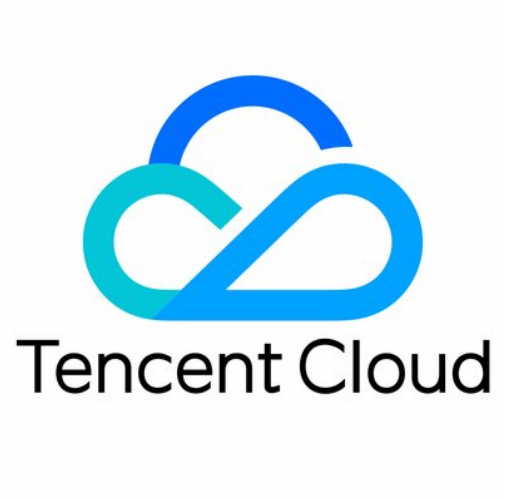
TDMQ for RabbitMQ

Product Introduction

Product Documentation



©2013-2022 Tencent Cloud. All rights reserved.

Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice

Tencent Cloud

All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

©2013-2022 Tencent Cloud. All rights reserved.

Product Introduction

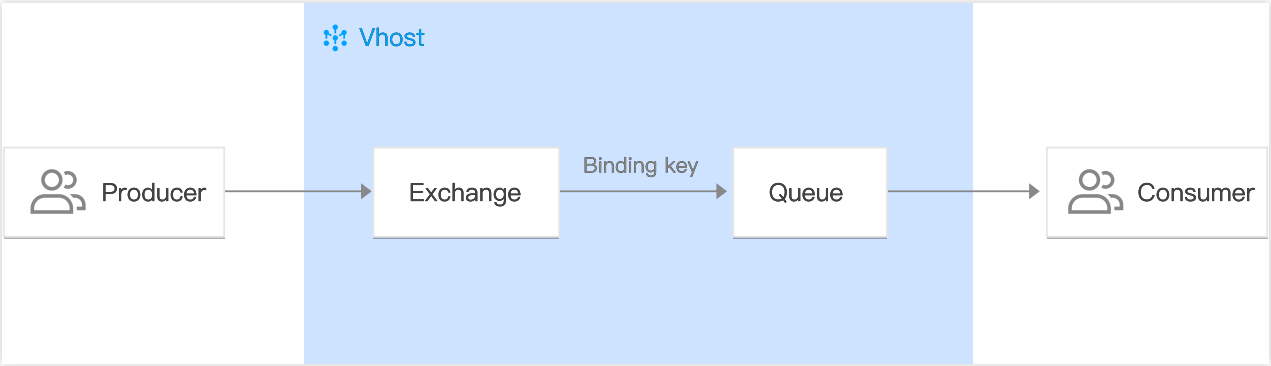
Overview

Last updated：2024-01-03 11:39:16

TDMQ for RabbitMQ is Tencent's proprietary message queue service. It supports the AMQP 0-9-1 protocol and is fully compatible with all components and principles of Apache RabbitMQ. It also has the underlying benefits of computing-storage separation and flexible scaling.
TDMQ for RabbitMQ has extremely flexible routing to adapt to the message delivery rules of various businesses. It can buffer the upstream traffic pressure and ensure the stable operations of the message system. It is often used to implement async communication and service decoupling between systems to reduce their mutual dependency, making it widely applicable to distributed systems in finance and government affairs industries.

Infrastructure

The basic concepts of TDMQ for RabbitMQ are as follows:
Producer: Sends messages to exchanges.
Vhost: Is used for logical isolation. Exchanges and queues of different vhosts are isolated from each other.
Exchange: Receives messages from producers and routes them to queues.
Queue: Caches messages for consumers to consume them.
Consumer: Pulls messages from queues for consumption.



For more concepts of TDMQ for RabbitMQ, see Basic Concepts.

©2013-2022 Tencent Cloud. All rights reserved.

