Building a real-time analytics platform

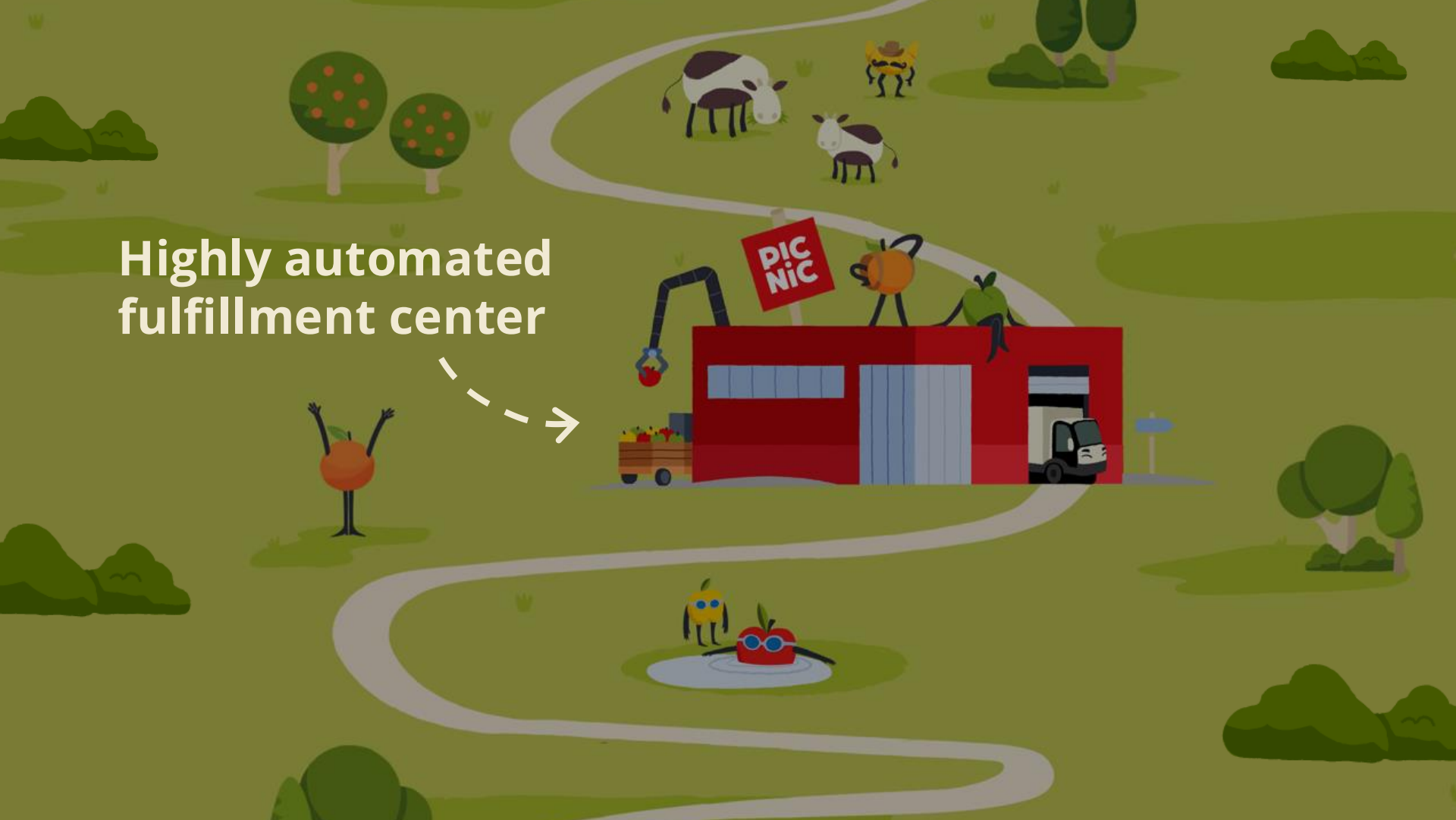
Max Sumrall
Tech @ Picnic

Picnic?

1. What is Picnic?


PIC NIC
V-930-RL

1M unique stores


Highly automated
fulfillment center
PIC
NIC


The last
mile

Fulfilment Centers - FCA Utrecht

× 20 more of these…

NL The Netherlands
DE Germany
FR France

Agenda

1. Picnic's *Real Time Insights* platform
2. Why ClickHouse's *flexibility* is so powerful
3. Sneak peek of *what's next*

