Monitoring End to End Latency Using Streams Messaging Manager

Date published: 2019-09-20
Date modified: 2022-05-12

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

End to end latency overview..... 4

Granularity of metrics for end-to-end latency..... 5

Enabling interceptors..... 5

Monitoring end to end latency for Kafka topic.....7

End to end latency use case.....11

End to end latency overview

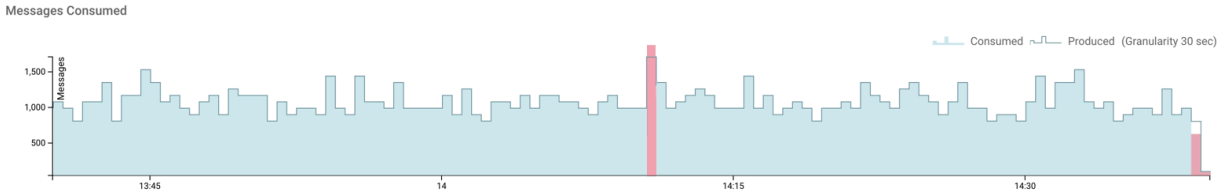
You can use Streams Messaging Manager (SMM) UI to monitor end-to-end latency in Kafka topics. By monitoring latency, you get a complete understanding of the time taken by a consumer to consume a message that is produced in a topic. You can also monitor the number of consumed messages across all the consumer groups for a topic within the selected time range.

Use the latency feature to achieve the following goals:

- Verify whether end-to-end processing time SLAs are met.
- Identify slow or lagging consumers.
- Verify whether messages are overconsumed or underconsumed.

You can find details about the number of messages produced in a topic, the number of messages consumed from a topic, and latency details during the consumption of the messages in the following two graphs in the SMM UI:

- Messages Consumed. The graph provides you the overall produced message count and consumed message count across all the consumer groups for a topic within the selected time range. Any discrepancy in the produced and consumed message count is highlighted in red.



  In the preceeding image, the linear formation represents the number of messages produced in last one hour, and the filled area represents the number of messages consumed in last one hour with a granularity of 30 seconds. The blue area signifies that all the produced messages are consumed. The red area represents a discrepancy in the produced and consumed message count and can either mean that messages are overconsumed or underconsumed.

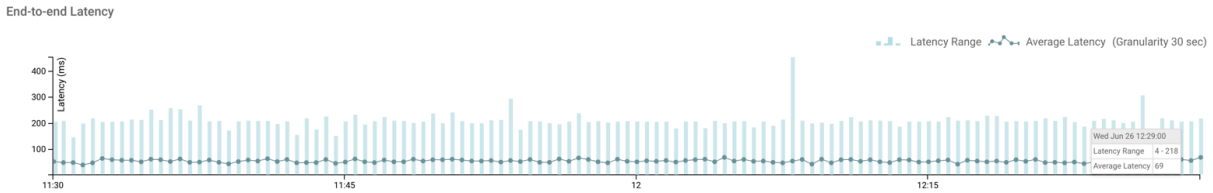
  In the image, there are two red areas. The first red area, from the left, indicates that the number of consumed messages is more than the number of produced messages. This represents an overconsumption of messages which can occur when a consumer group offset is reset to an older offset to reprocess messages, or when producers or consumers are shut down in an unclean manner.

  The last red area indicates that the number of consumed messages is less than the number of produced messages. This represents an underconsumption of messages which can occur when the consumer group offset is set to a newer offset causing the consumer group to skip processing some messages.

  The far right section of the graph shows the current processing window where consumers are still consuming the produced messages. Therefore this area is expected to be marked red and indicates an undercsonumption of messages.

  All other areas in the image are blue, which indicates that all the produced messages are consumed.

- End-to-end Latency. The End-to-end Latency graph provides you the details of the latency range and average latency in consuming the messages, which are produced in a particular topic, in milliseconds within the selected time range.



  In the above image, the vertical lines represent the latency range, and the dotted line represents the average latency in consuming the produced messages in last one hour with a granularity of 30 seconds. You can see that at

12:29:00 hours on Wed Jun 26, the latency range was between 4 - 218 milliseconds and the average latency was 69 milliseconds.

Note: You can also create alerts to receive notifications based on the conditions that you configure in the alert policy for monitoring latency in your system. For more information about creating alerts to monitor latency, see the *Managing Alert Policies* guide.

Related Information

Alert Policies Overview

Granularity of metrics for end-to-end latency

Learn how often Streams Messaging Manager (SMM) queries and retrieves latency metrics data from the REST API server.

SMM uses the REST API to display metrics. All the metrics are available for querying at two different granularities: 30 seconds and 15 minutes. The metrics are pre-aggregated for the defined buckets. Depending on how long the data is queried, granularity and varying dimensions of the topic, partition, consumer group ID and client ID, the data is calculated and rendered as JSON. Go through the following details before you start monitoring latency using SMM:

- When you select a period newer than current time by 24 hours, the data is retrieved from REST server with metrics granularity of 30 seconds.
- When you select a period older than current time by 24 hours, the data is retrieved from REST server with metrics granularity of 15 minutes.
- SMM UI polls the API for updates periodically (every 30 seconds if the selected period is newer than current time by 24 hours and every 15 minutes otherwise).
- By default, the 30 seconds granularity metrics are stored for 24 hours and the 15 minutes granularity metrics are stored for 2 weeks.

For information on REST APIs used in SMM, see the *REST API Reference* guide.

Enabling interceptors

You need to enable interceptors for consumers, producers, and KafkaStreams applications to enable Streams Messaging Manager (SMM) to fetch the metrics. If you do not enable the interceptors, you can not see any metrics in SMM.