Strengths

Last updated：2024-01-03 11:39:16

Open-Source Component Compatibility

TDMQ for RabbitMQ supports the AMQP 0-9-1 standard protocol and is fully compatible with the open-source RabbitMQ community and its queue, exchange, and vhost components, so that you can quickly migrate the metadata from RabbitMQ to Tencent Cloud at zero costs.

Diverse Features

TDMQ for RabbitMQ supports various messaging patterns of native RabbitMQ as well as dead letter switch and standby switch, eliminating your concerns over message loss caused by message expiration or routing failure.

Stability and Reliability

TDMQ for RabbitMQ features a persistent storage mechanism for high availability. The persistence of exchanges, queues, and messages ensures that the metadata and message content after service restart will never get lost. It stores messages in three replicas. When a physical machine is faulty, the data can be quickly migrated to guarantee the availability of three data replicas, achieving a 99.95% service availability.

High Scalability

TDMQ for RabbitMQ supports more queues and has higher scalability than open-source RabbitMQ. Its underlying system can automatically scale clusters based on the business scale in a way imperceptible to users.

Ease of Ops-Free Use

TDMQ for RabbitMQ provides access APIs and open-source SDKs for all programming languages on all versions. It offers the entire set of Ops services of the Tencent Cloud platform and monitors alarms in real time to help you quickly discover and solve problems and guarantee the service availability.

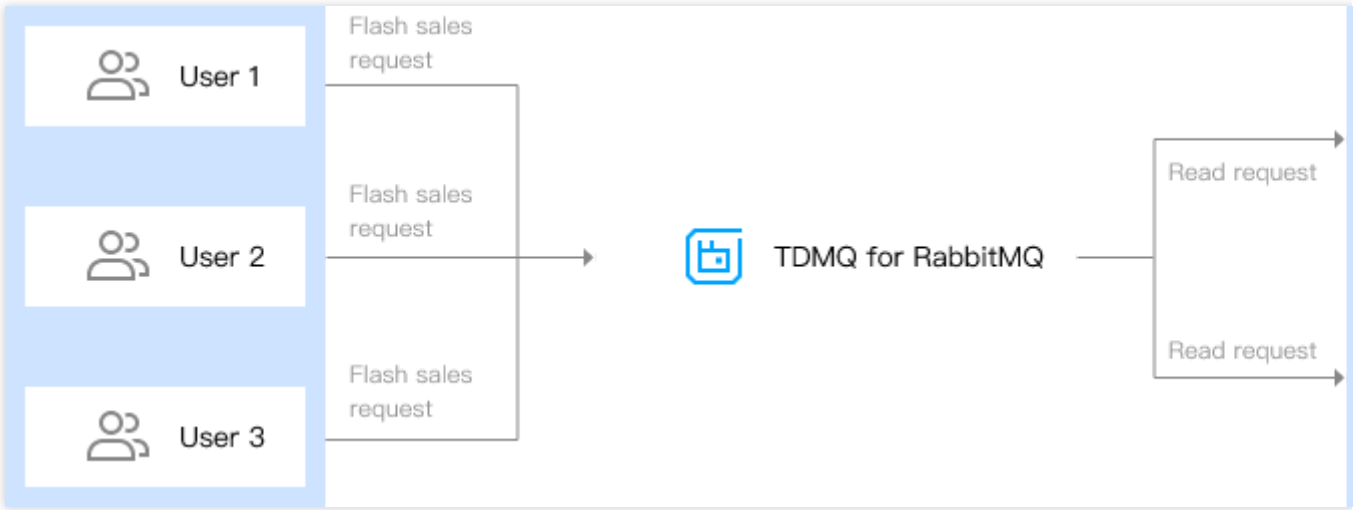
©2013-2022 Tencent Cloud. All rights reserved.

