RabbitMQ and Kafka

Prabhleen Bagri and Lindsey Amaro
Mentor: David Fox

August 11, 2022

LLNL-PRES-838831

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Team Members

- Prabhleen Bagri
- San Jose State University
- Computer Science
- Expected Grad: May 2023

- Lindsey Amaro
- Cal Poly – San Luis Obispo
- Mathematics
- Expected Grad: June 2024

What is a Message Broker?

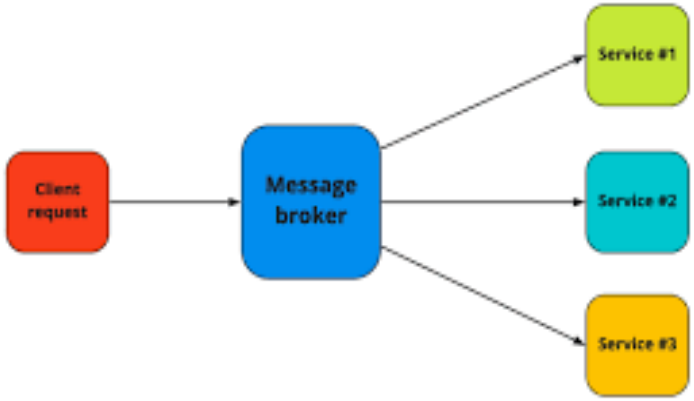


Diagram source: https://tsh.io/blog/message-broker/

Normally we would use TCP to send messages, but there are drawbacks

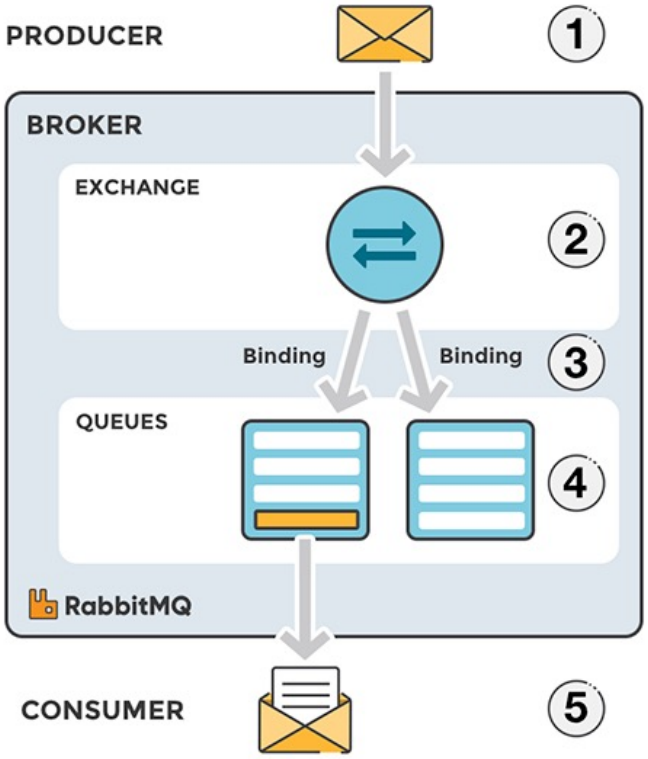
→ **SOLUTION : message brokers**

Message brokers sit in between two services that interact. They enable applications and systems to communicate with each other and exchange information.

- Acts as a buffer i.e. it holds messages until it is ready to be received
- Allows sender to issue message without knowing where receiver is
- Improved system performance because it allows asynchronous processing
- facilitates decoupling

RabbitMQ

- Message Broker
- Producer sends messages to exchanges, which then route those messages to queues that consumers can access
- Several Types of Exchanges support different routing patterns:
  - Direct
  - Fanout
  - Topic
- Messages deleted from queues once consumed
- Use Cases:
  - Situations that require complex sending patterns
  - Instances where quick message response is important
  - Applications that need to work with messaging protocols such as AMQP
- Companies that use RabbitMQ
  - T-Mobile
  - Reddit
  - Trivago



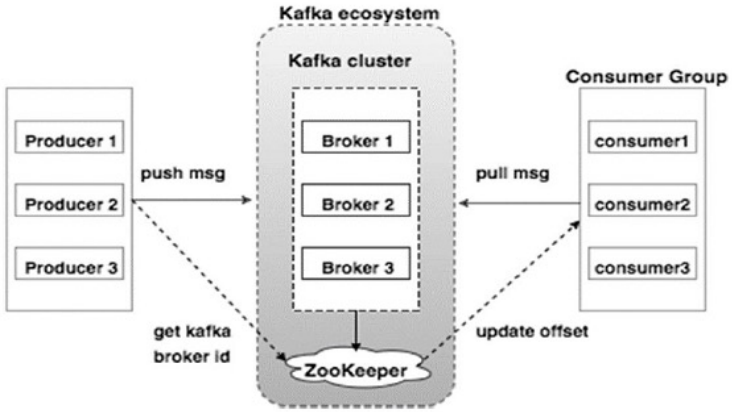
Picture Source: https://www.cloudamqp.com/blog/part1-rabbitmq-for-beginners-what-is-rabbitmq.html

