Distributed Message Service for Kafka

Troubleshooting

Issue	01
Date	2024-11-20

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

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

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

Contents

1 Troubleshooting Kafka Connection Exceptions

Overview

This section describes how to troubleshoot Kafka connection problems.

Problem Classification

If the connection to a Kafka instance is abnormal, perform the following operations to troubleshoot the problem:

- **Checking the Network**
- **Checking Consumer and Producer Configurations**
- **Checking for Common Errors on Java Clients**
- **Checking for Common Errors on the Go Client**

Checking the Network

Ensure that the client and the Kafka instance can be connected. If they cannot be connected, check the network.

For example, if you have enabled SASL for the Kafka instance, run the following command:

curl -kv *{ip}*:*{port}*

- If the network is normal, information similar to the following is displayed:

```
[root@ecs-5d2f ~]# curl -kv 192.168.0.52:9093
* Rebuilt URL to: 192.168.0.52:9093/
*   Trying 192.168.0.52...
* TCP_NODELAY set
* Connected to 192.168.0.52 (192.168.0.52) port 9093 (#0)
> GET / HTTP/1.1
> Host: 192.168.0.52:9093
> User-Agent: curl/7.61.1
> Accept: */*
>
Warning: Binary output can mess up your terminal. Use "--output -" to tell
Warning: curl to output it to your terminal anyway, or consider "--output
Warning: <FILE>" to save to a file.
* Failed writing body (0 != 7)
* Closing connection 0
```


- If the network is abnormal or disconnected, information similar to the following is displayed:

Solution:

1. Check whether the client and the Kafka instance are in the same VPC. If no, **establish a VPC peering connection**.
2. Check whether the security group rules are correctly configured. For details, see **How Do I Select and Configure a Security Group?**

Checking Consumer and Producer Configurations

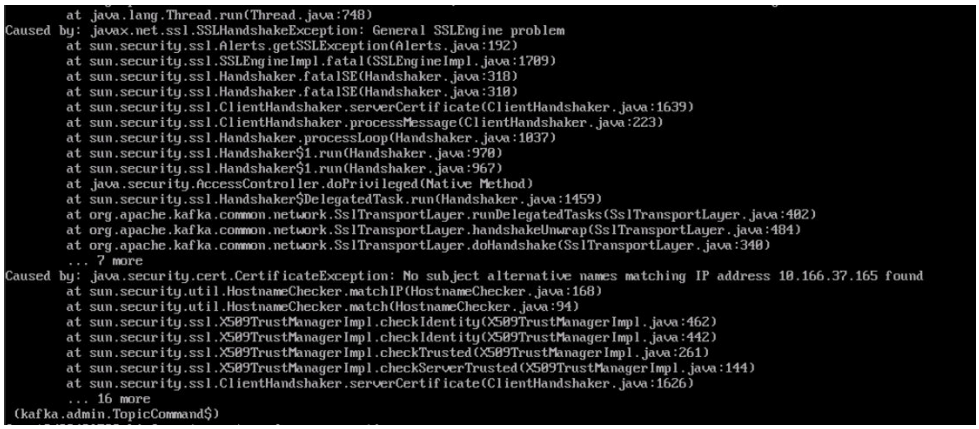
View logs to check whether the parameters printed during initialization of the consumer and producer are the same as those set in the configuration files.

If they are different, check the parameters in the configuration files.

Checking for Common Errors on Java Clients

- Error 1: Domain name verification is not disabled.

  The following error information is displayed:

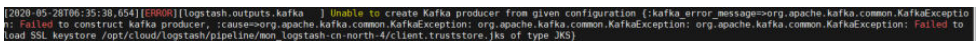


  Solution: Leave the **ssl.endpoint.identification.algorithm** parameter in the **consumer.properties** and **producer.properties** files empty to disable domain name verification.

```
ssl.endpoint.identification.algorithm=
```

- Error 2: SSL certificates fail to be loaded.

