LOG8415 : Lab 3

Advanced Concepts in Cloud Computing

Implementing a DB Cluster

Author

Oussama El-Figha (2324698)

December 03, 2024

Abstract

In this lab assignment, you are asked to set up MySQL cluster on Amazon EC2 and implement cloud design patterns. You are tasked with implementing and deploying an architecture by adding the Proxy and the Gatekeeper patterns. In the first part of the assignment, you will install, configure, and deploy a MySQL stand-alone. Next, you will apply your newly acquired skills to implement a distributed cluster of MySQL databases. You are asked to report your results and analysis by producing a report using LATEX format.

1 Introduction

The goal of this assignment is to implement a MySQL database cluster on AWS EC2 instances using cloud design patterns to ensure scalability, security, and reliability. The task includes:

- Setting up a MySQL standalone server with the Sakila database.
- Implementing a distributed MySQL cluster with data replication.
- Applying the Proxy pattern for efficient request routing.
- Implementing the Gatekeeper pattern to secure the system.
- Automating the entire process using Python scripts with the AWS SDK and Bash scripting.

This document reports the implementation process, benchmarking results, and key observations.

2 Benchmarking MySQL with Sysbench

To validate the MySQL setup, Sysbench was used for benchmarking. The following steps:

1. Install Sysbench on all EC2 instances.
2. Prepare the database with `sysbench oltp_read_write`.
3. Execute benchmarking tests for read and write operations.

Results: The benchmark confirmed the correctness of the MySQL standalone setup, with throughput and response times meeting expected performance levels for `t2.micro` instances.

3 Implementation of the Proxy Pattern

The Proxy pattern was implemented to route requests within the MySQL cluster. A `t2.large` instance was deployed to act as the Proxy server. Three routing modes were implemented:

- **Direct Mode:** All requests are forwarded directly to the manager node.
- **Random Mode:** Read requests are distributed randomly among worker nodes.
- **Customized Mode:** Read requests are routed to the worker node with the lowest ping time.

Implementation Details:

- The Proxy server was configured using FastAPI to handle incoming requests.
- Write requests were forwarded to the manager node to maintain data consistency.
- The logic for selecting worker nodes in Random and Customized modes was implemented using Python.

4 Implementation of the Gatekeeper Pattern

The Gatekeeper pattern was implemented to enhance system security. Two `t2.large` instances were deployed:

- **Gatekeeper:** Handles incoming requests, validates them, and forwards them to the Trusted Host.
- **Trusted Host:** Routes validated requests to the Proxy server.

Security Features:

- The Trusted Host was configured with strict firewall rules and minimal services to reduce the attack surface.
- Communication between the Gatekeeper and Trusted Host used private IPs and internal subnets.

5 Benchmarking the Clusters

The performance of the MySQL cluster was tested using 1000 read and 1000 write requests for each Proxy routing mode. The benchmarking process involved:

1. Sending requests to the Gatekeeper endpoint.

2. Measuring response times and throughput.
3. Logging the results for analysis.

Results:

Metric	**Value**
Total Requests	4000
Total Time	805.65 seconds
Throughput	4.96 requests/second
Average Response Time	201.46 ms

Table 1: Benchmarking results for the MySQL cluster.

6 Description of Implementation

Architecture: The system consists of:

- A manager node and two worker nodes configured for data replication.
- A Proxy server for request routing.
- A Gatekeeper and Trusted Host for secure access.

Automation: The entire setup was automated using the `main_script.py` script, which includes:

- Key pair creation and management.
- EC2 instance provisioning.
- MySQL installation and replication setup.
- Deployment of Proxy and Gatekeeper patterns.
- Benchmarking the system.

7 Summary of Results and Instructions

7.1 Summary of Results

The implementation demonstrated:

- Scalability through the Proxy pattern.
- Enhanced security via the Gatekeeper pattern.
- Automated deployment and benchmarking.

7.2 Instructions to Run the Code

1. Clone the repository `https://github.com/oussamaelfig/LOG8415_Final_Project` and navigate to the project directory.
2. Run `python3 main_script.py` to deploy the system.
3. Analyze the benchmarking results in `benchmark_log.txt`.
4. Use `python3 Utilities\terminate.py` to clean up the resources.