

ClickHouse

Meetup

**One2N office @ Rachna Ventura**  
Pune, India

September 20, 2025 at 10:30 AM IST



THANK YOU TO OUR HOST!



# Tech Talks



## Maximising Analytics with ClickHouse and Kafka Integration

Rakesh Puttaswamy, ClickHouse Principal Solutions Architect, & Siddesh Vyavahare, Confluent Advisory Solutions Engineer



## Sparse Data Storage and Query Patterns in ClickHouse

Amit Sanjay Sadafule, Co-Founder and Head of Technology @ Manthhan Software



## Why ThriveStack chose ClickHouse

Ankit Gupta, Senior Software Engineer @ ThriveStack



## ClickHouse for network and application analytics data for Cyber Security

Ajit Bhat, Engineering Manager, and Afzal Khan, Principal Software Engineer @ Netscout



# ClickHouse + Kafka Integration



# Rakesh Puttaswamy

## ClickHouse Lead Solution Architect

 [rakesh-p-gowda](https://www.linkedin.com/in/rakesh-p-gowda)



# Siddesh Vyavahare

## Confluent Advisory Solutions Engineer

 [siddesh-vyavahare-96229b150](https://www.linkedin.com/in/siddesh-vyavahare-96229b150)

Travel enthusiast, Once a Devops Engineer, Siddesh now queries customer problems and joins them with tailor-made solutions.

Passionate about Kafka & real time streaming use case pivoted from building data pipelines to flexing scalable real-time architectures.

Always chasing the next big challenge, professionally and personally.



# Agenda

**01**

**Introduction**

**02**

**Apache Kafka 101**

**03**

**Integrating Kafka with  
ClickHouse**

**04**

**Questions**

# Introduction: ClickHouse



# ClickHouse

2009

*Prototype*

2012

*Production*

2016

*Open Source*

2021

*ClickHouse Inc.*

2022

*ClickHouse  
Cloud*

**The Most Popular Analytics Database on the Planet**

**#1**

*Analytics DB on DB-Engines*

Over

**40,000**

*GitHub Stars*

Over

**200,000**

*Community Members*

# What is ClickHouse ?

ClickHouse is an **Open-Source**, columnar **OLAP** database  
Designed for **Blazing fast** analytics of massive volumes of data

1

**Speaks SQL fluently**



3

**Highly efficient storage**



2

**Processes data very fast**



4

**Easily scalable to any size**



# 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.*

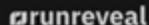
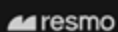
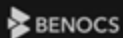


# Use cases



## Logs, events, traces

Monitor with confidence your logs, events, and traces. Detect anomalies, fraud, network or infrastructure issues, and more.



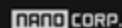
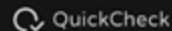
## Real-time Analytics

Power interactive applications and dashboards that analyze and aggregate large amounts of data on the fly. Run complex internal analytics in ms, not mins or hrs.



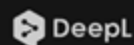
## Business intelligence

Interactively slice and dice your data for analysis, reporting, and building internal applications. Evaluate user behaviors, ad and media perf, market dynamics, and more.



## ML and Gen AI

Execute fast and efficient vector search. Plug-and-play Generative AI models from any provider. Use lightning-fast aggregations to power model training at petabyte scale.

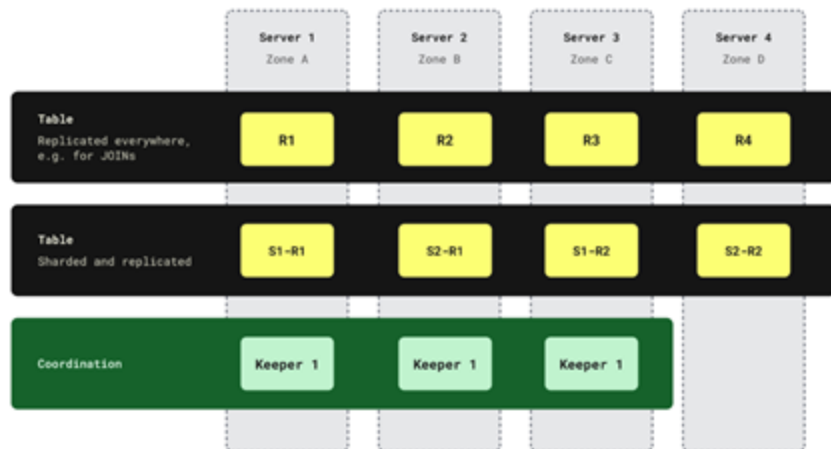


# ClickHouse

## Self-managed

- ✓ Open-source
- ✓ Flexible architecture
- ✓ Efficient and robust
- ✓ Support contracts available

Sample self-managed architecture

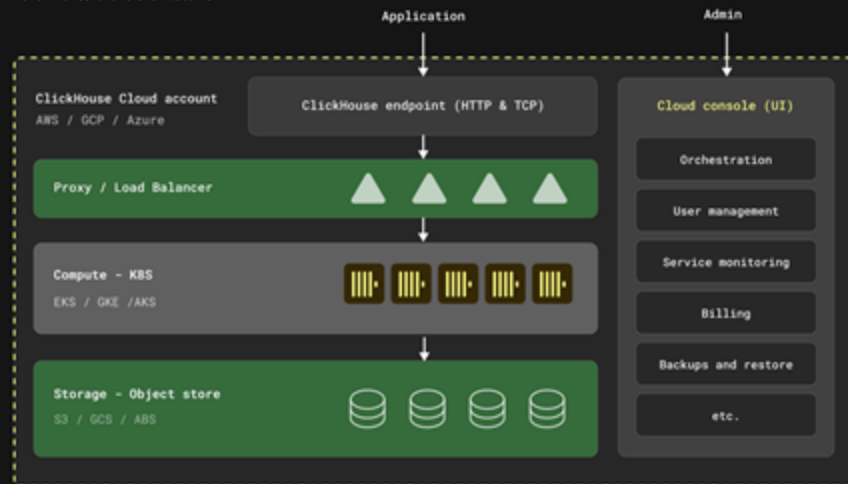


# ClickHouse

## Cloud

- ✓ Easy to use
  - ✓ Feature-rich
  - ✓ Fast
  - ✓ Scalable
  - ✓ Reliable
  - ✓ PAYG
- Managed for you  
Cloud-first features & tooling  
Automatically maximizes performance/efficiency  
Scale seamlessly  
Ensure reliability  
SaaS usage and capacity based pricing

ClickHouse Cloud architecture



# ClickHouse Architecture Patterns

# Row vs Column

completed_at	order_number	order_state	order_type	total_price
completed_at	order_number	order_state	order_type	total_price
completed_at	order_number	order_state	order_type	total_price

Row-based

**Excess disk Reads** We need data in column, but it's stored in rows. So entire rows are read.

completed_at	order_number	order_state	order_type	total_price
completed_at	order_number	order_state	order_type	total_price
completed_at	order_number	order_state	order_type	total_price

Column-Based

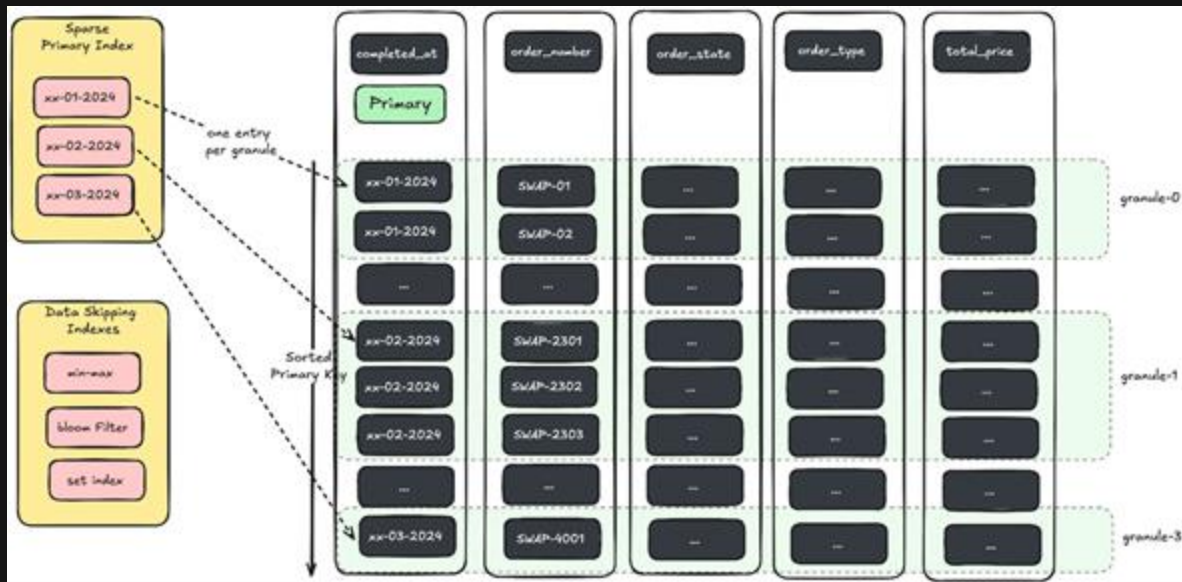
**Column Pruning** it only pick the subset of columns that are openly declared in the SQL query, thus eliminating 99% columns (990 of 1000 columns) from the disk read consideration.

## Vectorized Execution

data processed in **large arrays**, which could be the entire length of the column loaded into RAM.

+ **reduces CPU cache miss rates**

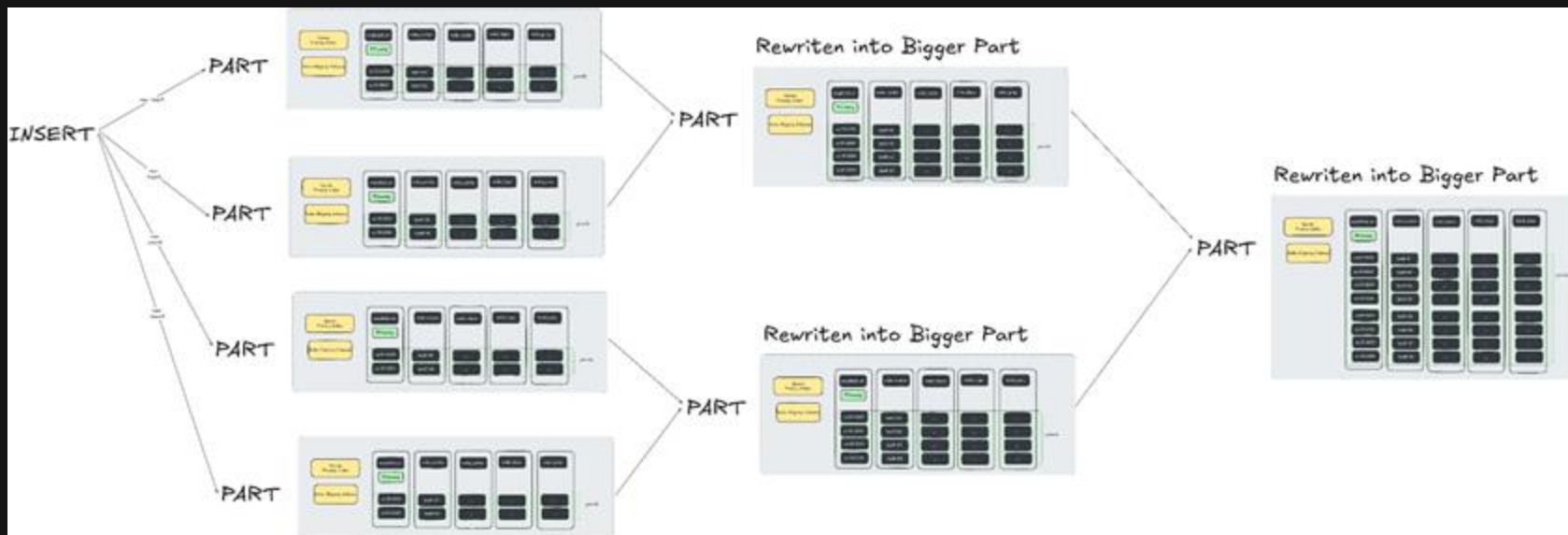
# Multiple Layers of Indexes



- **Granules** are chunks of rows, and **ClickHouse** groups rows into granules based on the `index_granularity` setting.
- **Sparse Primary Index:** ClickHouse uses the `sparse primary index` to quickly jump to the relevant granules, skipping over large portions of the data that don't need to be read.
- **Skip Indexes:** These allow **ClickHouse** to skip entire granules if it can determine from the index that no rows in the granule satisfy the query's filter.



