Data Lake, Real-time Analytics, or Both

Robert Hodges – Rohan Pednekar

Let's make some introductions

Robert Hodges

Database geek with 30+ years on DBMS systems. Day job: CEO at Altinity

Rohan Pednekar

Sr. Product Manager at Ahana, Open Source Evangelist

…And introduce our companies

Altinity

ahana™

Altinity.Cloud Platform for ClickHouse

Real-time data in the cloud, on Kubernetes, and on-prem

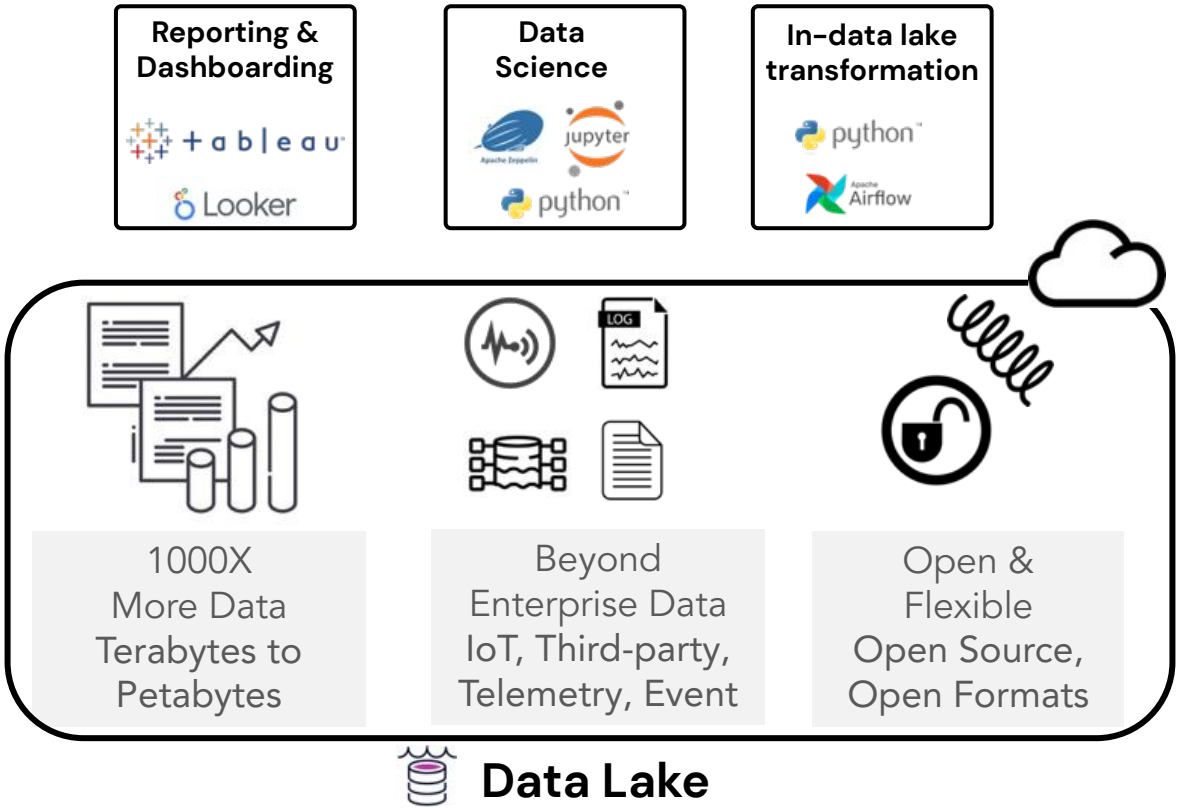
Fully managed Presto Service on AWS

Query your AWS S3 Data Lakes with SQL

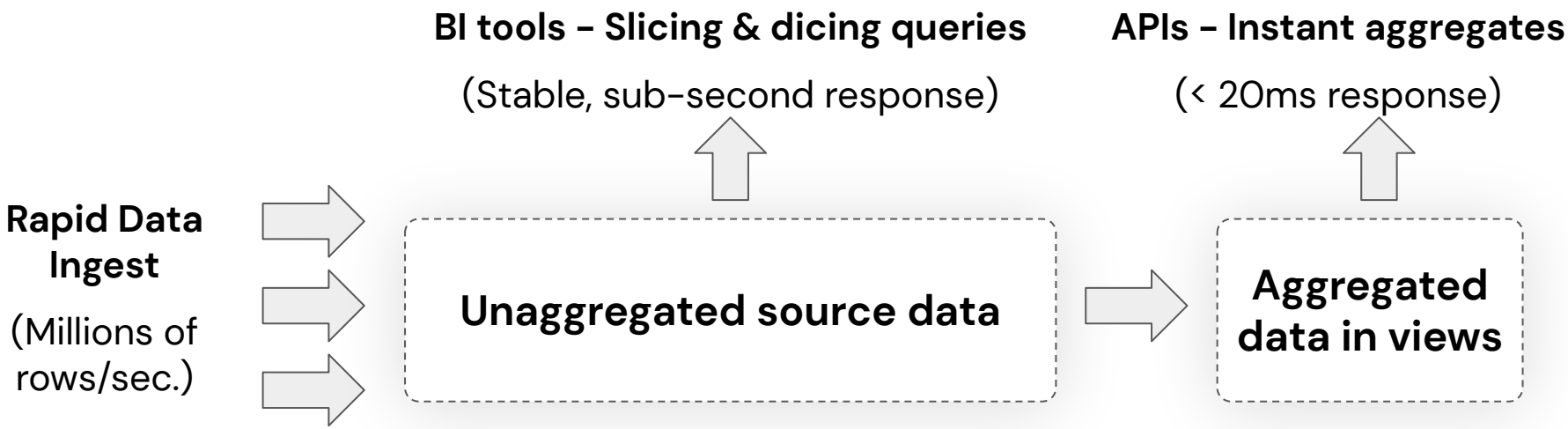
Data Lakes & Real Time Analytics

Let’s discuss data lake and real-time analytic approaches

What are data lake analytics?



What are real-time analytics?



Data Lakes & Real Time Analytics

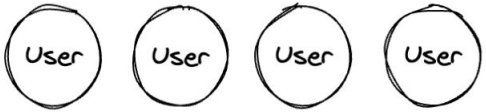
How do you know which approach is best for you?

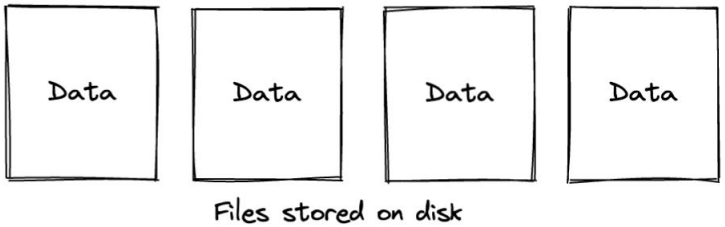
Presto: SQL Query Engine for big data

Today's Challenges for Data Engineers & Data Architects

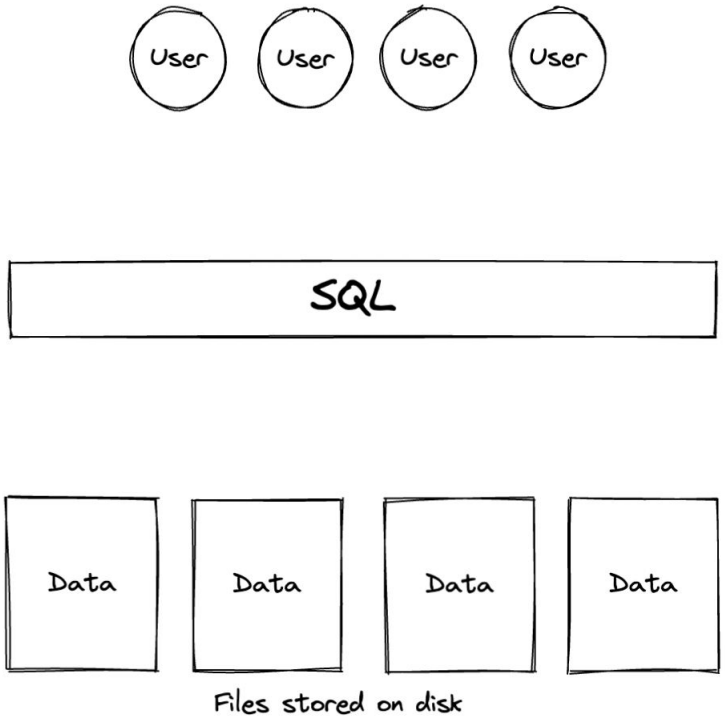
- Storage and Compute
- Diverse Data Sources
- Managing different SQL dialects
- Onboarding time
- Cost of proprietary systems

What do we need for cloud based analytics?

