Cloudera DataFlow for Data Hub 7.2.11

Streaming Analytics

Date published: 2019-12-16
Date modified: 2021-09-09

https://docs.cloudera.com/

Legal Notice

Contents

Streaming Analytics in Cloudera

Cloudera Streaming Analytics (CSA) offers real-time stream processing and streaming analytics powered by Apache Flink. Flink implemented on CDP provides a flexible streaming solution with low latency that can scale to large throughput and state. Additionally to Flink, CSA includes SQL Stream Builder to offer data analytical experience using SQL queries on your data streams.

Key features of Cloudera Streaming Analytics

SQL Stream Builder

SQL Stream Builder is a job management interface to compose and run Streaming SQL on streams, as well as to create durable data APIs for the results.

Cloudera Platform

Implementing Flink on the Cloudera Platform allows you to easily integrate with Runtime components, and have all the advantages of cluster and service management with Cloudera Manager.

Streaming Platform

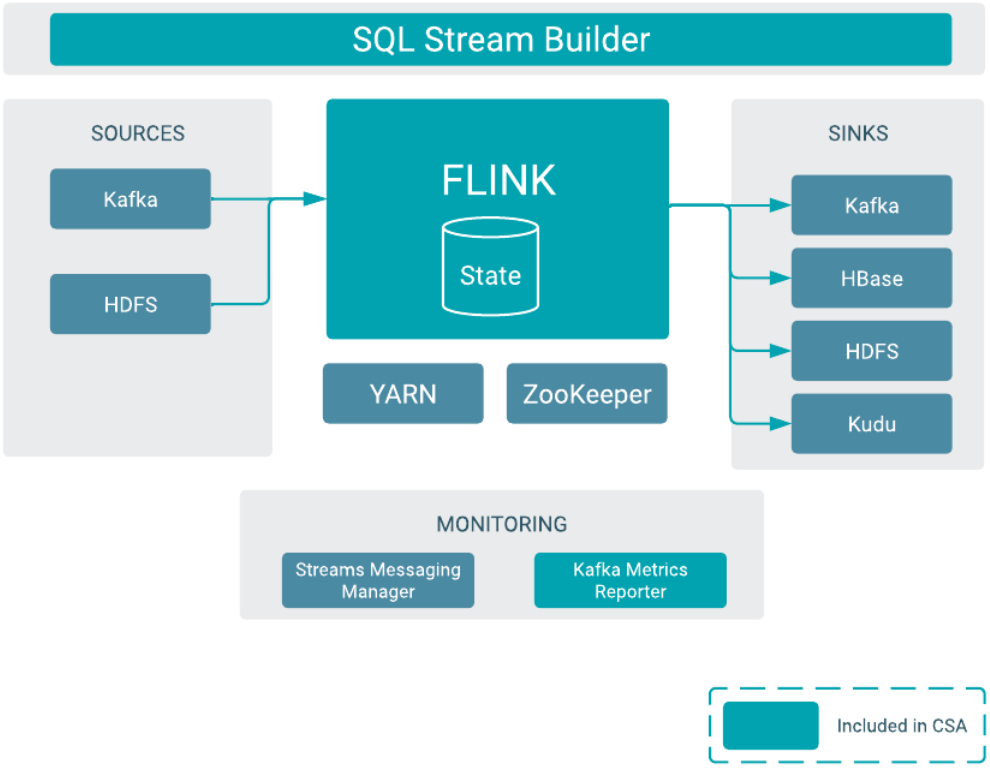
For streaming analytics, CSA fits into a complete streaming platform augmented by Apache Kafka, Schema Registry, Streams Messaging Manager in the Cloudera Runtime stack.

Supported Connectors

CSA offers Kafka, HBase, HDFS, Kudu and Hive as connectors to choose based on the requirements of your application deployment.