# Efficient Merging

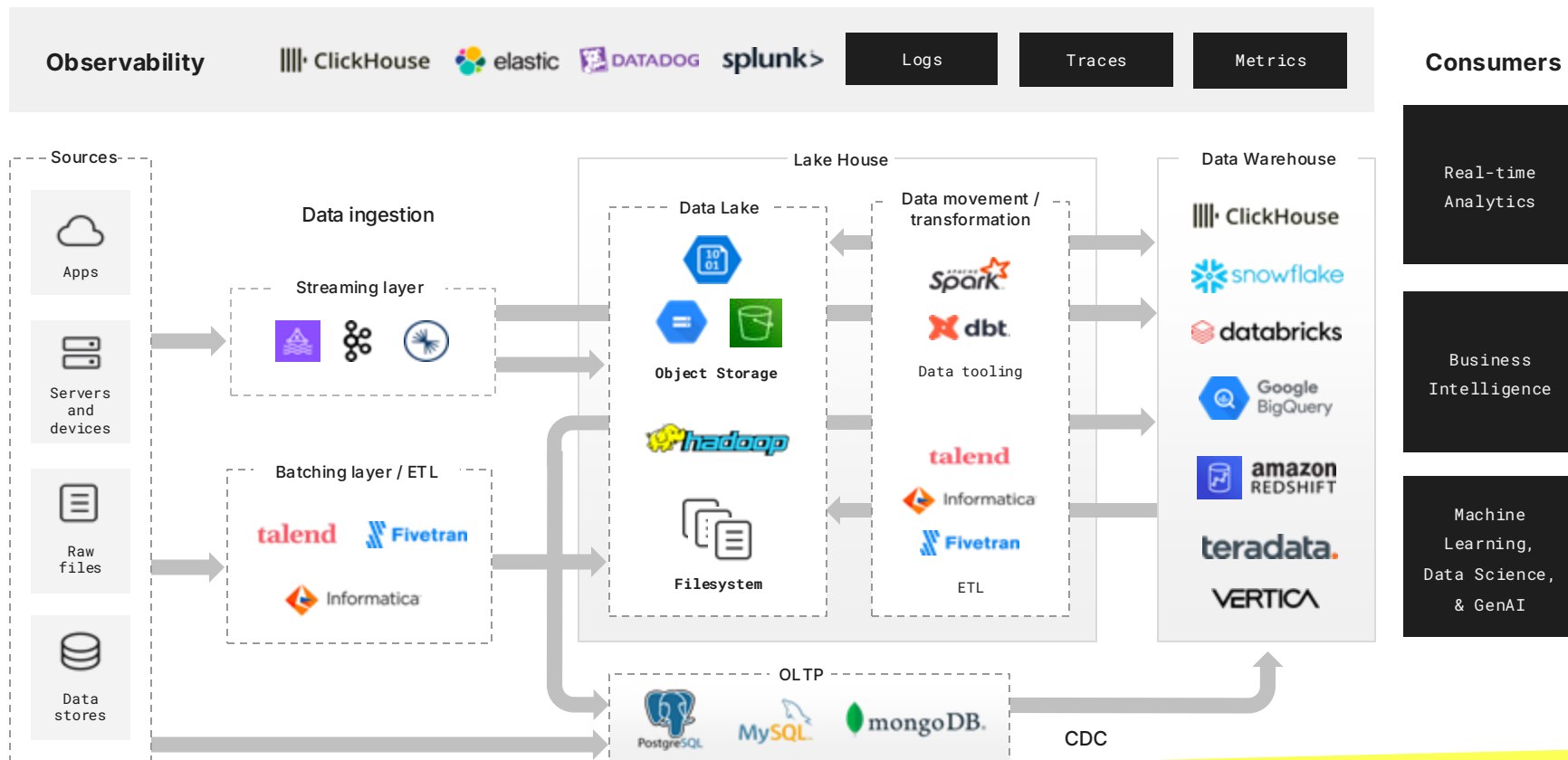


**Parts:** Each time data is inserted, a new part is created. Multiple parts can accumulate over time, which can slow down query performance.

To improve performance and reduce the number of parts, **ClickHouse** periodically runs merging operations.

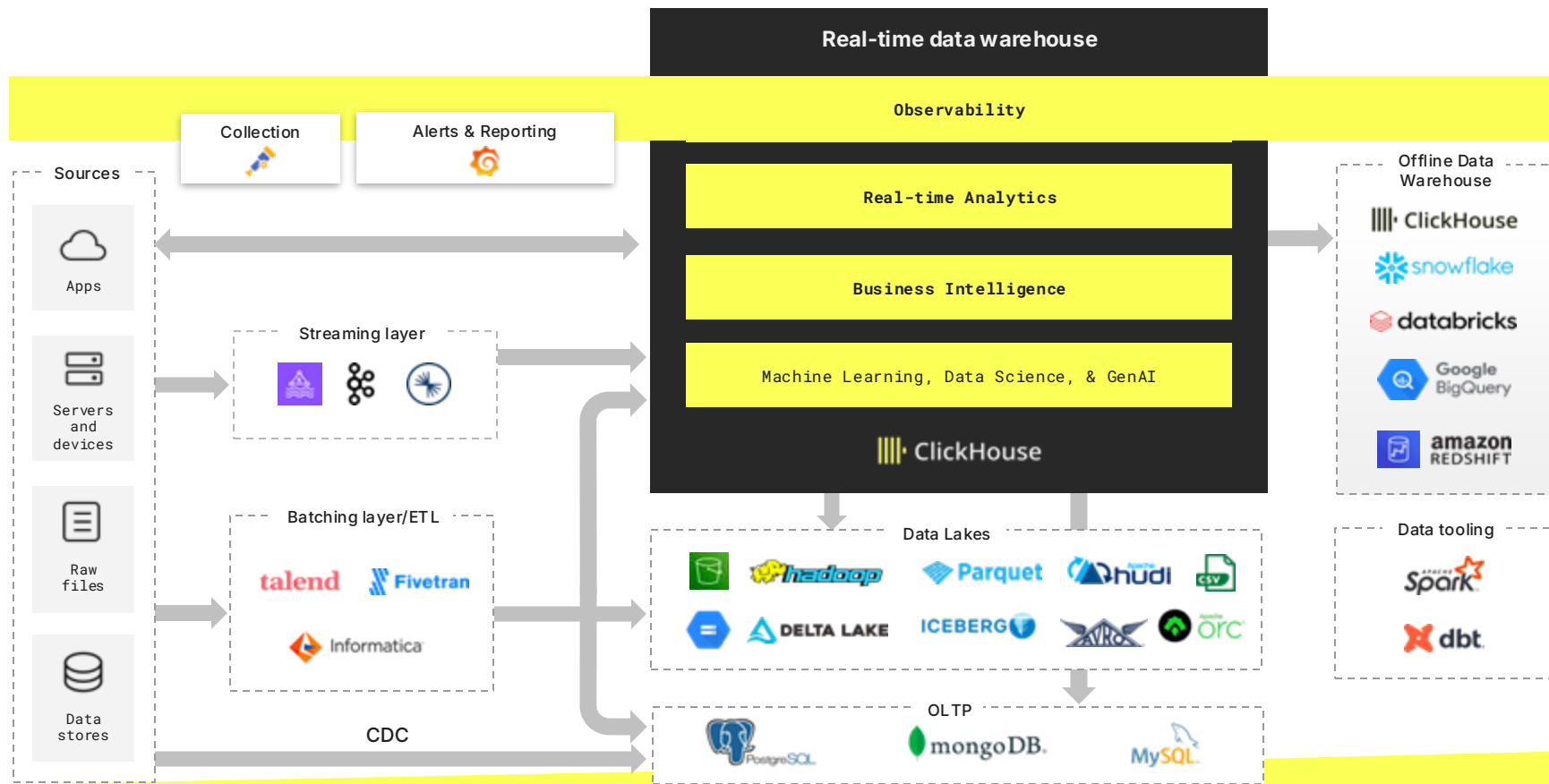
# Traditional Data Architecture

Operationally complex with skyrocketing expense



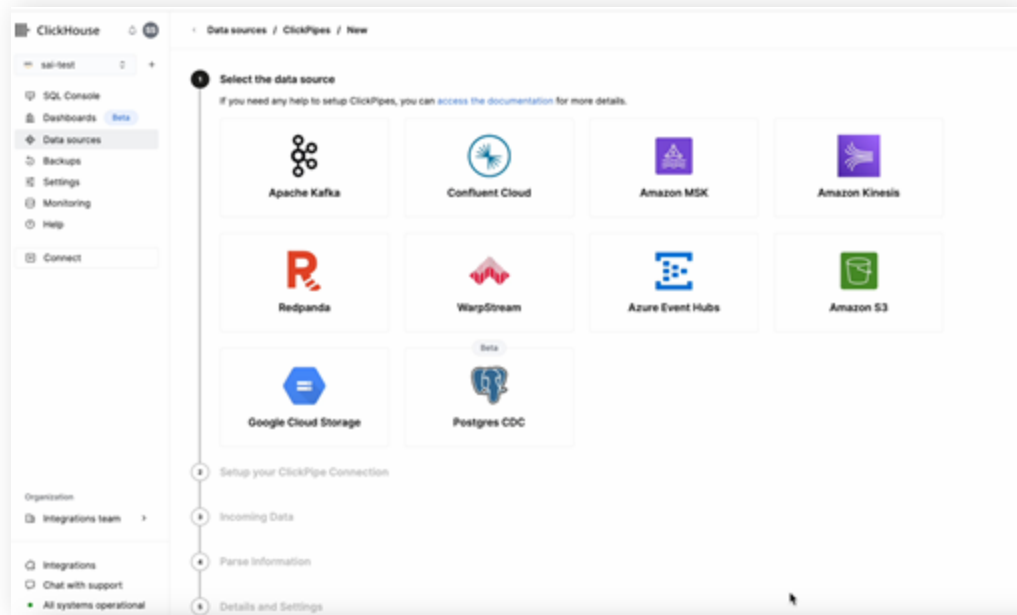
# Modern Data Architecture

Reduces system bloat and increases resource efficiency



# ClickPipes

- An integration engine that makes ingesting massive volumes of data from a diverse set of sources as simple as clicking a few buttons.
- Simplifies data ingestion from a variety of sources, including Kafka, Kinesis, Postgres, Amazon S3, and Google Cloud Storage.
- Scalable architecture ensures high throughput and low latency, ideal for demanding workloads.



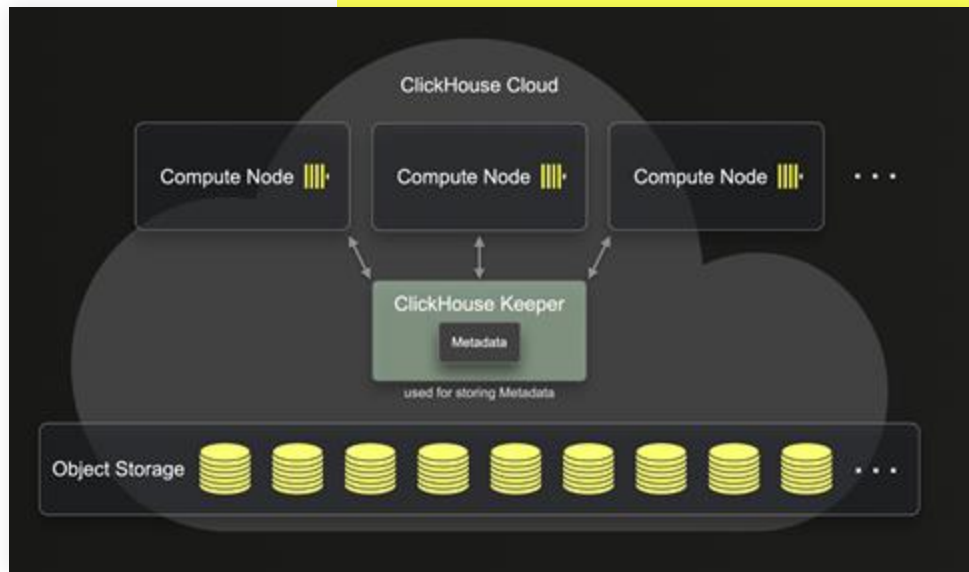
# Separation of compute and storage

**ClickHouse Cloud uses SharedMergeTree table engine, which allows storage and compute to be decoupled and scaled independently.**

Object storage is used as the primary store for data and local disks only for caching, metadata, and temporary storage. This provides uncompromised TCO and reliability guarantees.

