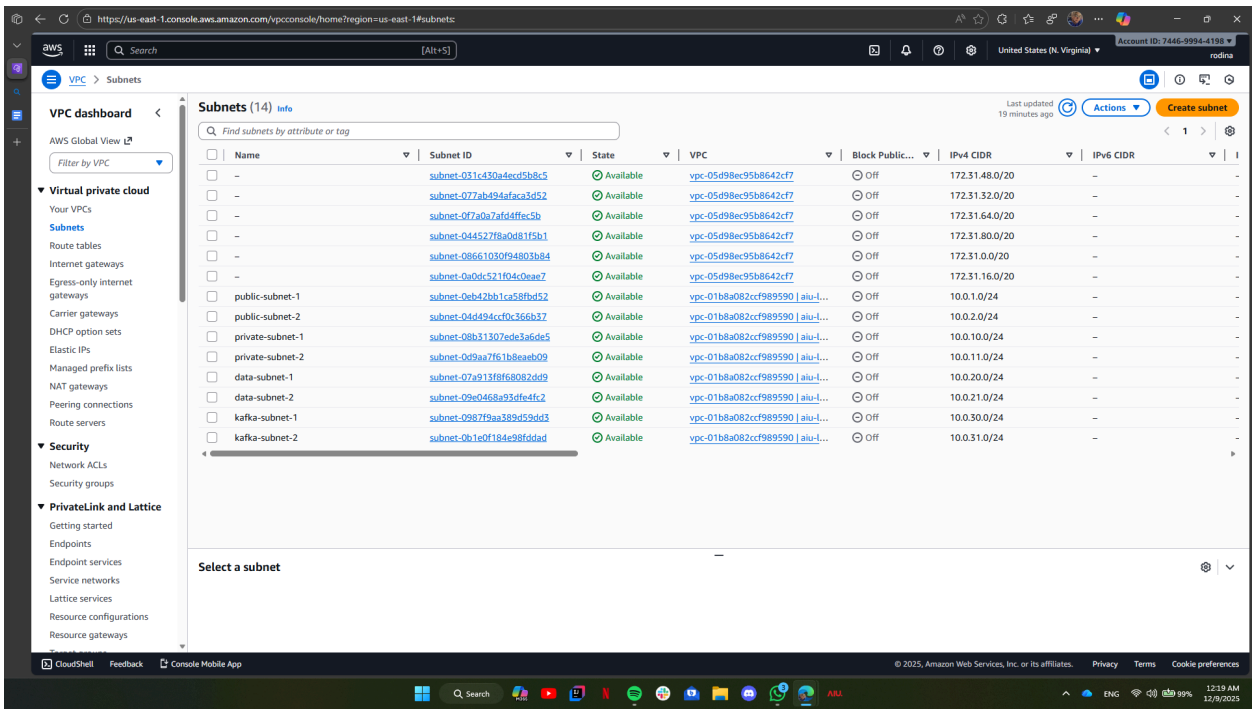
Aya Mamdouh Ahmed Elhabashy
22101087

VPC Creation:
Name: aiu-learning-platform-vpc

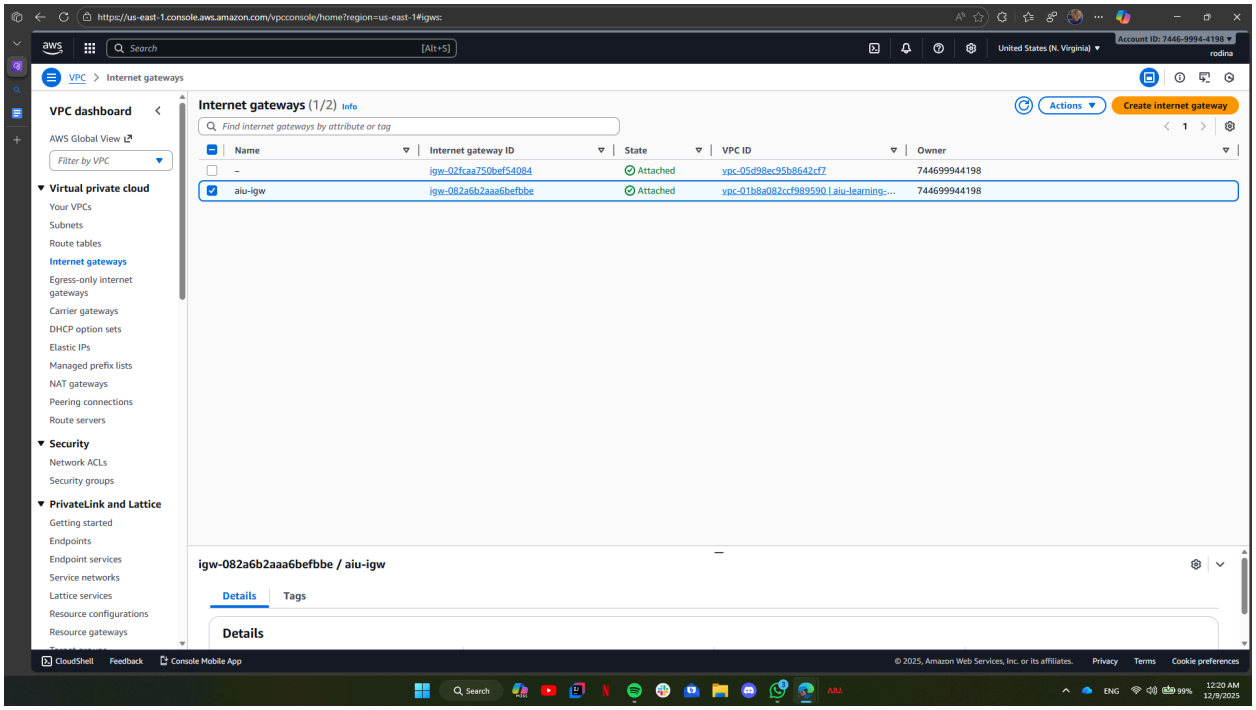
IPv4 CIDR: `10.0.0.0/16`

Tier	Subnets (2 AZs)	Purpose