Monitoring Solutions

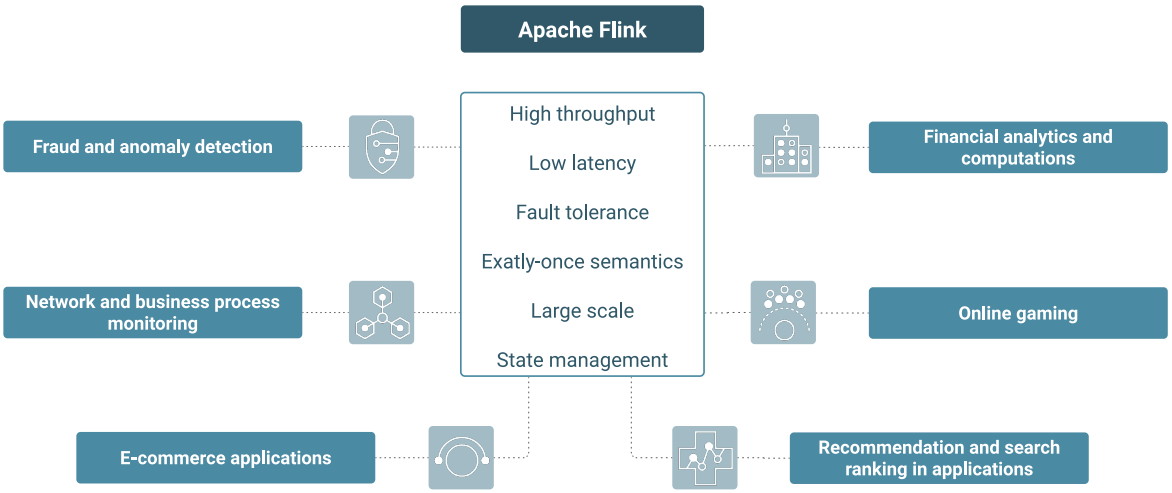
Within CSA, Kafka Metrics Reporter, Streams Messaging Manager and the reworked Flink Dashboard helps you monitor and troubleshoot your Flink applications.



What is Apache Flink?

Flink is a distributed processing engine and a scalable data analytics framework. You can use Flink to process data streams at a large scale and to deliver real-time analytical insights about your processed data with your streaming application.

Flink is designed to run in all common cluster environments, perform computations at in-memory speed and at any scale. Furthermore, Flink provides communication, fault tolerance, and data distribution for distributed computations over data streams. A large variety of enterprises choose Flink as a stream processing platform due to its ability to handle scale, stateful stream processing, and event time.

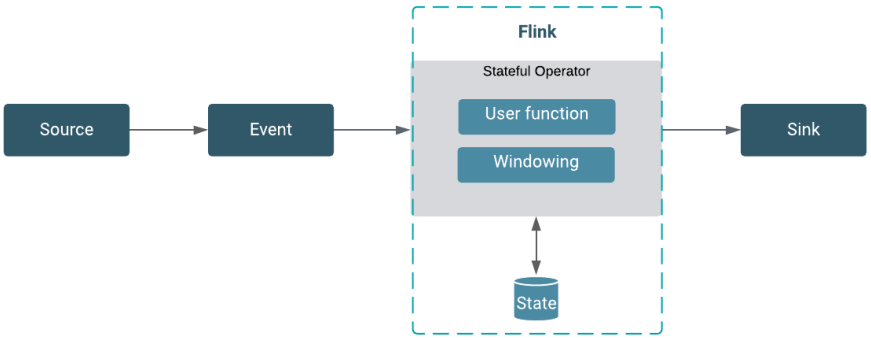


Core Features of Flink

Learn more about the specific details of Flink architecture, the DataStream API, how Flink handles event time and watermarks, and how state management works in Flink.

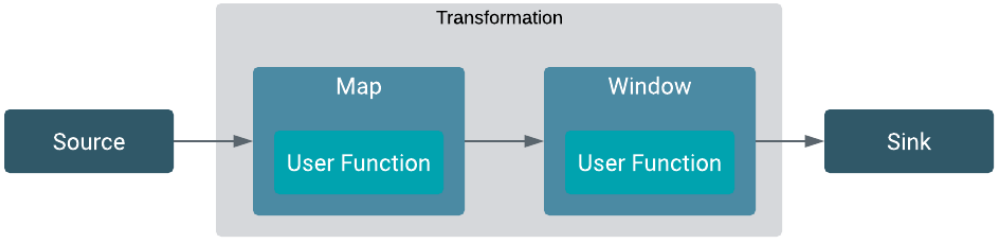
Architecture

The two main components for the task execution process are the Job Manager and Task Manager. The Job Manager on a master node starts a worker node. On a worker node the Task Managers are responsible for running tasks and the Task Manager can also run more than one task at the same time. The resource management for the tasks are completed by the Job manager in Flink. In a Flink cluster, Flink jobs are executed as YARN applications. HDFS is used to store recovery and log data, while ZooKeeper is used for high availability coordination for jobs.



