DELIVERING ANALYTICS AT SCALE

Our Journey to Ingest and Analyse 5 Billion Events / Day

Sharat Madanapalli
Head of Data and AI

ABOUT

*Recent picture but not comprehensive

Me

Started as a founding engineer @ Canopus (along with my PhD @ UNSW)

Worked on core product, data pipeline & produced some cool patents (applied AI)

Lead the Data and AI team:
analysts, scientists, MLEs and SDEs.

Canopus Tech Team

Network Analytics startup born out of UNSW

Bunch of amazing tech folks who solve challenges!

Built an AI-powered network analytics platform and deployed around the world!

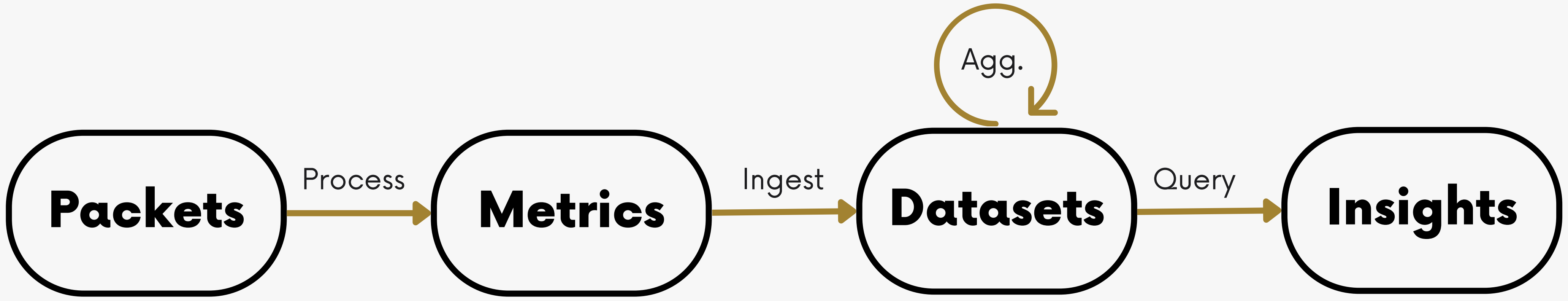
NETWORK MONITORING FOOTPRINT

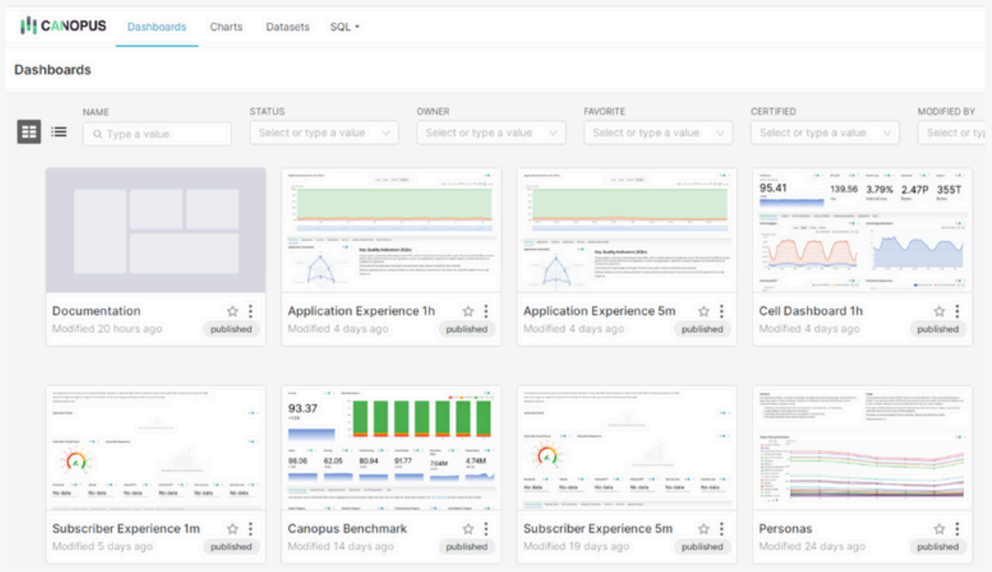
250+ Gbps Monitored Traffic Rates

2M+ Daily Active Users (DAU)

5B+ Daily Events Ingested & Analysed

NETWORK ANALYTICS





Packet Engine: Probe

Parses and extracts packet info

Tracks connections and hosts active on the network

Classifies applications using signatures and ML models

Measures performance in terms of RTT, Loss, Throughput etc.

