Design of a Distributed Messaging Queue: Part 2

Learn about the detailed design of a messaging queue and its management at the back-end servers.

We'll cover the following

- Back-end service
  - Primary-secondary model
  - A cluster of independent hosts

In the previous lesson, we discussed the responsibilities of front-end servers and metadata services. In this lesson, we'll focus on the main part of the design where the queues and messages are stored: the back-end service.

Back-end service

This is the core part of the architecture where major activities take place. When the front-end receives a message, it refers to the metadata service to determine the host where the message needs to be sent. The message is then forwarded to the host and is replicated on the relevant hosts to overcome a possible availability issue. The message replication in a cluster on different hosts can be performed using one of the following two models:

1. *Primary-secondary model*
2. *A cluster of independent hosts*

Before delving into the details of these models, let's discuss the two types of cluster managers responsible for queue management: *internal* and *external*

cluster managers. The differences between these two cluster managers are shown in the following table.

Internal versus External Cluster Managers

Internal Cluster Manager	External Cluster Manager
It manages the assignment of queues within a cluster.	It manages the assignment of queues across clusters.
It knows about each and every node within a cluster.	It knows about each cluster. However, it doesn't have information on every host that's present inside a cluster.
It listens to the heartbeat from each node.	It monitors the health of each independent cluster.
It manages host failure, instance addition, and removals from the cluster.	It manages and utilizes clusters.
It partitions a queue into several parts and each part gets a primary server.	It may split a queue across several clusters, so that messages for the same queue are equally distributed between several clusters.

