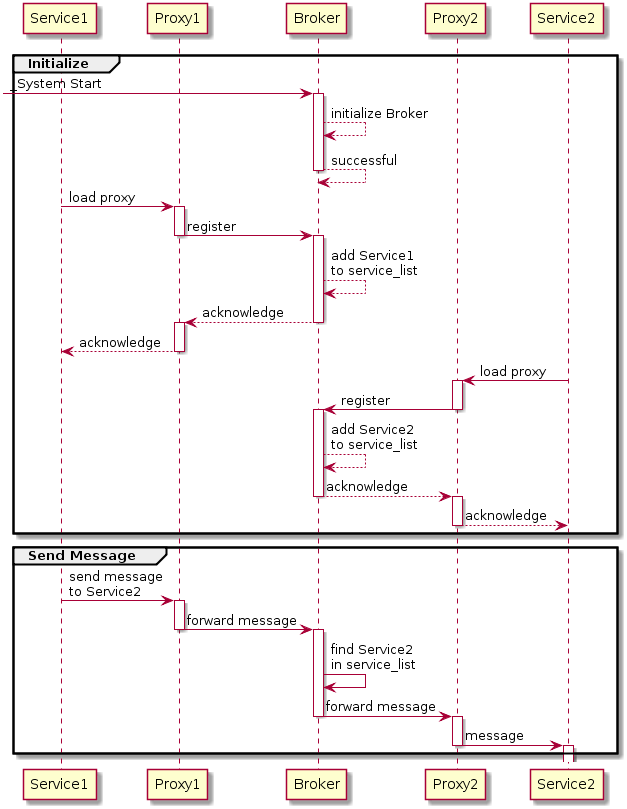
An introduction

**What is a Message Broker?**

In its core, a message broker is [“a program that translates a message to a formal messaging protocol of the sender, to the formal messaging protocol of the receiver”](https://en.wikipedia.org/wiki/Message_broker)



**So when do I need a Message Broker?**

This means that when you have a lot of messages (think thousands, millions, billions of messages) that it could be worth looking into a Message Broker to create a centralized store/processor for these messages, so that other applications or users can work with these messages — your single source of truth.

**Use Case #1**

Just think about when an enterprise acquires a startup. How would you integrate the applications that the startup wrote into your own enterprise? You could start by migrating the whole application towards your own database connections, or start porting it to another language. But this is most often a task of weeks / months. Rather than spending your time on doing this, it could be more beneficial to integrate the startup application in a centralized event store, so that you can adapt your other programs to work with these events. Events are a form of unified communication, so that you do not rely on the programming / scripting language behind it.

**Use Case #2**

What about when your organization wants to work with the new Internet of Things (IoT). Then you would have thousands of devices, but how do you manage the data of all those devices? Well this is where a Message Broker also excels in. Just send all your data of these devices towards your Message Broker, and it will take care of processing them.

**Which Message Brokers are out there to process my events?**

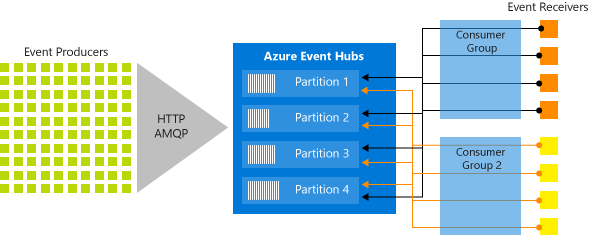
There are tons of message brokers out there (ActiveMQ, Kafka, RabbitMQ, OMS, JMS, Redis, Service Bus, …) however for me there are three popular ones (subjective).

**Azure Event Hub**

[What is Azure Event Hubs and why use it  
Overview and introduction to Azure Event Hubs - Cloud-scale telemetry ingestion from websites, apps, and devicesdocs.microsoft.com](https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-what-is-event-hubs)

Azure Event Hub allows you to set up a scalable Event Hub that suits your needs in a couple of seconds. It is a PaaS offering by Microsoft Azure, so that you do not need to manage it, but rather just consume it.

