

## Assignment 3 – Multi-Cloud Architecture Deployment



Public URL of the AWS Instance Website: <http://34.231.224.149/cos80001/photoalbum/album.php>

Team Member	Student ID	Responsibilities
<b>Alexander Blatchford</b>	101623699	<ul style="list-style-type: none"> <li>- Set up the OCI infrastructure, including the VCN, public and private subnets, and route tables.</li> <li>- Deployed the phpMyAdmin web server in the public subnet and Installed Apache, PHP, and configured phpMyAdmin to connect to the MySQL database.</li> <li>- Created and deployed the MySQL database instance, including database schema setup and testing via phpMyAdmin</li> </ul>
<b>Arun Ragavendhar Arunachalam Palaniyappan</b>	104837257	<ul style="list-style-type: none"> <li>- Handled the full VPN tunnel configuration across both clouds: created DRG, Customer Gateway, and VPN Connections on OCI and AWS and deployed the photoalbum website.</li> <li>- Configured all Firewalls and security components: Security Lists and Security groups on OCI, Security groups and NACL on AWS.</li> <li>- Enforced the least privilege principle and tested end-to-end connectivity across cloud environments.</li> </ul>
<b>Samsun Gulshan Sheik Dawood</b>	105009251	<ul style="list-style-type: none"> <li>- Set up the AWS infrastructure, including the VPC, public and private subnets, and route tables</li> <li>- Deployed and configured the AWS Bastion/Web server, including Apache, PHP, and Elastic IP assignment</li> <li>- Installed and deployed the photoalbum web application under the correct directory, connecting it to the OCI database via VPN</li> <li>- Created the test instance in the private subnet and performed ICMP ping tests to verify internal connectivity.</li> </ul>

Table 1: Team Member Names and Contribution

## Table of Contents

<i>I. INTRODUCTION.....</i>	<i>2</i>
<i>II. OCI INFRASTRUCTURE SETUP .....</i>	<i>2</i>
A. Virtual Cloud Network, Route Tabling and Subnet Configuration .....	2
B. Security Lists and NSGs .....	3
C. OCI Bastion Webserver Instance and Heatwave Database Instance .....	4
D. MySQL Database Deployment on OCI .....	6
E. Object Storage Configuration on OCI.....	6
<i>III. AWS INFRASTRUCTURE SETUP .....</i>	<i>6</i>
A. Virtual Private Cloud, Route Tabling and Subnet Configuration .....	6
B. Security Groups and Network ACLs .....	8
C. Web server setup on the AWS bastion web server and attaching elastic IP address.....	9
D. ICMP PING connectivity testing from Bastion to Test Instance .....	9
<i>IV. SITE-TO-SITE, TUNNELING AND VPN CONFIGURATION.....</i>	<i>10</i>
A. On OCI.....	10
B. On AWS.....	11
<i>V. WEB APPLICATION DEPLOYMENT AND CONNECTIVITY .....</i>	<i>12</i>
<i>VI. COMPARISON OF DIFFERENT VPN CONNECTION METHODS.....</i>	<i>13</i>
<i>VII. CHALLENGES AND LEARNINGS.....</i>	<i>14</i>
<i>VIII. CONCLUSION.....</i>	<i>14</i>
<i>IX. REFERENCES.....</i>	<i>14</i>

## I. INTRODUCTION

This report outlines the step-by-step process followed to deploy a multi-cloud photo album web application using Amazon Web Services (AWS) and Oracle Cloud Infrastructure (OCI). The objective was to split the system architecture, where the compute resources such as the web server run on AWS, while the data storage components—MySQL database and object storage—reside in OCI. A site-to-site IPsec connection was established between the AWS VPC and OCI VCN with a singular active tunnel between gateways on both clouds to ensure safe and restricted access between clouds, the AWS bastion/web server and the OCI database.

**Public URL of the AWS Instance Website:** <http://34.231.224.149/cos80001/photoalbum/album.php>

**OCI Tenancy:** s104837257, **Associated student Email:** 104837257@student.swin.edu.au

**AWS Account - Federated user:** [voclabs/user3898953=104837257@student.swin.edu.au](#)

## II. OCI INFRASTRUCTURE SETUP

### A. Virtual Cloud Network, Route Tabling and Subnet Configuration

In the Oracle Cloud Infrastructure (OCI), a Virtual Cloud Network (VCN) named "Assignment-3VCN" was set up in the us-ashburn-1 region using the 172.17.0.0/16 CIDR range. Two subnets were designed within this VCN. Public Subnet 1 (172.17.1.0/24) was created to contain the phpMyAdmin instance and Private Subnet 1 (172.17.3.0/24) was used to host the MySQL database, the VCN's CIDR is shown in Fig 1 and the subnet configuration is illustrated in Fig 2.

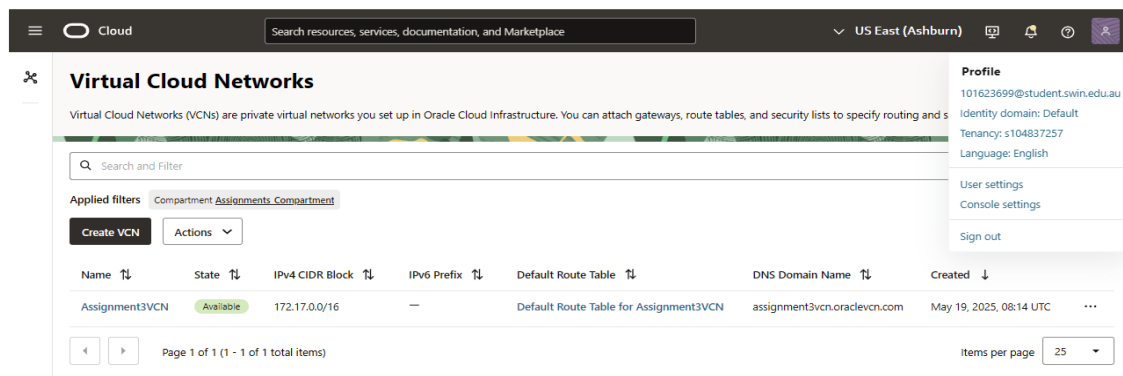


Fig 1: OCI VCN CIDR and Name

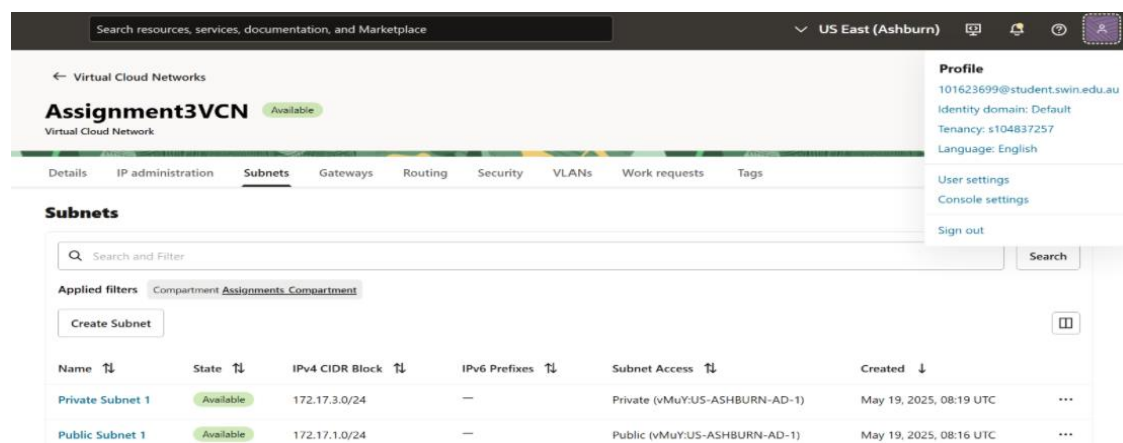


Fig 2: OCI Subnet Configuration

The public subnet was linked to a public route table with an Internet Gateway (Fig 3) to allow internet access. The private subnet was connected to a separate private route table (Fig 4) that restricted any external connectivity, enhancing internal resource security.

