

## DevOps Engineering on AWS Cloud

### **Project: Building a Highly Available, Scalable Web Application**

Group 3: Amina Merić & Selma Imširović

The goal of this project is to plan, design, build, and deploy a web application to the AWS Cloud in a way that is consistent with best practices of the AWS Well-Architected Framework.

- *First Phase:*

#### **Task 1:**

Starting with the first phase, we have planned out architectural design for the infrastructure and for every next phase, we created it using the Lucid Chart tool.

#### **Task 2:**

Additionally we estimated the costs that we will spend building the web application using the AWS Pricing Calculator. In the AWS Pricing Calculator we added all the services that we used in our infrastructure which are Amazon Private Cloud (VPC), Amazon RDS for MySQL and the Amazon EC2 instance. At the end our estimated monthly cost is 55.70 USD.

- *Second Phase:*

#### **Task 1:**

In this phase, we first created an Amazon Virtual Private Cloud (VPC) named "amina-vpc" in the US East (N. Virginia) / us-east-1 region. Our VPC consists of two public subnets and two private subnets. One pair of public and private subnets (public1 and private1) is in the us-east-1a availability zone, while the other pair (public2 and private2) is in the us-east-1b availability zone.

Next, we created an internet gateway and attached it to the VPC to enable communication with the internet. Then we created a Route Table for our VPC. We

added routes to this Route Table for the internet gateway and associated the Route Table with our public subnets. To allow the private subnets to access the internet, we created a NAT Gateway in one of the public subnets and added the necessary routes to the Route Table associated with the private subnets.

After successfully creating the Route Tables, we created two Security Groups: one for the EC2 instances and the other for the database. We configured the inbound rules for these Security Groups to allow the necessary inbound traffic. With this setup, we successfully launched our VPC.

## Task 2:

In the second task we launched an Amazon EC2 instance. For the amazon machine Image (AMI) we selected Ubuntu, and the instance type is t2.micro. For this instance we selected the VPC and subnets previously created, and attached the security groups from task one.

In order to install the web application and database on our virtual machine, we executed this script:

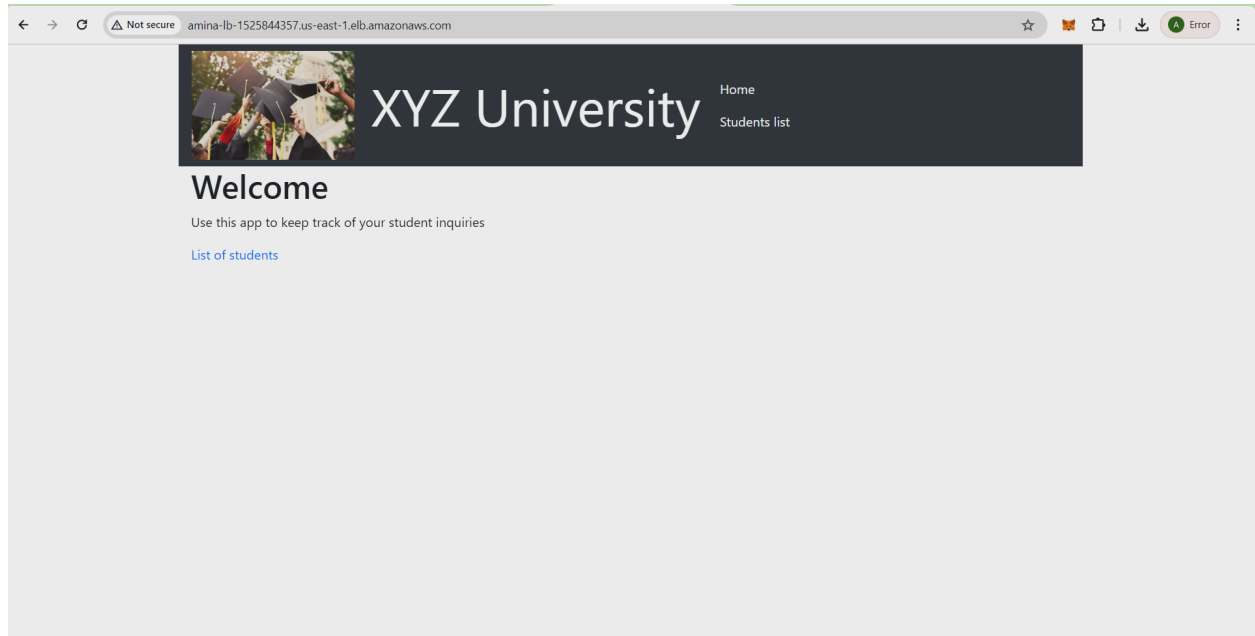
```
#!/bin/bash -xe
apt update -y
apt install nodejs unzip wget npm mysql-server -y
#wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCAP1-1-DEV/code.zip -P /home/ubuntu
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCAP1-1-79581/1-lab-capstone-project-1/code.zip -P /home/ubuntu
cd /home/ubuntu
unzip code.zip -x "resources/codebase_partner/node_modules/*"
cd resources/codebase_partner
npm install aws aws-sdk
mysql -u root -e "CREATE USER 'nodeapp' IDENTIFIED WITH mysql_native_password BY 'student12';"
mysql -u root -e "GRANT all privileges on *.* to 'nodeapp'@'%';"
mysql -u root -e "CREATE DATABASE STUDENTS;"
mysql -u root -e "USE STUDENTS; CREATE TABLE students(
    id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    address VARCHAR(255) NOT NULL,
    city VARCHAR(255) NOT NULL,
    state VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    phone VARCHAR(100) NOT NULL,
    PRIMARY KEY ( id ));"
sed -i 's/.*bind-address.*/bind-address = 0.0.0.0/' /etc/mysql/mysql.conf.d/mysqld.cnf
systemctl enable mysql
service mysql restart
export APP_DB_HOST=$(curl http://169.254.169.254/latest/meta-data/local-ipv4)
export APP_DB_USER=nodeapp
export APP_DB_PASSWORD=student12
export APP_DB_NAME=STUDENTS
export APP_PORT=80
npm start &
echo '#!/bin/bash -xe
cd /home/ubuntu/resources/codebase_partner
export APP_PORT=80
npm start' > /etc/rc.local
chmod +x /etc/rc.local
```

(Script can be found in the helper-scripts folder.)

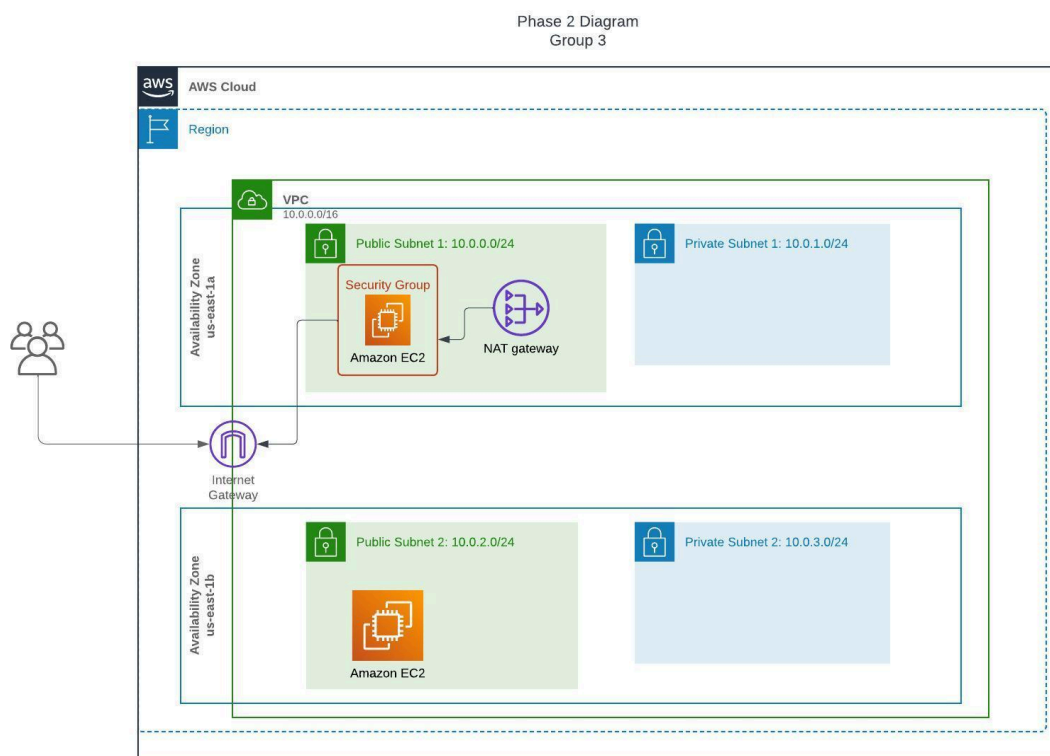
After that we launched the instance.