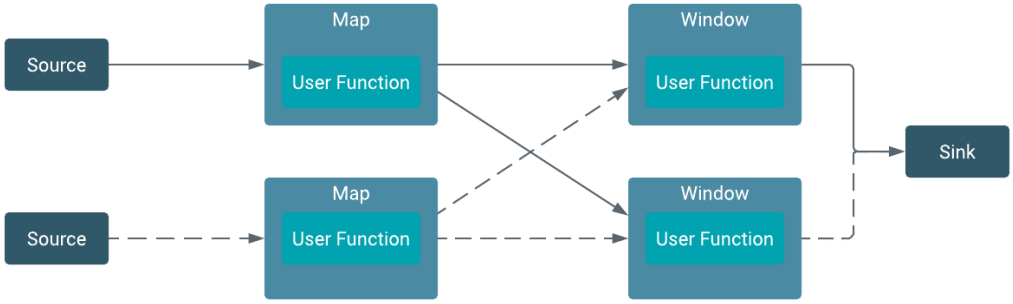
DataStream API

The DataStream API is used as the core API to develop Flink streaming applications using Java or Scala programming languages. The DataStream API provides the core building blocks of the Flink streaming application: the datastream and the transformation on it. In a Flink program, the incoming data streams from a source are transformed by a defined operation which results in one or more output streams to the sink.



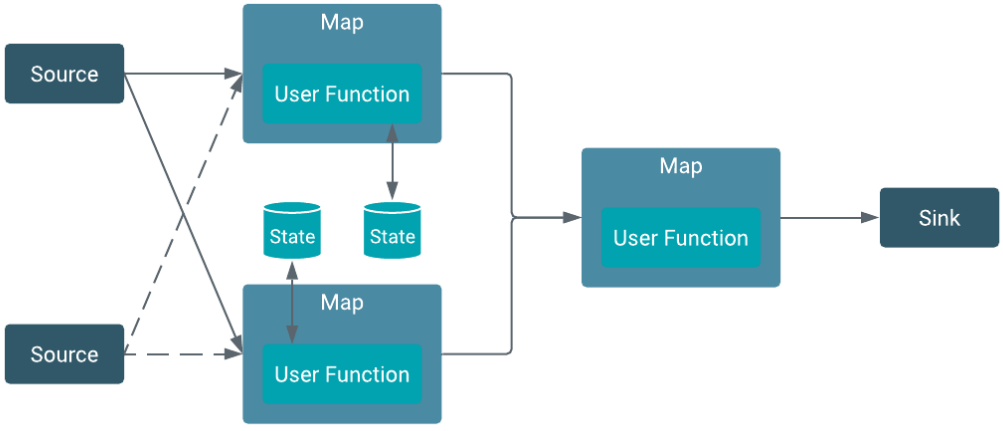
Operators

Operators transform one or more DataStreams into a new DataStream. Programs can combine multiple transformations into sophisticated data flow topologies. Other than the standard transformations like map, filter, aggregation, you can also create windows and join windows within the Flink operators. On a dataflow one or more operations can be defined which can be processed in parallel and independently to each other. With windowing functions, different computations can be applied to different streams in the defined time window to further maintain the processing of events. The following image illustrates the parallel structure of dataflows.



