COS20019 – Assignment 3

Serverless/Event-driven Architectural Design Report

Bui Minh Thuan  
104486358  
Tutorial Time: Wednesday 10:00AM *Date of Submission: 28/07/2024*

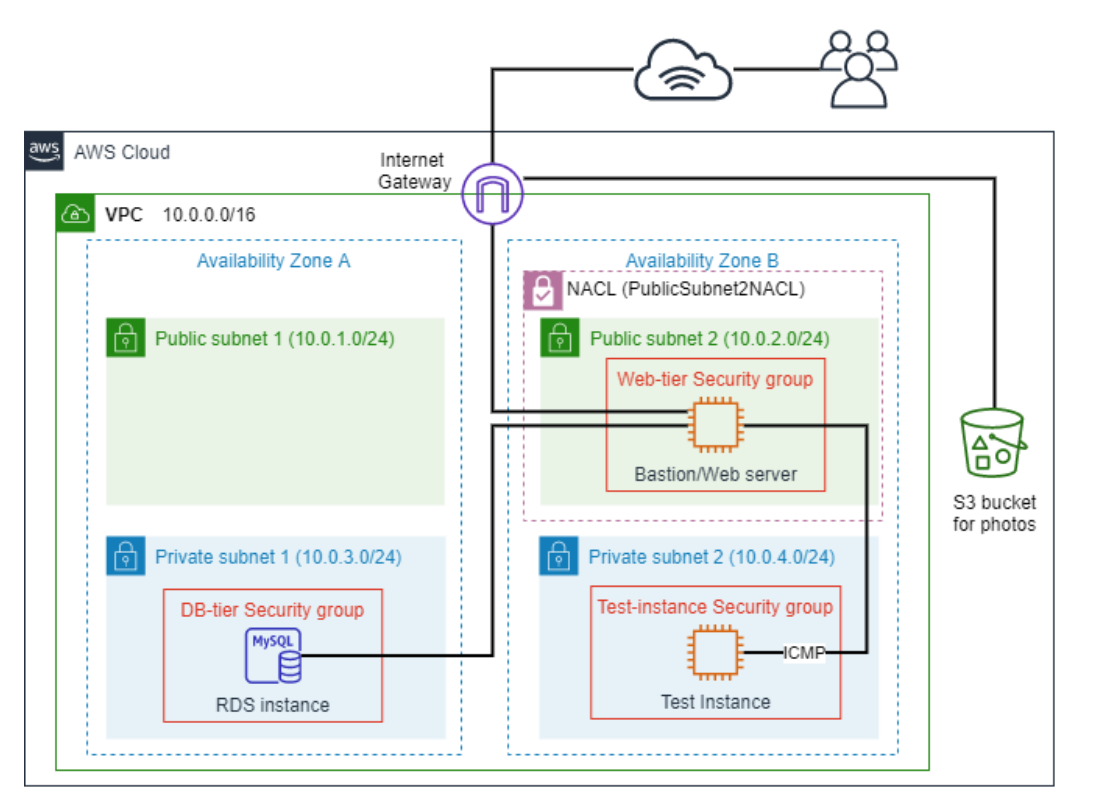
# Introduction

The Photo Album application, developed for seamless media management and sharing, has achieved remarkable success and the business requires advanced architecture to sustain its growth. This report presents a comprehensive cloud architecture design using Amazon Web Services (AWS) to ensure high performance, security, and global accessibility. The solution adopts a serverless, event-driven approach to minimize in-house administration and optimize resources. Key AWS services are selected to enhance content delivery, authentication, media processing, and data storage. Selection is based on design rationale, offering justification and comparison for the chosen AWS services, addressing performance, reliability, security, and cost considerations. This cloud architecture addresses current needs while providing flexibility and scalability for future expansion and feature enhancements

# Architecture Design

## Architectural diagram

In the first part, we propose our overall architecture design,….

  
Figure 1: Infrastructure diagram

Consider all the requirements in this business scenario disccuess below, we verify this diagram fullfil,….

## Requirements Fullfilment

Starting from Assignment 1B, Default VPC is no longer used, instead, in each task, we need to manually create the own VPC. In this Assignment, the created VPC will have 2 Availability Zones and 4 Subnets (2 public, 2 private). For route tables, 2 new route tables created, 1 for public and 1 for private. For the gateway, only use Internet Gateway.