Public	10.0.1.0/24, 10.0.2.0/24	Internet-facing ALB, NAT
Private	10.0.10.0/24,10.0.11.0/24	Container hosts
Data	10.0.20.0/24,10.0.21.0/24	RDS
Kafka	10.0.30.0/24,10.0.31.0/24	Kafka brokers & Zookeeper

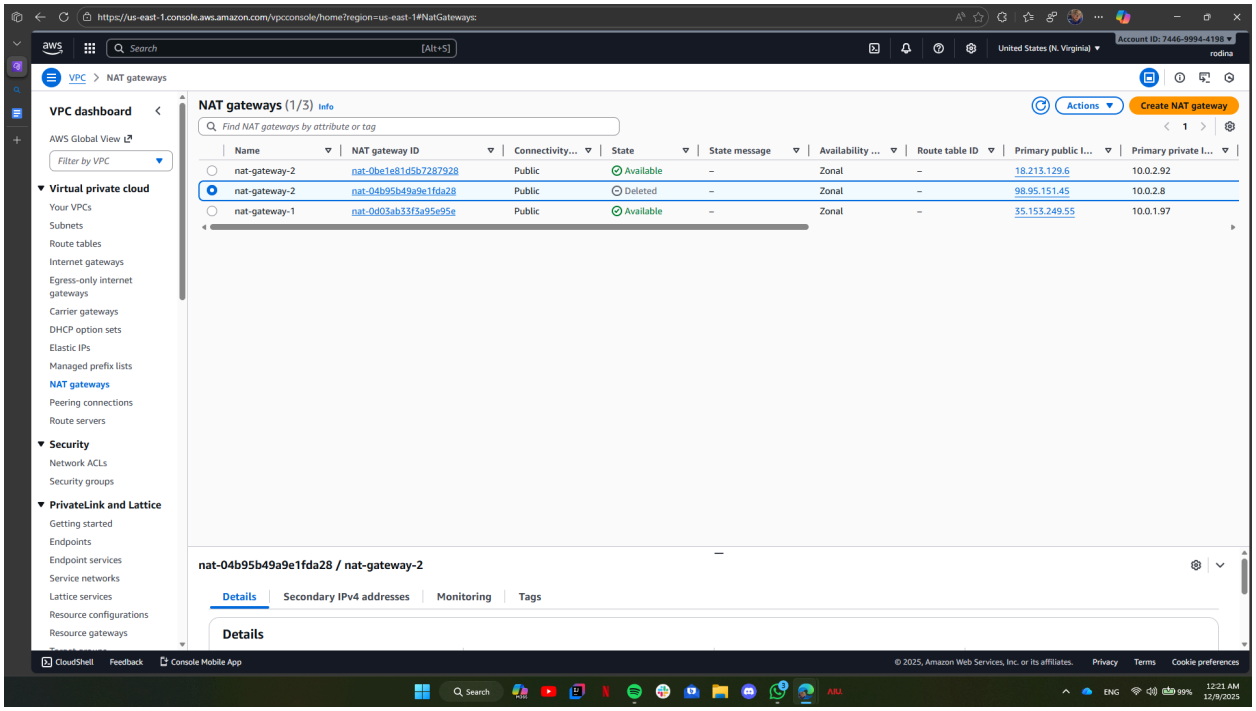
Subnets:



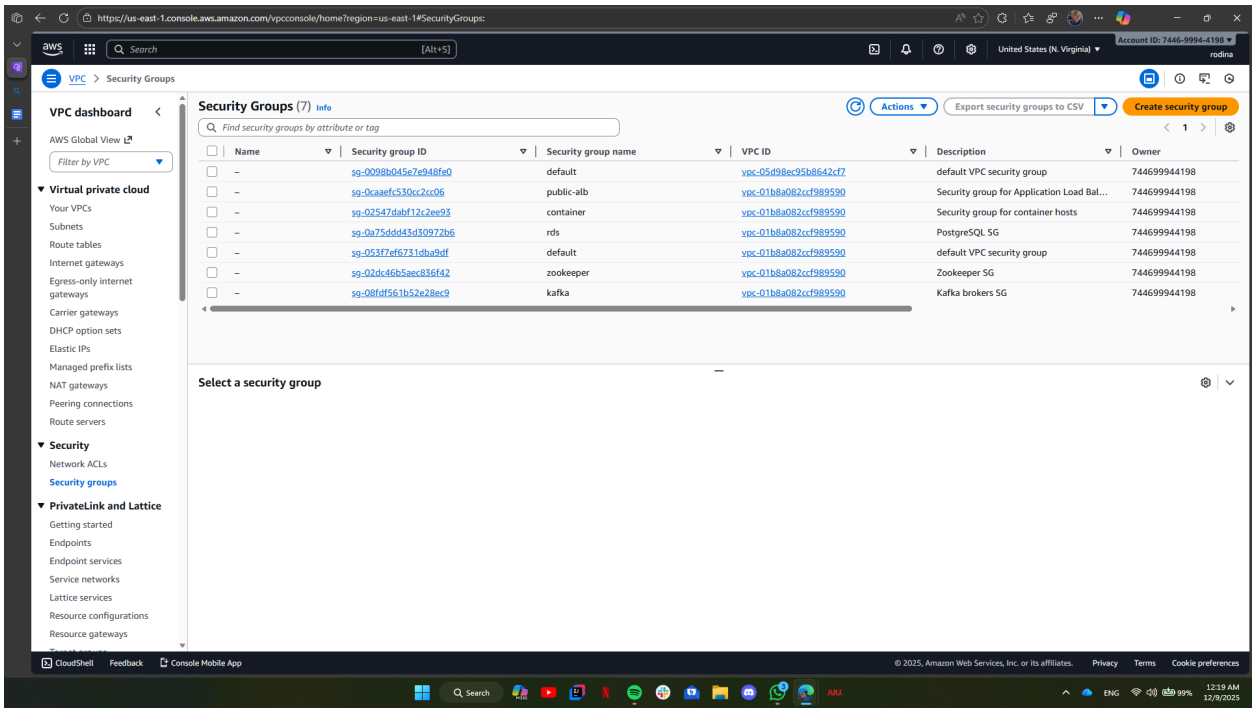
Internet Gateway:



NAT gateways:



Security Groups:



For Phase2 i worked on **AWS Apache Kafka**
I created the Kafka & Zookeeper

```
[ec2-user@ip-10-0-1-61 ~]$ sudo docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED         STATUS         PORTS                                                                                  NAMES
52f35f51808b   confluentinc/cp-kafka:7.4.2   "/etc/confluent/dock…"   12 minutes ago   Up 12 minutes   0.0.0.0:9092->9092/tcp, :::9092->9092/tcp                                              kafka
b024e44c1964   zookeeper:3.6.3       "/docker-entrypoint.…"   12 minutes ago   Up 12 minutes   2888/tcp, 3888/tcp, 0.0.0.0:2181->2181/tcp, :::2181->2181/tcp, 8080/tcp   zookeeper
[ec2-user@ip-10-0-1-61 ~]$
```

Using this:
For kafka :

```
sudo docker run -d --name kafka \
  --network kafka-net \
  -e KAFKA_BROKER_ID=1 \
  -e KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181 \
  -e KAFKA_LISTENERS=PLAINTEXT://0.0.0.0:9092 \
  -e KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://10.0.1.61:9092 \
  -e KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1 \
  -p 9092:9092 \
  confluentinc/cp-kafka:7.4.2
```

For zookeeper:

```
sudo docker run -d --name zookeeper \
  --network kafka-net \
  -p 2181:2181 \
```

```
  -v /home/ec2-user/zookeeper-data:/data \
  -e ZOO_MY_ID=1 \
  -e ZOO_PORT=2181 \
  -e ZOO_TICK_TIME=2000 \
  -e ZOO_INIT_LIMIT=5 \
  -e ZOO_SYNC_LIMIT=2 \
  zookeeper:3.6.3
```

And for creating the topics:

```
TOPICS=(
document.uploaded
document.processed
notes.generated
quiz.requested
quiz.generated
audio.transcription.requested
audio.transcription.completed
audio.generation.requested
audio.generation.completed
chat.message
)

for topic in "${TOPICS[@]}"; do
  sudo docker exec -it kafka /usr/bin/kafka-topics \
    --create \
    --bootstrap-server 10.0.1.61:9092 \
    --replication-factor 1 \
    --partitions 3 \
    --topic "$topic"
done
```

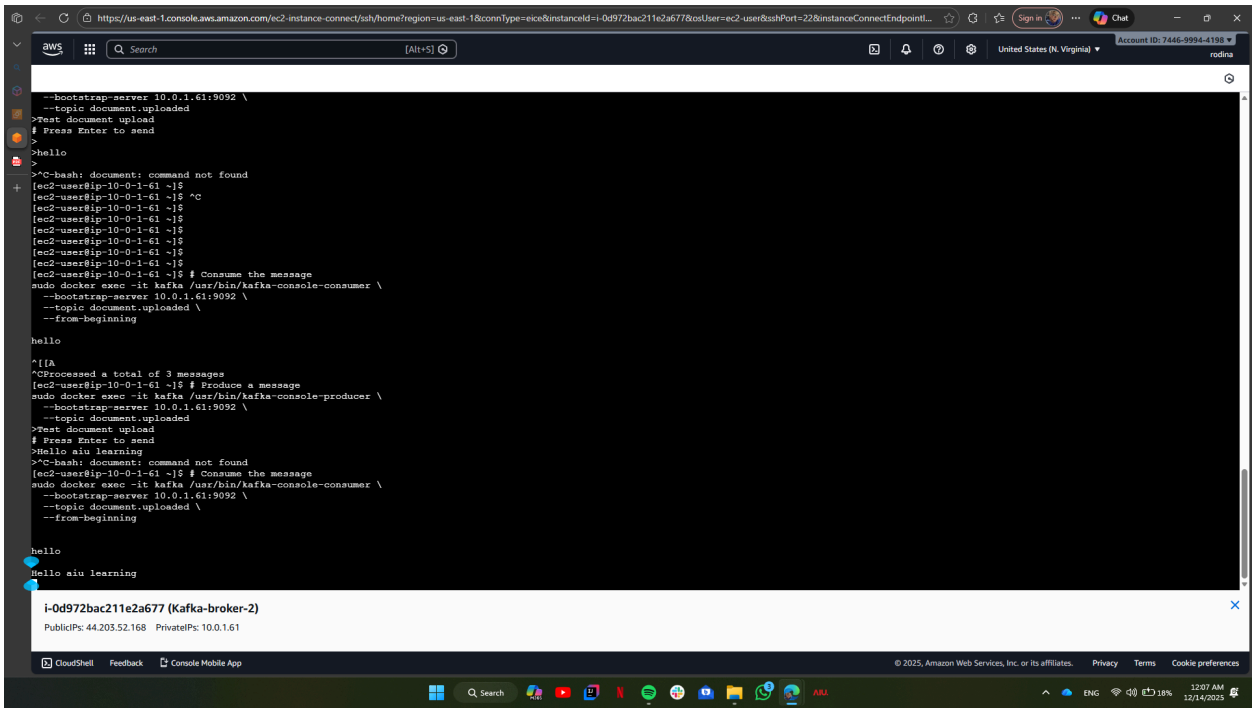
I did it on kafka broker1 but suddenly kafka & zookepeer stopped working so i used kafka broker 2 and started it all again