Use Cases

Last updated：2024-01-03 11:39:16

Traffic Peak Shifting for Flash Sales System

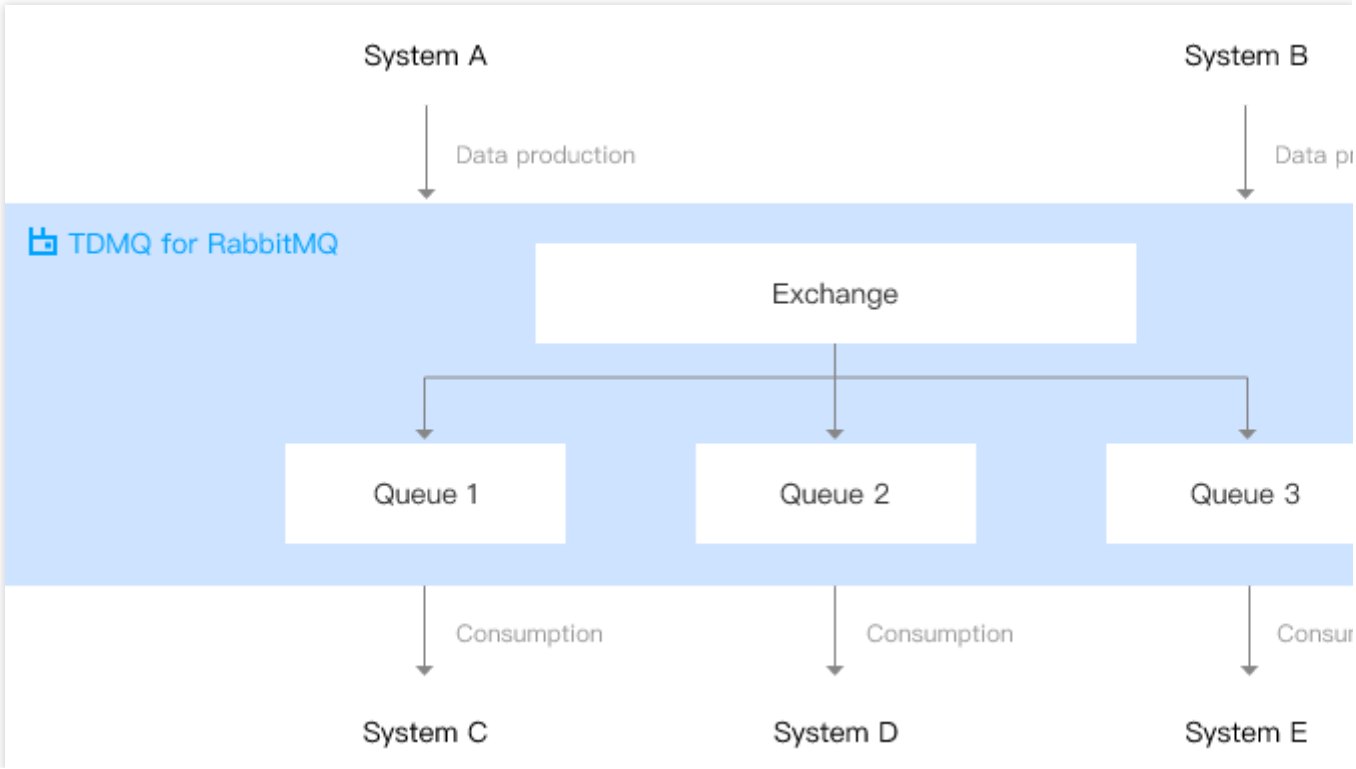
A flash sales system may fail due to traffic surges. TDMQ for RabbitMQ can mitigate the traffic pressure at the upstream to secure the stable operations of the message system.



Async Decoupling of Business Systems

The order data in a transaction system involves over 100 downstream business systems, such as shipping, logistics, and order. TDMQ for RabbitMQ can implement async communication and service decoupling between systems to reduce their mutual dependency, improve the processing efficiency, and ensure the system stability.