Data pipeline (this talk!)

Ingest: Take the raw metric stream and dump into DB.

- Avro Serialization
- Kafka
- Write it to DB

Aggregate: Transform raw data into “analysis-ready” datasets

- Roll ups in time
- Group by
- Join with customer metadata

Analysis and Insights

Address customer’s analytics needs

Query the data and show it on dashboards and GUI

- Query caching
- Creating appropriate “views”
- Modelling: Anomaly, Drilldown
- User permissions

AN EXAMPLE RECORD


```
{
  "timestamp": "2024-09-05T10:15:00Z",
  "user_id": "user12345",
  "client_ip": "192.168.1.5",
  "server_ip": "203.0.113.10",
  "download_bytes": 1256789,
  "upload_bytes": 456789,
  "internal_rtt": 15.6,
  "external_rtt": 75.2,
  "internal_loss": 0.01,
  "external_loss": 0.05
}
```


Metric

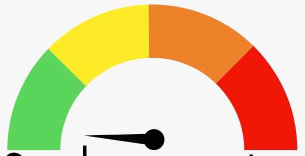
Timestamp	User ID	Client IP	Server IP	Download Bytes	Upload Bytes	Internal RTT (ms)	External RTT (ms)	Internal Loss (%)	External Loss (%)
2024-09-05T10:15:30Z	user12345	192.168.1.5	203.0.113.10	1,256,789	456,789	15.6	75.2	0.01	0.05
2024-09-05T10:20:45Z	user67890	192.168.1.12	203.0.113.11	3,456,123	789,123	22.3	80.5	0.03	0.07
2024-09-05T10:25:10Z	user54321	192.168.1.7	203.0.113.12	987,654	123,456	18.9	70.1	0.02	0.04

Dataset

OUR JOURNEY TO SCALE

Focusing on Ingest, Aggregate and Analyse

HUMBLE BEGINNINGS: CSV



Scale-o-meter

Ingest	Aggregate	Analyse
Probe writes metrics to a CSV file. Rotates the files (one per schema) every 24 hours.	No aggregates whatsover	Jupyter Notebooks + Pandas + Plotly Wrote scripts that produced daily reports over our data that we mailed to our customers.

Good starting point: Getting insights to customers quickly. Pandas on CSV still counts as data science!

Writes aren't fast enough. Uncompressed so storage won't scale.

Querying requires loading all the files in memory.

LET'S GET A SQL DB: TIMESCALE


Scale-o-meter

Ingest	Aggregate	Analyse
Probe writes metrics to the DB. Each file becomes a new table.	Probe pre-aggregated the data in memory and pushed down to DB.	Can now generate the same reports in an efficient way. Started to develop our full stack apps backed by Timescale.

Liked the convenience of SQL, multiple modules could query the data :)

Timescale recent "chunk" in RAM: Real-time dashboards ↗ but historical dashboards ↘

As we scaled, Timescale was taking up half the RAM on our servers -> query timeouts!

DB TO PIPELINE: KAFKA + CLICKHOUSE

Scale-o-meter

Ingest

Probe writes metrics to Kafka topics.

