

The background of the entire image is a night-time photograph of the Singapore skyline. On the left, the Marina Bay Sands hotel is prominently featured with its three towers and the SkyPark on top, all illuminated. To its right, the Esplanade - Theatres on the Bay is visible, along with the ArtScience Museum, which has a lotus-like shape. The Helix Bridge is in the foreground on the left, with its red and blue lights reflecting in the water. The water of Marina Bay is in the foreground, reflecting the city lights. The sky is dark blue.

ClickHouse

Meetup



October 3, 2024
6:00 PM SGT



Room2F Bras Basah
Singapore

Tech Talks



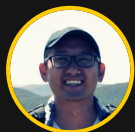
Roadmap & My Favourite Features

Tyler Hannan, Senior Director, Developer Advocacy
@ ClickHouse



Achieving 25K QPS with Sub-100ms P99 Latency in ClickHouse for Cost Savings Over Elasticsearch

Frank Chen, Expert OLAP Engineer @ Shopee



Security in ClickHouse Cloud - 2024 edition

San Tran, Application Security Engineer
@ ClickHouse



My Favourite Features

And a bit of roadmap

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.



ClickHouse Core

ClickHouse Core Roadmap 2024

<https://github.com/ClickHouse/ClickHouse/issues/58392>

- **SQL compatibility**
 - Enable Analyzer by default
 - JOINS reordering and extended pushdown
 - Correlated subqueries (with decorrelation)
- **Data storage & Interfaces**
 - Adaptive mode for async inserts
 - Semistructured data (prod readiness)
 - Lightweight updates (production readiness)
 - Transactions for Replicated tables
 - Vector search (production readiness)
 - Inverted indices (production readiness)
 - Support for Iceberg Data Catalog
 - Support for Hive-style partitioning
- **Security**
 - Encapsulation of access control for views
 - Resource scheduler
- **Query processing**
 - Parallel replicas with task callbacks (production readiness)
 - Parallel replicas with parallel distributed INSERT SELECT
 - Adaptive thresholds for data spilling on disk
- **Architectural Change**
 - Distributed cache
 - Stateless worker for background operations
 - Stateless worker for select queries (R&D)
- **Experiments & research**
 - PromQL support
 - Streaming queries
 - Key-value data marts
 - Unique key constraints



Faster Onboarding

Core recent features and roadmap

Insertion

Make it easier for user to insert data in ClickHouse:

- Semi-structured columns
- Async Insert by default
- Parallel replicas with parallel INSERT SELECT

Reading

Improved performance and better integration to make it easier to read data.

- Parallel replicas with task callbacks (production readiness)
- Improve Iceberg Table engine
- Improve Delta Lake Table Engine

Ease of Use

Making ClickHouse easier to use by simplifying features, making it more a "out of the box" experience:

- PromQL Support
- Adaptive threshold for data spilling on disk



Data Insertion and Manipulation

Core recent features and roadmap

Async Insert

Making asynchronous insert the default behavior in ClickHouse

Adjustable Asynchronous
Insert Timeout
Consistent INSERTs into
multiple MVs

Lightweight Operations

Delete data frequently without impacting performance

Lightweight operations at
query time
Materialization of lightweight
operations

Transactions

Bring ACID properties to use cases important to ClickHouse users

BEGIN TRANSACTION,
COMMIT and ROLLBACK
statements (experimental on
MergeTree)



More flexible analytics

Core recent features and roadmap

Analyzer by default

Enabling Analyzer by default opens many possibilities in ClickHouse

- Implement Analyzer in ClickHouse
- Enable Analyzer by default
- Simplify existing code
- Implement cost based optimizer

Extended JOIN support

Faster JOINS for more use cases; automatic JOIN algorithm selection

- Dynamic JOIN algorithm selection and JOIN rewrite (dep. Analyzer)
- Joins reordering and extended pushdown
- Support more operators for joins

Indexing / Search

More functionality and seamless migrations / ecosystem integrations

- Vector search indices (Production ready)
- Inverted indices (Production ready)



Blog / Engineering

ClickHouse Release 24.8 LTS



The ClickHouse Team

Sep 3, 2024

Another month goes by, which means it's time for another release!

ClickHouse version 24.8 contains **19 new features** 🎁 **18 performance optimisations** 🚀 **65 bug fixes** 🐛

This release is an LTS (Long Term Support) one, which means it will be supported for 12 months after release. To learn more about Stable and LTS releases, [see the documentation](#).

In this release, we've got the newly revamped JSON type, a table engine for time-series data, exactly-once processing of Kafka messages, and of course, join improvements!