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

Use Limits

Last updated：2024-04-23 11:25:56

This document describes the limits on some metrics and performance in the TDMQ for RabbitMQ exclusive cluster console. Do not exceed these limits during use to avoid exceptions.

Cluster

Limit	Description
Cluster Name Length	A cluster name is a string of 3 to 64 characters.
Cluster Name Specification	A cluster name cannot be empty and can only contain digits, letters, hyphens (-), and underscores (_).

Vhost

Limit	Description
Vhost Name Length	A Vhost name is a string of 1 to 64 characters.
Vhost Name Specification	A Vhost name cannot be empty and can only contain letters, digits, periods (.), hyphens (-), and underscores (_).
Vhost Quantity	Each cluster can have up to 20 Vhosts.

Exchange

Limit	Description
Exchange Name Length	An exchange name is a string of 1 to 64 characters.
Exchange Name Specification	An exchange name cannot be empty and can only contain letters, digits, periods (.), hyphens (-), and underscores (_).
Exchange Routing Type	Options include **direct**, **fanout**, **topic**, and **headers**. A routing type must be selected when an exchange is created.
Number of Exchanges	Each cluster can have up to 1000 exchanges.

Queue

Limit	Description

©2013-2022 Tencent Cloud. All rights reserved.

Queue Name Length	A queue name is a string of 1 to 64 characters.
Queue Name Specification	A queue name cannot be empty and can only contain letters, digits, period (.), hyphens (-), and underscores (_).
Number of Queues	Each cluster can have up to 1000 queues.

©2013-2022 Tencent Cloud. All rights reserved.

Relevant Concepts
Basic Concepts

Last updated：2024-01-03 11:39:16

This document describes the common terms in TDMQ for RabbitMQ.

Term	Definition
Channel	In each physical TCP connection of a client, multiple channels can be established, each of which represents a session task.
Connection	It refers to a physical TCP connection between a producer or consumer and TDMQ for RabbitMQ.
Vhost	A virtual host (vhost) is used for logical isolation and manages its own exchanges, queues, and bindings separately, so that applications can run securely in different vhosts without interfering with each other. There can be multiple vhosts under one instance, and there can also be multiple exchanges and queues in one vhost. A vhost must be specified when a producer or consumer connects to TDMQ for RabbitMQ.
Queue	A queue is an internal object of RabbitMQ used to store messages. Each message will be put into one or more queues.
Exchange	A producer sends a message to an exchange, which then routes the message to one or more queues based on its attributes or content (or discards it).
Routing key	When a producer sends a message to an exchange, it usually specifies a routing key to indicate the routing rule for the message, and the routing key needs to be used together with an exchange type and a binding key to eventually take effect. If the exchange type and binding key are fixed (which is generally the case), the producer can determine the message flow by specifying the routing key when sending the message to the exchange.
Binding	RabbitMQ associates an exchange with a queue through a binding, so that it can know how to correctly route a message to the specified queue.
Binding key	When a queue is bound to an exchange, a binding key is generally specified. Then, when a consumer sends a message to the exchange, the message will be routed to the bound queue if the binding key matches the routing key. If multiple queues are bound to the same exchange, these bindings can use the same binding key. The binding key does not take effect in all situations; instead, it depends on the exchange type. For example, a fanout exchange will ignore the binding key and route the message to all queues bound to the exchange.
Exchange	Commonly used exchange types in RabbitMQ include fanout, direct, and topic.

©2013-2022 Tencent Cloud. All rights reserved.

type	1. Fanout: a fanout exchange will route all messages sent to it to all queues bound to it. 2. Direct: a direct exchange will route messages to the queue whose binding key exactly matches the routing key. 3. Topic: a topic exchange is similar to a direct exchange. It supports multi-condition match and fuzzy match; that is, it will route messages to the queues bound to it by using routing key pattern match and string comparison.

©2013-2022 Tencent Cloud. All rights reserved.

