AWS ACADEMY-BUILDING A HIGHLY AVAILABLE, SCALABLE WEB APPLICATION

DEPI-Project

BY: OMAR AYMAN MOHAMED

OCTOBER 24, 2024

Acknowledgement

I would like to express my heartfelt gratitude to **Dr. Ahmed Elhamy** for his invaluable guidance and support throughout my learning journey. Your insights and encouragement have played a crucial role in my understanding of AWS architecture and cloud technologies. I appreciate the time and effort you dedicated to helping us grasp complex concepts, which has significantly enhanced my skills and confidence in this field. Thank you for being an inspiring educator and for your unwavering commitment to our success.

Table Of Contents

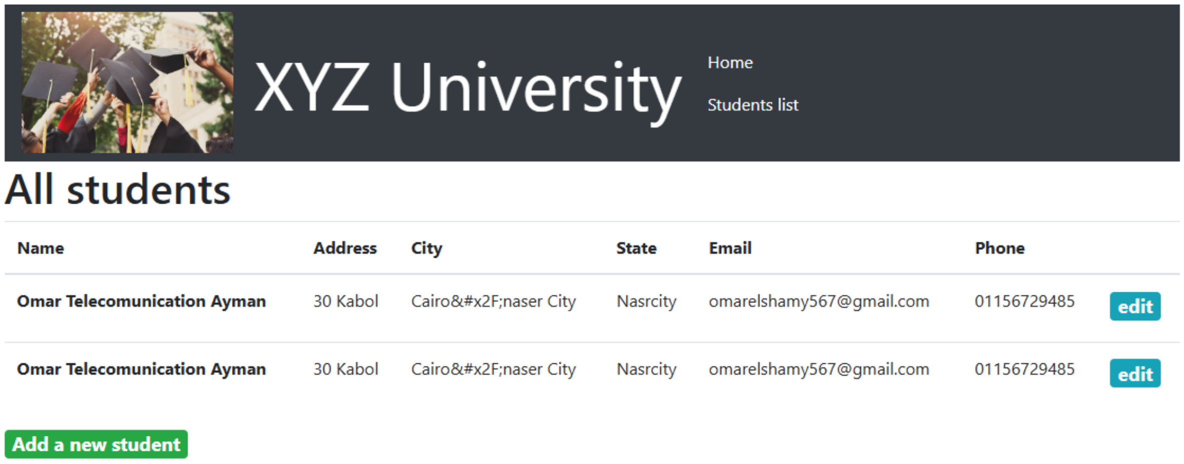
Contents

Introduction:

This project requires an understanding of core AWS services, such as compute, storage, networking, and database services. The project also requires knowledge of architectural best practices, such as high availability, scalability, and security. Students should have completed the AWS Academy Cloud Architecting course to gain this necessary knowledge. Students who have completed the AWS Academy Cloud Foundations course and are enrolled in the AWS Academy Cloud Architecting course can also try to complete this project with the help of course materials, labs from courses, and educator guidance. Knowledge of any programming language, such as Python or JavaScript, is an advantage but isn't mandatory.

The Scenario is to plan, design, build, and deploy the web application to the AWS Cloud in a way that is consistent with best practices of the AWS Well-Architected Framework. During the peak admissions period, the application must support thousands of users, and be highly available, scalable, load-balanced, secure, and high performing.

Create an architectural diagram to depict various AWS services and their interactions with each other. Estimate the cost of using services by using the AWS Pricing Calculator. Deploy a functional web application that runs on a single 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 appropriate network security settings for the web servers and database. Implement high availability and scalability in the deployed solution. Configure access permissions between AWS services.



All students

Name	Address	City	State	Email	Phone	
Omar Telecomunication Ayman	30 Kabol	Cairo&#x2F;naser City	Nasrcity	omarelshamy567@gmail.com	01156729485	edit
Omar Telecomunication Ayman	30 Kabol	Cairo&#x2F;naser City	Nasrcity	omarelshamy567@gmail.com	01156729485	edit

Planning the design and estimating the cost:

Planning design You can use several tools to create architecture design one of the recommended tools is a lucid chart, you would like to download an AWS Official architecture icon, and you would like to find references architecture Diagrams. To find the estimated cost in AWS using the AWS Pricing Calculator.

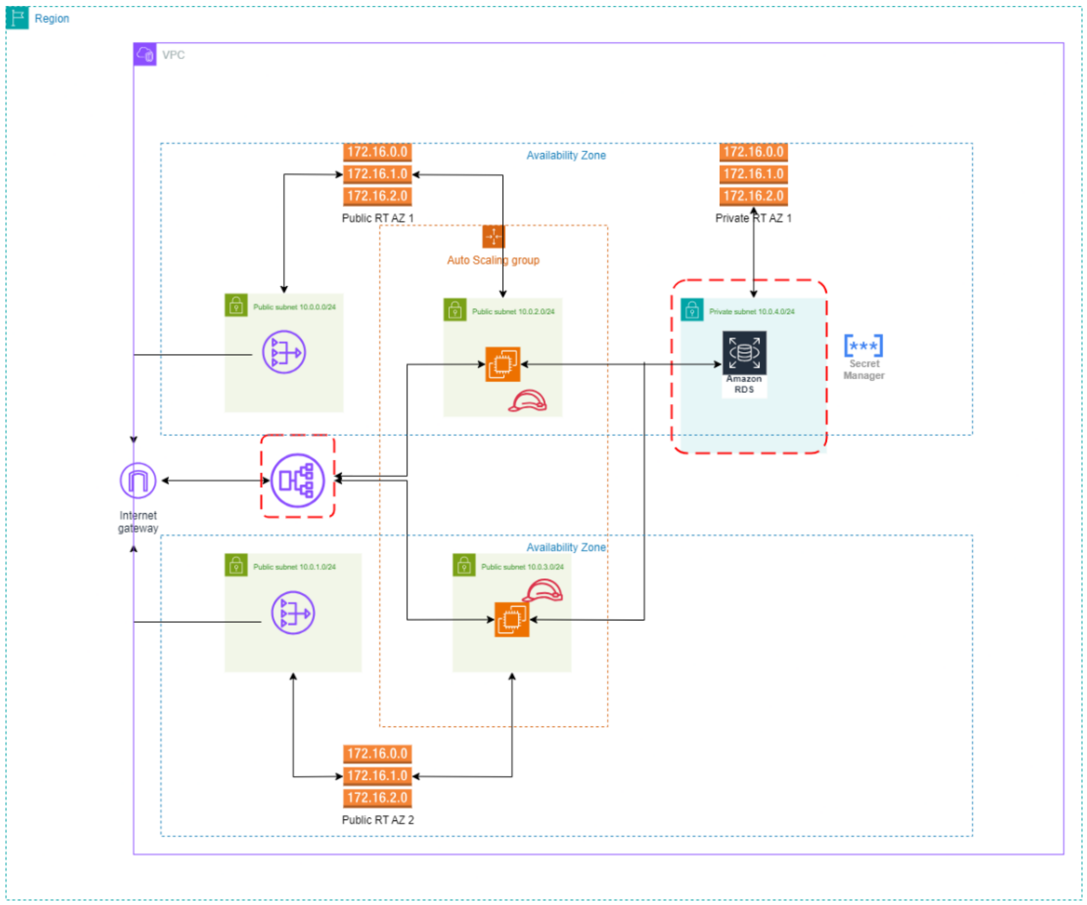
- **Functional:** The solution meets the functional requirements, such as the ability to view, add, delete, or modify the student records, without any perceivable delay.
- **Load balanced:** The solution can properly balance user traffic to avoid overloaded or underutilized resources.
- **Scalable:** The solution is designed to scale to meet the demands that are placed on the application.
- **Highly available:** The solution is designed to have limited downtime when a web server becomes unavailable.
- **Secure:** The database is secured and can't be accessed directly from public networks. The web servers and database can be accessed only over the appropriate ports. The web application is accessible over the internet. The database credentials aren't hardcoded into the web application.
- **Cost-optimized:** The solution is designed to keep costs low.
- **High performing:** The routine operations (viewing, adding, deleting, or modifying records) are performed without a perceivable delay under normal, variable, and peak loads.