### Task 3:

Last task was to actually test this deployment and to ensure that we can successfully access and navigate through it (screenshot can be seen below).



### Architectural diagram after second phase:



- *Third phase:*

The aim of this phase is to make our application runnable on separate virtual machines.

#### **Task 1:**

In this task we modified the inbound rules in the previously created security group that is associated with the database (MySQL Aurora) in order for the VPC to be able to access the instances of the database.

#### **Task 2:**

Second task was about the creation of the Amazon Relational Database using MySQL engine. The database instance is called "STUDENTS" and it resides in the private subnets within our VPC in order to ensure that it is not publicly accessible and to ensure that it is more secure.

#### **Task 3:**

In this task we needed to establish the Cloud9 environment in order to run AWS Command Line Interface (CLI) commands. We used the t3.micro EC2 instance which is configured with the SSH - Secure Shell access for secure and reliable connection and also our Cloud 9 environment has access to the RDS MySQL database.

#### **Task 4:**

With the fourth task we provisioned AWS Secret Manager. We did that by running this script:

```
aws secretsmanager create-secret \
  --name Mydbsecret \
  --description "Database secret for web app" \
  --secret-string "{\"user\":\"nodeapp\",\"password\":\"password\",\"host\":\"students.carz3c0jo1i2.us-east-1.rds.amazonaws.com\",\"db\":\"STUDENTS\"}"
```

(Script can be found in the helper-scripts folder.)

#### **Task 5:**

In the following we made a new EC2 instance in the same way as the previous one, only for the user data we inserted and executed the following script:

```
#!/bin/bash -xe
apt update -y
apt install nodejs unzip wget npm mysql-client -y
#wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCAP1-1-DEV/code.zip -P /home/ubuntu
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCAP1-1-79581/1-lab-capstone-project-1/code.zip -P /home/ubuntu
cd /home/ubuntu
unzip code.zip -x "resources/codebase_partner/node_modules/*"
cd resources/codebase_partner
npm install aws aws-sdk
export APP_PORT=80
npm start &
echo '#!/bin/bash -xe
cd /home/ubuntu/resources/codebase_partner
export APP_PORT=80
npm start' > /etc/rc.local
chmod +x /etc/rc.local
```

(Script can be found in the helper-scripts folder.)