Exchange

Last updated：2024-01-03 11:39:16

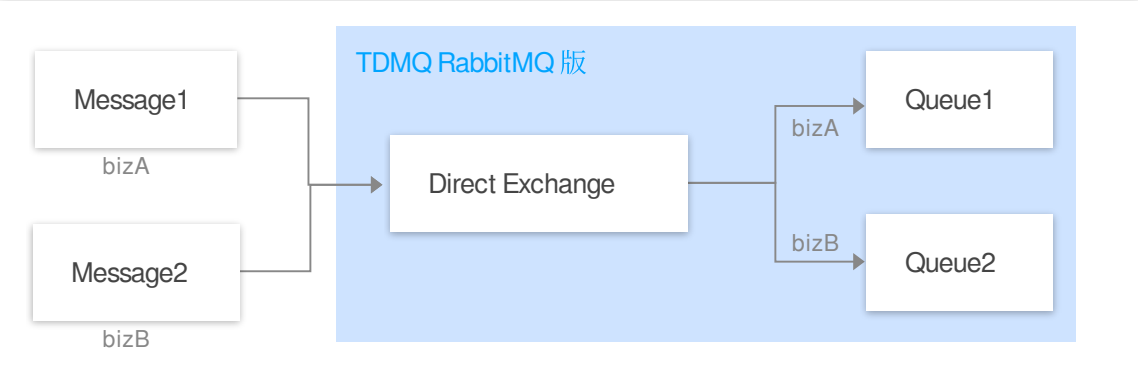
This document describes the concept, types, and usage of exchange in TDMQ for RabbitMQ.

Concept

Exchange is a message routing agent in TDMQ for RabbitMQ. A producer sends a message to an exchange, which then routes the message to one or more queues based on its attributes or content (or discards it). Then, a consumer pulls it from such queues for consumption.
TDMQ for RabbitMQ currently supports four types of exchange: Direct, Fanout, Topic, and Header.
Direct: a direct exchange will route messages to the queue whose binding key exactly matches the routing key.
Fanout: a fanout exchange will route messages to all queues bound to it.
Topic: a topic exchange supports multi-condition match and fuzzy match; that is, it will route messages to the queues bound to it by using routing key pattern match and string comparison.
Header: A header exchange has nothing to do with routing key and matches messages by the `Headers` attribute. When binding a queue to a header exchange, declare a map key-value pair to implement the binding.

Direct Exchange

Routing rule: a direct exchange will route messages to the queue whose binding key exactly matches the routing key.
Use case: this type of exchange is suitable for filtering messages by simple character identifiers and is often used for unicast routing.
Example:



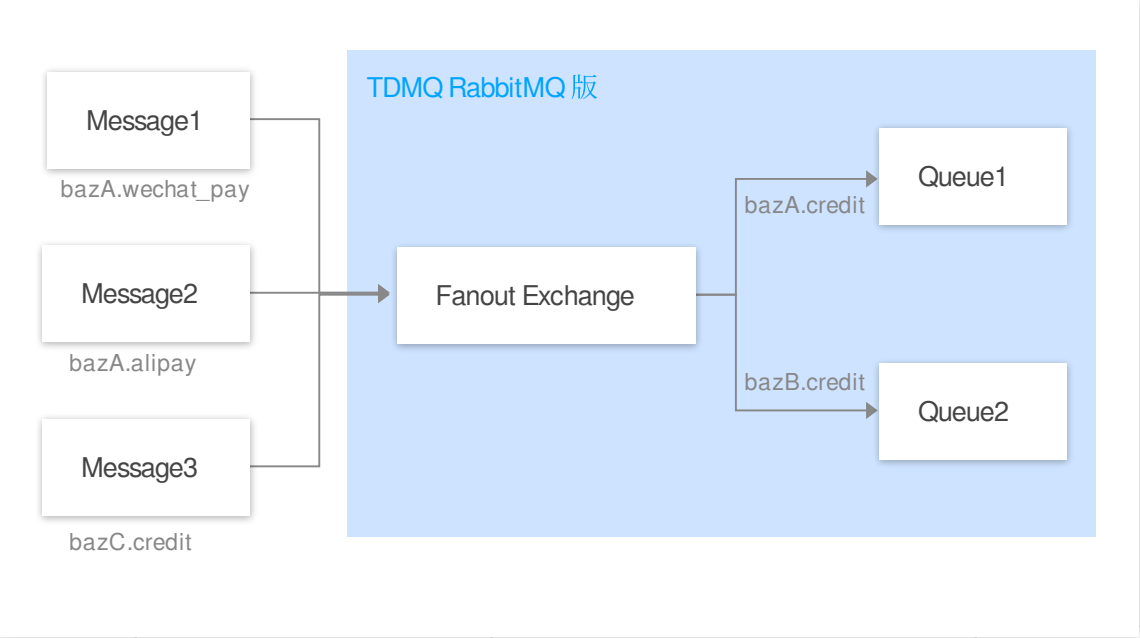
Message	Routing Key	Binding Key	Queue
Message 1	`bizA`	`bizA`	Queue 1

