Cloudera Data Flow for Data Hub 7.2.2

Ingesting Data into Azure Data Lake Storage

Date published: 2020-05-01
Date modified: 2020-06-16

https://docs.cloudera.com/

Legal Notice

Contents

Ingesting data into Azure Data Lake Storage..... 4
Understand the use case..... 4
Meet the prerequisites..... 4
Build the data flow..... 5
Create IDBroker mapping..... 7
Create controller services for your data flow..... 8
Configure the processor for your data source..... 10
Configure the processor for merging records..... 12
Configure the processor for your data target..... 13
Start the data flow..... 15
Verify data flow operation..... 16

Monitoring your data flow..... 16

Next steps..... 17

Ingesting data into Azure Data Lake Storage

You can use an Apache NiFi data flow to ingest data into Azure Data Lake Storage (ADLS) in CDP Public Cloud by following these steps.

Understand the use case

Learn how you can use a Flow Management cluster connected to a Streams Messaging cluster to build an end-to-end flow that ingests data to Azure Data Lake Storage (ADLS). This example use case shows you how to use Apache NiFi to move data from Kafka to ADLS.

Why move data to object stores?

Cloud environments offer numerous deployment options and services. There are many ways to store data in the cloud, but the easiest option is to use object stores. Object stores are extremely robust and cost/effective storage solutions with multiple levels of durability and availability. You can include them in your data pipeline, both as an intermediate step and as an end state. Object stores are accessible to many tools and connecting systems, and you have a variety of options to control access.

Apache NiFi for cloud ingestion

You can use Apache NiFi to move data from a range of locations into object stores in CDP Public Cloud.

As a part of Cloudera Data Flow, NiFi offers a scalable solution for managing data flows with guaranteed delivery, data buffering / pressure release, and prioritized queuing. It supports data transfer from nearly any type of data source to cloud storage solutions in a secure and governed manner.

You can use NiFi to move data from various locations into Azure Data Lake Storage (ADLS) in a quick and user-friendly way, by simply dragging and dropping a series of processors on the NiFi user interface. 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 associated with creating an ingest-focused data flow from Apache Kafka into ADLS folders. If you are moving data from a location other than Kafka, see the Getting Started with Apache NiFi for information about how to build a data flow, and about other data get and consume processor options.

Related Information

Getting Started with Apache NiFi
Ingesting Data into Apache Kafka
Ingesting Data into Apache HBase in CDP Public Cloud
Ingesting data into Apache Hive in CDP Public Cloud
Ingesting Data into Apache Kudu in CDP Public Cloud
Ingesting data into Amazon S3

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 and 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.
- You have a Flow Management Data Hub cluster running in your CDP Public Cloud environment.

- You have a Streams Messaging Data Hub cluster in the same CDP environment as the Flow Management cluster.
- Your CDP user has been added to the appropriate pre-defined Ranger access policies to allow access to the NiFi UI.
- You have set up a consumer group in Ranger, and you have a Ranger policy for Kafka allowing your CDP user to access this consumer group.
- You have a Ranger policy on your Kafka topic you will use as a data source allowing the CDP user to consume from the topic.
- You have created a target folder in your Azure Data Lake Storage (ADLS) account for the data to be moved to the cloud.
- You have created a Managed Service Identity in Azure with permissions to write to the ADLS target folder.

Related Information

Understanding roles and resource roles

Creating your first Flow Management cluster

Configuring Ranger policies for Apache NiFi

IDBroker

Build the data flow

Learn how you can create an ingest data flow to move data from Kafka to ADLS. This involves opening Apache NiFi in your Flow Management cluster, adding processors and other data flow objects to your canvas, and connecting your data flow elements.

About this task

You can use the `PutHDFS` or `PutAzureDataLakeStorage` processors to build your ADLS ingest data flows. Regardless of the type of flow you are building, the first steps in building your data flow are generally the same. 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 Nifi 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 `ConsumeKafkaRecord_2_0` processor for data input in your data flow.
   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 `ConsumeKafkaRecord_2_0` processor from the list.
   c) Click Add or double-click the required processor type to add it to the canvas.

   Note: This processor consumes messages from Kafka specifically built against the Kafka 2.0 Consumer API.