  The following error information is displayed:



  Solution:

  a. Check whether the **client.jks** file exists in the corresponding address.

  b. Check the permissions on the processes and files.

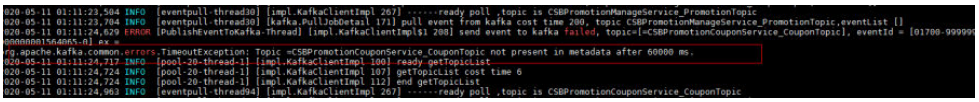
c. Check whether the **ssl.truststore.password** parameter in the **consumer.properties** and **producer.properties** files is correctly set.

ssl.truststore.password is the server certificate password, which must be set to **dms@kafka** and cannot be changed.

```
ssl.truststore.password=dms@kafka
```

- Error 3: The topic name is incorrect.

  The following error information is displayed:



  Solution: Create a new topic or enable the automatic topic creation function.

Checking for Common Errors on the Go Client

The Go client fails to connect to Kafka over SSL and the error "first record does not look like a TLS handshake" is returned.

Solution: If the instance was created before January 2021, enable the TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 or TLS_RSA_WITH_AES_128_CBC_SHA256 cipher suite (both are disabled by default). If the instance was created in or after January 2021, enable the TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 cipher suite.

2 Troubleshooting 6-Min Latency Between Message Creation and Retrieval

Symptom

The duration from message creation to retrieval occasionally reaches 6 minutes, which is not tolerable to services.

Possible Causes

1. Service requests are stacked and cannot be processed in time.

   According to the monitoring data, only up to 50 messages are stacked and up to 10 messages are created per second, which is within the processing capability limit, so this is not the cause of the symptom.

2. The EIP inbound traffic decreases.

   If the EIP technical support personnel cannot find any problem, this is not the cause of the symptom.

3. The consumer group is behaving abnormally.

   According to the server logs, the consumer group is going through frequent rebalance operations. While most rebalance operations are completed within seconds, some can take several minutes. Messages cannot be retrieved until the rebalance is complete.

   This is the cause of the symptom.

Detailed Analysis

A consumer group may exhibit the following three types of behavior in the log:

- **Preparing to rebalance group 1**

  The consumer group starts rebalance, and its status changes to **REABLANCING**.

- **Stabilized group**

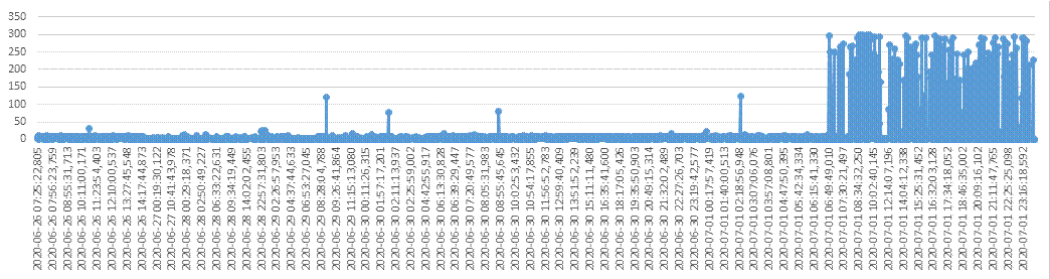
  The consumer group completes rebalance, and its status changes to **STABILIZED**.

- **Member consumer-xxx in group 1 has failed**

A consumer in a consumer group leaves the group if **the consumer has not communicated with the server for a long time**. This is usually triggered if the message processing is prolonged and the process is blocked.

The following figure shows the duration between **Preparing** and **Stabilized**. **The time shown in the figure is UTC+0, which is eight hours later than GMT +08:00.**

Figure 2-1 Consumer group rebalance



This set of data shows that rebalance performance of the consumer group deteriorates after 06:49 on July 1 (or 14:49 on July 1 GMT+08:00). As a result, the client becomes abnormal.