State and state backend

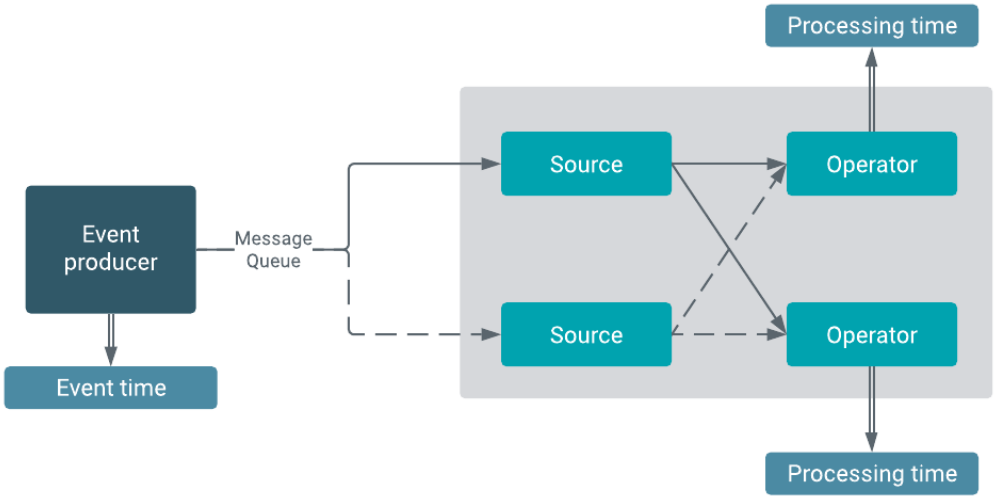
Stateful applications process dataflows with operations that store and access information across multiple events. You can use Flink to store the state of your application locally in state backends that guarantee lower latency when accessing your processed data. You can also create checkpoints and savepoints to have a fault-tolerant backup of your streaming application on a durable storage.



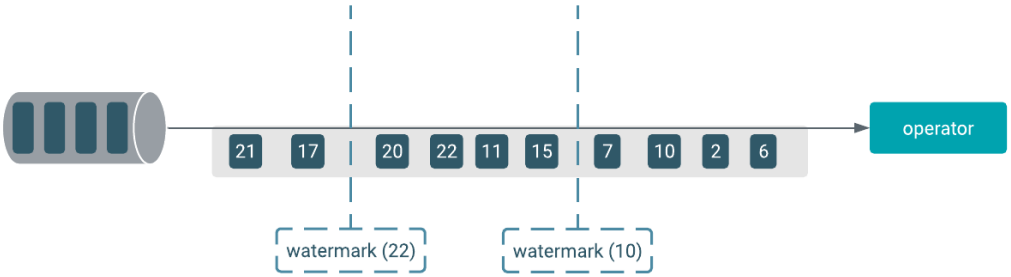
Event time and watermark

In time-sensitive cases where the application uses alerting or triggering functions, it is important to distinguish between event time and processing time. To make the designing of applications easier,

you can create your Flink application either based on the time when the event is created or when it is processed by the operator.