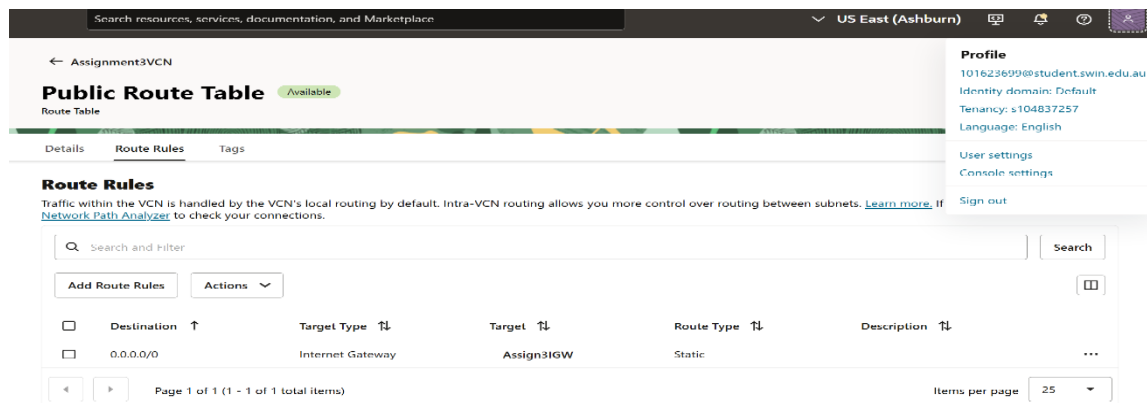


Fig 3: Public Route Table Route Rules

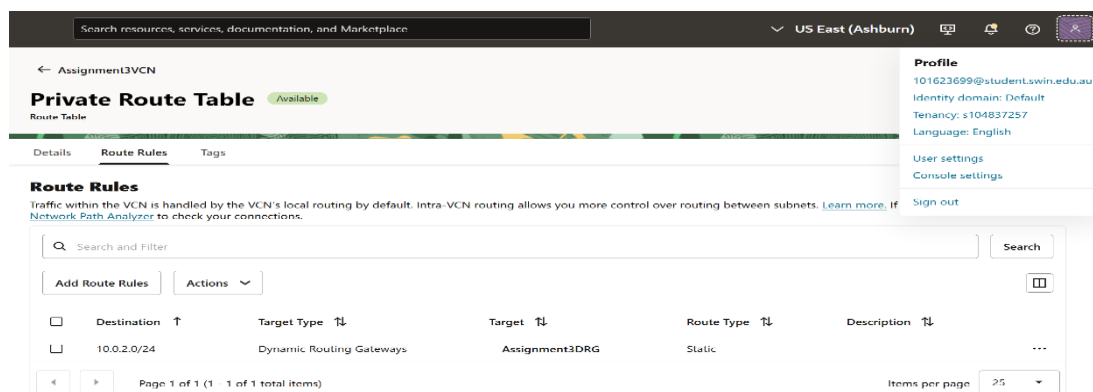


Fig 4: Private Route Table Route Rules

## B. Security Lists and NSGs

In OCI, security was managed using a singular security list for the public subnet called PublicSubnet1SL (Fig 7) and the default security list was used to manage the private subnet's security (Fig 6). This security was utilised in tandem with a Network Security Group (NSG) called Web-tier NSG (Fig 5) to be attached to the VNIC of the bastion instance on OCI. PublicSubnet1SL allows ingress for SSH and HTTP requests from anywhere. The default security list allows ingress from the webserver's restricted 32786-61000 ephemeral ports with a destination to the 3306 MySQL database port. Traffic to the database from the AWS Webserver's public subnet CIDR (10.0.2.0/24) is also allowed when directed to port 3306 from the ephemeral range. The NSG Web-tier NSG allows egress to the AWS bastion directed to the database from the ephemeral range and ingress from all anywhere for HTTP and SSH from the internet.

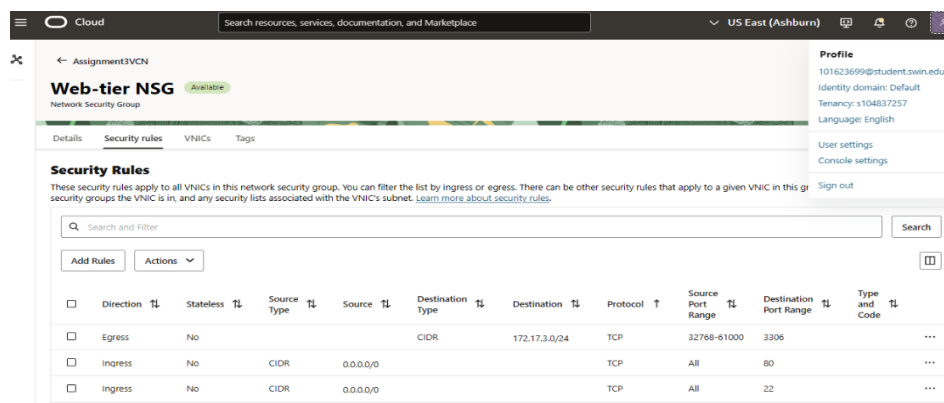


Fig 5: Network Security Group: Web-tier NSG

**Default Security List for Assignment3VCN** Available

Security List

Instance traffic is controlled by firewall rules on each Instance in addition to this Security List

Details Security rules Tags

**Ingress Rules**

Search and Filter

Add Ingress Rules Actions

Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows	Description
No	172.17.1.0/24	TCP	32768-61000	3306		TCP traffic for ports: 3306	...
No	10.0.2.0/24	TCP	32768-61000	3306		TCP traffic for ports: 3306	...

Page 1 of 1 (1 - 2 of 2 total items)

Items per page 25

**Egress Rules**

Search and Filter

Add Egress Rules Actions

No items to display

**Profile**

101623699@student.swin.edu.au  
Identity domain: Default  
Tenancy: s104837257  
Language: English

User settings  
Console settings  
Sign out

Fig 6: Default Security List (For DB Private Subnet)

**PublicSubnet1SL** Available

Security List

Instance traffic is controlled by firewall rules on each Instance in addition to this Security List

Details Security rules Tags

**Ingress Rules**

Search and Filter

Add Ingress Rules Actions

Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows	Description
No	0.0.0.0/0	TCP	All	22		TCP traffic for ports: 22 SSH Remote Login Protocol	...
No	0.0.0.0/0	TCP	All	80		TCP traffic for ports: 80	...

**Egress Rules**

Search and Filter

Add Egress Rules Actions

Stateless	Destination	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows	Description
No	172.17.3.0/24	TCP	32768-61000	3306		TCP traffic for ports: 3306	...

**Profile**

104837257@student.swin.edu.au  
Identity domain: Default  
Tenancy: s104837257  
Language: English

User settings  
Console settings  
Sign out

Fig 7: Security List: PublicSubnet1SL

### C. OCI Bastion Webserver Instance and Heatwave Database Instance

The Bastion/PHPadmin server was created in the public subnet and the name and IP addresses can be seen in Fig 8. The security and route table associations are illustrated in Fig 9, where the public route table is associated.