3. Add the `MergeRecord` processor.

   This processor merges together multiple record-oriented flowfiles into a single flowfile that contains all of the records of the input flowfiles. It works by creating bins and adding flowfiles to these bins until they are full.

   Once a bin is full, all of the flowfiles are combined into a single output flowfile, and that flowfile is routed to the merged relationship.

   When using the `ConsumeKafkaRecord_2_0` processor, you are pulling small-sized records, so it is practical to merge them into larger files before writing them to ADLS.

4. Add a processor for writing data to ADLS.

Option	
`PutHDFS` processor	The HDFS client writes to ADLS through the Azure Data Lake Storage REST API. This solution leverages centralized CDP security. You can use CDP usernames and passwords for central authentication, and all requests go through IDBroker.
`PutAzureDataLakeStorage` processor	It is an ADLS-specific processor that provides native support for Azure Data Lake Storage Gen 2. This processor is not integrated into CDP's authentication and authorization frameworks, it uses Azure's native authentication technology. If you use it for data ingestion, you need to provide the Azure access credentials file / information when configuring the processor. You can do this by providing your storage account name and storage account key.

   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 processor of your choice from the list.

   c) Click Add or double-click the required processor type to add it to the canvas.

5. Connect the processors to create the data flow by clicking the connection icon in the first processor, and dragging and dropping it on the second processor.

   A Create Connection dialog appears with two tabs: Details and Settings. You can configure the connection's name, flowfile expiration time period, thresholds for back pressure, load balance strategy and prioritization.

   a) Connect `ConsumeKafkaRecord_2_0` with MergeRecord.

   b) Add the success flow of the `ConsumeKafkaRecord_2_0` processor to the `MergeRecord` processor.

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

   d) Connect `MergeRecord` with your target data processor (`PutHDFS` / `PutAzureDataLakeStorage`).

   e) Add the merged flow of the `MergeRecord` processor to the target data processor.

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

6. Optionally, you can add funnels to your flow.

   a) `ConsumeKafkaRecord_2_0`: If any of the Kafka messages are pulled but cannot be parsed, the continents of the message will be written to a separate flowfile and that flowfile will be transferred to the parse.failure relationship. You can connect the failure queue coming from the processor to a failure funnel for parse failures.

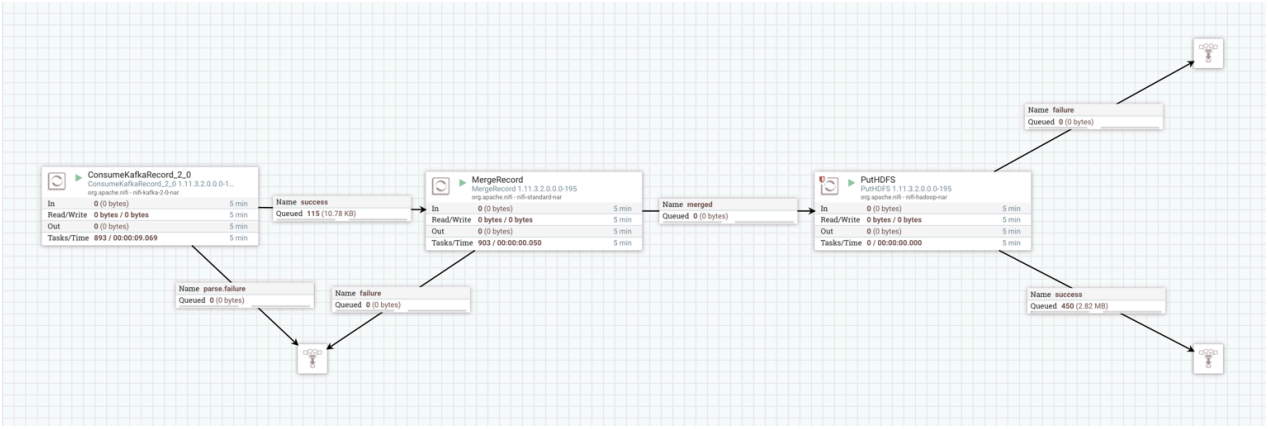
   b) `MergeRecord`: For merge failures, you can connect the failure queue coming from the processor to the failure funnel you used for ConsumeKafkaRecord_2_0.

   c) `PutHDFS` / `PutAzureDataLakeStorage`: You can add success and failure funnels at the end of the data flow and connect them with your target data processor. These funnels help you see where flow files are being routed when your flow is running.

