In case of nifi errors/issues in the container, change to one of the following nifi image versions in docker-compose.yml file:

1) apache/nifi:1.14.0
2) apache/nifi:1.12.0
3) pavansrivathsa/nifi

Apache NiFi:
Apache NiFi, is an open-source data flow management tool that provides a web-based interface for designing, managing, and monitoring data flows in real-time. It was initially developed by the National Security Agency (NSA) and was later made open source and donated to the Apache Software Foundation (ASF).

NiFi is designed to handle the flow of data between systems in a secure, reliable, and scalable manner. It provides a graphical interface for designing data flows using a visual canvas of interconnected processors, which represent data sources, processors, and data sinks. Users can create, modify, and manage data flows in real-time through the web-based interface, making it easy to implement data integration, data transformation, and data routing scenarios.

Some of the key features of NiFi include:

1. Data ingestion: NiFi can efficiently collect, aggregate, and process large volumes of data from various sources, such as databases, APIs, sensors, logs, and social media feeds.

2. Data transformation: NiFi provides a wide range of built-in processors for data transformation, enrichment, validation, and filtering. Users can also create custom processors to meet specific data processing requirements.

3. Data routing: NiFi supports data routing based on various conditions, such as content, schema, or metadata, allowing users to dynamically route data to different destinations based on business rules.

4. Data security: NiFi includes features such as SSL/TLS encryption, authentication, authorization, and data lineage tracking to ensure the security of data flows.

5. Data provenance: NiFi maintains a complete history of data flows, allowing users to track and audit data lineage, provenance, and modifications for compliance and troubleshooting purposes.

6. Extensibility: NiFi provides a plugin-based architecture that allows users to extend its functionality with custom processors, controller services, and reporting tasks.

7. Scalability: NiFi is designed to scale horizontally, allowing users to deploy it in a distributed and clustered mode to handle large-scale data processing requirements.

NiFi is widely used in various industries, such as finance, healthcare, telecommunications, government, and e-commerce, for data integration, data processing, data streaming, and data flow management. It has a vibrant community of users and contributors, and its active development is supported by the Apache Software Foundation.

Apache Kafka:
Apache Kafka is an open-source distributed event streaming platform used for building real-time data streaming and processing applications. It was initially developed by LinkedIn and later donated to the Apache Software Foundation, where it is now maintained as a top-level Apache project.

Kafka is designed to handle high-throughput, fault-tolerant, and scalable data streaming scenarios. It provides a distributed architecture with high availability, durability, and fault tolerance, making it suitable for use cases that require processing large volumes of data in real-time. Kafka is based on a distributed publish-subscribe model, where producers write data to Kafka topics, and consumers read from those topics, enabling asynchronous data processing and decoupling of data producers and consumers.

Some of the key features of Kafka include:

1. Distributed architecture: Kafka is designed to be distributed across multiple nodes, allowing for horizontal scalability and fault tolerance. It uses a distributed storage system that replicates data across multiple brokers, ensuring high availability and durability.

2. High-throughput data streaming: Kafka is capable of handling millions of events per second, making it suitable for high-volume data streaming scenarios.

3. Fault tolerance and durability: Kafka provides fault tolerance through data replication across multiple brokers, ensuring data durability and availability even in the presence of failures.

4. Scalability: Kafka is designed to scale horizontally, allowing users to add or remove brokers dynamically to handle changing data processing requirements.

5. Pub-Sub model: Kafka uses a publish-subscribe model, where producers write data to topics, and consumers read from those topics. This enables decoupling of data producers and consumers, allowing for flexibility and scalability in data processing.

6. Stream processing: Kafka has built-in support for stream processing through Kafka Streams, a stream processing library that allows users to process data in real-time and build stream processing applications.

7. Connectors and ecosystem: Kafka has a rich ecosystem of connectors that allow integration with various data sources and sinks, such as databases, message queues, data lakes, and analytics frameworks. It also integrates well with other Apache projects like Apache Flink, Apache Spark, and Apache NiFi.

Kafka is widely used in various industries, such as finance, e-commerce, social media, gaming, and telecommunications, for building real-time data streaming and processing applications, including log aggregation, data pipelines, event-driven architectures, and real-time analytics. It has a large and active community of users and contributors, and its development is supported by the Apache Software Foundation.