# From Kafka via Flink CDC Action framework, persisted into Apache Paimon on S3.

IoT Data packaged in JSON format pushed into Kafka Topic, extracted using Flink CDC, using Flink action framework & Stored into Apache Paimon.

(18 March 2025)

**Overview**

In a previous [blog](https://medium.com/@georgelza/from-mongodb-via-flink-cd-action-framework-persisted-into-apache-paimon-on-s3-008976cdbfff) we were inserting our JSON packaged IoT generated data into a MongoDB Collection and then using [Apache Flink’s flink\_paimon\_action](https://paimon.apache.org/docs/master/cdc-ingestion/overview/) framework to CDC source the data and sink (insert) it into a [Apache Paimon](https://paimon.apache.org/) database..

This time round, we’re going to publish the same data onto a Confluent Kafka Topic and then utilize the [Apache Flink’s flink\_paimon\_action](https://paimon.apache.org/docs/master/cdc-ingestion/overview/) framework again to [CDC](https://en.wikipedia.org/wiki/Change_data_capture) source the payload directly out of our Topic and store it directly into [Apache Parquet](https://parquet.apache.org/) files stored in our [Apache Paimon](https://paimon.apache.org/) based table, written onto S3 Object storage provided via a [MinIO](https://min.io/) server.

This will be a very real-world type of IoT flow that you will run across.

Our data is again generated via a Python IoT program (located in *app\_iot1/* and now containerized) created in the previous blog.

* The first *app\_iot1* creates our simplest IoT JSON payload.
  + This is accomplished by setting *TSHUMAN, STRUCMOD & DEVICETYPE* = 0
* In *app\_iot2* we extend the payload to include *TSHUMAN*=1 and *STRUCMOD*=1.
  + This adds a human readable date field and the location object to the metadata tag section.
* In *app\_iot3* we go one step further and add *DEVICETYPE=1* to the payload.
  + This adds a text string defining the device type.

Showing the versatility of JSON for IoT data and the ability along our entire path from source to data store to accommodate the dynamic data structure.

For catalog services we will be using the [Apache Hive](https://hive.apache.org/)’s and their [Metastore](https://cwiki.apache.org/confluence/display/hive/design#Design-Metastore) functionality as created in a previous blogs (but with a little version update applied recently).

For those that have been following my previous blogs, you will notice I’ve upgraded my [Confluent](https://www.confluent.io/) Kafka Cluster (now 7.7.1) and the [Apache Flink](https://flink.apache.org/) environment (now 1.19.1). The [Apache Paimon](https://paimon.apache.org/) stack has also been upgraded to 0.9.0

As always, all the code can be found in the [GIT repository](https://github.com/georgelza/DataPipeline-KakfaPaimon.git), and yes, we’re still using a substantial amount of Makefiles, Docker-compose.yml and Dockerfile’s.