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

Results

This example data flow has been created using the `PutHDFS` processor.



What to do next

If you are using the `PutHDFS` processor, configure IDBroker mapping authorization.

If you are using the `PutAzureDataLakeStorage`processor, you do not need IDBroker mapping, so proceed to configuring your processors in your data flow.

Related Information

Apache NiFi Documentation

Create IDBroker mapping

Learn how you can create the IDBroker mapping for the `PutHDFS` processor for your data flow. To enable your CDP user to utilize the central authentication features CDP provides and to exchange credentials for AWS access tokens, you have to map your CDP user to the correct IAM role.

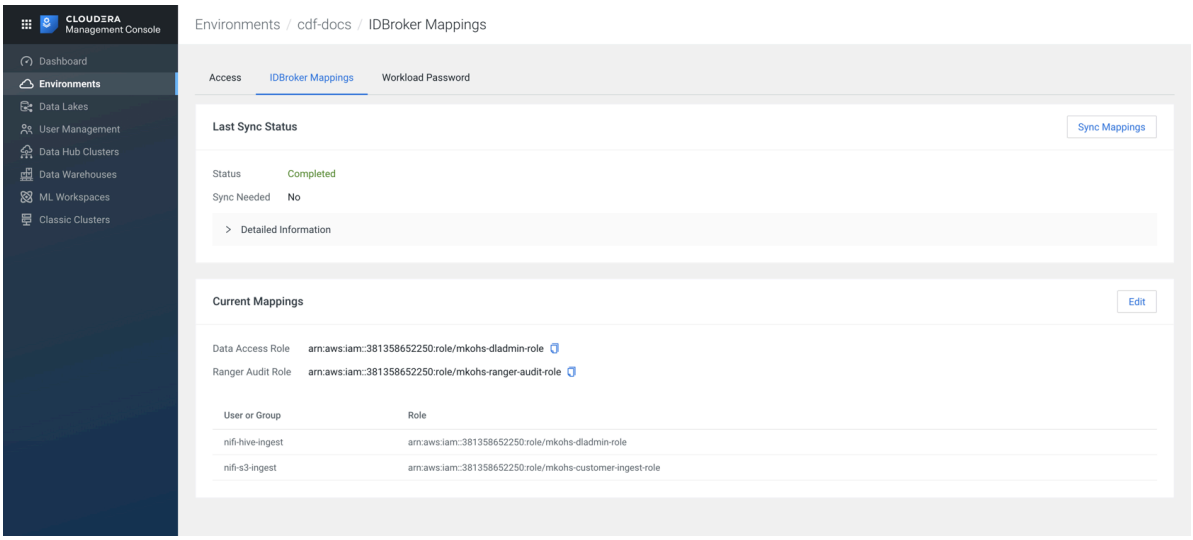
About this task

The option to add / modify these mappings is available from the Management Console in your CDP environment.

Note: This task is only needed if you use the `PutHDFS` processor for accessing your ADLS folder.

Procedure

1. Access IDBroker Mappings.
   a) To access IDBroker Mappings in your environment, click ActionsManage Access.
   b) Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.



2. Add your CDP user or group and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

   You can get the Azure Managed Identity Resource ID in your ADLS Storage Explorer by navigating to Managed Identities Your Managed IdentityPropertiesResource ID. The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
3. Click Save and Sync.

What to do next

Configure controller services for your data flow.

