TDMQ for CKafka

General References

Product Documentation

©2013-2022 Tencent Cloud. All rights reserved.

Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

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

Trademark Notice

Tencent Cloud

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

Service Statement

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

Contents

©2013-2022 Tencent Cloud. All rights reserved.

General References
Conducting Production and Consumption Pressure Testing on CKafka

Last updated：2024-01-09 15:02:47

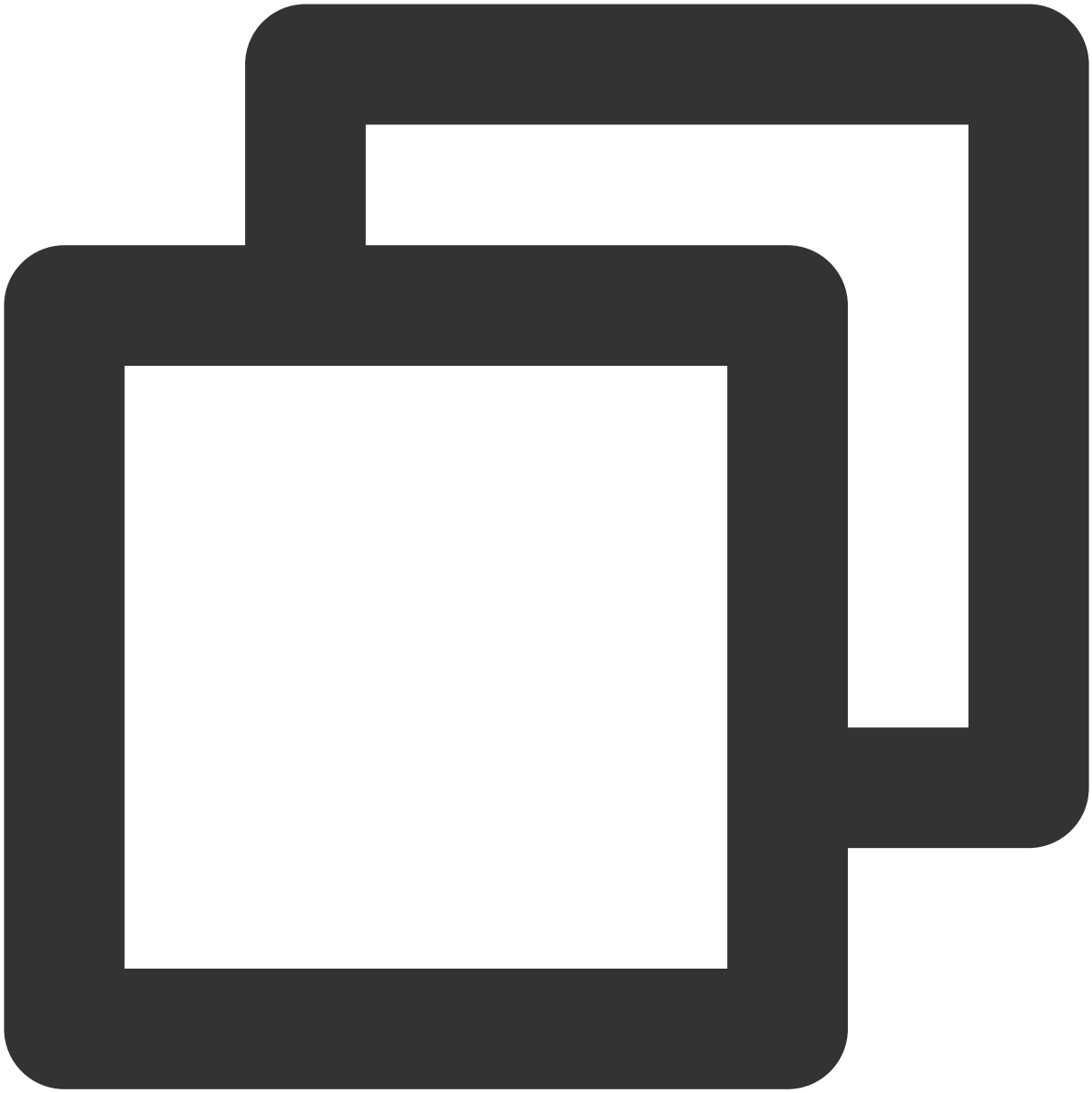
Testing Tool

The open-source script of the Kafka client can be used for Kafka producer and consumer performance testing. Test results are displayed mainly based on the size of messages sent per second (MB/second) and the number of messages sent per second (records/second).
Kafka producer test script: `$KAFKA_HOME/bin/kafka-producer-perf-test.sh`
Kafka consumer test script: `$KAFKA_HOME/bin/kafka-consumer-perf-test.sh`

Testing Command

Note:
The `ckafka vip:vport` in the following sample commands should be replaced by the actual IP and port assigned for your instance.
Sample command for production testing:

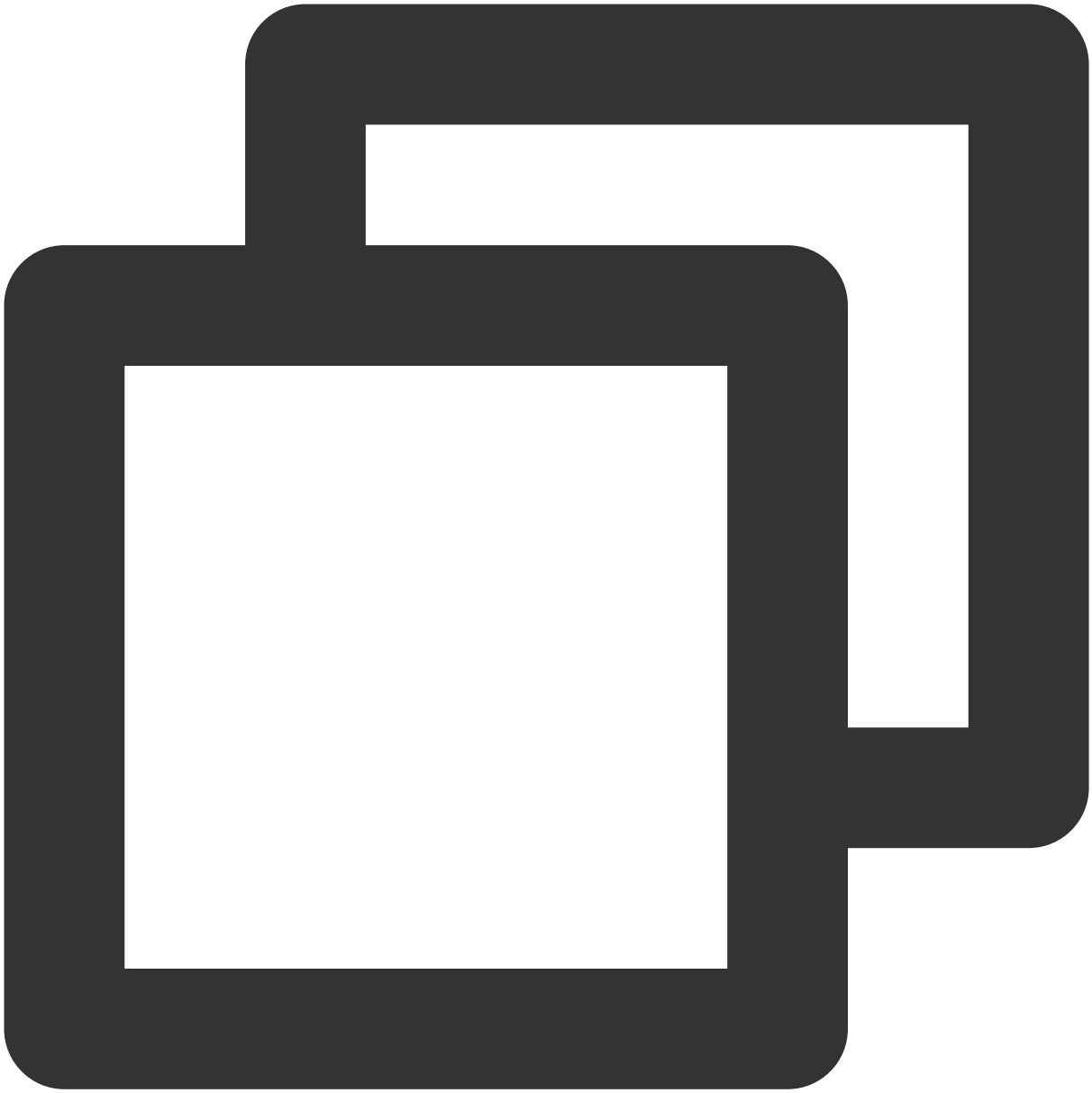
©2013-2022 Tencent Cloud. All rights reserved.



```
bin/kafka-producer-perf-test.sh
--topic test
--num-records 123
--record-size 1000
--producer-props bootstrap.servers= ckafka vip : port
--throughput 20000
```

Sample command for consumption testing:

©2013-2022 Tencent Cloud. All rights reserved.



```
bin/kafka-consumer-perf-test.sh
--topic test
--new-consumer
--fetch-size 10000
--messages 1000
--broker-list bootstrap.servers=ckafka vip : port
```

Suggestions

©2013-2022 Tencent Cloud. All rights reserved.

We recommend you create three or more partitions to increase throughput. This is because there must be at least three CKafka cluster nodes at the backend. If only one partition is created, it will be distributed in a single broker, which will affect CKafka performance.
As messages in each CKafka partition are ordered, the production performance will be affected if there are too many partitions. We recommend you create up to six partitions.
It is necessary to simulate concurrency with multiple clients to ensure the testing effect. We recommend you use multiple test servers as the pressure test clients (producers) and start multiple pressure test programs on each test server to increase concurrency. To avoid the high load of test servers, you are also recommended to start one producer every second rather than all producers simultaneously.

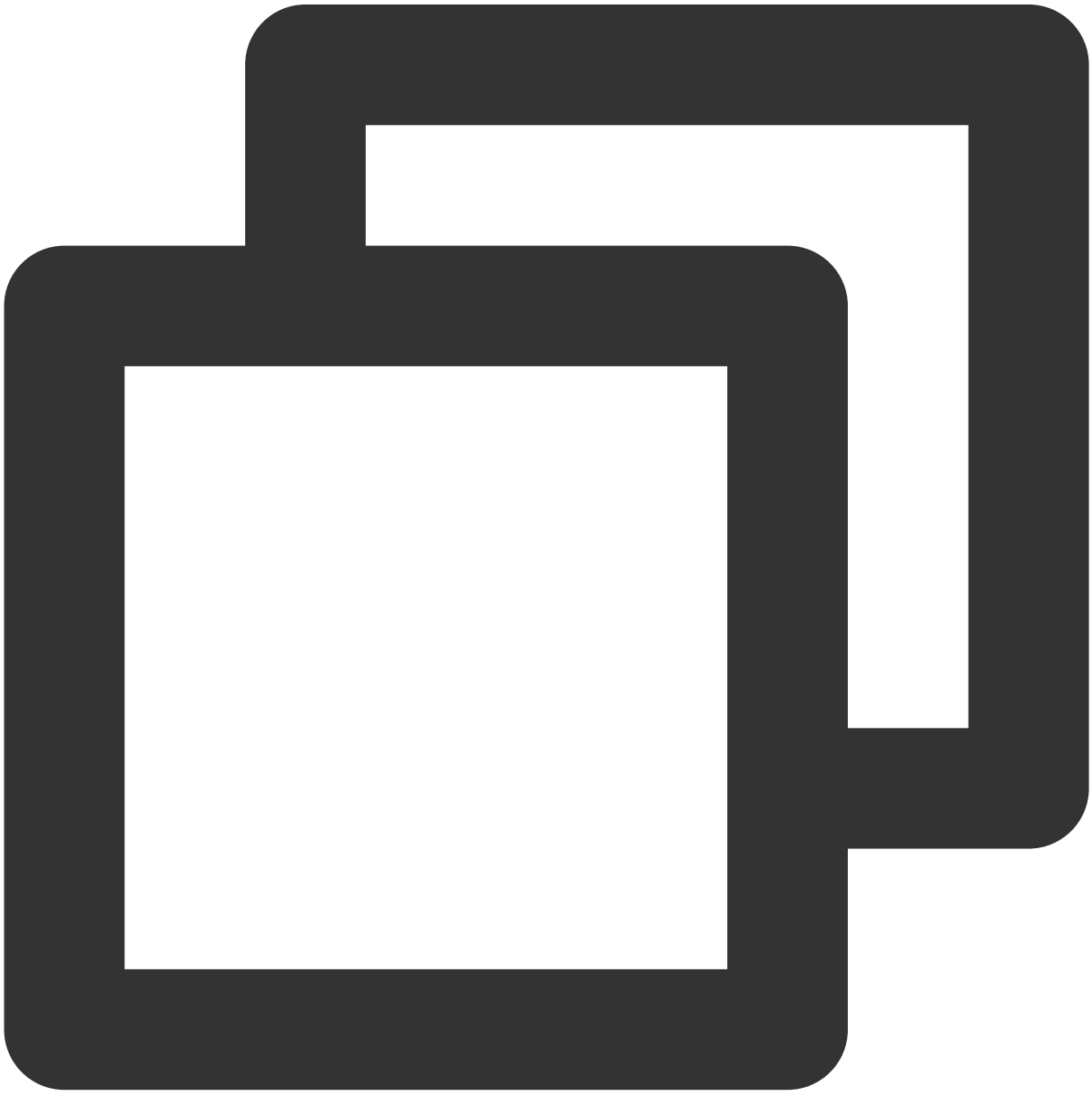
©2013-2022 Tencent Cloud. All rights reserved.