## Benefits include:

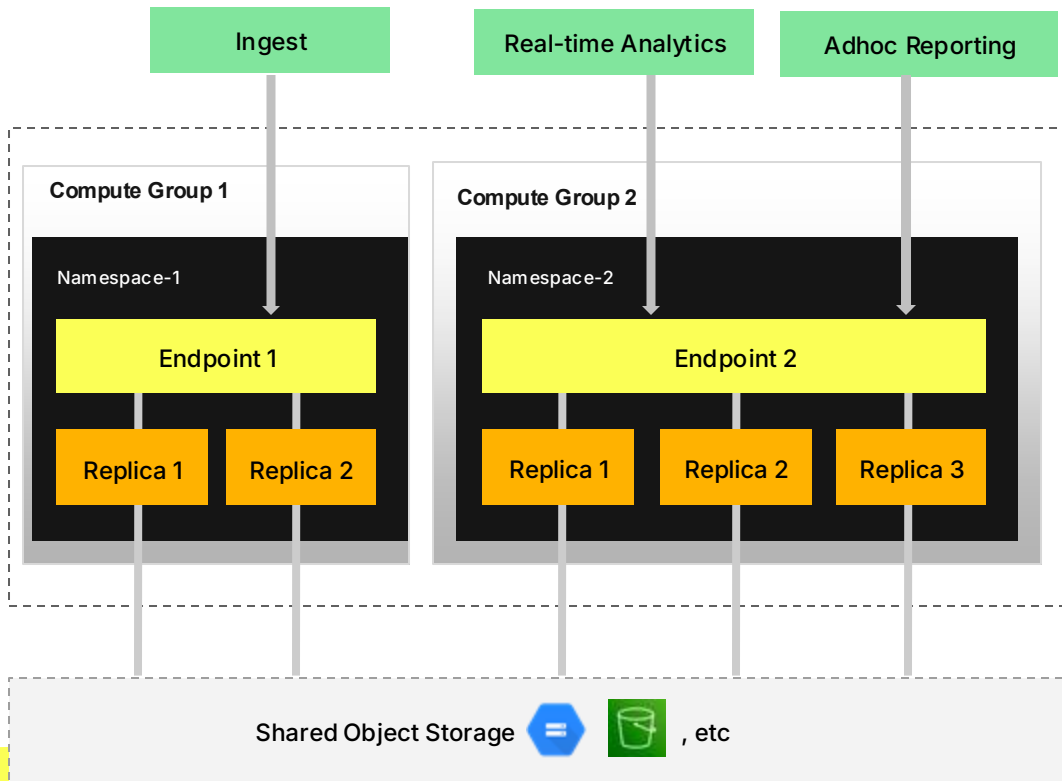
- 1 Shard/Replica - no manual rebalancing
- Virtually limitless storage
- Higher insert throughput
- Improved throughput of merges and mutations
- Faster scale-up and scale-down operations
- More lightweight consistency for selects
- ...and more!



# Compute-Compute Separation

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





# Introduction to Confluent

# The Rise of Event Streaming



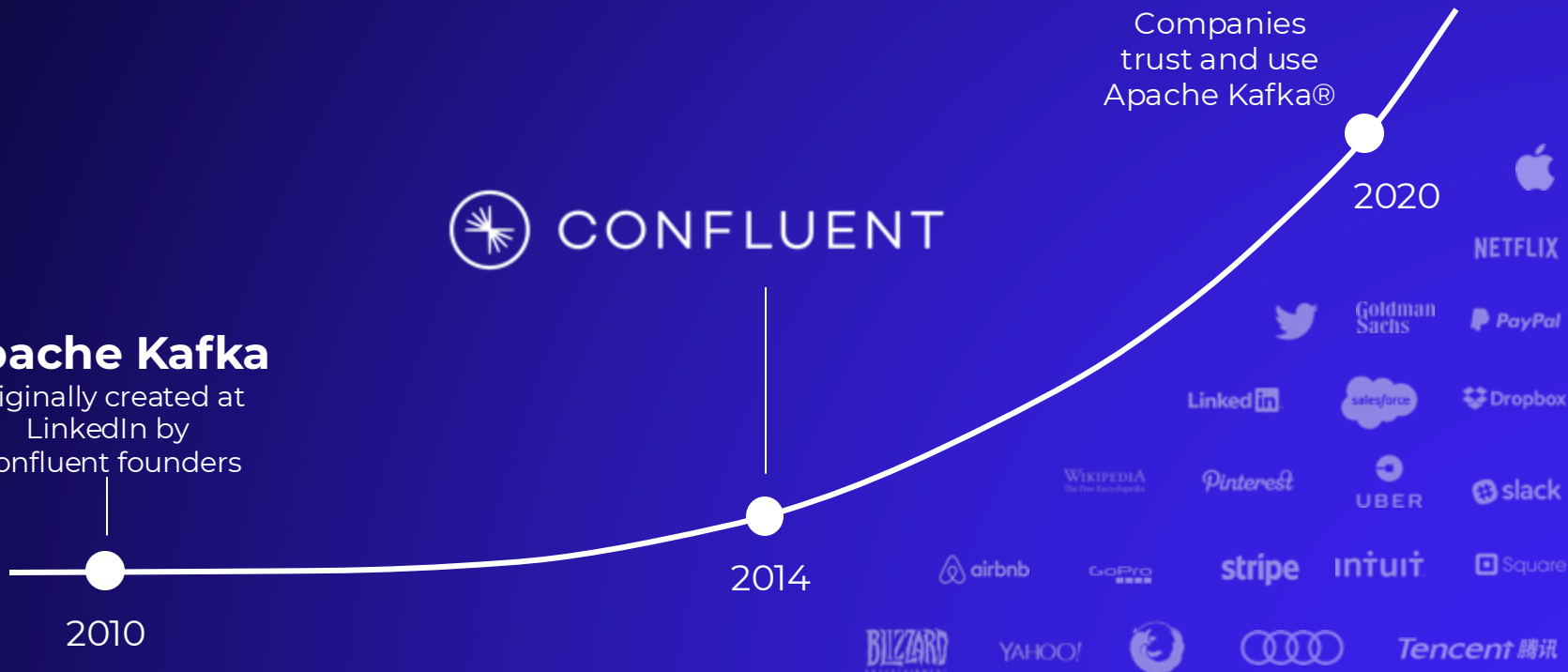
## Apache Kafka

originally created at  
LinkedIn by  
Confluent founders



# 80%

Fortune 100  
Companies  
trust and use  
Apache Kafka®

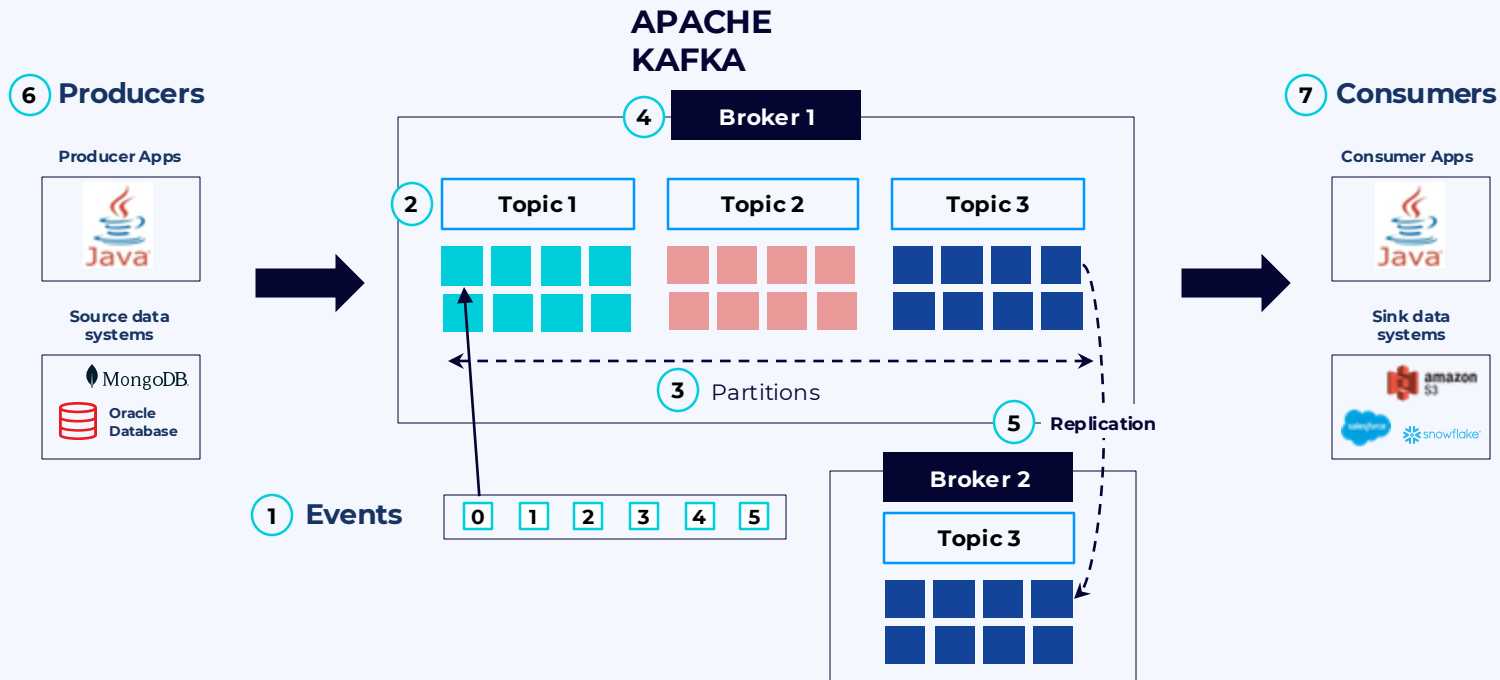






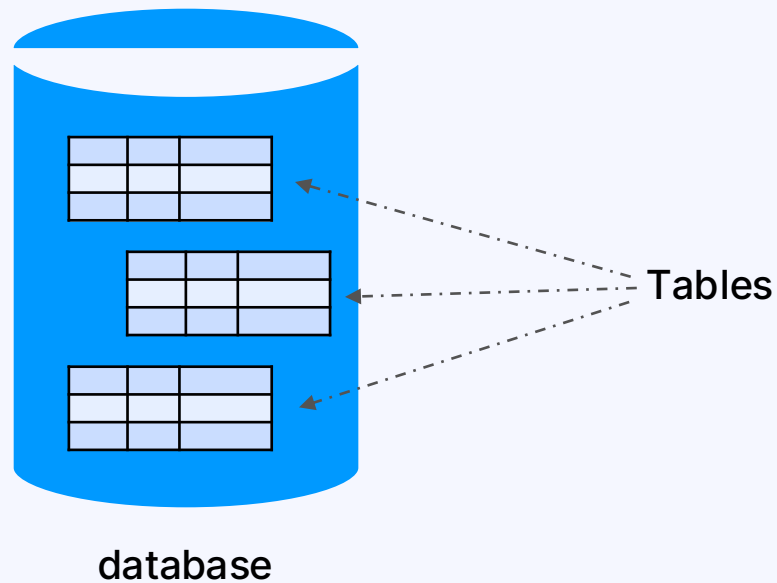
# Introducing Apache Kafka®

# Key components of Apache Kafka



**Apache Kafka** is a data streaming system that allows developers to react to new events as they occur in real time.

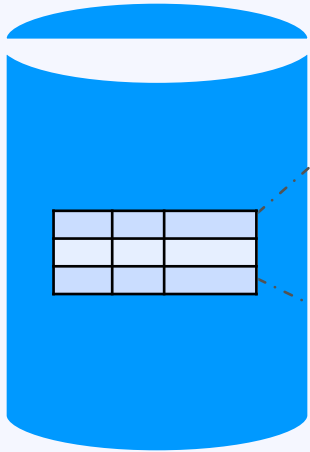
# Databases have tables





# Tables have rows and columns

## thermostat\_readings

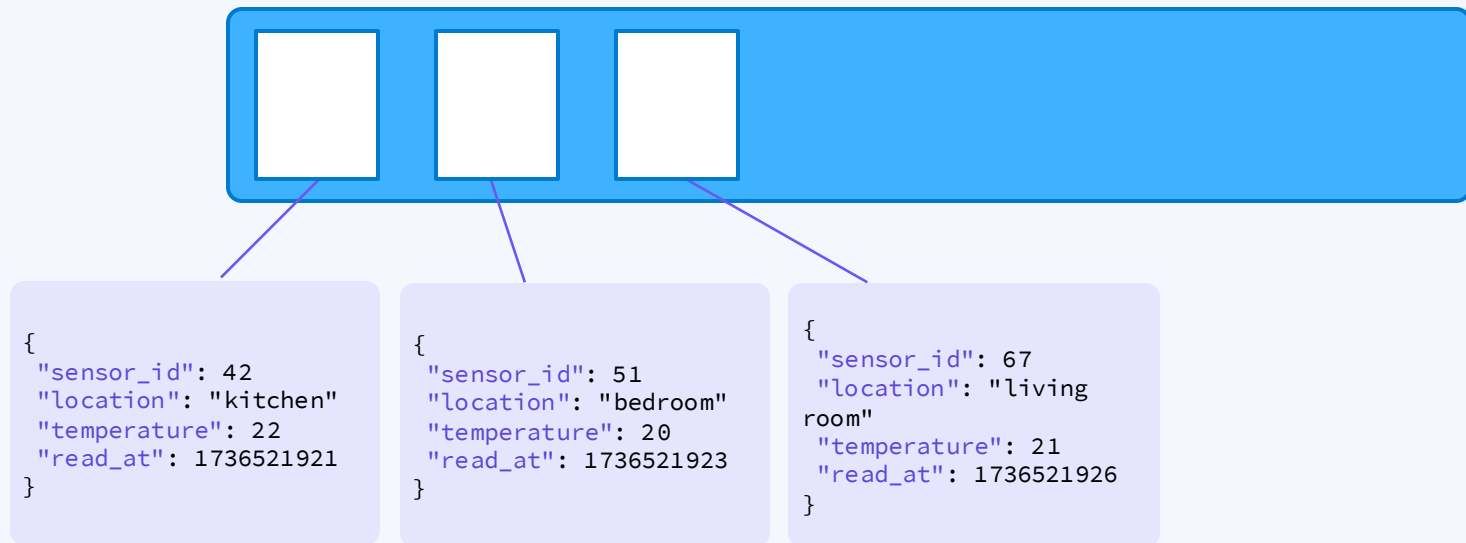


sensor_id	location	temperature	read_at
42	kitchen	22	1736521921
51	bedroom	20	1736521923
67	living Room	21	1736521926