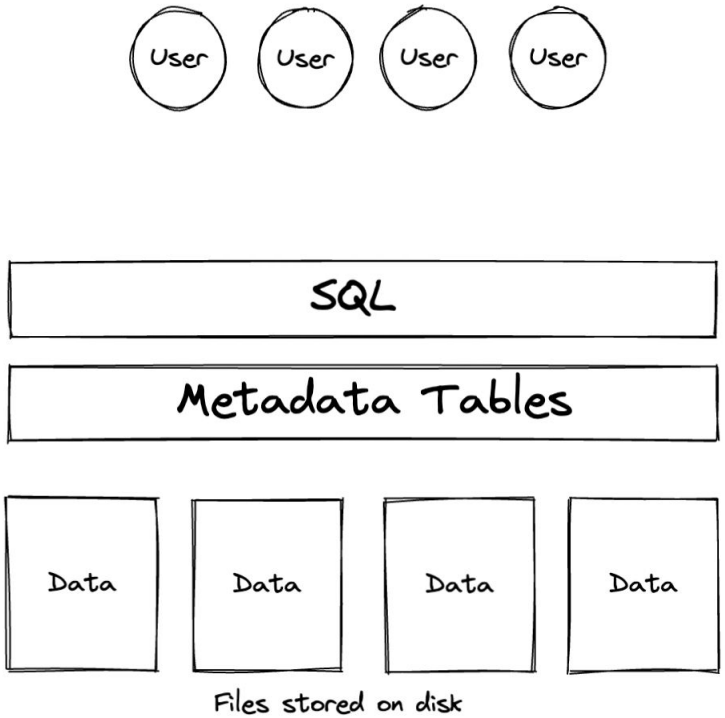




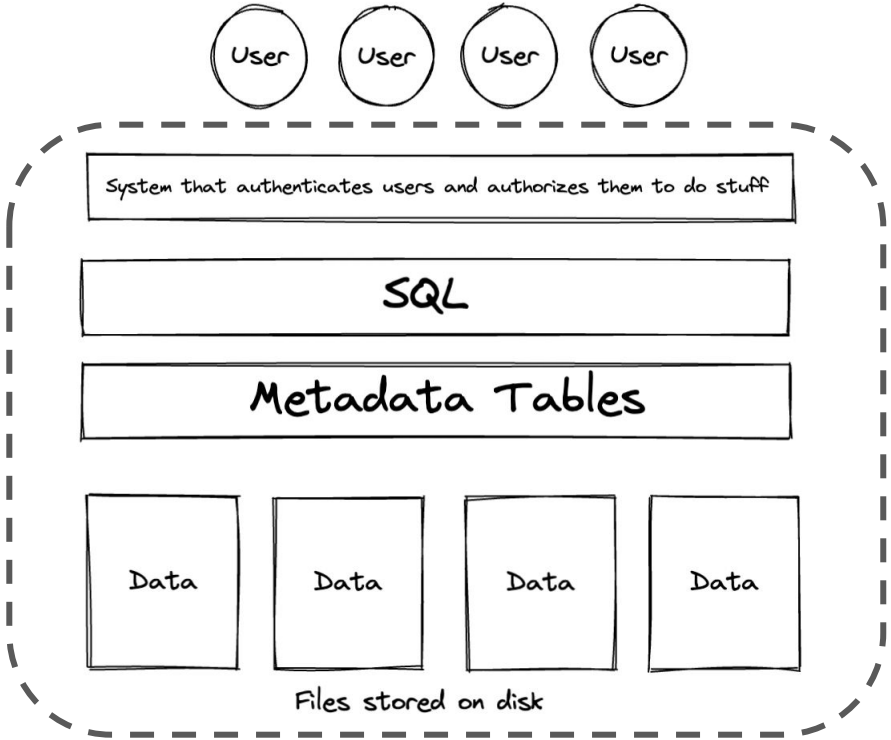
What do we need for cloud based analytics?



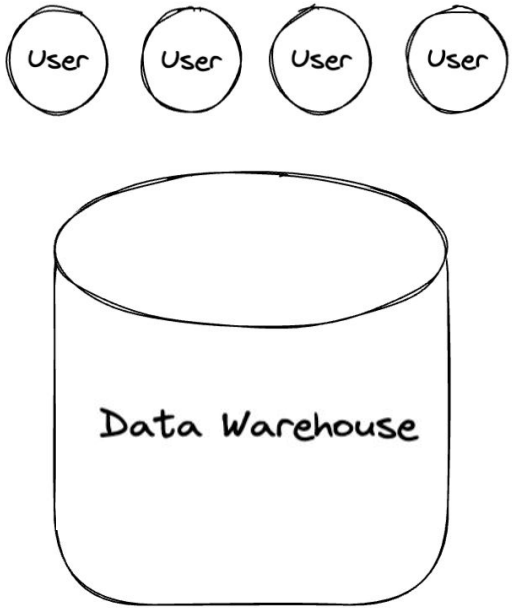
What do we need for cloud based analytics?



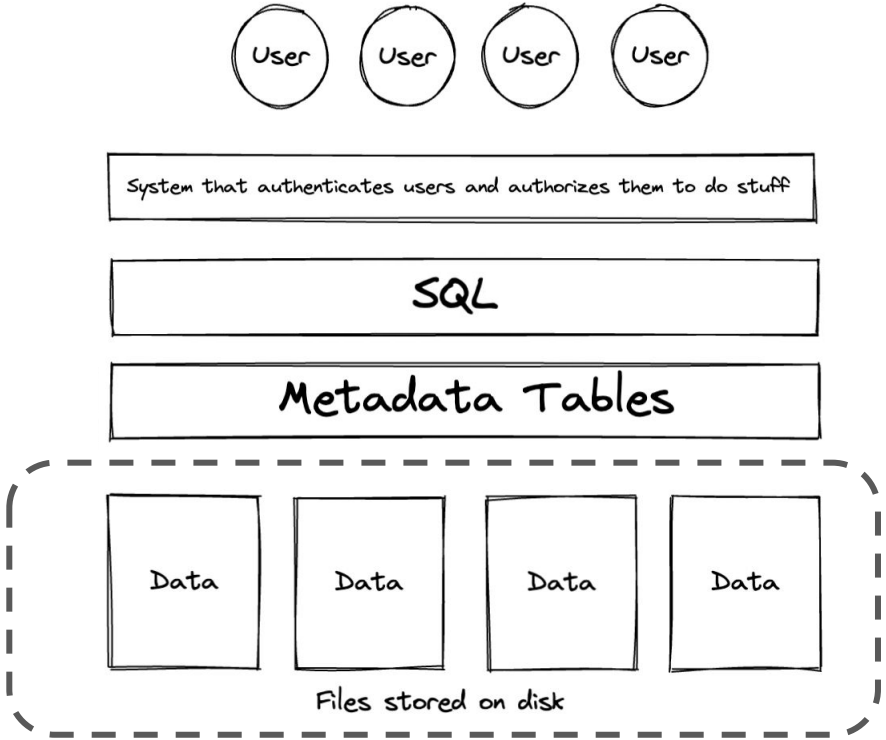
What do we need for cloud based analytics?



