DEPI AWS Graduation Project

Project 9 & 10 & 11

1 PROJECT ASSIGNMENT

Task	Name
Project 9	Esraa Atef Eldegwy
Project 10	Ahmed Khaled Ahmed
Project 11	Yousef Ashraf AbdelRazek Ashraf Abdo abdo

2 TABLE OF CONTENTS

3 DOCUMENT OVERVIEW

This document is created by group 1 of AWS cloud solution admin and architect track as graduation project document to give an overview of project 9, 10 and 11.

Project 9 will give more details about how to create Amazon virtual private cloud with private and public subnet with internet gateway and route table, also create EC2 and access it from public subnet.

Project 10 will give more details about how to create VPC peering between two VPCs with route tables configurations and also enable VPC flow logs to monitor network traffic.

Project 11 will give more details about how to create Amzon Cognito use pool and manage users with Amazon Cognito identity pool configuration for authentication to secure application using it.

3.1 Project Scope

Project Number	Project Name	Description
Project 9	Creating an Amazon virtual private cloud (VPC)	• Deploy a VPC. • Create public and private subnets. • Set up an internet gateway and attach it to the VPC. • Configure route tables for public internet access. • Launch an application server to validate the VPC setup.
Project 10	Creating VPC peering connection	• Create a VPC peering connection between two VPCs. • Configure route tables to utilize the VPC peering connection. • Enable VPC Flow Logs to monitor network traffic. • Test the VPC peering connection. • Analyze VPC flow logs for traffic insights.
Project 11	Securing applications using Amazon Cognito	• Create an Amazon Cognito user pool and manage users. • Update a web application to use the user pool for authentication. • Configure an Amazon Cognito identity pool for authorization. • Update the application to use the identity pool for secure access to AWS services.

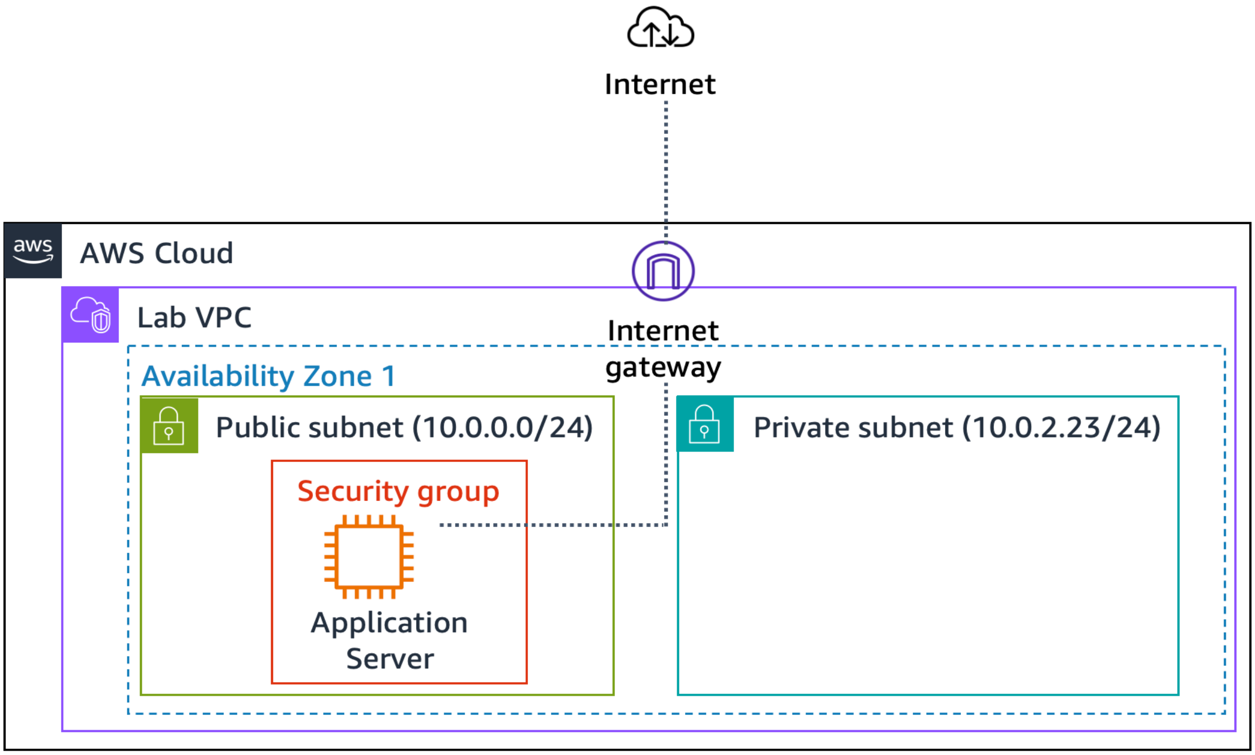
Table 1. Project Scope

4 PROJECT 9: CREATING AN AMAZON VIRTUAL PRIVATE CLOUD (VPC)

This project demonstrates how to create a Virtual Private Cloud (VPC) using Amazon VPC, enabling you to deploy a secure private network in AWS. You will learn to create a VPC, set up public and private subnets, attach an internet gateway, and launch an application server to test the VPC configuration.

Project Tasks:

- Deploy a VPC.
- Create public and private subnets.
- Set up an internet gateway and attach it to the VPC.
- Configure route tables for public internet access.
- Launch an application server to validate the VPC setup.



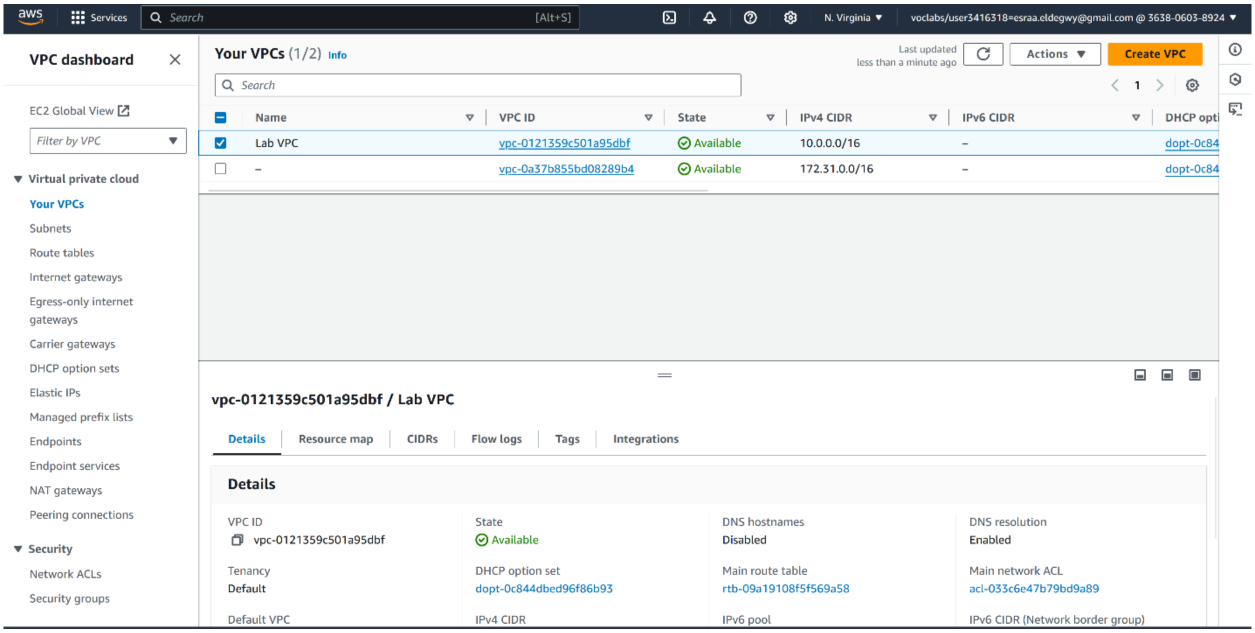
Task 1: Creating a VPC

A VPC is a virtual network that is dedicated to your Amazon Web Services (AWS) account. It is logically isolated from other virtual networks in the AWS Cloud. You can launch AWS resources, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, into the VPC. You can configure the VPC by modifying its IP address range and create subnets. You can also configure route tables, network gateways, and security settings.

- Navigate to the VPC service in the AWS Management Console.
- Create a new VPC with the following settings:
- **Name Tag**: Lab VPC
- **IPv4 CIDR Block**: 10.0.0.0/16
- Enable DNS hostnames for the VPC to assign friendly DNS names to EC2 instances.

Commands used on AWS CLI:

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16 --query Vpc.VpcId --output text
aws ec2 create-tags --resources vpc-0121359c501a95dbf --tags Key=Name,Value='Lab VPC'
aws ec2 modify-vpc-attribute --vpc-id vpc-0121359c501a95dbf --enable-dns-hostnames
```



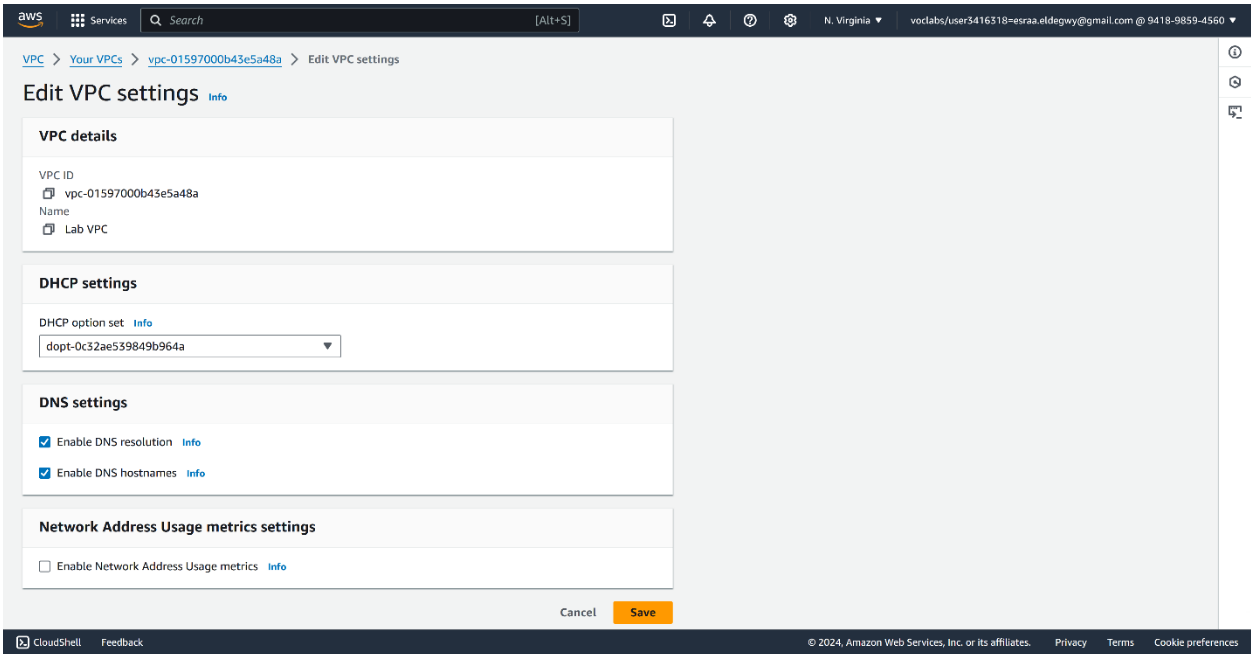
Your VPCs (1/2) Info

☐	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP opti
☑	Lab VPC	vpc-0121359c501a95dbf	✓ Available	10.0.0.0/16	–	dopt-0c84
☐	–	vpc-0a37b855bd08289b4	✓ Available	172.31.0.0/16	–	dopt-0c84

vpc-0121359c501a95dbf / Lab VPC

Details

VPC ID vpc-0121359c501a95dbf	State ✓ Available	DNS hostnames Disabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-0c844dbed96f86b93	Main route table rtb-09a19108f5f569a58	Main network ACL acl-033c6e47b79bd9a89
Default VPC	IPv4 CIDR	IPv6 pool	IPv6 CIDR (Network border group)



aws
Services
Search
[Alt+S]
N. Virginia ▼
voclabs/user3416318=esraa.eldegwy@gmail.com @ 9418-9859-4560 ▼
VPC > Your VPCs > vpc-01597000b43e5a48a > Edit VPC settings

Edit VPC settings Info

VPC details

VPC ID
vpc-01597000b43e5a48a
Name
Lab VPC

DHCP settings

DHCP option set Info
dopt-0c32ae539849b964a ▼

DNS settings

☑ Enable DNS resolution Info

☑ Enable DNS hostnames Info

Network Address Usage metrics settings

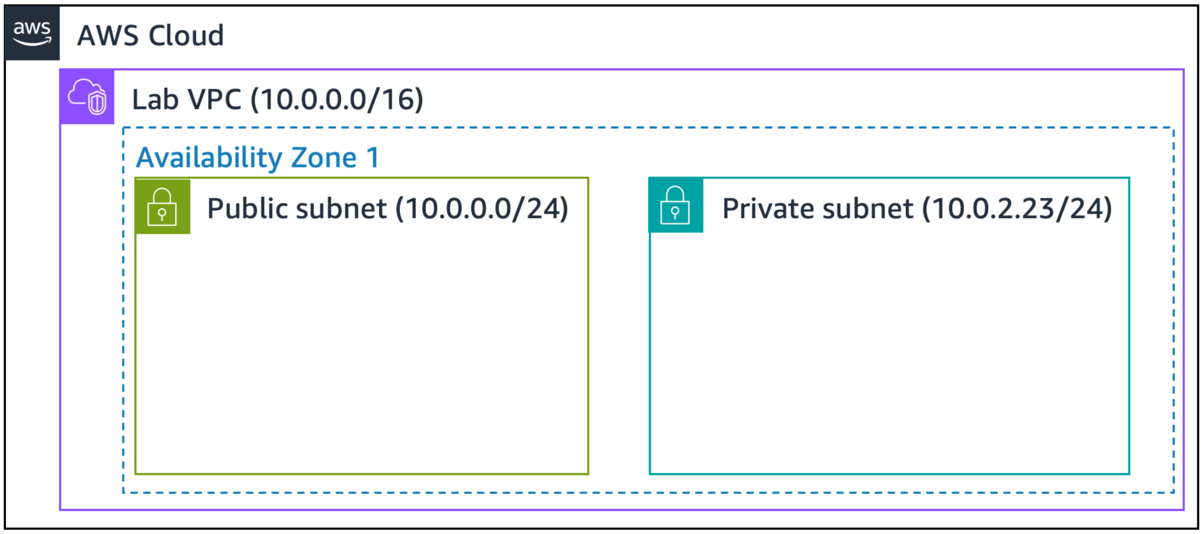
☐ Enable Network Address Usage metrics Info

Cancel
Save
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Task 2: Creating Subnets

A subnet is a subrange of IP addresses in the VPC. AWS resources can be launched into a specified subnet. Use a public subnet for resources that must be connected to the internet, and use a private subnet for resources that must remain isolated from the internet.

In this task, you create a public subnet and a private subnet:



Task 2.1: Creating a Public Subnet

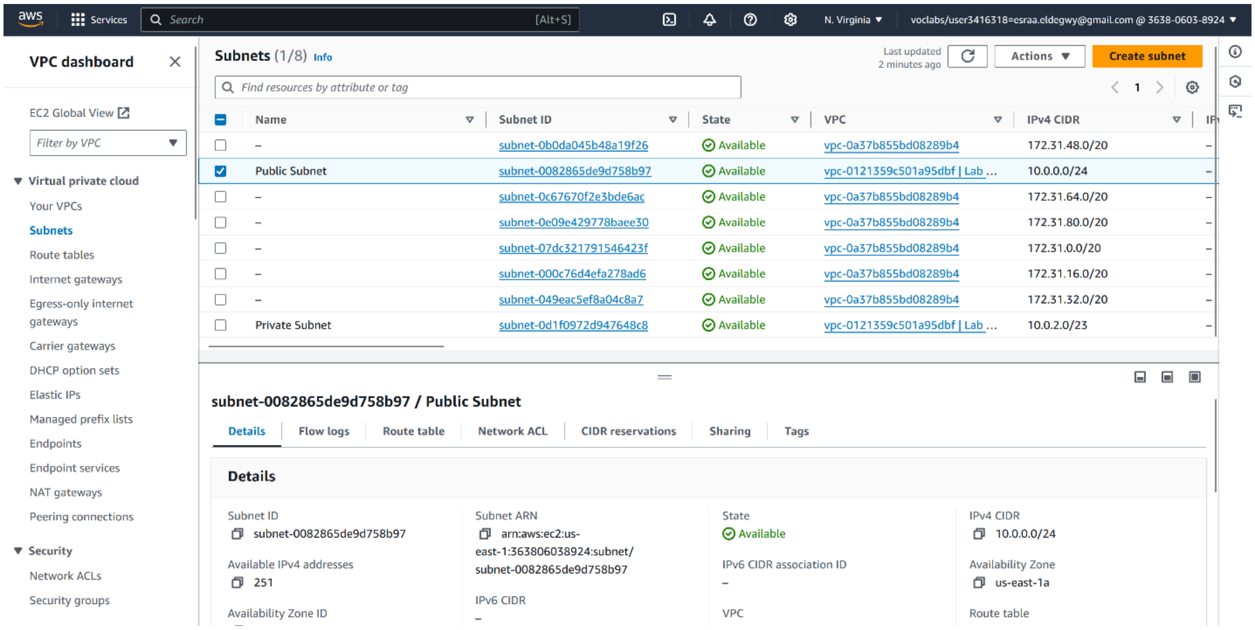
- Create a public subnet in the Lab VPC with the following settings:
- **Subnet Name**: Public Subnet
- **Availability Zone**: Choose the first Availability Zone.
- **IPv4 Subnet CIDR Block**: 10.0.0.0/24
- Enable auto-assign public IPv4 addresses for instances launched in this subnet.

Commands used on AWS CLI:

```
aws ec2 create-subnet --vpc-id vpc-0121359c501a95dbf --cidr-block 10.0.0.0/24 --availability-zone us-east-1a --query Subnet.SubnetId --output text
```

```
aws ec2 create-tags --resources subnet-0082865de9d758b97 --tags Key=Name,Value='Public Subnet'
```

