How to create a real-time scalable streaming app using Apache NiFi, Apache Pulsar, and Apache Flink SQL

Author: Tim Spann, Developer Advocate, StreamNative

Introduction

Building streaming applications can be a difficult task — often, just figuring out how to get started can be overwhelming. In this article, we'll cover how and when to use NiFi with Pulsar and Flink and how to get started. Apache NiFi, Apache Pulsar, and Apache Flink are robust, powerful open-source platforms that enable running any size application at any scale. This enables us to take our quickly developed prototype and deploy it as an unbreakable clustered application ready for internet-scale production workloads of any size.

Using a drag-and-drop tool can help reduce that difficulty and allow us to get the data flowing. By utilizing Apache NiFi, we can quickly go from ideation to data streaming as live events in our Pulsar topics. Once we have a stream, building applications via SQL becomes a much more straightforward premise. The ability to rapidly prototype, iterate, test, and repeat are critical in modern cloud applications.

We are now faced with a familiar scenario where it appears like traditional database-driven or batch applications that most data engineers and programmers use.

Before we cover how to get started with NiFi, Pulsar, and Flink, let’s discuss how and why these platforms work for real-time streaming. **ChatGPT** said:

How do I use Apache NiFi with Apache Pulsar and Apache Flink?

Apache NiFi, Apache Pulsar, and Apache Flink are all open-source technologies that can be used together to build powerful data processing pipelines. To use Apache NiFi with Apache Pulsar and Apache Flink, you would first need to install and configure each of these technologies on your system.

Once you have them installed and configured, you can use Apache NiFi to ingest data from various sources and route it to Apache Pulsar for real-time streaming and processing. You can then use Apache Flink to process the data streams from Apache Pulsar in real-time, and perform any necessary transformations or aggregations on the data.

To do this, you would need to create a data flow in Apache NiFi that ingests data from the desired sources and routes it to Apache Pulsar. You can then use the Apache Pulsar and Apache Flink connectors in Apache NiFi to connect to your Apache Pulsar and Apache Flink clusters and transfer data between them.

Overall, using Apache NiFi with Apache Pulsar and Apache Flink allows you to build highly scalable and efficient data processing pipelines that can handle large volumes of data in real-time.

Why use Apache NiFi with Apache Pulsar and Apache Flink?

Architects and developers have many options for building real-time scalable streaming applications, so why should they utilize the combination of Apache NiFi, Apache Pulsar, and Apache Flink? The initial reason I started utilizing this combination of open-source projects is the ease of getting started. I always recommend first starting with the simplest way for anyone exploring solutions to new use cases or problems. The simplest solution to start data flowing from a source or extract is usually Apache NiFi.

Apache NiFi is a drag-and-drop tool that works on live data, so I can quickly point to my source of data and start pulling or triggering data from it. Since Apache NiFi supports hundreds of sources, often, the data I want to access is a straightforward drag-and-drop. Once the data starts flowing, I can build an interactive streaming pipeline one step at a time in real time with live data flowing. I can examine the state of that data before building the next step. With the combination of inter-step queues and data provenance, I know the current state of the data and all the previous states with their extensive metadata. In an hour or less, I can usually build the

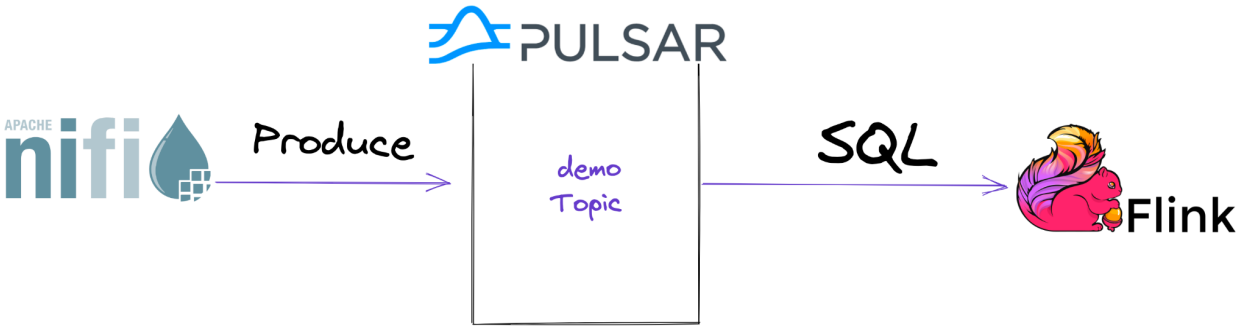
ingest, routing, transforming, and essential data enrichment. The final step of the Apache NiFi portion of our streaming application is to stream the data to Apache Pulsar utilizing the NiFi-Pulsar connector. Next in our application development process is to provide routing and additional enhancements before data is consumed from Pulsar.

Within Apache Pulsar, we can utilize Functions written in Java, Python, or Go to enrich, transfer, add schemas and route our data in real time to other topics.

When

Quick Criteria

Your data source	Recommended platform(s)
`JSON REST Feed`	Looks like a good fit for NiFi+
`Relational Tables CDC`	Looks like a good fit for NiFi+
`Complex ETL and Transformation`	Look at Spark + Pulsar
`Source requires joins`	Look at Flink applications
`Large batch data`	Look at native applications
`Mainframe data sources`	Look at existing applications that can send messages
Websocket streams of data	Looks like a good fit for NiFi+
`Clickstream Data`	Looks like a good fit for NiFi+
Sensor data from devices	Just stream directly to Pulsar via MQTT.



You may ask when I should use the combination of Apache NiFi - Apache Pulsar - Apache Flink to build my apps, and what if I only need Pulsar? That can be the case many times. Suppose you have an existing application that produces messages or events being sent to Apache Kafka, MQTT, RabbitMQ, REST, WebSockets, JMS, RPC, or RocketMQ. In that case, you can just point that program at a Pulsar cluster or rewrite to us the superior native Pulsar libraries.

After an initial prototype with NiFi, if it is too slow, you can deploy your flow on a cluster and resize with Kubernetes, expand out vertically with more RAM and CPU cores or look for solutions with my streaming advisors. The great thing about Apache NiFi is that there are many pre-built solutions, demos, examples, and articles for most use cases spanning from REST to CDC to logs to sensor processing.

If you hit a wall at any step, then perhaps Apache NiFi is not right for this data. If it is mainframe data, complex ingest rules require joins, many enrichment steps, and complex ETL or ELT. I suggest looking at custom Java code, Apache Spark, Apache Flink, or another tool.

If you don't have an existing application, but your data requires no immediate changes and comes from a known system, perhaps you can use a native Pulsar source. Check them out at https://hub.streamnative.io/. If you need to do some routing, enrichment, and enhancement, you may want to look at Pulsar Functions which can take your raw data in that newly populated topic event at a time to do that.

If you have experience with an existing tool such as Spark and have an environment, that may be a good way for you to bring this data into the Pulsar stream. This is especially true if there are a lot of ETL steps or you are combining it with Spark ML.

There are several items you should catalog about your data sources, data types, schemas, formats, requirements, and systems before you finalize infrastructure decisions.

A series of questions should be answered. These are a few basic questions.

1. Is this pipeline one that requires Exactly Once semantics?

2. What effects would duplicate data have on your pipeline?
3. What are the scale in events per second, gigabytes per second, and total storage and completion requirements?
4. How many infrastructure resources do you have?
5. What is the sacrifice for speed vs. cost?
6. How much total storage per day?
7. How long do you wish to store your data stream?
8. Does this need to be repeatable?
9. Where will this run? Will it need to run in different locations, countries, availability zones, on-premise, cloud, K8, Edge, ..
10. What does your data look like? Prepare data types, schemas, and everything you can about the source and final data. Is your data binary, image, video, audio, documents, unstructured, semi-structured, structured, normalized relational data, etc..…
11. What are the upstream and downstream systems?
12. Does NiFi, Pulsar, Flink, Spark, and other systems have native connectors or drivers for your system?
13. Is this data localized, and does it require translation for formatting or language?
14. What type of enrichment is required?
15. Do you require strict auditing, lineage, provenance, and data quality?
16. Who is using this data and how?
17. What team is involved? Data Scientists? Data Engineers? Data Analysts? Programmers? Citizen Streaming Developers?
18. Is this batch-oriented?
19. How long will this pipeline live?
20. Is this time series data?
21. Is Machine Learning or Deep Learning part of the flow or final usage?

