Distributed Message Service for Kafka

Service Overview

Issue	01
Date	2023-08-18

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 What Is DMS for Kafka?

Apache Kafka is distributed message middleware that features high throughput, data persistence, horizontal scalability, and stream data processing. It adopts the publish-subscribe pattern and is widely used for log collection, data streaming, online/offline system analytics, and real-time monitoring.

Distributed Message Service (DMS) for Kafka is a message queuing service based on Apache Kafka. This service provides Kafka premium instances. The computing, storage, and bandwidth resources used by an instance are exclusively occupied by the user. You can apply for instances as required. The instances can be used right out of the box, taking off the deployment and O&M pressure for you so that you can focus on developing your services.

Readers' Guide

This documentation introduces DMS for Kafka and its differences from Apache Kafka. You will learn about the detailed information about the specifications, console operations, and client access to instances of Huawei CloudDMS for Kafka.

For more information about the basic knowledge of Kafka or technical details about creating and retrieving messages, please go to the **official Apache Kafka website**.

2 Product Advantages

Huawei Cloud DMS for Kafka provides easy-to-use message queuing based on Apache Kafka. Services can be quickly migrated to the cloud without any change, reducing maintenance and usage costs.

- Rapid deployment

  Simply set instance information on the DMS for Kafka console, submit your order, and a complete Kafka instance will be automatically created and deployed.

- Service migration without modifications

  DMS for Kafka is compatible with open-source Kafka APIs and supports all message processing functions of open-source Kafka.

  If your application services are developed based on open-source Kafka, you can easily migrate them to Huawei Cloud DMS for Kafka after specifying a few authentication configurations.

  NOTE

  Kafka instances are compatible with Apache Kafka v1.1.0, v2.3.0, and v2.7. Keep the client and server versions the same.

- Security

  Operations on Kafka instances are recorded and can be audited. Messages can be encrypted before storage.

  In addition to Simple Authentication and Security Layer (SASL) authentication, Virtual Private Clouds (VPCs) and security groups also provide security controls on network access.

- Data reliability

  Kafka instances support data persistence and replication. Messages can be synchronously or asynchronously replicated between replicas and flushed to disk.

- High availability

  Kafka runs in clusters, enabling failover and fault tolerance so that services can run smoothly.

  Kafka instance brokers can be deployed across AZs to enhance service availability. Data is synchronized between different AZs based on Kafka's in-sync replica (ISR) mechanism. A topic must have multiple data copies and distribute them across ISRs. When ISR replication is normal, the recovery point objective (RPO) is close to 0.

- Simple O&M

  Huawei Cloud provides a whole set of monitoring and alarm services, eliminating the need for 24/7 attendance. Kafka instance metrics are monitored and reported, including the number of partitions, topics, and accumulated messages. You can configure alarm rules and receive SMS or email notifications on how your services are running in real time.

- Massive accumulation and scaling

  Kafka features high scalability because it runs in a distributed system, or cluster. You can configure up to 100 partitions for a topic. The storage space can be also expanded. This means that billions of messages can be accumulated, suitable for scenarios requiring high concurrency, high performance, and large-scale access.

- Flexible specifications

  You can customize the bandwidth and storage space for the instance and the number of partitions and replicas for topics in the instance.

3 Application Scenarios

Kafka is popular message-oriented middleware that features highly reliable, asynchronous message delivery. It is widely used for transmitting data between different systems in many industries, including enterprise application, payment, telecommunications, e-commerce, social networking, instant messaging, video, Internet of Things, and Internet of Vehicle.

Asynchronous Communication

Non-core or less important messages are sent asynchronously to receiving systems, so that the main service process is not kept waiting for the results of other systems, allowing for faster responses.

For example, Kafka can be used to send a notification email and SMS message after a user has registered with a website, providing fast responses throughout the registration process.

Figure 3-1 Serial registration and notification

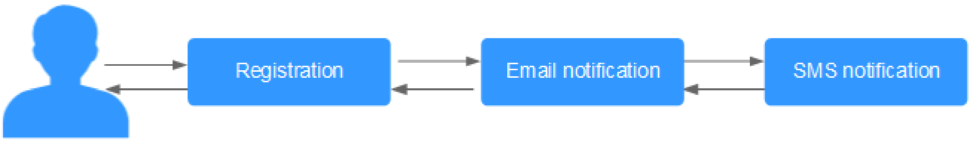
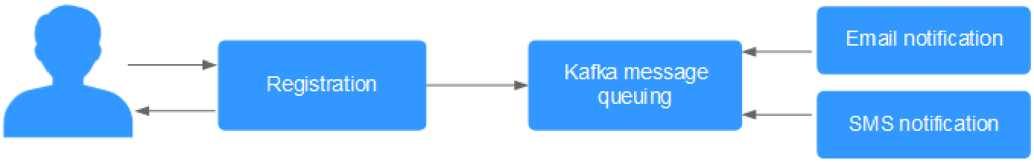


