑	Web Application-3	i-0eac48cbf003e6a83	Running	t2.micro	2/2 checks passec	View alarms +	us-east-
☐	Web Application-2	i-03d05d1db005d47a6	Running	t2.micro	2/2 checks passec	View alarms +	us-east-
☐	aws-cloud9-cloud9-3-38dc5fc41f22466ba33ff82dad55dab8	i-063619557df5641a3	Stopped	t3.small	–	View alarms +	us-east-

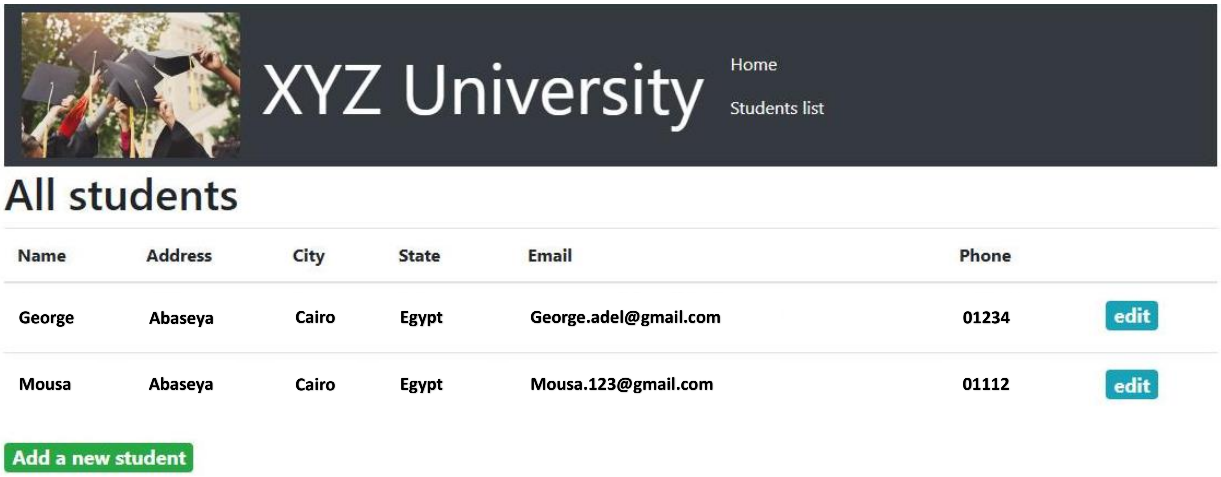
5- Migrating the Database:

The student records data is migrated from the EC2 instance to the new RDS database using AWS CLI commands and a provided migration script.

```
aws secretsmanager create-secret \
--name Mydbsecret \
--description "Database secret for web app" \
--secret-string "{\"user\":\"\",\"password\":\"\",\"host\":\"\",\"db\":\"\"}
mysqldump -h -u nodeapp -p --databases STUDENTS > data.sql
mysql -h -u nodeapp -p STUDENTS < data.sql
```

6- Testing the Application:

- Perform tests on the application to confirm functionality (view, add, delete, and modify records).
- Conduct a load test using AWS Cloud9 to evaluate how the application handles increased traffic.

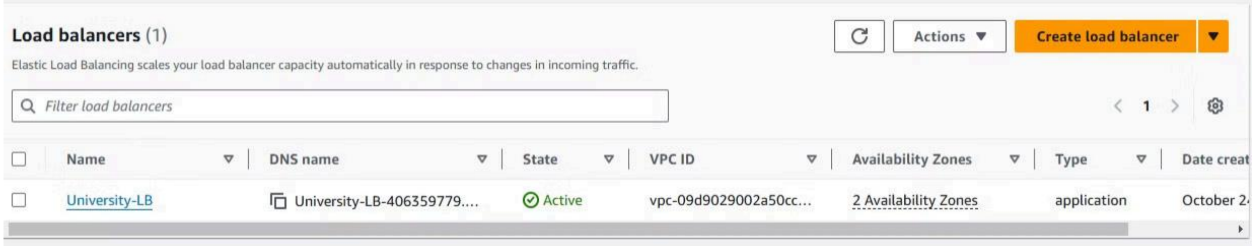


Name	Address	City	State	Email	Phone
George	Abaseya	Cairo	Egypt	George.adel@gmail.com	01234
Mousa	Abaseya	Cairo	Egypt	Mousa.123@gmail.com	01112

Phase 4: Implementing High Availability and Scalability:

1- Creating an Application Load Balancer (ALB):

- Set up an ALB to distribute traffic across multiple EC2 instances.
- Ensure at least two Availability Zones are configured to improve redundancy



