

When a relational database is not enough!

Scaling your history data...



Thomas Oftring

Head of Monitoring and Database



[Linkedin \(get connected\)](#)



OCTOBER 7 ▪ 9, 2026
RIGA ▪ LATVIA



PostgreSQL/TimescaleDb

The rock of the block

ZABBIX
SUMMIT²⁶
OCTOBER 7 ■ 9, 2026
RIGA ■ LATVIA



PostgreSQL

The trusted veteran

Big pool of established tools

- Command line tools
- UI tools

Well known to

- SQL Developers
- Database Administrators

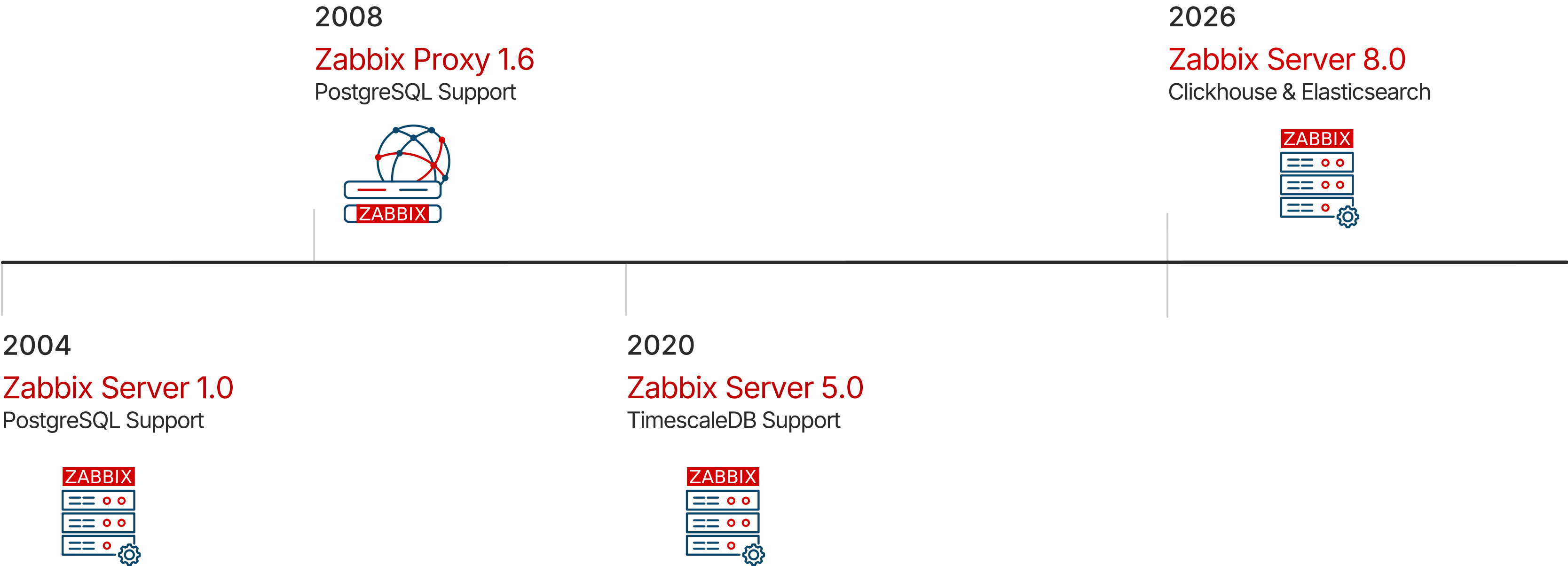
Lot of extensions/integrations

- TimescaleDb ;-)
- Foreign Data Wrappers
etc. 1000+

A rock as relational database

- ACID
- Transaction safe

Zabbix Database Engine Support (PostgreSQL and beyond)



ClickHouse

The new kid on the block

ZABBIX

SUMMIT²⁶

OCTOBER 7 ▪ 9, 2026

RIGA ▪ LATVIA

||||

ClickHouse

ClickHouse Trade Offs

Fewer tools

- Tools from ClickHouse itself
- Increasing ecosystem

Learning Curve

- Some things are different
- SQL query language available but more analytical
 - ARRAY JOIN
 - ASOF JOIN
 - ANY/ALL

