ClickHouse

How Open Source and ClickHouse are Changing the Cloud Data Warehouse Landscape

Hellmar Becker
ClickHouse

Aug 2025

ClickHouse

About me

- Solution Architect @ ClickHouse
- 25 years in analytics
- Love open source 🧡

ClickHouse

Table of contents

ClickHouse

ClickHouse

01 History of Data Warehousing

The Unbundling of the Cloud Data Warehouse

Tanya Bragin
Nov 7, 2023

We owe a lot to the cloud data warehouses, but their era of hegemony is coming to an end.

Over the last 10 years, companies like Snowflake modernized a whole industry, which previously relied on a closed and proprietary ecosystem of self-managed deployments (powered by Oracle, Teradata, and the like). They enabled organizations to move petabytes of critical workloads to the cloud, opening up these datasets to a wider range of integrations, collaboration, and applications – democratizing access to data and dramatically increasing its value.

https://clickhouse.com/blog/the-unbundling-of-the-cloud-data-warehouse

The History of Data Warehousing

Traditional data warehouses signaled the dawn of “big data”

30 years ago

10 years ago

Present

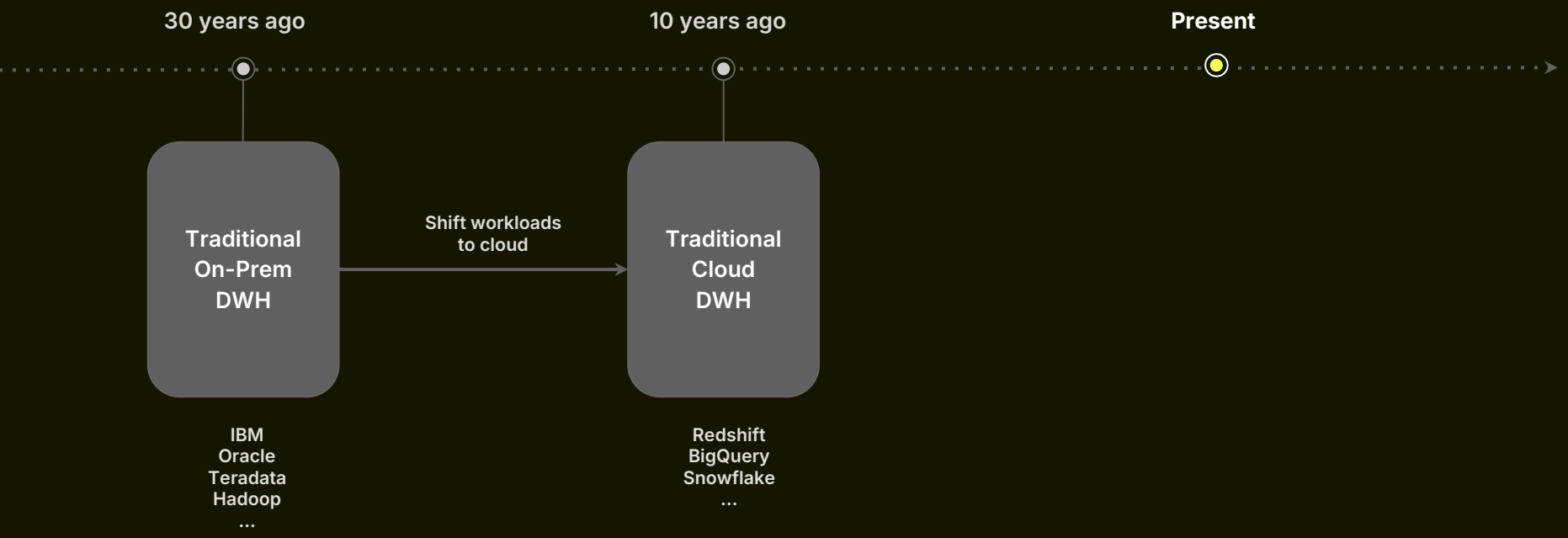
Traditional
On-Prem
DWH

IBM
Oracle
Teradata
Hadoop
...

https://clickhouse.com/blog/the-unbundling-of-the-cloud-data-warehouse

The History of Data Warehousing

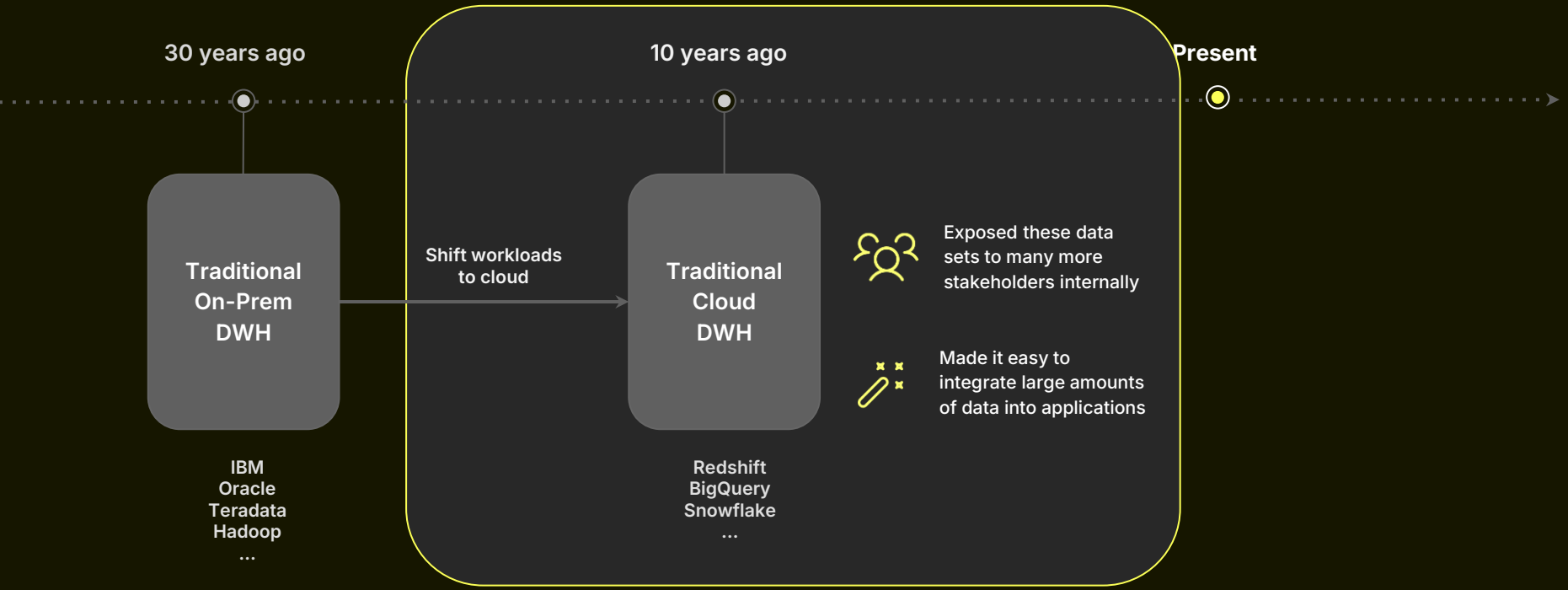
Cloud data warehouses democratized “big data” workloads



https://clickhouse.com/blog/the-unbundling-of-the-cloud-data-warehouse

The History of Data Warehousing

Cloud data warehouses democratized “big data” workloads



https://clickhouse.com/blog/the-unbundling-of-the-cloud-data-warehouse

What are real-time analytical applications?

ClickPy | chdb | 2011-01-01 to 2024-09-28 | Powered by ClickHouse

chdb

Author: auxten
Author Email: auxten@clickhouse.com
License: Apache-2.0
Summary: chDB is an in-process SQL OLAP Engine powered by ClickHouse

135 # Pull requests
12 # PR request openers
108 # Issues
66 # Forks
1.9k (+1870)

Downloads by period: 971 last day, 17K last week, 58K last month, 451K total
All version, 2.0.4 latest version

Downloads over time

Top versions: 1.2.1, 1.4.1, 1.1.0, 0.15.0, 1.2.0, 1.3.0

clickpy.clickhouse.com

ClickHouse

ClickPy

Powered by ClickHouse

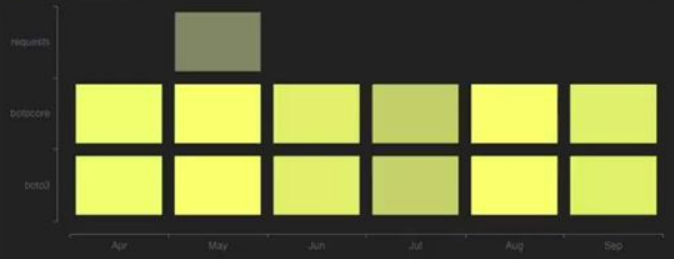
Analytics for PyPI packages

Browse through 696,572 Python packages from PyPI and over **1.10 trillion** downloads, updated daily