Interceptors publish the metrics to Kafka periodically. Metrics include counts on the producer side, and count average latency, and minimum and maximum latencies on the consumer side.

Steps to enable interceptors in your application

Add the following jar to the classpath of the application or as a dependency in the application:

```
<dependency>
 <groupId>com.hortonworks.smm</groupId>
 <artifactId>monitoring-interceptors</artifactId>
</dependency>
```

Steps to enable consumer interceptor

Perform the following steps to enable consumer interceptor:

1. Add the interceptor.classes property to consumer configuration that gets passed to the KafkaConsumer constructor.

2. Configure the client.id property as follows:

```
KafkaConsumer<Integer, String> createKafkaConsumer(String bootstrapServe
rs, String groupId, String clientIdentifier) {
    Properties properties = new Properties();
    properties.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapSe
rvers);
    properties.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, "org.ap
ache.kafka.common.serialization.IntegerDeserializer");
    properties.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, "org.a
pache.kafka.common.serialization.StringDeserializer");
    properties.put(ConsumerConfig.GROUP_ID_CONFIG, groupId);
    properties.put(ConsumerConfig.CLIENT_ID_CONFIG, clientIdentifier);
    //Add ConsumerInterceptor like this
    properties.put(ConsumerConfig.INTERCEPTOR_CLASSES_CONFIG,
       "com.hortonworks.smm.kafka.monitoring.interceptors.MonitoringCon
sumerInterceptor");
    return new KafkaConsumer<Integer, String>(properties);
}
```

Note: We recommend you to configure the client.id property. It helps in identifying the consumer instance. If you do not configure it, the latency metrics fetch the default consumer ID.

Steps to enable producer interceptor

Add the interceptor.classes property to producer configuration that gets passed to the KafkaProducer constructor, as follows:

```
KafkaProducer<Integer, String> createKafkaProducer(String bootstrapServers)
{
    Properties properties = new Properties();
    properties.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServers
);
    properties.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, "org.apach
e.kafka.common.serialization.IntegerSerializer");
    properties.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, "org.apac
he.kafka.common.serialization.StringSerializer");
    //Add ProducerInterceptor like this
    properties.put(ProducerConfig.INTERCEPTOR_CLASSES_CONFIG,
       "com.hortonworks.smm.kafka.monitoring.interceptors.MonitoringProduce
rInterceptor");
    return new KafkaProducer<Integer, String>(properties);
}
```

Steps to enable interceptors in KafkaStreams applications

Add the producer.interceptor.classes and consumer.interceptor.classes properties to Kafka Streams configurations, as follows:

```
void startKafkaStreams(StreamsBuilder builder) {
    KafkaStreams kstreams = new KafkaStreams(builder.build(), getKafkaStrea
msConfiguration());
    kstreams.start();
}
Properties getKafkaStreamsConfiguration() {
    Properties config = new Properties();
    config.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServers);
    config.put(StreamsConfig.APPLICATION_ID_CONFIG, appId);
    config.put(StreamsConfig.CLIENT_ID_CONFIG, clientId);

    //Add producer interceptor like this
```

```
    config.put(
      StreamsConfig.PRODUCER_PREFIX + ProducerConfig.INTERCEPTOR_CLASSES_CO
NFIG,
      "com.hortonworks.smm.kafka.monitoring.interceptors.MonitoringProduce
rInterceptor");
    //Add consumer interceptor like this
    config.put(
      StreamsConfig.CONSUMER_PREFIX + ConsumerConfig.INTERCEPTOR_CLASSES_C
ONFIG,
      "com.hortonworks.smm.kafka.monitoring.interceptors.MonitoringConsum
erInterceptor");

    return config;
}
```

Note: MonitoringProducerInterceptor publishes the producer metrics to the "__smm_producer_metrics" topic and MonitoringConsumerInterceptor publishes the consumer metrics to the "__smm_consumer_metrics" topic from their respective client applications. So if Authorization is enabled on kafka cluster, ensure that your client applications have access to the aforementioned topics.

Monitoring end to end latency for Kafka topic

To monitor end-to-end latency of Kafka topics, you can monitor the count of consumed messages across all Kafka consumer groups and the time taken by all Kafka consumer groups to consume messages that are produced in a Kafka topic, in a graphical way. You can also monitor the same for each Kafka consumer group, each client in a Kafka consumer group, and each partition in a Kafka topic.

About this task

Perform the following steps to monitor end-to-end latency in the Streams Messaging Manager (SMM) UI:

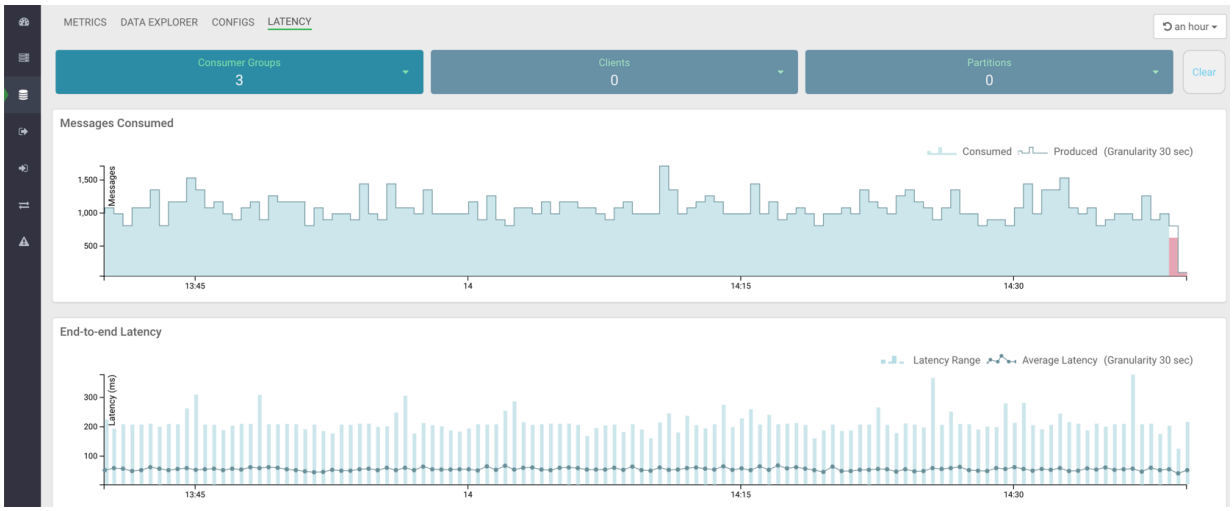
Procedure

1. Go to Topics in the SMM UI.
2. Select the topic you want to verify the details about.
3. Click the Profile icon beside the topic you select.

   This takes you to the Metrics page where you can find the Messages Consumed and End-to-end Latency graphs along with other topic details. On the Metrics page, these two graphs provide you an aggregated result of latency and count of consumed messages across all consumer groups.

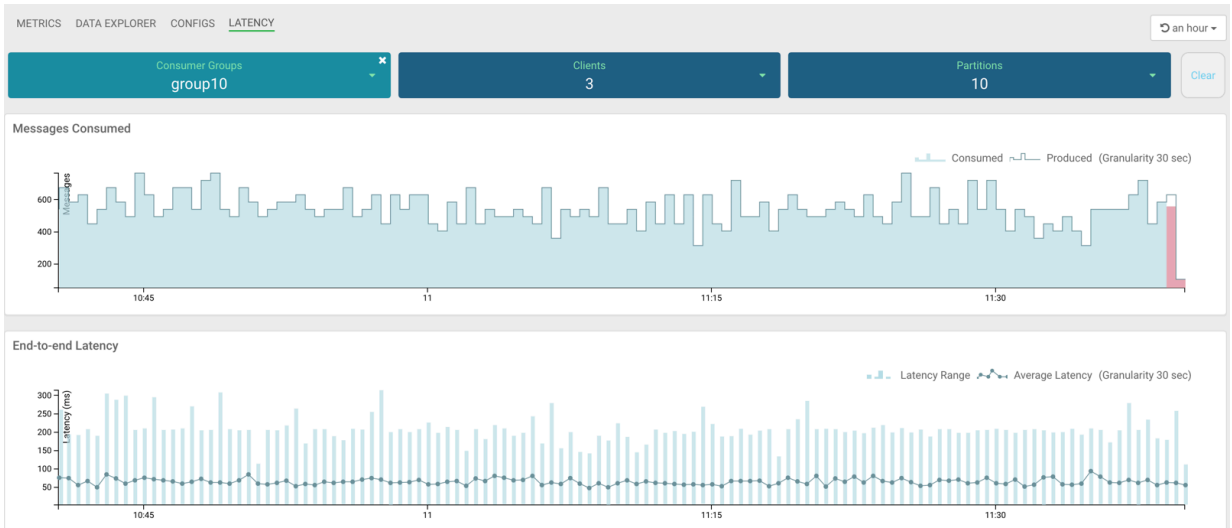
4. To verify the details by individual consumer groups, clients, and partitions, go to the Latency tab.

   The Latency page appears as shown in the following image:



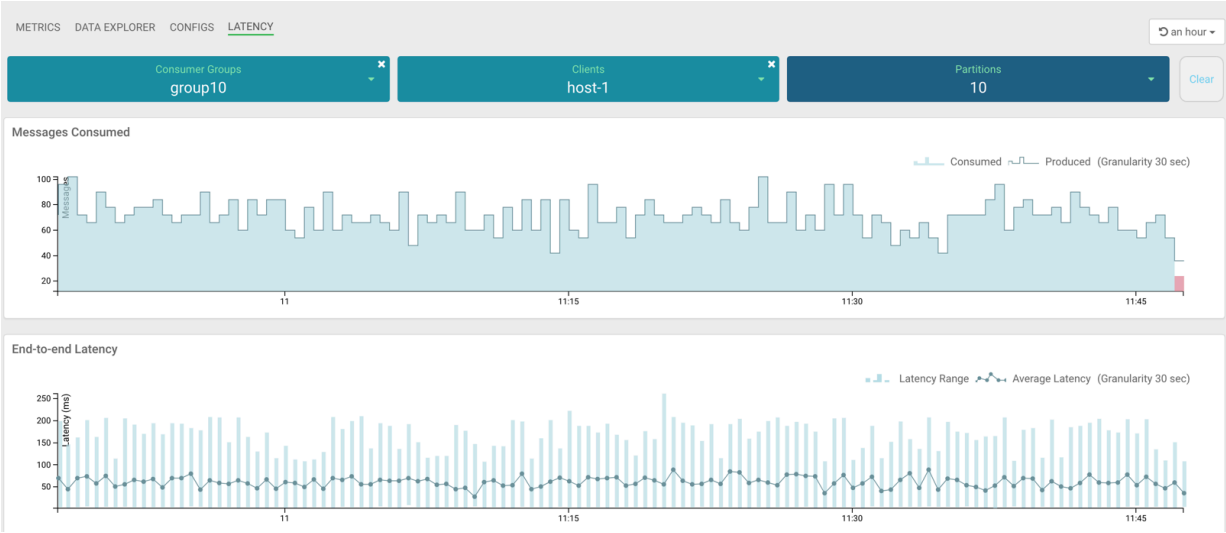
   The Latency view gives you a powerful snapshot of the end-to-end latency scenario: number of consumer groups for a topic, number of clients inside a particular consumer group, and number of partitions in a topic along with the Messages Consumed and End-to-end Latency graphs.