Assumptions

This project will be built in a controlled lab environment that has restrictions on services, features, and budget. Consider the following assumptions for the project:

- The application is deployed in one AWS Region.
- The website **does not need to be** available over HTTPS or a custom domain.
- The solution is deployed on *Ubuntu* machines by using the JavaScript code that is provided.
- Use the JavaScript code as written unless the instructions specifically direct you to change the code.
- The solution uses services and features within the restrictions of the lab environment.
- The ***database*** is hosted only in a **single Availability Zone.**
- The **website is publicly accessible without authentication.**

AWS Architecture Diagram:



Components:

1. <u>AWS Region:</u>

- A Region is a separate geographical area with its own set of AWS data centers. Each Region operates independently, with complete isolation from other Regions, allowing you to replicate applications across multiple regions for redundancy and disaster recovery.
- Key Features:
  - ❖ Each region contains at least two Availability Zones.
  - ❖ Data transfer between regions incurs additional costs.
  - ❖ You choose regions based on factors like legal requirements, latency to users, or redundancy.

2. <u>Virtual Private Cloud (VPC):</u>

- VPC is your isolated network within AWS. You can create subnets, route tables, gateways, and security configurations. The VPC allows for control over the IP address range, subnet creation, internet access, and inter-subnet traffic.

- Key Features:
    - ❖ CIDR Block: You can define a range of private IP addresses using CIDR notation (e.g., 10.0.0.0/16).
    - ❖ Subnet Types: Public, private, and isolated subnets for controlling which resources can access the internet.
    - ❖ Network ACLs: Additional layer of security controlling inbound/outbound traffic at the subnet level.

3. <u>Internet Gateway:</u>

- The IGW is a horizontally scaled, redundant, and highly available VPC component that enables communication between instances in your VPC and the internet.
- **Key Features**:
    - ❖ It provides a target in your VPC route tables for internet-routable traffic.
    - ❖ Supports both IPv4 and IPv6 traffic.
    - ❖ **Attach to VPC**: Must be explicitly attached to the VPC for internet access.

4. <u>NAT Gateway</u>

- NAT Gateway allows instances in a private subnet to connect to the internet but prevents unsolicited inbound connections.
- **Key Features**:
    - ❖ Managed by AWS, ensuring scalability and high availability.
    - ❖ You configure route tables so that private instances send outbound traffic to the NAT Gateway.
    - ❖ It is located in the public subnet and requires an Elastic IP.

5. <u>Security Groups:</u>

- Security groups act as virtual firewalls for your EC2 instances to control incoming and outgoing traffic.
- **Key Features**:
    - ❖ **Stateful**: Any traffic allowed in is automatically allowed out, and vice versa.
    - ❖ **Rules**: You define inbound and outbound rules. Rules specify protocol (TCP, UDP, ICMP), port ranges, and source/destination IP addresses.
    - ❖ Security groups are attached to instances or services (like RDS) and control traffic at the instance level.

6. <u>EC2 Instance:</u>

- Elastic Compute Cloud (EC2) instances are scalable virtual servers that run in the AWS cloud.
- **Key Features**:
  - ❖ **Instance Types**: Choose from various instance types (e.g., t3.micro, m5.large), depending on your application needs (CPU, memory, network throughput).
  - ❖ **AMI**: Amazon Machine Image (AMI) defines the OS and pre-installed software on the instance.
  - ❖ **Elastic IP**: You can assign static IP addresses (Elastic IPs) to EC2 instances for stable internet communication.

7. <u>Application Load Balancer:</u>

- ALB is a Layer 7 load balancer that distributes traffic based on the content of the request (e.g., HTTP headers, request paths).
- **Key Features**:
  - ❖ **Target Groups**: You define target groups (collections of EC2 instances or containers) for routing traffic.
  - ❖ **Listeners**: ALB uses listeners to check for incoming traffic on specific ports (e.g., 80 for HTTP or 443 for HTTPS).
  - ❖ **Sticky Sessions**: ALB supports session persistence to ensure the same client is always routed to the same backend.
  - ❖ **Health Checks**: Monitors target health and removes unhealthy instances from the load-balancing pool.

8. <u>Private Subnet:</u>

- A private subnet is not directly accessible from the internet, typically used for backend resources like databases or application servers.
- **Key Features**:
  - ❖ **Isolation**: Resources in a private subnet cannot be accessed directly from the internet.
  - ❖ **Outbound Access**: To allow outbound access (e.g., for updates), a NAT Gateway is used.
  - ❖ **Routing**: Route tables direct outbound traffic through the NAT Gateway instead of the Internet Gateway.

9. <u>Public Subnet:</u>

- A public subnet has a route to the Internet Gateway, making it accessible from the internet.
- **Key Features**:
  - ❖ **Public IPs**: Instances in a public subnet can have public IP addresses, allowing them to receive traffic from the internet.
  - ❖ **Web Servers**: Typically, web servers are hosted in public subnets to handle HTTP/S requests.

10. Route Table:

- Route tables contain rules that determine the traffic flow between subnets and to/from external networks.
- **Key Features**:
    - ❖ **Routes**: Each route consists of a destination (e.g., 0.0.0.0/0 for the internet) and a target (e.g., Internet Gateway or NAT Gateway).
    - ❖ **Subnet Association**: Each subnet is associated with one route table.
    - ❖ **Propagation**: Dynamic route propagation is possible when using VPNs or Direct Connect.

11. Auto Scaling Group:

- Automatically adjusts the number of EC2 instances to maintain availability and meet demand.
- **Key Features**:
    - ❖ **Scaling Policies**: Configure ASG to add/remove instances based on CloudWatch alarms, CPU usage, or custom metrics.
    - ❖ **Minimum and Maximum Capacity**: Define the number of instances that the group can scale in and out to maintain performance.
    - ❖ **Health Checks**: Monitors the health of EC2 instances and replaces them if they fail health checks.

12. Secrets Manager:

- Secrets Manager securely stores and retrieves secrets (e.g., API keys, database credentials).
- **Key Features**:
    - ❖ **Automatic Rotation**: Supports automatic rotation of secrets on a defined schedule.
    - ❖ **Encryption**: Secrets are encrypted at rest using AWS KMS (Key Management Service).
    - ❖ **Access Policies**: IAM policies control access to secrets, ensuring that only authorized resources can retrieve them.

13. Relational Database Service:

- RDS is a managed service for relational databases like MySQL, PostgreSQL, and SQL Server.
- **Key Features**:
    - ❖ **Automatic Backups**: Provides automatic backups and point-in-time recovery.
    - ❖ **Multi-AZ Deployments**: RDS supports deployment across multiple AZs for high availability.
    - ❖ **Read Replicas**: RDS allows creating read replicas to scale read-heavy applications.

14.Availability Zone:

- AZs are isolated data centers within an AWS region, designed for fault tolerance and high availability.
- **Key Features**:
  - ❖ **Isolation**: Each AZ is isolated from failures in other AZs, providing redundancy.
  - ❖ **Low Latency**: AZs are connected through low-latency, high-throughput networking.

