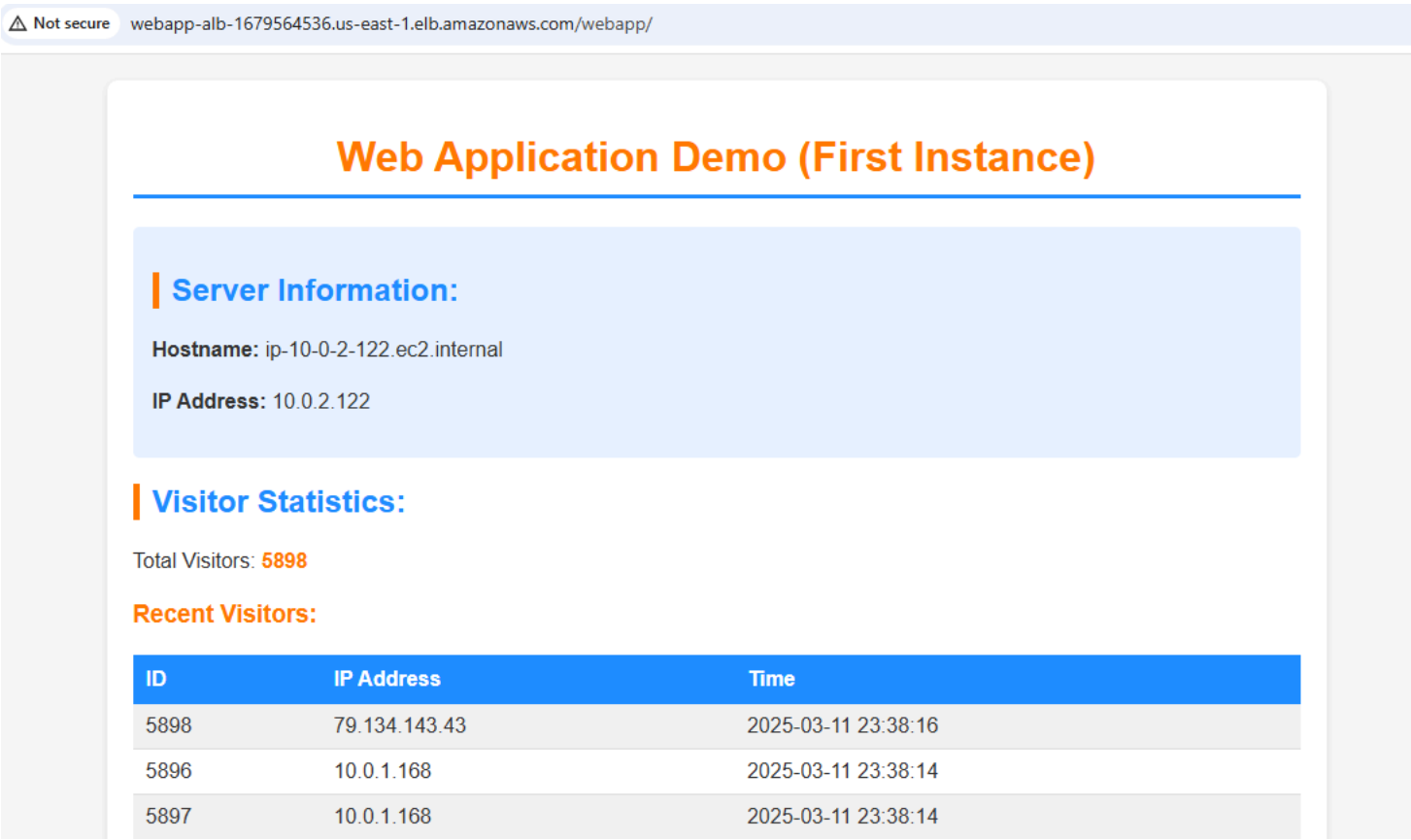
AWS Infrastructure Implementation

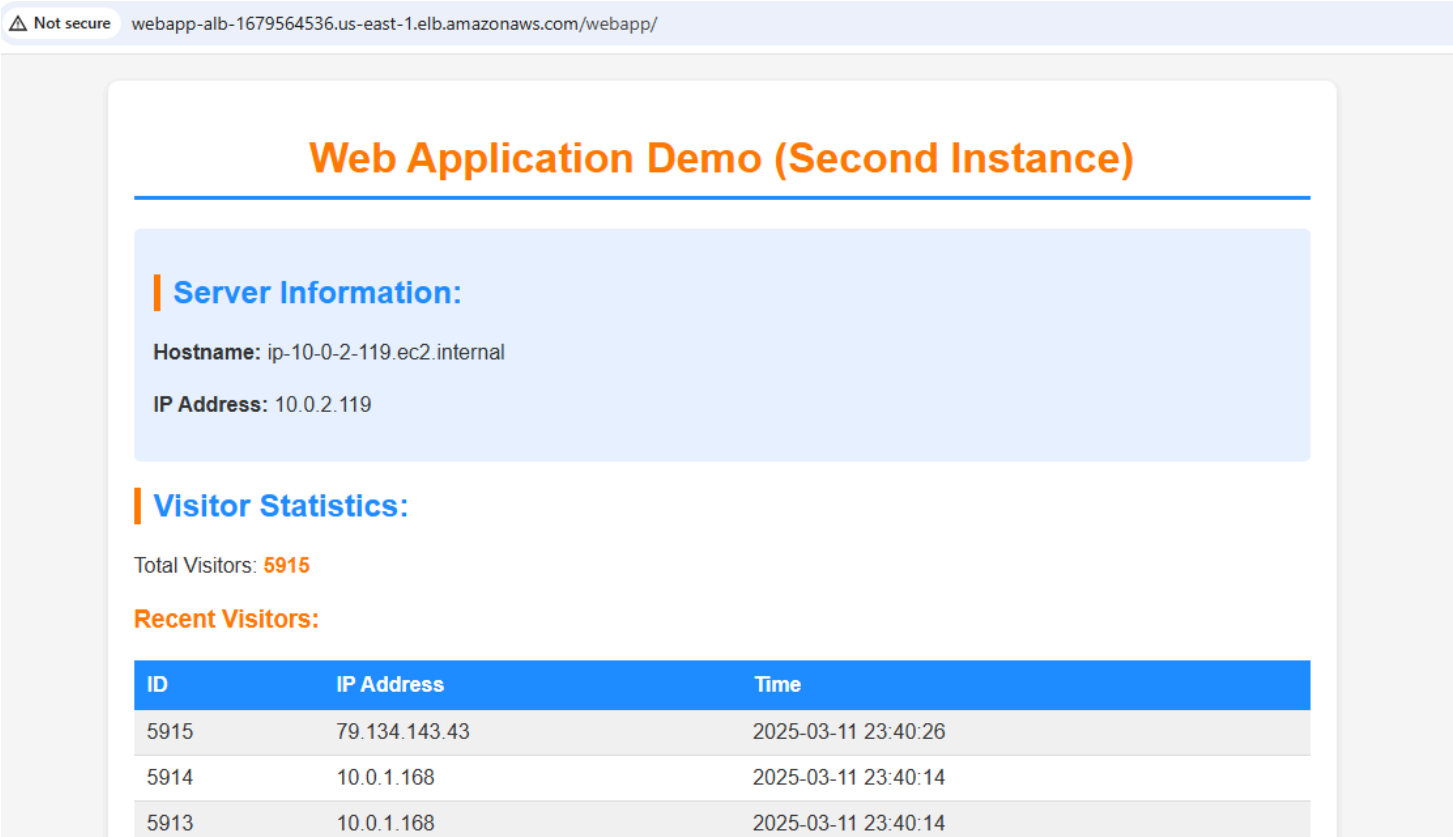
By Hamza Alhalabi

Table of Contents

Introduction

In this assignment, I built an three-tier infrastructure using Amazon Web Services. The infrastructure represents a web application using PHP and MySQL technologies and a group of hardware components that are connected with private and public networks to combine the accessibility and security of the application as needed.



