ITESO, Universidad Jesuita de Guadalajara

Proyecto Final: Aplicación Web Escalable y con Alta Disponibilidad

Diseño de Sistemas Escalables

Profesor: Jorge Alejandro

Carlos Esteban Calzada Diaz [738803]

Juan Bernardo Orozco Quirarte [745172]

Fecha de entrega: xx de mayo del 2025

Visión general

El presente proyecto forma parte del laboratorio del curso AWS Academy – Cloud Web Application Builder, y tiene como propósito desarrollar una aplicación web escalable y altamente disponible, aplicando los principios fundamentales del diseño de sistemas modernos en la nube. Esta actividad representa una experiencia práctica alineada con los desafíos reales que enfrentan las organizaciones al construir soluciones resilientes, seguras y eficientes.

La problemática a resolver parte de un escenario común en instituciones educativas: una universidad experimenta interrupciones y bajo rendimiento en su sistema de gestión de registros estudiantiles durante el periodo de admisiones, debido a la alta demanda de usuarios. Ante esta situación, se nos encomienda construir un prototipo funcional de la aplicación que responda eficazmente a estos desafíos mediante el uso de servicios de AWS, dentro de un entorno con limitaciones controladas.

El proyecto tiene como objetivos principales:

- Adquirir experiencia práctica en la arquitectura de soluciones escalables en la nube.
- Aplicar principios del AWS Well-Architected Framework en el diseño y despliegue del sistema.
- Desarrollar habilidades clave como el pensamiento crítico, la resolución de problemas y la comunicación efectiva en un contexto técnico.
- Comprender el ciclo de vida del desarrollo de software, desde la planificación inicial hasta la implementación funcional.

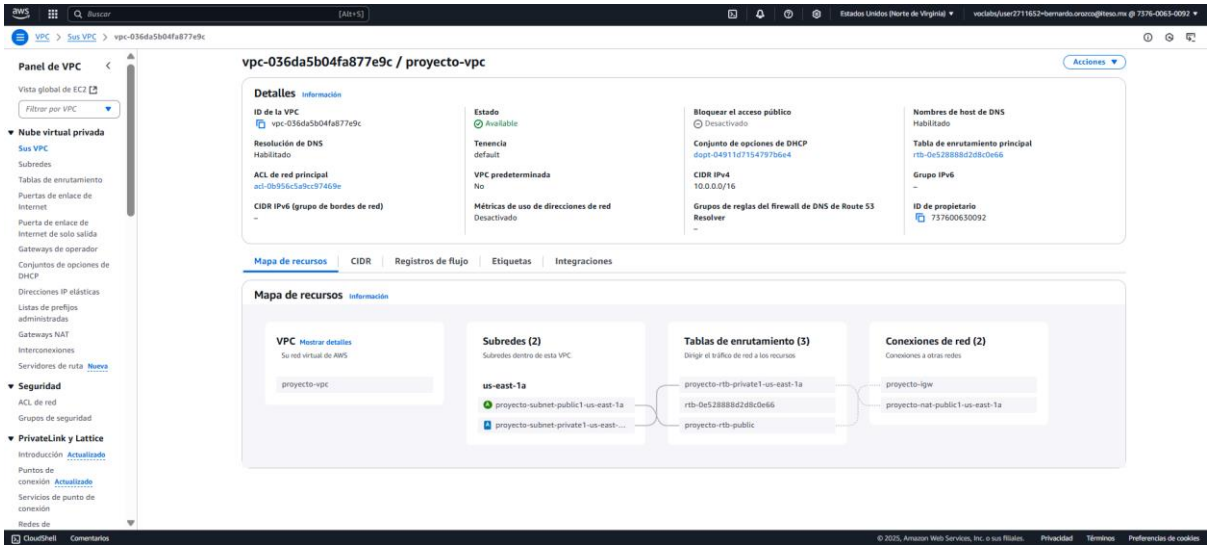
A lo largo del desarrollo se establecen supuestos clave, como el uso de una única región de AWS, el despliegue en servidores Ubuntu, la utilización de servicios limitados por el entorno de laboratorio y el acceso público sin autenticación, entre otros. Estos factores permiten enfocar el proyecto como una prueba de concepto (PoC), que puede ser expandida en el futuro hacia una solución más robusta.

El trabajo se estructura en fases progresivas: desde la planificación y estimación de costos, pasando por la creación de una versión básica funcional de la aplicación, hasta alcanzar una arquitectura desacoplada, escalable y con alta disponibilidad.

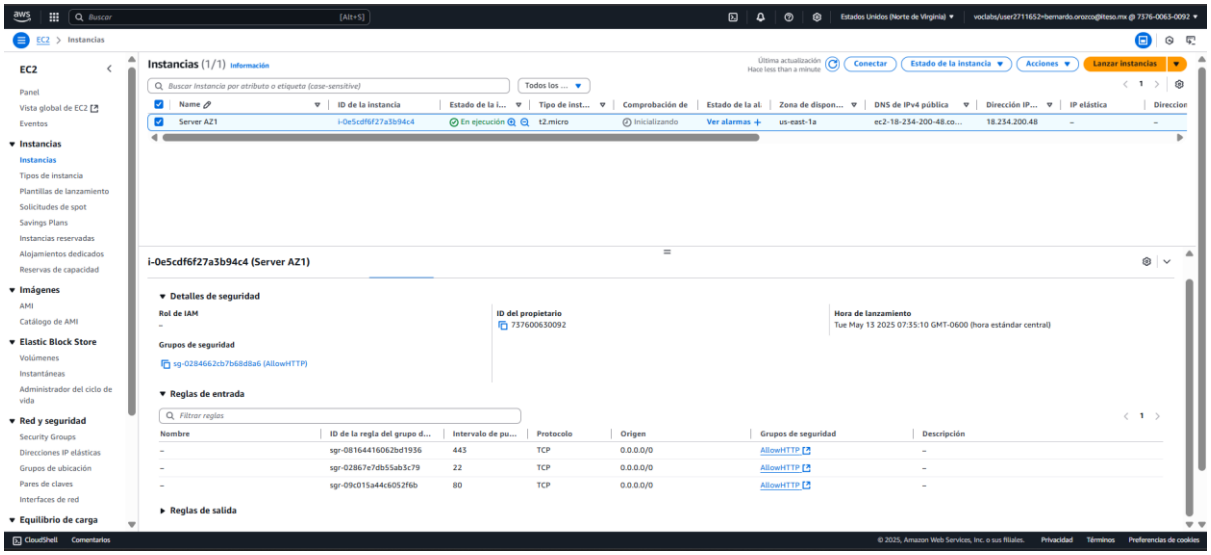
Este proyecto no solo consolida conocimientos técnicos sobre la nube y AWS, sino que también fortalece la capacidad para diseñar soluciones reales que respondan a necesidades específicas de rendimiento, seguridad y eficiencia operativa.

Solución general

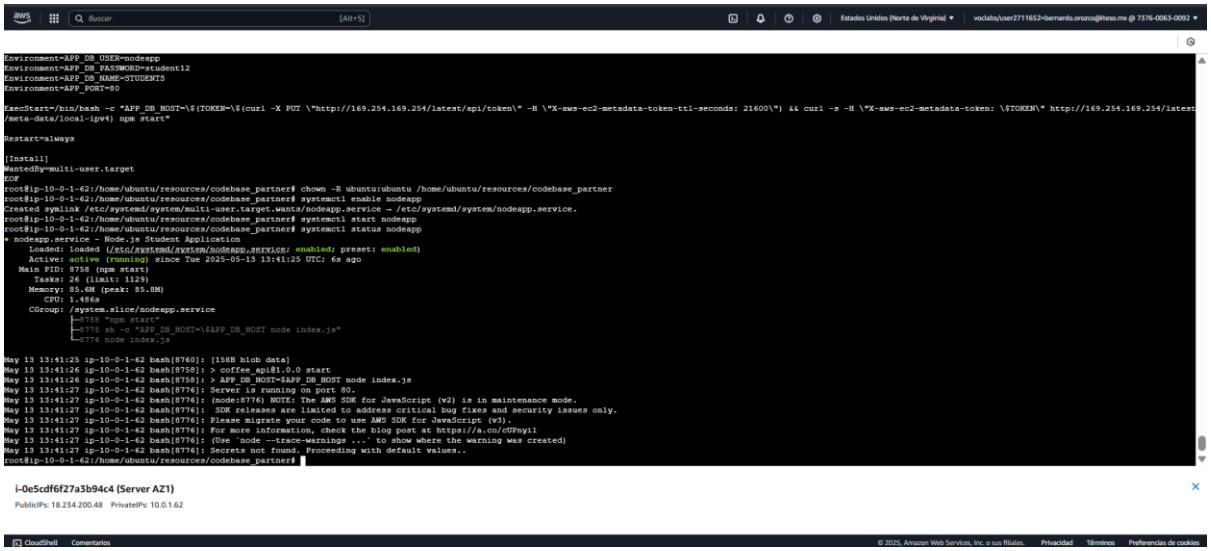
1. Se implementó una VPC con una subred pública y una subred privada. Se configuró un Internet Gateway (IGW) y una tabla de enrutamiento.
    i. La subred pública se conecta directamente al IGW para tener acceso a Internet.
    ii. La subred privada se enruta mediante un NAT Gateway para permitir conectividad saliente controlada.



