**COS80001 – Cloud Engineering**

**Assignment 3: Multi-Cloud Architecture Deployment**  
**Report Title:** Photo Album Application Hosted on AWS and OCI with Site-to-Site VPN Connectivity  
**Student Name:** Arun Ragavendhar Arunachalam Palaniyappan  
**Student ID:** 104837257  
**Tutorial Time:** Friday – 04:30 PM to 06:30 PM  
**OCI Tenancy Name:** s104837257

**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 VPN was established between the AWS VPC and OCI VCN to ensure secure, private communication for database queries, avoiding public internet routes. The architecture was designed to simulate real-world hybrid deployments with cost-efficiency and security in mind.

**II. INFRASTRUCTURE SETUP**

**A. AWS – Virtual Private Cloud and Subnet Configuration**

To begin the deployment, a custom Virtual Private Cloud (VPC) named "Assignment3VPC" was created within the us-east-1 region in AWS. The CIDR block assigned to this VPC was 10.0.0.0/16, offering a large range of private IP addresses. Inside this VPC, four subnets were designed to logically separate resources. Two subnets were marked as public: Public Subnet 1 with the IP range 10.0.1.0/24 and Public Subnet 2 with the IP range 10.0.2.0/24, where the web server was eventually placed. The other two, Private Subnet 1 (10.0.3.0/24) and Private Subnet 2 (10.0.4.0/24), were created for internal use, with the second used to host a test instance.

Public subnets were connected to an Internet Gateway using a dedicated route table that allowed internet access. The private subnets were connected to a separate route table that did not include internet access, ensuring these resources remained isolated and protected.

**Figure 1: AWS VPC and Subnet Configuration**

**B. AWS Security Groups and Network ACLs**

To protect the networked components, security was implemented using both Security Groups and Network ACLs. A security group named WebServerSG was applied to the EC2 instance in the public subnet. This group allowed HTTP traffic on port 80 and SSH traffic on port 22 from any external IP, enabling website access and administrative login. It also permitted ICMP (ping) traffic from the internal subnet to support connectivity testing. Another group, TestInstanceSG, was used on the private test instance and allowed all internal traffic for testing flexibility.

In addition to security groups, a custom Network ACL was created and attached to Public Subnet 2. This ACL allowed inbound HTTP and SSH connections, as well as outbound return traffic through ephemeral ports (1024–65535), which are commonly used for response communication. This combination of security layers ensured the system was both functional and secure.

**Figure 2: AWS Security Group Rules**  
**Figure 3: Network ACL Rules for Public Subnet 2**

**C. AWS EC2 Instance (Web Server)**

A virtual machine, or EC2 instance, was deployed in Public Subnet 2. This instance used the Amazon Linux 2023 operating system and was created with the t2.micro type, which is free-tier eligible and suitable for lightweight applications. A startup script was used during instance creation to automatically install Apache, PHP, and phpMyAdmin. To maintain a consistent IP address even after restarts, an Elastic IP was assigned to this instance, making it reliably accessible to the public.

**Figure 4: Web Server EC2 Setup and Elastic IP Binding**

**D. OCI – Virtual Cloud Network and Subnet Configuration**

In 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 IP range. Four subnets were designed within this VCN. Public Subnet 1 (172.17.1.0/24) and Public Subnet 2 (172.17.2.0/24) were reserved for any future external-facing services. Private Subnet 1 (172.17.3.0/24) was used to host the MySQL database, while Private Subnet 2 (172.17.4.0/24) was used to create an internal test instance.

The public subnets were linked to a route table with an Internet Gateway to allow internet access. The private subnets were connected to a separate route table that restricted any external connectivity, enhancing internal resource security.

**Figure 5: OCI VCN and Subnet Configuration**

**E. OCI Security Lists and NSGs**

In OCI, security was managed using a combination of Security Lists (SLs) and Network Security Groups (NSGs). The Web-tierNSG was configured to allow HTTP (port 80) and SSH (port 22) from any external IP. It also allowed ICMP traffic from internal private subnets and permitted MySQL traffic on port 3306 specifically from the AWS subnet range.

The TestInstanceNSG was designed to allow all traffic internally, useful for connectivity checks. The Security List for Private Subnet 1 was very restrictive and only allowed incoming traffic from the AWS web server’s IP range, ensuring only secure and intended access.

