Kamal Sigdel

Design and Implementation of a Secure and Cost-Effective Three-Tier Cloud Network on AWS

Metropolia University of Applied Sciences

Bachelor of Engineering

Degree Programme in Information Technology

Bachelor's Thesis

30 March 2026

Abstract

Author:	Kamal Sigdel
Title:	Design and Implementation of a Secure and Cost-Effective Three-Tier Cloud Network on AWS
Number of Pages:	40 pages + 15 appendices
Date:	30 March 2026
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Smart IoT Systems: Networks
Supervisor:	Markku Niiranen, Senior Lecturer

This thesis presents the design and implementation of a secure and cost-effective three-tier cloud network architecture using Amazon Web Services (AWS). The objective of the study was to demonstrate how proper network segmentation and controlled communication between system components can improve security and fault isolation while maintaining low operational cost.

The work was carried out using a project-based approach. The architecture was first designed based on common cloud networking principles and then implemented in a real AWS environment. Core services such as Virtual Private Cloud, Amazon instances, and Amazon Relational Database were used to build separate web, application, and database tiers. The system was evaluated through practical connectivity and security testing to verify correct communication between tiers and protection against unauthorized access. A cost analysis was also conducted based on actual usage and AWS pricing data.

The findings show that the implemented three-tier architecture functions correctly. The web tier allowed internet access, while the application and database tiers were secured in private subnets. Internal communication between tiers was limited to private IP addresses, and unauthorized access attempts were successfully blocked. In addition, the cost analysis confirmed that these security and segmentation goals were achieved without introducing significant monthly expenses.

The study demonstrates that secure and well-structured cloud network architectures can be implemented in a cost-effective manner when resources are carefully sized and managed. The proposed solution provides a practical reference model suitable for small organizations, educational environments, and budget-constrained projects.

Keywords: Amazon Web Services, Cloud Computing, Cloud Security, Cost Optimization, Network Segmentation, Three-Tier Architecture.

The originality of this thesis has been checked using the Turnitin Originality Check service.

Contents

Appendices

List of Abbreviations

AWS:	Amazon Web Services
AZ:	Availability Zone
DBIR:	Data Breach Investigations Report
EC2:	Elastic Compute Cloud
IaaS:	Infrastructure as a Service
IGW:	Internet Gateway
IP:	Internet Protocol
NAT:	Network Address Translation
PSQL:	Postgresql
RDS:	Relational Database Service
USD:	United States Dollar
VPC:	Virtual Private Cloud

1 Introduction

Cloud computing has become a widely adopted approach for deploying modern information systems due to its flexibility, scalability, and on-demand availability of resources. Instead of investing in physical infrastructure, organizations can use computing and networking resources as per their needs and pay for the usage. This makes cloud platforms attractive for startups, small organizations, and educational projects with limited budgets and operational resources. [1,2.]

Despite these advantages, moving systems to the cloud does not automatically ensure security. In practice, the security of a cloud-based system depends heavily on how the network is designed. Many small-scale deployments follow a minimal approach in which web services, application logic, and databases are placed within the same network layer. While this simplifies initial deployment, it often results in weak isolation between system components. According to industry reports, many cloud security issues are caused by exposed services, excessively permissive network rules, and insufficient network segmentation. These issues increase the risk of misconfiguration, unauthorized access, and poor fault isolation. [3,4,5,6,7,8.]

A common reason for this approach is the assumption that secure cloud architectures are complex and expensive to implement [9,10]. Advanced designs often rely on multiple managed services and high-availability features, which may not be suitable for small-scale environments due to cost constraints. Cloud cost management studies consistently report that a significant portion of cloud expenditure is caused by overprovisioned resources and always-on managed services that exceed actual workload requirements. [11,12,13.] As a result, security and cost are often perceived as opposing goals, leading to compromises in network segmentation and access control.

This study aims to design and implement a secure, cost-effective three-tier cloud network architecture on Amazon Web Services (AWS) that separates the

web, application, and database layers into distinct network segments with clearly defined communication paths. The architecture allows internet access only to the web tier and keeps the application and database tiers private. The design strictly controls traffic flow between components using routing and access control mechanisms. The system is validated through practical connectivity and security testing to confirm that the design behaves as intended. The study also evaluates cost-related design decisions to show how unnecessary expenses can be avoided while maintaining essential security requirements.

This thesis follows a project-based approach in which the architecture is first designed according to established cloud networking principles and then implemented using standard AWS services. The scope is limited to the network and infrastructure layer, including the configuration of a Virtual Private Cloud (VPC), public and private subnets, Elastic Compute Cloud (EC2) instances for web and application tiers, and Relational Database Service (RDS) database deployed in private subnets, and security groups to enforce access control. The architecture is intended as a small-scale reference solution suitable for budget-constrained projects, educational use, startups, and proof-of-concept environments. Advanced enterprise features such as load balancers, auto scaling, continuous monitoring and logging platforms, penetration testing, compliance validation, and multi-region or high-availability deployments are intentionally excluded, as they fall outside the scope of a cost-efficient reference architecture.

This thesis is organized into seven chapters. Following this introduction, the theoretical background related to cloud computing, AWS networking, and three-tier architectures is described in Chapter 2. Similarly, the materials, tools, and methods used in the implementation are presented in Chapter 3. After that, the analysis of the existing cloud architecture approaches and their limitations is presented in Chapter 4. The proposed solution and implementation details are explained in Chapter 5. The results of the tests and the findings are discussed

in Chapter 6. Finally, Chapter 7 concludes the study and outlines possible directions for future work.

2 Theoretical Background

This chapter describes the key concepts behind the design and implementation of the proposed cloud network architecture. The focus is on practical concepts that support the implementation work presented later in the thesis.

2.1 Basics of Cloud Computing

Cloud computing is a way of providing computing resources, such as servers, networks, and storage, as services over the internet. Instead of handling physical hardware themselves, users can use resources whenever needed from a cloud provider. [2,14.]

A key benefit of cloud computing is that users are charged only for the resources they use. However, security and reliability depend on the design of the cloud infrastructure, especially on the network structure [2,14,15,16]. The differences between the traditional on-premises infrastructure and cloud computing in terms of ownership, cost, security, scalability, deployment time, and maintenance are specified in Table 1.

Table 1. Comparison between traditional infrastructures and cloud computing.

Basis for comparison	Traditional on-premises infrastructure	Cloud computing
Infrastructure ownership	Hardware is owned and maintained by the organization.	Cloud providers own and manage the infrastructure.
Cost model	Fixed costs regardless of actual usage.	Cost incurred based on usage of the resources.
Security responsibility	Fully managed by the organization.	Managed by both providers and the users with shared responsibilities.
Scalability	Scaling can be done by purchasing and installing new hardware.	Resources can be scaled on demand.
Deployment speed	Deployment can take from days to months.	Resources can be deployed within minutes.
Maintenance	The organization is responsible for maintaining hardware and upgrades.	Maintenance is handled by the cloud provider.

As shown in Table 1, cloud computing is more flexible than traditional on-premises infrastructure. Resources can be used only when needed, systems can be set up quickly, and much of the maintenance is handled by the cloud provider. This makes cloud computing easier to manage and more cost-effective, especially for small or growing systems.

2.2 Introduction to Amazon Web Services

Amazon Web Services is a leading cloud platform, providing a broad set of services for computing, networking, storage, and security. AWS allows users to build systems without owning physical hardware and to control how resources are accessed and connected. In this study, AWS was selected mainly because it supports the exact network components needed to build and test a secure

three-tier architecture, such as a VPC, subnets, routing, and security controls [16].

Furthermore, AWS is widely adopted in organizations and is frequently used for significant workloads, which improves the real-world relevance of the implemented solution [17]. Table 2 compares major cloud platforms to justify the reason for selecting AWS as a cloud platform for this study.

Table 2. Reasons for choosing AWS as a cloud platform.

Basis for comparison	Amazon Web Services	Microsoft Azure	Google Cloud
Industry Adoption	Very widely used [17].	Very widely used but less frequently than AWS [17].	Very widely used, but less frequently than AWS and Microsoft Azure [17].
Market Position	Leader in Information as a Service (IaaS) provider [18].	Second-largest IaaS provider [18].	Among the top IaaS providers [18].
Fit for this thesis (network segmentation + tier isolation)	VPC enables isolated networking, subnets, routing, and security controls [6].	Comparable capabilities (not evaluated in this study).	Comparable capabilities (not evaluated in this study).

As shown in Table 2, Amazon Web Services was selected because it is widely adopted in the industry and provides strong support for network segmentation and tier isolation. Services such as VPC, EC2, and RDS support the architectural goals of this study and were already familiar through coursework at Metropolia. This combination of industry relevance, suitable technical features, and the author's prior knowledge makes AWS an appropriate platform for the implemented solution.

2.3 Virtual Private Cloud

Amazon Virtual Private Cloud enables users to build a private and isolated network inside AWS. It functions like a private network where users can set IP address ranges, subnets, routing, and access rules [6]. By default, VPCs isolate cloud resources from other AWS customers. This isolation forms the foundation for building secure architectures, as all communication entering or leaving the VPC can be explicitly controlled.

2.4 Subnetting and CIDR Addressing

Within a VPC, the network is divided into smaller segments called subnets. Each subnet is associated with a specific IP address range defined using Classless Inter-Domain Routing (CIDR). Subnetting allows resources to be grouped based on their role and access requirements. [19.] Subnets are divided into public and private subnets. Public subnets allow access to the internet, while private subnets are used for internal resources that should not be reachable from outside the network. This division is a key principle in secure cloud network design. [20.]

2.5 Three-Tier Architecture Model

In a three-tier architecture, a system is divided into the web, application, and database layers. Each layer or tier has a specific responsibility and communicates only with the tiers it depends on. The web tier handles user requests and is typically exposed to the internet. The application tier handles business logic and connects the web tier to the database. The database tier stores sensitive data and should remain private. This model improves security by reducing direct access to sensitive components and simplifying system maintenance by clearly separating responsibilities. [21.]

2.6 Network Segmentation and Security

Network segmentation involves separating a network into smaller areas to control the flow of traffic between them. In cloud environments, this is usually achieved using subnets, route tables, and security groups. Subnets are used to separate resources based on their role, such as public web servers and private backend systems. Route tables define where network traffic is allowed to go, while security groups control which connections are permitted between resources. [22,23.]

Security groups act like firewalls. They allow only the traffic that is needed and block everything else. By allowing communication only between required components, unnecessary access is reduced. This follows the principle of least privilege, where access is granted only when it is required. [22,23.]

2.7 High Availability and Fault Isolation

High availability means the ability of a system to continue to run even if part of the infrastructure fails. In AWS, this is possible by using more than one Availability Zone. Each Availability Zone (AZ) is physically separate, which reduces the risk of a single point of failure. [24,25.]

Fault isolation is improved when components are divided into different tiers and subnets. If an issue occurs in one tier, it is less likely to affect the other tiers. Although this study does not focus on enterprise-level availability features, basic availability principles are applied through multi-Availability Zone subnet design. [25.]

2.8 Cost Optimization in Cloud Network Design

Cost optimization is an important consideration when designing cloud networks. While cloud platforms offer many advanced features, not all of them are necessary for small-scale systems.

Cost-efficient design choices include selecting appropriately sized instance types, limiting the use of always-on managed services, and deploying databases in a single Availability Zone when high availability is not a primary requirement. By carefully choosing only the components required for the intended workload, it is possible to design a secure cloud architecture without incurring unnecessary cost. [15.]

3 Materials and Methods

This chapter describes how the study was carried out, including the methodology, tools, and technologies used, and the approach to implementation and evaluation. The purpose of this chapter is to explain how the proposed solution was designed, implemented, and validated.

3.1 Working Method

This study followed a project-based working method that combines conceptual design, practical implementation, and analytical evaluation. The work was carried out in clearly defined phases to ensure a structured and systematic process.

Figure 1 presents the main phases used in this study, from the initial design stage to the final documentation and evaluation.

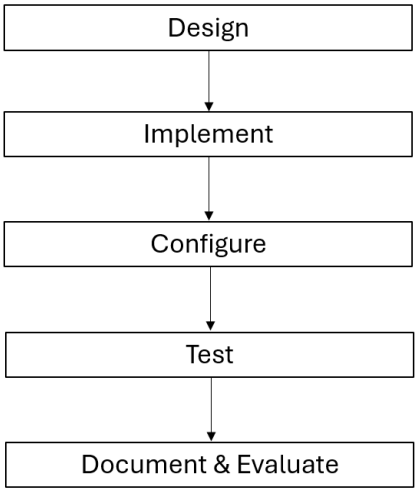


Figure 1. Main phases of the thesis work.

As shown in Figure 1, the process started with designing the cloud network architecture based on theoretical principles and best practices. The design was

then implemented in AWS by building the required network and infrastructure components. After implementation, security and access control were configured to enforce tier separation and least-privilege access. The solution was then tested through connectivity and restriction checks to confirm that allowed communication paths function correctly while blocked paths remain inaccessible. Finally, the results were documented and evaluated to assess whether the objectives related to security, functionality, and cost efficiency were achieved.

3.2 Tools and Technologies

The implementation of the three-tier architecture was carried out using AWS and supporting tools. AWS was selected because it provides mature networking and security services that support secure and cost-effective cloud architectures. The main tools and technologies used in this study are listed below.

- AWS was the cloud platform.
- VPC was used for network isolation.
- Public and private subnets were created to isolate the web, application, and database tiers.
- An Internet Gateway was configured to allow internet access to the web tier.
- Network Address Translation Gateway (NAT) was used to enable temporary outbound access from private subnets when needed.
- Amazon EC2 was used for hosting the web and application tier servers.
- Amazon RDS (PostgreSQL) was used for the database tier.
- Security Groups were created to control traffic flow between tiers.
- AWS Systems Manager Session Manager was used for secure administrative access without SSH exposure.
- Command-line tools such as curl and PSQL were used for connectivity and validation testing.