```
aws ec2 modify-subnet-attribute --subnet-id subnet-0082865de9d758b97 --map-public-ip-on-launch
```



Subnets (1/8) Info

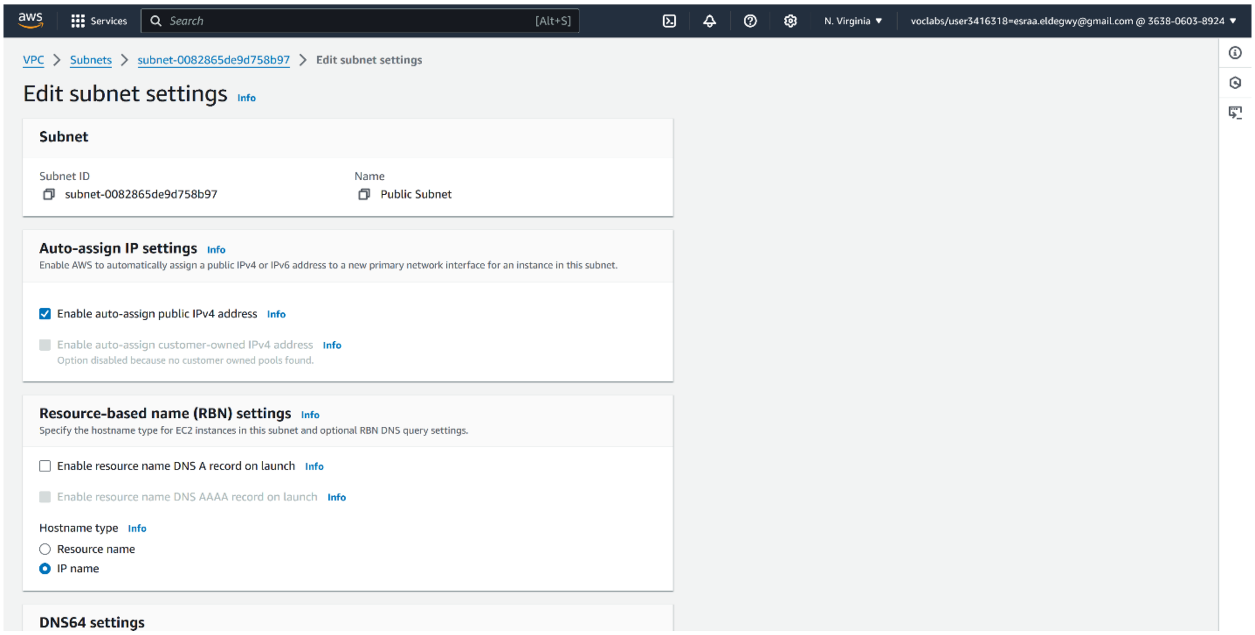
	Name	Subnet ID	State	VPC	IPv4 CIDR	IP
□	–	subnet-0b0da045b48a19f26	✓ Available	vpc-0a37b855bd08289b4	172.31.48.0/20	–
☑	Public Subnet	subnet-0082865de9d758b97	✓ Available	vpc-0121359c501a95dbf | Lab ...	10.0.0.0/24	–
□	–	subnet-0c67670f2e3bde6ac	✓ Available	vpc-0a37b855bd08289b4	172.31.64.0/20	–
□	–	subnet-0e09e429778baee30	✓ Available	vpc-0a37b855bd08289b4	172.31.80.0/20	–
□	–	subnet-07dc321791546423f	✓ Available	vpc-0a37b855bd08289b4	172.31.0.0/20	–
□	–	subnet-000c76d4efa278ad6	✓ Available	vpc-0a37b855bd08289b4	172.31.16.0/20	–
□	–	subnet-049eac5ef8a04c8a7	✓ Available	vpc-0a37b855bd08289b4	172.31.32.0/20	–
□	Private Subnet	subnet-0d1f0972d947648c8	✓ Available	vpc-0121359c501a95dbf | Lab ...	10.0.2.0/23	–

subnet-0082865de9d758b97 / Public Subnet

Details | Flow logs | Route table | Network ACL | CIDR reservations | Sharing | Tags

Details

Subnet ID: subnet-0082865de9d758b97
Subnet ARN: arn:aws:ec2:us-east-1:363806038924:subnet/subnet-0082865de9d758b97
State: ✓ Available
IPv4 CIDR: 10.0.0.0/24
Available IPv4 addresses: 251
IPv6 CIDR association ID: –
Availability Zone: us-east-1a
Availability Zone ID
IPv6 CIDR: –
VPC
Route table


aws
Services
Search
[Alt+S]
N. Virginia ▼
voclabs/user3416318=esraa.eldegwy@gmail.com @ 3638-0603-8924 ▼

VPC > Subnets > subnet-0082865de9d758b97 > Edit subnet settings

Edit subnet settings Info

Subnet

Subnet ID
subnet-0082865de9d758b97

Name
Public Subnet

Auto-assign IP settings Info
Enable AWS to automatically assign a public IPv4 or IPv6 address to a new primary network interface for an instance in this subnet.

☑ Enable auto-assign public IPv4 address Info

□ Enable auto-assign customer-owned IPv4 address Info
Option disabled because no customer owned pools found.

Resource-based name (RBN) settings Info
Specify the hostname type for EC2 instances in this subnet and optional RBN DNS query settings.

□ Enable resource name DNS A record on launch Info

□ Enable resource name DNS AAAA record on launch Info

Hostname type Info
○ Resource name
● IP name

DNS64 settings

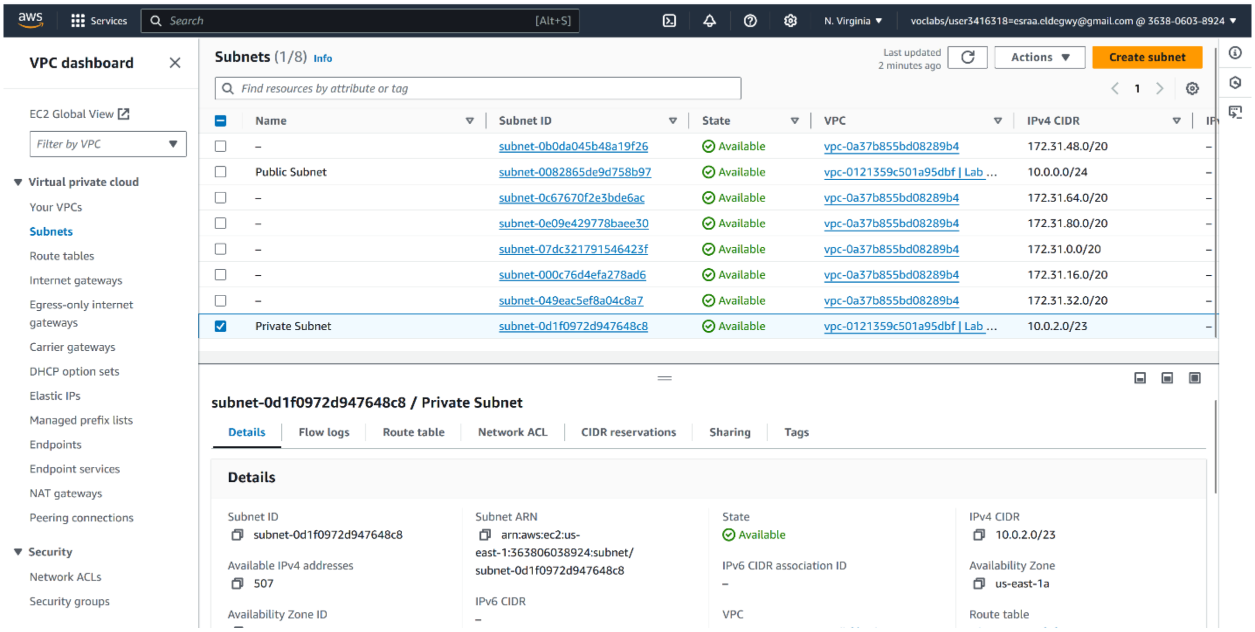
Task 2.2: Creating a Private Subnet

- Create a private subnet in the Lab VPC with the following settings:
- **Subnet Name**: Private Subnet
- **Availability Zone**: Choose the first Availability Zone.
- **IPv4 Subnet CIDR Block**: 10.0.2.0/23

Commands used on AWS CLI:

```
aws ec2 create-subnet --vpc-id vpc-0121359c501a95dbf --cidr-block 10.0.2.0/23 --availability-zone us-east-1a --query Subnet.SubnetId --output text
```

```
aws ec2 create-tags --resources subnet-0d1f0972d947648c8 --tags Key=Name,Value='Private Subnet'
```



Subnets (1/8)

	Name	Subnet ID	State	VPC	IPv4 CIDR
□	–	subnet-0b0da045b48a19f26	Available	vpc-0a37b855bd08289b4	172.31.48.0/20
□	Public Subnet	subnet-0082865de9d758b97	Available	vpc-0121359c501a95dbf | Lab ...	10.0.0.0/24
□	–	subnet-0c67670f2e3bde6ac	Available	vpc-0a37b855bd08289b4	172.31.64.0/20
□	–	subnet-0e09e429778baee30	Available	vpc-0a37b855bd08289b4	172.31.80.0/20
□	–	subnet-07dc321791546423f	Available	vpc-0a37b855bd08289b4	172.31.0.0/20
□	–	subnet-000c76d4efa278ad6	Available	vpc-0a37b855bd08289b4	172.31.16.0/20
□	–	subnet-049eac5ef8a04c8a7	Available	vpc-0a37b855bd08289b4	172.31.32.0/20
☑	Private Subnet	subnet-0d1f0972d947648c8	Available	vpc-0121359c501a95dbf | Lab ...	10.0.2.0/23

subnet-0d1f0972d947648c8 / Private Subnet

Details

Subnet ID: subnet-0d1f0972d947648c8
Subnet ARN: arn:aws:ec2:us-east-1:363806038924:subnet/subnet-0d1f0972d947648c8
State: Available
IPv4 CIDR: 10.0.2.0/23
Available IPv4 addresses: 507
IPv6 CIDR association ID: –
Availability Zone: us-east-1a
Availability Zone ID
IPv6 CIDR: –
VPC
Route table

VPC now has two subnets. However, the public subnet is totally isolated and cannot communicate with resources outside the VPC. Next, you configure the public subnet to connect to the internet through an internet gateway.

Task 3: Creating an Internet Gateway

An internet gateway is a horizontally scaled, redundant, and highly available VPC component. It allows communication between the instances in a VPC and the internet. It imposes no availability risks or bandwidth constraints on network traffic.
An internet gateway serves two purposes:

1. To provide a target in route tables that connects to the internet
2. To perform network address translation (NAT) for instances that were assigned public IPv4 addresses

In this task we will do the following:

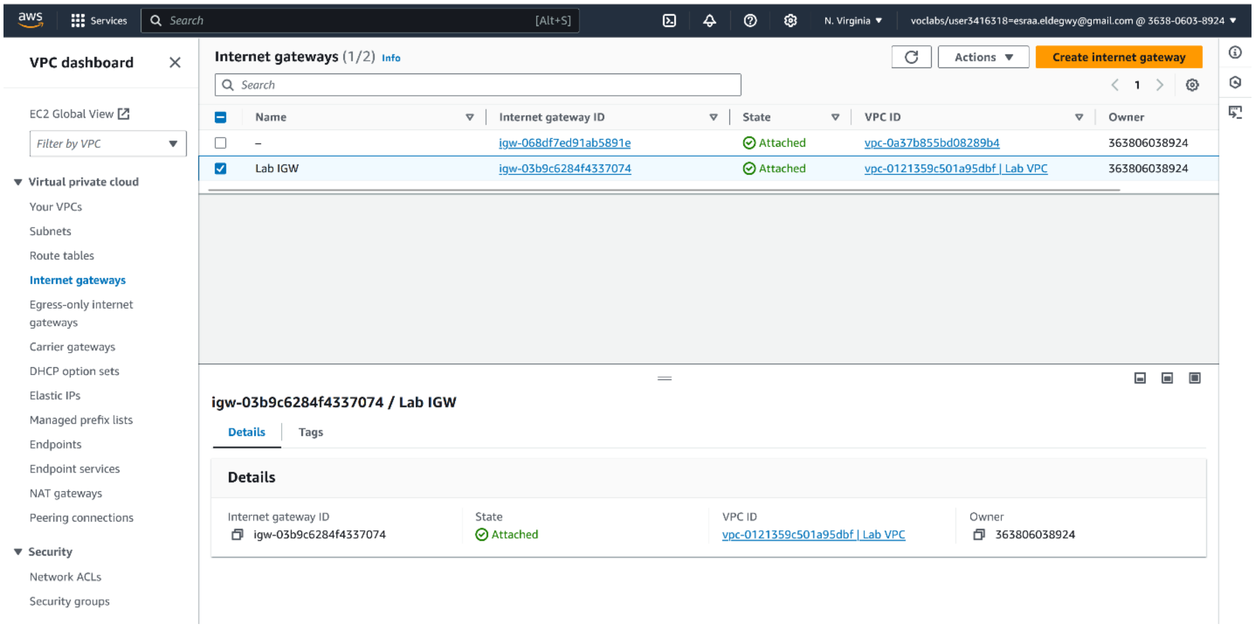
- Create an internet gateway named Lab IGW.
- Attach the internet gateway to the Lab VPC.

Commands used on AWS CLI:

```
aws ec2 create-internet-gateway --query InternetGateway.InternetGatewayId --output text
```

```
aws ec2 create-tags --resources igw-03b9c6284f4337074 --tags Key=Name,Value='Lab IGW'
```

```
aws ec2 attach-internet-gateway --vpc-id vpc-0121359c501a95dbf --internet-gateway-id igw-03b9c6284f4337074
```



Internet gateways (1/2) Info

	Name	Internet gateway ID	State	VPC ID	Owner
□	–	igw-068df7ed91ab5891e	Attached	vpc-0a37b855bd08289b4	363806038924
☑	Lab IGW	igw-03b9c6284f4337074	Attached	vpc-0121359c501a95dbf | Lab VPC	363806038924

igw-03b9c6284f4337074 / Lab IGW

Details

Internet gateway ID	State	VPC ID	Owner
igw-03b9c6284f4337074	Attached	vpc-0121359c501a95dbf | Lab VPC	363806038924

Task 4: Configuring Route Tables

A route table contains a set of rules, called routes, that are used to determine where network traffic is directed. Each subnet in a VPC must be associated with a route table because the table controls the routing for the subnet. A subnet can be associated with only one route table at a time, but you can associate multiple subnets with the same route table.

To use an internet gateway, a subnet's route table must contain a route that directs internet-bound traffic to the internet gateway. If a subnet is associated with a route table that has a route to an internet gateway, it is known as a public subnet.

In this task, you do the following:

- Create a public route table for internet-bound traffic.
- Add a route to the route table to direct internet-bound traffic to the internet gateway.
- Associate the public subnet with the new route table.

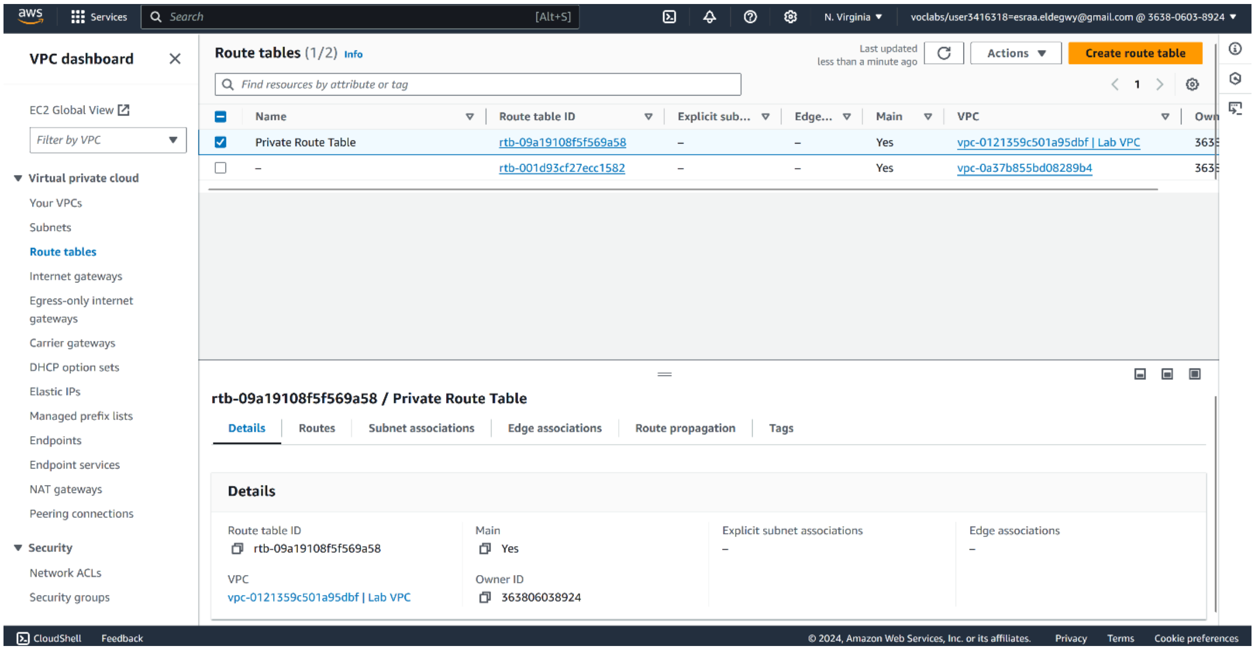
Commands used on AWS CLI:

```
aws ec2 create-tags --resources rtb-09a19108f5f569a58 --tags Key=Name,Value='Private Route Table'
```

```
aws ec2 create-route-table --vpc-id vpc-0121359c501a95dbf --query RouteTable.RouteTableId --output text
```

```
aws ec2 create-tags --resources rtb-0e096b4161659f8a9 --tags Key=Name,Value='Public Route Table'
```