15. IAM Role:

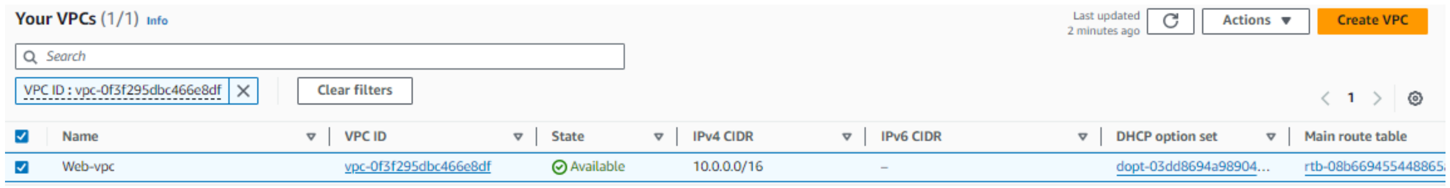
- An IAM Role is a set of permissions that define what AWS resources a service or user can access.
- **Key Features**:
  - ❖ **Temporary Credentials**: IAM roles provide temporary access credentials (using STS) instead of long-term credentials.
  - ❖ **Service Roles**: You can assign IAM roles to AWS services (like EC2 or Lambda) to access other AWS resources securely.

16. Cloud9:

- Cloud9 is an integrated development environment (IDE) in the cloud that supports real-time collaboration, debugging, and code execution.
- **Key Features**:
  - ❖ **Preconfigured Environment**: Comes with all necessary tools (e.g., AWS CLI, SDKs) to interact with AWS services.
  - ❖ **SSH Access**: Allows remote access to your EC2 instances for development and deployment tasks.

Working Steps:

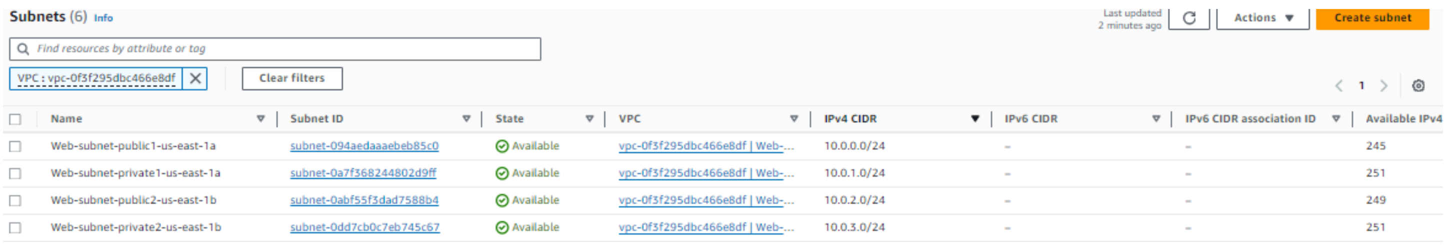
1.Creating a virtual network:



VPC ID : vpc-0f3f295dbc466e8df

☑	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set	Main route table
☑	Web-vpc	vpc-0f3f295dbc466e8df	Available	10.0.0.0/16	–	dopt-03dd8694a98904…	rtb-08b669455448865

Creating Subnets:



Subnets (6) Info

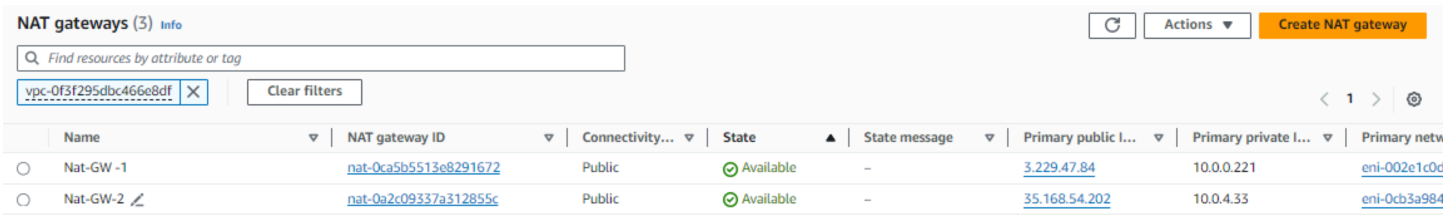
Last updated 2 minutes ago | Actions | Create subnet

Find resources by attribute or tag

VPC : vpc-0f3f295dbc466e8df | Clear filters

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID	Available IPv4
Web-subnet-public1-us-east-1a	subnet-094aedaaaebeb85c0	Available	vpc-0f3f295dbc466e8df | Web-...	10.0.0.0/24	–	–	245
Web-subnet-private1-us-east-1a	subnet-0a7f368244802d9ff	Available	vpc-0f3f295dbc466e8df | Web-...	10.0.1.0/24	–	–	251
Web-subnet-public2-us-east-1b	subnet-0abf55f3dad7588b4	Available	vpc-0f3f295dbc466e8df | Web-...	10.0.2.0/24	–	–	249
Web-subnet-private2-us-east-1b	subnet-0dd7cb0c7eb745c67	Available	vpc-0f3f295dbc466e8df | Web-...	10.0.3.0/24	–	–	251

Creating Nat Getways:



NAT gateways (3) Info

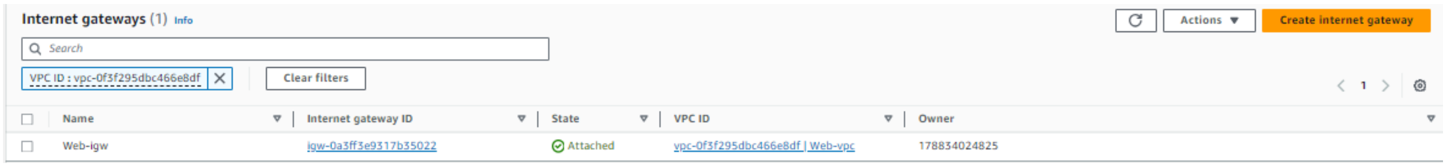
Actions | Create NAT gateway

Find resources by attribute or tag

vpc-0f3f295dbc466e8df | Clear filters

Name	NAT gateway ID	Connectivity...	State	State message	Primary public I...	Primary private I...	Primary netw
Nat-GW -1	nat-0ca5b5513e8291672	Public	Available	–	3.229.47.84	10.0.0.221	eni-002e1c0d
Nat-GW-2	nat-0a2c09337a312855c	Public	Available	–	35.168.54.202	10.0.4.33	eni-0cb3a984

Creating Internet Getways:



Internet gateways (1) Info

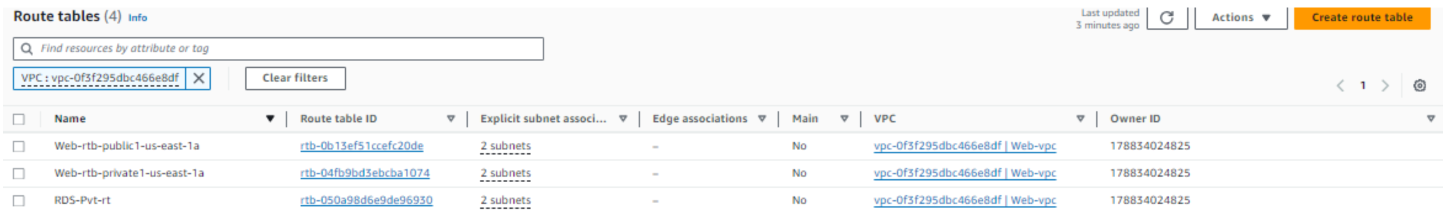
Actions | Create internet gateway

Search

VPC ID : vpc-0f3f295dbc466e8df | Clear filters

Name	Internet gateway ID	State	VPC ID	Owner
Web-igw	igw-0a3ff3e9317b35022	Attached	vpc-0f3f295dbc466e8df | Web-vpc	178834024825

Creating Rout Tables:



Route tables (4) Info

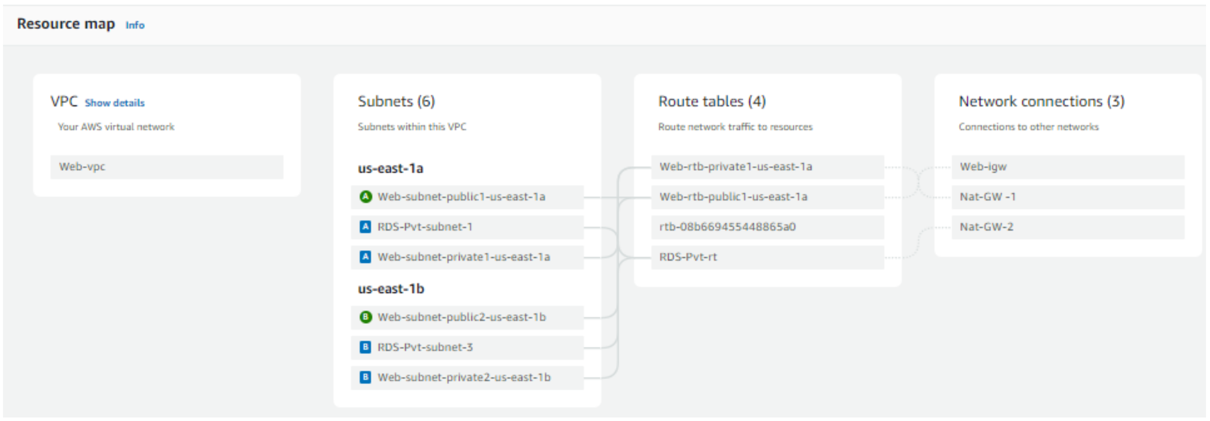
Last updated 3 minutes ago | Actions | Create route table

Find resources by attribute or tag

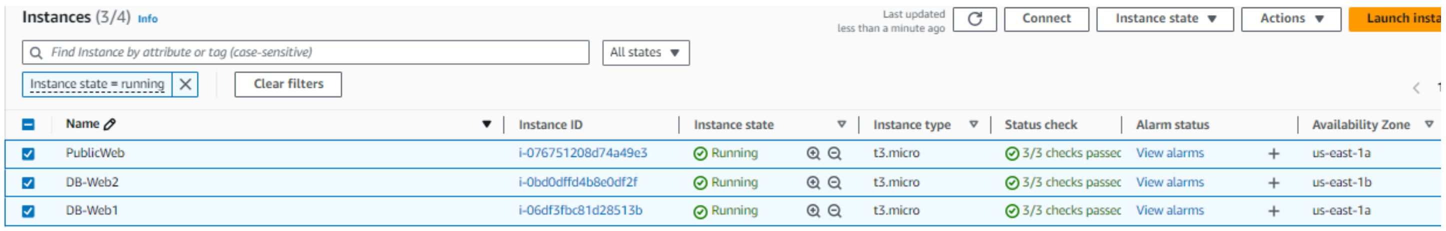
VPC : vpc-0f3f295dbc466e8df | Clear filters

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC	Owner ID
Web-rtb-public1-us-east-1a	rtb-0b13ef51ccefc20de	2 subnets	–	No	vpc-0f3f295dbc466e8df | Web-vpc	178834024825
Web-rtb-private1-us-east-1a	rtb-04fb9bd3ebcba1074	2 subnets	–	No	vpc-0f3f295dbc466e8df | Web-vpc	178834024825
RDS-Pvt-rt	rtb-050a98d6e9de96930	2 subnets	–	No	vpc-0f3f295dbc466e8df | Web-vpc	178834024825

Resource Map:



2.Creating Virtual Machine:



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
PublicWeb	i-076751208d74a49e3	Running	t3.micro	3/3 checks passec	View alarms	us-east-1a
DB-Web2	i-0bd0dffd4b8e0df2f	Running	t3.micro	3/3 checks passec	View alarms	us-east-1b
DB-Web1	i-06df3fbc81d28513b	Running	t3.micro	3/3 checks passec	View alarms	us-east-1a

Public Web User Data:

Purpose of the Script:

This script automates the setup of a web server and database on an AWS EC2 instance. It does the following:

- Installs required packages for running a Node.js application and MySQL.
- Downloads the application code and installs necessary dependencies.
- Configures a MySQL database, creates a user, and sets up a students table to store user data.
- Exposes MySQL to external connections and restarts the service.
- Exports environment variables to configure the application's connection to the database.
- Starts the Node.js web application.
- Sets up an auto-start script to ensure that the application restarts on system reboot.

This makes the EC2 instance a fully functioning web server, with a MySQL database backing it, that can handle incoming HTTP requests.

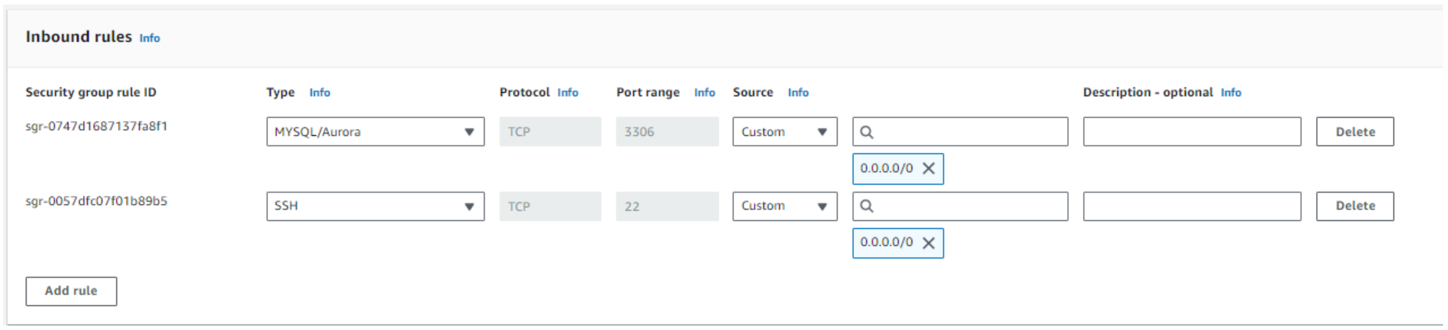
DB Web User Data:

Purpose of the Script:

- Updates the EC2 instance's package list and installs required software (Node.js, NPM, MySQL client, and utilities).
- Downloads and extracts the web application code from an AWS S3 bucket.
- Installs necessary dependencies, specifically the AWS SDK.
- Configures the Node.js application to run on port 80.
- Starts the application and configures it to automatically start whenever the instance is rebooted.
- This setup assumes the MySQL database is hosted separately, as only the MySQL client is installed. The web application will run on an EC2 instance and listen for HTTP requests.

