**Tech Design - Wakanda Weather Monitoring System**

**Table of Contents**

1. [Introduction](https://chat.openai.com/c/c3021eda-6702-471c-aa69-cc4bf9761050#introduction)
2. [System Architecture](https://chat.openai.com/c/c3021eda-6702-471c-aa69-cc4bf9761050#system-architecture)
3. [Sequence Diagrams](https://chat.openai.com/c/c3021eda-6702-471c-aa69-cc4bf9761050#sequence-diagrams)
   * [Data Ingestion and Processing](https://chat.openai.com/c/c3021eda-6702-471c-aa69-cc4bf9761050#data-ingestion-and-processing)
   * [Database Synchronization](https://chat.openai.com/c/c3021eda-6702-471c-aa69-cc4bf9761050#database-synchronization)
4. [Kafka Topic Definitions](https://chat.openai.com/c/c3021eda-6702-471c-aa69-cc4bf9761050#kafka-topic-definitions)
5. [Kafka Connectors](https://chat.openai.com/c/c3021eda-6702-471c-aa69-cc4bf9761050#kafka-connectors)
6. [Database Schema](https://chat.openai.com/c/c3021eda-6702-471c-aa69-cc4bf9761050#database-schema)
7. [Retry Policy Strategy](https://chat.openai.com/c/c3021eda-6702-471c-aa69-cc4bf9761050#retry-policy-strategy)
8. [Logger Mechanism](https://chat.openai.com/c/c3021eda-6702-471c-aa69-cc4bf9761050#logger-mechanism)
9. [Security Measures](https://chat.openai.com/c/c3021eda-6702-471c-aa69-cc4bf9761050#security-measures)
10. [Performance Considerations](https://chat.openai.com/c/c3021eda-6702-471c-aa69-cc4bf9761050#performance-considerations)

**Introduction**

The Tech Design document provides a comprehensive technical view of the Wakanda Weather Monitoring System, including system architecture, data flows, Kafka integration, database schema, security measures, performance considerations, a retry policy strategy, and a logger mechanism.

**System Architecture**

The system adheres to a microservices architecture, consisting of the following components:

* **Weather Satellites:** Responsible for transmitting weather readings.
* **Weather Stations:** Receive and process satellite readings.
* **Data Processing Component:** Calculates weather metrics and temporarily caches data.
* **Kafka:** Acts as the messaging system for real-time data ingestion and component communication.
* **Kafka Connectors:** Facilitate data integration between Kafka and the MySQL database.
* **MySQL Database:** Stores historical weather data and processed metrics.

A diagram of a network

Description automatically generated

**Description:** This sequence diagram illustrates how data synchronization is achieved between Kafka and the MySQL database, ensuring data persistence.

**Kafka Topic Definitions**

* **satellite-readings:** Kafka topics for receiving satellite readings.
* **data-processing:** Kafka topics for forwarding processed data.

**Kafka Connectors**

Two Kafka Connect connectors are configured:

1. **Satellite-to-Kafka Connector:** Ingests satellite readings into Kafka.
2. **Kafka-to-Database Connector:** Transfers data from Kafka topics to the MySQL database.

**Database Schema**

The MySQL database schema includes the following tables:

* **weather\_satellite:** Stores satellite information.
* **weather\_station:** Stores weather station details.
* **satellite\_readings:** Temporarily caches incoming satellite readings.
* **processed\_data:** Stores processed weather metrics.
* **forecasts:** Archives generated weather forecasts.

**Retry Policy Strategy**

To ensure system resilience, a retry policy strategy is implemented. Key components of this strategy are as follows:

* **Exponential Backoff:** In the event of a failure (e.g., communication issues with Kafka or the database), the system initiates an exponential backoff mechanism. This entails increasingly longer delays before retrying the operation, preventing overloading in case of transient issues.
* **Maximum Retry Attempts:** A defined maximum number of retry attempts is set for each operation. Upon reaching this limit, the system logs the failure and requires manual intervention.
* **Retry Delay:** Initial retry delays are minimal, with subsequent retries incorporating longer delays. This approach facilitates recovery from transient issues without causing immediate retry storms.
* **Error Handling:** All errors occurring during retries are logged for monitoring and analysis. Details include the nature of the error, the number of retry attempts, and the final status (success or failure).

**Logger Mechanism**

A comprehensive logging mechanism is integrated into the system to facilitate troubleshooting, monitoring, and auditing. The logging mechanism encompasses the following:

* **Log Levels:** Different log levels (e.g., INFO, WARN, ERROR, DEBUG) categorize log messages for various purposes and levels of detail.
* **Contextual Logging:** Log messages contain contextual information, such as component identification, timestamps, and triggering events.
* **Centralized Logging:** All log entries are centrally stored in a dedicated logging repository, providing easy access for analysis and debugging.
* **Log Retention:** A log retention policy is in place to efficiently manage log storage. It addresses archiving or purging of older logs based on retention periods and compliance requirements.
* **Real-time Monitoring:** Critical system operation-related log entries are continuously monitored in real-time. Alerts are generated for specific log entries necessitating immediate attention.
* **Log Rotation:** Log files undergo rotation at predefined intervals to prevent excessive disk space consumption.

The integration of the retry policy strategy and the logger mechanism ensures that the system gracefully recovers from transient failures and offers a robust mechanism for tracking and diagnosing issues. This approach significantly contributes to the system's reliability and maintainability.

**Security Measures**

* **Data Encryption:** Data transmission between system components is encrypted to maintain privacy.
* **Access Control:** Role-based access control restricts access to sensitive data.
* **Firewalls:** Firewalls safeguard the Kafka cluster and database from unauthorized access.

**Performance Considerations**

* **Scaling:** The system supports horizontal scaling by adding more weather stations and satellite-to-Kafka connectors.
* **Cache Management:** The data processing component effectively manages cache size to ensure optimal performance.
* **Optimized Queries:** Database queries are optimized to enhance data retrieval efficiency.

The combined implementation of these elements ensures the system's resilience, robustness, and effectiveness in monitoring and providing real-time weather data for Wakanda.