**Figure 6: NSG Rules and Security List Rules for OCI**

**III. DATABASE AND OBJECT STORAGE CONFIGURATION (OCI)**

**A. MySQL Database Deployment**

Within OCI, a managed MySQL instance was created using version 8.0.41 and the MySQL2 shape with 50GB of allocated storage. This instance was deployed into Private Subnet 1. Importantly, public access was disabled for this database to enhance security. The database was named photoalbum, and it contained a single table called photosDB. This table had the following fields: photo\_title for the name of the photo, description for a short caption, creation\_date for when the image was taken, keywords for search-related tags, and object\_reference, which stored the public URL of the image.

phpMyAdmin was used to manage the database and was accessed through the web server on AWS, connecting securely over the VPN.

**Figure 7: OCI MySQL Instance Configuration and Table Schema**

**B. Object Storage Configuration**

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. Six sample images were uploaded through the OCI console and tested using an incognito browser window to confirm public access worked as intended.

**Figure 8: Public Access Settings and Uploaded Images in Object Storage**

**IV. SITE-TO-SITE VPN CONFIGURATION**

**A. On OCI**

The first step to establish the VPN was to create a Dynamic Routing Gateway (DRG) 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) object was created in OCI using the AWS VPN 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.

**Figure 9: DRG, CPE, and VPN Setup in OCI**

**B. On AWS**

In AWS, a Virtual Private Gateway (VGW) was created and attached to the VPC. A Customer Gateway (CGW) was then defined using the public IP of the OCI VPN endpoint. 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 172.17.0.0/16 through the VPN tunnel.

Tunnel status was confirmed to be “UP” on both sides through the AWS and OCI dashboards.

**Figure 10: VPN Setup in AWS with Tunnel Configuration**

**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.

**Figure 11: PHP Application Configuration to Connect OCI Database**  
**Figure 12: Webpage Displaying Images and Metadata Fetched from OCI**

**VI. TESTING AND VALIDATION**

Testing was performed to ensure each part of the deployment worked as expected. A ping test was done from the AWS EC2 instance to the OCI database private IP to confirm tunnel connectivity. The MySQL database connection was verified through both phpMyAdmin and direct terminal commands. The web application’s frontend was tested in a browser to ensure images displayed properly with the correct metadata. Finally, the tunnel connection was validated using AWS and OCI monitoring dashboards.

**Figure 13: Ping Results and Tunnel Verification Screenshots**

**VII. COMPARISON OF CLOUD CONNECTION METHODS**

Different methods were explored for connecting cloud environments. DirectConnect, offered by AWS, is a dedicated, high-speed line suitable for large enterprises but comes at a high cost. FastConnect is OCI’s version of DirectConnect and serves the same purpose. On the other hand, IPsec VPN uses encrypted communication over the public internet. Although it is slightly slower, it is free, flexible, and ideal for academic or small business use cases. For this assignment, IPsec VPN was selected due to its ease of configuration and zero cost.

**VIII. CHALLENGES AND LEARNINGS**

Several challenges were encountered during this deployment. The VPN connection failed initially due to missing route entries in the AWS and OCI routing tables. This was fixed by properly adding routes for each other’s IP ranges. Another issue involved the OCI database not responding to the web server, which was traced to a missing rule in the NSG that allowed port 3306 from AWS. Once added, the connection worked smoothly.

A big learning was understanding how OCI and AWS differ in VPN setup and how route propagation works differently. The need for both inbound and outbound rules in Network ACLs and the stateless nature of firewalls were also valuable insights. This hands-on experience provided a deeper understanding of hybrid cloud architecture and security best practices.

**IX. CONCLUSION**

The final deployment demonstrated a secure, functional, and efficient multi-cloud setup. AWS hosted the web application, while OCI managed the MySQL database and image storage. A secure IPsec VPN tunnel ensured private communication between the two platforms. The solution fulfilled all project goals and gave practical experience in designing, building, and troubleshooting a hybrid cloud environment. This knowledge will be useful for future real-world applications of secure, scalable cloud systems.

**X. REFERENCES**

1. AWS VPN Documentation – <https://docs.aws.amazon.com/vpn/latest/s2svpn/>
2. OCI VPN Documentation – <https://docs.oracle.com/en-us/iaas/Content/Network/Tasks/settingupIPSec.htm>
3. OCI Object Storage – <https://docs.oracle.com/en-us/iaas/Content/Object/Concepts/objectstorageoverview.htm>
4. OCI MySQL Database – <https://docs.oracle.com/en-us/iaas/mysql-database/doc/index.html>
5. AWS Elastic IPs – <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html>