**Instances**

An instance is a compute host. Choose between virtual machines (VMs) and bare metal instances. The image that you use to launch an instance determines its operating system and

Search and Filter

Applied filters: Compartment: Assignments Compartment

Create instance Actions

Name	State	Public IP	Private IP	Shape	OCPU count	Memory (GB)	Availability domain	Fault domain	Created
Bastion/PHPadmin	Running	193.122.143.70	172.17.1.28	VM.Standard.E2.1	1	8	AD-1	FD-1	May 20, 2025, 01

Page 1 of 1 (1 - 1 of 1 total items)

Items per page 25

**Profile**

101623699@student.swin.edu.au  
Identity domain: Default  
Tenancy: s104837257  
Language: English

User settings  
Console settings  
Sign out

Fig 8: Bastion/PHPadmin OCI Instance

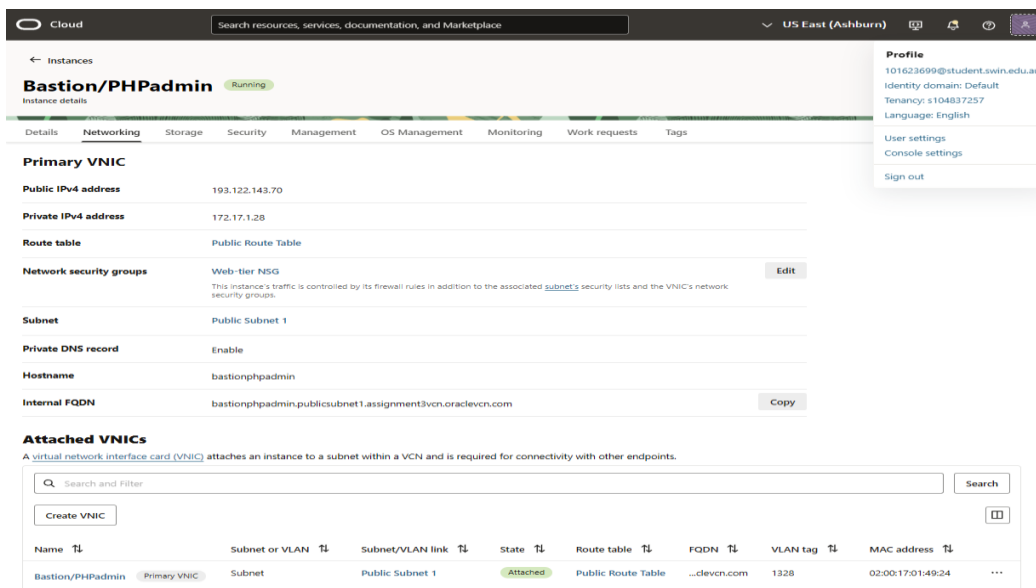


Fig 9: Bastion/PHPAdmin Route Table, Subnet and NSG Association

The MySQL Database was deployed using OCI's Heatwave Interface; the endpoint, subnet association and VCN association can be seen in Fig 10. The database was configured to have the shape as MySQL.2, with 2 ECPUs, 8.0.41 DB version, MySQL 2 Standalone config and no heatwave clustering with no backups according to specification (Fig 11).

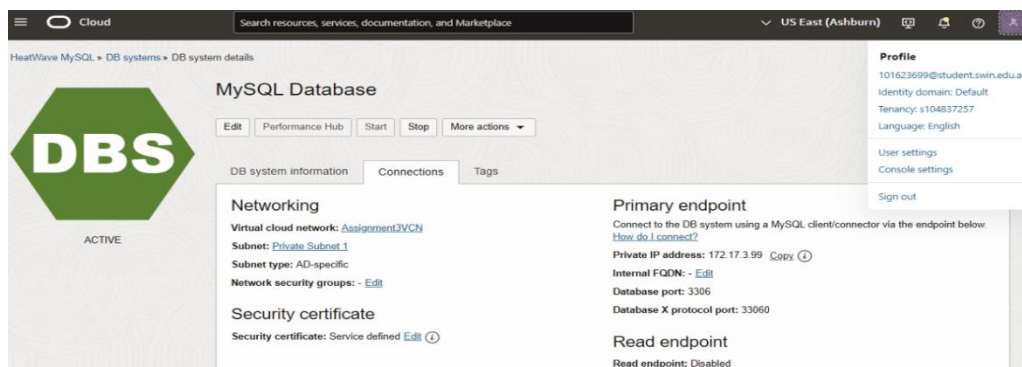


Fig 10: MySQL Database Endpoint, VCN and Private Subnet Association

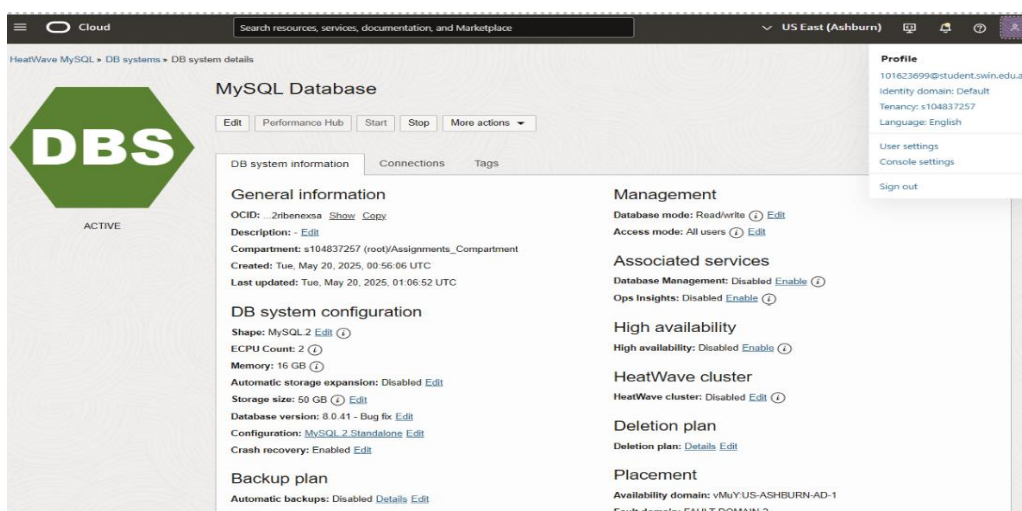
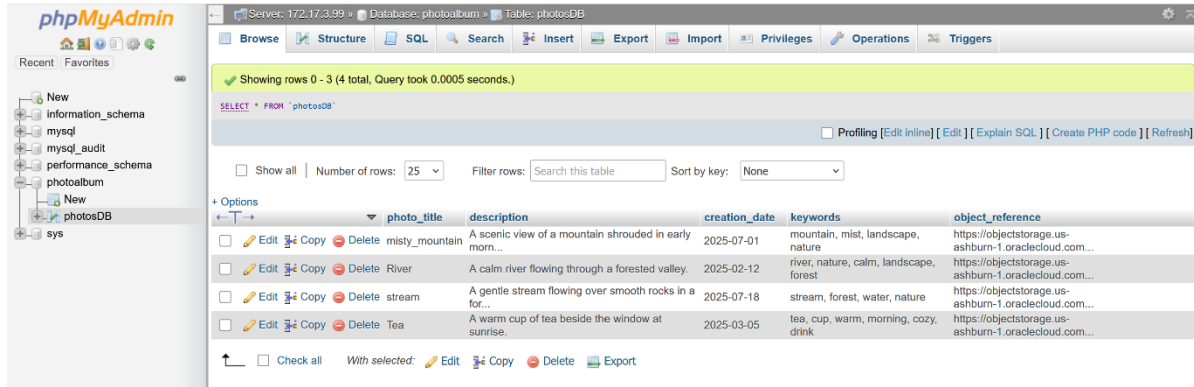


Fig 11: MySQL Database Deployment Details

## D. MySQL Database Deployment on OCI

In OCI, a managed MySQL database (version 8.0.41, MySQL2 shape, 50GB storage) was deployed in Private Subnet 1, with public access disabled for security. It used OCI's default security list, allowing ingress only from the OCI Bastion/Web Server and the AWS Bastion/Web Server. The database, named photoalbum, had one table called photosDB, with fields: photo\_title, description, creation\_date, keywords, and object\_reference—storing the photo's name, caption, date, tags, and image URL, as seen in Fig 12. It was managed using phpMyAdmin, accessed securely from the OCI Bastion/Web Server.



