TDMQ for CKafka

Product Introduction

Product Documentation

Contents

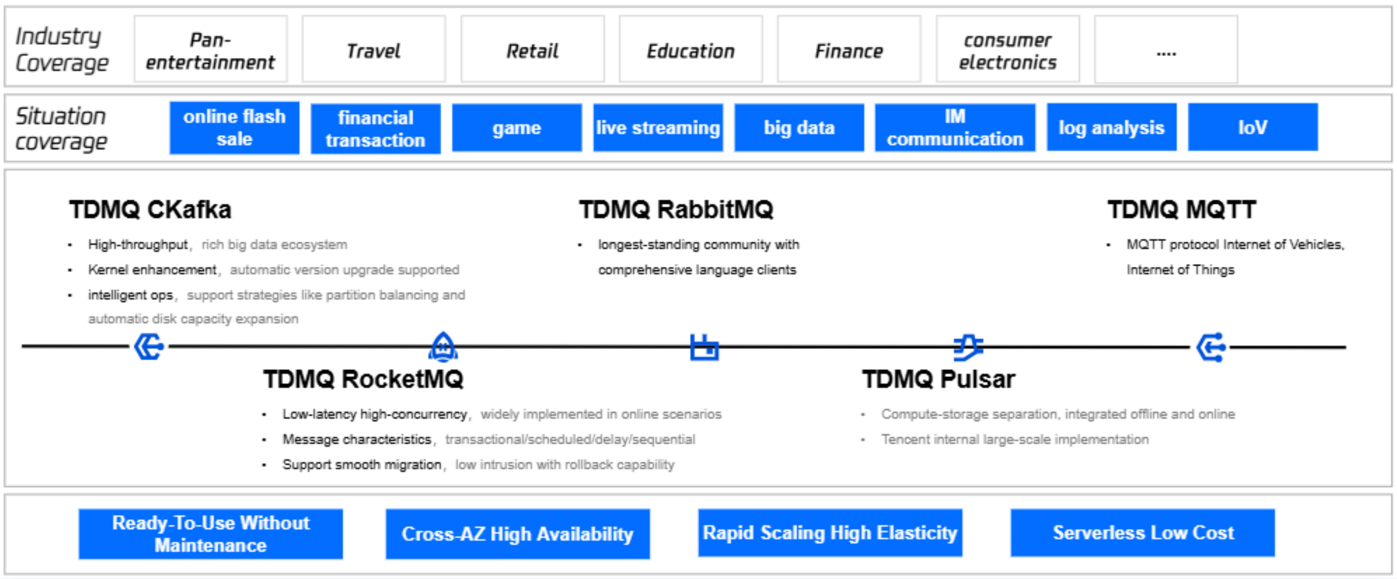
Product Introduction
TDMQ Product Series Introduction and Selection

Last updated：2025-09-19 17:35:32

Product Overview

TDMQ (Tencent Distributed Message Queue) is a middleware product series independently developed by Tencent Cloud. As a key component in distributed systems, it possesses **stable, reliable, high elasticity, and low-cost** characteristics, providing **asynchronous communication** as a foundational capability. Through **application decoupling**, it reduces system complexity and improves system availability and scalability.

- **Compatible with open-source mainstream protocols**, including five subproducts: CKafka, RocketMQ, RabbitMQ, Pulsar, and MQTT. Offers migration solution support with zero business code modification to reduce migration costs.
- Cover **online scenarios (e-commerce transactions, social live streaming)**, **offline scenarios (big data real-time compute, offline analysis)**, and **device-side scenarios (IoT, IoV)**, meeting the needs of **pan-entertainment, education, retail, travel, finance, healthcare** and other industries.



Advantages

Ready-To-Use Without Maintenance

TDMQ offers an out-of-the-box fully hosted service. Users can create a cluster with just one click, completely saying goodbye to cumbersome installation and deployment work. The optimized resource management interface, comprehensive monitoring metrics, and intelligent diagnostic tool drastically reduce Ops complexity and management costs.

Cross-AZ High Availability

TDMQ builds a comprehensive disaster recovery system through multi-layer technical measures. It implements a cross-AZ deployment architecture to prevent idc-level fault risks. With rate limiting protection policies, it dynamically adjusts traffic pressure to ensure cluster health. It also provides cross-cluster data replication capability, meeting various high availability scenario requirements from basic backup to multi-active deployment.

Rapid Scaling High Elasticity

TDMQ offers ultimate auto scaling capability. With one-click operation, rapid scaling of resources is achievable. The seamless adjustment of underlying cloud resources remains business-agnostic throughout the process, enabling easy coping with various emergent traffic scenarios.

Serverless Low Cost

TDMQ adopts a storage-compute separation architecture. The computing layer supports second-level auto scaling and can handle sudden traffic spikes without pre-allocated resources, maximizing resource usage rate. Meanwhile, the storage layer is infinitely scalable and used on a pay-as-you-go basis, reducing storage costs by 30%-50%.

Product Comparison

TDMQ can provide the appropriate product form and solution for different customer scenarios and needs. If you have any requirement, contact us for consultation.

Product	Product features	Product Strengths	Use Cases	Applicable Business
CKafka	• High-throughput, rich big data ecosystem • Kernel enhancement, automatic version upgrade supported • intelligent ops, support strategies like partition balancing and	High-throughput benchmark, stable performance, extensively applied	High-throughput offline scenarios	Collection of compressed logs, monitoring data aggregation, streaming data integration

	automatic disk capacity expansion			
Rocket MQ	• Low-latency high-concurrency, widely implemented in online scenarios • Message characteristics, transactional/scheduled/delay/sequential • Support smooth migration, low intrusion with rollback capability	Massive message accumulation, low delay, high-throughput, high reliability	High-reliability, low-latency online business scenario	async decoupling, peak shaving and valley filling, sequential message sending/receiving, distributed transaction consistency
Rabbit MQ	• Longest-standing community with comprehensive language clients	100% compatibility with open-source, providing flexible routing modes	Suitable for small and medium-sized online business scenarios	Flash sales, priority messaging, delayed message delivery, message broadcasting
Pulsar	• Compute-storage separation, integrated offline and online • Tencent internal large-scale implementation	Compute-storage separation, flexible scaling	Compatible with online and offline scenarios	Async decoupling, peak shaving and valley filling, sequential message sending/receiving, data sync
MQTT	• MQTT protocol Internet of Vehicles, Internet of Things	Lightweight protocol, supports a million clients online simultaneously	Business message delivery suitable for various mobile devices	Internet of Vehicles, Industrial Internet, IM communication, smart home

What Is TDMQ for CKafka

Last updated: 2025-09-19 17:35:32

TDMQ for CKafka Overview

TDMQ for CKafka is a distributed, high-throughput, highly scalable messaging system that is 100% compatible with open-source Kafka API versions 2.4, 2.8, and 3.2. Based on a publish/subscribe model, CKafka enables producers and consumers to interact asynchronously without waiting for each other through message decoupling. It has the advantages of high availability, data compression, and support for both offline and real-time data processing. It is suitable for scenarios such as log compression collection, monitoring data aggregation, and streaming data integration.

Features

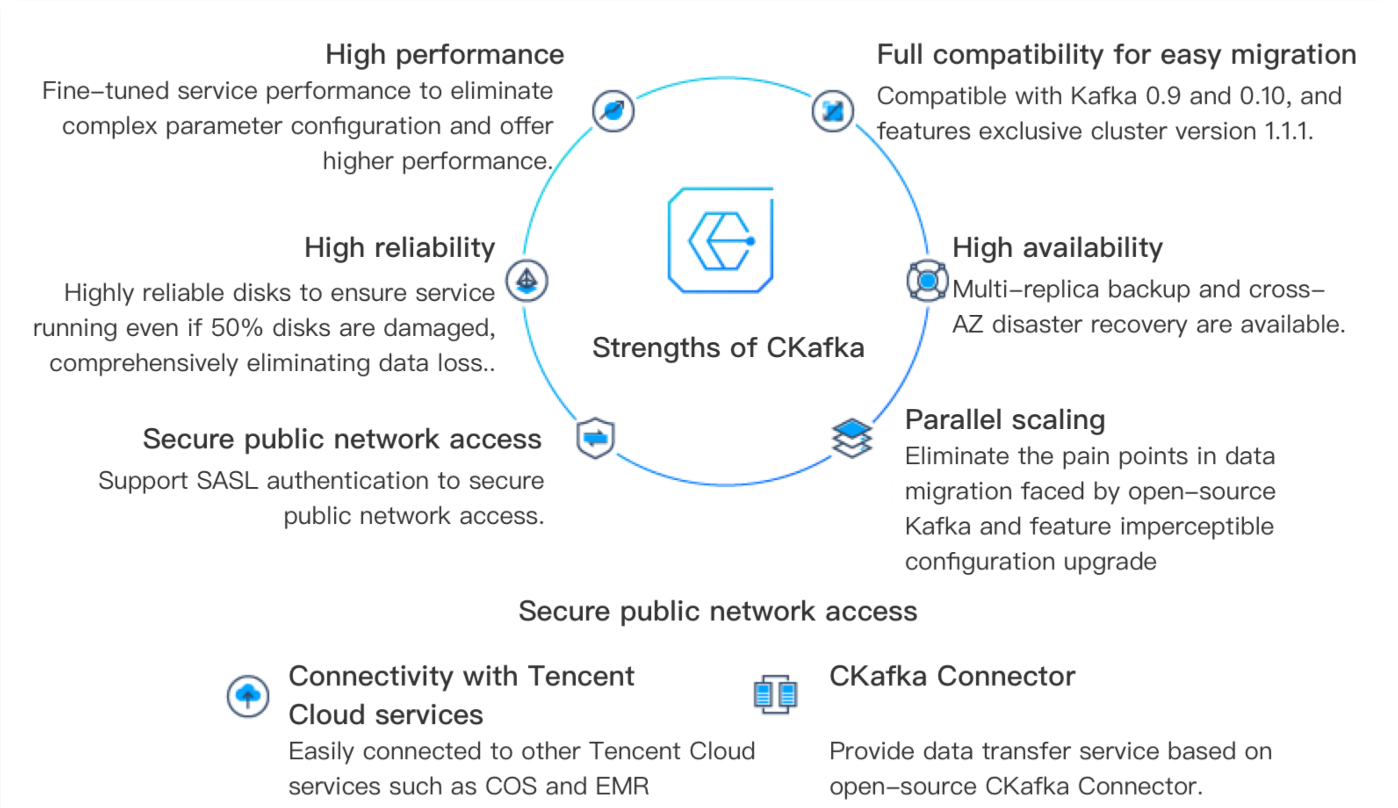
- **Decoupling producers and consumers**
  TDMQ for CKafka effectively decouples the relationship between producers and consumers, allowing for independent scaling or modification of the processing flow between producers and consumers while ensuring the same API constraints.
- **Peak shifting and valley filling**
  TDMQ for CKafka can withstand access traffic surges instead of completely crashing due to sudden overwhelming requests, which effectively boosts system robustness.
- **Sequential read/write**
  TDMQ for CKafka ensures the orderliness of messages within a partition. Like most message queues, it guarantees that data is processed in sequence, greatly improving disk efficiency.
- **Async communication**
  In scenarios where immediate message processing is not required, TDMQ for CKafka provides an asynchronous message processing mechanism. During high traffic periods, messages are simply placed in the queue and processed later when the traffic decreases, alleviating system pressure.

> **Note:**
> TDMQ for CKafka supports private deployment. You can submit a ticket for consultation.

Strengths

Last updated：2025-09-19 17:35:32

This document describes the strengths of TDMQ for CKafka compared to Apache Kafka.



Strengths

Full compatibility with Apache Kafka and easy migration

- TDMQ for CKafka is compatible with Kafka versions 2.4、2.8 and 3.2.
- Business systems using TDMQ for CKafka can migrate to the cloud without any changes, based on existing open-source Apache Kafka client-side code, and enjoy the high-performance message queue service provided by Tencent Cloud.

High performance

- TDMQ for CKafka team has further fine-tuned the service performance to eliminate complex parameter configuration and offer higher performance.
- TDMQ for CKafka offers UI-based upgrade and scaling, as well as a powerful IaaS layer that provides 10% to 20% more production capabilities than Apache Kafka.

High availability

- Leveraging Tencent's years of experience in monitoring technologies, TDMQ for CKafka provides comprehensive monitoring on clusters. Additionally, our professional operations team is available 24/7 to handle alerts, ensuring the high availability.
- Custom multi-AZ deployments in the same region is supported to improve disaster recovery ability.

High reliability

- The disks used are highly reliable, and the service will not be affected even if 50% disks are damaged.
- There are 2 replicas by default, and 3 replicas can be supported. The more replicas, the higher the reliability.

Parallel scaling

- When cluster traffic and disk capacity exceed the alarm threshold, auto-scaling is supported with the client unaffected.
- TDMQ for CKafka eliminates the pain points in data migration faced by open-source Kafka and features imperceptible configuration upgrade.

Data security

- TDMQ for CKafka provides various security features such as authentication, authorization, root account, and sub-account, enabling enterprise-level security protection.
- VPC: Access from more secure VPC is supported.
- SASL authentication is supported to secure public network access.
- Root account and sub-account: CAM features such as root account, sub-account, and collaborator are fully supported, enabling authorization between root account and sub-account as well as across organizational accounts.

Connectivity with Related Services

Integration with cloud services

TDMQ for CKafka can be easily connected to other Tencent Cloud services such as COS and EMR.

TDMQ for CKafka Connector

CKafka supports data transfer with open-source Kafka Connect, so that data can be exchanged between two Kafka clusters.

Use Cases

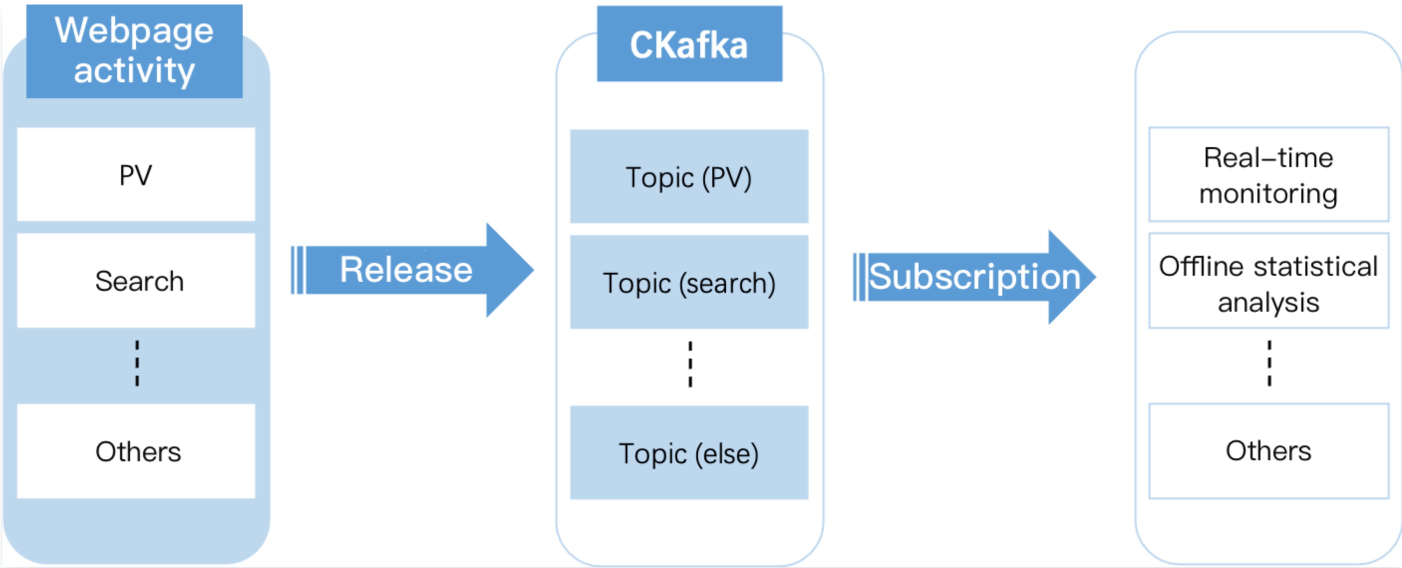
Last updated：2025-09-19 17:35:32

TDMQ for CKafka is widely used in big data scenarios, such as webpage tracking, behavior analysis, log aggregation, monitoring, streaming data processing, and online and offline data analysis.
You can simplify data integration in the following ways:

- Import messages from TDMQ for CKafka into COS, Stream Compute Service, and other data warehouses.
- Connect with other Tencent cloud products using Serverless Cloud Functions triggers.

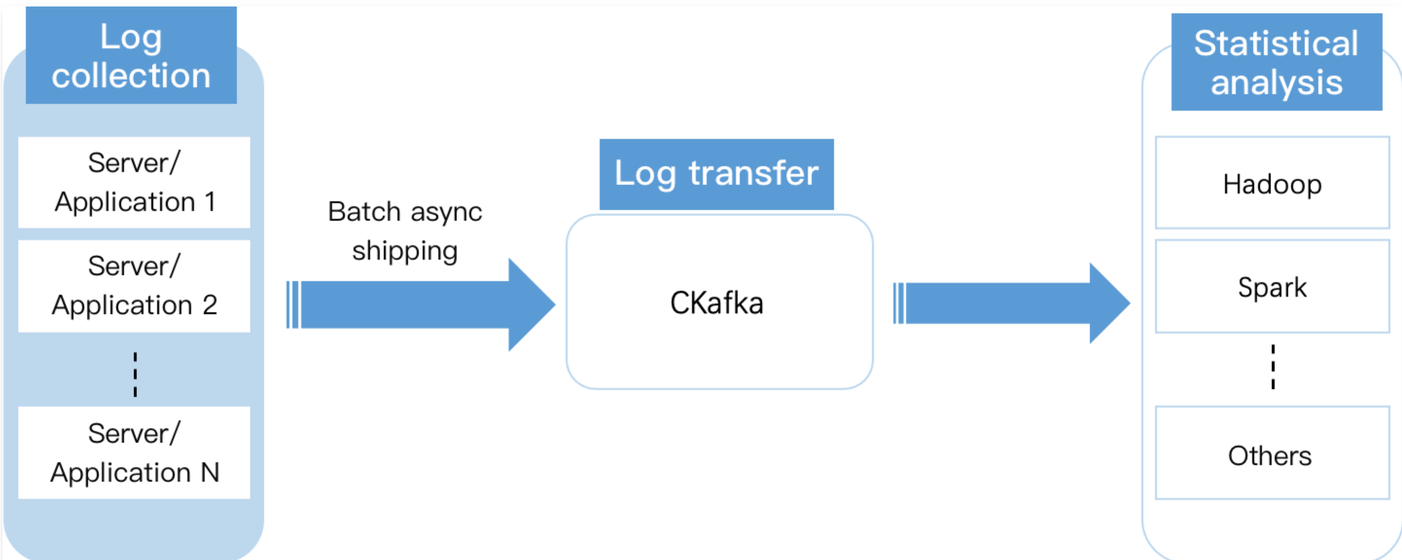
Web Tracking

TDMQ for CKafka can process website activities such as PV, search, and other user behaviors in real time and then publishes them to topics by type. These information flows can be used for real-time monitoring or offline statistical analysis.
As a large amount of activity information is generated in each user's page views, website activity tracking requires a very high throughput. TDMQ for CKafka can meet the requirements of high throughput and offline processing.



Log aggregation

The low-latency processing capability of TDMQ for CKafka makes it easier to process (consume) distributed data from multiple data sources. Under the same performance conditions, TDMQ for CKafka provides more durable persistent storage and lower end-to-end latency than a centralized data aggregation system.
The above features make TDMQ for CKafka clusters an ideal log collection center. Multiple servers and applications can asynchronously send operation logs in batches to TDMQ for CKafka clusters instead of saving them locally or in a database. TDMQ for CKafka clusters can submit and compress messages in batches, making the performance overhead almost negligible for producers. Uers can use systematic storage and analysis systems such as Hadoop to analyze the pulled logs.



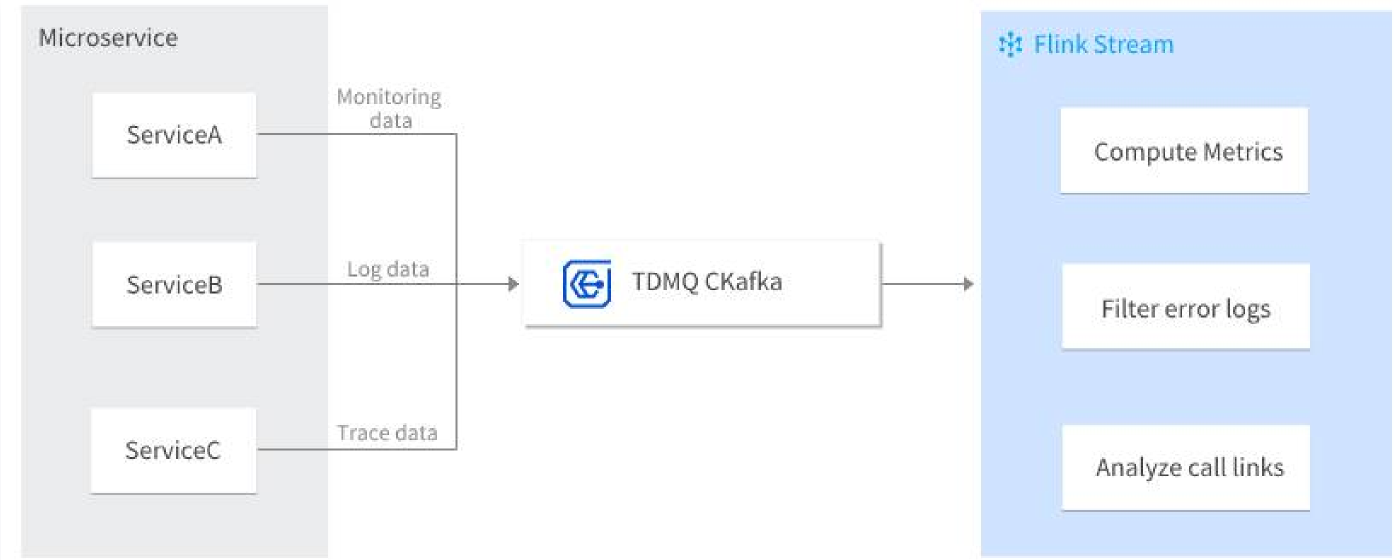
Big data

In some big data scenarios, a large amount of concurrent data needs to be processed and aggregated. This requires clusters to have excellent processing performance and high scalability. Moreover, TDMQ for CKafka clusters' data distribution mechanism, in terms of disk space allocation, message format processing, server

selection, and data compression, also makes them suitable for handling high numbers of real-time messages and aggregating distributed application data, which facilitates system OPS.
TDMQ for CKafka clusters can better aggregate, process, and analyze offline and streaming data.

User Link Observation

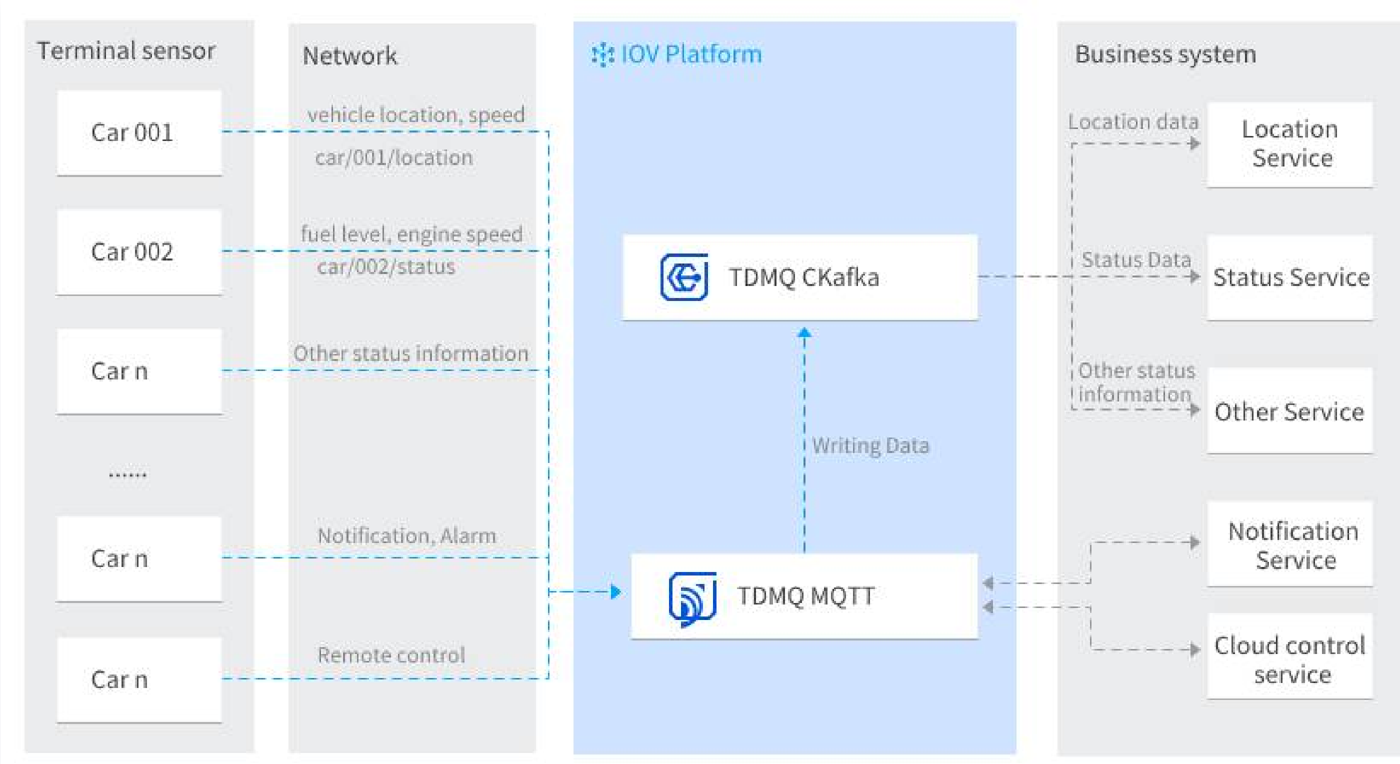
In a typical microservices architecture, the system consists of multiple independent services. Each service generates a large number of monitoring data (for example, CPU and memory usage), log data (for example, request logs and error logs), and Trace data (for example, service invocation links). To achieve comprehensive observability of the system, these data can be uniformly collected and sent to Kafka. Downstream consumes these data in real time through Flink Stream for aggregation, analysis, and anomaly detection, helping the Ops team quickly detect and troubleshoot problems, enhancing system stability and maintainability.



IoT Data Collection and Distribution

In an IoT scene, devices deliver device data to CKafka via the MQTT protocol and are distributed to different systems by the rule engine for further processing. For example, vehicles collect various information through sensors and controllers installed on them, such as vehicle location, speed, fuel level, engine status, etc. This information needs to be transmitted to the car manufacturer's server in real time or periodically so that data analysis, failure alarming, remote control and other operations can be performed.

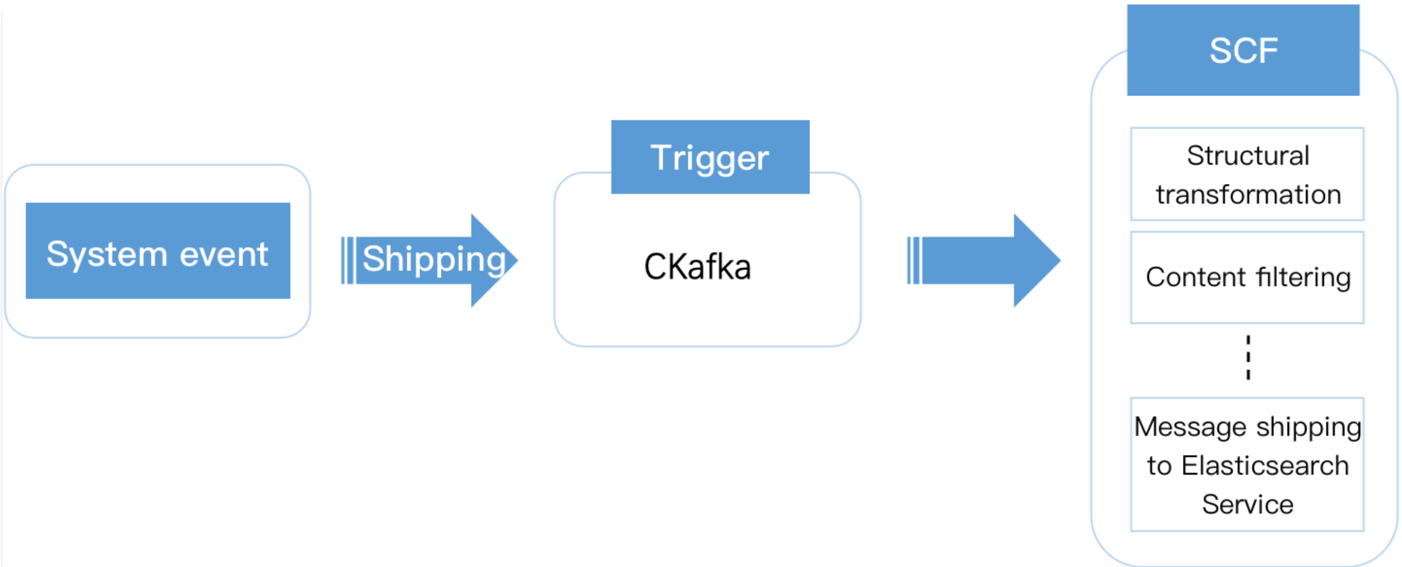
- Terminal devices access the MQTT Version of the Message Queue via the MQTT protocol and connect to the CKafka cluster through the rule engine to achieve data forwarding.
- IoV service platform, high-precision map service, location service and other IoV related applications can directly consume by subscribing to CKafka data. At the same time, two-way communication of vehicle control (remote control) messages can be realized through the MQTT Version of the Message Queue.



Serverless Cloud Functions triggers

TDMQ for CKafka can be used as SCF triggers, and when a message is received, a function can be triggered and the message will be passed to the function as event content. For example, when CKafka triggers a function, the function can transform the message structure, filter the message contents, or deliver the message to Elasticsearch Service (ES).

> **Note:**
> For more information on the availability of SCF, see Service Level Agreement for SCF.

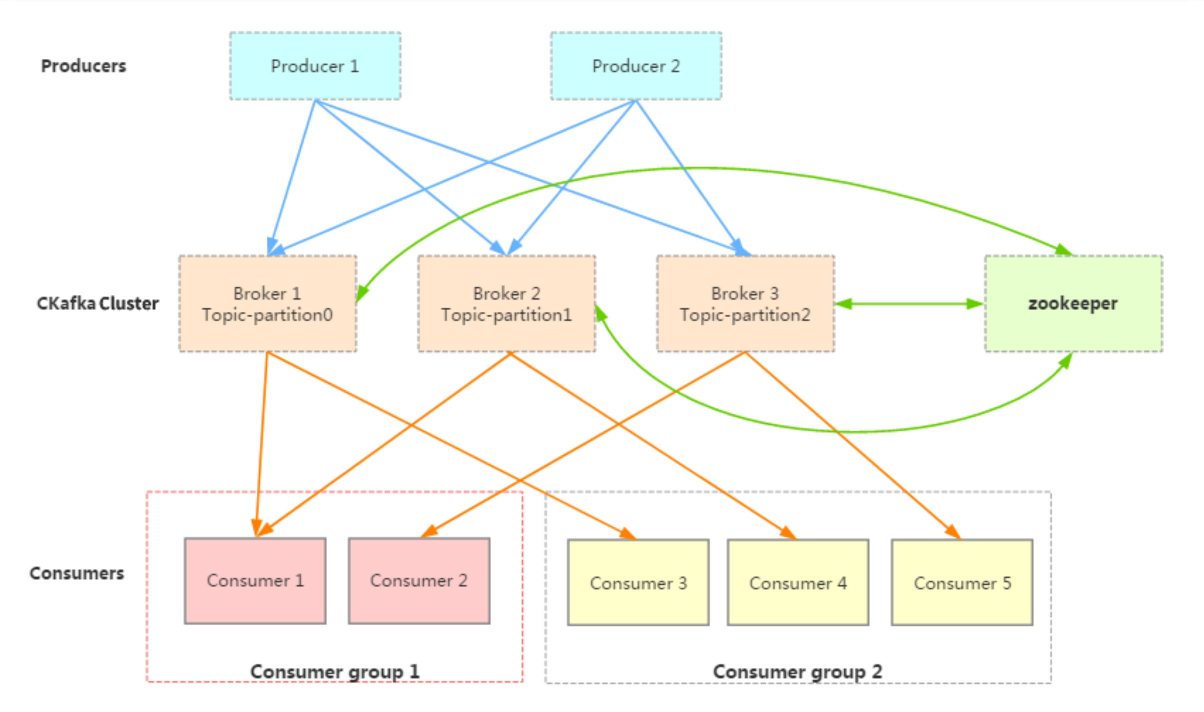

System event
Shipping
Trigger
CKafka
SCF
Structural transformation
Content filtering
Message shipping to Elasticsearch Service

Technology Architecture

Last updated：2025-09-19 17:35:32

Architecture Description

The architecture diagram of CKafka is as follows:



Producer

Producers of messages might generate messages such as those generated by web page activities, service logs, etc. Producers publish messages to the Broker cluster of CKafka in push model.

CKafka Cluster

- **Broker:** Server that stores messages. Brokers support horizontal scaling. The more nodes there are, the higher the cluster throughput rate.
- **Partition:** Partition. A Topic can have multiple partitions. Each Partition physically corresponds to a folder used to store the messages and index files in this Partition. A Partition can have multiple replicas to improve availability, but it will also increase storage and network overhead.
- **ZooKeeper:** Responsible for cluster configuration management, leader election, fault tolerance, etc.

Consumer

Message consumers. A Consumer is divided into several Consumer groups. Consumers consume messages from the Broker in pull mode.

Architecture Principle Explanation

High-Throughput

In the CKafka, there is a process where a large amount of network data is persisted to disks and disk files are sent over the network. The performance of this process directly affects the overall throughput of CKafka, mainly achieved through the following points:

- **Efficient disk usage**: Read and write data sequentially on the disk to improve disk utilization.
- Writing messages: Messages are written to the page cache and flushed to disk by asynchronous threads.
- Reading messages: Messages are directly transferred from the page cache to the socket and sent out.
- When the corresponding data is not found in the page cache, disk I/O will occur. The messages will be loaded from the disk into the page cache and then sent directly from the socket.
- **Zero-Copy Mechanism of Broker**: Use the sendfile system call to send data directly from page cache to network.
- **Reduce network overhead**
- Compress data to reduce network load.
- Batch processing mechanism: The producer writes data to the Broker in batches, and the consumer pulls data from the Broker in batches.

Data Persistence

Data persistence of the CKafka is mainly implemented by the following principles:

- **Storage distribution of Partitions in a Topic**
- In the file storage of CKafka, there are multiple different Partitions in the same Topic. Each Partition physically corresponds to a folder, used to store messages and index files of that Partition. For example, if you create two Topics, Topic1 has 5 Partitions and Topic2 has 10 Partitions, then 5 + 10 = 15 folders will be generated on the entire cluster accordingly.
- **File storage method in a Partition**
- Physically, a Partition consists of multiple segments. Each segment has an equal size. Sequential read/write operations are performed. Expired segments are quickly deleted to improve disk usage.

Horizontal Expansion (Scale Out)

- A topic can include multiple partitions, which can be distributed on one or multiple brokers.
- A consumer can subscribe to one or more Partitions.
- A producer is responsible for evenly distributing messages to the corresponding partitions.
- Messages in partitions are sequential.

Consumer Group

- CKafka does not delete consumed messages.

- Every consumer should belong to a group.
- Multiple consumers in the same Consumer Group do not consume the same Partition simultaneously.
- Different groups can consume the same message at the same time, supporting diversified processing (for queue mode and release/subscription mode).

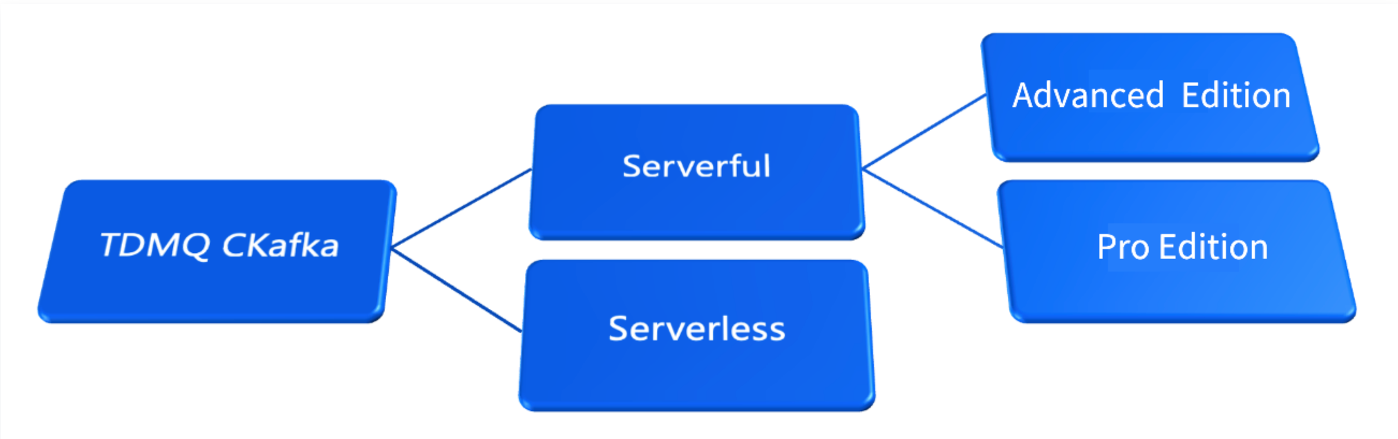
TDMQ Product Series Introduction and Selection
Capability Comparison

Last updated：2025-09-19 17:35:32

To meet user needs in different usage scenarios, TDMQ for CKafka version instance provides multiple forms `Serverful` and `Serverless` for users in various scenarios. When you proceed to purchase a CKafka cluster, it is advisable to make a product selection judgment based on comprehensive factors such as business application scenarios, product capabilities, and use cost.

Product series

Learn about TDMQ for CKafka product series in one chart.



- Serverful, the classic form of CKafka, allows users to purchase clusters of corresponding specifications as needed. With business growth, it is necessary to maintain certain focus on the CKafka cluster.
- Serverless, the brand-new form of CKafka, aims to completely release user's effort and focus more on business logic. Currently in public beta.

Serverful

Serverful cluster types are divided into Pro Edition, Advanced Edition, and Standard Version (no new purchases) for different application scenarios. The comparison is as follows:

Project	Project Description	Pro Edition	Advanced Edition	Standard Version (Stop New Purchases)

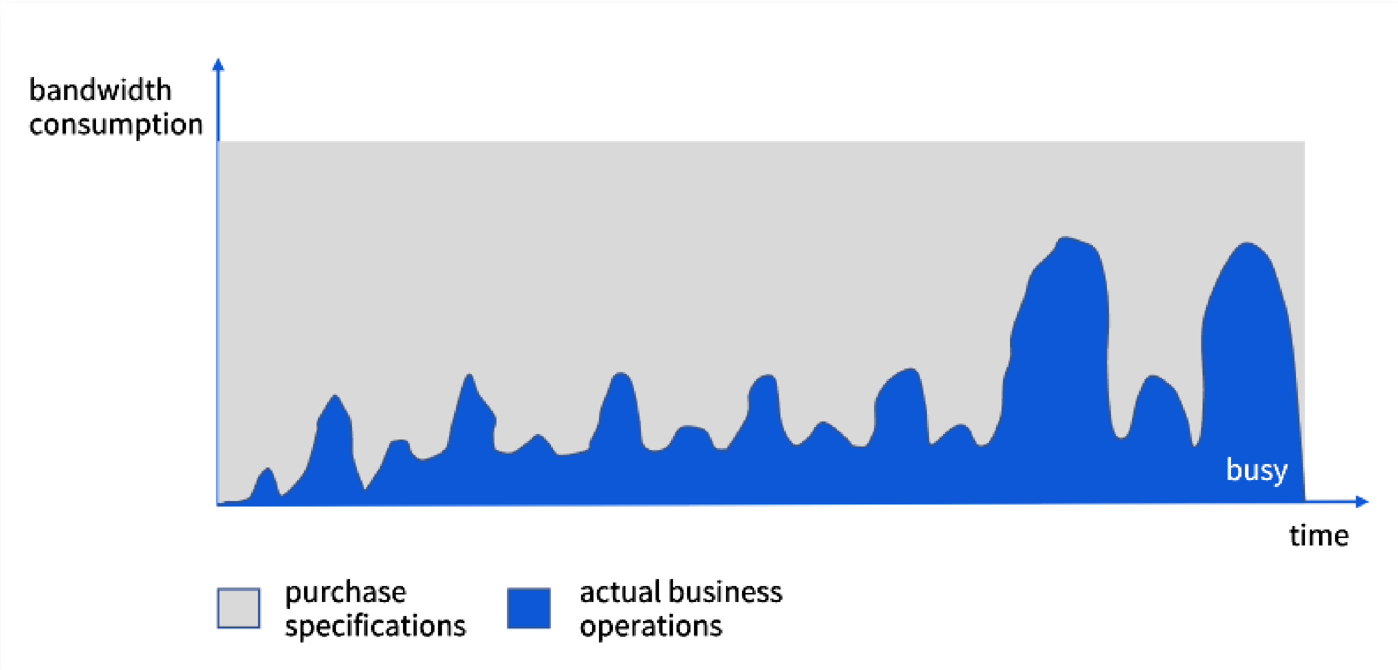
Architecture	Deployment Architecture	Dedicated instance	Share physical node resources	Sharing physical node resources
Stability	SLA assurance	• Service availability: 99.995% • Storage availability: See CBS SLA description	• Service availability: 99.95% • Storage availability: See CBS SLA description	• Service availability: 99.95% • Storage availability: See CBS SLA description
Specification	Bandwidth specification range	20~100000MB/s. For details, see Billing Overview.	20~360MB/s. For details, see Billing Overview.	40MB/s, 100MB/s, 150MB/s
	Storage	Support fixed storage and elastic storage. Among them, elastic storage form offers theoretically unlimited storage space on a pay-as-you-go basis.	Fixed storage only	Fixed storage only
	Scale-out	High degree of freedom, allowing separate scaling of bandwidth, Topic/Partition cap, and Disk	High degree of freedom, supports independent scaling of bandwidth, Topic/Partition cap, and Disk.	Low degree of freedom, allows separate disk scaling
	Public network traffic	Supports upgrading public network bandwidth. If you have higher bandwidth requirements, you can make an additional payment to proceed to purchase. For specific prices, see Billing Overview.	3 Mbps free public network bandwidth is provided by default	1 Mbps free public network bandwidth is provided by default
isolati	Topic	✓	×	×

on	Throttling			
Security Control	SSL Authentication	✓	×	×
Intelligent Ops	Disk Water Level Processing	Dynamic message retention policy Automatic Disk Scale-Out	Dynamic message retention policy	
	Elastic bandwidth	✓	×	×
	Automatic Partition Processing	✓	×	×
	Quick diagnosis	✓	×	×
Monitoring alarm	Advanced Monitoring	✓	×	×
	Metric sorting	✓	×	×
	Event log	✓	×	×
	Export to Prometheus	✓	×	×
High Availability	Cross-AZ High-Availability Deployment	✓	✓	×
	Cross-	✓	×	×

	AZ Migration			
Upgrade	Engine Version Online Upgrade	✓	×	×
	Broker Fix and Upgrade	rapid upgrade	shared cluster extended cycle	shared cluster, extended cycle
advanced service	technology services	Provide parameter optimization consulting services, target specific business scenarios for customized parameter configuration. You can submit a ticket to apply.	Basic fault handling and troubleshooting	Basic fault handling and troubleshooting

Serverless

After running analysis on a large amount of online clusters, we found some common symptoms exist.



1. High cost. Business typically shows peak and trough patterns within a time period. To ensure business stability, reserve specifications according to business peak values.
2. Scaling out is slow. Since the current CKafka classic form has technical architecture limitations, migrating TB-level data may take several days when scaling out brokers.

To address the above issues, we launched the next-generation form of CKafka, the Serverless edition. Following is the feature list during the beta test.

Note:

1. Currently, CKafka Serverless is in public beta. Please use it in offline scenarios or testing environments.
2. Free during the public beta. An account can create up to 5 instances in a single region.
3. Launched in the Guangzhou and Singapore regions.

Project	Project Description	Serverless Edition
Specification	Bandwidth specification range	20-360 MB/s, capacity expansion only in the current period.
	Storage	Elastic storage form.
	Scaling Out	Production/consumption bandwidth can be scaled separately. The maximum quantity of topics and partitions varies by specification.
	Message retention time	24 hours - 72 hours.
	Public Network Traffic	Unsupported during the beta test.
Security Control	SSL Authentication	Unsupported during the beta test.
Message Query	-	✓
intelligent ops	one-click diagnosis	✓
High	Cross-AZ	High-availability deployment with 2 default AZs.

Availability	High-Availability Deployment	

Selection Suggestion

Last updated：2025-09-19 17:44:57

Selection Overview

Before purchasing a TDMQ for CKafka instance, you need to consider key factors such as price, performance, workload, and business scenario to choose the most suitable instance specification. Different forms of CKafka instances vary in specification and capacity performance, and their resource computation logic also differs. This section provides a detailed introduction to the specification features and calculation method of each version instance, thereby helping you perform rapid selection.

Product Form Selection

Product Form

- CKafka offers two product series, Serverful and Serverless, to meet the needs of different business scenarios.
- Product form and detailed feature differences are described in Capability Comparison.

Product Form Selection Recommendation

For `Serverful` editions (including Advanced Edition and Pro Edition) and `Serverless` editions, you can select based on specification range, differentiated features, and adaptation scene:

Recommended Version		Specification Range	Differentiated Features	Adaptation Scene
Serverful Edition	Advanced Edition	• Bandwidth range: 20 MB/s — 360 MB/s • Number of partitions: 400—1800 starting, can be expanded separately	• Basic Monitoring • Instance traffic throttling • dual-AZ • Migration is not supported for availability zones • Support public network access. Public network bandwidth is not scalable.	• Business traffic for small and medium-sized businesses • Cost control • short-term test and development
	Professional	• Bandwidth range: 20 MB/s — 100000 MB/s	• Basic & Advanced Monitoring	• High business traffic

	Edition	• Number of partitions: 400—6000 starting, can be expanded separately	• Prometheus monitoring • Instance & Topic traffic throttling • Intelligent ops • Up to four availability zones • Migration is supported for availability zones • Support public network access. Public network bandwidth is scalable.	• High stability requirements • Fine-grained Ops management • High availability requirement
Serverless Edition		• Specification bandwidth range: 20 MB/s — 360 MB/s • Actual available bandwidth: 40 MB/s — 720 MB/s • Read/write bandwidth can be set separately • Number of partitions: 400—1800, not scalable	• Standard Monitoring • Intelligent ops • Elastic storage • dual-AZ • Public network access not currently supported	• Business traffic for small and medium-sized businesses • High business flexibility • High demand for auto scaling capability

Specification Selection

After selecting the version, you can choose appropriate specifications based on your business usage. The main factors affecting specifications are:

Factors	Description
Bandwidth	• Write bandwidth and read bandwidth. Write bandwidth requires consideration of replica factors in calculation, while read bandwidth does not. • For example: Your business write traffic totals 50MB/s with 3 replicas, and message consumption traffic is 200MB/s. At this point, the purchased instance must meet write bandwidth ⩾ 50 * 3 = 150MB/s and read bandwidth ⩾ 200MB/s. You need to at least select an instance specification with 200MB/s bandwidth.
Storage	• Consider data writing speed, retention duration, and number of replicas.

	• For example: With a daily average write speed of 50 MB/s, message retention period of 72 hr (3 days), and 3 replicas, the required storage space is at least: 50MB/s * 3600 * 72 * 3, approximately 38000 GB.
Number of Partitions	• Consider the impact of replicas. • For example: If your instance has 10 Topics, each with 20 partitions and 3 replicas, the actual number of partitions used is: 10 * 20 * 3 = 60 partitions.
Number of Replicas	The number of replicas can impact bandwidth, storage, and partition estimation, and an excessively large number of partitions in a single cluster may lead to cluster instability.
Number of topics	The number of topics follows the partition count of the cluster purchase. If you require support for more topics, you can choose to extend the partition.
Cluster load	If you regularly enable transactional messages, compression algorithms, and other modes that consume cluster CPU, you need to pay special attention to advanced monitoring of cluster load metrics. See cluster capacity planning practical tutorial.

Note

Although CKafka (Message Queue) supports scaling out parameters such as bandwidth, storage, and number of partitions, its running state does not solely depend on these dimensions. The stability of the entire cluster is influenced by multiple combined factors. The following factors may affect cluster stability. It is advisable to include them in the specification selection metrics and select matching specifications based on your real-world usage scenarios:

Influencing Factors	Description
Cluster TPS is too high	Excessive cluster TPS will consume a very large amount of underlying machine resources and have a significant impact on cluster stability. It is recommended to comprehensively monitor the cluster TPS and load condition to determine whether a higher configured cluster is needed.
Whether to enable compression	Enabling the compression service consumes certain CPU resources for message validation. Different compression algorithms have different impact on CPU. For specific compression algorithm details, see CKafka Data Compression.
Whether to enable transaction message	After enabling transaction messages, the transaction coordinator may impact cluster CPU, memory, and network bandwidth metrics. High-concurrency transactions can cause lock contention, leading to write latency and a possible 30%—50% drop in cluster throughput. For transaction management details, see CKafka Transaction Management.
Whether to	After enabling message idempotence, the cluster will consume resources

enable message idempotence	(CPU/memory/network) as the price paid for **strong data consistency in a single partition**. It cannot resolve cross-session or cross-partition issues and may also impact the cluster as a whole.
Consumer checkpoint submission is frequent	Frequent consumer checkpoint submission is a "reliability-for-throughput" trade-off strategy. It may simultaneously amplify message loss and performance risk, putting significant pressure on the cluster Broker. It is recommended to comprehensively evaluate the cluster load to determine whether a higher-configuration cluster is needed.

Version support for open source Kafka

Last updated: 2025-09-19 17:35:32

Apache Kafka has evolved through approximately 30 versions from 0.7.x to 3.3.x, which can be divided into four development stages: 0.x, 1.x, 2.x, and 3.x. Currently, TDMQ for CKafka provides commercial cloud-compatible versions for these four stages, covering the mainstream Kafka versions used by users.
CKafka continuously iterates features, fixes bugs, and addresses stability risks on the basis of the open-source version. While maintaining full compatibility with open source, it undergoes deep optimization for actual enterprise application scenarios, providing users with a more stable and reliable service experience for messages.
This document introduces the open-source version support mechanism of TDMQ for CKafka, including support status, lifecycle explanation, selection recommendation, capability difference, and version upgrade instructions.

Version Number Introduction

The CKafka version is expressed as x.y.z_va.b.c.

- x.y.z means the compatible open-source Kafka version, as follows:
  - x major version: major update, may contain incompatible API changes.
  - y minor version: new features, backward compatible.
  - z means patch version: Bug fixes or minor optimizations, with no new features introduced.
- a.b.c refers to the CKafka enhanced version provided by Broker.

For example, 2.8.1_v1.0.0 means the enhanced version based on open source Kafka 2.8.1. See Broker version upgrade record for specific version update details.

Open-Source Version Support Notes

Tencent Cloud CKafka is compatible with open-source Kafka. Higher versions are fully backward compatible with lower versions. The following is the compatibility note:

CKafka Version	Compatible Community Edition	Compatibility
0.10.2 (discontinued)	⩽ 0.10.x	100%
1.1.1 (discontinued)	⩽ 1.1.x	100%
2.4.1	⩽ 2.4.x	100%
2.8.1 (Recommended)	⩽ 2.8.x	100%
3.2.3	⩽ 3.2.x	100%

Note:

About CKafka 2.4.1 version description:

When launching version 2.4, CKafka was developed based on the stable Apache Kafka community branch 2.4.1. The community later had a brief development branch 2.4.2, but after several fixes and merges, it was eventually decided to revoke version 2.4.2. To maintain consistency with the community edition, CKafka has adjusted the originally labeled version 2.4.2 to match the community version 2.4.1. Currently, the CKafka 2.4.1 product line is fully compatible with all features of the Apache Kafka community version 2.4.1.

Lifecycle Explanation of CKafka Version

CKafka, Message Queue is a standalone code branch independently maintained by Tencent Cloud, 100% compatible with the Kafka protocol, featuring high performance, high availability, high reliability, high elasticity, and high security. All its versions have a lifecycle independent of the open-source community, uniformly maintained and managed by the Tencent Cloud Message Queue team. To ensure business stability, plan your upgrade schedule according to the product lifecycle and upgrade to supported versions promptly.

<table>
<tr><th>CKafka Version</th><th>Lifecycle Stage</th><th>EOM Time</th><th>Estimated EOFS Date</th><th>Estimated EOS Date</th></tr>
<tr><td>0.10.2</td><td>EOM</td><td>March 6, 2024</td><td>None</td><td>None</td></tr>
<tr><td>1.1.1</td><td>EOM</td><td>March 6, 2024</td><td>None</td><td>None</td></tr>
<tr><td>2.4.1(2.4.2)</td><td>GA</td><td rowspan="3" colspan="3">None</td></tr>
<tr><td>2.8.2</td><td>GA</td></tr>
<tr><td>3.2.3</td><td>GA</td></tr>
</table>

Note:

- EOM: End of Marketing, stop the sale of this version.
- EOS: End of Service & Support, disable service for this version. After this date, no technical support will be provided for the software version.

CKafka Version Capability Difference Comparison

Project	Capability	2.4.1	2.8.1	3.2.3

Product form	Elastic storage	×	Professional Edition support	×
Isolation	Traffic throttling	Professional Edition support		×
Security	SASL authentication	Supports PLAIN and SCRAM mechanisms.		Supports PLAIN mechanism
	ACL policy	Supports CIDR configuration		CIDR cannot be configured
	SSL encryption	Professional Edition support		×
Monitoring	Cluster load metrics	×	Professional Edition support	

CKafka Version Selection Recommendation

Use Cases	Selection Recommendation

Self-built Kafka migrate to the cloud	• Matching version available:
	  ○ Select a CKafka version that matches the major version of your self-built Kafka. For example, if your self-built Kafka is version 2.8.1, select CKafka version 2.8.1.
	• No matching version available:
	  ○ Upward compatibility: Recommend selecting a higher version. For example, if your self-built version is 2.8.0, recommend using version 2.8.1; if your self-built version is 1.1.1, recommend using version 2.4.1 or higher (each Broker version has backward compatibility).
	  ○ Not recommended to downgrade: Avoid selecting an earlier version of CKafka when your self-built version is higher, to prevent risks of incompatibility.

First-time use of Kafka	For version function differences, refer to CKafka Version Capability Difference. Unless otherwise needed, recommend choose CKafka v2.8.1 (stable feature, excellent compatibility).

Upgrading CKafka Version

Minor Version Upgrade Instructions

CKafka supports minor version upgrade through the console. For specific operations, refer to Upgrading Instance Kernel Minor Version.

Major Version Upgrade Instructions

CKafka does not support major version upgrade through the console currently. Console upgrade capacity will be provided subsequently. If you need to upgrade instance kernel major version, see Migrate Cluster for relevant guidelines and perform metadata and message data migration via migration solution.

Comparison with Apache Kafka

Last updated：2025-09-19 17:35:32

The details of TDMQ for CKafka and self-built open-source Kafka are as follows:

Project	Feature Item	TDMQ CKafka Version	Self-Built Kafka
Basic Feature	Topic Parameter Configuration	Support console adjustment of multiple parameters, including number of partitions, number of replicas, message cleanup policy, minimum sync replica count, duration of Segment sharding scrolling, message retention time, etc.	Command line configuration makes it difficult for business personnel to perform custom operations.
	Change in Topic partition count	Support updating the Topic partition count on the console.	Command line configuration makes it difficult for business personnel to perform custom operations.
	Sending Message	Support sending messages in the console.	It is not supported.
	Consumption progress reset	Support resetting consumer offset on the console.	Manual backend configuration is complex and prone to errors.
	Management ability	• Support visualization management of clusters, topics, consumption groups, etc.; more fine-grained management capabilities can be provided through tags. • Support statistics on Topic production and consumption traffic, with traffic details clear at a glance.	Needs to be collocated with an open-source management system, with relatively weak ease of use.
Intelligent operations	**Disk automatic balancing**	Support disk data balancing between Brokers for CKafka cluster instances with data skew problems, reduce the pressure of	It is not supported.

and mainte nance		data skew on the cluster, and decrease the need for disk capacity.	
	Intelligent inspection	Provide one-click diagnostic capability, can actively troubleshoot cluster issues and potential risks, and based on Tencent Cloud Expert experience, provide problem resolutions, automatically summarize health check results to generate reports, and achieve a closed-loop operation and maintenance experience.	It is not supported.
	Intelligent Disk Configuration	Support automatic handling of disk water level. When the disk water level reaches the threshold, support dynamically adjusting message retention or automatically scaling the disk to avoid impacting the business.	It is not supported.
Isolati on	Multi-instance	Creating multiple instances is supported based on one physical cluster.	It is not supported.
	Bandwidth quota control	Each instance supports limiting usage based on bandwidth.	It is not supported.
	Topic Traffic Throttling	Support traffic throttling in the Topic dimensionality, maximize the assurance of business reliability.	It is not supported.
	Disk quota control	Each instance supports limiting usage by disk.	It is not supported.
	Multiple Access Point Configuration	Supports configuring instance routing policy and setting access points in the specified VPC.	Unsupported. Sending messages requires specifying the Broker address, and receiving messages requires ZooKeeper coordination and scheduling. Not flexible enough.
	Separation of	The control plane and data plane	It is not supported.

	control plane and data plane	are separated in architecture. Management and control operations are implemented by calling control components through the TencentCloud API SDK. Data plane operations are called through the open-source Kafka SDK, fully compatible with open source.	
	Root Account Isolation	Based on the Tencent Cloud parent-child accounts system, implement data logical isolation between root accounts.	It is not supported.
Monitoring and Alarms	Out-of-the-Box	Mature standardized deployment and monitoring solutions, ready to use.	Not supported. Users need to self-build and maintain an open-source monitoring system. Moreover, if the monitoring system cannot meet the requirements, it is difficult to customize.
	Monitoring Alarm	Provide rich monitoring metrics such as production and consumption peak bandwidth, production and consumption average bandwidth, production and consumption message count, disk utilization, instance connections, and unconsumed message count. Support multi-dimensional monitoring and alerts for instances, Topics, and consumption groups.	Not supported. Additional integration with open-source third-party components is required.
	Metric Sorting	Support viewing the top 10 Topic production and consumption traffic and top 10 disk usage, and the top 10 consumption speed of Consumer Groups.	It is not supported.
High Availability	**Cross-AZ High Availability Deployment**	Support cross-AZ high-availability deployment and mature recovery plan. Support upgrading from	Supported but more complicated, with deployment solutions and parameters to be designed.

		single-AZ deployment to cross-AZ deployment.	
	Cross-Availability Zone Disaster Recovery	Support instance-level synchronization of data, support data replication and migration feature between instances in different regions, and ALL data and metadata of the instance will be synchronized.	Supported but more complicated, and the open-source solution has certain defects.
Security compliance	Topic-dimensional ACL access control	Support controlling access permissions based on user + IP range. Permission configuration supports down to the instance + Topic dimensionality, and support two modes: blocklist and allowlist.	Configuring parameters via the command line is cumbersome and prone to errors.
	SASL password authentication	Support console configuration of SASL authentication.	Configuring parameters via the command line is cumbersome and prone to errors.
	SSL authentication	Support console configuration of SSL certificate.	Configuring parameters via the command line is cumbersome and prone to errors.
	Manage operation permission control	Console operations are based on Tencent Cloud CAM permission control for secure and compliant access.	It is difficult to exercise granular control over management and control operation permissions.
	Operations are traceable	Management and control operations are docked with CloudAudit and are traceable.	It is not supported.
Other	**Online Upgrade Capability**	CKafka provides the capability for version upgrades, allowing for seamless upgrades to the latest version when bugs or security vulnerabilities occur in the community edition.	Not supported. Manual operation is required. Manual upgrade may affect the business.

High Availability

Cluster High Reliability

Last updated：2025-09-19 14:32:46

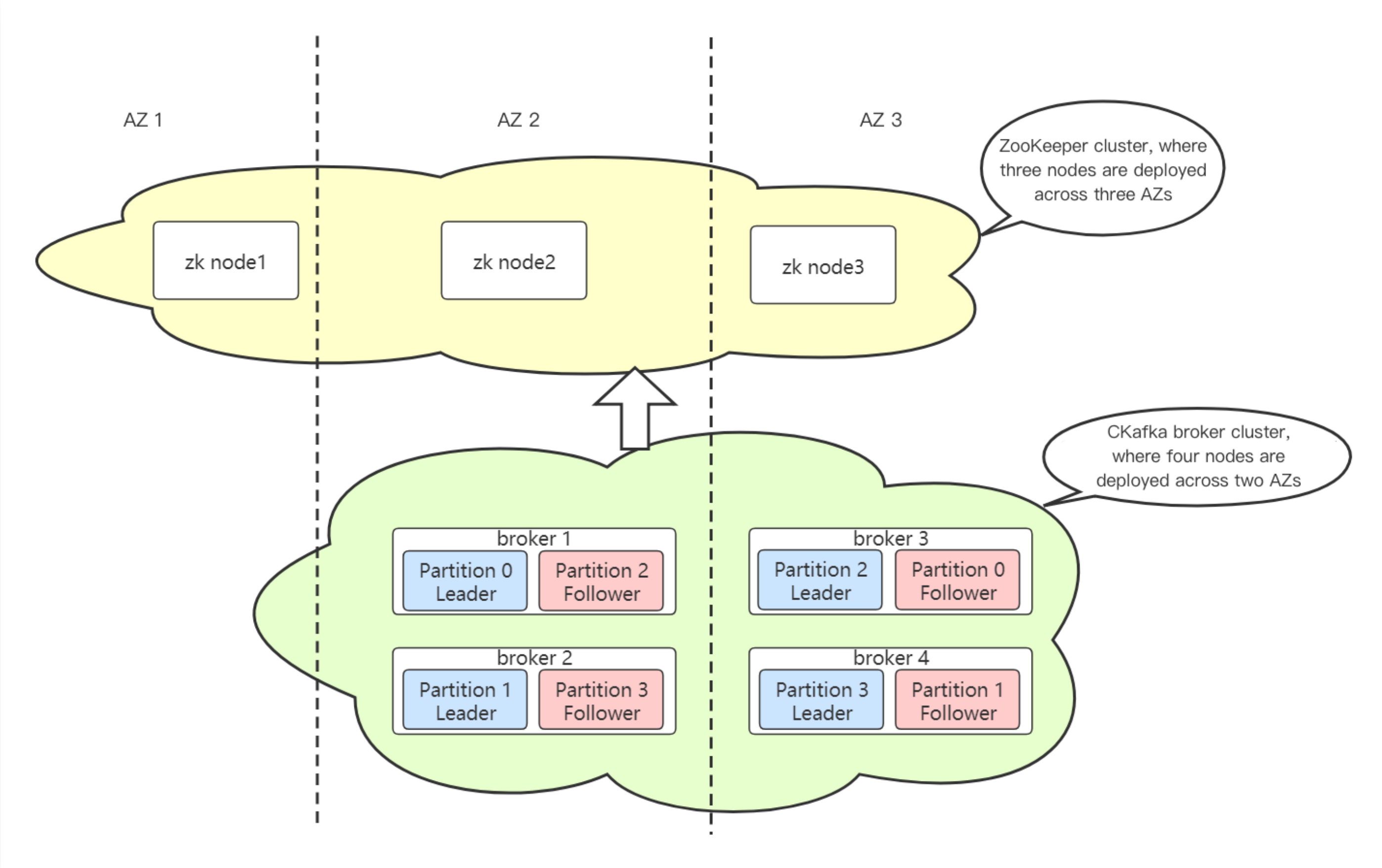
TDMQ for CKafka provides cluster-level high availability capability. With multi-AZ deployment and the availability zone migration feature, it effectively improves service stability and disaster recovery capability.

Cluster-Level High Availability Description

High Availability Capability	Applicable Versions	Description	Reference Documentation
Cross-AZ deployment	Advanced Edition Pro Edition	When purchasing a CKafka instance in a region with three or more AZs, you can choose multiple availability zones for deployment. Partition replicas are forcibly distributed across nodes in different AZs. This deployment method enables your instance to provide services properly even if a single availability zone becomes unavailable. • Pro Edition supports up to four AZs. • Advanced Edition supports up to two AZs.	Cross-AZ Deployment
Modify instance AZ	Professional Edition	Migrate the instance to another AZ within the same region. After migrating availability zones, all attributes of the instance, configurations, and connection address will not change. • Migrate from one availability zone to another: The instance availability zone is under full load or other situations that impact instance performance. • Migrate from one availability zone to multiple availability zones: Improve disaster recovery capability of the instance and implement cross IDC disaster recovery.	Migrate instance AZ

Cross-AZ Deployment

Deployment Architecture



The cross-availability zone deployment architecture of CKafka is divided into network layer, data layer and control layer.

Network Layer

CKafka exposes a VIP to clients. After connecting to the VIP, clients get metadata information on partitions (this metadata usually maps address information to different ports under the same VIP). This VIP can failover to another availability zone at any time. When a certain AZ is unavailable, the VIP automatically drifts to another available node in this region, thereby achieving cross-availability zone disaster recovery.

Data Layer

The CKafka data layer and Native Kafka use the same distributed deployment method, with multiple replicas distributed across different Broker nodes in different AZs. For each partition, there is a Leader-follower relationship between nodes. If the Leader encounters an exception or goes offline, the cluster control node (Controller) will elect a new partition Leader to handle requests for that partition.

For clients, when an AZ becomes unavailable, if the Leader of a Topic partition is located on a Broker node in the unavailable AZ, the established connections may experience timeouts or be closed. After the partition Leader node fails, the Controller (if the Controller node fails, the remaining nodes will elect a new Controller) will elect a new Leader node to provide services. The Leader switchover time is at the second level (the specific switch time is directly proportional to the number of cluster nodes and metadata size). Clients will perform a Scheduled Refresh of Topic partition metadata and connect to the new Leader node for production consumption.

Control layer

The control layer of CKafka and Native Kafka implement the same technical solution, depending on zookeeper for service discovery and cluster Controller election of Broker nodes. **CKafka instances supporting cross-availability zone deployment have zookeeper cluster nodes (abbreviated as zk nodes) deployed in three AZs (or IDCs).** If any zk node in an AZ disconnects due to failure, the entire zookeeper cluster can still provide services normally.

Pros and Cons of Cross-AZ Deployment

Strengths

It can significantly enhance cluster disaster recovery capability. When unexpected network fluctuations, restart due to power failure, or other force majeure risks occur in a single AZ, it ensures clients can restore message production and consumption after a short waiting time for reconnection.

Disadvantages

If cross-AZ deployment is adopted, since partition replicas are distributed in multiple AZs, message replication incurs additional cross-AZ network latency compared with a single AZ. This latency directly affects the write time consumed for production (when the client ACK parameter is greater than 1 or equal to -1, all). Currently, the cross-AZ latency in major regions such as Guangzhou, Shanghai, and Beijing is generally 10ms—40ms.

Cross-AZ Deployment Scenario Analysis

Single AZ Unavailable

After a single AZ becomes unavailable, as explained earlier, clients will disconnect and reconnect, and after reconnection, the service can still be provided normally.
Since the management API service currently does not support cross-AZ deployment, if a single AZ becomes unavailable, you may be unable to create topics, configure ACL policies, or view monitoring through the console. However, this will not affect the production and consumption of existing services.

Network Isolation between Two AZs

If network isolation occurs between two AZs, making them unable to communicate with each other, a cluster split-brain event may appear. In this case, nodes in both AZs provide services, but data writes from one AZ may be deemed as dirty data after cluster recovery.

Consider the following scenario: when the cluster Controller node and one zk node in the ZooKeeper cluster experience network isolation from other nodes. At this point, the other nodes will re-elect a new Controller (since the majority of ZooKeeper cluster nodes have normal network communication, the Controller election can succeed). However, the isolated Controller still considers itself the Controller node, leading to a split-brain situation in the cluster.
At this point, client writes need to be considered case by case. For example, when the client's ACK policy equals -1 or all, and the number of replicas is 2, assuming the cluster has 3 nodes, a split-brain scenario may result in a 2:1 distribution. Partitioned writes from the original Leader in the 1-node region will report errors, while the other side will proceed with normal writing. However, if there are 3 replicas and ACK = -1 or all is configured, neither side will write successfully. At this point, further handling solutions must be determined based on specific parameter configurations.
After the cluster network is restored, clients can resume production and consumption without any operation. However, since the server will normalize the data again, the data of one split node will be directly truncated. But for multi-replica cross-AZ data storage methods, this kind of truncation will not bring data loss.

Multi-AZ Disaster Recovery Limit

Capacity Constraint

CKafka implements multi-AZ disaster recovery through the geographic distribution of underlying resources. In case of AZ failure, CKafka will switch the Partition Leader to a Broker node in an available AZ, so the underlying resources carrying traffic will become fewer. Therefore, there will be capacity issues. The following is the resource available case for single availability zone failure:

- For a 2-AZ disaster recovery instance, usable capacity = instance / 2. To ensure proper usage, the customer needs to reserve 100% redundant resources.
- For a 3-AZ disaster recovery instance, usable capacity = instance / 3 * 2. To ensure proper usage, the customer needs to reserve 50% redundant resources.
- For a 4-AZ disaster recovery instance, usable capacity = instance / 4 * 3. To ensure proper usage, the customer needs to reserve 33.3% redundant resources.
- For an n-AZ disaster recovery instance, usable capacity = instance / n * (n - 1). To ensure proper usage, the customer needs to reserve 1 / (n - 1) redundant resources.

Parameters and Configuration Limits

CKafka allows modifying the Topic-level configuration `min.insync.replicas` in the console. This configuration is valid only when the client sets ack=-1, to ensure a message is considered successful only after being synchronized by `min.insync.replicas` replicas simultaneously (for example: Topic replicas = 3, min.insync.replicas=2, message production is required to be synchronized by at least 2 replicas to be considered successful).
Therefore, the Topic configuration (`min.insync.replicas` < number of AZs) must be set to ensure production availability in case of AZ failure. The Topic configuration `min.insync.replicas` < Topic replicas must be set

to ensure production availability in case of single node failure. The rule is: `min.insync.replicas` < Topic replicas <= number of AZs.

- 2-AZ instance, Topic replicas = 2, min.insync.replicas = 1
- 3-AZ instance, Topic replicas = 2, min.insync.replicas = 1
- 3-AZ instance, Topic replicas = 3, min.insync.replicas <= 2

Data High Reliability

Last updated: 2025-09-19 14:57:41

Tencent Cloud Message Queue CKafka is a high-throughput distributed message system widely used in enterprise applications. High data reliability is one of the important features of CKafka. This documentation introduction describes the mechanisms for high reliability of data in CKafka.

Multi-Replica and Election Mechanism

Multiple Replicas

The multi-replica design can enhance system availability and reliability. CKafka recommends 3 replicas for message topics and at least 2 replicas for production use.
Replicas are usually evenly distributed across the entire cluster broker nodes. The replica allocation algorithm is as follows:

1. Sort all brokers (assuming there are n brokers in total) and the partitions to be allocated.
2. Allocate the i-th partition to the (i mod n)-th broker.
3. Allocate the j-th replica of the i-th partition to the ((i + j) mod n)-th broker.

Leader Election Mechanism

TDMQ for CKafka dynamically maintains an ISR (in-sync replicas) in ZooKeeper. All replicas in the ISR keep up with the Leader. Only members in the ISR may be elected as the Leader.

- Assuming f + 1 replicas in ISR, one Partition can tolerate f Replica failures while ensuring committed messages are not lost.
- Assuming a total of 2f + 1 Replicas (including Leader and Follower), f + 1 Replicas must replicate the message before commit. To ensure correct election of a new Leader, failed Replicas cannot exceed f.

Leader Switchover Principle

When creating a new Topic, the CKafka broker cluster allocates a leader for each partition and evenly distributes the partitions of the current Topic across all brokers.
However, after using for a period of time, uneven partition allocation on brokers may occur, or exceptions such as `BrokerNotAvailableError` and `NotLeaderForPartitionError` may appear during production consumption.
This is usually caused by a leader switch in the partition. Typical scenarios:

- When a certain partition leader's broker encounters unexpected circumstances, such as network interruption, program crash, or hardware failure, making it unable to communicate with the broker controller, the current Topic partition will undergo a leader switch, and the leader will migrate to a follower partition.
- When the Kafka cluster sets `auto.leader.rebalance.enable = true` to perform automatic

rebalance, or when brokers are manually added/removed and rebalance is manually triggered, leader switching may occur due to partition automatic balance.

When a leader switch occurs due to broker disconnection:

- If the client sets `ack = all` and `min.insync.replicas > 1` , since the message is confirmed on both the leader partition and follower partition, the message will not be lost.
- If the client sets `ack = 1` , messages within the `replica.lag.time.max.ms` period may not be synchronized to the follower partition, possibly causing message loss.
- When a leader switch is triggered by rebalance due to manual/automatic operations (such as instance upgrade, single AZ switching, cross-AZ instance migration) while the broker is normal, it will not cause message loss for the following reasons:
- If the client sets `ack = all` and `min.insync.replicas > 1` , since the message is confirmed on both the leader and follower partitions, the message will not be lost.
- If the client sets `ack = 1` , leader switch will automatically sync the offset in the partition, so the message will not be lost.

Acks Parameter Configuration

1. acks = all (or -1)

- When the producer sets the acks parameter to all (or -1), it means the producer needs to wait for all ISR replicas to confirm receipt of the message before considering the message sent successfully.
- This configuration provides the highest data reliability but may increase message send delay. For example, in a partition with 3 replicas, after the producer sends a message, it has to wait for both the Leader replica and two Follower replicas to confirm receipt before the message is considered sent successfully.

2. acks = 1

- When the acks parameter is set to 1, the producer only needs to wait for the Leader replica to confirm receipt of the message before considering it sent successfully. The sync status of Follower replicas does not affect the producer's judgment.
- This configuration provides a balance between data reliability and message send delay. If the Leader replica fails after message acknowledgment and the Follower replicas have not yet fully synchronized the message, it may lead to MSG_LOST.

3. acks = 0

- When the acks parameter is set to 0, the producer does not need to wait for any replica confirmation and can consider the message sent successfully. This configuration offers the lowest latency for message sending but also the minimum data reliability. If a Broker node fails before receiving the message, the message will be lost.

Relationship between Replica Quantity, Acks, and Min.Insync.Replicas Parameters

Key Rules and Configuration Suggestions

1. Correlation between replica quantity and min.insync.replicas
- Mandatory constraint: min.insync.replicas ⩽ number of replicas, otherwise the Broker will deny write requests.
- Recommended configuration:
  - Production environment: replica quantity = 3 & min.insync.replicas = 2 (balance disaster recovery and performance)
  - Financial scenario: number of replicas = 5 & min.insync.replicas = 3 (strong consistency requirement)
2. acks and min.insync.replicas linkage rules
- acks = all effective condition:
  - Must cooperate with min.insync.replicas ⩾ 2, otherwise when only the Leader remains in ISR, acks = all degenerates into acks = 1, unable to ensure data security.
- acks = 1 potential risk:
  - If the Leader replica fails and the Follower is unsynced, messages may be lost (compensatory measures should be done through retries and idempotency).

Common Replica Number, Acks, and Min.Insync.Replicas Usage Scenarios for Data Reliability

Number of Replicas	acks	min.insync.replicas	Behavior Feature	Data Reliability
any value	0	any value	–Producer does not wait for confirmation – Messages may not be written to any replica and get lost	min: no data reliability assurance
2	all	1	– Degenerates into acks = 1 (ISR may only include the Leader) – Tolerate 1 Broker failure	low: data may be lost during Leader failure

3	1	1	– Producer only requires Leader confirmation – ISR may only include the Leader – Tolerate 2 Broker failures	moderate: data may be lost during Leader failure with no Follower synchronization
3	a ll	2	–The producer needs to wait for at least 2 replicas in the ISR to confirm. – ISR includes the Leader and at least 1 Follower – Tolerate 1 Broker failure	High: Data integrity is maintained when at least 2 replicas in the ISR are live.
3	a ll	3	– The producer needs to wait for all 3 replicas to confirm. – ISR must contain all replicas – Tolerate 0 Broker failure	Top: In extreme cases only allow 0 node faults
5	a ll	3	– The producer needs to wait for 3 replicas to confirm. – ISR includes the Leader and 2 Followers – Tolerate 2 Broker failures	High: Data security is ensured when most replicas are live.

Scenarios of Data Loss and Solution

This section will introduce the factors that impact the data reliability of TDMQ for CKafka through the production side, server side (CKafka), and consumer side, and provide the solutions.

How to deal with data loss on the production side?

Causes of Data Loss

When the producer sends data to the TDMQ for CKafka, the data may be lost due to network jitter, and the TDMQ for CKafka does not receive the data. Possible scenarios:

- When the network load is high or the disk is busy, the producer has no retry mechanism.
- The disk exceeds the purchase specification limit, such as an instance disk spec of 9000GB. If the disk is not scaled out in time after being full, data cannot be written to TDMQ for CKafka.
- Suddenly or continuously growing peak traffic exceeds the purchase specification limit, such as an instance peak throughput specification of 100MB/s. If peak throughput exceeds the limit for a long time without timely scaling out, data writing to CKafka will slow down. With the producer queue timeout mechanism, data cannot be written to TDMQ for CKafka.

Solution

- The producer enables the failure retry mechanism for the important data.
- For disk usage, set up monitoring and alarm policy when configuring the instance to take preventive measures. When the disk is full, you can upgrade in time in the console (upgrading between non-exclusive CKafka instances is smooth and non-stop, or disk can also be upgraded separately), or reduce disk storage by changing the message retention time.
- To minimize message loss on the producer side, you can tune the buffer size through `buffer.memory` and `batch.size` (in bytes). The larger the buffer, the better. If the producer goes down for some reason, the more data there is in the buffer, the more garbage needs to reclaim, and recovery will be slower. You should **pay attention to the number of messages produced by the producer, average message size** (monitoring indicators in TDMQ for CKafka).
- Configure the producer-side acks
- When a producer sends data to a leader, the level of data reliability can be set through the acks parameter and min.insync.replicas.
- acks and configuration details. Refer to relationship between parameters: replica quantity, acks, and min.insync.replicas.

Recommended Parameter Values

This parameter value is for reference only. The actual value depends on the actual business situation.

- Retry mechanism: `message.send.max.retries=3;retry.backoff.ms=10000;`
- Guaranteed high reliability: `request.required.acks=-1;min.insync.replicas=2;`
- Guaranteed high performance: `request.required.acks=0;`
- Reliability + performance: `request.required.acks=1;`

How to deal with data loss on the server (CKafka)?

Causes of Data Loss

- The leader of the partition crashes before completing the backup of the replica followers. Even if a new leader is elected, the data is lost because it is not backed up in time.
- The disk writing mechanism of open source Kafka is asynchronous, that is, data is first stored in PageCache. When not officially written to disk, if the broker disconnects, restarts, or fails, data in PageCache may be lost since it hasn't been written to disk.
- Disk failure causes the loss of data that has been written to disk.

Solution

- Open source Kafka uses multiple replicas. The official recommendation is to use replicas to ensure data integrity. At this point, data will be lost only when multiple replicas and multiple brokers fail simultaneously, which offers much higher data reliability than a single replica. Therefore, TDMQ CKafka enforces dual-replica topics, configurable to three replicas.

- CKafka service is configured with more reasonable parameters log.flush.interval.messages and log.flush.interval.ms to write data to disk.
- CKafka performs special processing on disk to ensure data reliability is not affected when part of the disk is damaged.

Recommended Parameter Values

A replica in an asynchronous state can be elected as a leader: `unclean.leader.election.enable=false // off`

How to deal with data loss on the consumer side?

Causes of Data Loss

- The offset is committed before the data is actually consumed. If the consumer crashes during the process but the offset has been refreshed, the consumer will miss a piece of data and need to reset the offset of the consumer group to retrieve the data.
- The consumption and production rates are different for too long, and the message retention time is too short, resulting in the message being expired and deleted before it is consumed in time.

Solution

- Reasonably configure the parameter auto.commit.enable. When equal to true, it indicates automatic submission. It is recommended to use scheduled commit to avoid frequent offset commits.
- Monitor consumer situations and adjust data retention time appropriately. Monitor the current consumer offset and the number of unconsumed messages, and configure alarms to prevent messages from being expired and deleted due to the slow consumption.

Troubleshooting Plan for the Data Loss

Printing Partition and Offset Locally for Troubleshooting

The print information code is as follows:

```
Future<RecordMetadata> future = producer.send(new ProducerRecord<>
(Topic, messageKey, messageStr));
RecordMetadata recordMetadata = future.get();
log.info("partition: {}", recordMetadata.partition());
log.info("offset: {}", recordMetadata.offset());
```

- If partition and offset are printed, it indicates the message sent has been successfully saved on the server. At this point, you can query messages at related positions through the message query tool.

- If partition and offset are not printed, it means the message was not saved by the server, and the client needs to retry.

Use Limits

Last updated: 2025-09-19 17:35:33

This document outlines some limitations on metrics and performance in TDMQ for CKafka. Be mindful not to exceed these limits during use to avoid any issues.

Item	Description
Number of topics	The topic limit depends on product specifications. For details, see Billing Overview | Tencent Cloud.
Number of partitions	• One topic can support up to 3,000 partitions. • The instance-level partition limit includes replicas, which are typically 2 or 3. • Reducing the number of partitions is not supported.
Partition throughput	• In the case of `ack = 1`, the throughput of a single TDMQ for CKafka partition is between 30 and 60 MB/sec due to factors such as TDMQ for CKafka's partition architecture, business data size, and request frequency. • When using `ack = -1` (strong consistency), **we recommend maintaining the throughput of a single TDMQ for CKafka partition between 10 and 20 MB/sec**. This helps ensure that the stability of request duration is not impacted by factors such as TDMQ for CKafka's partition architecture, business data size, and request frequency. • In scenarios involving idempotency or transactions, CKafka server load may increase, impacting the maximum throughput per partition. If your business is latency-sensitive, it is recommended to test with different cluster specifications.
Duration	TDMQ for CKafka is a high-throughput, high-volume message queue and cannot guarantee low latency for every request. It is recommended to set the timeout as follows: • For the producer, when ACK = 1, the timeout period defaults to **30s.** • For the producer, when ACK = -1, the timeout period defaults to **60s.** • The timeout period for the consumer is set to **60s**.
Number of consumer groups	• For Standard Edition, the recommended number of instance-level consumer groups is up to 50. • For Advanced and Professional Editions, it is recommended that the number of consumer groups per instance not exceed 500. Contact Us for upgrade options.
Client Usage	• It is recommended to use the CKafka control console or TencentCloud API

	for managing clusters, topics, and other metadata. • It is not recommended to directly use AdminClient to create topics, modify partitions, or perform other metadata operations to avoid data inconsistency in the console. Such actions are planned to be blocked in the future.
Instance	• The region attribute of instances cannot be changed. • The maximum number of connections for the instance is 50,000. Exceeding this limit will cause the client cannot create new connections. If the maximum value is unreasonable in actual business, contact us to submit a request for modification.
Version	• Compatible with open-source versions 2.4, 2.8, and 3.2
Routes	An instance can create up to five routes, with only one of them being a public network route.
Public network bandwidth	Not supported.
Exposing ZooKeeper	Not supported.
Exposing underlying resources	It is not supported so as to avoid risks caused by user's operation.
Message size	It cannot exceed 12 MB; otherwise, messages will fail to be sent.
Tag	Each Tencent Cloud resource can have up to 50 tags.
Concurrent operations in the console	To prevent consistency issues from high concurrency in instance metadata operations, some interfaces are locked to limit requests. Currently, only 20 concurrent console requests per instance are allowed to ensure stability and success (including SDK API calls).
Data reliability	Due to CKafka's own replica synchronization and ACK mechanism, there may be some extreme cases where data loss occurs. For example, when the number of replicas is 2 and ack = 1, if a leader switch occurs in the partition and the replica has not been synchronized at this point, data loss will occur. If your business cannot tolerate data loss, please ensure to refer to CKafka Data Reliability Statement for client configuration.

Note:
Due to TDMQ for CKafka's partition architecture, business data size, request frequency, and the stability of the physical layer, TDMQ for CKafka cannot guarantee that each request has low

latency.

- However, we strive to ensure the following two requirements:
  - The percentage of low-latency production for a single instance each month aligns with the instance's SLA.
  - The percentage of low-latency consumption for a single instance each month aligns with the instance's SLA.
- The calculation formulas are as follows:
  - The percentage of low-latency production for a month = (Number of minutes in the month when 99.9th percentile latency production time per minute is 30 seconds or less / Total service minutes in the month) × 100%.
  - The percentage of low-latency consumption for a month = (Number of minutes in the month when 99.9th percentile latency consumption is 60 seconds or less / Total service minutes in the month) × 100%.

Regions and AZs

Last updated：2025-09-19 17:35:33

A region is the physical location of an IDC. Availability zone (AZ) refers to the physical IDC of Tencent Cloud in the same region with independent power supplies and network resources. For more information, see CVM-Regions and AZs.

Supported regions

China

Region		Code
South China	Guangzhou	ap-guangzhou
	Shenzhen	ap-shenzhen
	Shenzhen Finance	ap-shenzhen-fsi (Available only for financial institutions and enterprises; apply through Online Consultation)
East China	Shanghai	ap-shanghai
	Shanghai Finance	ap-shanghai-fsi (Available only for financial institutions and enterprises; apply through Online Consultation)
	Nanjing	ap-nanjing
	Hangzhou	ap-hangzhou
North China	Beijing	ap-beijing
	Beijing Finance	ap-beijing-fsi (Available only for financial institutions and enterprises; apply through Online Consultation)
	Tianjin	ap-tianjin
Central China	Changsha	ap-changsha
Southwest China	Chengdu	ap-chengdu
	Chongqing	ap-chongqing
Hong Kong/Macao/Taiwan	Taipei (China)	ap-taipei

(China)	Hong Kong (China)	ap-hongkong

Other countries and regions

Region		Code
Southeast Asia	Singapore	ap-singapore
	Bangkok	ap-bangkok
	Jakarta	ap-jakarta
Northeast Asia	Seoul	ap-seoul
	Tokyo	ap-tokyo
US West	Silicon Valley	na-siliconvalley
US East	Virginia	na-ashburn
South America	São Paulo	sa-saopaulo
Europe	Frankfurt	eu-frankfurt

Selection of Regions and AZs

When selecting a region and AZ, take the following into consideration:

- The geographic locations of TDMQ for CKafka instances, your business, and your target users: We recommend that you choose the region closest to your customers when purchasing a TDMQ for CKafka instance to reduce latency and improve access speed.
- Relationship between CKafka and other Tencent Cloud services:

  When you select other Tencent Cloud services, we recommend that you try to locate them all in the same region and AZ to allow them to communicate with each other through the private network, reducing access latency and increasing access speed.
- High availability and disaster recovery.

  Even if you have just one VPC, we still recommend that you deploy your businesses in different AZs to prevent a single point of failure and enable cross-AZ disaster recovery.
- There may be network latency among different AZs. We recommend that you assess your business requirements and find the optimal balance between high availability and low latency.
- If you need access to TDMQ for CKafka instances in other countries or regions, we recommend that you select an instance in those other countries or regions. If you use a TDMQ for CKafka instance in China to access servers in other countries or regions, you may encounter much higher network latency.

Resource availability

The following table describes global, regional, and AZ-specific TDMQ for CKafka resources.

Resource	Resource ID Format (8-Digit String of Numbers and Letters)	Type	Description
User account	Unlimited	Globally unique	Users can use the same account to access Tencent Cloud resources from around the world.
TDMQ for CKafka instance	ckafka-xxxxxxxx	TDMQ for CKafka instances can only be used in a single AZ of a single region.	TDMQ for CKafka instances can only be created in a specific AZ.

Related Operations

Migrating instance to another AZ

CKafka supports instance migration across AZs in the same region. If you need this, submit a ticket for assistance.

Relevant cloud services

Last updated：2025-10-16 17:21:45

Cloud Service	Related Introduction
Operation Audit	**CloudAudit (CAudit)** is a service that supports supervision, compliance check, operation review, and risk auditing for your Tencent Cloud account. After authorizing CloudAudit, it will start recording operational activities in your console and API. Meanwhile, the audit console also offers a convenient log retrieval function with multiple dimensions.
Config Audit	**Config Audit (Config)** helps you centrally audit and govern cloud resources. It continuously records and evaluates configuration messages and change operations of Tencent Cloud resources scattered across your account in each region, thereby helping you efficiently achieve autonomous supervision and standardized operations.
CLS	**Cloud Log Service (CLS)** is a SaaS Service that supports integrated observability for Log and Metric data collection, ETL, search and analysis, visualization, and alarming. It supports the use of Kafka Producer SDK to upload logs to CLS.
Tencent Cloud Smart Advisor	**Tencent Cloud Smart Advisor** is based on an IaC base, with a product philosophy of one platform and multiple applications. It provides a unified visualization Cloud architecture IDE and vertical applications in multiple ITOM domains: Cloud inspection, capacity monitoring, Chaotic Fault Generator experiment, Cloud escort, health dashboard, and contingency plan management, to build a reassuring Cloud Governance Platform for customers. • Tencent Smart Advisor - Chaotic Fault Generator provided the CKafka Broker High Disk I/O Load Experiment Actions to test the business system's ability to recover from unexpected delays when faced with high disk I/O loads on CKafka Broker, thereby improving business security and stability.
Tencent Cloud CDC	**Tencent Cloud CDC (Cloud Dedicated Cluster)** is a newly launched managed infrastructure product designed to extend centralized public cloud services into near-field services within the user's IDC. Combining the advantages of public cloud and local IDC, users can leverage the rich capabilities of the public cloud while ensuring lower latency and enhanced data security.
COS	**Cloud Object Storage (COS)** is a distributed storage service provided by Tencent Cloud to store massive files. Users can store and view data at any time over the network. Tencent Cloud COS enables all users to use a data storage service with high scalability, low cost, reliable and secure. • CKafka message backup is a feature provided by Tencent Cloud COS (Cloud Object Storage) based on SCF (Serverless Cloud Function). It helps users

	transfer CKafka messages to COS, making it easy to analyze and download the data. • Data export to CKafka is a data lake solution provided by Tencent Cloud COS (Cloud Object Storage) based on SCF (Serverless Cloud Function). It helps users export data in file formats such as CSV and JSON to Tencent Cloud CKafka within the same region for data aggregation and analysis of massive messages and log data.
SCF	**SCF (Serverless Cloud Function)** is a serverless execution environment provided by Tencent Cloud for enterprises and developers. It helps you run code without purchasing or managing servers, making it an ideal computing platform for scenarios like real-time file processing and data processing. Simply write core code in the supported languages on the SCF platform and set execution conditions to run code securely with elastic scaling on Tencent Cloud infrastructure. • With the aid of SCF and CKafka trigger, it eliminates the complexity of storing Kafka messages to COS, ES, DB and more.
Tencent Cloud Observability Platform (TCOP)	**Tencent Cloud Observability Platform (TCOP)** offers an integrated monitoring solution based on all types of monitoring data including metrics, traces, logs, and events, combined with powerful visualization and alarm capabilities. It meets your end-to-end unified monitoring requirements, improves operation and maintenance troubleshooting efficiency, and provides protection for stable and continuous business operations. • You can configure monitoring and alerts to comprehensively monitor CKafka instances, receive alarm notifications through multiple notification channels promptly upon alarm occurrence, and timely understand your instance status. • You can consume CKafka Message Queue content by configuring the CKafka Connector. The CKafka Connector uses the Pull model, where the event connector actively pulls CKafka content and routes related events to more services through rule routing.
Tencent Cloud TCHouse-C	**Tencent Cloud TCHouse-C** is an enterprise-level cloud data warehouse service built on the open-source OLAP engine ClickHouse. It enables rapid deployment of a PB-level real-time data warehouse in just a few minutes, achieving ultra-fast real-time analysis of large-scale data. TCHouse-C's kernel is highly compatible with the open-source edition of ClickHouse, drastically enhancing product stability, security, and operational convenience, so you do not need to worry about the underlying infrastructure and can focus on enhancing data value. • You can seamlessly import data from Cloud services like CKafka into Tencent Cloud Data Warehouse for analysis.
TencentCloud Managed	**TencentCloud Managed Service for Prometheus (TMP)** is a highly available, fully managed service based on open-source Prometheus. It is highly integrated with Tencent Kubernetes Engine (TKE) and compatible with various application

Service for Prometheus	components in the open-source ecosystem. Combined with the alerting functions of Tencent Cloud Observability Platform (TCOP) and Prometheus Alertmanager, it offers out-of-the-box efficient operations and maintenance capabilities, reducing development and O&M costs.
TencentCloud Managed Service for Grafana (TCMG)	**TencentCloud Managed Service for Grafana (TCMG)** is a managed service developed by Tencent Cloud based on the popular open-source visualization project Grafana, in collaboration with Grafana Labs. TCMG provides secure, maintenance-free Grafana capabilities with built-in plugins for multiple Tencent Cloud data sources, such as Prometheus monitoring service, container service, log service, Graphite, and InfluxDB, achieving unified data visualization. With TCMG and preset cloud service monitoring plugins, you can visualize component data from Tencent Cloud Observability Platform, including but not limited to CVM, cloud databases, CKafka, and Redis.
Stream computing	**Stream Compute Service (SCS)** is a real-time analysis tool in the big data product ecosystem. It is an Enterprise-level Real-time Big Data Analysis Platform based on Apache Flink, featuring one-stop development, seamless connection, sub-second latency, low cost, and secure and stable performance. SCS aims to maximize the value of enterprise data and accelerate the construction process of real-time digitalization. It seamlessly connects with mainstream Tencent Cloud data products such as Message Queue, database, and data warehouse, as well as self-built open-source big data components. It supports scaling out Connectors as needed to integrate with various external data systems. Fully compatible with Apache Flink, it enables smooth migration to the cloud for self-built systems.
Cloud Access Management	**Cloud Access Management (CAM)** helps you securely and conveniently manage access to Tencent Cloud services and resources. You can use CAM to create sub-users, user groups, and roles, and control their access scope through policies. CAM supports SSO capabilities for users and roles, allowing you to set up interoperability between enterprise users and Tencent Cloud based on specific management scenarios.
VPC	**Virtual Private Cloud (VPC)** is a logically isolated network space that you define in Tencent Cloud. You can build logically isolated, user-customized network spaces for resources such as CKafka to enhance the security of users' cloud resources and meet the needs of different use cases.
Tencent Cloud container service	**TKE (Tencent Kubernetes Engine)** provides a high-performance container management service with containers as the core, based on native Kubernetes and highly scalable. It integrates seamlessly with Tencent Cloud IaaS products to assist customers in quickly realizing business containerization.
CLB	**Cloud Load Balancer (CLB)** is a service that distributes traffic to multiple backend servers. It can expand the service capacity of application systems externally through traffic distribution and improve their availability by eliminating single

	points of failure. The CLB service uses a virtual IP address (VIP) to virtualize multiple backend servers in the same region into a high-performance, highly available application service pool. It distributes network requests from clients to the server pool based on the specified method.
TencentCloud API	**TencentCloud API** is the cornerstone of Tencent Cloud's open ecosystem, featuring easy automation, easy remote calls, high compatibility, and low system requirements. With TencentCloud API, you just need a small amount of code to quickly operate Cloud services. For frequently called features, it can improve efficiency. You can also combine TencentCloud APIs to implement more advanced features. TencentCloud API provides interfaces for various resources of Tencent Cloud products, allowing you to quickly operate Cloud services and more easily manage your cloud resources.
CVM	**Cloud Virtual Machine (CVM)** offers you secure and reliable elastic computing services. In just a few minutes, you can obtain and enable CVM on the cloud to meet your computing needs. Based on changing business needs, you can scale up or down computing resources in real time. CVM supports billing by resources actually used, helping you save computing costs. Using CVM can significantly reduce your software and hardware procurement cost and simplify IT operations and maintenance.
CBS	**Cloud Block Storage (CBS)** provides persistent block-level storage services for Cloud Virtual Machines. The data in CBS is automatically stored with multi-replica redundancy within the AZ to mitigate the risk of data single points of failure, offering up to 99.9999999% data reliability.
microservice platform	**Tencent Service Framework (TSF)** is a PaaS platform centered around applications and microservices, offering full application lifecycle management, data-driven operation, three-dimensional monitoring, and service governance capabilities.