References:

https://dev.to/tspannhw/did-the-user-really-ask-for-exactly-once-fault-tolerance-3fek
https://www.datainmotion.dev/2021/01/migrating-from-apache-storm-to-apache.html
https://thenewstack.io/pulsar-nifi-better-together-for-messaging-streaming/

How to Get Started with NiFi + Pulsar + Flink (FLiPN)

Getting started with N+P+F is a simple process. Follow the step-by-step instructions below to configure Apache Pulsar to build full FLiPN applications with Apache Pulsar, Apache NiFi, and Apache Flink.

The easiest way is to use Docker, you will need at least 16 GB of RAM for things to run smoothly.

> **Note**:
> If you need to change ports, you can update the **docker-compose.yml** file. If you are running on a Mac M1 or M2, you probably want to set the platform for Docker to **platform "linux/x86_64".**

Follow the step-by-step instructions below to configure Apache Pulsar to build complete FLiPN applications with Apache NiFi and Apache Flink:

1. You must have Docker and Docker Compose
2. You must have Git installed.
3. Docker must be running.
4. Clone this repository locally https://github.com/tspannhw/create-nifi-pulsar-flink-apps
5. Download to **nifi** directory in that new folder.
    a. Option 1: Download and build your own connector with maven and Java 11+. https://github.com/streamnative/pulsar-nifi-bundle
    b. Option 2: Download https://search.maven.org/remotecontent?filepath=io/streamnative/connectors/nifi-pulsar-nar/1.18.0/nifi-pulsar-nar-1.18.0.nar and https://search.maven.org/remotecontent?filepath=io/streamnative/connectors/nifi-pulsar-client-service-nar/1.18.0/nifi-pulsar-client-service-nar-1.18.0.nar
6. Run the Pulsar and NiFi clusters in docker via **docker-compose up**
7. Run the Flink SQL cluster in docker via:
    sudo docker run --rm -it --platform "linux/x86_64" --volume flink:/flink --name "flink" streamnative/pulsar-flink:1.15.1.4 /bin/bash
8. You will be connected to the Flink cluster and in a command shell.
9. Run ./bin/start-cluster.sh
10. Run ./bin/sql-client.sh
11. You will now be in the Flink SQL Client.
12. You can now connect Flink to Pulsar by creating a catalog.

```
CREATE CATALOG pulsar
  WITH (
    'type' = 'pulsar-catalog',
    'catalog-admin-url' = 'http://<Your PC Name>:8080',
    'catalog-service-url' = 'pulsar://<Your PC Name>:6650'
  );
```

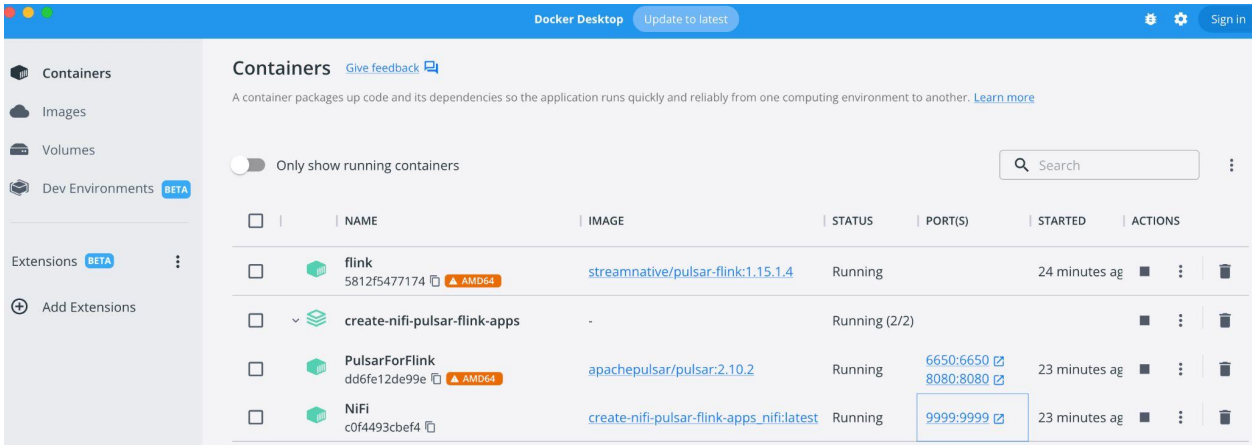
13. Now you will use that catalog.
```
USE CATALOG pulsar;
```
14. We will create a database for our topic.
```
CREATE DATABASE sql_examples;
```

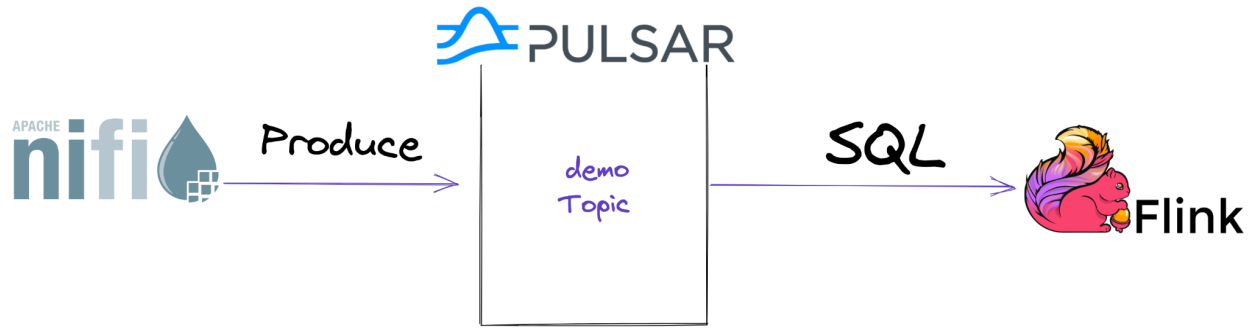
15. Then use that database to create a table for our topic.
```
USE sql_examples;

CREATE TABLE citibikenyc (
      num_docks_disabled DOUBLE,
      eightd_has_available_keys STRING,
      station_status STRING,
      last_reported DOUBLE,
      is_installed DOUBLE,
      num_ebikes_available DOUBLE,
      num_bikes_available DOUBLE,
      station_id DOUBLE,
      is_renting DOUBLE,
      is_returning DOUBLE,
      num_docks_available DOUBLE,
      num_bikes_disabled DOUBLE,
      legacy_id DOUBLE,
      valet STRING,
      eightd_active_station_services STRING,
      ts DOUBLE,
      uuid STRING
) WITH (
  'connector' = 'pulsar',
  'topics' = 'persistent://public/default/citibikenyc',
  'format' = 'json'
);
```

SQL Command	Purpose
`SHOW TABLES`	Show tables in the current database
`USE CATALOG pulsar`	Use the **pulsar** catalog
`SHOW CURRENT DATABASE`	Display the current database
`SHOW DATABASES`	Display all the databases
`CREATE TABLE citibikenyc …`	Create a table in the current database

`DESC citibikenyc`	Describe the columns of a topic
`show create table citibikenyc`	Display the DDL to recreate a table
`select * from citibikenyc`	Query the topic for current data and all events that arrive.
`CREATE CATALOG pulsar ..`	Create the catalog to connect to pulsar.

Congratulations! After completing the steps above, you've configured Pulsar to work with Apache NiFi and Apache Flink. Now we'll discuss building a quick application. It's so easy; my cat, Sploot, can write one.