Configuration Guide for Common Parameters in CKafka

Last updated：2024-01-09 15:02:48

Broker Configuration Parameter Description

The following are the configurations of a CKafka Broker for your reference:



©2013-2022 Tencent Cloud. All rights reserved.

```
# Maximum message length in bytes.
message.max.bytes=1000012

# Whether to allow automatic creation of topics. The default value is false. Curren
auto.create.topics.enable=false

# Whether to allow topic deletion by calling the API.
delete.topic.enable=true

# The maximum request length allowed for a Broker is 16 MB.
socket.request.max.bytes=16777216

# Each IP can establish up to 5,000 connections with a Broker.
max.connections.per.ip=5000

# Offset retention period. The default value is 7 days.
offsets.retention.minutes=10080

# Everyone is allowed to access when there is no ACL configuration.
allow.everyone.if.no.acl.found=true

# The log segment size is 1 GB.
log.segment.bytes=1073741824

# The log rolling check interval is 5 minutes. If the retention period is set to le
log.retention.check.interval.ms=300000
```

Note:

For configurations not listed here, see the open-source Kafka default configurations.

Topic Configuration Parameter Description

1. Number of partitions

From the producer's point of view, writes to different partitions are completely in parallel; from the consumer's point of view, the number of concurrencies depends entirely on the number of partitions (if there are more consumers than partitions, there will definitely be idle consumers). It is important to select an appropriate number of partitions to fully play the performance of the CKafka instance.

The number of partitions should be determined based on the throughput of production and consumption, ideally through the following formula:

Note:

Num = max(T/PT, T/CT) = T/min(PT, CT)

©2013-2022 Tencent Cloud. All rights reserved.

Num represents the number of partitions, T the target throughput, PT the maximum production throughput by the producer to a single partition, and CT the maximum consumption throughput by the consumer from a single partition. The number of partitions is equal to T/PT or T/CT, whichever is larger.

In practice, the actual PT is determined by batch size, compression algorithm, acknowledgement mechanism, number of replicas, and so on, while the actual CT is subject to business logic, which varies according to the actual conditions.

We recommend that the number of partitions be greater than or equal to that of consumers to achieve maximum concurrency. For example, if there are 5 consumers, there should be 5 or more partitions.

However, having too many partitions will lower production throughput and increase time consumed by elections and thus need to be avoided. See the following for reference:

A single partition can implement sequential writes of messages.

A single partition can only be consumed by a single consumer process in the same consumer group.

A single consumer process can consume multiple partitions simultaneously, so partition limits the concurrency of consumers.

The more partitions there are, the longer it takes to elect a leader upon failure.

Offset can be down to the partition level. The more partitions there are, the more time the offset query consumes.

The number of partitions can be dynamically increased but not reduced. However, an increase will result in message rebalance.

2. Number of replicas

At present, the number of replicas must be at least 2 to ensure availability. To ensure high reliability, we recommend maintaining at least 3 replicas.

Note:

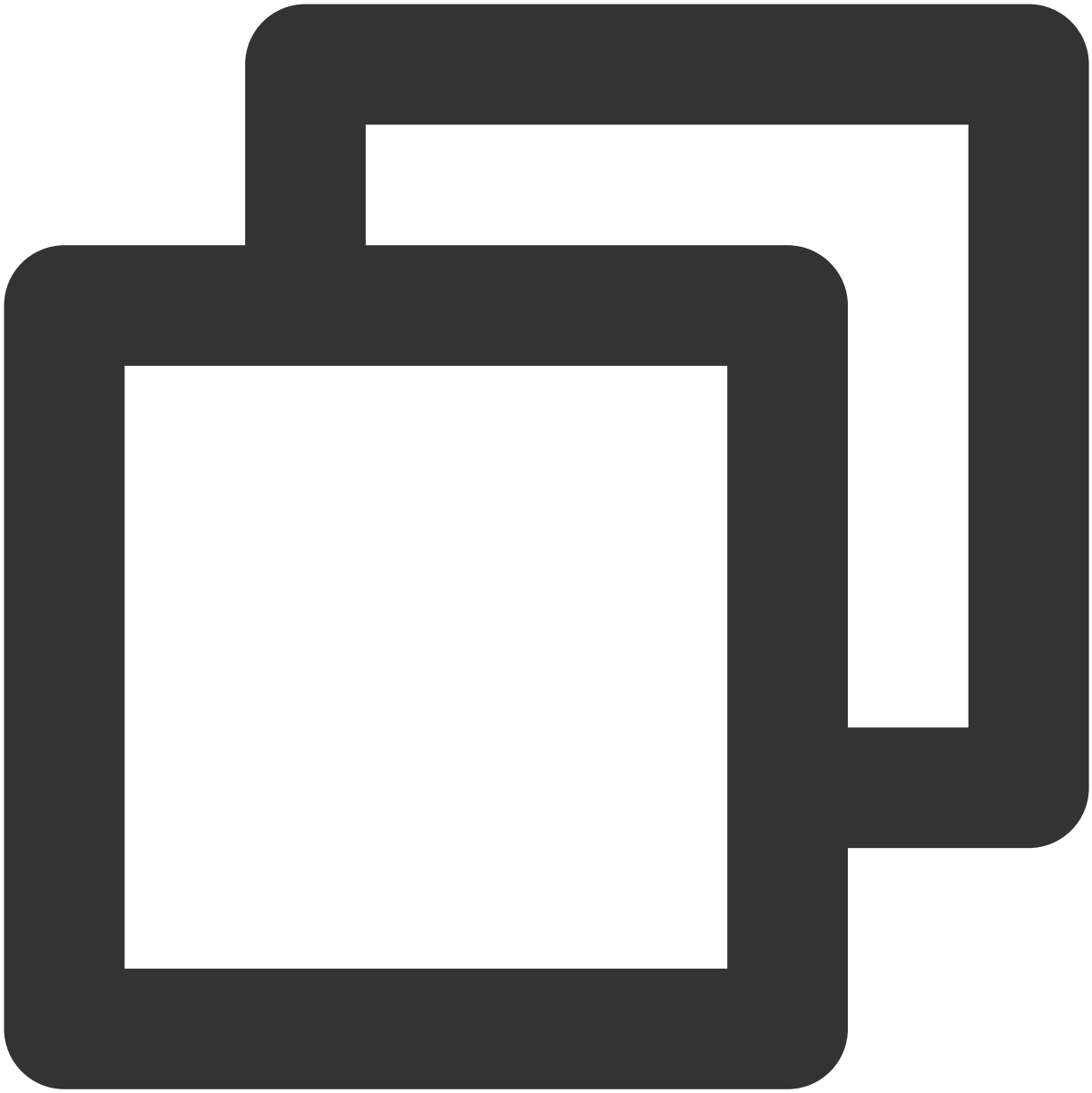
The number of replicas will affect the production/consumption traffic; for example, if there are 3 replicas, the actual traffic will be 3 times the production traffic.

3. Log retention period

The `log.retention.ms` configuration of a topic is set through the retention period of the instance in the console.

4. Other topic-level configurations

©2013-2022 Tencent Cloud. All rights reserved.

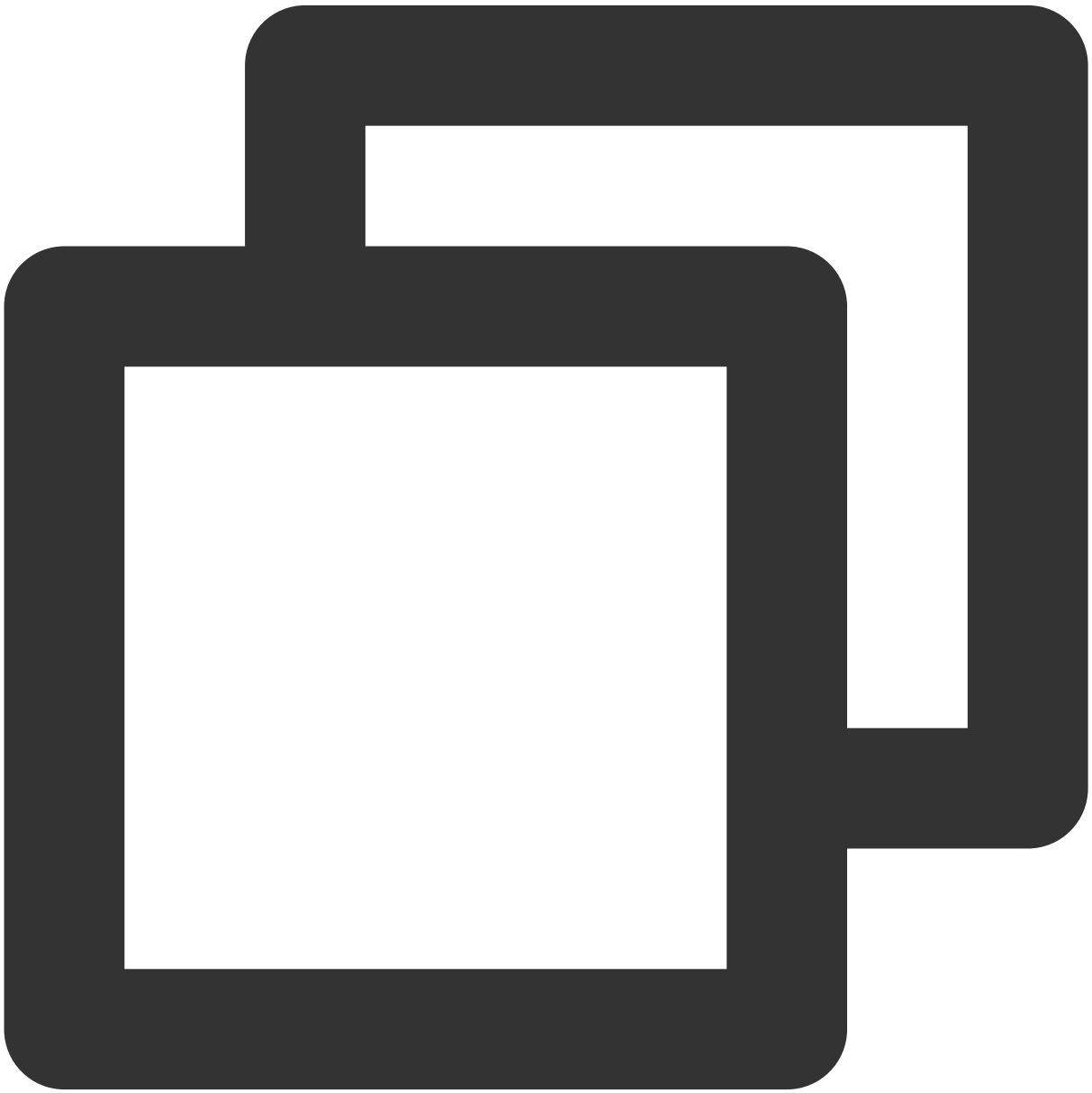


```
# Maximum message length at the topic level.
max.message.bytes=1000012

# Messages in the 0.10.2 version are in the V1 format.
message.format.version=0.10.2-IV0

# Replica not in ISR can be selected as a leader; in this case, availability is hig
unclean.leader.election.enable=true

# Minimum number of replicas for producer requests submitted by ISR. If the number
min.insync.replicas=1
```

©2013-2022 Tencent Cloud. All rights reserved.