This set of tools covered the required implementation, security configuration, and testing tasks.

3.3 Design and Implementation Approach

The design and implementation followed the principle of least privilege, meaning each part of the system is allowed to communicate only with what it truly needs [26]. The solution follows a three-tier structure made up of the web tier, application tier, and database tier. The web tier is placed in public subnets and is the only part accessible from the internet. The application and database tiers are placed in private subnets, where the application tier has no direct internet access, and the database tier can only be accessed by the application tier.

The implementation was completed step by step. First, the network foundation was built by creating the VPC, subnets, and route tables. After that, the EC2 instances were deployed, and security groups were configured to control traffic between tiers. For administration, private instances were accessed using AWS Systems Manager instead of opening SSH to the public internet.

3.4 Evaluation and Testing Strategy

The implemented architecture was evaluated using practical connectivity and security testing. The evaluation focused on verifying that the architectural objectives of network segmentation, controlled inter-tier communication, and protection against unauthorized access were correctly implemented.

Testing was conducted through real connection attempts using standard command-line tools. This approach was selected to reflect actual system behaviour and to provide concrete evidence of how traffic is handled within the network, without introducing additional testing tools or services that would increase operational cost. The evaluation included the following tests:

- Verification that the web tier is accessible from the internet via its public IP address.
- Confirmation that the application tier is not accessible directly from the internet.

- Validation that the database tier is accessible only from the application tier.
- Confirmation that unauthorized access attempts from external and internal sources are blocked.

Both successful and failed connection attempts were documented to demonstrate that permitted communication paths function as designed and that restricted paths are effectively denied.

This testing approach is sufficient to support the conclusions of the study because it directly validates the core security mechanisms introduced by the three-tier architecture. The primary focus of the work is network-level security achieved through subnet isolation, routing, and security group rules. The performed tests confirm that these mechanisms enforce the intended access boundaries.

Advanced security testing areas, such as monitoring, logging and analysis, intrusion detection, penetration testing, vulnerability scanning, and compliance validation, were not included. These activities are typically part of operational security and continuous monitoring in production environments rather than architectural validation. In addition, encryption in transit and at rest is provided by AWS managed services by default, reducing the need for separate encryption testing within the scope of this study.

Overall, the evaluation strategy aligns with cloud networking best practices, in which validating connectivity, isolation, and access control is sufficient to confirm the correctness and security of a network architecture at the design and implementation levels [27].

4 Current State Analysis

This chapter reviews commonly used cloud architecture models and highlights their limitations in terms of security, scalability, and cost efficiency. The purpose of this analysis is to explain why simpler architectures may be insufficient for secure cloud deployments and to justify the need for the proposed three-tier solution.

4.1 One-Tier Architecture

In a one-tier architecture, all system components are deployed within a single layer. The web server, application logic, and database often run on the same instance or within the same network segment. This approach is simple to set up and requires minimal configuration. [28,29.]

While suitable for small experiments or prototypes, one-tier architectures offer very limited security. Because all components share the same environment, exposing the system to the internet also exposes internal services such as the database. Fault isolation is also weak, as a failure in one component can affect the entire system. [28,29.]

4.2 Two-Tier Architecture

A two-tier architecture divides the system into a frontend tier and a backend tier. The frontend interacts with the users, while the backend handles the system logic and stores data. In cloud environments, this often means distinguishing the web server from the database server. [28,29.]

Compared to a one-tier architecture, this model improves security by isolating the database from direct internet access. It also allows limited scalability and better performance tuning. As a result, two-tier architectures are commonly used in small to medium-sized applications. [28,29.]

However, the application logic and data layer are still tightly coupled, and internal network separation remains limited. As the system grows, access control rules become more complex, and fault isolation is still constrained when multiple responsibilities are handled within the same tier. [28,29.]

4.3 Limitations of One-Tier and Two-Tier Architectures

Both one-tier and two-tier architectures suffer from limited network segmentation. When multiple system responsibilities share the same tier or network segment, enforcing strict access control becomes difficult.

These architectures also increase the attack surface, as internal services may become accessible through misconfigurations. Troubleshooting and maintenance become more challenging when failures are not isolated to a specific tier. As a result, simpler architectures are often insufficient for applications that handle sensitive data or require stronger security guarantees. [28,29.]

4.4 Security and Cost Challenges

Designing a secure cloud architecture requires balancing security, reliability, and cost. Industry reports and cloud best-practice frameworks consistently show that security incidents and unnecessary cloud expenses often arise not from a lack of security tools, but from architectural design choices such as insufficient network segmentation, lack of fault isolation, and overuse of always-on managed services [6,7,8,9,10,11,12,13]. These challenges are particularly relevant for small-scale and budget-constrained environments. The challenges that are widely observed in real-world cloud deployments are documented in industry security reports, and cloud cost management studies are discussed in the following sections.

4.4.1 Cloud Misconfiguration and Poor Network Segmentation

Based on the Verizon Data Breach Investigations Report (DBIR), misconfiguration and improper access control are among the most common causes of cloud-related security incidents [8]. Exposed services, overly permissive network rules, and insufficient network segmentation significantly increase the attack surface of cloud environments. In small or simplified deployments, multiple system components are often placed within the same network layer, which increases the risk of unauthorized access and data exposure. [3,6,8.]

4.4.2 Lack of Fault Isolation and Cascading Failures

Cloud reliability guidelines, such as those presented in Google's Site Reliability Engineering (SRE) literature, emphasize fault isolation as a fundamental architectural principle [30]. Systems that lack component isolation are more vulnerable to cascading failures, in which a fault in one component can affect the entire system. When application logic and data storage are tightly coupled or deployed within the same network segment, troubleshooting becomes more difficult, and system resilience is reduced. [3,30.]

4.4.3 Cloud Cost Inefficiency and Overprovisioning

Cloud cost management studies indicate that a significant portion of cloud spending is wasted due to overprovisioned resources and unused services. The Flexera State of the Cloud Report consistently reports that approximately thirty percent of cloud expenditure is wasted, often due to deploying services that exceed actual workload requirements. [11.] This issue is especially common when enterprise-grade configurations are applied to low-traffic or experimental systems.

4.4.4 Always-On Managed Services and Baseline Costs

The Cost Optimization pillar from AWS states that continuously running services such as load balancers, NAT Gateways, and multi–Availability Zone deployments add fixed costs to the system [12]. While these services are essential for high-availability and mission-critical systems, they are not always required for small-scale or non-critical environments. Applying such services by default can significantly increase operational costs without providing proportional benefits. Careful service selection based on actual needs is, therefore, critical for cost-efficient cloud design.

4.4.5 Publicly Exposed Cloud Resources as Attack Vectors

Threat intelligence reports from IBM X-Force and Palo Alto Networks Unit 42 identify publicly exposed cloud resources, such as databases and management interfaces, as common targets for attack. Direct internet exposure increases the likelihood of exploitation, particularly when combined with misconfiguration or weak access controls. Deploying sensitive resources in private networks and limiting access through controlled communication paths reduces this risk. [4,5.]

4.5 Requirements for the Proposed Solution

Based on the analysis of existing architectures and their limitations, the proposed solution must meet the following requirements:

- Clear division between web, application, and database components.
- Restriction of direct internet access to the web tier only.
- Controlled and minimal communication paths between tiers.
- Improved fault isolation through tier separation.
- Cost efficiency is suitable for small-scale and budget-constrained environments.

These requirements serve as the foundation for the design and implementation of the three-tier cloud network architecture.

5 Proposed Solution and Implementation

This chapter presents the design and implementation of the proposed three-tier cloud network architecture in Amazon Web Services. The focus is on explaining what was built, how it was implemented, and how security and cost-efficiency were achieved through architectural decisions.

5.1 Overall Architecture Design

The proposed solution is based on a three-tier cloud network architecture that consists of a web tier, an application tier, and a database tier. Each tier had a clearly defined responsibility and was isolated at the network level to improve security and fault isolation.

The web tier was the only component exposed to the internet and was responsible for receiving user requests. The application tier manages backend logic and links the web and database tiers. The database tier stored sensitive data and was accessible only from the application tier. As seen below, Figure 2 illustrates the design of the three-tier cloud network architecture.

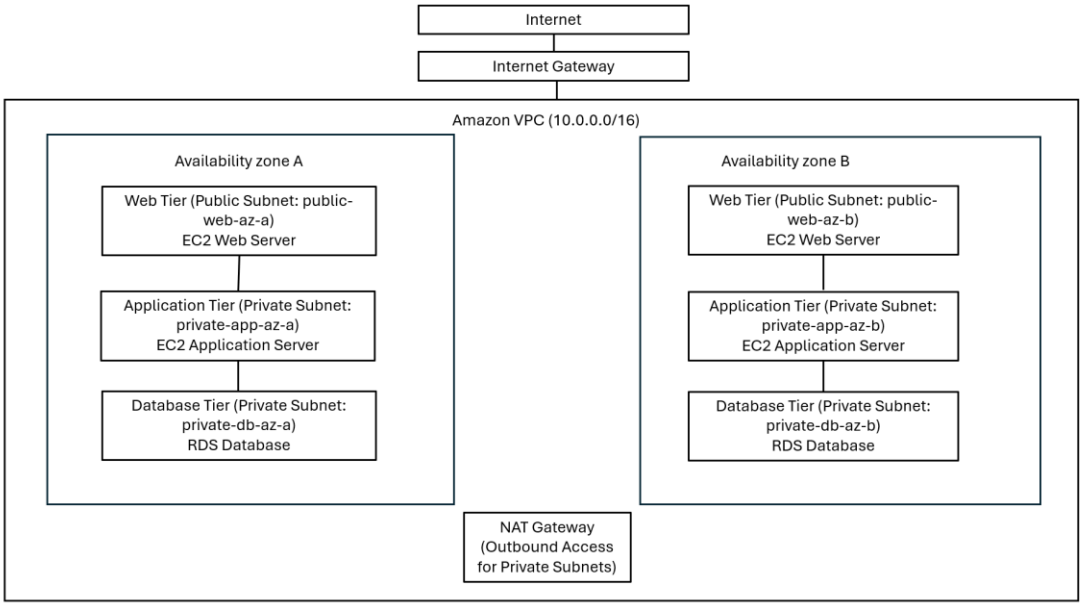


Figure 2. High-level three-tier cloud network architecture in AWS.

As illustrated in Figure 2, the system is set up within one Amazon VPC and uses multiple Availability Zones to increase availability and fault isolation. The web tier is deployed in public subnets, while the application and database tiers are in private subnets. An Internet Gateway provides internet access only for the web tier, and all communication between the tiers takes place using private IP addresses within the VPC.

This layered design limits unnecessary access paths and reduces the attack surface while keeping the architecture simple and suitable for small-scale environments.

5.2 VPC and Subnet Design

The entire system was deployed within a single Amazon Virtual Private Cloud named three-tier-vpc. The VPC was assigned the CIDR block 10.0.0.0/16, providing sufficient address space for all tiers and future expansion. Figure 3 illustrates the structure of the VPC and the division of the subnets.

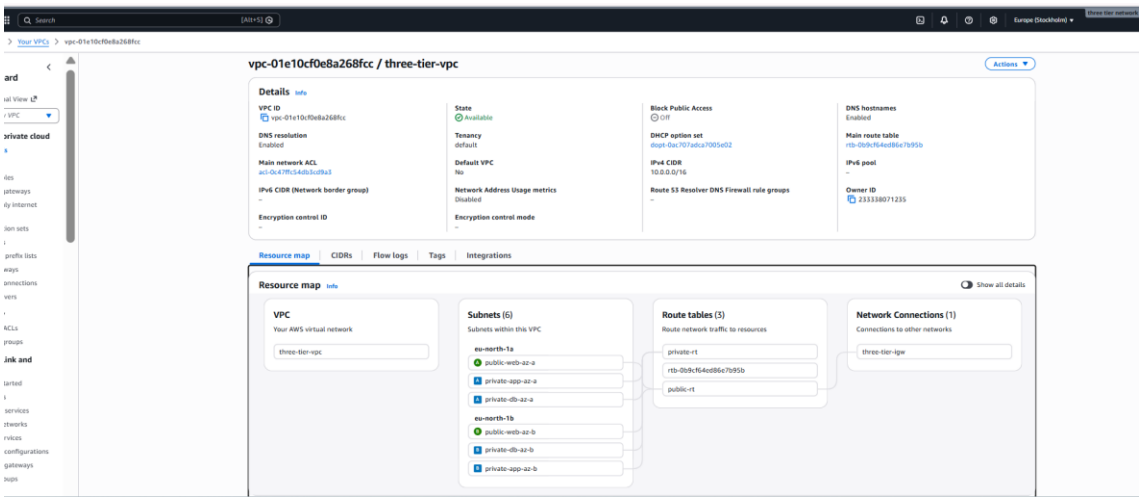


Figure 3. VPC structure showing public and private subnets across two availability zones.

Figure 3 shows that subnets were distributed across two availability zones to improve availability and fault isolation. Two public subnets were created for the web tier. In contrast, two private subnets were created for both the application tiers and the database tiers, respectively. This subnet design ensured that only the web tier had direct internet access, while internal tiers remained isolated within private networks.

The confirmation of VPC creation, internet gateway configuration, elastic Internet Protocol (IP) allocation, NAT gateway setup, route table configuration, and subnet associations is provided in Appendices 1 to 10.

5.3 Web Tier Implementation

An Amazon EC2 instance in a public subnet was used for the web tier and was given a public IPv4 address for internet access. A lightweight web server was used to receive HTTP requests and serve as the entry point of the system. Requests that require backend processing are forwarded internally to the

application tier using private IP communication within the VPC. Figure 4 shows the Amazon EC2 instance used to implement the web tier.