### Web access

### Get

### Web access

### Get

### Web access

### Get

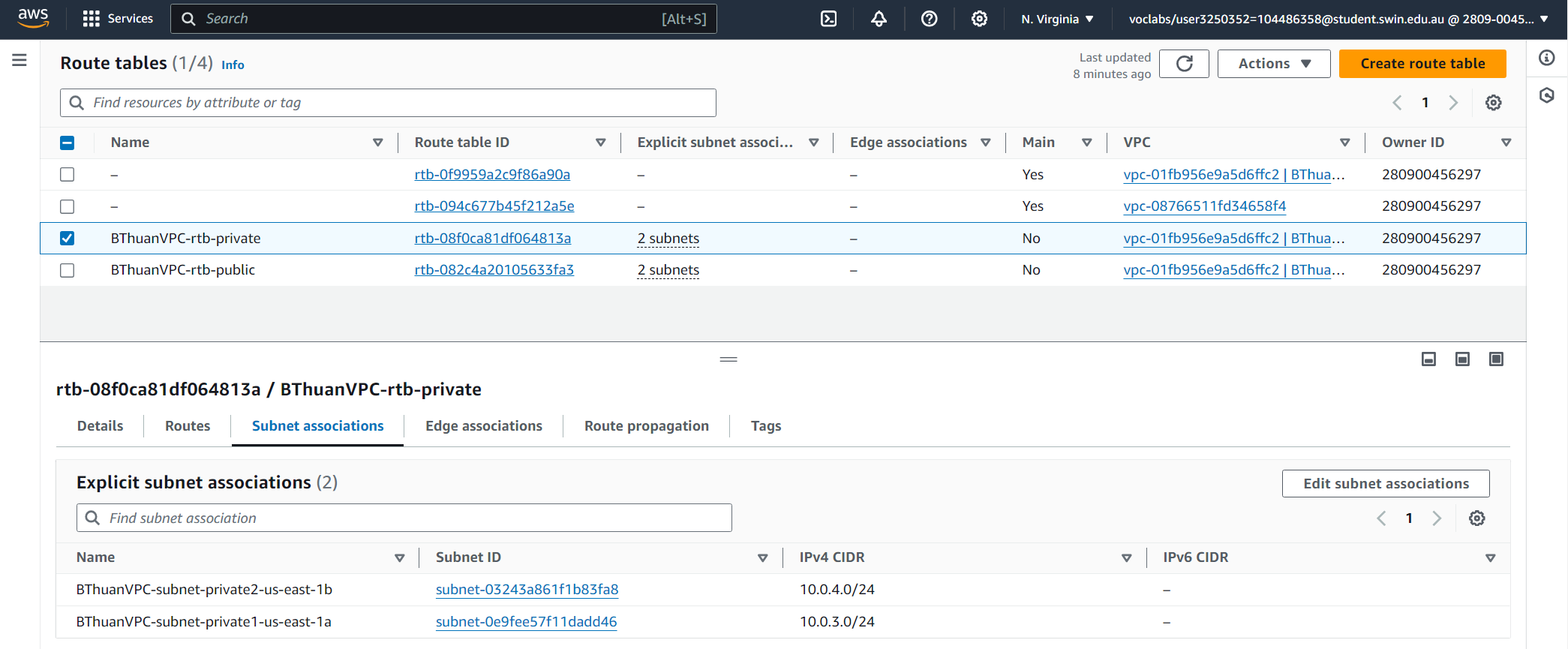
### Web access

### Get

## Service Description

Starting from Assignment 1B, Default VPC is no longer used, instead, in each task, we need to manually create the own VPC. In this Assignment, the created VPC will have 2 Availability Zones and 4 Subnets (2 public, 2 private). For route tables, 2 new route tables created, 1 for public and 1 for private. For the gateway, only use Internet Gateway.

