PROJECT REPORT

BIG DATA ECOSYSTEM

Noé PHAM & Elanore LELIEVRE

ING 5 BIG DATA & IA GR01

Introduction

The objective of this project was to implement a distributed data pipeline using open-source distributed systems, specifically the **ELK Stack** (Elasticsearch, Logstash, Kibana) integrated with a **MySQL** database. The goal was to demonstrate the use of these technologies to efficiently ingest, process, and visualize data from a relational database in a real-time environment. The project aimed to establish the entire setup using **Docker Compose** to manage the containers for Elasticsearch, Logstash, Kibana, and MySQL.

The main challenges encountered during the project revolved around ensuring seamless connectivity between **Logstash** and both **Elasticsearch** and **MySQL**. Misconfigurations, incorrect parameters, and network issues created significant hurdles during the setup.

Step 1: Setting Up MySQL

The first step in the project was to set up the **MySQL** database. To begin with, a MySQL container was created using Docker. A popular example dataset, **Chinook**, was used for this project. The following key actions were taken during this step:

1. **Docker Container for MySQL**: A MySQL container was created using the official MySQL Docker image. This involved configuring the docker-compose.yml file to define the MySQL service, setting the root password, and ensuring that MySQL was properly configured to allow connections from Logstash.
2. **Database Initialization**: Once the container was running, the Chinook database was initialized by running SQL scripts that populated the database with tables like Albums, Artists, Customers, and Invoices.
3. **Networking Configuration**: One of the challenges during the MySQL setup was configuring the network properly. In the containerized environment, it is crucial that the **Logstash** container can access the **MySQL** container. Misconfigured networks or missing links in the Docker Compose file could cause Logstash to fail when trying to connect to MySQL.
4. **Testing MySQL**: To verify that MySQL was working correctly, simple SQL queries were run against the Chinook database to ensure data was properly populated and accessible.

Step 2: Implementing the ELK Stack

The next major step involved setting up the **ELK Stack** (Elasticsearch, Logstash, and Kibana), which required defining each component in the docker-compose.yml file and ensuring that they could work together smoothly.

1. **Elasticsearch Setup**:
    - Elasticsearch was the first service added to the Docker Compose file. This required specifying the container image, configuring the memory limits, and setting the appropriate environment variables such as the ELASTIC_PASSWORD and KIBANA_PASSWORD.

    - After launching the container, Elasticsearch was verified by connecting to its REST API to confirm that it was accessible via the browser and that it could store and index data.

2. **Kibana Setup**:
    - Kibana was configured next, pointing it to the running Elasticsearch instance. Kibana allows users to query and visualize the data stored in Elasticsearch. The Kibana container was set to connect to Elasticsearch over the specified ports and ensured that the Kibana interface was accessible via the browser at localhost:5601.

Step 3: Configuring Logstash to Connect to MySQL and Elasticsearch

Logstash was the most challenging component of the setup, as it required two crucial configurations: connecting to **MySQL** for data ingestion and pushing data to **Elasticsearch** for indexing. This configuration involved creating a Logstash pipeline with input, filter, and output sections.

1. **Logstash and MySQL Connectivity**:
    - The primary difficulty in this phase was ensuring that **Logstash** could access the MySQL database container. The connection to MySQL required the use of the **JDBC plugin** in Logstash. The correct JDBC URL was crucial, along with the credentials for connecting to MySQL.
    - Initially, Logstash failed to connect to MySQL due to incorrect hostnames. The hostname had to be set to the name of the MySQL service in the Docker Compose network (mysql-db) rather than localhost or 127.0.0.1. Additionally, issues with firewall settings and network bridging between containers delayed the initial connection.

2. **Logstash and Elasticsearch Connectivity**:
    - The next step was configuring Logstash to send the data to Elasticsearch. The configuration for the output plugin involved specifying the Elasticsearch host and port, along with authentication credentials.
    - Logstash failed to push data to Elasticsearch due to authentication errors, which were traced back to incorrect environment variables or missing configurations for ELASTIC_PASSWORD in the .env file.
    - Furthermore, Logstash containers faced networking issues that prevented them from connecting to Elasticsearch running in another container. This issue was resolved by ensuring that both containers were part of the same Docker Compose network and using the correct service name (elasticsearch) as the hostname for Elasticsearch in the Logstash output configuration.

3. **Data Ingestion from MySQL to Elasticsearch**:

  - After resolving the connectivity issues, data ingestion from MySQL to Elasticsearch began. This involved defining an SQL query in the Logstash input section to extract data from MySQL and then mapping this data into Elasticsearch indexes.
  - One challenge in this phase was ensuring that the data from MySQL was appropriately transformed before being ingested into Elasticsearch. Field renaming and type mappings were necessary to ensure consistency in the Elasticsearch index.

4. **Logstash Debugging**:
  - During the setup, frequent errors such as connection timeouts and configuration mismatches were encountered. The Logstash logs were invaluable in diagnosing issues related to data ingestion and output. By analyzing the logs, it was possible to identify misconfigured plugins or missing environment variables and address them step-by-step.

Step 4: Visualizing Data in Kibana

Once the data was ingested into Elasticsearch, the final step was to visualize it using **Kibana**. The Kibana interface was accessed through a web browser, and Elasticsearch indices were loaded to create dashboards and visualizations.

1. **Kibana Dashboards**: Kibana provided an easy way to interact with the data stored in Elasticsearch. Simple visualizations, such as bar charts and tables, were created to analyze the data. The configurations for Kibana were minimal, as the major challenge was ensuring that the data was correctly indexed in Elasticsearch.

Problems Encountered

Throughout the implementation, several issues were encountered:

1. **Connectivity Between Logstash and MySQL**:
  - Initial connection failures were primarily due to incorrect hostnames and networking issues in Docker Compose. Logstash could not resolve the mysql-db container by the hostname localhost or 127.0.0.1, leading to connection errors. This was resolved by configuring the network settings correctly in Docker Compose.
2. **Authentication Issues**:
  - Logstash encountered multiple authentication errors when trying to push data to Elasticsearch. The issue was traced back to missing or incorrect environment variables such as ELASTIC_PASSWORD. Ensuring that the correct credentials were passed in the .env file resolved the issue.
3. **Data Ingestion Failures**:

- Misconfigured SQL queries or incorrect field mappings caused issues with data ingestion. Logstash was not able to ingest data from MySQL into Elasticsearch in a consistent format, which required several debugging sessions to correct field names and mappings.

4. **Elasticsearch Memory Issues**:
    - Elasticsearch required significant memory to handle large amounts of data, and initially, the memory limits specified in the Docker Compose configuration were insufficient. Increasing the memory allocation resolved these performance issues.

Conclusion

The integration of **MySQL**, **Elasticsearch**, **Logstash**, and **Kibana** presented multiple challenges, particularly related to network configurations and authentication issues. However, by carefully configuring each component, troubleshooting connectivity problems, and analyzing Logstash logs, the project was successfully implemented. The final system allowed for real-time data ingestion, storage, and visualization, demonstrating the power and flexibility of open-source distributed systems like the ELK Stack. Despite the challenges, this project provided valuable experience in handling real-world issues related to distributed system configurations and deployments.