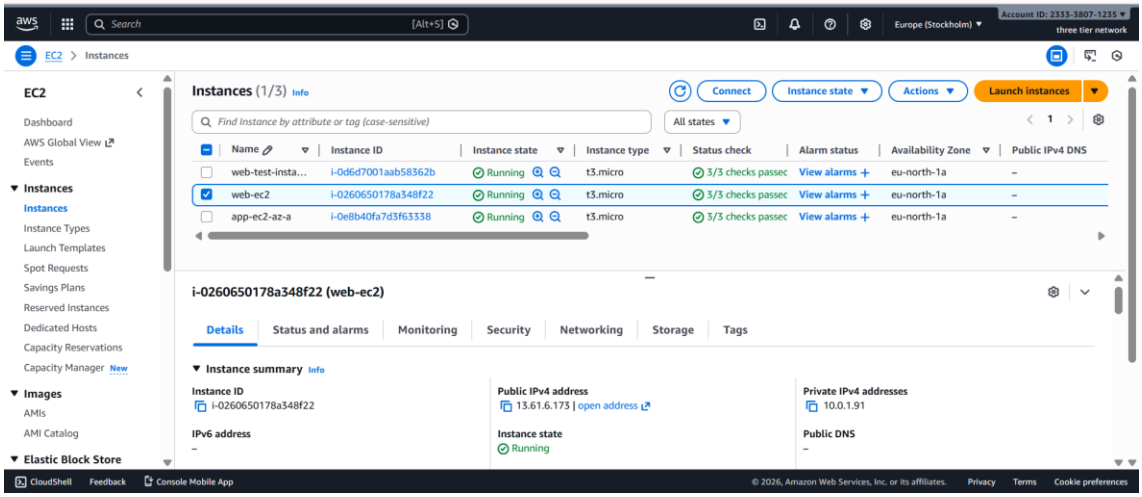


Figure 4. Deployment of web tier EC2 in a public subnet with a public IPv4 address.

Figure 4 illustrates the operational Amazon EC2 instance used for the web tier. The instance is running successfully, with all health checks passed, confirming that the web tier is correctly deployed and available for handling incoming requests.

The web tier does not store sensitive data and does not have direct access to the database tier to reduce the attack surface and limit the impact of potential security breaches.

5.4 Application Tier Implementation

The application tier is deployed on an Amazon EC2 instance in a private subnet without a public IP address. The application tier handles backend logic and manages communication between the web tier and the database. It acts as a controlled intermediary, ensuring that only valid and authorized requests reach the database tier. Figure 5 shows the deployment of the application tier EC2 instance in a private subnet.

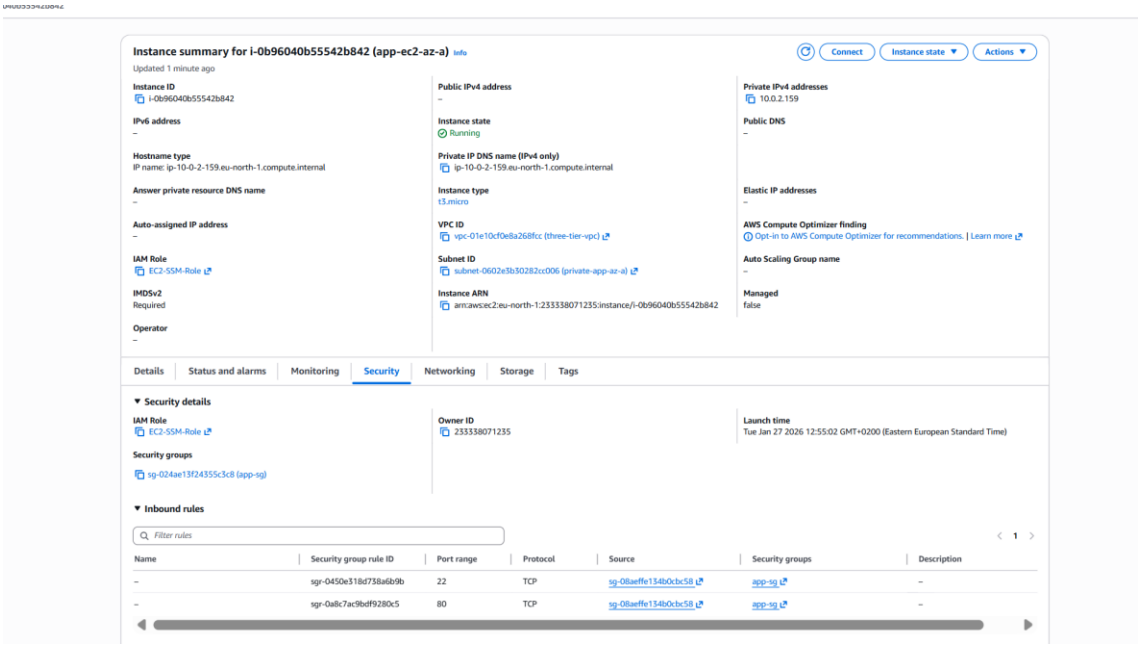


Figure 5. Deployment of the application tier EC2 in a private subnet.

As seen in Figure 5, the application tier EC2 instance is running inside a private subnet and has only a private IP address assigned. No public IPv4 address is present, which verifies that the instance could not be reached online.

Administrative access to the application tier was secured using AWS Systems Manager, allowing management and troubleshooting without exposing SSH access or opening additional network ports.

5.5 Database Tier Implementation

The database tier was implemented using Amazon RDS with PostgreSQL. The database was deployed to private subnets and configured to be inaccessible to the public. Access to the database was restricted so that only the application tier could connect to it. This ensures that sensitive data is protected from both internet access and direct access from the web tier. Figure 6 shows the Amazon RDS instance used to implement the database tier of the architecture.

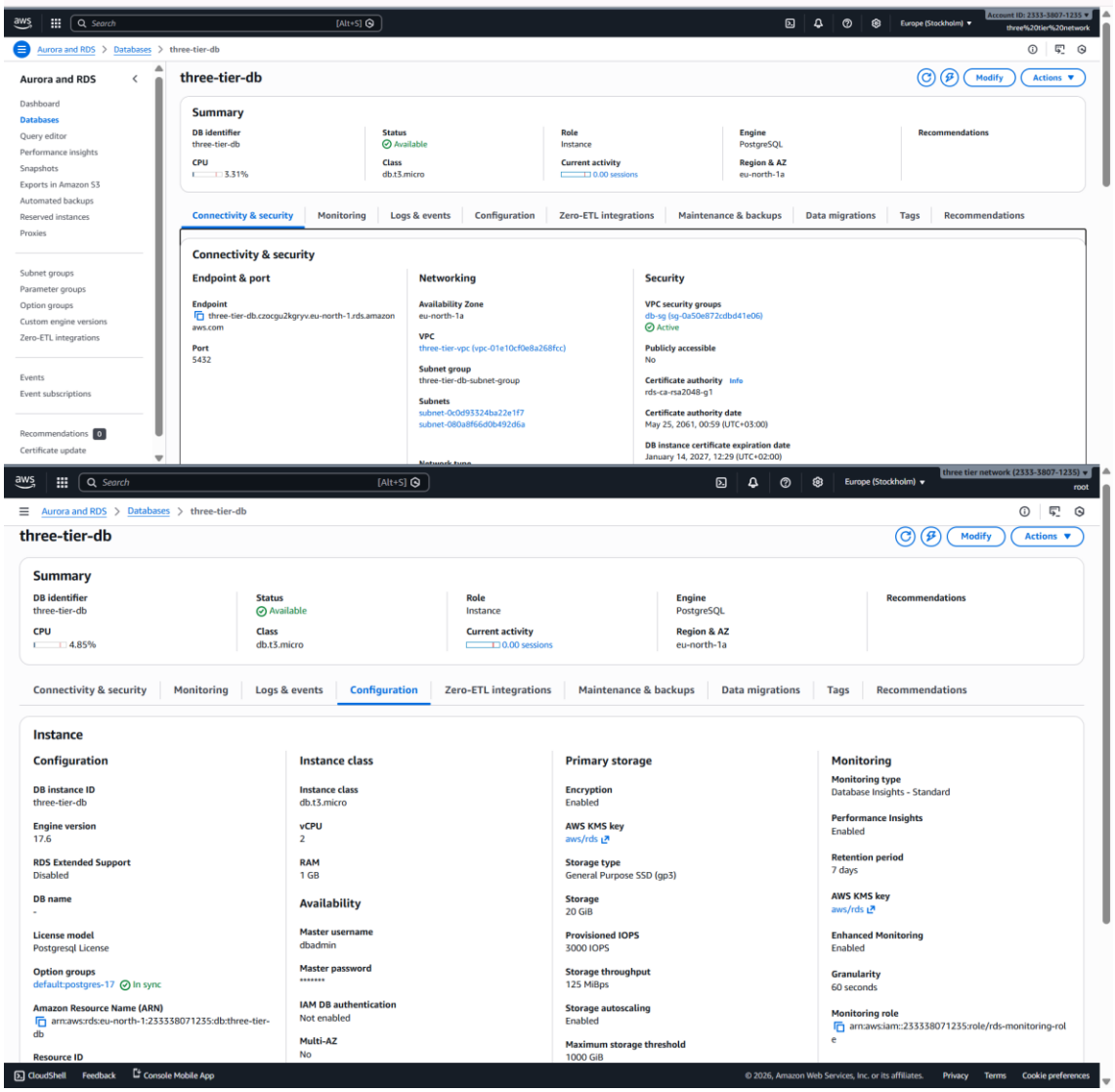


Figure 6. Deployment of Amazon RDS PostgreSQL in private subnets.

Figure 6 verifies that the PostgreSQL database is running in a private subnet and is not publicly accessible. The database endpoint is reachable only within the VPC, ensuring that it cannot be accessed directly from the internet or the web tier. Connectivity is restricted so that only the application tier is allowed to communicate with the database, to keep sensitive data more secure.

5.6 Security Groups and Traffic Flow Control

Traffic between the system tiers was controlled using AWS Security Groups. Security Groups act as virtual firewalls that define which types of traffic are allowed to reach each resource. In this architecture, Security Groups were used as the primary mechanism to enforce controlled communication between tiers and to prevent unnecessary access.

Each tier was assigned a dedicated Security Group to support the principle of least privilege. Web-sg was the security group of the web tier, which allows inbound traffic from the internet only on the required web ports. Similarly, app-sg was the security group of the application tier, which allows inbound traffic only from the web tier. The database tier uses a Security Group named db-sg, which allows inbound database connections only from the application tier.

This configuration creates a strict and predictable traffic flow through the system. User requests enter the system through the web tier, are forwarded to the application tier for processing, and finally reach the database tier when data access is required. Direct communication between the web tier and the database tier is not permitted.

By blocking all other traffic paths, the Security Group design reduces the system’s attack surface and improves fault isolation. Even if one tier is compromised, the Security Groups limit lateral movement within the system. This demonstrates how effective security can be achieved through correct network segmentation and access control rather than additional security services.

Figure 7 shows the security group used for the web tier EC2 instance. It defines the inbound traffic allowed from the Internet.

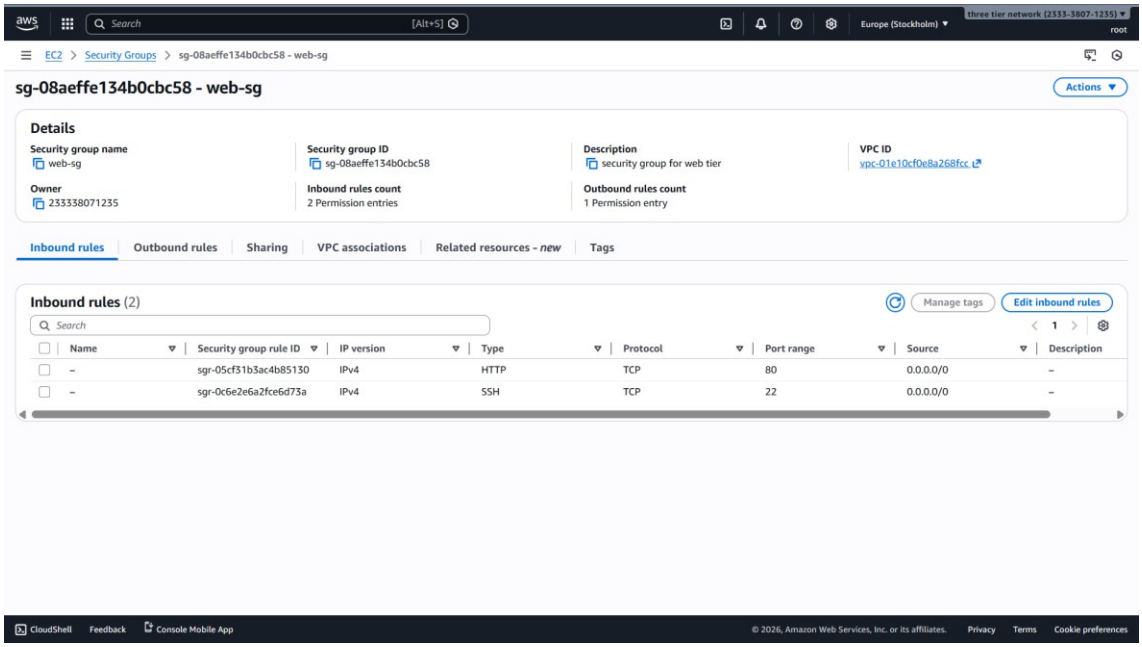


Figure 7. Security group applied to the web tier.

The inbound rules from Figure 7 allow HTTP traffic from the Internet to reach the web tier. This provides access to the application while keeping other services protected. Figure 8 shows the security group used for the application tier EC2 instance. It controls which sources are allowed to access backend services.