Figure 3-2 Asynchronous registration and notification using message queues

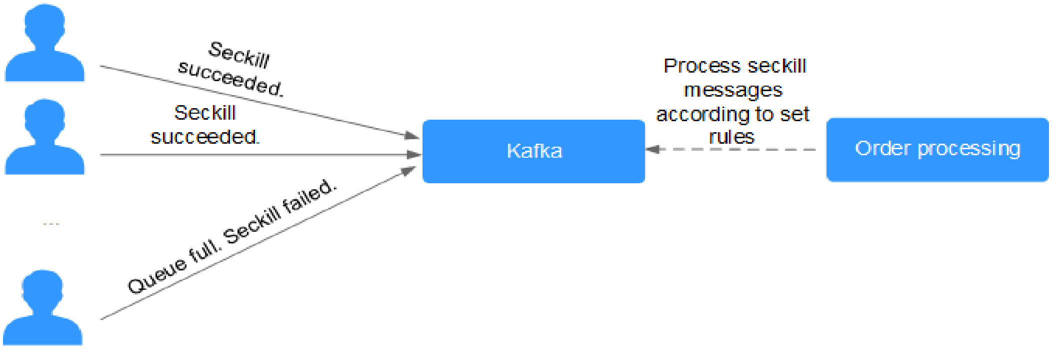


Traffic Control

In e-commerce systems or large-scale websites, there is a processing capability gap between upstream and downstream systems. Traffic bursts from upstream systems with high processing capabilities may have a large impact on downstream systems with lower processing capabilities. For example, online sales promotions involve a huge amount of traffic flooding into e-commerce systems. Kafka provides a three-day buffer by default for hundreds of millions of messages, such as orders and other information. In this way, message consumption systems can process the messages during off-peak periods.

In addition, flash sale traffic bursts originating from frontend systems can be handled with Kafka, keeping the backend systems from crashing.

Figure 3-3 Traffic burst handling using Kafka


Seckill succeeded.
Seckill succeeded.
...
Queue full. Seckill failed.
Kafka
Process seckill messages according to set rules
Order processing

Log Synchronization

In large-scale service systems, logs of different applications are collected for quick troubleshooting, full-link tracing, and real-time monitoring.

Kafka is originally designed for this scenario. Applications asynchronously send log messages to message queues over reliable transmission channels. Other components can read the log messages from message queues for further analysis, either in real time or offline. In addition, Kafka can collect key log information to monitor applications.

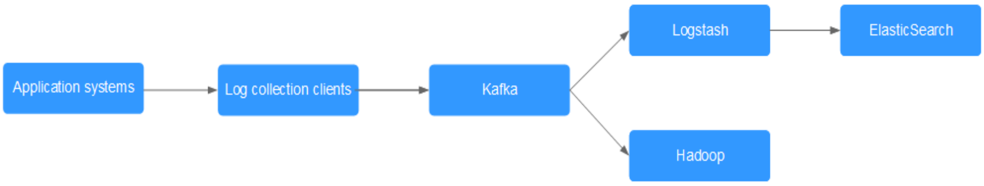
Log synchronization involves three major components: log collection clients, Kafka, and backend log processing applications.

1. The log collection clients collect log data from a user application service and asynchronously send the log data in batches to Kafka clients.

   Kafka clients receive and compress messages in batches. This only has a minor impact on the service performance.

2. Kafka persists logs.
3. Log processing applications, such as Logstash, subscribe to messages in Kafka and retrieve log messages from Kafka. Then, the messages are searched for by file search services or delivered to big data applications such as Hadoop for storage and analysis.

Figure 3-4 Log synchronization process



NOTE

Logstash is for log analytics, Elasticsearch is for log search, and Hadoop is for big data analytics. They are all open-source tools.

4 Specifications

Kafka Instance Specifications

Kafka instances are compatible with open-source Kafka v1.1.0, v2.3.0, and v2.7. The instance specifications are represented by the ECS flavor and the number of brokers. Available options are kafka.2u4g.cluster, kafka.4u8g.cluster, kafka.8u16g.cluster, kafka.12u24g.cluster, and kafka.16u32g.cluster.

NOTE

In the following table, transactions per second (TPS) are calculated assuming that the size of a message is 1 KB.