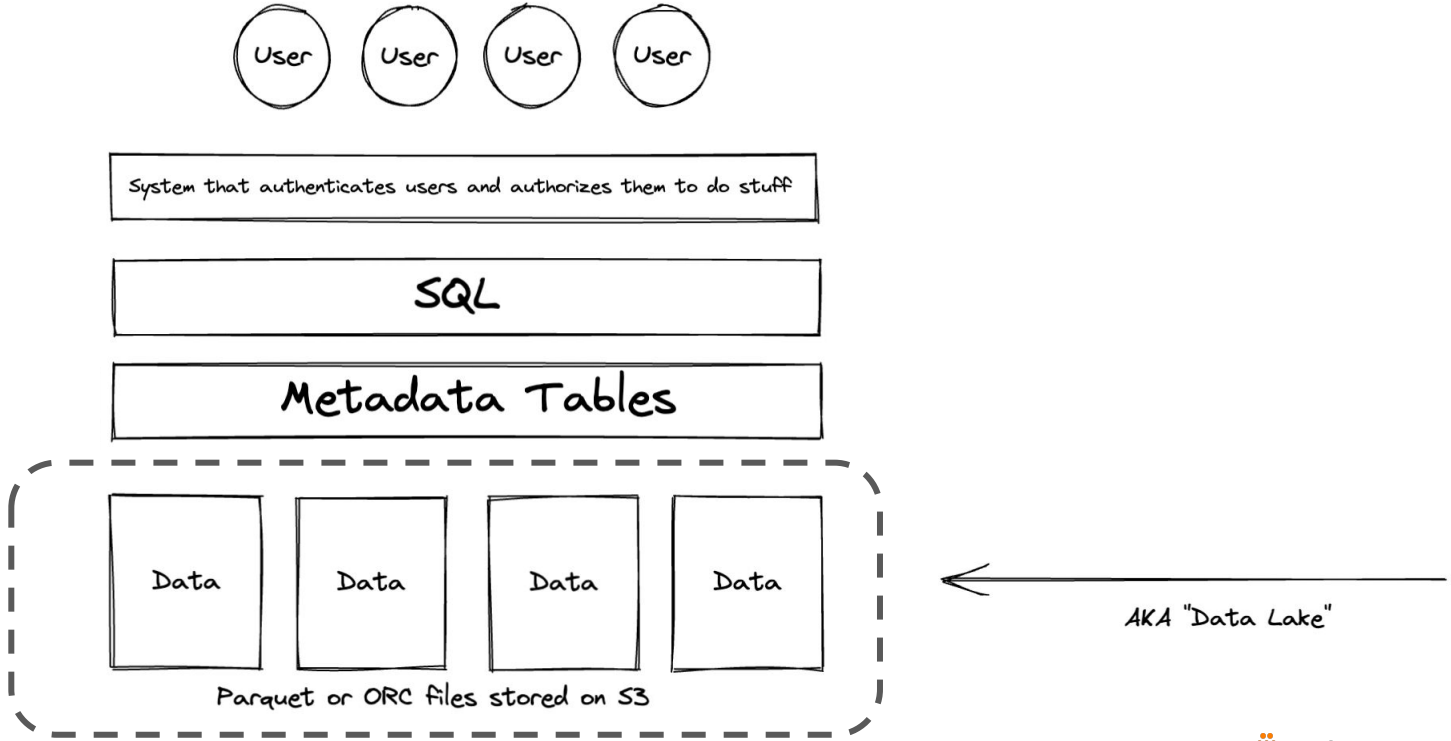
Cloud Data Warehouse is an answer



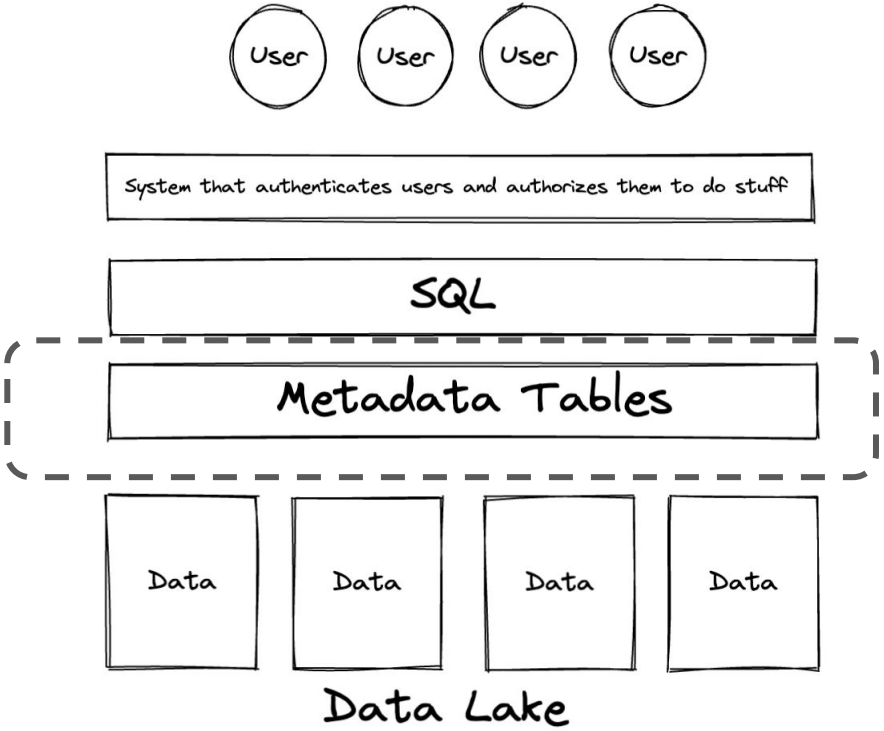
Where can we save money?



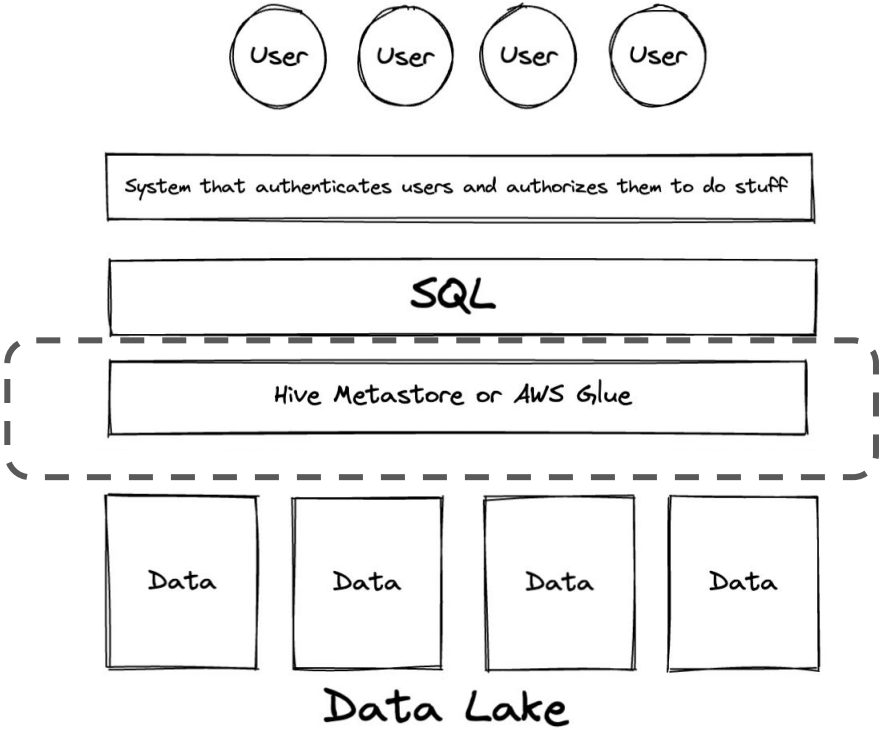
Where can we save money? Storage!



