YCompany - eClaim System’s Message Stream Provider

DAR Document



Nagarro Software Pvt. Ltd.

Sanjeet Chauhan

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| Version | Date | Author/Contributor | Comments |
| 1.0 | 28/06/2019 | Sanjeet Chauhan | Initial Version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[1 Introduction 4](#_Toc15751960)

[1.1 Objective and scope of document 4](#_Toc15751961)

[2 Requirements at a Glance 5](#_Toc15751962)

[3 Available tools 6](#_Toc15751963)

[3.1 RabbitMQ 6](#_Toc15751964)

[3.1.1 Features 6](#_Toc15751965)

[3.2 Apache Kafka 6](#_Toc15751966)

[3.2.1 Features 6](#_Toc15751967)

[4 Comparison Analysis 7](#_Toc15751968)

[4.1 Comparison 7](#_Toc15751969)

[5 Recommendation 7](#_Toc15751970)

[6 Assumptions 8](#_Toc15751971)

[7 Risks 9](#_Toc15751972)

[8 Appendix 10](#_Toc15751973)

[8.1 References 10](#_Toc15751974)

# Introduction

As per requirement of the YComapany – eClaim design, it is required to choose a message streaming provider that provides the desired level of performance, efficiency ans scalability.

This document is for analyzing two most popular message stream provider available in the marketplace and providing a detailed comparison of the available option. Comparison is made based on intended features using a point-based scoring system to identify the best fit.

## Objective and scope of document

The objective of system is to identify all the features required for a database engine to qualify for fittment to the system. Futher, not all features are equally important, some carry a higher weightage other may carry lower weightage.

Also, there are number of stream provider available in the market place but not all available options are compared in this document. Only the databases that are relied & trusted worldwide are cosidered for detailed comparision.

# Requirements at a Glance

The system require a message streaming provider that provides easy integration, massive scale and great performance. The business operation involves lot of data flow and transactions over medium and complex business logic accorss whole system hence it is equally important to log alost each and every activity and errors and exception throught out the system. Logging on such massive scale require a robust message stream provider to deal with thousands of thousands messages in few seconds or minutes. Logged data further can be used to diagnose erros or exception in detail also it will open a scope to monitor and anlyze data by other data analytical tool.

Comparison of the available message stream provider will be done on following major features:

* Performance
* Ease of development & integration
* Security & Compliance
* Replication
* Support & Online Help
* Manageability
* Scalability

# Available tools

There are many message stream provider available in the marketplace being used for enterprises application. Most common are : RabbitMQ, Apache Kafka, ZeroMQ, Apache ActiveMQ, IBM MQ etc.

We are considering RabbitMQ and Apache Kafka for comparison as these are the most trusted, most widely used and can be integrated easily in the proposed system.

## RabbitMQ

RabbitMQ is an open-source message-broker software (sometimes called message-oriented middleware) that originally implemented the Advanced Message Queuing Protocol (AMQP) and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol (STOMP), Message Queuing Telemetry Transport (MQTT), and other protocols.

### Features

* Messages are routed through exchanges before arriving at queues.
* Several RabbitMQ servers on a local network can be clustered together, forming a single logical broker.
* Queues can be mirrored across several machines in a cluster, ensuring that even in the event of hardware failure your messages are safe.
* RabbitMQ clients for almost any language.
* RabbitMQ ships with an easy-to use management UI that allows you to monitor and control every aspect of your message broker.
* RabbitMQ offers tracing in case messaging system is misbehaving.

## 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. Its storage layer is essentially a "massively scalable pub/sub message queue designed as a distributed transaction log,"[3] making it highly valuable for enterprise infrastructures to process streaming data. Kafka can also connect to external systems (for data import/export) via Kafka Connect and provides Kafka Streams.

### Features

* Apache Kafka can handle scalability and do it without downtime.
* Kafka can work with a huge volume of data streams, easily.
* Kafka clusters can handle failures with the masters and databases.
* it is very reliable as it is distributed, partitioned and fault tolerant.
* It is durable because Kafka uses Distributed commit logs, which means messages persist on disk as fast as possible.
* For both publishing and subscribing messages, Kafka has high throughput. Even if many TBs of messages are stored, it maintains stable performance.
* Kafka is very fast and guarantees zero downtime and zero data loss.
* There are many ways by which applications can plug in and make use of Kafka. In addition, Kafka offers ways to write new connectors as needed.

# Comparison Analysis

Both providers will be compared on various features such as performance, storage, scalability and many more listed in the compare matrix below.

## Comparison

Feature based relative scoring of all the three database engines is mentioned below

|  |  |  |
| --- | --- | --- |
| Feature | Apache Kafka | RabbitMQ |
| Enterprise support | **No** | **Yes** |
| Documentation | **Limited** | **Yes** |
| Benchmark | **100k/sec** | **20k/sec** |
| Storage | **Yes** | **No** |
| Order storage delivery | **Yes** | **No** |
| Queue content persistence | **Yes** | **No** |
| Message deletion | **After exceeding size** | **Immediately after consumtion** |
| Queue data compression | **Yes** | **No** |
| Available client in | **Approx 10 client** | **Approx 13 client** |
| Different consumers on same data set | **Yes** | **Yes** |
| Conditional message routing | **Limited** | **Yes** |
| SSL support | **No** | **Yes** |
| Message replay | **Yes** | **No** |
| Message rejection | **No** | **Yes** |
| Load balancing | **Yes** | **Yes** |

# Recommendation

Based on the comparision above, it is recommended that Apache Kafka be used as Message stream provider as it gives some really needed feature with no price like 100k/sec storage capability , message replay etc.

# Assumptions

1. It is assumed that data logged via message streaming can also be taken to analyse later hence it was considered that a high scale and high performance stream provider is needed.

# Risks

* From management perspective, the current design relies on cloud services for various IT operations, so minor risk is involved if migration to on premises design is considered. All these tasks will be handled by the on premises team.

# Appendix

## References

1. <https://content.pivotal.io/blog/understanding-when-to-use-rabbitmq-or-apache-kafka>
2. <http://kth.diva-portal.org/smash/get/diva2:813137/FULLTEXT01.pdf>