Solace Spark Connector

User Guide

Solace Corporation

Version 1.2.0

Table of Contents

Solace Spark Connector

Solace Spark Connector streams data from Solace PubSub+ broker to Spark Data Sources. The connector is based Spark Data Source API.

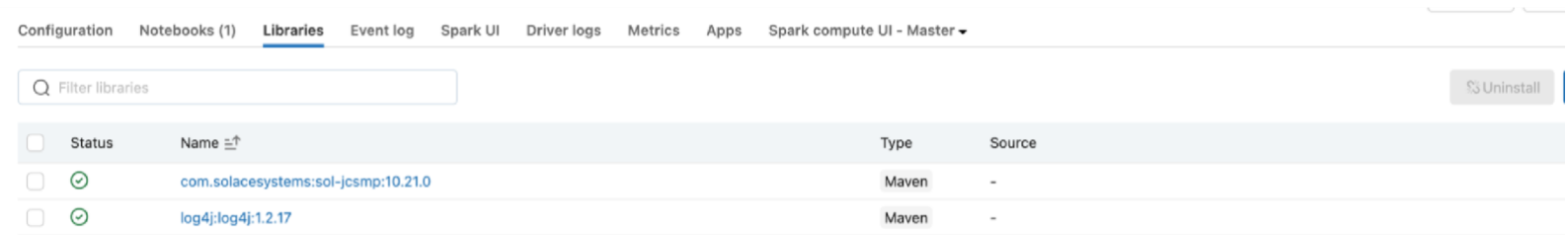
Requirements

Apache Spark 3.1.2, Scala 2.12

Installation on Databricks Cluster

Create a Databricks cluster with Runtime: 9.1 LTS with above Spark version and follow the steps below

1. In the libraries section upload the pubsubplus-connector-spark jar and also add the following maven dependencies using install new option available in Databricks cluster libraries section



Running as a Job

If you are running Spark Connector as a job, use the jar files provided as part of distribution to configure your job. In case of thin jar you need to provide dependencies as in above screenshot for the job. For class name and other connector configuration please refer sample scripts and configuration option sections

Thin Jar vs Fat Jar

Thin jar is light weight jar where only Spark Streaming API related dependencies are included. In this case Solace and Log4j related dependencies should be added during configuration of Spark Job. Spark supports adding these dependencies via maven or actual jars.

Solace JCSMP – 10.21.0
Log4j – 1.2.17

Fat jar includes all the dependencies and it can be used directly without any additional dependencies.

Solace Spark Schema

Solace Spark Connector transforms the incoming message to Spark row with below schema.

Column Name	Column Type
Id	String
Payload	Binary
PartitionKey	String
Topic	String
TimeStamp	Timestamp
Headers	Map<string, binary>

Using Sample Script

Databricks

1. **Solace_Read_Stream_Script.txt**

Create a new notebook in Databricks environment and set the language to Scala. Copy the Script to notebook and provide the required details. Once started, script reads data from Solace Queue and writes to parquet files.

2. **Read_Parquet_Script.txt**

This scripts reads the data from parquet and displays the count of records. The output should match the number of records available in Solace queue before processing.

Spark Job or Spark Submit

Spark Job or Spark Submit requires jar as input. You can convert above Scala scripts to jar and provide it as input and add pubsubplus-connector-solace jar(thin or fat) as dependency. In case of thin jar please note that additional dependencies need to be configured as mentioned in Thin Jar vs Fat Jar section.

Configuration Options

Config Option	Type	Valid Values	Default Value	Description
host	String	tcp(s)://hostname:port	Empty	Fully Qualified Solace Hostname with protocol and port number
vpn	String		Empty	Solace VPN Name
username	String		Empty	Solace Client Username
password	String			Solace Client Password
queue	String		Empty	Solace Queue name

batchSize	Integer		1	Set number of messages to be processed in batch. Default is set to 1
ackLastProcessedMessages	Boolean		false	Set this value to true if connector needs to determine
				processed messages in last run. The connector purely depends on offset file generated during Spark commit. In some cases connector may acknowledge the message based on offset but same may not be available in your downstream systems. In general we recommend leaving this false and handle process/reproces s of messages in downstream systems

skipDuplicates	Boolean		false	Set this value to true if connector needs check for duplicates before adding to Spark row. This scenario occurs when the tasks are executing more than expected time and message is not acknowledged before the start of next task. In such cases the message will not be added to Spark row.

offsetIndicator	String		MESSAGE_ID	Set this value if your Solace Message has unique ID in message header. Supported Values are 1. MESSAGE_ID 2. CORRELATION_ID 3. APPLICATION_MESSAGE_ID 4.<CUSTOM_USER_PROPERTY> CUSTOM_USER_PROPERTY refers to one of headers in user properties Header. Note: Default value uses replication group message ID property as offset indicator. A replication group message ID is an attribute of Solace messages, assigned by the event broker delivering the messages to the queue and topic

				endpoints, that uniquely identifies a message on a particular queue or topic endpoint within a high availability (HA) group and replication group of event brokers.
includeHeaders	Boolean		false	Set this value to true if message headers need to be included in output
partitions	Integer		1	Sets the number of consumers for configured queue. Equal number of partitions are created to process data received from consumers

createFlowsOnSameSession	Boolean		false	If enabled consumer flows are enabled on same session. The number of consumer flows is equal to number of partitions configured. This is helpful when users want to optimize on number of connections created from Spark. By default the connector creates a new connection for each consumer.