2. Se lanzó una instancia EC2 (Ubuntu, tipo t2.micro) dentro de la VPC llamada “proyecto”. Se configuró un grupo de seguridad que permite tráfico entrante para conectarse a la aplicación AllowHttp.



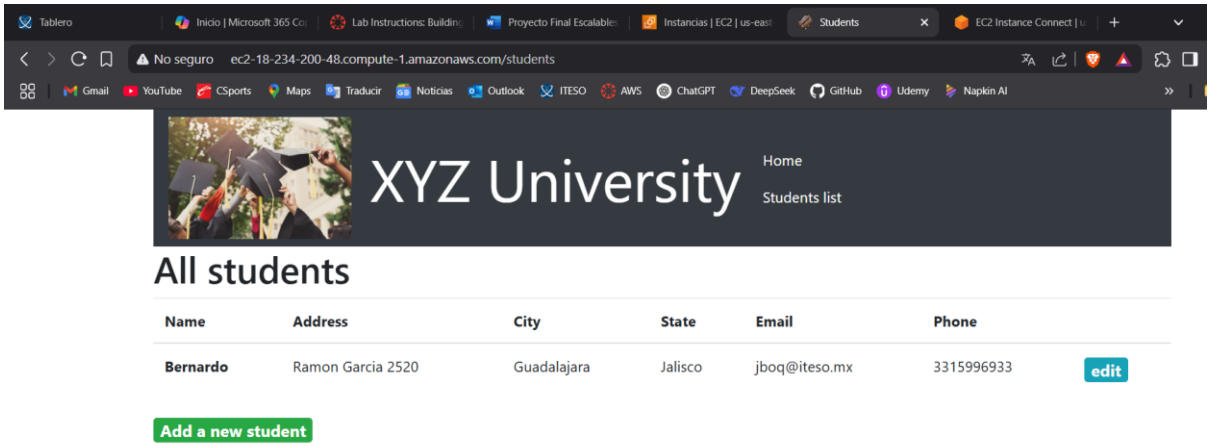
3. Se accedió a la instancia mediante el cliente basado en navegador de EC2. Desde ahí se ejecutó un script (UserdataScript-phase-2.sh) que instala los requerimientos de la aplicación web.



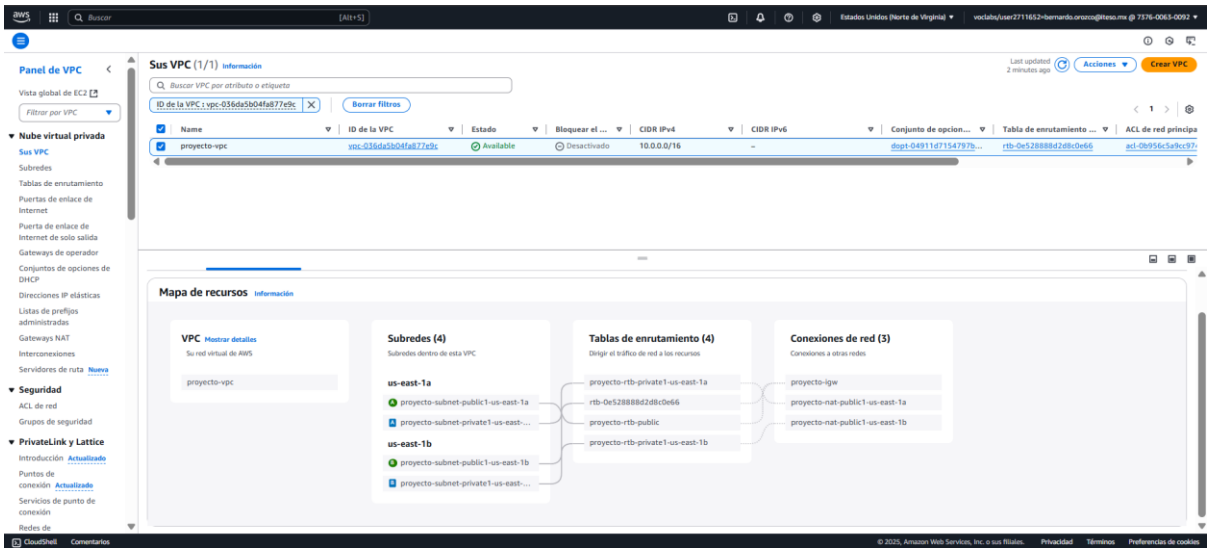

```
#!/bin/bash -xe

# Update and install packages
apt update -y
apt install nodejs unzip wget npm mysql-server -y

# Download and extract code
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCAP1-1-91571/1-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

# Configure MySQL
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 ));"

# Configure MySQL to accept connections from any IP
sed -i 's/.*bind-address.*/bind-address = 0.0.0.0/' /etc/mysql/mysql.conf.d/mysqld.cnf
systemctl enable mysql
service mysql restart
```

```
# Create systemd service file
cat > /etc/systemd/system/nodeapp.service << EOF
[Unit]
Description=Node.js Student Application
After=network.target mysql.service

[Service]
Type=simple
User=root
WorkingDirectory=/home/ubuntu/resources/codebase_partner

Environment=APP_DB_USER=nodeapp
Environment=APP_DB_PASSWORD=student12
Environment=APP_DB_NAME=STUDENTS
Environment=APP_PORT=80

ExecStart=/bin/bash -c "APP_DB_HOST=\$(TOKEN=\$(curl -X PUT
\"http://169.254.169.254/latest/api/token\" -H \"X-aws-ec2-metadata-token-ttl-seconds: 21600\") && curl -
s -H \"X-aws-ec2-metadata-token: \$TOKEN\" http://169.254.169.254/latest/meta-data/local-ipv4) npm
start"

Restart=always

[Install]
WantedBy=multi-user.target
EOF

# Set correct permissions
chown -R ubuntu:ubuntu /home/ubuntu/resources/codebase_partner

# Enable and start the service
systemctl enable nodeapp
systemctl start nodeapp

# Verify the service is running
systemctl status nodeapp
```

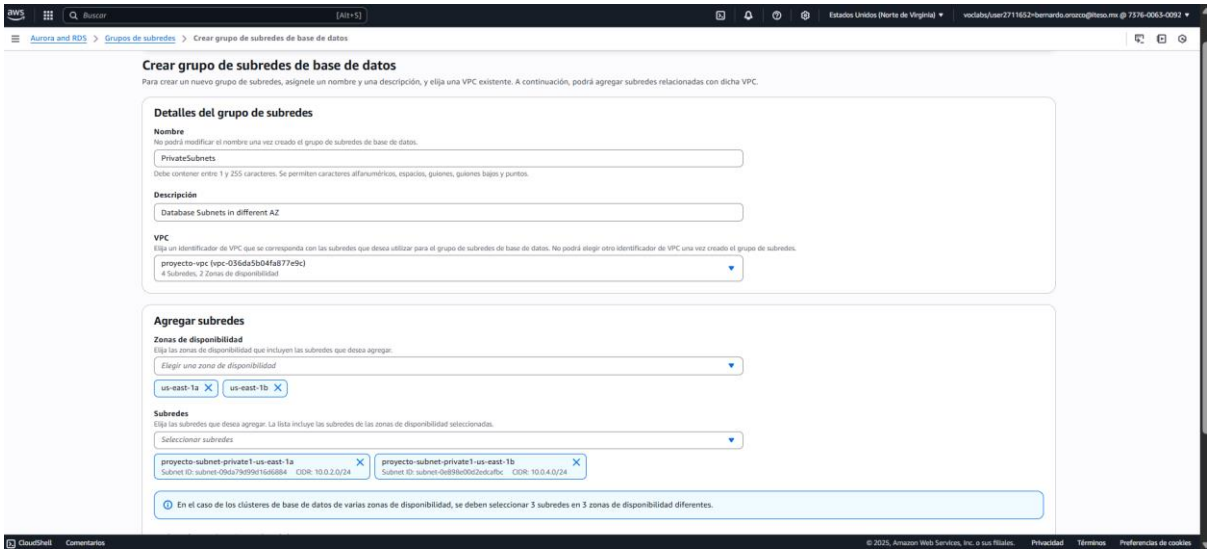
4. Al finalizar la instalación, la aplicación quedó accesible mediante la IP pública o DNS de la instancia, permitiendo operaciones CRUD desde el navegador.



