

Ingesting Data into Apache Kafka

Date published: 2019-12-16

Date modified: 2021-10-25



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

Ingesting data into Apache Kafka.....	4
Understand the use case.....	4
Meet the prerequisites.....	4
Build the data flow.....	5
Create controller services for your data flow.....	6
Configure the processor for your data source.....	8
Configure the processor for your data target.....	9
Start the data flow.....	11
Verify data flow operation.....	11
Monitoring end to end latency for Kafka topic.....	14
 Monitoring your data flow.....	 18
 Next steps.....	 19
 Appendix – Schema example.....	 19

Ingesting data into Apache Kafka

You can use an Apache NiFi data flow to ingest data into Apache Kafka by following these steps.

Understand the use case

Learn how to use NiFi to move data from a range of locations into Kafka in CDP Public Cloud.

Apache NiFi as a part of Cloudera Data Flow offers a scalable way of managing data flows with guaranteed delivery, data buffering / pressure release, and prioritized queuing.

You can use it to ingest data into Apache Kafka by simply dragging and dropping a series of processors on the NiFi user interface. In addition to facilitating the data flow into Kafka, the processors provide an out-of-the-box integration with Streams Messaging Manager (SMM) by configuring the metrics interceptors.

When the data flow is ready, you can visually monitor and control the pipeline you have built.

This use case walks you through the steps of creating a data flow that generates FlowFiles with random CSV data and writes this data to Apache Kafka in CDP Public Cloud. This gets you started with creating a Kafka ingest data flow. If you want to use a data source other than generated FlowFiles, see the *Getting Started with Apache NiFi* for information about how to build a data flow, and about get and consume data processor options.

Related Information

[Getting Started with Apache NiFi](#)

[Ingesting Data into Apache HBase](#)

[Ingesting data into Apache Hive](#)

[Ingesting Data into Apache Kudu](#)

[Ingesting Data into Amazon S3 Buckets](#)

[Ingesting Data into Azure Data Lake Storage](#)

Meet the prerequisites

Use this checklist to make sure that you meet all the requirements before you start building your data flow.

- You have a CDP Public Cloud environment.
- You have a CDP username (it can be your own CDP user or a CDP machine user) and a password set to access Data Hub clusters.

The predefined resource role of this user is at least EnvironmentUser. This resource role provides the ability to view Data Hub clusters and set the FreeIPA password for the environment.

- Your user is synchronized to the CDP Public Cloud environment.
- Your CDP user has the correct permissions set up in Ranger allowing access to NiFi and Kafka.
- You have a Flow Management Data Hub cluster in your CDP environment.
- You have a Streams Messaging Data Hub cluster in the same CDP environment as the Flow Management cluster.
- You have created a Kafka topic in your Streams Messaging cluster to send data to.
- You have created a sample schema in your Streams Messaging cluster you want to use for writing data to Kafka.

See the *Appendix* for an example schema.

Related Information

[Understanding roles and resource roles](#)

[Creating your first Flow Management cluster](#)

[Creating your first Streams Messaging cluster](#)

[Creating a Kafka topic](#)

[Creating a new schema in Schema Registry](#)

[Authorizing Flow Management cluster access in CDP Public Cloud](#)

[Appendix – Schema example](#)

Build the data flow

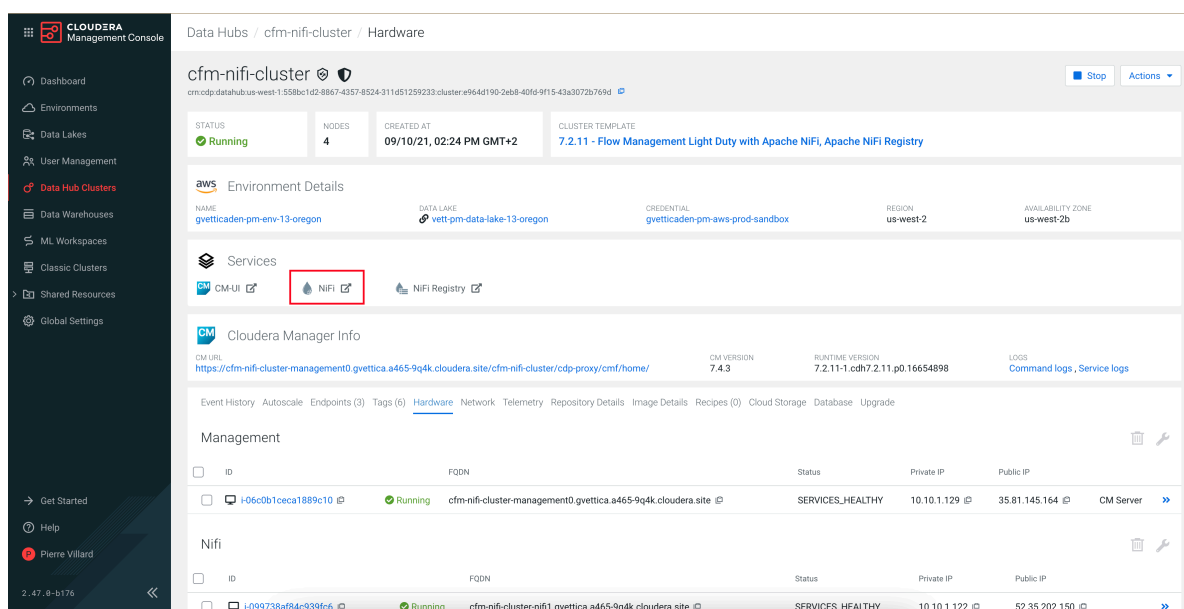
Learn how to create NiFi data flows easily with a number of processors and flow objectives to choose from.

