

Turbocharging Zabbix Performance With Clickhouse Database Backend

Aleksandrs Petrovs-Gavrilovs, Senior Trainer and Engineer, Zabbix SIA



Why ClickHouse?

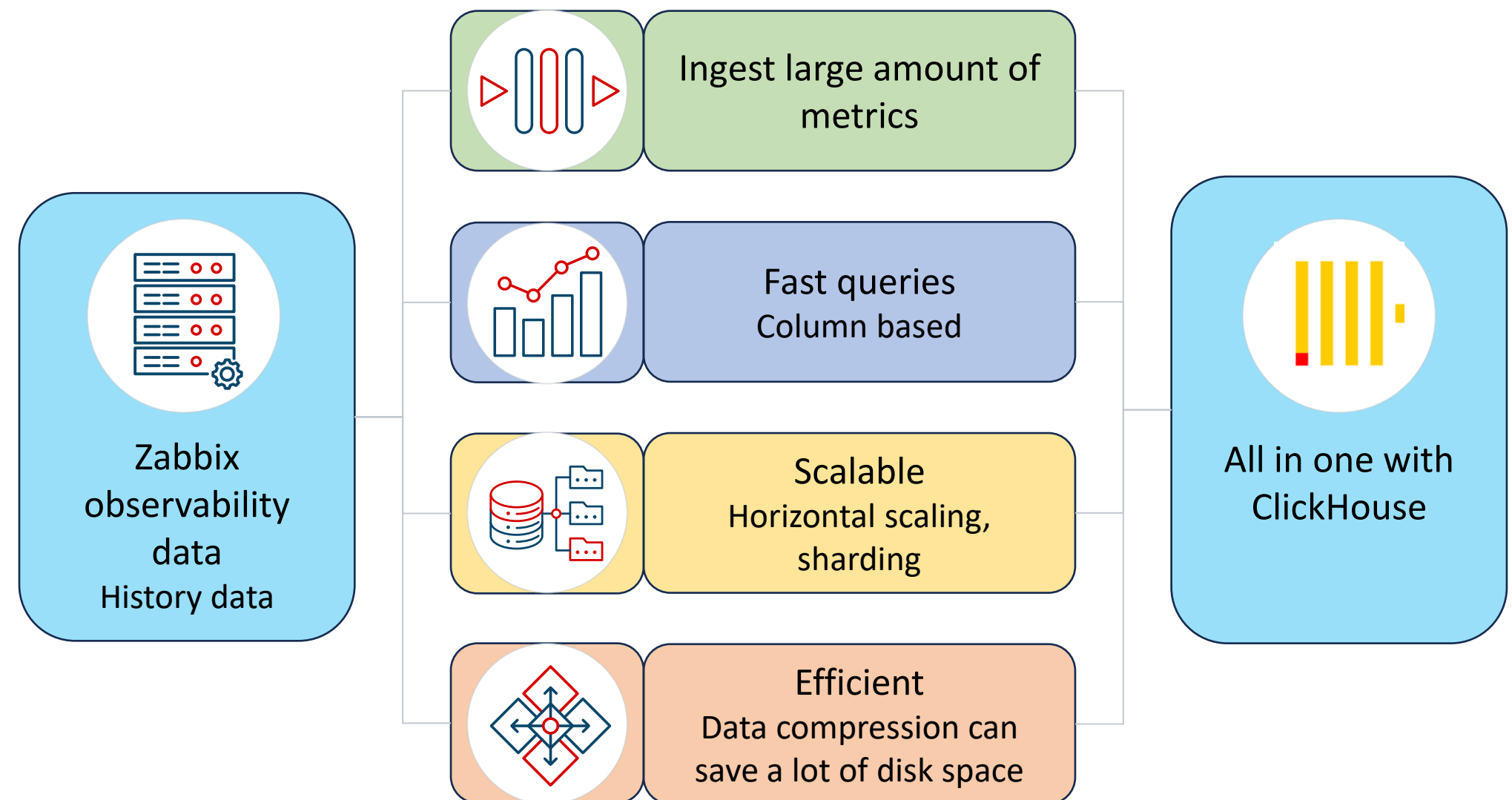


Scaling

Zabbix

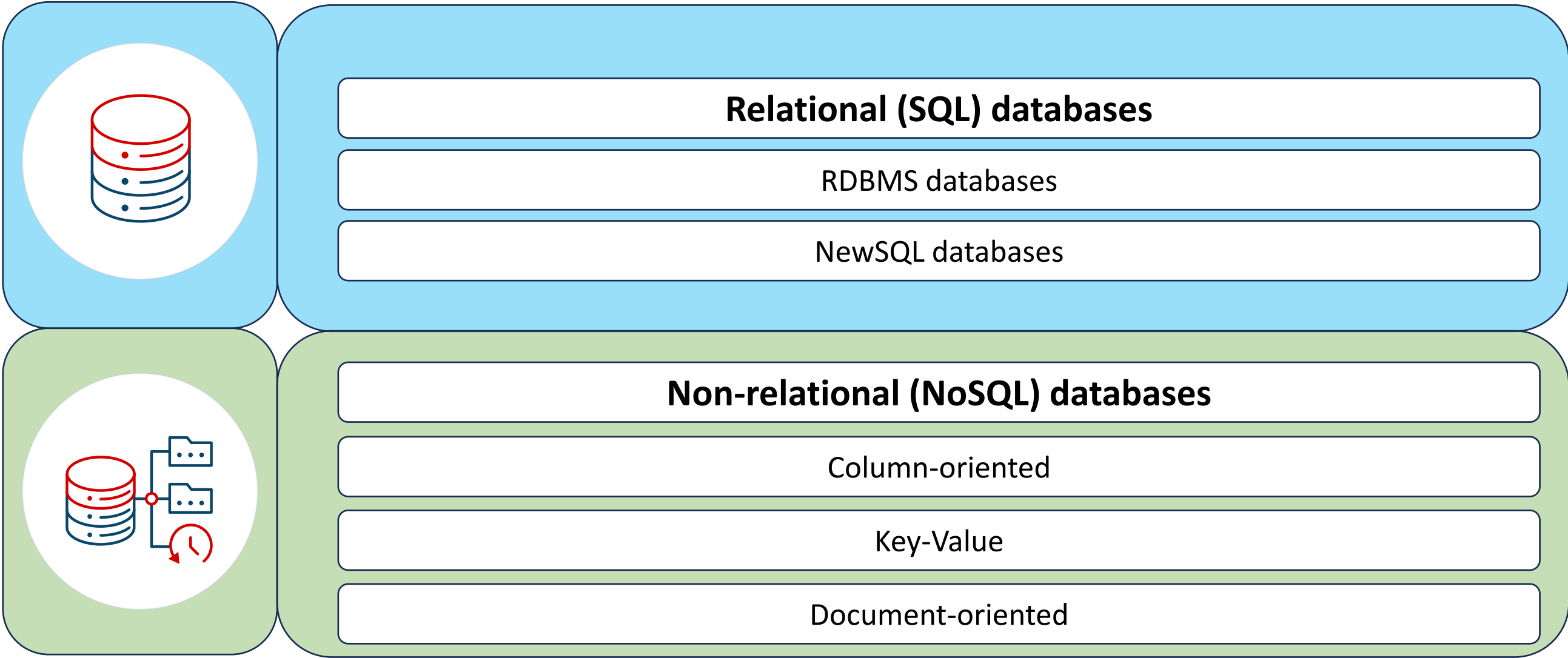
Using ClickHouse allows to ingest large amount of observability metrics