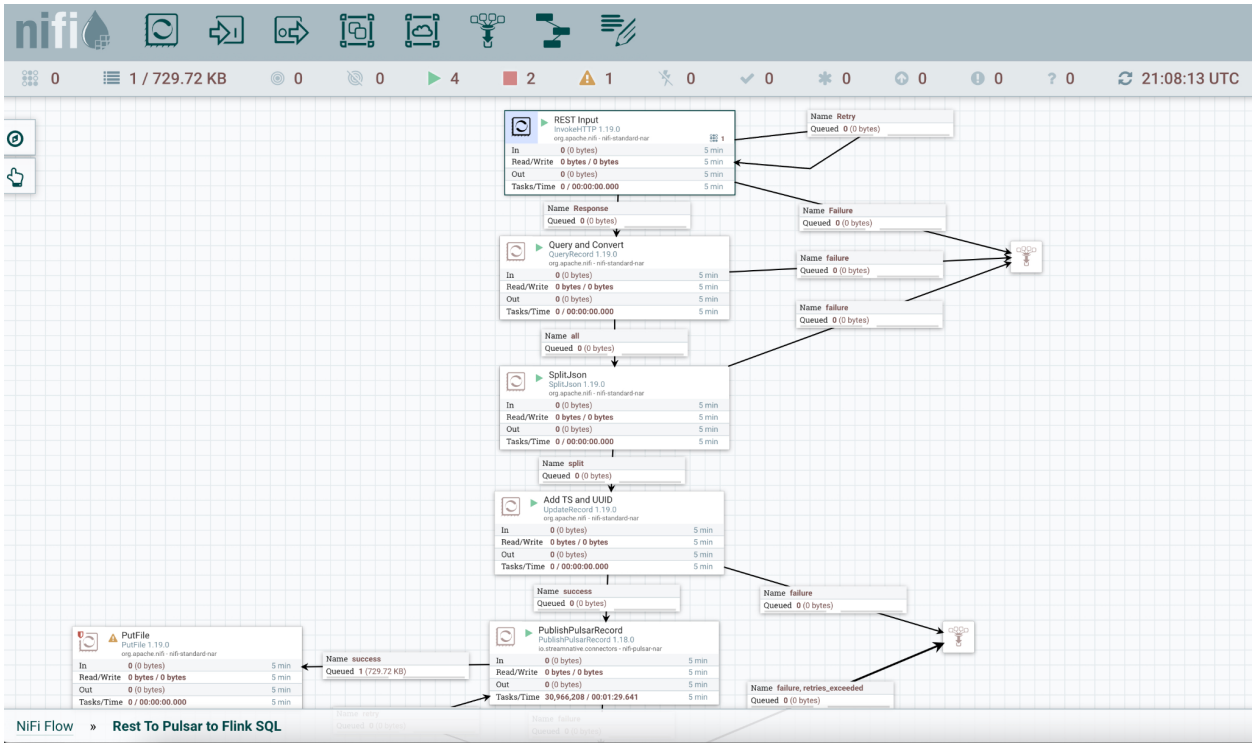


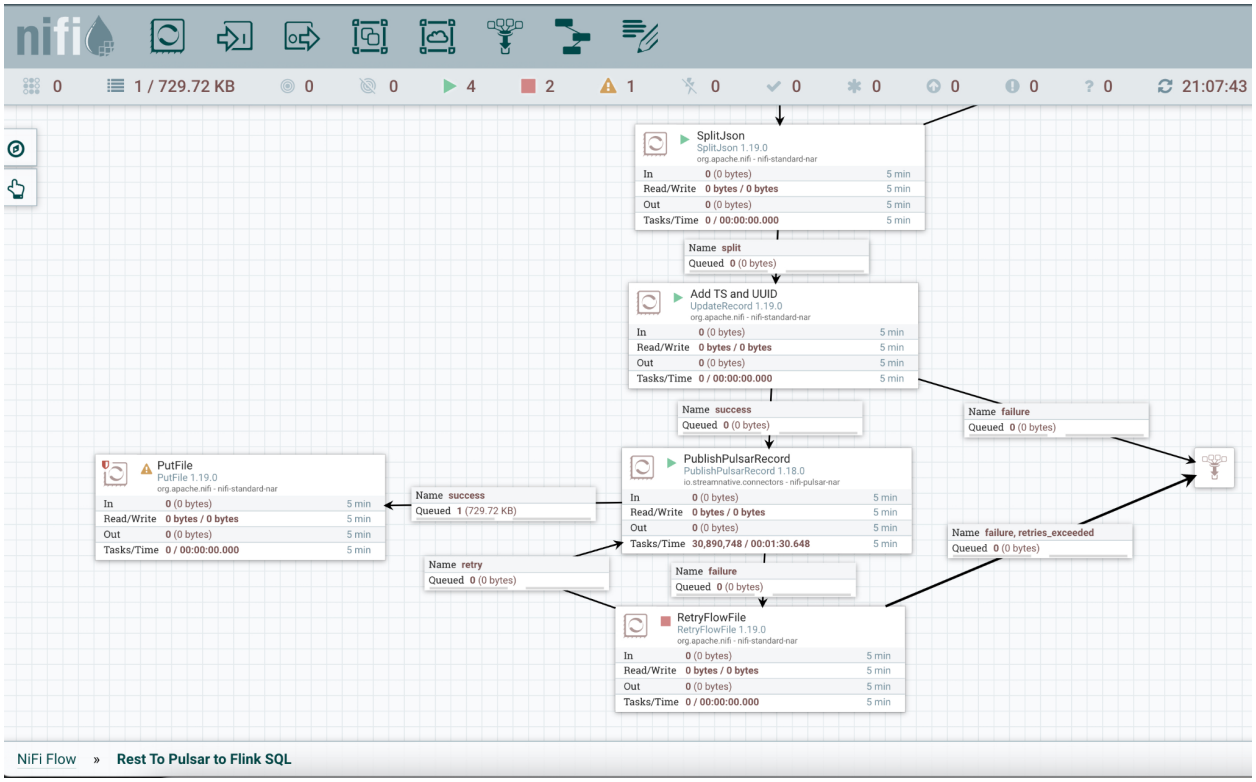


Building a FLiPN App The Easy Way

We created our topic and table so that the hard part is completed. In a production system, the catalog would automatically contain all the data from our topics defined with schemas.

1. From a browser, navigate to your local NiFi as http://localhost:9999/nifi/
2. You can build an app by dragging and dropping NiFi controls to your NiFi canvas. Or you can load one from a JSON file or a NiFi registry. A finished example NiFi flow is in the github directory **nifi**. You can upload it from there by adding a Processor Group and uploading it.
3. To run a flow, highlight the first processor and click Run.
4. Once data flows to Pulsar, you can start a SQL query in the Flink SQL client.
5. You have now built a NiFi-Pulsar-Flink app.
6. See an example of a more advanced one that includes a Pulsar function; see https://github.com/tspannhw/pulsar-transit-function