About this task

You can use the `PublishKafka2RecordCDP` processor to build your Kafka ingest data flow. Regardless of the type of flow you are building, the basic steps in building your data flow are to open NiFi, add your processors to the canvas, and connect the processors to create the flow.

Procedure

1. Open NiFi
in
Data Hub.
 - a) To access the NiFi service in your Flow Management Data Hub cluster, navigate to Management Console serviceData Hub Clusters.
 - b) Click the tile representing the Flow Management Data Hub cluster you want to work with.
 - c) Click the NiFi icon in the Services section of the cluster overview page to access the NiFi UI.



You will be logged into NiFi automatically with your CDP credentials.

2. Add the `GenerateFlowFile` processor for data input.

This processor creates FlowFiles with random data or custom content.

- a) Drag and drop the processor icon into the canvas. This displays a dialog that allows you to choose the processor you want to add.
- b) Select the `GenerateFlowFile` processor from the list.
- c) Click Add or double-click the required processor type to add it to the canvas.

You will configure the `GenerateFlowFile` processor to define how to create the sample data in *Configure the processor for your data source*.

3. Add the PublishKafka2RecordCDP processor for data output.



Note: There are various processors in NiFi for ingesting data into Kafka.

PublishKafka2RecordCDP uses a configured record reader to read the incoming flow files as records, and then it uses a configured record writer to serialize each record for publishing to Kafka. As Streams Messaging clusters in Data Hub include Kafka version 2.x, use PublishKafka2RecordCDP in your data flow.

You will configure the PublishKafka2RecordCDP processor in *Configure the processor for your data target*.

4. Connect the two processors to create a flow.

- Drag the connection icon from the first processor, and drop it on the second processor.

A Create Connection dialog appears with two tabs: Details and Settings.

- Configure the connection.

You can configure the connection's name, flowfile expiration time period, thresholds for back pressure, load balance strategy and prioritization.

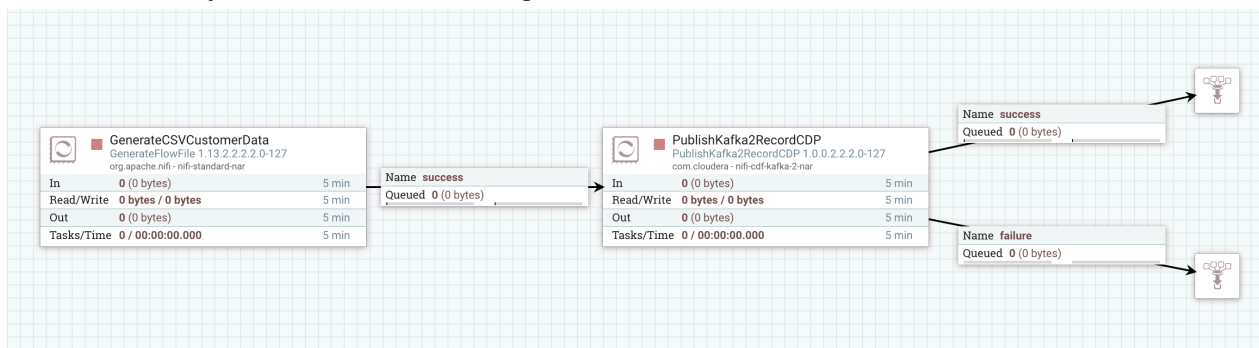
- Click Add to close the dialog box and add the connection to your data flow.

5. Optionally, you can add success and failure funnels to your data flow, which help you see where flow files are being routed when your flow is running. Connect the PublishKafka2RecordCDP to these funnels with success and failure connections.

If you want to know more about working with funnels, see the *Apache NiFi User Guide*.

Results

Your data flow may look similar to the following:



What to do next

Create controller services for your data flow. You will need these services later on, when configuring the PublishKafka2RecordCDP processor.

Related Information

[Building a NiFi data flow](#)

[Apache NiFi User Guide](#)

Create controller services for your data flow

Learn how to create and configure controller services for the Kafka ingest data flow.

About this task

You can add controller services that can provide shared services to be used by the processors in your data flow.

Procedure

1. To add a Controller Service to your flow, right-click on the canvas and select Configure from the pop-up menu. This displays the Controller Services Configuration window.
2. Select the Controller Services tab.
3. Click the + button to display the Add Controller Service dialog.
4. Select the required Controller Service and click Add.
5. Click the Configure icon in the right-hand column and configure the necessary options.
6. When you have finished configuring the options you need, click Apply to save the changes.
7. Click the Enable button (flash) in the far-right column of the Controller Services tab to enable the Controller Service.

Example

The following controller services are used in this Kafka ingest example:

- HortonworksSchemaRegistry Controller Service
- CSVReader Controller Service
- AvroRecordSetWriter Controller Service

See below for property details.