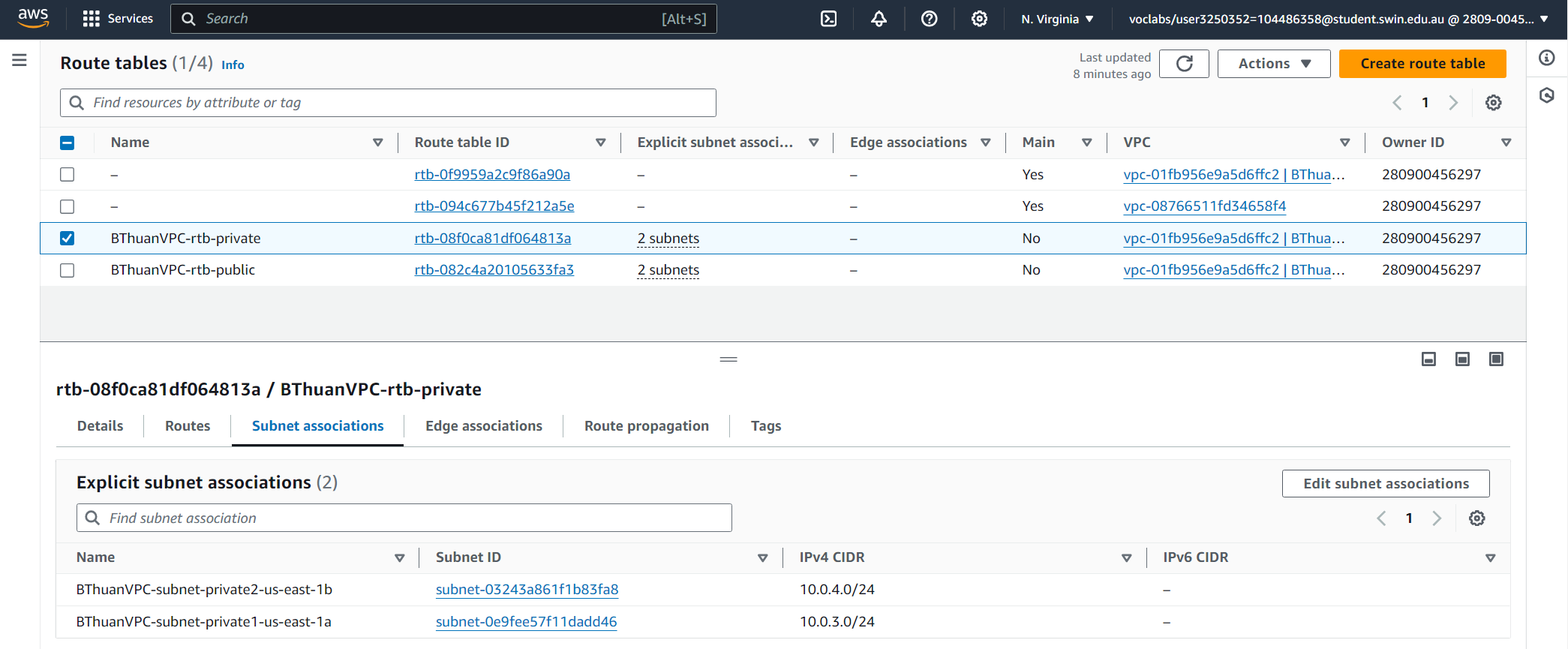
This step is not difficult, but we need to be careful when set up rules for the routable as well as assign IP to each subnet, ensuring each subnet gets the accurate range of IP as required in the architecture diagram, as well as subnet corresponds to the correct route tables.

  
Figure 6: Private subnet 1,2 associates with Private Route Table

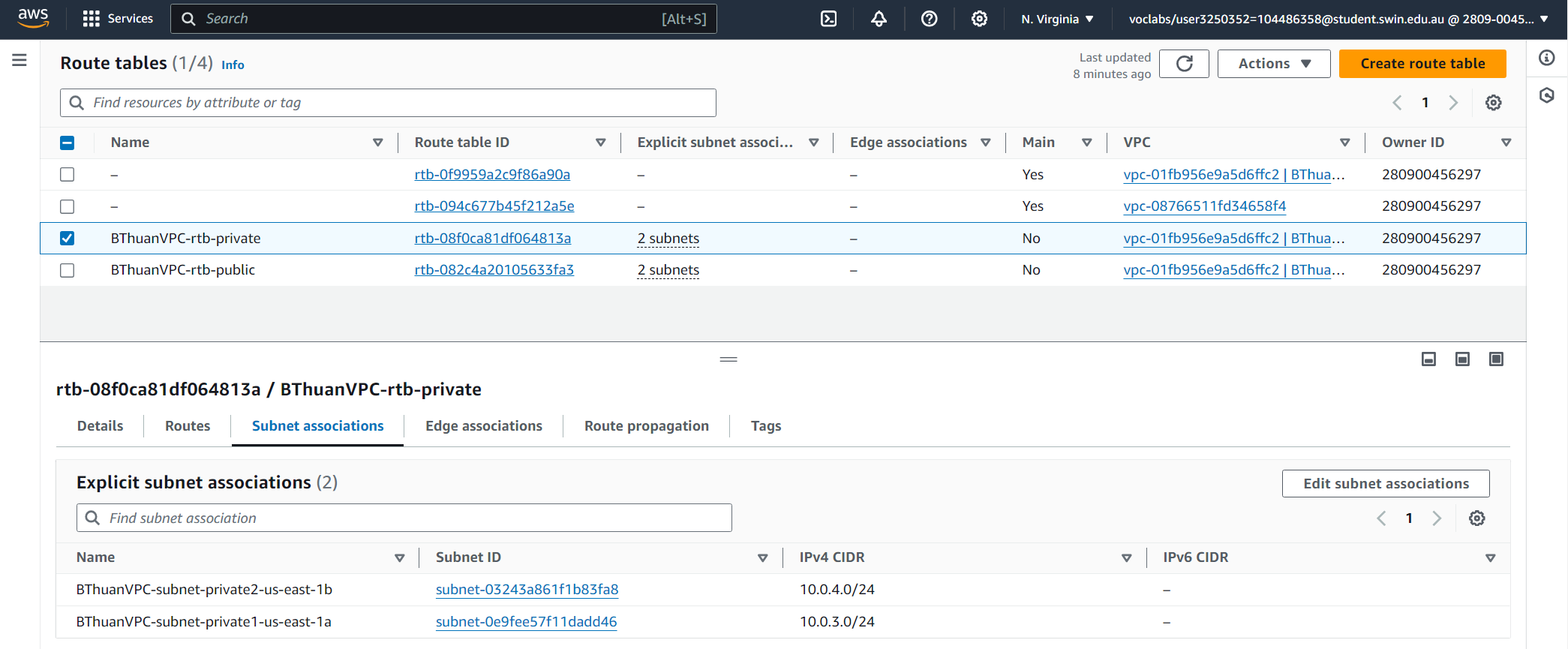
## Collaboration Diagram

Starting from Assignment 1B, Default VPC is no longer used, instead, in each task, we need to manually create the own VPC. In this Assignment, the created VPC will have 2 Availability Zones and 4 Subnets (2 public, 2 private). For route tables, 2 new route tables created, 1 for public and 1 for private. For the gateway, only use Internet Gateway.