Related Information

IDBroker

Onboarding CDP users and groups for cloud storage

Create a provisioning credential for Azure

Create controller services for your data flow

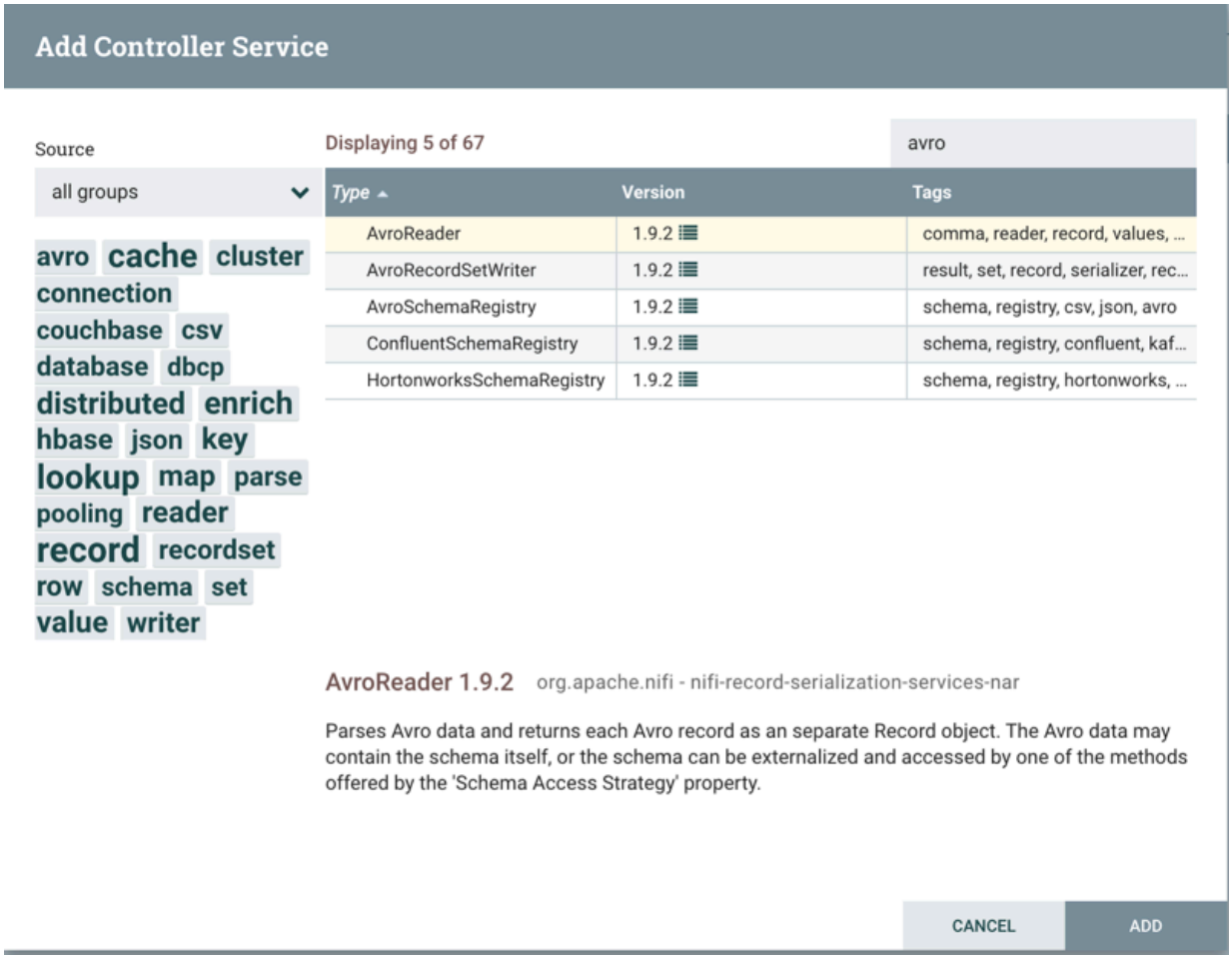
Learn how you can create and configure controller services for an ADLS ingest data flow in CDP Public Cloud. Controller services provide shared services that can be used by the processors in your data flow. You will use these Controller Services later when you configure your processors.

Procedure

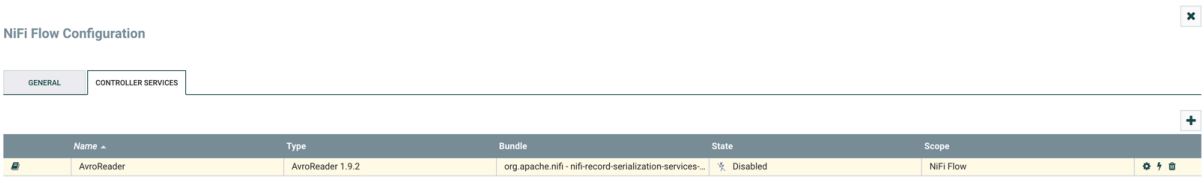
1. To add a Controller Service to your flow, right-click on the Canvas and select Configure from the pop-up menu. This displays the Controller Services Configuration window.
2. Select the Controller Services tab.