HortonworksSchemaRegistry Controller Service

In this data flow, the data source is Kafka. To define the Schema Registry URL property of the HortonworksSchemaRegistry Controller Service, provide the URL of the schema registry that you want to connect to. You can find the master hostname on the Streams Messaging cluster overview page when selecting the Hardware tab.

Event History Hardware Cloud Storage Tags (8) Gateway Recipes (0) Repository Details Image Details Network							
Master							
Search							
☐	ID	FQDN	Status	Private IP	Public IP		
☐	i0c1485c11de919ec9	Running docs-messaging-master0.cdf-docs.a465-9q4k.cloudera.site	SERVICES_HEALTHY	10.10.169.72	54.188.233.219	CM Server	

Table 1: HortonworksSchemaRegistry Controller Service properties

Property	Description	Example value for ingest data flow
Schema Registry URL	Provide the URL of the schema registry that this Controller Service should connect to, including version. In the format: https://host:7790/api/v1	https://docs-messaging-master0.cdf-docs.a465-9q4k.cloudera.site:7790/api/v1
SSL Context Service	Specify the SSL Context Service to use for communicating with Schema Registry. Use the pre-configured SSLContextProvider.	Default NiFi SSL Context Service
Kerberos Principal	Specify the user name that should be used for authenticating with Kerberos. Use your CDP workload username to set this Authentication property.	srv_nifi-kafka-ingest
Kerberos Password	Provide the password that should be used for authenticating with Kerberos. Use your CDP workload password to set this Authentication property.	password

CSVReader Controller Service

Table 2: CSVReader Controller Service Properties

Property	Description	Example value for ingest data flow
Schema Access Strategy	Specify how to obtain the schema to be used for interpreting the data.	Use String Fields From Header
Treat First Line as Header	Specify whether or not the first line of CSV should be considered a Header or a record.	true

AvroRecordSetWriter Controller Service

Table 3: AvroRecordSetWriter Controller Service Properties

Property	Description	Example value for ingest data flow
Schema Write Strategy	Specify how the schema for a Record should be added to the data.	HWX Content-Encoded Schema Reference
Schema Access Strategy	Specify how to obtain the schema to be used for interpreting the data.	Use 'Schema Name' Property
Schema Registry	Specify the Controller Service to use for the Schema Registry.	CDPSchemaRegistry
Schema Name	Specify the name of the schema to look up in the Schema Registry property. See the <i>Appendix</i> for an example schema.	customer

What to do next

Configure the processor for your data source.

Related Information

[Adding Controller Services for data flows](#)

[Apache NiFi Documentation](#)

[Appendix – Schema example](#)

Configure the processor for your data source

Learn how to configure a data source processor for the Kafka ingest data flow.

About this task

You can set up a data flow to move data into Apache Kafka from many different locations. This example assumes that you are using sample data generated by the `GenerateFlowFile` processor. If you are moving data from a certain location, see the *Apache NiFi Getting Started* for information on how to build a data flow, and about other data ingest processor options.

Procedure

1. Launch the Configure Processor window, by right clicking the `GenerateFlowFile` processor and selecting Configure. A configuration dialog with the following tabs is displayed: Settings, Scheduling, Properties, and Comments.
2. Configure the processor according to the behavior you expect in your data flow.

The `GenerateFlowFile` processor can create many FlowFiles very quickly. Setting the run schedule to a reasonable value is important so that the flow does not overwhelm the system.

- When you have finished configuring the options you need, save the changes by clicking the Apply button.

Make sure that you set all required properties, because you cannot start the processor until all mandatory properties have been configured.

Example

The following settings and properties are used in this example:

Table 4: GenerateFlowFile processor scheduling

Scheduling	Description	Example value for ingest data flow
Run Schedule	Run schedule dictates how often the processor should be scheduled to run. The valid values for this field depend on the selected Scheduling Strategy.	500 ms

Table 5: GenerateFlowFile processor properties

	Description	Example value for ingest data flow
Custom text	If Data Format is text and if Unique FlowFiles is false, you can provide custom to be used as the content of the generated FlowFiles. The expression statement in the example value generates a random ID between 1 and 10 000, with random last names assigned.	<pre>customer_id, customer_name \${random():mod(10000):plus(1)}, Smith \${random():mod(10000):plus(1)}, Johnson \${random():mod(10000):plus(1)}, Steward</pre>

What to do next

Configure the processor for your data target.

Related Information

[Configuring a processor](#)

[Getting started with Apache NiFi](#)

[Apache NiFi Documentation](#)

Configure the processor for your data target

Learn how to configure a data target processor for the Kafka ingest data flow.

About this task

You can set up a data flow to move data to many locations. This example assumes that you are moving data to Apache Kafka. If you want to move data to a different location, see the other use cases in the *Cloudera Data Flow for Data Hub* library.

Procedure

- Launch the Configure Processor window, by right-clicking the PublishKafka2RecordCDP processor and selecting Configure.

A configuration dialog with the following tabs is displayed: Settings, Scheduling, Properties, and Comments.

- Configure the processor according to the behavior you expect in your data flow.

- When you have finished configuring the options you need, click Apply to save the changes.

Make sure that you set all required properties, because you cannot start the processor until all mandatory properties are configured.