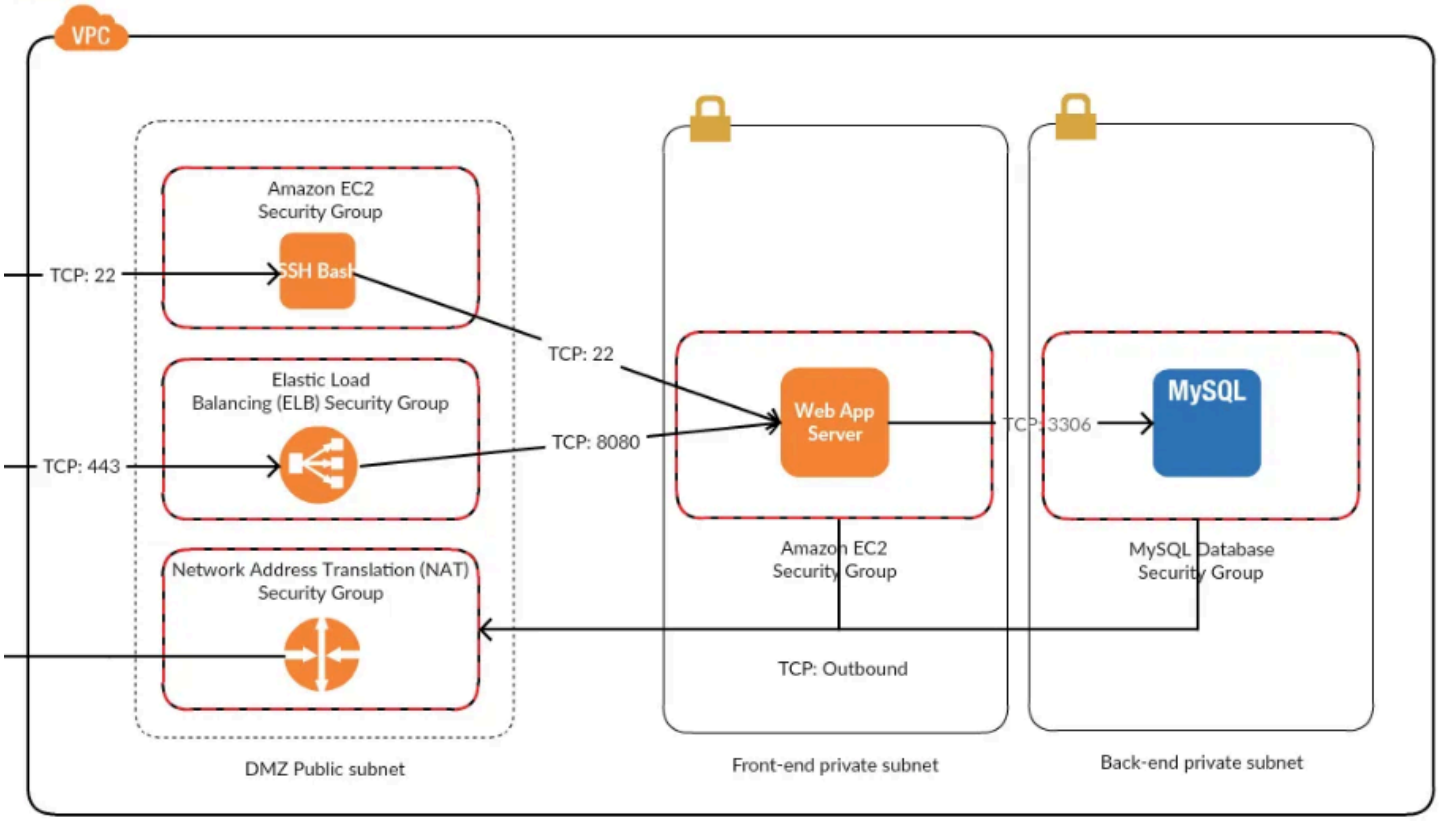


Project Requirements

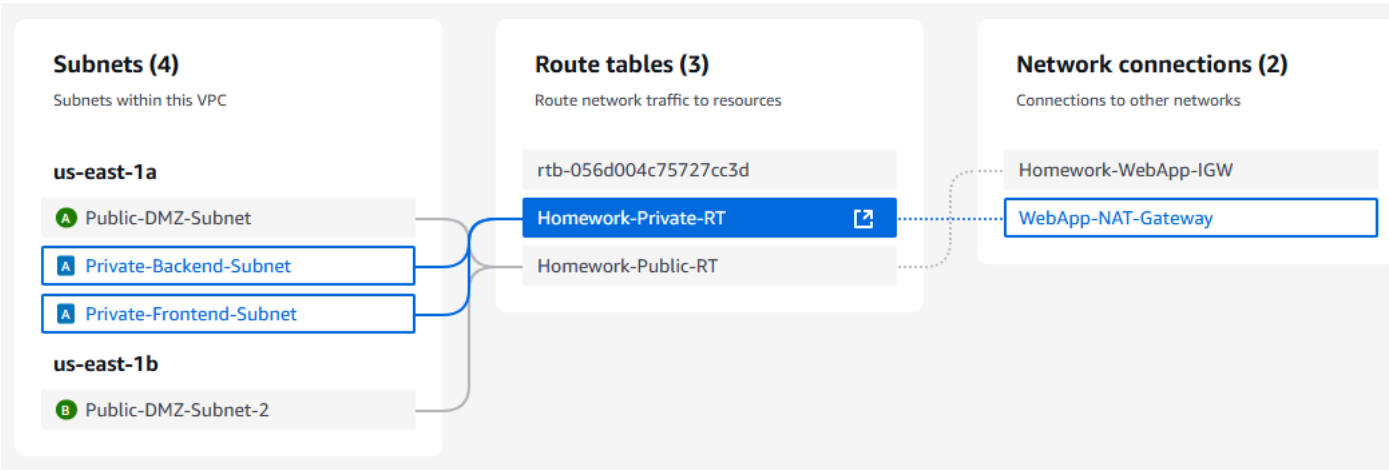
- AWS requires a kind of (**complex**) signing up process, as it gives free services for new user (one year of free-tier features), it requires a valid credit card and details information about the user and their address. This process may take time more than expected.
- I installed **AWS CLI** in my device before starting, but I did not rely on it while building the assignment infrastructure.
- I also installed **Terraform** to benefit from IoC freatures and a tool called **Terraformer** that convert a built services into infrastructure code. I used them when I was studying the lectures but I did not have the time to convert my infrastructure into IoC code, so I skipped it.

Architecture Overview

- All the components were inside one virtual private cloud **VPC**, this VPC have one internet gateway that manages the connections with the network outside our private cloud. There are **three subnet**, one public called "**DMZ**- subnet", and two private called "**frontend**-subnet" and "**backend**-subnet".



- The most challenging parts were route tables and security groups, I added **two route tables** "**private**-RT" and "**public**-RT" with nedded rules.



- The implementation required the creation of **five security groups**, each one with custom rules for inbound and outbound networking traffic.

Security Groups (8) Info

Actions | Export security groups to CSV

Find resources by attribute or tag

Name	Security group ID	Security group name	VPC ID
–	sg-0807c4b21114dc0cc	NAT-SG	vpc-0da8
–	sg-0444c68da3998847a	SSH-Bastion-SG	vpc-0da8
–	sg-034b231ede1889df7	ELB-SG	vpc-0da8
–	sg-00b10423daaadca89	MySQL-SG	vpc-0da8
–	sg-02aa7f5fcdc799ba1	default	vpc-0da8
–	sg-0a5333929acd56859	WebApp-SG	vpc-0da8

- The implementation contains **four EC2** instances, one for the **SSH-Bastion** host server, and two for the **web server** and the last one for **MySQL database server**.