5. Select any consumer group from the Consumer Groups drop-down, as shown in the following image:



   In the image, group10 consumer group is selected. The Latency tab displays that there are 3 clients in the group10 consumer group, and 10 partitions are available in the topic.

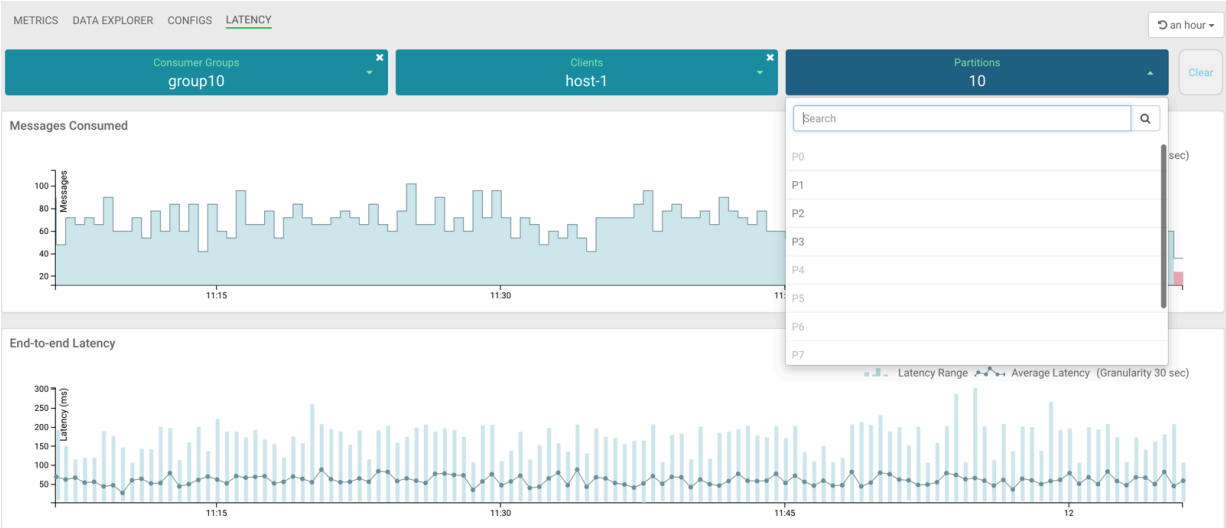
6. Select any client from the Clients drop-down, as shown in the following image:



In the image, host-1 client is selected. At this instance, the Messages Consumed and End-to-end Latency graphs display data for only the host-1 client. Here you can monitor the number of messages produced, number of messages consumed, latency range, and average latency for host-1 only. Hover your mouse over the graphs and get data at any point of time in the selected time range. You can see in the Messages Consumed graph that host-1 consumed all the messages that are produced and actively consuming data at the latest time. You can see in the End-to-end Latency graph that the latency range and average latency are under 250 milliseconds.

7. To get details about the partitions from where host-1 is consuming data, click Partitions.

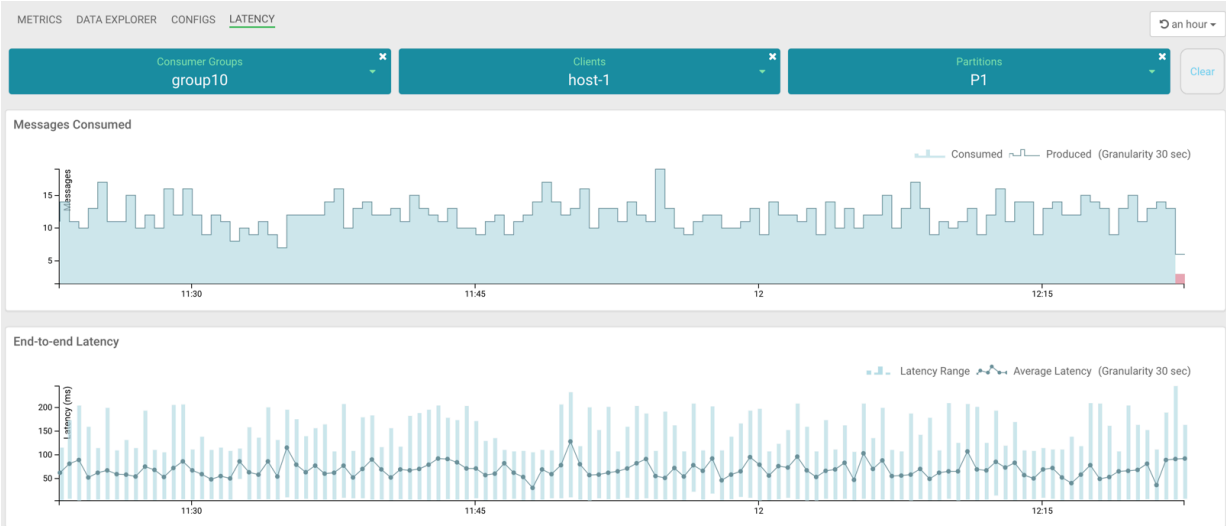
   The list of partitions in the topic appears, as shown in the following image:



In the image, you can see that host-1 is consuming data from 3 partitions: P1, P2, and P3. Other partitions are inactive for host-1.