Search for a package

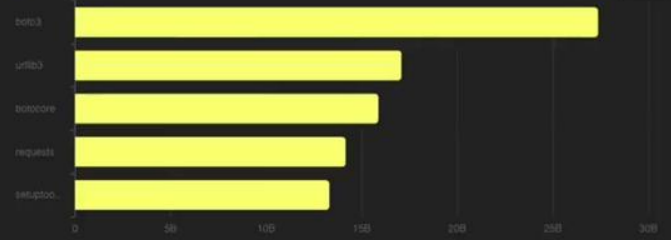
Recent releases

On popular packages



Top Repos

88B downloads



Emerging repos

Top 7

- **insight-cli** - 6.3M downloads in the last 3 months
- **sagemaker-core** - 3M downloads in the last 3 months
- **sparganothis-vim** - 1.9M downloads in the last 3 months
- **langgraph-checkpoint** - 615K downloads in the last 3 months
- **mtmdb** - 608K downloads in the last 3 months
- **primp** - 585K downloads in the last 3 months
- **bio-grumpy** - 454K downloads in the last 3 months

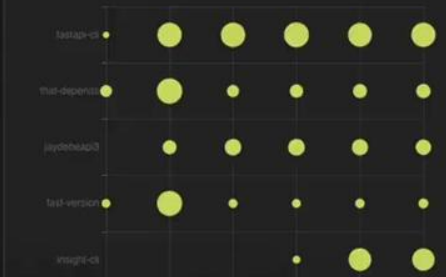
Needing a refresh

Top 7

- **urllib3** - 17B downloads, last updated on 18 February 2024
- **setuptools** - 13B downloads, last updated on 13 March 2024
- **six** - 11B downloads, last updated on 05 May 2021
- **python-dateutil** - 11B downloads, last updated on 01 March 2024
- **typing-extensions** - 10B downloads, last updated on 25 February 2024
- **charset-normalizer** - 9.5B downloads, last updated on 01 November 2023
- **wheel** - 9.4B downloads, last updated on 11 March 2024

Hot packages

96M downloads

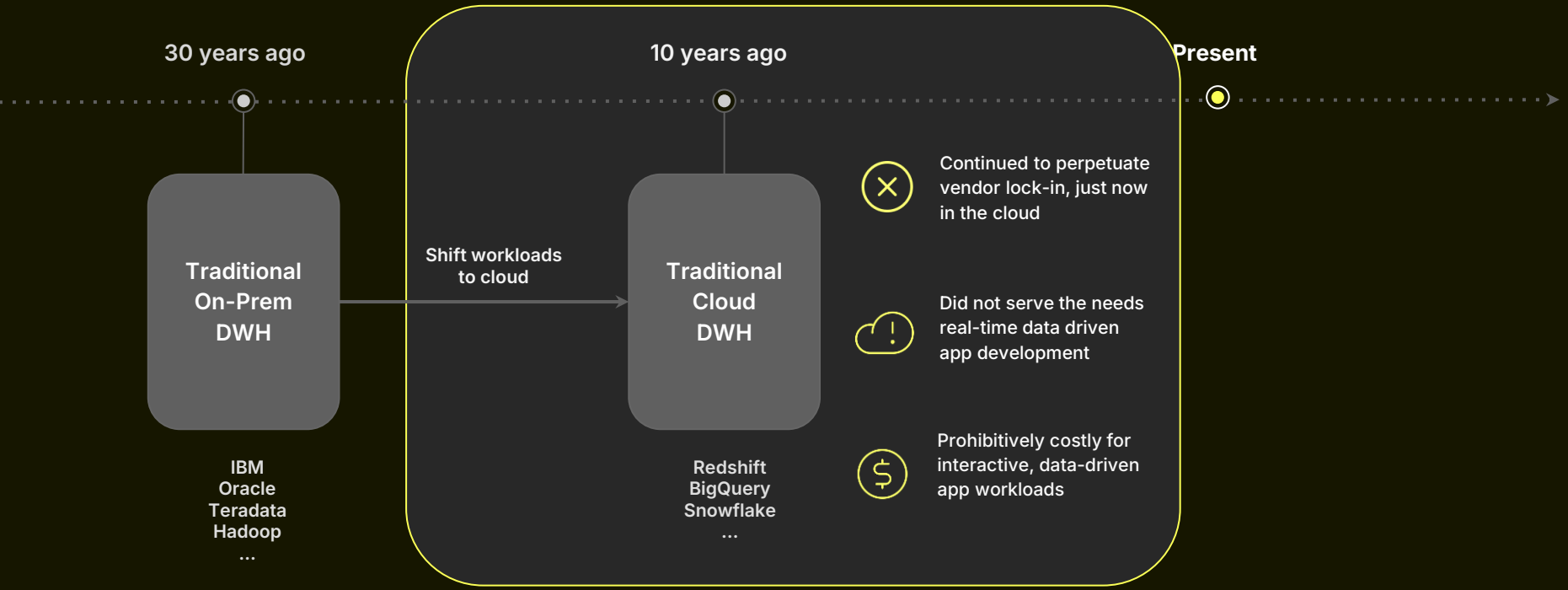


clickpy.clickhouse.com

ClickHouse

The History of Data Warehousing

... but also created problems



https://clickhouse.com/blog/the-unbundling-of-the-cloud-data-warehouse

The History of Data Warehousing

Meanwhile open-source directions in data ecosystem continue to evolve

30 years ago

10 years ago

Present

Traditional On-Prem DWH

IBM
Oracle
Teradata
Hadoop
...

Shift workloads to cloud

Traditional Cloud DWH

Redshift
BigQuery
Snowflake
...

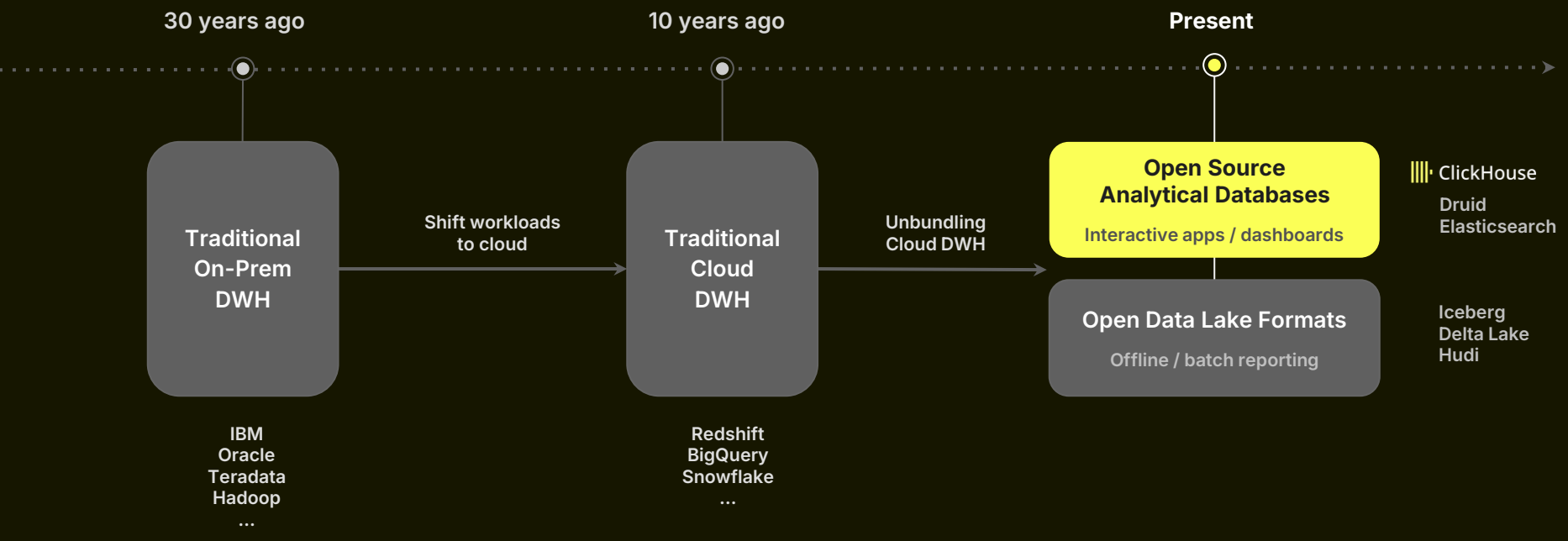
Open-source alternatives for serving real-time apps emerged