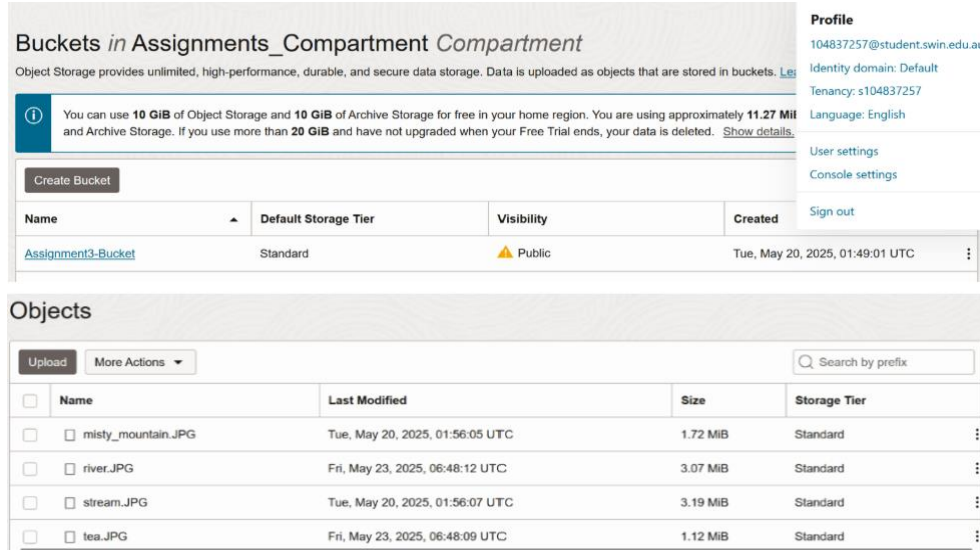
The screenshot shows the phpMyAdmin interface for the 'photoalbum' database, specifically the 'photosDB' table. The table structure is defined as follows:

photo_title	description	creation_date	keywords	object_reference
misty_mountain	A scenic view of a mountain shrouded in early morn...	2025-07-01	mountain, mist, landscape, nature	https://objectstorage.us-ashburn-1.oraclecloud.com...
River	A calm river flowing through a forested valley.	2025-02-12	river, nature, calm, landscape, forest	https://objectstorage.us-ashburn-1.oraclecloud.com...
stream	A gentle stream flowing over smooth rocks in a for...	2025-07-18	stream, forest, water, nature	https://objectstorage.us-ashburn-1.oraclecloud.com...
Tea	A warm cup of tea beside the window at sunrise.	2025-03-05	tea, cup, warm, morning, cozy, drink	https://objectstorage.us-ashburn-1.oraclecloud.com...

Fig 12: photosDB table schema and data records from phpMyAdmin

## E. Object Storage Configuration on OCI

A bucket named Assignment-3\_bucket was created using OCI's Object Storage service. Public access was granted at the bucket level, which meant all uploaded photos were accessible via direct public URLs. This removed the need to individually modify permissions for each file. Four sample images were uploaded through the OCI console and tested using an incognito browser window to confirm public access worked as intended.



The screenshot shows the OCI Object Storage console for the 'Assignment3-Bucket'. The bucket is configured with 'Standard' storage tier and 'Public' visibility. The following table lists the uploaded objects:

Name	Last Modified	Size	Storage Tier
misty_mountain.JPG	Tue, May 20, 2025, 01:56:05 UTC	1.72 MiB	Standard
river.JPG	Fri, May 23, 2025, 06:48:12 UTC	3.07 MiB	Standard
stream.JPG	Tue, May 20, 2025, 01:56:07 UTC	3.19 MiB	Standard
tea.JPG	Fri, May 23, 2025, 06:48:09 UTC	1.12 MiB	Standard

Fig 13: Public Access Settings and Uploaded Images in Object Storage

## III. AWS INFRASTRUCTURE SETUP

### A. Virtual Private Cloud, Route Tabling and Subnet Configuration

To begin the deployment, a custom Virtual Private Cloud (VPC) named "Assignment3VPC" was created within the us-east-1-b region in AWS. The CIDR block assigned to this VPC was 10.0.0.0/16, inside this VPC two subnets were designed, one as public subnet and the other as a private subnet (Fig 14). Public Subnet 2 with the



IP range 10.0.2.0/24 and Private Subnet 2 (10.0.4.0/24) were created for AWS, with the public for the Bastion/Web server and the private used to host a test instance.

A public route table was created and attached for the public subnet. Likewise, a private route table was created and attached to the private subnet. Public subnets were connected to an Internet Gateway using the public route table that allowed internet access (Fig 16). The private subnet was connected to a separate route table that did not include internet access (Fig 15). The OCI database subnet CIDR (172.17.3.0/24) and the target AWS VPG to reach it, were added in the public route table, allowing the AWS bastion/web server to access the Database on the OCI Cloud through the VPN tunnel.

