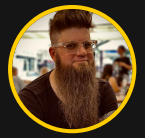
ClickHouse Jakarta Meetup

October 2024

ClickHouse

Tech Talks



An Introduction to ClickHouse and My Favourite Features

Tyler Hannan, Senior. Director, Developer Advocacy @ ClickHouse

Building the Backbone: Leveraging ClickHouse for Real-Time Data in High-Frequency Trading

Nahwin Rajan, Co-Founder & CEO In Stealth Mode

Driving Jakarta's Electric Motorcycle Transformation with ClickHouse

Andi Pangeran, Principal Software Engineer @ Electrum

An Introduction to ClickHouse

And my favourite features

2024

ClickHouse

What is ClickHouse?

Open source

Developed since
2009 - OSS 2016
31k+ Github stars
1.3k+ contributors
500+ releases

column-oriented

Best for aggregations
Files per column
Sorting and indexing
Background merges

distributed

Replication
Sharding
Multi-master
Cross-region

OLAP database

Analytics use cases
Aggregations
Visualizations
Mostly immutable data

Key Features

Some of the cool things ClickHouse can do

1 Speaks SQL

Most SQL-compatible UIs, editors, applications, frameworks will just work!

2 Lots of writes

Up to several million writes per second - per server.

3 Distributed

Replicated and sharded, largest known cluster is 4000 servers.

4 Highly efficient storage

Lots of encoding and compression options - e.g. 20x from uncompressed CSV.

5 Very fast queries

Scan and process even billions of rows per second and use vectorized query execution.

6 Joins and lookups

Allows separating fact and dimension tables in a star schema.

Database for Interactive Experiences

Make any data fast

User-facing Analytics	Internal Analytics	Observability
Spotify	Walmart	CLOUDFLARE
Bloomberg	Deutsche Bank	GitLab
twilio	lyft	NETFLIX
CONTENTSQUARE	PostHog	Disney+

Benchmarks

- Our own at https://clickhouse.com/benchmark/dbms/
  - 6x faster than Vertica
  - 25x faster than Greenplum
  - 50x faster than TimescaleDB
- Others, see https://github.com/ClickHouse/ClickHouse/issues/22398 e.g.:
  - 3-5x faster than RedShift (sometimes 50x)
  - 5-6x faster than Elasticsearch
  - 6-10x faster than Druid
- Storage
  - Ebay: 90% less hardware than Druid
  - Contentsquare: 6x more data than Elasticsearch

Creating tables

ClickHouse SQL Basics

```
CREATE TABLE events
(
    timestamp DateTime,
    tenant    LowCardinality(String),
    type      Enum8(...),
    value     Float64,
    attr      Array(String),
    flags     Map(String, Boolean)
)
ENGINE = MergeTree
ORDER BY (tenant, timestamp)
```

- Other engines:
  - ReplacingMergeTree
  - CollapsingMergeTree
  - AggregatingMergeTree
- Integration engines:
  - Kafka, RabbitMQ
  - MySQL, PostgreSQL, MongoDB
  - JDBC, ODBC
  - S3, HDFS
  - EmbeddedRocksDB
- Adding Replicating- in front makes an engine replicate data (e.g. ReplicatingMergeTree)

Inserting directly

ClickHouse SQL Basics

```
INSERT INTO people VALUES ('Obi-Wan Kenobi', 57, ...) ('Yoda', 900, ...)
(...)
```

- Batching is important! Either batch yourself or turn on asynchronous inserts:

```
SET async_insert = true
INSERT INTO people VALUES ('Obi-Wan Kenobi', 57, ...)
INSERT INTO people VALUES ('Yoda', 900, ...)
INSERT INTO people VALUES (...)
```


Inserting from external sources

ClickHouse SQL Basics

- ClickHouse has many built-in table functions to read external data:

```
INSERT INTO table SELECT * FROM s3(...)
INSERT INTO table SELECT * FROM file(...)
INSERT INTO table SELECT * FROM url(...)
INSERT INTO table SELECT * FROM mysql(...)
INSERT INTO table SELECT * FROM postgresql(...)
INSERT INTO table SELECT * FROM jdbc(...)

And more!
```

- Read the docs at clickhouse.com/docs!