Open data lake formats evolved for cold / warm storage

https://clickhouse.com/blog/the-unbundling-of-the-cloud-data-warehouse

The History of Data Warehousing

Leading to unbundling of the cloud data warehouse in the present

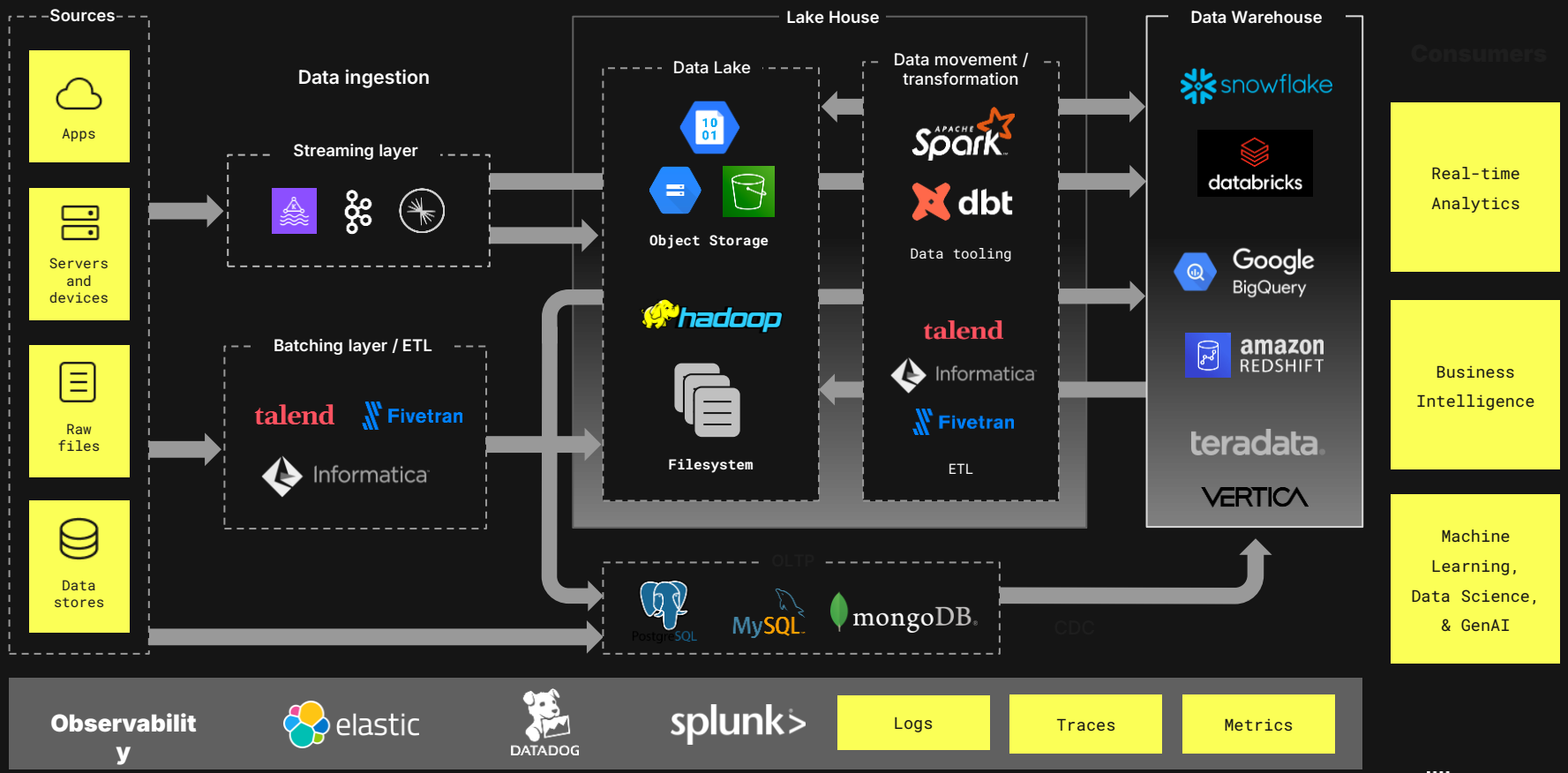


https://clickhouse.com/blog/the-unbundling-of-the-cloud-data-warehouse

ClickHouse

02 What are Data Lakes?

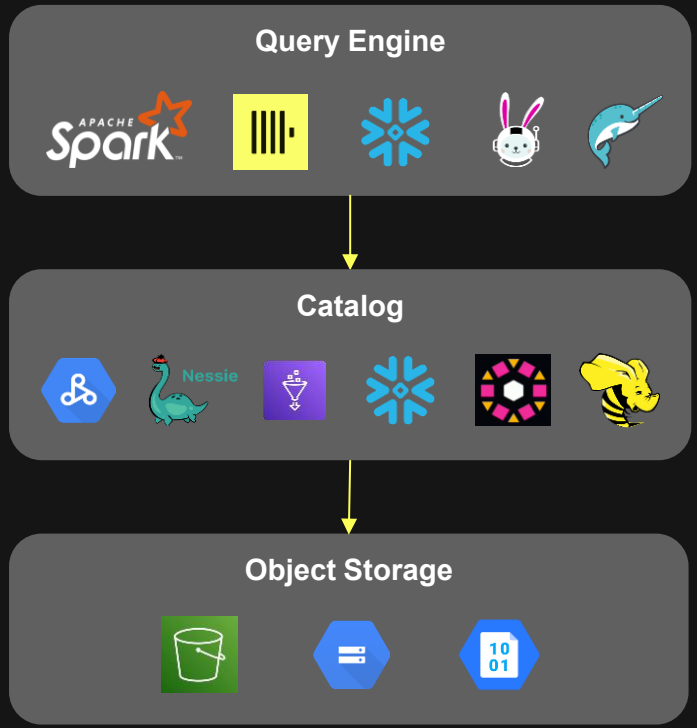
Typical Data Stack



ClickHouse

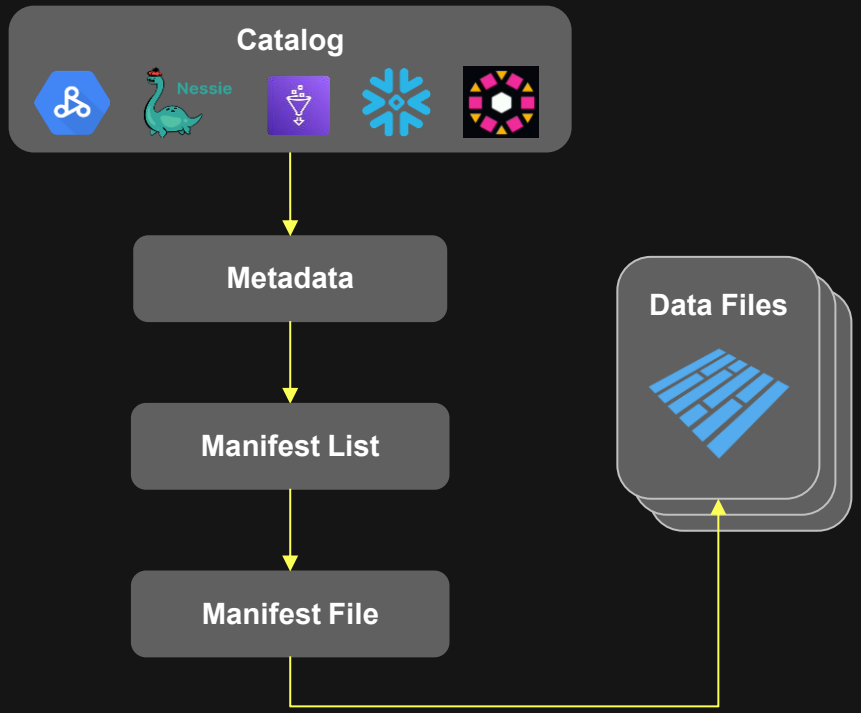
What is a data lake?

- The goal is to have many different query engines can connect to a catalog and query the underlying format
- Compatible with many different object storages
- Allows for open data sharing between disparate vendors and technologies



What is a data lake?

- **Table Format**
  Allows to view a collection of data files as a unified table
- **Table Management**
  Snapshots, Schema evolution, deletes
- **Data Catalog**
  Listing of all data assets stored in the data lake



What are open data lake formats?

Let’s ask ChatGPT

Data lake formats

File formats

Alternatives Comparison Table

Format	Type	Key Features	Strengths	Use Cases
Apache Iceberg	Table format	Schema evolution, partitioning, time travel	Supports large datasets, versioning	Data lakes, big data analytics
Delta Lake	Storage layer	ACID transactions, batch and streaming support	Combines data lake and warehouse benefits	Reliable ETL, data pipelines
Apache Hudi	Storage format	Incremental data processing, ACID transactions	Efficient upserts and deletes	Real-time data ingestion, ETL
Parquet	Columnar	High performance, efficient compression	Fast querying for analytics	Data warehousing, BI
ORC	Columnar	Optimized for Hadoop, effective for Hive	High read performance for large datasets	Big data processing, Hive
Avro	Row-based	Schema evolution, data serialization	Good for streaming and serialization	Data serialization, messaging

Offer more features on top of file formats

- Snapshots
- Schema evolution
- Time travel
- …

ClickHouse

ClickHouse

02 What are Analytical Databases?

What are analytical databases?

Let’s ask ChatGPT

- **Open source analytical databases**: Optimized for fast querying and analysis of large datasets, often using columnar storage.
- **Key features**: High-speed querying, scalable, cost-effective, and often community-driven.
- **Use cases**: Data warehousing, business intelligence, reporting, and real-time analytics.