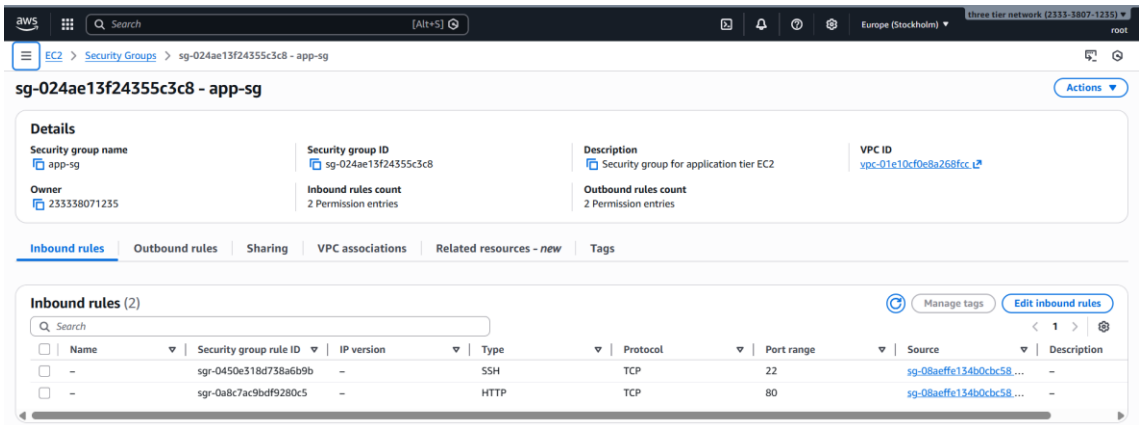


Figure 8. Security group applied to the application tier.

The application tier accepts inbound traffic only from the web tier security group. This ensures that backend services are not accessible directly from the internet or other sources.

Figure 9 shows the security group used for the database tier. It defines the access rules for database connections.

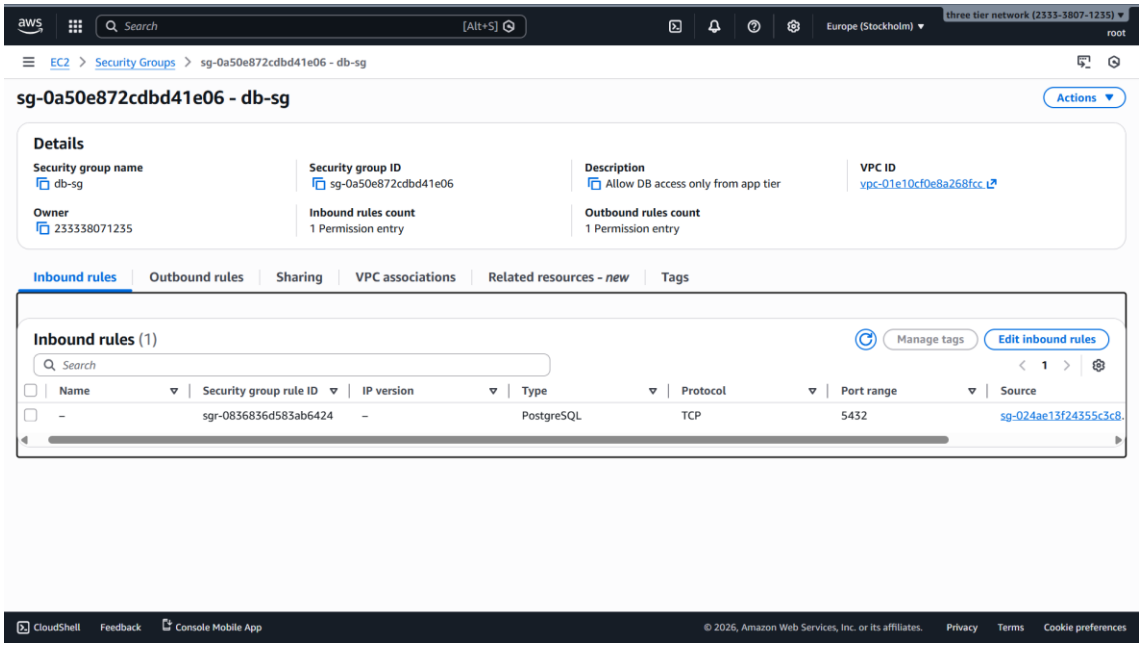


Figure 9. Security group applied to the database tier.

The database security group allows PostgreSQL traffic only from the application tier. Direct access from the web tier or the internet is not permitted to protect sensitive data. The outbound rule configuration of the database security group is shown in Appendix 11.

5.7 Access Management Using AWS Systems Manager

Administrative access to EC2 instances in private subnets was managed using AWS Systems Manager Session Manager. This approach eliminates the need to open SSH ports or manage SSH keys for private instances.

Using Session Manager, administrative access is logged and secured via AWS Identity and Access Management (IAM), improving overall security and auditability. Figure 10 shows the IAM role attached to the EC2 instances to enable access through AWS Systems Manager.

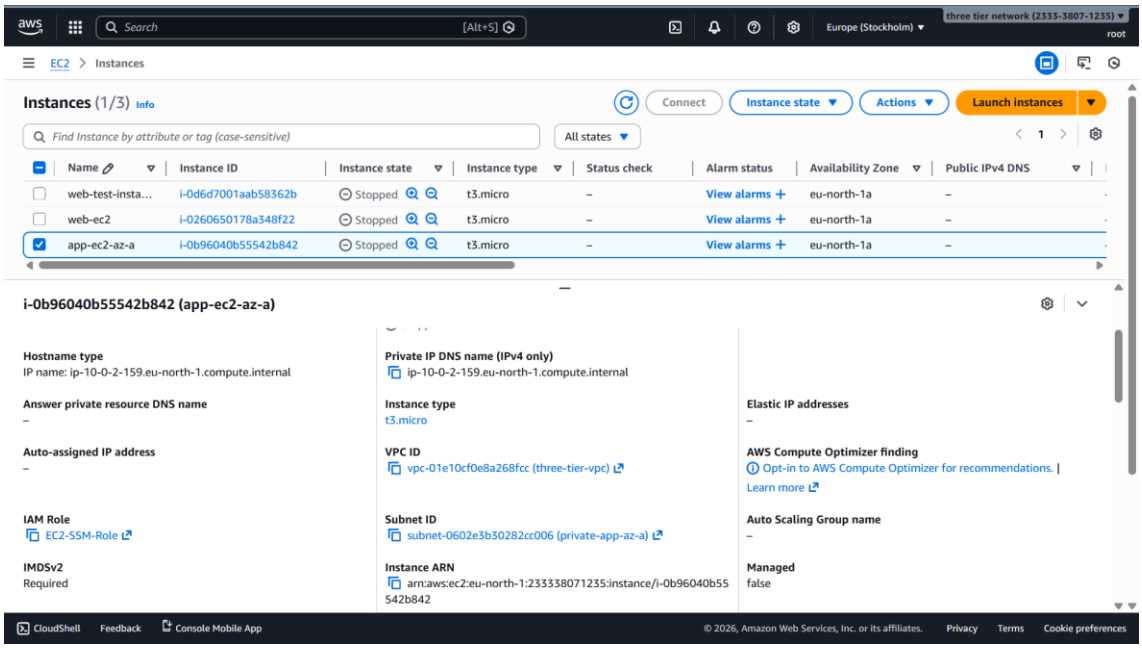


Figure 10. EC2 instance IAM role configured for AWS Systems Manager session manager access.

Therefore, Figure 10 confirms that an AWS Systems Manager-compatible IAM role is attached to the EC2 instances. This role provides administrative access via Session Manager without requiring SSH access or opening inbound ports. Access to private instances is managed securely using IAM.

5.8 Cost Optimization Decisions

Cost efficiency was a key design goal of the proposed solution. Several design decisions were made to minimize operational costs while maintaining the required level of security and functionality. Cost efficiency was ensured by carefully choosing services and configurations that provide security without unnecessary cloud costs.

5.8.1 Selection of EC2 Instance Types

Small EC2 instance (t3.micro) types were selected for both the web and application tiers, as the workload consisted only of basic request handling and connectivity testing. This avoided unnecessary compute capacity while still supporting all functional requirements of the architecture.

5.8.2 Database Deployment Strategy

The database tier was deployed using Amazon RDS in a single Availability Zone. While multi-AZ configurations improve availability, they significantly increase cost. For the scope of this study, which focused on network security and segmentation rather than high availability, a single-AZ deployment was considered sufficient and more cost-effective. Additionally, a small database instance size was selected because the workload was limited to basic connectivity testing and low-volume queries, which further reduced operational costs by avoiding unnecessary overprovisioning while still meeting all functional requirements.

5.8.3 Exclusion of Additional Managed Services

Services such as load balancers, auto scaling groups, and managed firewall solutions were intentionally excluded from the design. While these services offer benefits in production environments, they introduce additional recurring costs and complexity. The security objectives of the architecture were achieved through network segmentation and access control, making these services unnecessary for the scope of this work.

5.8.4 Temporary Use of the NAT Gateway

A NAT Gateway was deployed temporarily to allow outbound internet access from private subnets during system setup and configuration. This access was required for tasks such as installing software packages and enabling AWS-

managed services. After these tasks were completed, the NAT Gateway was deleted to avoid continuous hourly charges, demonstrating conscious cost management.

5.8.5 Resource Management After Testing

After implementation and testing activities were completed, EC2 instances were stopped to prevent unnecessary compute costs. This approach reflects practical cost awareness and demonstrates how cloud resources can be actively managed to avoid paying for unused capacity.

6 Testing, Results, and Discussion

This chapter presents the testing methodology, summarizes the results of the connectivity and security tests, evaluates cost-related outcomes, and discusses the findings of the implemented three-tier architecture.

6.1 Testing Methodology

The testing methodology focused on verifying that the implemented three-tier cloud architecture behaves as designed in terms of connectivity, security, and access control. Rather than relying on simulations or theoretical validation, testing was conducted using real network connections in the deployed AWS environment.

The tests were designed to validate two key aspects of the architecture. First, required communication paths between system components were tested to ensure that legitimate traffic is allowed. Second, unauthorized access attempts were tested to confirm that direct access to restricted tiers is correctly blocked.

Testing was performed using standard command-line tools and cloud-native access mechanisms provided by AWS. These included browser-based access for the web tier, internal connectivity tests between EC2 instances, database connection attempts using the RDS endpoint, and administrative access via AWS Systems Manager Session Manager.

Both successful and failed test results were documented. Successful connections demonstrate that required communication paths function correctly, while failed connection attempts provide evidence of effective network segmentation and access control. This approach ensures that the security design of the architecture is validated through practical observation rather than configuration assumptions.

6.2 Connectivity Test Results

Connectivity testing was performed to verify that communication between system components follows the designed three-tier architecture. The tests focused on confirming that only the required communication paths are allowed, while unauthorized access attempts are blocked.

6.2.1 Web Tier Accessibility Test

The first test verified that the web tier is reachable from the internet. The web server was accessed using its public IPv4 address through a standard web browser. This confirms that the web tier is correctly exposed to external users and functions as the system's entry point. Figure 11 shows the details of the web tier, including the assigned public IPv4 address, and access of the browser using the public IP address.

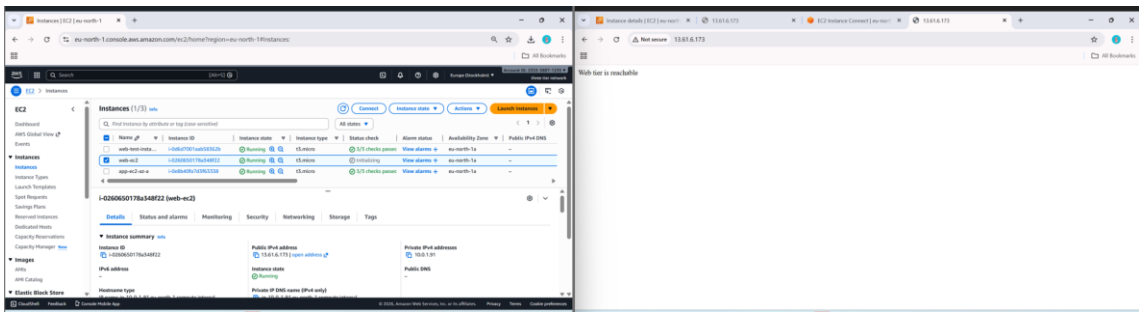


Figure 11. Web tier EC2 instance details and browser access using the public IP address.

Figure 11 shows the configuration of the web tier EC2 instance, including its assigned public IPv4 address and running state. The browser view on the right confirms that the web server is reachable from the internet using this public IP address. This verifies that the web tier is correctly exposed to external users while acting as the only internet-facing component of the architecture.

6.2.2 Internal Connectivity Between Web and Application Tiers

In order to validate internal communication between the web tier and the application tier, a connectivity test was performed from the web tier EC2 instance to the application tier using private IP communication. This test verifies that internal traffic is allowed according to the architecture design while keeping the application tier inaccessible from the public internet.