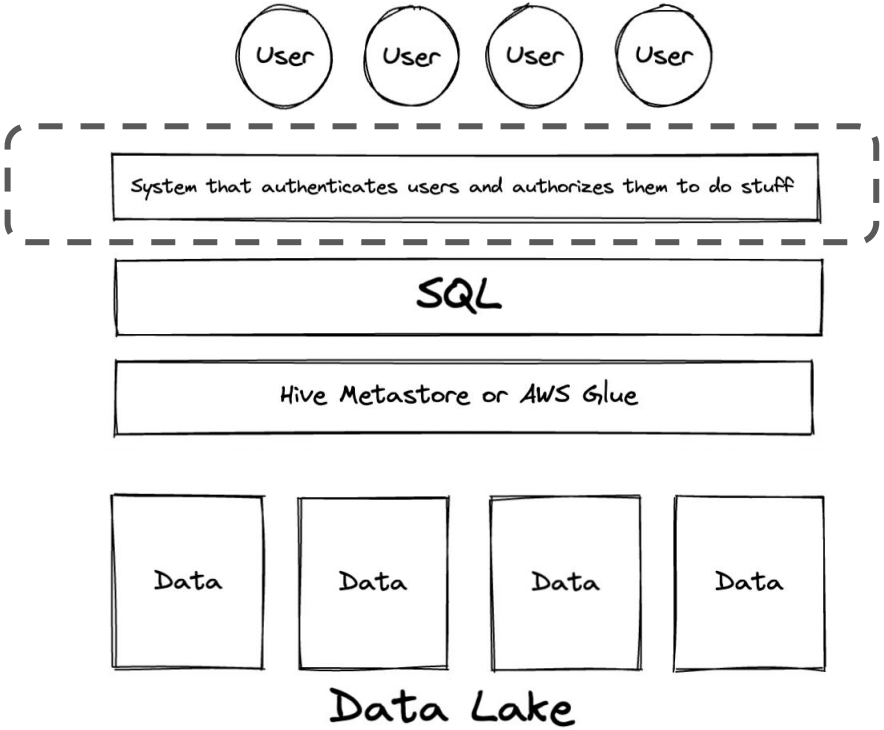
Metadata tables -> Catalog



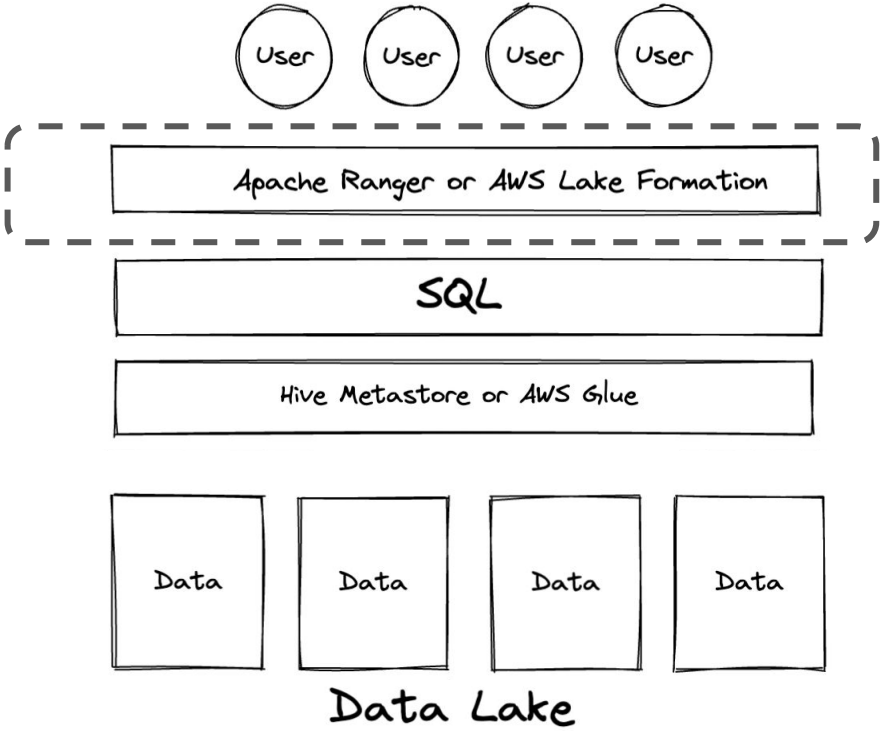
Metadata tables -> Catalog



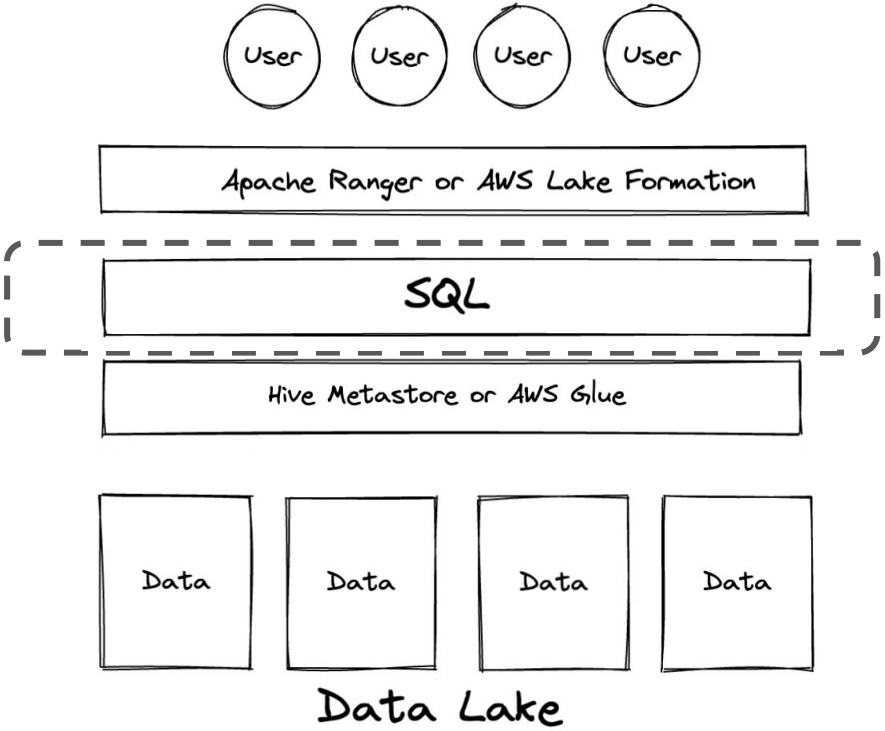
You probably already use something else here



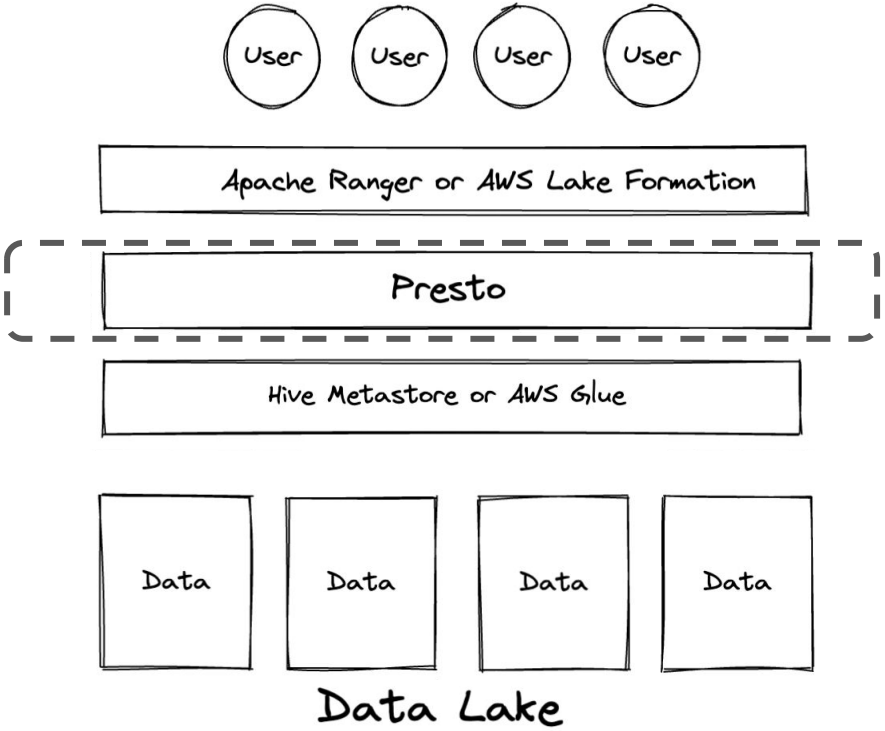
May as well be Open Source!



But what about SQL?

