

Aya Mamdouh Ahmed Elhabashy  
22101087

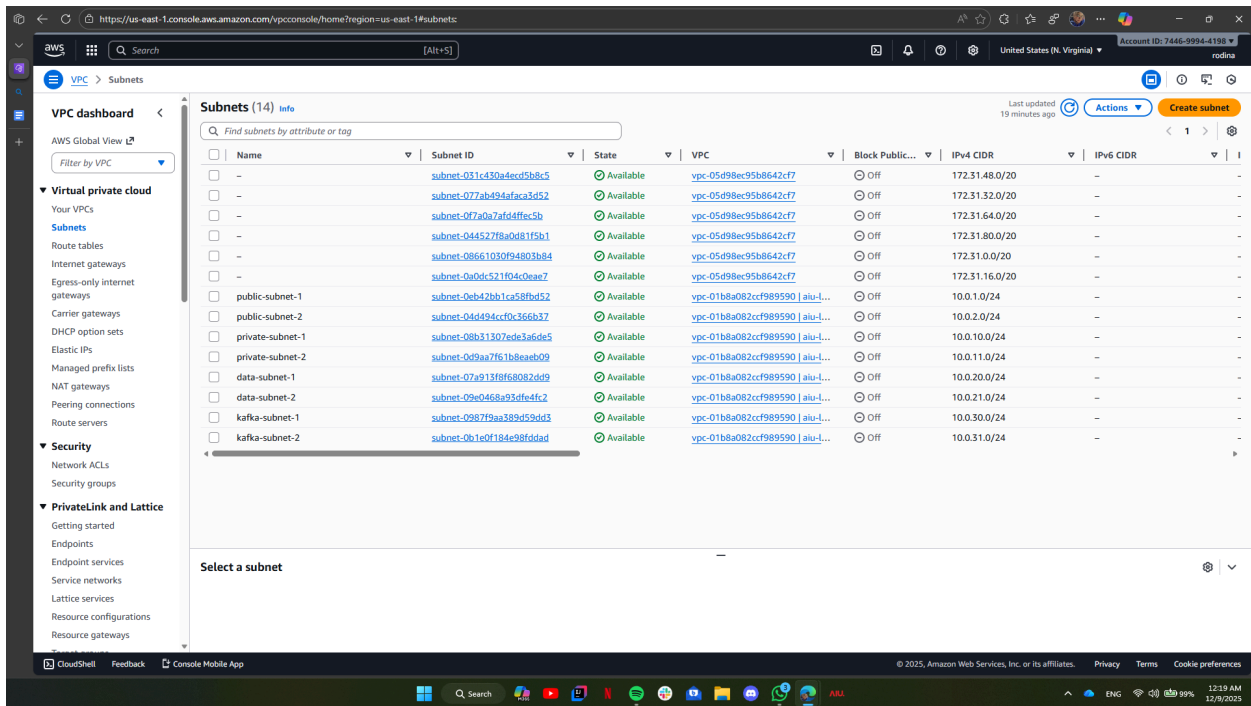
VPC Creation:

Name: ai-u-learning-platform-vpc

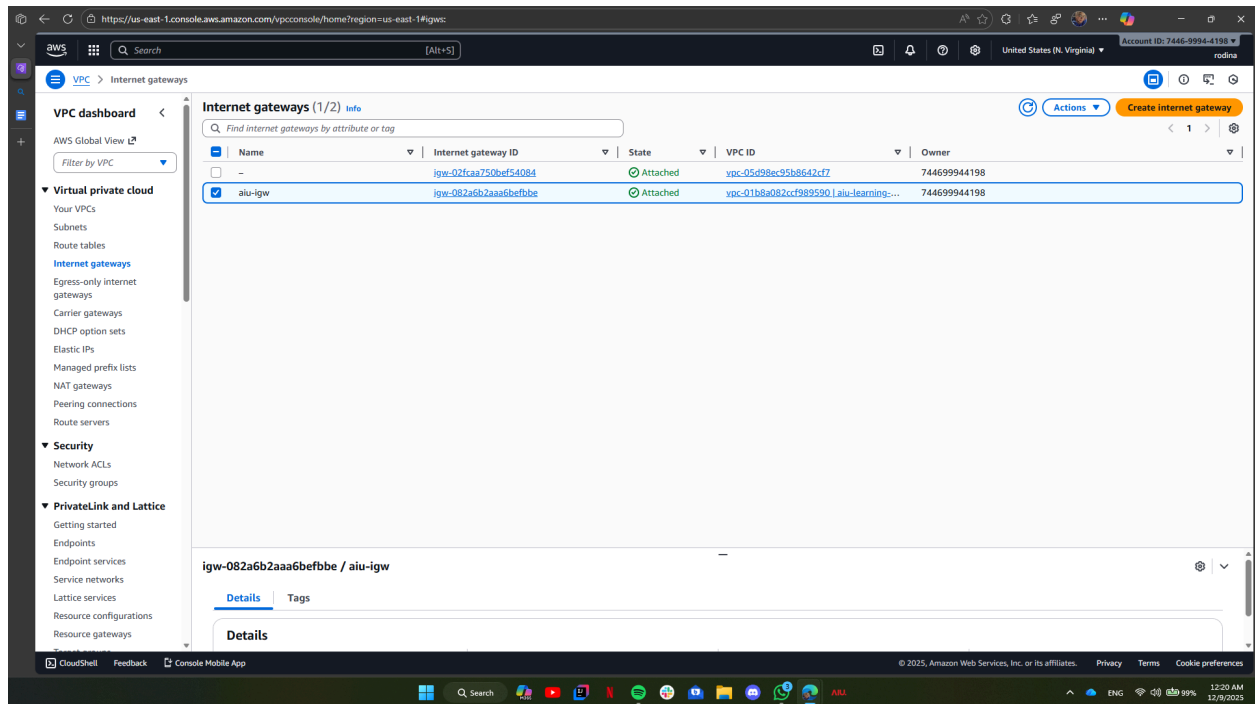
IPv4 CIDR: 10.0.0.0/16

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

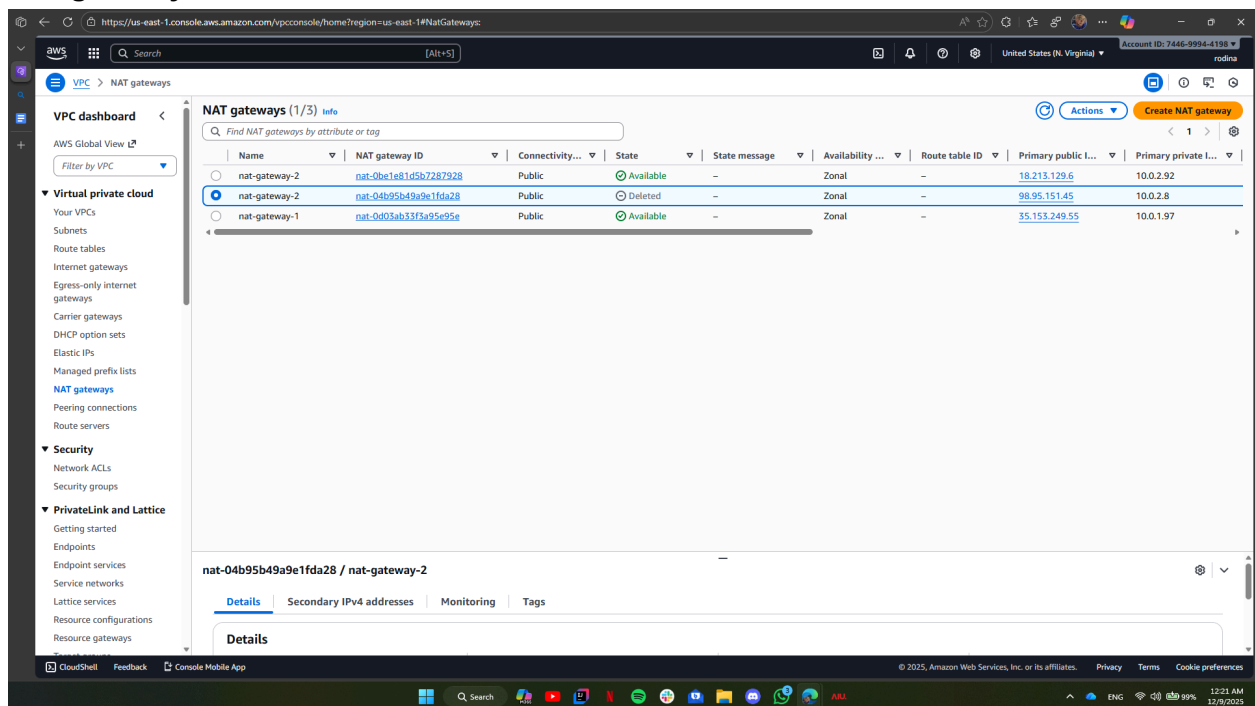
Subnets:



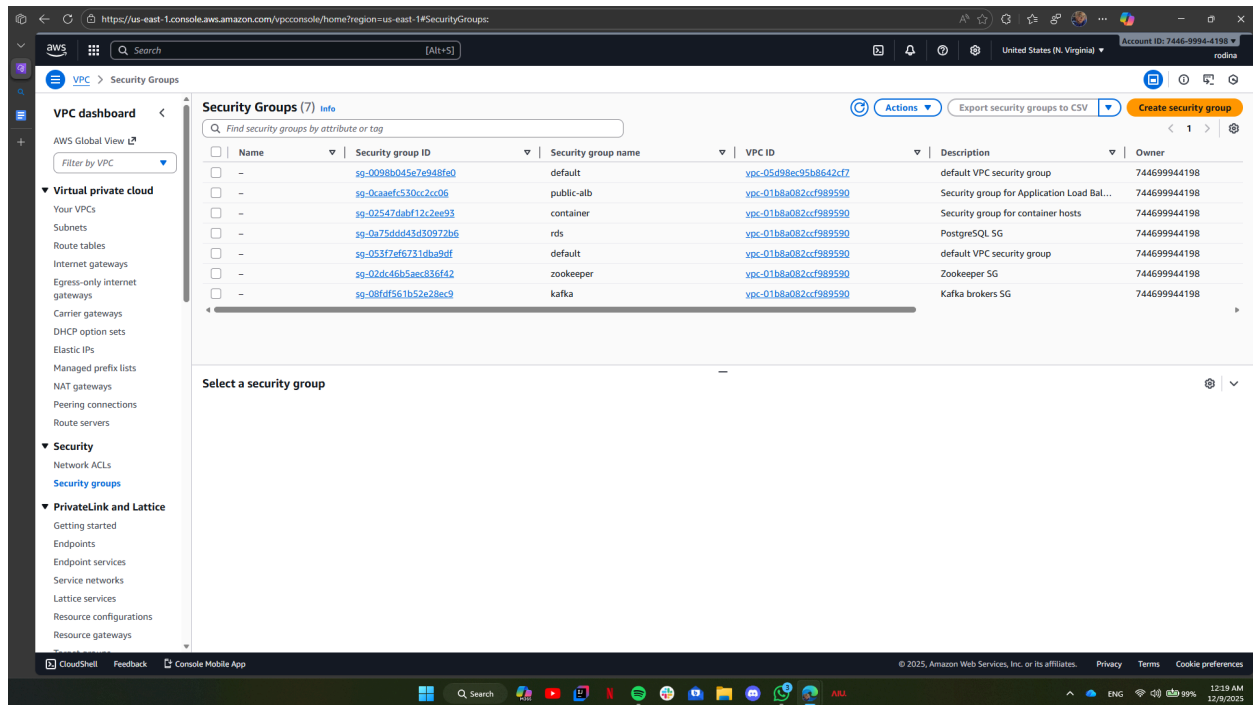
Internet Gateway:



## 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
32f35f51808b   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 & zookeeper stopped working so i used kafka broker 2 and started it all again

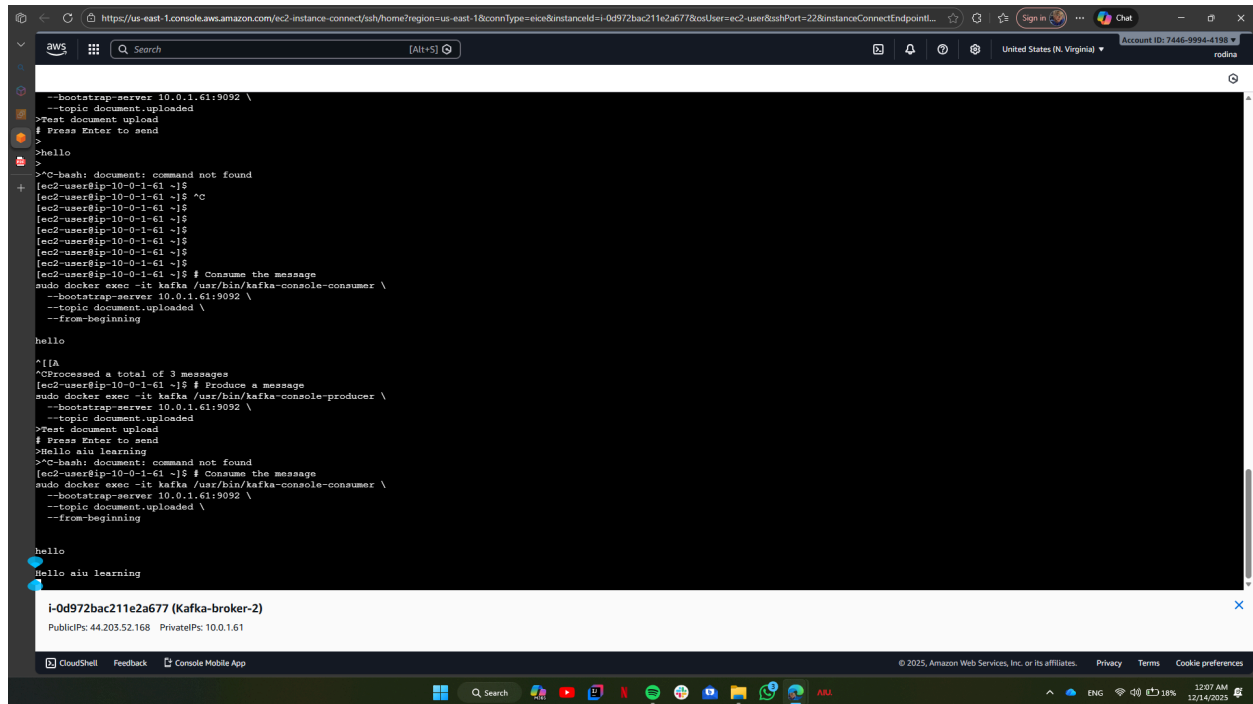
I created the producer & consumer using this:

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
```



```
--bootstrap-server 10.0.1.61:9092 \  
--topic document.uploaded  
>Test document upload  
# Press Enter to send  
>  
hello  
>  
>^C-bash: document: command not found  
[ec2-user@ip-10-0-1-61 ~]$  
[ec2-user@ip-10-0-1-61 ~]$ ^C  
[ec2-user@ip-10-0-1-61 ~]$  
[ec2-user@ip-10-0-1-61 ~]$  
[ec2-user@ip-10-0-1-61 ~]$  
[ec2-user@ip-10-0-1-61 ~]$  
[ec2-user@ip-10-0-1-61 ~]$  
[ec2-user@ip-10-0-1-61 ~]$ # 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  
  
hello  
  
^[[A  
^CProcessed a total of 3 messages  
[ec2-user@ip-10-0-1-61 ~]$ # 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  
>  
Hello aiu learning  
>  
>^C-bash: document: command not found  
[ec2-user@ip-10-0-1-61 ~]$ # 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  
  
hello  
Hello aiu learning
```

i-0d972bac211e2a677 (Kafka-broker-2)  
PublicIPs: 44.203.52.168 PrivateIPs: 10.0.1.61

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.

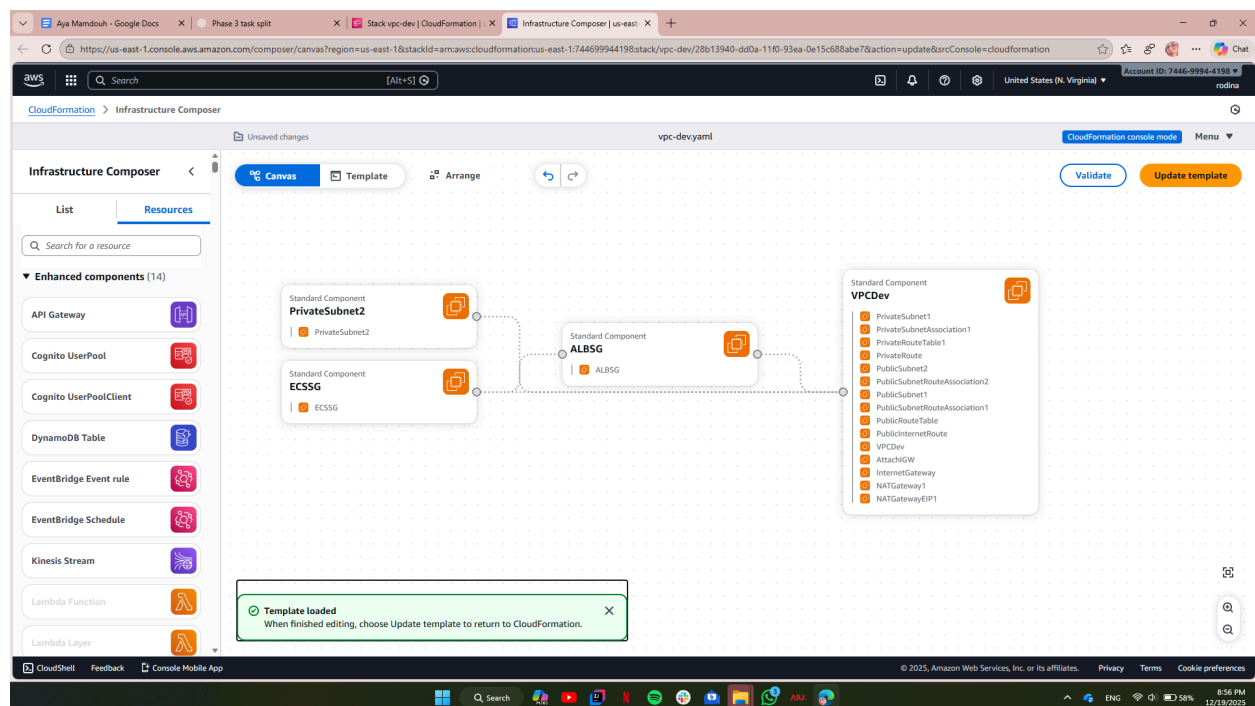
To enable outbound internet access for private resources without exposing them directly, a NAT Gateway was deployed in the public subnet and connected to private route tables. This design follows AWS best practices for secure networking.

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 Elhamdulillah

Stacks (1)

Filter status: Active

View nested

Stacks

Stacks

vpc-dev

2025-12-19 20:40:15 UTC+0200

UPDATE\_COMPLETE

vpc-dev

Delete Update stack Stack actions Create stack

Stack info Events Resources Outputs Parameters Template Change sets Git sync

Overview

Stack ID: [arn:aws:cloudformation:us-east-1:744699944198:stack/vpc-dev/28b13940-dd0a-11f0-93ea-de15c688abe7](#)

Description: Modular VPC infrastructure with public and private subnets, NAT gateways, security groups, and environment separation for dev, staging, and prod.

Status: UPDATE\_COMPLETE

Status reason: -

Parent stack: -