8. Select any active partition from the list.

   The Latency tab displays the transaction details between host-1 and the selected partition (for example, P1), as shown in the following image:

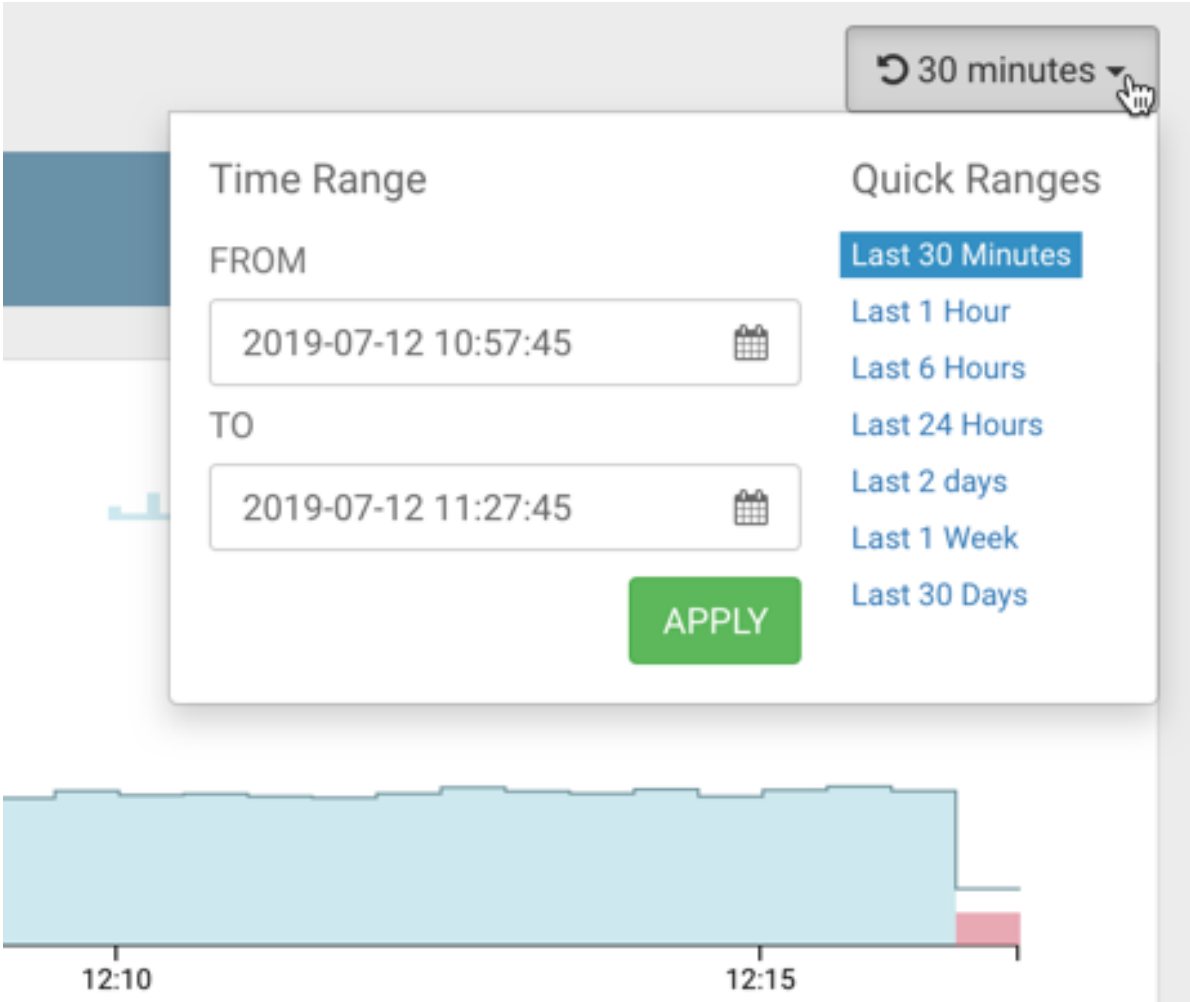


   Now you have got details for host-1 client. Likewise, you can fetch details for other clients.

9. Follow steps 6 through 8 to fetch data for all other clients.

10. Follow steps 5 through 8 to fetch data for all other consumer groups.

- To clear all the selections at once, click the Clear button at the top-right corner of the page.
- To clear selection for Consumer Groups, Clients, or Partitions, click the delete icon located on each drop-down.
- To select a different time range, use the Time Range and Quick Ranges options at the top-right corner of the page, as shown in the following image:



End to end latency use case

You can use end-to-end latency feature available in Streams Messaging Manager (SMM) to handle real-time scenarios including measurement of end-to-end processing time, identification of slow or lagging consumers, and verification of over-consumption or under-consumption of messages.

Verify whether end-to-end processing time SLAs are met

A service-level agreement (SLA) is a commitment between a service provider and a service user. Particular aspects of the service are agreed between the service provider and the service user. The most common component of SLA is that the services should be provided to the user as agreed upon in the contract. For example, you agreed upon an average latency value and a maximum latency value for message consumption with Cloudera. Therefore, after a producer produces messages, if the messages take the agreed amount of time to be consumed by consumers, the SLA would be met.

1. Go to Topics in the SMM UI.

2. Select the topic you want to verify the details about.
3. Click the Profile icon beside the topic you select.
4. Check the latency graph and see if the average and maximum latencies are as expected.
5. If the latencies are not as expected, go to the Latency tab.
6. Check whether the number of clients is as expected. If not, then you might want to check the missing client instance.
7. If the number of clients are as expected, then check if there is any spike in the message count. Select a period of 1 week in the Time Range pane and see if there is a surge in incoming messages that could explain the time SLA violation.
8. If the time SLA appears to be breached even after all the checks turned positive, go to Use Case 2.