Table 4-1 Kafka instance specifications

Flavor	Brokers	Maximum TPS per Broker	Maximum Partitions per Broker	Maximum Consumer Groups per Broker	Maximum Client Connections per Broker	Storage Space	Traffic per Broker (MB/s)
kafka.2u4g.cluster	3–30	30,000	250	20	2000	300 GB–300,000 GB	100
kafka.4u8g.cluster	3–30	100,000	500	100	4000	300 GB–600,000 GB	200
kafka.8u16g.cluster	3–30	150,000	1000	150	4000	300 GB–900,000 GB	250
kafka.12u24g.cluster	3–30	200,000	1500	200	4000	300 GB–900,000 GB	375

Flavor	Brokers	Maximum TPS per Broker	Maximum Partitions per Broker	Maximum Consumer Groups per Broker	Maximum Client Connections per Broker	Storage Space	Traffic per Broker (MB/s)
kafka.16u32g.cluster	3–30	250,000	2000	200	4000	300 GB–900,000 GB	500

Instance Specifications and Network Bandwidth

The network bandwidth of a Kafka instance consists of the following:

1. Network bandwidth used by the instance brokers
2. Bandwidth of the disk used by the instance brokers. For details, see **Disk Types and Performance**.

Note:

- By default, Kafka tests are performed in the tail read scenario (that is, only the latest production data is consumed) instead of the cold read scenario (that is, historical data is consumed from the beginning).
- The bandwidth of an instance with an old flavor (such as 100 MB/s) is the total network bandwidth of the instance's all brokers.

Traffic calculation of instances with new flavors (such as kafka.2u4g.cluster) is described as follows:

- The read/write ratio is 1:1.
- The default number of topic replicas is 3.
- Total network traffic = Traffic per broker x Broker quantity
- Total instance traffic = Service traffic + Data replication traffic between brokers

Assume that the current flavor is kafka.2u4g.cluster, the traffic per broker is 100 MB/s, and the number of brokers is 3. What are the total network traffic, maximum read traffic, and maximum write traffic of the instance?

1. Total network traffic = Traffic per broker x Broker quantity = 100 MB/s x 3 = 300 MB/s
2. Maximum read traffic = Total instance network traffic/Default number of replicas/2 = 300 MB/s/3/2= 50 MB/s
3. Maximum write traffic = Total instance network traffic/Default number of replicas/2 = 300 MB/s/3/2 = 50 MB/s

Flavor Selection

- kafka.2u4g.cluster with 3 brokers

  Recommended for up to 6000 client connections, 60 consumer groups, and 90,000 TPS

- kafka.4u8g.cluster with 3 brokers

  Recommended for up to 12,000 client connections, 300 consumer groups, and 300,000 TPS

- kafka.8u16g.cluster with 3 brokers

  Recommended for up to 12,000 client connections, 450 consumer groups, and 450,000 TPS

- kafka.12u24g.cluster with 3 brokers

  Recommended for up to 12,000 client connections, 600 consumer groups, and 600,000 TPS

- kafka.16u32g.cluster with 3 brokers

  Recommended for up to 12,000 client connections, 600 consumer groups, and 750,000 TPS

Storage Space Selection

Kafka instances support multi-replica storage. The storage space is consumed by all replicas. When creating an instance, specify its storage space based on the expected service message size and the number of replicas.

For example, if the estimated message size is 100 GB, the disk capacity must be at least: 100 GB x Number of replicas + 100 GB (reserved space).

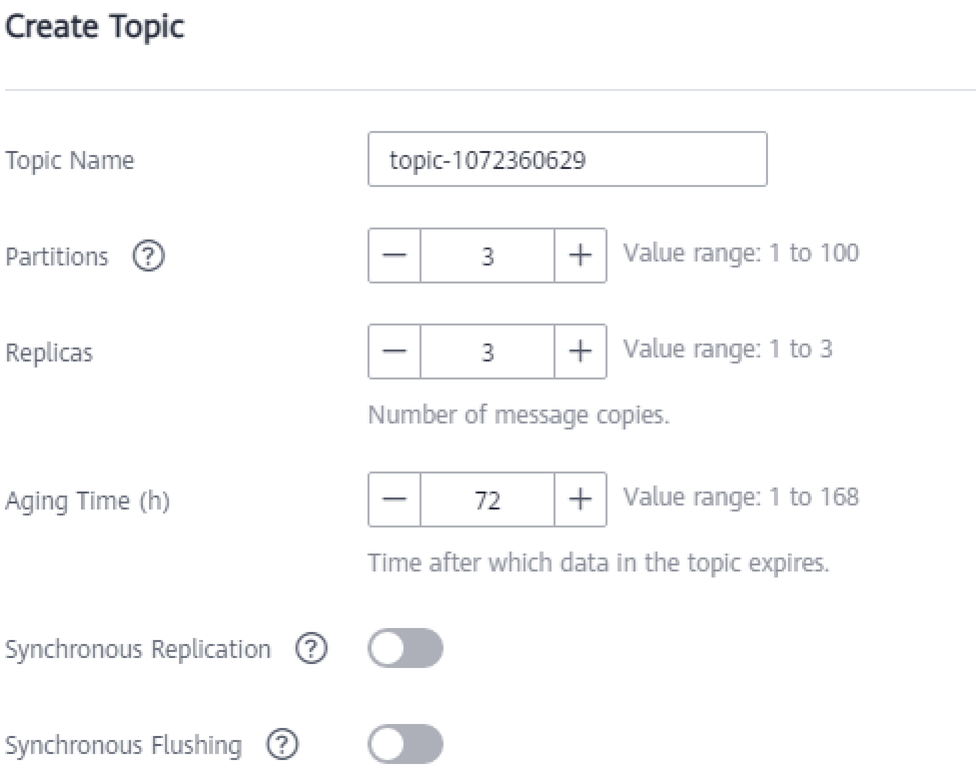
The storage space can be expanded as your service grows.