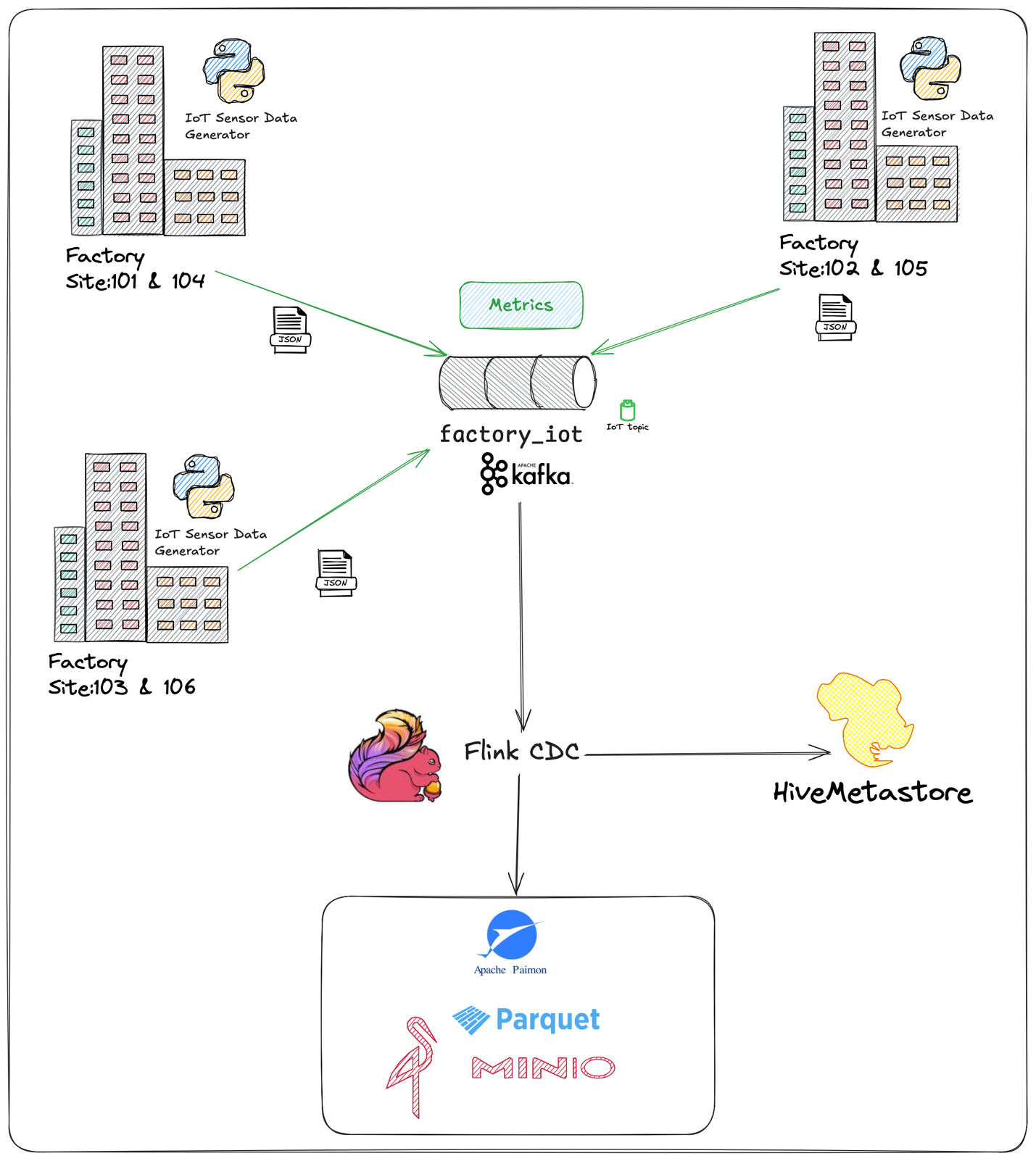
I’ve decided not to digress as usual and do allot of other “things” like aggregation. We’ve demonstrated that a couple of times now.

Ok, back to why we’re here.

We will use our Python application to create a IoT data stream as if we’re reading sensors in factories fitted to machinery. These “[JSON](https://www.json.org/)” documents will be pushed onto our Kafka Topic called *factory\_iot*. The topic “*key”* is set to the *siteId* of the various factories.

I grouped the factories into 3 sets, North, South and East. (simulating a sort of regional distribution).

We will then use the [Kafka CDC capability via the action frame](https://paimon.apache.org/docs/0.9/flink/cdc-ingestion/kafka-cdc/) to replicate this directly into an Apache Paimon Table.



Once the scaffolding is up, this is it, and this is all of it. Of course you can make it allot more complicated.

First start by running *app\_iot.*

This will start publishing our very simply payload as below onto *factory\_iot* topic.

You can run the *app\_iot1* by either going into the *app\_iot1* directory and executing *run1.bsh* or you can execute *make* *iot1\_datagen w*hile located in the *devlab* directory.