### Authentication

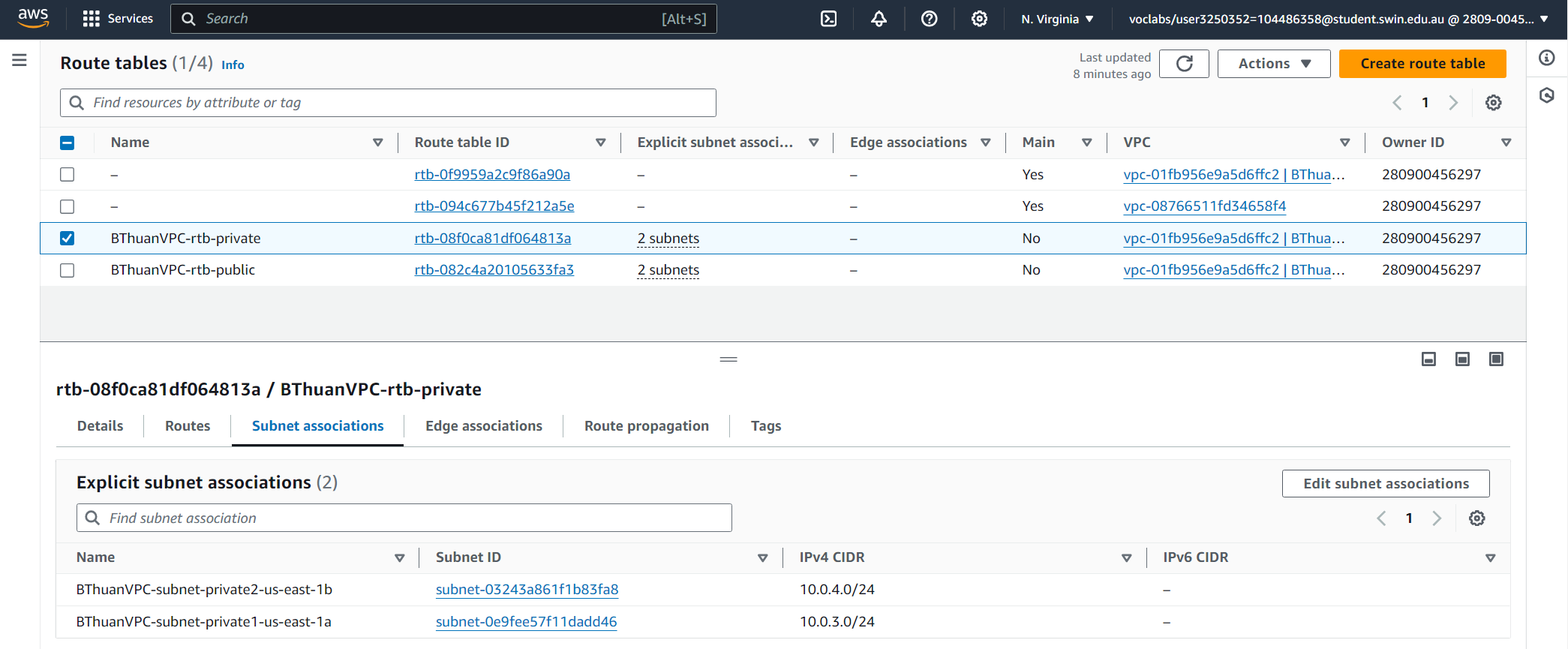
  
Figure 6: Private subnet 1,2 associates with Private Route Table

### Web Access

  
Figure 6: Private subnet 1,2 associates with Private Route Table

### Get Image

### Upload

  
Figure 6: Private subnet 1,2 associates with Private Route Table

# Design Rationale

After setting up the AWS architecture, the second part of the assignment focuses on deploying the website in this architecture. In this assignment, the website is a photo album, in which images are stored on AWS, and will be displayed whenever user access the web page. Most of the code for the website is already provided in the instruction folder, so my main tasks will be modifying the code to match the architecture, deploying it to the web server, and testing it functionality.

## Design Justification

We define standards in our solution to successfully meet the daily operational needs of the PhotoAlbum website according to business requirements, ensuring a secure, scalable, high-performance, and cost-effective cloud architecture. Our serverless architecture is designed with a clear workflow for each microservice, divided into three tiers—presentation, application, and data—making management and website function implementation convenient. ????