Identify slow or lagging consumers

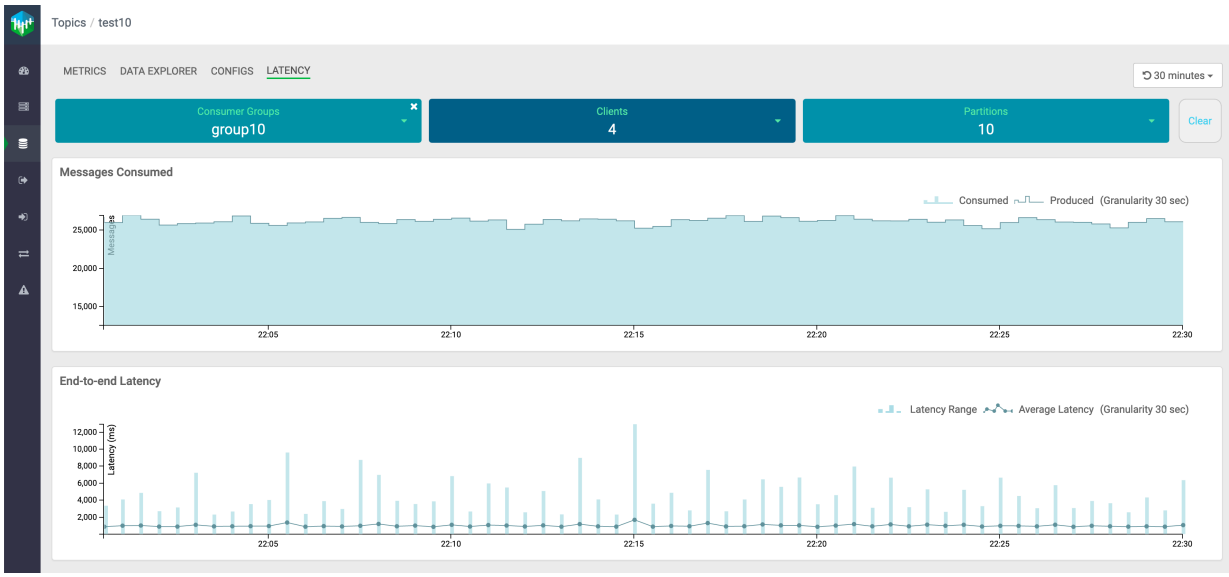
A stream based application flow involves application polling for messages, fetching and processing the messages, performing an optional blocking operation like interacting with database or local file system and then application polling for the messages again. However, the delay due to message processing, system bottleneck, or external bottleneck might become very long in some scenarios and you might want to understand which of the process instances is facing issues.

1. Go to Topics in the SMM UI.
2. Select the topic you want to verify the details about.
3. Click the Profile icon beside the topic you select.
4. Go to the Latency tab.

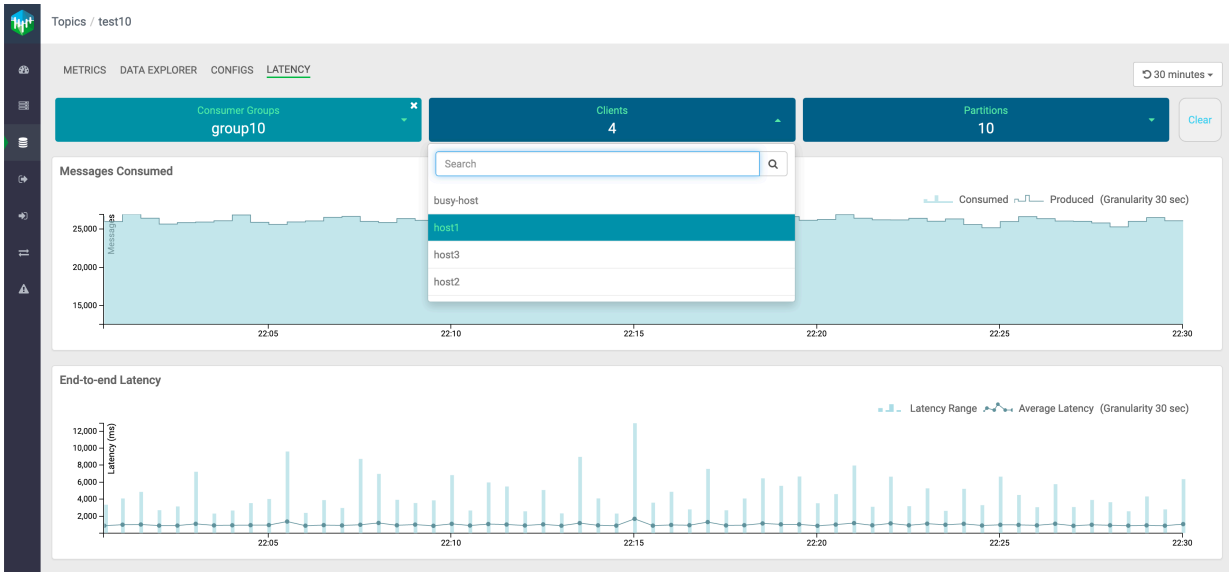
5. After you select a group, inspect the latency and message counts for each client.

   This could lead you to the slow consumer.

   Let us walk through an example.