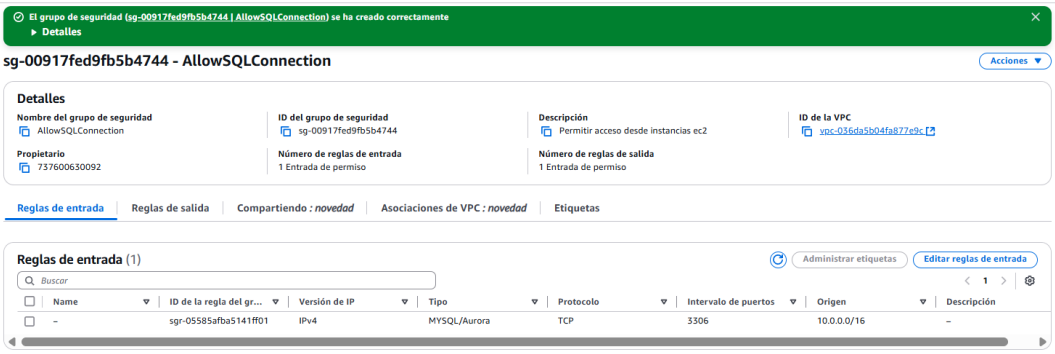
5. Se agregó una segunda zona de disponibilidad (AZ) con otra subred pública y privada. Se configuró un segundo NAT Gateway y se actualizaron las tablas de enrutamiento para que ambas zonas pudieran operar de forma similar.



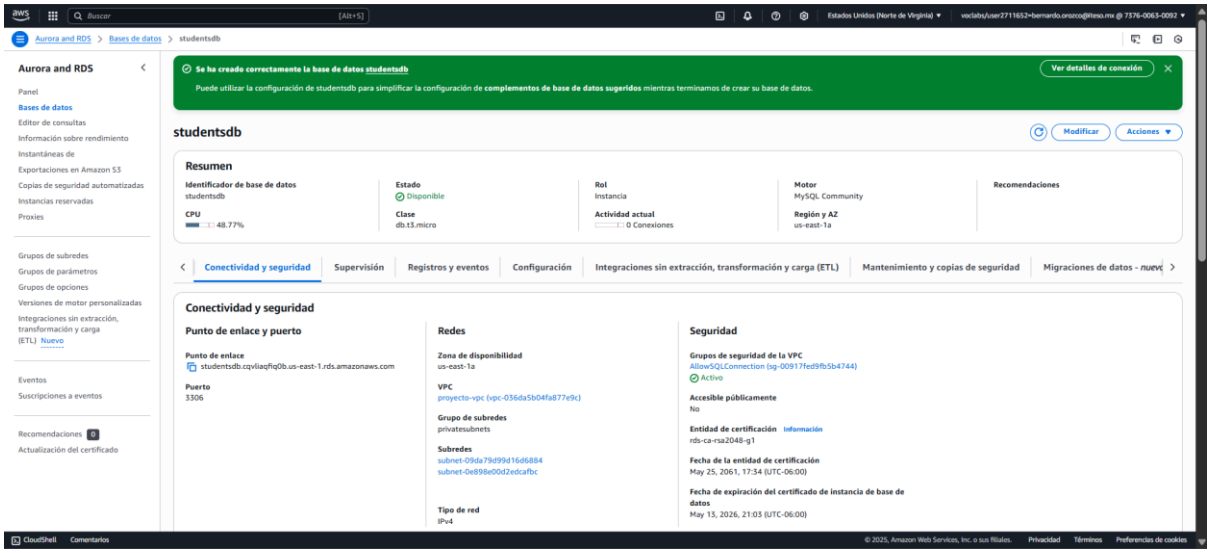
6. Se creó un grupo de subredes (subnet group) para la base de datos, seleccionando las subredes privadas de ambas zonas de disponibilidad.



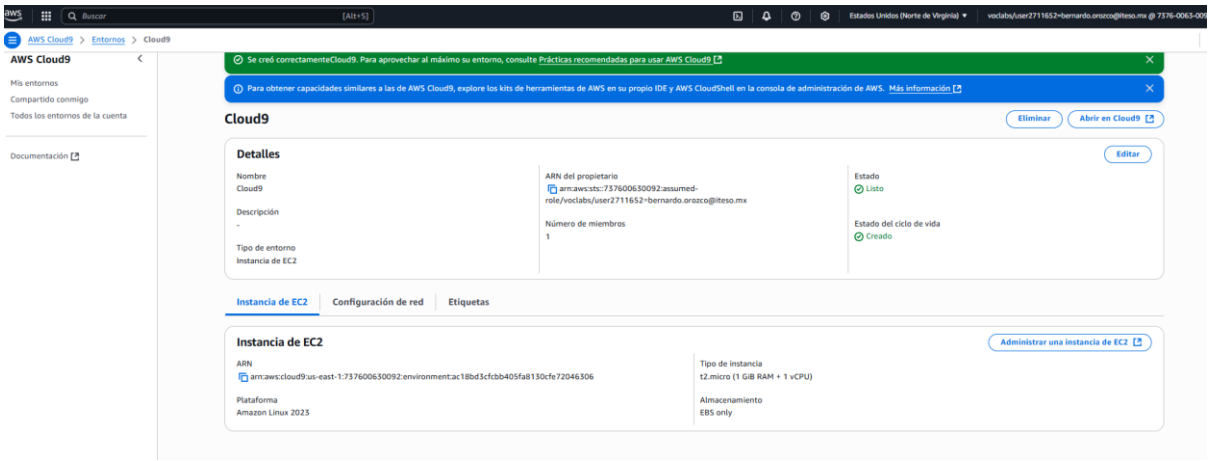
7. Se creó un nuevo grupo de seguridad AllowSQLConnection para asociarlo a la base de datos, permitiendo únicamente conexiones desde las instancias EC2 de la VPC.



8. Se desplegó una base de datos MySQL en Amazon RDS, con una instancia t3.micro, almacenamiento SSD (20 GiB) y acceso desde una sola zona de disponibilidad, considerando las limitaciones del laboratorio.
    a. Usuario: admin
    b. Contraseña: qpalzmwoskxn
    c. Permisos de entrada puerto 3306 desde el grupo de seguridad AllowSQLConnection.