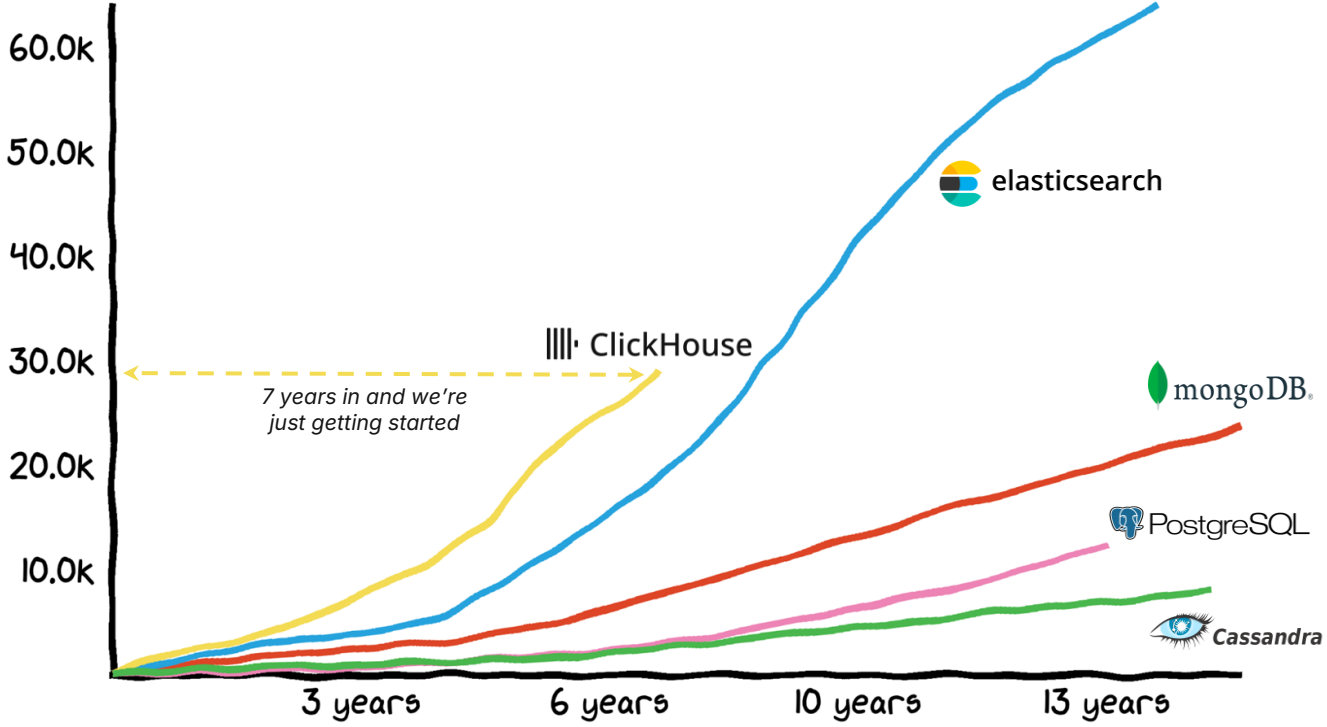
Best and fastest example:

- **ClickHouse**: Known for its industry-leading speed, real-time query performance, and scalability.

Comparison:

- **Legacy data warehouses (e.g., Teradata, IBM Netezza)**: Expensive, slower for large-scale queries, harder to scale, and often on-premise.
- **Cloud data warehouses (e.g., Snowflake, Google BigQuery)**: More flexible, better scalability, and simpler to manage, but can become expensive at scale.

ClickHouse adoption



ClickHouse OpenSource

- 36.9k Stars
- 6.8k Forks
- 1.6k Contributors
- 349 Active contributors
- 7.9k Slack members

ClickHouse Cloud

- Processed hundreds of millions of queries
- Thousands of cloud trial and paid customers since Dec 2022

What is ClickHouse?

Open source

- Developed since 2009
- OSS 2016
- 27k+ Github stars
- 1.6k+ contributors
- 300+ 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

We're powered by seriously big data

Ahrefs collects, processes, and stores large amounts of data for search marketing professionals.

Similar to search engines, we crawl the entire web 24/7 and store petabytes (1PB = 1000TB) of information about live websites – like how they link to each other and what keywords they rank for in search results.

We're also really proud of our backend infrastructure. Since no existing solution could keep up with the volume of data we operate, most of our software infrastructure was built in-house.

4300
Servers

658K
CPU cores

4PB
RAM

33PB
HDD

476PB
SSD

$900M
The amount we'd have to spend on the cloud in 3 years without our own infrastructure.

ClickHouse Use Cases

CLOUDFLARE

> ClickHouse helps us efficiently and reliably analyze logs across **trillions of Internet requests** to identify malicious traffic and provide customers with rich analytics

Real-time analytics datastore

Adevinta

> In comparison, BigQuery was **less performant and 2x more expensive** due to its pricing model that charges based on bytes scanned, as tested with 22 qps, using a single table of 20B rows and 20 TB of data.

Alternative to traditional data warehouse

SONY

> We ingest **tens of millions** of video streaming events into ClickHouse Cloud and generate dashboards for analysis to monitor, alert & troubleshoot the QOS and QOE of our customers in real-time.

Backend for observability platforms

COGNITIV™

> Data science as a discipline is not like engineering, where you can build in phases methodically. Iteration time is incredibly key to a data science team workflow.

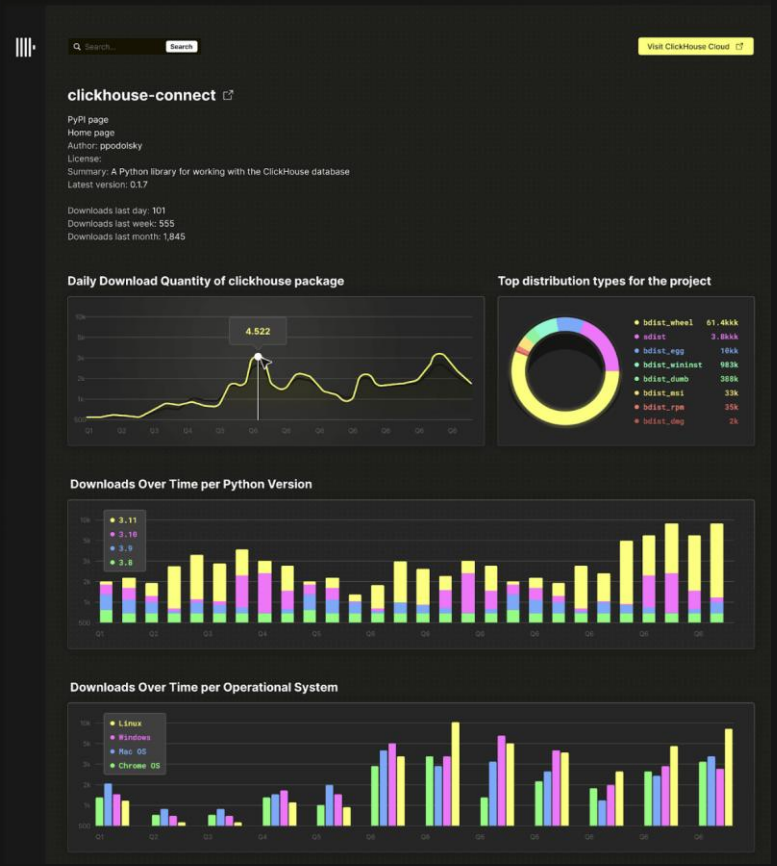
Model training and inference engine

Why ClickHouse?

https://clickpy.clickhouse.com/

The Challenge

Build an interactive, data driven application for Python package analytics



https://clickhouse.com/blog/clickhouse-vs-snowflake-for-real-time-analytics-benchmarks-cost-analysis

ClickHouse vs Snowflake

ClickHouse outperforms other cloud data warehouses and saves on costs

ClickHouse

vs

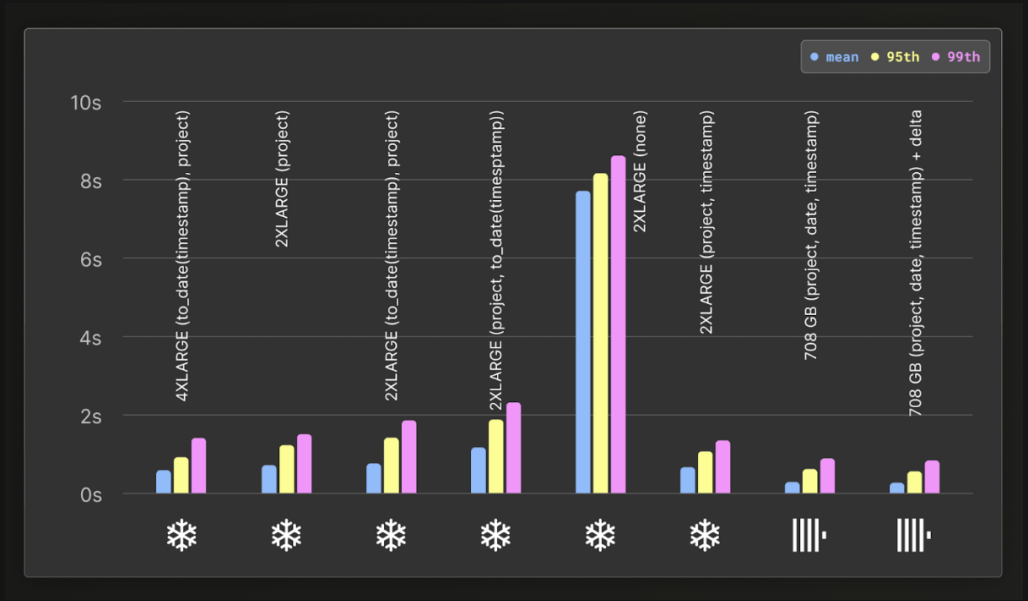
snowflake

3-5x	decrease in compute usage and costs
38x	more effective compression
2x+	faster

https://clickhouse.com/blog/clickhouse-vs-snowflake-for-real-time-analytics-benchmarks-cost-analysis

ClickHouse vs Snowflake - Speed

https://clickpy.clickhouse.com/



Key Takeaways

Clustering is critical to Snowflake's performance for real-time analytics, with an average response time of over 7s for non-clustered performance.