(if you execute the app by running the bash script method you can run *app\_iot2* & *app\_iot3* similarly by running *run2.bsh* & *run3.bsh* from within *app\_iot1* directory), otherwise you can run *app\_iot2* and *app\_iot3* by executing *make* *iot1\_datagen* and *make* *iot1\_datagen.*

{

"ts" : 123421452622,

"metadata" : {

"siteId" : 1009,

"deviceId" : 1042,

"sensorId" : 10180,

"unit" : "Psi"

},

"measurement" : 1013.3997

}

Once it is running then execute the below by executing *make jm* from within the *devlab* directory which will open up the Flink job manager and put you at the command line.

You can simply copy/paste the below and hit enter.

This will start our “subscribing” process on our source topic (*factory\_iot)* and pushing the data into our target Apache Paimon database/table.

NOTE: Make sure to first start the data generator as per above so that when the below command is executed there is some shape of a payload with a structure to infer a schema from. If we don’t do it this way then we would first have had to create the table structure manually.

/o*pt/flink/bin/flink run \*

*/opt/flink/lib/paimon/paimon-flink-action-0.9.0.jar \*

*kafka\_sync\_table \*

*-Dexecution.checkpointing.interval=10s \*

*-Dexecution.checkpointing.num-retained=5 \*

*-Dstate.checkpoints.num-retained=10 \*

*-Dpipeline.name='sync-kafka-topic-to-paimon-s3' \*

*--kafka\_conf properties.bootstrap.servers=broker:29092 \*

*--kafka\_conf topic=factory\_iot \*

*--kafka\_conf properties.group.id=123456 \*

*--kafka\_conf value.format=json \*

*--kafka\_conf scan.startup.mode=earliest-offset \*

*--catalog\_conf metastore=hive \*

*--catalog\_conf uri=thrift://metastore:9083 \*

*--warehouse s3a://warehouse/paimon/ \*

*--database iot \*

*--table factory\_iot \*

*--table\_conf sink.parallelism=4*

*NOTE: even though the documentation does not call for scan.startup.mode=earliest-offset please do include it, even when sourcing a new topic.*

Ok, so this is funky. And simple. Let’s test schema evolution.

Execute the *app\_iot2*. This as you would have seen above adds the *TSHUMAN* and *LOCATION* object to the payload.

{

"ts" : 123421452622,

"metadata" : {

"siteId" : 1009,

"deviceId" : 1042,

"sensorId" : 10180,

"unit" : "Psi",

"ts\_human" : "2024-10-02T00:00:00.869Z",

"location": {

"latitude": -26.195246,

"longitude": 28.034088

}

},

"measurement" : 1013.3997

}

And lastly as we’re feeling very lucky, run *app\_iot3* which will add the *DEVICETYPE* field to the metadata tag.

{

"timestamp" : "2024-10-02T00:00:00.869Z",

"metadata" : {

"siteId" : 1009,

"deviceId" : 1042,

"sensorId" : 10180,

"unit" : "Psi",

"ts\_human" : "2024-10-02T00:00:00.869Z",

"location": {

"latitude": -26.195246,

"longitude": 28.034088

},

"deviceType" : "Oil Pump",

},

"measurement" : 1013.3997

}

And that’s it… a very short blog compared to previous ;).

So, in summary, we build a data pipeline from Kafka Topic, that can handle schema evolution, have that data flow into our Apache Paimon data store stored as Apache Parquet files.

I think this is pretty neat… Hope you enjoyed the exploration.

Good luck, this is all fraught with rabbit holes, as always, so many and you can disappear so easily… but then that’s ½ the fun.



*Note: to execute this blog start with README.md located in the root folder and work from there, it will tell you exactly what to execute in which order to download all the dependencies and build everything.*

**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

<https://www.linkedin.com/in/george-leonard-945b502/>

<https://medium.com/@georgelza>