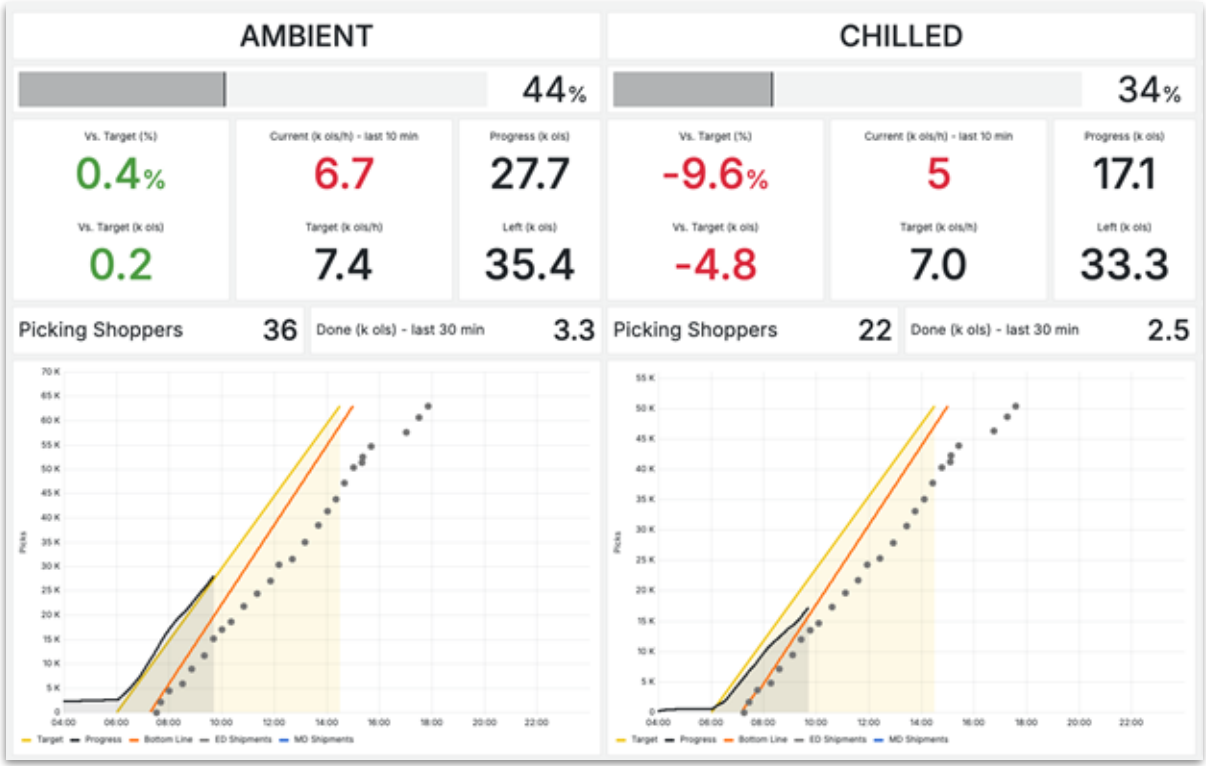
Real-time Insights

Why?
We need real-time information on supply chain processes!

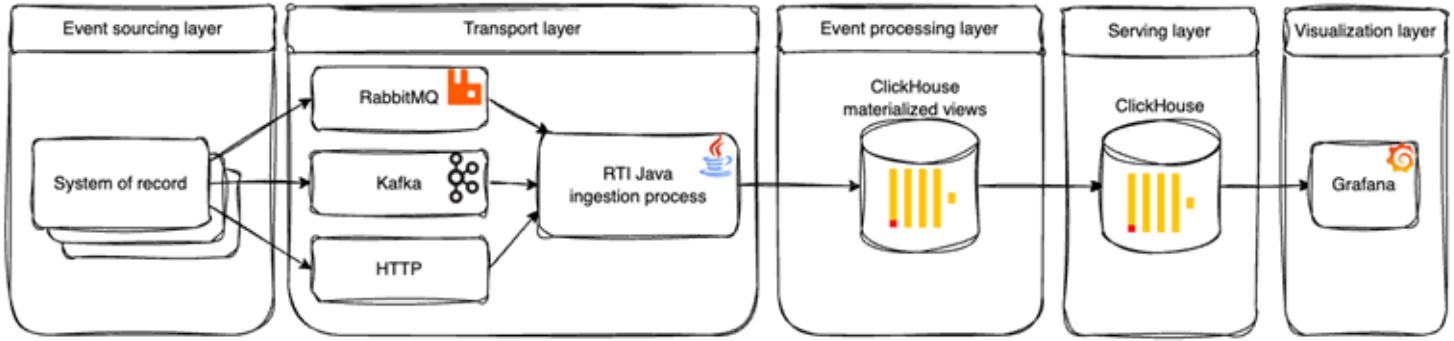
Solution
The *real-time insights* (RTI) platform.

Goals
- **Self-service** platform
- Real-time data, not delayed
- Improve operational efficiency

Real time observability into supply chain operations



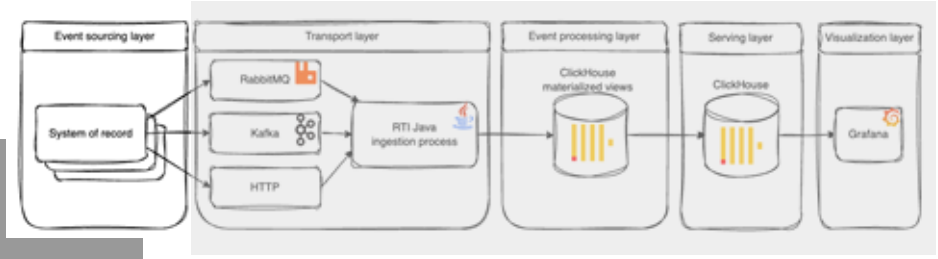
Architecture



Events

The Event Sourcing layer collects various types of data:

- **Backend data**: Data from server-side applications
- **Frontend data**: Clickstream data from the mobile app
- **Slowly moving dimensional data**: Product assortment, vehicle information, warehouse layout

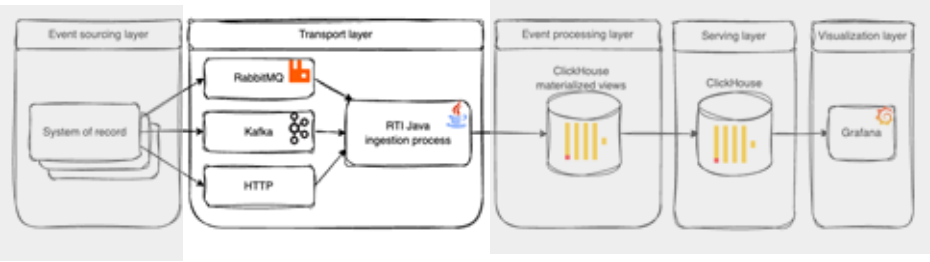



```yaml
sources:
  - name: sources
    tables:
      - name: calcite_analyzer_analytics_events
        meta:
          type: KAFKA
          topic:
            "{{ env_var('PEE_MARKET') }}-{{
            env_var('PEE_ENVIRONMENT')}}.calcite.analyzer"
        columns: *default_columns
      - name: delivery_capacity_update
        meta:
          type: RMQ
          exchange: "picnic.capacity"
          routing_key: "picnic.capacity.update"
        columns: *default_columns
```

Transport Layer

The Transport Layer uses technologies such as **RabbitMQ** and **Apache Kafka**. These technologies ensure efficient and reliable data transport, enabling real-time data flow.