For clusters with comparable resources, ClickHouse outperforms Snowflake by at least 3x on the mean and 2x on the 95th and 99th percentile.

ClickHouse, with 177 vCPUs, even outperforms a 4X-LARGE Snowflake warehouse with 1024 vCPUs. This suggests our specific workload gains no benefit from further parallelization, as described by Snowflake.

https://clickhouse.com/blog/clickhouse-vs-snowflake-for-real-time-analytics-benchmarks-cost-analysis

ClickHouse vs Snowflake - Data Compression

Database	ORDER BY/CLUSTER BY	Total size (TiB)	Compression ratio on Parquet
Snowflake	-	1.99	4.39
Snowflake	(to_date(timestamp), project)	1.33	6.57
Snowflake	(project)	1.52	5.75
Snowflake	(project, to_date(timestamp))	1.77	4.94
Snowflake	(project, timestamp)*	1.05	8.32
ClickHouse	(project, date, timestamp)	0.902	9.67
ClickHouse	(project, date, timestamp) + delta codec	0.87	10.05

Most optional query performance

Most optimal compression

Key Takeaways

Clustering in Snowflake is essential for good compression and query performance in our use case.

The best clustering key chosen for our Snowflake schema resulted in reducing the data size by 40%.

Despite including the extra column date, the best compression in ClickHouse is nonetheless better than the most optimal Snowflake configuration by almost 20% (0.87TiB vs. 1.05TiB).

https://clickhouse.com/blog/clickhouse-vs-snowflake-for-real-time-analytics-benchmarks-cost-analysis

ClickHouse vs Snowflake - Costs

https://clickpy.clickhouse.com/

Database	Specification	Compute Cost per hour ($)	Compute Cost per month ($)	Data Storage Cost per month ($)	Total Cost per month ($)
Snowflake (standard)	2X-LARGE	64	$46,080	$28.73	$46,108
Snowflake (Enterprise)	2X-LARGE	96	$69,120	$28.73	$69,148
Snowflake (standard)	4X-LARGE	256	$184,320	$28.73	$184,348
ClickHouse	708GB	20.3196	$14,630	$42.48	$14,672

The above shows Snowflake is over 3x more expensive to run a production application than ClickHouse Cloud. For comparable performance between both systems, through Enterprise tier features, users pay 4.7x more with Snowflake compared to ClickHouse Cloud.

Key Takeaways

Snowflake costs can cascade quickly with enterprise features

Many pricing dimensions and hidden costs

Clustering and materialized views incur charges

ClickHouse Cloud is significantly more cost effective when the services/warehouses need to be active, less so if querying is ad hoc

https://clickhouse.com/blog/clickhouse-vs-snowflake-for-real-time-analytics-benchmarks-cost-analysis

ClickHouse

04 Combining analytical databases and data lakes

Combining data lakes with analytical databases

Do you still need to load data into analytical database?

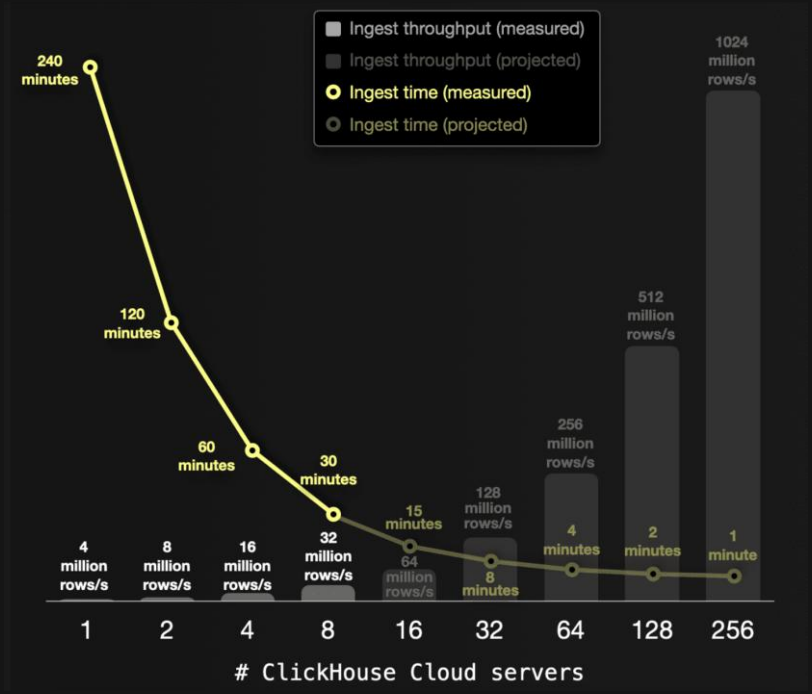
Questions to ask

- How much faster is it to write data locally vs Data Lake?
- How much faster is it to run queries locally vs Data Lake?

Combining data lakes with analytical databases

Speed of writes

Speed of writes depends on the writer resources and configuration



https://clickhouse.com/blog/supercharge-your-clickhouse-data-loads-part2

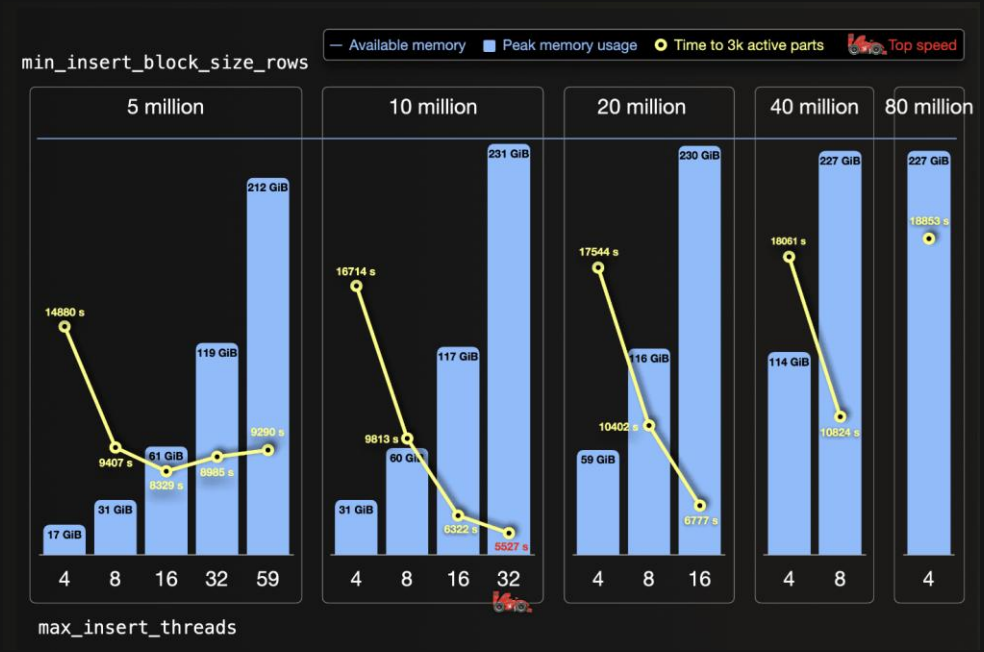
ClickHouse

Combining data lakes with analytical databases

Speed of writes

Real-time analytical databases are optimized for streaming ingestion at high throughput and low latency with many specialized controls

As a result, latency of writes to data lakes are slower than real-time analytical databases



https://clickhouse.com/blog/supercharge-your-clickhouse-data-loads-part2

ClickHouse

Combining data lakes with analytical databases

Speed of reads

ClickBench

- Open source benchmark for analytics
- Reads against data lakes are 2-14x slower