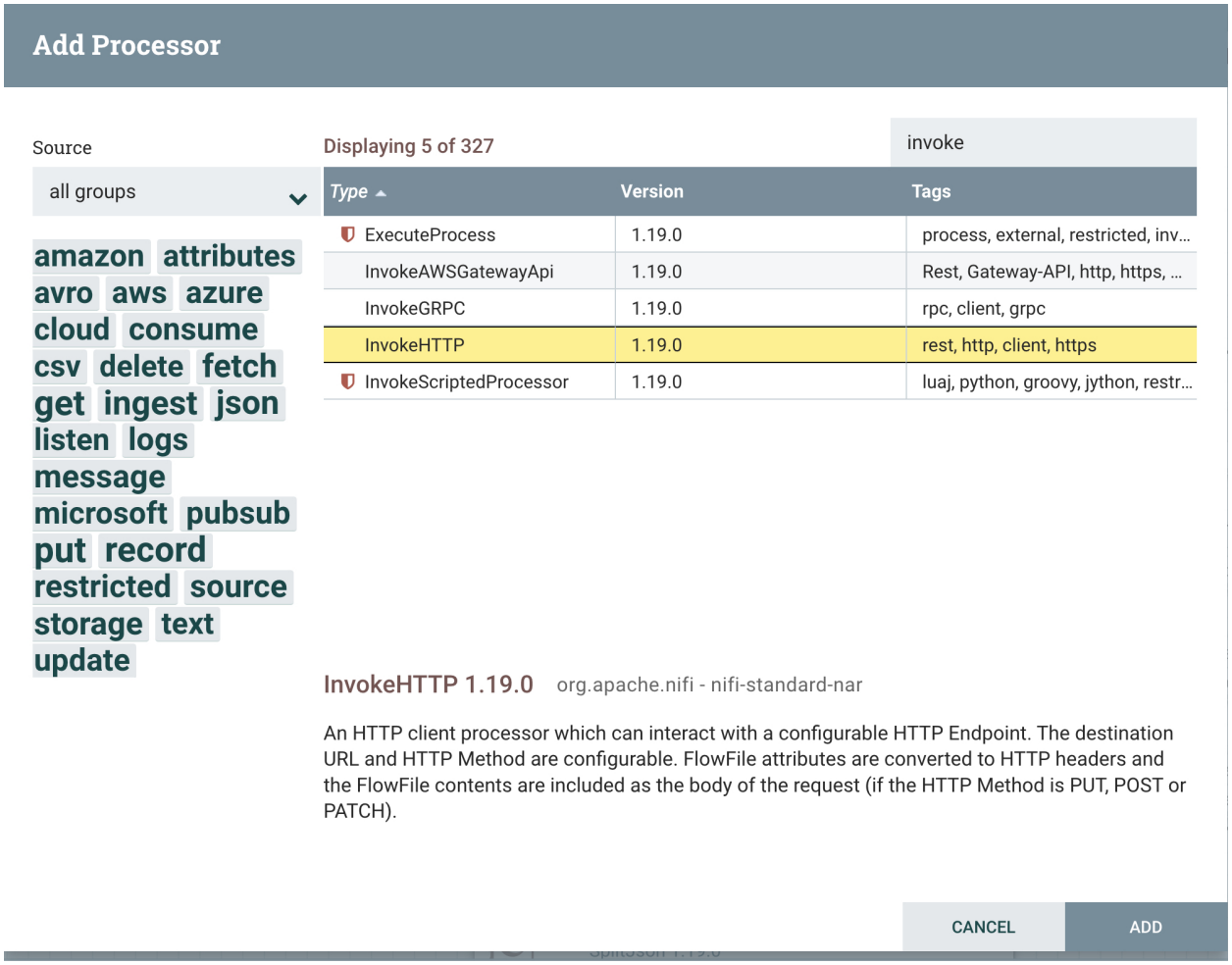




Build an Apache NiFi flow:

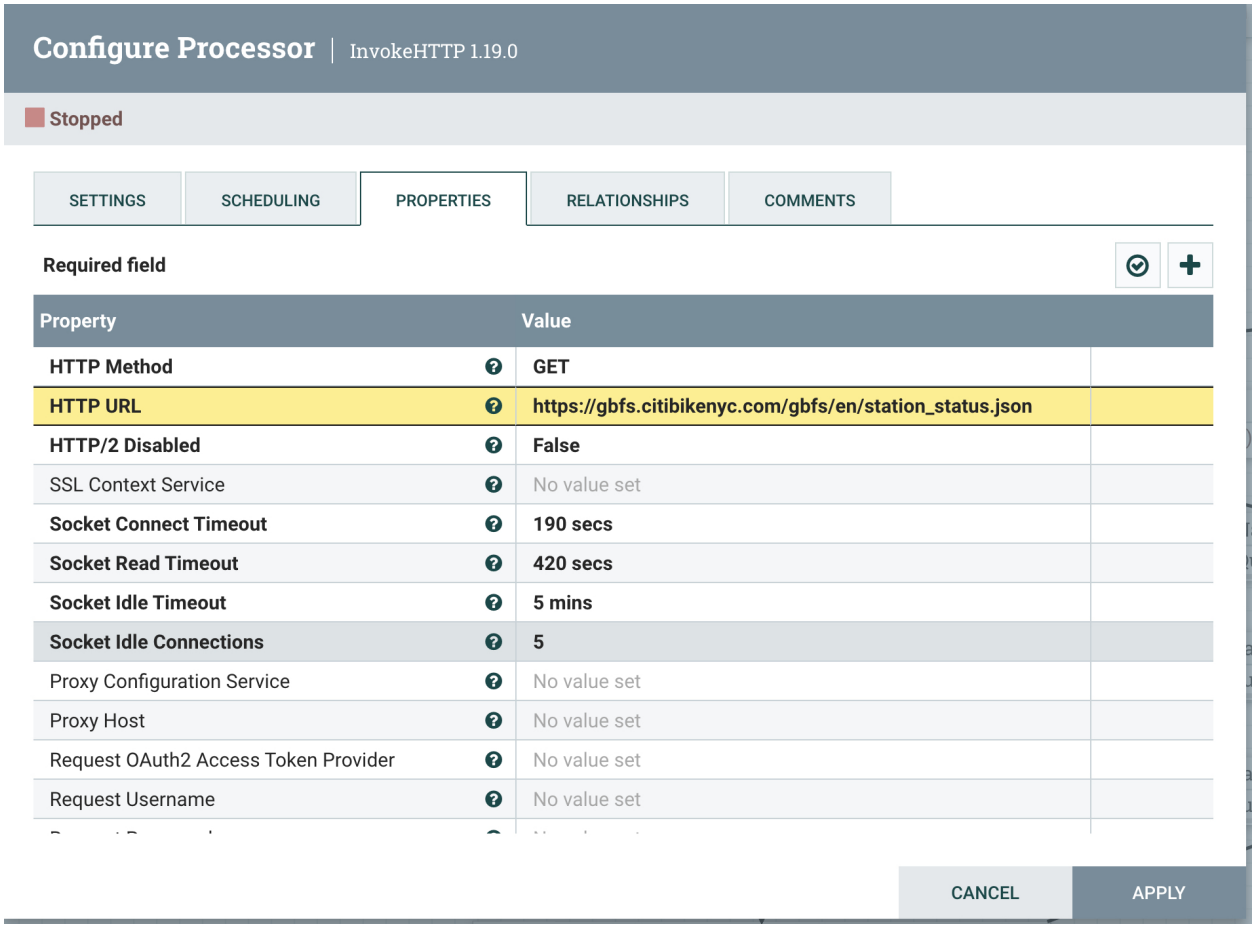
Add processors to the canvas.