Instances (4) Info

Last updated less than a minute ago | Connect | Instance state | Actions

Find Instance by attribute or tag (case-sensitive) | All states

Name	Instance ID	Instance state	Instance type	Status check
First-Web-Server	i-06c45a5d3bbaf0ccb	Running	t2.micro	2/2 checks passed
MySQL-Server	i-04022c6ead24cece8	Running	t2.micro	2/2 checks passed
WebApp-Basti...	i-0afca5f34b52b5ba3	Running	t2.micro	2/2 checks passed
Second-Web-S...	i-01bc1339f96c1a55f	Running	t2.micro	2/2 checks passed

- The **Application Load Balancer** (ABI) was critical for distributing the traffic betwee the two web servers. I created a **target group** containing the two servers, and used ABI instance to connect it with the target group with the suitable ports.
- At the end, I added a **NAT gateway** instance to enable the privae subnets instances to access the internet to setup what they need like PHP and MySQL.
- This was the high-level architicture for this infrastucture, I will try to add more details in the next section.

Implementation Steps

- Setting up the VPC:
  - This step is direct, just set the name of the VPC and specify the CIDR block, I chose 10.0.0.0/16 which gives the larger number of IP addresses inside the private cloud.

IPv4 CIDRs Info

Address family	CIDR	Status
IPv4	10.0.0.0/16	Associated

- Configuring subnets:
  - I created three subnets, one public and two private, and gave them suitable IPv4 CIDR blocks to distribute the IP between them in a reasonable way.

Subnets (10) Info

Last updated less than a minute ago | Actions | Creat

Find resources by attribute or tag

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
Public-DMZ-Subnet	subnet-011e9ce26355cc69d	Available	vpc-0da88e211551b62a2 | Ho...	Off	10.0.1.0/24
Private-Backend-Subnet	subnet-01e5f387f58ebbb14	Available	vpc-0da88e211551b62a2 | Ho...	Off	10.0.3.0/24
Private-Frontend-Subnet	subnet-05d6b22fad25156f4	Available	vpc-0da88e211551b62a2 | Ho...	Off	10.0.2.0/24

- Attaching internet gateway:
  - I added a custom IGW, it don't need much configurations.

Internet gateways (2) Info

Actions

Search

Name	Internet gateway ID	State	VPC ID
Homework-WebApp-IGW	igw-05d340535db22ccad	Attached	vpc-0da88e2

- Configuring route tables:
  - The public route table forwards the traffic of the public subnets to the internet gateway.

rtb-06cc85a11067ec828 / Homework-Public-RT

Details Info

Route table ID	Main	Explicit subnet associations
rtb-06cc85a11067ec828	No	2 subnets
VPC	**Owner ID**	
vpc-0da88e211551b62a2 | Homework-WebApp-VPC	864981740908	

Routes | Subnet associations | Edge associations | Route propagation | Tags

Routes (2)

Filter routes

Destination	Target	Status
0.0.0.0/0	igw-05d340535db22ccad	Active
10.0.0.0/16	local	Active

  - The pribate route table forwards the traffic of the two private subnets to the NAT gateway instance.

rtb-07c9f12a7c85b5157 / Homework-Private-RT

Details Info

Route table ID	Main	Explicit subnet associations
rtb-07c9f12a7c85b5157	No	2 subnets
VPC	**Owner ID**	
vpc-0da88e211551b62a2 | Homework-WebApp-VPC	864981740908	

Routes | Subnet associations | Edge associations | Route propagation | Tags

Routes (2)

Filter routes

Destination	Target	Status
0.0.0.0/0	nat-06481ba2919ef0da4	Active
10.0.0.0/16	local	Active

- Configuring security groups
  - Firstly, I created the SSH-bastion host security group and added the rules to allow only the admin to access it using SSH (TCP 22) and the specified key-pair.