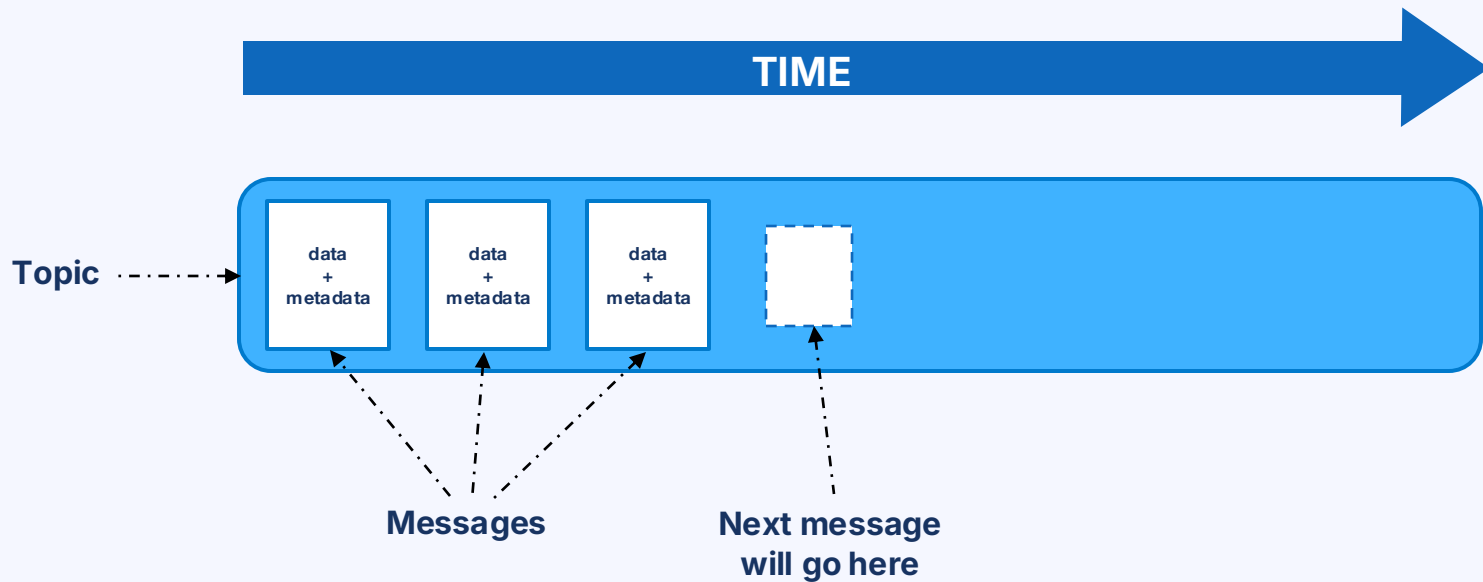
# Messages stored in receiving order

thermostat\_readings





# Topics store data in logs



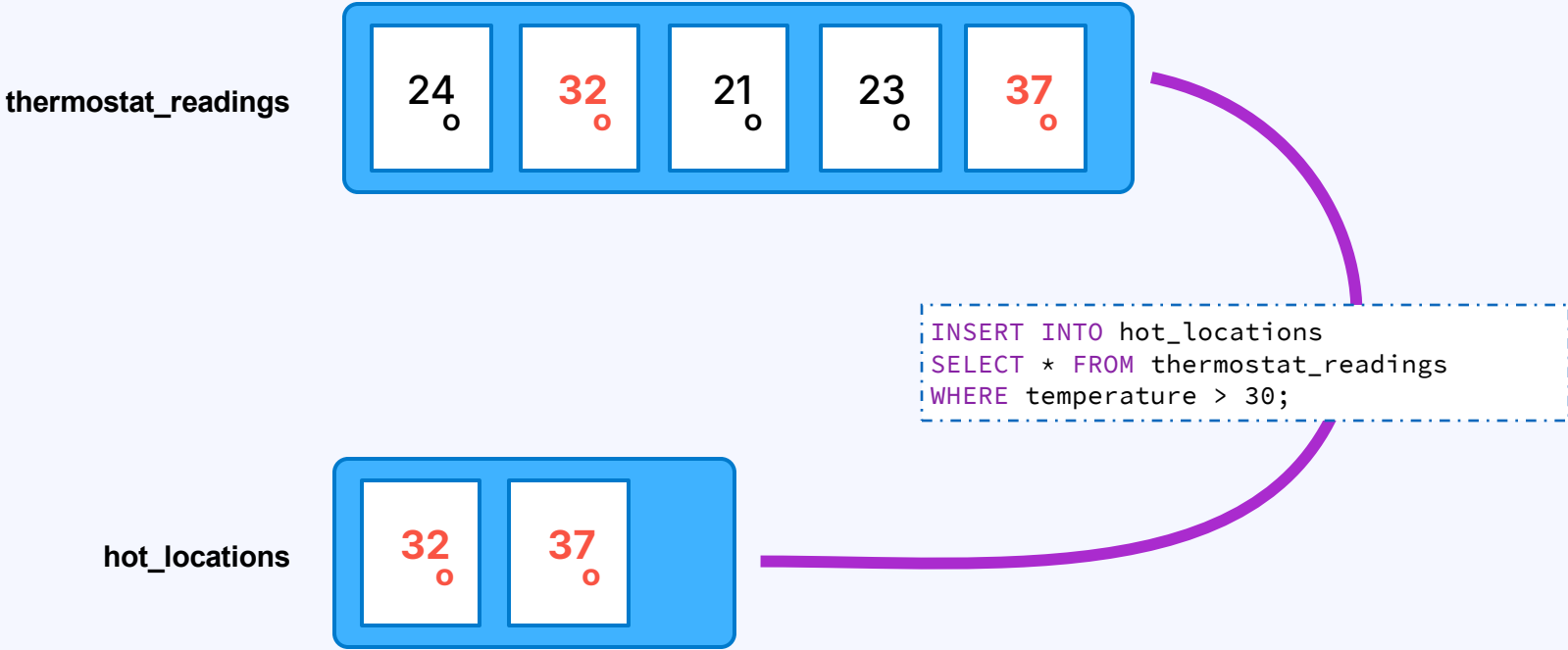


# Topics organize your data





# Derive topics from other topics







# Brokers



# Cluster

## Kafka Cluster



Broker 1



Disk



Broker 2



Disk



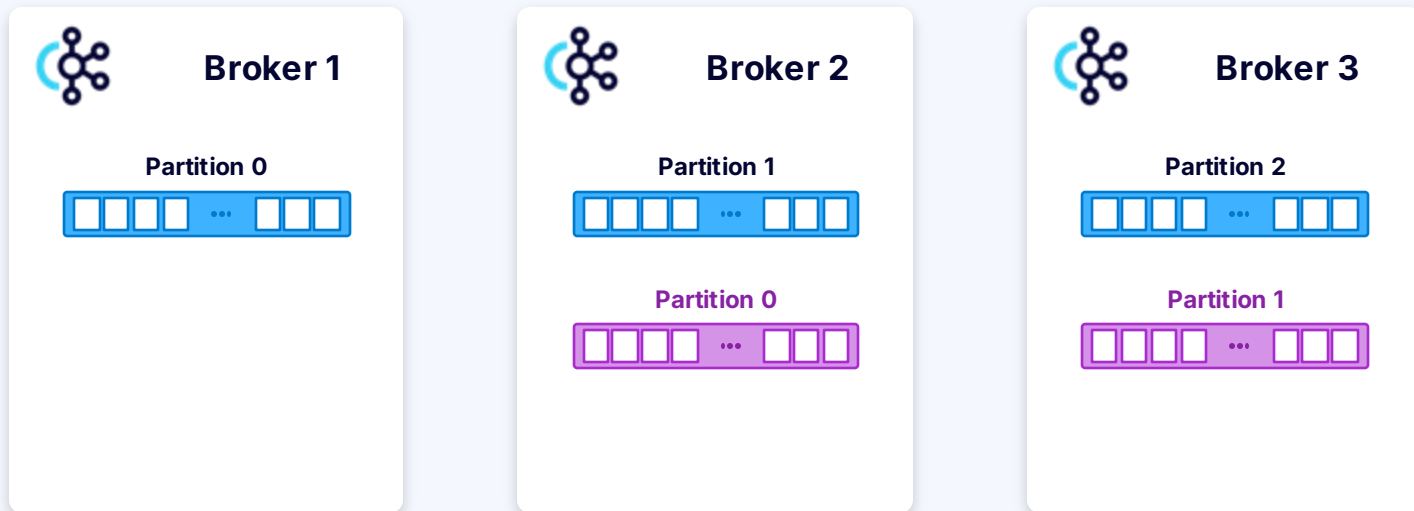
Broker 3



Disk

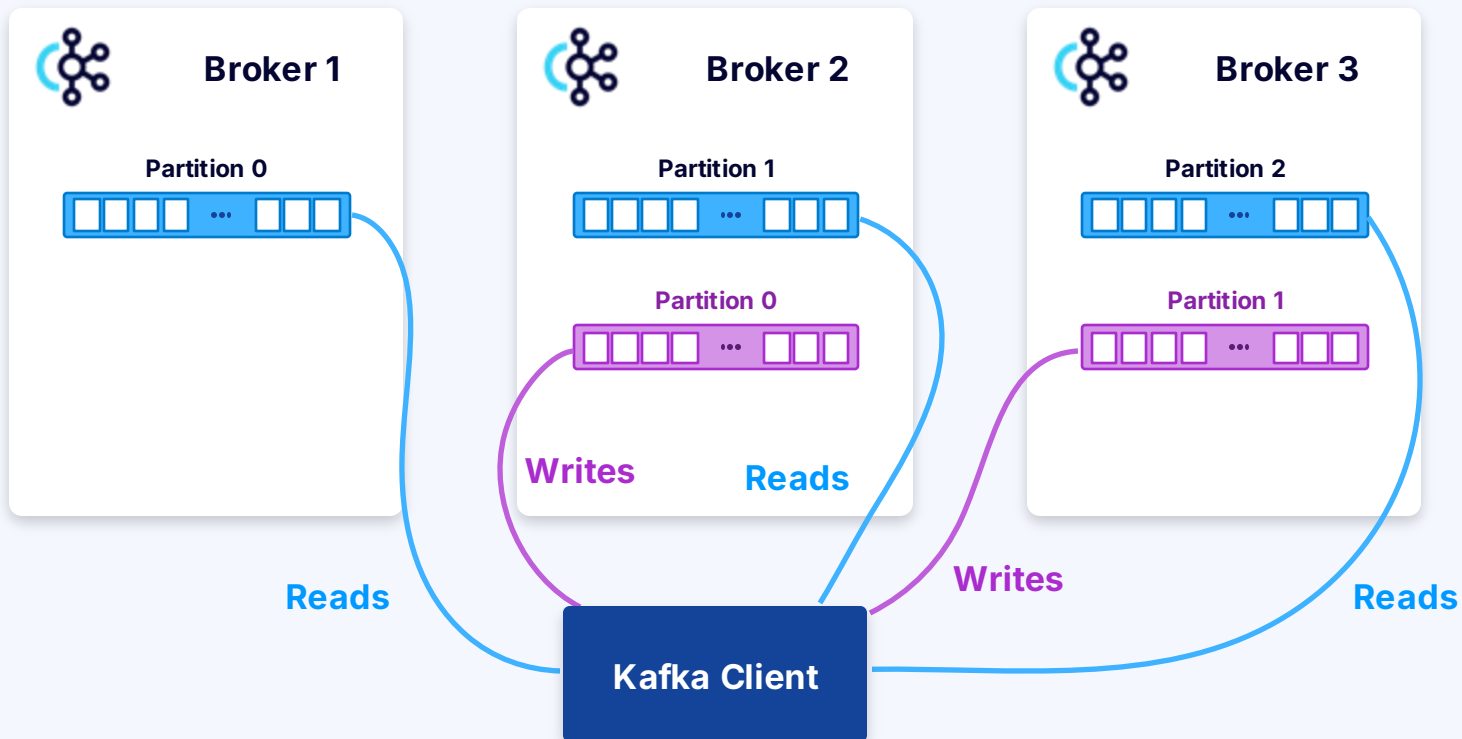