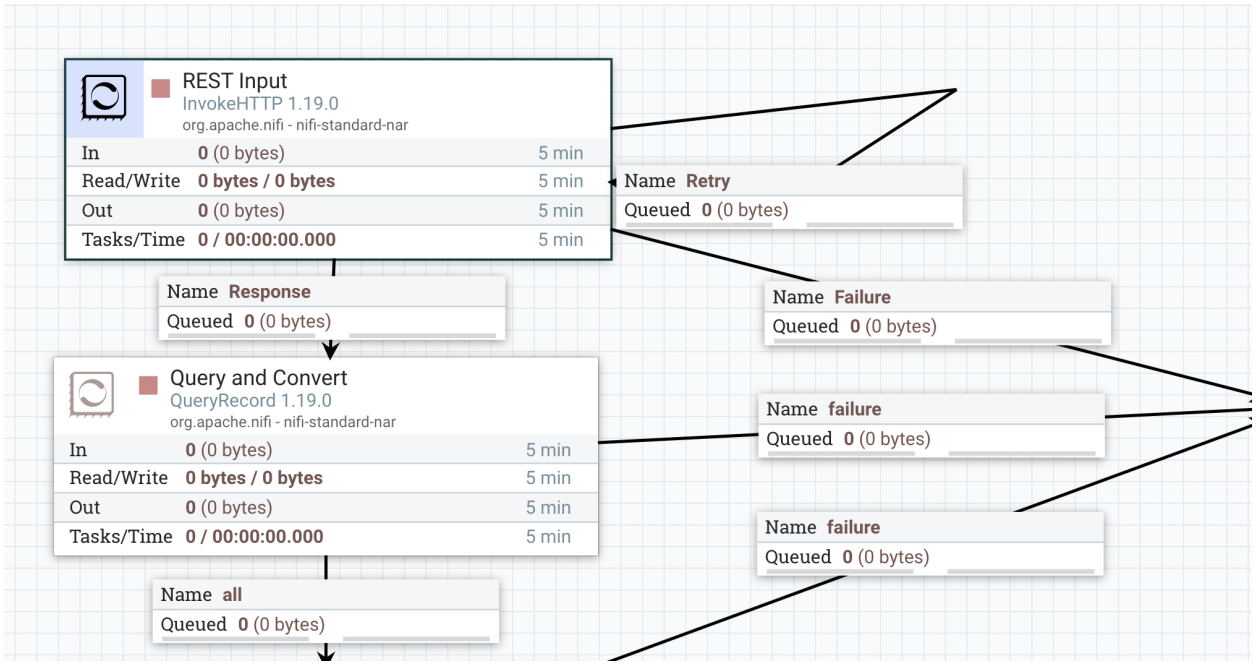
Then pick the ones you need, following our suggestions in the following steps.



InvokeHTTP

HTTP URL: https://gbfs.citibikenyc.com/gbfs/en/station_status.json
Schedule for every 10 minutes. We will run this once for a demo.

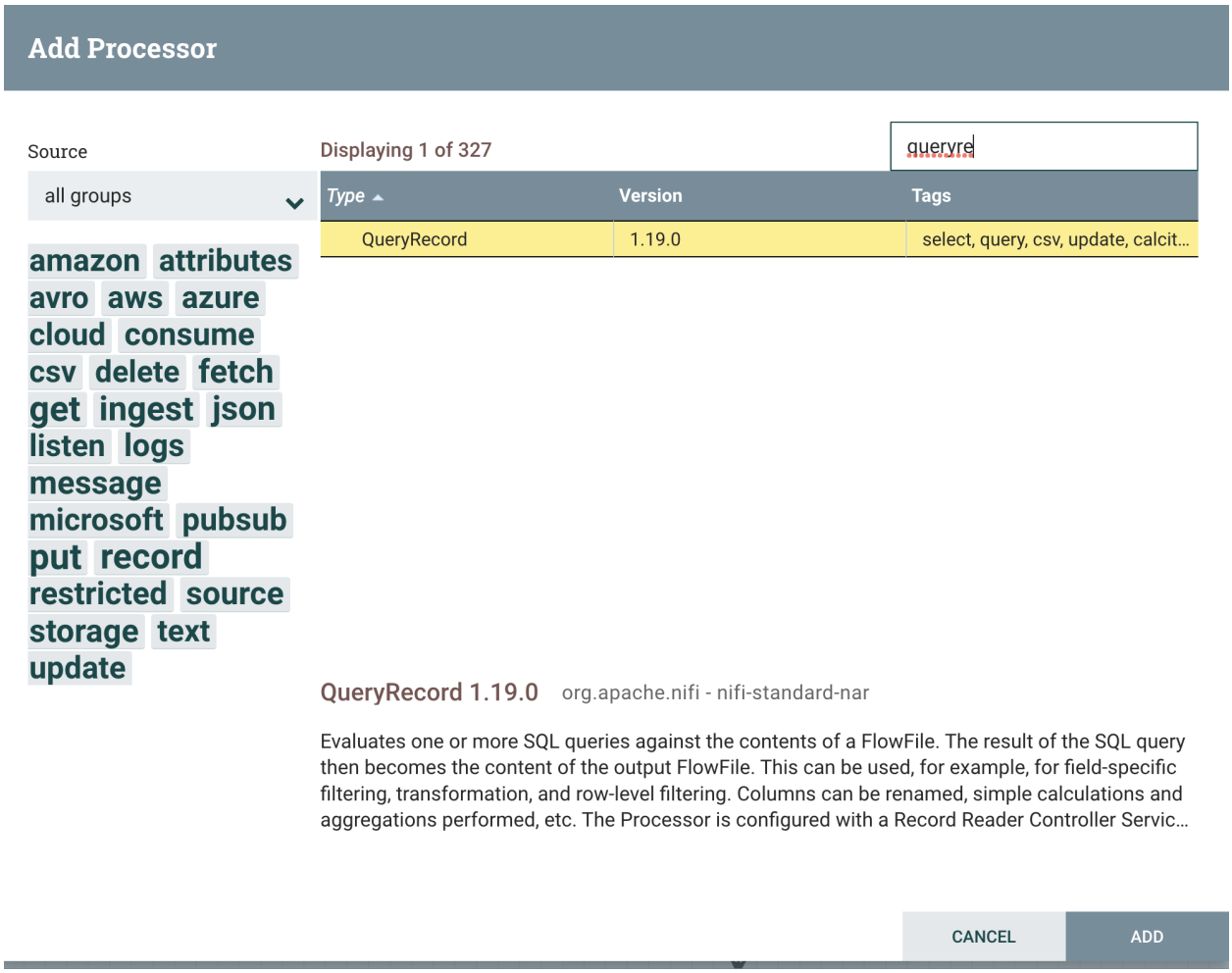


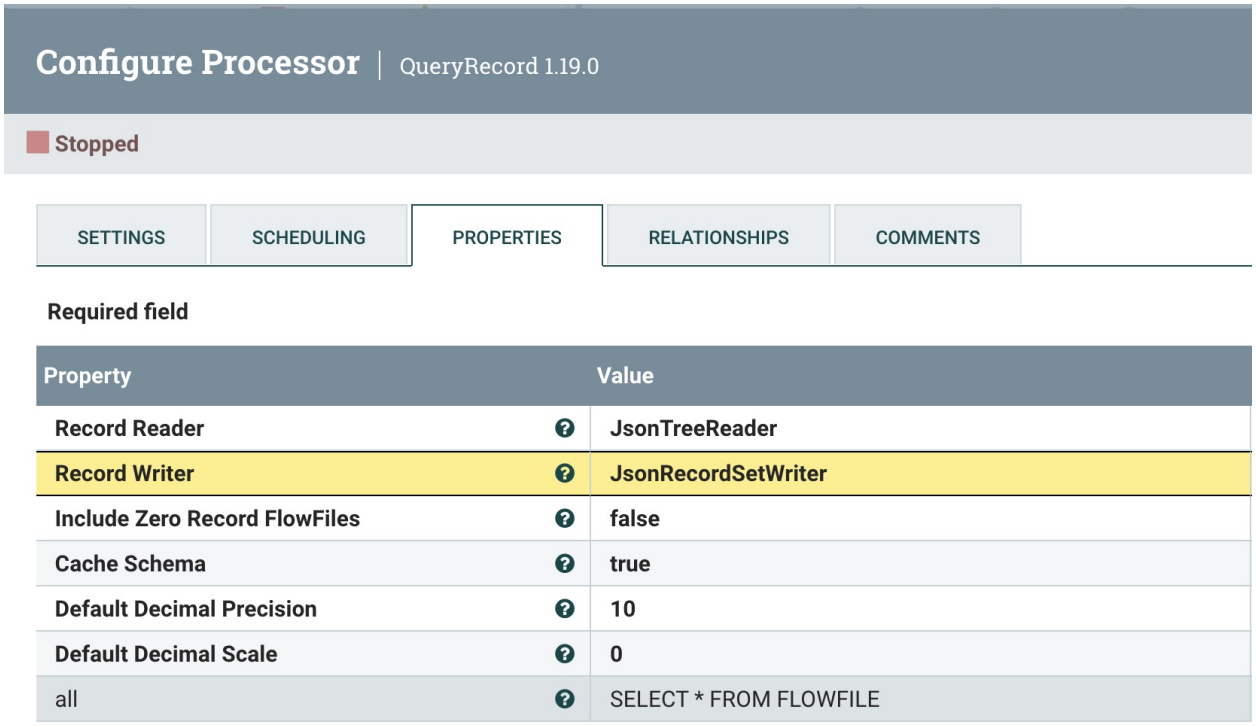


QueryRecord

Create a new **JsonTreeReader** with defaults.