Kafka

- **A distributed event store and streaming platform**
- Has servers and clients that communicate over TCP Network Protocol
- Can be deployed on hardware, VMs, containers, cloud environments
- Used by over 80% of the fortune 100

Use cases:

- Decouple dependencies by streaming events
- Messaging
- Location & activity tracking
- Commit log
- Log aggregation



- Server: is run as a cluster of servers that can span multiple datacenters
- Highly scalable and fault-tolerant
- Client: allows you to write distributed applications and microservers that read, write, and process streams of data in parallel

diagram source: https://www.tutorialspoint.com/apache_kafka/apache_kafka_cluster_architecture.htm

Objectives

- Install and configure RabbitMQ and Kafka in student cluster environment
- Create basic sender and receiver Python applications for RabbitMQ and Kafka to ensure each service is working
- Explore HPC uses cases for both services

Image Source: https://kafka.apache.org/

Image Source: https://www.rabbitmq.com/

RabbitMQ Install and Configuration

RabbitMQ on RHEL Installation Steps:

1. Import necessary RPM's
2. Configure a Yum repository for RabbitMQ
3. Install the RabbitMQ server and its dependencies
4. Create a RabbitMQ user
5. Create a Virtual Host for the RabbitMQ user to connect to
6. Set permissions, so the user and virtual host can recognize each other

```
[root@xenon4 rabbit]# python3 send.py
 [x] Sent 'Hello World!'
```

```
[root@xenon5 rabbit]# python3 receive.py
 [*] Waiting for messages. To exit, press CTRL+C
 [x] Received 'Hello World!'
```

Kafka Install and Configuration

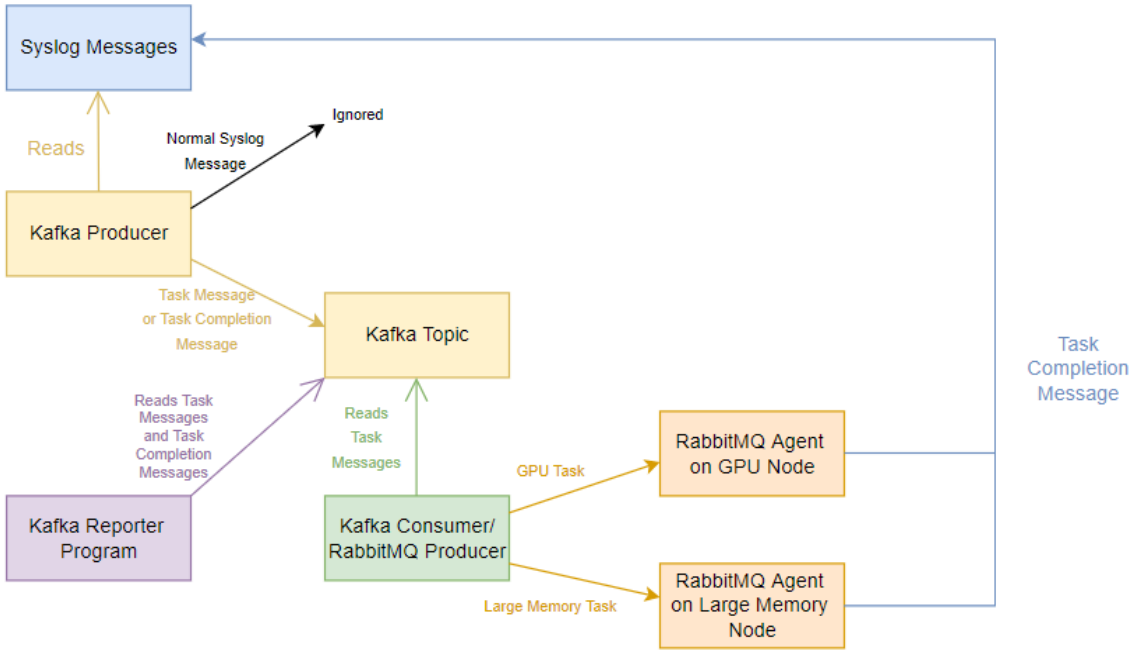
```
[root@radon4 ~]# python3 producerApp.py
sending message...
message sent successfully...
```

```
received message =  ConsumerRecord(topic='test-topic'
partition=0, offset=3, timestamp=1660178191347, times
mp_type=0, key=None, value={'name': 'abc', 'email': '
c@example.com'}, headers=[], checksum=None, serialize
key_size=-1, serialized_value_size=43, serialized_hea
r_size=-1)
```

- Download latest Kafka release and extract it
- Create user to run Kafka and Zookeeper
- Start Kafka environment (create and start a service for both)
- To send messages, create producer and consumer programs on separate nodes using KafkaProducer / KafkaConsumer APIs