What's new in ClickHouse

19 new features

- ★ Multiquery Mode By Default
- ★ Virtual Column_etag
- ★ Hive-Style Partitioning
- ★ printf function
- ★ Tagged Query Cache
- ★ Control Of Projections During Merges

8 performance optimisations

- ★ Optimisations for Subcolumns
- ★ OPTIMIZE Query For Join Tables
- ★ Faster Merges
- ★ Faster Drop Database

65 bug fixes

★ Interesting new features ★

- ★ Analyzer In Production
- ★ New Kafka Engine
- ★ TimeSeries Engine
- ★ JSON Data Type

The New Kafka Engine

The Kafka engine exists in ClickHouse since 2017
— it implements streaming consumption and data pipelines from Kafka.

Its downside: non-atomic commit to Kafka and to ClickHouse, leading to the possibility of duplicates in the case of retries.

Now there is an option to manage the offsets in Keeper:

```
SET allow_experimental_kafka_offsets_storage_in_keeper = 1;  
  
CREATE TABLE ... ENGINE = Kafka(  
    'localhost:19092', 'topic', 'consumer', 'JSONEachRow')  
SETTINGS  
    kafka_keeper_path = '/clickhouse/{database}/kafka',  
    kafka_replica_name = 'r1';
```

The New Kafka Engine

```
CREATE TABLE ... ENGINE = Kafka(  
    'localhost:19092', 'topic', 'consumer', 'JSONEachRow')  
SETTINGS  
    kafka_keeper_path = '/clickhouse/{database}/kafka',  
    kafka_replica_name = 'r1';
```

With the new option it does not rely on Kafka to track the offsets, and does it by itself with ClickHouse Keeper.

If an insertion attempt fails, it will take exactly the same chunk of data and repeat the insertion, regardless of network or server failures.

This enables deduplication and makes the consumption **exactly-once**.

Developer: János Benjamin Antal.

TimeSeries Engine

Now ClickHouse supports **Prometheus protocols** for remote write and read.

The new, **TimeSeries** Engine implements storage for metrics.

```
SET allow_experimental_time_series_table = 1;  
  
CREATE TABLE tbl ENGINE = TimeSeries; -- the default options.  
  
CREATE TABLE tbl ENGINE = TimeSeries  
    DATA ENGINE = MergeTree  
    TAGS ENGINE = ReplacingMergeTree  
    METRICS ENGINE = ReplacingMergeTree;
```

Developer: Vitaly Baranov.

TimeSeries Engine

```
$ cat /etc/clickhouse-server/config.d/prometheus.yaml
prometheus:
  port: 8053
  handlers:
    my_rule_1:
      url: '/write'
      handler:
        type: remote_write
        database: default
        table: tbl
    my_rule_2:
      url: '/read'
      handler:
        type: remote_read
        database: default
        table: tbl
    my_rule_3:
      url: '/metrics'
      handler:
        type: expose_metrics
```


TimeSeries Engine

ClickHouse is listening the Prometheus protocol and ready to receive metrics.

TimeSeries engine is simple to use, but allows many customizations:

- put some tags (e.g., hostname) into separate columns;
- adjust table's primary key;
- adjust column types;
- ...

But there is more work to do:

- support for PromQL;

Developer: Vitaly Baranov.

OVERLAY

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

```
1. | res |  
   | Hello, test! |
```



OVERLAY

```
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
```

```
1. [res  
   Hello, test!]
```

Describing A Release

```
$ sudo docker run --rm clickhouse/clickhouse-server:24.8 clickhouse-local --query  
"SELECT * FROM system.contributors ORDER BY name" > contributors_24.8.txt
```

```
$ sudo docker run --rm clickhouse/clickhouse-server:24.9 clickhouse-local --query  
"SELECT * FROM system.contributors ORDER BY name" > contributors_23.9.txt
```



Describing A Release

```
numbersix at thevillage in ~/clickhouse-mac
$ ./clickhouse local --query "
    SELECT arrayStringConcat(groupArray(line), ', ')
    FROM file('contributors_24.9.txt', LineAsString)
    WHERE line NOT IN (
        SELECT *
        FROM file('contributors_24.8.txt', LineAsString))
    FORMAT TSVRaw"
```

```
1on, Alexey Olshanskiy, Alexis Arnaud, Austin Bruch, Denis Hananein, Dergousov,
Gabriel Mendes, Konstantin Smirnov, Kruglov Kirill, Marco Vilas Boas, Matt Woenker,
Maxim Dergousov, Michal Tabaszewski, NikBarykin, Oleksandr, Pedro Ferreira, Rodrigo
Garcia, Samuel Warfield, Sergey (Finn) Gnezdilov, Tuan Pham Anh, Zhigao Hong,
baolin.hbl, gao chuan, haozelong, imddba, kruglov, leonkozlowski, m4xxx1m, marco-vb,
megao, mmav, neoman36, okunev, siyuan
```