### **Operational Excellence**

We define standards in our solution to successfully meet the daily operational needs of the PhotoAlbum website according to business requirements, ensuring a secure, scalable, high-performance, and cost-effective cloud architecture. Our serverless architecture is designed with a clear workflow for each microservice, divided into three tiers—presentation, application, and data—making management and website function implementation convenient.

New feature development and code deployment are efficiently managed using AWS CodeCommit. CodeCommit is our fully managed source control service that hosts our Git repositories. By using CodeCommit, we enable PhotoAlbum developers to collaborate, commit code changes,  and manage branches. This facilitates continuous integration and delivery, ensuring the website can rapidly evolve according to business requirements.

To maintain operational health and performance, we utilize Amazon CloudWatch and AWS CloudTrail. CloudWatch provides real-time monitoring of AWS resources and applications, collecting and tracking metrics, logging files, and setting alarms. With CloudTrail, we can track system changes and analyze their effects on system health and performance. Amazon Route 53 further contributes to our operational excellence by performing health checks of web endpoints. Our approach enables us to respond quickly to events and adapt to the evolving business needs of the PhotoAlbum website

### **Security**

We have implemented a robust identity and access management system to ensure the security of user data. To secure user authentication sessions, we use Amazon Cognito, which provides access tokens for authenticated users who match the designated user pool. These users are then granted temporary credentials associated with IAM roles created by Cognito, which restrict their access to only the necessary PhotoAlbum website services to manage permissions. Sensitive data such as tokens and generated temporary symmetric keys are encrypted with AWS KMS in transit and at rest, ensuring comprehensive data protection. Asymmetric keys are used to encrypt the symmetric keys generated from Cognitoa and to verify digital signatures and data integrity.

Our multi-layered security infrastructure includes Amazon WAF (Web Application Firewall), which has managed rules by AWS experts to automatically detect and mitigate SQL injection and XSS attacks. We also use WAF to filter web traffic such as HTTP headers and mitigate DDOS attacks by setting thresholds for requests from individual IP addresses. We monitor our architecture security using CloudWatch and CloudTrail. AWS CloudTrail captures and logs all API activity within our environment, providing detailed visibility into user actions and service interactions for auditing and threat detection. Meanwhile, AWS CloudWatch monitors system performance and generates alarms based on specific log patterns and metrics, enabling us to detect and respond to security events in real time.

### **Reliability**

We focus on ensuring the reliability, continuous operation, and robustness of the PhotoAlbum website, even under adverse conditions. To achieve this, we have implemented several AWS services designed to handle failures automatically and recover reliably: AWS Lambda, S3, and SQS. These serverless services are inherently resilient, managed by AWS to recover from failures without requiring manual intervention. For example, Lambda automatically retries failed executions, S3 ensures data durability with multiple copies across different locations, and SQS manages message durability and availability. DynamoDB plays a critical role in our reliability strategy by replicating data across multiple regions working alongside our serverless architecture’s automatic scaling.

Additionally, DynamoDB’s horizontal scaling capability partitions data to distribute the load, which increases system availability by adding more resources as needed, rather than upgrading existing ones. These resources are cached using CloudFront at edge locations globally, reducing the load on the origin servers, and ensuring content availability even during high-traffic periods. This cross-region replication enhances data availability and fault tolerance, ensuring that data remains accessible even if a region experiences an outage. To handle web errors,  we utilize Route 53 DNS failover routing’s automatic redirection of traffic to alternate locations in case of errors, ensuring high availability and reliability.

We avoid the challenges of capacity planning by using CloudWatch for comprehensive monitoring and insights into our system’s performance to help us make informed decisions and manually adjust resources as needed. Our automated deployment and scaling strategy further reduce human error and enhance reliability. This automation ensures that the website remains functional and responsive, even in the event of crashes or other disruptions.

### **Performance Efficiency**

For performance efficiency, we have implemented serverless technologies, leveraging AWS Amplify to host our website with asynchronous Lambda processing. This approach eliminates the overhead of server management and ensures that our resources are dynamically allocated based on real-time demand. This aligns with the principle of democratizing advanced technologies, making sophisticated, scalable solutions accessible and easy to implement using AWS-managed services. We selected DynamoDB, known for its exceptional performance and scalability, efficiently handles large volumes of data with low latency while seamlessly scaling horizontally to manage increased traffic. By utilizing DynamoDB's on-demand capacity mode, we align with the principles of mechanical sympathy, avoiding the complexities of manual capacity planning. This approach ensures our database operations are inherently efficient and adapt dynamically to workload demands, enhancing overall performance,

Our website’s main performance comes from executing Lambda function code in response to various triggers, such as changes in data or system states, and user actions. In our architecture, we incorporate a pub/sub mechanism with Amazon SNS to publish messages and notify AWS SQS, which then triggers Lambda functions to process tasks asynchronously. This setup allows us to handle numerous user requests and uploads concurrently, maintaining high performance and ensuring the system remains responsive under heavy load. This design also supports frequent experimentation by allowing quick implementation and function testing.