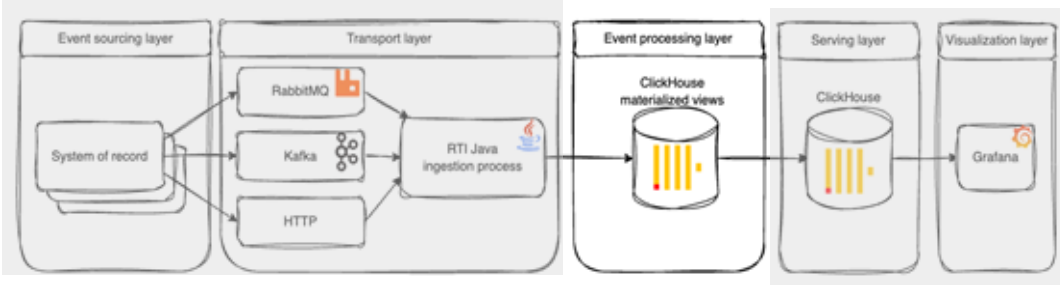
A Java application is responsible for ingesting data into **ClickHouse**.



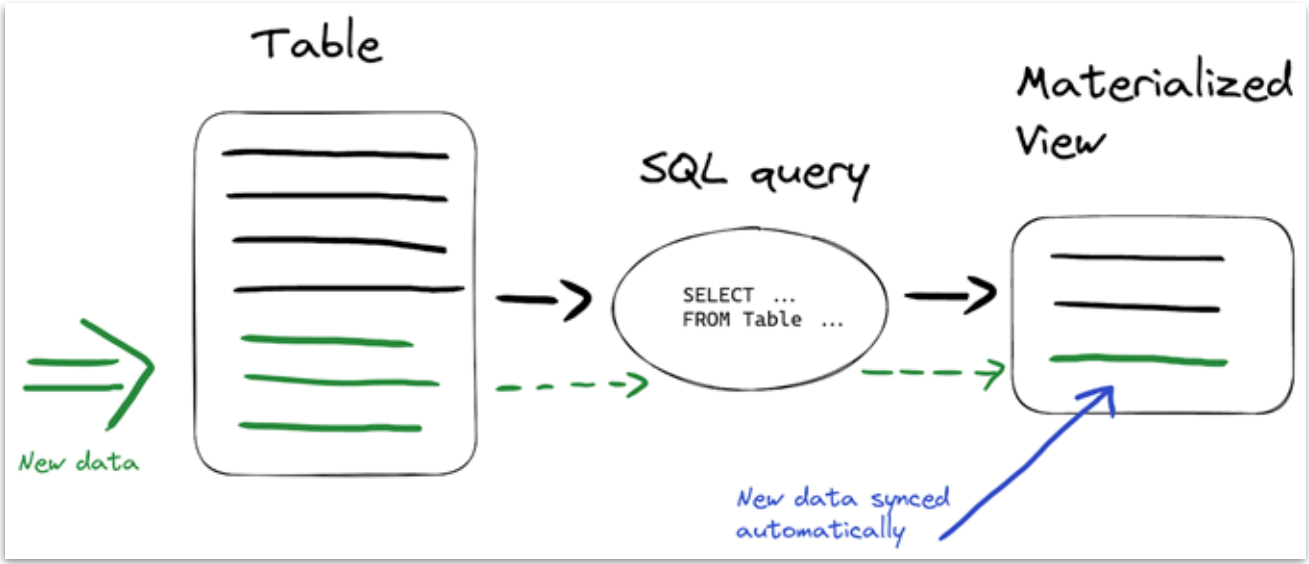
Event Processing Layer

The Processing Layer is responsible for transforming and aggregating raw data.

- **materialized views**: Parses and aggregate raw events
- **refreshable materialized views**: periodic updates



Materialized views in the processing layer



Materialized views in the processing layer

```
Row 1:
──────
record_content_str:  {"warehouse_id":"FC1","arrival_event_type":"CLOSE_ARRIVAL","arrival_id"
record_metadata_str: {"SOURCE_TYPE":"KAFKA","rabbitmq.headers.Picnic-Trace-Id":"QZkc4zaq"}
load_ts:             2025-06-25 06:55:40.234

Row 2:
──────
record_content_str:  {"warehouse_id":"FC1","arrival_event_type":"CREATE_ARRIVAL","arrival_ic
record_metadata_str: {"SOURCE_TYPE":"KAFKA","rabbitmq.headers.Picnic-Trace-Id":"jIFpTGd2"}
load_ts:             2025-06-21 08:46:46.170
```

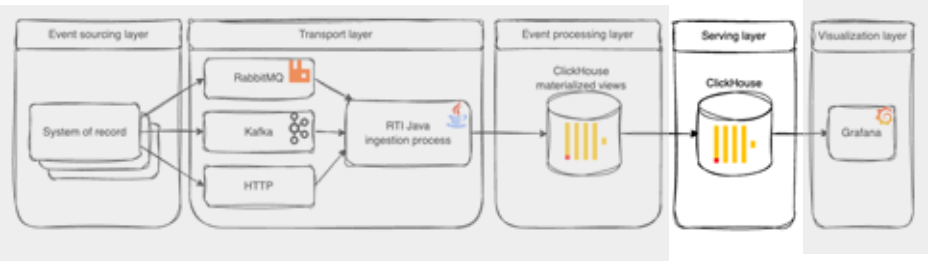

```
Row 1:
──────
event_time:                2025-06-25 06:55:40.230
warehouse_id:              FC1
arrival_event_type:        CLOSE_ARRIVAL
arrival_id:                ARR-FC1-458b5190-4f60-440b-930c-6b80213fec35
arrival_time:              2025-06-25 04:45:00.000
logistic_service_provider: TOUFLET
dock:                      D-01
driver_name:               .
license_plate:             .
arrival_status:            ARRIVED
goods_temperature1:        0
goods_temperature2:        0
goods_temperature3:        0
```

Serving Layer

The Serving Layer is implemented with **ClickHouse**, utilizing different table engines tailored for specific data serving needs.