# Brokers host partitions





# Brokers handle reads and writes

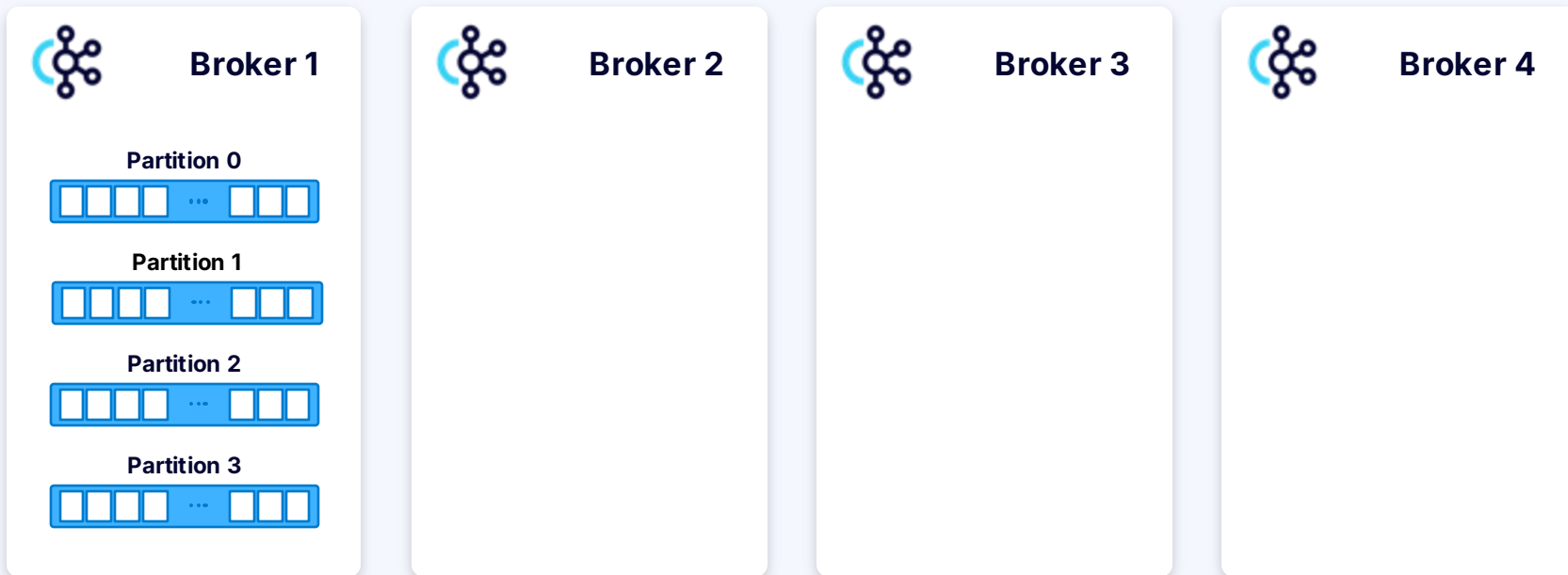




# Replication

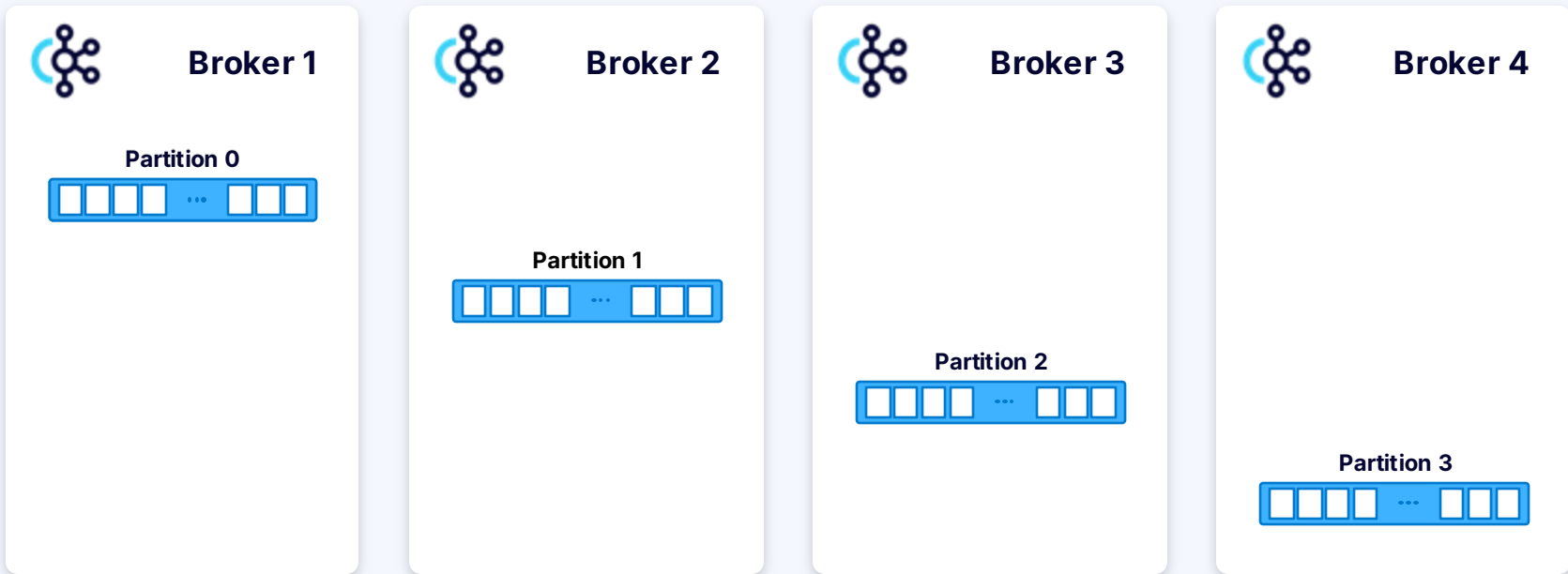


# Kafka replicates partitions across brokers



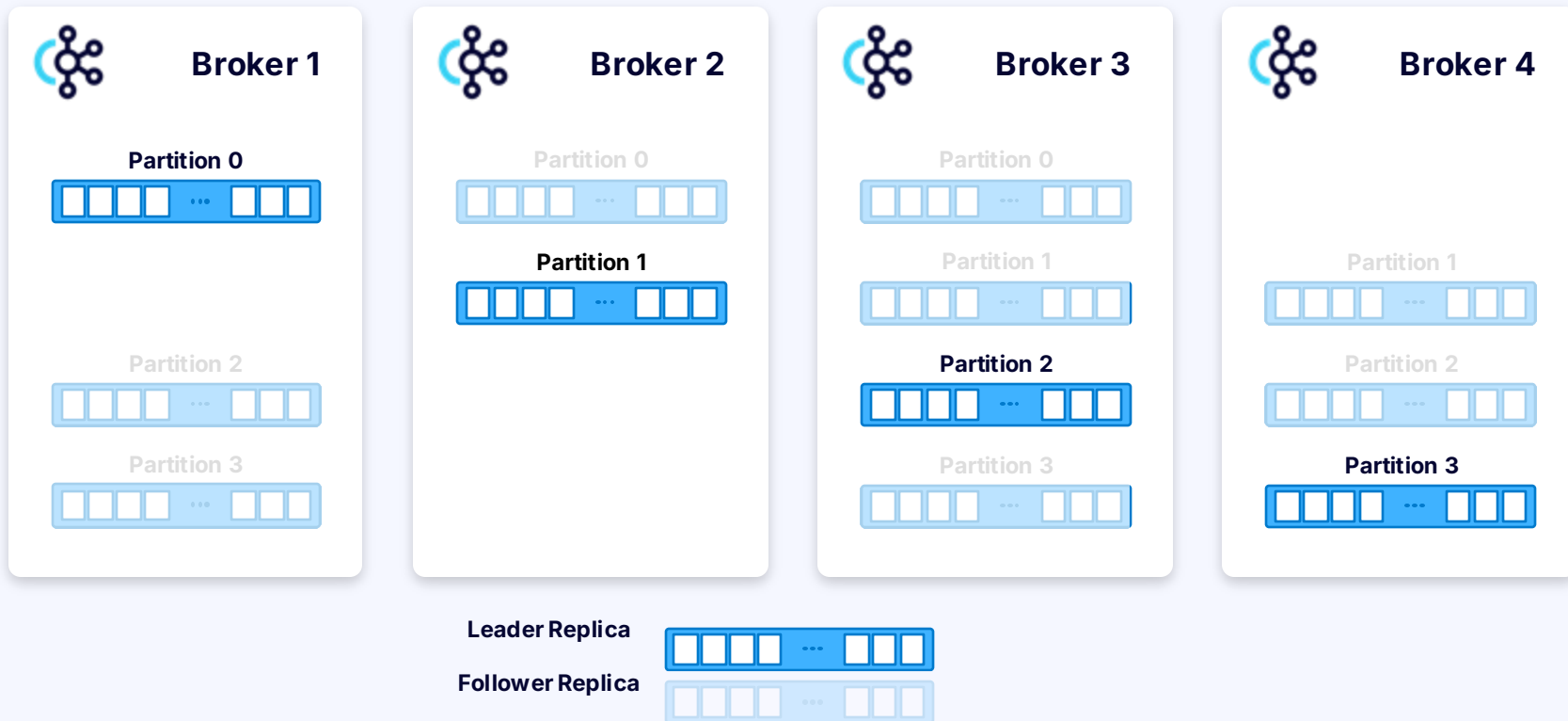


# Kafka replicates partitions across brokers





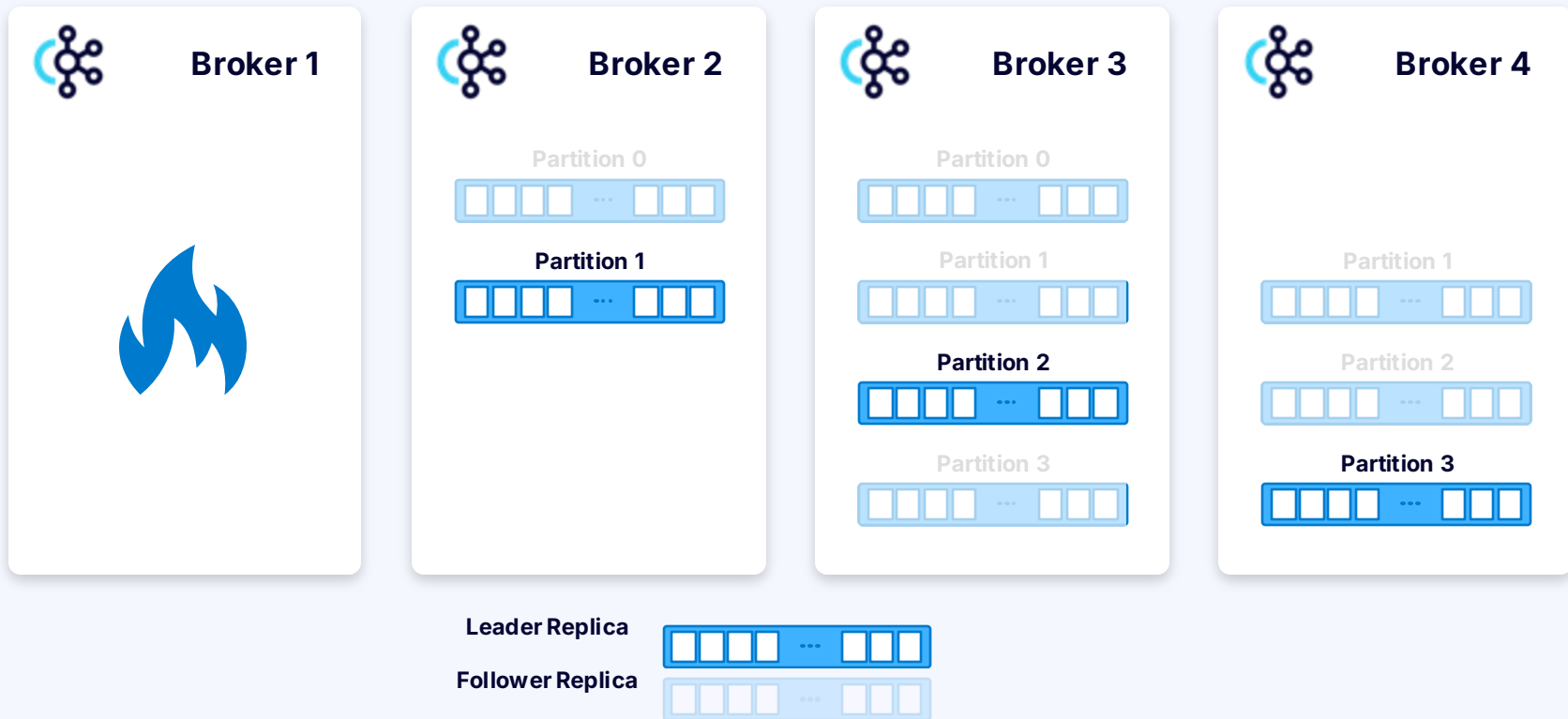
# Kafka replicates partitions across brokers