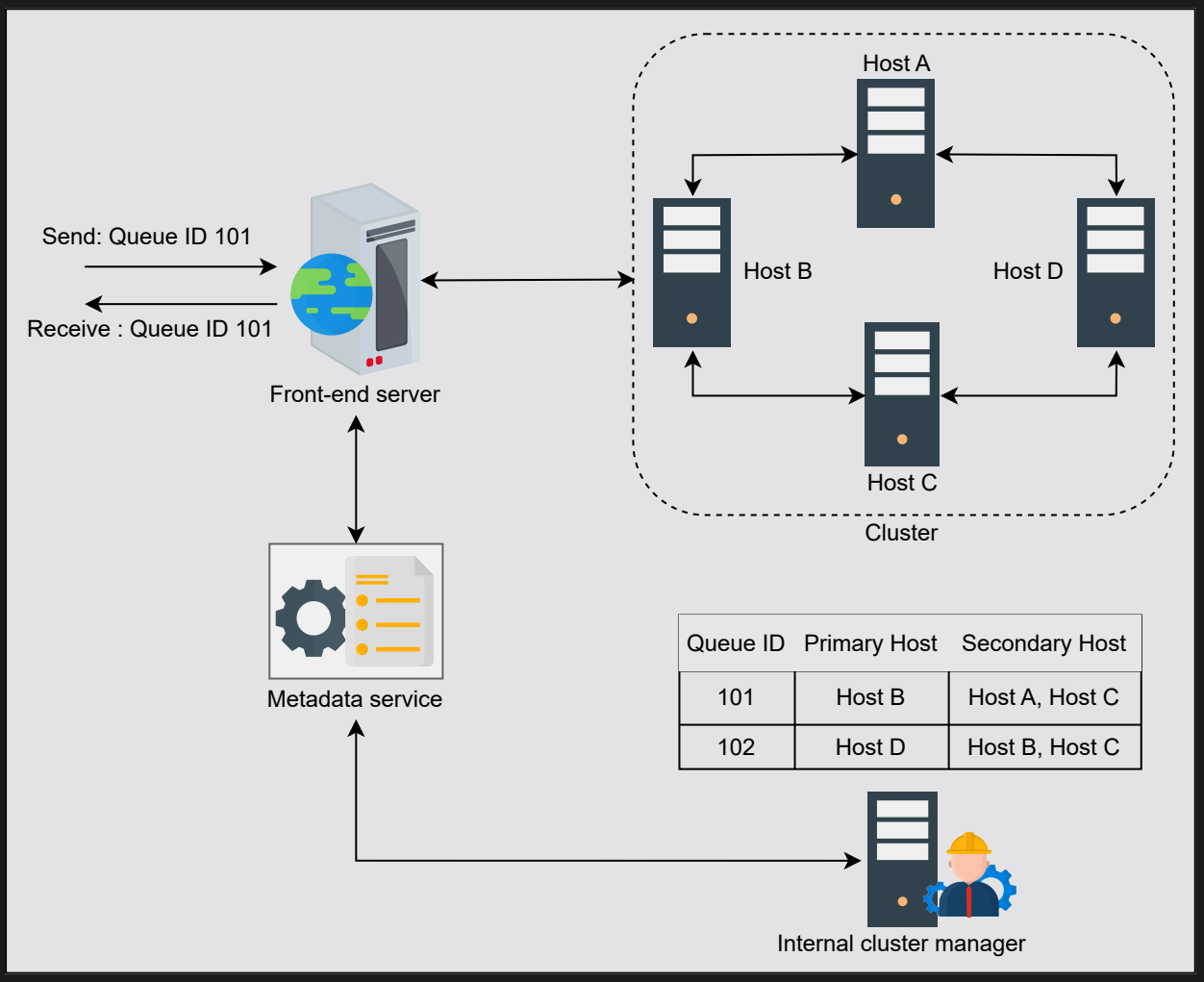
Primary-secondary model

In the **primary-secondary model**, each node is considered a **primary host** for a collection of queues. The responsibility of a primary host is to receive requests for a particular queue and be fully responsible for data replication. The request is received by the front-end, which in turn communicates with the metadata service to determine the primary host for the request.

For example, suppose we have two queues with the identities 101 and 102 residing on four different hosts A, B, C, and D. In this example, instance B is the

primary host of queue 101 and the secondary hosts where the queue 101 is replicated are A and C. As the front-end receives message requests, it identifies the primary server from the internal cluster manager through the metadata service. The message is retrieved from the primary instance, which is also responsible for deleting the original message upon usage and all of its replicas.

As shown in the following illustration, the internal cluster manager is a component that's responsible for mapping between the primary host, secondary hosts, and queues. Moreover, it also helps in the primary host selection. Therefore, it needs to be reliable, scalable, and performant.



Primary-secondary model of distributed queue: A request is received for a queue with ID 101, which is served accordingly

A cluster of independent hosts

In the approach involving a **cluster of independent hosts**, we have several clusters of multiple independent hosts that are distributed across data centers.

As the front-end receives a message, it determines the corresponding cluster via the metadata service from the external cluster manager. The message is then forwarded to a random host in the cluster, which replicates the message in other hosts where the queue is stored.

Point to Ponder

Question

How does a random host within a cluster replicate data—that is, messages—in the queues on other hosts within the same cluster?