Producer Configuration Guide

The following describes common parameter settings for the Producer client. We recommend adjusting them based on your actual business scenarios:



```
# The producer will attempt to bundle and send the messages sent to the same partit
batch.size=16384
```

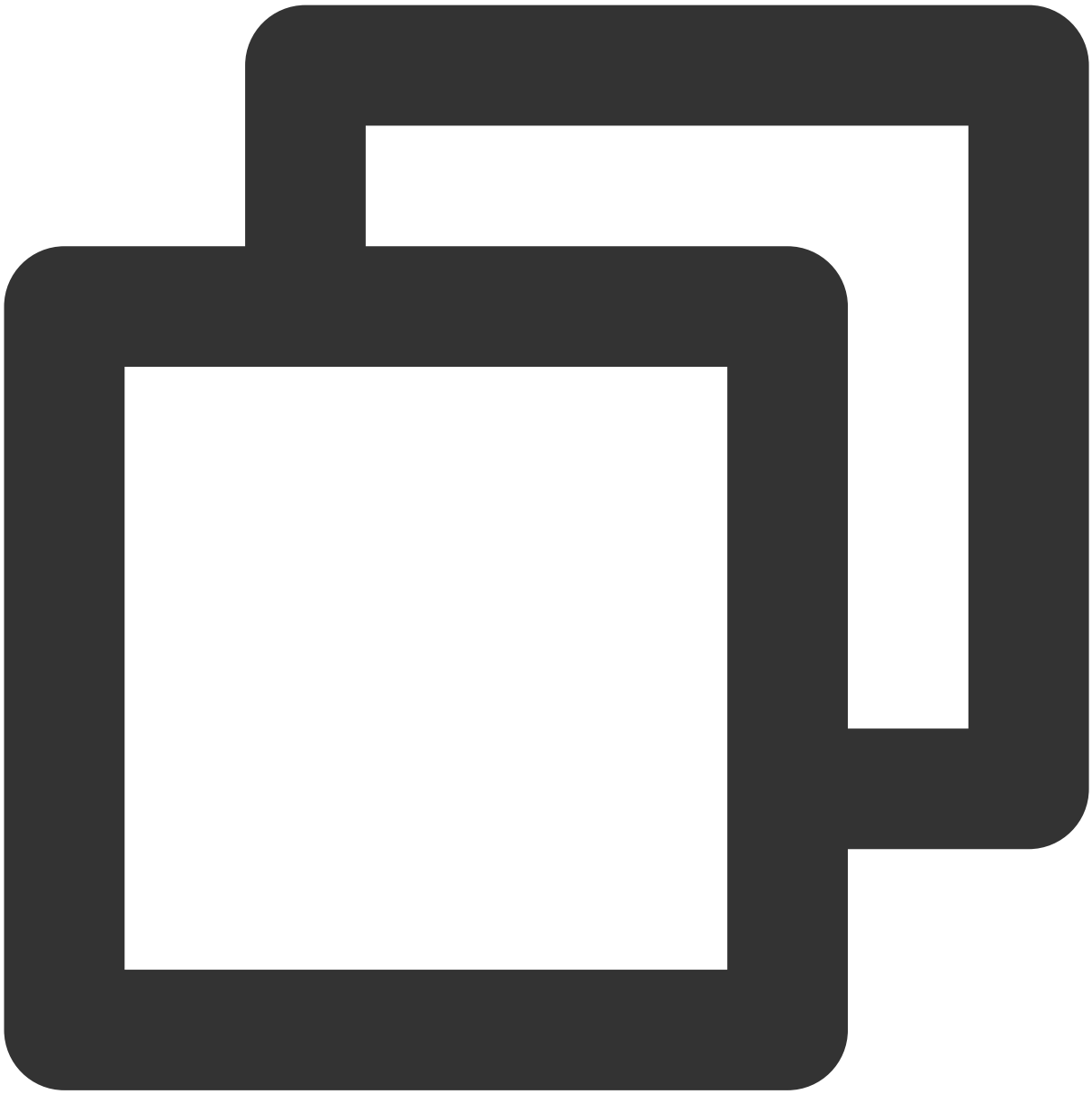
©2013-2022 Tencent Cloud. All rights reserved.

```
# The following describes the 3 ACK mechanisms supported by a Kafka producer:
# -1 or all: the Broker responds to the producer and continues to send the next mes
# 0: the producer continues to send the next message or next batch of messages with
# 1: the producer sends the next message or next batch of messages after it receive

# If users do not configure this, the default value will be 1. Users can customize
acks=1

# Control the maximum time a production request waits in the Broker for replica syn
timeout.ms=30000

# Configure the memory that the producer uses to cache messages to be sent to the B
buffer.memory=33554432

# If messages are produced faster than they are sent by the sender thread to the Br
max.block.ms=60000

# Set the time to send the scheduled message, so that more messages can be sent in
linger.ms=0

# Maximum size of the request packet that the producer can send, which defaults to
max.request.size=1048576

# Compression format configuration. Currently, version 0.9 and earlier do not suppo
compression.type=[none, snappy, lz4]

# Timeout period for the client to send a request to the Broker, which cannot be sm
request.timeout.ms=30000

# Maximum number of unacknowledged requests that the client can send on each connec
max.in.flight.requests.per.connection=5


# Number of retries upon request error. It is recommended that you set the paramete
retries=0

# Retry interval upon request failure.
retry.backoff.ms=100
```

Consumer Configuration Guide

©2013-2022 Tencent Cloud. All rights reserved.

The following describes common parameter settings for the Consumer client. We recommend adjusting them based on your actual business scenarios:



```
# Whether to sync the offset to the Broker after a message is consumed, so the late
auto.commit.enable=true

# Interval for the automatic submission of offset when auto.commit.enable=true is c
auto.commit.interval.ms=5000

# Mode to initialize the offset when no offset is configured for the Broker (such a
# earliest: reset to the minimum offset in the partition.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
# latest: reset to the maximum offset in the partition. This is the default value.
# none: throw an OffsetOutOfRangeException exception without resetting the offset.
auto.offset.reset=latest

# Identify the consumer group to which the consumer belongs.
group.id=""

# Consumer timeout period when the Kafka consumer groups are used. If the Broker do
session.timeout.ms=10000

# Interval at which the consumer sends a heartbeat when the Kafka consumer groups a
heartbeat.interval.ms=3000

# Maximum interval allowed for calling the poll again when the Kafka consumer group
max.poll.interval.ms=300000

# Minimum data size returned by a fetch request. The default value is 1 B, indicati
fetch.min.bytes=1

# Maximum data size returned by a fetch request. The default value is 50 MB.
fetch.max.bytes=52428800

# Fetch request wait time.
fetch.max.wait.ms=500

# Maximum data size returned by each partition in a fetch request. The default valu
max.partition.fetch.bytes=1048576

# Number of records returned in one poll call.
max.poll.records=500

# Client request timeout period. If no response is received after this time elapses
request.timeout.ms=305000
```

©2013-2022 Tencent Cloud. All rights reserved.

Connecting to Legacy Self-Built Kafka

Last updated：2024-01-09 15:02:48

CKafka is compatible with the Producer and Consumer APIs of Apache Kafka 0.9 and above (currently, directly purchasable versions include v0.10.2, v1.1.1, v2.4.1, v2.8.1, and v3.2.3). To connect to an on-premises Kafka of an earlier version (such as v0.8), partial rewriting of APIs is needed. This document compares the Producer and Consumer APIs of Kafka 0.8 and its earlier versions, and describes how to rewrite the APIs.

Kafka Producer

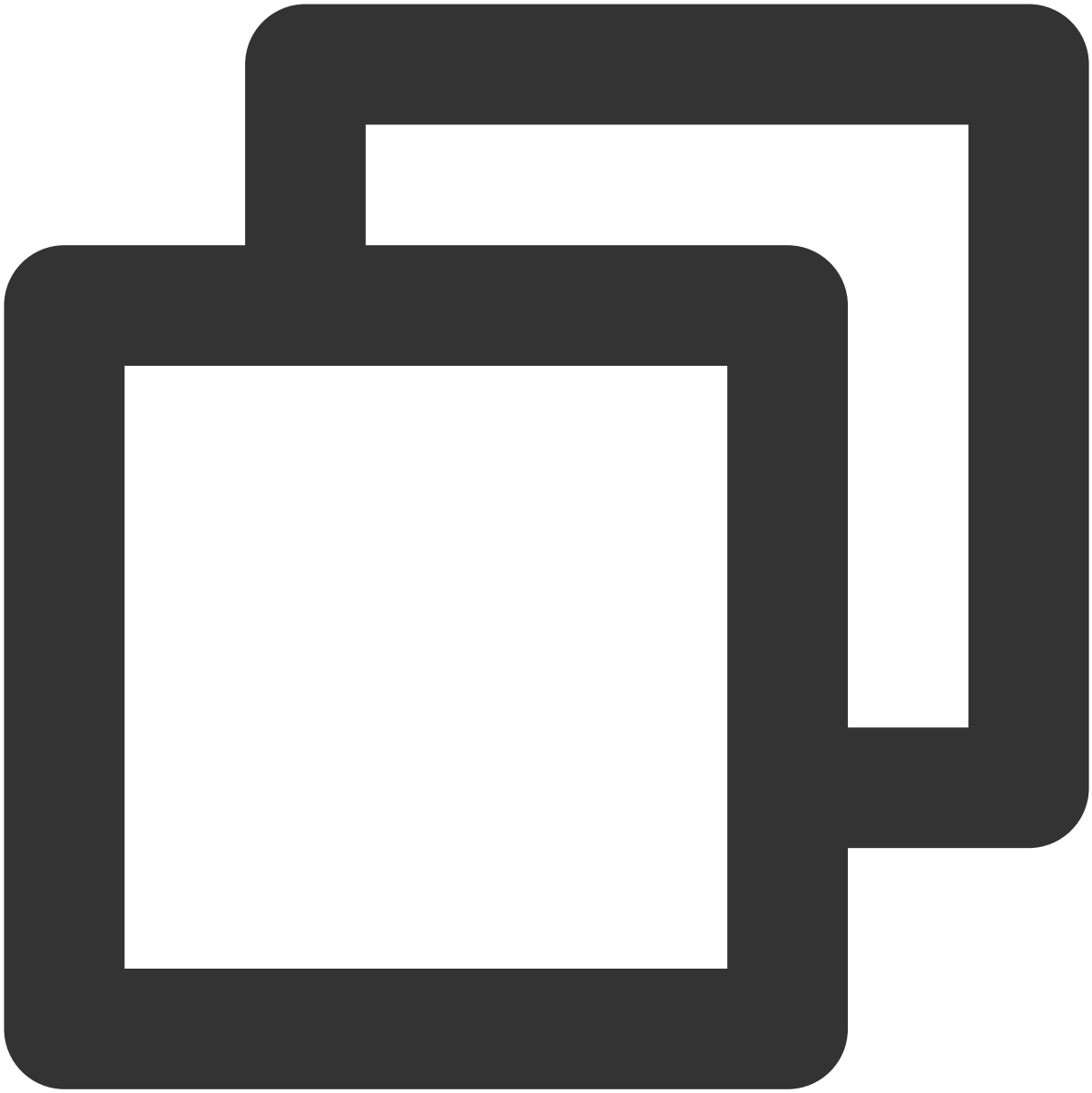
Overview

In Kafka 0.8.1, the Producer API is rewritten. This Producer client version is officially recommended because it provides better performance and more features. The community will maintain the new version of the Producer API (referred to as the New Producer API).

Comparison between new producer API and old producer API

New Producer API Demo

©2013-2022 Tencent Cloud. All rights reserved.

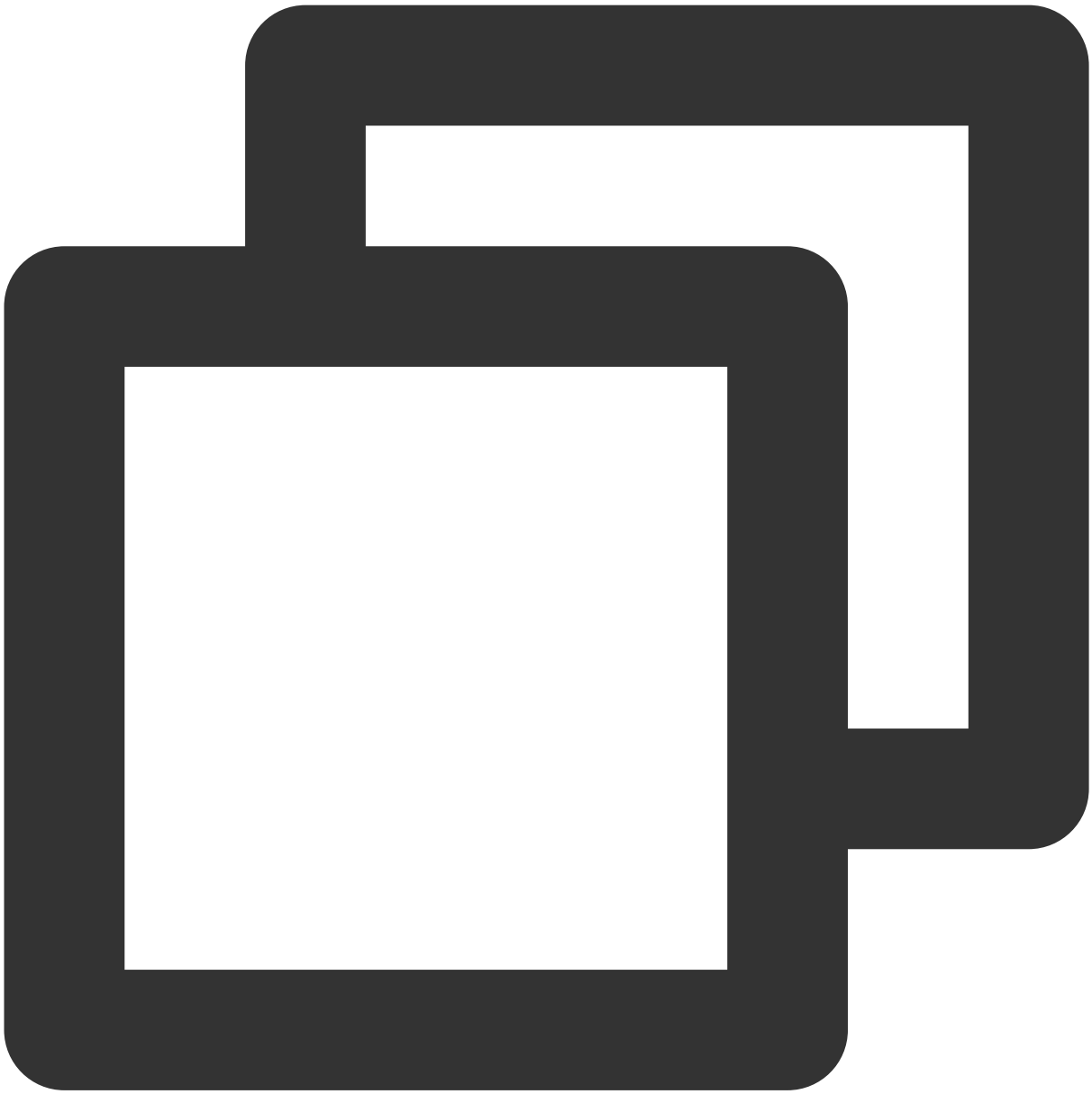


```
Properties props = new Properties();
props.put("bootstrap.servers", "localhost:4242");
props.put("acks", "all");
props.put("retries",0);
props.put("batch.size", 16384);
props.put("linger.ms", 1);
props.put("buffer.memory", 33554432);
props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer
props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializ
Producer<String, String> producer = new KafkaProducer<>(props);
producer.send(new ProducerRecord<String, String>("my-topic", Integer.toString(0), I
```

©2013-2022 Tencent Cloud. All rights reserved.

```
producer.close();
```

Old Producer API Demo



```
Properties props = new Properties();
props.put("metadata.broker.list", "broker1:9092");
props.put("serializer.class", "kafka.serializer.StringEncoder");
props.put("partitioner.class", "example.producer.SimplePartitioner");
props.put("request.required.acks", "1");
ProducerConfig config = new ProducerConfig(props);
Producer<String, String> producer = new Producer<String, String>(config);
KeyedMessage<String, String> data = new KeyedMessage<String, String>("page_visits",
```

©2013-2022 Tencent Cloud. All rights reserved.

```
producer.send(data);
producer.close();
```

As shown in the previous code, the basic usage of the new and old versions are the same, except for some parameter settings. This means the cost of API partial rewriting is not high.