3. Creating Security Groups:

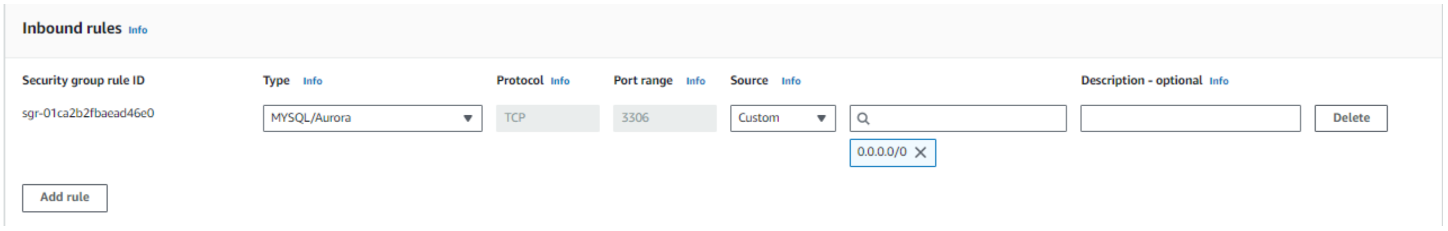
EC2-SG:



Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-0747d1687137fa8f1	MYSQL/Aurora	TCP	3306	Custom	0.0.0.0/0	
sgr-0057dfc07f01b89b5	SSH	TCP	22	Custom	0.0.0.0/0	

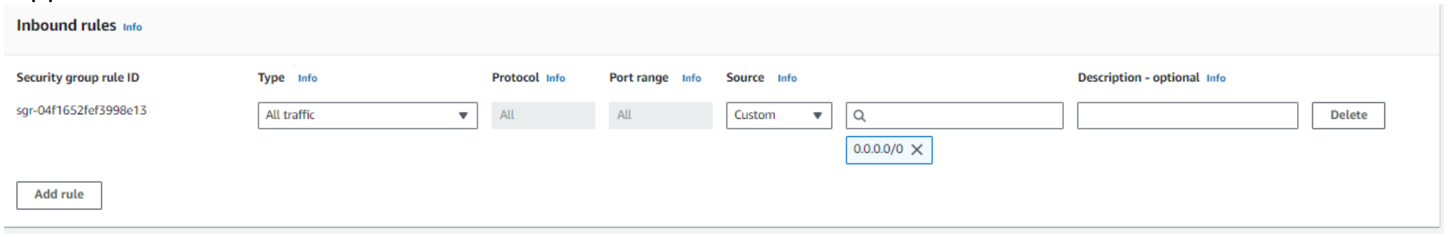
Database -SG:



Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-01ca2b2fbaead46e0	MYSQL/Aurora	TCP	3306	Custom	0.0.0.0/0	

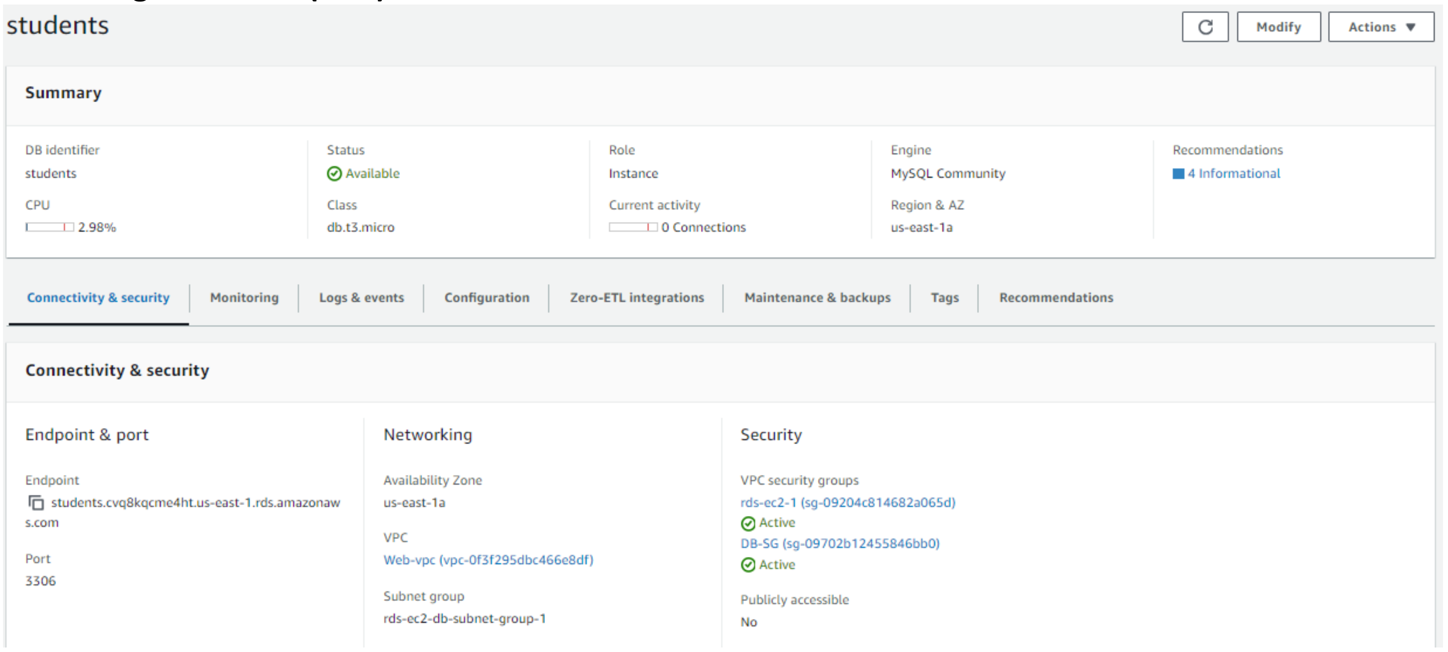
Application Loadbalancer -SG:



Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-04f1652fef3998e13	All traffic	All	All	Custom	0.0.0.0/0	

4.Creating Data Base (RDS):



students

Modify | Actions

Summary

DB identifier	Status	Role	Engine	Recommendations
students	Available	Instance	MySQL Community	4 Informational
CPU	Class	Current activity	Region & AZ	
2.98%	db.t3.micro	0 Connections	us-east-1a	

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

Connectivity & security

Endpoint & port

Endpoint
students.cvq8kqcme4ht.us-east-1.rds.amazonaws.com

Port
3306

Networking

Availability Zone
us-east-1a

VPC
Web-vpc (vpc-0f3f295dbc466e8df)

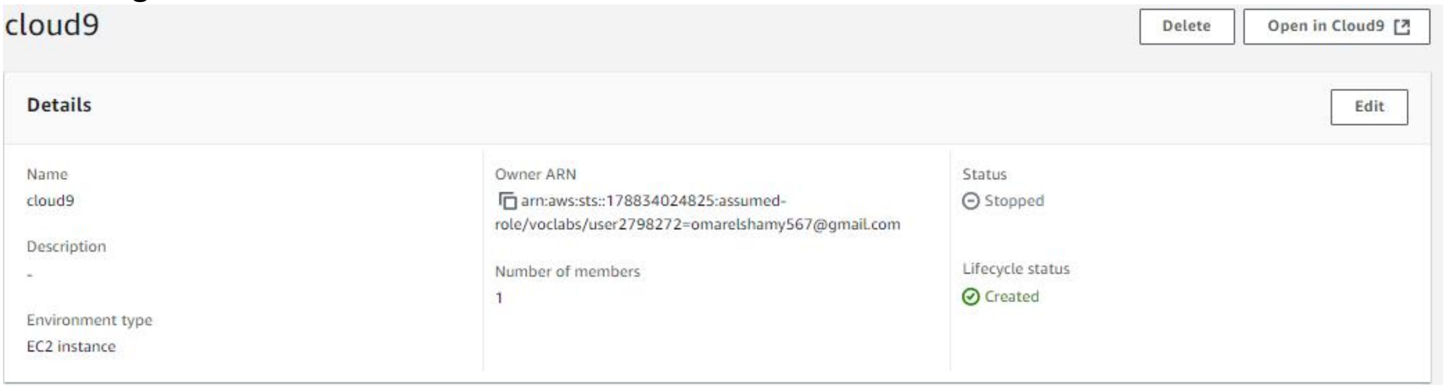
Subnet group
rds-ec2-db-subnet-group-1

Security

VPC security groups
rds-ec2-1 (sg-09204c814682a065d)
Active
DB-SG (sg-09702b12455846bb0)
Active

Publicly accessible
No

5.Creating Cloud9 Environment:



cloud9

Delete | Open in Cloud9

Details Edit

Name
cloud9

Description
-

Environment type
EC2 instance

Owner ARN
arn:aws:sts::178834024825:assumed-role/voclabs/user2798272=omarelshamy567@gmail.com