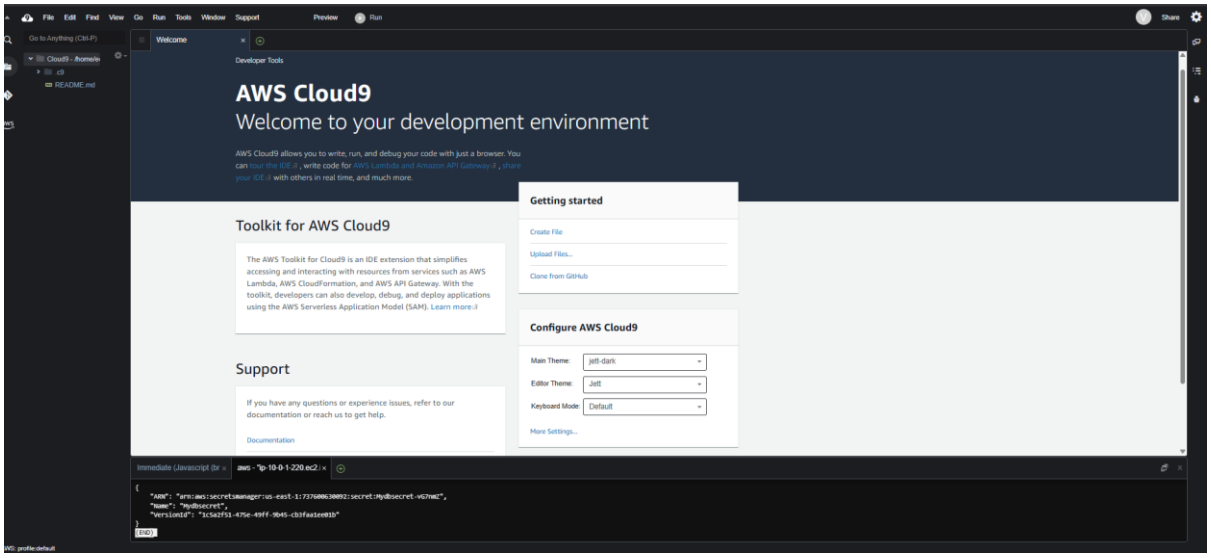


9. Se creó un entorno en AWS Cloud9 con Amazon Linux. Se configuró para ejecutar dentro de una subred pública, permitiendo acceso a Internet.

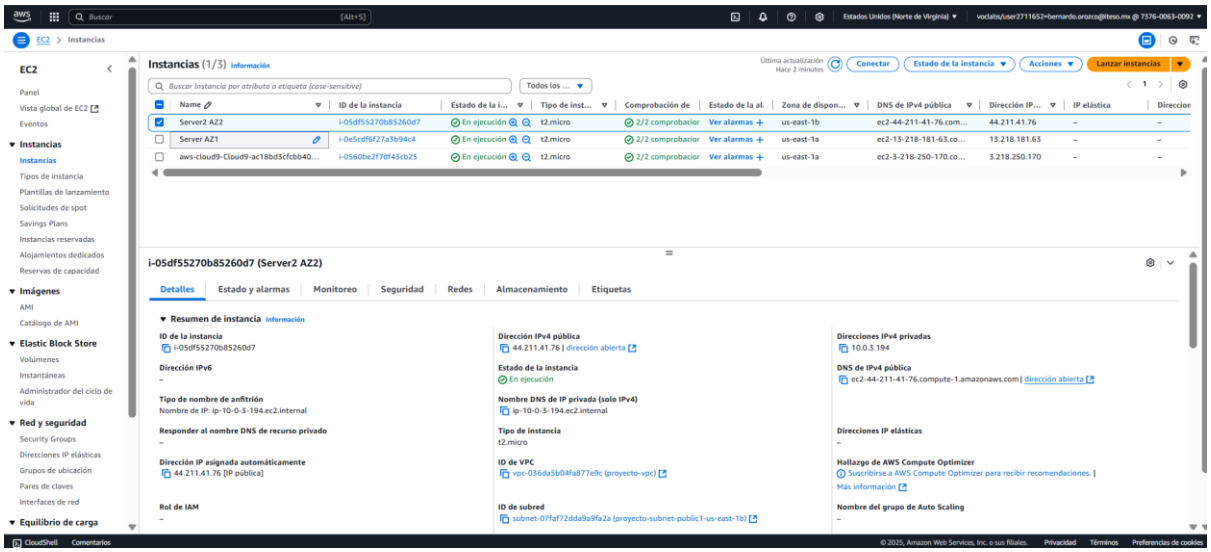


10. Desde Cloud9 se creó un secreto en AWS Secrets Manager que almacena las credenciales y datos de conexión para la base de datos RDS.

```
aws secretsmanager create-secret --name Mydbsecret --description "Database secret for web app" --secret-string "{\"user\":\"admin\",\"password\":\"qpalzmwoskxn\",\"host\":\"studentsdb.cqvliaqfiq0b.us-east-1.rds.amazonaws.com\",\"db\":\"STUDENTS\"}"
```



11. Se lanzó otra instancia EC2 (Ubuntu, t2.micro) en una subred pública. Se configuró con un grupo de seguridad y se le asignó un rol (LabInstanceProfile) con permisos para acceder al secreto de RDS.

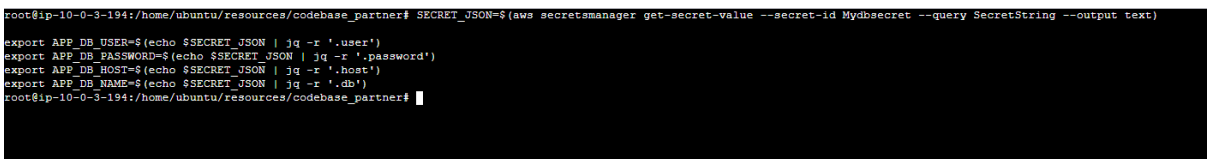


12. Se instaló la nueva aplicación web en esta instancia utilizando el script UserdataScript-phase-3.sh, que incluye la recuperación de las credenciales desde Secrets Manager.

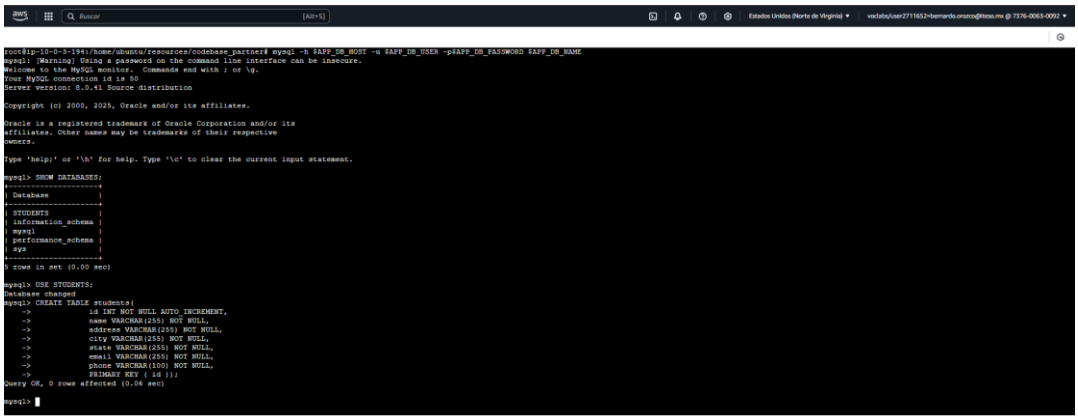

```
#!/bin/bash -xe
apt update -y
apt install nodejs unzip wget npm mysql-client -y
#wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCAP1-1-DEV/code.zip -P /home/ubuntu
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCAP1-1-91571/1-lab-capstone-project-1/code.zip -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
```

13. Se cargaron las credenciales del secreto como variables de entorno en la instancia para que la aplicación pudiera conectarse a la base de datos.

```
SECRET_JSON=$(aws secretsmanager get-secret-value --secret-id Mydbsecret --query SecretString --output text)

# Exporta las variables de entorno usando jq para extraer los valores del JSON
export APP_DB_USER=$(echo $SECRET_JSON | jq -r '.user')
export APP_DB_PASSWORD=$(echo $SECRET_JSON | jq -r '.password')
export APP_DB_HOST=$(echo $SECRET_JSON | jq -r '.host')
export APP_DB_NAME=$(echo $SECRET_JSON | jq -r '.db')
```

