# A Proposal to utilize Modern Realtime Data Streaming Technologies to Manage Data Centre Resources.

A SNMP based stream of data (OIDs) sourced using Apache Flink SNMP Source connector, stored into Apache Fluss.

(7 July 2025)

# Overview

In a previous [blog](https://medium.com/@georgelza/building-a-factory-monitoring-iot-pipeline-8880a7e34f70) we explored various IoT data streams, most all based on industrial use cases.

Reality is, IoT is also very valuable in the IT world, not at first obvious…

How can we re-vitalize an industry standard technology like [SNMP](https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol) (Simple Network Management Protocol), using modern data streaming concepts to improve monitoring, management, proactive maintenance to protect our ever-increasing reliance in larger and larger data center due to technologies like AI, ML, RAG, etc... there are many newer 3 letter acronyms if you go looking.

If anyone has ever looked at SNMP, they would have notice the sheer volume of information available, exposed. It also maps very well to the root concepts coined during the early days [BIG Data](https://en.wikipedia.org/wiki/Big_data) era, namely VVV (Velocity, Volume, Variety).

With this we do sit with the problem that VVV also causes, how do we manage it, how do we consume the data volume, store the volume, expose abnormalities.

To analyze it, well we have all the 3LA (Three Letter Acronyms) as previously, AI, ML, RAG, MCP... But none of these are of any value if we don't have data to work with.

[SNMP](https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol) itself is an old technology, but it was never built for the modern world, for the new volumes we are talking here about, and it has no "concept" of our very popular 3LA demanding/getting all the attention.

And here we sit, lets introduce modern Data streaming technologies... Let’s use these concepts to source our data and get it to our 3LA project of choice.

This is the combination 1, 2 no more like 3 projects.

1. The first was this idea, this blog, but it then required project 2,
2. Write an Apache Flink SNMP source connector, and then project 3,
3. The ability to load SNMP MIB files into a designated table to be accessible via our Apache Flink/Apache Fluss stack. A [SNMP MIB Parser](https://github.com/georgelza/SNMP-MIB-Parser) was written using Python using the PySNMP Python Package.

Well, project 1 is done (you are reading it, hehehe), project 2 is 90% done ([SNMPv3](https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol#Version_3) auth outstanding), project 3... let’s not talk, or well, maybe by the time you read this I might have completed it.

First up, I am not a developer, I'm a technologist (day job is Technology Architect, but for 25+ yrs I've primarily specialized in the data space) with a curious mind that get me into allot of trouble...

Java is not my language, make that never have been, I'm mildly proficient in [Python](https://www.python.org/) and [Golang](https://www.google.com/search?client=safari&rls=en&q=Golang&ie=UTF-8&oe=UTF-8), never written [Java](https://www.java.com/en/) until I started looking at [Apache Flink](https://flink.apache.org/) source connectors...

As such, I had the idea, and between [Google Gemini](https://gemini.google.com/) and myself we had allot of conversations, arguments, restarts. Giving it the right comments, questions you can make it eventually provide you with the correct structured framework. At which point I added the bits I figured I needed.

Eventually I got the [Apache Flink SNMP Source Connector](https://github.com/georgelza/SNMP-Flink-Source-connector.git) completed.

Learned allot and can't say I'm not mildly impressed with what I created.

This Project: [GIT REPO](https://github.com/georgelza/DataPipeline-SNMP_Flink_Fluss.git)

## Regarding Stack:

Regarding our [Apache Fluss](http://fluss.apache.org/) stack. I simply did not configure (run) the lakehouse sink to our deep storage, based [Apache Paimon](https://paimon.apache.org/) open tables format stored on HDFS layer, for these labs as we’ve shown that working in the previous blog.