Use avro serialization

Clickhouse Kafka Engine

Aggregate

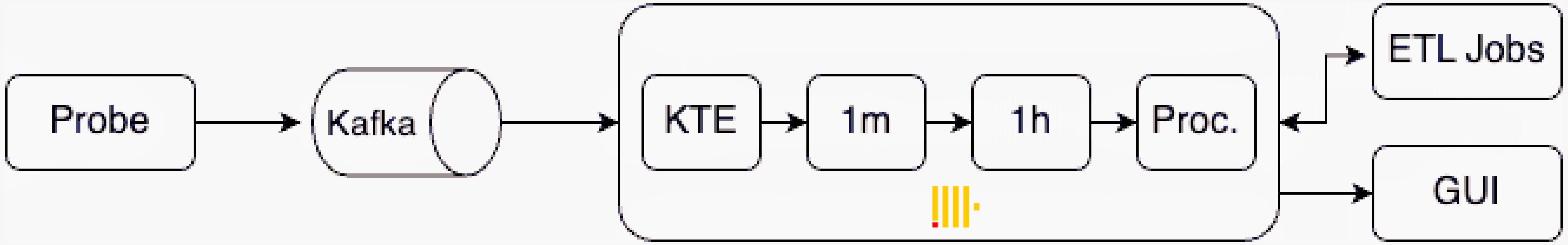
Clickhouse Materialized Views

- Materialize from kafka table into a merge tree table
- Roll up into time buckets

Analyse

Introduced scheduled ETL jobs to transform the data via Prefect.

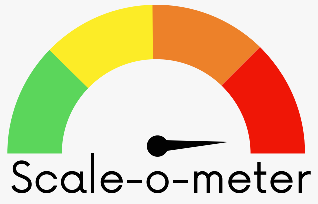
GUIs got faster.



PURE MAGIC! Clickhouse was fast at low RAM and disk usage! Served us good until 10-100 millions of rows / day.

The customers always want more: Won't scale to billions of rows!

5B+ EVENT ANALYTICS PIPELINE



Scale Ingest

Implemented Kafka-Go-Connect instead of kafka table engine:

- ch-go lib
- convert from avro -> native format batches
- async write mode.

Use SSDs as landing zone and TTL move to HDDs.

Scale Aggregate

Real-time Aggregates using Flink

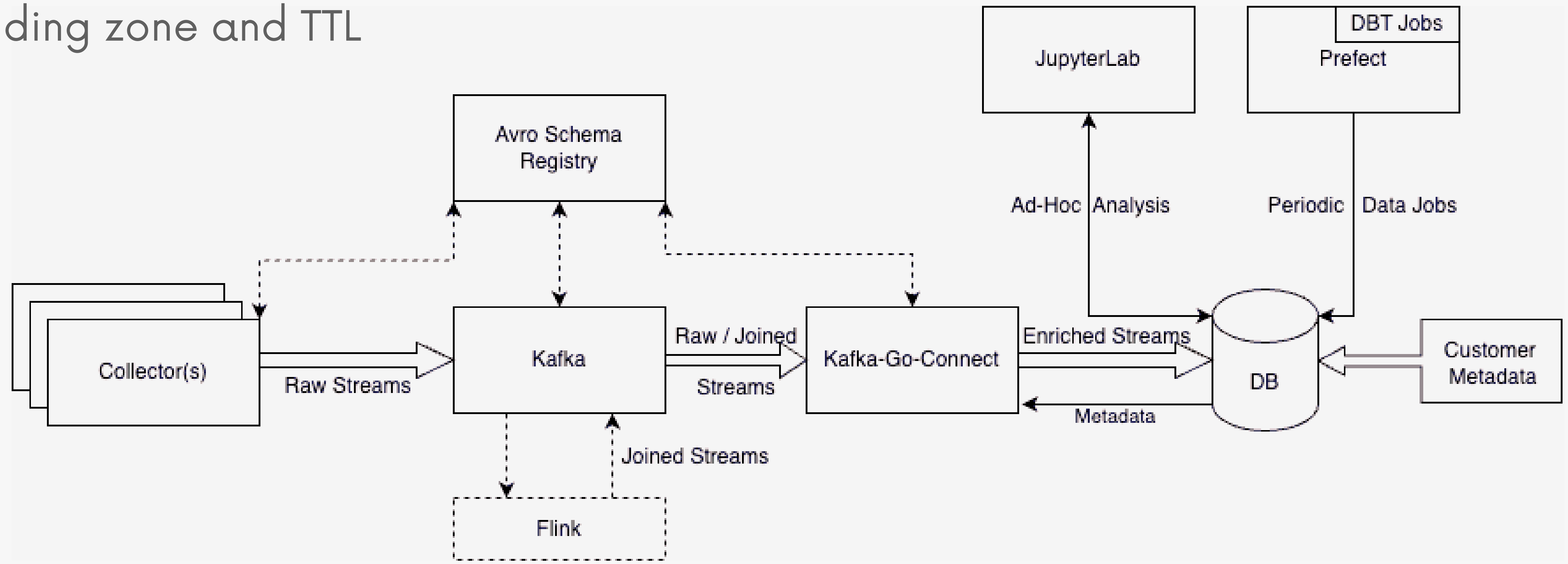
DBT to manage roll ups vs. MVs

Split up incoming data streams into two nodes. Agg independantly.