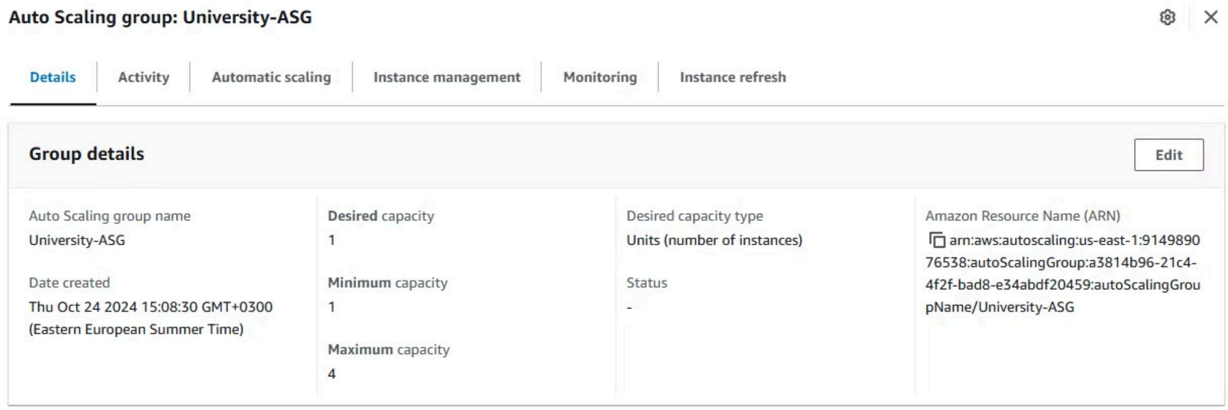
Load balancers (1)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

□	Name	DNS name	State	VPC ID	Availability Zones	Type	Date creat
□	University-LB	University-LB-406359779....	✓ Active	vpc-09d9029002a50cc...	2 Availability Zones	application	October 2

2- Auto Scaling Group:

To enable automatic scaling, an Auto Scaling group is set up with a launch template based on the web server configuration. The Auto Scaling group dynamically adjusts the number of instances in response to CPU utilization.



Auto Scaling group: University-ASG

Group details

Auto Scaling group name
University-ASG

Date created
Thu Oct 24 2024 15:08:30 GMT+0300
(Eastern European Summer Time)

Desired capacity
1

Minimum capacity
1

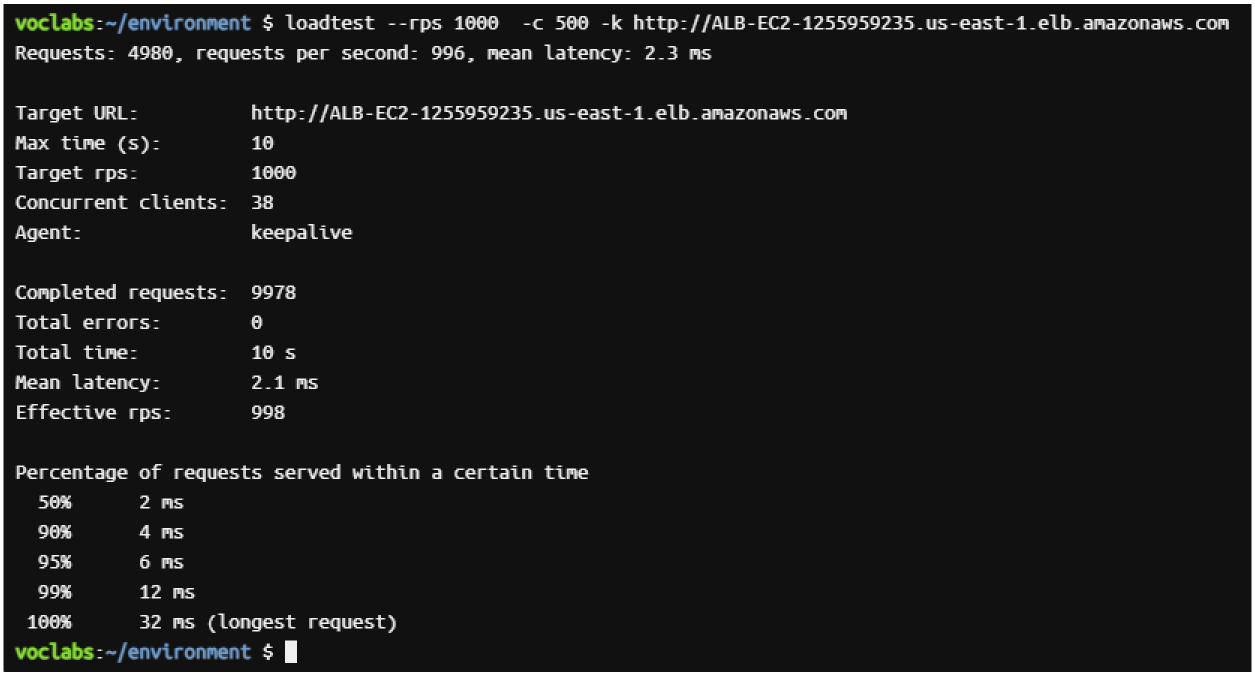
Maximum capacity
4

Desired capacity type
Units (number of instances)

Status
-

Amazon Resource Name (ARN)
arn:aws:autoscaling:us-east-1:914989076538:autoScalingGroup:a3814b96-21c4-4f2f-bad8-e34abdf20459:autoScalingGroupName/University-ASG

3- Testing and Load Balancing:



```
voclabs:~/environment $ loadtest --rps 1000  -c 500 -k http://ALB-EC2-1255959235.us-east-1.elb.amazonaws.com
Requests: 4980, requests per second: 996, mean latency: 2.3 ms

Target URL:          http://ALB-EC2-1255959235.us-east-1.elb.amazonaws.com
Max time (s):        10
Target rps:          1000
Concurrent clients:  38
Agent:               keepalive

Completed requests:  9978
Total errors:        0
Total time:          10 s
Mean latency:        2.1 ms
Effective rps:       998

Percentage of requests served within a certain time
  50%      2 ms
  90%      4 ms
  95%      6 ms
  99%      12 ms
 100%      32 ms (longest request)
voclabs:~/environment $ █
```