

TDMQ for RabbitMQ

Product Introduction

Product Documentation



Copyright Notice

©2013-2019 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



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

Product Introduction

- Overview

- Strengths

- Use Cases

- Use Limits

- Relevant Concepts

 - Basic Concepts

 - Exchange

Product Introduction

Overview

Last updated : 2022-07-04 15:32:57

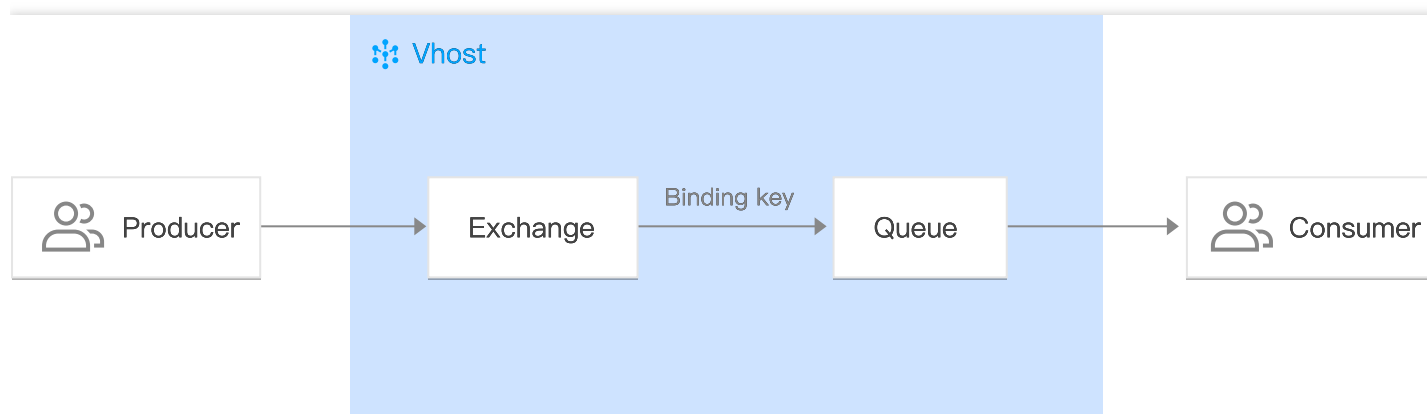
TDMQ for RabbitMQ is Tencent's proprietary message queue service. It supports the AMQP 0-9-1 protocol and is fully compatible with all components and principles of Apache RabbitMQ. It also has the underlying benefits of computing-storage separation and flexible scaling.

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](#).

Strengths

Last updated : 2022-07-04 15:32:57

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

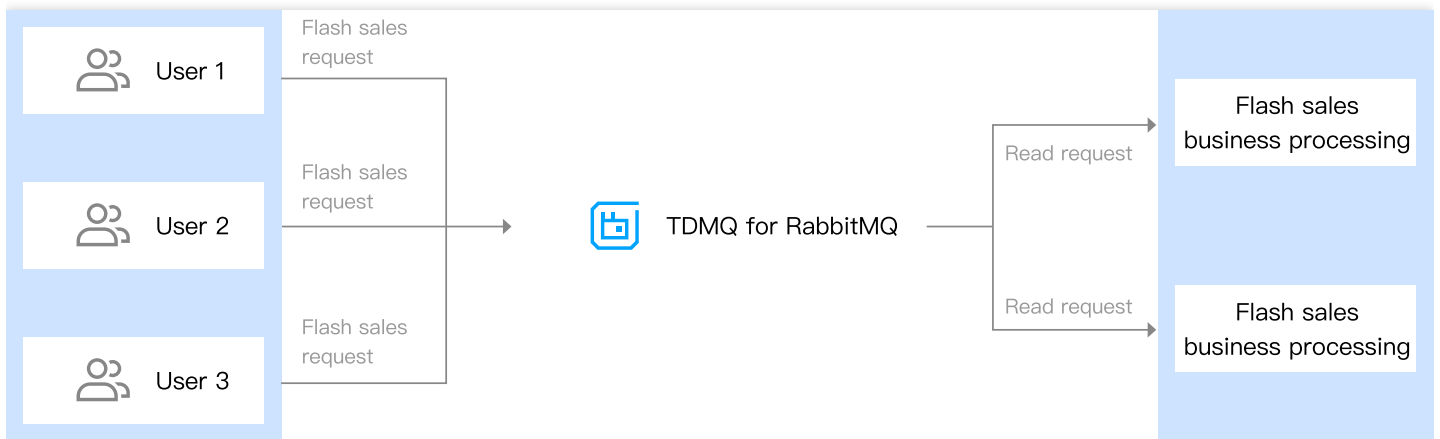
TDMQ for RabbitMQ provides access APIs and open-source SDKs for all programming languages on all versions. It offers the entire set of Ops services of the Tencent Cloud platform and monitors alarms in real time to help you quickly discover and solve problems and guarantee the service availability.