## Task 6:

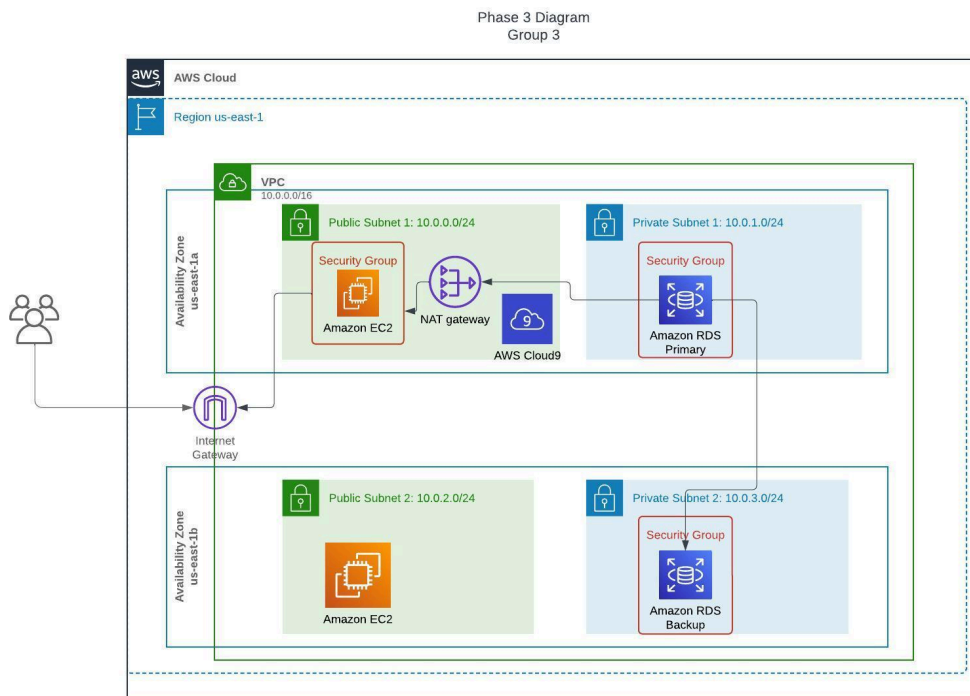
In this task we migrated the database from the original EC2 instance to the new one which we created in Cloud9. On the Cloud9 we run the following script:

```
mysqldump -h 10.0.0.73 -u nodeapp -p --databases STUDENTS > data.sql
mysql -h students.carz3c0jo1i2.us-east-1.rds.amazonaws.com -u nodeapp -p STUDENTS < data.sql
```

(Script can be found in the helper-scripts folder.)

With this we successfully migrated the database and made sure that everything works properly.

## Architectural diagram after third phase:



- *Fourth phase*

Phase four was about ensuring the high availability and scalability. Because of that we launched the Application Load Balancer.

### Task 1:

We created an Application Load Balancer and configured its name, schema, AZs (two availability zones: *us-east-1a* and *us-east-1b*), security settings and routing.

### Task 2:

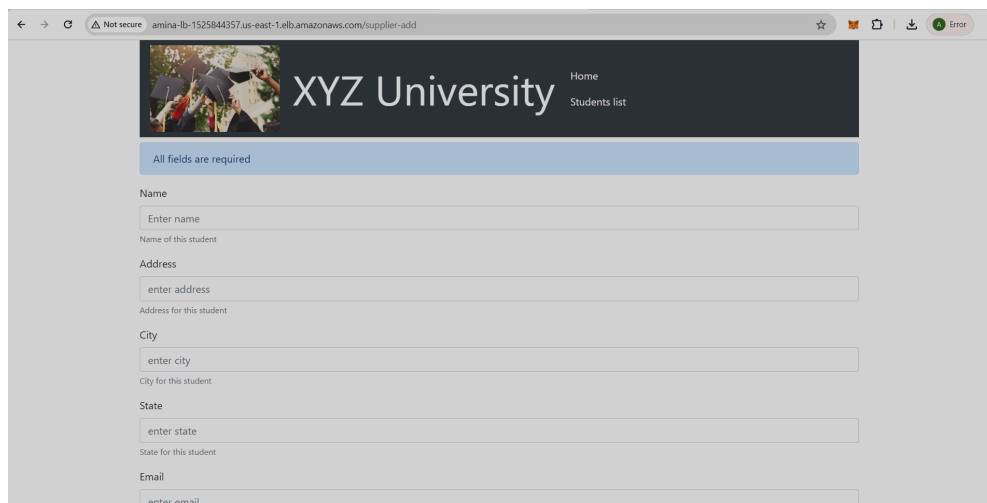
After creating LB, we set up the EC2 instance by creating a Launch Template, selecting the appropriate AMI and instance type, and enabling auto assigned IPv4 addresses for public access. Then we configured the auto scaling group by selecting VPC in subnets and attaching LB.

This setup ensures that our web application is publicly accessible and can dynamically scale to handle different loads, providing high availability and scalability.

### Task 3:

We tested the application by performing some basic CRUD operations in order to see if everything works correctly and fine.