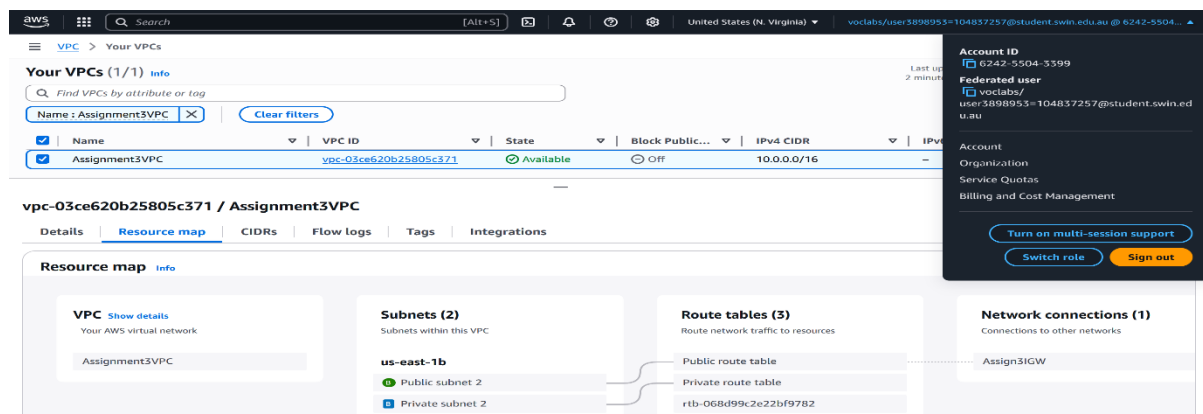


Fig 14: Assignment3VPC, its CIDR, Subnet and Route Table Associations

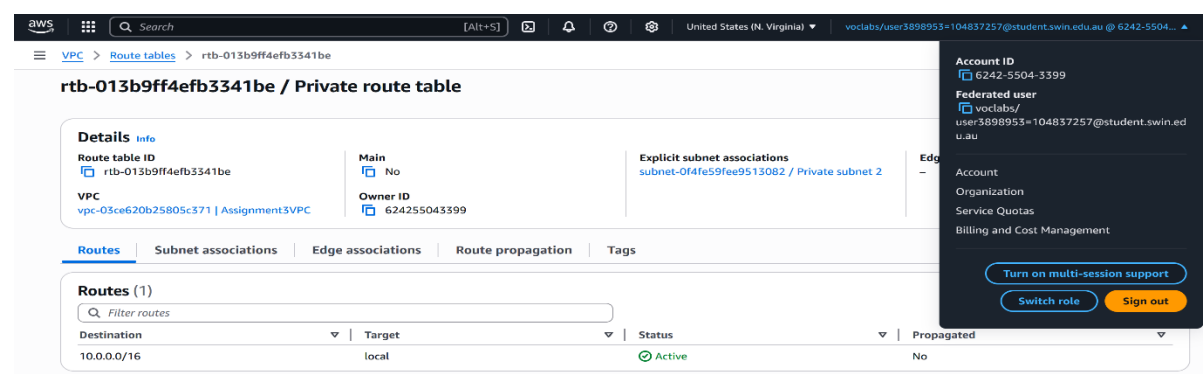


Fig 15: Routes associated with Private Route Table

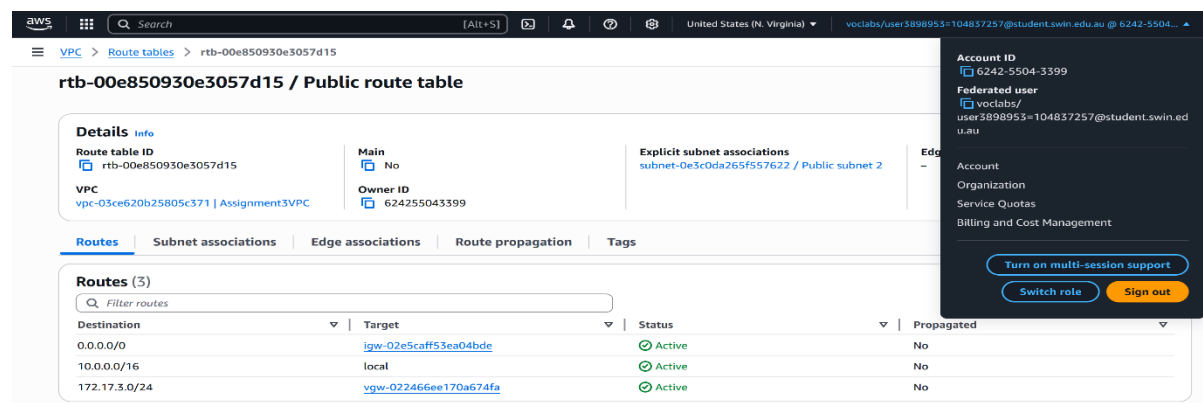


Fig 16: Routes associated with Public Route Table



## B. Security Groups and Network ACLs

On AWS, the Web-tier Security Group controlled access to the Bastion/Web Server. It allowed SSH (port 22) and HTTP (port 80) from anywhere for remote access and public hosting. Outbound traffic was restricted to only what was needed - MySQL (port 3306) to the OCI database (172.17.3.0/24) and ICMP to the test instance (10.0.4.0/24), ensuring private, secure communication (Fig 17).

The PublicSubnet2NACL allowed inbound traffic on ports 22, 80, ICMP, and ephemeral ports (32768–61000), while outbound traffic was limited to ports 1024–65535 and ICMP to support return traffic (Fig 19). The Test-instance Security Group allowed only ICMP both ways, enabling ping tests while blocking everything else (Fig 18). All traffic was tightly controlled under the least privilege principle—only essential communication was permitted, with everything else denied.

**sg-0244bdd4a52f20d26 - Web-tier Security group**

**Details**

Security group name Web-tier Security group	Security group ID sg-0244bdd4a52f20d26	Description Security group for web server
Owner 624255043399	Inbound rules count 2 Permission entries	Outbound rules count 2 Permission entries

**Inbound rules (2)**

Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
sgr-0610490d21a1b87df	IPv4	SSH	TCP	22	0.0.0.0/0	-
sgr-0ee9b9593a08a4f8c	IPv4	HTTP	TCP	80	0.0.0.0/0	-

**Outbound rules (2)**

Security group rule ID	IP version	Type	Protocol	Port range	Destination	Description
sgr-0c446b2edc4b617b7	IPv4	MySQL/Aurora	TCP	3306	172.17.3.0/24	-
sgr-07f4ad151dcdcf699	IPv4	All ICMP - IPv4	ICMP	All	10.0.4.0/24	-

Fig 17: Web-tier Security group with inbound and outbound rules

**sg-0547b5ae118c8ea63 - Test-instance Security group**

**Details**

Security group name Test-instance Security group	Security group ID sg-0547b5ae118c8ea63	Description Security group for test instance
Owner 624255043399	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry

**Inbound rules (1)**

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-0818ca1403ad9a8a6	IPv4	All ICMP - IPv4	ICMP	All	10.0.2.0/24

**Outbound rules (1)**

Name	Security group rule ID	IP version	Type	Protocol	Port range	Destination
-	sgr-05e2d997bae3ecb02	IPv4	All ICMP - IPv4	ICMP	All	10.0.2.0/24

Fig 18: Test-instance Security group with inbound and outbound rules

**PublicSubnet2NACL**

**Details**

Network ACL ID: `acl-0a89f60a974697908`

Associated with: `subnet-0e3c0da265f557622 / Public subnet 2`

Default: No

Owner: `624255043399`

**Inbound rules (5)**

Rule number	Type	Protocol	Port range	Source	Allow/Deny
100	HTTP (80)	TCP (6)	80	0.0.0.0/0	Allow
110	SSH (22)	TCP (6)	22	0.0.0.0/0	Allow
120	All ICMP - IPv4	ICMP (1)	All	10.0.4.0/24	Allow
130	Custom TCP	TCP (6)	32768 - 61000	172.17.3.0/24	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

**Outbound rules (3)**

Rule number	Type	Protocol	Port range	Destination	Allow/Deny
100	Custom TCP	TCP (6)	1024 - 65535	0.0.0.0/0	Allow
110	All ICMP - IPv4	ICMP (1)	All	10.0.4.0/24	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

Fig 19: PublicSubnet2NACL with inbound and outbound rules

### C. Web server setup on the AWS bastion web server and attaching elastic IP address

A script was run on the AWS bastion/web server to install Apache, PHP, and phpMyAdmin to deploy and host the photoalbum web site. To keep the public IP address as the same, even after restarts, an Elastic IP was created and attached as shown below in Fig 20.

**Instance summary for i-0a872917f4753f804 (Bastion/Web server)**

Updated less than a minute ago

Instance ID: `i-0a872917f4753f804`

IPV6 address: -

Hostname type: IP name: `ip-10-0-2-242.ec2.internal`

Answer private resource DNS name: -

Auto-assigned IP address: -

IAM Role: -

IMDSv2: Required

Operator: -

Public IPv4 address: `34.231.224.149` | open address

Instance state: `Running`

Private IP DNS name (IPv4 only): `ip-10-0-2-242.ec2.internal`

Instance type: `t2.micro`

VPC ID: `vpc-03ce620b25805c371` (Assignment3VPC)

Subnet ID: `subnet-0e3c0da265f557622` (Public subnet 2)

Instance ARN: `arn:aws:ec2:us-east-1:624255043399:instance/i-0a872917f4753f804`

Private IPV4 addresses: `10.0.2.242`

Public DNS: -

Elastic IP addresses: `34.231.224.149` (Assignment3-ElasticIP) [Public IP]

AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Auto Scaling Group name: -

Managed: false

Fig 20: Bastion/Webserver with the elastic IP address

### D. ICMP PING connectivity testing from Bastion to Test Instance

The test instance resides in the private subnet (10.0.4.0/24) and its private IP address is 10.0.4.68 (Fig 21). An ICMP ping was carried out from the bastion/webserver to this test instance to test connectivity and this was successful as shown below in Fig 22.

