**Message Broker**

Contents

[1 Document Control 3](#_Toc98318692)

[1.1 Change Record 3](#_Toc98318693)

[1.2 Reviewer 3](#_Toc98318694)

[1.3 Approver 3](#_Toc98318695)

[2 Document Purpose 3](#_Toc98318696)

[3 Message Broker 3](#_Toc98318697)

[3.1 RabbitMQ 3](#_Toc98318698)

[3.1.1 Features of RabbitMQ 4](#_Toc98318699)

[3.1.2 Who uses RabbitMQ 4](#_Toc98318700)

[3.2 Apache Kafka 4](#_Toc98318701)

[3.2.1 Features of Kafka 5](#_Toc98318702)

[3.2.2 Who uses Kafka 5](#_Toc98318703)

[3.3 Amazon Kinesis 5](#_Toc98318704)

[3.3.1 Features of Kinesis 6](#_Toc98318705)

[3.3.2 Who uses Kinesis 6](#_Toc98318706)

[3.4 Azure Service Bus 6](#_Toc98318707)

[3.4.1 Features of Service Bus 6](#_Toc98318708)

[4 Comparison 6](#_Toc98318709)

[5 Appendix 9](#_Toc98318710)

# Document Control

## Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** | **Change reference** |
| 03/15/2022 | Prajeesh T S | 1.1 | Initial version |

## Reviewer

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Approval/Review Date** |
| **Anoop Jose** | Staff Software Architect |  |

## Approver

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Approval/Review Date** |
| **Anoop Jose** | Staff Software Architect |  |

# Document Purpose

This document provides details regarding details of available message brokers and the comparison

# Message Broker

## RabbitMQ

*RabbitMQ is the most widely deployed open-source message broker. RabbitMQ is used worldwide at small startups and large enterprises. RabbitMQ is lightweight and easy to deploy on-premises and in the cloud.*

RabbitMQ is a distributed broker since it executes as a cluster of nodes in which the queues are distributed as against the nodes, for fault tolerance and high-end availability. It is based on a push model that is considered best for low latency messaging and queue-based architecture.

### Features of RabbitMQ

* Facilitates users to set rules for message delivery
* Offers multiprotocol support
* Flexible, secure, supports synchronous and asynchronous communication
* Large community support, multiple resources, and articles
* Suitable for many languages and technologies like Java, .NET, Ruby, NodeJS, external plug-ins, etc.
* Can be leveraged on various OS and cloud-based infrastructure
* Robust messaging, clustering, easy to use
* Scalability to encompass thousands of messages per second

### Who uses RabbitMQ

* Reddit
* Citizens Bank
* Stack
* USAA
* Alibaba Travels
* General Motors
* Trivago
* Palo Alto Networks
* Robinhood
* Freedom Financial
* Accenture

## Apache Kafka

Apache Kafka *is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.*

Kafka is competent to enable applications for processing, persisting, and reprocessing streamed data. It makes use of a routing key for sending messages through a direct routing approach. It is known as a distributed event streaming platform.

It offers a great degree of availability and can be clustered across multiple servers. In Kafka, the producer is responsible for writing and storing messages in the cluster whereas the consumer is responsible for consuming messages from the cluster. Kafka works on a pull-based approach in which users can ask for message matches from specified offsets.

##### 

### Features of Kafka

* The compatible, durable, simple routing approach
* Optimized for high throughput ingestion data stream
* Permits users to batch and replay messages
* Easily scales horizontally and lets many consumers handle it simultaneously
* Inbuilt security features with client authentication and controlled access
* Active and large community support with multiple documentation
* Easy to learn, scalable fault-tolerant, and accurate
* Best for real-time processing of data
* Supports multitenancy and ideal for Big Data projects

### Who uses Kafka

* LinkedIn
* Netflix
* Twitter
* Coursera
* PayPal
* Cisco
* Tumblr
* Spotify
* Etsy
* Foursquare
* Uber
* Mozilla
* Airbnb
* Cloudflare
* Oracle

## Amazon Kinesis

Kinesis is a cloud based real-time processing service. Kinesis producers can push data as soon as it is created to the stream. Kinesis breaks the stream across shards (similar to partitions), determined by your partition key. Each shard has a hard limit on the number of transactions and data volume per second. If you exceed this limit, you need to increase your number of shares. Much of the maintenance and configuration is hidden from the user. AWS allows ease of scaling with users only paying for what they use.