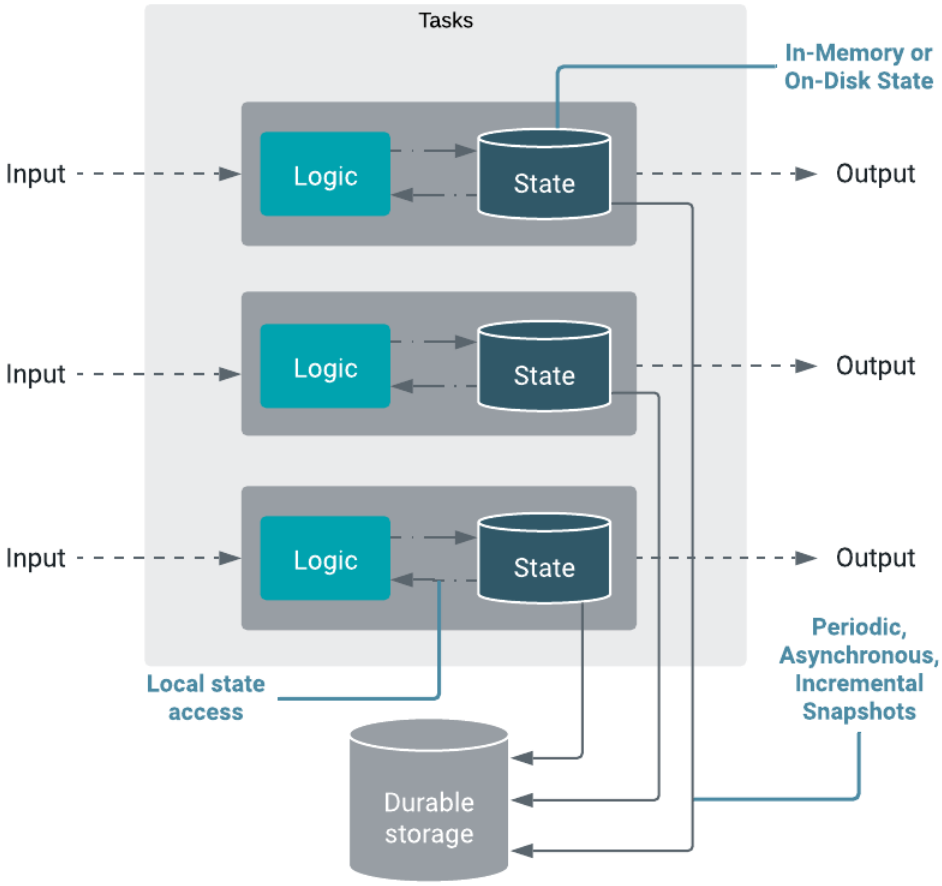


With only the event time, it is not clear when the events are processed in the application. To track the time for an event time based application, watermark can be used.



Checkpoints and savepoints

Checkpoints and savepoints can be created to make the Flink application fault tolerant throughout the whole pipeline. Flink contains a fault tolerance mechanism that creates snapshots of the data stream continuously. The snapshot includes not only the dataflow, but the state attached to it. In case of failure, the latest snapshot is chosen and the system recovers from that checkpoint. This guarantees that the result of the computation can always be consistently restored. While checkpoints are created and managed by Flink, savepoints are controlled by the user. A savepoint can be described as a backup from the executed process.



Related Information

Flink application structure
Configuring RocksDB state backend
Enabling checkpoints for Flink applications
Enabling savepoints for Flink applications

Introduction to SQL Stream Builder

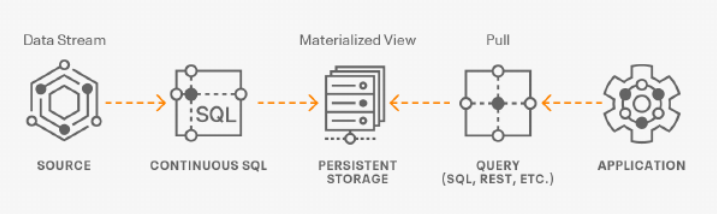
Cloudera Streaming Analytics offers an easy to use and interactive SQL Stream Builder as a service to create queries on streams of data through SQL.

The SQL Stream Builder (SSB) is a comprehensive interactive user interface for creating stateful stream processing jobs using SQL. By using SQL, you can simply and easily declare expressions that filter, aggregate, route, and otherwise mutate streams of data. SSB is a job management interface that you can use to compose and run SQL on streams, as well as to create durable data APIs for the results.

What is Continuous SQL?

SSB runs Structured Query Language (SQL) statements continuously, this is called Continuous SQL or Streaming SQL. Continuous SQL can run against both bounded and unbounded streams of data. The results are sent to a sink of some type, and can be connected to other applications through a Materialized View interface. Compared to traditional SQL, in Continuous SQL the data has a start, but no end. This means that queries continuously process results. When

you define your job in SQL, the SQL statement is interpreted and validated against a schema. After the statement is executed, the results that match the criteria are continuously returned.



Integration with Flink

SSB runs in an interactive fashion where you can quickly see the results of your query and iterate on your SQL syntax. The executed SQL queries run as jobs on the Flink cluster, operating on boundless streams of data until cancelled. This allows you to author, launch, and monitor stream processing jobs within SSB as every SQL query is a Flink job. You can use Flink and submit Flink jobs without using Java, as SSB automatically builds and runs the Flink job in the background.

As a result of Flink integration, you are able to use the basic functionalities offered by Flink. You can choose exactly once processing, process your data stream using event time, save your jobs with savepoints, and use Flink DDL to create tables and use custom connectors based on your requirements. As a result of customizable connectors, you are able to enrich your streaming data with data from slowly changing connectors.