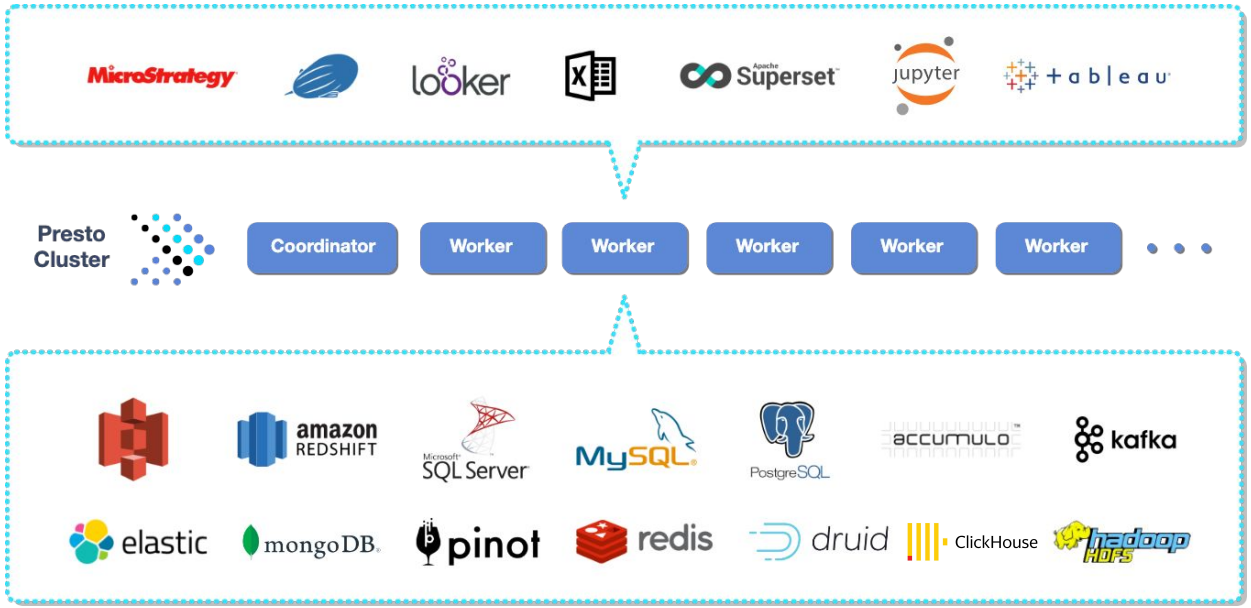


Welcome to the Open SQL Data Lakehouse!



What is Presto?

- Open source, distributed SQL query engine for the data lake & lakehouse
- Designed from ground up for fast analytic queries against data of any size
- Query in place - no need to move data
- Federated querying - join data from different source formats

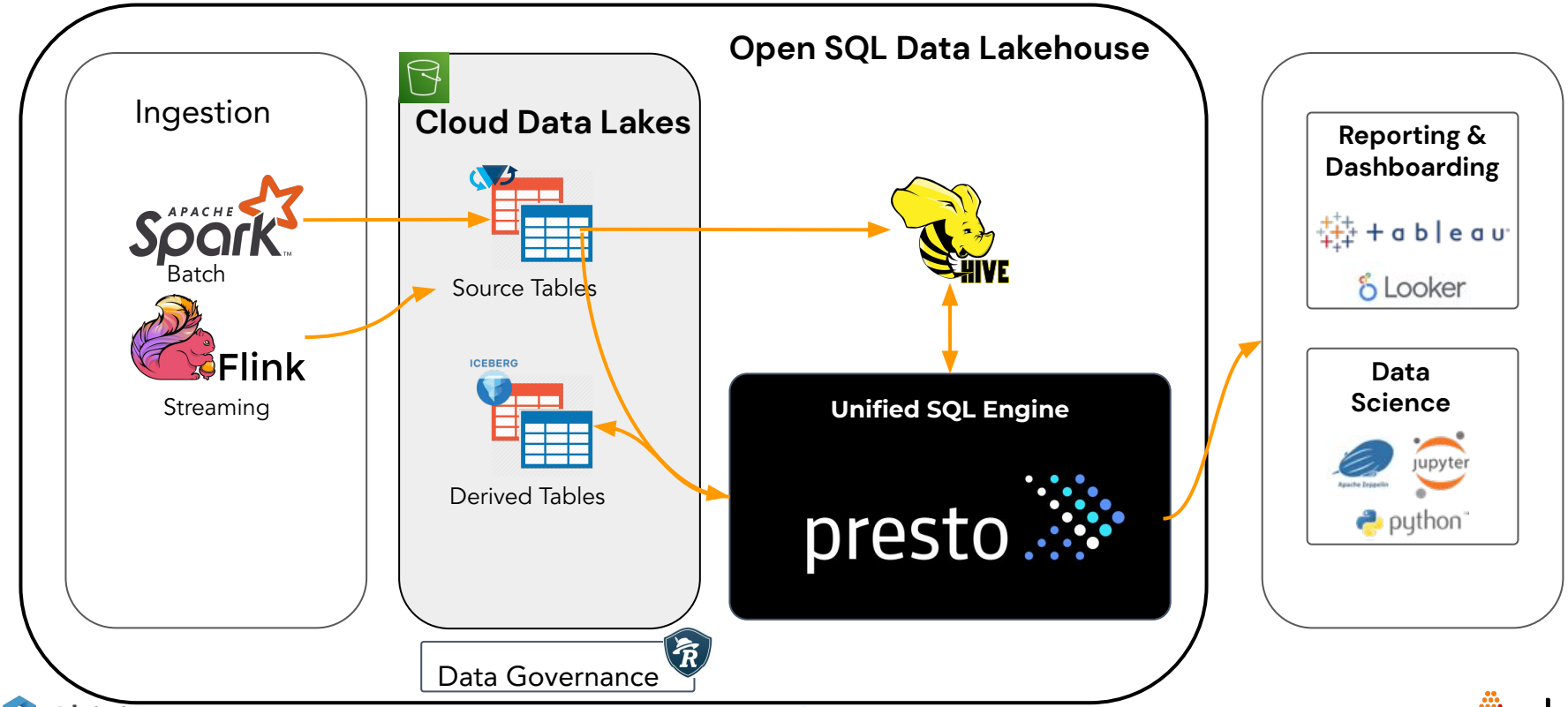


Outcome - Presto for Data Lake Analytics

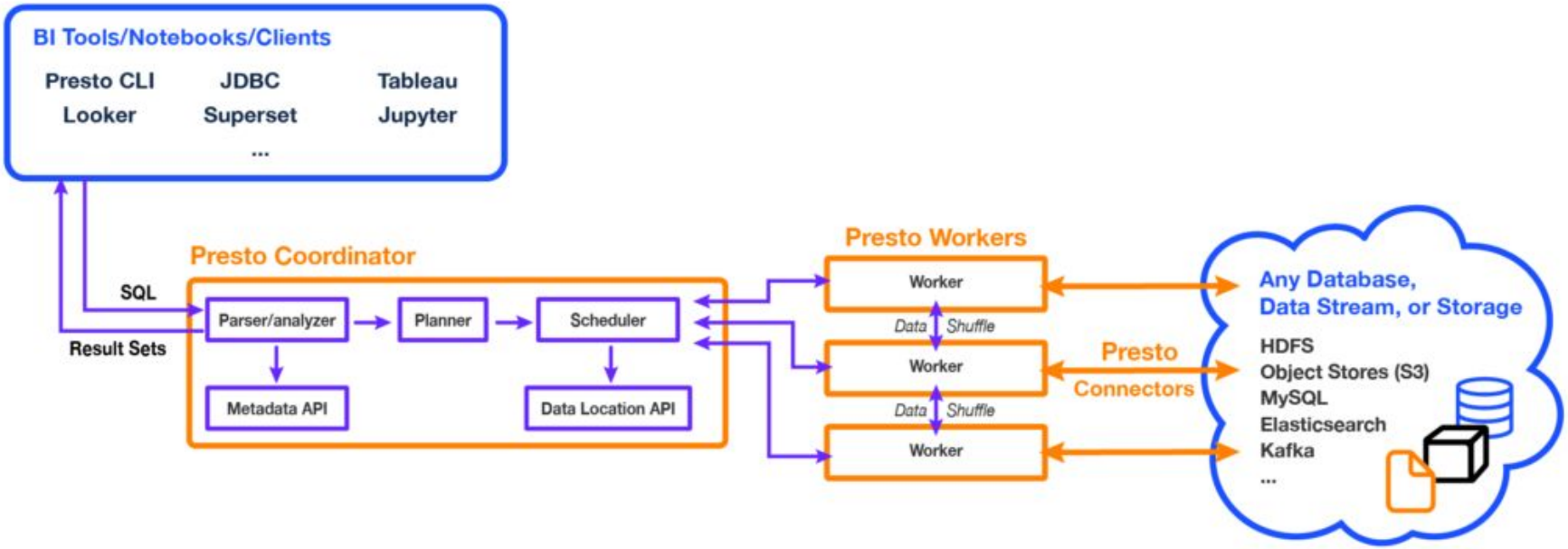
- Storage-Compute segregation
- Query Federation
- Unified SQL access
- Faster Onboarding and No Data Downtimes
- Better price performance