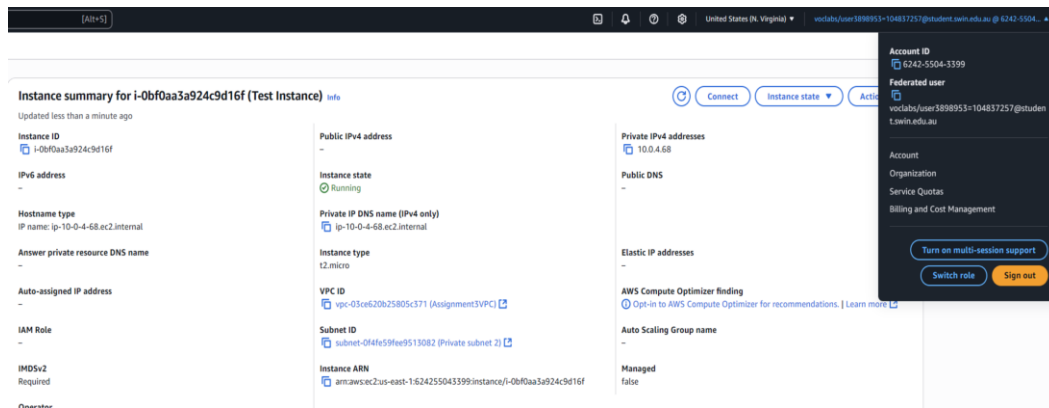


Fig 21: Test Instance with its Private IP Address

```

_/m/'
Last login: Sat May 24 10:05:28 2025 from 18.206.107.29
[ec2-user@ip-10-0-2-242 ~]$ ping 10.0.4.68
PING 10.0.4.68 (10.0.4.68) 56(84) bytes of data.
 64 bytes from 10.0.4.68: icmp_seq=1 ttl=127 time=0.463 ms
 64 bytes from 10.0.4.68: icmp_seq=2 ttl=127 time=0.694 ms
 64 bytes from 10.0.4.68: icmp_seq=3 ttl=127 time=0.442 ms
 64 bytes from 10.0.4.68: icmp_seq=4 ttl=127 time=0.354 ms
 64 bytes from 10.0.4.68: icmp_seq=5 ttl=127 time=0.728 ms
 64 bytes from 10.0.4.68: icmp_seq=6 ttl=127 time=0.401 ms
 64 bytes from 10.0.4.68: icmp_seq=7 ttl=127 time=0.765 ms
 64 bytes from 10.0.4.68: icmp_seq=8 ttl=127 time=0.381 ms
 64 bytes from 10.0.4.68: icmp_seq=9 ttl=127 time=0.515 ms
 64 bytes from 10.0.4.68: icmp_seq=10 ttl=127 time=1.21 ms
 64 bytes from 10.0.4.68: icmp_seq=11 ttl=127 time=0.350 ms
 64 bytes from 10.0.4.68: icmp_seq=12 ttl=127 time=0.734 ms
^C
--- 10.0.4.68 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11471ms
rtt min/avg/max/mdev = 0.350/0.586/1.212/0.242 ms
[ec2-user@ip-10-0-2-242 ~]$

```

Fig 22: Successful ICMP ping from Bastion to Test Instance

#### IV. SITE-TO-SITE, TUNNELING AND VPN CONFIGURATION

##### A. On OCI

The first step to establish the VPN was to create a dummy AWS CGW and then create an OCI Dynamic Routing Gateway (DRG) (Fig 23) and attach it to the VCN. This DRG acts like a virtual router that can send traffic to other networks. Next, a Customer Premises Equipment (CPE) (Fig 24) object was created in OCI using a dummy AWS CGW's outside tunnel IP address. An IPsec connection was then formed between OCI and AWS using a shared secret. Finally, the private subnet route table was updated to send any traffic bound for 10.0.0.0/16 through the DRG (Fig 25). The tunnels that are interconnected have one that is not configured portrayed in Fig 26.