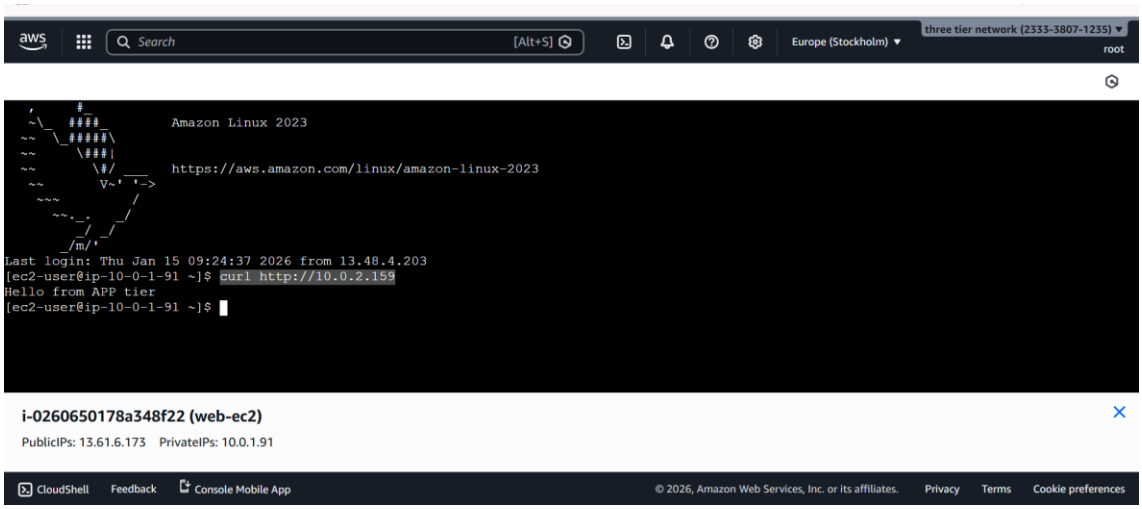


Figure 12. Successful internal connectivity test from the web tier to the application tier using private IP.

Figure 12 shows a successful HTTP request sent from the web tier instance to the private IP address of the application tier using the curl command. The response confirms that the application tier is reachable only from within the VPC and that internal communication between the tiers is functioning as intended. This result demonstrates that the application tier is correctly isolated from external access while remaining accessible to the web tier.

6.2.3 Application Tier to Database Tier Connectivity

To verify secure communication between the application tier and the database tier, a connectivity test was performed from the application tier EC2 instance to the database. The test was conducted using the database endpoint and

standard PostgreSQL client tools. This validation ensures that the database is reachable only from the application tier, as defined by the three-tier architecture design.

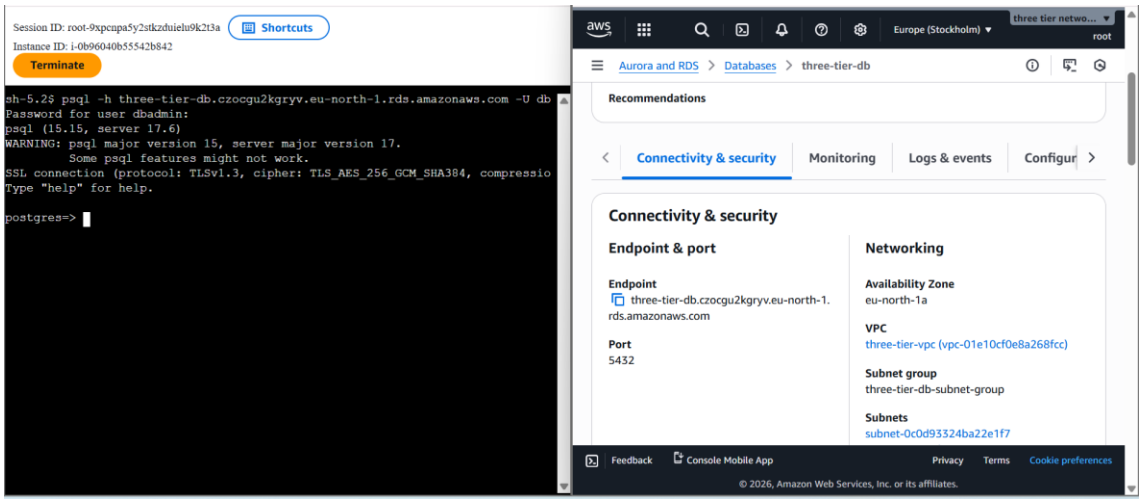


Figure 13. Successful connection from the application tier to the database.

Figure 13 shows a successful PostgreSQL connection established from the application tier to the database using the RDS endpoint and port 5432. The PSQL prompt confirms that authentication was successful and that the database accepted the connection over a secure TLS-encrypted channel. This result demonstrates that the database tier is correctly isolated from both the internet and the web tier, while remaining accessible to the application tier as intended.

6.3 Security Validation Results

Security validation was performed to confirm that the implemented three-tier architecture correctly enforces network isolation and access control rules. The tests focused on verifying that only intended communication paths are allowed and that all unauthorized access attempts are blocked according to the security group configuration.

The validation was carried out by attempting connections from different sources, including the public internet and internal tiers, and observing whether access was permitted or denied.

6.3.1 Internet Access to the Application Tier (Blocked)

To verify that the application tier is not directly accessible from the internet, an SSH connection attempt was made from a local machine to the private IP address of the application tier EC2 instance.

This test was expected to fail because the application tier is deployed in a private subnet and does not have a public IP address. In addition, the security group of the application tier does not allow inbound traffic from external sources.

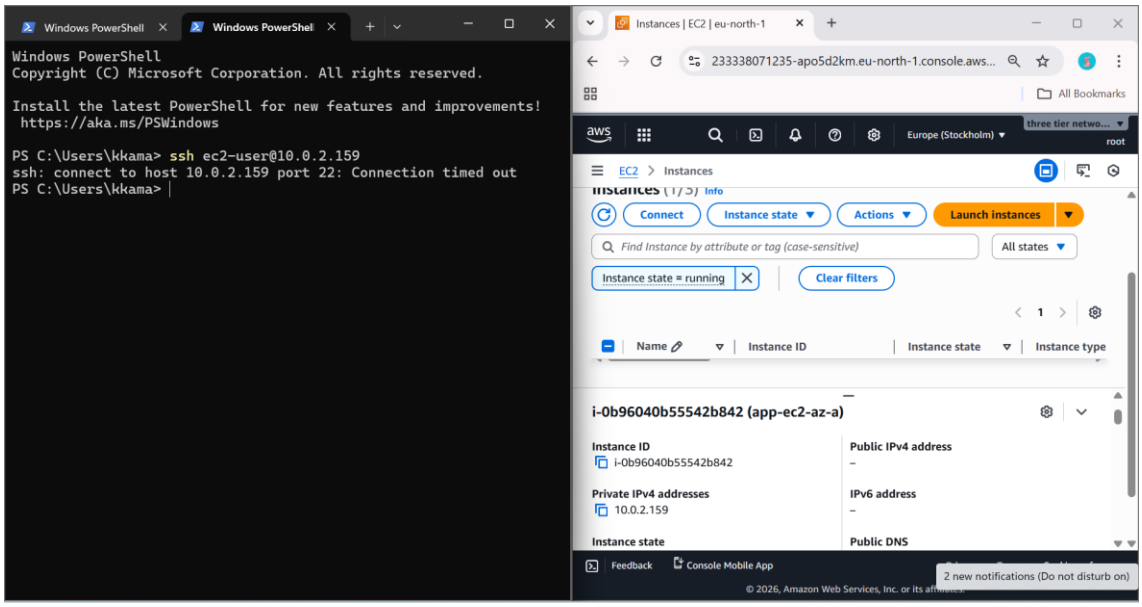


Figure 14. Internet access to the application tier is blocked.

As shown in Figure 14, the connection attempt timed out, confirming that the application tier cannot be accessed directly from the internet. This result validates that the private subnet placement and security group rules are functioning as intended.

6.3.2 Web Tier Access to the Database Tier (Blocked)

To further validate tier isolation, a database connection attempt was made from the web tier EC2 instance directly to the database endpoint using the PostgreSQL client. This connection was expected to fail because the database tier is configured to accept connections only from the application tier security group and not from the web tier.

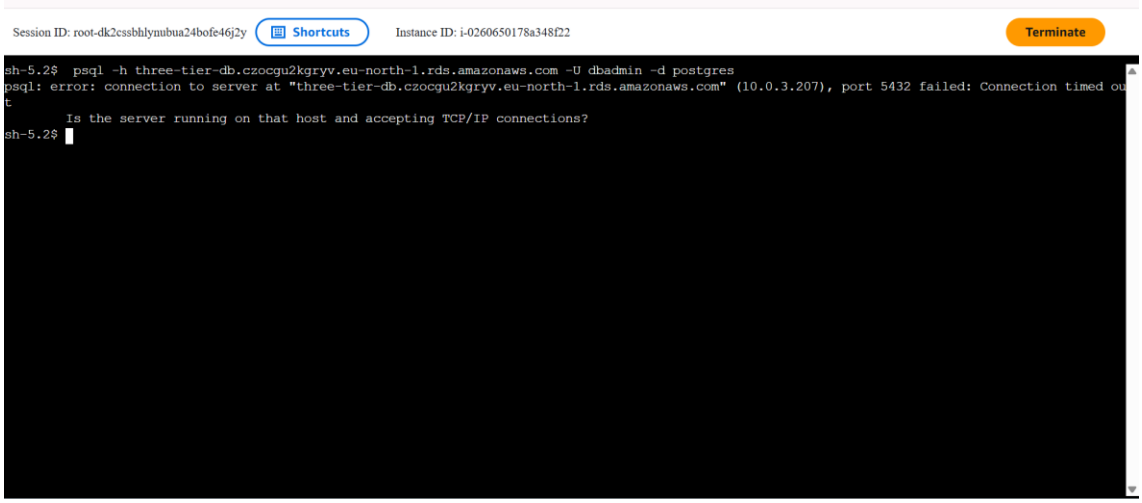


Figure 15. Direct database access from the web tier is blocked.

The failed connection confirms that the web tier has no direct access to the database tier. This enforces strict separation between tiers and reduces the risk of sensitive data exposure in case the web tier is compromised.

6.3.3 Internet Access to the Database Tier (Blocked)

As a final security validation, a network connectivity test was performed to access the database endpoint directly from the public internet. This test verifies that the database is fully isolated from external access and is not reachable over the internet.

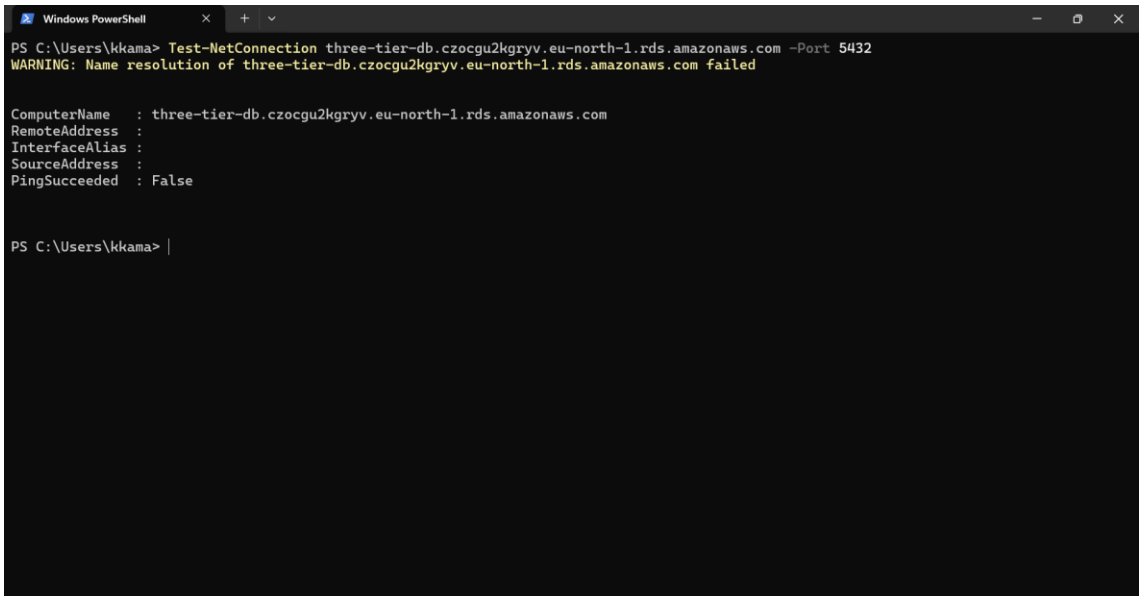


Figure 16. Internet access to the database tier is blocked.

The failed connectivity test in Figure 16 confirms that the database tier is not exposed to the internet. The database remains accessible only through internal VPC communication, ensuring strong protection for sensitive data.

6.4 Cost Evaluation Results

The cost evaluation examined whether the implemented architecture achieved its objective of maintaining security and structural separation without creating unnecessary operational expenses. As discussed in Sections 4.4.3 and 4.4.4, high cloud costs are commonly caused by over-provisioned resources and the continuous use of always-on managed services rather than by the three-tier architectural model itself. This section evaluates whether the cost drivers were avoided in practice.

The system was deployed using small, general-purpose instance types. Both the web and application tiers used t3.micro EC2 instances (on-demand rate: 0.0108 USD per hour in the Stockholm region) [31]. The database tier used a db.t3.micro Amazon RDS instance with 20 GiB of gp3 storage in a Single-Availability Zone configuration, with an approximate monthly cost of 16.27 USD

[32]. These instance sizes were sufficient for the workload, which primarily consisted of request handling and connectivity validation. No unnecessary compute capacity or scaling mechanisms were introduced.

The NAT Gateway was created only temporarily for outbound access during setup and testing. Continuous use of a NAT Gateway would add a fixed baseline cost of 0.046 USD per hour [33]. By deleting it after validation, recurring charges were avoided. Similarly, EC2 instances were stopped when not actively used.

According to the AWS Cost and Usage Report, the total infrastructure cost for January was 31.85 USD. This amount reflects temporary resource usage rather than continuous 24/7 operation. The detailed usage report is provided in Appendix 12.

To provide a consistent baseline comparison, the minimum monthly cost of one-tier, two-tier, and three-tier deployments was calculated assuming continuous operation for 720 hours (24/7 for one month) and continuous NAT Gateway usage.

Table 3. Minimum monthly cost comparison.

Architecture	Components	Approximate monthly cost in USD
One-tier	1 EC2 + NAT	40.90
Two-tier	1 EC2 + 1 RDS + NAT	57.17
Three-tier	2 EC2 + 1 RDS + NAT	64.94

The calculations shown in Table 3 are based on 0.0108 USD/hour per EC2 instance, 16.27 USD/month for db.t3.micro RDS (Single-AZ), and 0.046 USD/hour for NAT Gateway [31,32,33].