3. Click the + button to display the Add Controller Service dialog.

4. Select the required Controller Service and click Add.



5. Perform any necessary Controller Service configuration tasks by clicking the Configure icon in the right-hand column.



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

7. Enable the Controller Service by clicking the Enable button (flash) in the far-right column of the Controller Services tab.

Example

In this example we are using the following controller services:

AvroReader Controller Service

Table 1: AvroReader 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.	HWX Content-Encoded Schema Reference
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.	customer

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 the first line of CSV should be considered a header or a record.	true

CSVRecordSetWriter Controller Service

Table 3: CSVRecordSetWriter 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.	Do Not Write Schema
Schema Access Strategy	Specify how to obtain the schema to be used for interpreting the data.	Use 'Schema Name' Property
Schema Name	Specify the name of the schema to look up in the Schema Registry property. See the Appendix for an example schema.	customer

What to do next

Configure the processors in your data flow.

Related Information

Adding Controller Services for data flows

Apache NiFi Documentation

Configure the processor for your data source

Learn how you can configure the `ConsumeKafkaRecord_2_0` data source processor for your ADLS ingest data flow. You can set up a data flow to move data to Azure Data Lake Storage from many different locations. This

example assumes that you are streaming data from Kafka and shows you the configuration for the relevant data source processor.

Procedure

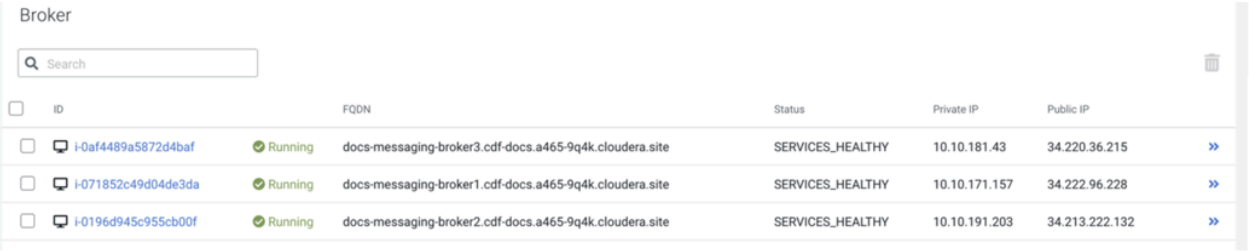
1. Launch the Configure Processor window, by right-clicking the `ConsumeKafkaRecord_2_0` processor and selecting Configure.

   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.
3. 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, as you cannot start the processor until all mandatory properties have been configured.

Example

In this example data flow, the data source is 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.



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 table includes a description and example values for the properties required to configure the example ingest data flow. For a complete list of `ConsumeKafkaRecord_2_0` properties, see the *Apache Nifi Documentation*.

Table 4: ConsumeKafkaRecord_2_0 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(s)	Provide the name of the Kafka Topic(s) to pull from.	Customer
Record Reader	Specify the Record Reader to use for incoming FlowFiles.	AvroReader
Record Writer	Specify the Record Writer to use in order to serialize the data before sending to Kafka.	CSVRecordSetWriter
Security Protocol	Specify the protocol used to communicate with Kafka brokers.	SASL_SSL
SASL Mechanism	Specify the SASL mechanism to use for authentication.	PLAIN

Property	Description	Example value for ingest data flow
Username	Use your CDP workload username to set this Authentication property.	srv_nifi-kafka-ingest
Password	Use your CDP workload password to set this Authentication property.	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
Group ID	Provide the consumer group ID to identify consumers that are within the same consumer group.	nifi-adls-ingest

Note: If you want to move data to ADLS from a location other than Kafka, see the *Apache NiFi Getting Started* for information about other data ingest processor options.

What to do next

Configure the processor for merging your records.