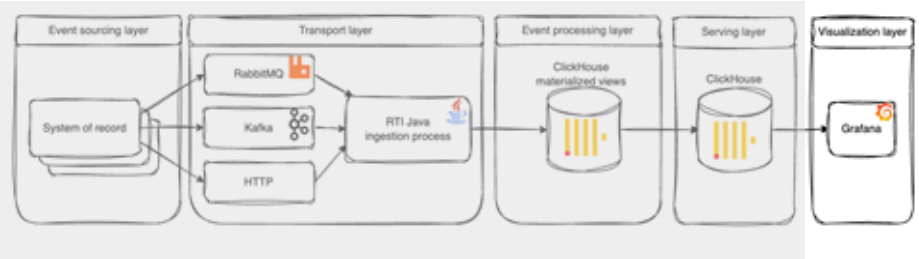
To query data from ClickHouse, analysts can write **free-format SQL queries**.

Right indexing is crucial for performance.



Visualization layer

Grafana renders visualizations from ClickHouse data



Queries 4 | Transformations 0 | Alert 0

A | ClickHouse

Editor Type: SQL Editor | Query Builder

Query Type: Table | Logs | Time Series | Traces

```
with 'AMBIENT' as temp_zone
, timestamp_values as (
    select
            multiIf(
                temp_zone = 'CHILLED',
                toDateTime($chilled_start_time),
                temp_zone = 'FROZEN',
                toDateTime($frozen_start_time),
                toDateTime($ambient_start_time)
            ) as start_time,
            multiIf(
                temp_zone = 'CHILLED',
                toDateTime($chilled_end_time),
                temp_zone = 'FROZEN',
                toDateTime($frozen_end_time),
                toDateTime($ambient_end_time)
            ) as end_time,
            toHour(start_time, 'Europe/Amsterdam') >= 12 as evening_process
        from generate_series(1,1,1) as data
)
```

Platform Engineering

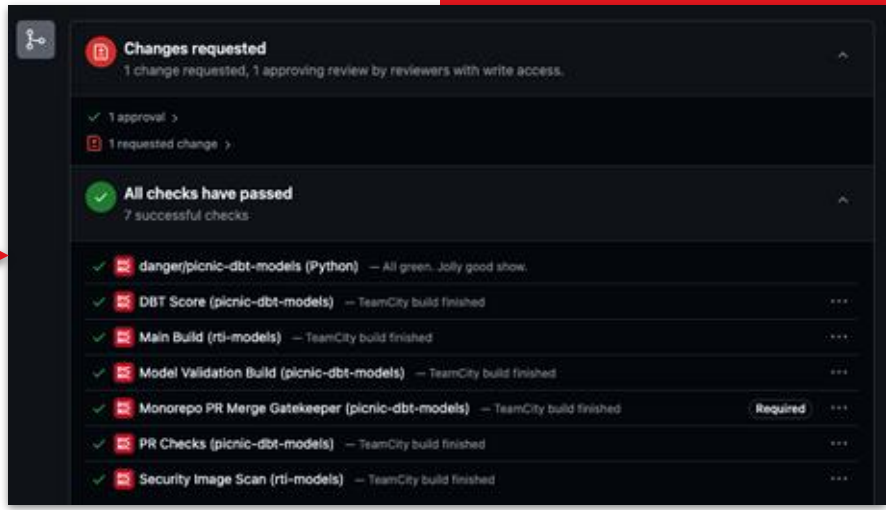
RTI is built upon the **platform engineering paradigm**

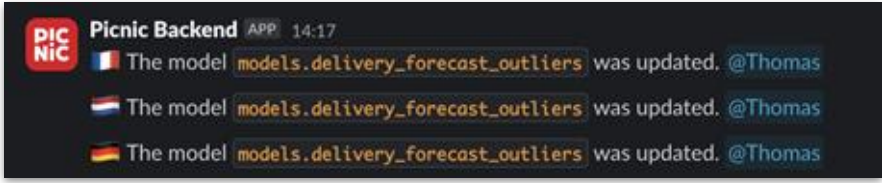
- Minimize technical effort for platform users
- Serve a variety of different teams in the company
- Enable users to focus on business logic

Minimizing Technical Heavy Lifting

Platform features:

- **Data ingestion** by configuration (yaml)
- **PR checks**: Ensuring query integrity and quality
- **CI/CD pipelines**: Automated deployment of changes, including notifications
- **Monitoring and alerting**: The platform team ensures performance and availability
- **GDPR** compliance features





ClickHouse & dbt

- **dbt:** Configure data as code
  - Manages tables and their dependencies
  - Enables **reprocessing** of raw data