Scale Analyse

Resource distribution: route query based on table between 2 servers.

Slow queries got converted into DBT models.



LESSONS LEARNT

Tips and Takeaways

MOVING FROM NATIVE MVS TO DBT

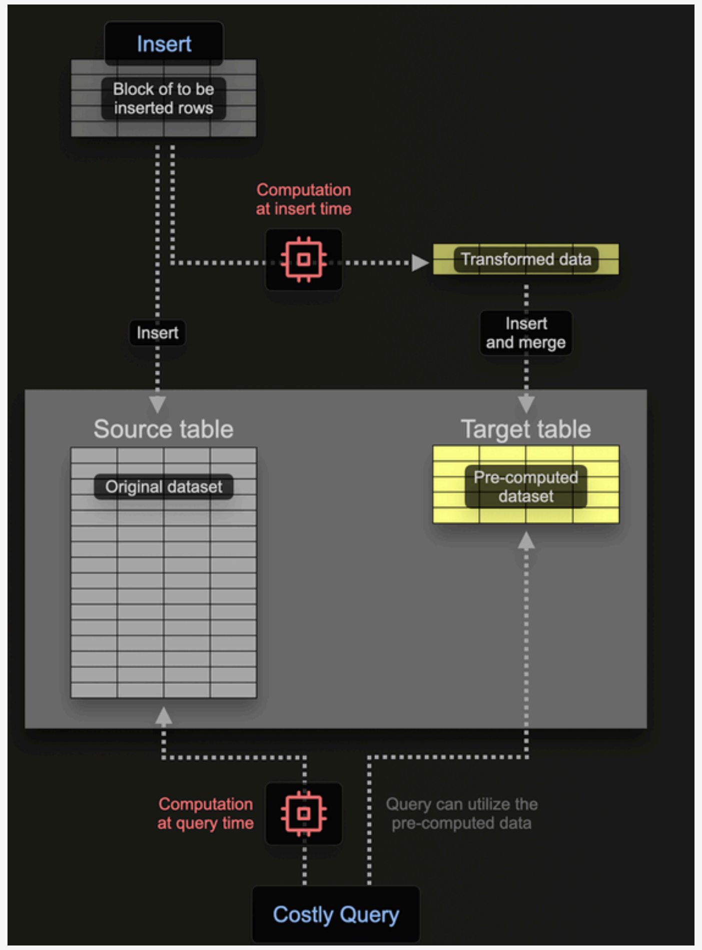
Native MVs

Aggregate block of rows and store in a target table.

Good for Simple Aggregation (E.g. sum, count etc.) on a single stream via Plain SQL.

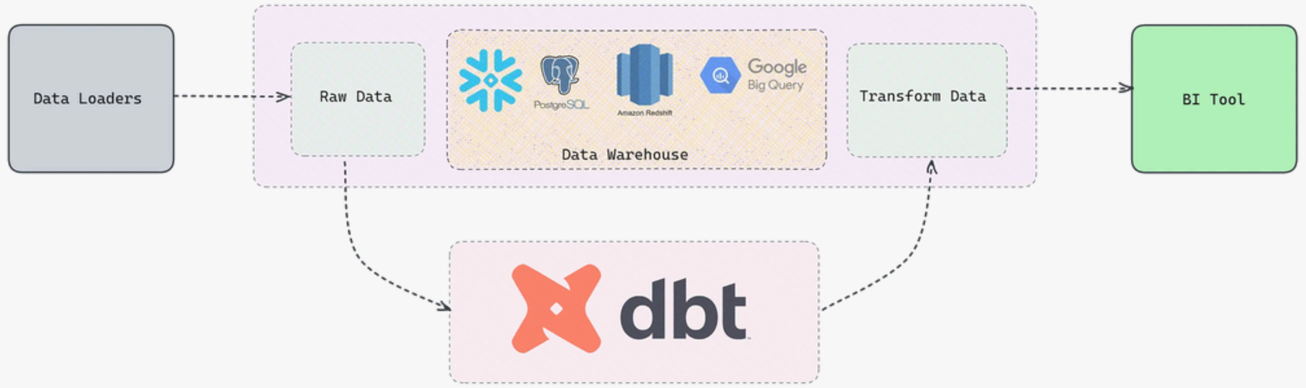
Complex Aggregation is a two stage materialization: state tables and "merge queries".