# Kafka replicates partitions across brokers



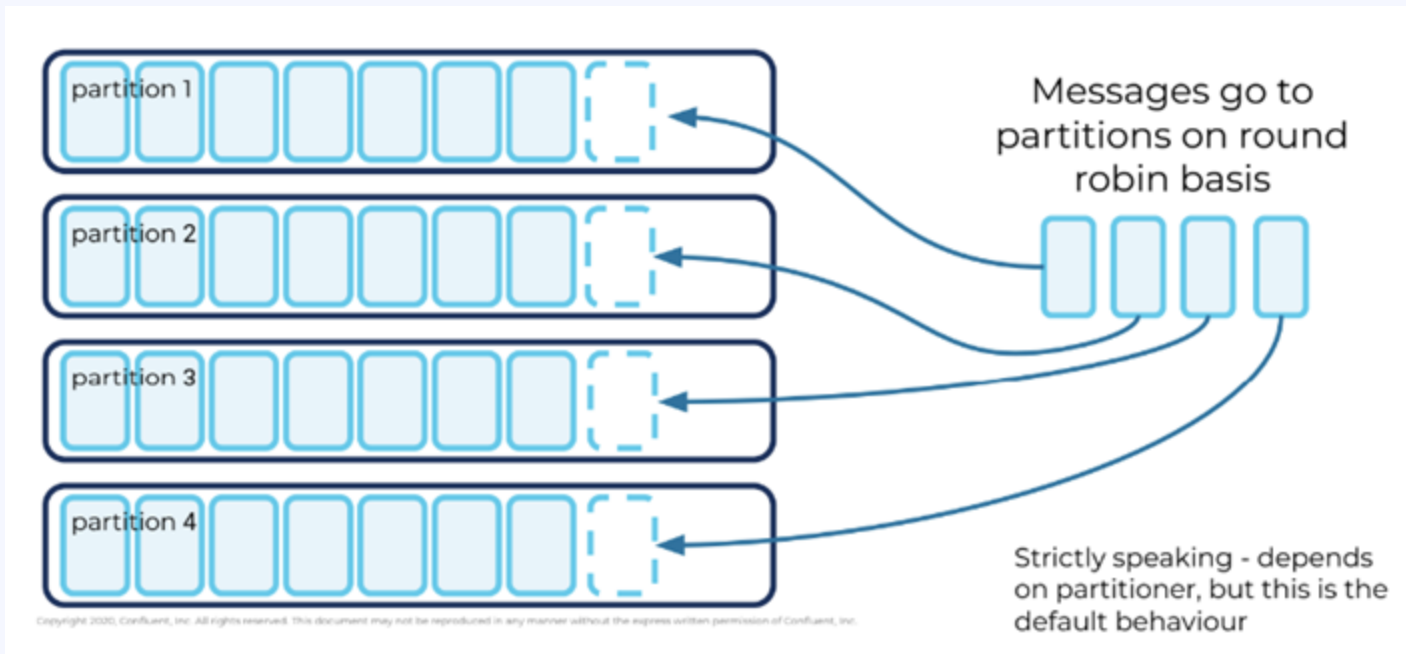


# Kafka Producer API

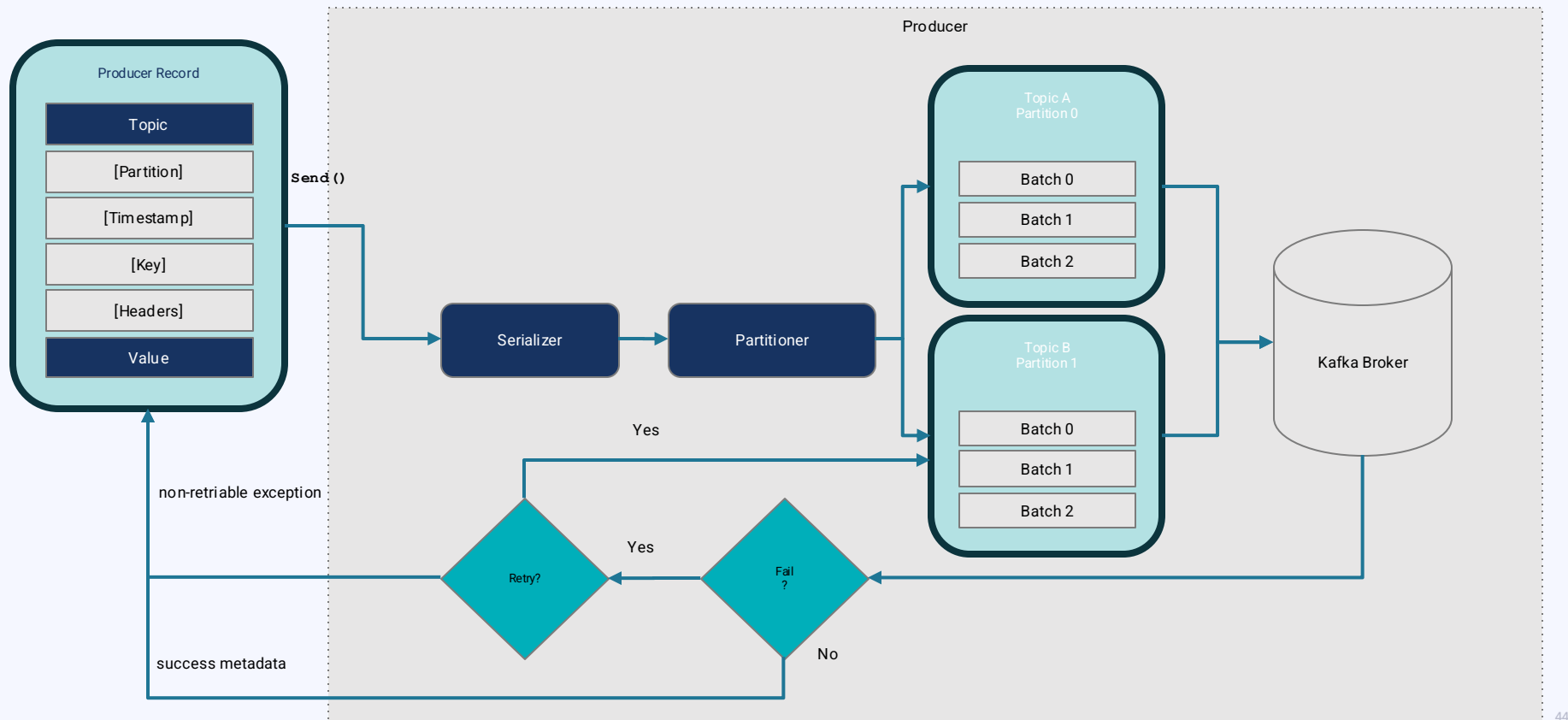
# Kafka Producer API



Producers are the entry gate to Kafka, they don't just send messages they batch, retry, and ensure delivery semantics such as at-most-once, at-least-once, or exactly-once.



# What happens inside a producer?



# Key Configurations for Better Resiliency



## Pro Tips

- Always use `acks=all` unless absolutely OK with potential data loss
- Combine with `min.insync.replicas ≥ 2` and appropriate retry settings
- For strict ordering under retries, enforce idempotence and limit in-flight requests
- Understand that Kafka acknowledgements don't coat-to-disk, they're just memory-based unless FS is forced



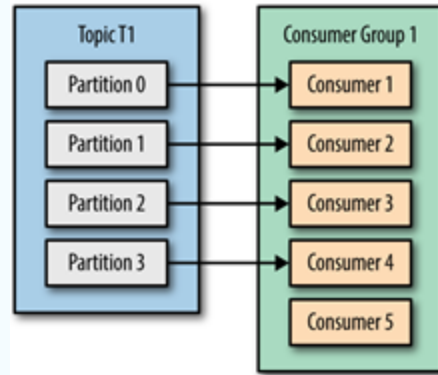
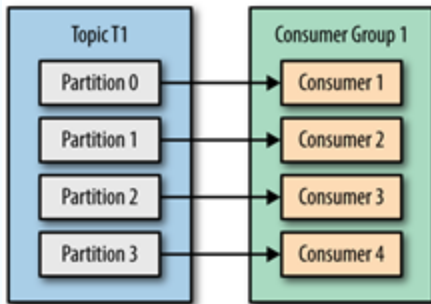
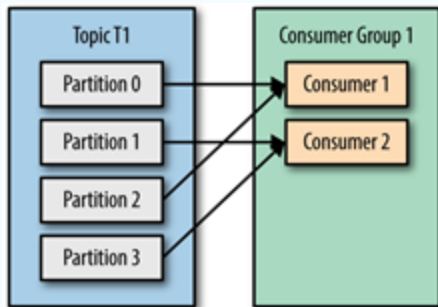
# Consuming from Kafka

# Consumers



## Partitions

- Basis for scalability
- No partition will be assigned to more than one consumer in the same group



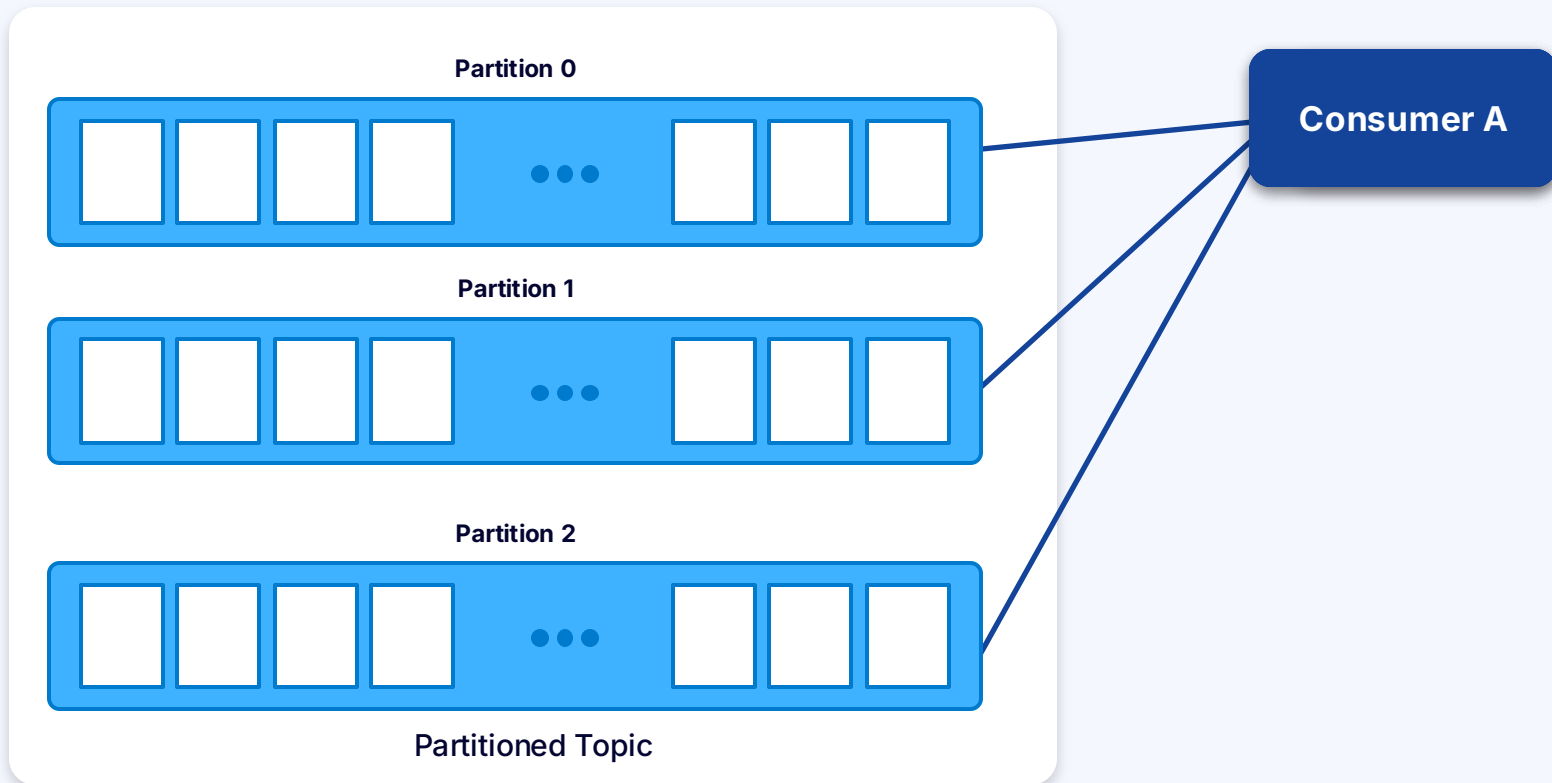
## Key parameters

```
# of partitions  
fetch.min.bytes=1  
fetch.max.wait.ms=500ms  
max.partition.fetch.bytes=10MB
```

```
fetch.max.bytes=50MB  
max.poll.records=500  
max.poll.interval.ms=5min  
auto.commit.interval.ms=5s (if being used)
```