Adding the student:



The screenshot shows a web browser window with the URL `amina-lb-1525844357.us-east-1.elb.amazonaws.com/supplier-add`. The page header features the XYZ University logo and navigation links for Home and Students list. A blue message bar states "All fields are required". The form contains the following fields:

- Name**: A text input field with the placeholder "Enter name".
- Name of this student**: A text input field with the placeholder "enter address".
- Address**: A text input field with the placeholder "enter address".
- City**: A text input field with the placeholder "enter city".
- State**: A text input field with the placeholder "enter state".
- Email**: A text input field with the placeholder "enter email".

Not secure amina-lb-1525844357.us-east-1.elb.amazonaws.com/supplier-add

Name  
Enter name  
Name of this student

Address  
enter address  
Address for this student

City  
enter city  
City for this student

State  
enter state  
State for this student

Email  
enter email  
Email for this student

Phone  
enter phone  
Phone number for this student

Submit

Getting the student's data:

Not secure amina-lb-1525844357.us-east-1.elb.amazonaws.com/students

XYZ University Home  
Students list

### All students

Name	Address	City	State	Email	Phone
Amina Meric	Stupska 19 D&#x2F;il	Sarajevo	Federation	amina.meric@stu.ibu.edu.ba	061000222 <a href="#">edit</a>
Selma Imsirovic	Stupska 19 D&#x2F;il	Tuzla	Federation	selmai@gmail.com	061000222 <a href="#">edit</a>

[Add a new student](#)

## Task 4:

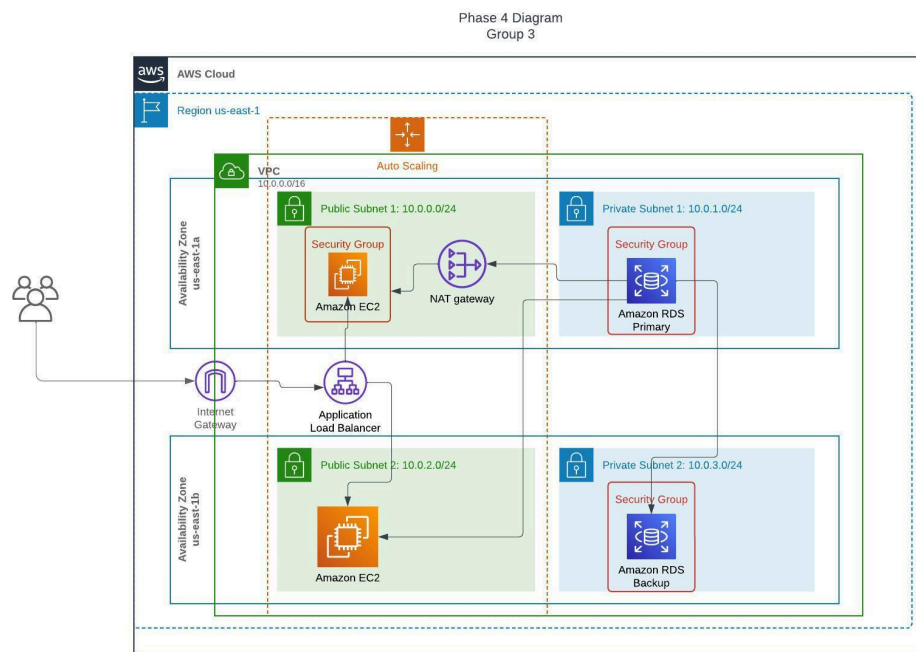
At the end we executed the following script on Cloud9 against the load balancer:

```
npm install -g loadtest
loadtest --rps 1000 -c 500 -k http://amina-lb-1525844357.us-east-1.elb.amazonaws.com/students
```

(Script can be found in the helper-scripts folder.)

By this we successfully completed this project that covers various aspects of architecture design, cost estimation, web application deployment, network configuration, security measure setup, high availability and scalability, and access permission management.

### Final Architectural diagram:



The goal and sour focus was to obtain a comprehensive understanding and practical experience with AWS services.