The comparison shows that the primary cost increase across architectures is caused by continuously running the NAT Gateway and database service rather

than by the architectural separation itself. The difference between the two-tier and three-tier models is mainly the additional small EC2 instance used to logically separate the web and application layers. While this slightly increases the monthly cost, it improves isolation, reduces the attack surface, and strengthens fault containment.

During configuration testing, alternative database options were also evaluated. When a larger RDS instance type was selected in a Single-AZ configuration, the estimated monthly cost increased to approximately 131.61 USD. When the same larger instance type was combined with a multi-AZ deployment, the estimated monthly database cost further increased to approximately 193.35 USD. The detailed cost estimates of RDS are provided in Appendices 13 to 15.

These results demonstrate that three-tier segmentation does not necessarily require high expenditure. When instance sizes are carefully selected and unnecessary managed services are avoided, secure architectural separation can be implemented with predictable and controlled monthly costs.

6.5 Discussion of Findings

The testing results demonstrate that the three-tier architecture successfully improves security and fault isolation through network segmentation. Compared to simpler architectures, the implemented solution limits internet exposure to the web tier and strictly controls internal communication paths.

The use of private subnets and security groups proved effective in protecting sensitive components such as the application and database tiers. At the same time, the architecture remained relatively simple to implement and manage, making it suitable for small organizations or educational environments.

The result shows that thoughtful architectural design and correct configuration can significantly improve system security without increasing complexity or cost.

The proposed architecture does not replace enterprise-grade cloud designs but provides a cost-efficient alternative for scenarios where high availability and automatic scaling are not strict requirements. When system criticality or traffic volume increases, additional services such as load balancers, multi-AZ databases, and permanent NAT Gateways can be introduced without changing the core three-tier structure demonstrated in this study.

6.6 Limitations of the Implemented Solution

Although the implemented architecture meets its objectives, it has certain limitations. The system does not include advanced features such as logging, continuous monitoring, intrusion detection, vulnerability scanning, penetration testing, load balancing, auto scaling, or multi-region deployment, which would be required for high-traffic or enterprise-level systems.

The database is deployed in a single Availability Zone, which limits availability in case of zone-level failures. Additionally, performance testing under heavy load was not conducted, as the focus of this study is on network design and security rather than performance optimization. These limitations are acceptable within the scope of this study and reflect the design choices aimed at balancing simplicity and cost efficiency.

7 Conclusions and Future Work

This study focused on the design and implementation of a secure and cost-effective three-tier cloud network architecture using Amazon Web Services. The main goal was to demonstrate that strong security and clear fault isolation can be achieved through correct network segmentation and access control, without relying on complex or expensive cloud services.

The proposed architecture was implemented within a single VPC using public and private subnets distributed across multiple availability zones. The web tier was placed in public subnets and exposed to the internet, while the application and database tiers were deployed in private subnets and protected from direct external access. Communication between tiers was strictly controlled using security groups, following the principle of least privilege.

The implemented solution was validated through practical testing. Connectivity tests confirmed that required communication paths between tiers functioned correctly, while security tests showed that unauthorized access attempts from the internet and from incorrect tiers were successfully blocked. These results demonstrate that the architecture behaves as intended and provides a clear improvement in security compared to simpler one-tier or two-tier designs.

Cost efficiency was maintained by selecting lightweight service configurations and avoiding unnecessary components such as load balancers, auto scaling, and multi-region deployments. The use of managed services was limited to essential components, and resources were stopped or removed when no longer required. This shows that secure cloud architectures can be built in a cost-effective manner, making the solution suitable for small organizations, educational environments, and budget-constrained use cases.

Although the implemented architecture does not include advanced scalability or high-availability features, these limitations were intentional and aligned with the scope of the study. The design provides a practical foundation that can be extended in the future with additional services if required. Overall, the study demonstrates that well-structured cloud network architectures can effectively balance security, simplicity, and cost when design decisions are made carefully and based on actual requirements.

References

1 Mell P, Grance T. The NIST Definition of Cloud Computing. Gaithersburg (MD): National Institute of Standards and Technology; 2011. Available from: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf. Accessed January 17, 2026.

2 IBM. What is Cloud Computing? Armonk, NY: IBM; 2024. Available from: https://www.ibm.com/think/topics/cloud-computing. Accessed January 17, 2026.

3 Amazon Web Services. AWS Fault Isolation Boundaries. Seattle (WA): Amazon Web Services; 2023. Available from: https://docs.aws.amazon.com/pdfs/whitepapers/latest/aws-fault-isolation-boundaries/aws-fault-isolation-boundaries.pdf. Accessed January 17, 2026.

4 IBM Security. 2025 Threat Intelligence Index. Armonk (NY): IBM Corporation; 2025. Available from: https://www.ibm.com/thought-leadership/institute-business-value/report/2025-threat-intelligence-index. Accessed February 8, 2026.

5 Palo Alto Networks Unit 42. Cloud Threat Report: Expanding Attack Surface. Santa Clara (CA): Palo Alto Networks; 2024. Available from: https://unit42.paloaltonetworks.com/cloud-threat-report-expanding-attack-surface/. Accessed February 8, 2026.

6 Amazon Web Services. Amazon Virtual Private Cloud User Guide. Seattle (WA): Amazon Web Services; 2024. Available from: https://docs.aws.amazon.com/pdfs/vpc/latest/userguide/vpc-ug.pdf. Accessed January 18, 2026.

7 Fidelis Security. Cloud Misconfigurations Causing Data Breaches. Bethesda (MD): Fidelis Security; 2023. Available from: https://fidelissecurity.com/threatgeek/threat-detection-response/cloud-misconfigurations-causing-data-breaches/. Accessed February 8, 2026.

8 Verizon. Data Breach Investigations Report 2025. New York (NY): Verizon; 2025. Available from: https://www.verizon.com/business/resources/T794/reports/2025-dbir-data-breach-investigations-report.pdf. Accessed February 8, 2026.

9 Cisco. 2022 Global Hybrid Cloud Trends Report. Cisco Systems; 2022. Available from: https://www.cisco.com/c/en_au/solutions/hybrid-cloud/2022-trends-report-cte.html. Accessed February 24, 2026.

10 OpenText. Public Cloud: Unexpected Costs & Issues Affecting Customers and Employees – a Global Survey of IT Professionals and Executives.

OpenText; 2022. Available from: https://www.opentext.com/uk/media/report/public-cloud-unexpected-costs-and-issues-affecting-customers-and-employees-report-en.pdf. Accessed February 24, 2026.

11 Flexera. State of the Cloud Report 2025. Itasca (IL): Flexera; 2025. Available from: https://resources.flexera.com/web/pdf/Flexera-State-of-the-Cloud-Report-2025.pdf. Accessed February 8, 2026.

12 Amazon Web Services. AWS Well-Architected Framework: Cost Optimization Pillar. Seattle (WA): Amazon Web Services; 2024. Available from: https://docs.aws.amazon.com/pdfs/wellarchitected/latest/cost-optimization-pillar/wellarchitected-cost-optimization-pillar.pdf. Accessed February 8, 2026.

13 IBM Security. 2025 Threat Intelligence Index. Armonk (NY): IBM Corporation; 2025. Available from: https://www.ibm.com/thought-leadership/institute-business-value/report/2025-threat-intelligence-index. Accessed February 8, 2026.

14 Amazon Web Services. What is Cloud Computing? Seattle (WA): Amazon Web Services; 2024. Available from: https://aws.amazon.com/what-is-cloud-computing/. Accessed January 18, 2026.

15 Amazon Web Services. AWS Pricing Overview. Seattle (WA): Amazon Web Services; 2024. Available from: https://aws.amazon.com/pricing/. Accessed January 18, 2026.

16 Amazon Web Services. Amazon Web Services Overview. Seattle (WA): Amazon Web Services; 2023. Available from: https://docs.aws.amazon.com/pdfs/whitepapers/latest/aws-overview/aws-overview.pdf. Accessed January 18, 2026.

17 Flexera. State of the Cloud Report 2025. Itasca (IL): Flexera; 2025. Available from: https://resources.flexera.com/web/pdf/Flexera-State-of-the-Cloud-Report-2025.pdf. Accessed February 8, 2026.

18 Gartner. Gartner Says Worldwide IaaS Public Cloud Services Market Grew 22.5 percent in 2024. Stamford (CT): Gartner; 2025. Available from: https://www.gartner.com/en/newsroom/press-releases/2025-08-06-gartner-says-worldwide-iaas-public-cloud-services-market-grew-22-point-5-percent-in-2024. Accessed February 18, 2026.

19 Amazon Web Services. Subnet CIDR Blocks. Seattle (WA): Amazon Web Services; 2024. Available from: https://docs.aws.amazon.com/vpc/latest/userguide/subnet-sizing.html. Accessed January 18, 2026.

20 Amazon Web Services. Subnets for Your VPC. Seattle (WA): Amazon Web Services; 2024. Available from:

https://docs.aws.amazon.com/vpc/latest/userguide/configure-subnets.html. Accessed January 18, 2026.

21 Amazon Web Services. Building a Three-Tier Architecture on a Budget. Seattle (WA): Amazon Web Services; 2023. Available from: https://aws.amazon.com/blogs/architecture/building-a-three-tier-architecture-on-a-budget/. Accessed January 18, 2026.

22 Amazon Web Services. Security Groups for Your VPC. Seattle (WA): Amazon Web Services; 2024. Available from: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-groups.html. Accessed January 18, 2026.

23 Amazon Web Services. AWS Well-Architected Framework – Security Pillar. Seattle (WA): Amazon Web Services; 2024. Available from: https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/. Accessed January 18, 2026.

24 Amazon Web Services. High Availability and Scalability on AWS. Seattle (WA): Amazon Web Services; 2024. Available from: https://docs.aws.amazon.com/whitepapers/latest/real-time-communication-on-aws/high-availability-and-scalability-on-aws.html. Accessed January 18, 2026.

25 Amazon Web Services. Fault Tolerance and Fault Isolation. Seattle (WA): Amazon Web Services; 2024. Available from: https://docs.aws.amazon.com/whitepapers/latest/availability-and-beyond-improving-resilience/fault-tolerance-and-fault-isolation.html. Accessed January 18, 2026.

26 Palo Alto Networks. What is the Principle of Least Privilege? Santa Clara (CA): Palo Alto Networks; 2024. Available from: https://www.paloaltonetworks.com/cyberpedia/what-is-the-principle-of-least-privilege. Accessed January 20, 2026.

27 Amazon Web Services. AWS Well-Architected Framework. Seattle (WA): Amazon Web Services; 2024. Available from: https://docs.aws.amazon.com/pdfs/wellarchitected/latest/framework/wellarchitected-framework.pdf#welcome. Accessed January 30, 2026.

28 Amazon Web Services. Serverless Multi-Tier Architectures Using Amazon API Gateway and AWS Lambda. Seattle (WA): Amazon Web Services; 2023. Available from: https://docs.aws.amazon.com/pdfs/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/serverless-multi-tier-architectures-api-gateway-lambda.pdf. Accessed January 31, 2026.

29 IBM. Three-Tier Architecture. Armonk (NY): IBM; 2024. Available from: https://www.ibm.com/think/topics/three-tier-architecture. Accessed January 21, 2026.

30 Google. Addressing Cascading Failures. In: Site Reliability Engineering: How Google Runs Production Systems. Mountain View (CA): Google; 2016. Available from: https://sre.google/sre-book/addressing-cascading-failures/. Accessed February 5, 2026.

31 Amazon Web Services. Amazon EC2 on-Demand Pricing. Seattle (WA): Amazon Web Services; 2024. Available from: https://aws.amazon.com/ec2/pricing/on-demand/. Accessed February 5, 2026.

32 Amazon Web Services. Amazon RDS Pricing. Seattle (WA): Amazon Web Services; 2024. Available from: https://aws.amazon.com/rds/pricing/. Accessed February 8, 2026.

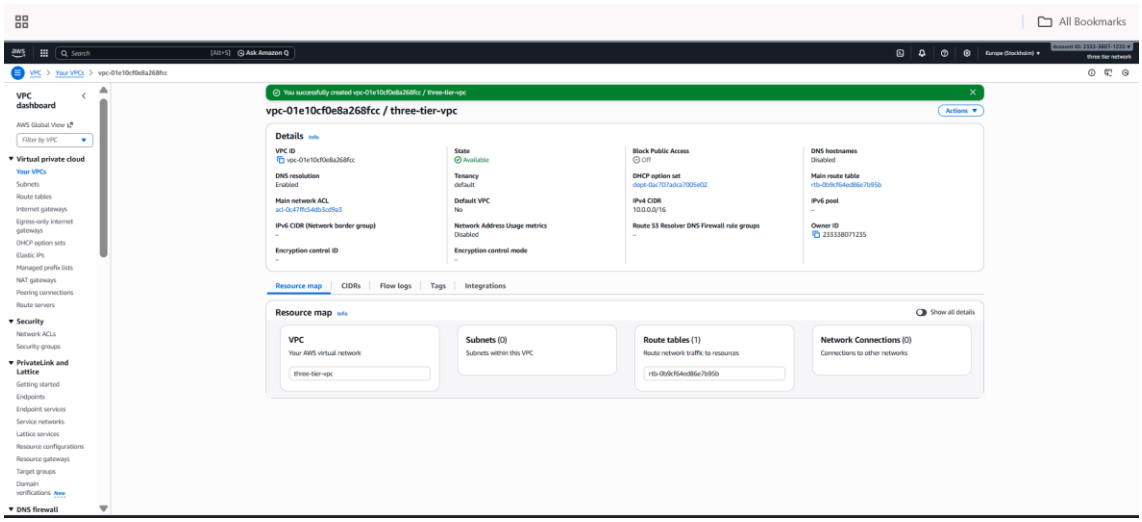
33 Amazon Web Services. Amazon VPC Pricing. Seattle (WA): Amazon Web Services; 2024. Available from: https://aws.amazon.com/vpc/pricing/. Accessed February 8, 2026.