Topic Quantity

There are limits on the topic quantity and the aggregate number of partitions in the topics. When the partition quantity limit is reached, you can no longer create topics.

The number of topics is related to the maximum number of partitions allowed (see **Figure 4-1**) and the specified number of partitions in each topic (see **Table 4-1**).

Figure 4-1 Setting the number of partitions



Create Topic

Topic Name: topic-1072360629

Partitions ⓘ: 3 — Value range: 1 to 100

Replicas: 3 — Value range: 1 to 3

Number of message copies.

Aging Time (h): 72 — Value range: 1 to 168

Time after which data in the topic expires.

Synchronous Replication ⓘ: ○

Synchronous Flushing ⓘ: ○

The maximum number of partitions allowed for an instance with kafka. 2u4g.cluster and 3 brokers is 750.

- If the number of partitions of each topic in the instance is 3, the maximum number of topics is 750/3 = 250.
- If the number of partitions of each topic in the instance is 1, the maximum number of topics is 750/1 = 750.

5 Comparing Kafka, RabbitMQ, and RocketMQ

Feature	RocketMQ	Kafka	RabbitMQ
Priority queue	Not supported	Not supported	Supported. It is recommended that the priority be set to 0–10.
Delayed queue	Supported	Not supported	Supported
Dead letter queue	Supported	Not supported	Supported
Message retry	Supported	Not supported	Not supported
Retrieval mode	Pull-based and push-based	Pull-based	Pull-based and push-based
Message broadcasting	Supported	Supported	Supported
Message tracking	Supported	Supports offset and timestamp tracking.	Not supported. Once a message retrieval has been acknowledged, RabbitMQ will be notified that the message can be deleted.
Message accumulation	Supported	Supports higher accumulation performance than RabbitMQ thanks to high throughput.	Supported
Persistence	Supported	Supported	Supported

Feature	RocketMQ	Kafka	RabbitMQ
Message tracing	Supported	Not supported	Supported by the firehose feature or the rabbitmq_tracing plugin. However, rabbitmq_tracing reduces performance and should be used only for troubleshooting.
Message filtering	Supported	Supported	Not supported, but can be encapsulated.
Multi-tenancy	Supported	Not supported	Supported
Multi-protocol	Compatible with RocketMQ.	Only supports Apache Kafka.	RabbitMQ is based on AMQP and supports MQTT and STOMP.
Multi-language	Supports clients in multiple programming languages.	Kafka is written in Scala and Java and supports clients in multiple programming languages.	RabbitMQ is written in Erlang and supports clients in multiple programming languages.
Throttling	Planned	Supports throttling on producer or consumer clients.	Supports credit-based throttling on producers, a mechanism that triggers protection from within.
Ordered message delivery	Message order is maintained within a queue.	Supports partition-level FIFO.	Not supported. Supports FIFO only for single-threaded message queuing without advanced features such as delayed queues or priority queues.
Security	Supports SSL authentication.	Supports SSL and SASL authentication and read/write permissions control.	Similar to Kafka.
Transactional messages	Supported	Supported	Supported

6 Comparing DMS for Kafka and Open-Source Kafka

DMS for Kafka is compatible with open-source Kafka and has customized and enhanced Kafka features. In addition to the advantages of open-source Kafka, DMS for Kafka provides more reliable and useful features.

Table 6-1 Differences between DMS for Kafka and open-source Kafka

Category	Item	DMS for Kafka	Open-source Kafka
Ease of use	Readily available	Instances can be created intuitively within minutes and used right out of the box with visualized operations and real-time monitoring.	Preparing server resources and installing and configuring the software is time-consuming and prone to mistakes.
	APIs	Instances can be managed easily by calling RESTful APIs.	N/A
Costs	On-demand use	Multiple specifications are available to suit different needs. The instance broker quantity and disk space can be expanded without downtime.	Expenses are incurred for setting up a message service and occupying underlying resources.
	Fully managed	Services are readily available without requiring additional hardware resources or expenses.	Users must prepare hardware resources and set up the service by themselves, and bear high usage and maintenance costs.

Category	Item	DMS for Kafka	Open-source Kafka
Proven success	Mature	DMS has been deployed in many Huawei Cloud products and proven successful in large e-commerce events. It is also used in the clouds of carrier-grade customers across the world, and meets strict carrier-grade reliability standards. DMS closely follows up with community updates to continuously fix known open-source vulnerabilities and add support for new features.	Using open-source software requires lengthy self-development and verification and has had few successful cases.
	Feature-rich	While maintaining 100% open-source compatibility, DMS further optimizes open-source code to improve performance and reliability, and provides message querying, and many other features.	Functionality is limited and requires self-development.
Reliability	Highly available	DMS supports cross-AZ deployment to improve reliability. In addition, automatic fault detection and alarms ensure reliable operations of key services.	High availability requires self-development or open-source code implementation, which are costly and cannot guarantee reliability.
	Simple O&M	O&M is entirely transparent to tenants with a full set of monitoring and alarm functions. O&M personnel will be informed of any exceptions, eliminating the need for 24/7 attending.	Users need to develop and optimize O&M functions, especially alarm notification functions. Otherwise, manual attendance is required.
	Secure	DMS uses VPC isolation and SSL channel encryption.	Security must be hardened by users themselves.

7 Notes and Constraints

This section describes the notes and constraints on DMS for Kafka.

Instance

Table 7-1 Instance notes and constraints

Item	Notes and Constraints
Kafka ZooKeeper	Kafka clusters are managed using ZooKeeper. Opening ZooKeeper may cause misoperations and service losses. Currently, ZooKeeper is used only within Kafka clusters and does not provide services externally.
Version	● The service version can be 1.1.0, 2.3.0, or 2.7. Kafka instances cannot be upgraded once they are created. ● Clients later than version 0.10 are supported. Use a version that is consistent with the service version.
Logging in to the VM where the Kafka brokers reside	Not supported
Storage	● The storage space can be expanded but cannot be reduced. ● You can expand the storage space up to 20 times.
Broker quantity	The broker quantity can be increased but cannot be decreased.
VPC, subnet, and AZ	After an instance is created, its VPC, subnet, and AZ cannot be modified.
Kerberos authentication	Not supported

Item	Notes and Constraints
Client connections from each IP address	Each Kafka broker allows a maximum of 1000 connections from each IP address by default. Excess connections will be rejected.

Topic

Table 7-2 Topic notes and constraints

Item	Notes and Constraints
Total number of topic partitions	The total number of topic partitions is related to the instance specifications. For details, see **Specifications**. Kafka manages messages by partition. If there are too many partitions, message creation, storage, and retrieval will be fragmented, affecting the performance and stability. If the total number of partitions of topics reaches the upper limit, you cannot create more topics.
Number of partitions in a topic	Based on the open-source Kafka constraints, the number of partitions in a topic can be increased but cannot be decreased.
Topic quantity	The topic quantity is related to the total number of topic partitions and number of partitions in each topic. For details, see **Specifications**.
Automatic topic creation	Supported. If automatic topic creation is enabled, the system automatically creates a topic when a message is created in or retrieved from a topic that does not exist. This topic has the following default settings: 3 partitions, 3 replicas, aging time 72 hours, and synchronous replication and flushing disabled. After you change the value of the **log.retention.hours**, **default.replication.factor**, or **num.partitions** parameter, automatically created topics later use the new value. For example, if **num.partitions** is set to **5**, an automatically created topic will have the following settings: 5 partitions, 3 replicas, aging time 72 hours, and synchronous replication and flushing disabled.
Synchronous replication	If a topic has only one replica, synchronous replication cannot be enabled.

Item	Notes and Constraints
Replica quantity	Single-replica topics are not recommended. If an instance node is faulty, an internal service error may be reported when you query messages in a topic with only one replica. Therefore, you are not advised to use a topic with only one replica.
Aging time	The value of the **log.retention.hours** parameter takes effect only if the aging time has not been set for the topic. For example, if the aging time of Topic01 is set to 60 hours and **log.retention.hours** is set to 72 hours, the actual aging time of Topic01 is 60 hours.
Batch importing and exporting topics	Batch export is supported, but batch import is not supported.
Topic name	If a topic name starts with a special character, for example, a number sign (#), monitoring data cannot be displayed.
Delay queues	Not supported

Consumer Group

Table 7-3 Consumer group notes and constraints

Item	Notes and Constraints
Creating consumer groups, consumers, and producers	Consumer groups, consumers, and producers are generated automatically when you use the instance.
Resetting the consumer offset	Messages may be retrieved more than once after the offset is reset.
Consumer group name	If a consumer group name starts with a special character, for example, a number sign (#), monitoring data cannot be displayed.

Message

Table 7-4 Message notes and constraints

Item	Notes and Constraints
Message size	The maximum length of a message is 10 MB. If the length exceeds 10 MB, the production fails.

User

Table 7-5 User notes and constraints

Item	Notes and Constraints
Number of users	A maximum of 20 SASL_SSL users can be created for a Kafka instance.

8 Related Services

- Cloud Trace Service (CTS)

  CTS generates traces to provide you with a history of operations performed on cloud service resources. The traces include operation requests sent using the management console or open APIs, as well as the operation results. You can view all generated traces to query, audit, and backtrack performed operations.

  For details about the operations recorded by CTS, see **Operations Logged by CTS**.

- Virtual Private Cloud (VPC)

  Kafka instances run in VPCs and use the IP addresses and bandwidth of VPC. Security groups of VPCs enhance the security of network access to the Kafka instances.

- Elastic Cloud Server (ECS)

  An ECS is a basic computing unit that consists of vCPUs, memory, OS, and EVS disks. Kafka instances run on ECSs. A broker corresponds to an ECS.

- Elastic Volume Service (EVS)

  EVS provides block storage services for ECSs. All Kafka data, such as messages, metadata, and logs, is stored in EVS disks.

- Cloud Eye

  Cloud Eye is an open platform that provides monitoring, alarm reporting, and alarm notification for your resources in real time.

  NOTE

  The values of all Kafka instance metrics are reported to Cloud Eye every minute.

- Elastic IP (EIP)

  The EIP service provides independent public IP addresses and bandwidth for Internet access. Kafka instances bound with EIPs can be accessed over public networks.

- Tag Management Service (TMS)

  TMS is a visualized service for fast and unified cross-region tagging and categorization of cloud services.

  Tags facilitate Kafka instance identification and management.

9 Basic Concepts

DMS for Kafka of Huawei Cloud uses Kafka as the message engine. This chapter presents explanations of basic concepts of Kafka.

Topic

A topic is a category for messages. Messages are created, retrieved, and managed in the form of topics.

Topics adopt the publish-subscribe pattern. Producers publish messages into topics. One or more consumers subscribe to the messages in the topics. The producers and consumers are not directly linked to each other.

Producer

A producer publishes messages into topics. The messages are then delivered to other systems or modules for processing as agreed.

Consumer

A consumer subscribes to messages in topics and processes the messages. For example, a monitoring and alarm platform (a consumer) subscribing to log messages in certain topics can identify alarm logs and then send SMS or email alarm notifications.

Broker

A broker is a Kafka process in a Kafka cluster. Each process runs on a server, so a broker includes the storage, bandwidth, and other server resources.

Partition

A topic is divided into partitions. Messages are distributed to multiple partitions to achieve scalability and fault tolerance.

Replica

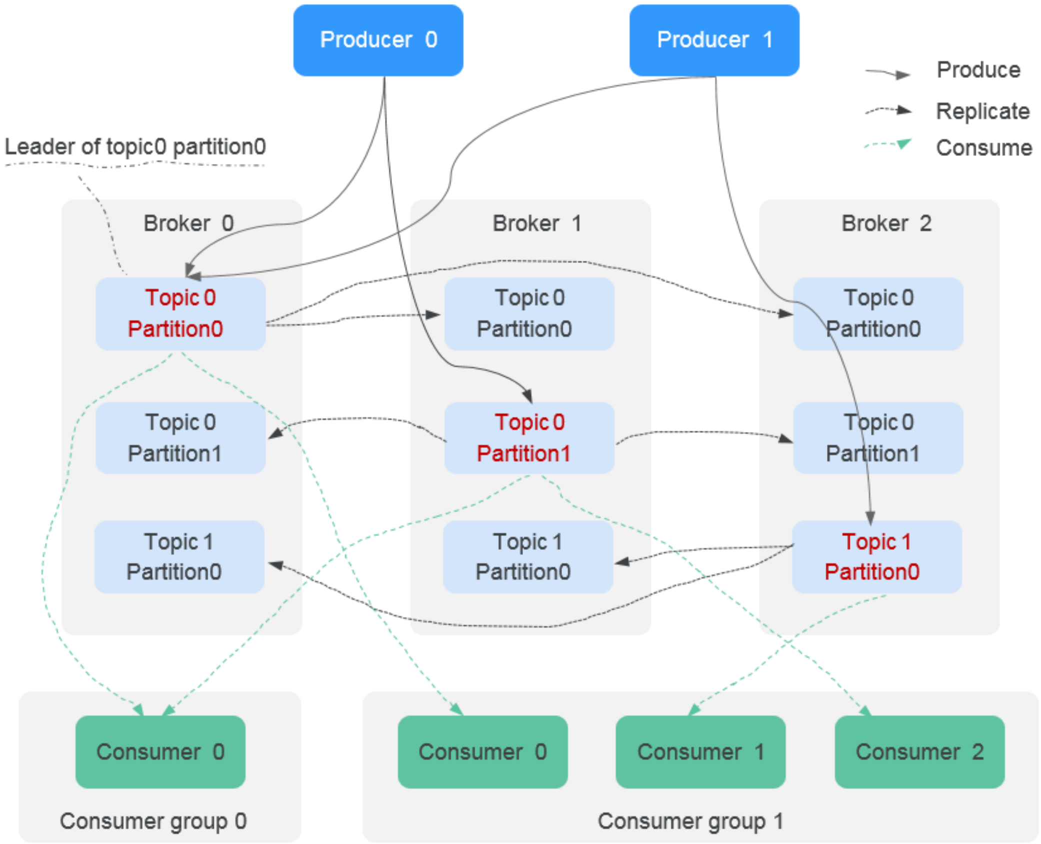
A replica is a redundant copy of a partition in a topic. Each partition can have one or more replicas, enabling message reliability.

Messages in each partition are fully replicated and synchronized, preventing data loss if one replica fails.

Each partition has one replica as the leader which handles the creation and retrievals of all messages. The rest replicas are followers which replicate the leader.

Topics and partitions are logical concepts, while replicas and brokers are physical concepts. The following diagram shows the relationships between partitions, brokers, and topics in messages streaming.

Figure 9-1 Kafka message streaming



Aging Time

The period that messages are retained for. Consumers must retrieve messages before this period ends. Otherwise, the messages will be deleted and can no longer be retrieved.

10 Permissions Management

You can use Identity and Access Management (IAM) to manage DMS for Kafka permissions and control access to your resources. IAM provides identity authentication, permissions management, and access control.

You can create IAM users for your employees, and assign permissions to these users on a principle of least privilege (PoLP) basis to control their access to specific resource types. For example, you can create IAM users for software developers and assign specific permissions to allow them to use Kafka instance resources but prevent them from being able to delete resources or perform any high-risk operations.

If your HUAWEI ID does not require individual IAM users for permissions management, skip this section.

IAM can be used free of charge. You pay only for the resources in your account.

For more information, see **IAM Service Overview**.

NOTE

Permissions policies of DMS for Kafka are based on DMS. Therefore, when assigning permissions, select DMS permissions policies.

DMS for Kafka Permissions

By default, new IAM users do not have any permissions assigned. To assign permissions to these new users, add them to one or more groups, and attach permissions policies or roles to these groups.

DMS for Kafka is a project-level service deployed and accessed in specific physical regions. When assigning DMS for Kafka permissions to a user group, specify region-specific projects where the permissions will take effect. If you select **All projects**, the permissions will be granted for all region-specific projects. When accessing DMS for Kafka, the users need to switch to a region where they have been authorized to use this service.

You can grant permissions by using roles and policies.

- Roles: A type of coarse-grained authorization mechanism that provides only a limited number of service-level roles. When using roles to grant permissions, you also need to assign dependency roles. However, roles are not an ideal choice for fine-grained authorization and secure access control.

- Policies: A type of fine-grained authorization mechanism that defines permissions required to perform operations on specific cloud resources under certain conditions. This mechanism allows for more flexible policy-based authorization for securer access control. For example, you can grant DMS for Kafka users only the permissions for managing instances. Most policies define permissions based on APIs. For the API actions supported by DMS for Kafka, see **Permissions Policies and Supported Actions**.

Table 10-1 lists all the system-defined roles and policies supported by DMS for Kafka.

Table 10-1 System-defined roles and policies supported by DMS for Kafka

Role/Policy Name	Description	Type	Dependency
DMS FullAccess	Administrator permissions for DMS. Users granted these permissions can perform all operations on DMS.	System-defined policy	None
DMS UserAccess	Common user permissions for DMS, excluding permissions for creating, modifying, deleting, and scaling up instances.	System-defined policy	None
DMS ReadOnlyAccess	Read-only permissions for DMS. Users granted these permissions can only view DMS data.	System-defined policy	None
DMS VPCAccess	VPC operation permissions to assign to DMS agencies.	System-defined policy	None
DMS KMSAccess	KMS operation permissions to assign to DMS agencies.	System-defined policy	None
DMS Administrator	Administrator permissions for DMS.	System-defined role	This role depends on the **Tenant Guest** and **VPC Administrator** roles.

NOTE

System-defined policies contain OBS actions. Due to data caching, the policies take effect five minutes after they are attached to a user, user group, or enterprise project.

Table 2 lists the common operations supported by each DMS for Kafka system policy. Select the policies as required.

Table 10-2 Common operations supported by each system-defined policy of DMS for Kafka

Operation	DMS FullAccess	DMS UserAccess	DMS ReadOnlyAccess
Creating instances	√	×	×
Modifying instances	√	×	×
Deleting instances	√	×	×
Modifying instance specifications	√	×	×
Restarting instances	√	√	×
Querying instance information	√	√	√

Fine-grained Authorization

To use a custom fine-grained policy, log in to the IAM console as an administrator and select the desired fine-grained permissions for DMS. **Table 10-3** describes fine-grained permission dependencies of DMS for Kafka.

Table 10-3 Fine-grained permission dependencies of DMS for Kafka

Permission	Description	Dependency
dms:instance:get	Viewing instance details	None
dms:instance:getConnectorSinkTask	Viewing dumping task details	None
dms:instance:getBackgroundTask	Viewing background task details	None
dms:instance:modifyAuthInfo	Changing an instance password	None
dms:instance:resetAuthInfo	Resetting an instance password	None

Permission	Description	Dependency
dms:instance:scale	Scaling up an instance	● vpc:vpcs:get ● vpc:ports:create ● vpc:securityGroups:get ● vpc:ports:get ● vpc:subnets:get ● vpc:vpcs:list ● vpc:publicIps:get ● vpc:publicIps:list ● vpc:ports:update ● vpc:publicIps:update
dms:instance:connector	Enabling dumping	● vpc:vpcs:get ● vpc:ports:create ● vpc:securityGroups:get ● vpc:ports:get ● vpc:subnets:get ● vpc:vpcs:list ● vpc:publicIps:get ● vpc:publicIps:list ● vpc:ports:update ● vpc:publicIps:update
dms:instance:deleteConnectorSinkTask	Deleting a dumping task	None
dms:instance:modify	Modifying an instance	● vpc:vpcs:get ● vpc:ports:create ● vpc:securityGroups:get ● vpc:ports:get ● vpc:subnets:get ● vpc:vpcs:list ● vpc:publicIps:get ● vpc:publicIps:list ● vpc:ports:update ● vpc:publicIps:update
dms:instance:deleteBackgroundTask	Deleting a background task	None
dms:instance:modifyStatus	Restarting an instance	None
dms:instance:createConnectorSinkTask	Creating a dumping task	None

Permission	Description	Dependency
dms:instance:delete	Deleting an instance	None
dms:instance:create	Creating an instance	- vpc:vpcs:get - vpc:ports:create - vpc:securityGroups:get - vpc:ports:get - vpc:subnets:get - vpc:vpcs:list - vpc:publicIps:get - vpc:publicIps:list - vpc:ports:update - vpc:publicIps:update
dms:instance:listConnectorSinkTask	Viewing the dumping task list	None
dms:instance:list	Viewing the instance list	None

Helpful Links

- **What Is IAM?**
- **Creating a User and Granting DMS for Kafka Permissions**
- **Permissions Policies and Supported Actions**

11 Billing

Huawei Cloud DMS for Kafka supports pay-per-use. For details, see **Pricing Details**.

Billing Items

Huawei Cloud DMS for Kafka is billed based on Kafka instance specifications and storage space.

Table 11-1 DMS for Kafka billing

Billing Item	Description
Instance	● Kafka instances are billed based on their ECS flavor and broker quantity. When purchasing an instance, select appropriate ECS flavors and the number of brokers based on service evaluation. **Table 11-2** lists the performance per broker. ● Kafka instances can be billed on a pay-per-use (hourly) basis.

Billing Item	Description
Storage space	● Instances are billed based on the storage space. For each type of instance specification, you can choose the high I/O or ultra-high I/O disk type to meet your service requirements. You can specify the number of replicas. For example, if the disk size required to store message data is 500 GB and there are three replicas, the disk capacity should be at least: 500 GB x 3 = 1500 GB. ● Storage space can be specified with increments of 100 GB. For details about the storage space range, see **Table 11-2**. ● The storage space can be billed on a pay-per-use (hourly) basis.

Table 11-2 Kafka instance specifications

Flavor	Brokers	Maximum TPS per Broker	Maximum Partitions per Broker	Maximum Consumer Groups per Broker	Maximum Client Connections per Broker	Storage Space	Traffic per Broker (MB/s)
kafka. 2u4g.cluster	3–30	30,000	250	20	2000	300 GB–300,000 GB	100
kafka. 4u8g.cluster	3–30	100,000	500	100	4000	300 GB–600,000 GB	200
kafka. 8u16g.cluster	3–30	150,000	1000	150	4000	300 GB–900,000 GB	250
kafka. 12u24g.cluster	3–30	200,000	1500	200	4000	300 GB–900,000 GB	375

Flavor	Brokers	Maximum TPS per Broker	Maximum Partitions per Broker	Maximum Consumer Groups per Broker	Maximum Client Connections per Broker	Storage Space	Traffic per Broker (MB/s)
kafka.16u32g.cluster	3–30	250,000	2000	200	4000	300 GB–900,000 GB	500

Billing Modes

Pay-per-use (hourly) mode: More flexible, enabling you to start and stop services anytime. You pay only for what you use. The minimum time unit is one hour. Less than an hour is recorded as an hour.

Changing Configurations

- You can change the number of brokers for a Kafka instance. You will then be billed based on the new specifications immediately after the change.
- You can also change the storage space of Kafka. You will be billed based on the new storage space immediately after the storage space increase. Storage space can only be increased, and cannot be decreased. The minimum increment is 100 GB.