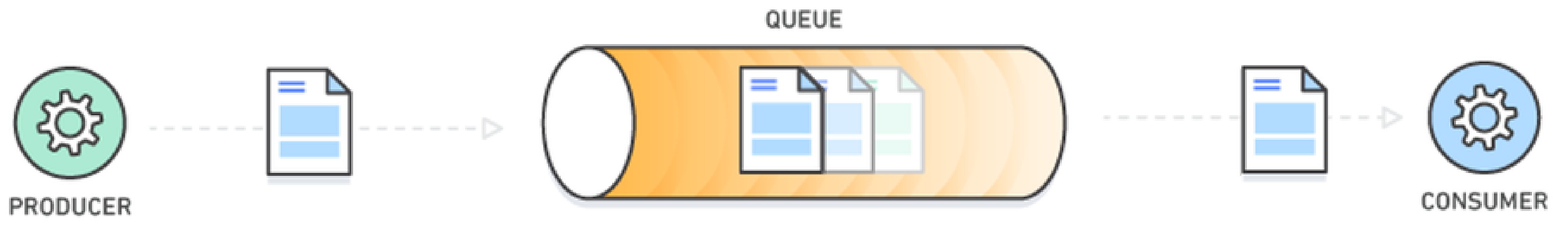
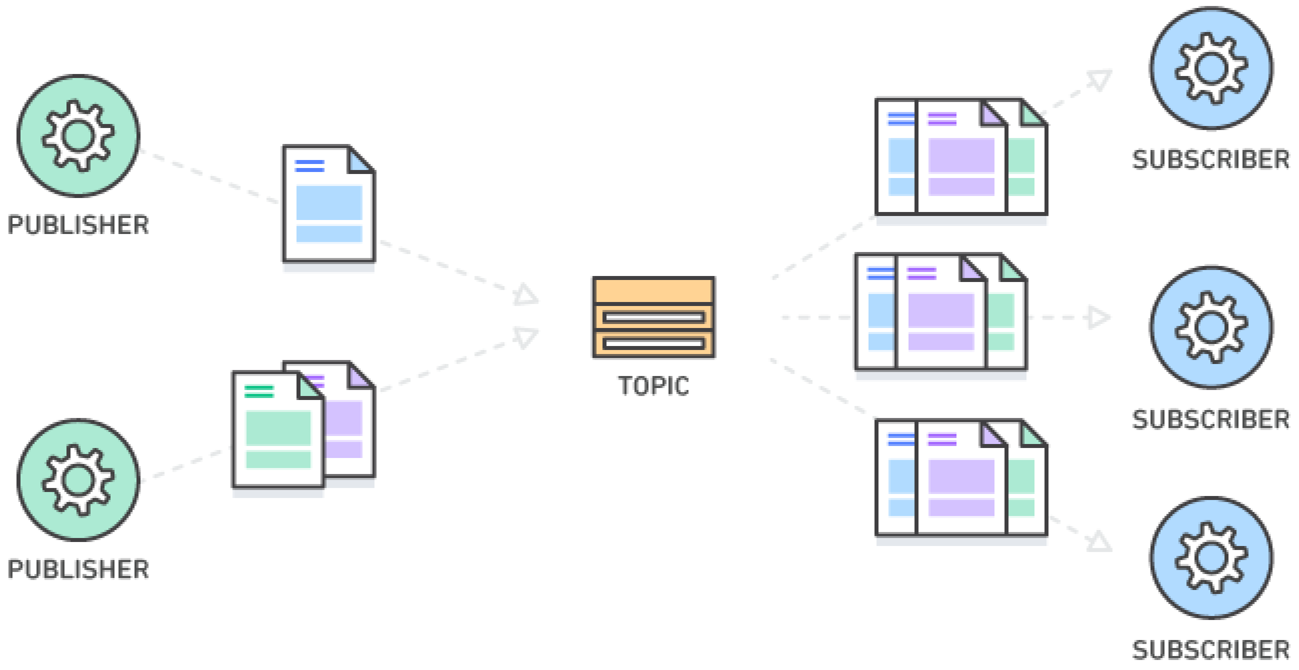
Queue model vs. Topic model

The point-to-point model is also known as the **queue model:**



The publisher or subscriber model is also simply known as the **topic model**:



Characteristics difference

In the queue model, there is an acknowledgment of the identity of the receiver and oftentimes the sender. In the topic model, there is anonymity in the identities of both the subscriber and publisher.