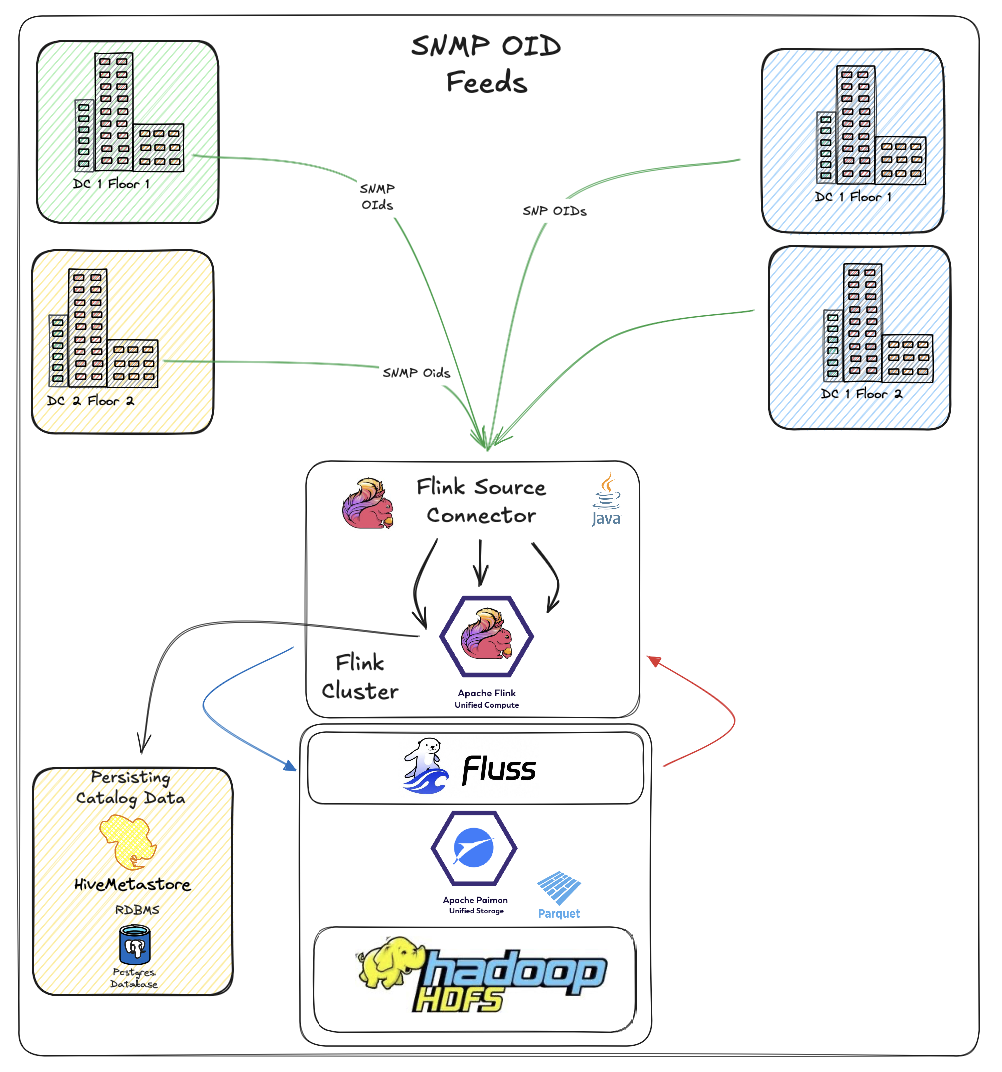
Our data is source via SNMP agent endpoints. I list various SNMP data simulators available in case you don’t have access to hardware that you can probe.

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, the [Apache Flink](https://flink.apache.org/) environment is now based 1.20.1. The [Apache Paimon](https://paimon.apache.org/) stack has also been upgraded to 0.9.0, [Apache Fluss](http://fluss.apache.org/) 0.7.0.