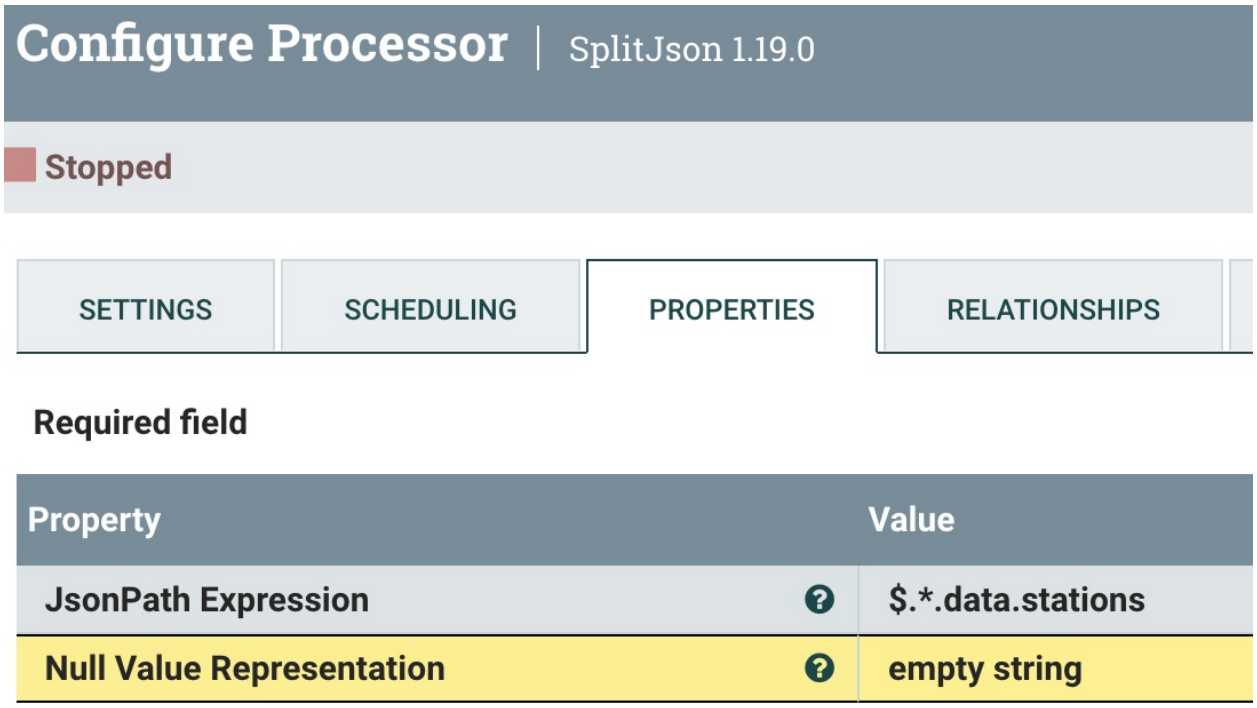
Create a new **JsonRecordSetWriter** with defaults.





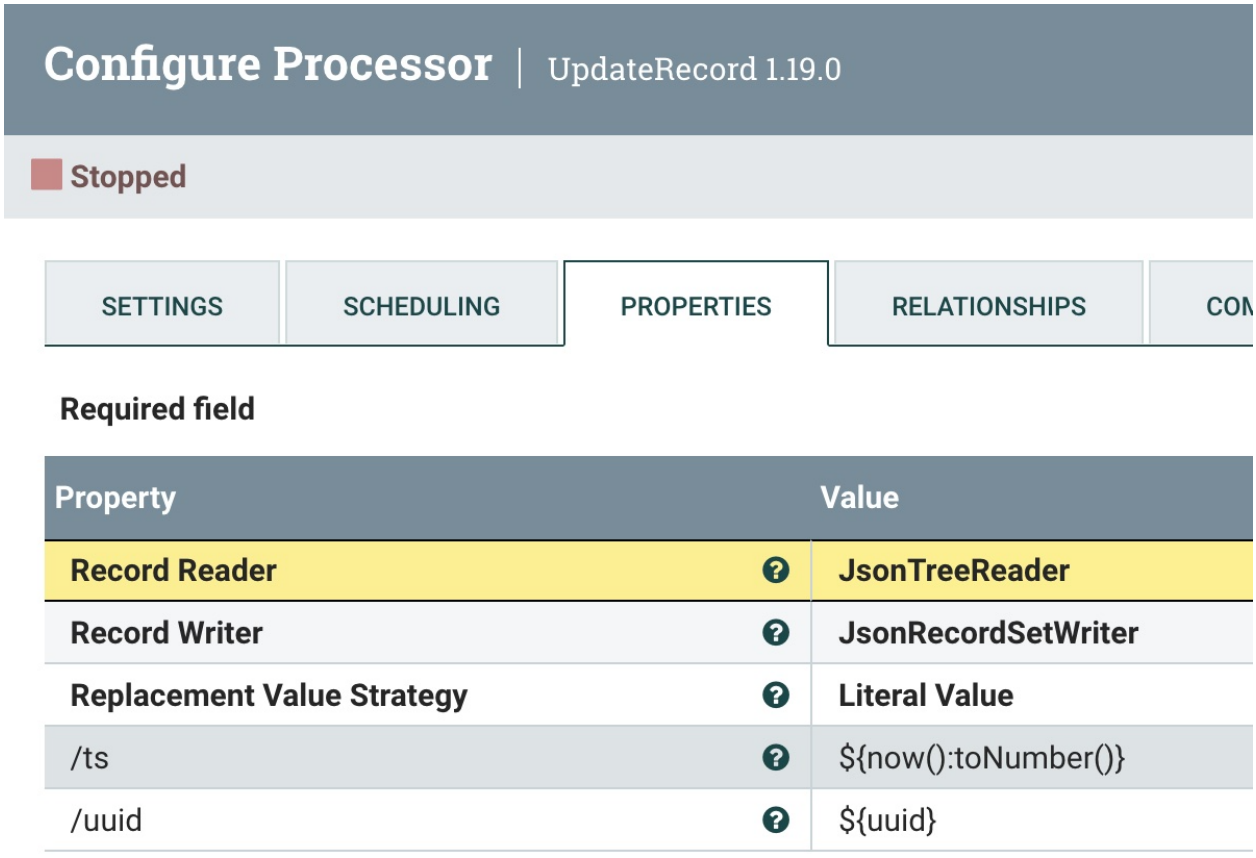
SplitJson

$.*.data.stations



UpdateRecord

/ts ${now():toNumber()}
/uuid ${uuid}



Create a Pulsar Controller.

Controller Service Details | StandardPulsarClientService 1.18.0

SETTINGS | PROPERTIES | COMMENTS

Required field

Property	Value
Pulsar Service URL	**pulsar://Timothys-MBP:6650**
Pulsar Client Authentication Service	No value set
Maximum concurrent lookup-requests	5000
Maximum connects per Pulsar broker	1
I/O Threads	1
Keep Alive interval	30 sec
Listener Threads	1
Maximum lookup requests	50000
Maximum rejected requests per connection	50
Operation Timeout	30 sec
Stats interval	60 sec
Allow TLS Insecure Connection	false
Enable TLS Hostname Verification	false
Use TCP no-delay flag	**false**

The **Pulsar Service URL** is pulsar://<YourPCName>:6650.