Related Information

Configuring a Processor
Apache NiFi Documentation
Getting Started with Apache NiFi

Configure the processor for merging records

Learn how you can configure the `MergeRecord` processor for your ADLS ingest data flow. You can use it to merge together multiple record-oriented flow files into a large flow file that contains all records of your Kafka data input.

Procedure

1. Launch the Configure Processor window, by right clicking the `MergeRecord` processor and selecting Configure.

   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.
3. When you have finished configuring the options you need, save the changes by clicking Apply.

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

Example

In this example the following settings and properties are used:

Table 5: MergeRecord processor scheduling

Scheduling	Description	Example value for ingest data flow
Automatically Terminate Relationships		original

Table 6: MergeRecord processor properties

Property	Description	Example value for ingest data flow
RecordReader	Specify the Controller Service to use for reading incoming data.	CSVReader

Property	Description	Example value for ingest data flow
RecordWriter	Specify the Controller Service to use for writing out the records.	CSVRecordSetWriter
Merge Strategy	Specify the algorithm used to merge records. The Bin-Packing Algorithm generates a FlowFile populated by arbitrarily chosen FlowFiles.	Bin-Packing Algorithm
Minimum Number of Records	Specify the minimum number of records to include in a bin.	900
Maximum Number of Records	Specify the maximum number of Records to include in a bin.	1000

What to do next

Configure the processor for your data target.

Related Information

Configuring a Processor

Apache NiFi Documentation

Getting Started with Apache NiFi

Configure the processor for your data target

Learn how you can configure the data target processor for your ADLS ingest data flow. This example assumes that you are moving data to Azure Data Lake Storage and shows you how to configure the corresponding processors.

Procedure

1. Launch the Configure Processor window by right clicking the processor you added for writing data to ADLS (`PutHDFS` or `PutAzureDataLakeStorage`) and selecting Configure.

   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.

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

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

   Note: Cloudera recommends using the `PutHDFS` processor for writing data to ADLS in CDP Public Cloud as it leverages the CDP authentication framework.

   In this example, the following properties are used for `PutHDFS`:

Table 7: PutHDFS processor properties

Property	Description	Example value for ingest data flow
Hadoop Configuration Resources	Specify the path to the core-site.xml configuration file. Make sure that the default file system (fs, default.FS) points to the ADLS bucket you are writing to. **Note:** The core-site.xml file is present on every Flow Management cluster. Cloudera Manager stores the right core-site.xml file in the same /etc directory for every cluster.	/etc/hadoop/conf.cloudera.core_settings/core-site.xml
Kerberos Principal	Specify the Kerberos principal (your CDP username) to authenticate against CDP.	srv_nifi-adls-ingest
Kerberos Password	Provide the password that should be used for authenticating with Kerberos.	password

Property	Description	Example value for ingest data flow
Directory	Provide the path to your target directory in Azure expressed in an abfs compatible path format.	abfs://<YourFileSystem>@<YourStorageAccount>.dfs.core.windows.net/<TargetPathWithinFileSystem>

You can leave all other properties as default configurations.

For a complete list of `PutHDFS` properties, see the *processor documentation*.

If you want to use the `PutAzureDataLakeStorage` processor to store the data in ADLS, you have to configure your Azure connection in a secure way by providing:

Note:

`PutAzureDataLakeStorage` has several limitations:

- You can add files to read-only buckets
- There is no check for file overwriting. It is possible to overwrite data.
- To add files to a bucket root level, set the destination with an empty string, rather than " / ".

- Storage Account Name - the name of your Azure storage account that holds the containers where you want to write to)
- Storage Account Key or your SAS Token - the authentication key that allows you to write data to the Azure storage account

  Note: By default, all resources in Azure Storage are secured, and are available only to the account owner. You can grant clients general access to your storage account sharing the storage account key, or you can give more granular access for certain users to specific resources in your storage account using the SAS Token.
- Filesystem Name - the name of your Azure storage account file system where you want to write to
- Directory Name - the name of the folder within your filesystem where you want to write to

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

What to do next

Your data flow is ready to ingest data into ADLS. Start the flow.

Related Information

Configuring a processor

Apache NiFi Documentation

Data ingest use cases in Cloudera Data Flow for Data Hub

Start the data flow

When your flow is ready, you can begin ingesting data into Azure Data Lake Storage folders. Learn how to start your ADLS ingest data flow.