The following table summarizes the supported connectors and how they can be used in SSB:

Connector	Type	Description
Kafka	source/sink	Supported as exactly-once-sink
Hive	source/sink	Can be used as catalog
Kudu	source/sink	Can be used as catalog
Schema Registry	source/sink	Can be used as catalog
JDBC	source/sink	Can be used with Flink DDL. PostgreSQL, MySQL and Hive are supported.
Filesystems	source/sink	Filesystems such as HDFS, S3 and so on. Can be used with Flink DDL
Webhook	sink	Can be used as HTTP POST/PUT with templates and headers
PostgreSQL	sink	Materialized View connection for reading views. Can be used with anything that reads PostgreSQL wire protocol
REST	sink	Materialized View connection for reading views. Can be used with anything that reads REST (such as notebooks, applications, and so on)

Key features of SSB

SQL Stream Builder (SSB) within Cloudera supports out-of-box integration with Flink and Kafka, as virtual table sink and source. For integration with Business Intelligence tools you can create Materialized Views.

Tables

Tables are a core abstraction in SSB. Just like typical databases, they provide the interface for running queries. Data can be queried from tables, and results can be sent to tables. Tables do not have native storage in SSB, rather they reference data connectors for Kafka, Hive, Kudu and so on. If needed, tables have a schema definition as well. In the case of Kafka, table definition includes

a rich interface for defining schema, and run-time characteristics like timestamps and various consumer/producer settings.

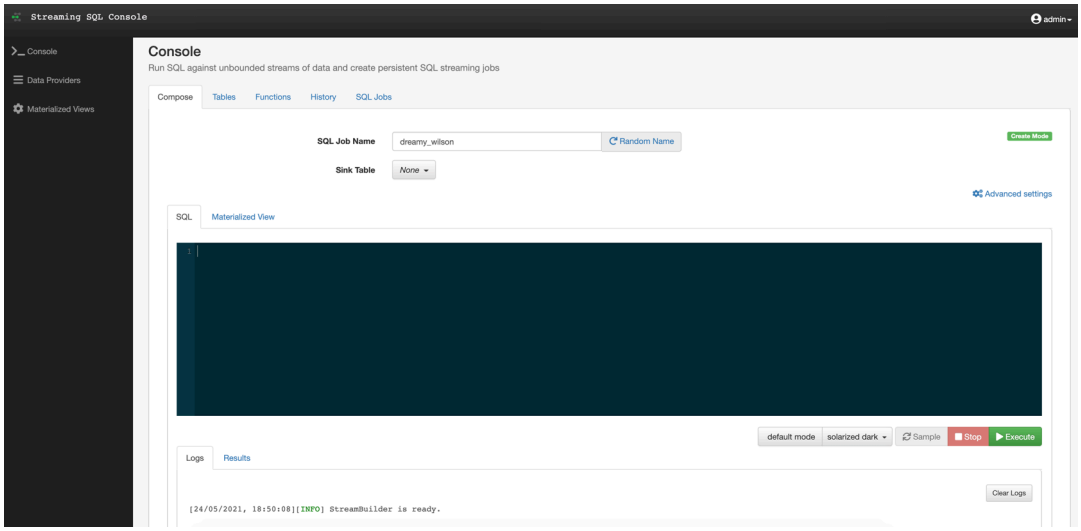
For more information about the supported data connectors in SSB, see the Integration with Flink section.

Catalog Support

In addition to creating tables manually, you can access tables in external systems using the supported Flink catalogs. SSB supports Hive, Kudu and Schema Registry as catalogs.

Streaming SQL Console

SSB comes with an interactive user interface that allows you to easily create, and manage your SQL jobs in one place. It allows you to create and iterate on SQL statements with robust tooling and capabilities. Query parsing is logged to the console, and results are sampled back to the interface to help with iterating on the SQL statement as required.



Materialized Views

SSB has the capability to materialize results from a Streaming SQL query to a persistent view of the data that can be read through REST and over the PG wire protocol. Applications can use this mechanism to query streams of data in a way of high performance without deploying additional database systems. Materialized Views are built into the SQL Stream Builder service, and require no configuration or maintenance. The Materialized Views act like a special kind of sink, and can even be used in place of a sink. They require no indexing, storage allocation, or specific management.

Detect Schema

SSB is capable of reading JSON messages in a topic, identifying their data structure, and sampling the schema to the UI. This is an useful function when you do not use Schema Registry.