Detailed Comparison

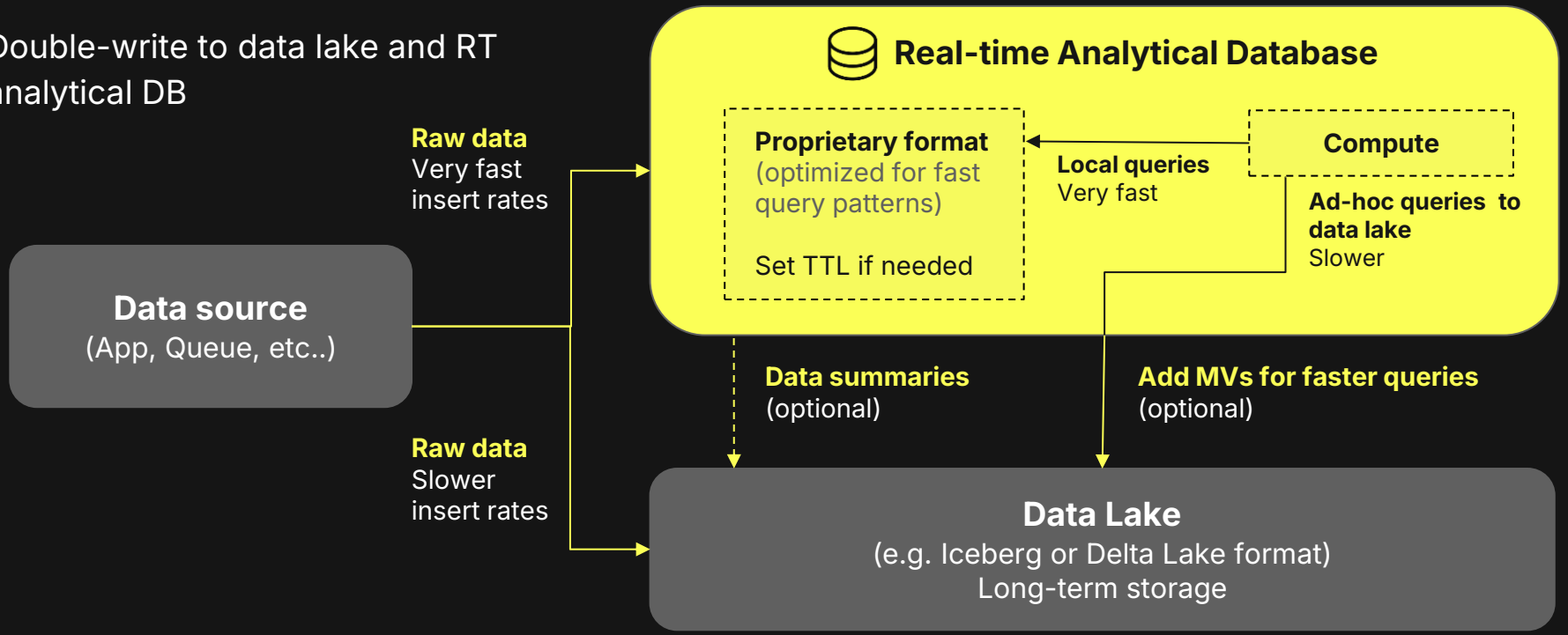
	ClickHouse (c6a.4xlarge, 500gb gp2)	ClickHouse (Parquet, partitioned) (c6a.4xlarge, 500gb gp2)	ClickHouse (data lake, partitioned) (c6a.4xlarge, 500gb gp2)	ClickHouse (Iceberg) (c6a.4xlarge, 500gb gp2)	ClickHouse (Iceberg, Partitioned) (c6a.4xlarge, 500gb gp2)	Athena (Iceberg) (serverless)
Load time:	494s (×1.00)	0	0	0	0	0
Data size:	13.40 GiB (×1.00)	13.73 GiB (×1.02)	13.73 GiB (×1.02)	0.00 GiB (×0.00)	0.00 GiB (×0.00)	0.00 GiB (×0.00)
Q0.	0.002s (×1.00)	0.040s (×4.17)	0.479s (×48.75)	0.249s (×21.58)	0.185s (×16.25)	2.059s (×172.42)
Q1.	0.007s (×1.00)	0.085s (×5.59)	0.751s (×44.76)	0.662s (×39.53)	0.805s (×47.94)	2.380s (×140.59)
Q2.	0.028s (×1.00)	0.144s (×5.13)	1.474s (×49.47)	0.989s (×33.30)	1.250s (×42.00)	2.591s (×86.70)
Q3.	0.027s (×1.00)	0.110s (×3.24)	1.026s (×28.00)	0.905s (×24.73)	1.388s (×37.78)	2.975s (×80.68)
Q4.	0.339s (×1.00)	0.429s (×1.26)	1.227s (×3.54)	1.113s (×3.22)	1.560s (×4.50)	2.948s (×8.48)
Q5.	0.443s (×1.00)	0.646s (×1.45)	1.475s (×3.28)	1.305s (×2.90)	1.660s (×3.69)	2.989s (×6.62)
Q6.	0.009s (×1.00)	0.099s (×5.74)	0.934s (×49.68)	0.617s (×33.00)	0.826s (×44.00)	2.448s (×129.37)
Q7.	0.011s (×1.00)	0.087s (×4.62)	0.643s (×31.10)	0.609s (×29.48)	0.743s (×35.86)	2.229s (×106.62)
Q8.	0.490s (×1.00)	0.600s (×1.22)	1.407s (×2.83)	1.288s (×2.60)	2.423s (×4.87)	3.193s (×6.41)
Q9.	0.533s (×1.00)	0.696s (×1.30)	2.163s (×4.00)	1.905s (×3.67)	3.124s (×5.77)	2.878s (×5.32)
Q10.	0.132s (×1.00)	0.301s (×2.19)	1.192s (×8.46)	1.212s (×8.61)	1.700s (×12.04)	2.710s (×19.15)
Q11.	0.149s (×1.00)	0.355s (×2.30)	1.536s (×9.72)	1.285s (×8.14)	1.862s (×11.77)	2.560s (×16.16)
Q12.	0.537s (×1.00)	0.792s (×1.47)	1.592s (×2.93)	1.384s (×2.55)	2.184s (×4.01)	3.141s (×5.76)
Q13.	0.775s (×1.00)	1.088s (×1.40)	2.489s (×3.18)	2.211s (×2.83)	4.385s (×5.60)	3.592s (×4.59)
Q14.	0.627s (×1.00)	0.923s (×1.46)	1.608s (×2.54)	1.546s (×2.44)	2.509s (×3.95)	2.509s (×4.00)
Q15.	0.491s (×1.00)	0.639s (×1.30)	1.321s (×2.66)	1.233s (×2.48)	1.820s (×3.65)	1.958s (×3.93)
Q16.	1.638s (×1.00)	2.013s (×1.23)	3.124s (×1.90)	3.000s (×1.83)	4.316s (×2.63)	3.379s (×2.06)
Q17.	0.886s (×1.00)	1.276s (×1.44)	2.367s (×2.65)	2.177s (×2.44)	5.293s (×5.92)	2.657s (×2.98)
Q18.	3.242s (×1.00)	4.364s (×1.35)	5.672s (×1.75)	3.698s (×1.14)	5.354s (×1.65)	3.582s (×1.10)
Q19.	0.014s (×1.00)	0.101s (×4.63)	0.705s (×29.70)	0.704s (×33.50)	1.316s (×55.25)	2.322s (×97.17)
Q20.	0.573s (×1.00)	1.838s (×3.17)	3.242s (×5.50)	3.536s (×6.00)	6.994s (×12.01)	3.645s (×6.27)
Q21.	0.622s (×1.00)	2.318s (×3.68)	4.013s (×6.37)	4.261s (×6.76)	5.995s (×9.50)	3.842s (×6.09)
Q22.	1.041s (×1.00)	5.124s (×4.88)	6.887s (×6.56)	8.549s (×8.14)	10.446s (×9.95)	3.935s (×3.75)
Q23.	2.383s (×1.00)	18.346s (×7.67)	21.136s (×8.84)	22.779s (×9.52)	43.773s (×18.30)	7.754s (×3.24)
Q24.	0.151s (×1.00)	0.462s (×2.93)	1.893s (×11.82)	1.615s (×10.09)	2.496s (×15.57)	2.892s (×18.02)
Q25.	0.123s (×1.00)	0.352s (×2.72)	1.159s (×8.79)	1.118s (×8.48)	2.677s (×20.20)	2.772s (×20.92)
Q26.	0.146s (×1.00)	0.456s (×2.99)	1.673s (×10.79)	1.495s (×9.65)	2.284s (×14.71)	3.484s (×22.40)
Q27.	0.607s (×1.00)	2.089s (×3.40)	3.476s (×5.65)	4.302s (×6.99)	7.625s (×12.37)	3.854s (×6.26)
Q28.	8.792s (×1.86)	9.576s (×2.02)	10.896s (×2.30)	11.134s (×2.35)	21.368s (×4.51)	4.732s (×1.00)
Q29.	0.023s (×1.00)	0.122s (×4.00)	1.244s (×38.00)	4.822s (×146.42)	4.960s (×150.61)	4.519s (×137.24)
Q30.	0.424s (×1.00)	0.691s (×1.62)	1.907s (×4.42)	2.076s (×4.81)	4.178s (×9.65)	3.132s (×7.24)
Q31.	0.603s (×1.00)	1.104s (×1.82)	3.320s (×5.43)	2.929s (×4.79)	3.862s (×6.32)	3.800s (×6.22)
Q32.	3.792s (×1.00)	4.589s (×1.21)	5.814s (×1.53)	6.323s (×1.67)	8.212s (×2.16)	4.325s (×1.14)
Q33.	3.004s (×1.00)	4.657s (×1.55)	5.609s (×1.86)	6.024s (×2.00)	11.604s (×3.85)	3.244s (×1.08)
Q34.	3.012s (×1.00)	4.580s (×1.52)	5.542s (×1.84)	6.029s (×2.00)	8.928s (×2.96)	3.883s (×1.29)
Q35.	0.884s (×1.00)	1.070s (×1.21)	1.688s (×1.90)	1.029s (×1.16)	1.432s (×1.61)	2.875s (×3.23)
Q36.	0.034s (×1.00)	0.161s (×3.89)	0.447s (×10.39)	4.393s (×100.07)	3.729s (×84.98)	3.172s (×72.32)
Q37.	0.021s (×1.00)	0.093s (×3.32)	0.387s (×12.81)	3.800s (×122.90)	3.601s (×116.48)	4.000s (×129.35)
Q38.	0.013s (×1.00)	0.111s (×5.26)	0.450s (×20.00)	4.050s (×176.52)	3.592s (×156.61)	3.560s (×155.22)
Q39.	0.064s (×1.00)	0.259s (×3.64)	0.613s (×8.42)	6.772s (×91.65)	5.576s (×75.49)	3.444s (×46.68)
Q40.	0.011s (×1.00)	0.064s (×3.52)	0.330s (×16.19)	2.011s (×96.24)	2.419s (×115.67)	3.020s (×144.29)
Q41.	0.009s (×1.00)	0.064s (×3.89)	0.351s (×19.00)	2.197s (×116.16)	2.698s (×142.11)	3.909s (×206.26)
Q42.	0.008s (×1.00)	0.060s (×3.89)	0.261s (×15.06)	1.532s (×85.67)	1.748s (×97.67)	2.297s (×128.17)