Compatibility description

Producer API 0.8.x can be connected to CKafka successfully without partial rewriting. We recommend using the New Kafka Producer API.

Kafka Consumer

Overview

The open-source Apache Kafka 0.8 provides two types of the Consumer API:
High Level Consumer API (blocking configuration details)
Simple Consumer API (support for parameter configuration adjustment)
Kafka 0.9.x has introduced the New Consumer API that inherits the features of the two types of the old consumer API (v0.8) and reduces the load on ZooKeeper.
The following describes how to transform the old consumer API (v0.8) to the new consumer API (v0.9).

Comparison between new consumer API and old consumer API

Old consumer API (v0.8)

High Level Consumer API (Demo)
The High Level Consumer API can meet general requirements if you care only about data, except for message offset. This API, built on the consumer group logic, blocks the offset management, and supports Broker fault handling and Consumer load balancing. It allows developers to get started with the Consumer client quickly.
Consider the following when you use the High Level Consumer API:
If the number of consumer threads is greater than the number of partitions, certain consumer threads cannot obtain data.
If the number of partitions is greater than the number of threads, certain threads consume more than one partition.
The changes in partitions and consumers will affect rebalancing.
Low Level Consumer API (Demo)
The Low Level Consumer API is recommended if you need message offset and features like repeated consumption or skip read, or if you want to consume specific partitions and ensure more consumption semantics. But in this case, you need to handle offsets and Broker exceptions.
When using the Low Level Consumer API, you need to:
Track and maintain the offset and control the consumption progress.

©2013-2022 Tencent Cloud. All rights reserved.

Find the leader of partitions for the topic, and deal with partition changes.

New consumer API (v0.9)

Kafka 0.9.x has introduced the New Consumer API that inherits the features of the two types of Old Consumer API while providing consumer coordination (High Level API) and lower-level access to customize consumption policies. The New Consumer also simplifies the consumer client and introduces a central coordinator to solve the herd effect and split-brain problems resulting from the separate connections to ZooKeeper, and to reduce the load on ZooKeeper.

Advantages:

Introduces coordinators

The current version of High Level Consumer has the herd effect and split-brain problems. Placing failure detection and rebalancing logics into a highly available central coordinator solves both problems while greatly reducing the load on the ZooKeeper.

Allows you to assign partitions

To keep certain states of each local partition unchanged, you need to keep partition mappings unchanged. Some other scenarios are designed to associate the Consumer with the region-dependent Broker.

Allows you to manage offsets

You can manage offsets as needed to implement repeated consumption, skipped consumption, and other semantics.

Triggers callbacks after rebalancing based on your specifications

Provides non-blocking Consumer API

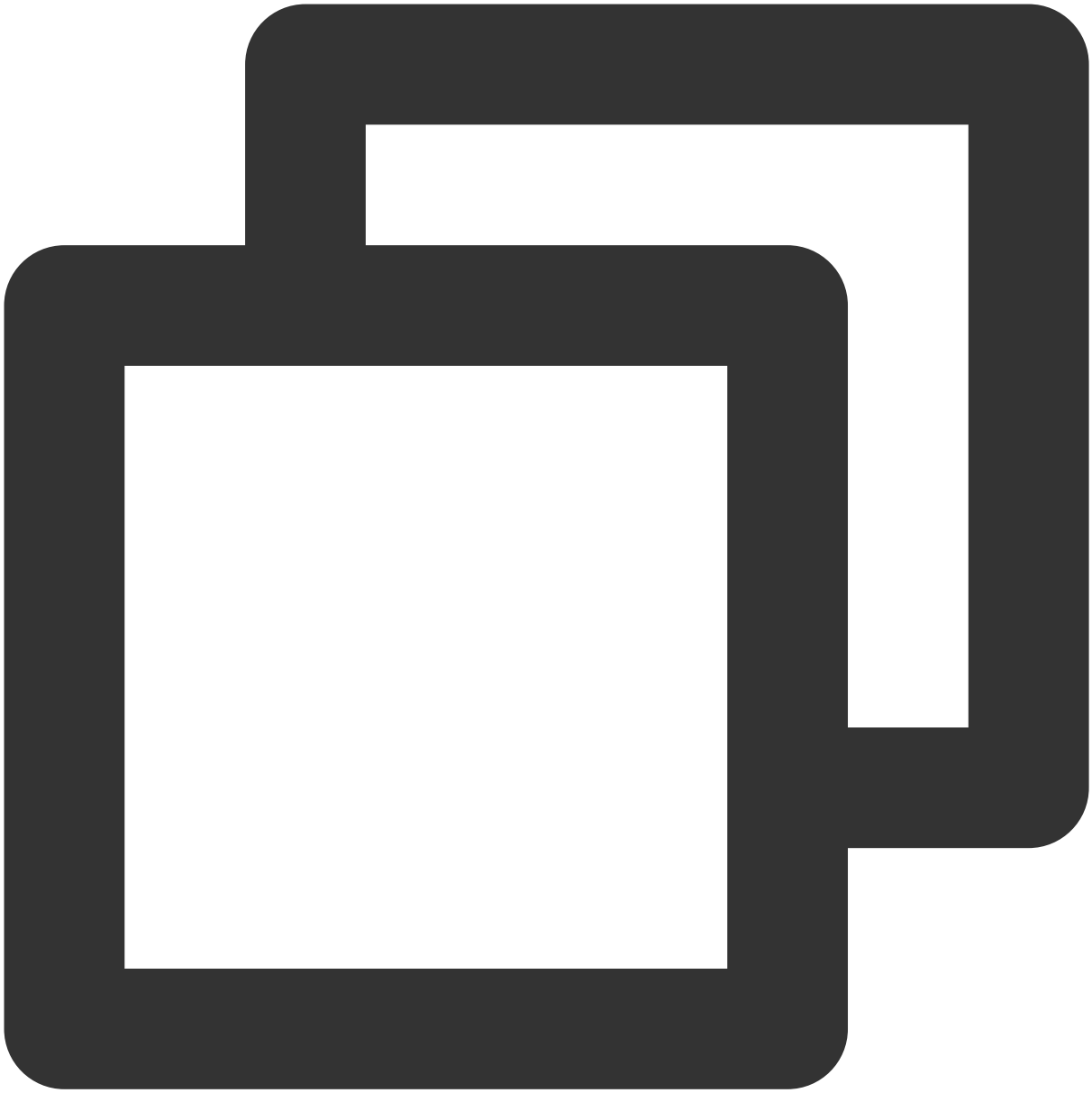
Comparison between new consumer API and old consumer API

Type	Version	Automatic Offset Storage	Manual Offset Management	Automatic Exception Handling	Automatic Rebalance Handling	Automatic Leader Search	Pros and Cons
High Level Consumer	Earlier than v0.9	Yes	No	Yes	Yes	Yes	Herd effect and split br
Simple Consumer	Earlier than v0.9	No	Yes	No	No	No	Various exceptions need to be handled
New Consumer	Later than v0.9	Yes	Yes	Yes	Yes	Yes	Mature and recommend for the curre version

Transforming old consumer to new consumer

New Consumer

©2013-2022 Tencent Cloud. All rights reserved.

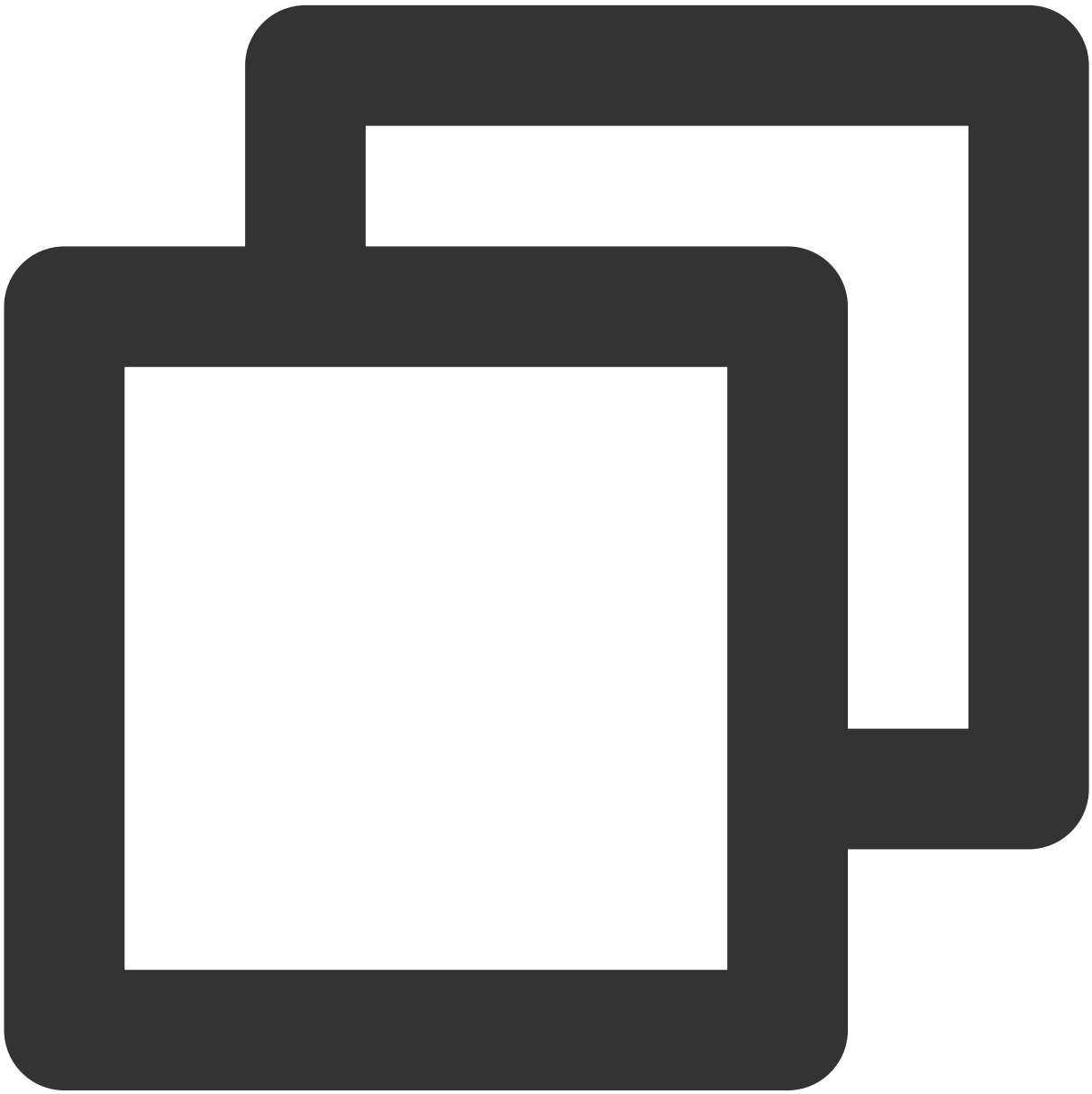


```
//The main configuration difference is that the ZooKeeper parameters are replaced.
Properties props = new Properties();
props.put("bootstrap.servers", "localhost:9092");
props.put("group.id", "test");
props.put("enable.auto.commit", "true");
props.put("auto.commit.interval.ms", "1000");
props.put("session.timeout.ms", "30000");
props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserial
props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeseri
//Compared with old consumers, new consumers are easier to create.
KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
consumer.subscribe(Arrays.asList("foo", "bar"));
while (true) {
    ConsumerRecords<String, String> records = consumer.poll(100);
    for (ConsumerRecord<String, String> record : records)
       System.out.printf("offset = %d, key = %s, value = %s", record.offset(), reco
```

Old Consumer （High Level）



```
//Old consumers require ZooKeeper
Properties props = new Properties();
props.put("zookeeper.connect", "localhost:2181");
props.put("group.id", "test");
```

©2013-2022 Tencent Cloud. All rights reserved.

```
props.put("auto.commit.enable", "true");
props.put("auto.commit.interval.ms", "1000");
props.put("auto.offset.reset", "smallest");
ConsumerConfig config = new ConsumerConfig(props);
//Require connector creation
ConsumerConnector connector = Consumer.createJavaConsumerConnector(config);
//Create a message stream
Map<String, Integer> topicCountMap = new HashMap<String, Integer>();
topicCountMap.put("foo", 1);
Map<String, List<KafkaStream<byte[], byte[]>>> streams =
 connector.createMessageStreams(topicCountMap);
//Obtain data
KafkaStream<byte[], byte[]> stream = streams.get("foo").get(0);
ConsumerIterator<byte[], byte[]> iterator = stream.iterator();
MessageAndMetadata<byte[], byte[]> msg = null;
while (iterator.hasNext()) {
     msg = iterator.next();
     System.out.println(//
            " group " + props.get("group.id") + //
            ", partition " + msg.partition() + ", " + //
             new String(msg.message()));
}
```

Comparing with the old consumer, the new consumer has simpler coding and uses Kafka addresses instead of ZooKeeper parameters. In addition, the new consumer has parameter settings for interactions with coordinators, in which the default settings are suitable for general use.

Compatibility description

Both CKafka and the new version of Kafka in the open-source community support the rewritten new consumer API, which blocks the interaction between the consumer client and ZooKeeper (ZooKeeper is not exposed to users any longer). The new consumer solves the herd effect and split brain problems resulting from direct interaction with ZooKeeper, and integrates the features of the old consumer, thus making the consumption more reliable.

©2013-2022 Tencent Cloud. All rights reserved.

Suggestions for CKafka Version Selection

Last updated：2024-01-09 15:02:48

This document describes the compatibility of CKafka with open-source Kafka and helps you select a CKafka version that is more suitable for your business according to your business needs.

Overview

Open-Source Kafka has about 20 versions ranging from v0.7.x to v2.8.x. From the perspective of message queue, it can be divided into three stages: v0.x, v1.x, and v2.x. At present, Tencent Cloud provides four corresponding versions for these three community development stages: v0.10, v1.1, v2.4, and v2.8, which basically cover commonly used Kafka versions.
Among them, the two major versions v1.x and v2.x mainly contain optimizations and improvements of Kafka Streams but don't introduce many major features in terms of the message engine (v2.x has great improvements of the transaction feature though). Kafka Streams is greatly improved on v2.x. Therefore, if you need to use these features, please select v2.x at least.

Compatibility Description

CKafka is perfectly compatible with open-source Kafka with full backward compatibility with lower versions. For example, if you build Kafka v0.10 on your own, you can select CKafka v0.10, v1.1.1, or v2.4.1 in the cloud, but if you build Kafka on a higher version, we recommend you not select a lower version (because it is uncertain whether your business uses features of higher versions).
The following describes the compatibility:

CKafka Version	Compatible Community Versions	Compatibility
0.10.2	≤ 0.10.x	100%
1.1.1	≤ 1.1.x	100%
2.4.1	≤ 2.4.x	100%
2.8.1	≤ 2.8.x	100%

Note for CKafka v2.4.1
When CKafka launched v2.4, the stable branch in the community was v2.4.1. Later, the community launched a development branch v2.4.2. After some repairs were merged, it was positioned as v2.4.2. Then, the community

©2013-2022 Tencent Cloud. All rights reserved.

deleted v2.4.2, so there was no v2.4.2 in the community eventually. Therefore, the v2.4.2 previously displayed by CKafka is aligned with the current v2.4.1.

Suggestions for CKafka Version Selection

If you migrate your self-built Kafka to the cloud, we recommend you select the corresponding major version. For example, if your self-built Kafka is on v1.1.0, please select CKafka v1.1.
If the corresponding version cannot be found in the cloud, we recommend you select a higher version. For example, if your self-built Kafka is on v1.0.0, we recommend you use CKafka v1.1.1, and if it is on v0.11.x, we also recommend you use CKafka v1.1.1 (because each version of Broker is backward compatible).

©2013-2022 Tencent Cloud. All rights reserved.

CKafka Data Reliability Description

Last updated：2024-01-09 15:02:47

This document describes the factors that affect the reliability of CKafka from the perspectives of the producer, the server (CKafka), and the consumer, respectively, and provides corresponding solutions.

What should I do if data gets lost on the producer?

Causes of data loss

When the producer sends data to CKafka, the data may get lost due to network jitters, and CKafka will not receive the data. Other possible causes are as follows:
The network load is high or the disk is busy, and the producer does not have a retry mechanism.
The purchased disk capacity is exceeded. For example, if the disk capacity of an instance is 9,000 GB and it is not expanded promptly after being used up, data cannot be written to CKafka.
Sudden or continuously increased peak traffic exceeds the purchased peak throughput. For example, if the peak throughput of the instance is 100 MB/sec, but it is not scaled up promptly after the peak throughput is exceeded for a long period of time, data writes to CKafka will become slower. In this case, if the producer has a queuing timeout mechanism in place, data cannot be written to CKafka.

Solutions

Enable the retry mechanism on the producer for important data.
To avoid data loss caused by improper disk usage, set monitoring and alarm policies as preventive measures when configuring the instance.
When the disk capacity is used up, upgrade the instance timely in the console. Upgrading Ckafka instances of Standard Edition will not interrupt the service. The disk capacity can be expanded separately. You can also shorten the message retention period to reduce disk usage.
To minimize the loss of messages on the producer, you can fine-tune the size of the buffer by using `buffer.memory` and `batch.size` (in bytes). A larger buffer is not necessarily better. When the producer fails for any reason, more data in the buffer means more garbage to be recycled, which slows down data recovery. **Pay close attention to the number of messages produced by the producer and the average message size** (through the rich set of monitoring metrics available in CKafka).
Configure acknowledgment (ACK) for the producer.
When the producer sends data to the leader, it can set the data reliability level by using the `request.required.acks` and `min.insync.replicas` parameters.
When `acks = 1` (default value), the leader in the ISR has successfully received a message sent by the producer, and the next message can be sent. If the leader goes down, the data unsynced to its followers will get lost.

©2013-2022 Tencent Cloud. All rights reserved.

When `acks = 0`, the producer sends the next message without waiting for acknowledgment from the broker. In this case, data transfer efficiency is the highest, but data reliability is the lowest.

Note:

When the producer is configured with `acks = 0`, if the current instance is throttled, in order for the server to provide services normally, the server will actively close the connection to the client.

When `acks = -1` or `acks = all`, the producer needs to wait for the acknowledgment of message receipt from all the followers in the ISR before sending the next message, which ensures the highest reliability.

Even if `acks` is configured as above, there is no guarantee that data will never get lost. For example, when there is only one leader in the ISR (the number of members in the ISR may increase or decrease in certain circumstances, and in some cases, only one leader is left), the value of `acks` will be `1`. Therefore, you also need to configure the `min.insync.replicas` parameter in the CKafka console by enabling the advanced configuration in **Topic Management** > **Edit Topic**. This parameter specifies the minimum number of replicas in the ISR, and its default value is 1. It only takes effect when `acks = -1` or `acks = all`.

Recommended parameter values

These parameter values are for reference only, and the actual values depend on the actual conditions of your business.

Retry mechanism: `message.send.max.retries=3;retry.backoff.ms=10000;`

Guarantee of high reliability: `request.required.acks=-1;min.insync.replicas=2;`

Guarantee of high performance: `request.required.acks=0;`

Reliability + performance: `request.required.acks=1;`

What should I do if data gets lost on the broker (CKafka)?

Causes of data loss

The partition's leader goes down before the followers complete the data backup. Even if a new leader has been selected, data will get lost because it has not been backed up yet.

Open-source Kafka stores data to disks in an async manner. Specifically, data is first stored in PageCache before persistence. If the broker disconnects, restarts, or fails, the data stored in PageCache will get lost because it has not been stored persistently to the disks yet.

Stored data may get lost due to disk failures.

Solutions

Open-source Kafka has multiple replicas that are used to ensure data integrity. Data will get lost only if multiple replicas and brokers fail at the same time, so data reliability is higher than that in the single-replica case. Therefore, CKafka requires at least two replicas for a topic and supports configuring three replicas.

CKafka performs data flushing by configuring more reasonable parameters, such as `log.flush.interval.messages` and `log.flush.interval.ms`.

©2013-2022 Tencent Cloud. All rights reserved.

In CKafka, the disk is specially designed to ensure that data reliability will not be compromised even if the disk is partially damaged.

Recommended parameter values

Whether a replica that is not in sync status can be elected as a leader:

```
unclean.leader.election.enable=false // Disabled
```

What should I do if data gets lost on the consumer?

Causes of data loss

The offset is committed before data is consumed. If the consumer goes down in the process but the offset has been updated, the consumer will miss a data entry, and the consumer group will have to reset the offset in order to retrieve it.
The consumption speed differs significantly from the production speed, but the message retention period is too short; therefore, the message will be deleted upon expiration before it is consumed.

Solutions

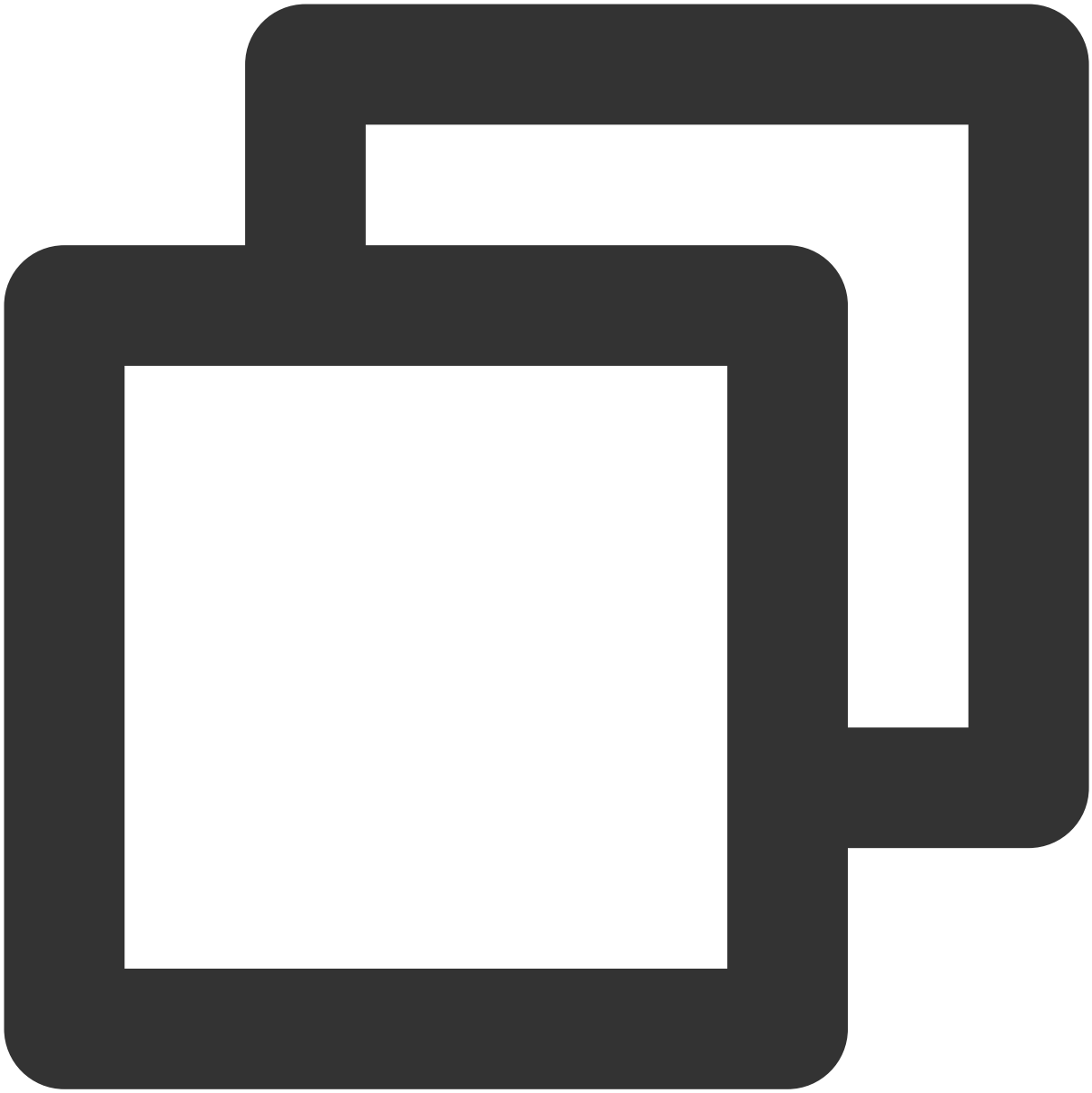
Configure the `auto.commit.enable` parameter appropriately. When it is set to `true`, the commit is performed automatically. We recommend that you use the scheduled commit feature to avoid committing offsets frequently.
Monitor the consumer and correctly adjust the data retention period. Monitor the consumption offset and the number of unconsumed messages, and configure an alarm to prevent messages from being deleted upon expiration due to slow consumption.

Troubleshooting data loss

Printing partition and offset information locally for troubleshooting

Below is the code for printing information:

©2013-2022 Tencent Cloud. All rights reserved.



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

If the partition and offset can be printed out, the currently sent message has been correctly saved on the server. At this time, you can use the message query tool to query the information of the relevant offset.

If the partition and offset information cannot be printed out, the message has not been saved on the server, and the client needs to retry.

©2013-2022 Tencent Cloud. All rights reserved.

Connector
Database Change Subscription
Canal Format of MySQL Subscription Message

Last updated：2024-01-09 15:02:47

Overview

When using CKafka Connector to subscribe to change operations in MySQL, you can select multiple message formats, and the default format is Debezium. In addition, the system provides compatibility with other message formats. This document describes the message formats compatible with the **official custom format**.

Official Format 1

Official format 1 currently is supported only for DML messages, while the DDL message format is the same as the Canal format.

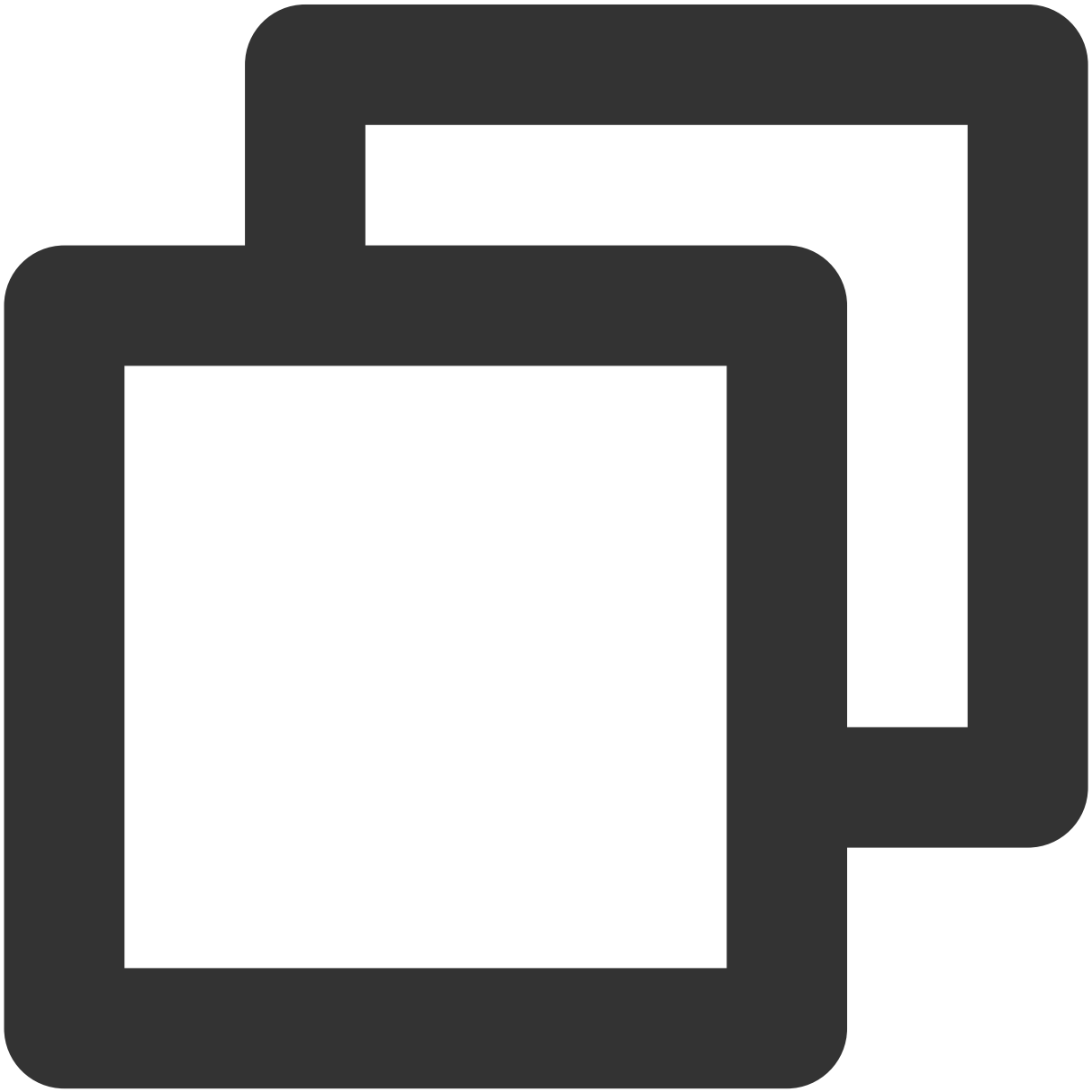
Field	Description
BINLOG_NAME	Binlog filename
BINLOG_POS	Binlog position
DATABASE	Database name
EVENT_SERVER_ID	It is null currently
GLOBAL_ID	GTID information if GTID is enabled
GROUP_ID	It is null currently
NEW_VALUES	If `type` is `U`, it is the row information after the update in JSON format. If `type` is `D`, it is null. If `type` is `I`, it is the inserted row information in JSON format.
OLD_VALUES	If `type` is `U`, it is the row information before the update in JSON format. If `type` is `D`, it is the deleted row information in JSON format. If `type` is `I`, it is null.

©2013-2022 Tencent Cloud. All rights reserved.

TABLE	Table name
TIME	Log generation time
TYPE	Log type. Valid values: `U` (UPDATE); `D` (DELETE); `I` (INSERT).

DDL Format

create database

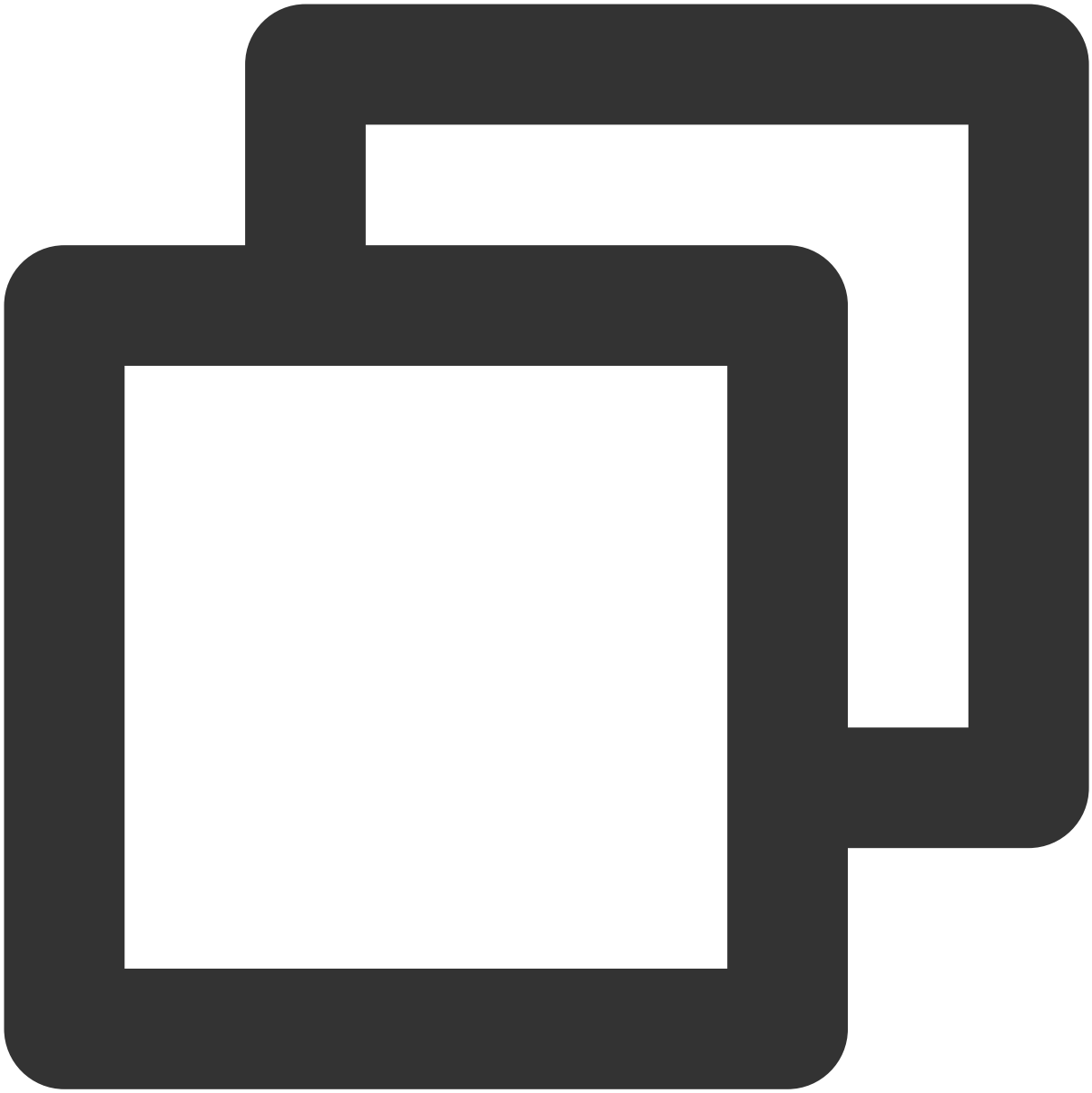


©2013-2022 Tencent Cloud. All rights reserved.

```
{
    "data": null,
    "database": "dip_test",
    "es": 1655812326,
    "id": 0,
    "isDdl": true,
    "mysqlType": null,
    "old": null,
    "pkNames": null,
    "sql": "CREATE DATABASE `dip_test` CHARSET utf8mb4 COLLATE utf8mb4_0900_ai_ci",
    "sqlType": null,
    "table": "",
    "ts": 1655812326,
    "type": "QUERY"
}
```


drop database

©2013-2022 Tencent Cloud. All rights reserved.

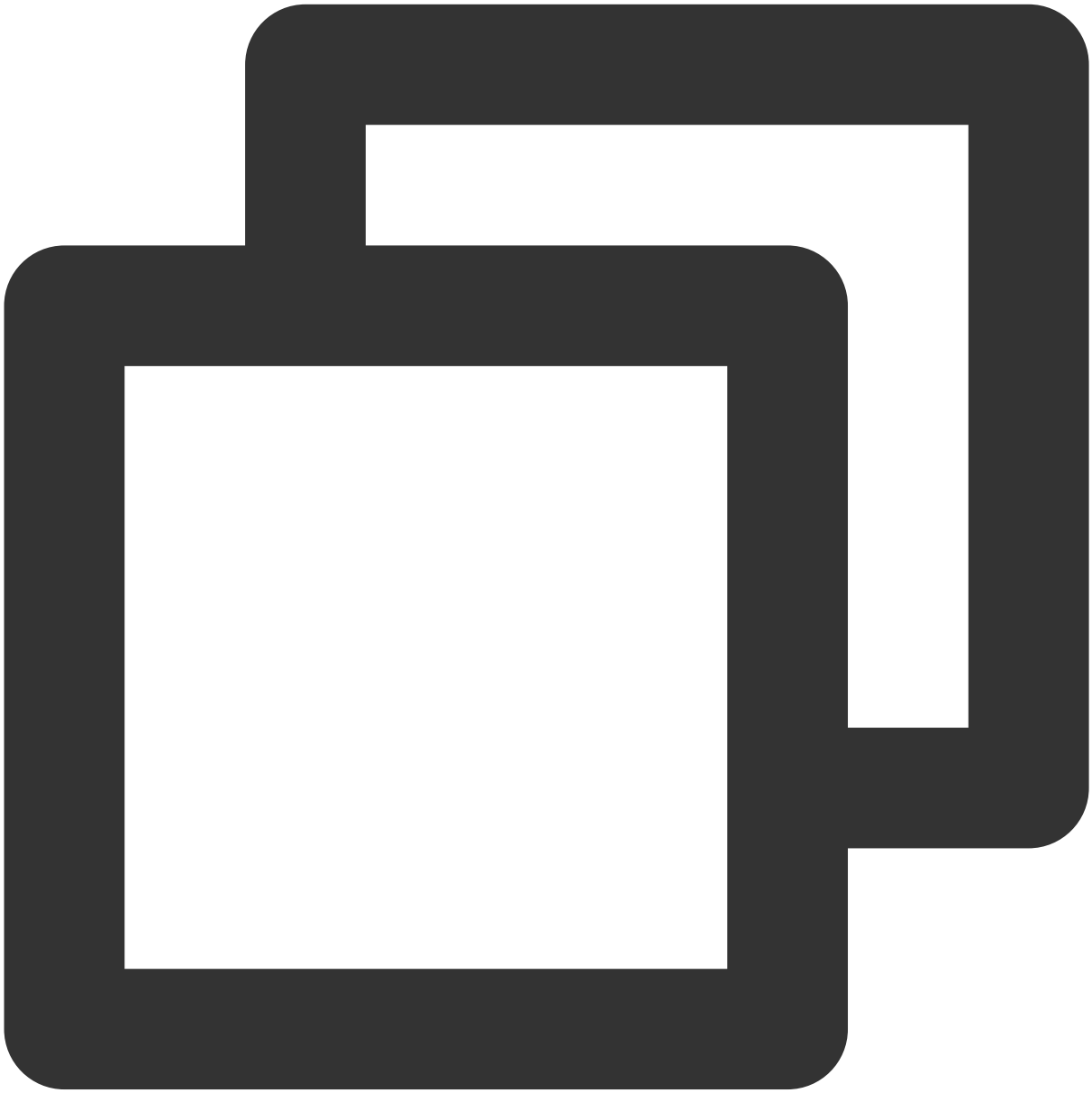


```
{
    "data": null,
    "database": "dip_test",
    "es": 1655812326,
    "id": 0,
    "isDdl": true,
    "mysqlType": null,
    "old": null,
    "pkNames": null,
    "sql": "DROP DATABASE IF EXISTS `dip_test`",
    "sqlType": null,
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    "table": "",
    "ts": 1655812326,
    "type": "QUERY"
}
```