Our global reach is enhanced through CloudFront, which allows our website to go global in minutes alongside our serverless automatic scaling and global content delivery. By caching content close to users, CloudFront offers high performance, and significantly reduces latency, providing a fast and seamless user experience regardless of their geographic location. We used CloudWatch to monitor real-time metrics and insights regarding system performance, enabling us to maintain perform efficiency of the application architecture

### **Cost Optimization**

<https://aws.amazon.com/pricing/>

<https://calculator.aws/#/>

We reduce costs through our serverless architecture by leveraging the pay-as-you-go approach, ensuring we only pay for the computing time used rather than maintaining expensive, idle physical instances. By choosing the on-demand capacity mode for DynamoDB, we eliminate the need for capacity guessing, reducing costs by charging only for actual read and write requests. Additionally, our managed services such as AWS SNS, SQS, and DynamoDB offload the responsibility of maintaining and updating infrastructure to AWS, lowering the need for IT experts and increasing our ROI. To manage cost-effectiveness, we use AWS Cost Explorer and CloudWatch to monitor monthly costs and application statistics, ensuring optimal resource utilization and continuous cost monitoring.

## Alternative Solutions

AWS offers a wide range of services for various kinds of workload. For a same task, we can easily find multiple solutions with its own unique benefits. Therefore, comparing between these services are necessary when propose an architecture design?????

### **Architecture Design: 3-tier architecture or 2-tier architecture**

<https://byjus.com/gate/difference-between-two-tier-and-three-tier-database-architecture/>

<https://medium.com/@paulndemo/2-and-3-tier-architecture-4a473e5ced3d>

In a two-tier architecture, the client communicates directly with the server. While this is simpler to build and maintain, it can lead to performance bottlenecks and security issues, especially when the number of users increases.

Meanwhile, in a three-tier architecture, the application is divided into three layers: presentation layer (frontend), the logic layer (backend), and the data layer (database). This separation allows three-tier architecture to operate at a comparatively faster pace due to its ability to efficiently distribute and manage the load across multiple layers without overloading any single component. A three-tier architecture enables independent upgrading or replacement of any one of the three tiers, enhancing flexibility and maintainability. This allows the system to scale more effectively, avoiding degraded performance by managing separate connections for each client through distinct layers. Additionally, by preventing direct client-database communication it is comparatively more secure in the long run.

Therefore, for our use case of providing low-latency content delivery for the Photo Album application, a three-tier architecture is the optimal choice.

### **Virtual machine vs Serverless computing**

<https://aws.amazon.com/serverless/>

<https://www.techrepublic.com/article/serverless-computing-pros-and-cons-5-benefits-and-3-drawbacks/>

The company’s system, running on t2.micro EC2 instances with over 80% capacity usage, is slow and costly. They seek a serverless solution to improve performance, reduce costs, and enhance global response times.

Serverless services offer a pay-as-you-go model, so the company only pays for resources when they are used, with no costs for idle run time, this helps optimize expenses. Serverless solutions can automatically scale to handle sudden traffic spikes without manual configuration. Services like AWS Lambda can scale from 0 to thousands of requests per second.

For low latency, we use serverless functions to greatly reduce the latency experienced by end users because serverless functions don’t operate from an origin server, so there’s no single location to which an end user’s traffic has to be directed.

We design a serverless architecture to help the company reduce costs, improve performance and deliver a superior user experience by addressing latency issues and ensuring efficient scaling.

### **Caching Options: Cloudfront vs ElastiCache**

Providing users with low-latency content delivery is crucial for the Photo Album application due to its requirement for quick image and video access. The two popular caching options for AWS are Amazon CloudFront and Amazon ElastiCache. Understanding the use cases and differences between these services is essential for optimizing the application's performance and user experience.

Amazon ElastiCache is a managed service that provides in-memory data to cache database queries and session data, reducing the load on the primary database by storing frequently accessed data in memory. ElastiCache is suited for real-time fetching of dynamic data from databases rather than static files. However, for our use case, the primary caching needed is for static content delivery (i.e., images, videos) rather than the queries.

On the other hand, AWS CloudFront caches image and video files at edge locations worldwide, which is essential for our application's requirement for global accessibility and low latency.  It also integrates seamlessly with other AWS services, enhancing scalability, reliability, and security. Therefore, CloudFront is the optimal choice for our use case of caching static media files

### **Database: SQL vs NoSQL**

Efficient database management is crucial for the Photo Album application due to its requirement for quick media access. Two popular database options for AWS are Amazon RDS for SQL databases and Amazon DynamoDB for NoSQL databases. Understanding the use cases and differences between these services is essential for optimizing the application's performance and user experience.