- **dbt-score:** Picnic's open-source *linter* for dbt metadata

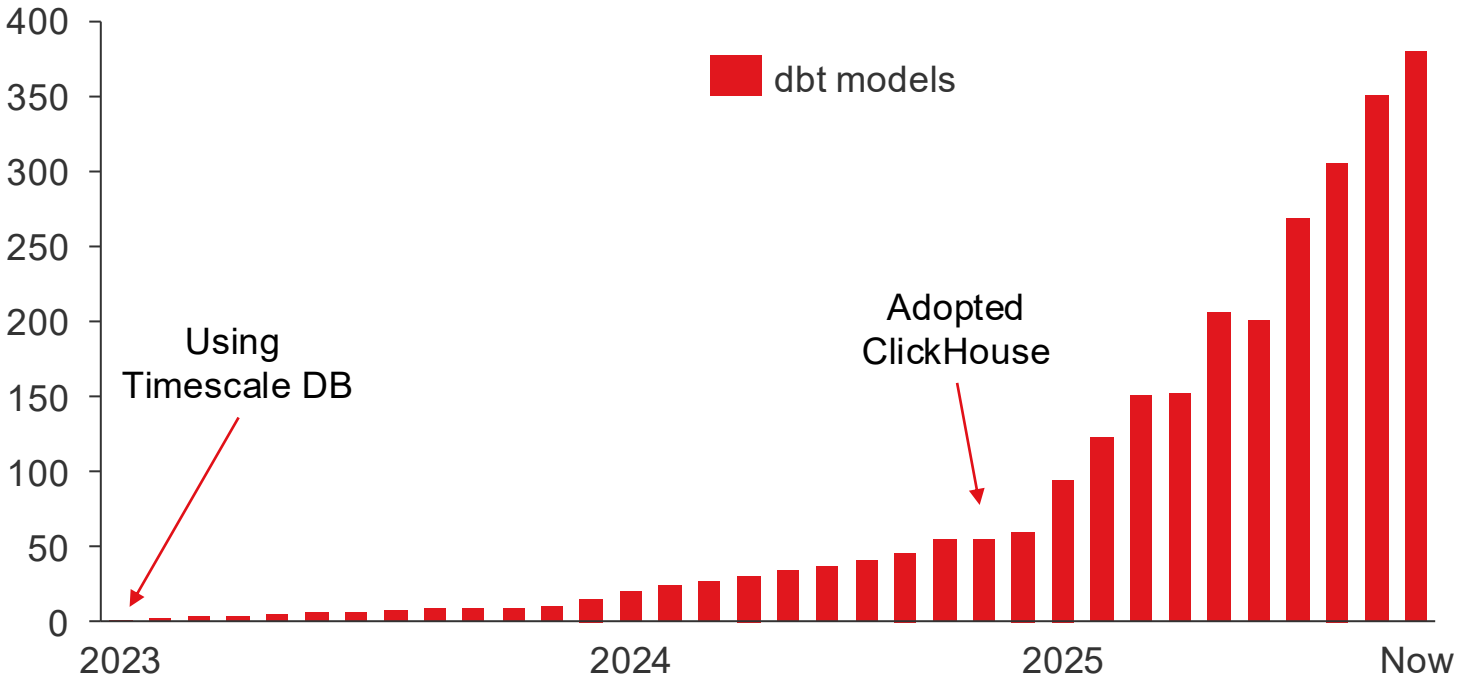
Path to ClickHouse

- **TimescaleDB**
  - familiar Postgres SQL
  - *Hypertables* and *Continuous Aggregates*
- **As we scaled:**
  - complex tuning
  - hard to run as a shared platform.

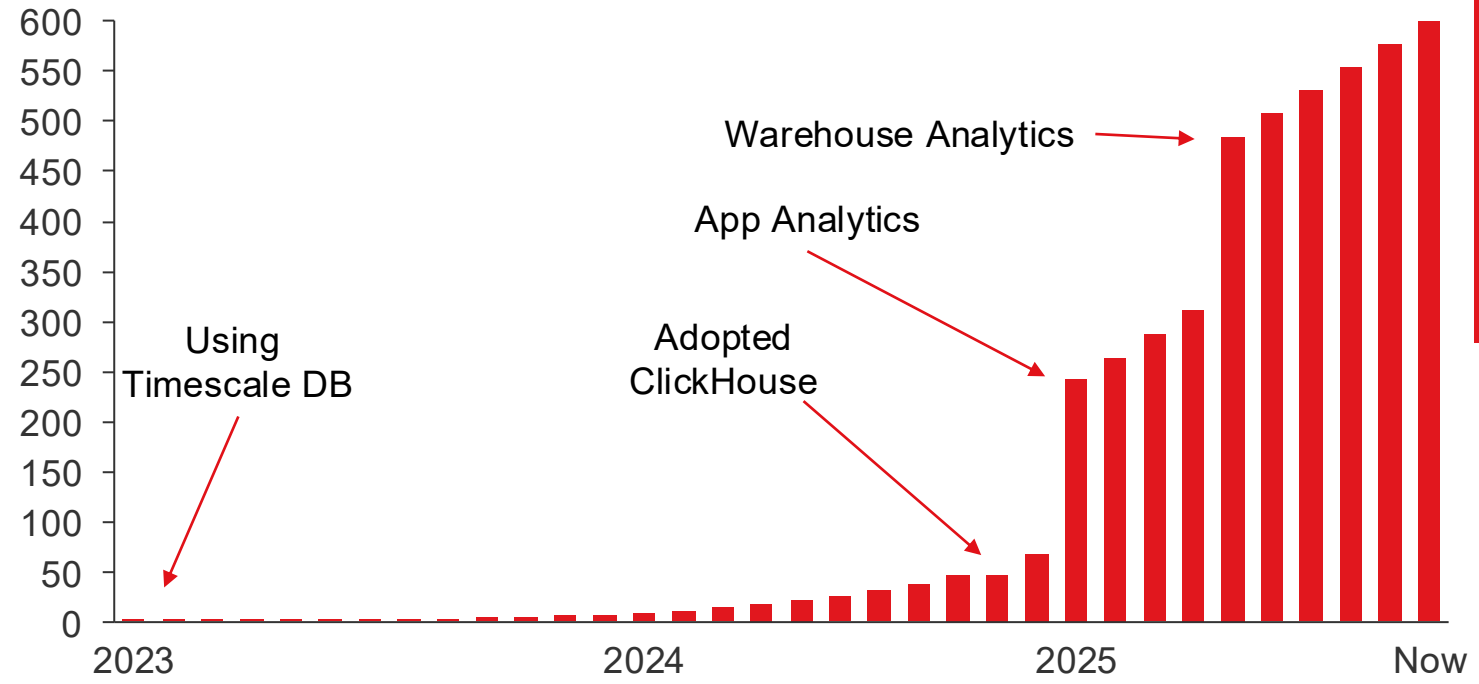
Path to ClickHouse

- **We needed:**
  - High Performance
  - Cheap data retention
  - Query isolation
  - Simple management and maintenance
  - SQL
- **Why ClickHouse**
  - Materialized Views
  - Object Storage
  - TTLs
  - Easy scaling

Path to ClickHouse



Path to ClickHouse



ClickHouse Flexibility

Deep Dive on GDPR + PII Data

- Only keep data as long as necessary for its purpose
  - ✅ ClickHouse TTL set to 4 days by default

- Only see data you ***Need*** for your job.
  - *AMS* managers only need to see *AMS* employee data.
  - ⚠️ ***How can we do this with Grafana and ClickHouse?***