Appendices

The appendices provide supporting screenshots and configuration evidence for implementation and cost evaluation discussed in the main body of the thesis.

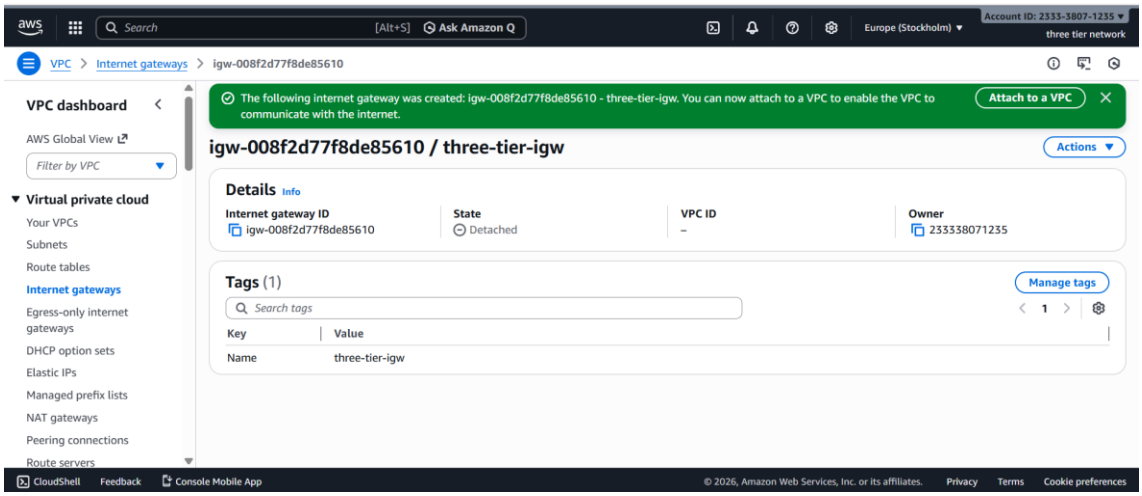
Appendix 1: VPC Created

The following picture confirms the creation of a three-tier VPC used in the study.



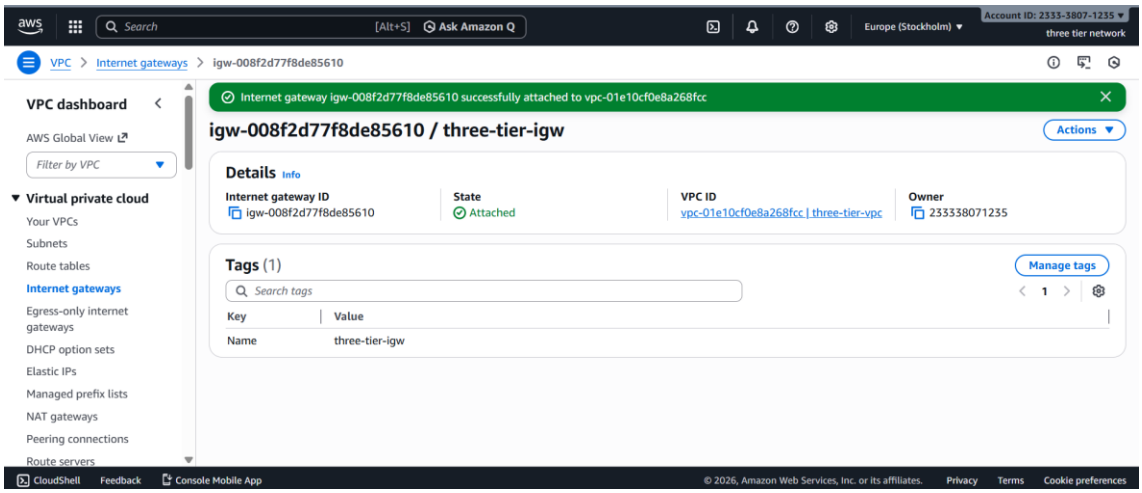
Appendix 2: Creation of Internet Gateway

The screenshot shows the successful creation of an internet gateway.



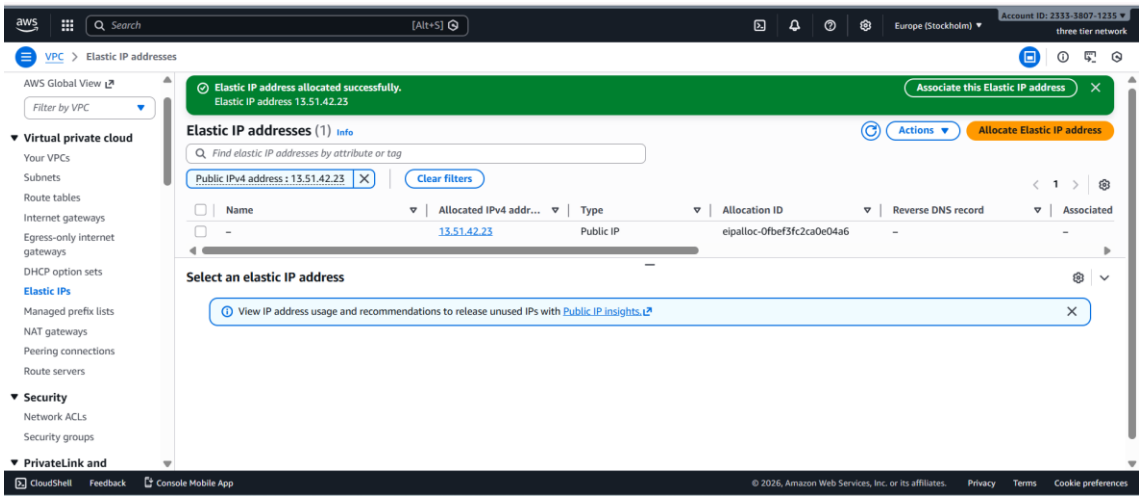
Appendix 3: Attachment of an Internet Gateway to VPC

The following figure shows the successful attachment of the Internet Gateway VPC.



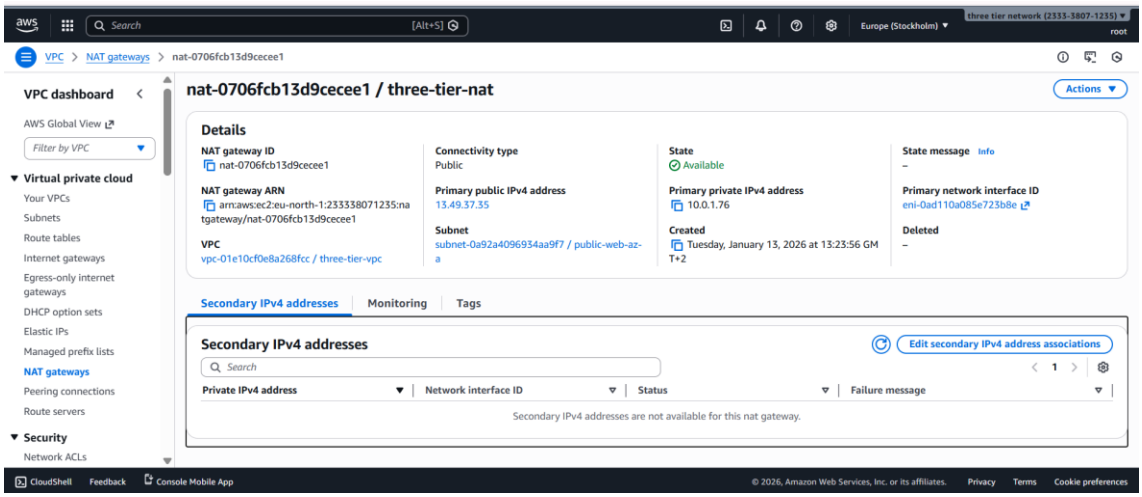
Appendix 4: Elastic IP Allocation/

The following figure shows the allocation of elastic IP address.



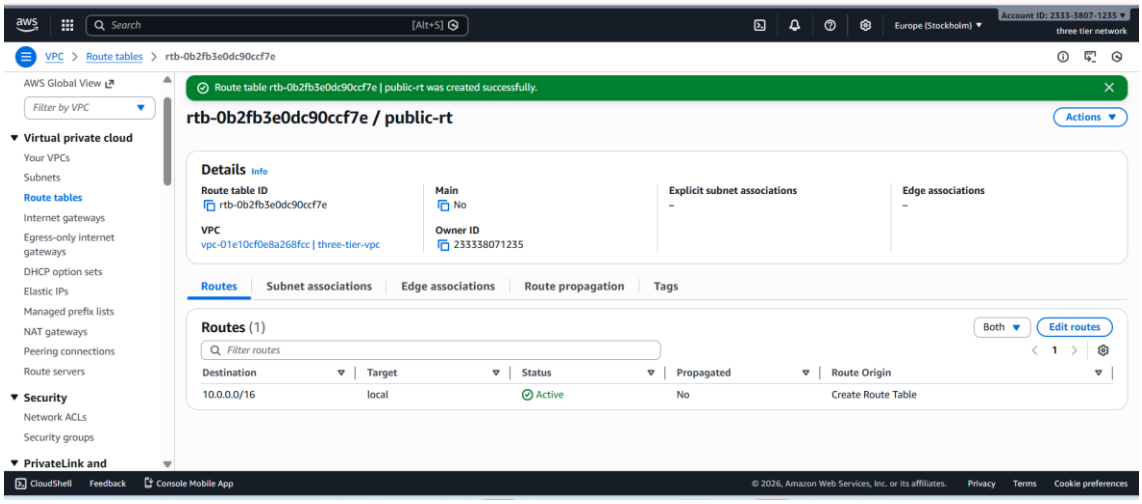
Appendix 5: NAT Gateway Configuration

The following image shows the details of the NAT gateway that was created temporarily for outbound access of private subnets during testing.



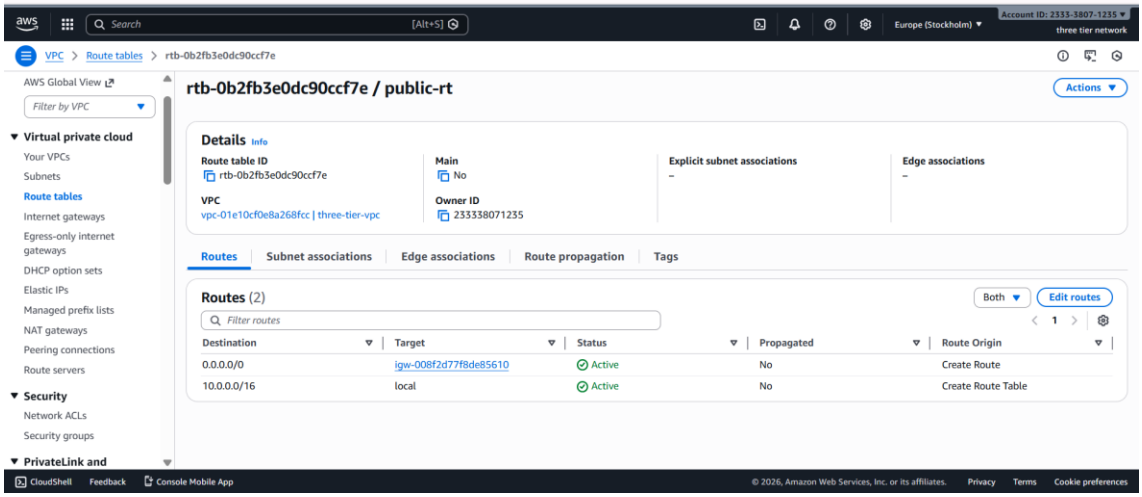
Appendix 6: Creation of Public Route Table

The image shows the creation of the public route table configured with the route to the internet gateway, enabling outbound internet access for resources deployed in the public subnets.



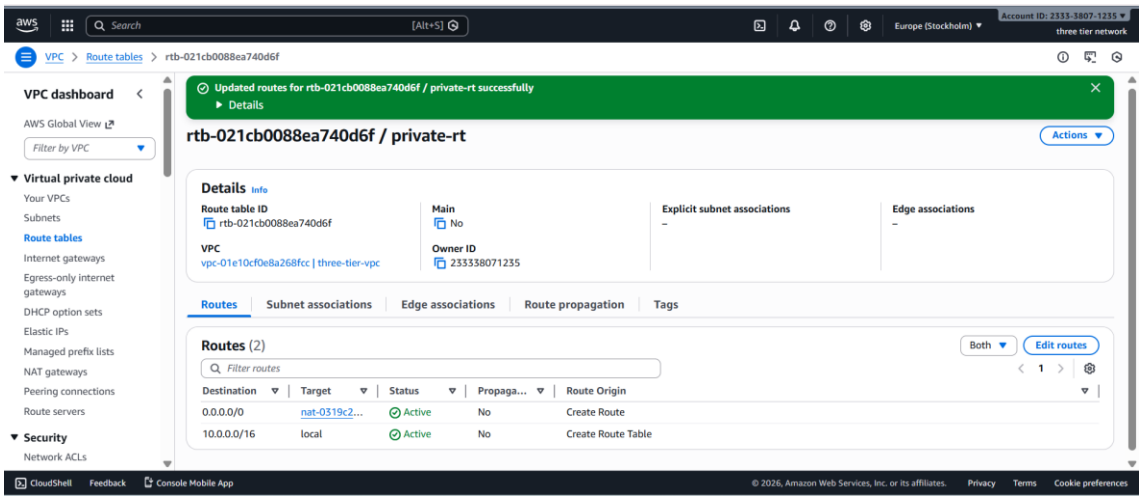
Appendix 7: Verification of Public Route Table

The image shows the correct configuration of the public route table.