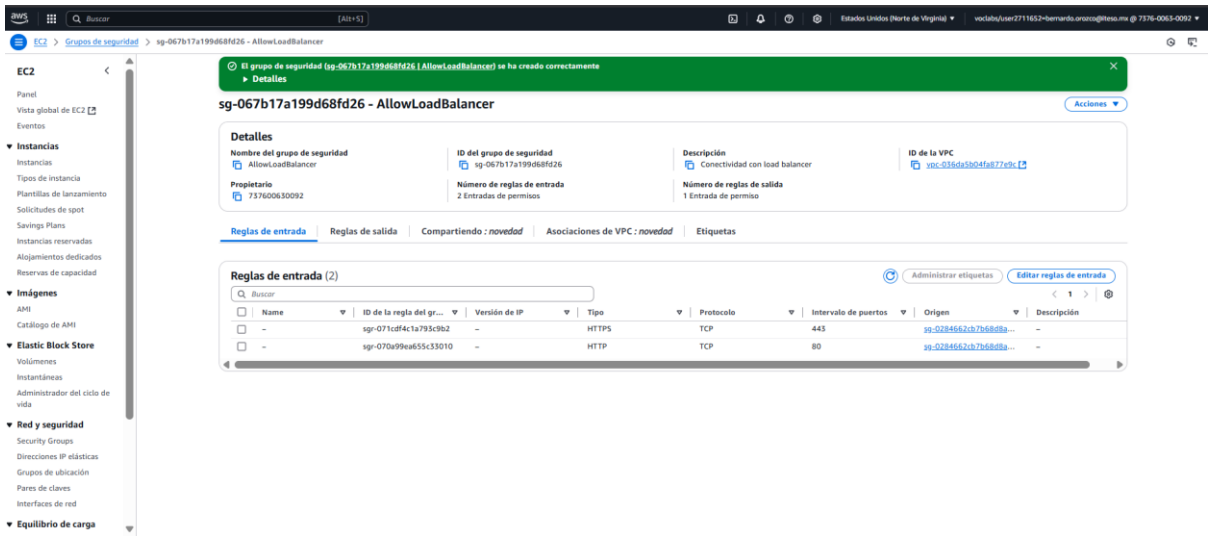


14. Debido a restricciones del entorno de laboratorio, se conectó manualmente a la base de datos y se creó la tabla students directamente con una instrucción SQL.

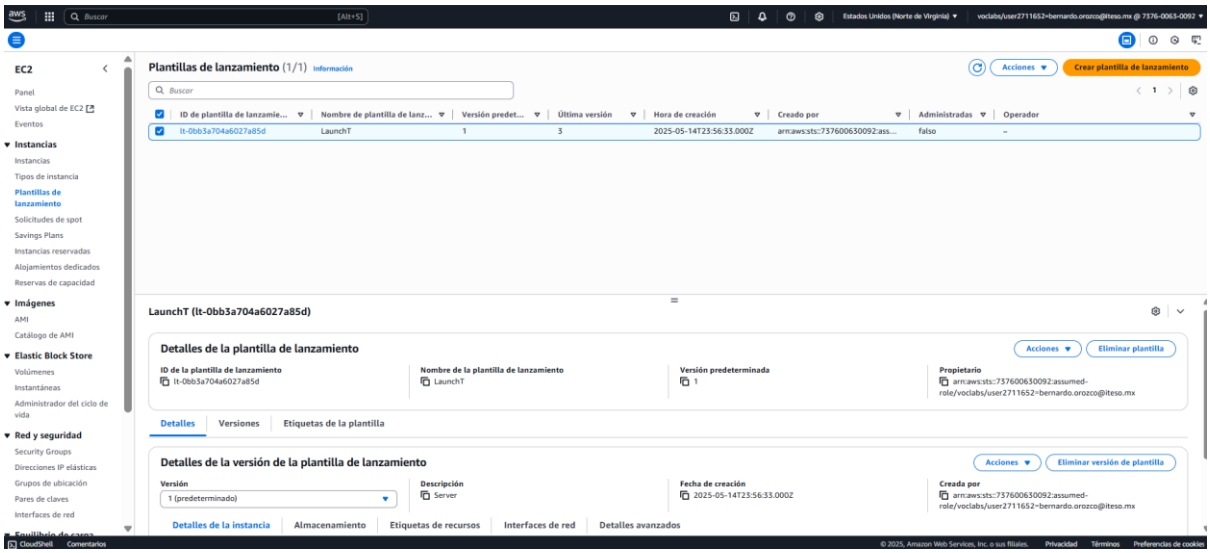



```
mysql -h $APP_DB_HOST -u $APP_DB_USER -p$APP_DB_PASSWORD $APP_DB_NAME

USE STUDENTS;
-- Creamos la tabla presente en UserdataScript-phase-2.sh
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 ));
```

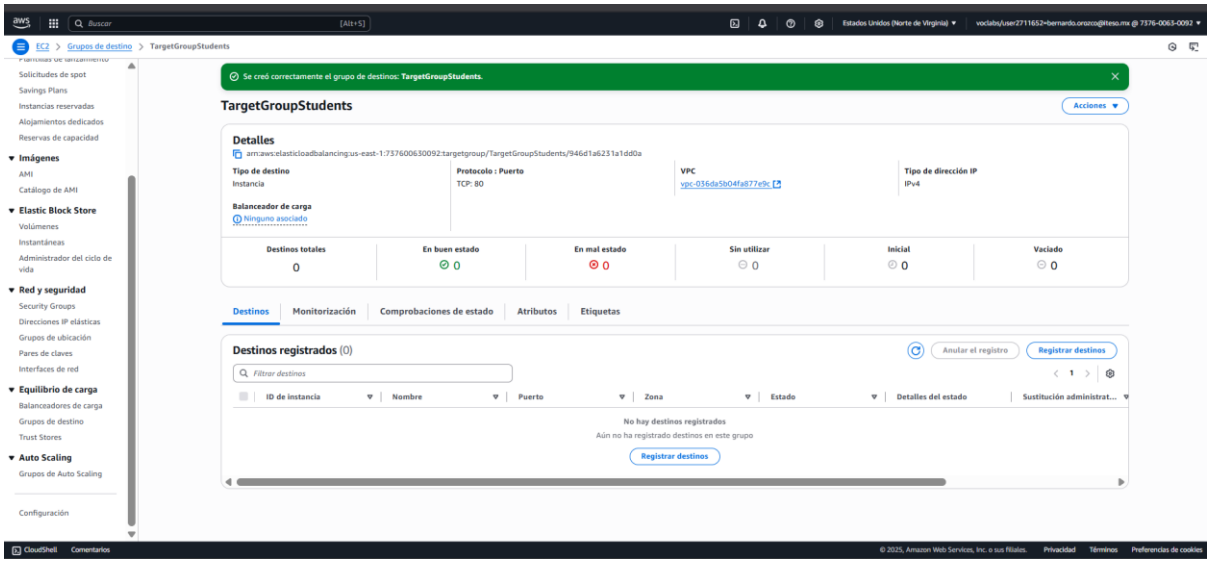
15. Se creó un grupo de seguridad llamado AllowLoadBalancer, que permite tráfico HTTP proveniente del grupo de seguridad AllowHttp, el cual será asociado al balanceador de carga.



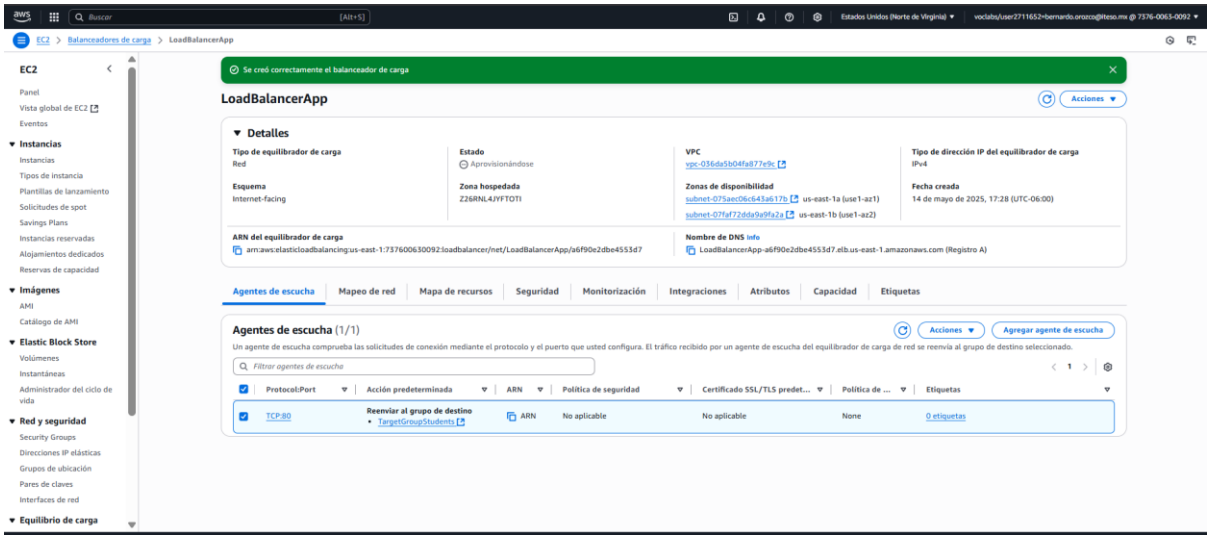
16. Se generó un Launch Template basado en una instancia funcional (llamada Server), removiendo el grupo AllowHttp y asociando AllowLoadBalancer.