Fewer integrations

- 60+ federations to other systems
- 100+ data formats for read/write

Different operational ways

- Limited transactions
- Shards
- Replicas

Arrived for the Big Data Era

Columnar storage engine

- Excellent compression
- Very fast aggregations
- Can skip irrelevant data

Economic Appeal

- Low hardware requirements
- Simplicity

Real time analytics

- 100 – 1000 x faster workloads
- Sub-second response on billion-rows
- Made for aggregations

Made and designed for

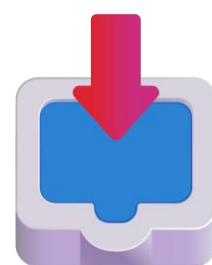
- Logs
- Metrics
- Tracing data

Why move the history to ClickHouse



**Faster Queries
on Time Series**

**Faster
Zabbix dashboards
Zabbix graphs**



**High Frequency
Ingestion at Scale**

**Sustains heavy writes
No impact on reads**



**Offloads
PostgreSQL**

**More resources
for configuration
and
Zabbix Server logic**



**Scales
Horizontally**

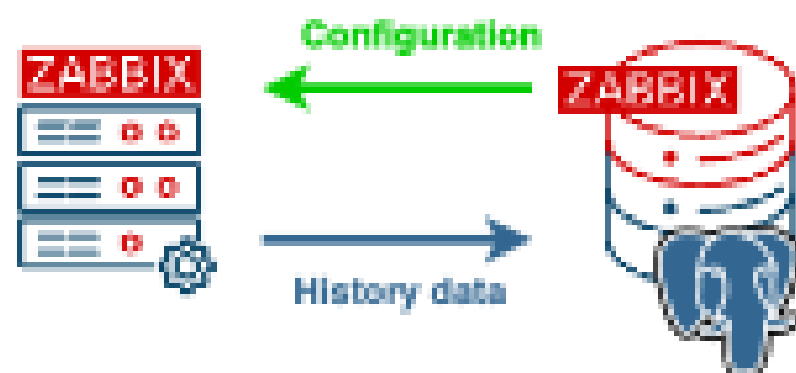
**Scale horizontally
when needed
(Shards)**

Zabbix 8.0

How to upgrade – Prepare ClickHouse



Upgrade to Zabbix 8.0 and History Data Migration



ClickHouse



Zabbix Server

- Follow the usual upgrade path to 8.0
- Check logs if all is running
- Let Zabbix collect data to PostgreSQL

Install & Configure ClickHouse

- Prepare everything for ClickHouse
- Configure ClickHouse
- Check that everything is working
- Get familiar with the new database system

History Data Migration

- Option 1:
Migrate Data to ClickHouse Offline
- Option 2:
Migrate Data Online with CDC to reduce downtime

Zabbix 8.0

History Migration to ClickHouse



two ways?!?



Install and configure ClickHouse

Install and setup service

- Use the official ClickHouse repository
- Use official ClickHouse container

Check config files

- Configure at least config.xml
- Pay attention to logging

Create Zabbix database and user

- Database names are case sensitive
- User accounts are case sensitive

Create tables in the database

- Execute Zabbix deployment script
- Change TTL in config file to your needs
- Check created tables

```
# Create tables in the ClickHouse database
./usr/share/zabbix/sql-scripts/clickhouse/history_all.sh \
--user zabbix \
--password <password> \
--db zabbix \
--server http://localhost:8123

clickhouse-client --user zabbix --password <password> \
--query "SHOW TABLES FROM zabbix;"

# history
# history_json
# history_log
# history_str
# history_text
# history_uint
```

History Data Migration

With Downtime



History Data Migration with Downtime

Stop all Zabbix services

- Stop all Zabbix server services and Web Frontends
- Prepare configuration files

Get migration scripts from Zabbix Git repo

- Check the README file
- Make sure to have enough space for export

Export the data from PostgreSQL

- Execute Zabbix export script
- Execute parallel if your export disk is fast enough