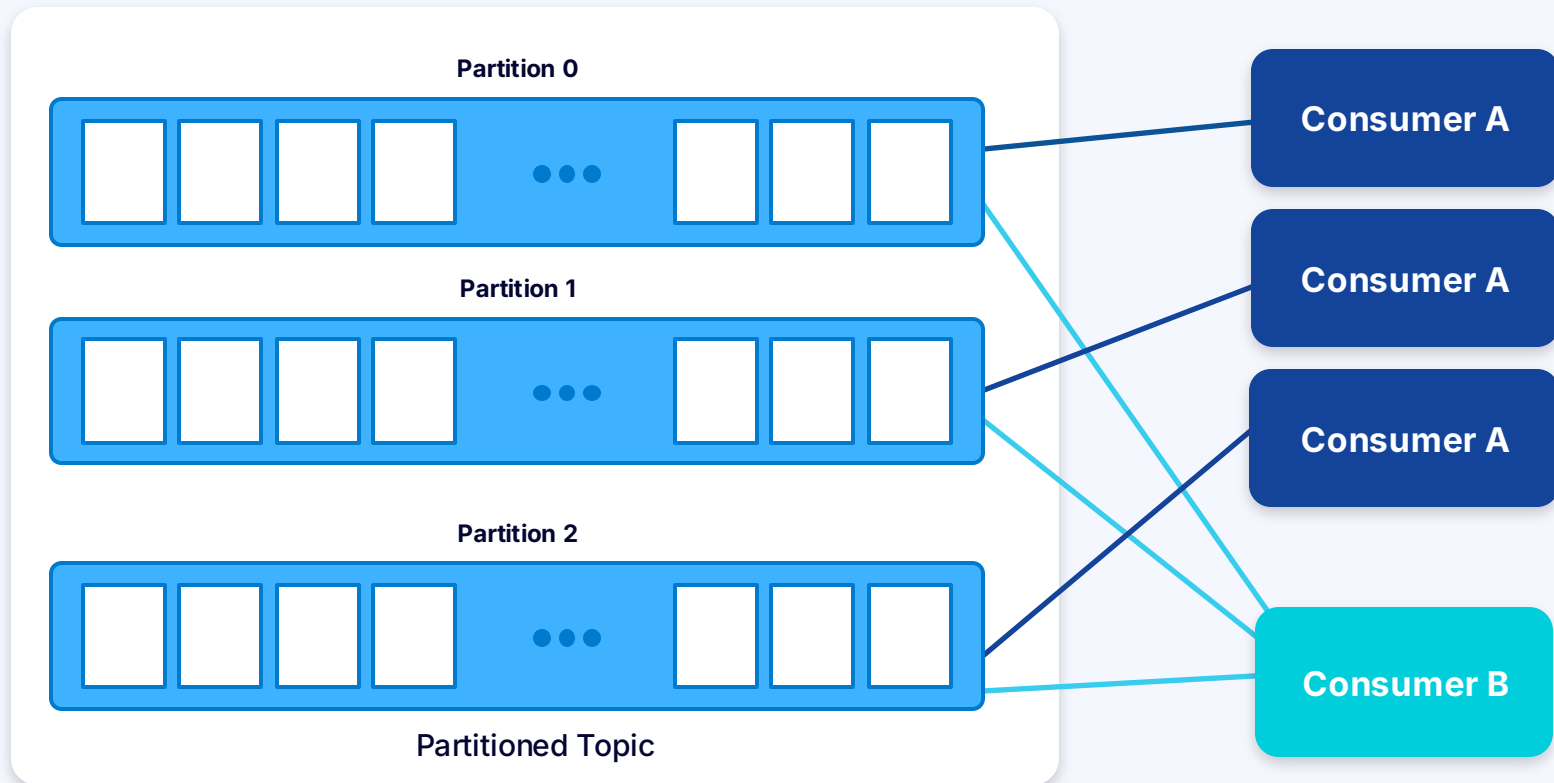
# Scaling with Consumer Groups





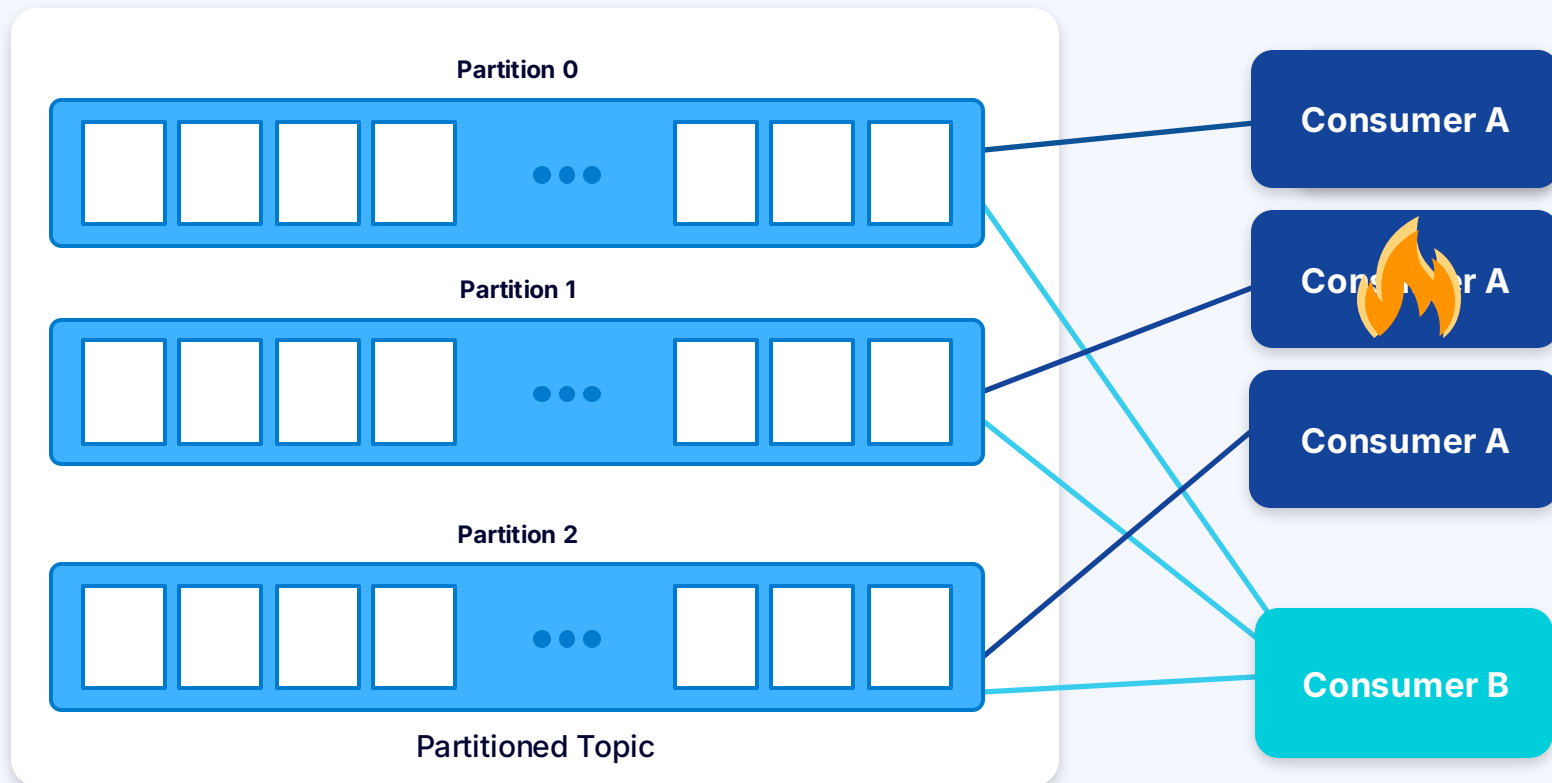


# Scaling with Consumer Groups



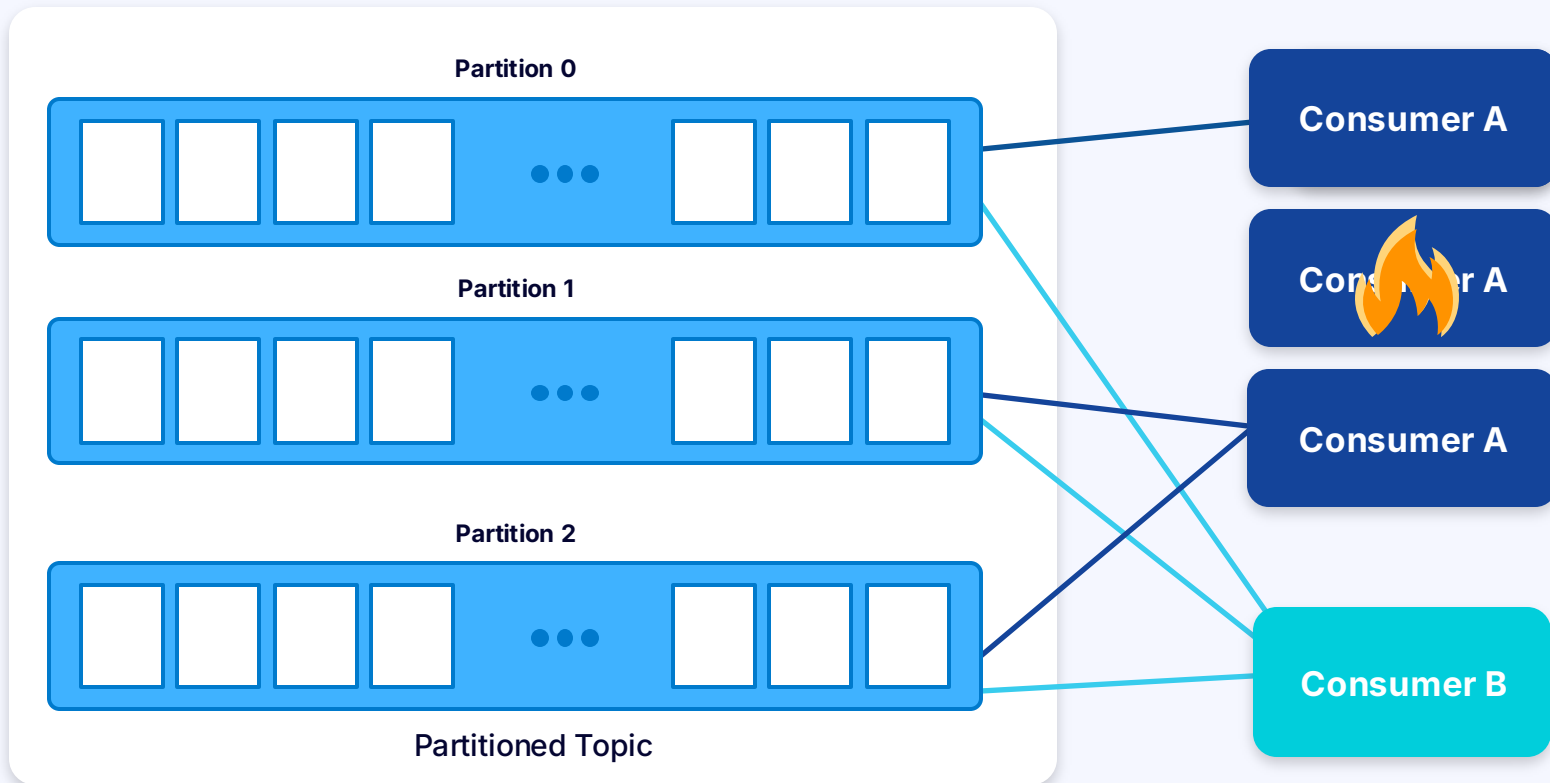


# Rebalancing





# Rebalancing



# Key Configurations and Concepts



## Key Concepts to Highlight:

- `poll()` loop fundamentals
- Consumer group membership & partition assignment
- Offsets: when/how to commit (`commitSync()` vs `commitAsync()`)
- Heartbeat vs processing heartbeat intervals
- Critical configs: `max.poll.interval.ms`, `session.timeout.ms`, `heartbeat.interval.ms`, `max.poll.records`