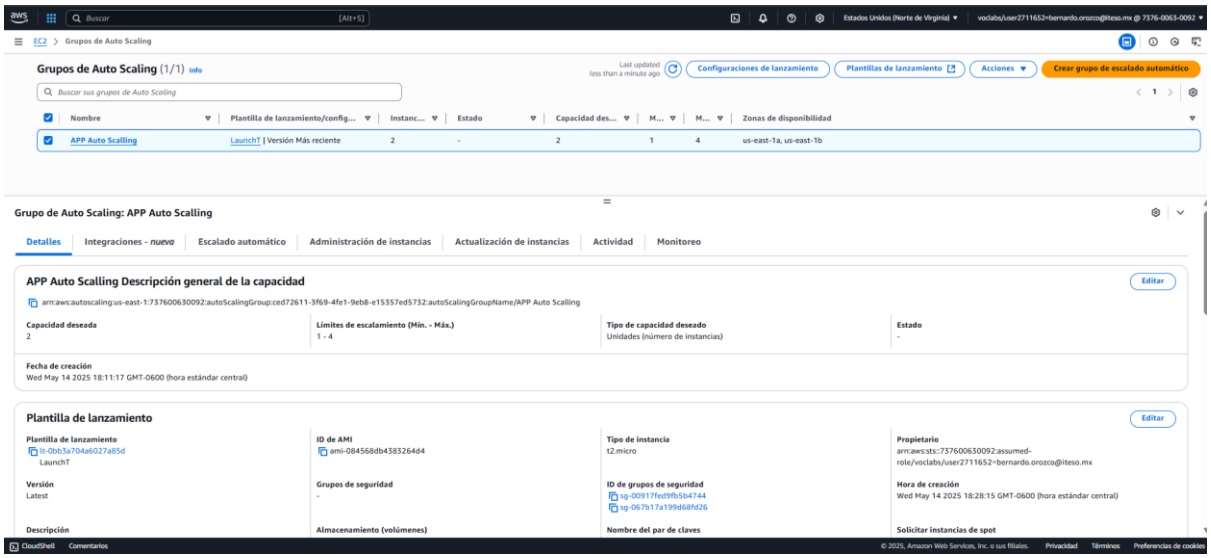
17. Se creó un Target Group para EC2 en la misma VPC, al cual se asociarán las instancias que estén detrás del balanceador de carga.

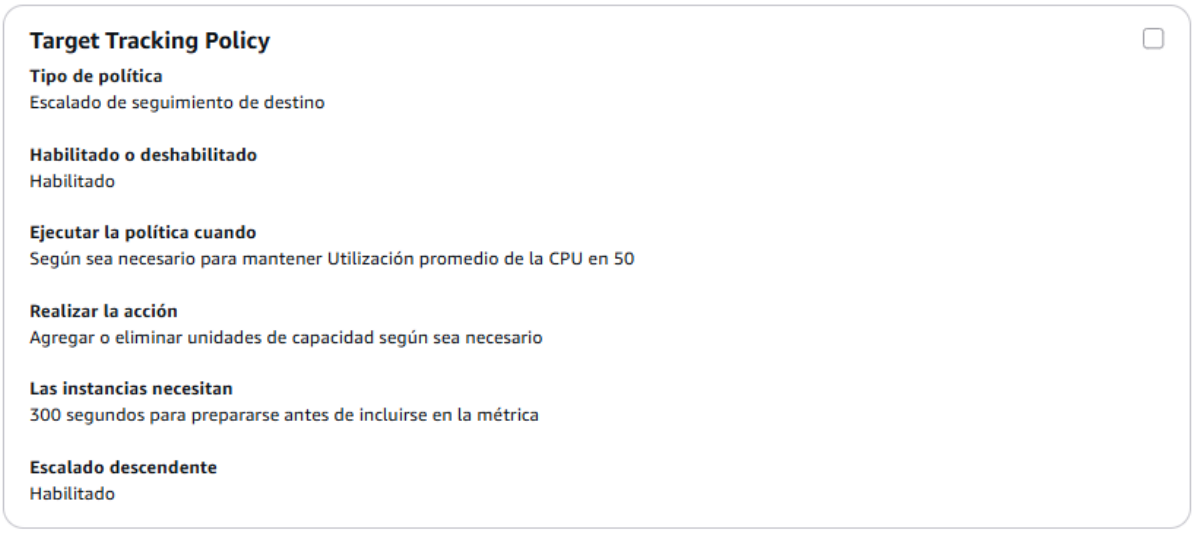


18. Se configuró un Application Load Balancer (ALB), asociado al grupo de seguridad AllowHttp, al Target Group y a las subredes públicas de ambas zonas de disponibilidad.



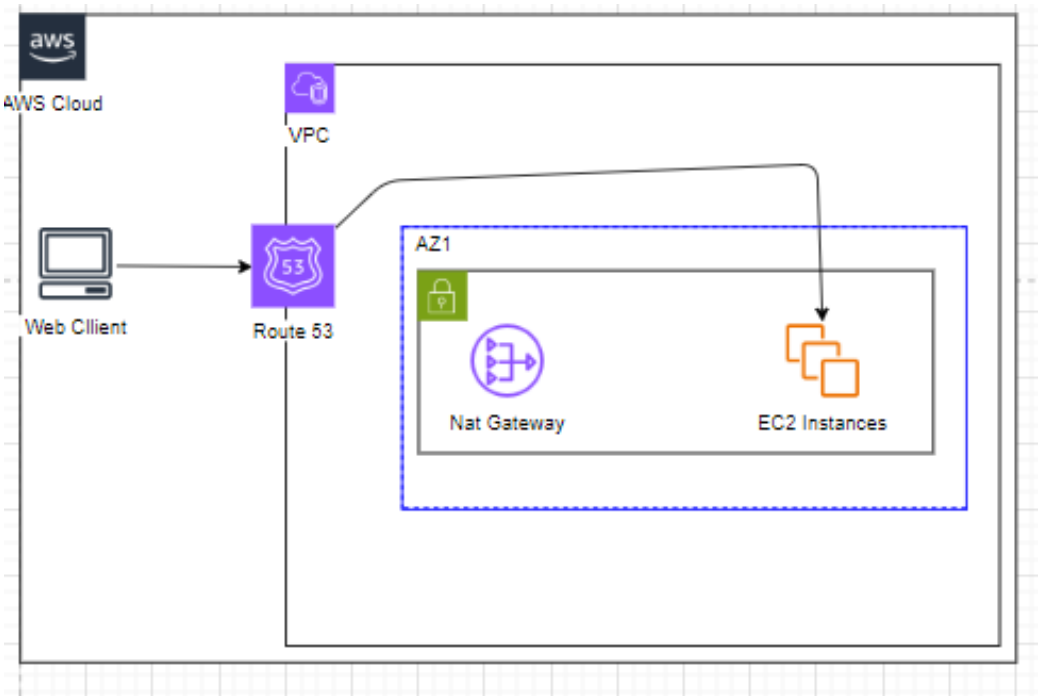
19. Se creó un grupo de Auto Scaling que escala automáticamente según métricas de CloudWatch, utilizando la plantilla de lanzamiento y conectado al ALB.