create table



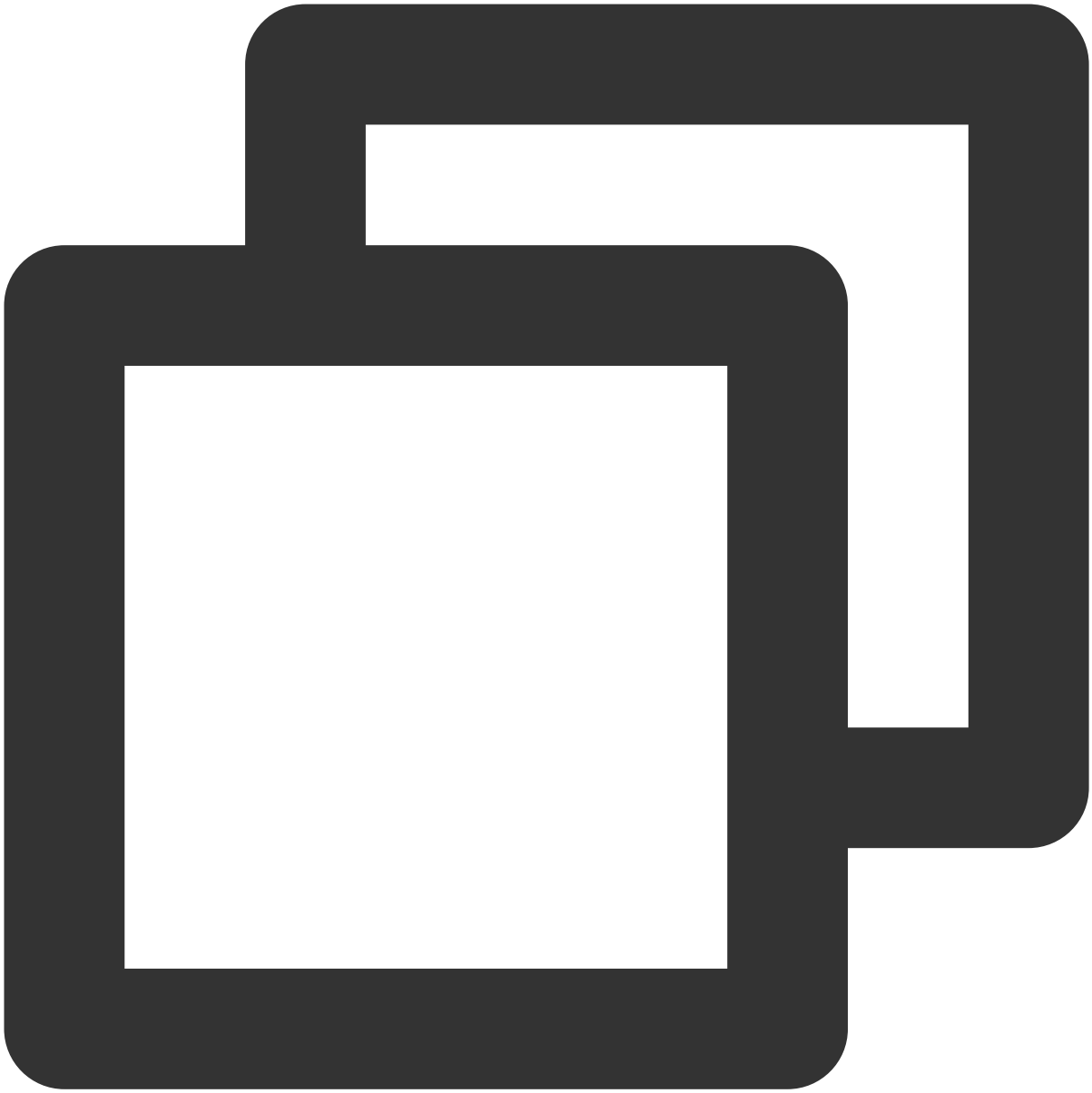
```
{
    "data": null,
    "database": "dip_test",
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    "es": 1655812326,
    "id": 0,
    "isDdl": true,
    "mysqlType": null,
    "old": null,
    "pkNames": null,
    "sql": "CREATE TABLE `customers` (
  `id` int NOT NULL AUTO_INCREMENT,
  `first_name` varchar(255) NOT NULL,
  `last_name` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`),
  KEY `ix_id` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1041 DEFAULT CHARSET=utf8",
    "sqlType": null,
    "table": "customers",
    "ts": 1655812326,
    "type": "CREATE"
}
```


alter table

©2013-2022 Tencent Cloud. All rights reserved.

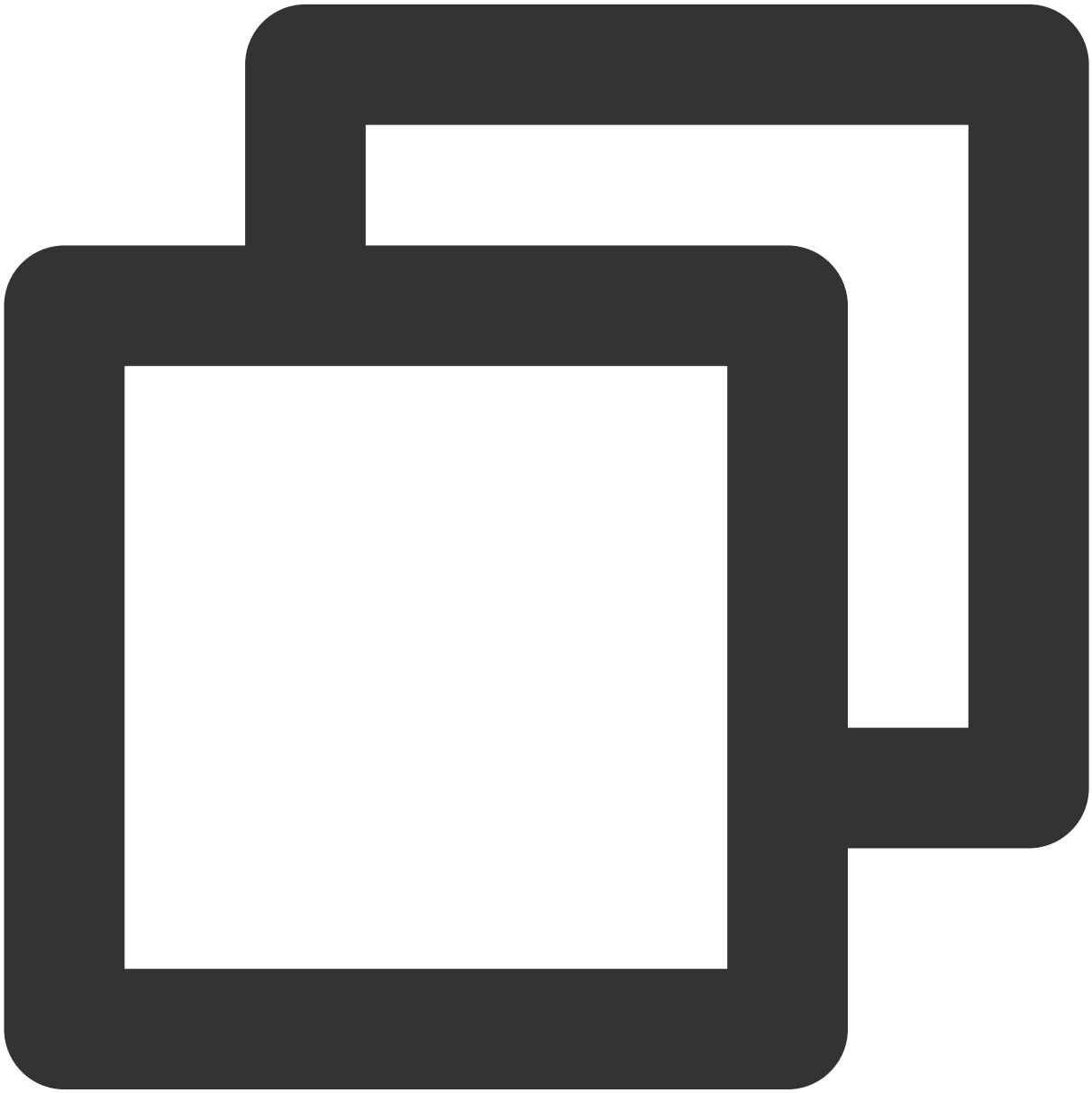



```
{
    "data": null,
    "database": "test",
    "es": 1655782153,
    "id": 0,
    "isDdl": true,
    "mysqlType": null,
    "old": null,
    "pkNames": null,
    "sql": "ALTER TABLE `user` ADD COLUMN `createtime` datetime NULL DEFAULT CURREN
    "sqlType": null,
```


©2013-2022 Tencent Cloud. All rights reserved.

```
    "table": "user",
    "ts": 1655782153,
    "type": "ALTER"
}
```

drop table



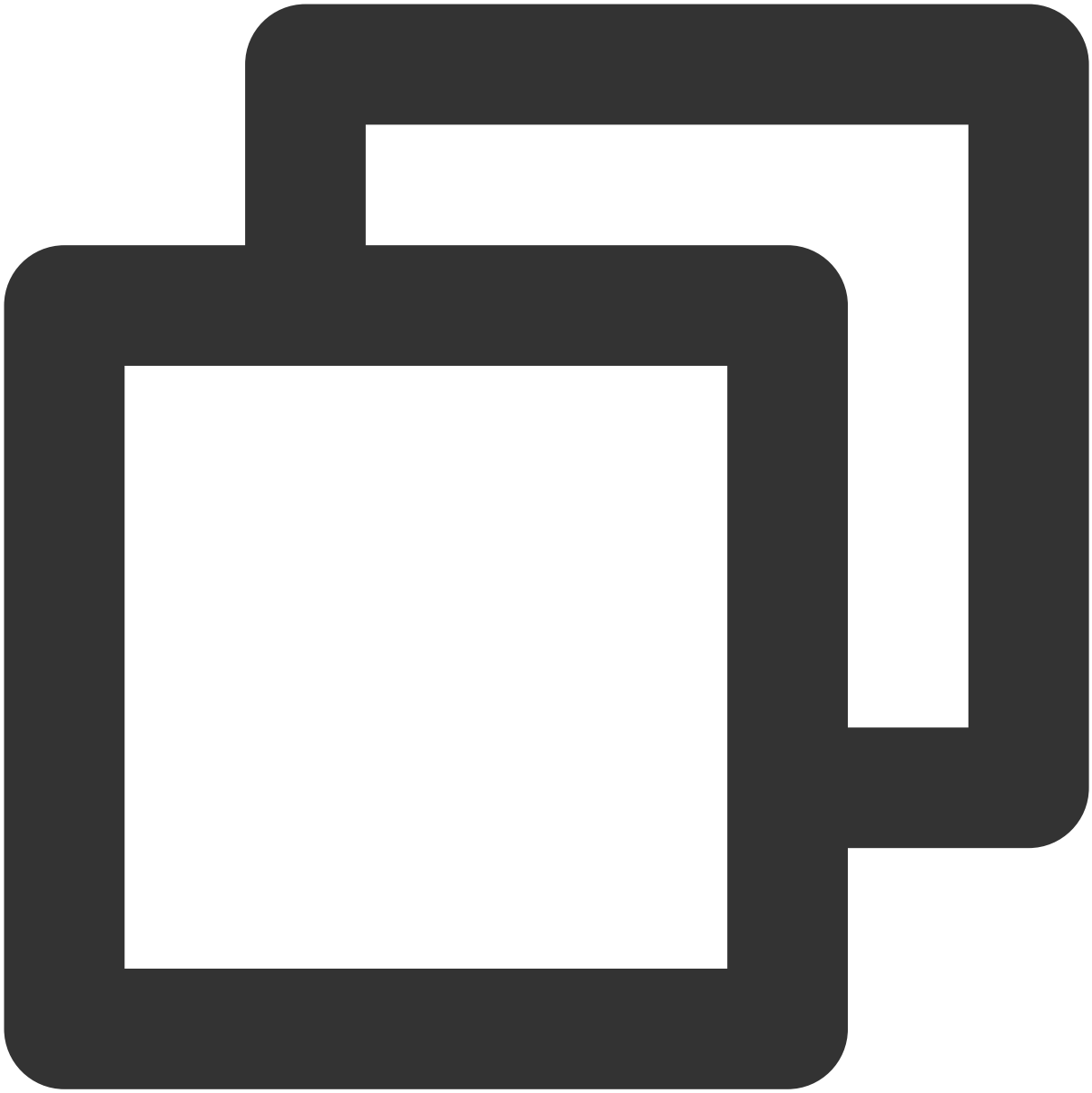
```
{
    "data": null,
    "database": "dip_test",
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    "es": 1655812326,
    "id": 0,
    "isDdl": true,
    "mysqlType": null,
    "old": null,
    "pkNames": null,
    "sql": "DROP TABLE IF EXISTS `dip_test`.`customers`",
    "sqlType": null,
    "table": "customers",
    "ts": 1655812326,
    "type": "ERASE"
}
```

rename table

©2013-2022 Tencent Cloud. All rights reserved.