# Kafka Connect

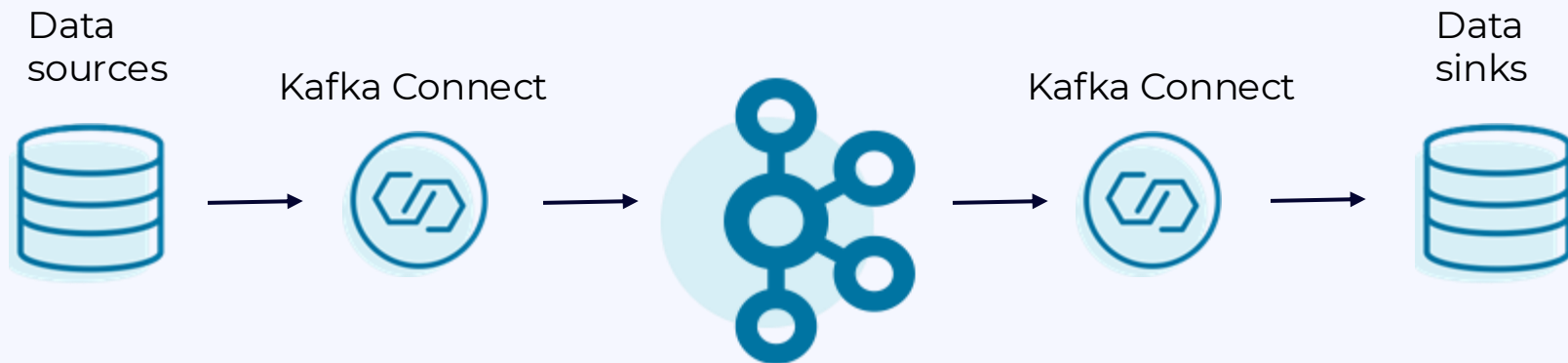
No/Low Code connectivity to many systems

# Kafka Connect



No-Code way of connecting known systems (databases, object storage, queues, etc) to Apache Kafka

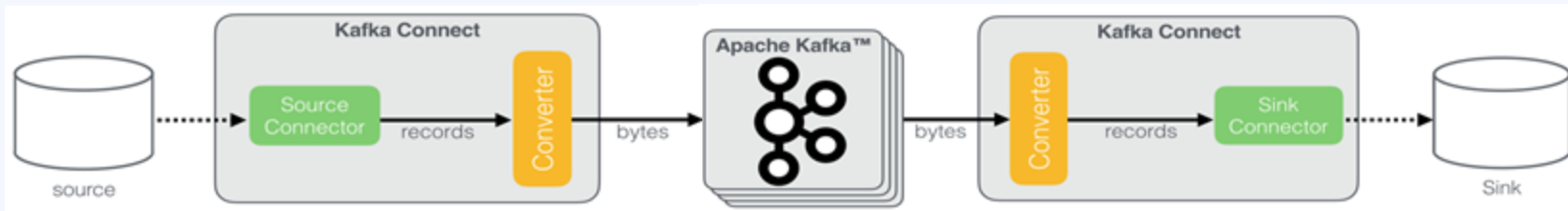
Some code can be written to do custom transforms and data conversions though maybe out of the box Single Message Transforms and Converters exist



# Kafka Connect



Convert between the **source and sink record objects** and the **binary format** used to persist them in Kafka.



JSON, Avro, Protobuf and Others

Learn Apache Kafka®  
with Confluent



Join your local Kafka User Group!



[meetup.com/pune-kafka/](https://meetup.com/pune-kafka/)

Ask questions, share knowledge and chat  
with your fellow community members!



[cnfl.io/ask-the-community](https://cnfl.io/ask-the-community)

## Try Confluent Cloud Today!

Get \$400 worth of free credits for  
your first 30 Days.



**SKIP THE PAYWALL**

+ another \$25 in Credits!

Promo Code

**CONFLUENTDEV1**



[cnfl.io/cloudmeetupgift](https://cnfl.io/cloudmeetupgift)



# Integrating Kafka with ClickHouse

$(\text{real-time})^2$

# Existing Kafka Integration Options

## 01 Kafka Table Engine

```
CREATE TABLE queue (  
    timestamp UInt64,  
    level String,  
    message String  
)  
ENGINE = Kafka('localhost:9092', 'topic', 'group1', 'JSONEachRow');
```

## 02 ClickHouse Kafka Connect Sink

### JDBC Connector (Source and Sink)

### HTTP Sink Connector

### ClickHouse Kafka Connect Sink

the official Kafka Connect Sink connector for ClickHouse.

#### Installation

##### Confluent Hub CLI installation

Use the [Confluent Hub client](#) to install this connector with:

```
confluent-hub install clickhouse/clickhouse-kafka-connect:0.6.12
```

##### Download installation

Or download the ZIP file and extract it into one of the directories that is listed on the Connect worker's `plugin.path` configuration properties. This must be done on each of the installations where Connect will be run.

[Download](#)

Plugin type: Sink

Enterprise support: [Q](#)

## 03 ClickPipes

Cloud / Data Ingestion

### ClickPipes

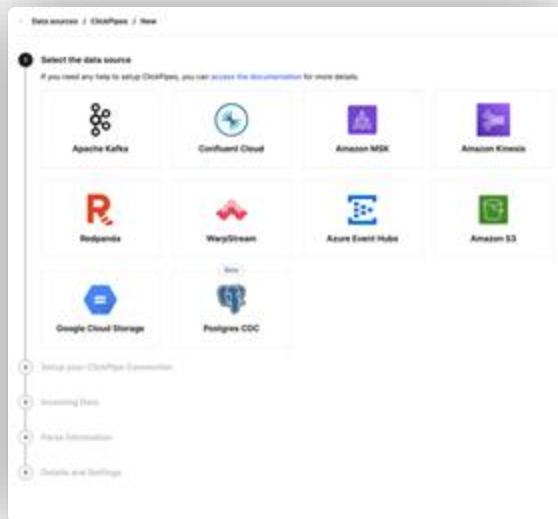
An integration engine that makes ingesting massive volumes of data from a diverse set of sources as simple as clicking a few buttons. Only available in ClickHouse Cloud.

[Get started today](#)

[View documentation](#)

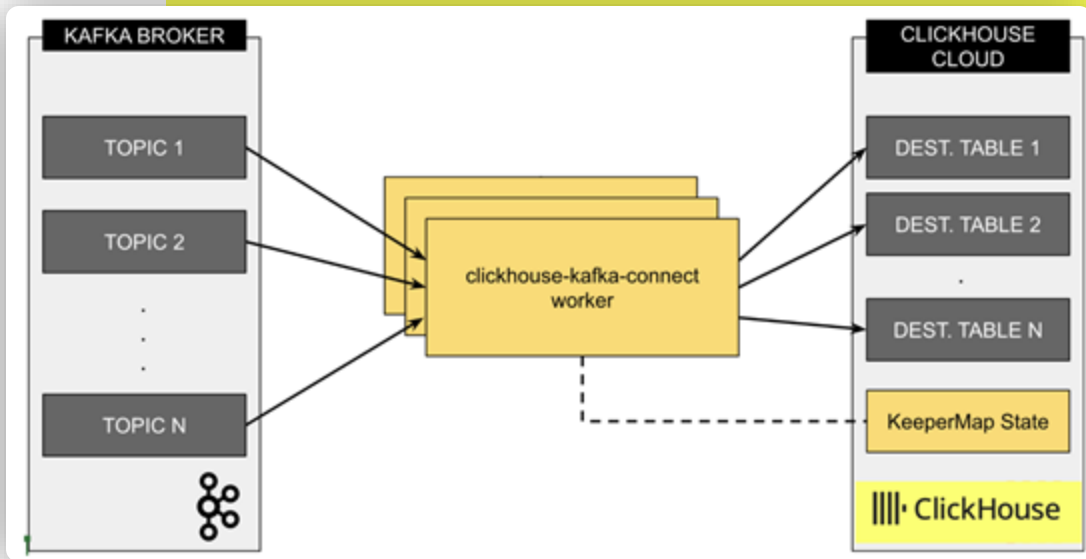


Blazing-fast Postgres to ClickHouse CDC with our new ClickPipe connector — now in Private Preview. [Learn more](#)



## ClickHouse Kafka Connect Sink

- Shipped with out-of-the-box **exactly-once semantics** powered by ClickHouse core feature named **KeeperMap** (table engine used as a state store by the connector) and allows for minimalistic architecture
- **Core integration:** Built, maintained, and supported by ClickHouse
- Tested **continuously** against ClickHouse Cloud.
- Data inserts with a **declared schema** (schema-based data, e.g. Avro, Protobuf, etc.) and **schemaless** (e.g. JSON)
- Support for all data types of ClickHouse



ClickPipes *noun*  
/klik paɪps/

Cloud-Native experience  
to ingest data from  
remote data sources to  
ClickHouse Cloud



ClickHouse

clickpipes-test

SQL Console

Dashboards

Data sources

Backups

Settings

Monitoring

Help

Connect

Organization

Integrations team

Integrations

Chat with support

All systems operational

Data sources / ClickPipes / New

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

Postgres CDC

2 Setup your ClickPipe Connection

If you need any help to connect to Confluent [access the documentation](#) for more detail

Integration name

clickpipe

Description

Custom Confluent Ingestion

API key

60EYKZV7MDC7WNG

API secret

.....

SASL Mechanism

SASL\_PLAIN

Consumer group

clickpipes-58743992-8a79-43c5-882a-c703c7b381ca

Servers

pkc-312vc0-ap-southeast-1.amazonaws.com:9092

SSL certificate (Beta)

Read Docs

Use Schema Registry

Read Docs

Back

Next: Incoming Data

3 Incoming Data

Topic

topic\_0

Offset selection (Beta)

From beginning

We expect JSON format

If a schema registry is not provided, we expect JSON format in

Sample Data

Sample displayed from partition: 4

Offset: 0

Timestamp: 173388103372

```
{
  "time": 231,
  "agent": "MacOS/5.8 (Windows NT 10.0; Win64; x64) ",
  "ip": "100.145.8.144",
  "referrer": "",
  "referrer_user": "",
  "request": "GET /v1/status.html HTTP/1.1",
  "status": "404",
  "time": "231",
  "user_id": 1
}
```

4 Parse Information

Now you need to create a new table or select an existing one where the incoming data can be stored. The table would have columns for each relevant piece of information such as date, amount, payer, and payment method.

New table

Existing table

Database

default

Table

topic\_0

Setting key

Select an option

Column settings

Name	Type	Default value	Nullable()
time	int64	0	<input type="checkbox"/>
agent	String	0	<input type="checkbox"/>
ip	int64	0	<input type="checkbox"/>
referrer	String	0	<input type="checkbox"/>
referrer_user	String	0	<input type="checkbox"/>
request	String	0	<input type="checkbox"/>
status	int64	1	<input type="checkbox"/>
time	int64	0	<input type="checkbox"/>
user_id	int64	0	<input type="checkbox"/>

Advanced settings

Back

Next: Details and Settings

# ClickPipes for Confluent Cloud

62



# Connect with ClickHouse



GitHub



ClickHouse  
Academy

## ClickHouse Cloud free trial



***\$100 additional credits***  
(total \$400 trial credits  
for 30 days)

## 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

## We are Hiring. Come Work with Us!



# Questions