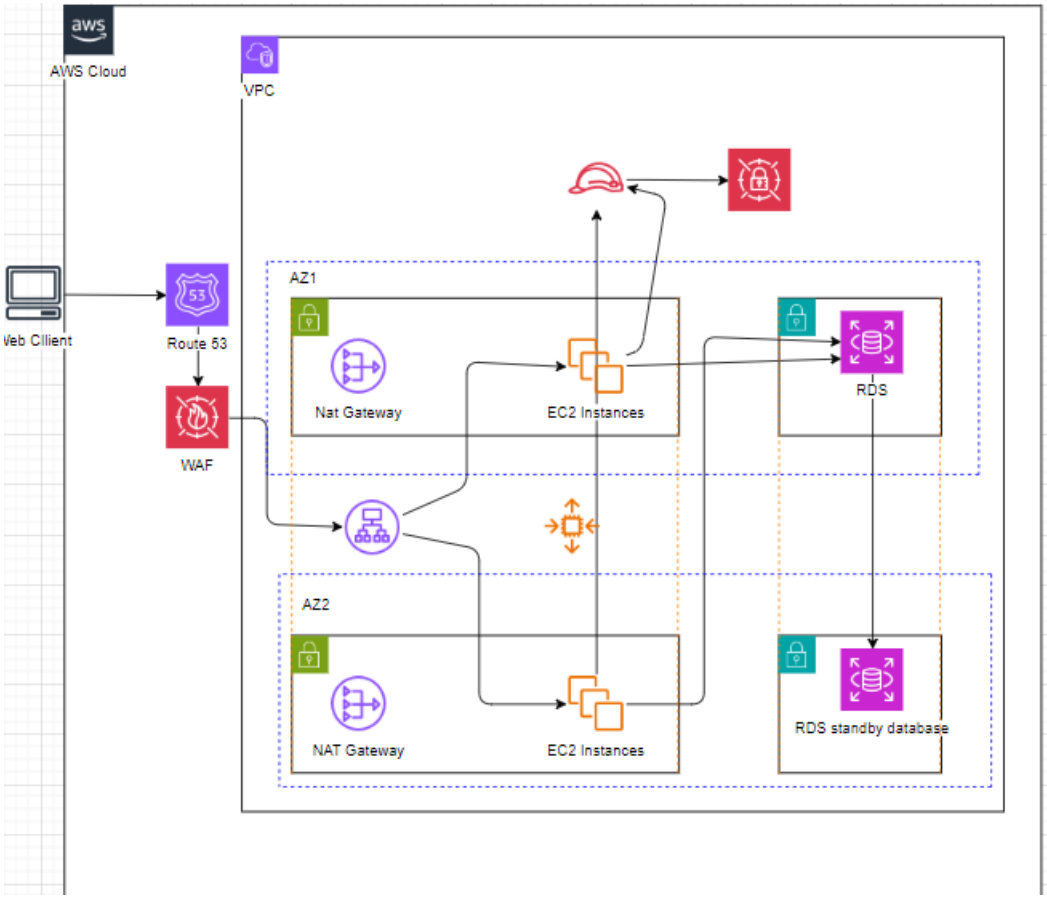


Diagramade arquitectura

Arquitectura Acoplada

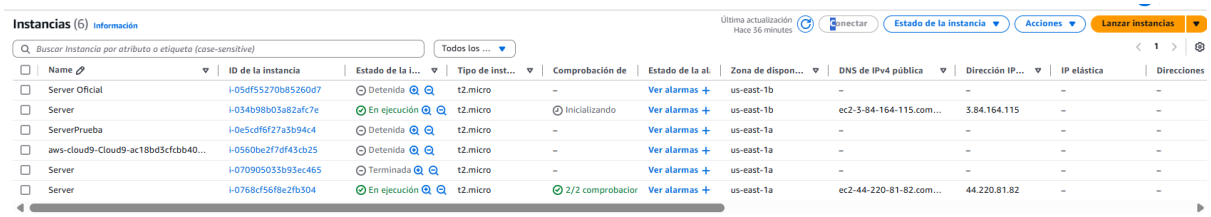


Arquitectura Desacoplada

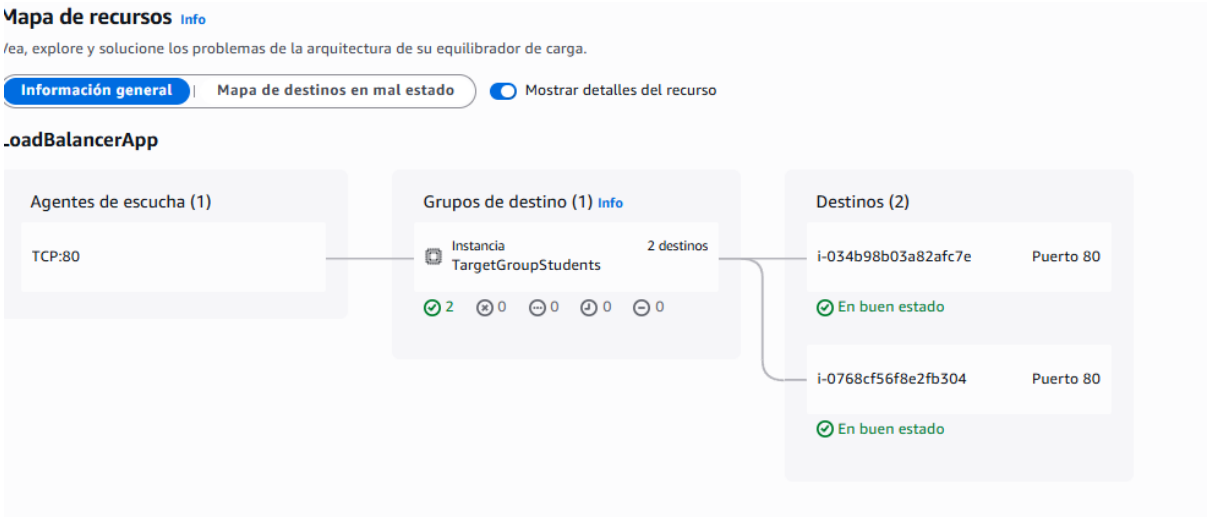


Demo de aplicación

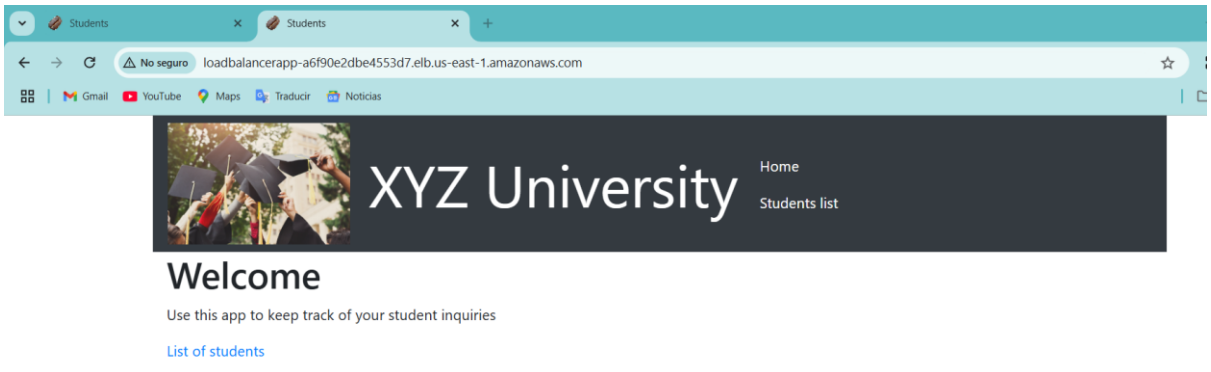
Tenemos varias instancias, las que son de tipo o nombre “Server” son aquellas que levanta el grupo de auto escalado de acuerdo con métricas de CloudWatch, específicamente el uso del CPU. Podemos ver que de 3 que hay, solamente dos están corriendo, esta sucede porque se configuraron dos servidores como el numero deseado de instancias encendidas, al terminar una otra se levanta automáticamente.