Authentication

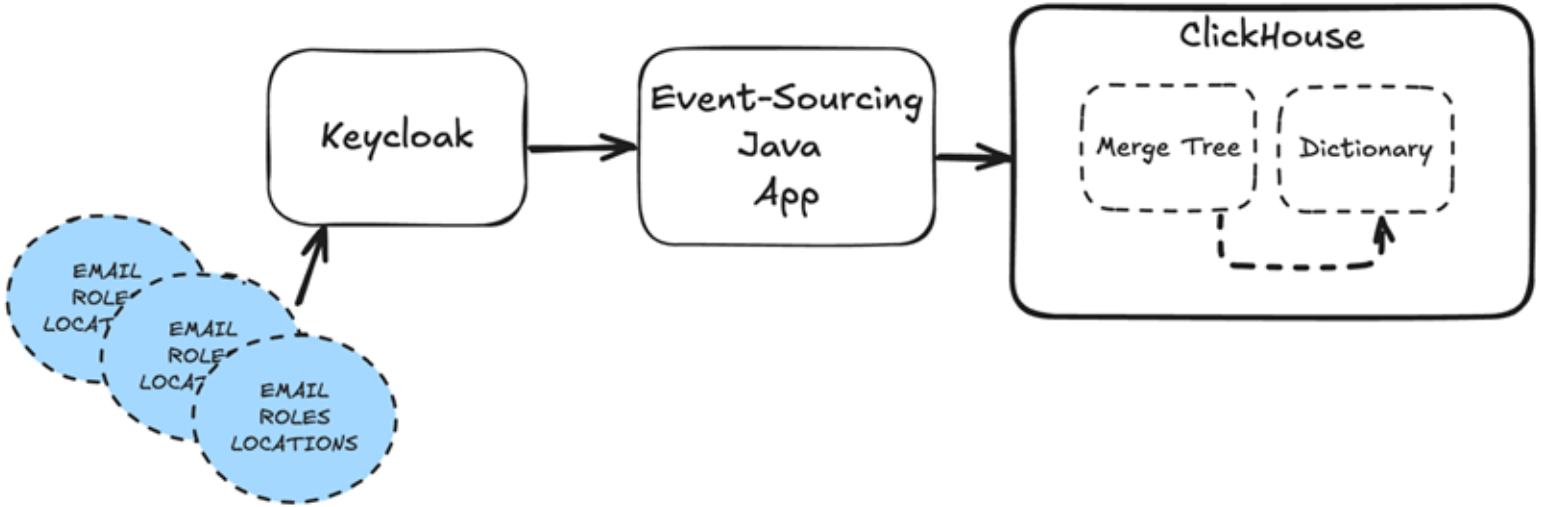
1. **ClickHouse function:**

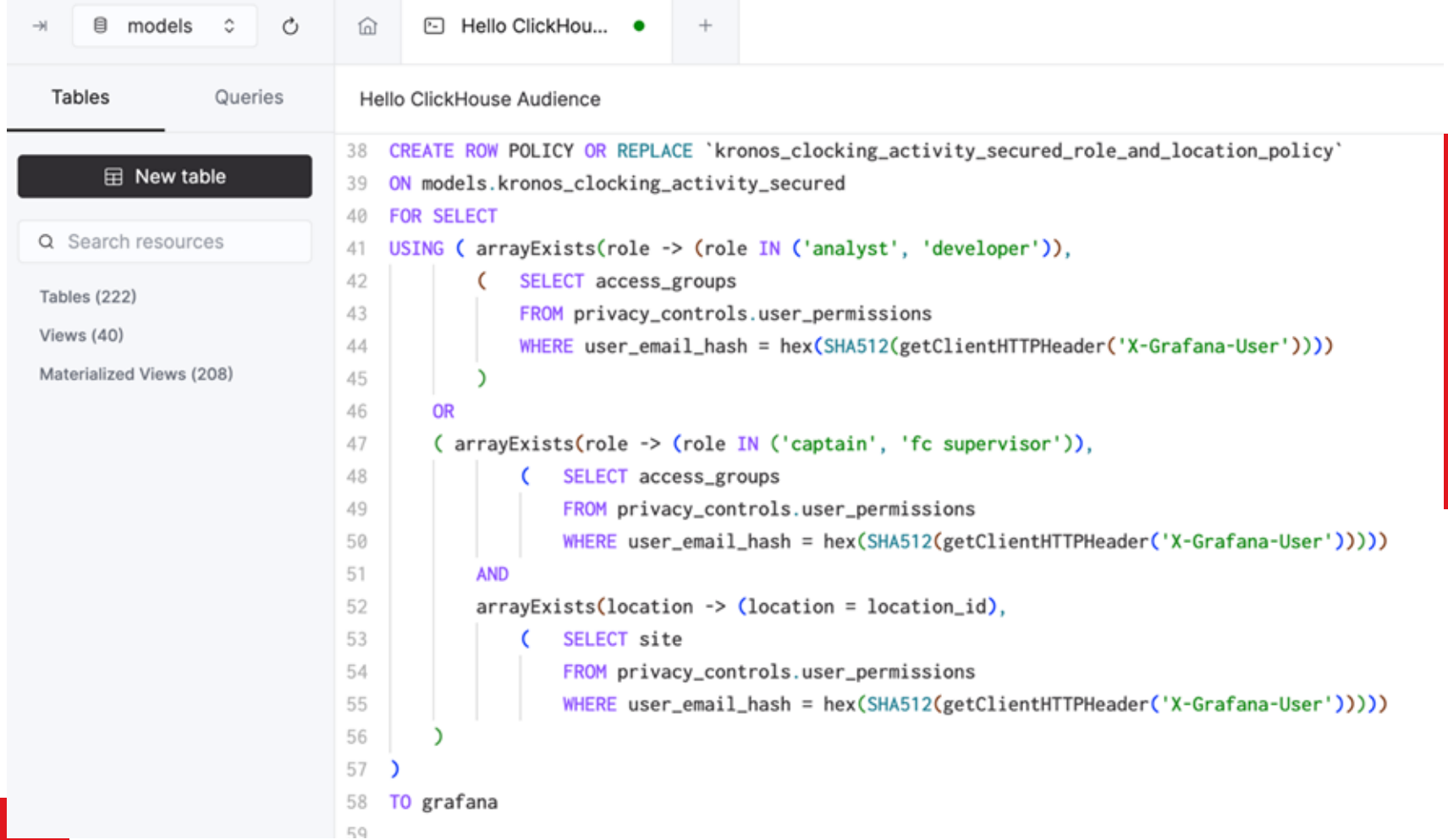
```
SELECT getClientHTTPHeader('X-Grafana-User')
```