ClickHouse

Combining data lakes with analytical databases

Typical architectures - Fast data path

Double-write to data lake and RT analytical DB

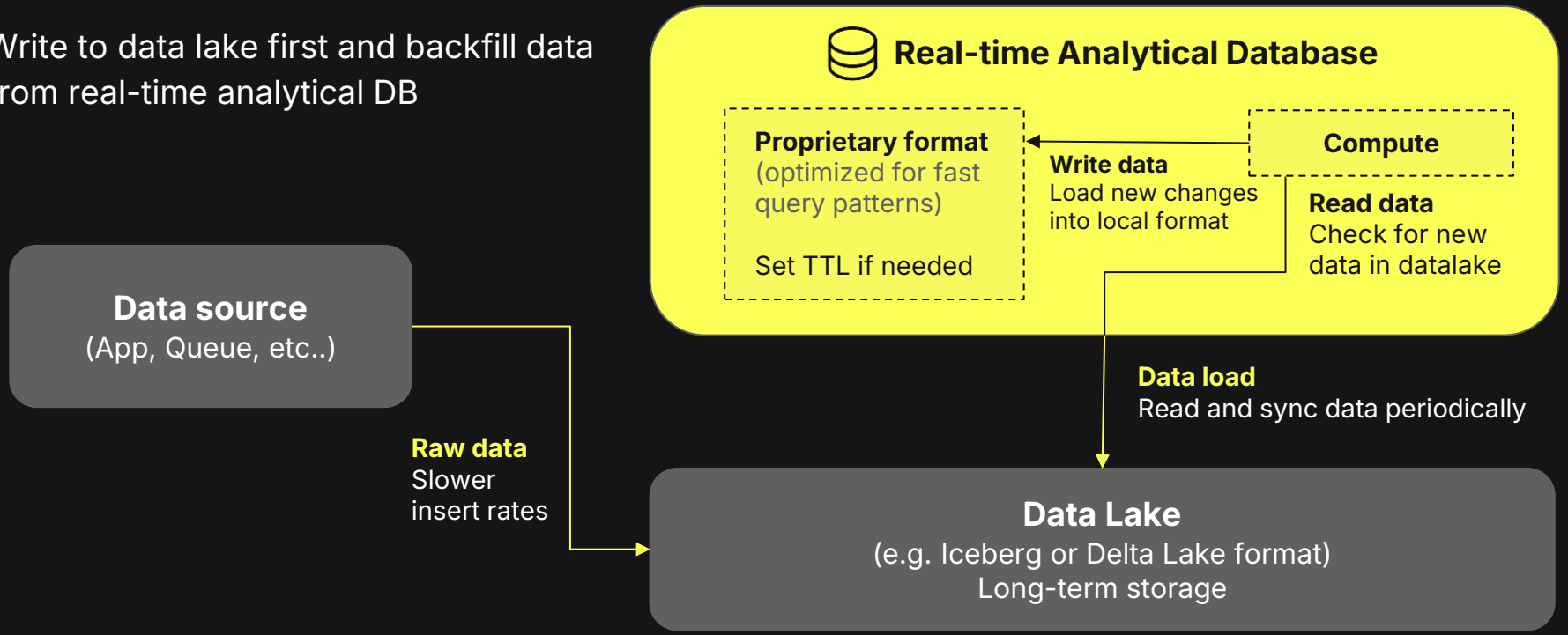


ClickHouse

Combining data lakes with analytical databases

Typical architectures - Slow data path

Write to data lake first and backfill data from real-time analytical DB

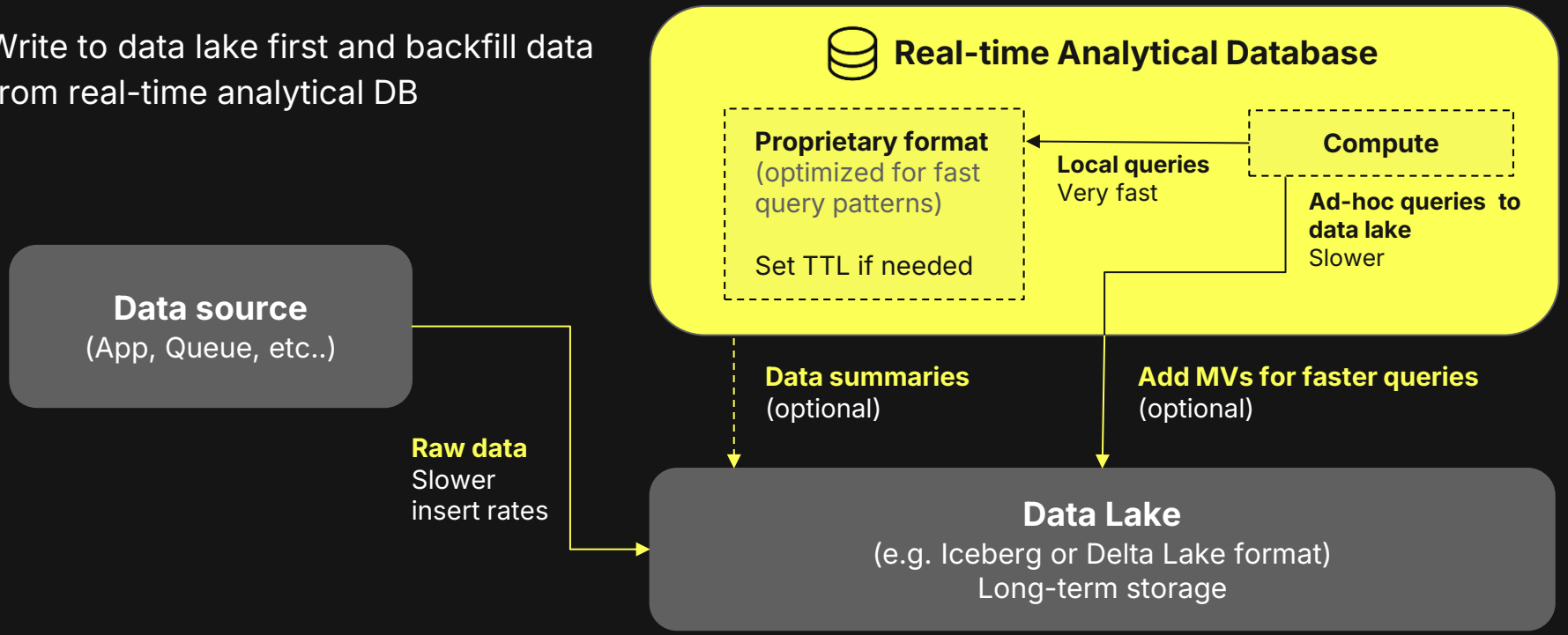


ClickHouse

Combining data lakes with analytical databases

Typical architectures - Slow data path

Write to data lake first and backfill data from real-time analytical DB



ClickHouse

ClickHouse

ClickHouse Usage Examples

ClickHouse

Demo Resources

Tutorial

https://clickhouse.com/docs/en/getting-started/example-datasets/environmental-sensors

ClickHouse Install

curl https://clickhouse.com/ | sh

https://clickhouse.com/cloud

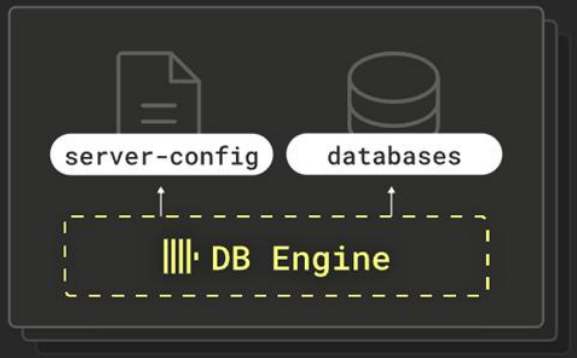
More Demos

https://clickhouse.com/demos

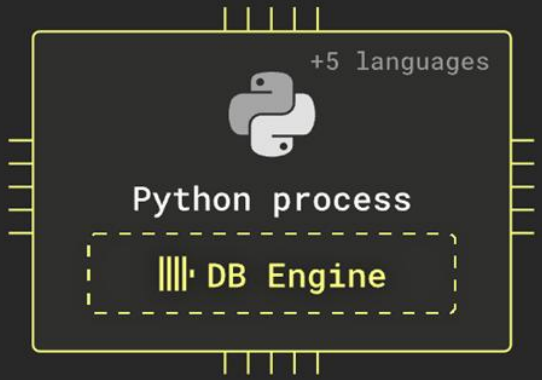
https://play.clickhouse.com

ClickHouse deployment options

Single binary, no dependencies



ClickHouse



chDB: ClickHouse In-Process

ClickHouse Local

Real-Time Data Analytics Pla

clickhouse.com/clickhouse

Going to re:Invent this December? Come party with us and The Chainsmokers →

ClickHouse

Products Use cases Docs Resources Pricing Contact us 36.9k Sign in Get started

- Internet of Things (IoT)
- Observability
- User behavior analytics

Start using ClickHouse in minutes

Install ClickHouse for MacOS, Linux, and FreeBSD.

```
$ curl https://clickhouse.com/ | sh
```

Or install for Windows, Docker or see other install options.