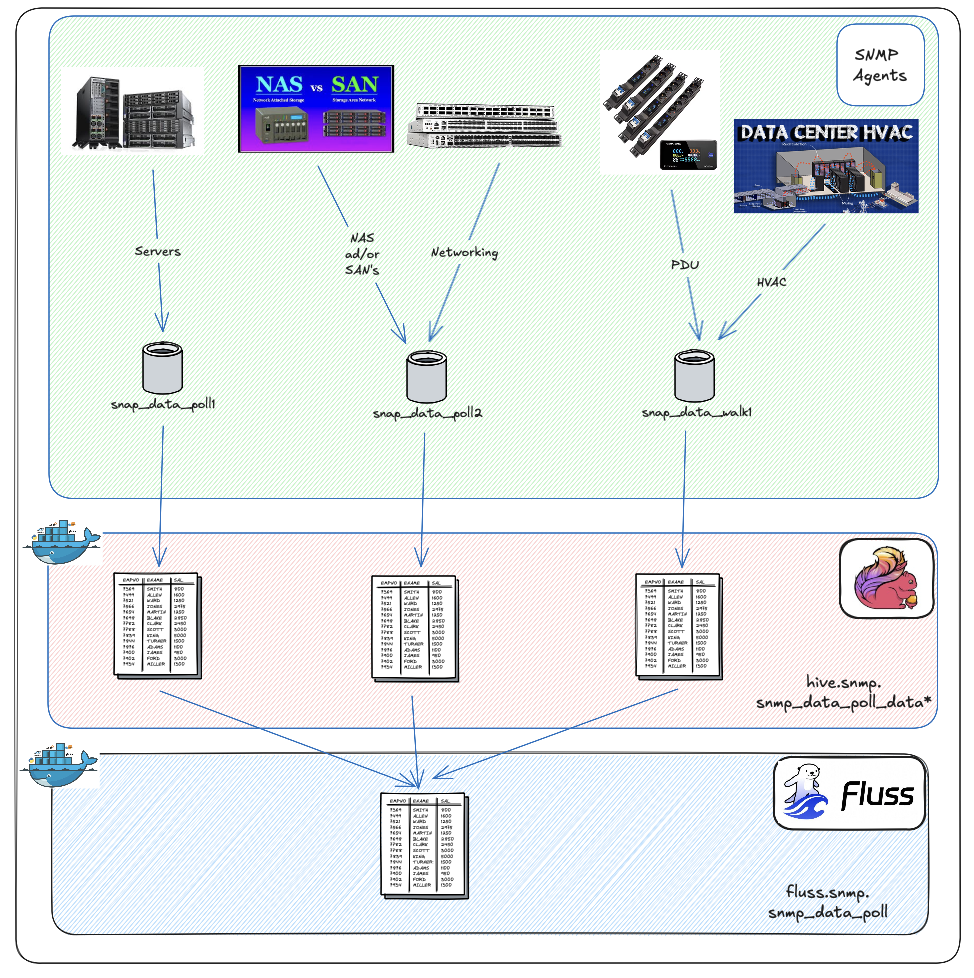
As always, all the code can be found in the [GIT repository](https://github.com/georgelza/DataPipeline-SNMP_Flink_Fluss.git), and yes, we’re still using a substantial amount of *Makefiles*, *Docker-compose.yml* and *Dockerfile’s*

## Simple Overview Diagram



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

To start navigate to the *<root>* folder and read the *README.md* file. This will give a similar overview as per above, that will then direct you to build the basic scaffolding using (<*root>/infrastructure/),* after which you will be directed to *<root>/devlab0/README.md* instructing you how to build and run the various examples.



# Source to SNMP via connector into Apache Flink:

This is above flow is accomplished using the below Flink SQL. You will find there are 3 SQL files, one for various agents to poll using GET method, a 2nd to execute a WALK on and the last is either GET or WALK, but using SNMPv3. This will use our Apache Flink SNMP Source Connector as created in our other blog.

See: <*root>/devlab0/creFlinkFlows/2.1.snmp\_pll\_data\_get.sql*

CREATE TABLE hive.snmp.snmp\_poll\_data2 (

device\_id VARCHAR(255) NOT NULL

,metric\_oid VARCHAR(255) NOT NULL

,metric\_value VARCHAR(1000) NOT NULL

,data\_type VARCHAR(50) NOT NULL

,instance\_identifier VARCHAR(255) NULL

,ts TIMESTAMP(3) NOT NULL

,WATERMARK FOR ts AS ts - INTERVAL '5' SECONDS

,PROC\_TIME AS PROCTIME()

) WITH (

'connector' = 'snmp'

,'target' = '172.16.10.2:161,'172.16.10.3:161'

,'snmp.version' = 'SNMPv1'

,'snmp.community-string' = 'abfr24'

,'snmp.poll\_mode' = 'GET'

,'oids' = '.1.3.6.1.2.1.1.5.0'

,'snmp.interval\_seconds' = '10'

,'snmp.timeout\_seconds' = '5'

,'snmp.retries' = '2'