Joining two streams is challenging with MVs (there's no wait time)



Source: https://clickhouse.com/docs/en/materialized-view

DBT



Source: https://medium.com/@sagar.bhandge0310/dbt-data-build-tool-overview-67cc77e761ea

General purpose data transformation tool via templated SQL

Write "Models" (basically SQL queries) => tables (via config)

Supports dependancies between models and incremental materialization only new rows added (for timeseries datasets)

For continuous data, `dbt run` has to be scheduled at regular intervals using schedulers like cron, prefect or airflow.

ANALYZE AND MATERIALIZE LOOP

Identify Slow Queries

Use Clickhouse Query Log table to observe repeated slow queries.

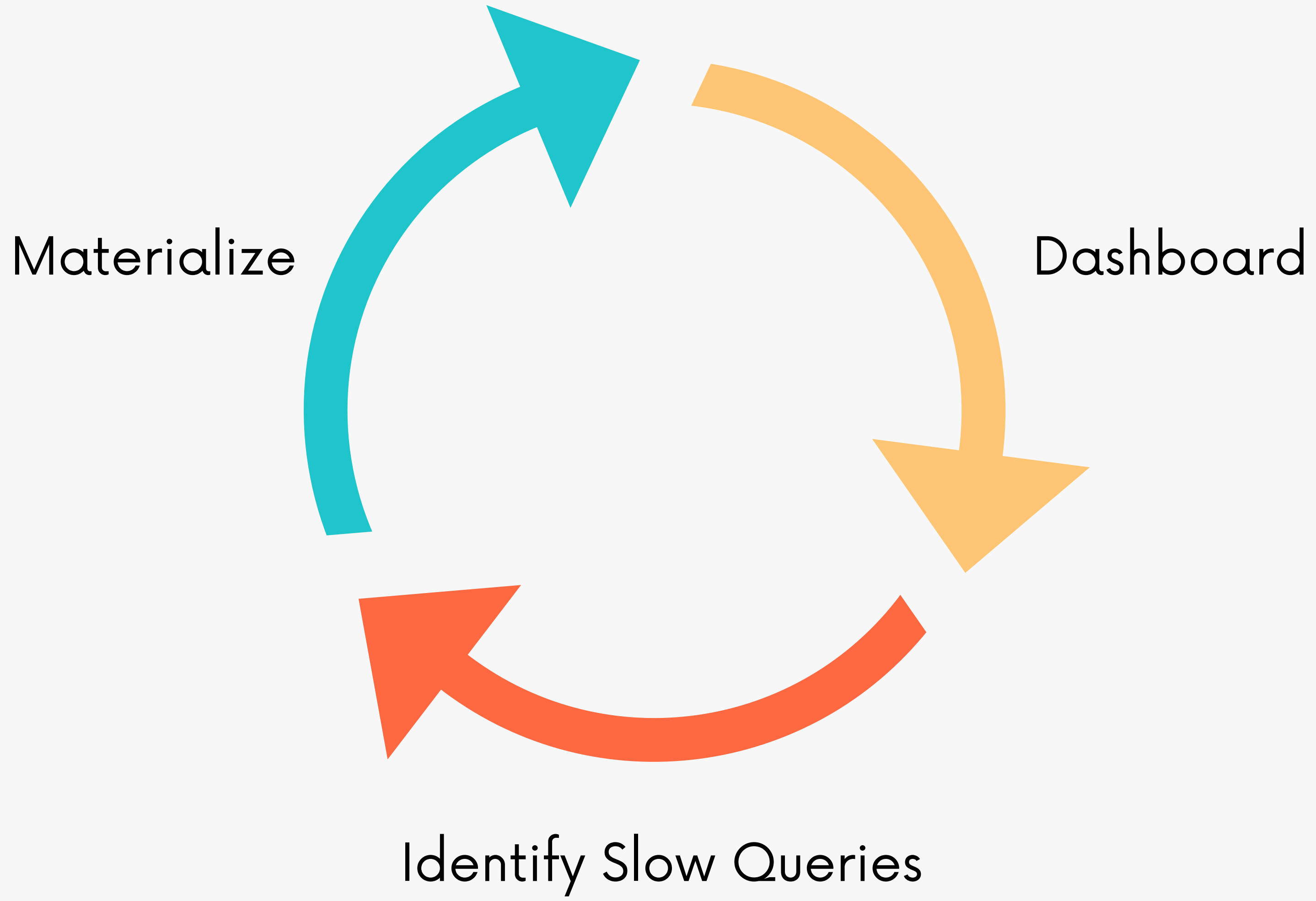
Materialize

Use DBT to build an MV using the query. DBT will automatically handle table creation etc. (PK order given via config)

Schedule incremental updates to MVs using Prefect/Airflow

Dashboard

Continue building more analytics for customers.



MONITORING THE PIPELINE