Inbound rules (1)

Manage tags | Edit inbou

Search

	IP version	Type	Protocol	Port range	Source	Description
073	IPv4	SSH	TCP	22	0.0.0.0/0	SSH access only

  - The web servers security group allows the acces of SSH (on TCP 22) for the admin (Bastion-Host is the source), and allows requests traffic from the load balancer on TCP 8080 (ALB is the source).

Inbound rules (4)

Manage tags | Edit inbound

Search

Security group rule ID	IP version	Type	Protocol	Port range	Source
sgr-097a8ae10376eedd4	–	Custom TCP	TCP	8080	sg-034b231ede1889df...
sgr-0e0f5d09069c328fb	–	SSH	TCP	22	sg-0444c68da3998847...
sgr-0f345de4d1e11705a	–	All traffic	All	All	sg-0807c4b21114dc0cc...
sgr-03f23f162d21d8ef9	–	HTTP	TCP	80	sg-034b231ede1889df...

  - MySQL server security group was similar with web servers security group in the SSH access, it also allows MySQL access from the web servers on the port TCP 3306.

Inbound rules (2)

Manage tags | Edit

Search

Security group rule ID	IP version	Type	Protocol	Port range	Source
sgr-005efbd4031c00383	–	MYSQL/Aurora	TCP	3306	sg-0a5333929ac
sgr-059bf9d510f5101be	–	SSH	TCP	22	sg-0444c68da39

  - Load balancer security group was a bit problematic for me, I managed the inbound and outbound traffic to the web servers as well.

Inbound rules (2)

Manage tags | Edi

Search

Security group rule ID	IP version	Type	Protocol	Port range	Source
sgr-046715a9d2f8215c1	IPv4	HTTP	TCP	80	0.0.0.0/0
sgr-0f412f630cd8b2b8b	IPv4	HTTPS	TCP	443	0.0.0.0/0

Outbound rules (1)

Manage tags | E

Search

Security group rule ID	IP version	Type	Protocol	Port range	Destination
sgr-036b0550de627ce51	IPv4	All traffic	All	All	0.0.0.0/0

- The last one was the NAT gateway security group, it was the least complicated one between all previous security groups.

- Deploying EC2 instances:
  - First EC2 instance was the SSH-Bastion host instance, it was special because it contained a public address and public internet connection.

Instance summary for i-0afca5f34b52b5ba3 (WebApp-Bastion-Host) Info

Connect

Updated 4 minutes ago

Instance ID i-0afca5f34b52b5ba3	**Public IPv4 address** 3.87.28.197 | open address	**Private IPv4 addresses** 10.0.1.235
IPv6 address –	**Instance state** Running	**Public IPv4 DNS** –
Hostname type IP name: ip-10-0-1-235.ec2.internal	**Private IP DNS name (IPv4 only)** ip-10-0-1-235.ec2.internal	

  - Other three EC2 insatances for web servers and MySQL server were almost the same in creation.

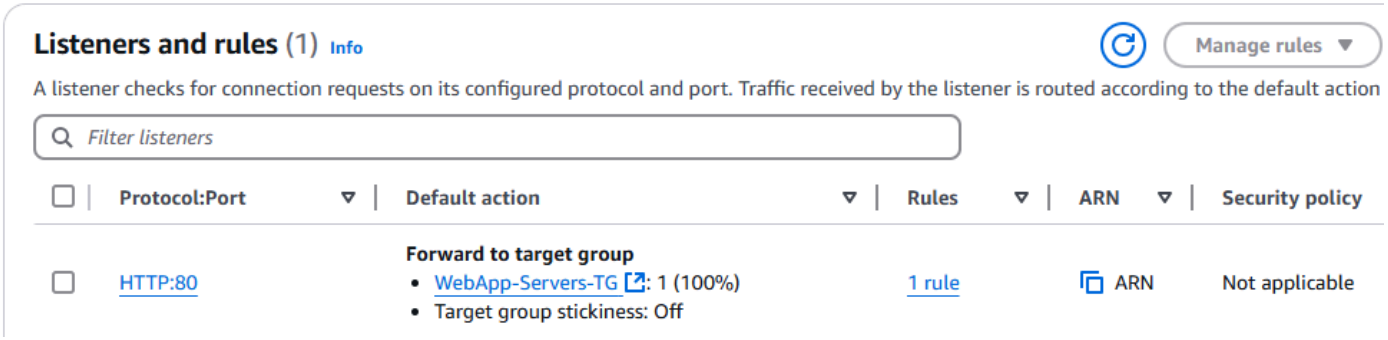
- Setting up the servers:
  - I forwarded the SSH key-pair from my local computer to the Bastion host and then to the web servers and MySQL server, in this way, I became able to install what I want on the machines, I installed **Apache** and **PHP** on the web servers, and installed **MySQL** (MariaDB) to the database server by running the required command, and by creating PHP files and databases when needed.