PublishPublicRecord

Configure the Pulsar Producer.

Processor Details

SETTINGS | SCHEDULING | PROPERTIES | RELATIONSHIPS | COMMENTS

Required field

Property	Value
Record Reader	**JsonTreeReader**
Record Writer	**JsonRecordSetWriter**
Pulsar Client Service	**StandardPulsarClientServiceDocker**
Topic Name	**persistent://public/default/citibikenyc**
Async Enabled	**false**
Maximum Async Requests	50
Batching Enabled	**false**
Batching Max Messages	1000
Batch Interval	10 ms
Block if Message Queue Full	**false**
Compression Type	**None**
Message Routing Mode	**Round Robin Partition**
Message Demarcator	No value set
Max Pending Messages	1000

The **Topic Name** is ***persistent://public/default/citibikenyc***.

Use the **Pulsar Client Service** that you created before.

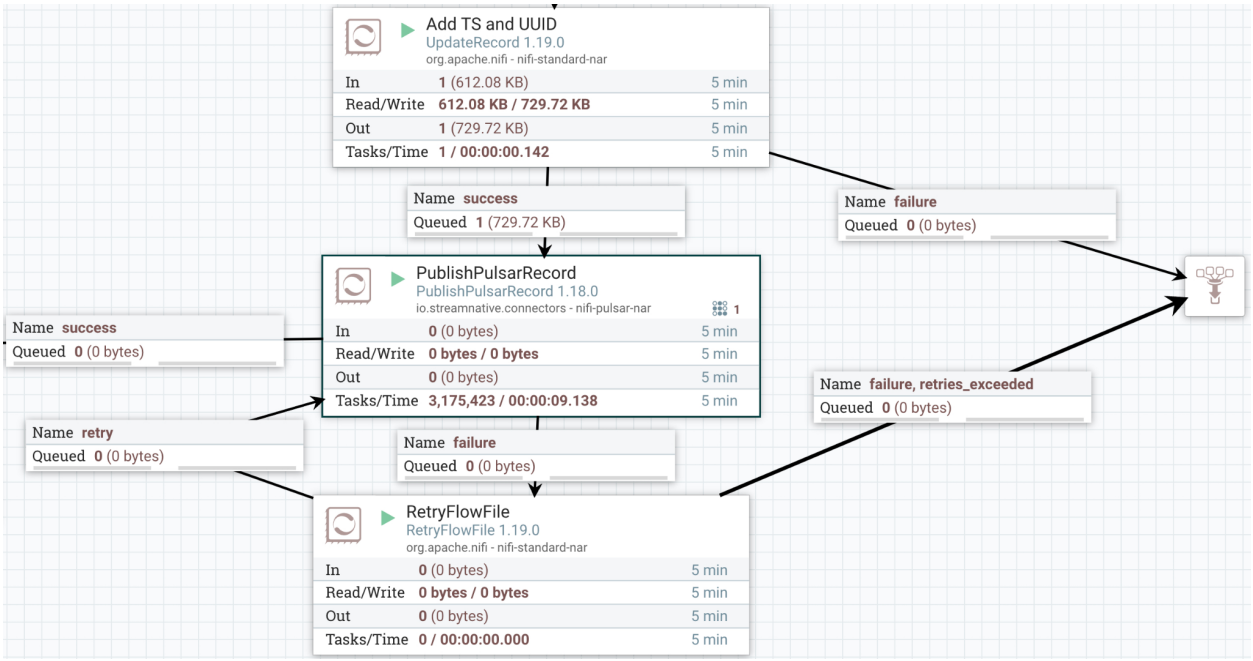
Use existing **JsonTreeReader**.

Use existing **JsonRecordSetWriter**.

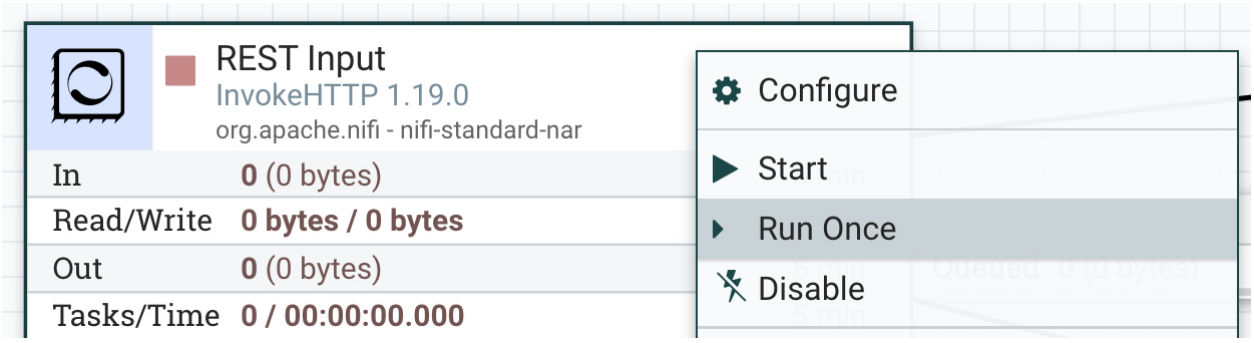
RetryFlowFile

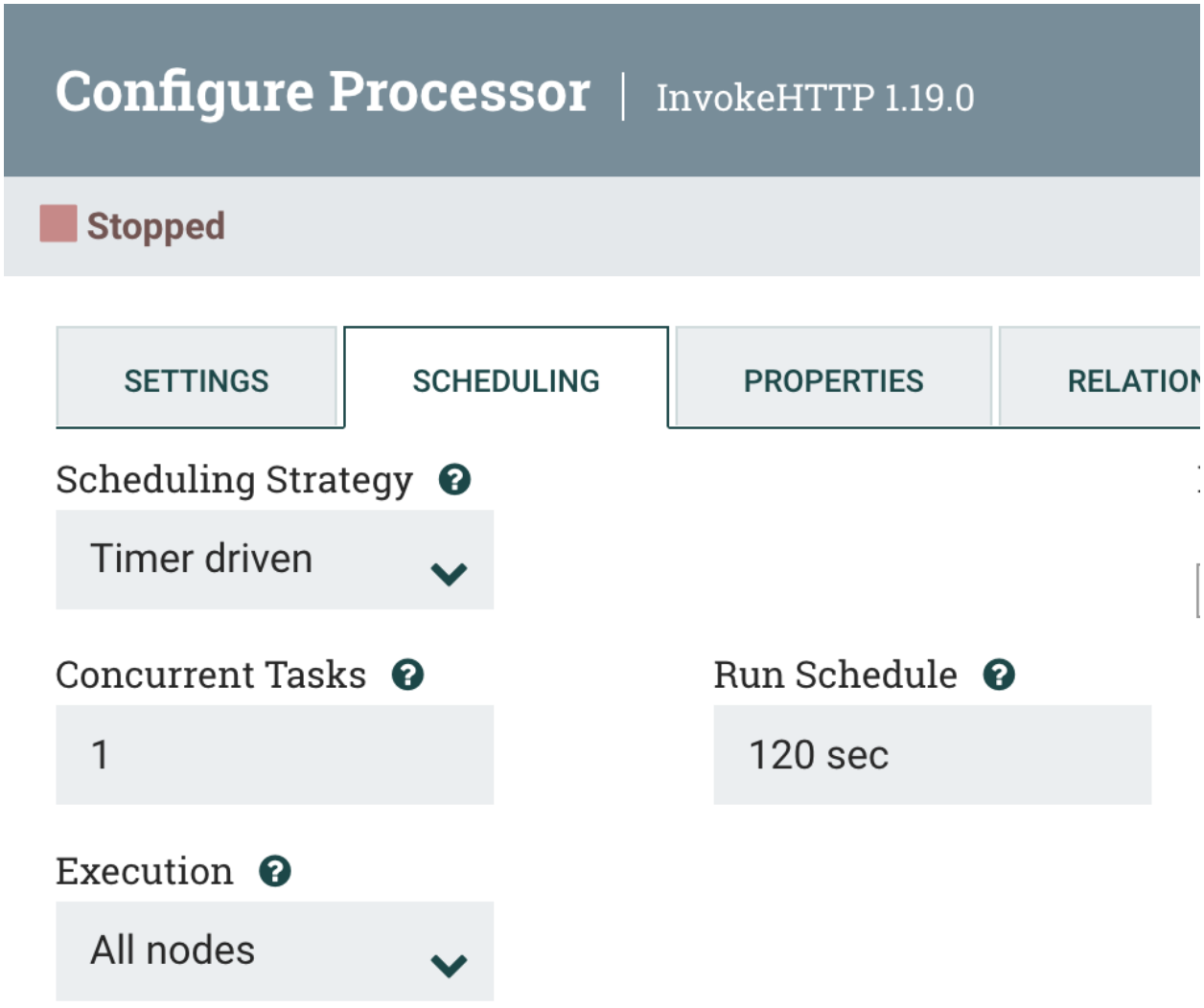
We use the default settings to retry if we fail.