©2013-2022 Tencent Cloud. All rights reserved.

Message 2	`bizB`	`bizB`	Queue 2

Fanout Exchange

Routing rule: this type of exchange will route messages to all queues bound to it.
Use case: this type of exchange is suitable for broadcast message scenarios. For example, a distribution system can use a fanout exchange to broadcast various status and configuration updates.
Example



Message	Routing Key	Binding Key	Queue
Message 1	`bazA.wechat_pay`	`bazA.credit` , `bazB.credit`	Queue 1, Queue 2
Message 2	`bazA.alipay`	`bazA.credit` , `bazB.credit`	Queue 1, Queue 2
Message 3	`bazC.credit`	`bazA.credit` , `bazB.credit`	Queue 1, Queue 2

Topic Exchange

Routing rule: this type of exchange supports multi-condition match and fuzzy match; that is, it will route messages to the queues bound to it by using routing key pattern match and string comparison.

©2013-2022 Tencent Cloud. All rights reserved.

The wildcards supported by topic exchanges include asterisk "*" and pound sign "#".

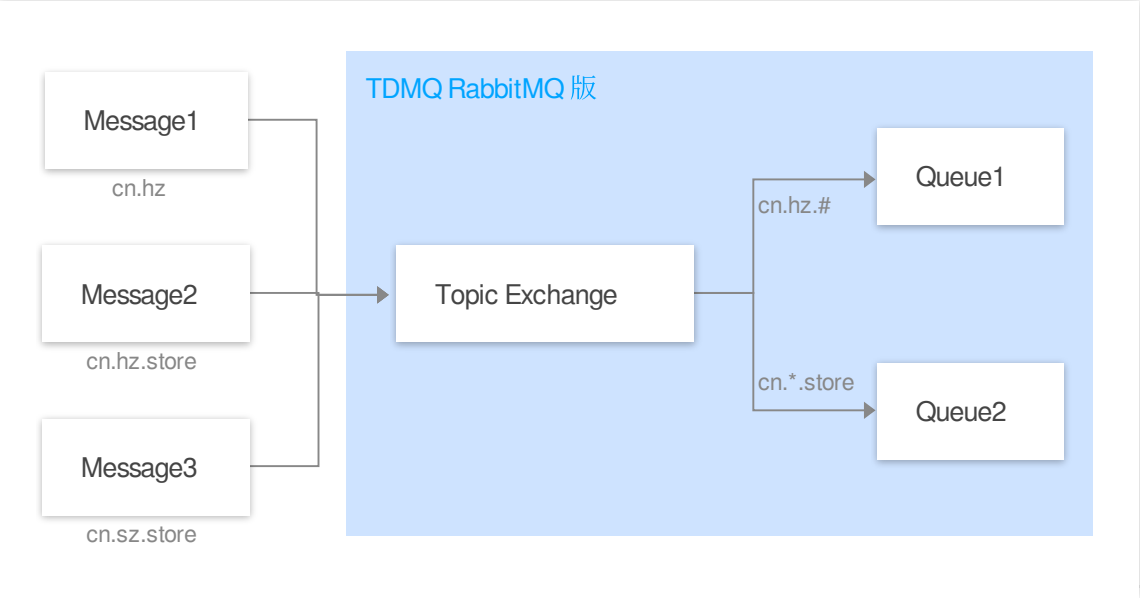
"*" represents a word, such as `sh` .