Use Cases

Last updated : 2022-02-11 14:24:35

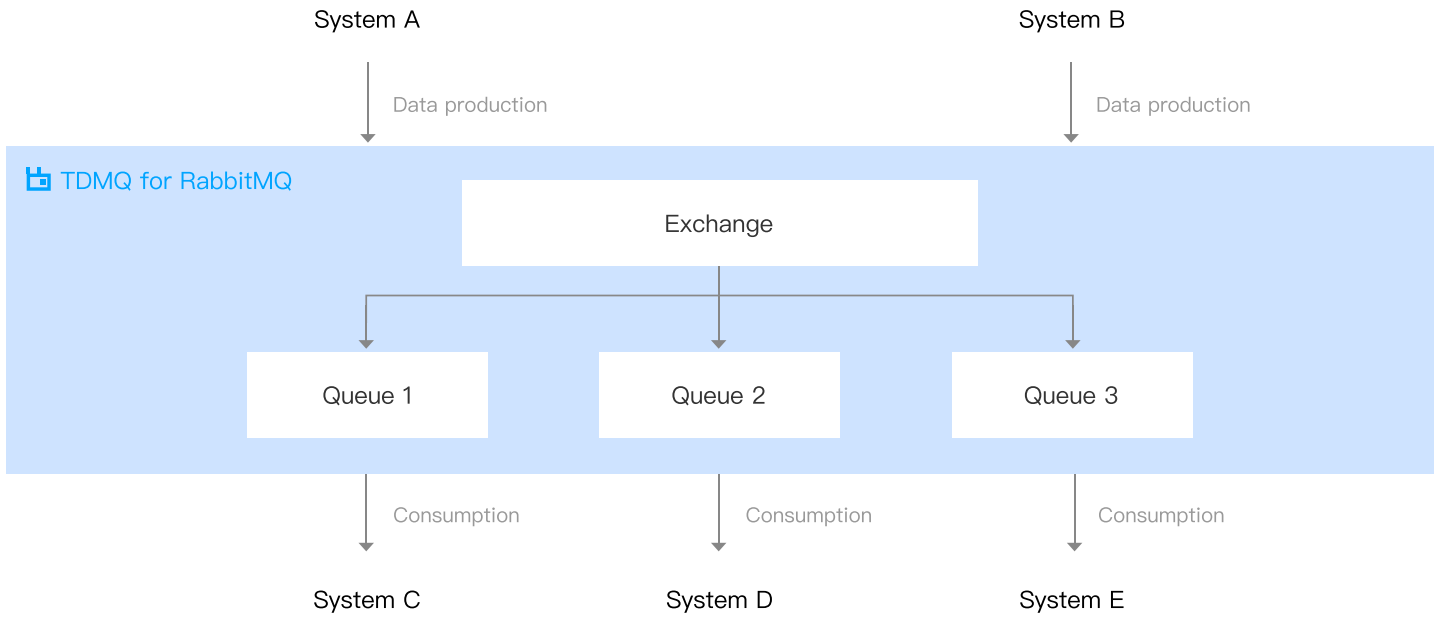
Traffic Peak Shifting for Flash Sales System

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.



Use Limits

Last updated : 2022-07-04 15:32:57

This document lists the limits of certain metrics and performance in TDMQ for RabbitMQ. Be careful not to exceed the limits during use to avoid exceptions.