Input Transform

In case you are not aware of the incoming data structure or raw data is being collected from for example sensors, you can use the Input Transform to clean up and organize the incoming data before querying. Input transforms also allow access to Kafka header metadata directly in the query itself. Input transforms are written in Javascript and compiled to Java bytecode deployed with the Flink jar.

User Defined Functions

You can create customized and complex SQL queries by using User Defined Functions to enrich your data, apply computations or a business logic on it. User defined functions are written in Javascript, and compiled to Java bytecode deployed with the Flink jar.

Related Information

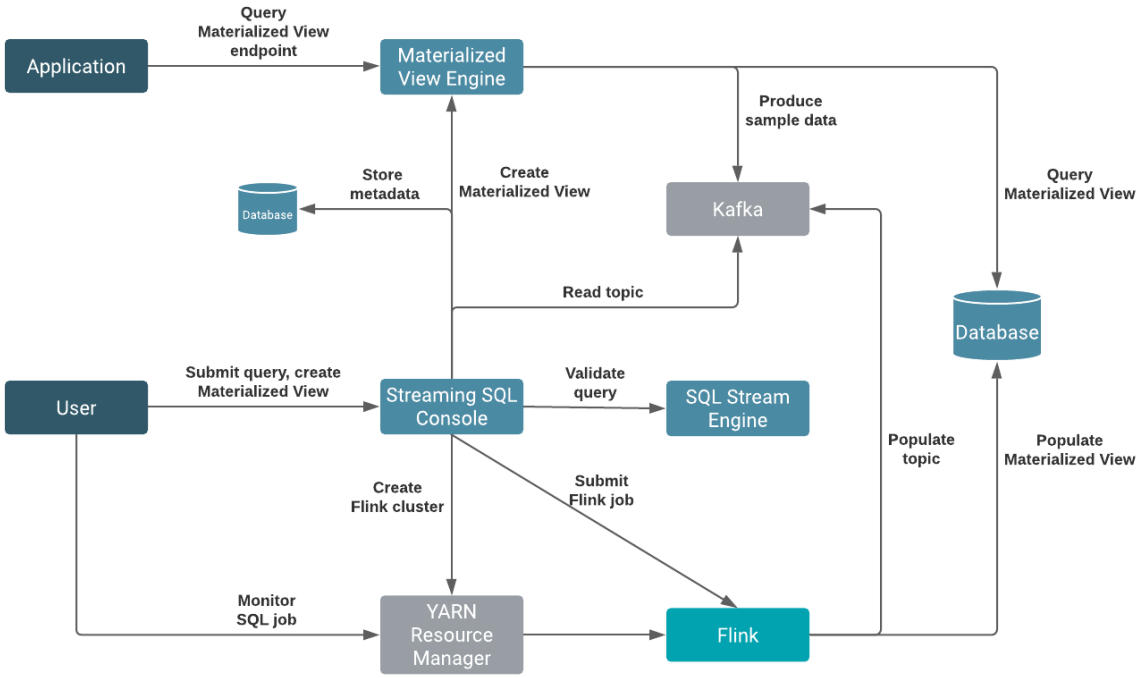
Integration with Flink

SQL Stream Builder architecture

The SQL Stream Builder (SSB) service is integrated on Cloudera Data Platform (CDP) and connected to Flink and its services. The SSB architecture includes Streaming SQL Console, SQL Stream Engine and Materialized View Engine. These main components within SSB are responsible for executing jobs, populating topics, creating metadata and querying data that happens in the background.

SSB consists of the following main components:

- SQL Stream Engine
- Streaming SQL Console
- Materialized View Engine



The primary point of user interaction for SQL Stream Builder is the Console component. When you submit a query using the Streaming SQL Console, a Flink job is automatically created in the background on the cluster. SSB also requires a Kafka service on the same cluster. This mandatory Kafka service is used to automatically populate topics for the websocket output. The websocket output is needed for sampling data to the Console, and when no table is added to output the results of the SQL query.

When a Materialized View query is submitted, Flink generates the data to the Materialized View database from which the Materialized View Engine queries the required data.

In CDP Public Cloud, PostgreSQL is supported as a default database for SQL Stream Builder and the Materialized View database.