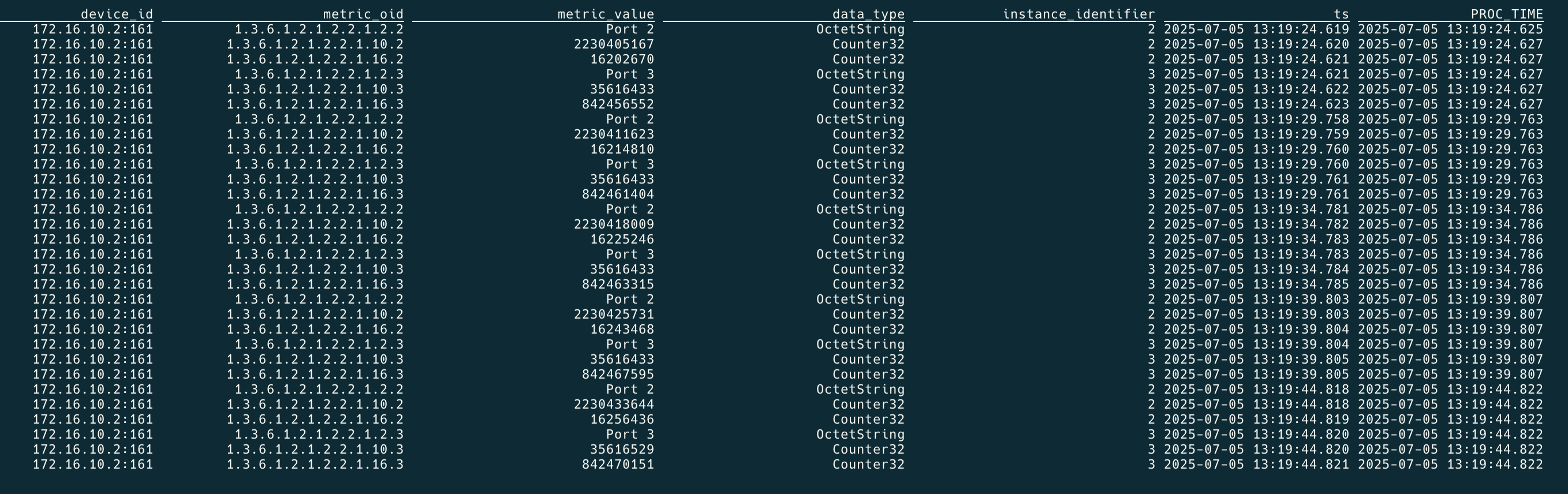
);

NOTE: the above NULL is important. Without it the insert will fail.

At this point we can run the following commands which resulted in results as per screen grabs.

use hive.snmp;

select \* from hive.snmp.snmp\_data\_poll2;



We first need to create our Fluss target table, this is accomplished using the following Flink SQL. For the Apache Fluss tier we have a single table.

See: <*root>/devlab0/creFlinkFlows/3.1.creFlussTarget.sql*

With Apache Fluss 0.7.0 we now have auto partitioning based on or multiple columns.

CREATE OR REPLACE TABLE fluss.snmp.snmp\_poll\_data (

device\_id VARCHAR(255) NOT NULL

,metric\_oid VARCHAR(255) NOT NULL

,metric\_value VARCHAR(1000)

,data\_type VARCHAR(50)

,instance\_identifier VARCHAR(255)

,ts TIMESTAMP(3)

,partition\_month STRING

,WATERMARK FOR ts AS ts - INTERVAL '5' SECONDS

) PARTITIONED BY (partition\_month) WITH (

'bucket.num' = '3'

,'table.datalake.enabled' = 'true'

,'table.auto-partition.time-unit' = 'MONTH'

,'table.auto-partition.num-precreate' = '5'

,'table.auto-partition.num-retention' = '60'

,'table.auto-partition.time-zone' = 'Africa/Johannesburg'

);

We can now push from our step 1 tables into the Fluss table using the following Flink SQL. Again I’m only showing one insert, see the below script for the 3 insert statements.

See: <*root>/devlab0/creFlinkFlows/3.2.creFlussInserts.sql*

INSERT INTO fluss.snmp.snmp\_poll\_data

(device\_id, metric\_oid, metric\_value, data\_type, instance\_identifier, ts, partition\_month)

SELECT

device\_id AS device\_id

,metric\_oid AS metric\_oid

,metric\_value AS metric\_value

,data\_type AS data\_type

,instance\_identifier AS instance\_identifier

,ts AS ts

,DATE\_FORMAT(TO\_TIMESTAMP\_LTZ(ts, 3), 'yyyyMM') AS partition\_month

FROM hive.snmp.snmp\_poll\_data1;

At this point, you have an active data stream which can be enriched and analysed.



# And In Summary.

We now have a SNMP based data stream from our source device referred to as a SNMP Agent, through Apache Flink using a Source connector and up into Apache Fluss for Analytical storage.

Apache Fluss (now an Apache incubator project).

We can further use various Apache Flink capabilities, i.e.: PyFlink to externally call AI, ML, RAG and MCP services.

We’ve also previously showed some “simple’ aggregation, and windowing via Apache Flink SQL and it’s windowing capabilities.

We have further introduced the reader to real time (and long term lakehouse) storage using Apache Fluss as our Open Table format, using industry recognized Apache Parquet as Open Files format, persisted on either HDFS or S3.

As always, I’m predictable, but I really do think this is pretty amazing… Hope you enjoyed the exploration, the journey up to now.

The next idea is brewing already ;)

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. If you have any problems, welcome to reach out to me via one of the below profiles.*

**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>