Number of members
1

Status
Stopped

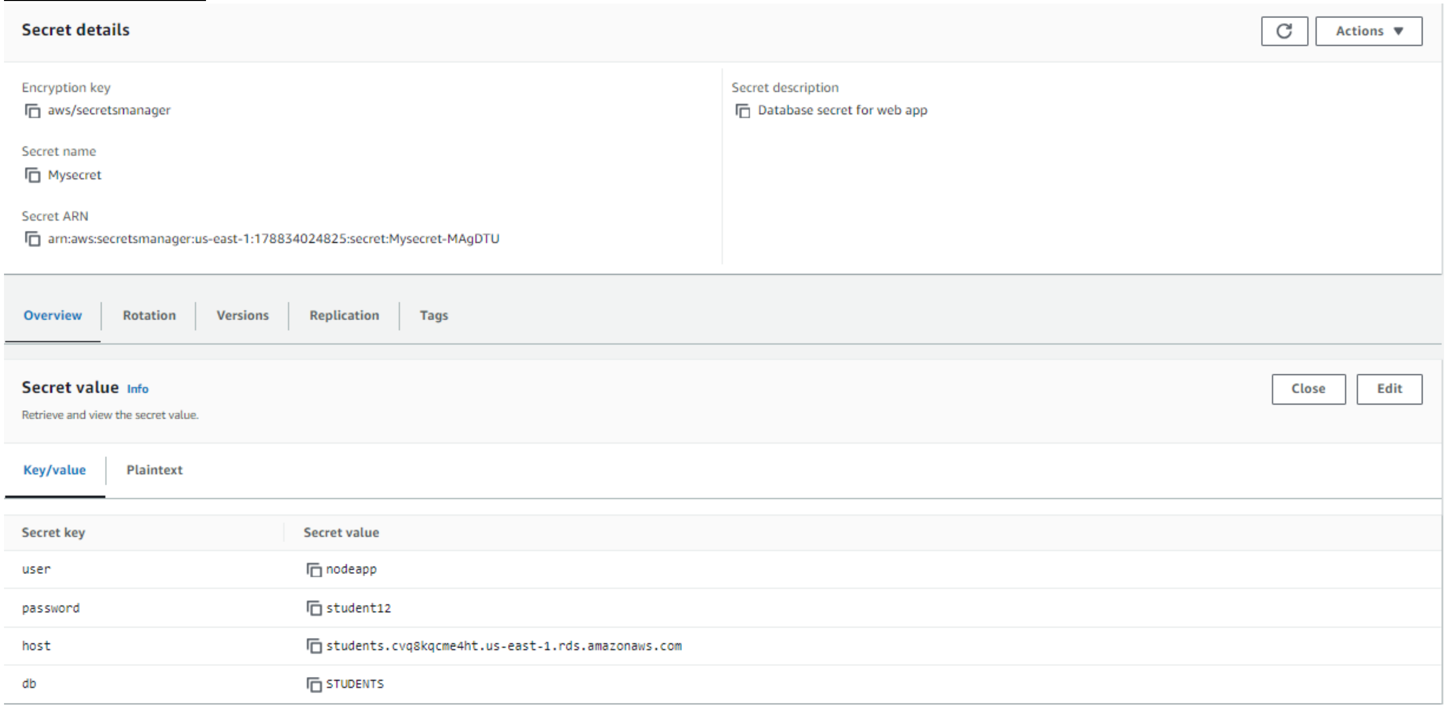
Lifecycle status
Created

Creating Secret Manger:

```
# Use following script to create secret in secret manager for accessing your database via the web application
# Replace the values for below placeholders with the actual values you have used to configure the service
#<RDS Endpoint>
#<password>
#<username>
#<dbname>

aws secretsmanager create-secret \
    --name Mysecret \
    --description "Database secret for web app" \
    --secret-string "{\"user\":\"nodeapp\",\"password\":\"student12\",\"host\":\"students.cvq8kqcme4ht.us-east-1.rds.amazonaws.com\",\"db\":\"STUDENTS\"}"
```

Secret Manger:



Secret details

Encryption key
aws/secretsmanager

Secret name
Mysecret

Secret ARN
arn:aws:secretsmanager:us-east-1:178834024825:secret:Mysecret-MAgDTU

Secret description
Database secret for web app

Secret value
Retrieve and view the secret value.

Secret key	Secret value
user	nodeapp
password	student12
host	students.cvq8kqcme4ht.us-east-1.rds.amazonaws.com
db	STUDENTS

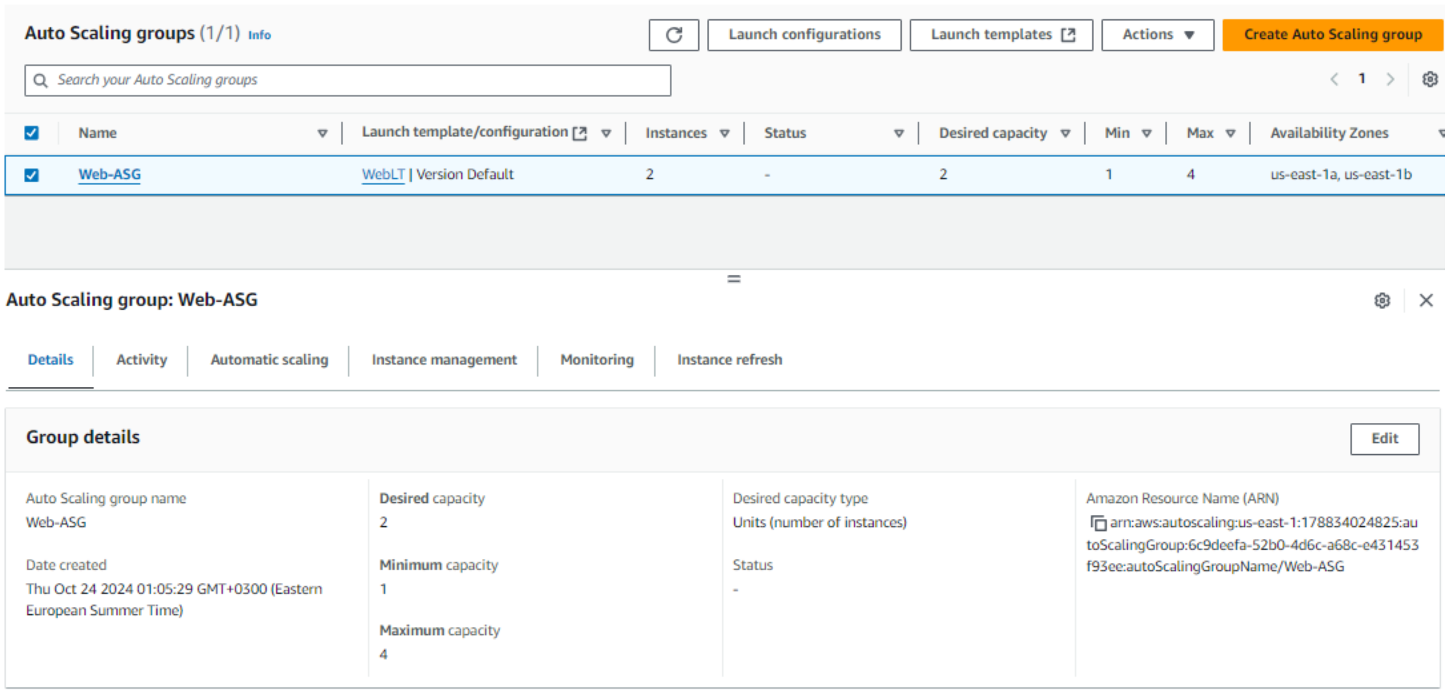
Migration:

```
## Migration
# Following command exports the data from existing server.
# Prerequisites - This script expects mysql-client, mysqlpdump to be present
# AWS Cloud9 environment already has all the prerequisites installed for running these scripts

# Replace the <EC2instancePrivateip> with internal IP address of the EC2 instance (CapstonePOC) created in Phase-2 earlier.
# Provide the password when prompted

students.cvq8kqcme4ht.us-east-1.rds.amazonaws.com
#Following command imports the data into RDS database. Replace <RDSEndpoint> with the RDS Database endpoint you noted after RDS Database created in earlier steps.
#when prompted, enter password you provided during the time of database creation
mysql -h students.cvq8kqcme4ht.us-east-1.rds.amazonaws.com -u nodeapp -p  STUDENTS < data.sql
```