Let's look at an eCommerce app powered by Presto

Open SQL Data Lakehouse for eCommerce: Powered by Presto



Presto Scalable Architecture



Demo Time

1. Query S3 Data
2. Join AWS glue table and MySQL table with Presto

Real-time Analytics with ClickHouse

Real-time analytic challenges

- Load millions of rows per second from event stream fire hoses
- Fixed, low latency response to arbitrary slicing/dicing queries
- ~10ms response to requests from services
- Scale to very large datasets
- High cost efficiency

ClickHouse is a SQL Data Warehouse

Understands SQL

Runs on bare metal to cloud

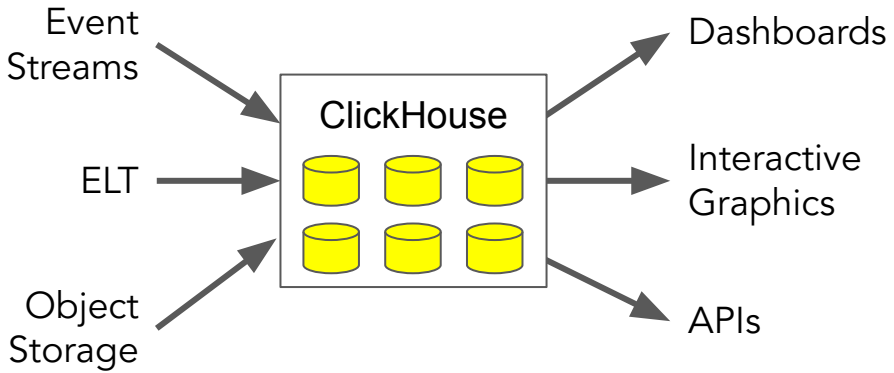
Shared nothing architecture

Stores data in columns

Parallel and vectorized execution

Scales to many petabytes

Is Open source (Apache 2.0)



It's a popular engine for real-time analytics

Seeing is believing

Demo Time!

Round up the usual performance suspects

Codecs

Data Types

Sharding

Read Replicas

Data Partitioning

Compression

Tiered Storage

In-RAM dictionaries

Skip Indexes

Projections

Vectorized Query

Primary key index

“One Big Table” design: multiple entities in a single table

Reading
- msg_type=’reading’
- sensor_id
- time
- temperature

Restart
- msg_type=’restart’
- sensor_id
- time

Error
- msg_type=’err’
- sensor_id
- time
- message

What does the sensor table look like?

```
CREATE TABLE IF NOT EXISTS readings_zstd (
  sensor_id Int32 Codec(DoubleDelta, ZSTD(1)),
  sensor_type UInt16 Codec(ZSTD(1)),
  location LowCardinality(String) Codec(ZSTD(1)),
  time DateTime Codec(DoubleDelta, ZSTD(1)),
  date ALIAS toDate(time),
  temperature Decimal(5,2) Codec(T64, ZSTD(10))
)
Engine = MergeTree
PARTITION BY toYYYYMM(time)
ORDER BY (location, sensor_id, time);
```

Optimized data types

Codecs + ZSTD compression

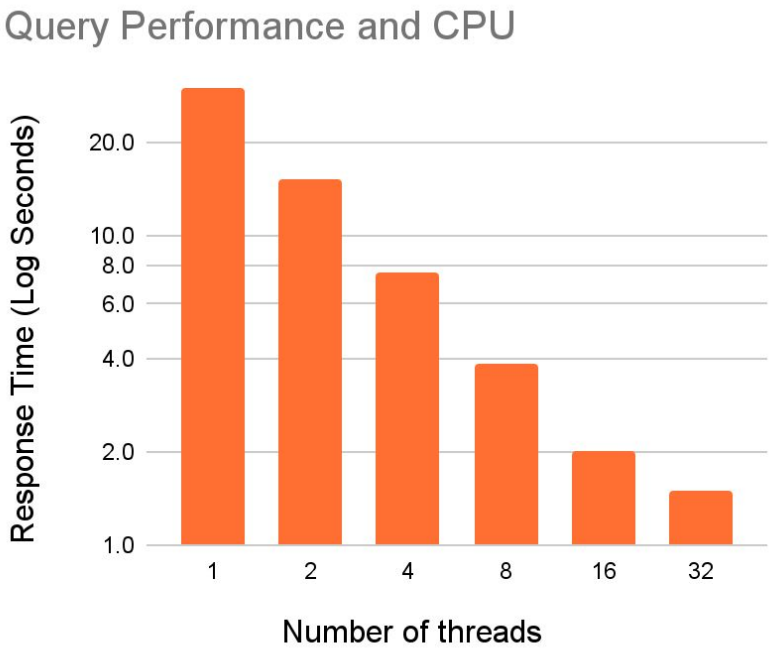
ALIAS column

Time-based partitioning

Sorting by key columns + time

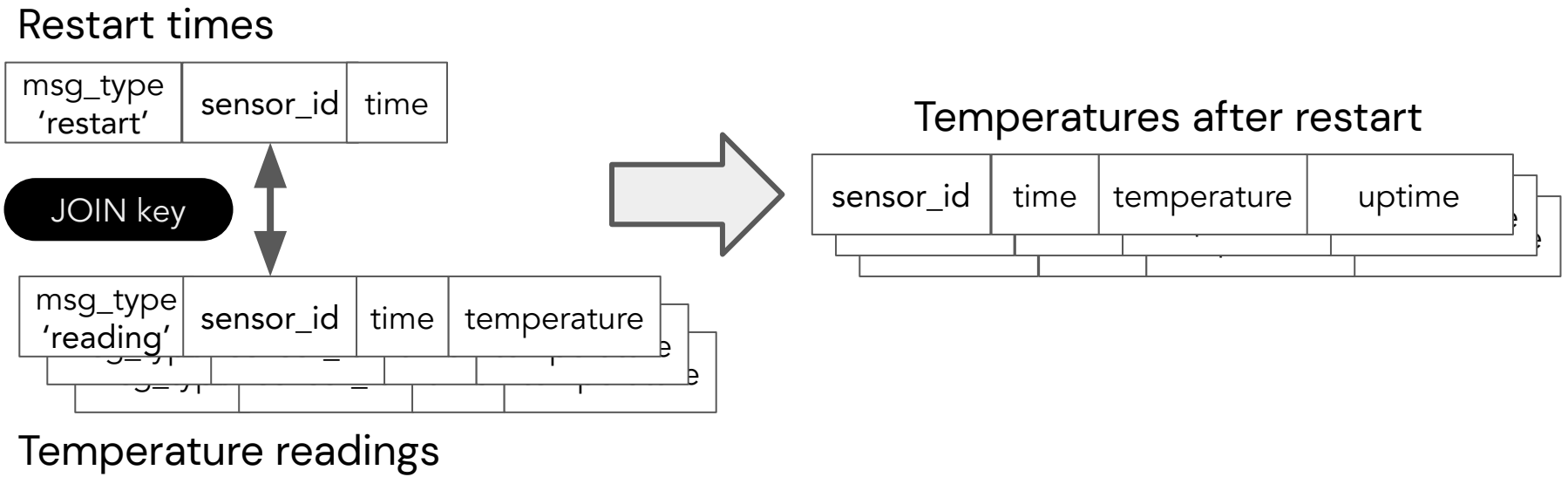
Linear query scaling using -If combinators

```
--Query over 1 Billion rows
set max_threads = 16;
SELECT
    toYYYYMM(time),
    countIf(msg_type = 'reading'),
    countIf(msg_type = 'restart'),
    min(temperature),
    round(avg(temperature)),
    max(temperature)
FROM test.readings_multi
WHERE sensor_id BETWEEN 0 and 10000
GROUP BY month ORDER BY month ASC;
```