JSON Object Format

JSON Data Type

```
SET allow_experimental_json_type = 1;

CREATE TABLE test (time DateTime, data JSON) ORDER BY time;

-- or with parameters:
data JSON(
    max_dynamic_paths = N,
    max_dynamic_types = M,
    some.path TypeName,           -- a type hint
    SKIP path.to.skip,           -- ignore some paths
    SKIP REGEXP 'paths_regexp')
```

Developer: Pavel Kruglov.

JSON Data Type

How it works:

- Analyzes the JSON and infers data types for every path.
- Stores every path and every distinct type as a subcolumn.
- Up to the maximum number, when it will fallback to storing the rest of the paths together.

It enables fast column-oriented storage and queries on arbitrary semistructured data!

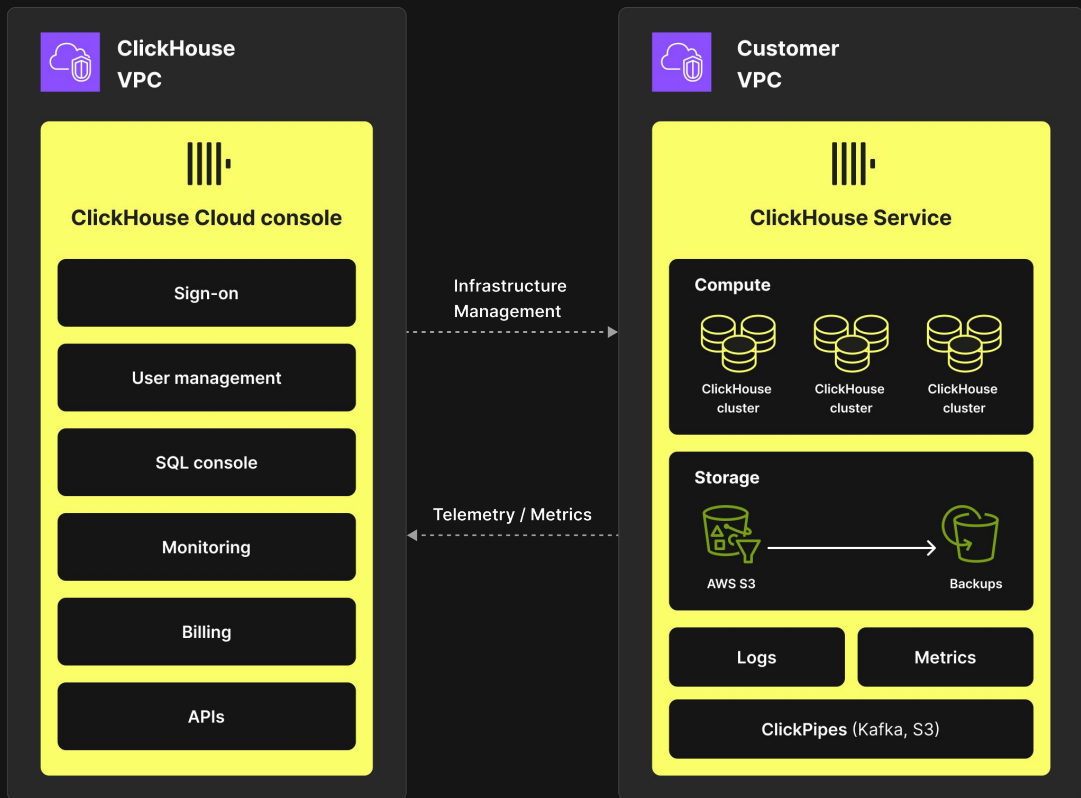
Developer: Pavel Kruglov.

Saved Queries as APIs



BYOC Architecture (AWS)[Private Preview]

- Compute and Storage will reside in Customer' VPC
- Only service health metrics will be accessible from the ClickHouse VPC for Monitoring and Management
- All web assets will be hosted on ClickHouse VPC - includes User Management, Service Configuration, SQL Console,...



Compute-Compute Separation[Private Preview]

With Compute-Compute separation, services can allocate dedicated compute for specific operations - eg: Streaming Ingest vs Adhoc Reporting, while sharing the same storage

- Compute units can be scaled independently
- Eliminates bottlenecks due to resource contention