"#" represents zero, one, or more words separated by period ".", such as `cn.hz` .

Use case

This type of exchange is often used for multicast routing. For example, you can use a topic exchange to distribute data about specific geographic locations.

Example



Message	Routing Key	Binding Key	Queue
Message 1	`cn.hz`	`cn.hz.#`	Queue 1
Message 2	`cn.hz.store`	`cn.hz.#` , `cn.*.store`	Queue 1, Queue 2
Message 3	`cn.sz.store`	`cn.*.store`	Queue 2

Header Exchange

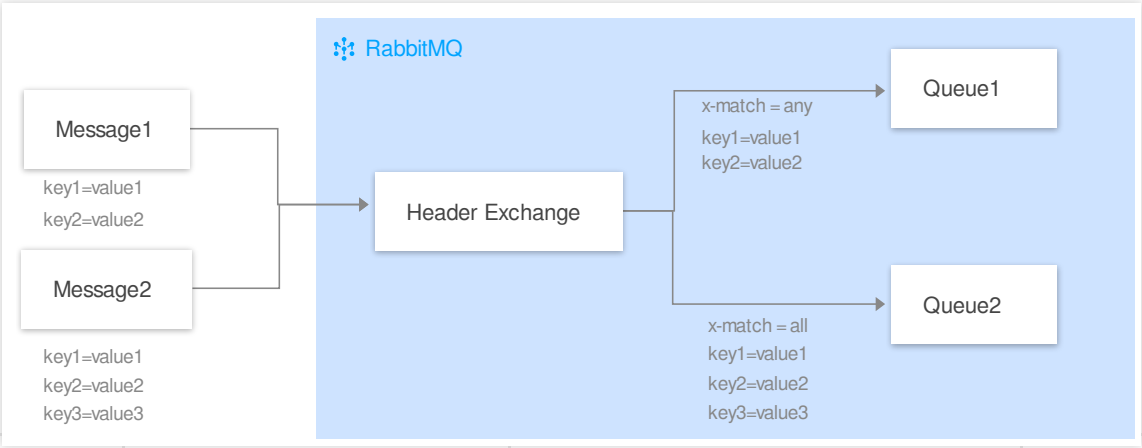
A header exchange has nothing to do with routing key and matches messages by the `Headers` attribute. When binding a queue to a header exchange, declare a map key-value pair to implement the binding. When a message is sent to RabbitMQ, the `Headers` attribute of the message will be obtained to match the key-value pair specified during exchange binding, and the message will be routed to the queue only if there is a full match.

The x-match rule has two types:

x-match = all: A message can be received only if all key-value pairs are matched.

x-match = any: A message can be received as long as any key-value pair is matched.

©2013-2022 Tencent Cloud. All rights reserved.



Message	Message Headers Attribute	Binding Headers Attribute	Queue

Message 1	`key1=value1`
	`key2=value2`
		`x-match = any`
		`key1=value1`
		`key2=value2`
			Queue 1

Message 2	`key1=value1` `key2=value2` `key3=value3`	`x-match = any` `key1=value1` `key2=value2` `x-match = all` `key1=value1` key2=value2` `key3=value3`	Queue 1, Queue 2

©2013-2022 Tencent Cloud. All rights reserved.