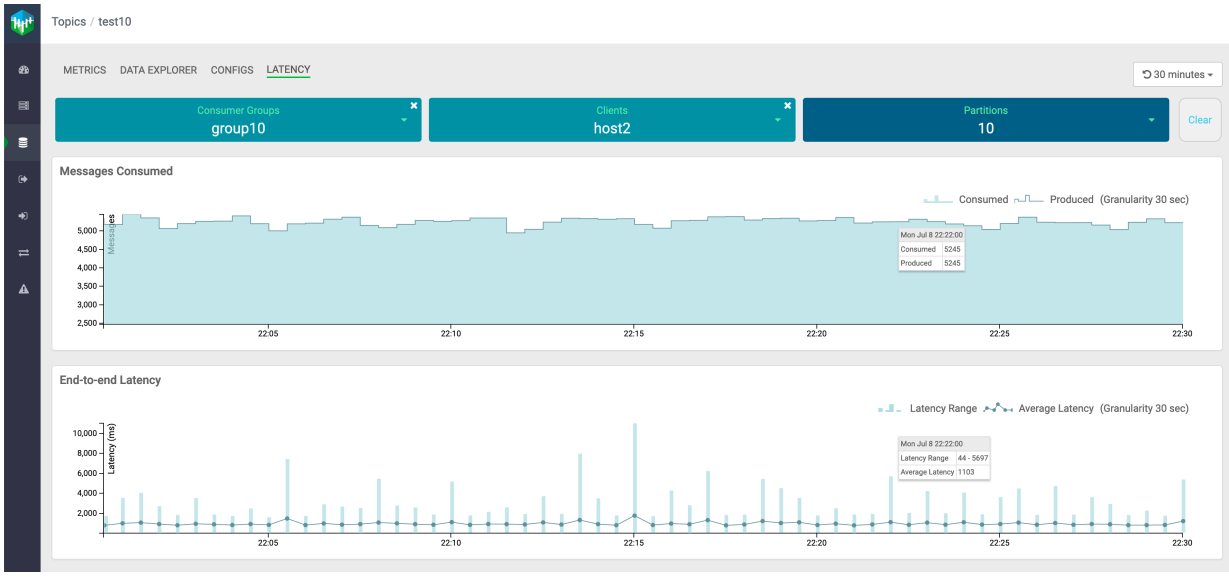


In the above image, you select group10 consumer group to inspect the latency and message counts for each client.

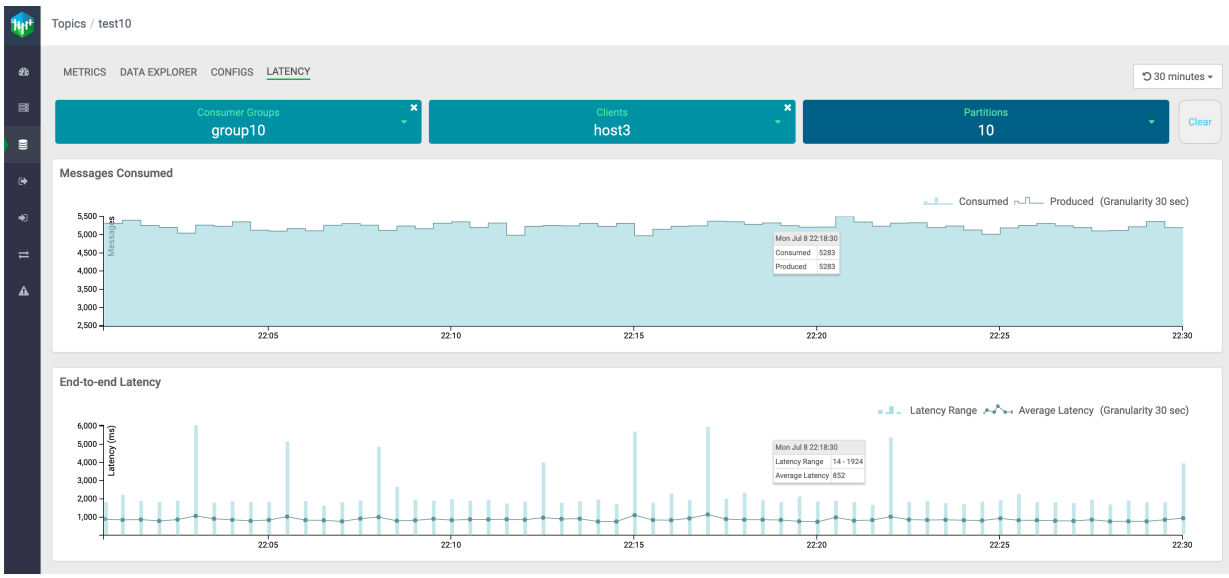


In the above image, you see the list of active clients for group10: host1, host2, host3, and busy-host. Now you need to select each client and inspect the latency and message counts.