6.Creating Auto Scaling Group:



Auto Scaling groups (1/1)

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
Web-ASG	WebLT | Version Default	2	-	2	1	4	us-east-1a, us-east-1b

Auto Scaling group: Web-ASG

Group details

Auto Scaling group name
Web-ASG

Date created
Thu Oct 24 2024 01:05:29 GMT+0300 (Eastern European Summer Time)

Desired capacity
2

Minimum capacity
1

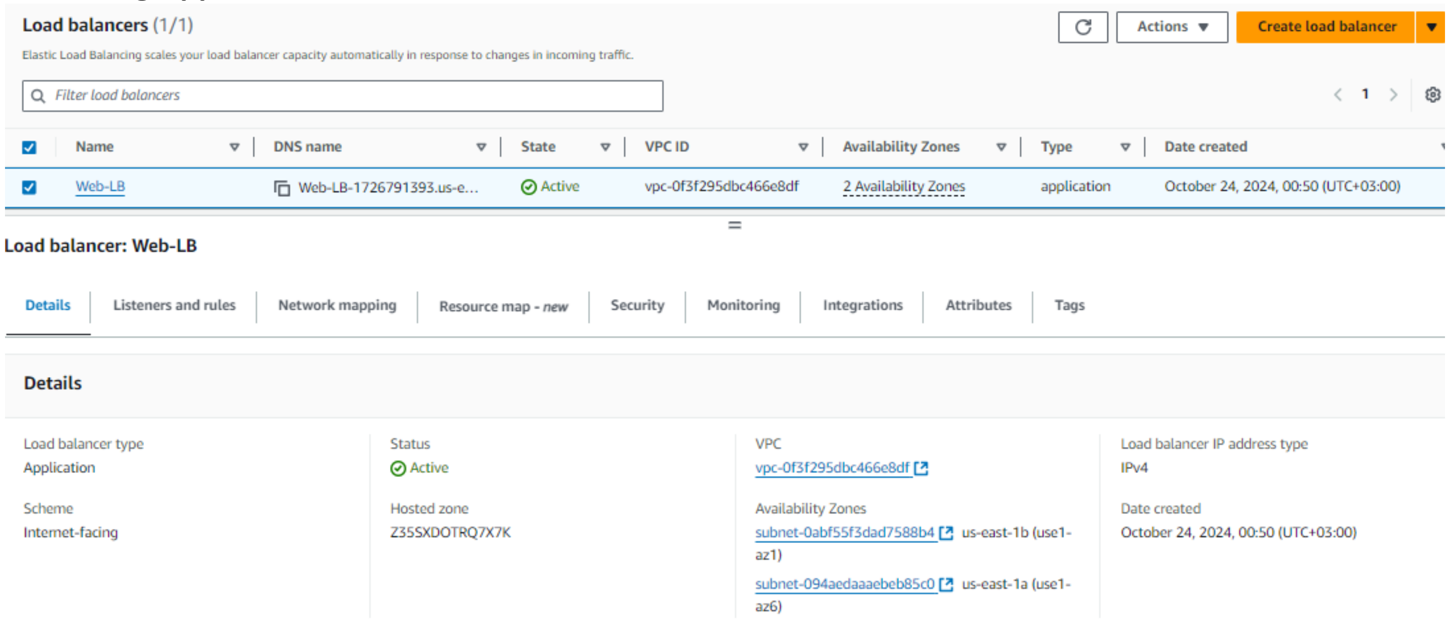
Maximum capacity
4

Desired capacity type
Units (number of instances)

Status
-

Amazon Resource Name (ARN)
arn:aws:autoscaling:us-east-1:178834024825:autoScalingGroup:6c9deefa-52b0-4d6c-a68c-e431453f93ee:autoScalingGroupName/Web-ASG

7.Creating Application Loadbalancer:



Load balancers (1/1)

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 created
Web-LB	Web-LB-1726791393.us-e...	Active	vpc-0f3f295dbc466e8df	2 Availability Zones	application	October 24, 2024, 00:50 (UTC+03:00)

Load balancer: Web-LB

Details

Load balancer type Application	Status Active	VPC vpc-0f3f295dbc466e8df	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone Z35SXDOTRQ7X7K	Availability Zones subnet-0abf55f3dad7588b4 us-east-1b (use1-az1) subnet-094aedaaaebeb85c0 us-east-1a (use1-az6)	Date created October 24, 2024, 00:50 (UTC+03:00)

8.ALB Test using cloud9:

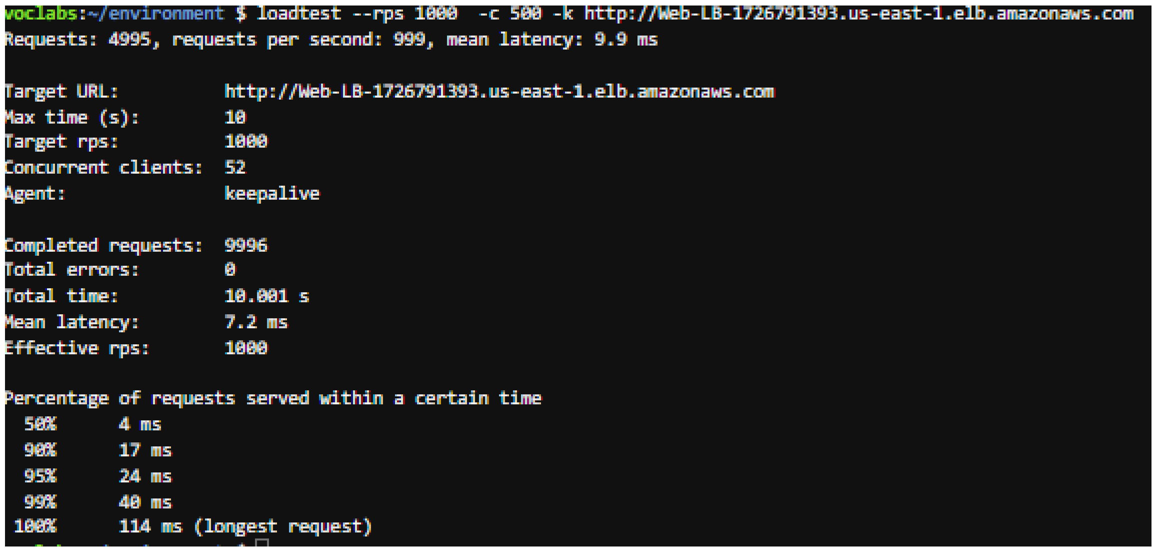
```
#Following command installs #loadtest package to perform load testing on the application
#Prerequisites are mentioned in the resources section

npm install -g loadtest

#Following command performs load testing on the given URL. replace the URL with Loadbalancer (or Public IP of EC2 instance)
# Press ctrl +C to stop the script

loadtest --rps 1000  -c 500 -k http://ALB-EC2-1750404363.us-east-1.elb.amazonaws.com
```

Result:



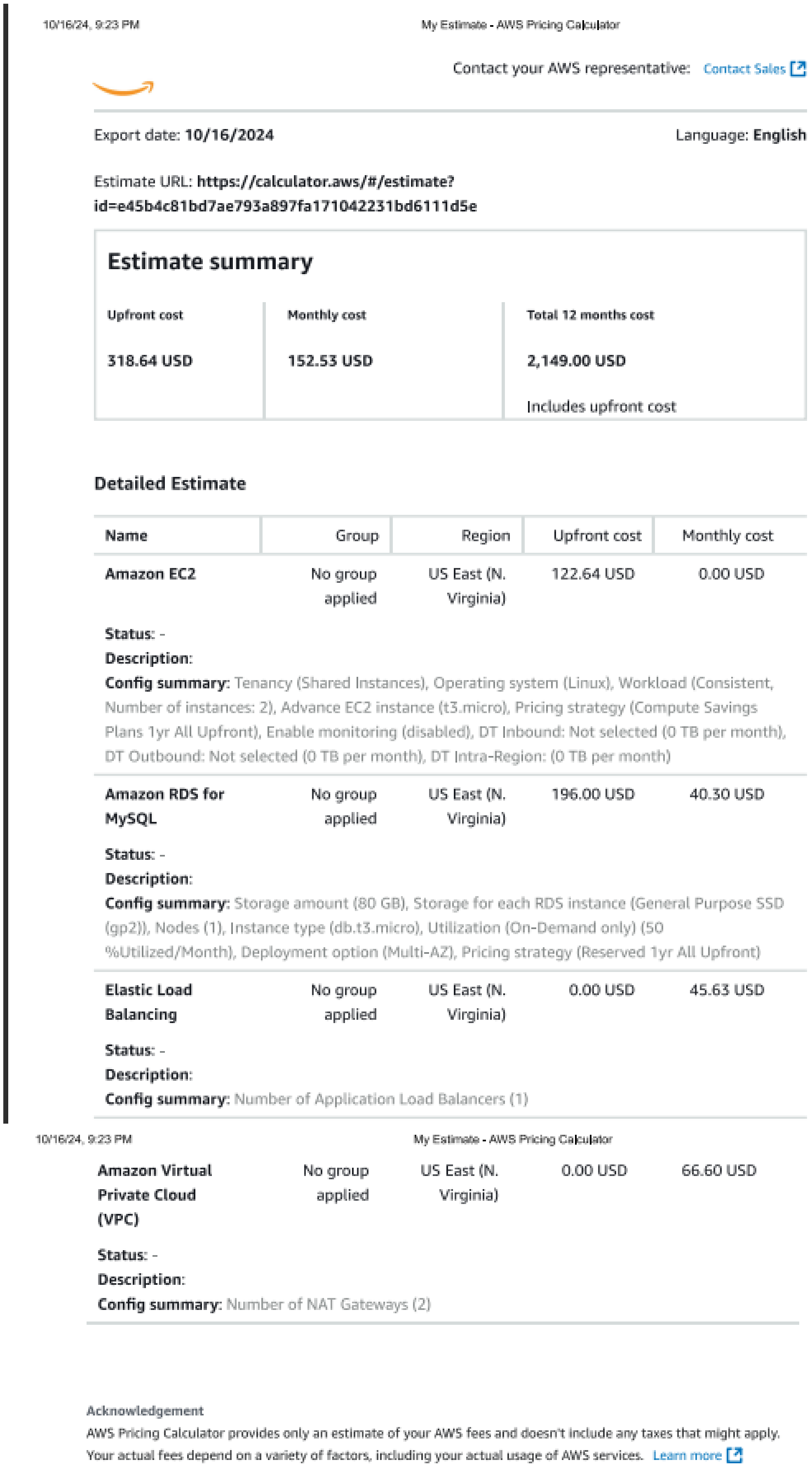
```
voclabs:~/environment $ loadtest --rps 1000  -c 500 -k http://Web-LB-1726791393.us-east-1.elb.amazonaws.com
Requests: 4995, requests per second: 999, mean latency: 9.9 ms

Target URL:          http://Web-LB-1726791393.us-east-1.elb.amazonaws.com
Max time (s):        10
Target rps:          1000
Concurrent clients:  52
Agent:               keepalive

Completed requests:  9996
Total errors:        0
Total time:          10.001 s
Mean latency:        7.2 ms
Effective rps:       1000

Percentage of requests served within a certain time
  50%      4 ms
  90%      17 ms
  95%      24 ms
  99%      40 ms
 100%      114 ms (longest request)
```

Estimated Coste:



10/16/24, 9:23 PM My Estimate - AWS Pricing Calculator

Contact your AWS representative: Contact Sales

Export date: **10/16/2024** Language: **English**

Estimate URL: **https://calculator.aws/#/estimate?id=e45b4c81bd7ae793a897fa171042231bd6111d5e**

Estimate summary

Upfront cost	Monthly cost	Total 12 months cost
318.64 USD	152.53 USD	2,149.00 USD Includes upfront cost

Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Amazon EC2	No group applied	US East (N. Virginia)	122.64 USD	0.00 USD
Amazon RDS for MySQL	No group applied	US East (N. Virginia)	196.00 USD	40.30 USD
Elastic Load Balancing	No group applied	US East (N. Virginia)	0.00 USD	45.63 USD
Amazon Virtual Private Cloud (VPC)	No group applied	US East (N. Virginia)	0.00 USD	66.60 USD

Amazon EC2:

Status: -
Description:
Config summary: Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 2), Advance EC2 instance (t3.micro), Pricing strategy (Compute Savings Plans 1yr All 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)

Amazon RDS for MySQL:

Status: -
Description:
Config summary: Storage amount (80 GB), Storage for each RDS instance (General Purpose SSD (gp2)), Nodes (1), Instance type (db.t3.micro), Utilization (On-Demand only) (50 %Utilized/Month), Deployment option (Multi-AZ), Pricing strategy (Reserved 1yr All Upfront)

Elastic Load Balancing:

Status: -
Description:
Config summary: Number of Application Load Balancers (1)

10/16/24, 9:23 PM My Estimate - AWS Pricing Calculator

Amazon Virtual Private Cloud (VPC):

Status: -
Description:
Config summary: Number of NAT Gateways (2)

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 lab provided hands-on experience in deploying a highly available, scalable Node.js web application on AWS infrastructure. Through a combination of automation, database management, and cloud integration, I successfully implemented a full-stack web application with a MySQL backend hosted on an EC2 instance. The use of Bash scripts streamlined the deployment process, ensuring consistency, security, and persistence in case of instance reboots.

The lab also highlighted the importance of setting environment variables for database credentials and using AWS SDK for seamless integration with cloud services. Additionally, it showcased the scalability and reliability of AWS as a platform for hosting modern web applications.

What I Learned Through This Lab:

1. **Automation of Deployment**: I learned how to automate the deployment of a Node.js web application on an AWS EC2 instance using Bash scripts. This ensures efficient, consistent infrastructure and application deployment.
2. **Working with AWS SDK**: I gained experience with the AWS SDK for JavaScript (Node.js), enabling applications to interact with AWS services programmatically, automating cloud operations.
3. **Database Setup and Configuration**: I learned how to install and configure MySQL, including creating databases, users, and tables. I also learned to adjust MySQL's bind address to allow external access, which is crucial for multi-instance environments.
4. **Application Persistence on Reboot**: I configured scripts to ensure the web application restarts automatically after system reboots, vital for ensuring up time in production systems.
5. **Environment Variables and Security**: I understood the importance of securely managing sensitive information, like database credentials, through environment variables, allowing flexibility and protection in configuration.
6. **Using AWS EC2:** I learned how to integrate AWS services like EC2 into the workflow, demonstrating AWS's ability to host and manage scalable web applications efficiently.
7. **Hands-on Experience with Node.js Applications**: This lab provided practical experience in deploying and managing a real-world Node.js application, enhancing my skills in building cloud-based solutions.