2. Grafana forwards header with authenticated user.
   a. *Picnic uses Keycloak for Grafana access*

Authorization

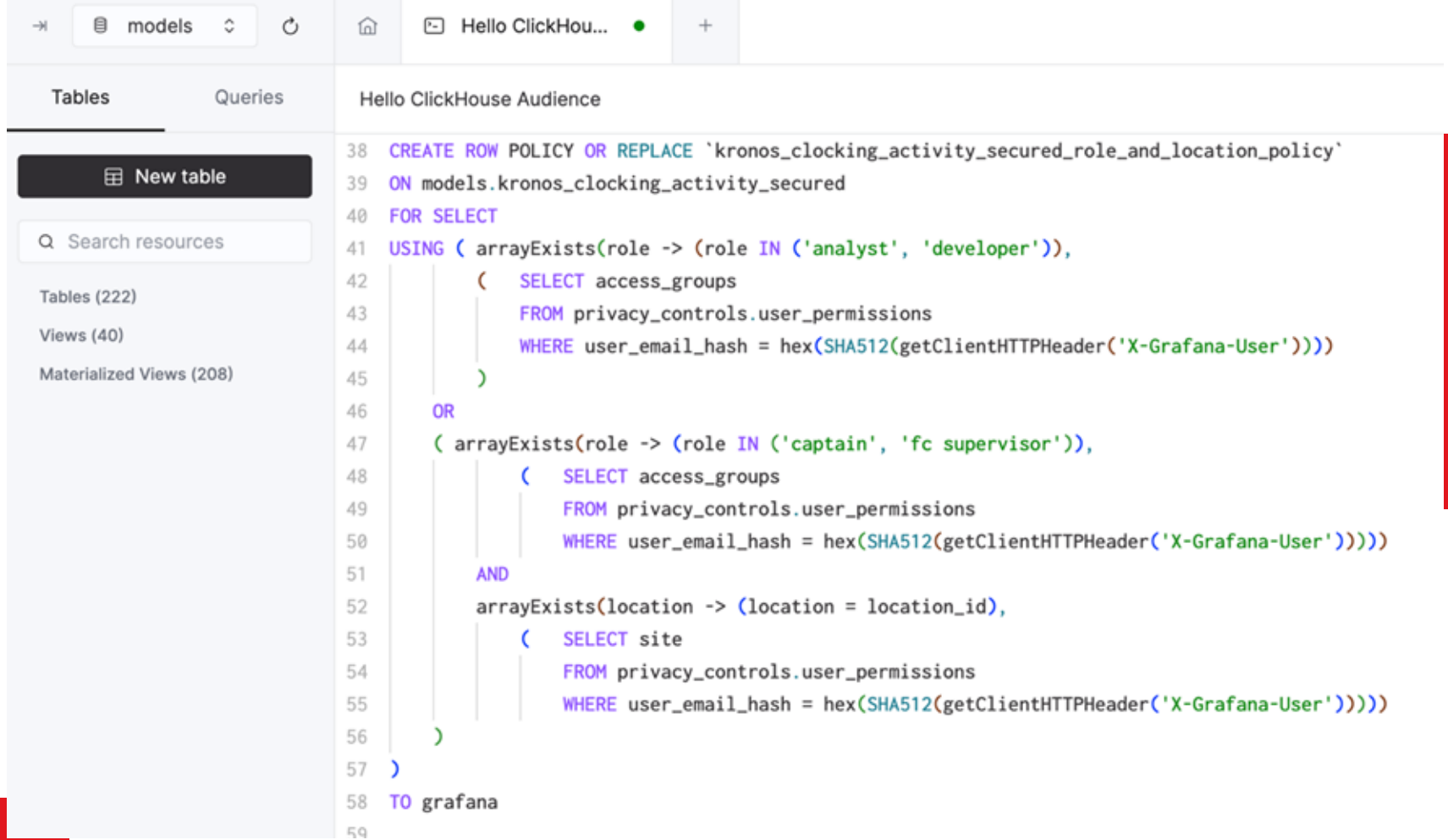
- Replicate user permissions as ClickHouse dictionary.
- dbt script to automatically create row policy tailored for each table




models
Hello ClickHou...
Tables
Queries
Hello ClickHouse Audience
New table
Search resources
Tables (222)
Views (40)
Materialized Views (208)
CREATE ROW POLICY OR REPLACE `kronos_clocking_activity_secured_role_and_location_policy`
ON models.kronos_clocking_activity_secured
FOR SELECT
USING (arrayExists(role -> (role IN ('analyst', 'developer')),
(SELECT access_groups
FROM privacy_controls.user_permissions
WHERE user_email_hash = hex(SHA512(getClientHTTPHeader('X-Grafana-User'))))
)
OR
(arrayExists(role -> (role IN ('captain', 'fc supervisor')),
(SELECT access_groups
FROM privacy_controls.user_permissions
WHERE user_email_hash = hex(SHA512(getClientHTTPHeader('X-Grafana-User')))))
AND
arrayExists(location -> (location = location_id),
(SELECT site
FROM privacy_controls.user_permissions
WHERE user_email_hash = hex(SHA512(getClientHTTPHeader('X-Grafana-User')))))
)
)
TO grafana

Policy as Code, Validate on Github Pull Request

1. Automated GDPR scanning.
    a. real PII → **build fails** until policy added.
    b. false positive → column-level skip (re-triggers on future edits).
2. Custom **dbt** scripts auto-generates/updates **ROW POLICY**.
3. All changes reviewed, versioned, auditable.