Import data to ClickHouse

- Execute Zabbix import script
- Pay attention to the ClickHouse logs
- Truncate unneeded ClickHouse logs

Export PostgreSQL history tables (export dir is /tmp no parameter)

```
psql -U zabbix -d zabbix -h 127.0.0.1 -f history_export_pgsql.sql
```

```
psql -U zabbix -d zabbix -h 127.0.0.1 -f history_json_export_pgsql.sql
```

```
psql -U zabbix -d zabbix -h 127.0.0.1 -f history_log_export_pgsql.sql
```

```
psql -U zabbix -d zabbix -h 127.0.0.1 -f history_str_export_pgsql.sql
```

```
psql -U zabbix -d zabbix -h 127.0.0.1 -f history_text_export_pgsql.sql
```

```
psql -U zabbix -d zabbix -h 127.0.0.1 -f history_uint_export_pgsql.sql
```

Import history tables to ClickHouse

```
./history_import.sh -s http://localhost:8123 -d zabbix -u zabbix -p zabbix -i /tmp
```

```
./history_json_import.sh -s http://localhost:8123 -d zabbix -u zabbix -p zabbix -i /tmp
```

```
./history_log_import.sh -s http://localhost:8123 -d zabbix -u zabbix -p zabbix -i /tmp
```

```
./history_str_import.sh -s http://localhost:8123 -d zabbix -u zabbix -p zabbix -i /tmp
```

```
./history_text_import.sh -s http://localhost:8123 -d zabbix -u zabbix -p zabbix -i /tmp
```

```
./history_uint_import.sh -s http://localhost:8123 -d zabbix -u zabbix -p zabbix -i /tmp
```

History Data Migration

Near-Zero Downtime with Change-Data-Caption (CDC)



History Data Migration with Near-Zero Downtime – Preparation

Install and prepare ClickHouse

- Create Zabbix database and user
- Create Zabbix tables

Prepare the existing PostgreSQL

- Check log level. Must be "logical"
- Create replication user
- Check pg_hba.conf for replication user

Create PostgreSQL publication

- Create publication only for needed tables
- Check that publication is working

```
-- postgresql.conf (requires restart)
```

```
-- wal_level = logical
```

```
-- max_replication_slots = 10
```

```
-- max_wal_senders = 10
```

```
-- Create a dedicated replication user
```

```
CREATE ROLE cdc_replicator WITH LOGIN REPLICATION  
PASSWORD 'StrongPassword!';
```

```
GRANT SELECT ON ALL TABLES IN SCHEMA public TO  
cdc_replicator;
```

```
-- Create a publication for the tables to migrate
```

```
CREATE PUBLICATION zbx_history_pub
```

```
FOR TABLE history, history_uint, history_json, history_str,  
history_text, history_log;
```

History Data Migration with PeerDB and CDC – Move the Data

Install and prepare PeerDB

- Clone git repo [PeerDB](#)
- Configure and spin up container (Docker/Podman)

Configure data source and target

- Configure PostgreSQL source
- Configure ClickHouse target

Configure mirror for data transfer

- Create mirror for every history table
- Avoid single mirror for all tables

-- Via PeerDB's SQL interface

-- (psql -h localhost -p 9900 -U peerdb peerdb)

```
CREATE MIRROR history_cdc_mirror FROM pg_source TO ch_target
FOR
$$
SELECT * FROM history
$$
WITH (
    destination_table_name = 'history',
    do_initial_snapshot = true,
    publication_name = 'zbx_history_pub'
);
```

History Data Migration with PeerDB and CDC – Replication Lag Monitoring

Track PostgreSQL

- Check replication slot
- Check replication state

Track ClickHouse

- Check latest values for all tables
- Should increase regularly if everything is ok

Track PeerDB mirror status

- Create publication only for needed tables
- Check that publication is working

[PeerDB mirror status](#)

```
# check PeerDB mirror status
```

```
# Available states:
```

```
# - STATUS_SETUP → Only during setup phase
```

```
# - STATUS_SNAPSHOT → During initial data load for history
```

```
# - STATUS_RUNNING → Change Data Capture (CDC) mode
```

```
SELECT mirror_name,  
       status,  
       workflow_id  
FROM peerdb_stats.mirror_status  
WHERE mirror_name = 'history_cdc_mirror';
```

