Log Ingestor

A Logs Ingestion and Analysis System for Distributed Systems

Project - IT214 - Database Management System
Prof. P M Jat

Group Members

1. Dhruv Jain - 202301272
2. Jevik Rakholiya - 202301276
3. Rujal Jiyani - 202301277

Objective :

Design and implement a PostgreSQL Database for efficient ingestion of logs and analysis of a distributed environment/system by optimal segregation of logs and quick querying logics.

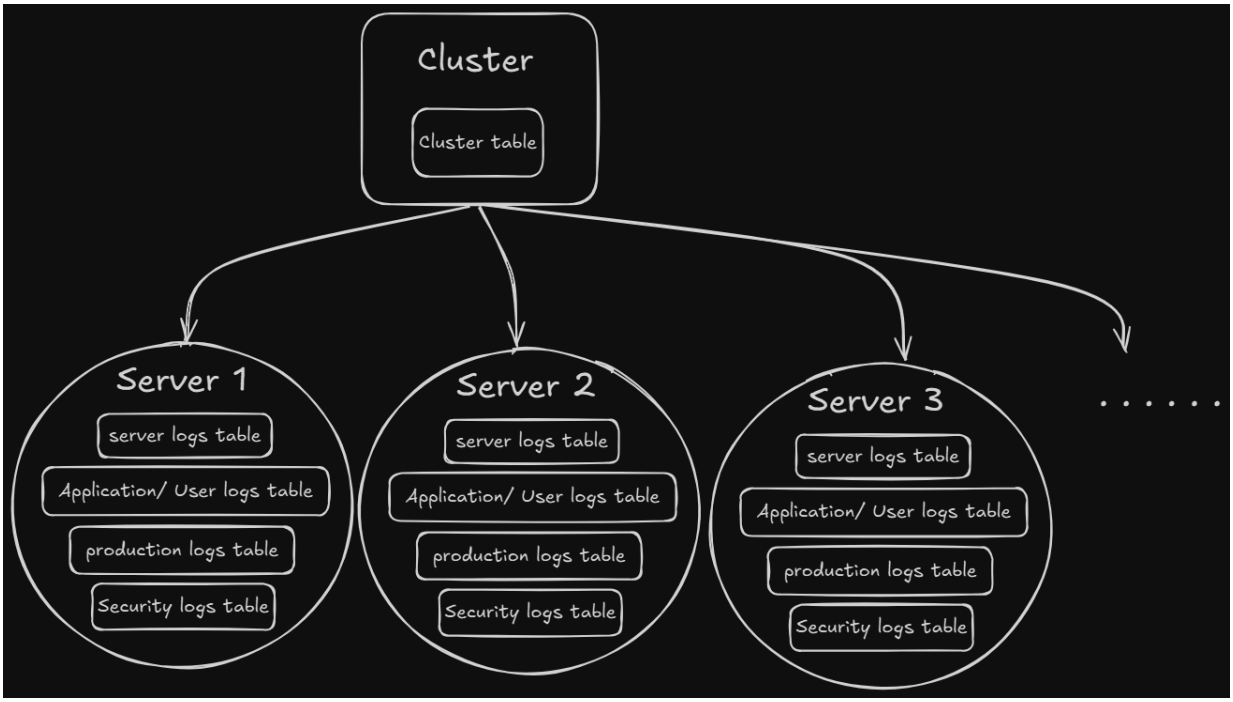
Scenario Description :

In modern IT environments, organizations run multiple servers, applications, and security systems and that generate massive amounts of log data (a computer-generated file that records information about activities, operations, and usage patterns). These logs contain crucial information about system performance, security events, and user activities. This log files are essential for monitoring the server and system working , which generates the need of systematic and efficient management of log files by a centralized log management system.

However, managing logs in a distributed environment is challenging due to:

- Scattered Logs – Logs are stored across different servers, making it difficult to retrieve and analyze them.
- Unstructured Data – Raw log files are often unstructured and difficult to query.
- Slow Troubleshooting – Without a centralized system, debugging issues becomes time-consuming.
- Security & Compliance Risks – Organizations must track failed login attempts, firewall activity, and suspicious behavior for cybersecurity compliance.

Our Intuitive Approach :



Explanation:

As modern applications run across **multiple servers, microservices, and APIs,** they generate **huge volumes of log data** in real-time. A centralized log database would lead to high storage consumption and slow query performance.

To solve this, we implement **Sharding**, a database partitioning technique that **distributes logs across multiple smaller databases**, improving query **efficiency and scalability**. So here we instead of creating a single table for each individual format of log files, we created a separate smaller composition of tables under each server storing log files of each format for that particular server only.

Also the log files of each company can be in **different format** company wise, therefore each server will have a **uniform log format** based on the company which is using that server. Different Log Formats are mentioned after this section.

Also, In this system, we **divide logs across multiple databases (shards) per server** while maintaining a **central metadata table** to track the location of logs.i.e **Cluster Table.** Our Cluster will contain a table for information of each server like, log formats, service provider(AWS, Google Cloud, Microsoft Azure, Apache etc..), billing of that server, server id etc.