> If you can't measure it, you can't manage it!
> Peter Drucker

Leading metrics: a potential future issue

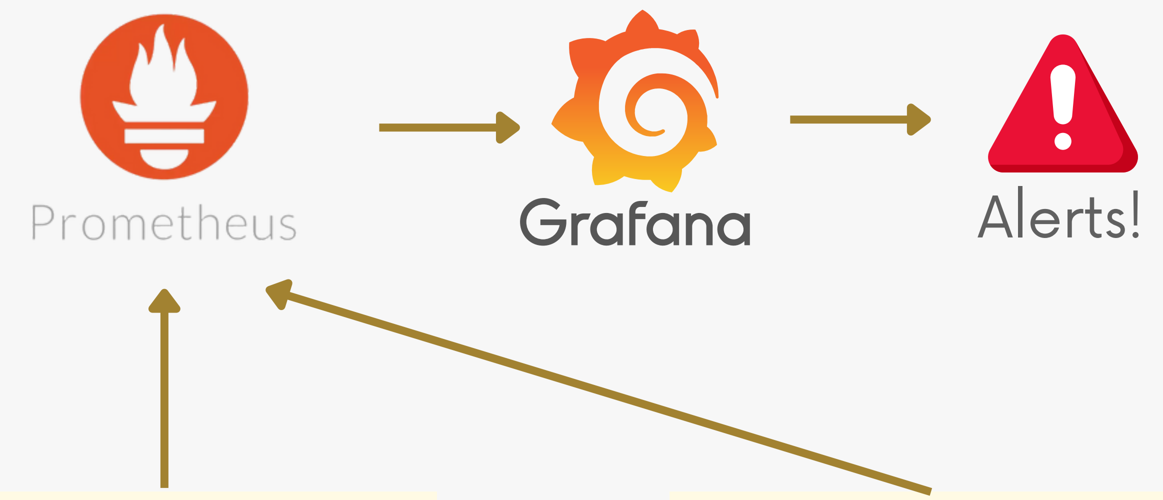
- Kafka lag increasing
- Clickhouse merges taking more time
- SSDs filling up (cascading effect)
- Batches taking more time to write (instrumented our kafka-connect)

Alerting metrics: Indicate an immediate issue

- Data lag on the table
- Running out of memory, disk etc.
- ETL jobs failing

Clickhouse System Tables are an immensely useful source!

Our Design



Out-of-the-box

Kafka Lags

Clickhouse Stats

Flink Stats etc.

Custom Metrics

Clickhouse Lag

Batch-write time

Job Status etc.

Thank you

My Top 3 takeaways

Start small and scale up: focus on delivering value through data first.

Right Basics/Arch > Tools: Get the tool/lib if you can't write the functionality yourself. E.g. ClickHouse :P

Monitor is pre-req to scale: Invest in measuring and monitoring the right metrics. Instrument via code if needed.

Let's connect on LinkedIn!