The company currently uses a relational database with Amazon RDS which is optimized for structured data and complex queries rather than unstructured data. However, the current setup has proven to be costly and slow for our use case of storing and retrieving large media files (images, videos). Although SQL databases maintain data integrity and complex relationships, they lack the ability to store and handle the disparate media data required by the website.

Amazon DynamoDB, on the other hand, provides near-real-time response time and predictable performance with serverless scalability. It supports key-value and document data models, making it ideal for applications requiring quick read/write access to large volumes of data. Furthermore, DynamoDB is a NoSQL database, which means it’s more flexible and cheaper to scale horizontally using distributed clusters compared to RDS SQL vertical scaling of expensive servers. For our use case, DynamoDB’s ability to handle rapid access to stored media and scale to meet increasing demands makes it the optimal choice.

### **Web hosting: Amplify vs Amazon S3**

Both Amazon S3 and AWS Amplify are used for web hosting. While S3 provides a simpler solution for hosting static websites, Amplify offers a more comprehensive and suitable platform for complex and scalable web applications.

Amazon S3 (Simple Storage Service) does have the capability to host static websites, making it a reliable option for serving HTML, CSS, and JavaScript files. However, hosting websites is not its primary function, making it harder to extend and integrate with other AWS resources compared to Amplify.

On the other hand, AWS Amplify provides a comprehensive set of tools and services for building, deploying, and scaling full-stack web and mobile applications. Amplify’s managed services minimize the need for in-house systems administration, making it perfect for businesses seeking to streamline operations and focus on development. Unlike S3, Amplify offers features like seamless integration with other AWS services, serverless computing with AWS Lambda, and robust authentication and analytics. This makes Amplify particularly suitable for your rapidly growing Photo Album application, which requires a scalable, serverless, and event-driven architecture to support current and future needs, making it a superior choice over S3 for hosting static websites

### **Media transcoding: Elastic Transcoder vs AWS MediaConverter**

AWS Elemental MediaConvert and Amazon Elastic Transcoder are both AWS services designed for transcoding media files stored in S3 into formats suitable for various playback devices. Comparing these two is essential for determining which service best meets specific business requirements, particularly for applications with increasing demand and diverse media needs.

Amazon Elastic Transcoder is a straightforward, cloud-based media transcoding service. It offers a simple setup with presets for common formats, allowing easy conversion of media files into multiple formats within a single job. While it is practical for basic transcoding needs, it lacks advanced control over encoding settings and supports fewer input and output formats, which can be limiting for more complex requirements.

AWS Elemental MediaConvert, however, is more advanced and flexible. It supports a broader range of formats, provides finer control over video quality and encoding settings, and includes features like adaptive bitrate streaming and built-in content encryption. For the Photo Album application, which anticipates exponential growth and diverse media formats, MediaConvert is more suitable. Its scalability, advanced features, and automatic media processing upon S3 upload make it ideal for handling increased demand, ensuring high-quality delivery, and accommodating future needs like video transcoding and additional processing capabilities.

### **Decoupling architecture: AmazonSNS vs AWS EventBridge**

When designing serverless applications, it's essential to consider why implementing either the pub-sub or queue pattern is more beneficial than calling Lambda functions directly. Direct invocation can lead to tight coupling between services, introducing several drawbacks such as increased dependencies, reduced modularity, and harder maintenance. Additionally, relying on a single endpoint can create bottlenecks, leading to performance issues and potential failures under high load. Messaging patterns like queue and pub-sub effectively decouple services and distribute the load more efficiently, enhancing the system's reliability and scalability. In our architecture, the queue pattern allows messages to be sent to a queue and processed asynchronously by consumer services, decoupling the producer (e.g., API Gateway) from the consumer (e.g., Lambda function), and buffering messages to smooth out traffic spikes. This prevents consumers from being overwhelmed and is particularly useful between API Gateway and Lambda functions. Conversely, the pub-sub pattern broadcasts messages to multiple subscribers simultaneously, enabling real-time updates and decoupled communication between producers and consumers. This is ideal for scenarios where multiple services need to react to events independently, such as in the transcoding section, where multiple services can process the same event concurrently and flexibly.

While both Amazon SNS (Simple Notification Service) and Amazon EventBridge can trigger Lambda functions, SNS is often the more practical choice for S3 and pub-sub architectures due to its simplicity and performance. EventBridge offers advanced routing and extensive AWS integrations, but for straightforward, many-to-many communication, SNS excels. Its ability to deliver messages to multiple subscribers, including Lambda functions, ensures prompt reaction to new S3 objects, enhancing system responsiveness and decoupling. SNS supports various communication protocols, making it versatile and efficient in distributing messages to multiple endpoints simultaneously. This makes SNS an ideal solution for ensuring immediate and reliable message delivery in pub-sub architectures, facilitating effective communication and timely processing tasks.

## Quantitive compare

The final step involves writing the code to display images and their meta-data when users open the website. Because the code has been provided in the instruction folder, I don’t need to write everything from scratch. What I need to do is adjust some constants, which includes student’s information (in this case is me, used to identify assignment) and database constants (used to connect to the database for this assignment).

