**Real Time Event Streaming Pipeline**

**Technologies:**

* Kafka
* Zoo Keeper
* C# / .Net
* MySQL
* Dockers Containers
* Java EE
* RestAPI
* Nodejs
* Bootstrap
* D3.js Visualization

**Introduction**

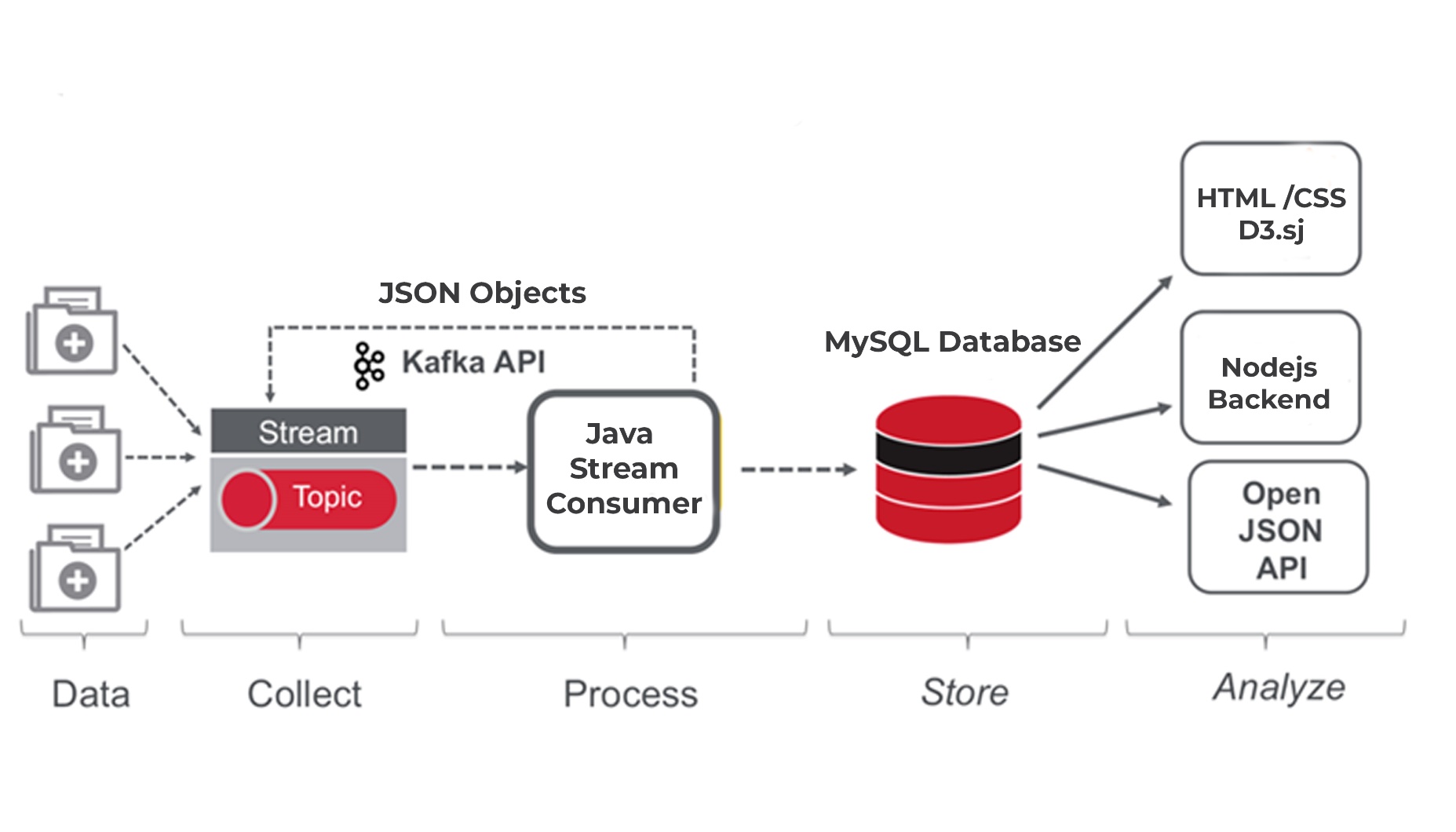
Apache Kafka is a distributed publish-subscribe messaging system and a robust queue that can handle a high volume of data and enables you to pass messages from one end-point to another. Kafka is suitable for both offline and online message consumption. Kafka messages are persisted on the disk and replicated within the cluster to prevent data loss. Kafka is built on top of the ZooKeeper synchronization service. It integrates very well with Apache Storm and Spark for real-time streaming data analysis.

Benefits

Following are a few benefits of Kafka −

* **Reliability** − Kafka is distributed, partitioned, replicated and fault tolerance.
* **Scalability** − Kafka messaging system scales easily without down time..
* **Durability** − Kafka uses Distributed commit log which means messages persists on disk as fast as possible, hence it is durable..
* **Performance** − Kafka has high throughput for both publishing and subscribing messages. It maintains stable performance even many TB of messages are stored.

Kafka is very fast and guarantees zero downtime and zero data loss.

**Solution Architecture:**

**Working:**

I have created a Rest API end point which listens to the events fired at any time and pushes events to Apache Kafka – Apache. Kafka is an open-source stream-processing software platform developed by LinkedIn and donated to the Apache Software Foundation, written in Scala and Java. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. Kafka uses Publisher and Subscriber based system in which Publisher Publishes messages to topics and Subscriber subscribe to the topic and can receive those messages.

1. **Producer (Nodejs) :** Receives the requests and pushes it to the queue with a specific topic. In this project Producer is a NodeJs RestAPI which listens to the events and pushes them to KAFKA
2. **Consumer (Java EE ) :** Consumer in this project is Java Based Application which subscribe to the topics and picks up events one by one from the queue and does the processing on those events to generate reports.
3. **Front-End Application (HTML/CSS) :** Front End Application takes the data from Consumer and displays in graphical format users.
4. **Simulator :** I have also created a .Net Desktop application to simulate the event fire so that we can send the events one by one to producer.

**Pre-Requisites:**

* Docker Engine must be installed on the computer.
* Node V. 10.16.0
* Java 1.8 or higher
* .Net framework 4.5 (Optional)

**Installation Steps:**

1. Run Kafka Docker.bat – It will start Apache Kafka, Zookeper and KSQL
2. Run SQL Docker.bat – It will start MySQL Docker and will create the database automatically
3. Run Create Topics.bat – It will create all the necessary topics for Apache kafka for the events
4. Run Producer – It will start the producer for the Apache Kafka
5. Run Consumer – It will start the Java Based Consumer for Apache Kafka
6. Go to \Even\_Fire\_DOTNET\Even\_Fire\bin\Debug Run Event Fire – It will simulate the event firing to producer by reading from the file
7. Go to <http://localhost:4000/> and you following web based (screenshot below) front end will show up where you can see the reports.
8. Please go to watch a live working Demo <https://youtu.be/WCJk0pPsuSg> on Youtube (Must)