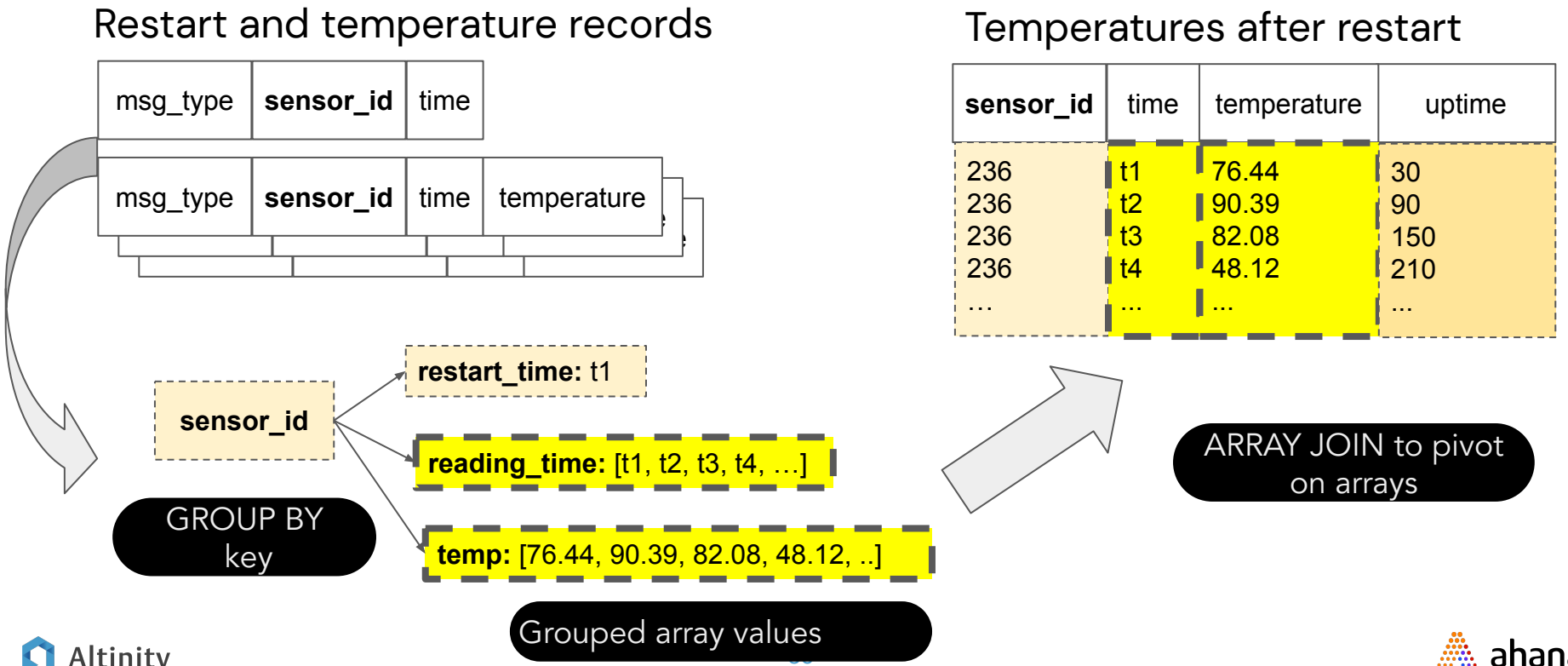


What about joins within a big table schema?

Use case: join restarts with temperature readings



Aggregation can implement joins!



Finding the last restart is an aggregation task!