History Data Migration with PeerDB and CDC – Final Cutover

Preparation

- Prepare Zabbix Server/Web Frontend configuration
- Confirm near zero CDC lag

Stop data input on PostgreSQL

- Stop Zabbix Server/Web Frontend
- Wait until CDC lag is zero all WALs deployed

Start Zabbix Environment with new config

- Start Web Frontend first and check graphs for history data
- Start Zabbix Server and pay attention to log

Stop CDC pipeline after sucessfull migration

- Disable and drop PeerDB mirror
- Drop PostgreSQL publication and replication slot

```
# Zabbix Server config (/etc/zabbix/zabbix_server.conf)
```

```
HistoryProvider=clickhouse;value_types="uint,dbl,str,log,text,json",  
url=http://localhost:8123,db=zabbix,username=zabbix,password=xx
```

```
# Zabbix Web Frontend config (/etc/zabbix/web/zabbix.conf.php)
```

```
$HISTORY_PROVIDERS[] = [  
    'types' ⇒ ['uint', 'dbl', 'str', 'log', 'text', 'json'],  
    'provider' ⇒ 'clickhouse',  
    'url' ⇒ 'http://localhost:8123',  
    'db' ⇒ 'zabbix',  
    'username' ⇒ 'zabbix',  
    'password' ⇒ 'Rms0104!'  
];
```

```
-- Via PeerDB's (psql -h localhost -p 9900 -U peerdb peerdb)
```

```
SELECT mirror_name, status, workflow_id  
FROM peerdb_stats.mirror_status  
WHERE mirror_name = 'history_cdc_mirror';
```

Backup & Restore – How to Handle It

PostgreSQL backup solutions

- [pg_backrest \(The proofed workhorse\)](#)
- [pg_hardstorage \(The new wave\)](#)
- [barman \(The command center\)](#)

ClickHouse backup solutions

- ClickHouse SQL BACKUP/RESTORE
- [clickhouse-backup](#)

How to fit it together

- Take PostgreSQL Backup
- Take ClickHouse Backup
- Restore PostgreSQL Backup to Point of Time
- Restore ClickHouse Backup
- Trim ClickHouse Database to PostgreSQL Point of Timer Recovery Timestamp

Restore PostgreSQL to an exact timestamp

```
pgbackrest --stanza=zabbix restore --type=time \  
--target="2026-10-09 17:00:00"
```

Restore ClickHouse nearest backup after PostgreSQL PITR

```
clickhouse-backup restore zabbix_backup_20260915_020000
```

-- Trim ClickHouse history* tables past the PostgreSQL restore point (T)
ALTER TABLE zabbix.history DELETE WHERE clock > toUnixTimestamp('2026-10-09 17:00:00');

Lessons Learned & Pitfalls



ClickHouse
Keep smiling
:)

ClickHouse "client"