I created the producer & consumer using this:

And finally the producer sends a message to the consumer

```
# Produce a message
sudo docker exec -it kafka /usr/bin/kafka-console-producer \
  --bootstrap-server 10.0.1.61:9092 \
  --topic document.uploaded
>Test document upload
# Press Enter to send
```

```
# Consume the message
sudo docker exec -it kafka /usr/bin/kafka-console-consumer \
  --bootstrap-server 10.0.1.61:9092 \
  --topic document.uploaded \
  --from-beginning
```



I created the infrastructure as code “IAC”

WHY?

Infrastructure as Code Implementation

In this phase, AWS CloudFormation was used to implement the infrastructure using Infrastructure as Code (IaC). This approach allows the entire AWS environment to be defined declaratively in a YAML template, ensuring consistency, repeatability, and automation across deployments.

The infrastructure was designed in a modular manner, starting with a Virtual Private Cloud (VPC) that serves as an isolated network boundary for all application resources. Public and private subnets were created across multiple Availability Zones to support high availability and fault tolerance. Public subnets host internet-facing components

such as the Application Load Balancer, while private subnets are reserved for backend services like ECS tasks and databases, improving security.

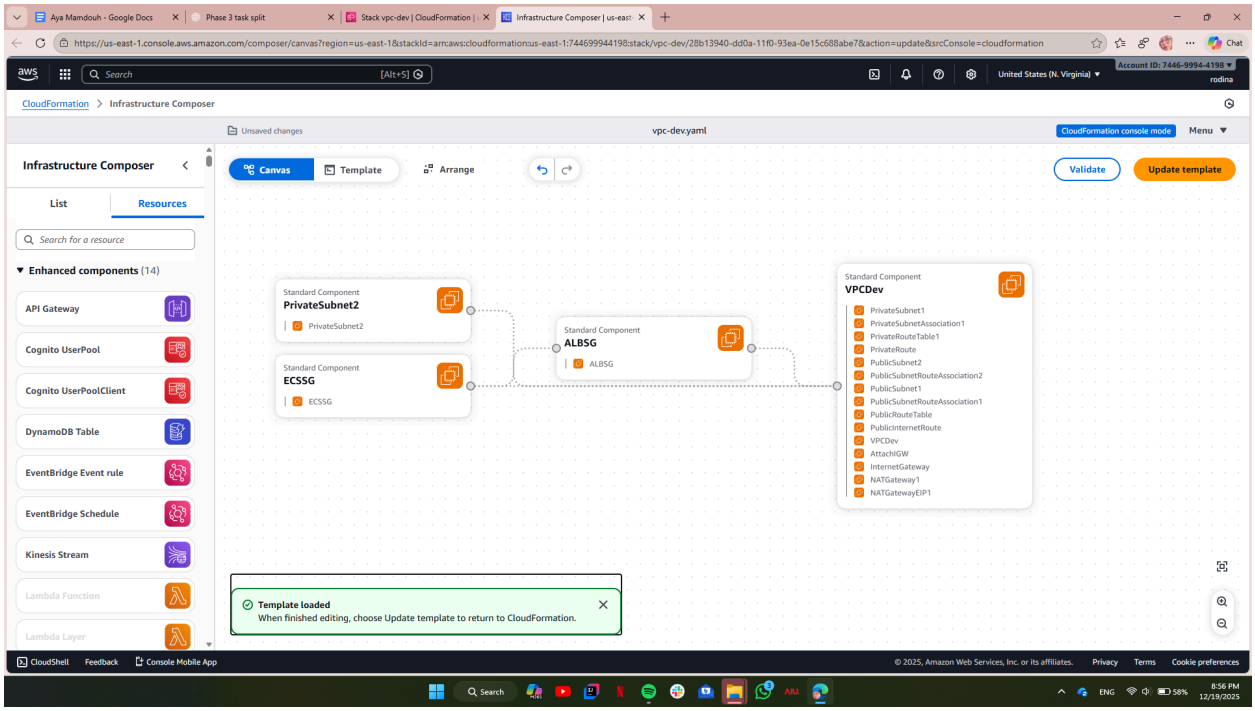
To enable outbound internet access for private resources without exposing them directly, a NAT Gateway was deployed in the public subnet and connected to private route tables. This design follows AWS best practices for secure networking.