Deleted time: -

Last drift check time: -

IAM role: -

Detailed status: -

Root stack: -

Created time: 2025-12-19 20:40:15 UTC+0200

Updated time: 2025-12-19 20:56:21 UTC+0200

Drift status: NOT\_CHECKED

Termination protection: Deactivated

Latest operations

Operation 1

And that the resources:

Stacks (1)

Filter status: Active

View nested

Stacks

Stacks

vpc-dev

2025-12-19 20:40:15 UTC+0200

UPDATE\_IN\_PROGRESS

Resources (20)

Search resources

Logical ID	Physical ID	Type	Status	Module
NATGateway2	<a href="#">nat-0298bd8dce6e76137</a>	AWS::EC2::NatGateway	CREATE_COMPLETE	-
NATGatewayEIP1	<a href="#">98.95.91.7</a>	AWS::EC2::EIP	CREATE_COMPLETE	-
NATGatewayEIP2	<a href="#">98.86.206.231</a>	AWS::EC2::EIP	CREATE_COMPLETE	-
PrivateRoute	<a href="#">rtb-0c1637e017d1705edf0.0.0.0/0</a>	AWS::EC2::Route	CREATE_COMPLETE	-
PrivateRouteTable1	<a href="#">rtb-0c1637e017d1705ed</a>	AWS::EC2::RouteTable	CREATE_COMPLETE	-
PrivateSubnet1	<a href="#">subnet-04629f5441c4ea1fb</a>	AWS::EC2::Subnet	CREATE_COMPLETE	-
PrivateSubnet2	<a href="#">subnet-0da4593a92d029c42</a>	AWS::EC2::Subnet	CREATE_COMPLETE	-
PrivateSubnetAssociation1	<a href="#">rtbassoc-021f33593822d7c47</a>	AWS::EC2::SubnetRouteTableAssociati on	CREATE_COMPLETE	-
PublicInternetRoute	<a href="#">rtb-0bcf3cc95e223fdcc0.0.0.0/0</a>	AWS::EC2::Route	CREATE_COMPLETE	-
PublicRouteTable	<a href="#">rtb-0bcf3cc95e223fdcc</a>	AWS::EC2::RouteTable	UPDATE_COMPLETE	-
PublicSubnet1	<a href="#">subnet-0cc64d4619676bacd</a>	AWS::EC2::Subnet	UPDATE_COMPLETE	-
PublicSubnet2	<a href="#">subnet-0846a8ebc1741f633c</a>	AWS::EC2::Subnet	UPDATE_COMPLETE	-
PublicSubnetRouteAssociation1	<a href="#">rtbassoc-0184625846c3b1473</a>	AWS::EC2::SubnetRouteTableAssociati on	CREATE_COMPLETE	-
PublicSubnetRouteAssociation2	<a href="#">rtbassoc-08316c4d97e684d59</a>	AWS::EC2::SubnetRouteTableAssociati on	CREATE_COMPLETE	-
VPCDev	<a href="#">vpc-0a9b377e051a51953</a>	AWS::EC2::VPC	CREATE_COMPLETE	-