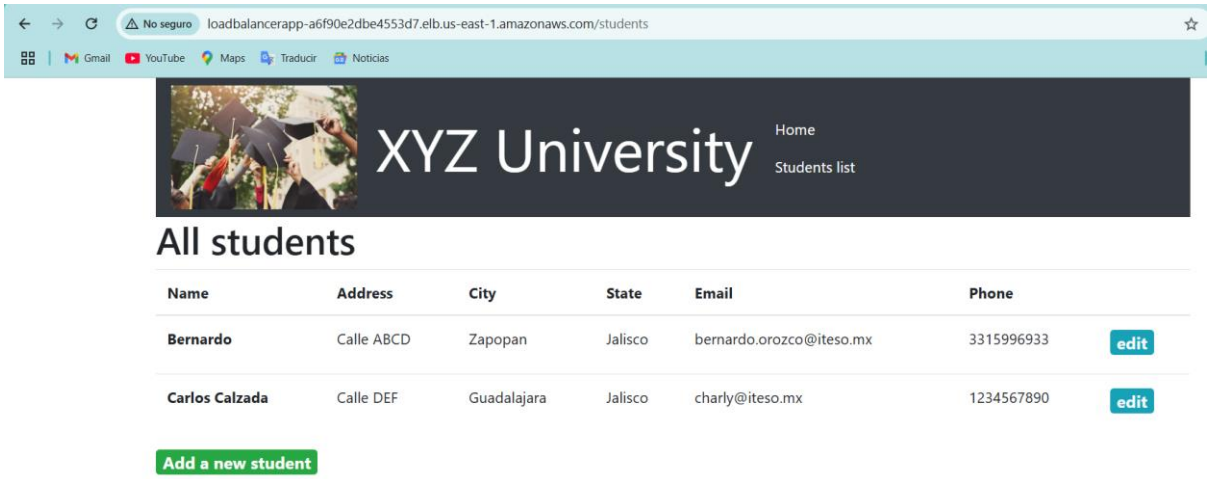
Mapa de recursos de ELB



Accedemos al DNS del ELB



Registramos estudiantes



Podemos editar, eliminar y agregar mas estudiantes.

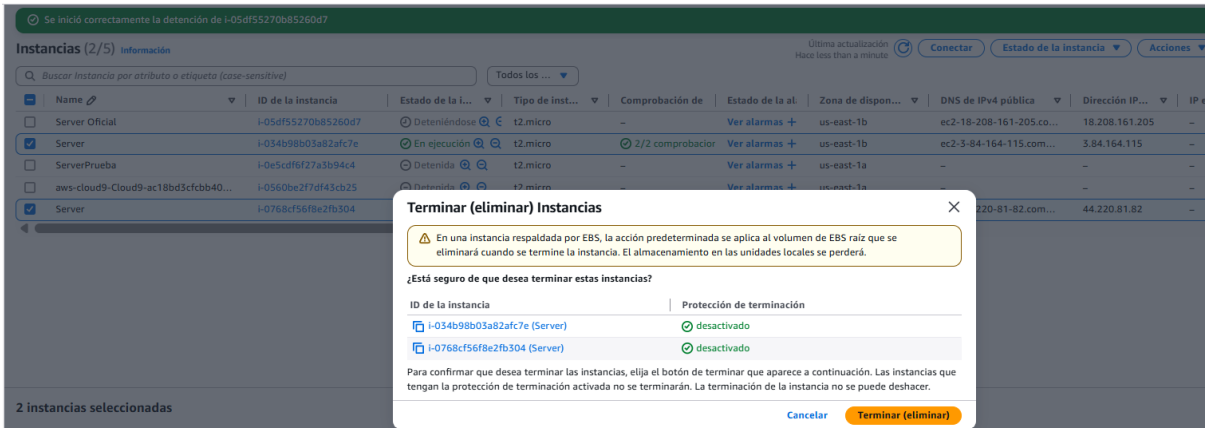
La prueba de que todo se guarda en la base de datos, nos conectaremos a nuestra instancia de mysql RDS por medio Server Oficial (el segundo que implementamos y en el cual basamos la plantilla de lanzamiento ya que este tiene permitido el tráfico entrante) con el siguiente comando.

```
mysql -h $APP_DB_HOST -u $APP_DB_USER -p$APP_DB_PASSWORD $APP_DB_NAME
```

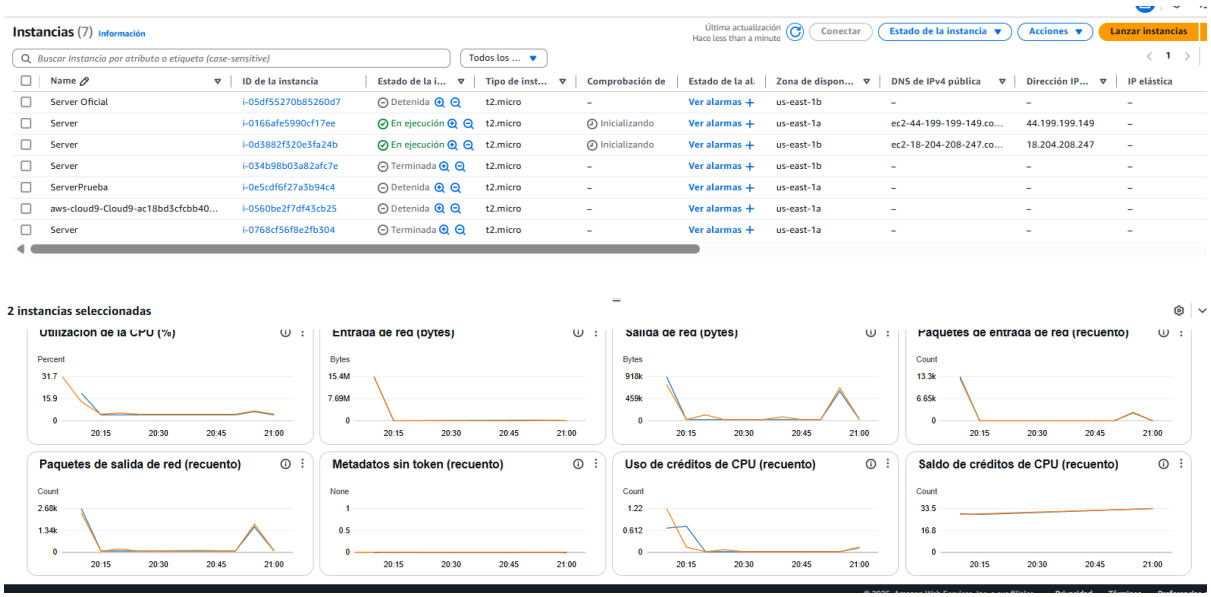
```
mysql> USE STUDENTS;
Database changed
mysql> SELECT * FROM
Display all 773 possibilities? (y or n)
mysql> SELECT * FROM students;
+----+----------------+------------+-------------+---------+-------------------------+------------+
| id | name           | address    | city        | state   | email                   | phone      |
+----+----------------+------------+-------------+---------+-------------------------+------------+
|  2 | Bernardo       | Calle ABCD | Zapopan     | Jalisco | bernardo.orozco@iteso.mx | 3315996933 |
|  3 | Carlos Calzada | Calle DEF  | Guadalajara | Jalisco | charly@iteso.mx         | 1234567890 |
+----+----------------+------------+-------------+---------+-------------------------+------------+
2 rows in set (0.00 sec)
```

El registro fue exitoso.

Vamos a terminar nuestros servers para verificar el funcionamiento del AutoScalling Group.



Automaticamente dos instancias se inicializan. Haremos una prueba de carga a ambas para comparar el uso del load balancer mandando miles de solicitudes al load balancer.



Comparativa de métricas de ambas instancias después de hacer la prueba de carga

Lecciones aprendidas

Durante el desarrollo de este proyecto aprendimos muchísimo, no solo sobre los servicios de AWS, sino también sobre cómo diseñar una arquitectura real siguiendo buenas prácticas y una de las cosas más importantes que entendimos fue la importancia de planear la red antes de lanzar cualquier recurso.

Al principio parecía que las VPC y subredes eran muy simples, pero una vez que separamos la base de datos en una subred privada y la aplicación en una pública en dos zonas de disponibilidad, todo tuvo sentido.

También aprendimos a pensar en seguridad desde el inicio, por ejemplo, usar Secrets Manager para no tener contraseñas en el código, y configurar correctamente los Security Groups para limitar solo el tráfico necesario.

Otro aprendizaje clave fue cómo automatizar el escalado con Auto Scaling y balancear tráfico con el Load Balancer, antes pensábamos que eso era solo para empresas grandes pero ahora vimos que realmente mejora el rendimiento y la disponibilidad el trabajar con herramientas como Cloud9 facilitó mucho el desarrollo porque todo estaba en la nube, sin tener que preocuparnos por configuraciones locales y finalmente, aprendimos que la nube no es solo lanzar instancias EC2, sino pensar en diseño, eficiencia, seguridad, y costos. Nos llevamos una visión mucho más completa y profesional de cómo se hace una arquitectura bien pensada en AWS.

Badge de curso completado

https://www.credly.com/badges/4114fdf3-d10e-480f-af1a-3f730d0c903c/public_url