Example

In this example data flow, we are writing data to Kafka. You can create the modified Kafka broker URLs using the broker hostnames and adding port :9093 to the end of each FQDN. You can find the hostnames on the Streams Messaging cluster overview page when selecting the Hardware tab.

Broker					
Search					
☐	ID		FQDN	Status	Private IP Public IP
☐	i-0af4489a5872d4baf		docs-messaging-broker3.cdf-docs.a465-9q4k.cloudera.site	SERVICES_HEALTHY	10.10.181.43 34.220.36.215
☐	i-071852c49d04de3da		docs-messaging-broker1.cdf-docs.a465-9q4k.cloudera.site	SERVICES_HEALTHY	10.10.171.157 34.222.96.228
☐	i-0196d945c955cb00f		docs-messaging-broker2.cdf-docs.a465-9q4k.cloudera.site	SERVICES_HEALTHY	10.10.191.203 34.213.222.132



Note: Property values can be parameterized. For example, you can create a parameter context to hold Kafka connection information and apply it to the Kafka Brokers property instead of adding the broker URLs individually.

The following properties are used for the PublishKafka2RecordCDP processor:

Table 6: PublishKafka2RecordCDP processor properties

Property	Description	Example value for ingest data flow
Kafka Brokers	Provide a comma-separated list of known Kafka Brokers. In the format <host>:<port>.	Docs-messaging-broker1.cdf-docs.a465-9q4k.cloudera.site:9093,docs-messaging-broker2.cdf-docs.a465-9q4k.cloudera.site:9093,docs-messaging-broker3.cdf-docs.a465-9q4k.cloudera.site:9093
Topic Name	Provide the name of the Kafka Topic to publish to.	Customer
Record Reader	Specify the Record Reader to use for incoming FlowFiles.	ReadCustomerCSV
Record Writer	Specify the Record Writer to use in order to serialize the data before sending to Kafka.	WriteCustomerAvro
Use Transactions	Specify whether or not NiFi should provide Transactional guarantees when communicating with Kafka.	false
Delivery Guarantee	Specify the requirement for guaranteeing that a message is sent to Kafka.	Guarantee Single Node Delivery
Security Protocol	Specify the protocol used to communicate with Kafka brokers.	SASL_SSL
SASL Mechanism	Specify the SASL mechanism to use for authentication.	PLAIN
Username	Use your CDP workload username to set this Authentication property.	srv_nifi-kafka-ingest

Property	Description	Example value for ingest data flow
Password	Use your CDP workload password to set this Authentication property. This will use the specified CDP user to authenticate against Kafka. Make sure that this user has the correct permissions to write to the specified topic.	password
SSL Context Service	Specify the SSL Context Service to use for communicating with Kafka. Use the pre-configured SSLContextProvider.	Default NiFi SSL Context Service

What to do next

Your data flow is ready to ingest data into Kafka. You can now start the data flow.

Related Information

[Configuring a processor](#)

[Apache NiFi User Guide](#)

[Data ingest use cases in Cloudera Data Flow for Data Hub](#)

Start the data flow

Learn how to start your Kafka ingest data flow.

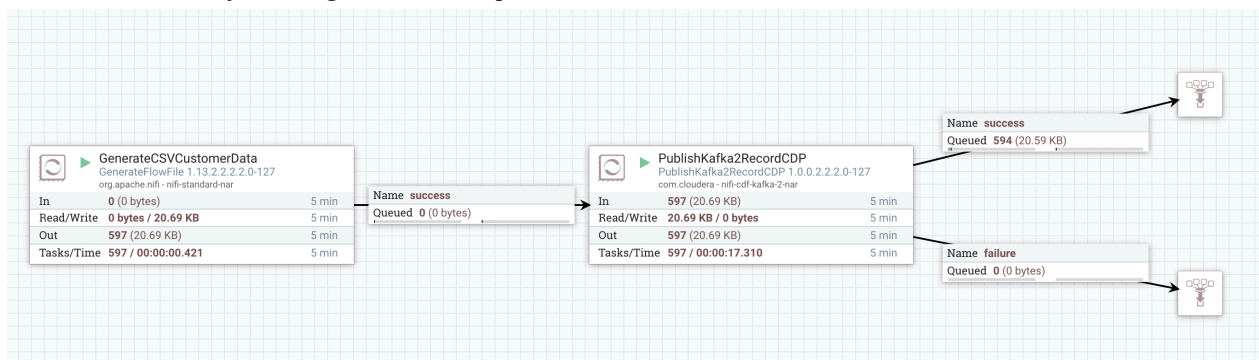
Procedure

1. Select all the data flow components you want to start.
2. Click the Start icon in the Actions toolbar.

Alternatively, right-click a single component and choose Start from the context menu.

Results

Your flow should be running without any errors. Data should be generated in the form of FlowFiles, and the files should be written to your designated Kafka topic.



What to do next

It is useful to check that data is running through your flow without any errors.

Related Information

[Apache NiFi User Guide](#)

Verify data flow operation

Learn how you can verify the operation of your Kafka ingest data flow.

About this task

There are a number of ways to check that data is running through the flow you have built and it actually appears in Kafka.

Procedure

- You can verify that NiFi processors are not producing errors.
- You can look at the processors in the UI to see the amount of data that has gone through them. You can also right-click on the processors, or on connections to view status history.