```
aws ec2 create-route --route-table-id rtb-0e096b4161659f8a9 --destination-cidr-block 0.0.0.0/0 --gateway-id igw-03b9c6284f4337074
aws ec2 associate-route-table --route-table-id rtb-0e096b4161659f8a9 --subnet-id subnet-0082865de9d758b97
```



Route tables (1/2) Info

	Name	Route table ID	Explicit sub...	Edge...	Main	VPC	Own
☑	Private Route Table	rtb-09a19108f5f569a58	–	–	Yes	vpc-0121359c501a95dbf | Lab VPC	363
□	–	rtb-001d93cf27ecc1582	–	–	Yes	vpc-0a37b855bd08289b4	363

rtb-09a19108f5f569a58 / Private Route Table

Details

Route table ID: rtb-09a19108f5f569a58

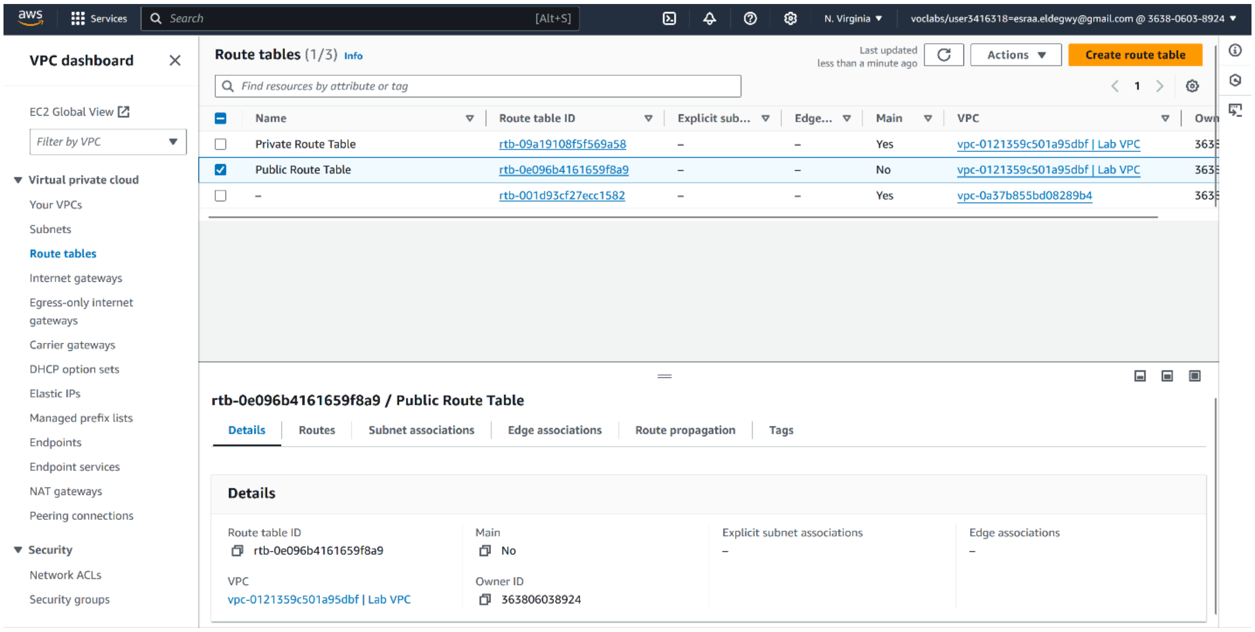
Main: Yes

Explicit subnet associations: –

Edge associations: –

VPC: vpc-0121359c501a95dbf | Lab VPC

Owner ID: 363806038924



Route tables (1/3) Info

	Name	Route table ID	Explicit sub...	Edge...	Main	VPC	Own
□	Private Route Table	rtb-09a19108f5f569a58	–	–	Yes	vpc-0121359c501a95dbf | Lab VPC	363
☑	Public Route Table	rtb-0e096b4161659f8a9	–	–	No	vpc-0121359c501a95dbf | Lab VPC	363
□	–	rtb-001d93cf27ecc1582	–	–	Yes	vpc-0a37b855bd08289b4	363

rtb-0e096b4161659f8a9 / Public Route Table

Details

Route table ID: rtb-0e096b4161659f8a9

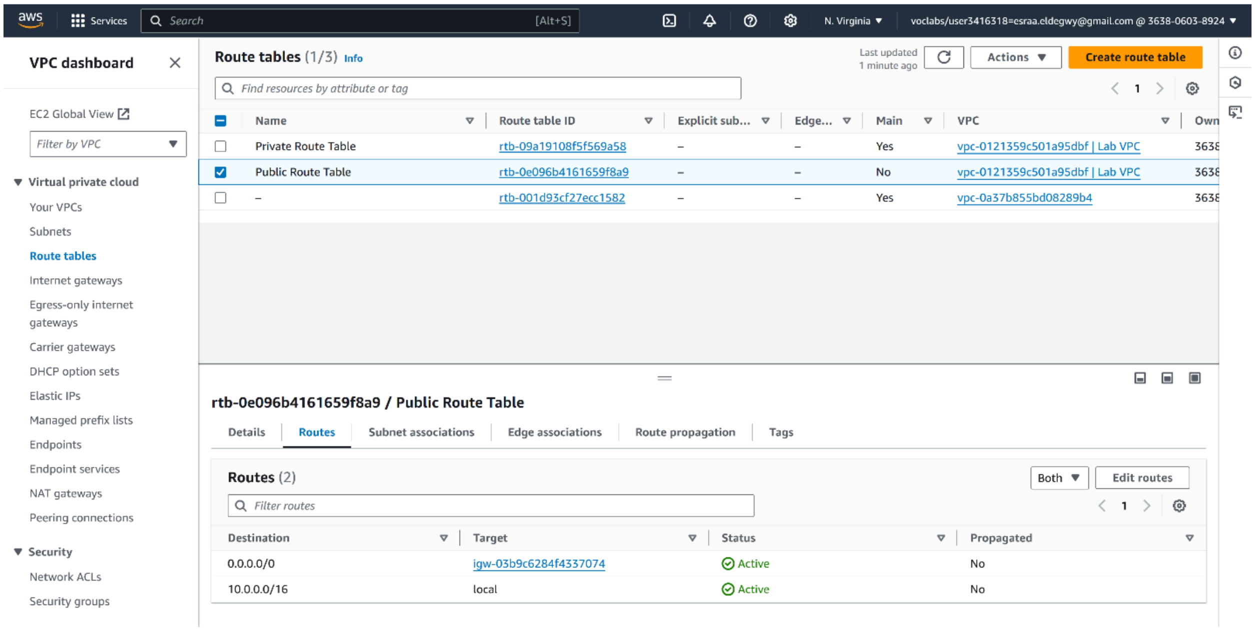
Main: No

Explicit subnet associations: –

Edge associations: –

VPC: vpc-0121359c501a95dbf | Lab VPC

Owner ID: 363806038924



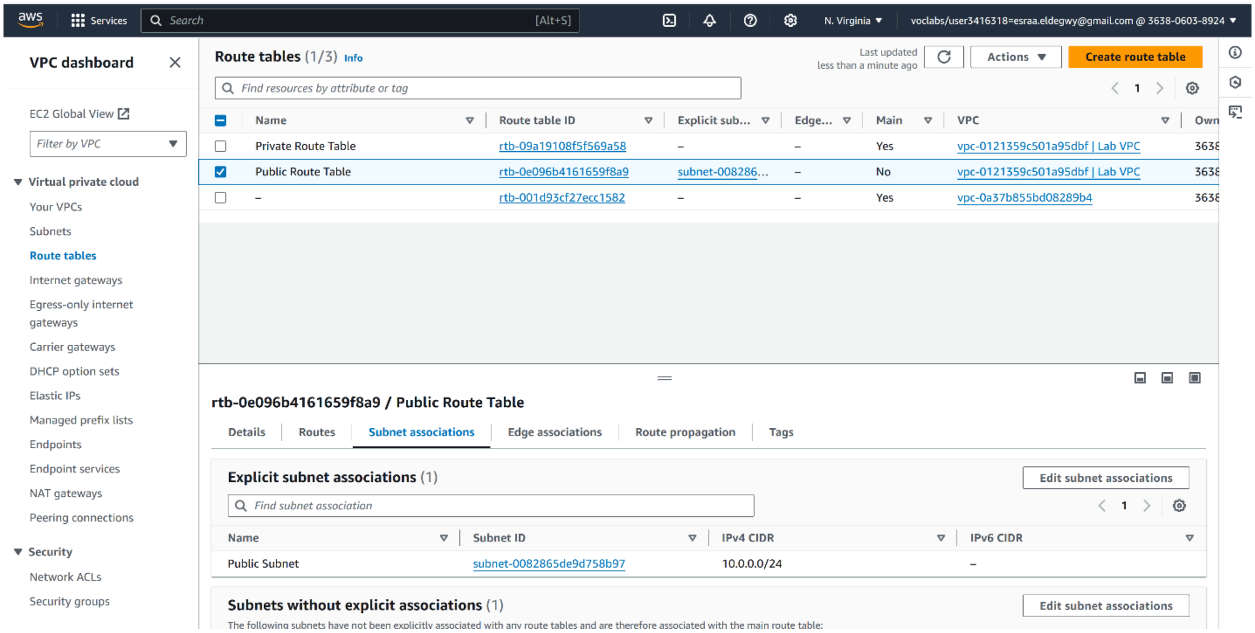
Route tables (1/3) Info

	Name	Route table ID	Explicit sub...	Edge...	Main	VPC	Own
□	Private Route Table	rtb-09a19108f5f569a58	–	–	Yes	vpc-0121359c501a95dbf | Lab VPC	3638
☑	Public Route Table	rtb-0e096b4161659f8a9	–	–	No	vpc-0121359c501a95dbf | Lab VPC	3638
□	–	rtb-001d93cf27ecc1582	–	–	Yes	vpc-0a37b855bd08289b4	3638

rtb-0e096b4161659f8a9 / Public Route Table

Routes (2)

Destination	Target	Status	Propagated
0.0.0.0/0	igw-03b9c6284f4337074	✓ Active	No
10.0.0.0/16	local	✓ Active	No



Route tables (1/3) Info

	Name	Route table ID	Explicit sub...	Edge...	Main	VPC	Own
□	Private Route Table	rtb-09a19108f5f569a58	–	–	Yes	vpc-0121359c501a95dbf | Lab VPC	3638
☑	Public Route Table	rtb-0e096b4161659f8a9	subnet-008286...	–	No	vpc-0121359c501a95dbf | Lab VPC	3638
□	–	rtb-001d93cf27ecc1582	–	–	Yes	vpc-0a37b855bd08289b4	3638

rtb-0e096b4161659f8a9 / Public Route Table

Explicit subnet associations (1)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
Public Subnet	subnet-0082865de9d758b97	10.0.0.0/24	–

Subnets without explicit associations (1)

The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

Task 5: Creating a Security Group for the Application Server

A *security group* acts as a virtual firewall for instances to control inbound and outbound traffic. Security groups operate at the level of the elastic network interface for the instance. Security groups do not operate at the subnet level. Thus, each instance can have its own firewall that controls traffic. If you do not specify a particular security group at launch time, the instance is automatically assigned to the default security group for the VPC.

In this task, you create a security group that gives users the ability to access your application server through HTTP.

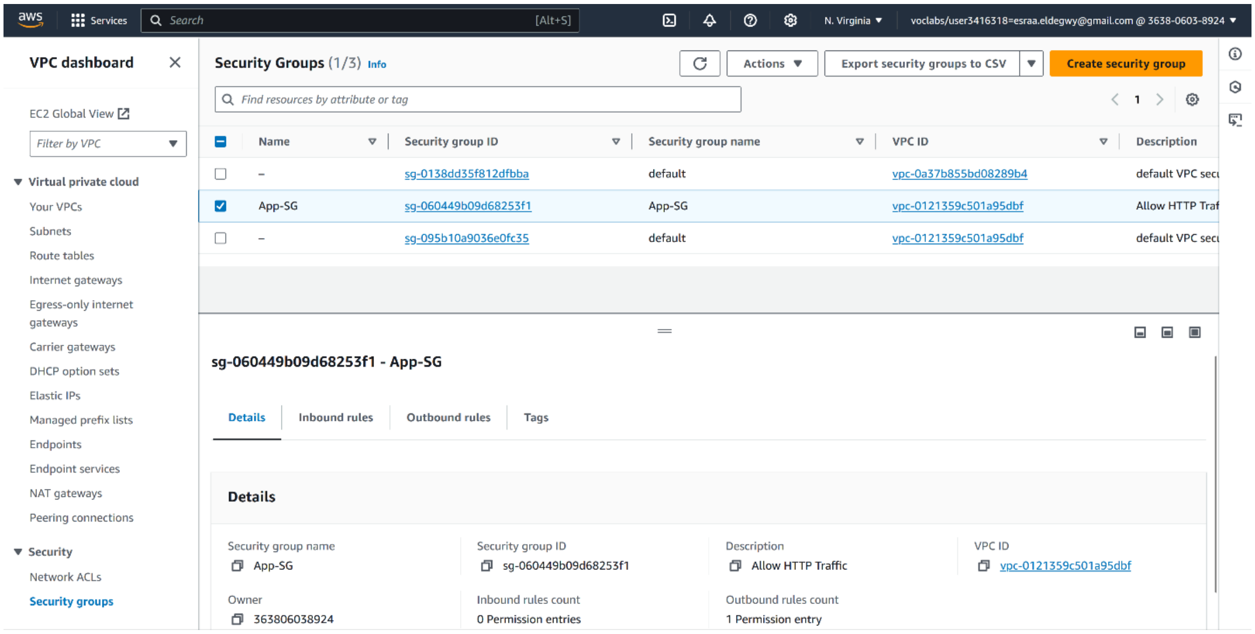
- Create a security group named App-SG to allow HTTP traffic:
- **Inbound Rule**: Allow HTTP (port 80) from Anywhere (0.0.0.0/0).
- This security group will be used by the application server launched in the public subnet.

Commands used on AWS CLI:

```
aws ec2 create-security-group --group-name App-SG --description "Allow HTTP Traffic" --vpc-id vpc-0121359c501a95dbf
```

```
aws ec2 create-tags --resources sg-060449b09d68253f1 --tags Key=Name,Value='App-SG'
```

```
aws ec2 authorize-security-group-ingress --group-id sg-060449b09d68253f1 --ip-permissions 'IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[{CidrIp=0.0.0.0/0,Description="Allow Web Access"}]'
```



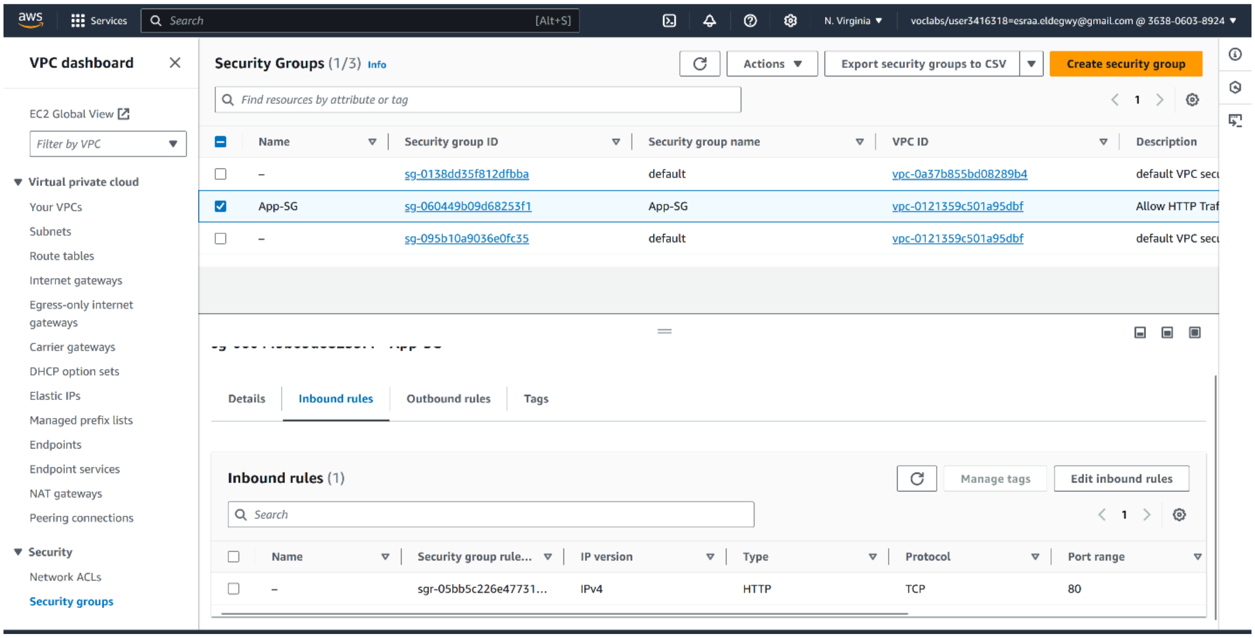
Security Groups (1/3) Info

	Name	Security group ID	Security group name	VPC ID	Description
□	–	sg-0138dd35f812dfbba	default	vpc-0a37b855bd08289b4	default VPC secu
☑	App-SG	sg-060449b09d68253f1	App-SG	vpc-0121359c501a95dbf	Allow HTTP Traf
□	–	sg-095b10a9036e0fc35	default	vpc-0121359c501a95dbf	default VPC secu

sg-060449b09d68253f1 - App-SG

Details

Security group name	Security group ID	Description	VPC ID
App-SG	sg-060449b09d68253f1	Allow HTTP Traffic	vpc-0121359c501a95dbf
Owner	**Inbound rules count**	**Outbound rules count**	
363806038924	0 Permission entries	1 Permission entry	

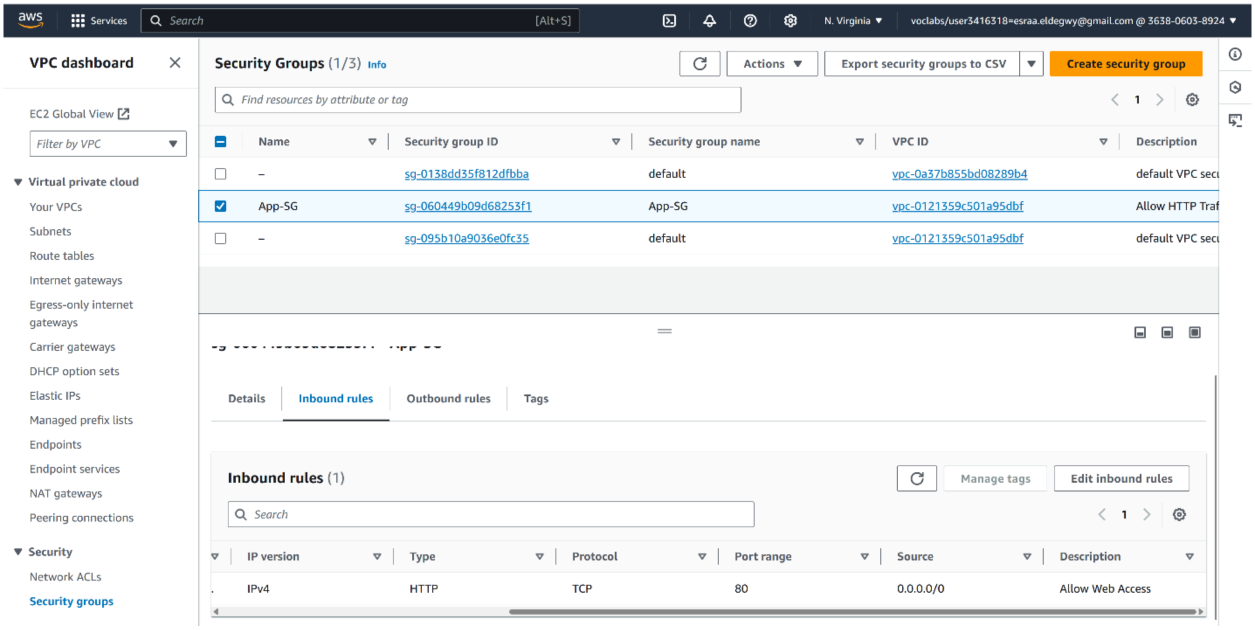


Security Groups (1/3) Info

	Name	Security group ID	Security group name	VPC ID	Description
□	–	sg-0138dd35f812dfbba	default	vpc-0a37b855bd08289b4	default VPC secu
☑	App-SG	sg-060449b09d68253f1	App-SG	vpc-0121359c501a95dbf	Allow HTTP Traf
□	–	sg-095b10a9036e0fc35	default	vpc-0121359c501a95dbf	default VPC secu

Inbound rules (1)

	Name	Security group rule...	IP version	Type	Protocol	Port range
□	–	sgr-05bb5c226e47731...	IPv4	HTTP	TCP	80



Security Groups (1/3) Info

	Name	Security group ID	Security group name	VPC ID	Description
□	–	sg-0138dd35f812dfbba	default	vpc-0a37b855bd08289b4	default VPC secu
☑	App-SG	sg-060449b09d68253f1	App-SG	vpc-0121359c501a95dbf	Allow HTTP Traf
□	–	sg-095b10a9036e0fc35	default	vpc-0121359c501a95dbf	default VPC secu

Inbound rules (1)

IP version	Type	Protocol	Port range	Source	Description
IPv4	HTTP	TCP	80	0.0.0.0/0	Allow Web Access

Task 6: Launching an Application Server in the Public Subnet

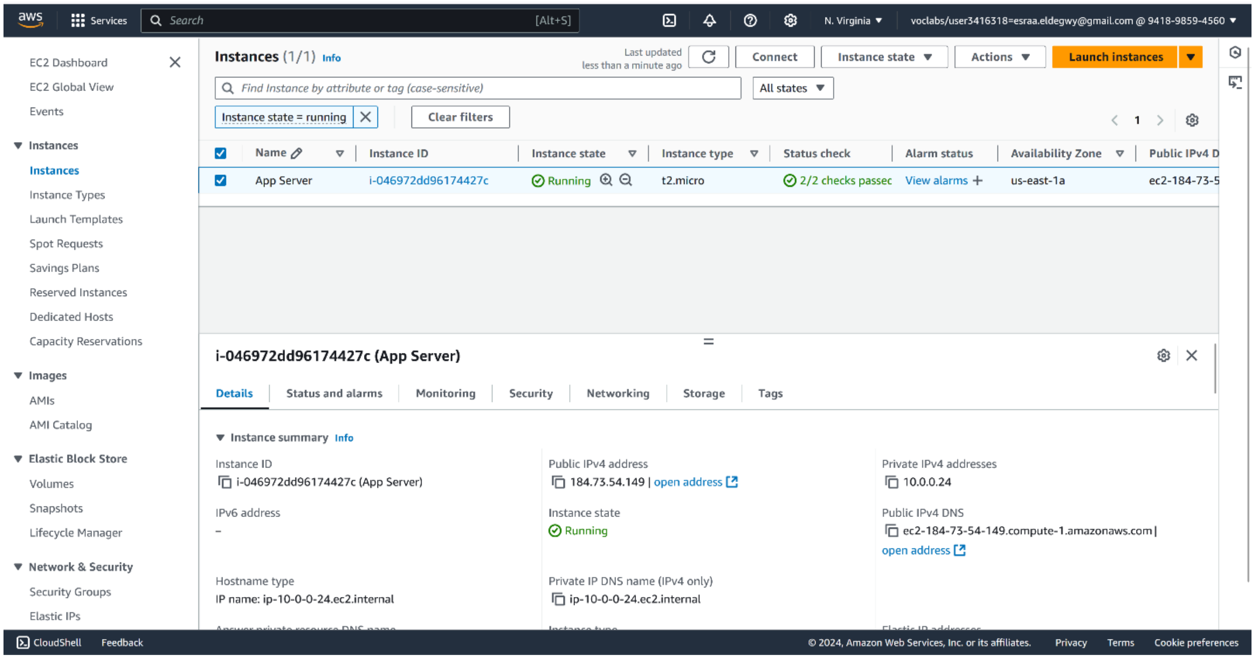
To test that your VPC is correctly configured, you now launch an EC2 instance into the public subnet. You also confirm that you can access the EC2 instance from the internet.

- Launch an EC2 instance (App Server) into the public subnet using the following settings:
- **Instance Type**: t2.micro
- **Subnet**: Public Subnet
- **Security Group**: App-SG
- **User Data**: Use the provided script to install and configure the web application.
- Validate the setup by accessing the instance's public IPv4 DNS in a web browser. Ensure the Inventory application loads successfully, indicating the correct VPC configuration.

Commands used on AWS CLI:

```
aws ec2 run-instances --image-id ami-0fff1b9a61dec8a5f --count 1 --instance-type t2.micro --key-name vockey --subnet-id subnet-0082865de9d758b97 --security-group-ids sg-060449b09d68253f1 --iam-instance-profile Name=Inventory-App-Role --user-data file:C:\Users\esraae\my_script.txt
```