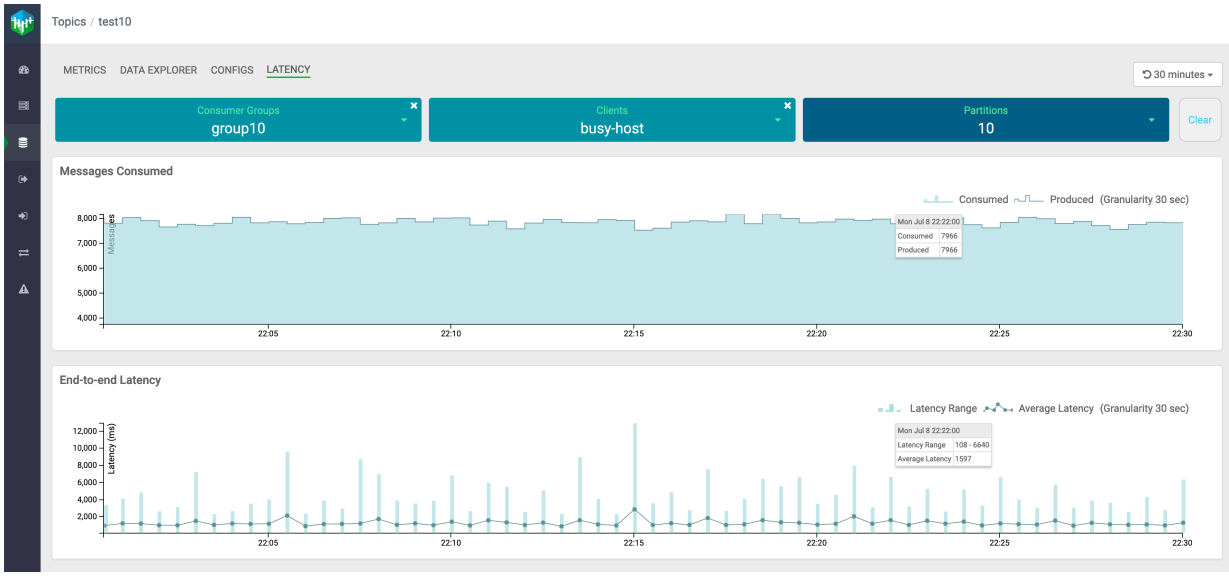
In the above image, you can see that host1 consumed all the messages produced, and the average latency and latency range are under a good range.



In the above image, you can see that host2 consumed all the messages produced. Also, there are few spikes in latency range, but the average latency is under a good range.



In the above image, you can see that host3 consumed all the messages produced. Also, there are occasional spikes in latency range, but the average latency is under a good range.



In the above image, you can see that busy-host consumed all the messages produced. There are frequent spikes in latency range, and both latency range and average latency are high. Hence, busy-host is the slow consuming client.

It is possible that all the clients are experiencing longer latencies. This could imply that there is a common bottleneck. For example, clients are interacting with external store over network and having delays in consuming the messages due to network issues.

If there is a single client that is experiencing slowness, you must check the message counts for other clients, and system parameters like CPU and memory.

This addresses your need to identify the slow consuming application.

Verify whether messages are overconsumed or underconsumed

There could be an overconsumption of messages. This might occur due to the following reasons:

- If the producers and consumers are shut down in an unclean manner or the producers and consumers went down in an unexpected manner. For example, Kafka producer produced some messages but it shut down before the

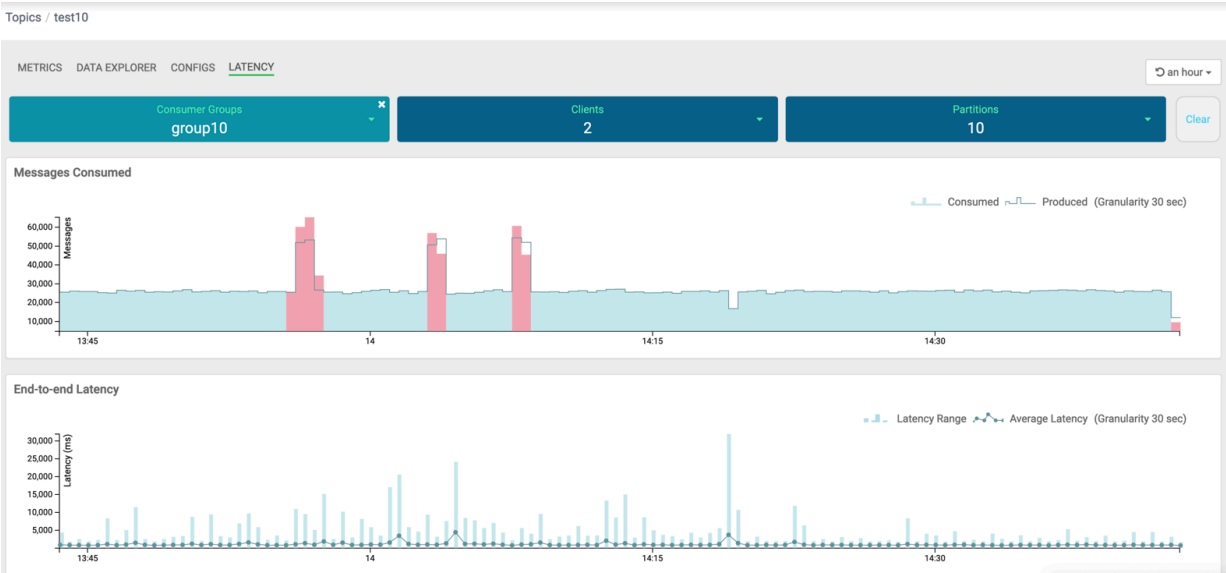
producer received any acknowledgement from the broker. Similarly, Kafka consumer consumed a few messages but got shut down before it could commit the offset at this latest point.

- If the consumer is reset to an older offset (reprocessing scenarios).

If the consumer is reset to a new offset (real-time application requirement), there could be an underconsumption of messages.There could be over or under consumption of messages if the cluster is in an unhealthy state.

1. Go to Topics in the SMM UI.
2. Select the topic you want to verify the details about.
3. Click the Profile icon beside the topic you select.
4. Go to the Latency tab.
5. After you select a group, inspect the produced message and consumed message counts for each client in Messages Consumed graph.

   This helps you to verify whether the consumers are consuming all the messages that are produced in a topic. You might also identify occurrences of any overconsumption or underconsumption of messages.



   In the image, you can see that for group10 consumer group, there are three red spikes in the Messages Consumed graph.

   The first spike, from the left, indicates that the number of consumed messages is more than the number of produced messages. So, this is an overconsumption of messages.

   The second and the third spike indicate overconsumption of messages followed by an underconsumption of messages.