- You can verify in Streams Messaging Manager that messages appear in the defined Kafka topic.
To check this:
 - Navigate to your Streams Messaging Data Hub cluster.
 - Click Streams Messaging Manager in the Services section of the Cluster overview page.

The screenshot shows the Cloudera Management Console interface for a cluster named 'docs-messaging'. The left sidebar contains navigation links for Dashboard, Environments, Data Lakes, User Management, Data Hub Clusters, Data Warehouses, ML Workspaces, and Classic Clusters. The main content area displays cluster details including Status (Running), Nodes (4), Created At (03/23/20, 04:21 AM GMT+1), and Cluster Template (7.1.0 - Streams Messaging Light Duty: Apache Kafka, Schema Registry, Streams Messaging Manager). Below this, the 'Services' section lists CM-UI, Schema Registry, and Streams Messaging Manager (highlighted with a red box). The 'Cluster Manager Info' section provides the CM URL, version (7.1.0), and platform version (7.1.0-1.cdh7.1.0.p0.1922354). The 'Master' section shows a table of nodes with columns for ID, FQDN, Status, Private IP, and Public IP.

- Search for your previously created topic and verify that your NiFi flow is visible, and it shows the producer and consumer as well.

The screenshot shows the Cloudera Streams Messaging Manager interface. The 'Overview' tab is active, displaying a table of topics. The 'customer' topic is selected, showing details such as Replication Factor (3), InSync Replicas (9 Of 9), Total messages (5,691), and Retention Period (168 hrs). The 'Producers' section shows one active producer (producer-5) with 5.7k messages. The 'Consumer Groups' section shows one active consumer group (nifi-s3-ingest) with 13 messages. The interface includes filters for Producers (1), Brokers (3), Topics (1), and Consumer Groups (2).

- To check the messages written to the Kafka topic, click the blue Profile icon at the end of the Topic row and choose the Data Explorer tab.

Topics / customer

METRICS DATA EXPLORER CONFIGS LATENCY

DESERIALIZER: Keys: String Values: String

FROM OFFSET: 126413 TO OFFSET: 126428

Offset	Timestamp	Key	Value
126422	Mon, Mar 30 2020, 11:57:10	null	⬢R Johnson
126421	Mon, Mar 30 2020, 11:57:10	null	⬢u Smith
126420	Mon, Mar 30 2020, 11:57:09	null	b Smith
126419	Mon, Mar 30 2020, 11:57:09	null	⬢ Johnson
126418	Mon, Mar 30 2020, 11:57:09	null	⬢ Smith
126417	Mon, Mar 30 2020, 11:57:09	null	⬢ Smith
126416	Mon, Mar 30 2020, 11:57:09	null	⬢L Johnson
126415	Mon, Mar 30 2020, 11:57:09	null	⬢ & Smith
126414	Mon, Mar 30 2020, 11:57:08	null	⬢ Smith
126413	Mon, Mar 30 2020, 11:57:08	null	⬢Y Johnson

1 - 10 of 15

- e) Since we are using an Avro schema in this example, the messages are encoded and not entirely readable. You can select the Avro Deserializer (Avro in both Key and Value fields) and make the data readable.

Topics / customer

DESERIALIZER: Keys: Avro Values: Avro

KEY SCHEMA NAME: CUSTOMERK KEY SCHEMA VERSIONS: Select...

VALUE SCHEMA NAME: CUSTOMER VALUE SCHEMA VERSIONS: 1 Hide schema text

```
{
  "type": "record",
  "namespace": "cloudera.example",
  "name": "customer",
  "fields": [
    {
      "name": "customer_id",
      "type": "int",
      "doc": "primary customer ID"
    },
    {
      "name": "customer_name",
      "type": "string",
      "doc": "customer last name"
    }
  ]
}
```

FROM OFFSET: 126413 TO OFFSET: 126428

Offset	Timestamp	Key	Value
126413	Mon, Mar 30 2020, 11:57:08	null	{"customer_id": 5698, "customer_name": "Johnson"}
126414	Mon, Mar 30 2020, 11:57:08	null	{"customer_id": 8494, "customer_name": "Smith"}
126415	Mon, Mar 30 2020, 11:57:09	null	{"customer_id": 2478, "customer_name": "Smith"}

Related Information

[Apache NiFi User Guide](#)

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:

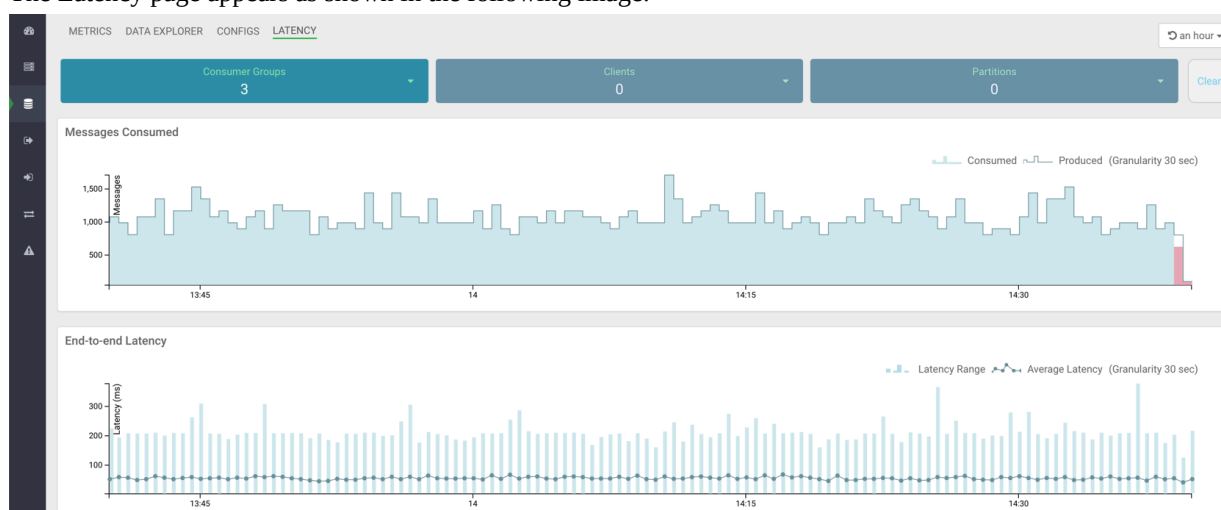
Procedure

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.

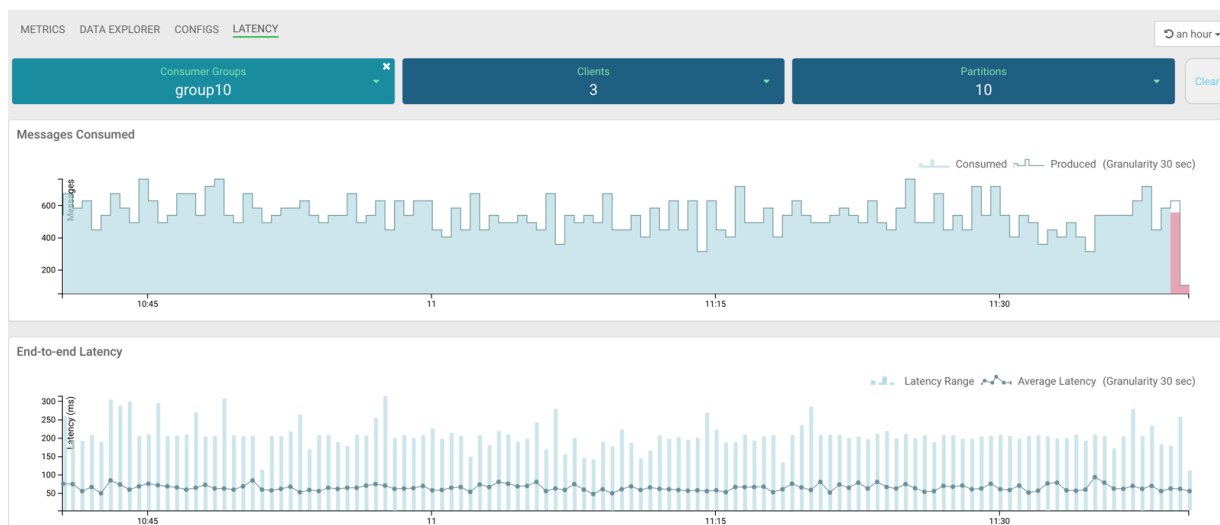
4. To verify the details by individual consumer groups, clients, and partitions, go to the Latency tab.

The Latency page appears as shown in the following image:



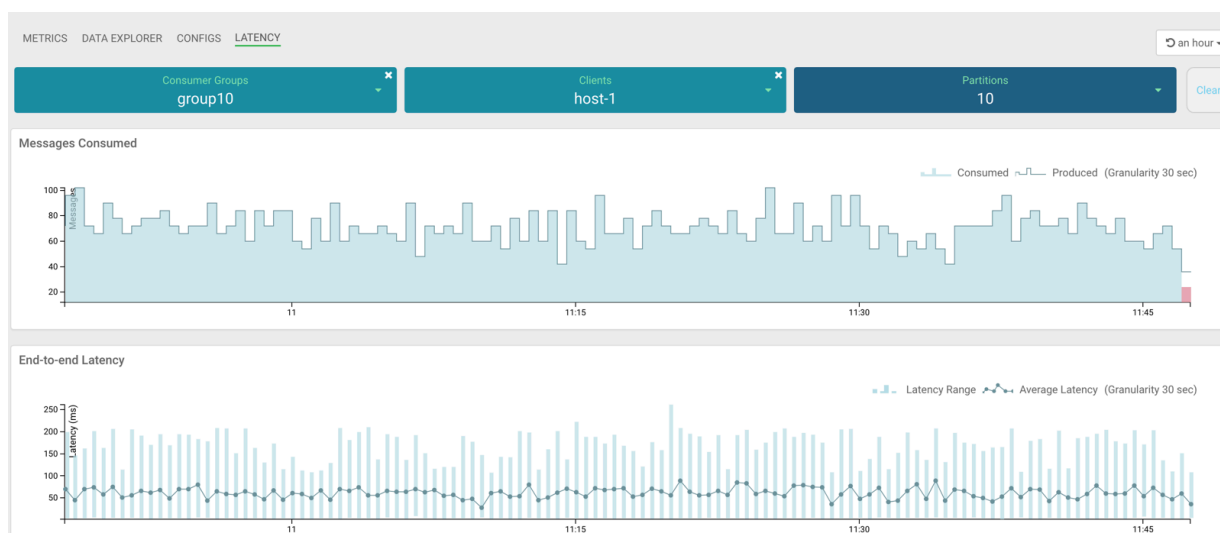
The Latency view gives you a powerful snapshot of the end-to-end latency scenario: number of consumer groups for a topic, number of clients inside a particular consumer group, and number of partitions in a topic along with the Messages Consumed and End-to-end Latency graphs.