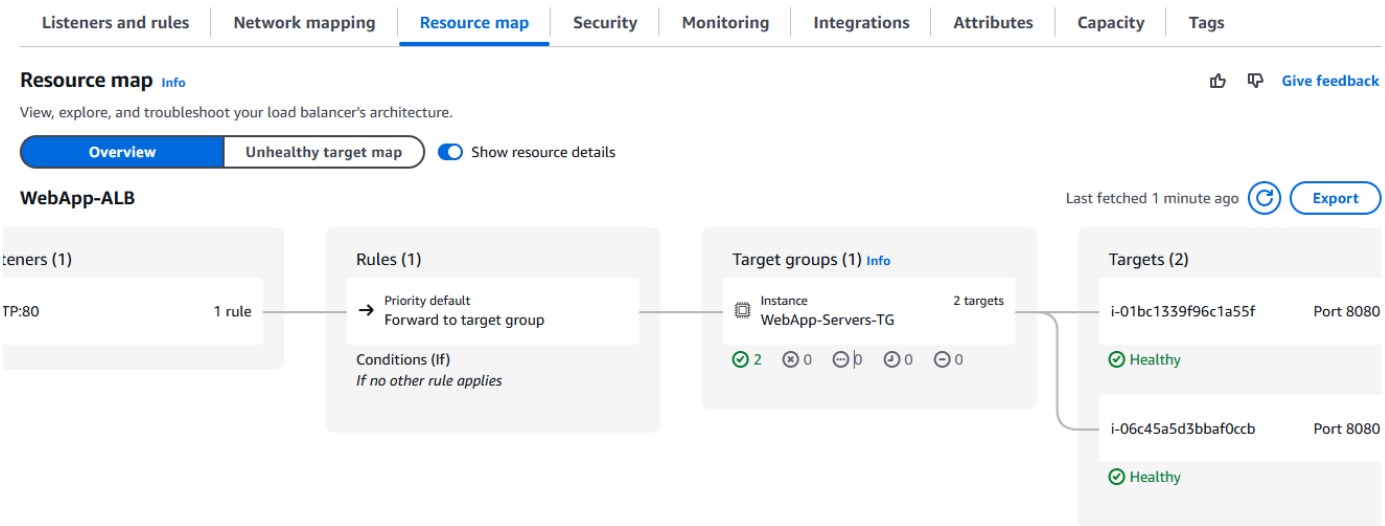
```
[ec2-user@ip-10-0-3-78 ~]$ sudo yum install mariadb105-server -y
Last metadata expiration check: 0:06:43 ago on Tue Mar 11 11:31:52 2025.
Dependencies resolved.
=================================================================
 Package     Arch   Version                    Repository   Size
=================================================================
Installing:
 mariadb105-server
             x86_64 3:10.5.25-1.amzn2023.0.1   amazonlinux  11 M
Installing dependencies:
 mariadb-connector-c
             x86_64 3.1.13-1.amzn2023.0.3      amazonlinux 196 k
 mariadb-connector-c-config
             noarch 3.1.13-1.amzn2023.0.3      amazonlinux 9.2 k
 mariadb105  x86_64 3:10.5.25-1.amzn2023.0.1   amazonlinux 1.6 M
 mariadb105-common
             x86_64 3:10.5.25-1.amzn2023.0.1   amazonlinux  29 k
 mariadb105-errmsg
             x86_64 3:10.5.25-1.amzn2023.0.1   amazonlinux 213 k
 mysql-selinux
             noarch 1.0.4-2.amzn2023.0.3       amazonlinux  36 k
```

```
MariaDB [(none)]> CREATE DATABASE webapp_db;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> CREATE USER 'hamza'@'%' IDENTIFIED BY 'usertiger';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON webapp_db.* TO 'hamza'@'%';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| webapp_db          |
+--------------------+
4 rows in set (0.000 sec)
```

```php
<?php
// Database connection parameters
$servername = "10.0.3.78";  // Private IP of your backend MariaDB server
$username = "hamza";          // The user you created in MariaDB
$password = "usertiger";        // Your database password
$dbname = "webapp_db";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

function getUserIP() {
    // Check for X-Forwarded-For header first (for clients behind proxies/load balancers)
    if (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
        // HTTP_X_FORWARDED_FOR can contain multiple IPs (client, proxies) separated by commas
        // The leftmost IP is typically the original client IP
        $ips = explode(',', $_SERVER['HTTP_X_FORWARDED_FOR']);
        return trim($ips[0]);
    }
    // If no forwarded IP, try the standard REMOTE_ADDR
    elseif (!empty($_SERVER['REMOTE_ADDR'])) {
        return $_SERVER['REMOTE_ADDR'];
    }
    // If both methods fail
    return 'Unknown';
}
```