Environment separation was achieved using CloudFormation parameters, allowing the same template to be reused for development, staging, and production environments. Resource naming and tagging dynamically adapt based on the selected environment, reducing duplication and simplifying management.

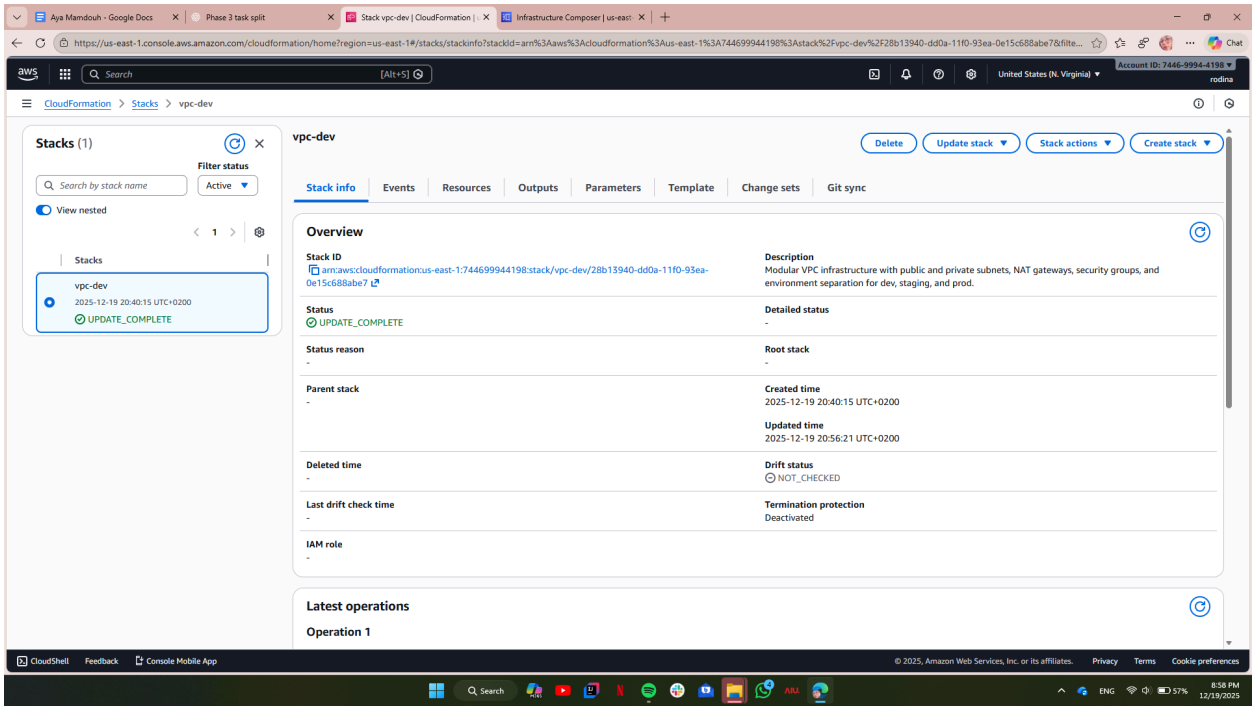
Security was enforced through the use of dedicated security groups. The Application Load Balancer security group allows HTTP and HTTPS traffic from the internet, while the ECS security group only permits traffic originating from the load balancer, implementing the principle of least privilege.

Finally, essential infrastructure identifiers such as VPC ID, subnet IDs, and security group IDs were exposed using CloudFormation outputs. These outputs enable seamless integration with CI/CD pipelines, allowing automated deployments to reference the infrastructure reliably.

That was the canvas:



It said completed Elhamdullah



And that the resources: