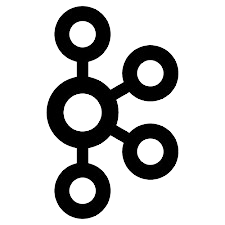
APACHE KAFKA



**TABLE OF CONTAIN:**

[1. USER GUIDE: 1](#_Toc98420642)

[1.1 INTRODUCTION: 1](#_Toc2047292655)

[1.2 PRE-REQUISITES: 2](#_Toc2116114375)

[1.3 RUNNING AND MONITORING THE JOB: 4](#_Toc902928884)

[1.3.1 WITHOUT USING AIRBYTE: 4](#_Toc1496330837)

[1.3.2 USING AIRBYTE: 6](#_Toc1217753898)

[2. TECHNICAL DOCUMENTATION: 8](#_Toc452573630)

[2.1 KAFKA: 9](#_Toc968237975)

[2.2 Data flow: 11](#_Toc1534485078)

# USER GUIDE:

## INTRODUCTION:

This document outlines the Kafka streaming job, which captures real-time changes from the MySQL binlog and streams them to Kafka topics. The job ensures seamless integration between a MySQL database and a Kafka streaming platform, providing real-time data processing with minimal latency and high reliability.

The key features of this job include:

* **Real-time Change Data Capture**: Captures MySQL binlog events and streams them to Kafka in real-time.
* **Schema Change Handling**: Automatically adapts to changes in the database schema without causing data loss.
* **Duplicate Handling**: Ensures that duplicate data is properly managed and only unique records are processed.
* **Checkpointing and Fault Tolerance**: Implements checkpointing to allow recovery from a specific binlog position in case of failure.
* **Streaming to Kafka Topics**: Configures data to be streamed to appropriate Kafka topics based on predefined criteria.

This guide is intended to help for managing data pipelines involving MySQL and Kafka. It covers the configuration and monitoring of the job, ensuring it runs efficiently and meets business requirements.

In the following sections, you will find detailed instructions on how to set up, configure, and monitor the Kafka streaming job.

## PRE-REQUISITES:

Before setting up and running the Kafka streaming job, ensure that the following requirements are met:

1. **Software Requirements:**

* **MySQL**: Version 5.7 or higher (above 8 is recommended), with binlog enabled.Binlog format should be mixed-based if you want to capture schema level changes (like; add,delete,modify columns) if not row-based can capture row level changes. (<https://dev.mysql.com/downloads/installer/>)
* **Kafka/confluent-kafka**: Version 2.5 or higher (latest one is recommended), properly configured and running.(<https://kafka.apache.org/downloads>) (<https://docs.confluent.io/platform/current/installation/installing_cp/zip-tar.html>)
* **Zookeeper**: Required by Kafka for coordination (no need to download and instillation cause it comes with kafka)
* **Java**: JDK 1.8 or higher installed (for Kafka) (java 11 is recommended). ([https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html)](https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html)
* **Python**: I am gonna use python install python (python 3 is recommended) also install python library to connect with mysql and kafka; confluent\_kafka, pymysqlreplication, pymysql (<https://www.python.org/downloads/>) (pip install PyMySQL, pip install pymysqlreplication, pip install confluent-kafka)
* **Docker** (optional): For running Kafka, Zookeeper, and MySQL in containers and airbyte too.

1. **Permissions and Access:**

* **MySQL User**: A MySQL user with the necessary permissions for reading binlog data. Ensure the user has the following privileges:
* -REPLICATION SLAVE
* -REPLICATION CLIENT
* -SELECT
* **Kafka Access**: Ensure you have access to Kafka brokers and the necessary topics for streaming.
* **Network Access**: Ensure that the MySQL server, Kafka brokers, and the machine running the Kafka Connect worker can communicate over the network.

**System Requirements:**

* **Memory**: At least 4 GB of RAM for running Kafka, MySQL, and the Kafka Connect worker.
* **Disk Space**: Ensure there’s enough disk space for Kafka and MySQL logs (especially binlogs).
* **CPU**: A multi-core processor is recommended for handling large-scale real-time data processing.

#### **iv.**Environment Setup:****

* **Kafka Cluster**: A properly configured and running Kafka cluster with brokers, topics, and Zookeeper. (setup >> <https://www.youtube.com/watch?v=BkL5blN057M&list=PLxv3SnR5bZE82Cv4wozg2uZvaOlDEbO67&index=3> and <https://www.youtube.com/watch?v=l6Ta75ZSXi4&list=PLxv3SnR5bZE82Cv4wozg2uZvaOlDEbO67&index=5>)
* **Binlog Configuration**: MySQL binlog must be enabled with proper configuration (binlog\_format set to MIXED/ROW).

## RUNNING AND MONITORING THE JOB:

### 1.3.1 WITHOUT USING AIRBYTE:

1. **Running zookeeper server:**

To run zookeeper server we need to open terminal and direct where the kafka is,By running following command we can start zookeeper-server.

For windows:

Screenshot from 2024-10-03 13-47-38

For linux:

Screenshot from 2024-10-03 13-47-22

For zookeeper server to start we nees two file ‘zookeeper-server-start’ and ‘zookeeper.properties’. The server-start file is; generally it can be found in ‘./bin’ and ‘zookeeper-server-start.bat’ or ‘zookeeper-server-start.sh’ or with out (‘.bat’ or ‘.sh’) just check the actual file name

and properties file can be found in ‘./etc/kafka’ if you use confluent\_kafka and if you use kafka then can be found in ‘./etc’

Name of the file should be checked some times it is just present as ‘zookeeper’ or ‘zookeeper.properties’.

1. **Running kafka server:**

Likewise we open another terminal and direct where the kafka is and run the following command;

For windows:

Screenshot from 2024-10-03 13-47-58

For linux:

Screenshot from 2024-10-03 13-47-46

For basics of kafka like; topic creation, producing messages, consuming messages you can check it out >> (<https://www.youtube.com/watch?v=pj6VoHAAXGQ&list=PLxv3SnR5bZE82Cv4wozg2uZvaOlDEbO67&index=4>)

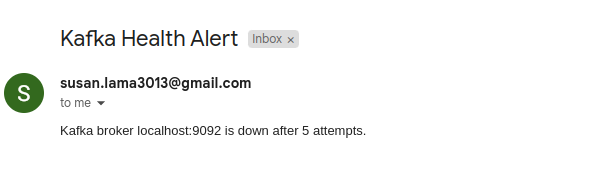
And you can clone my github repository and there I setup two types of environments;

1. Single-node kafka and zookeeper-server
2. Three-node kafka and Three-node zookeeper-servers

If you satisfy the pre-requisites the python program will run, you just need to slight adjustments which as mentioned in program.

demo vedio link >> <https://youtu.be/T2taO5jspZ8>

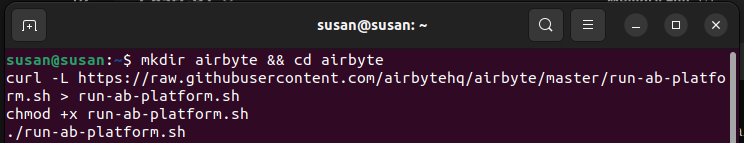
For monitoring the code/job compile a cdc.log file, we can monitor from there and if any errors like servers go down, the job gonna alert with email.



### 1.3.2 USING AIRBYTE:

**i. Configuration:**

First up all configure airbyte either using docker or without docker,



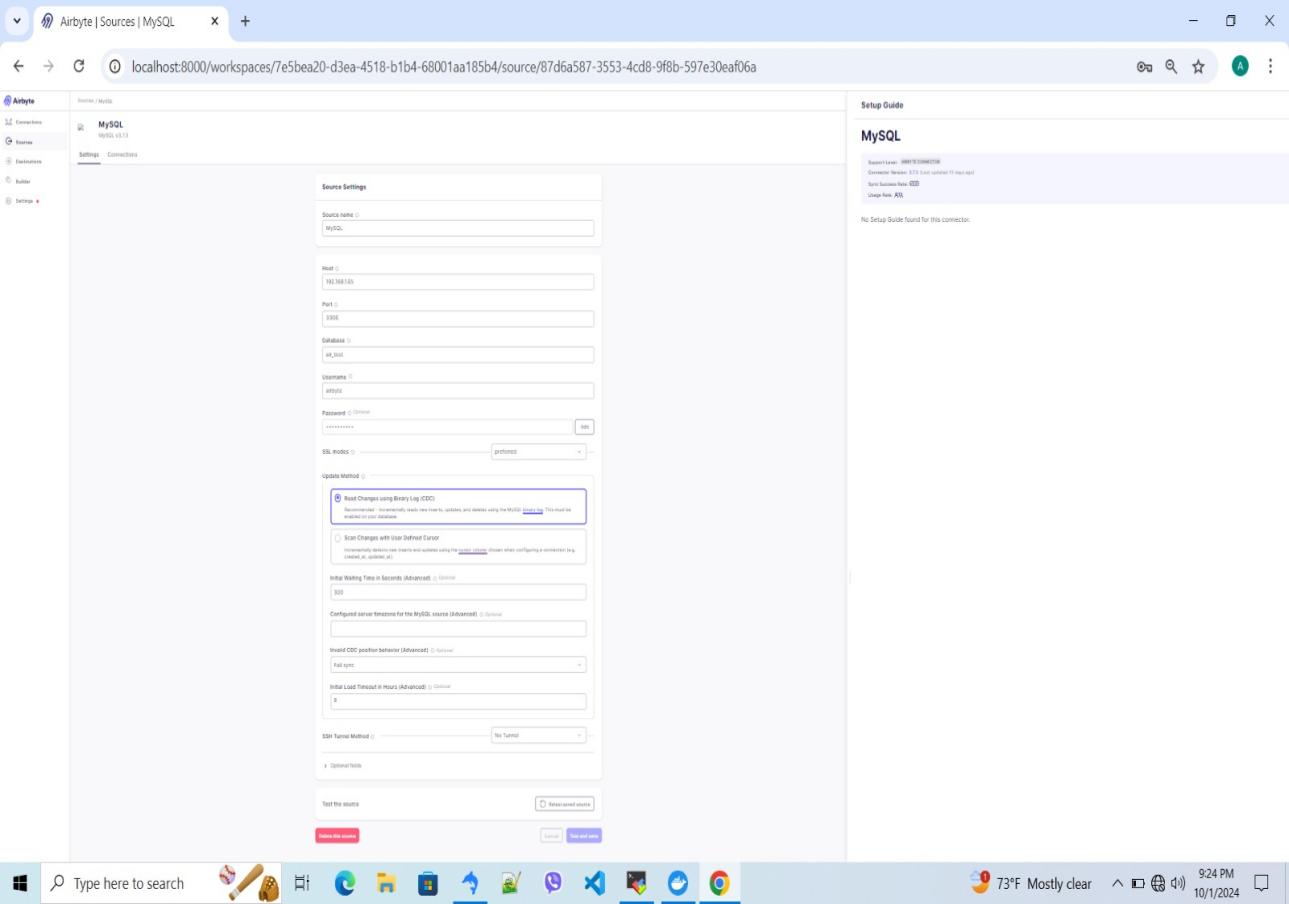
This commands gonna import airbyte necessary files (images) of airbyte and you need to compose up ‘docker-compose.yml’ file.

refer >><https://www.youtube.com/watch?v=oeshl0H1JcU&t=170s>

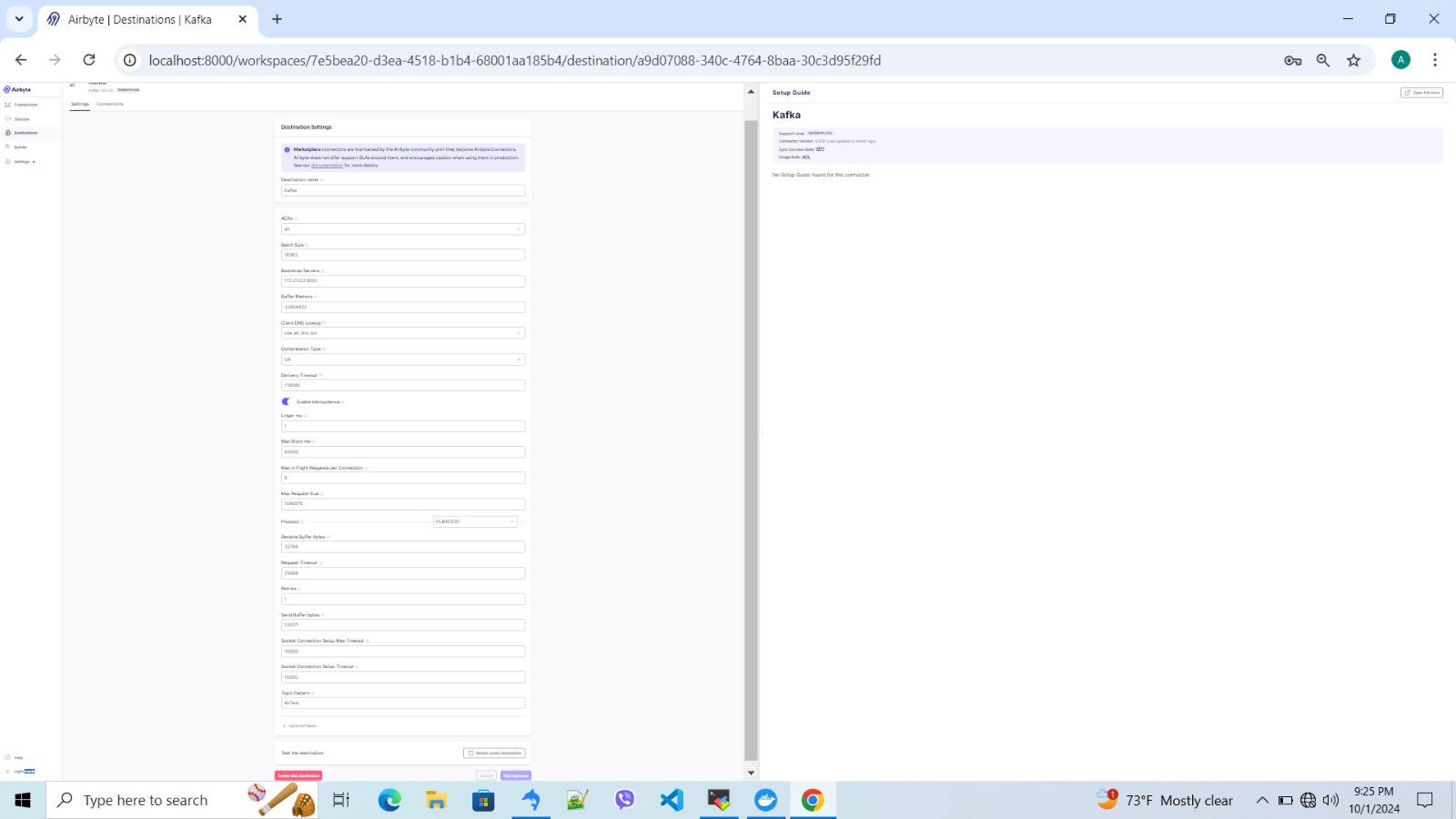
1. **Running:**

refer >> <https://airbyte.com/how-to-sync/mysql-to-kafka>

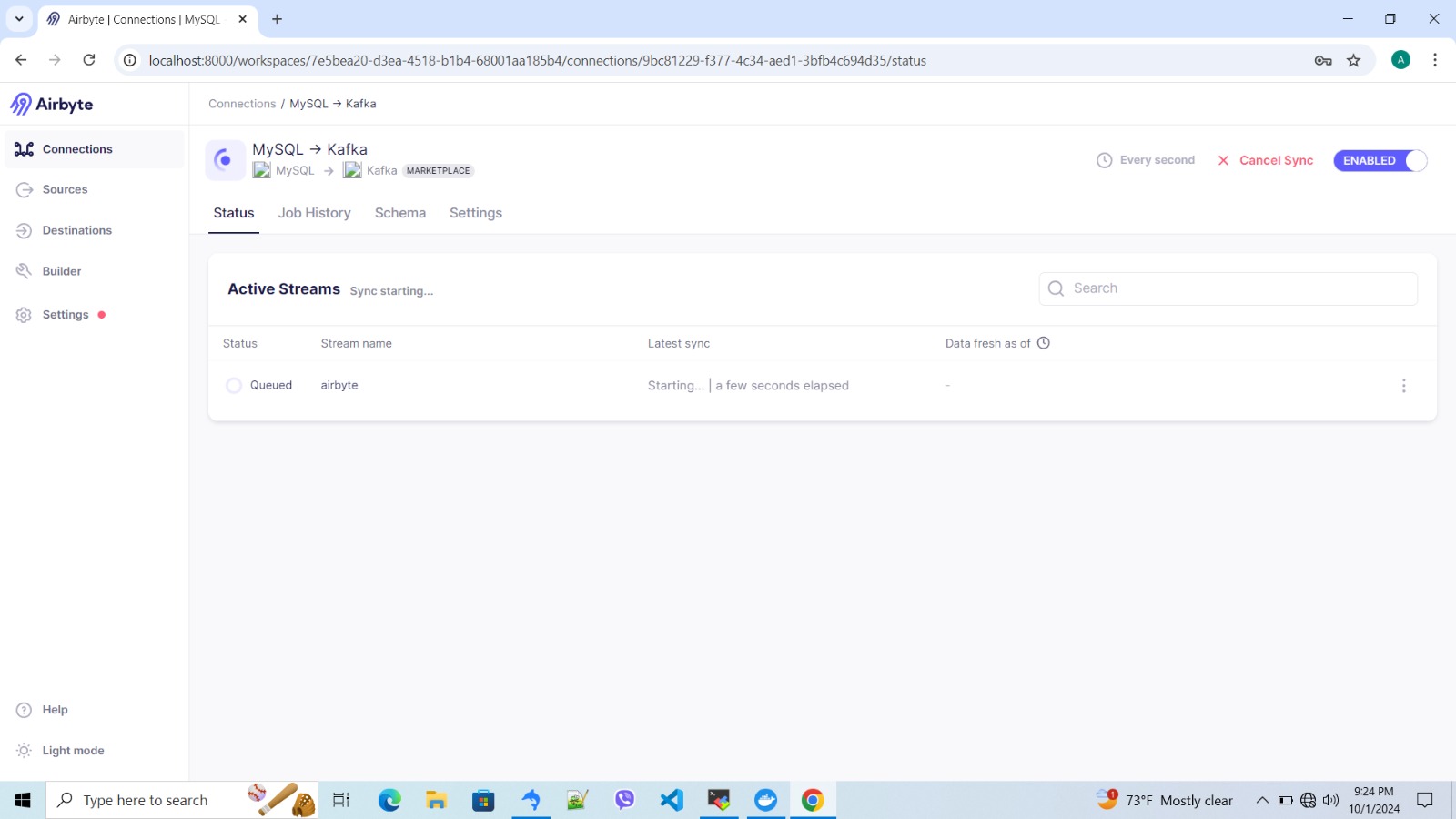
Fisrt up all setup mysql source:



Setup kafka destination:



Make a connection:



demo vedio link >> <https://www.youtube.com/watch?v=nulPXuRvjZU>

# TECHNICAL DOCUMENTATION:

This project aims to build a real-time data streaming pipeline that captures changes from a MySQL database and streams them into Kafka topics.

**key components:**

* **MySQL**: The primary database where data changes occur.
* **Airbyte**: An integration platform that captures MySQL binlog changes and pushes them to Kafka.
* **Kafka**: A distributed messaging system where data from MySQL is streamed to various topics, to be consumed by other systems or services.
* **Docker (optional)**: The container platform used to run the components in isolated environments.

db

KAFKA

server

MySQL

db

app

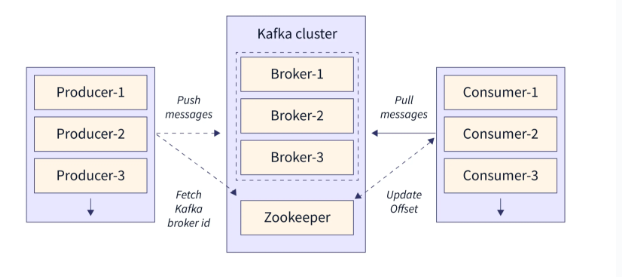
db

fig: Architecture of Mysql Cdc and passing to Kafka

## 2.1 KAFKA:

Apache Kafka is an open-source distributed event streaming platform that is used for building real-time data pipelines and streaming applications. Kafka enables the transfer of large amounts of data in real-time, providing both scalability and fault tolerance.

KAFKA ARCHITECTURE:



1. broker/ brokers(cluster):

* A Kafka **broker** is a server that runs Kafka and handles the exchange of messages between producers and consumers.

1. kafka topics:

* **Topics** are fundamental to Kafka's architecture. They are used to categorize and organize the data being streamed. Each topic is a **log of events** to which producers write and from which consumers read.

1. topic partition:

* Topics are divided into partitions, which allow Kafka to scale horizontally. Each partition is ordered, and consumers read from it sequentially.

1. producers:

* Kafka **producers** are applications or services that publish messages to Kafka topics. In our case, **Airbyte/python client** acts as the producer, publishing changes from the MySQL database into Kafka topics.

1. consumers:

* Kafka **consumers** are applications or services that subscribe to topics to read and process data. Each consumer reads data from one or more topics.

1. zookeeper:

* Zookeeper manages Kafka’s metadata, tracks broker health, and coordinates leader elections for partitions. It ensures high availability, fault tolerance, and seamless broker failover in Kafka clusters.

## 2.2 Data flow:

start

Y

N

alert email

errors/ exceptions

stop

pass to kafka

server active?

binlog capturing