sensor_id	time	msg_type
236	2019-01-10 20:00:13	restart

sensor_id	time	msg_type
236	2019-01-10 21:07:56	restart

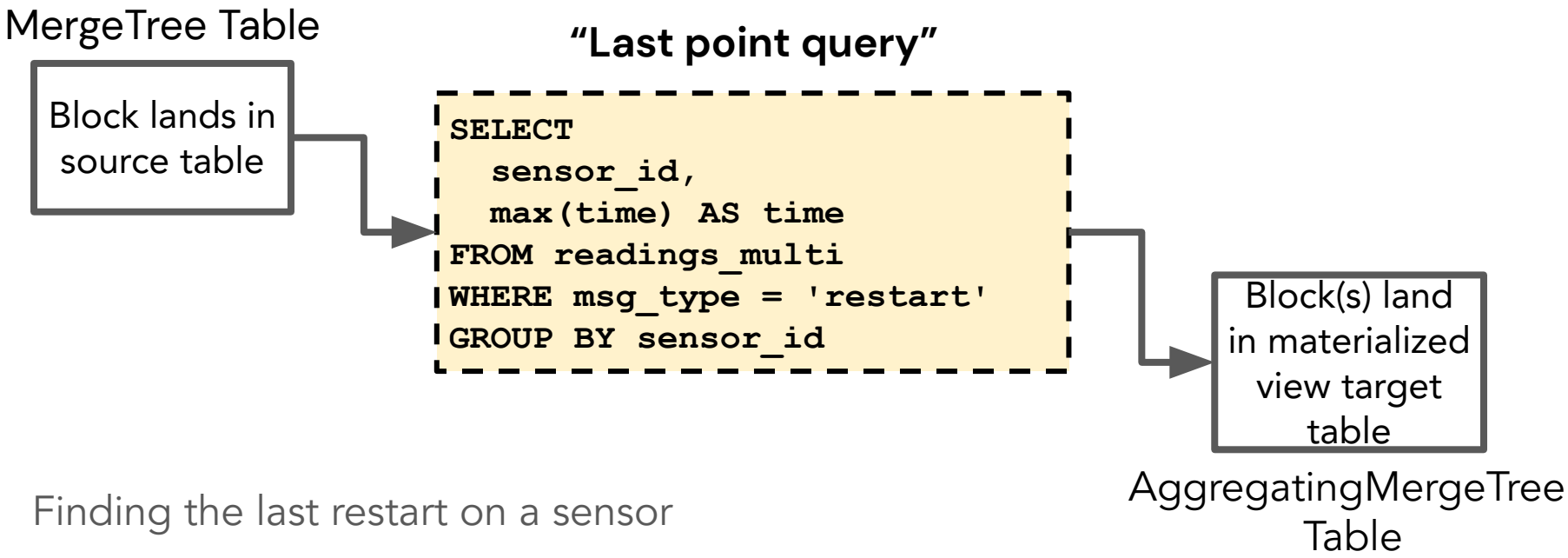
GROUP BY key

sensor_id	time	msg_type
236	2019-01-10 21:07:56	restart

Max value

Matching row value

Use materialized views to “index” data



Finding the last restart on a sensor

Outcome - ClickHouse for Real-Time Analytics

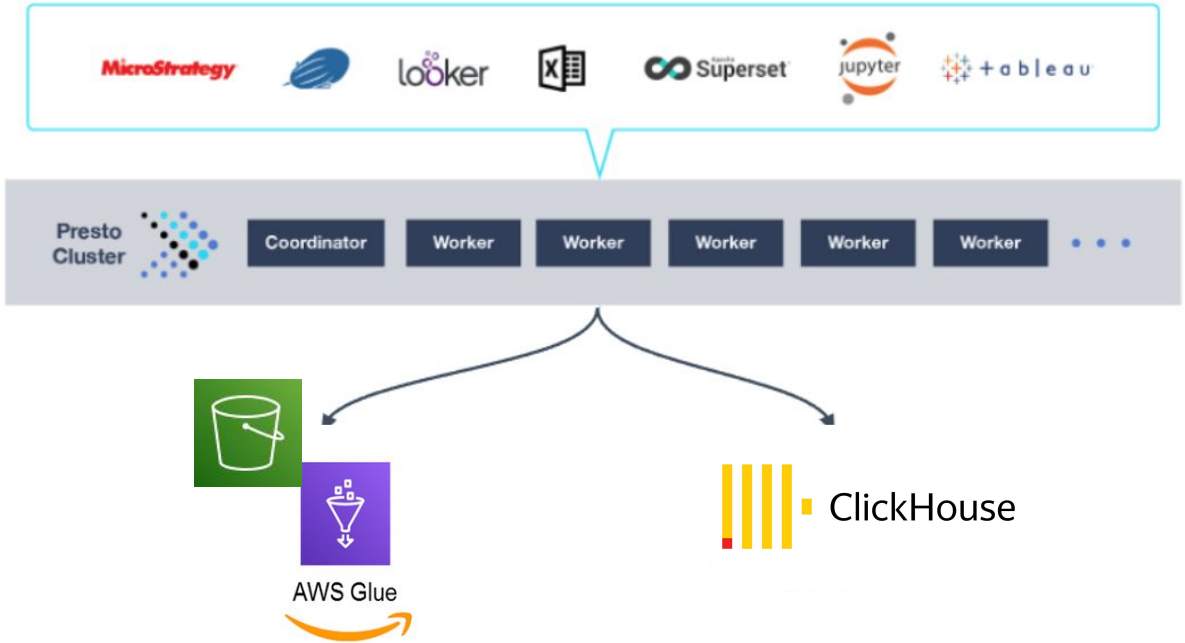
- Convenient integration to ingest: event streams, object storage, ELT, …
- Fast response on unaggregated source data
- Pre-aggregated response within time to render a web pages server-side
- Scale resources to maintain constant response
- Cost-efficient user-facing tenant APIs and visualization

Mixing and Matching

Data Lakes & Real Time Analytics

Let’s look at how you might deploy these architectures together

Presto Clickhouse Connector

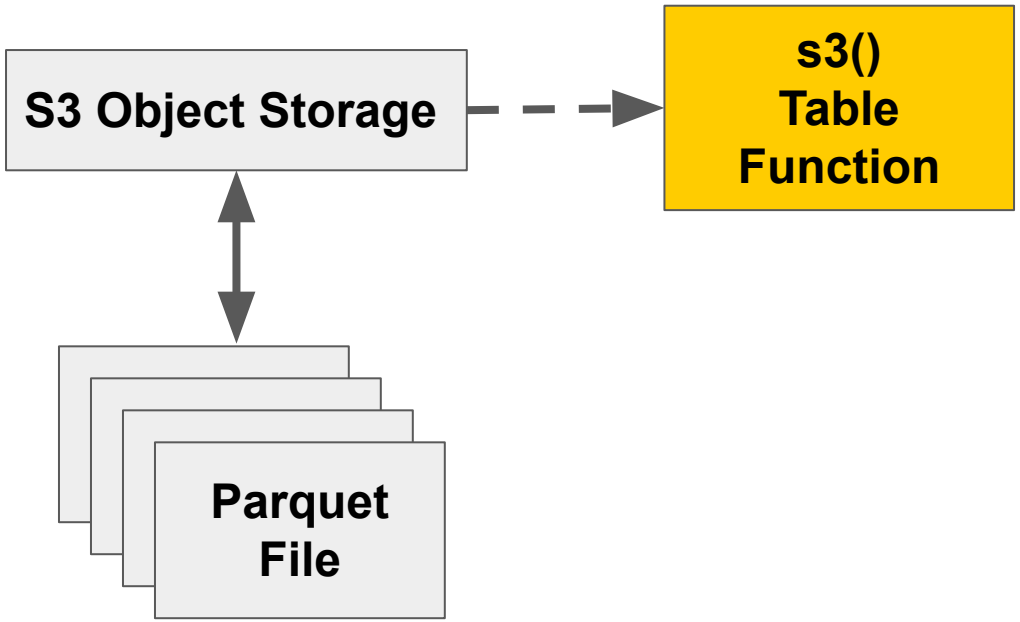


Query Federation

Join AWS glue table and Clickhouse Table with Presto

```
select name, sum(totalprice) as total
from clickhouse.ahana.customer AS c
LEFT JOIN
glue.ecom.orders AS o
ON c.custkey=o.custkey
GROUP BY name
ORDER BY total DESC LIMIT 10;
```

ClickHouse can read data from S3




```
SELECT
   max(a),
   sum(b)
FROM s3(...)
```

Example of reading Parquet data in ClickHouse

```
SELECT max(temperature), min(temperature)
FROM
s3('https://s3.us-east-1.amazonaws.com/.../readings*.parquet',
'Parquet')
WHERE sensor_type=1

max(temperature)|min(temperature)|
----------------+----------------+
          125.62|          -11.11|
```

Wrap-up

Summary points

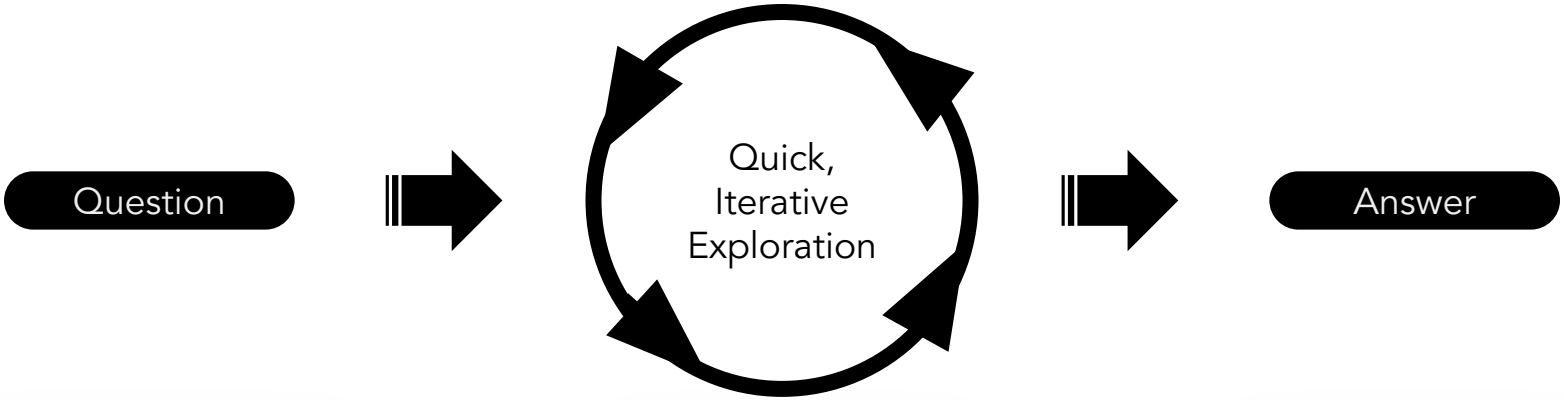
- Data lakes with Presto gives data engineers & data architects more flexibility, better price performance and 1 unified interface for their data
- Real-time analytics with ClickHouse offer fast reaction and constant query response on rapidly arriving data
- You can mix approaches, too
  - Deploy Clickhouse with Presto to get access to your real-time data along with your other data sources and data lakes
  - Read data lake files directly from ClickHouse

Thank you!
Questions?

https://altinity.com
Altinity.Cloud
Contact Altinity

https://ahana.io
Ahana Cloud
Contact Ahana

Real-time analytics in action: service log management



Why do node.js backends fail to process transactions?

Slice and dice queries on detailed log data

Bug introduced in latest upgrade

And here's the code…

```
SELECT sensor_id, reading_time, temp, reading_time,
  reading_time - restart_time AS uptime
FROM (
WITH toDateTime('2019-04-17 11:00:00') as start_of_range
SELECT sensor_id, groupArrayIf(time, msg_type = 'reading') AS
reading_time,
    groupArrayIf(temperature, msg_type = 'reading') AS temp,
    anyIf(time, msg_type = 'restart') AS restart_time
FROM test.readings_multi rm
WHERE (sensor_id = 2555)
  AND time BETWEEN start_of_range AND start_of_range + 600
GROUP BY sensor_id)
ARRAY JOIN reading_time, temp
```

Not everyone's cup of tea, but it works!!!

A famous data scientist on the subject of data…

It is a capital mistake to theorize before one has data.

Sherlock Holmes
(aka Arthur Conan Doyle)
A Scandal in Bohemia

ClickHouse Server Architecture