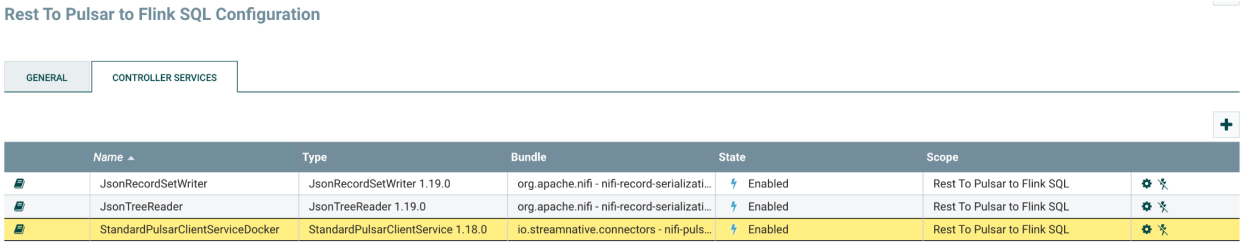
You are now ready to run and test your application.



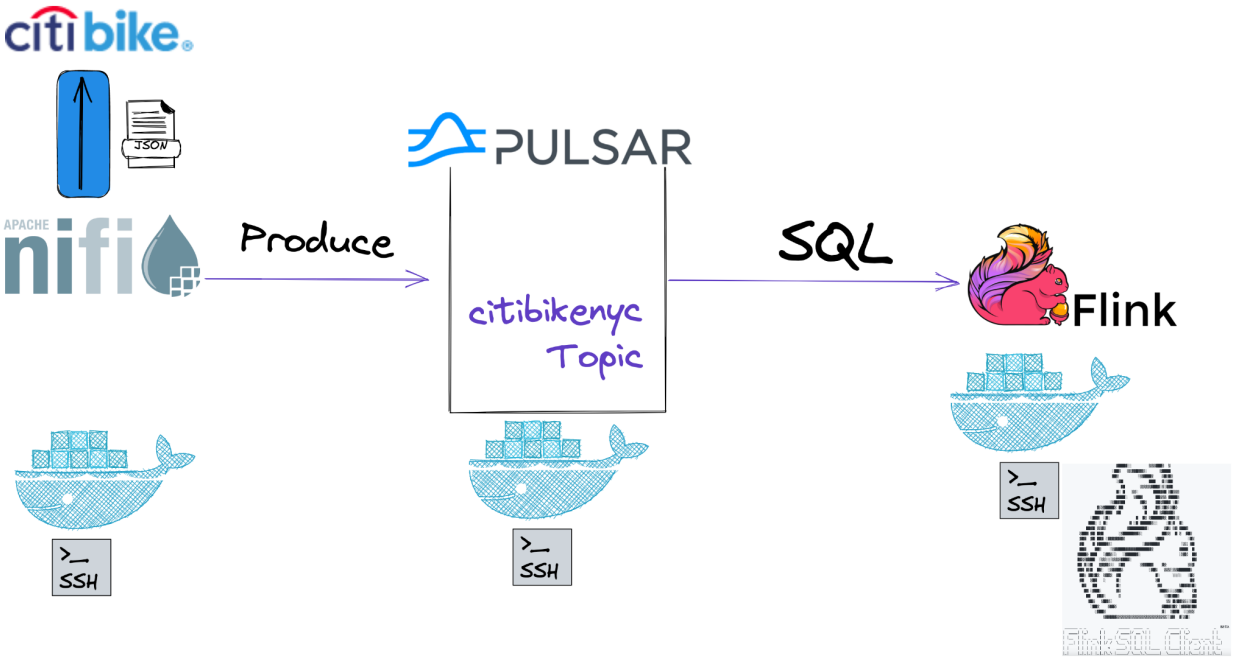
Or you can schedule it to run.







This has been tested on a MacBookPro with M1 Processor (ARM).



Resources

- https://thenewstack.io/pulsar-nifi-better-together-for-messaging-streaming/
- https://www.youtube.com/watch?v=zlSbJxrmgh0
- https://medium.com/@tspann/streaming-nft-analytics-with-apache-nifi-and-apache-pulsar-flipn-stack-6c1d944d1087
- https://medium.com/@tspann/lets-monitor-all-the-things-b884b86b166c
- https://streamnative.io/blog/engineering/2022-04-14-what-the-flip-is-the-flip-stack/
- https://www.slideshare.net/bunkertor/designing-eventdriven-applications-with-apache-nifi-apache-flink-apache-spark-and-apache-pulsar

Important

- https://github.com/tspannhw/pulsar-transit-function
- https://github.com/streamnative/flink-example/blob/main/sql-examples/sql-example.md
- https://github.com/tspannhw/FLiP-Current22-LetsMonitorAllTheThings
- https://github.com/streamnative/pulsar-flink-patterns
- https://github.com/tspannhw/Meetup-YourFirstEventDrivenApp
- https://github.com/tspannhw/pulsar-thermal-pinot
- https://github.com/gaoran10/pulsar-docker-compose

Events Dec 2022
https://dev.to/tspannhw/learn-how-to-use-apache-pulsar-apache-flink-and-apache-nifi-16i7

Timothy Spann
why would a developer create a NiFi + Pulsar + Flink application?

A developer may create a NiFi + Pulsar + Flink application to build a scalable and performant data pipeline for real-time data processing and analysis. Apache NiFi is a powerful tool for data ingestion and routing, Apache Pulsar is a high-performance distributed messaging system, and Apache Flink is a powerful stream processing engine. Together, these technologies can be used to build a robust data pipeline that can process and analyze data in real time.

What are the benefits of using Apache Pulsar with Apache NiFi and Apache Flink over another approach?

There are several benefits to using Apache Pulsar with Apache NiFi and Apache Flink in a data pipeline. First, Apache Pulsar provides a scalable and performant messaging system for data ingestion and routing. It can handle large volumes of data and process it quickly, making it well suited for use in a real-time data pipeline.

Second, Apache NiFi is a powerful tool for data flow management and routing. It allows developers to easily design, implement, and monitor data pipelines, making it easier to build and maintain complex data processing systems.

Third, Apache Flink is a powerful stream processing engine that can be used to perform real-time analysis and processing of data streams. It offers high-throughput and low-latency processing, making it well-suited for use in a real-time data pipeline.

Overall, using Apache Pulsar with Apache NiFi and Apache Flink in a data pipeline can provide a scalable, performant, and easy-to-use solution for real-time data processing and analysis.