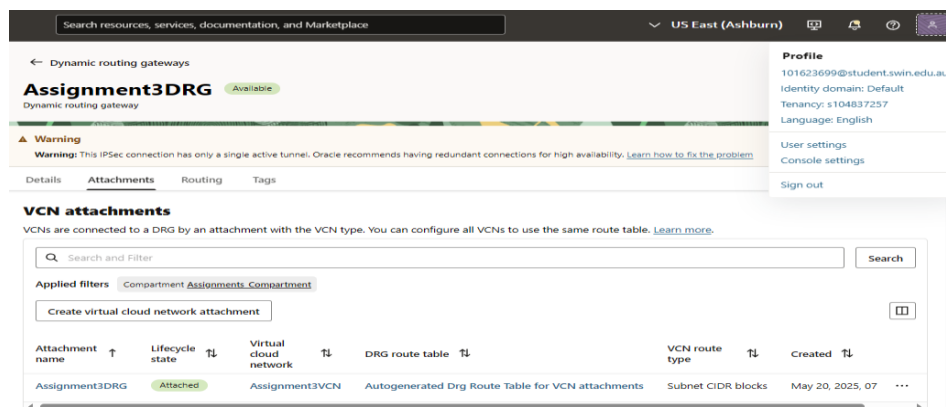


Fig 23: Assignment3DRG OCI DRG with VCN Attachment

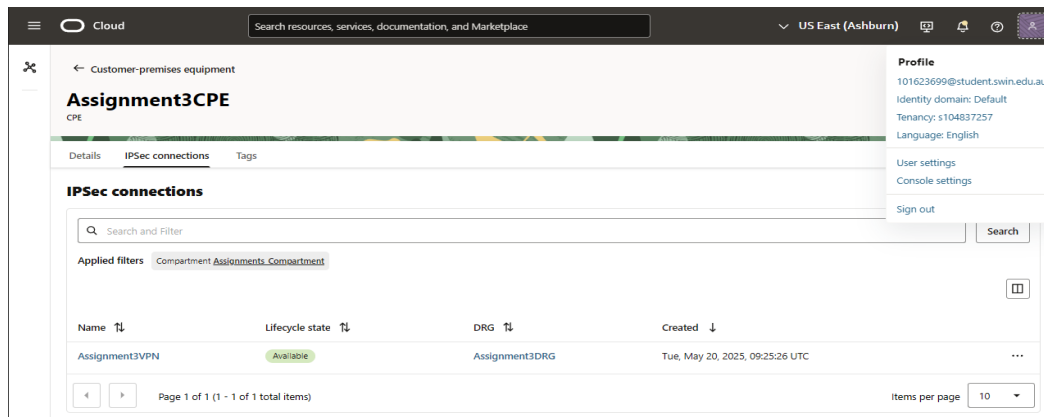


Fig 24: Assignment3CPE OCI CPE with VPN and DRG Attachments

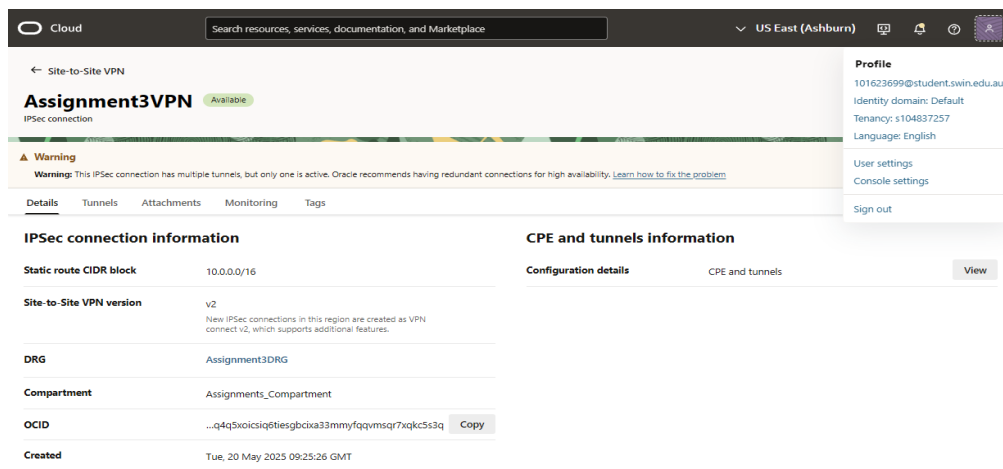


Fig 25: Assignment3VPN OCI VPN with AWS CIDR Block

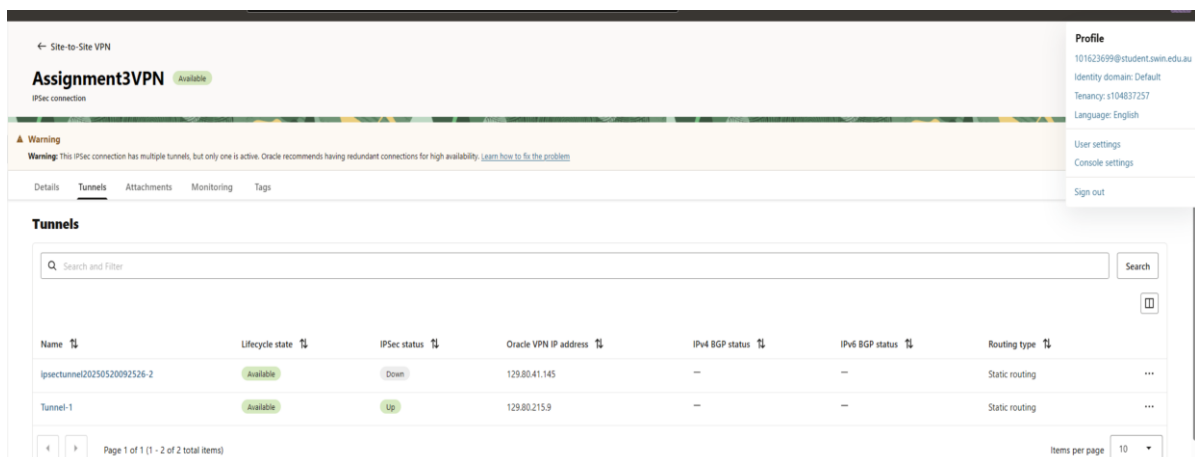


Fig 26: Assignment3VPN OCI Tunnel Setup (1 Down and 1 Up)

## B. On AWS

In AWS, a Virtual Private Gateway (VGW) was created and attached to the VPC (Fig 27). A Customer Gateway (CGW) was then defined using the public IP of the OCI VPN endpoint (Fig 28). A VPN connection was established by combining the VGW and CGW with the same shared secret used on the OCI side. To complete the setup, the route table for the AWS private subnets was updated to forward traffic bound for 172.17.3.0/24 through the VPN tunnel. Tunnel status was confirmed to be “UP” on both sides through the AWS and OCI dashboards (Fig 29).

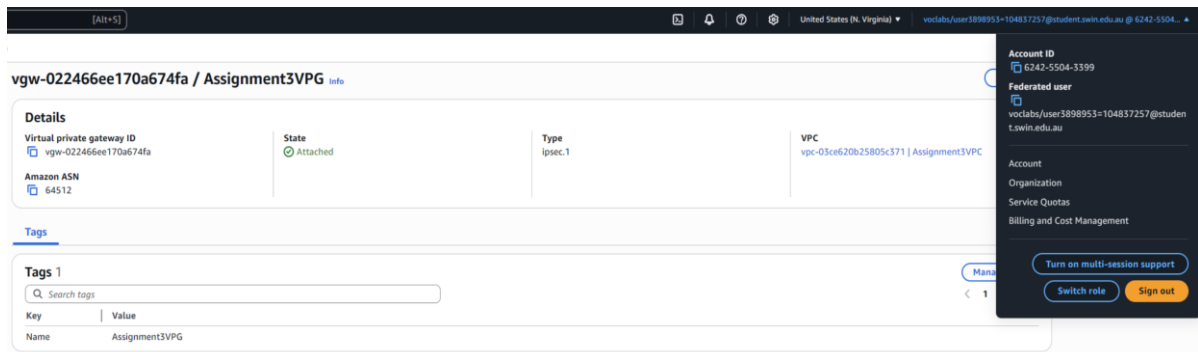


Fig 27: Assignment3VPG with AWS VPG to VPC attachment

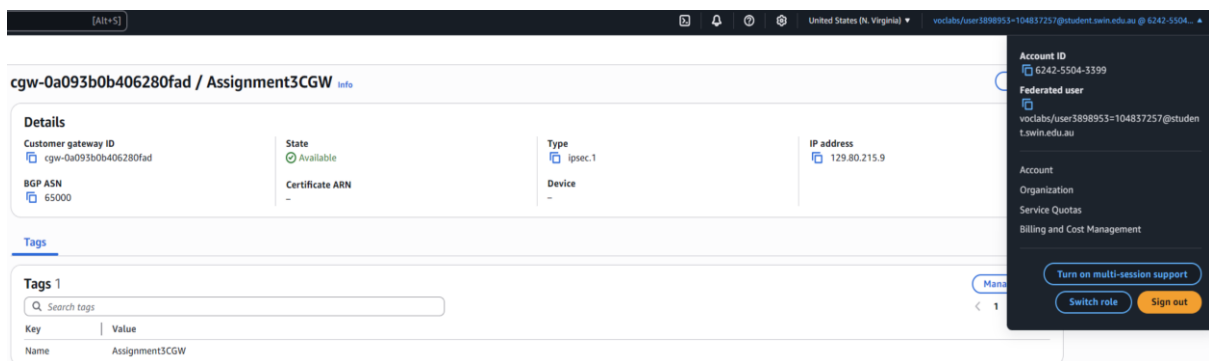


Fig 28: Assignment3CGW with outside Public IP of the OCI side of the VPN tunnel

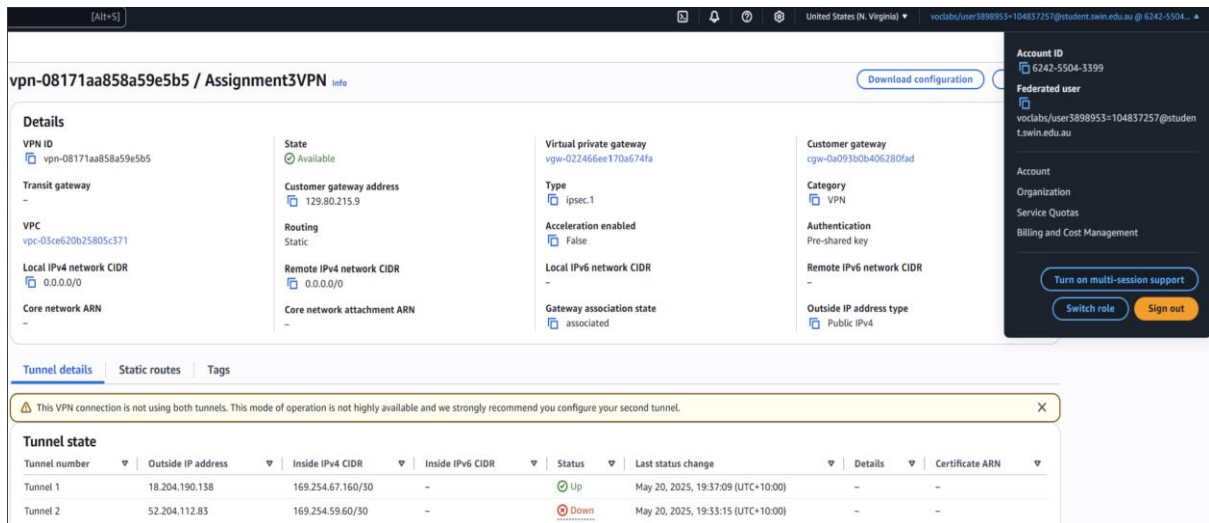


Fig 29: Assignment3VPN AWS Tunnel Setup (1 Down and 1 Up)

## V. WEB APPLICATION DEPLOYMENT AND CONNECTIVITY

The PHP web application named photoalbum\_v3.0 was deployed inside the EC2 instance under the `/var/www/html/cos80001/photoalbum/` directory. The constants.php file was updated to use the private IP address of the OCI database, ensuring database traffic did not leave the VPN.

The application was tested by loading the album page via the public Elastic IP. It successfully displayed all images and their associated metadata from the MySQL database detailed as shown below in Fig 30.