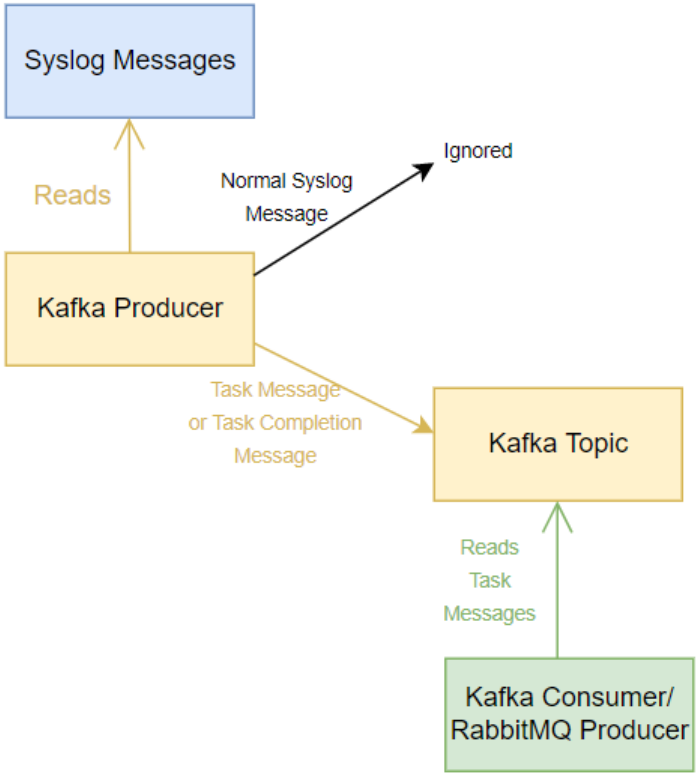
Example Use Case Model

- Besides sending messages, RabbitMQ and Kafka can also be used to send tasks
- These services could work together to send and record tasks being done
- **Objective**: Use Kafka and RabbitMQ to handle jobs issued by syslog

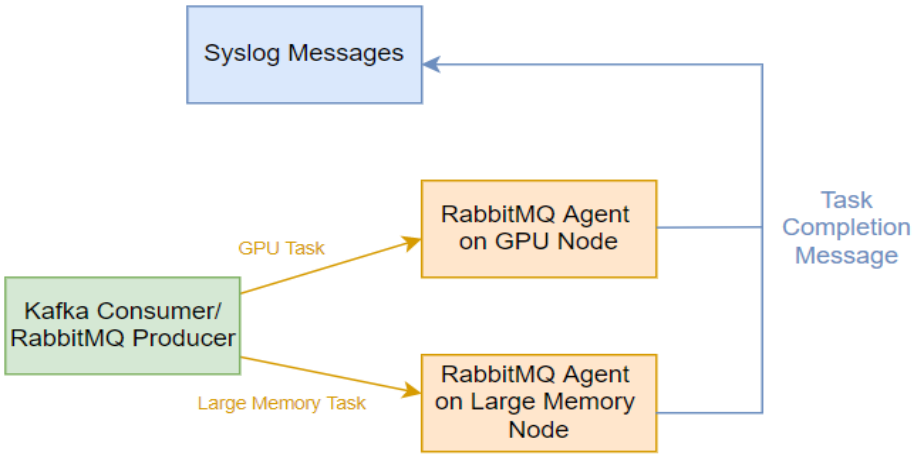


Transferring Task Messages

- Kafka Producer continually reads syslog messages
- Task message triggers Producer to publish to Kafka Topic
- Kafka Consumer reads from topic and passes task message to RabbitMQ Producer



Delegating and Recording Tasks



- RabbitMQ sends task to appropriate agent
- Task completion message sent to syslog
- Kafka Recorder program lists when tasks were issued and completed

Challenges

- Filtering Messages
  - No straightforward method for filtering data sent into the Kafka topic
- Incorporating Kafka in demo
  - Limited access to large data sets
  - Made finding use case difficult
- Connecting RabbitMQ and Kafka
  - Because of their similar functionality, finding a practical model using both services to fit one use case was a challenge

Future Goals

- Explore clustering
  - Scalability
  - High Availability
- Explore additional clients and usage models
  - Other client libraries with RabbitMQ and Kafka (C++, Java, etc.)
  - Advanced Configurations (integrating Celery with RabbitMQ, KafkaStreams with Kafka, etc.)
- Data Retention and Management, particularly with Kafka

References

- https://www.rabbitmq.com/
- https://www.cloudamqp.com/blog/part1-rabbitmq-for-beginners-what-is-rabbitmq.html
- https://www.upsolver.com/blog/kafka-versus-rabbitmq-architecture-performance-use-case#:~:text=Kafka%20offers%20much%20higher%20performance,for%20big%20data%20use%20cases
- https://kafka.apache.org/
- https://betterprogramming.pub/why-do-we-need-message-broker-7382ce0e46c6
- https://www.ibm.com/cloud/learn/message-brokers