```
aws ec2 create-tags --resources i-0f657f3b115346b00 --tags Key=Name,Value='App Server'
```



Instances (1/1) Info

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
☑	App Server	i-046972dd96174427c	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-184-73-5

i-046972dd96174427c (App Server)

Instance summary Info

Instance ID
i-046972dd96174427c (App Server)

Public IPv4 address
184.73.54.149 | open address

Private IPv4 addresses
10.0.0.24

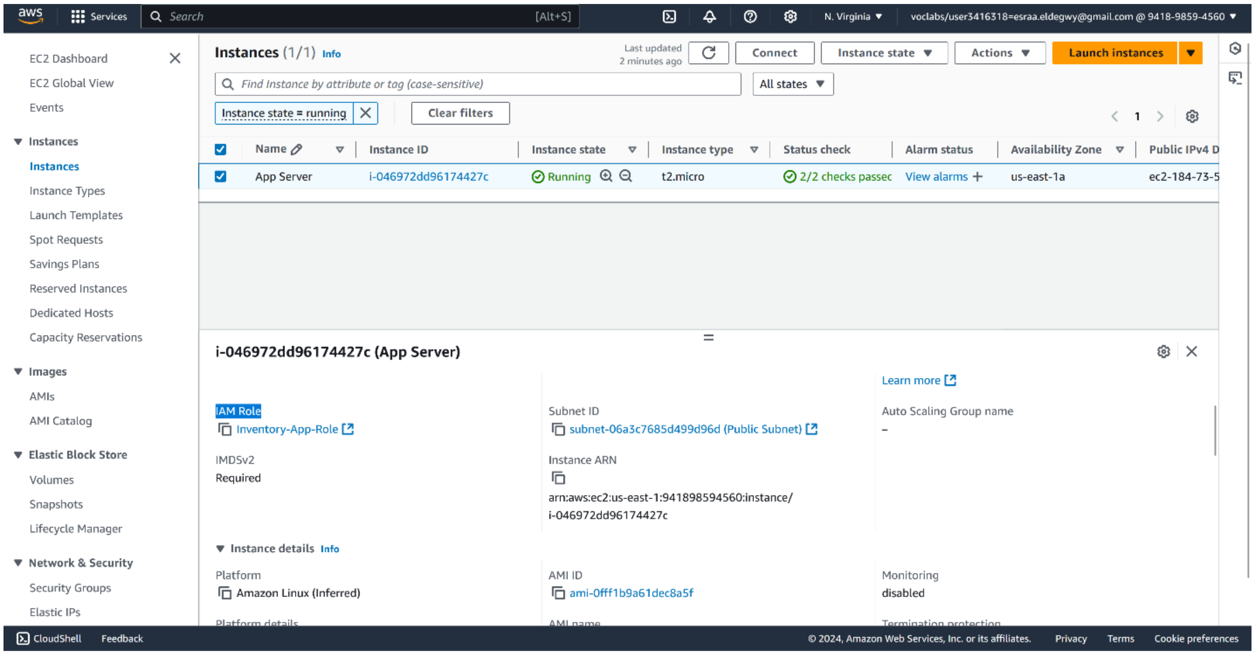
IPv6 address
–

Instance state
Running

Public IPv4 DNS
ec2-184-73-54-149.compute-1.amazonaws.com | open address

Hostname type
IP name: ip-10-0-0-24.ec2.internal

Private IP DNS name (IPv4 only)
ip-10-0-0-24.ec2.internal



Instances (1/1) Info

Last updated 2 minutes ago

Instance state = running

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
☑	App Server	i-046972dd96174427c	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-184-73-5

i-046972dd96174427c (App Server)

Learn more

IAM Role
Inventory-App-Role

Subnet ID
subnet-06a3c7685d499d96d (Public Subnet)

Auto Scaling Group name
–

IMDSv2
Required

Instance ARN
arn:aws:ec2:us-east-1:941898594560:instance/i-046972dd96174427c

▼ Instance details Info

Platform
Amazon Linux (Inferred)

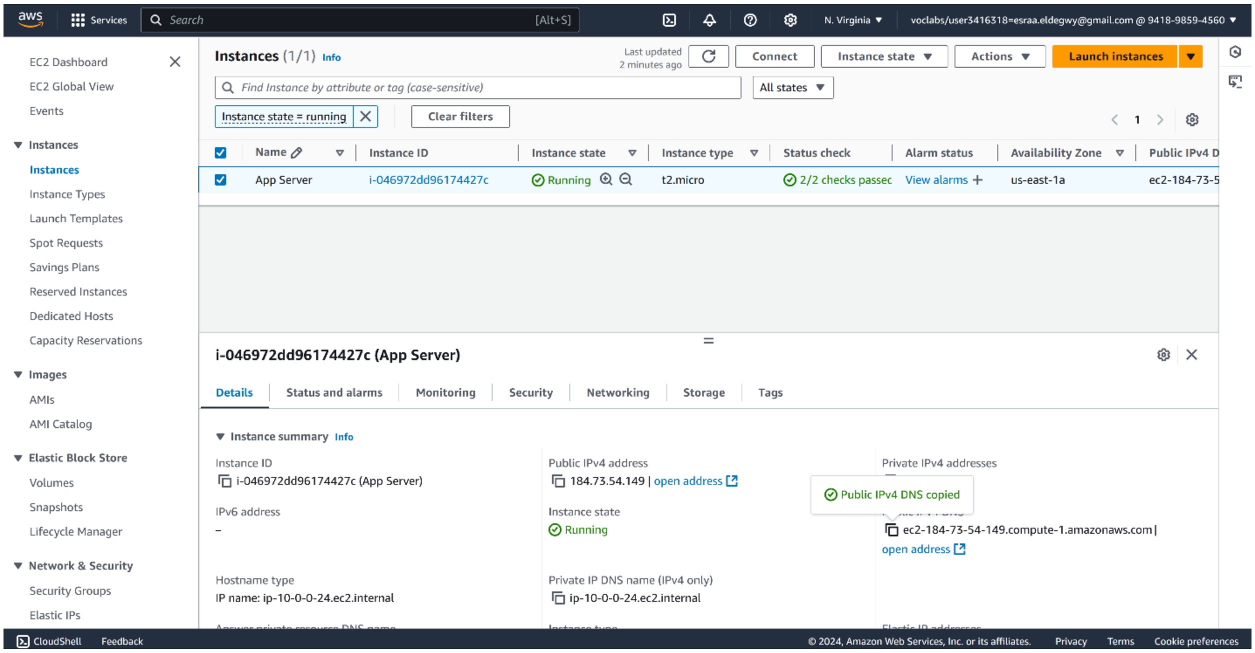
AMI ID
ami-0fff1b9a61dec8a5f

Monitoring
disabled

Platform details

AMI name

Termination protection



Instances (1/1) Info

Last updated 2 minutes ago

Instance state = running

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
☑	App Server	i-046972dd96174427c	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-184-73-5

i-046972dd96174427c (App Server)

▼ Instance summary Info

Instance ID
i-046972dd96174427c (App Server)

Public IPv4 address
184.73.54.149 | open address

Private IPv4 addresses

Public IPv4 DNS copied

IPv6 address
–

Instance state
Running

ec2-184-73-54-149.compute-1.amazonaws.com |
open address

Hostname type
IP name: ip-10-0-0-24.ec2.internal

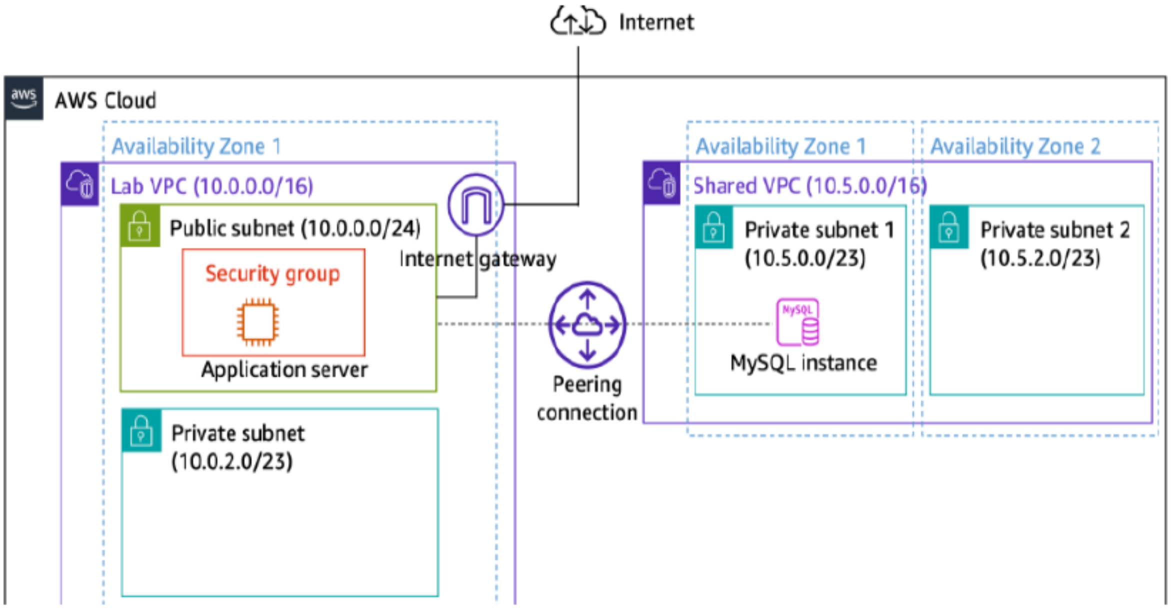
Private IP DNS name (IPv4 only)
ip-10-0-0-24.ec2.internal

5 PROJECT 10: CREATING A VPC PEERING CONNECTION

This project demonstrates how to create a private VPC peering connection between two VPCs. VPC peering allows secure and private communication between resources in different VPCs. By the end of this project, you will have established a peering connection, configured routing, enabled flow logs, and tested the connection.

Project Tasks:

- Create a VPC peering connection between two VPCs.
- Configure route tables to utilize the VPC peering connection.
- Enable VPC Flow Logs to monitor network traffic.
- Test the VPC peering connection.
- Analyze VPC flow logs for traffic insights.



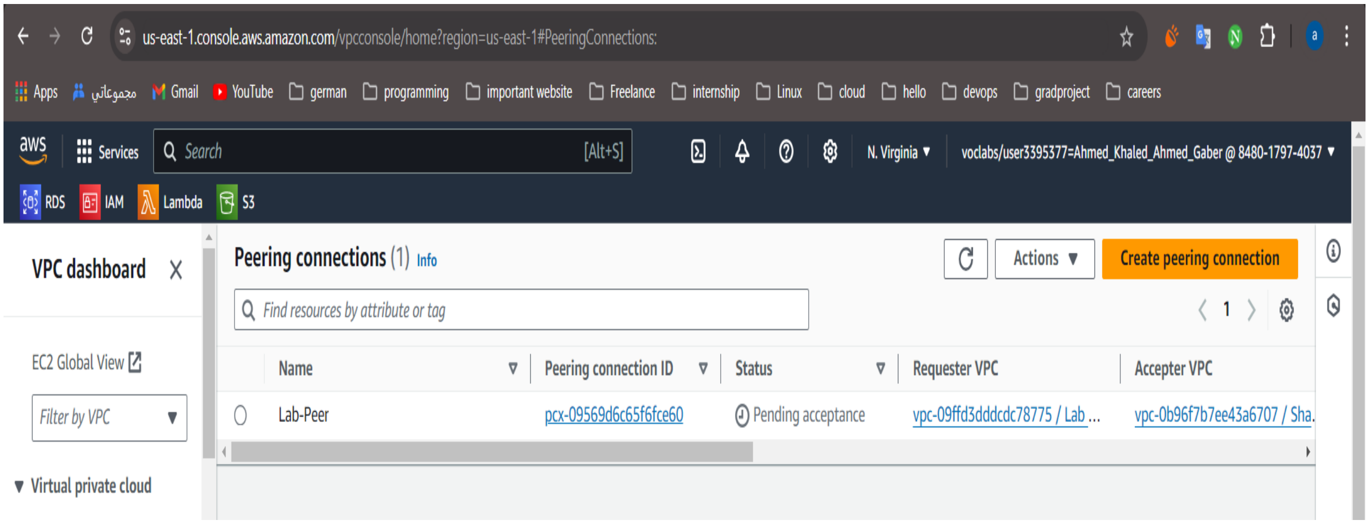
Task 1: Creating a VPC Peering Connection

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account. The VPCs can be in different Regions (also known as an inter-Region VPC peering connection).
Create a new VPC peering connection with the following settings:

- **Name**: Lab-Peer
- **Requester VPC**: Lab VPC
- **Accepter VPC**: Shared VPC
- After creating the peering connection, accept the request to establish the connection.

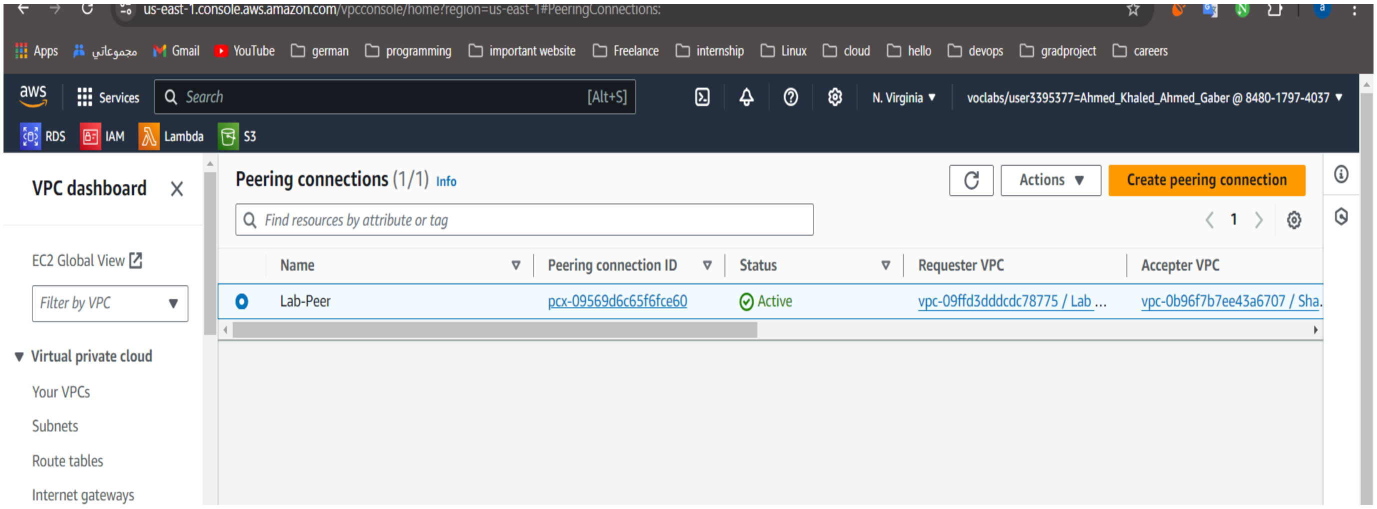
Commands used on AWS CLI:

```
aws ec2 create-vpc-peering-connection\
    --vpc-id vpc-09ffd3dddcdc78775\
   --peer-vpc-id vpc-0b96f7b7ee43a6707\
    --tag-specifications 'ResourceType=vpc-peering-connection,Tags=[{Key=Name,Value=Lab-
Peer}]'


aws ec2 accept-vpc-peering-connection \
   --vpc-peering-connection-id pcx-09569d6c65f6fce60
```



Peering connections (1)

	Name	Peering connection ID	Status	Requester VPC	Accepter VPC
○	Lab-Peer	pcx-09569d6c65f6fce60	Pending acceptance	vpc-09ffd3dddcdc78775 / Lab ...	vpc-0b96f7b7ee43a6707 / Sha.


us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#PeeringConnections:

Peering connections (1/1)

	Name	Peering connection ID	Status	Requester VPC	Accepter VPC
●	Lab-Peer	pcx-09569d6c65f6fce60	✓ Active	vpc-09ffd3dddcdc78775 / Lab ...	vpc-0b96f7b7ee43a6707 / Sha.

Task 2: Configuring Route Tables

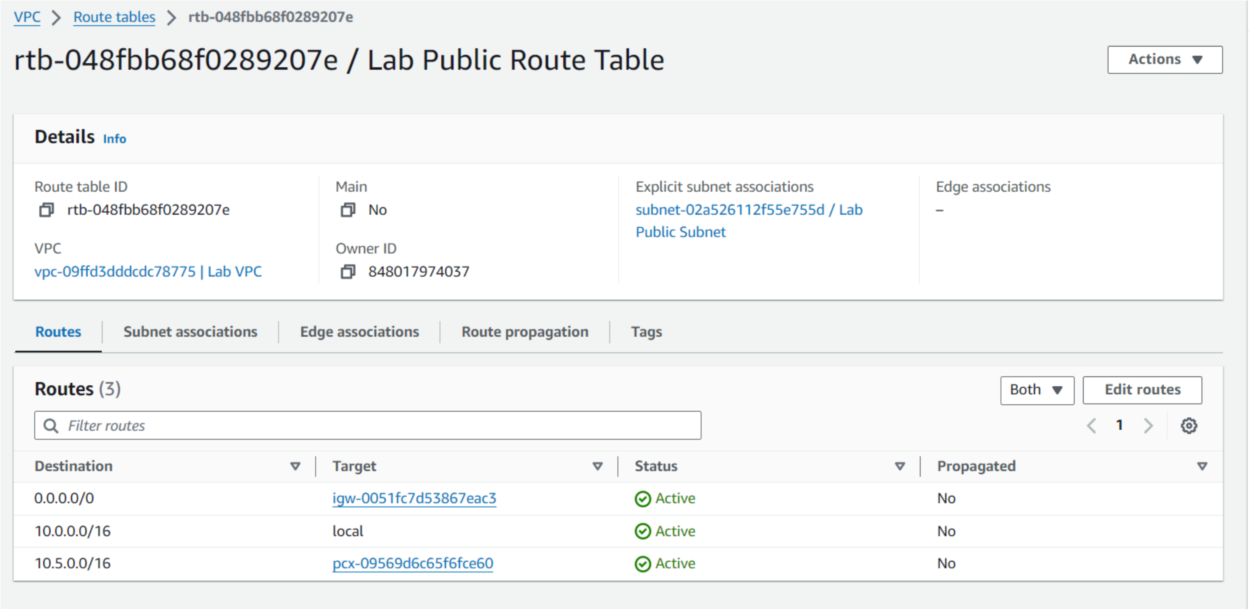
Update the route tables of both VPCs to route traffic through the peering connection:

- For **Lab VPC**: Add a route that directs traffic destined for 10.5.0.0/16 (Shared
- VPC’s CIDR block) to the Lab-Peer peering connection.
- For **Shared VPC**: Add a route that directs traffic destined for 10.0.0.0/16 (Lab
- VPC’s CIDR block) to the Lab-Peer peering connection.

Commands used on AWS CLI:

```
aws ec2 create-route \
   --route-table-id rtb-048fbb68f0289207e\
   --destination-cidr-block 10.5.0.0/16 \
   --vpc-peering-connection-id pcx-09569d6c65f6fce60
```