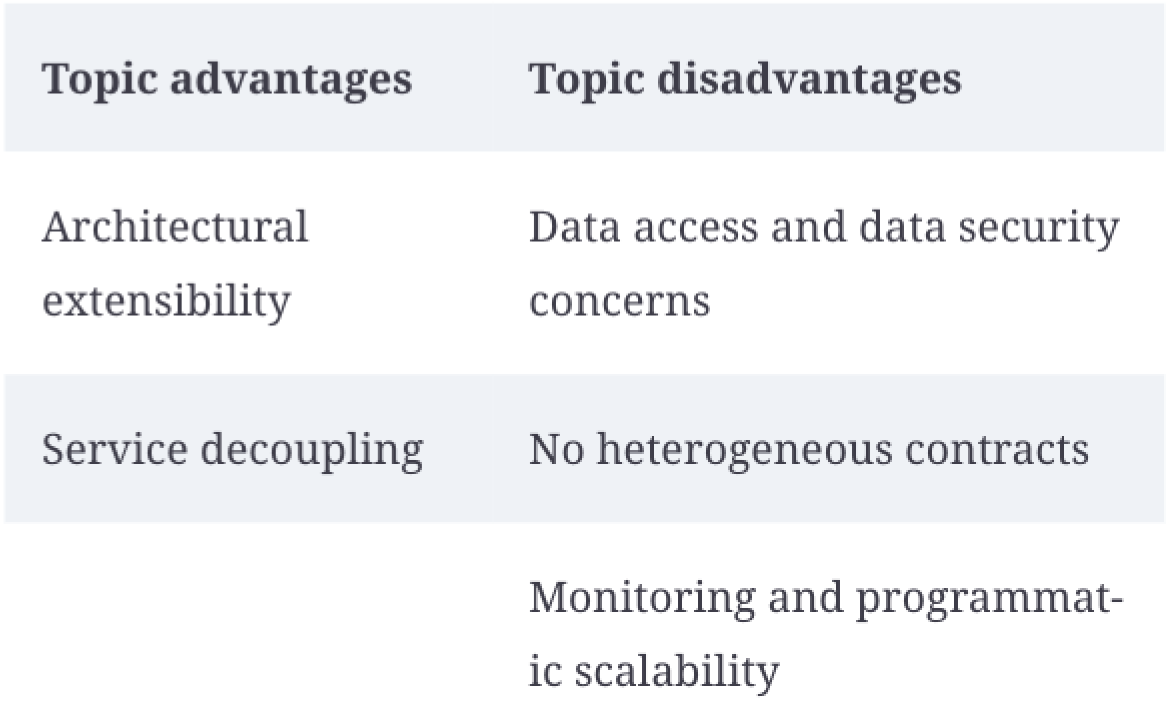
Queue model is only allowed one recipient; topic, on the other hand, can have multiple recipients.

In queue model, the sender and receiver do not have to be both active at the same time. In the topic model, timing is very vital.

In the queue model, the sender will receive a notification when the message gets to the receiver. The topic model, on the other hand, will not notify you with such, and there is even a risk that you will have no subscribers.

Review of Trade-offs for Topic model

Table 2-1. Trade-offs for topics



Topic advantages	**Topic disadvantages**
Architectural extensibility	Data access and data security concerns
Service decoupling	No heterogeneous contracts
	Monitoring and programmatic scalability

Think about how the different characteristics listed above relate to the trade-offs here.

When to use Topic model

Use this pattern when:

- An application needs to broadcast information to a significant number of consumers.
- An application needs to communicate with one or more independently-developed applications or services, which may use different platforms, programming languages, and communication protocols.
- An application can send information to consumers without requiring real-time responses from the consumers.
- The systems being integrated are designed to support an eventual consistency model for their data.
- An application needs to communicate information to multiple consumers, which may have different availability requirements or uptime schedules than the sender.

This pattern might not be useful when:

- An application has only **a few consumers** who need significantly **different information** from the producing application.
- An application requires **near real-time interaction** with consumers.

Think about how that's related to the different characteristics listed above.

Why are they important?

In modern cloud architecture, applications are decoupled into smaller, independent building blocks that are easier to develop, deploy and maintain. Both models provide communication and coordination for these distributed applications.

References

http://www.differencebetween.net/technology/internet/difference-between-queue-and-topic/#ixzz79CvDxatg

https://aws.amazon.com/pub-sub-messaging/
https://aws.amazon.com/message-queue/
https://docs.microsoft.com/en-us/azure/architecture/patterns/publisher-subscriber