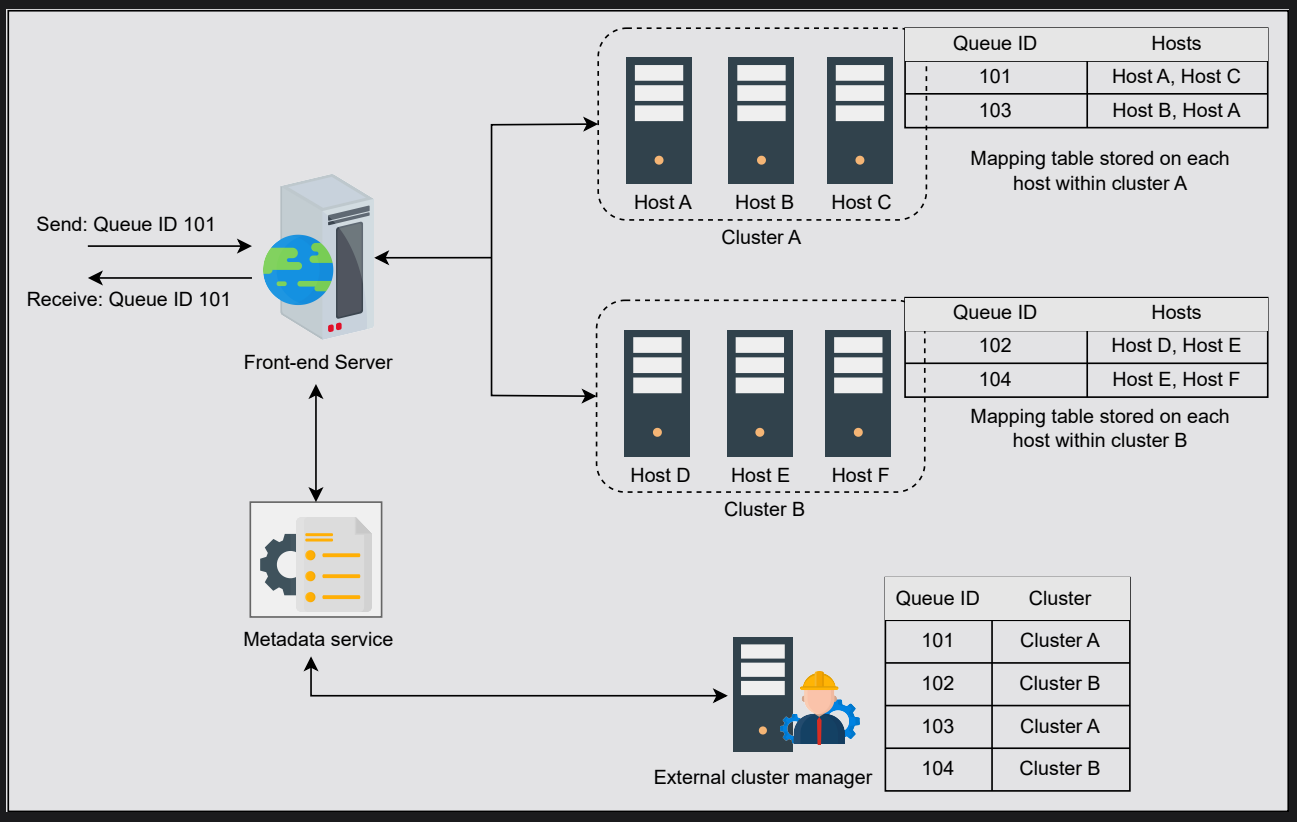
Show Answer

The same process is applied to receive message requests from the consumer. Similar to the first approach, the randomly selected host is responsible for message delivery and cleanup upon a successful processing of the message.

Furthermore, another component called an **external cluster manager** is introduced, which is accountable for maintaining the mapping between queues and clusters, as shown in the following figure. The external cluster manager is also responsible for queue management and cluster assignment to a particular queue.

The following figure illustrates the cluster of independent hosts. There are two clusters, A and B, which consist of several nodes. The external cluster manag has the mapping table between queues and their corresponding cluster. Whenever a front-end receives a request for a queue, it determines the corresponding cluster for the queue and forwards the request to the cluster

where the queue resides. The nodes within that cluster are responsible for storing and sending messages accordingly.



A cluster of independent hosts that consist of distributed queues

Point to Ponder

Question

What kind of anomalies can arise while replicating messages on other hosts?

Show Answer

We have completed the design of a distributed messaging queue and discussed the two models for organizing the back-end server. We also described the management process of queues and how messages are processed at the back-

end. Furthermore, we discussed how back-end servers are managed through different cluster managers.

In the next lesson, we discuss how our system fulfills the functional and non-functional requirements that were described earlier in the chapter.

Back

Design of a Distributed Messaging Q…

Completed

Next

Evaluation of a Distributed Messaging…