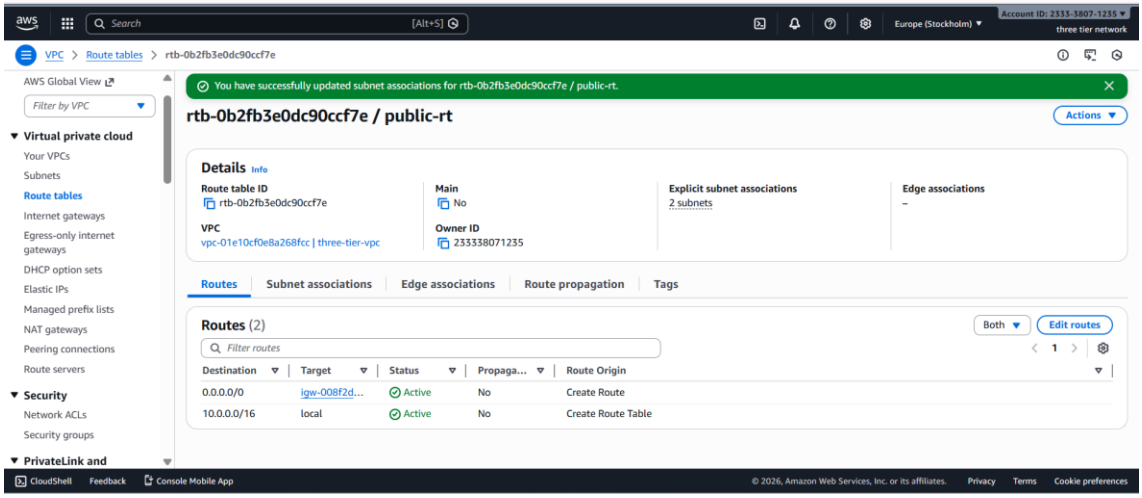
Appendix 8: Details of the private route table

The figure shows the configuration details of the private route table.



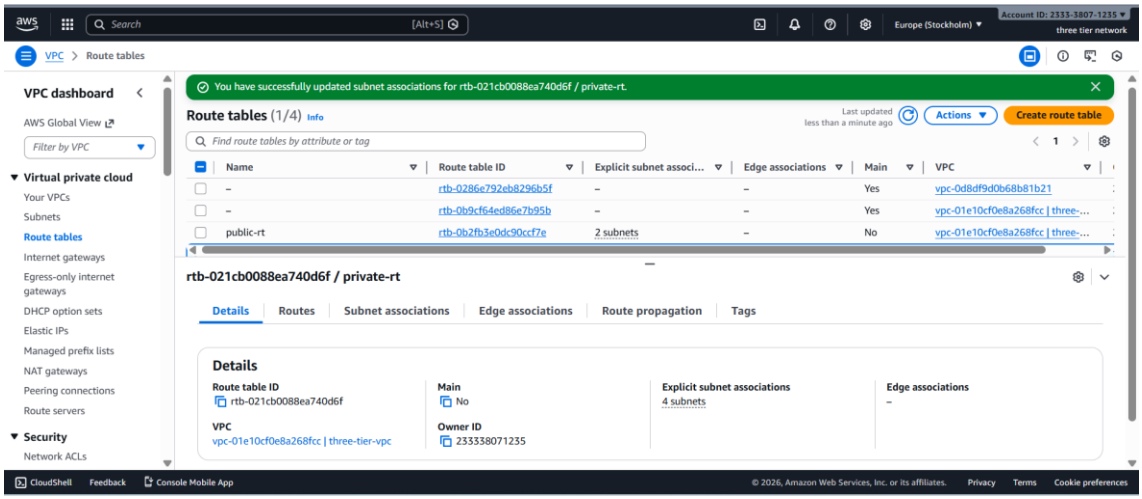
Appendix 9: Public Subnet Association

The image shows the association between the public route table and the public subnet, confirming that internet-bound traffic is correctly routed through the internet gateway.



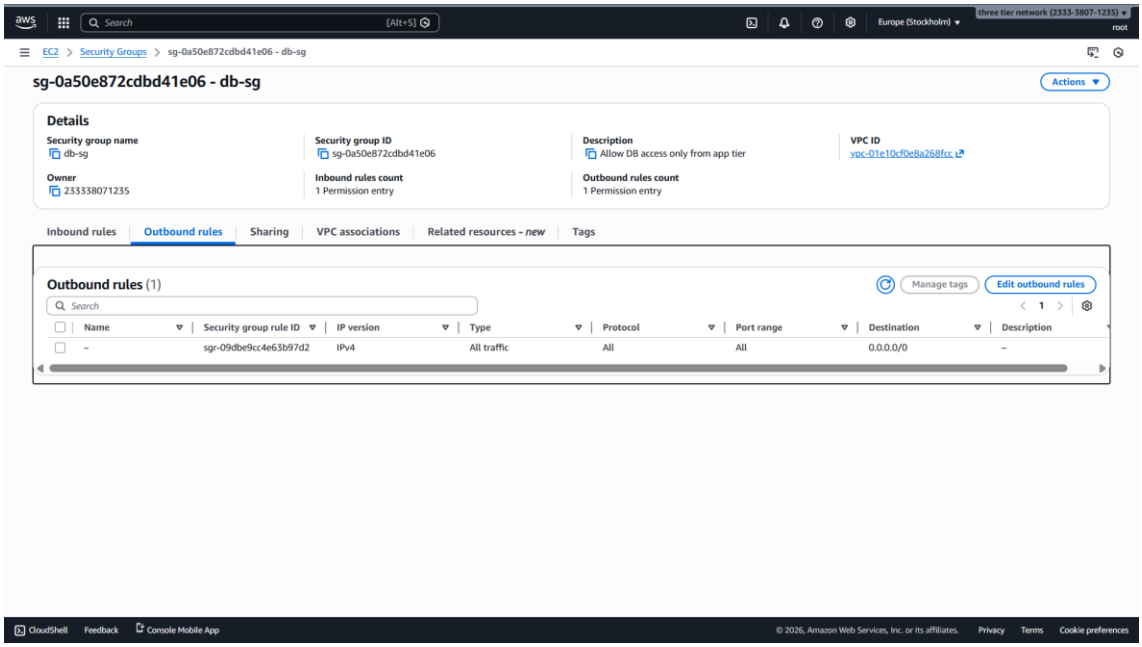
Appendix 10: Private Subnet Association

The figure shows the association between the private route table and the private subnets, confirming that internal traffic is routed within the VPC and not directly exposed to the internet.



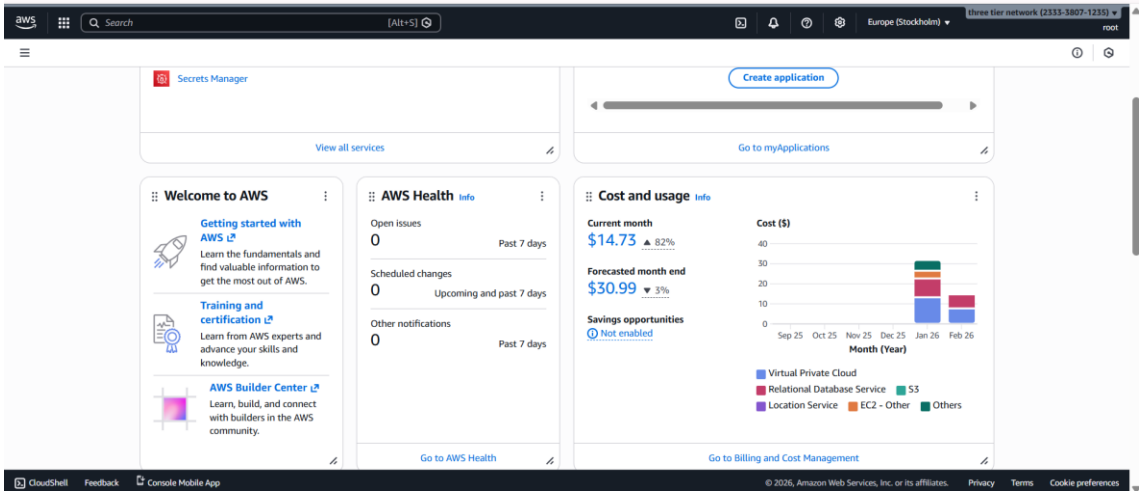
Appendix 11: Outbound rules of the database

The figure shows the configuration of the outbound rule of the database.



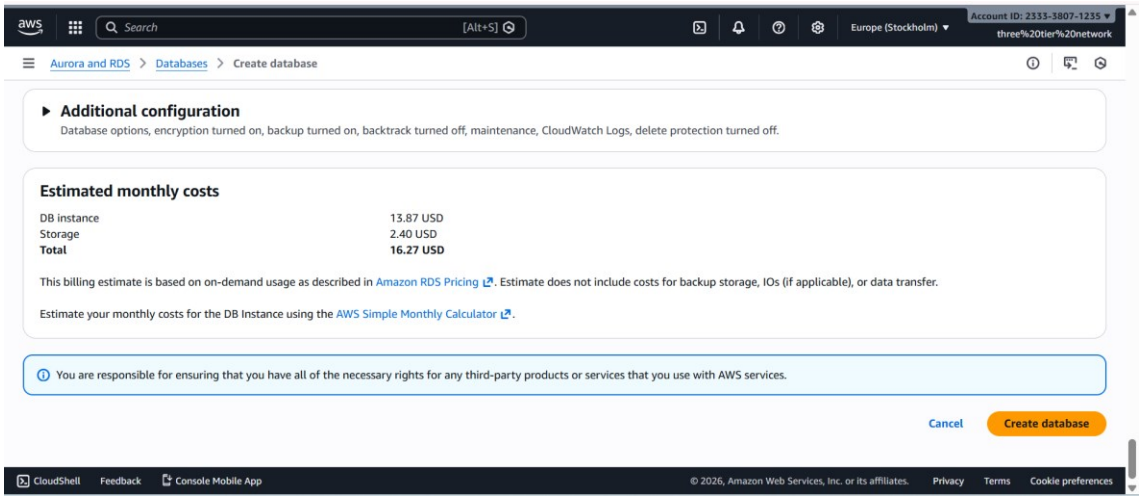
Appendix 12: Usage report of AWS for the thesis project

This figure presents the AWS billing and usage report for the study showing the breakdown of the service consumption.

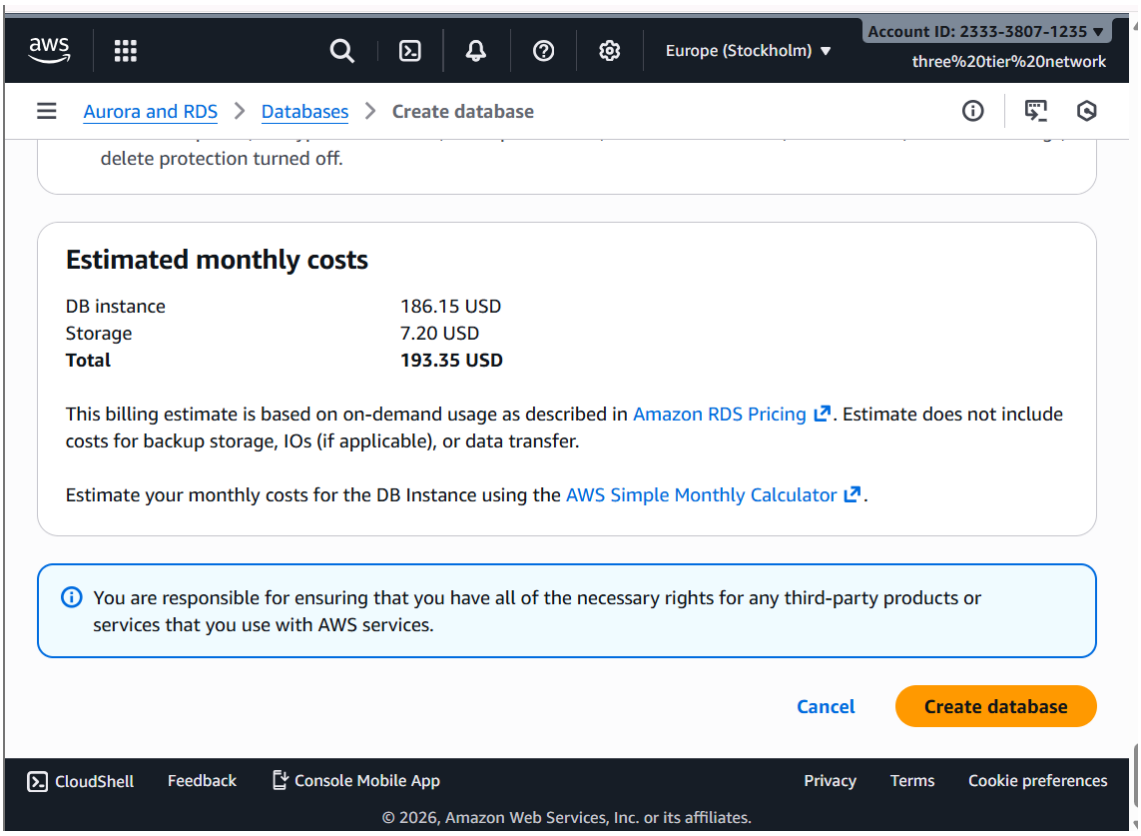


Appendix 13: Estimated monthly cost of Amazon RDS

The following image shows the estimated monthly cost of the RDS (db.t3.micro) in a single availability zone used for the project.



Appendix 14: Estimated monthly cost of Amazon RDS for a larger database with multi-availability zones



Appendix 15: Estimated monthly cost of Amazon RDS for a larger database with a single availability zone