Root Cause

Sometimes, **a consumer cannot respond to rebalancing in a timely manner. As a result, the entire consumer group is blocked** until the consumer responds.

Workaround

1. Use different consumer groups for different services to reduce the impact of a single consumer blocking access.
2. **max.poll.interval.ms** sets the maximum interval for a consumer group to request message consumption. If a consumer does not initiate another consumption request before timeout, the server triggers rebalancing. You can increase the default value of **max.poll.interval.ms**.

Solution

1. Use different consumer groups for different services.
2. Optimize the service processing logic to improve the processing efficiency and reduce the blocking time.

Background Knowledge

A consumer group can be either **REBALANCING** or **STABILIZED**.

- **REBALANCING**: If a consumer joins or leaves a consumer group, the metadata of the consumer group changes and **no consumers in the consumer group can retrieve messages**.
- **STABILIZED**: The metadata has been synchronized by all consumers in the consumer group, including existing ones. Rebalancing has completed and the

consumer group is stabilized. Consumers in the consumer group **can retrieve messages normally**.

A consumer group works as follows:

1. A consumer leaves or joins the group, changing the consumer group metadata recorded at the server. The server updates the consumer group status to **REBALANCING**.
2. The server **waits for all consumers** (including existing ones) to synchronize the latest metadata.
3. After **all consumers** have synchronized the latest metadata, the server updates the consumer group status to **STABILIZED**.
4. Consumers retrieve messages.

3 Troubleshooting Message Creation Failures

Symptom

The system displays the error message "Disk error when trying to access log file on the disk".

Root Cause

The disk usage of the broker is too high.

Solution

Expand the disk space by referring to **Modifying Kafka Instance Specifications**.

4 Troubleshooting Topic Deletion Failures

Symptom

A deleted topic still exists.

Root Cause

Automatic topic creation has been enabled for the instance, and a consumer is connecting to the topic. If services are not stopped, message creation will continue, and new topics will be automatically created.

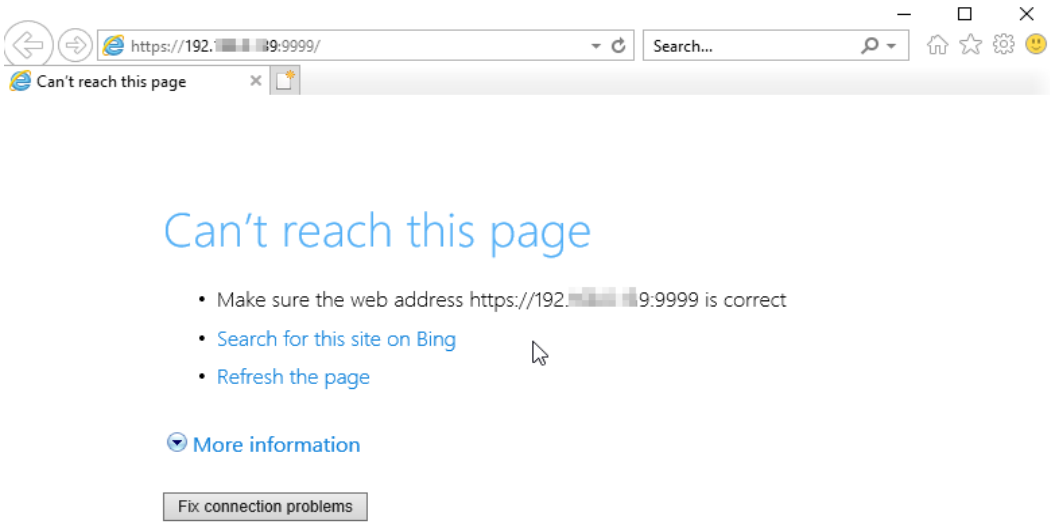
Solution

Disable automatic topic creation for the instance and then try again to delete the topic. For details, see **Configuring Automatic Topic Creation**.