Watch this getting started video to learn more about ClickHouse.

console.clickhouse.cloud/warehouses

ClickHouse

Select a service

Select a service to access the SQL console, sizing options and more.

Or visit the support portal to access your tickets.

Learn

Support

Organization

UI Team Productio...

Integrations

Chat with support

All systems operational

All warehouses - UI Team Production Analytics

+ Create new warehouse

Host your data in a warehouse, use one or more compute services to access it. Learn more

TB-AirQuality	383.2 GB	N. Virginia (us-east-1)
behemot log demo	1.4 GB	Ohio (us-east-2)
test creation	0	Ohio (us-east-2)
Docs Testing	25.1 MB	Oregon (us-west-2)
test-auth	0	Ohio (us-east-2)
security	121.1 GB	Ohio (us-east-2)

1 service

ClickHouse data query options

Local, federated, and even hybrid queries are supported

Open source

- **Federated queries on data in S3 (Parquet, ORC, Avro, ...)**
- **Support for Iceberg, Delta Lake, Hudi tables**
- **Local queries on data loaded to ClickHouse**

ClickHouse Cloud

- **SQL Console**
- **AI query co-pilot**

Environmental Sensors Data

Sensor.Community is a contributors-driven global sensor network that creates Open Environmental Data. The data is collected from sensors all over the globe. Anyone can purchase a sensor and place it wherever they like. The APIs to download the data is in GitHub and the data is freely available under the Database Contents License (DbCL).

> **Info**
> The dataset has over 20 billion records, so be careful just copying-and-pasting the commands below unless your resources can handle that type of volume. The commands below were executed on a **Production** instance of ClickHouse Cloud.

1. The data is in S3, so we can use the `s3` table function to create a table from the files. We can also query the data in place. Let's look at a few rows before attempting to insert it into ClickHouse:

```
SELECT *
FROM s3(
    'https://clickhouse-public-datasets.s3.eu-central-1.amazonaws.com/sensors/monthly/2019-06_bmp180.csv.z
    'CSVWithNames'
   )
LIMIT 10
SETTINGS format_csv_delimiter = ';';
```

The data is in CSV files but uses a semi-colon for the delimiter. The rows look like:

```
┌─sensor_id─┬─sensor_type─┬─location─┬──────lat─┬──────lon─┬─timestamp───────────┬─pressure─┬─altitude─┬─pres
│      9119 │ BMP180      │     4594 │ 50.994   │  7.126   │ 2019-06-01T00:00:00 │   101471 │ NULL     │ NULL
```

ClickHouse data load options

Many choices in open source and in ClickHouse Cloud

Open source

- **Command Line Client** - INSERT ... SELECT <- Demo 1
- **S3** Table Function
- **Iceberg, Delta Lake, Hudi** Table Functions
- **S3Queue** Table Engine

ClickHouse Cloud

- **ClickPipes managed ingest** <- Demo 2

```
Connected to ClickHouse server version 24.6.1.

clickhouse-cloud :) CREATE TABLE sensors
(
    sensor_id UInt16,
    sensor_type Enum('BME280', 'BMP180', 'BMP280', 'DHT22', 'DS18B20', 'HPM', 'HTU21D', 'PMS1003', 'PMS3003', 'PMS5003', 'PMS6003', 'PMS7003', 'PPD42NS', 'SDS011'),
    location UInt32,
    lat Float32,
    lon Float32,
    timestamp DateTime,
    P1 Float32,
    P2 Float32,
    P0 Float32,
    durP1 Float32,
    ratioP1 Float32,
    durP2 Float32,
    ratioP2 Float32,
    pressure Float32,
    altitude Float32,
    pressure_sealevel Float32,
    temperature Float32,
    humidity Float32,
    date Date MATERIALIZED toDate(timestamp)
)
ENGINE = MergeTree
ORDER BY (timestamp, sensor_id);
```

console.clickhouse.cloud/services/b42d2649-7a10-4252-b6c0-1e9119c6c72a/imports/0e471e3d-bb11-4751-b272-b565dbfc3dce

ClickHouse

MyAirQuality Servi...

- SQL Console
- Dashboards
- Data sources
- Backups
- Settings
- Monitoring
- Help

Connect

Organization

UI Team Productio...

- Integrations
- Chat with support
- All systems operational

Data sources / ClickPipes

1 **Select the data source**

If you need any help to setup ClickPipes, you can access the documentation for more details.

Apache Kafka | Confluent Cloud | Amazon MSK | Amazon Kinesis

Redpanda | WarpStream | Azure Event Hubs | Amazon S3

Google Cloud Storage | New PostgreSQL CDC

2 Setup your ClickPipe Connection

3 Incoming Data

Environmental Sensors Data

Sensor.Community is a contributors-driven global sensor network that creates Open Environmental Data. The data is collected from sensors all over the globe. Anyone can purchase a sensor and place it wherever they like. The APIs to download the data is in GitHub and the data is freely available under the Database Contents License (DbCL).

Info

The dataset has over 20 billion records, so be careful just copying-and-pasting the commands below unless your resources can handle that type of volume. The commands below were executed on a **Production** instance of ClickHouse Cloud.

1. The data is in S3, so we can use the `s3` table function to create a table from the files. We can also query the data in place. Let's look at a few rows before attempting to insert it into ClickHouse:

```
SELECT *
FROM s3(
    'https://clickhouse-public-datasets.s3.eu-central-1.amazonaws.com/sensors/monthly/2019-06_bmp180.csv.z
    'CSVWithNames'
   )
LIMIT 10
SETTINGS format_csv_delimiter = ';';
```

The data is in CSV files but uses a semi-colon for the delimiter. The rows look like:

```
┌─sensor_id─┬─sensor_type─┬─location─┬──────lat─┬──────lon─┬─timestamp───────────┬─────pressure─┬─altitude─┬─pres
│      9119 │ BMP180      │     4594 │ 50.994   │  7.126   │ 2019-06-01T00:00:00 │       101471 │ NULL     │ NULL
│     21210 │ BMP180      │    10762 │ 42.206   │ 25.326   │ 2019-06-01T00:00:00 │        99525 │ NULL     │ NULL
│     19660 │ BMP180      │     9978 │ 52.434   │ 17.056   │ 2019-06-01T00:00:04 │       101570 │ NULL     │ NULL
│     12126 │ BMP180      │     6126 │ 57.908   │  16.49   │ 2019-06-01T00:00:05 │    101802.56 │ NULL     │ NULL
```

ClickHouse

Summary / Next steps

Data Platform Options

Key building blocks to consider in your next data architecture

Data Warehouses

Pros: Turnkey solution

- Mature offerings from established vendors

Cons: Lock-in & Cost

- Data not available to other tools and query engines
- Expensive due to prevailing pricing models & legacy architectures

Data Lakes

Pros: Open data access

- Built on open object storage and data formats

Cons: Still evolving

- Lots of confusion about which format & data catalog is “best”
- Insert and query speeds are slow for real-time apps

Analytical DBs

Pros: Speed & Cost

- Fastest insert & query speeds

Cons: Part of the solution

- Most organizations will need to combine analytical databases with data lakes (compatibility is important)

Migration Best Practices

For unbundling *your* cloud data warehouse

1. Identify workloads that cost significant $$
2. Understand reasons for high costs
3. Migrate interactive workloads to analytical DBs
4. Migrate rarely / adhoc access data to open data lake
5. Profit

ClickHouse

ClickHouse

How open source is re-shaping the cloud data warehouse landscape

Further reading

https://clickhouse.com/blog/the-unbundling-of-the-cloud-data-warehouse

Find me

https://twitter.com/hellmar_becker

https://www.linkedin.com/in/hellmarbecker/

Hellmar Becker @ ClickHouse | **Aug 2025**

Try ClickHouse for your use case

- ClickHouse Cloud
- Download open source

Learn

- Academy / Certifications
- Blogs / YouTube

Engage with our community

- Community Slack
- Monthly Community Calls
- Meetups / Events

ClickHouse

ClickHouse

Connect with ClickHouse

slack from Salesforce

X

GitHub

Meetups and events

ClickHouse Academy

Try ClickHouse for your use case

- ClickHouse Cloud
- Download open source

Learn

- Academy / certifications
- Blogs / YouTube

Engage with our community

- Community Slack
- Monthly Community calls
- Meetups / events

ClickHouse