Cluster

Limit	Description
Maximum number of clusters per region	5
Cluster name length	3-64 characters

Vhost

Limit	Description
Vhost name length	1-64 characters
Maximum TPS per vhost	8,000
Maximum number of client connections per vhost	8,000
Maximum number of vhosts per cluster	10

Exchange

Limit	Description
Exchange name length	1-64 characters
Maximum number of exchanges per cluster	1,000

Queue

Limit	Description
Maximum number of queues per cluster	1,000
Queue name length	1-64 characters

Message

Limit	Description
Maximum message retention period	15 days
Maximum message delay	10 days
Maximum message size	4 MB
Consumption offset reset	15 days

Channel

Limit	Description
Maximum number of channels per connection	1,000

Public Network Bandwidth

Limit	Description
Public network bandwidth per account	5 Mbps

Relevant Concepts

Basic Concepts

Last updated : 2022-02-11 14:25:00

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.

Term	Definition
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	<p>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.</p> <p>If multiple queues are bound to the same exchange, these bindings can use the same binding key.</p> <p>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.</p>
Exchange type	<p>Commonly used exchange types in RabbitMQ include fanout, direct, and topic.</p> <ul style="list-style-type: none">• Fanout: a fanout exchange will route all messages sent to it to all queues bound to it.• Direct: a direct exchange will route messages to the queue whose binding key exactly matches the routing key.• 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.

Exchange

Last updated : 2022-02-11 14:25:10

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 three types of exchange: direct, fanout, and topic.

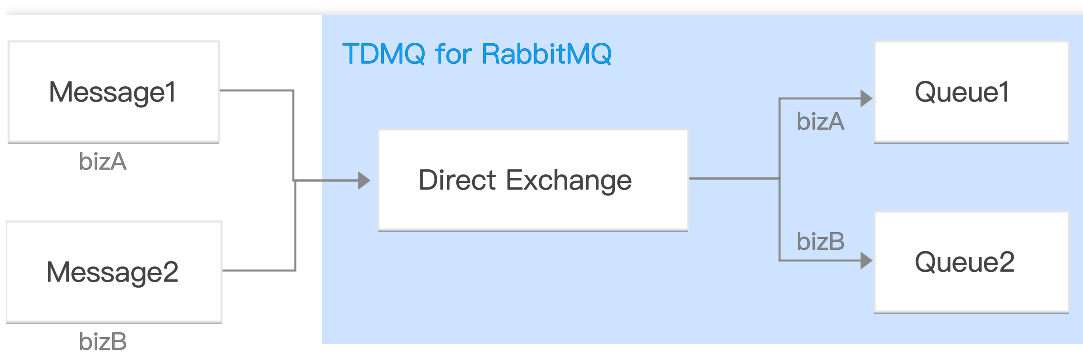
- Direct: a direct exchange will route messages to the queue whose binding key exactly matches the routing key.
- Fanout: a fanout exchange will route messages to all queues bound to it.
- Topic: a topic exchange supports multi-condition match and fuzzy match; that is, it will route messages to the queues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

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.

Sample:



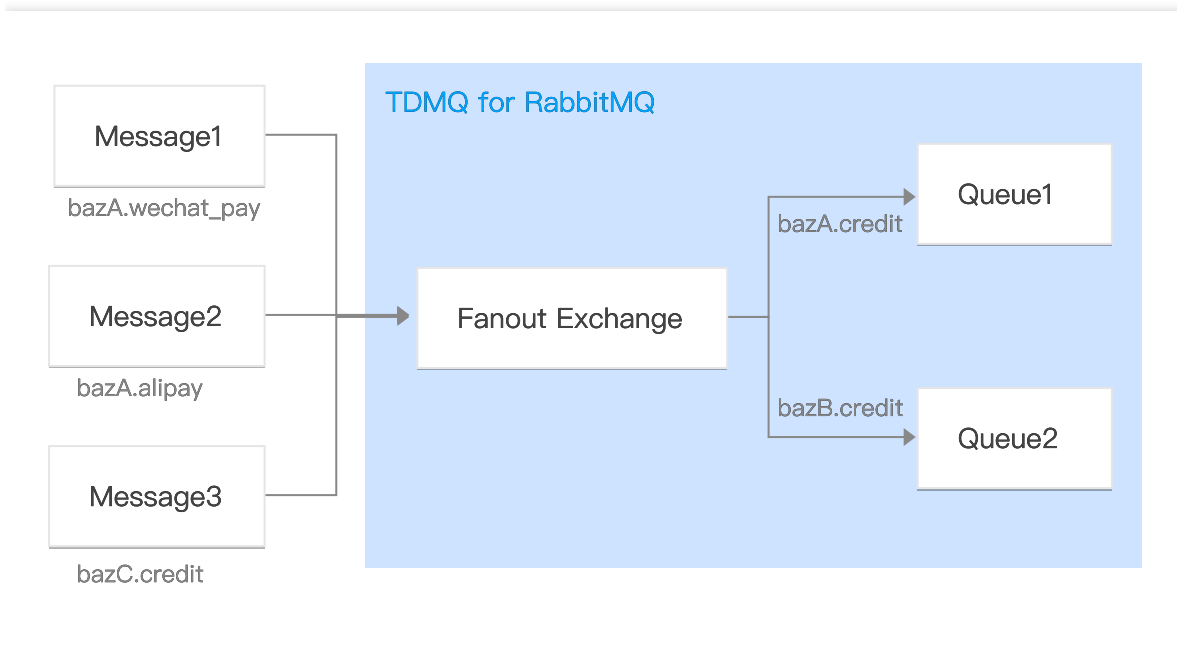
Message	Routing Key	Binding Key	Queue
Message 1	bizA	bizA	Queue 1
Message 2	bizB	bizB	Queue 2

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.

Sample:



Message	Routing Key	Binding Key	Queue
Message 1	bazA.wechat_pay	bazA.credit , bazB.credit	Queue 1, Queue 2

Message	Routing Key	Binding Key	Queue
Message 2	<code>bazA.alipay</code>	<code>bazA.credit</code> , <code>bazB.credit</code>	Queue 1, Queue 2
Message 3	<code>bazC.credit</code>	<code>bazA.credit</code> , <code>bazB.credit</code>	Queue 1, Queue 2

Topic Exchange

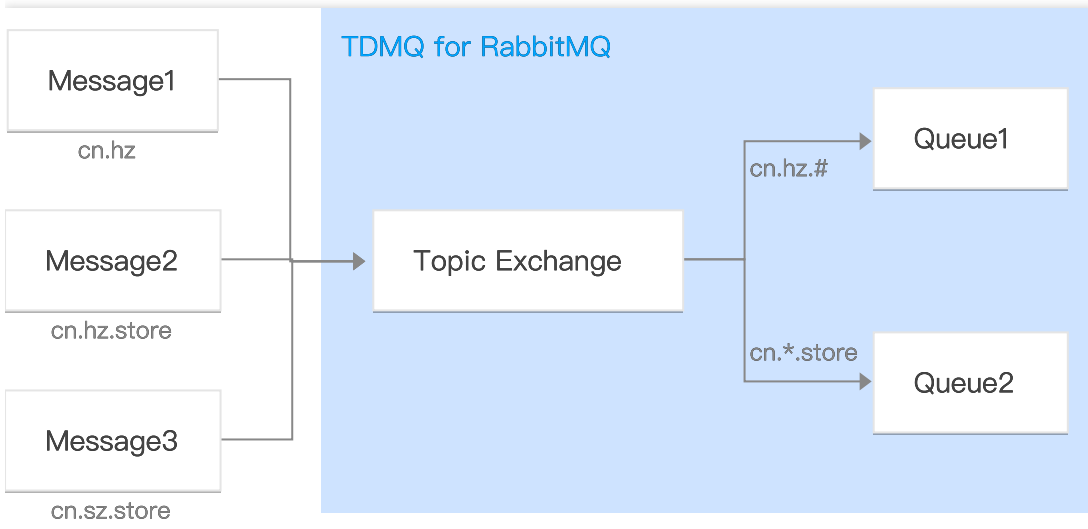
Routing rule: this type of exchange supports multi-condition match and fuzzy match; that is, it will route messages to the queues bound to it by using routing key pattern match and string comparison.

- 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.

Sample:



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