```
aws ec2 create-route \
   --route-table-id rtb-03a8a34851c54b5f5\
   --destination-cidr-block 10.0.0.0/16 \
   --vpc-peering-connection-id pcx-09569d6c65f6fce60
```


VPC > Route tables > rtb-048fbb68f0289207e
Actions ▼
Routes | Subnet associations | Edge associations | Route propagation | Tags
Both ▼
Edit routes
Filter routes
1

rtb-048fbb68f0289207e / Lab Public Route Table

Details Info

Route table ID
rtb-048fbb68f0289207e

Main
No

Explicit subnet associations
subnet-02a526112f55e755d / Lab Public Subnet

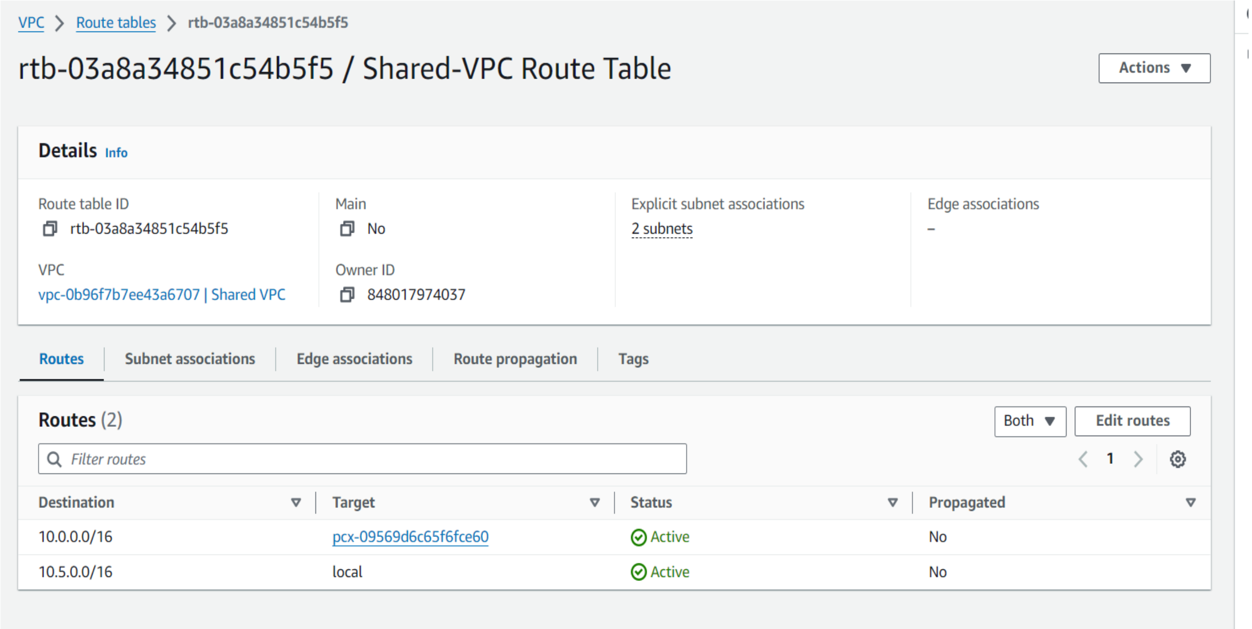
Edge associations
–

VPC
vpc-09ffd3dddcdc78775 | Lab VPC

Owner ID
848017974037

Routes (3)

Destination	Target	Status	Propagated
0.0.0.0/0	igw-0051fc7d53867eac3	✓ Active	No
10.0.0.0/16	local	✓ Active	No
10.5.0.0/16	pcx-09569d6c65f6fce60	✓ Active	No


VPC > Route tables > rtb-03a8a34851c54b5f5
Actions ▼
Routes | Subnet associations | Edge associations | Route propagation | Tags
Both ▼
Edit routes
Filter routes
1

rtb-03a8a34851c54b5f5 / Shared-VPC Route Table

Details Info

Route table ID
rtb-03a8a34851c54b5f5

Main
No

Explicit subnet associations
2 subnets

Edge associations
–

VPC
vpc-0b96f7b7ee43a6707 | Shared VPC

Owner ID
848017974037

Routes (2)

Destination	Target	Status	Propagated
10.0.0.0/16	pcx-09569d6c65f6fce60	✓ Active	No
10.5.0.0/16	local	✓ Active	No

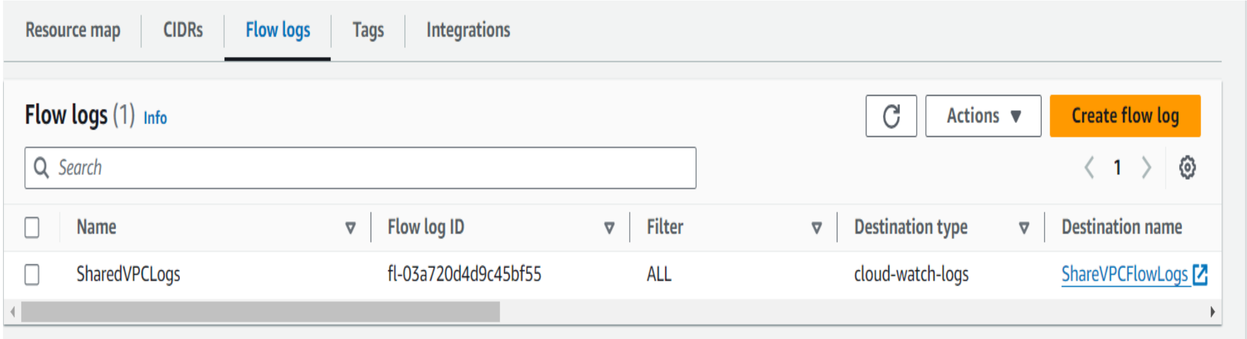
Task 3: Enabling VPC Flow Logs

Enable VPC Flow Logs to monitor traffic moving across the peered VPCs:
Navigate to **Shared VPC**, select **Flow Logs**, and create a flow log with the following settings:

- **Name**: SharedVPCLogs
- **Destination**: Send to CloudWatch Logs
- **Log Group**: ShareVPCFlowLogs
- **IAM Role**: vpc-flow-logs-Role

Commands used on AWS CLI:

```
aws ec2 create-flow-logs \
   --resource-type VPC \
   --resource-id vpc-0b96f7b7ee43a6707\
   --traffic-type ALL \
   --log-destination-type cloud-watch-logs \
   --log-group-name ShareVPCFlowLogs \
   --deliver-logs-permission-arn arn:aws:iam::848017974037:role/Vpc-flow-logs-Role\
   --max-aggregation-interval 60 \


aws ec2 create-tags \
   --resources fl-03a720d4d9c45bf55\
   --tags Key=Name,Value=SharedVPCLogs
```



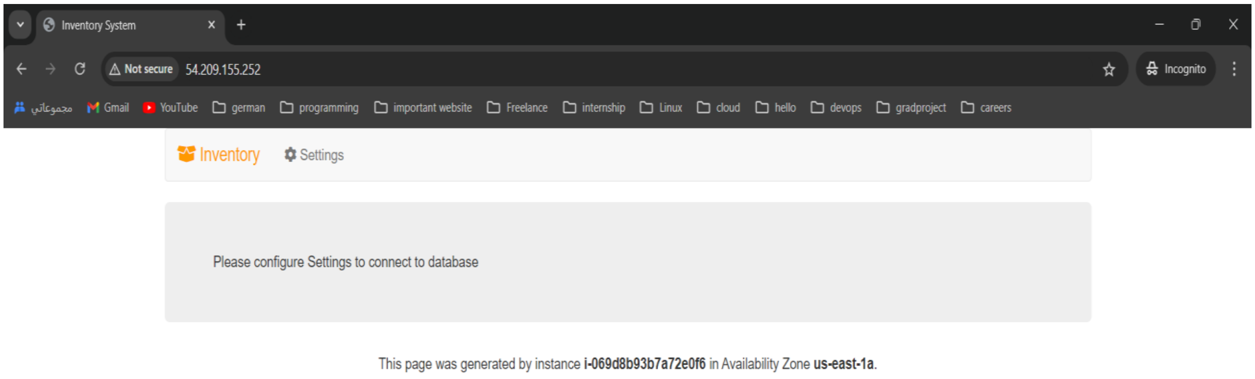
Flow logs (1) Info

□	Name	Flow log ID	Filter	Destination type	Destination name
□	SharedVPCLogs	fl-03a720d4d9c45bf55	ALL	cloud-watch-logs	ShareVPCFlowLogs

Task 4: Testing the VPC Peering Connection

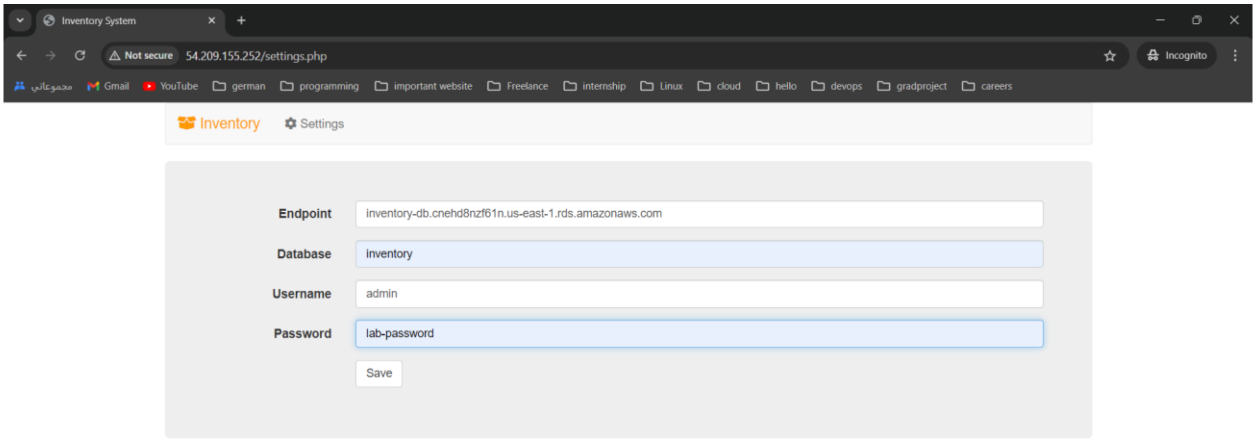
- Test the VPC peering connection by configuring the inventory application to connect to the database in the Shared VPC.
- Access the application using the public IP of the EC2 instance in Lab VPC, then input the database connection details (endpoint, database name, username, and password).
- Verify that the application successfully connects to the database, indicating that the peering connection is functioning correctly.

Connected through the public ip of ec2



Please configure Settings to connect to database

This page was generated by instance **i-069d8b93b7a72e0f6** in Availability Zone **us-east-1a**.



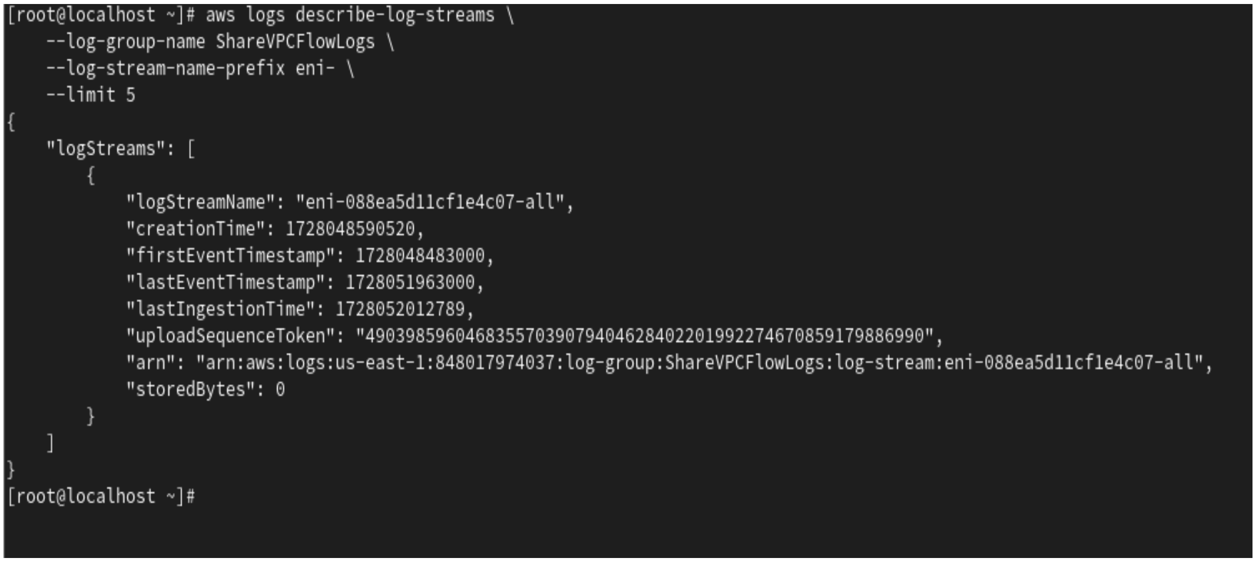
Endpoint	inventory-db.cnehd8nzf61n.us-east-1.rds.amazonaws.com
Database	inventory
Username	admin
Password	lab-password

Task 5: Analyzing the VPC Flow Logs

- Analyze the flow logs in CloudWatch to observe traffic patterns between the application and database.
- Look for log entries with port 3306, representing traffic between the application server and the database.

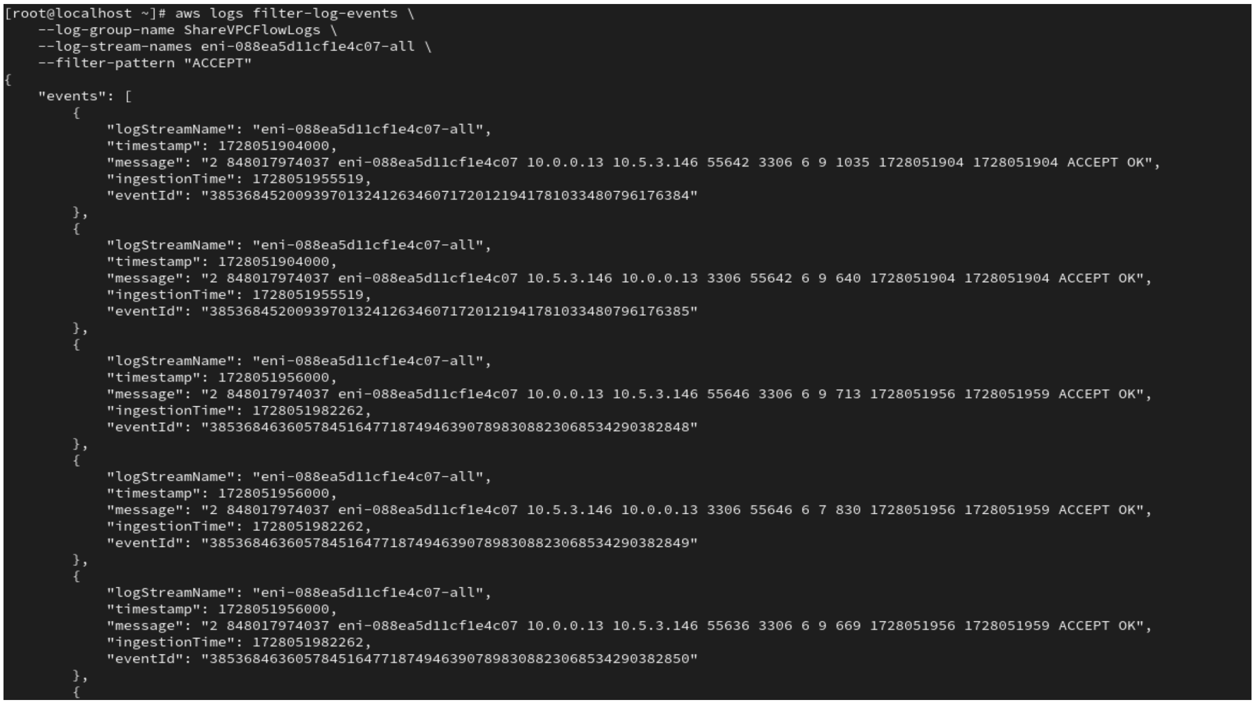
Commands used on AWS CLI:

To know how many enivironment



```
[root@localhost ~]# aws logs describe-log-streams \
    --log-group-name ShareVPCFlowLogs \
    --log-stream-name-prefix eni- \
    --limit 5
{
    "logStreams": [
        {
            "logStreamName": "eni-088ea5d11cf1e4c07-all",
            "creationTime": 1728048590520,
            "firstEventTimestamp": 1728048483000,
            "lastEventTimestamp": 1728051963000,
            "lastIngestionTime": 1728052012789,
            "uploadSequenceToken": "49039859604683557039079404628402201992274670859179886990",
            "arn": "arn:aws:logs:us-east-1:848017974037:log-group:ShareVPCFlowLogs:log-stream:eni-088ea5d11cf1e4c07-all",
            "storedBytes": 0
        }
    ]
}
[root@localhost ~]#
```

Filtered logs on the accepted entries on port 3306 ,it represents traffic



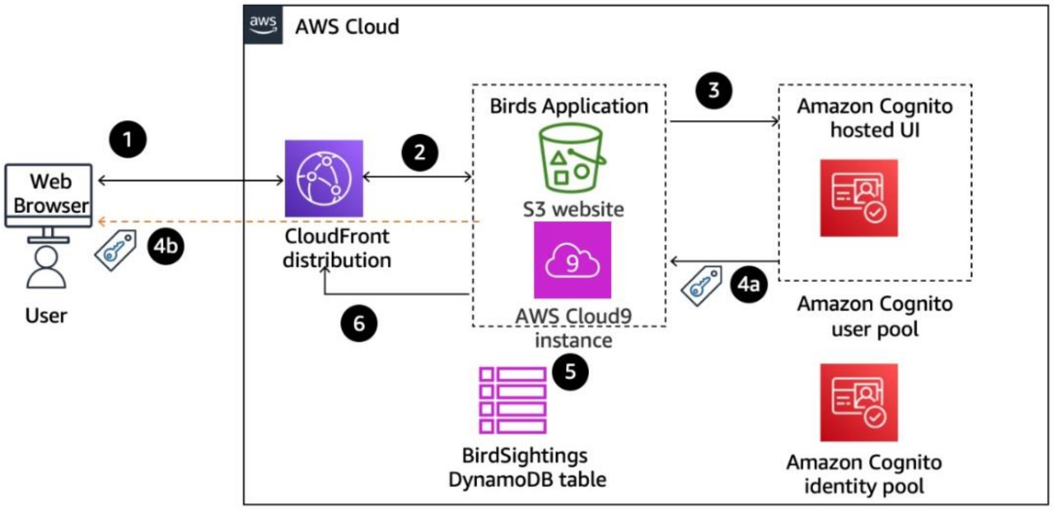
```
[root@localhost ~]# aws logs filter-log-events \
    --log-group-name ShareVPCFlowLogs \
    --log-stream-names eni-088ea5d11cf1e4c07-all \
    --filter-pattern "ACCEPT"
{
    "events": [
        {
            "logStreamName": "eni-088ea5d11cf1e4c07-all",
            "timestamp": 1728051904000,
            "message": "2 848017974037 eni-088ea5d11cf1e4c07 10.0.0.13 10.5.3.146 55642 3306 6 9 1035 1728051904 1728051904 ACCEPT OK",
            "ingestionTime": 1728051955519,
            "eventId": "38536845200939701324126346071720121941781033480796176384"
        },
        {
            "logStreamName": "eni-088ea5d11cf1e4c07-all",
            "timestamp": 1728051904000,
            "message": "2 848017974037 eni-088ea5d11cf1e4c07 10.5.3.146 10.0.0.13 3306 55642 6 9 640 1728051904 1728051904 ACCEPT OK",
            "ingestionTime": 1728051955519,
            "eventId": "38536845200939701324126346071720121941781033480796176385"
        },
        {
            "logStreamName": "eni-088ea5d11cf1e4c07-all",
            "timestamp": 1728051956000,
            "message": "2 848017974037 eni-088ea5d11cf1e4c07 10.0.0.13 10.5.3.146 55646 3306 6 9 713 1728051956 1728051959 ACCEPT OK",
            "ingestionTime": 1728051982262,
            "eventId": "38536846360578451647718749463907898308823068534290382848"
        },
        {
            "logStreamName": "eni-088ea5d11cf1e4c07-all",
            "timestamp": 1728051956000,
            "message": "2 848017974037 eni-088ea5d11cf1e4c07 10.5.3.146 10.0.0.13 3306 55646 6 7 830 1728051956 1728051959 ACCEPT OK",
            "ingestionTime": 1728051982262,
            "eventId": "38536846360578451647718749463907898308823068534290382849"
        },
        {
            "logStreamName": "eni-088ea5d11cf1e4c07-all",
            "timestamp": 1728051956000,
            "message": "2 848017974037 eni-088ea5d11cf1e4c07 10.0.0.13 10.5.3.146 55636 3306 6 9 669 1728051956 1728051959 ACCEPT OK",
            "ingestionTime": 1728051982262,
            "eventId": "38536846360578451647718749463907898308823068534290382850"
        },
        {
```

6 PROJECT 11: SECURING APPLICATIONS USING AMAZON COGNITO

This project will guide you through securing a web application using Amazon Cognito for authentication and authorization. You will configure Amazon Cognito user and identity pools to manage user access and secure API calls to AWS services like Amazon DynamoDB.

Project Tasks:

- Create an Amazon Cognito user pool and manage users.
- Update a web application to use the user pool for authentication.
- Configure an Amazon Cognito identity pool for authorization.
- Update the application to use the identity pool for secure access to AWS services.



Task 1: Preparing the Lab Environment

- Download and set up the Birds web application in AWS Cloud9 using the provided setup script.
- Note down key information like the S3 bucket name and CloudFront distribution domain for later use.

Notes:

- S3 bucket: c132429a3358550l7868655t1w838031074636-s3bucket-kyrcrcexbbuu
- CloudFront distribution domain: d8elk35v953vc.cloudfront.net
- User pool ID: us-east-1_q5bQkndNw
- App client ID: 73a96gujc3bhcndv8a5r9t835b
- Amazon Cognito domain prefix: https://nada.auth.us-east-1.amazoncognito.com
- Identity pool ID: us-east-1:f3b16a03-7439-43e1-a64f-bc4d860bb3cb

Run the following commands to retrieve the code and install the Birds application that you use in this project.
From this output, copy and paste the following values into your text file. You need these values in later steps:

- Record the S3 bucket name. The value is similar to the following: **c1234567890abcdefghi-s3bucket-123456abcd3**

- Record the CloudFront distribution domain name. The value is similar to the following: **d123456acbdef.cloudfront.net**

In the AWS Cloud 9 IDE, in the explorer window to the left of the terminal, expand the **website/scripts** folder.

Open the **config.js** file.

In the file, replace *<cloudfront-domain>* with the CloudFront distribution domain that you recorded into your text editor.

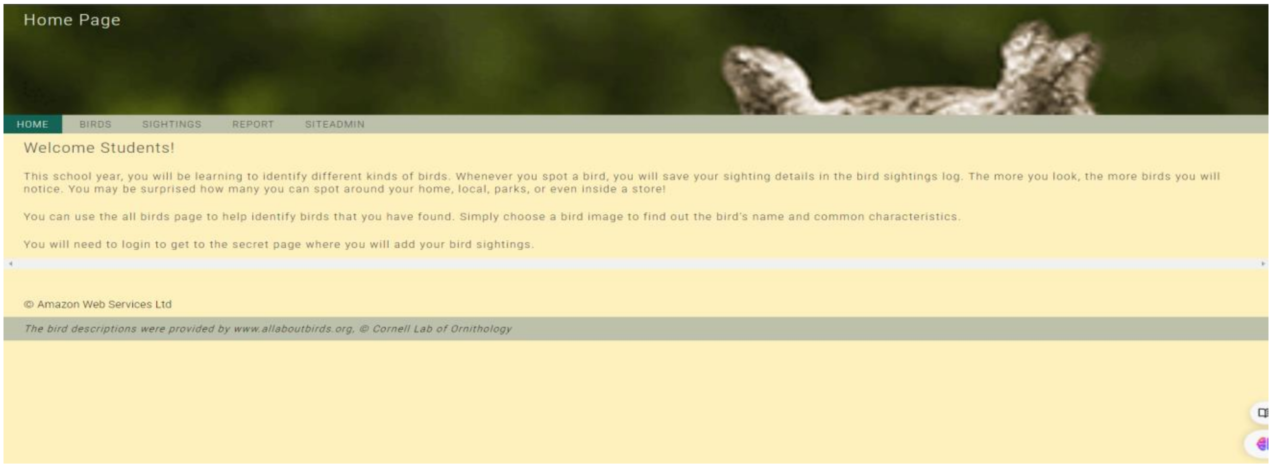
```
cd /home/ec2-user/environment
aws s3 cp website s3://c132429a3358550l7868655t1w838031074636-s3bucket-hlks4qa8sdvo/
--recursive --cache-control "max-age=0"
```

To start the node server, run the following commands:

```
cd /home/ec2-user/environment/node_server
npm start
```

Task 2: Reviewing the Birds Website

Explore the Birds web application to understand its structure and identify protected pages that will require authentication.



Welcome Students!

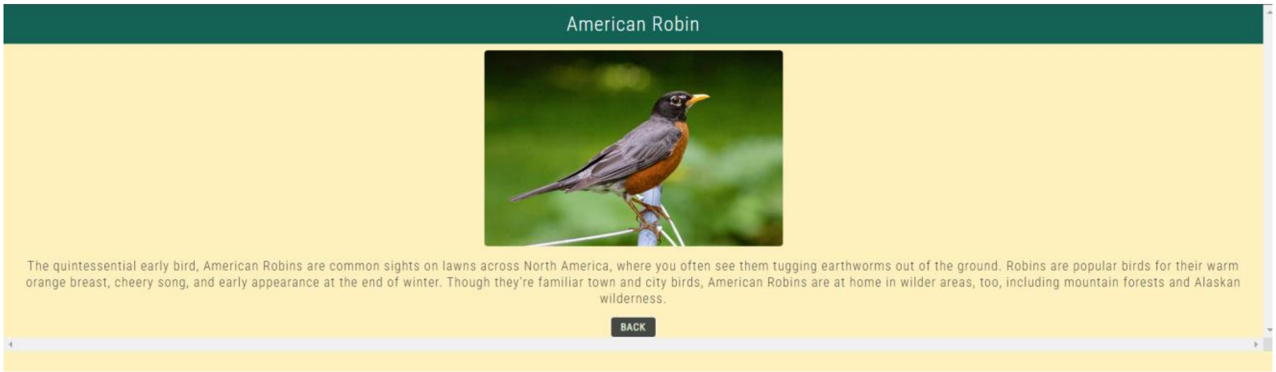
This school year, you will be learning to identify different kinds of birds. Whenever you spot a bird, you will save your sighting details in the bird sightings log. The more you look, the more birds you will notice. You may be surprised how many you can spot around your home, local, parks, or even inside a store!

You can use the all birds page to help identify birds that you have found. Simply choose a bird image to find out the bird's name and common characteristics.

You will need to login to get to the secret page where you will add your bird sightings.

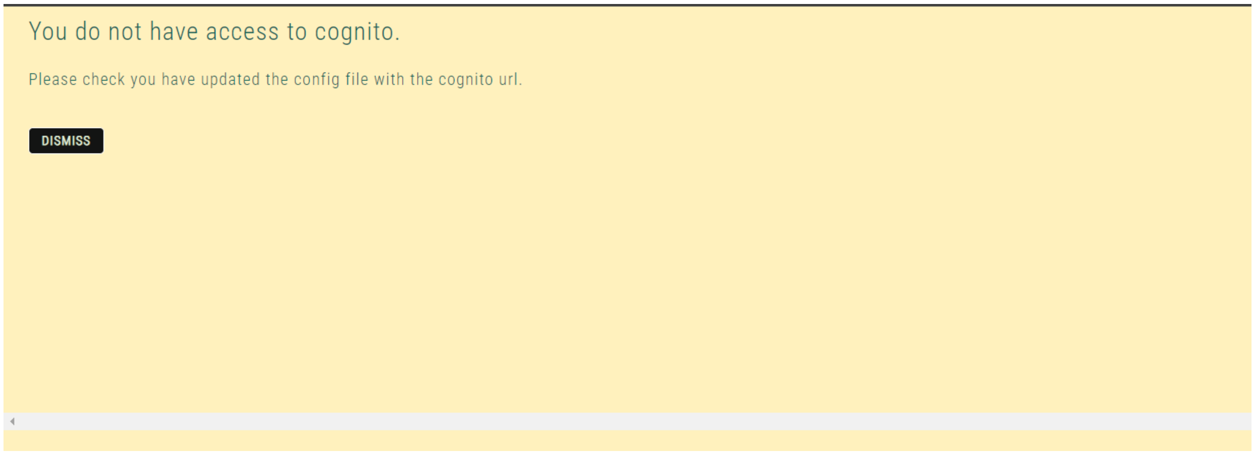
© Amazon Web Services Ltd

The bird descriptions were provided by www.allaboutbirds.org, © Cornell Lab of Ornithology



American Robin

The quintessential early bird, American Robins are common sights on lawns across North America, where you often see them tugging earthworms out of the ground. Robins are popular birds for their warm orange breast, cheery song, and early appearance at the end of winter. Though they're familiar town and city birds, American Robins are at home in wilder areas, too, including mountain forests and Alaskan wilderness.



You do not have access to cognito.

Please check you have updated the config file with the cognito url.

Note:

These pages with bird descriptions are not protected. Anyone can get to this content.

Choose **SIGHTINGS**.

Note: This page requires users to log in before they can see the content.

Task 3: Configuring the Amazon Cognito User Pool

Task 3.1: Creating a User Pool

- Create a user pool named bird_app with custom settings for authentication and session management.
- Configure the hosted UI, app client, and set callback URLs to integrate with the web application.

Task steps:

On the AWS Management Console, in the search box, enter and choose Cognito to open the Amazon Cognito console.

In the navigation pane on the left, choose **User pools**.

Choose **Create user pool**.

For **Step 1: Configure sign-in experience**, configure the following options:

For **Provider types**, notice that **Cognito user pool** is selected.

For **Cognito user pool sign-in options**, configure the following options:

- Choose **User name**.
- Choose **Allow users to sign in with a preferred user name**.

Note: When creating a user pool, the options recommended are suitable for the lab environment. In real-life applications, you might configure these options based on what your application needs. You can check the **Info** link for each option for more information about the recommended options.

Choose **Next**.

For **Step 2: Configure security requirements**, configure the following options:

- In the **Password policy** section for **Password policy mode**, keep the default selection: **Cognito defaults**.
- In the **Multi-factor authentication** section for **MFA enforcement**, choose **No MFA**.
- In the **User account recovery** section for **Self-service account recovery**, clear **Enable self-service account recovery - Recommended**.

Choose **Next**.

For **Step 3: Configure sign-up experience**, configure the following options:

- In the **Self-service sign-up** section for **Self-registration**, clear **Enable self-registration**.
- In the **Attribute verification and user account confirmation** section for **Cognito-assisted verification and confirmation**, clear **Allow Cognito to automatically send messages to verify and confirm - Recommended**.

Choose **Next**.

For **Step 4: Configure message delivery**, for **Email provider**, choose **Send email with Cognito**.

Choose **Next**.

For **Step 5: Integrate your app**, configure the following options:

- For **User pool name**, enter bird_app.
- For **Hosted authentication pages**, choose **Use the Cognito Hosted UI**.

Note: The Amazon Cognito hosted UI is convenient when integrating your application because it builds hosted sign-up, sign-in, and OAuth 2.0 service endpoints in Amazon Cognito; otherwise, you need to use Amazon Cognito API operations to perform sign-up and sign-in.

- In the **Domain** section for **Domain type**, notice that **Use a Cognito domain** is selected.
- For **Cognito domain**, enter a unique prefix string to create a domain name (for example, labbirdapp).
- Confirm that the message *Available* appears, which confirms that the domain name is available for use.

Note: If the domain is not available, try different combinations to choose a unique name.

- Copy and paste your unique domain name to your text editor as the **Amazon Cognito domain prefix** for use later.
- In the **Initial app client** section for **App type**, leave the default option: **Public client**.
- For **App client name**, enter bird_app_client.
- For **Allowed callback URLs**, enter https://<cloudfront-domain>/callback.html, and replace <*cloudfront-domain*> with the CloudFront distribution domain from your text editor.

Note: The updated URL should be similar to the following:
https://d123456acbdef.cloudfront.net/callback.html

Expand **Advanced app client settings**, and configure the following options:

- For **Authentication flows**, from the dropdown list, choose **ALLOW_USER_PASSWORD_AUTH**, and clear **ALLOW_USER_SRP_AUTH**.

- For **OAuth 2.0 grant types**, ensure that **Authorization code grant** is chosen. From the dropdown list, choose **Implicit grant**.
- For **OpenID Connect scopes**, ensure that **Email** and **OpenID** are chosen, and clear **Phone**.

Choose **Next**.

For **Step 6: Review and create**, review the settings to ensure that they're correct, and choose **Create user pool**.

Task 3.2: Adding Users to the User Pool

- Add test users (testuser and admin) to the user pool with specified credentials.
- Create an Administrators group and add the admin user to manage role-based access.

Task steps

From the **User pools** list, choose the link to the **bird_app** user pool.

Copy and past the **User pool ID** into your text editor as the **User pool ID** for use later.

In the Amazon Cognito console, on the **Users** tab, choose **Create user**.

Create a user with the following details:

- For **User name**, enter testuser.
- For **Password**, enter Lab-password1$.

Choose **Create user**.

Create another user with the following details:

- For **User name**, enter admin.
- For **Password**, enter Admin123$.

Choose **Create user**.

Note: This user has administrator permissions.

- Choose the **Groups** tab.
- Choose **Create group**.
- On the **Create group** page for **Group name**, enter Administrators.
- Choose **Create group**.
- Choose the **Administrators** group that you created.
- Choose **Add user to group**.
- From the list of users, select **admin**.
- Choose **Add**.

Note: You have added the **admin** user to the **Administrators** group.

To return to the earlier menu, choose the **bird_app** link at the top of the page.

- Choose the **App integration** tab.
- In the **App clients and analytics** section, choose the **bird_app_client** link.
- From the **App client: bird_app_client** page, copy the **Client ID** to your text editor as the **App client ID** for use later.
- On the **App client: bird_app_client** page from the **Hosted UI** section, choose **View Hosted UI**.
- In the new browser tab that opens, enter the username and password that you used to create the **testuser**, and choose **Sign in**.

You are prompted to change your password.

Follow the instructions to change the password for the **testuser**.

After changing the password, you are redirected to the application home page.

Note: This redirection occurs because during the configuration, you provided this page as a callback page.

This was done to verify the configuration of the user pool. You repeat this process later for the **admin** user when you open the application administrator site.

Note: Do not store these credentials in the browser cache if prompted.

Keep this browser tab with the Birds website open.

Adding users to the user pool

From the User pools list, choose the link to the bird_app user pool.

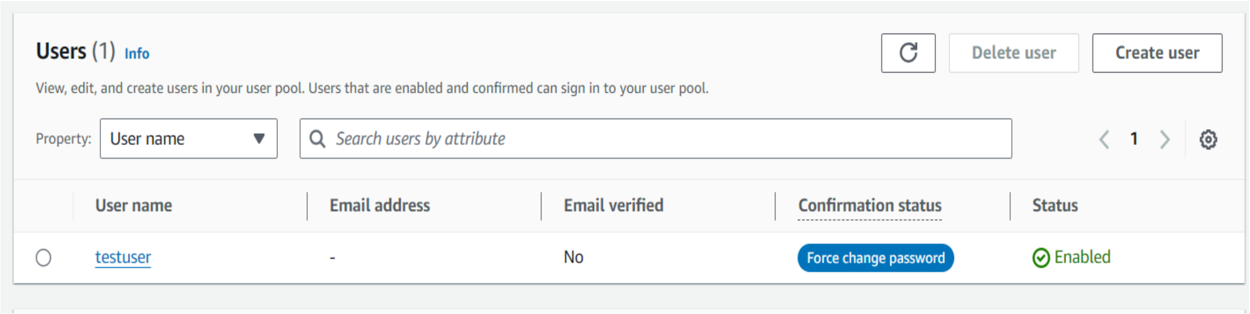
Copy and past the User pool ID into your text editor as the User pool ID for use later.

In the Amazon Cognito console, on the Users tab, choose Create user.

Create a user with the following details:

- For User name, enter testuser.
- For Password, enter Lab-password1$.

Choose Create user.



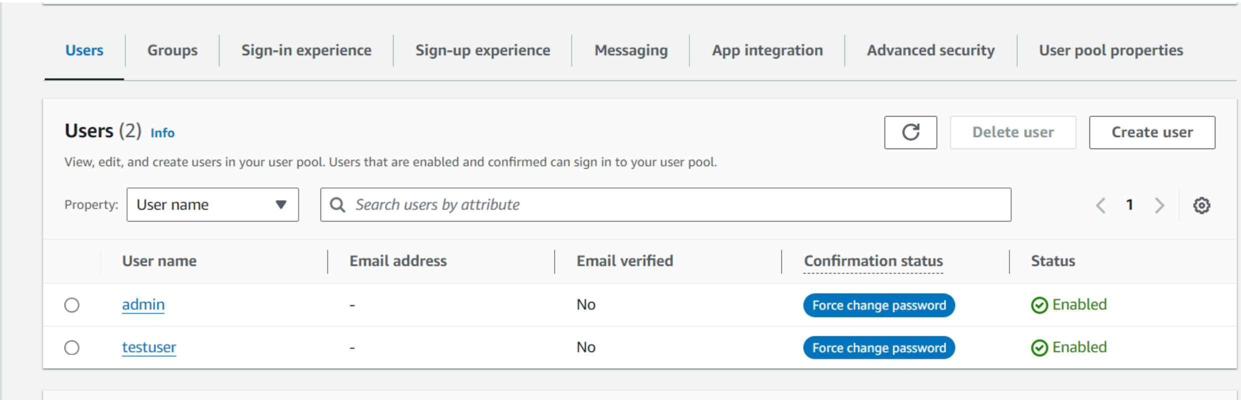
View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.

	User name	Email address	Email verified	Confirmation status	Status
○	testuser	-	No	Force change password	✓ Enabled

Create another user with the following details:

- For User name, enter admin.
- For Password, enter Admin123$.

Choose Create user.



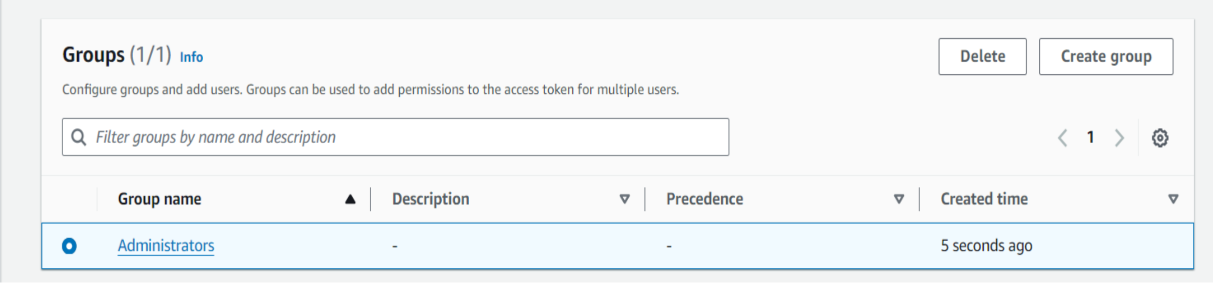
View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.

	User name	Email address	Email verified	Confirmation status	Status
○	admin	-	No	Force change password	✓ Enabled
○	testuser	-	No	Force change password	✓ Enabled

Note: This user has administrator permissions.

- Choose the Groups tab.
- Choose Create group.
- On the Create group page for Group name, enter Administrators.

- Choose Create group.



Groups (1/1) Info

Configure groups and add users. Groups can be used to add permissions to the access token for multiple users.

	Group name	Description	Precedence	Created time
●	Administrators	-	-	5 seconds ago

- Choose the Administrators group that you created.
- Choose Add user to group.
- From the list of users, select admin.
- Choose Add.

Note: You have added the admin user to the Administrators group.

To return to the earlier menu, choose the bird_app link at the top of the page.

- Choose the App integration tab.
- In the App clients and analytics section, choose the bird_app_client link.
- From the App client: bird_app_client page, copy the Client ID to your text editor as the App client ID for use later.
- On the App client: bird_app_client page from the Hosted UI section, choose View Hosted UI.
- In the new browser tab that opens, enter the username and password that you used to create the testuser, and choose Sign in.
- You are prompted to change your password.

Follow the instructions to change the password for the testuser.

After changing the password, you are redirected to the application home page.

Note: This redirection occurs because during the configuration, you provided this page as a callback page.

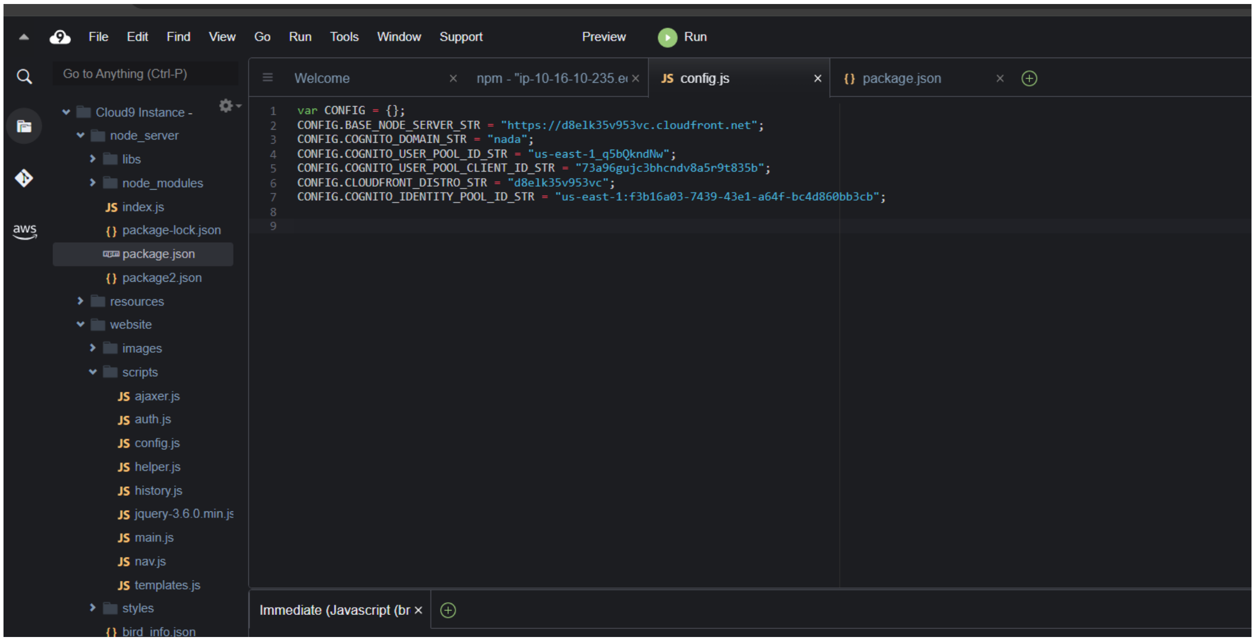
This was done to verify the configuration of the user pool. You repeat this process later for the admin user when you open the application administrator site.

Note: Do not store these credentials in the browser cache if prompted.

Keep this browser tab with the Birds website open.

Task 4: Updating the Application to Use the User Pool for Authentication

- Update the application's configuration files in AWS Cloud9 to integrate the user pool.
- Deploy the updated code to the S3 bucket and restart the node server to apply changes.



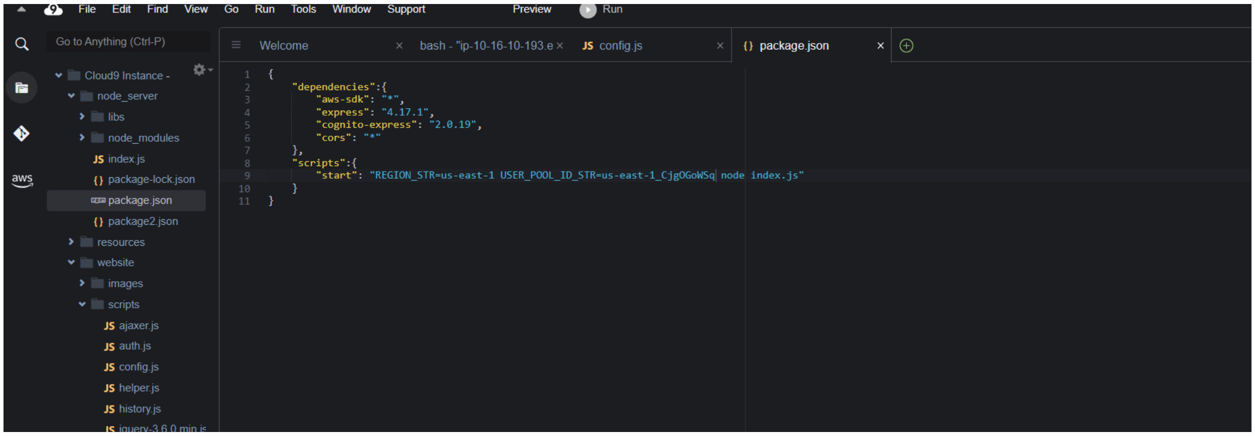
```
var CONFIG = {};
CONFIG.BASE_NODE_SERVER_STR = "https://d8elk35v953vc.cloudfront.net";
CONFIG.COGNITO_DOMAIN_STR = "nada";
CONFIG.COGNITO_USER_POOL_ID_STR = "us-east-1_q5bQkndNw";
CONFIG.COGNITO_USER_POOL_CLIENT_ID_STR = "73a96gujc3bhcndv8a5r9t835b";
CONFIG.CLOUDFRONT_DISTRO_STR = "d8elk35v953vc";
CONFIG.COGNITO_IDENTITY_POOL_ID_STR = "us-east-1:f3b16a03-7439-43e1-a64f-bc4d860bb3cb";
```

Command used:

```
cd /home/ec2-user/environment

aws s3 cp website s3:// c132429a3358550l7868655t1w838031074636-s3bucket-ugmd7wyrbbow
/ --recursive --cache-control "max-age=0"
```

These commands update the JavaScript packages required for the application to use the Amazon Cognito user pool, along with other configuration changes made earlier.

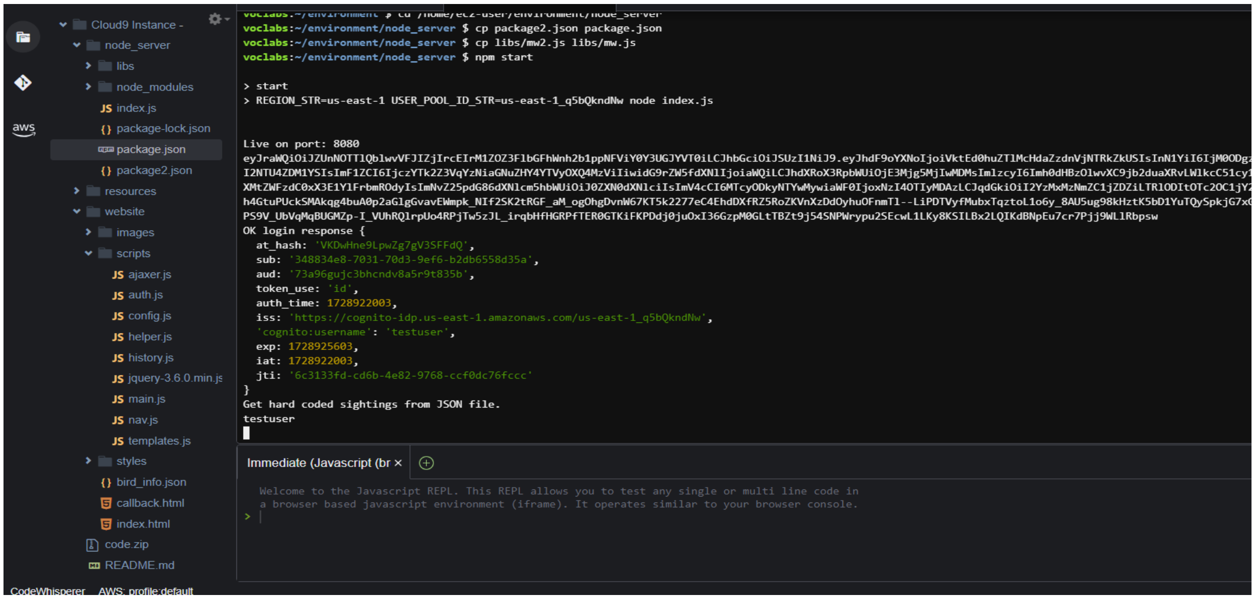
```
cd /home/ec2-user/environment/node_server

cp package2.json package.json

cp libs/mw2.js libs/mw.js
```



```
{
    "dependencies":{
        "aws-sdk": "*",
        "express": "4.17.1",
        "cognito-express": "2.0.19",
        "cors": "*"
    },
    "scripts":{
        "start": "REGION_STR=us-east-1 USER_POOL_ID_STR=us-east-1_CjgOGoWSq node index.js"
    }
}
```

Task 5: Testing the User Pool Integration with the Application

- Test the updated application by logging in with the testuser credentials to access protected pages.
- Verify that only authenticated users can access protected content.

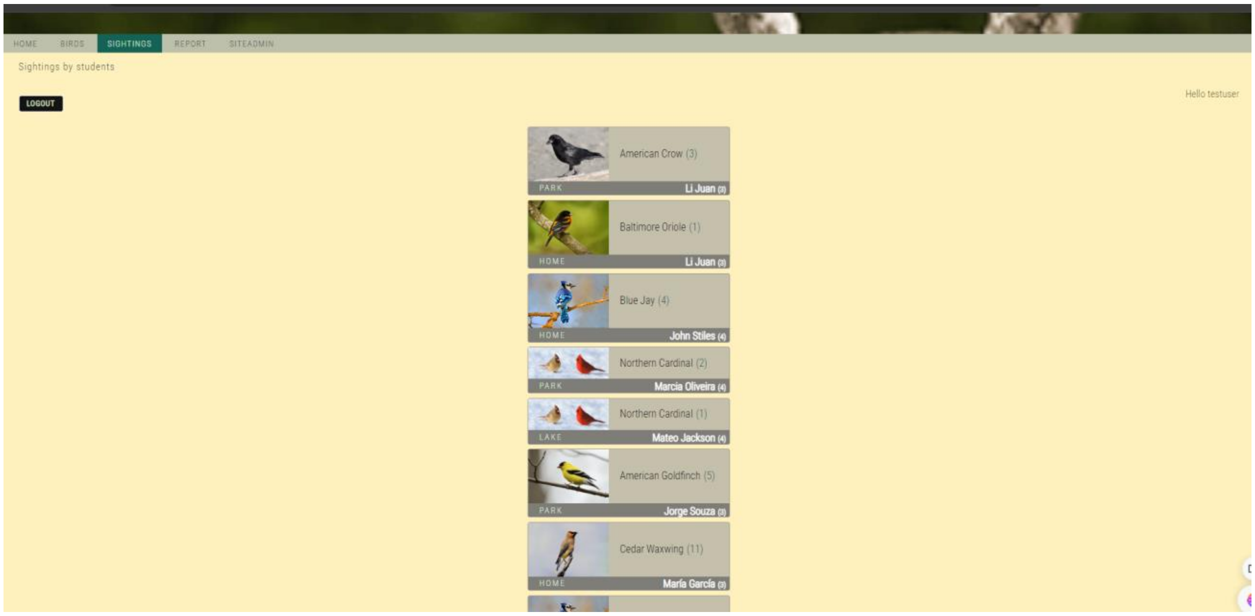


```
voclabs:~/environment $ cd /home/ec2-user/environment/node_server
voclabs:~/environment/node_server $ cp package2.json package.json
voclabs:~/environment/node_server $ cp libs/mw2.js libs/mw.js
voclabs:~/environment/node_server $ npm start

> start
> REGION_STR=us-east-1 USER_POOL_ID_STR=us-east-1_q5bQkndNw node index.js


Live on port: 8080
eyJraWQiOiJZUnNOTTlQblwvVFJIZjIrcEIrM1ZOZ3FlbGFhWnh2b1ppNFViY0Y3UGJYVT0iLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoiVktEd0huZTlMcHdaZzdnVjNTRkZkUSIsInN1YiI6IjM0ODgz
I2NTU4ZDM1YSIsImF1ZCI6IjczYTk2Z3VqYzNiaGNuZHY4YTVyOXQ4MzViIiwidG9rZW5fdXNlIjoiaWQiLCJhdXRoX3RpbWUiOjE3Mjg5MjIwMDMsImlzcyI6Imh0dHBzOlwvXC9jb2duaXRvLWlkcC51cy1
XMtZWFzdC0xX3E1YlFrbmROdyIsImNvZ25pdG86dXNlcm5hbWUiOiJ0ZXN0dXNlciIsImV4cCI6MTcyODkyNTYwMywiaWF0IjoxNzI4OTIyMDAzLCJqdGkiOiI2YzMxMzNmZC1jZDZiLTRlODItOTc2OC1jY2
h4GtuPUckSMAkqg4buA0p2aGlgGvavEWmpk_NIf2SK2tRGF_aM_ogOhgDvnW67KT5k2277eC4EhdDXfRZ5RoZKVnXzDdOyhuOFnmTl--LiPDTVyfMubxTqztoL1o6y_8AU5ug98kHztK5bD1YuTQySpkjG7x
PS9V_UbVqMqBUGMZp-I_VUhRQlrpUo4RPjTw5zJL_irqbHfHGRPfTER0GTKiFKPDdj0juOxI36GzpM0GLtTBZt9j54SNPWrypu2SEcwL1LKy8KSILBx2LQIKdBNpEu7cr7Pjj9WLlRbpsw
OK login response {
  at_hash: 'VKDwHne9LpwZg7gV3SFFdQ',
  sub: '348834e8-7031-70d3-9ef6-b2db6558d35a',
  aud: '73a96gujc3bhcndv8a5r9t835b',
  token_use: 'id',
  auth_time: 1728922003,
  iss: 'https://cognito-idp.us-east-1.amazonaws.com/us-east-1_q5bQkndNw',
  'cognito:username': 'testuser',
  exp: 1728925603,
  iat: 1728922003,
  jti: '6c3133fd-cd6b-4e82-9768-ccf0dc76fccc'
}
Get hard coded sightings from JSON file.
testuser
```

```
Welcome to the Javascript REPL. This REPL allows you to test any single or multi line code in
a browser based javascript environment (iframe). It operates similar to your browser console.
>
```

To start the node server, run the following command at the prompt:

npm start

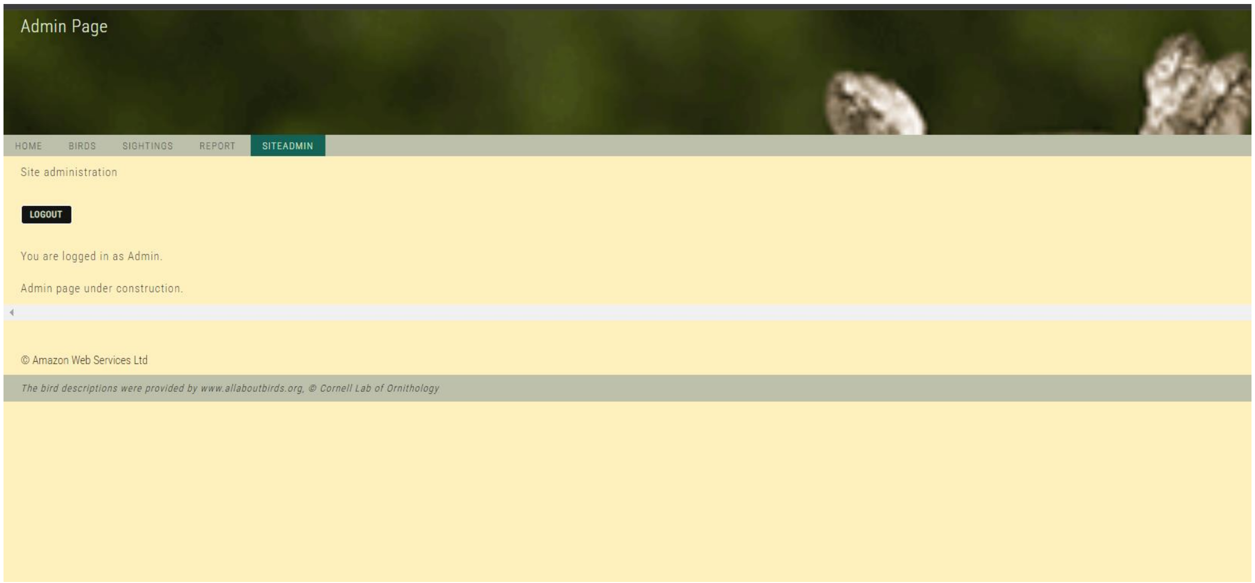


- Choose **LOGIN**.
- On the prompt, provide the credentials for **testuser**.
- After successfully logging in, the birds listing is displayed.

Note: If you are redirected to the **Home** page, navigate to the **SIGHTINGS** page again.

This test proves that only a user who has been authenticated by the Amazon Cognito user pool can access the protected pages of the application. The user pool centrally stores your application usernames and passwords, which makes your users and their authentication information more secure and convenient to manage.

- Choose **SITEADMIN**.
- You see a message stating that you need to log in as an administrator to access the administration page.
- If you see the message *You need Admin credentials to see Admin page*, choose **DISMISS**.
- On the **SITEADMIN** page, choose **ADMIN LOGIN**.
- On the login prompt, choose **Sign in as a different user?**



Admin Page

Site administration

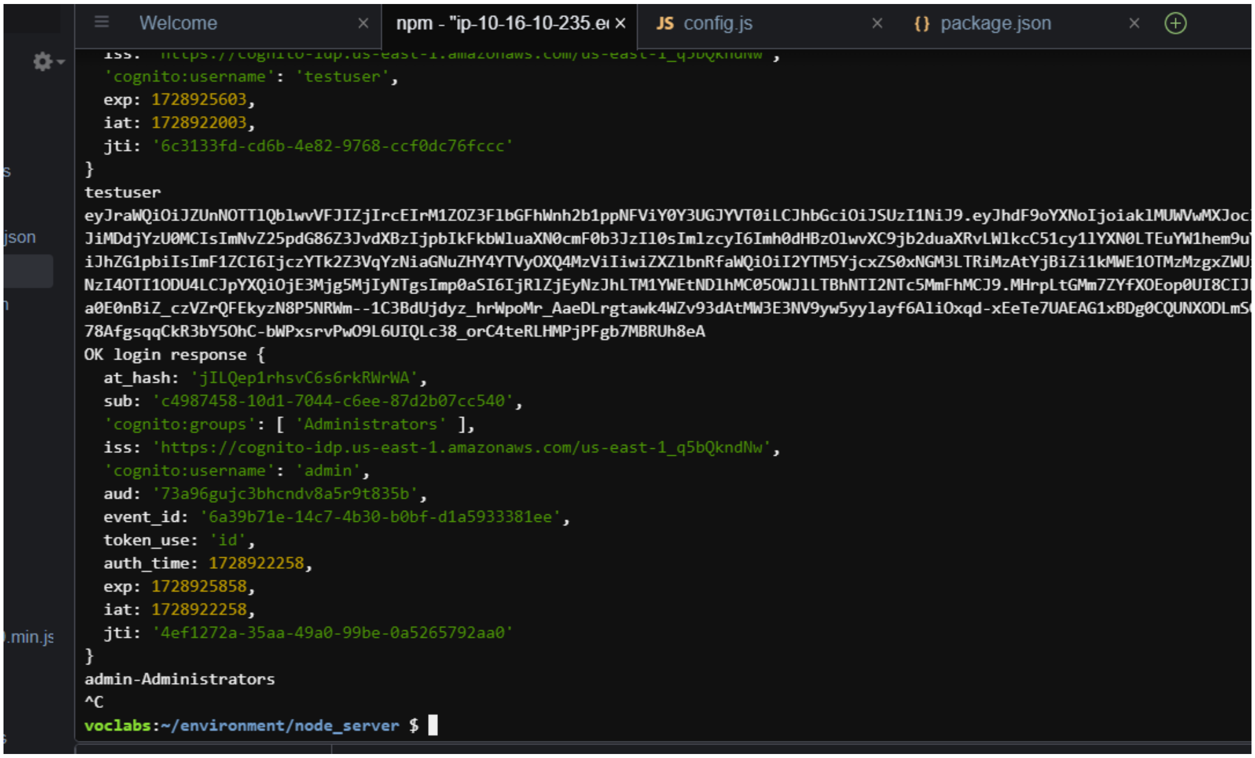
You are logged in as Admin.

Admin page under construction.

© Amazon Web Services Ltd

The bird descriptions were provided by www.allaboutbirds.org, © Cornell Lab of Ornithology

Use the **admin** login credentials to log in.



