

# Prometheus long-term storage on ClickHouse

If you already run ClickHouse, you already run your Prometheus long-term storage. Native support is arriving, and the gaps are small.

# ClickHouse, a popular analytics database

## WHAT IT IS

- **Column-oriented database** built for real-time analytics (OLAP)
- **Plain SQL** over large datasets, with fast scans and high compression
- **Open source since 2016**, Apache 2.0, written in C++
- **Born at Yandex**, now ClickHouse Inc.: self-hosted or ClickHouse Cloud

## HOW POPULAR

**50k+**

GitHub stars on ClickHouse/ClickHouse<sup>1</sup>

**2.6k+**

contributors over ten years of open source<sup>2</sup>

Sources: <sup>1</sup> GitHub, October 2026 · <sup>2</sup> ClickHouse, Open House 2026 recap

# Long-term storage, in moving parts

|                 | Thanos                                                | Mimir                                                                         | ClickHouse                                        |
|-----------------|-------------------------------------------------------|-------------------------------------------------------------------------------|---------------------------------------------------|
| Services to run | Sidecar or Receive, Querier, Store Gateway, Compactor | Distributor, Ingester, Querier, Store Gateway, Compactor (or monolithic mode) | clickhouse-server, plus Keeper when you replicate |
| Object storage  | Required                                              | Required                                                                      | Optional: just one disk tier                      |
| Query           | PromQL                                                | PromQL                                                                        | PromQL (subset) and SQL                           |
| Scaling         | Per component                                         | Per component                                                                 | Add replicas; shards need planning                |

Already running ClickHouse? **You already run your long-term storage.**

# ClickHouse speaks Prometheus natively

```
-- ClickHouse
SET enable_time_series_table = 1;
CREATE TABLE prometheus.metrics
ENGINE = TimeSeries;

# prometheus.yml
remote_write:
- url: https://ch:8443/prometheus/api/v1/write
  ?database=prometheus&table=metrics

# Grafana: Prometheus datasource on /prometheus
```

## 24.8 LTS · Aug 2024

TimeSeries table engine lands as experimental: remote write and remote read.

## 2026

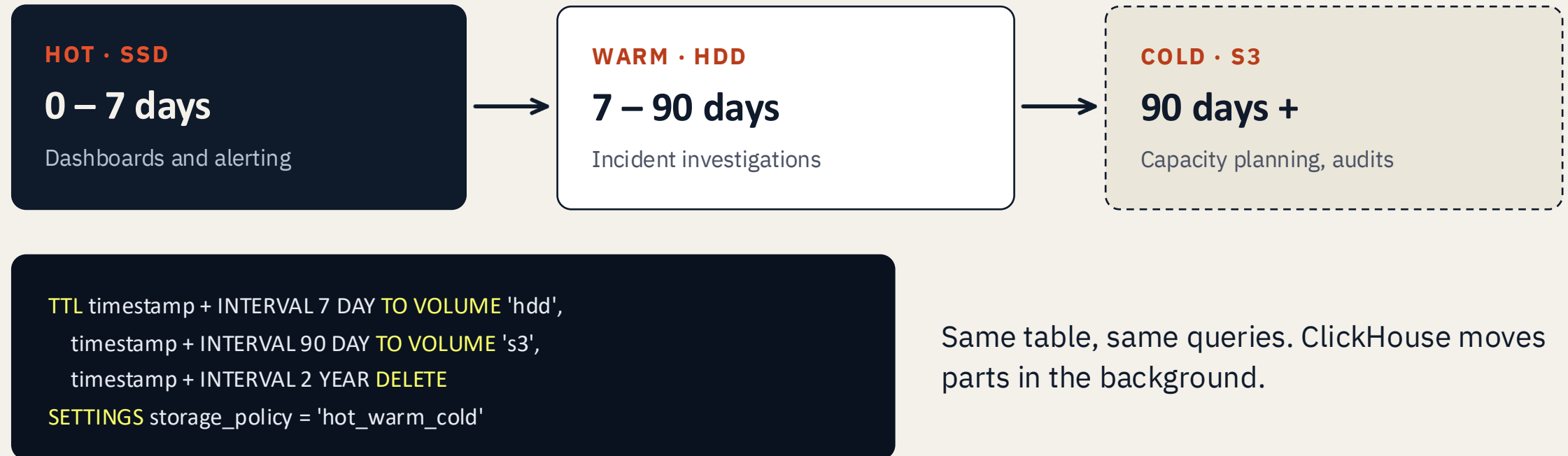
PromQL HTTP API: query, query\_range, series, labels, metadata. Grafana works unchanged.