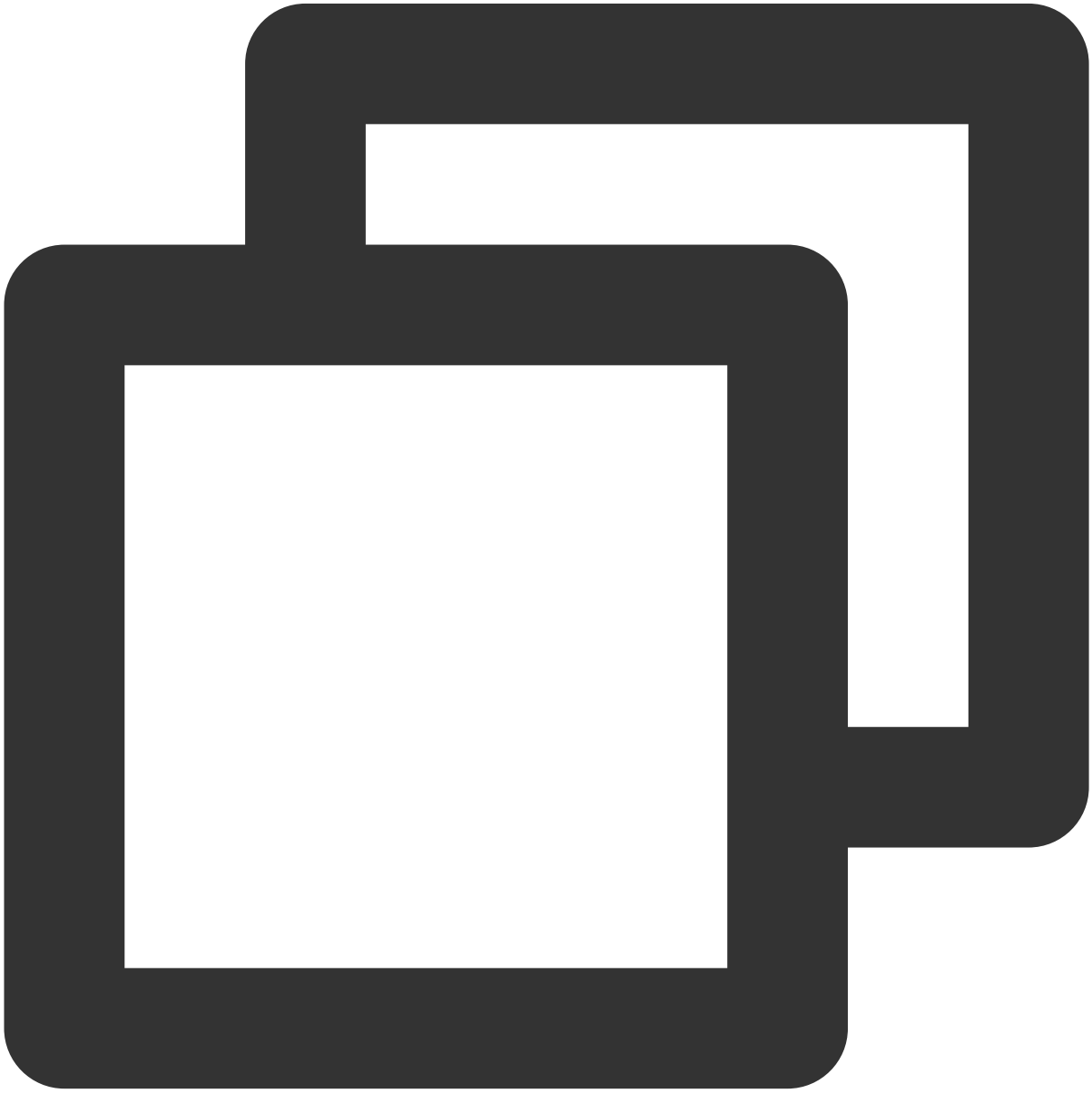
```
{
    "data": null,
    "database": "testDB",
    "es": 1656300979748,
    "id": 0,
    "isDdl": true,
    "mysqlType": null,
    "old": null,
    "pkNames": null,
    "sql": "rename table test to t_test",
    "sqlType": null,
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    "table": "t_test",
    "ts": 1656300979748,
    "type": "RENAME"
}
```

DML Format

insert

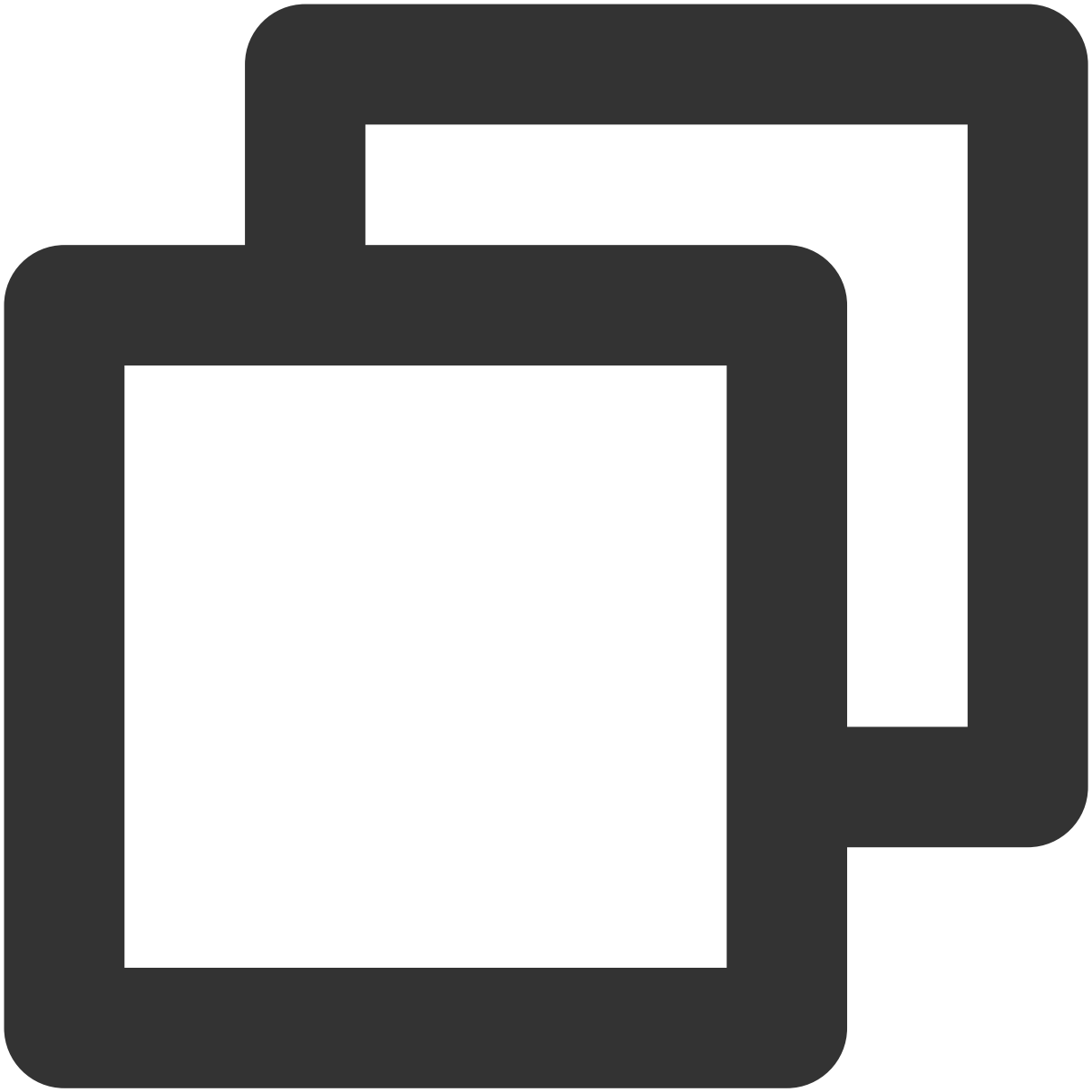


©2013-2022 Tencent Cloud. All rights reserved.

```
{
    "BINLOG_NAME": "mysql-bin.000003",
    "BINLOG_POS": 154,
    "DATABASE": "inventory",
    "EVENT_SERVER_ID": null,
    "GLOBAL_ID": null,
    "GROUP_ID": null,
    "NEW_VALUES": {
        "last_name": "Kretchmar",
        "id": "1004",
        "first_name": "Anne",
        "email": "annek@noanswer.org"
    },
    "OLD_VALUES": null,
    "TABLE": "customers",
    "TIME": "19700101080000",
    "TYPE": "I"
}
```

update

©2013-2022 Tencent Cloud. All rights reserved.



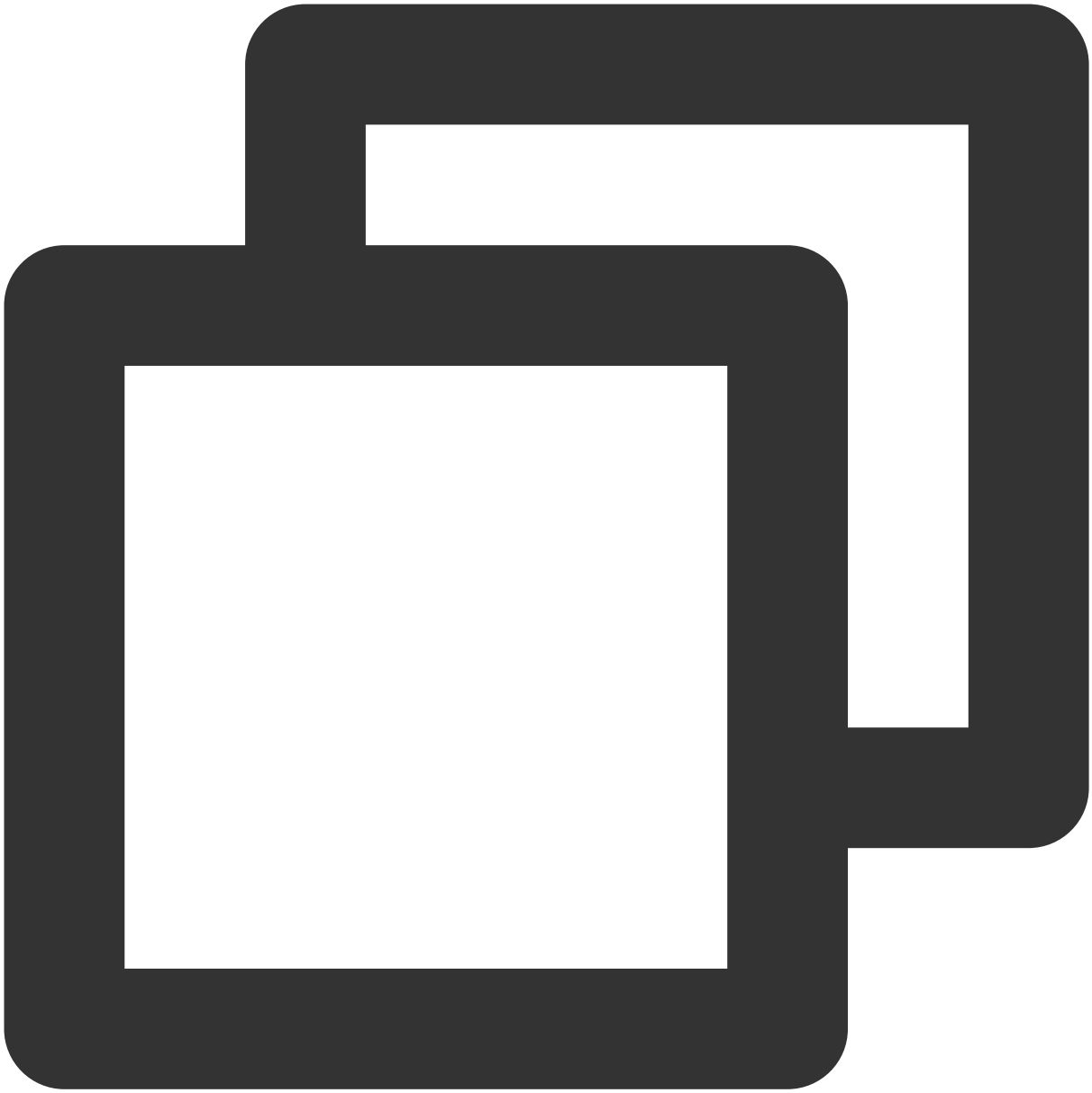

```
{
    "BINLOG_NAME": "mysql-bin.000003",
    "BINLOG_POS": 484,
    "DATABASE": "inventory",
    "EVENT_SERVER_ID": null,
    "GLOBAL_ID": null,
    "GROUP_ID": null,
    "NEW_VALUES": {
        "last_name": "Kretchmar",
        "id": "1004",
        "first_name": "Anne Marie",
```


©2013-2022 Tencent Cloud. All rights reserved.

```
        "email": "annek@noanswer.org"
    },
    "OLD_VALUES": {
        "last_name": "Kretchmar",
        "id": "1004",
        "first_name": "Anne",
        "email": "annek@noanswer.org"
    },
    "TABLE": "customers",
    "TIME": "20160611015029",
    "TYPE": "U"
}
```

delete

©2013-2022 Tencent Cloud. All rights reserved.




```
{
    "BINLOG_NAME": "mysql-bin.000003",
    "BINLOG_POS": 805,
    "DATABASE": "inventory",
    "EVENT_SERVER_ID": null,
    "GLOBAL_ID": null,
    "GROUP_ID": null,
    "NEW_VALUES": null,
    "OLD_VALUES": {
        "last_name": "Kretchmar",
        "id": "1004",
```


©2013-2022 Tencent Cloud. All rights reserved.

```
        "first_name": "Anne Marie",
        "email": "annek@noanswer.org"
    },
    "TABLE": "customers",
    "TIME": "20160611020502",
    "TYPE": "D"
}
```

©2013-2022 Tencent Cloud. All rights reserved.