```
  iss: 'https://cognito-idp.us-east-1.amazonaws.com/us-east-1_q5bQkndNw',
  'cognito:username': 'testuser',
  exp: 1728925603,
  iat: 1728922003,
  jti: '6c3133fd-cd6b-4e82-9768-ccf0dc76fccc'
}
testuser
eyJraWQiOiJZUnNOTTlQblwvVFJIZjIrcEIrM1ZOZ3FlbGFhWnh2b1ppNFViY0Y3UGJYVT0iLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoiaklMUWVwMXJoc
JiMDdjYzU0MCIsImNvZ25pdG86Z3JvdXBzIjpbIkFkbWluaXN0cmF0b3JzIl0sImlzcyI6Imh0dHBzOlwvXC9jb2duaXRvLWlkcC51cy1lYXN0LTEuYW1hem9u
iJhZG1pbiIsImF1ZCI6IjczYTk2Z3VqYzNiaGNuZHY4YTVyOXQ4MzViIiwiZXZlbnRfaWQiOiI2YTM5YjcxZS0xNGM3LTRiMzAtYjBiZi1kMWE1OTMzMzgxZWU
NzI4OTI1ODU4LCJpYXQiOjE3Mjg5MjIyNTgsImp0aSI6IjRlZjEyNzJhLTM1YWEtNDlhMC05OWJlLTBhNTI2NTc5MmFhMCJ9.MHrpLtGMm7ZYfXOEop0UI8CIJ
a0E0nBiZ_czVZrQFEkyzN8P5NRWm--1C3BdUjdyz_hrWpoMr_AaeDLrgtawk4WZv93dAtMW3E3NV9yw5yylayf6AliOxqd-xEeTe7UAEAG1xBDg0CQUNXODLmS
78AfgsqqCkR3bY5OhC-bWPxsrvPwO9L6UIQLc38_orC4teRLHMPjPFgb7MBRUh8eA
OK login response {
  at_hash: 'jILQep1rhsvC6s6rkRWrWA',
  sub: 'c4987458-10d1-7044-c6ee-87d2b07cc540',
  'cognito:groups': [ 'Administrators' ],
  iss: 'https://cognito-idp.us-east-1.amazonaws.com/us-east-1_q5bQkndNw',
  'cognito:username': 'admin',
  aud: '73a96gujc3bhcndv8a5r9t835b',
  event_id: '6a39b71e-14c7-4b30-b0bf-d1a5933381ee',
  token_use: 'id',
  auth_time: 1728922258,
  exp: 1728925858,
  iat: 1728922258,
  jti: '4ef1272a-35aa-49a0-99be-0a5265792aa0'
}
admin-Administrators
^C
voclabs:~/environment/node_server $ 
```

After successfully logging in, the message *Admin page under construction* is displayed.

Note: If you are redirected to the **Home** page, navigate to the **SITEADMIN** page again.

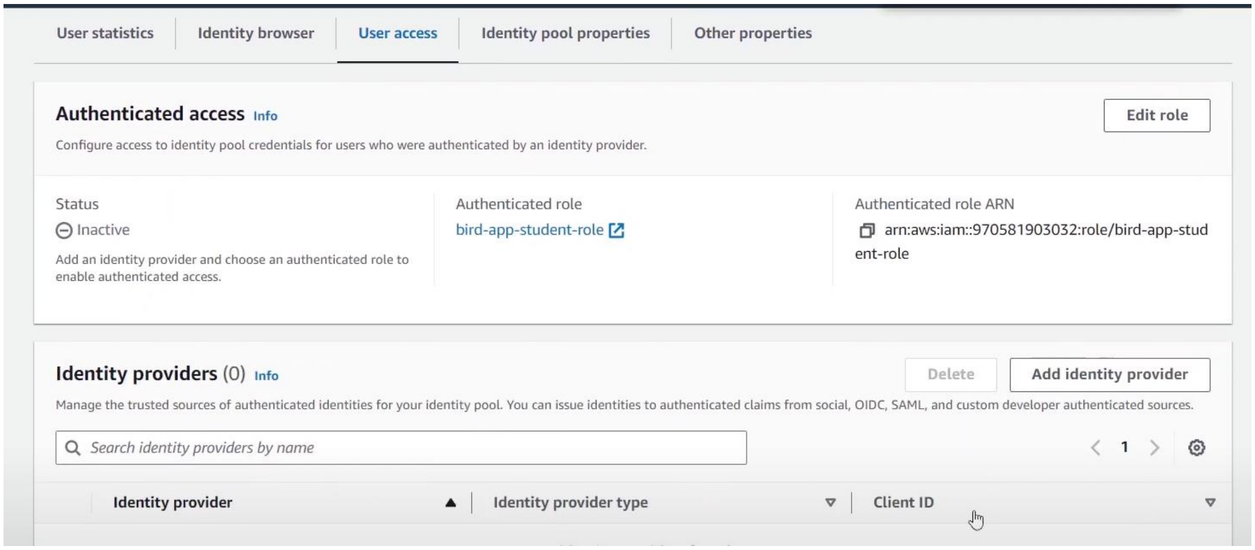
This test proves that the Amazon Cognito user pool can manage role-based access control to your application and provide secure access to the protected pages of the application based on the role.

Task 6: Configuring the Amazon Cognito Identity Pool

- Configure the pre-created Amazon Cognito identity pool to link it with the user pool for secure AWS service access.
- Set the correct user pool ID, app client ID, and configure default roles for authenticated access.

The Amazon Cognito identity pool was created for you when you launched the lab environment. In this task, you configure the Amazon Cognito identity pool to work with the Birds application.

- On the Amazon Cognito console in the navigation pane, choose **Identity pools**.
- Choose the **bird_app_id_pool** link.
- Copy and paste the **Identity pool ID** into your text editor as the **Identity pool ID** for use later.



Authenticated access Info

Configure access to identity pool credentials for users who were authenticated by an identity provider.

Status
⊖ Inactive
Add an identity provider and choose an authenticated role to enable authenticated access.

Authenticated role
bird-app-student-role

Authenticated role ARN
arn:aws:iam::970581903032:role/bird-app-student-role

Identity providers (0) Info

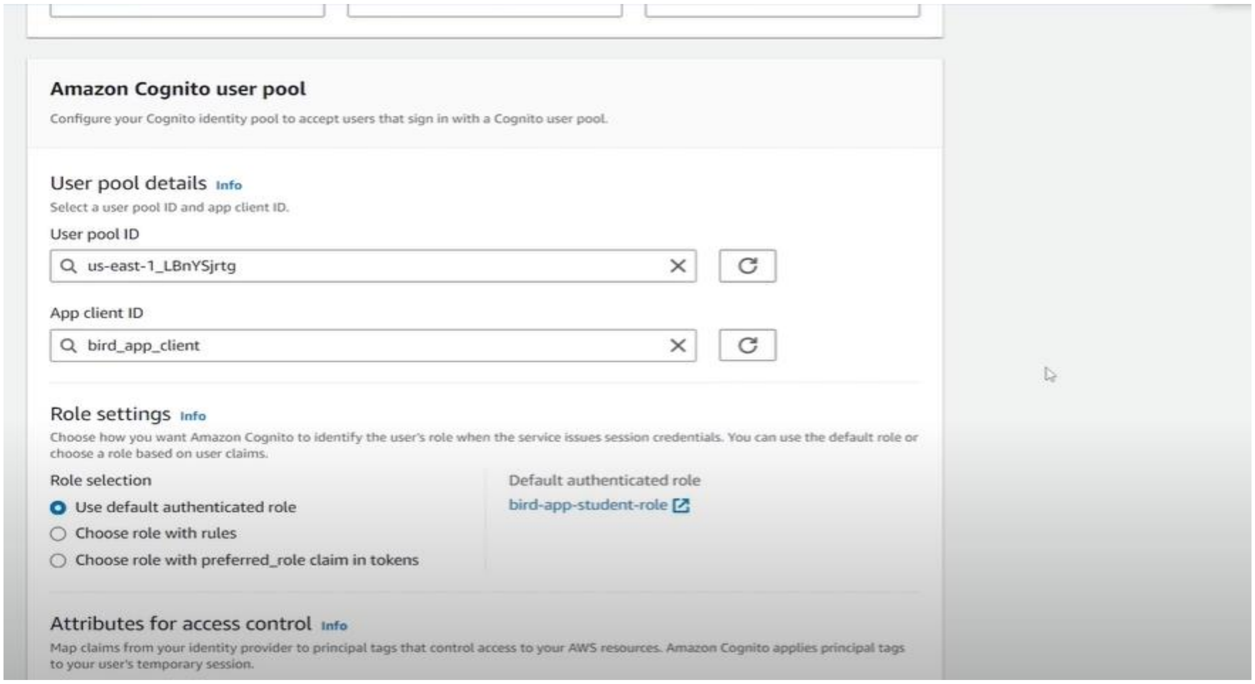
Manage the trusted sources of authenticated identities for your identity pool. You can issue identities to authenticated claims from social, OIDC, SAML, and custom developer authenticated sources.

Identity provider	Identity provider type	Client ID

On the Amazon Cognito console, choose the **User access** tab.

- Choose **Add identity provider**.
- Choose **Amazon Cognito user pool**.
- For **User pool ID**, choose the user pool with the name **bird_app**.
- For **App client ID**, choose the application with the name **bird_app_client**.
- In the **Role settings** section, notice that **Default authenticated role** is displayed.
- Choose **Save changes**.
- Review the **Authenticated access** section.

Notice that the **Authenticated role** has been configured to use the default role, which is assigned to users when they successfully log in. You could set up additional rules to assign different AWS Identity and Access Management (IAM) roles to different users. For this phase of application development, you only keep one role assigned to the users.



Amazon Cognito user pool

Configure your Cognito identity pool to accept users that sign in with a Cognito user pool.

User pool details Info

Select a user pool ID and app client ID.

User pool ID: us-east-1_LBnYSjrtg

App client ID: bird_app_client

Role settings Info

Choose how you want Amazon Cognito to identify the user's role when the service issues session credentials. You can use the default role or choose a role based on user claims.

Role selection

● Use default authenticated role

○ Choose role with rules

○ Choose role with preferred_role claim in tokens

Default authenticated role: bird-app-student-role

Attributes for access control Info

Map claims from your identity provider to principal tags that control access to your AWS resources. Amazon Cognito applies principal tags to your user's temporary session.

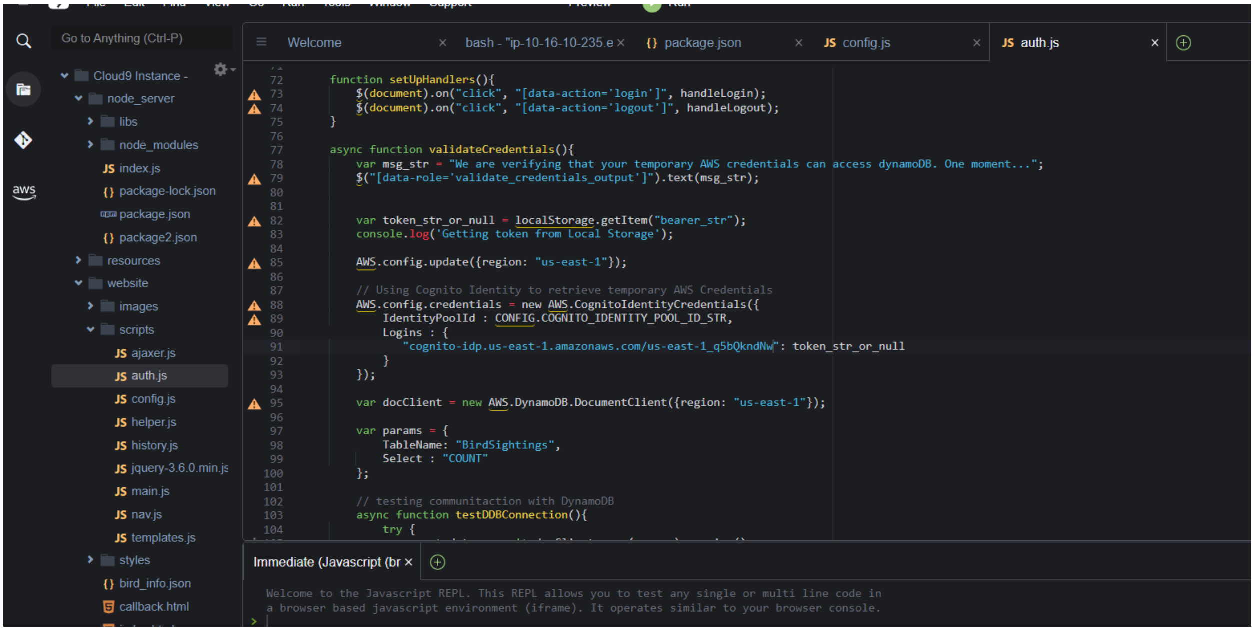
Task 7: Updating the Application to Use the Identity Pool for Authorization

- Update the application configuration to use the identity pool for secure communication with AWS services.

- Modify the config.js and auth.js files to include identity pool information and deploy the updates.

As with the user pool, the application needs to be updated so it can interact with the identity pool. In this task, you make the necessary updates to the Birds application.

- First, you update the Birds web application.
- Return to the browser tab where the AWS Cloud9 IDE is open.
- In the terminal window where the node server is running, press Ctrl+C to stop the NodeJs server:
- Next, you update the config.js file.
- In the Explorer window, expand the **website/scripts** folder.
- Open the **config.js** file.
- Remove the two forward slashes (**//**) from the beginning of the last line of code to uncomment the code, and replace *<cognito-identity-pool-id>* with the identity pool ID that you saved earlier.
- To save the file, choose **File** > **Save**.



```
72      function setUpHandlers(){
73          $(document).on("click", "[data-action='login']", handleLogin);
74          $(document).on("click", "[data-action='logout']", handleLogout);
75      }
76
77      async function validateCredentials(){
78          var msg_str = "We are verifying that your temporary AWS credentials can access dynamoDB. One moment...";
79          $("[data-role='validate_credentials_output']").text(msg_str);
80
81
82          var token_str_or_null = localStorage.getItem("bearer_str");
83          console.log('Getting token from Local Storage');
84
85          AWS.config.update({region: "us-east-1"});
86
87          // Using Cognito Identity to retrieve temporary AWS Credentials
88          AWS.config.credentials = new AWS.CognitoIdentityCredentials({
89              IdentityPoolId : CONFIG.COGNITO_IDENTITY_POOL_ID_STR,
90              Logins : {
91                 "cognito-idp.us-east-1.amazonaws.com/us-east-1_q5bQkndNw": token_str_or_null
92              }
93          });
94
95          var docClient = new AWS.DynamoDB.DocumentClient({region: "us-east-1"});
96
97          var params = {
98              TableName: "BirdSightings",
99              Select : "COUNT"
100         };
101
102         // testing communitaction with DynamoDB
103         async function testDDBConnection(){
104             try {
```

```
Welcome to the Javascript REPL. This REPL allows you to test any single or multi line code in
a browser based javascript environment (iframe). It operates similar to your browser console.
>
```

Next, you update the auth.js file.

From the **website/scripts** folder, open the **auth.js** file.

In the file, replace *<cognito-user-pool-id>* with the user pool ID. The placeholder is on or around line 91 in the file.

Important: Ensure that you use the user pool ID here and not the identity pool ID.

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({

IdentityPoolId : CONFIG.COGNITO_IDENTITY_POOL_ID_STR,

Logins : {

"cognito-idp.us-east-1.amazonaws.com/<cognito-user-pool-id>": token_str_or_null

}

});
```

The updated code is similar to the following:

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({

IdentityPoolId : CONFIG.COGNITO_IDENTITY_POOL_ID_STR,
Logins : {

"cognito-idp.us-east-1.amazonaws.com/us-east-1_AAAA1111": token_str_or_null

}

});
```

Note: This section of code uses the COGNITO_IDENTITY_POOL_ID_STR variable, which you set in the config.js file. The code uses this identifier to request credentials from the identity pool. Notice that this code block also passes the user pool ID and token_str_or_null, which holds the authentication token. The identity pool uses this information to verify the user in the user pool. If the user and token pass the validation, the identity pool sends AWS credentials back to the application.

To save the file, choose **File** > **Save**.

Next, you push the updated website code to the S3 bucket.

In the AWS Cloud9 IDE terminal, enter the following command and replace *<s3-bucket>* with the name of the S3 bucket. Run your revised command.

```
cd /home/ec2-user/environment

aws s3 cp website s3://<s3-bucket>/ --recursive --cache-control "max-age=0"
```

Ensure that the node server is still running. If it is not, run the following commands to start it again:

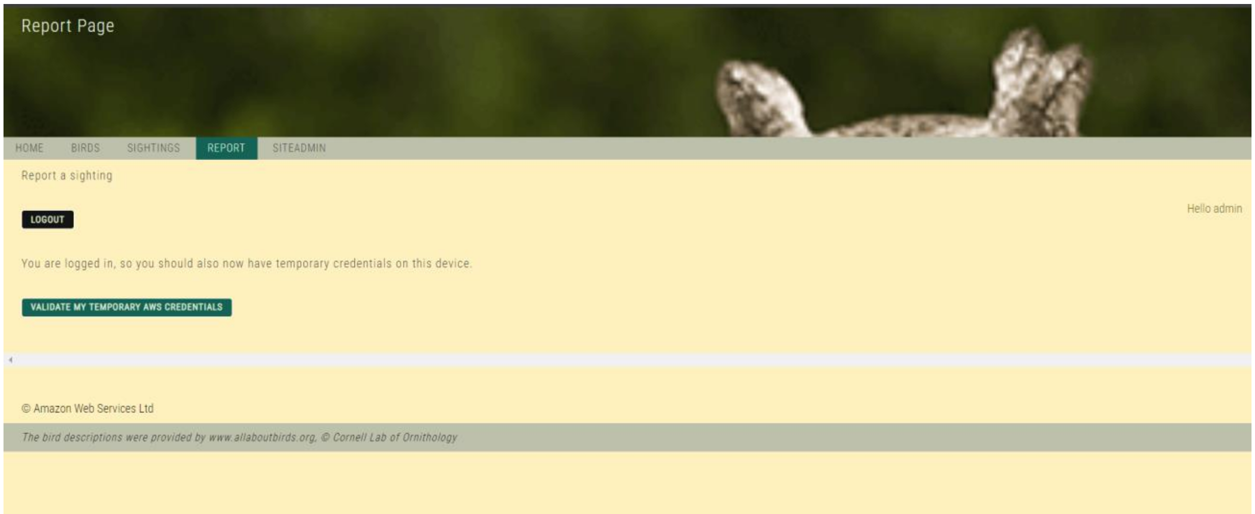
```
cd /home/ec2-user/environment/node_server

npm start
```

Task 8: Testing the Identity Pool Integration with the Application

- Test the application to ensure it can access AWS services using temporary credentials provided by the identity pool.
- Validate the integration by performing actions like querying the DynamoDB table using the generated credentials.

In this task, you test the updated Birds application to ensure that you can access temporary AWS credentials. With these temporary credentials, you are able to access AWS services based on the roles that were defined when you set up the identity pool. Remember that your identity pool is configured to associate authenticated users with an IAM role that allows access to a DynamoDB table.

- Return to the browser tab where the Birds application is open.
- Choose **HOME**, and refresh the page to ensure that your browser is using the updated code.
- Choose **REPORT**.
- Choose **LOGIN**, and log in with the **testuser** credentials that you created in the user pool.



Report a sighting

You are logged in, so you should also now have temporary credentials on this device.

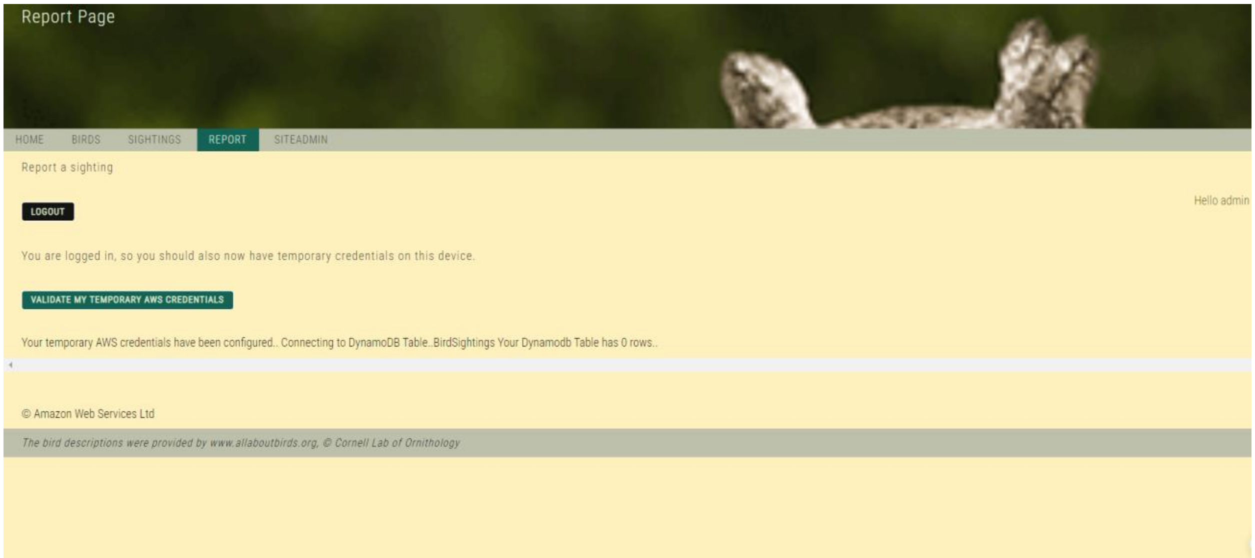
© Amazon Web Services Ltd

The bird descriptions were provided by www.allaboutbirds.org, © Cornell Lab of Ornithology

Note: You might have to log out of the **admin** account before you can log in with the **testuser** account.

After logging in, choose **REPORT** again.

To verify that you now have access to the temporary AWS credentials that you can use to interact with a DynamoDB table, choose **VALIDATE MY TEMPORARY AWS CREDENTIALS**.



Report a sighting

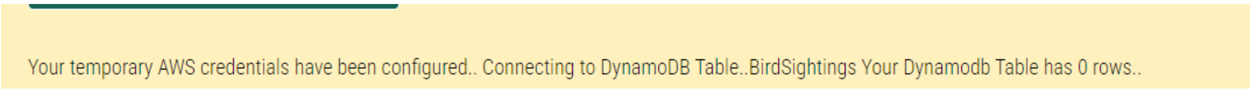
You are logged in, so you should also now have temporary credentials on this device.

Your temporary AWS credentials have been configured.. Connecting to DynamoDB Table..BirdSightings Your Dynamodb Table has 0 rows..

© Amazon Web Services Ltd

The bird descriptions were provided by www.allaboutbirds.org, © Cornell Lab of Ornithology

The application now uses the identity pool to generate temporary credentials and uses the same credentials to access the **BirdSightings** DynamoDB table. After successfully connecting to the database table, the application attempts to count the number of rows (0) and returns the following message: *Your temporary AWS credentials have been configured.. Connecting to DynamoDB Table..BirdSightings Your Dynamodb Table has 0 rows..*



Your temporary AWS credentials have been configured.. Connecting to DynamoDB Table..BirdSightings Your Dynamodb Table has 0 rows..

This test verifies that you have correctly configured the Amazon Cognito identity pool and the application. Users who are logged in are able to access temporary credentials to communicate with the DynamoDB database.

Note: You can access the DynamoDB table from the AWS Management Console and verify that the table has 0 rows..

Conclusion

Congratulations! You now have successfully done the following:

- Created an Amazon Cognito user pool
- Added users to the user pool
- Updated the example application to use the user pool for authentication
- Configured the Amazon Cognito identity pool
- Updated the example application to use the identity pool for authorization