5 Troubleshooting Failure to Log In to Kafka Manager in Windows

Symptom

After the Kafka Manager address is entered in the address box of the browser in Windows, the login fails and an error is displayed.



Root Cause

1. The Windows server and the Kafka instance are not in the same VPC and subnet, or the security group configurations are incorrect.
2. Kafka Manager is abnormal.

Solution

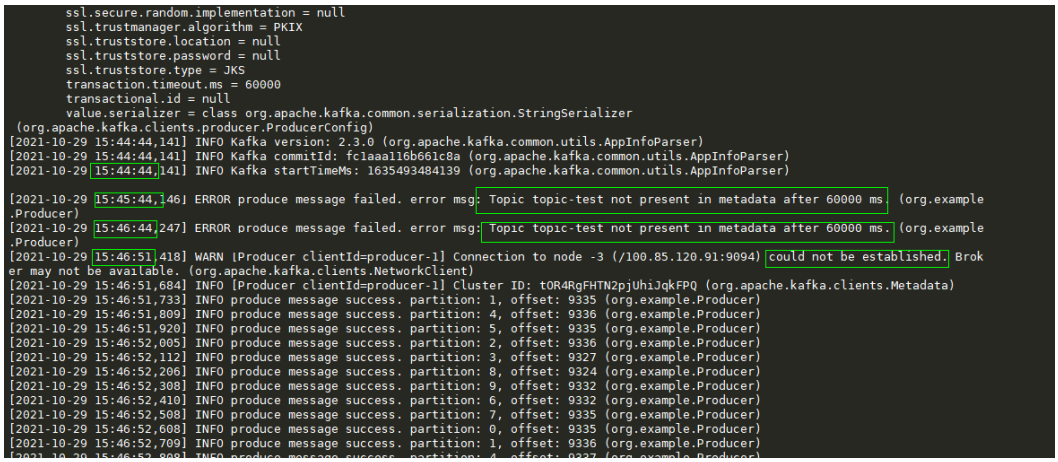
1. Check whether the Windows server and the Kafka instance are in the same VPC and subnet.
   - If they are in the same VPC and subnet, go to **2**.
   - If they are not in the same VPC and subnet, change the VPC and subnet of the Windows server to the same as those of the Kafka instance.

2. Check whether the security group is correctly configured. For details on how to configure a security group, see **How Do I Select and Configure a Security Group?**
   - If the security group is correctly configured, go to **3**.
   - If the security group is not correctly configured, modify the configuration.
3. On the Kafka console, restart Kafka Manager. For details, see **Restarting Kafka Manager**.

6 Troubleshooting Error "Topic {{topic_name}} not present in metadata after 60000 ms" During Message Production or Consumption

Symptom

For a Kafka instance deployed in multiple AZs, if one of the AZs is faulty, error message "Topic {{topic_name}} not present in metadata after 60000 ms" may be reported on the Kafka client during message production or consumption, as shown in the following figure.



Solution

You can use any of the following methods to solve this problem:

- Upgrade the Kafka client to v2.7 or later, and set **socket.connection.setup.timeout.ms** to a value greater than 1s and less than the value of **request.timeout.ms** divided by the number of Kafka server nodes.
- Change the value of **request.timeout.ms** of the Kafka client to a value greater than 127s.

- Change the Linux network parameter **net.ipv4.tcp_syn_retries** of the Kafka client to **3**.

7 Flink 1.15 Consumption Progress Submission Failure

Symptom

To consume Kafka messages in Flink 1.15, the consumption progress fails to be submitted, and the error messages "COORDINATOR_NOT_AVAILABLE" are thrown.

Root Cause

Bug on the Kafka client used by Flink 1.15: When the consumption progress fails to be submitted for once, the client sets the coordinator to unavailable and it cannot be automatically restored.

Solution

- Restart the Flink job.
- Upgrade Flink to 1.16 or later.