Reading data

ClickHouse SQL Basics

- Don’t do this! (at least not too often)

```
SELECT * FROM table
```

- And also not this! (at least not too often)

```
SELECT ... FROM table WHERE id = '<uuid>'
```

- The majority of your queries should be:

```
SELECT avg(something) FROM table WHERE toYear(timestamp) = 2022
```

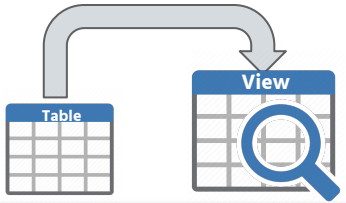

Views

ClickHouse SQL Basics

- Saving a query as a view (no data movement):


```
CREATE VIEW view AS SELECT ... FROM table ...
```

- Continuously processing new data from a table into another table:

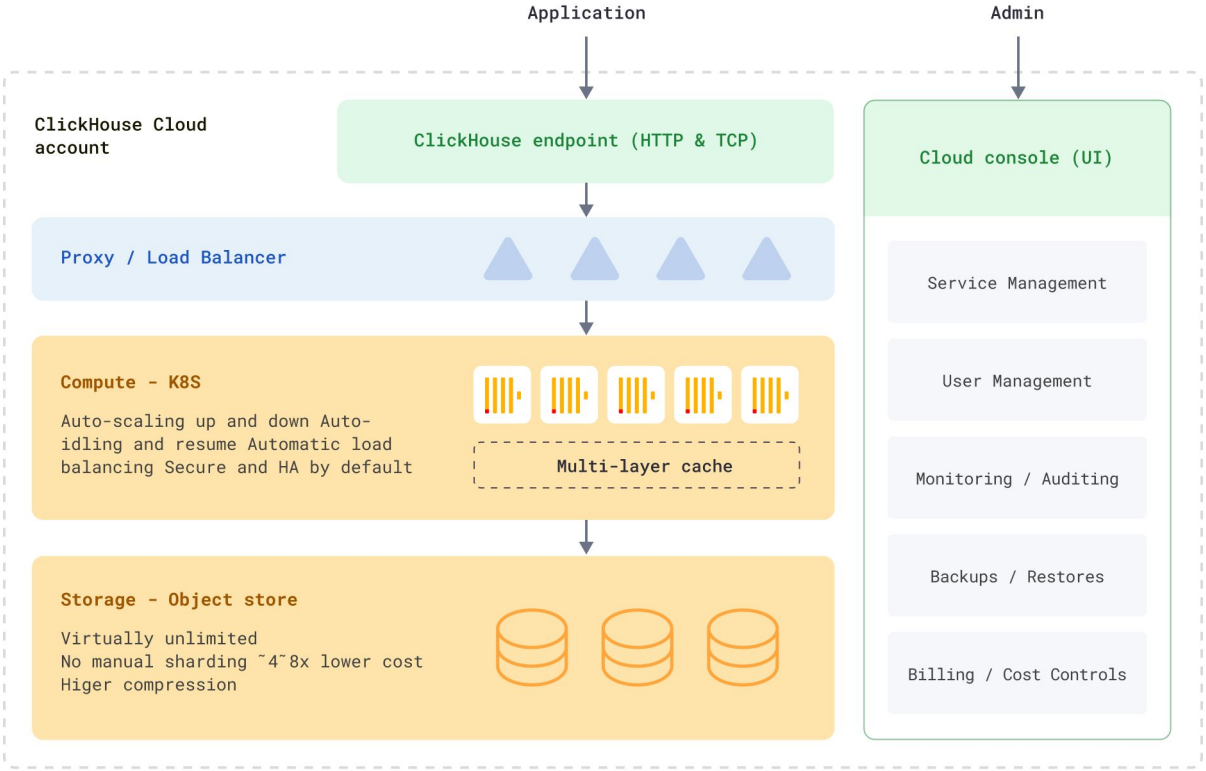



```
CREATE MATERIALIZED VIEW view AS SELECT avg(...) FROM table GROUP BY ...
```

Ecosystem and Integrations

- Interface
  - HTTP, native TCP, CLI
  - JDBC, ODBC
  - Client libraries: Go, Python, Javascript, etc.
- Data ingest
  - Kafka, S3, HDFS, RabbitMQ (native)
  - MySQL, PostgreSQL, MongoDB, Redis (native)
  - Vector (logs and metrics)
  - Airbyte (ELT)
  - JDBC, ODBC (Spark, Flink, etc.)
- Query editor
  - Open source: DBeaver, VS Studio
  - Proprietary: DataGrip
- Visualization
  - Open source: Grafana, Superset, Metabase
  - Proprietary: Tableau, Power BI

ClickHouse Cloud Architecture



For more, go to clickhouse.com/learn

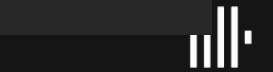
OVERLAY

```
SELECT overlay('Hello, world!', 'test', 8, 5) AS res
```

```
   ┌─res─────────┐
1. │ Hello, test! │
```

```
WITH 'Hello, world!' AS s,
     'test' AS replacement,
     8 AS pos,
     5 AS length
SELECT concat(substring(s, 1, pos - 1), replacement, substring(s, pos + length))
AS res
```

```
   ┌─res─────────┐
1. │ Hello, test! │
```

JSON Object Format

Saved Queries as APIs