Further expanding the server cluster will have details about server and that will be linked to the tables of different log type files, say **Server 1,** is runned by **AWS and using syslog format** will have its own server, production, user, security log tables and all will have the log files of the same format , similarly say **Server 2** runned by **Apache and using CLF log** format will also follow the same thing and so on. The plus point of this is that it will help to maintain the same log format files under the same server database which will make easy access to the log file.

The different section/types of log files that are being stored as a table under a server are as follows :

Log Sources & Types of Logs Captured :

Log Source	Example Logs Captured
Server Logs	Application and Microservices logs, Server startup/shutdown, resource usage (CPU, RAM, Disk)
Production Logs	Deployment logs(Minor/Major updates logs), Crash logs
Security Logs	Brute-force attacks, unauthorized access attempts, Blocked/allowed connections, DDoS detection, Bandwidth usage, VPN connections, IP blacklisting
Application /User Logs	API requests, user interactions like successful & failed logins/logout , errors & exceptions, password changes, CURD operations
Cluster Logs	Billing of cloud services, upscaling and downscaling of servers

Different Log Formats :

Log Format	Used By	Example
Common Log Format (CLF)	Apache, Nginx	`127.0.0.1 - - [10/Feb/2025:14:22:31 +0000] "GET /index.html HTTP/1.1" 200 5120`
Extended Log Format (ELF)	IIS, Proxies	`2025-02-10 14:22:31 192.168.1.1 GET /login 200 5120`
JSON Logs	Kubernetes, APIs	`{ "timestamp": "2025-02-10T14:22:31Z"j, "method": "GET", "status": 200 }`
Syslog (RFC 5424)	Linux, AWS EC2, Google GCE, Azure, Firewalls	`<34>1 2025-02-10T14:22:31Z server1 sshd 1234 INFO "User login successful"`
Nginx Logs	Nginx Web Server	`192.168.1.1 - - [10/Feb/2025:14:22:31 +0000] "GET /home HTTP/1.1" 200 5120`

Application Users :

1. System Administrator
2. Company Administrator
3. Security Analysts
4. Developers & DevOps Engineers

I. [Use Cases] - System Administrator :

The System Administrator is responsible for monitoring server health, resource usage, and failures. The Log Ingestion System helps administrators collect, store, and analyze server logs in real-time. This ensures system stability, performance optimization, and quick resolution of issues.

Server Health and Resource Usage : The system administrator can check the server health status by checking the CPU usage, Memory Consumption, Disk Space, Network Traffic by optimized query and can generate reports.

System Crash : The system administrators can survey if a server crashes by querying the security and server logs.

Billing : The system administrator can track the usage and resource consumption of a particular server runned by a company, and then eventually can charge the company for the use. The system admin can query about resource usage and calculate.

II. [Use Cases] - Company Administrator :

This use case deals with the company who is using the servers, it helps the company to analyze and trace its progress. The company administrator can monitor biling of servers, increment in users of their product after an Ad campaign and also track the working of the developers.

Progress Report : The company can query about the traffic on the server which helps them to estimate their progress. Further, they can query about the number of users active on the company server.

Billing : On the basis of the resources used , the company can estimate the amount to be paid to the service provider.

III. [Use Cases] - Security Analysts

Security Analysts require an in-depth knowledge of events that led to a crash by analyzing logs from different sources (servers, firewalls, cloud services, applications). Security analysts can leverage it for threat detection, forensic analysis, compliance, and incident response.

Unauthorized Access : By querying the user log files the analyst can check about the number of unauthorized access and failed login attempts to the system.

Firewall Threats : The security analysts can also check upon the firewall and also monitor the firewall penetration or VPN interference.

DDoS detection : By monitoring network traffic and tracking request made by a user ip, they can defend servers against DDoS attacks.

IV. [Use Cases] - Developers & DevOps Engineers

Deploying code to a server requires several process, each generating feebacks which are logged, these feebacks helps developers to find bugs in their code, optimize the runtime etc. Developers and DevOps engineers scale their code according to the network traffic.

Production Success/Crash : Production logs helps developers to find bugs, optimize their code, improve quality of code etc.

Application/Micorservice : By analyzing how users interact with their application and how much API requests a mircoservice might handle, Developers can optimize their code to reduce response time of each API request.

Overview of all use cases:

Company Administrations - View growth rate, user increase, billing of server

System Administrators – Monitor server health, resource usage, and crashes.

Security Analysts – Detect and investigate unauthorized access, failed login attempts, and firewall threats.

Developers & DevOps Engineers – Debug application errors, API failures, and database issues.

Queries :

1. **[Queries] System Administrator:**
    - Server Health - Memory, Disk Space and CPU usage in a time period
    - Network Traffic - Querying peak time of usage
    - System Crash
    - Billing - Overall cost of running a cluster

2. **[Queries] Company Administrator:**
    - Progress Report - Number of new users logins, Increase in Network Traffic according to season, Ads campaigns success etc..
    - Billing - Overall cost of running a cluster

3. **[Queries] Security Analysts:**
    - Brute-force attacks - unauthorized access attempts
    - Blocked/allowed connections - IP blacklisting
    - DDoS detection
    - VPN connections

4. **[Queries] Developers & DevOps Engineers:**
    - Production Success/Crash - Application and Microservice prodcution logs of a feature
    - Network Traffic - Querying peak usage of a microservice