## 26.9 · Sep 2026

Promoted from experimental to private preview: enable\_time\_series\_table. Not GA yet.

BUILT INTO CLICKHOUSE · A FEW LINES OF CONFIG

# Hot on SSD, warm on HDD, cold on S3



Durations are examples · storage policy defined in the server config

# What SQL adds on top of PromQL

## Cardinality forensics

Which metric or label is eating your storage.

```
SELECT metric_name, count()  
FROM timeSeriesTags(  
    prometheus.metrics)  
GROUP BY 1 ORDER BY 2 DESC
```

## Join business data

CPU cost per customer, SLA per contract.

```
SELECT c.name, avg(value)  
FROM cpu_usage u  
JOIN customers c  
    USING (tenant)
```

## PromQL inside SQL

Filter, join and export PromQL results.

```
SELECT * FROM  
prometheusQuery(  
    prometheus.metrics,  
    'rate(up[5m]]', now())
```



PATH 2 · A THIN LAYER ON TOP

# What ClickHouse doesn't do for you (yet)

## A stable Prometheus API

Remote write, remote read and PromQL, while the native engine matures.

## Rollups PromQL uses

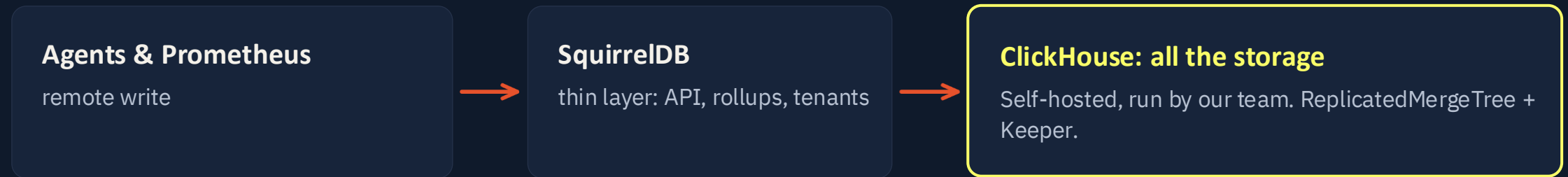
15-minute buckets kept up to date on insert; long ranges read them automatically.

## Multi-tenancy

Tenant isolation, per-series TTL, mutable labels.

**Pure ClickHouse rollups?** MATERIALIZED VIEW → AggregatingMergeTree, queried in SQL

# In production at Bleemeo



---

## 1.5 B

per point on disk, on par with the Prometheus TSDB

---

### A node restarts?

Replication catches it up on its own.

---

### Need to grow?

Add a replica; ClickHouse copies the data.

# Three takeaways

01

A credible Prometheus long-term store; the native path is maturing.

02

Tiering and retention are table settings, not extra services.

03

SQL turns your metrics into data you can join, audit and analyze.

**Trade-offs, honestly:** PromQL is a subset · native engine in preview · you run ClickHouse yourself.



**Try it on your next retention problem, and tell us what you build.**

[clickhouse.com/docs](https://clickhouse.com/docs) → Prometheus HTTP API and PromQL

Multi-tenancy or PromQL rollups today: [github.com/bleemeo/squirrelldb](https://github.com/bleemeo/squirrelldb)

Lionel Porcheron · [github.com/lporcheron](https://github.com/lporcheron)