### Features of Kinesis

* It helps in streaming from Millions of Devices
* It doesn't require Server Administration.
* It is Cost Efficient.
* Ease of Use.
* High Scalability.

### Who uses Kinesis

* Netflix
* Veritone

## Azure Service Bus

Azure Service Bus is a messaging service on cloud used to connect any applications, devices, and services running in the cloud to any other applications or services. As a result, it acts as a messaging backbone for applications available in the cloud or across any devices.

### Features of Service Bus

* Message sessions
  + To create a first-in, first-out (FIFO) guarantee in Service Bus, use sessions. Message sessions enable exclusive, ordered handling of unbounded sequences of related messages. To allow for handling sessions in high-scale, high-availability systems, the session feature also allows for storing session state, which allows sessions to safely move between handlers
* Dead-letter queue
* Scheduled delivery
* Autodelete on idle
* Duplicate detection

# Comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Apache Kafka** | **Rabbit MQ** | **Amazon Kinesis** | **Azure Service Bus** |
| Licensing | Open Source | Open Source | Commercial | Commercial |
| Installation | Can be installed and run in local machine | Can be installed and run in local machine | Cloud Service | Cloud Service |
| Security | Kafka supports client-side security features like:  Encrypt data-in-transit between your applications and Kafka brokers.  Client authentication.  Client authorization. | RabbitMQ does not encrypt data at rest. Use a filesystem that offers encryption. | For data security, you can use server-side encryption with AWS KMS master keys to encrypt data stored in your data stream. | All traffic to the service can be routed through that private endpoint, so no gateways, NAT devices, ExpressRoute or VPN connections, or public IP addresses are needed. |
| Administration Complexity | Medium | Medium | Low [SaaS] | Low [SaaS] |
| Supported Languages | C++ Go Java .NET Python Scala and more | Java .NET C++ Ruby Python PHP JavaScript Scala and more | Java .NET Python Ruby Node.js | .NET Java |
| Message Order | Provides messages ordering because of its partitions. Messages are sent to topic by message key | Not supported | Not Supported | Yes |
| Message Lifetime | Kafka is a log which means the messages are always there, we can specify message retention policy. | It’s a queue messages are done away once consumed. | Stores data for 24 hours by default which can be increase up to 7 days by changing configuration. | Time TO Live property can be configured to set the message life time |
| Delivery Guaranty | at-least-once, at-most-once, exactly-once | at-least-once, at-most-once | at-least-once | at-least-once, at-most-once |
| Message Priority | A message cannot be sent with a priority level, nor be delivered in priority order.  All messages are stored and delivered in the order in which they are received regardless of how busy the consumer side is | We can specify message priority and messages are consumed based on priority | Not Available | A low priority queue and a high priority queue could be created where the high priority items are worked before the low priority queue items. |
| Performance | Kafka leverages sequential disk I/O operations and thus demands less hardware.  Offers much higher performance. It uses sequential disk I/O to boost performance | RabbitMQ controls its messages almost in-memory, using a big cluster (30+ nodes)  Process a million messages per second but requires more resources (around 30 nodes) | Good | Good |
| Support for Transactions | Kafka supports transactions | RabbitMQ does not support transactions |  | A transaction groups two or more operations together into an execution scope. Service Bus allows you to group operations against multiple messaging entities within the scope of a single transaction. A message entity can be a queue, topic |
| Scale | Can send up to a million’s messages per second. | Based on configuration and resources. | Auto Scale | Auto Scale |
| Use Case | Best fit in event driven development where data must flow between multiple components in the application. Using this distributed message  bus model gives a great deal of scalability.  Large amounts of data. | Best fit in job shaped applications,  Complex routing |  |  |
|  |  |  |  |  |

#### 

# Appendix

* [Azure Service Bus messaging - advanced features - Azure Service Bus | Microsoft Docs](https://docs.microsoft.com/en-us/azure/service-bus-messaging/advanced-features-overview#:~:text=%20Azure%20Service%20Bus%20-%20advanced%20features%20,processing%2C%20setting%20a%20time%20when%20the...%20More%20)
* [Message Brokers Comparison (timebase.info)](https://timebase.info/why-timebase/message-brokers-comparison.html)
* [Message Broker Comparison | RabbitMQ, Kafka, ActiveMQ, Kestrel (thestories.net)](http://www.thestories.net/message-broker-comparison-rabbitmq-kafka-activemq-kestrel/)