... drives you crazy



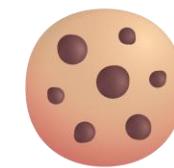
ClickHouse retention

... some Time-To-Live



ClickHouse system logs

... the cookie monster of disk space



ClickHouse HA

... do not forget High Availability



ClickHouse client

... drives you crazy



clickhouse

clickhouse client

clickhouse-client

clickhouse-client --host 127.0.0.1

```
← Progress: 0.00 rows, 0.00 B (0.00 rows/s., 0.00 B/s.)
↵ Progress: 0.00 rows, 0.00 B (0.00 rows/s., 0.00 B/s.)
↑ Progress: 5.00 rows, 101.00 B (4.59 thousand rows/s., 92.70 KB/s.)

  name
1. INFORMATION_SCHEMA
2. default
3. information_schema
4. system
5. zabbix_ghost

5 rows in set. Elapsed: 0.001 sec.

:) quit
Bye.
[root@tosummit ~]# clickhouse
ClickHouse local version 26.8.2.7 (official build).

:) SHOW DATABASES;

SHOW DATABASES

Query id: c531e3dc-c57f-4eba-b7bb-72ad51443460

→ Progress: 0.00 rows, 0.00 B (0.00 rows/s., 0.00 B/s.)
↵ Progress: 0.00 rows, 0.00 B (0.00 rows/s., 0.00 B/s.)
↓ Progress: 4.00 rows, 81.00 B (2.97 thousand rows/s., 60.20 KB/s.)

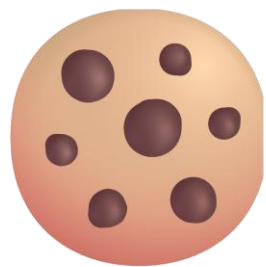
  name
1. INFORMATION_SCHEMA
2. default
3. information_schema
4. system

4 rows in set. Elapsed: 0.002 sec.

:) quit
Bye.
[root@tosummit ~]#
```

ClickHouse system logs

The cookie monster of disk space



ClickHouse is logging a lot of stuff:

- **system.query_log**
- **system.part_log**
- **system.text_log**

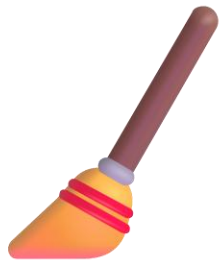
Query id: 91f56c15-cc44-4b7e-9a83-f87c693f45c4

	database	table	partition	rows	size
1.	system	part_log	202605	265046	21.21 MiB
2.	system	part_log	202606	7697017	687.72 MiB
3.	system	part_log	202607	53113997	4.98 GiB
4.	system	query_log	202608	347014232	32.83 GiB
5.	system	part_log	202608	55495091	5.11 GiB
6.	system	text_log	202608	1633427	13.21 MiB
7.	system	query_log	202609	37336890	3.58 GiB
8.	system	part_log	202609	4786917	453.31 MiB
9.	system	text_log	202609	127498	935.69 KiB

9 rows in set. Elapsed: 0.004 sec.

ClickHouse retention

Who does the work?



Zabbix Server
Housekeeper
... does nothing :-)

Clickhouse
Time-To-Live
... does the magic :-)

```
SHOW CREATE TABLE zabbix.history

Query id: 5861a6d5-45c3-439c-91d3-c521e57d197c

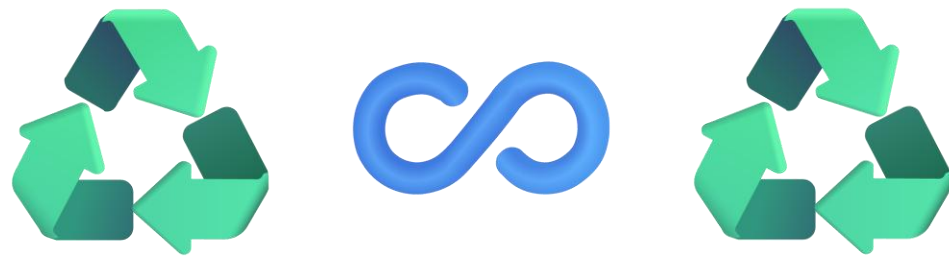
1. statement
CREATE TABLE zabbix.history
(
    `itemid` UInt64,
    `clock_ns` DateTime64(9),
    `value` Float64
)
ENGINE = MergeTree
PARTITION BY toYYYYMM(clock_ns)
PRIMARY KEY (itemid, clock_ns)
ORDER BY (itemid, clock_ns)
TTL clock_ns + toIntervalSecond(15552000)
SETTINGS index_granularity = 8192

1 row in set. Elapsed: 0.664 sec.
```

Numeric (float)	25w 5d
Character	25w 5d
Log	25w 5d
Numeric (unsigned)	25w 5d
Text	25w 5d
JSON	25w 5d