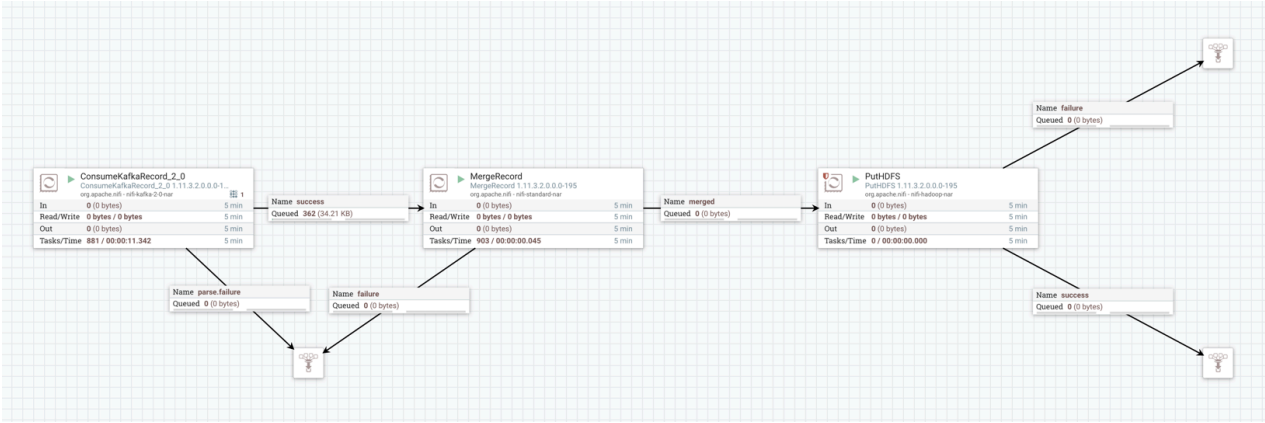
Procedure

1. To get your flow to work, 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. Data should be read from Kafka and it should be written to your predefined Azure ADLS folder.

Results

This example data flow has been created using the `PutHDFS` processor.



What to do next

It is useful to check that data is running through the flow you have created without any errors.

Verify data flow operation

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

About this task

There are a number of ways you can check that data is running through the flow you have built.

Procedure

1. You can verify that NiFi processors are not producing errors.
2. You can look at the processors in the UI, where you can 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.
3. To verify that the data generated appears in ADLS folder specified in the processor, return to your ADLS Storage Explorer, where you should see your files listed in the target folder of your file system.

   You may have to refresh the page depending on your browser/settings.

Monitoring your data flow

Learn about the different monitoring options for your ADLS ingest data flow in CDP Public Cloud.

You can monitor your data flow for information about health, status, and details about the operation of processors and connections. NiFi records and indexes data provenance information, so you can conduct troubleshooting 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 amount of time.

If you are worried about data being queued up, you can check how much data is currently queued. Process groups also conveniently show the totals for any queues within them. This can often indicate if there is a bottleneck in your flow somewhere, and how far the data has got through that pipeline.

Another option to check that data has fully passed through your flow is to check out data provenance to see the full history of your data.

Next steps

Learn about the different options that you have after building a simple ADLS ingest data flow in CDP Public Cloud.

Moving data to the cloud is one of the cornerstones of any cloud migration. Cloud environments offer numerous deployment options and services. This example data flow enables you to easily design more complex data flows for moving and processing data as part of cloud migration efforts.

You can build a combination of on-premise and public cloud data storage. You can use this solution as a path to migrate your entire data to the cloud over time—eventually transitioning to a fully cloud-native solution or to extend your existing on-premise storage infrastructure, for example for a disaster recovery scenario. Cloud storage can provide secure, durable, and extremely low-cost options for data archiving and long-term backup for on-premise datasets.

You can also use cloud services without storing your data in the cloud. In this case you would continue to use your legacy on-premise data storage infrastructure, and work with on-demand, cloud-based services for data transformation, processing and analytics with the best performance, reliability and cost efficiency for your needs. This way you can manage demand peaks, provide higher processing power, and sophisticated tools without the need to permanently invest in computer hardware.