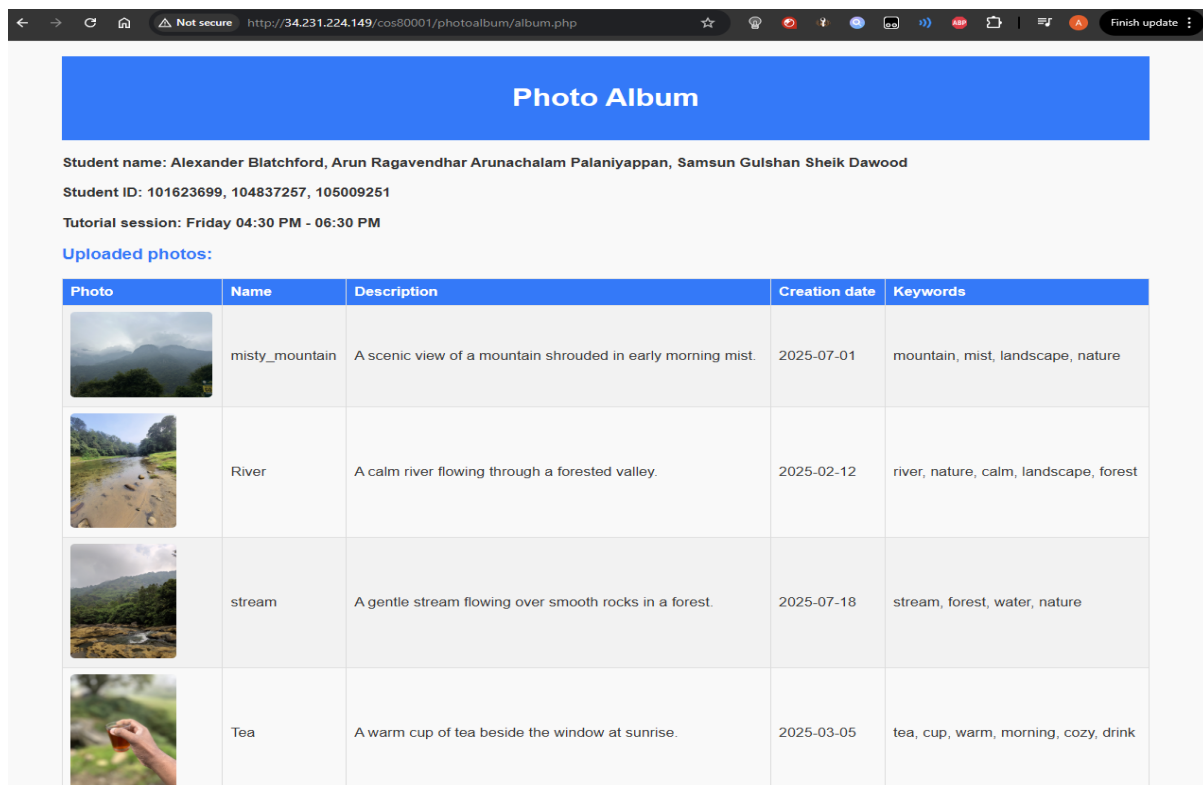


Fig 30: Web Deployment of photoalbum.php

## VI. COMPARISON OF DIFFERENT VPN CONNECTION METHODS

To connect AWS and OCI, three options were considered: AWS Direct Connect, OCI FastConnect, and IPsec VPN.

AWS Direct Connect and OCI FastConnect both offer private network links between cloud platforms. These connections do not use the public internet—instead, they rely on a dedicated physical line provided through partners like Telstra, Equinix, or Megaport, connecting directly to the cloud provider's infrastructure. These private lines support bandwidth speeds ranging from 1 Gbps to 100 Gbps, depending on the service tier and provider. They are designed for low latency (as low as 1–5 ms) and high reliability, which is important for enterprise workloads like large datacentres, banks, and also for military/defense usages.

However, these private links are costly and complex to set up. For example, AWS Direct Connect charges can start from \$0.25 to \$0.30 per GB transferred, with monthly port fees ranging from \$72 to \$2,250, depending on speed. Setup can also take several weeks, as it may involve third-party coordination and physical access to colocation facilities. So, these options are not well suited for short-term academic, testing, or lightweight deployments.

IPsec VPN, on the other hand, creates a secure connection over the public internet. It uses encryption to protect all data during transmission, without needing physical infrastructure. It can be set up fully in software through the cloud provider's console, with most connections ready in under an hour. While it does not offer guaranteed speeds or consistent latency—since it depends on internet conditions—it is completely free aside from regular data transfer costs, and provides enough security for non-production and small-scale use.

For this project, IPsec VPN was the most practical choice. It allowed safe, encrypted communication between AWS and OCI without any extra cost or delays. Since the goal was to build a working system for learning and testing as a part of the assignment, and not to run a high-traffic or business service, the added cost and

complexity of Direct Connect or FastConnect were unnecessary. IPsec VPN provided exactly what was needed: a simple, fast, and secure way to connect the two clouds.

## VII. CHALLENGES AND LEARNINGS

Some issues came up during the setup. The first problem was with the VPN connection. The tunnel could not be brought up at first, and traffic between AWS and OCI wasn't working. This was because the route tables in both clouds were missing the correct CIDR entries. After adding the proper routes and pointing them to the VPN gateways, the tunnel came up and the connection started working as expected.

Next, the AWS web server couldn't connect to the OCI MySQL database. This was due to a missing rule in the OCI default security list, which didn't allow ingress traffic on port 3306. Once the rule was added to allow traffic from the AWS subnet, the issue was resolved. There was also a problem with pinging the test instance in AWS. The pings failed because the NACL didn't have the inbound rules for ICMP. Since NACLs are stateless, both directions need to be set manually. This was fixed by updating the NACL and the test instance's Security Group. These problems helped the team understand how routing and security rules work differently in AWS and OCI, and the importance of allowing only the traffic that is needed as per system configuration.

## VIII. CONCLUSION

This assignment successfully demonstrated how to build and deploy a working multi-cloud system using AWS and OCI. The web server and application were hosted on AWS, while the MySQL database, phpMyAdmin interface, and object storage were deployed on OCI. The two environments were connected using an IPsec VPN tunnel that allowed secure, private communication without using public access for database traffic.

Each part of the setup followed the least privilege principle, with only required ports and IPs allowed in all security components. All systems were tested for connectivity and access, including successful ICMP communication between subnets and secure data flow from the web application to the remote database. The deployment met all the project requirements and provided hands-on experience with setting up networks, routing, VPNs, firewalls, and cloud-based application hosting across two different platforms. The process also gave the team a clear understanding of how AWS and OCI differ in their network and security setups, and how to configure them to work together in a hybrid cloud setup. This experience will be valuable in future industry roles that involve working with cloud infrastructure and setting up secure, multi-cloud systems.

## IX. REFERENCES

- [1] Amazon Web Services, "AWS Site-to-Site VPN," *AWS Documentation*. [Online]. Available: <https://docs.aws.amazon.com/vpn/latest/s2svpn/>
- [2] Oracle Cloud Infrastructure, "Setting Up an IPsec VPN," *OCI Documentation*. [Online]. Available: <https://docs.oracle.com/en-us/iaas/Content/Network/Tasks/settingupIPSec.htm>
- [3] Oracle Cloud Infrastructure, "Object Storage Overview," *OCI Documentation*. [Online]. Available: <https://docs.oracle.com/en-us/iaas/Content/Object/Concepts/objectstorageoverview.htm>
- [4] Oracle Cloud Infrastructure, "OCI MySQL Database Service," *OCI Documentation*. [Online]. Available: <https://docs.oracle.com/en-us/iaas/mysql-database/doc/index.html>
- [5] Amazon Web Services, "Elastic IP Addresses," *AWS Documentation*. [Online]. Available: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html>