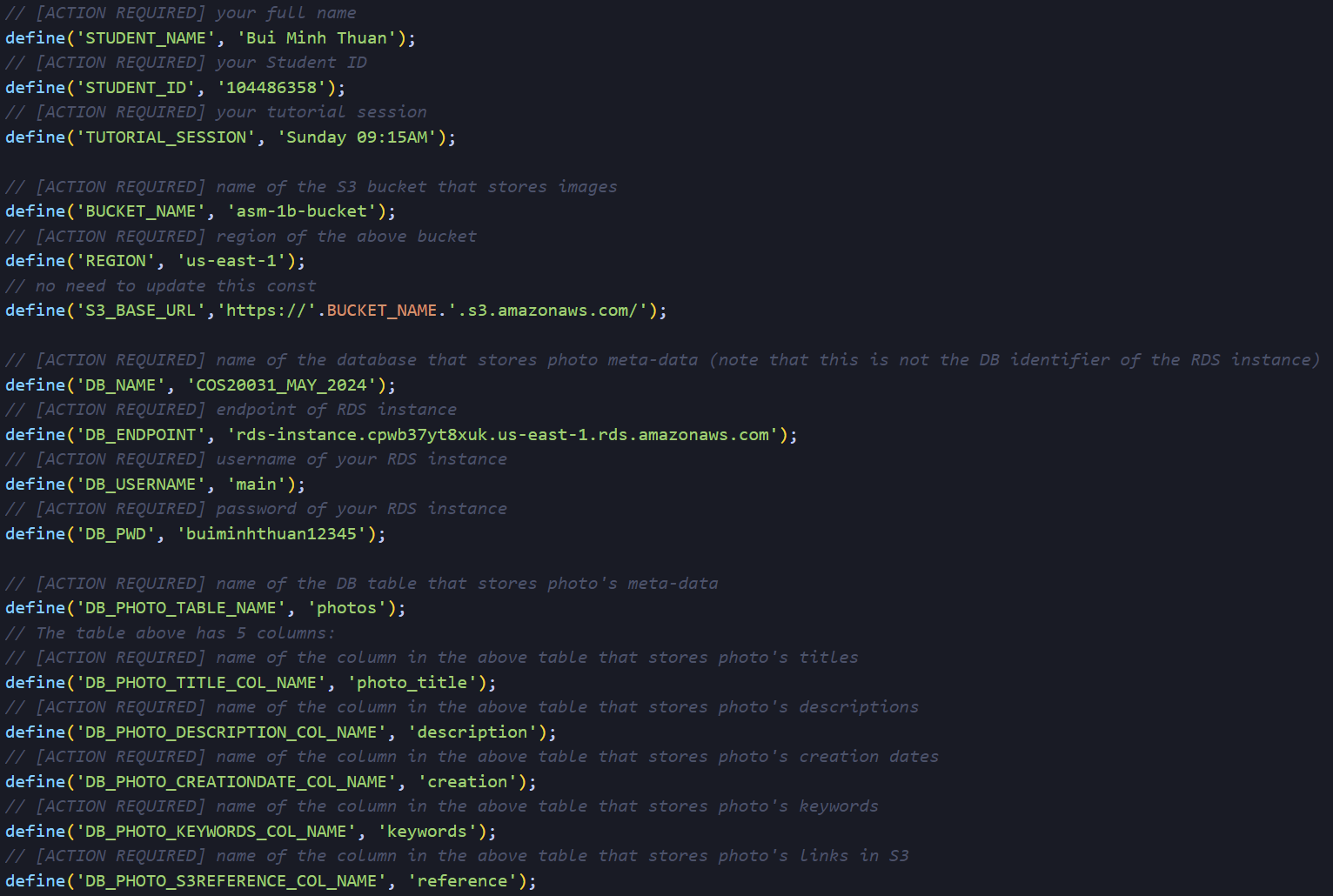


Figure 32: Updated constant variables

You can see the output above: my personal information on top, and a table contains all images in the album. In this table, the meta-data is fetched directly from the RDS database, while the image is taken from the link stored in it.

# Conclusion

Throughout this report, I summarized my work for all tasks in Assignment 1B – Unit COS20019 Cloud Computing Architecture. This report covered specifications details (VPC, EC2,.. settings), testing actions (web server access, ICMP, SSH remote connection,..) as well as problems I encounter during doing the assignment.

# References

1. Pope, M. (2014, May 21). *Securely Connect to Linux Instances Running in a Private Amazon VPC | Amazon Web Services*. Amazon AWS. Retrieved June 16, 2024, from <https://aws.amazon.com/vi/blogs/security/securely-connect-to-linux-instances-running-in-a-private-amazon-vpc/>
2. Amazon Web Services. (2021). Network ACLs. In Amazon Virtual Private Cloud User Guide. Retrieved June 16, 2024, from <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html#nacl-ephemeral-ports>.