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.

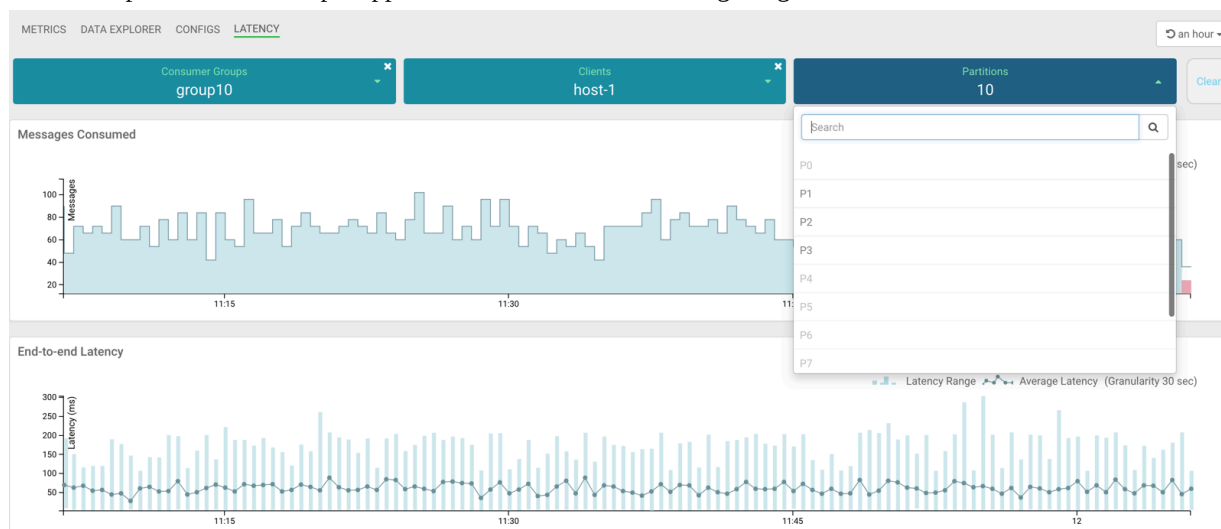
6. Select any client from the Clients drop-down, as shown in the following image:



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.

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

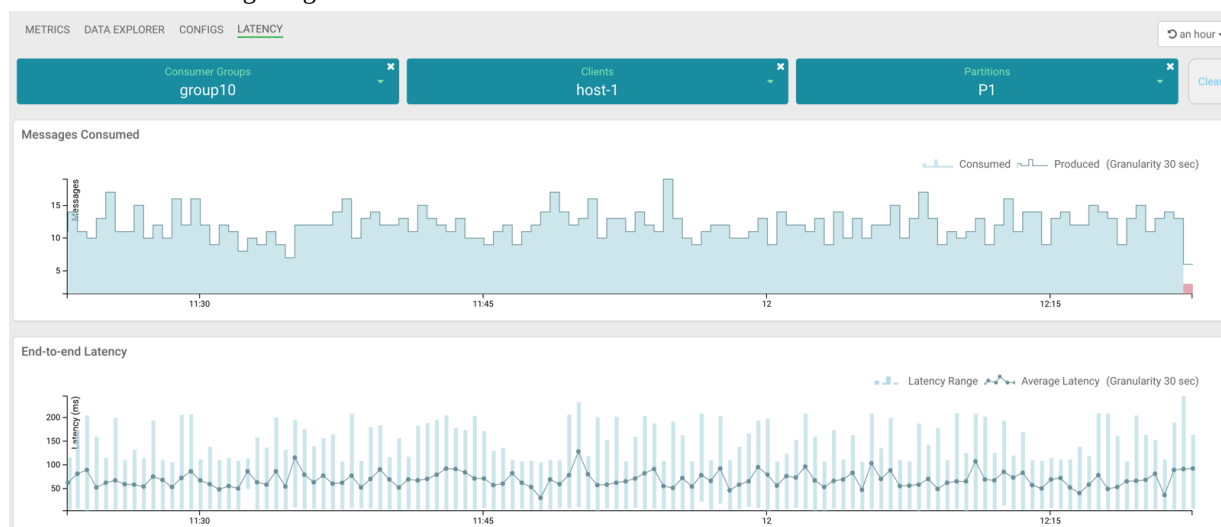
The list of partitions in the topic appears, as shown in the following image:



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.