models
Hello ClickHou...
Tables
Queries
Hello ClickHouse Audience
New table
Search resources
Tables (222)
Views (40)
Materialized Views (208)
CREATE ROW POLICY OR REPLACE `kronos_clocking_activity_secured_role_and_location_policy`
ON models.kronos_clocking_activity_secured
FOR SELECT
USING (arrayExists(role -> (role IN ('analyst', 'developer')),
(SELECT access_groups
FROM privacy_controls.user_permissions
WHERE user_email_hash = hex(SHA512(getClientHTTPHeader('X-Grafana-User'))))
)
OR
(arrayExists(role -> (role IN ('captain', 'fc supervisor')),
(SELECT access_groups
FROM privacy_controls.user_permissions
WHERE user_email_hash = hex(SHA512(getClientHTTPHeader('X-Grafana-User')))))
AND
arrayExists(location -> (location = location_id),
(SELECT site
FROM privacy_controls.user_permissions
WHERE user_email_hash = hex(SHA512(getClientHTTPHeader('X-Grafana-User')))))
)
)
TO grafana

dbt security policy

```yaml
meta:
  security:
    full_access_roles: [Developer, Analyst]
    location_based_access_roles: [Captain, FC Supervisor]
    location_column: location_id
  owner: "Max Sumrall"
  team: "Analytics Platform Team"
  alert_slack_users:
    - "U6WSLBS9Z" # Max Sumrall
```

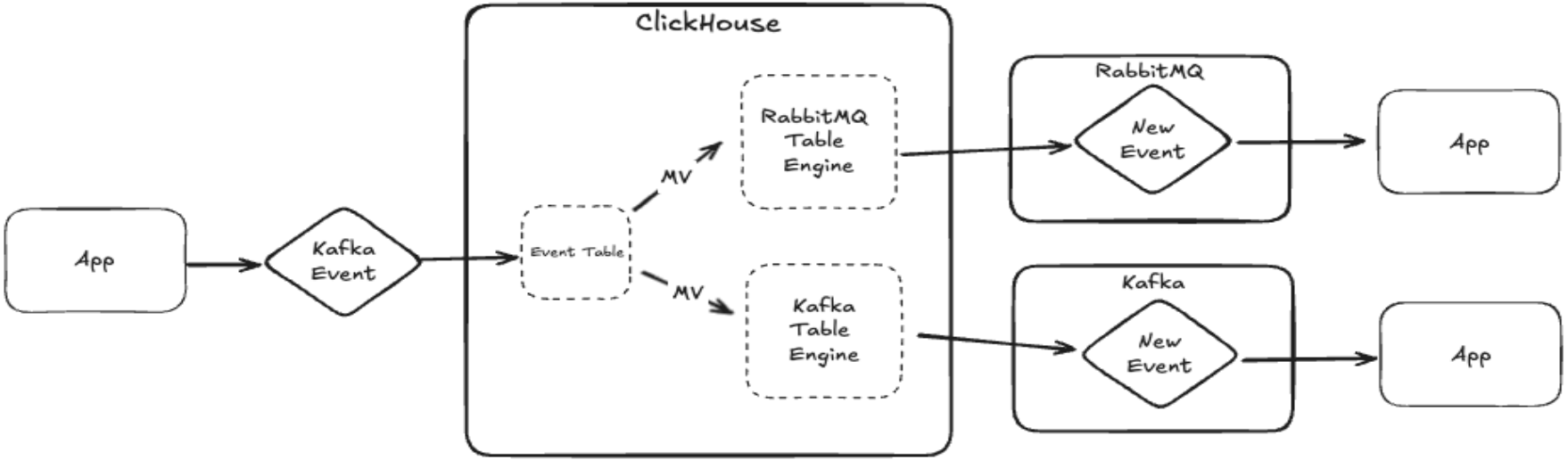
Up Next

Up Next

ClickHouse as Stream Processor

Flink
SQL

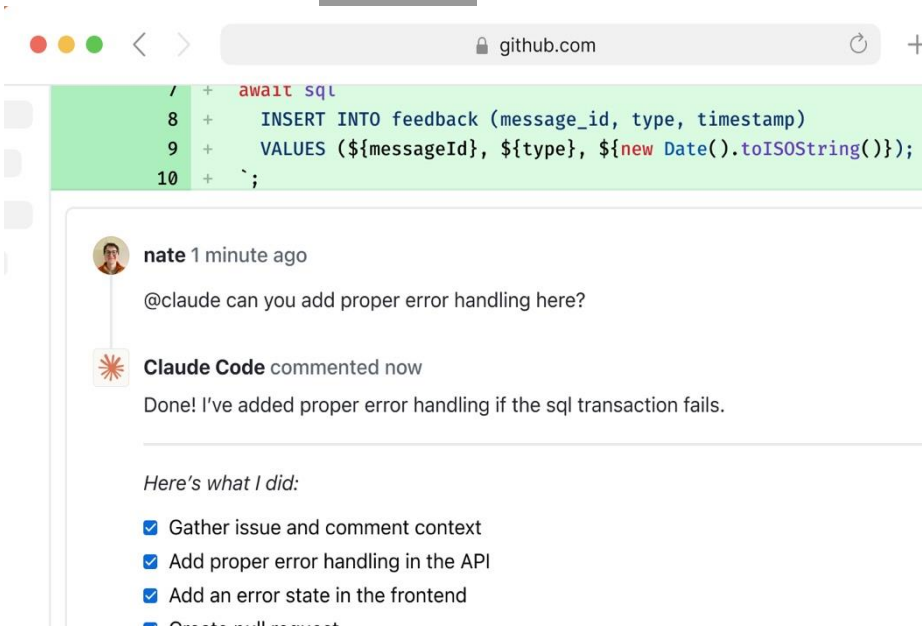
ClickHouse
SQL
+ Table engines



Up Next

AI Model Assistant

- Claude Code GitHub Action
- All models have metadata, examples
- Combine with documentation, tasks
- Advise on proposed changes
- Ask AI Agent business need, answered using dbt metadata and ClickHouse data



Thank you!

Max Sumrall

picnic.jobs