Distributed Log Aggregator - Milestone 2 Documentation

Prisacaru Bogdan-Paul, 343C4

December 14, 2025

1 Overview

The **Distributed Log Aggregator** is a scalable, microservices-based log aggregation and analytics platform. It is built using **Docker Swarm**, **FastAPI**, **Keycloak**, and **OpenSearch**.

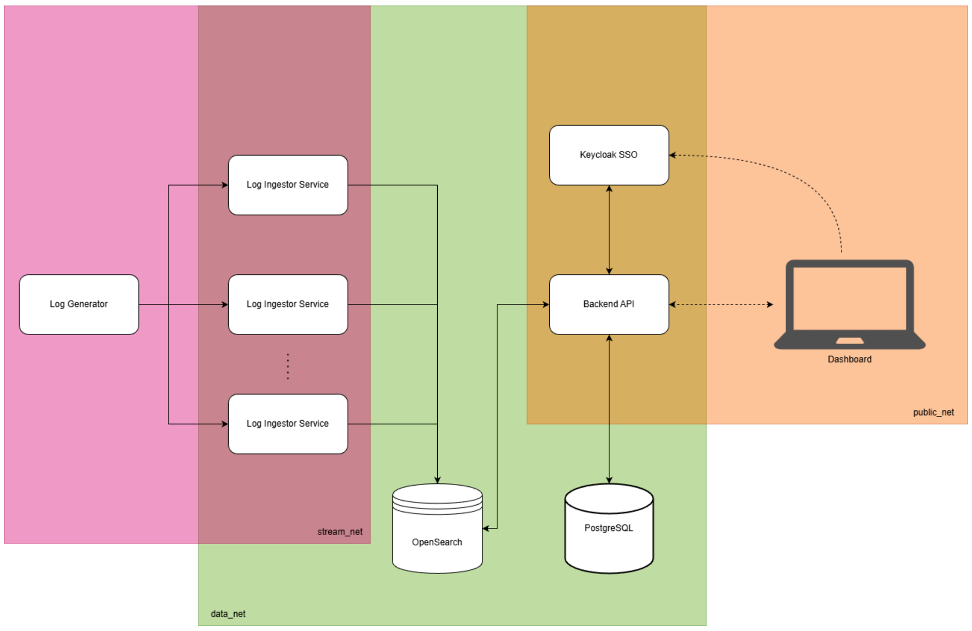

Figure 1: Milestone 2 System Architecture Overview

2 Architecture

The system is composed of the following microservices:

- **Dashboard Service**: A FastAPI-based frontend and API providing a user interface for log visualization, filtering, and exporting. It handles authentication via Keycloak.
- **Ingestor Service**: A lightweight service dedicated to receiving logs from various sources and indexing them into OpenSearch.
- **Log Generator**: A utility service that generates mock log traffic to simulate a real-world distributed system.
- **OpenSearch**: The core search and analytics engine used to store and query logs.
- **Keycloak**: Handles Identity and Access Management (IAM) and Single Sign-On (SSO).
- **PostgreSQL**: Stores user profile data and application-specific metadata.

3 Key Features

3.1 Authentication

- **Single Sign-On (SSO)**: Powered by Keycloak (OpenID Connect).

3.2 Role-Based Access Control (RBAC)

The system implements secure access management with distinct roles. The permissions are defined as follows:

Feature	**Admin**	**Developer**	**Viewer**
Login via SSO	✓	✓	✓
Search & Filter	✓	✓	✓
Time Window	Unlimited	Unlimited	Last 3 Hours
Log Visibility	All Levels	All Levels	INFO/WARN Only
Export Data	✓	✓	×

3.3 Advanced Filtering & Search

- **Full-Text Search**: Search log messages using OpenSearch.
- **Structured Filters**: Filter by **Service**, **Log Level**, and **Time Range**.
- **Pagination**: Efficiently browse through logs.

3.4 Data Export

- Export filtered logs to **CSV** or **JSON** formats.
- The export functionality respects active filters (e.g., filtering for "ERROR" results in an export containing only errors).

3.5 Scalable Ingestion

- Implements a time-based indexing strategy (`app-logs-YYYY.MM.DD`) for efficient storage management.

4 Setup & Deployment

4.1 Prerequisites

- Docker & Docker Compose
- Docker Swarm initialized (`docker swarm init`)

4.2 Deployment Steps

1. **Build the Services**:

```
docker compose build
```

2. **Deploy the Stack**:

```
docker stack deploy -c docker-compose.yml log_stack
```

3. **Access the Dashboard**: Open `http://localhost:8000` in your browser.
4. **Access Keycloak Console**: Open `http://localhost:8080`.

5 Default Credentials

5.1 Application Users (Log Realm)

Role	Username	Password
Admin	`admin`	`admin`
Developer	`developer`	`developer`
Viewer	`viewer`	`viewer`

5.2 Keycloak Administration Console

- **URL**: `http://localhost:8080`
- **Username**: `admin`
- **Password**: `admin`

6 Testing

To generate traffic, ensure the log generator is running:

```
docker service scale log_stack_log-generator=1
```

To stop generation:

```
docker service scale log_stack_log-generator=0
```

6.1 Automated Testing with Postman

A comprehensive Postman collection is provided in `tests/postman_collection.json` to validate the system's functionality and security requirements.

The collection includes tests for:

- **Authentication**: Verifying login flows for Admin, Developer, and Viewer roles using Keycloak's API.
- **RBAC Enforcement**: Ensuring that restricted endpoints (e.g., `/export`) are accessible only to authorized roles.
- **Log Ingestion**: Verifying that logs are correctly accepted by the Ingestor Service.
- **Search Functionality**: Validating that ingested logs are indexed and searchable via the Dashboard.

To run the tests:

1. Import the `postman_collection.json` file into Postman.
2. Ensure the stack is running locally.
3. Run the collection using the Postman Collection Runner.

7 Project Repository

The complete source code for this milestone is available on GitHub:

`https://github.com/BogdanPaul15/Distributed-Log-Aggregator`