- 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:

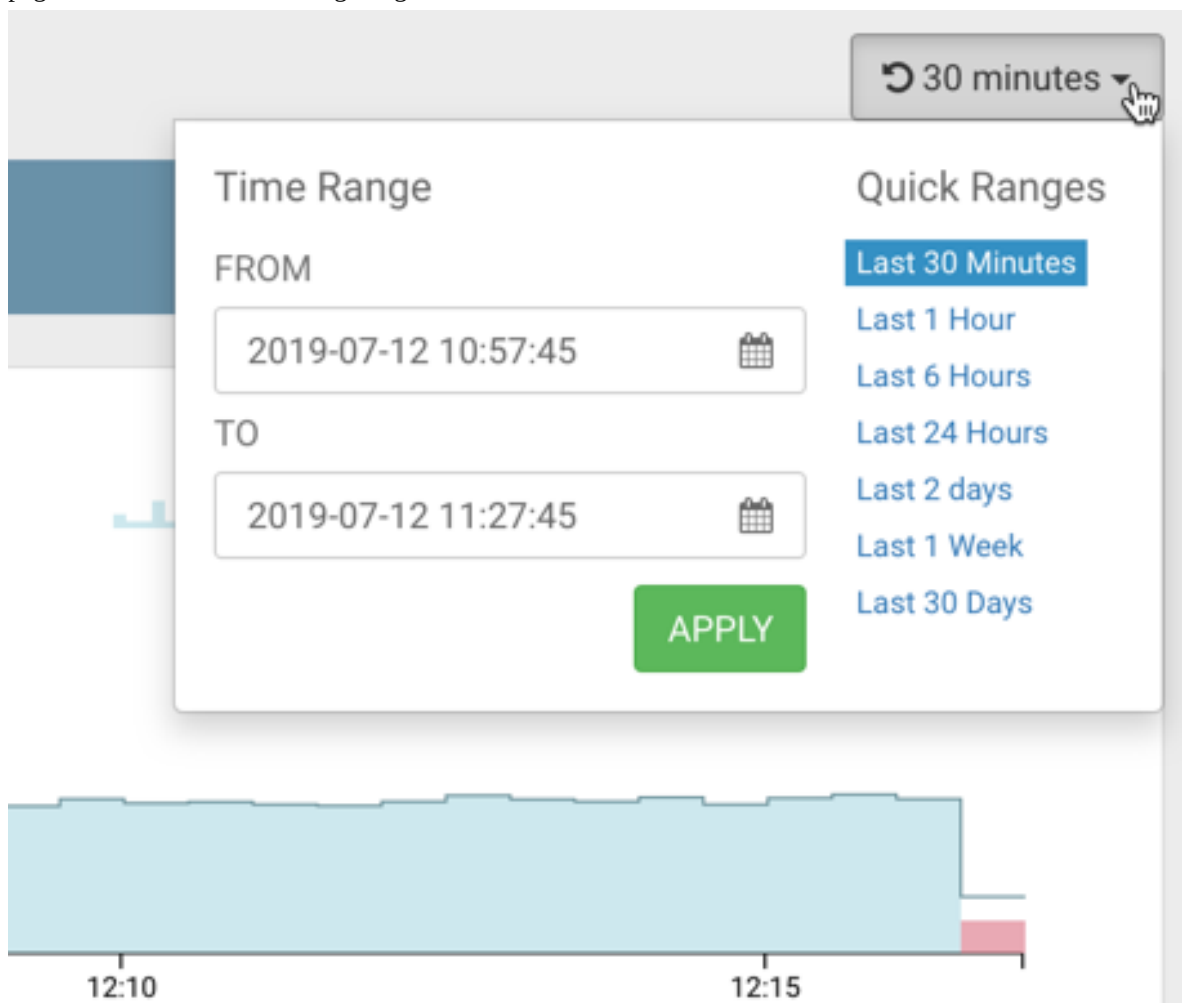


Now you have got details for host-1 client. Likewise, you can fetch details for other clients.

- 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:



Monitoring your data flow

Learn about the different monitoring options for your Kafka ingest data flow.

You can monitor your data flow for information about health and status, and for details about the operation of processors and connections. NiFi records and indexes data provenance information, so you can troubleshoot any issues in real time.

- Data statistics are rolled up on a summary screen (the little table icon on the top right toolbar which lists all the processors). You can use the **MonitorActivity** processor to alert you, if for example you have not received any data in your flow for a specified period of time.
- If you are worried about data being queued up, you can check how much data is currently queued. Process groups show the totals for all queues within them. This can often indicate if there is a bottleneck in your flow, and how far the data has got through in that pipeline.
- Another option to confirm that data has fully passed through your flow is to check out data provenance to see the full history of your data.

Related Information

[Monitoring a Data Flow](#)

Next steps

Learn some ideas on how to build on top of the Kafka ingest flow management use case.

You have built a data flow that generates sample data and moves this data into the Kafka topic. The data moved to Kafka can be picked up by any external services, or you can build another NiFi data flow to ingest these Kafka messages to for example HBase or Hive.

Related Information

[Data ingest use cases in Cloudera Data Flow for Data Hub](#)

Appendix – Schema example

This is a sample schema for the Kafka ingest data flow presented in this use case.

customer.json

```
{
  "type": "record",
  "namespace": "cloudera.example",
  "name": "customer",
  "fields": [
    {
      "name": "customer_id",
      "type": "int",
      "doc": "primary customer ID"
    },
    {
      "name": "customer_name",
      "type": "string",
      "doc": "customer last name"
    }
  ]
}
```