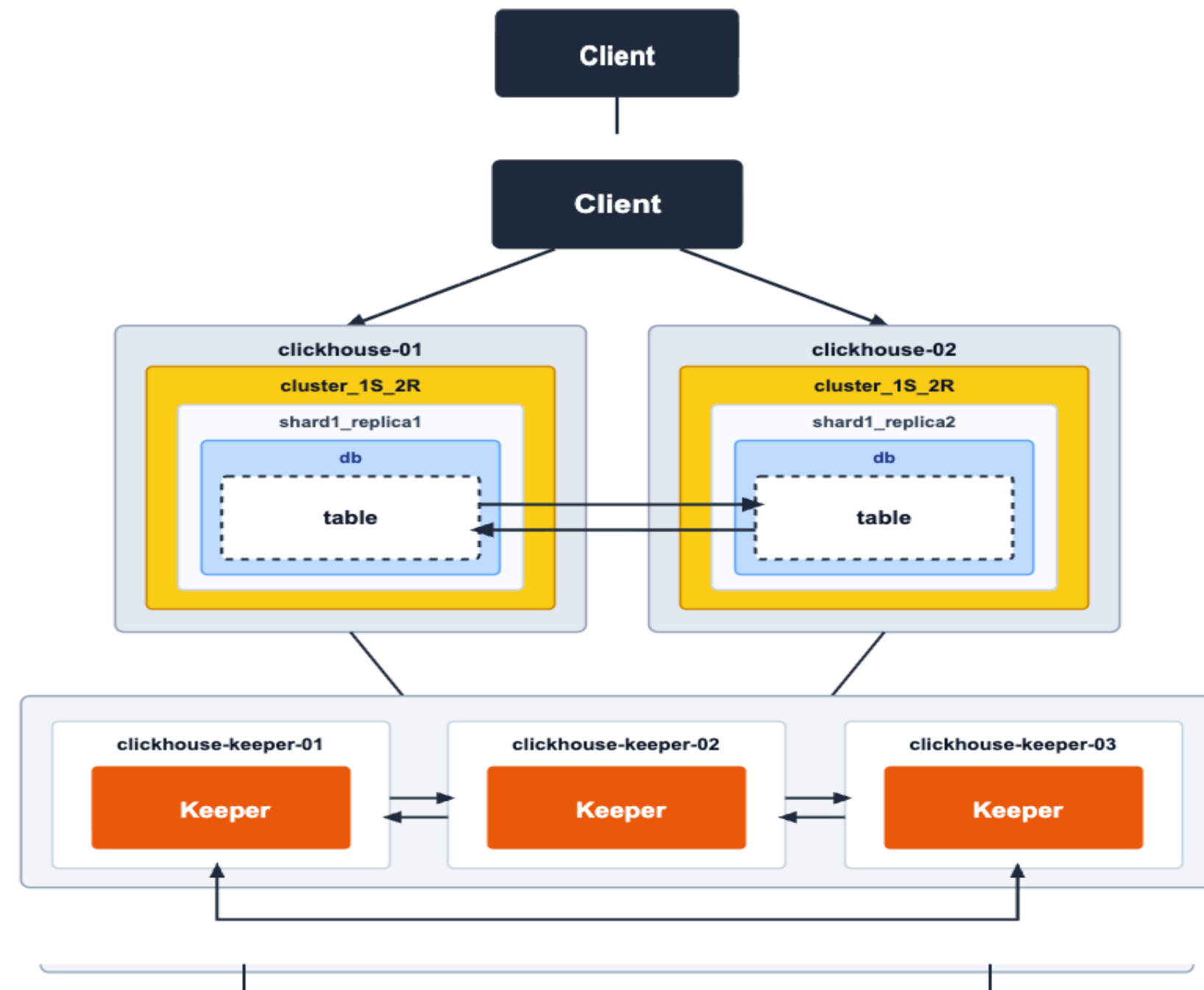
ClickHouse HA

Do not forget High Availability



ClickHouse

- **Shards**
used for scaling
- **Replicas**
used for replication



Don't put Zabbix brain in ClickHouse



Only put its longterm memory there!



Q&A

Thank you for your patience!



[Linkedin \(get connected\)](#)



OCTOBER 7 ▪ 9, 2026
RIGA ▪ LATVIA



Image sources

[ClickHouse Replication](#)

[ClickHouse Replication & Scaling](#)