- Configuring load balancing:
  - Application load balancer was added to organize the requests traffic to the web app servers, it sends the traffic to the servers on the port TCP 8080 after recieving the request on HTTP 80.





```
Load balancer name is: WebApp-ALB-1679564536.us-east-1.elb.amazonaws.com
```

- Setting NAT gateway:
  - This process was of course before the preparation of the servers because NAT is the only way to access the internet for private instances like servers in my infrastructure.
  - All traffic that are managed by the private route table should be forwarded to this gateway.

nat-06481ba2919ef0da4 / WebApp-NAT-Gateway

Details

NAT gateway ID nat-06481ba2919ef0da4	**Connectivity type** Public	**State** Available	**State message** Info –
NAT gateway ARN arn:aws:ec2:us-east-1:864981740908:natgateway/nat-06481ba2919ef0da4	**Primary public IPv4 address** 52.6.182.193	**Primary private IPv4 address** 10.0.1.47	**Primary network interface ID** eni-05c1c0bae9f276371
VPC vpc-0da88e211551b62a2 / Homework-WebApp-VPC	**Subnet** subnet-011e9ce26355cc69d / Public-DMZ-Subnet	**Created** Tuesday, March 11, 2025 at 14:03:55 GMT+3	**Deleted** –

Challenges and Solutions

- The main challenge was "*how to start?*", I take the initial steps following Dr. Motassem guides in implementing the example in the lecture. After that I was capable to move forward by myself.
- I faced a problem with the network that took a lot of time for me to figure it out, I did not know how to connect the private servers to the internet to install and preare the servers' environment, I tried to use a normal EC2 instance as a NAT instance but it required a lot of configuration, so I used an independent NAT gateway instance to connect internet to the private subnets.
- There were some problems with the load balance and servers in terms of network ports (80 or 8080) and HTTP or HTTPS, then I relized that HTTPS is not usable by anyone as it requires SSL certificate.
- When writing the PHP code, I faced some errors but I used a LLM to fix it.
- I tried to add more thing like Route53 but I found it expensive without real addition for the project.
- I had the intention to write the whole project as IoC code, but I did not have that much time.

Conclusion

I learnt how to build a basic infrastructure with cloud services like AWS, it was the first time for me buying a computing service over the internet.

I got knowledge in networking types and ports, as I took the course in the university but did not really applied it in the real-world.