To provide full interoperability and binary compatibility across platforms, Event Hub uses the Advanced Message Queuing Protocol (AMQP) which is an open standard, downloadable by everyone: <https://www.iso.org/standard/64955.html>



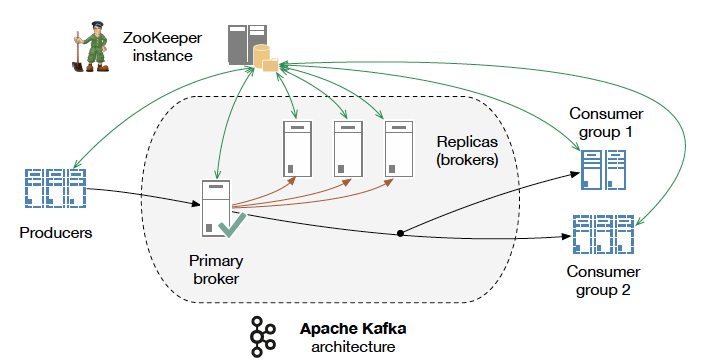
Azure Event Hub Architecture — <https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-features>

**Apache Kafka**

<https://kafka.apache.org/>

Apache Kafka is a Message Broker originally developed by LinkedIn and open sourced early 2011. It is just like Azure Event Hub a platform capable of handling millions of events.

The main difference between Apache Kafka and Event Hubs is that Apache Kafka is mostly installed through an IaaS offering rather than a PaaS offering. This trend however is changing and more and more Cloud Vendors are now offering PaaS versions of Kafka as well. Apache Kafka also has a big open source community behind it. <https://github.com/apache/kafka>

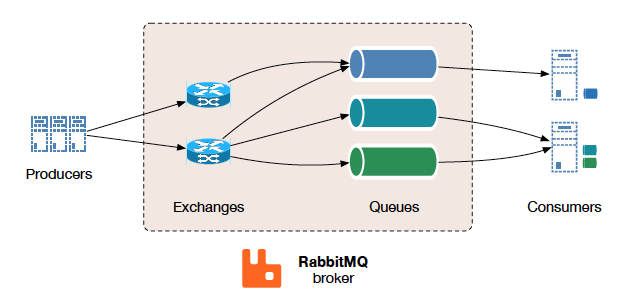


Apache Kafka Architecture — <http://kth.diva-portal.org/smash/get/diva2:813137/FULLTEXT01.pdf>

**RabbitMQ**

[RabbitMQ - Messaging that just works  
RabbitMQ is lightweight and easy to deploy on premises and in the cloud. It supports multiple messaging protocols…www.rabbitmq.com](https://www.rabbitmq.com/)

RabbitMQ was one of the first open source message brokers, developed to implement AMQP to work across different platforms and languages.



RabbitMQ Architecture — <http://kth.diva-portal.org/smash/get/diva2:813137/FULLTEXT01.pdf>

**Message Broker/Middleware**

According to Wikipedia:

*“Message-oriented middleware (MOM) is software or hardware infrastructure supporting sending and receiving messages between distributed systems. MOM allows application modules to be distributed over heterogeneous platforms and reduces the complexity of developing applications that span multiple operating systems and network protocols.”*

Message brokers do many things such as:

* Decouple message publisher and consumer
* Store the messages
* Routing of messages
* Monitoring and management of messages

Such message broker services typically used to lack standards, and existing commercial implementations had proprietary implementation and API issues to look at. This was up until recently an issue which can now be addressed through AMQP. AMQP is an open standard application layer protocol for message-oriented middleware.

From [the AMQP website](http://www.amqp.org/):

*“AMQP is an Open Standard for Messaging Middleware.*

*By complying to the AMQP standard, middleware products written for different platforms and in different languages can send messages to one another. AMQP addresses the problem of transporting value-bearing messages across and between organizations in a timely manner.*

*AMQP enables complete interoperability for messaging middleware; both the networking protocol and the semantics of broker services are defined in AMQP.”*

In null-set, AMQP defines:

* Where to send messages (Routing)
* How to get there (Delivery)
* What goes in must come out (Fidelity)

[AMQP](http://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol) is a standard  wire level protocol (communicating with a remote machine or getting data from point to point) and has many [implementations](http://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol#Implementations).

**RabbitMQ** is one such open source message broker software that implements AMQP.