- **Fast Queries:** Column-based storage makes aggregation and summation queries fast and efficient.
- **Scalable:** ClickHouse scales horizontally by adding more nodes, with all nodes contributing to query processing (sharding)
- **Efficient Storage:** Compression can significantly reduce the amount of disk space required for stored data.



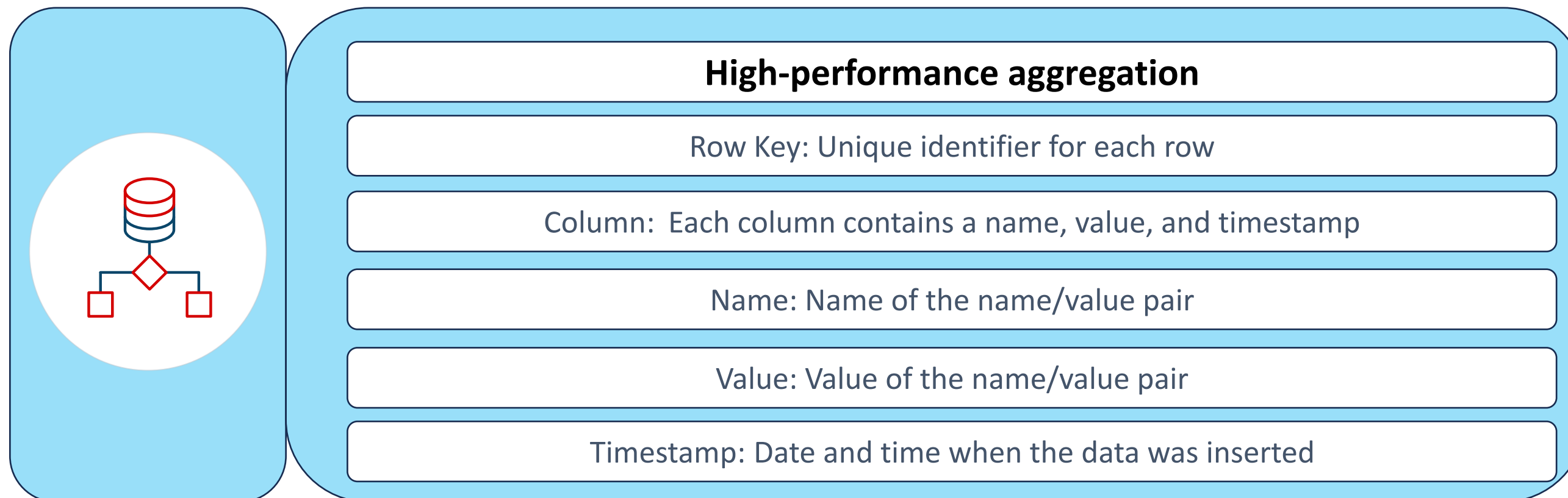
Diferrent DB types

SQL and NoSQL



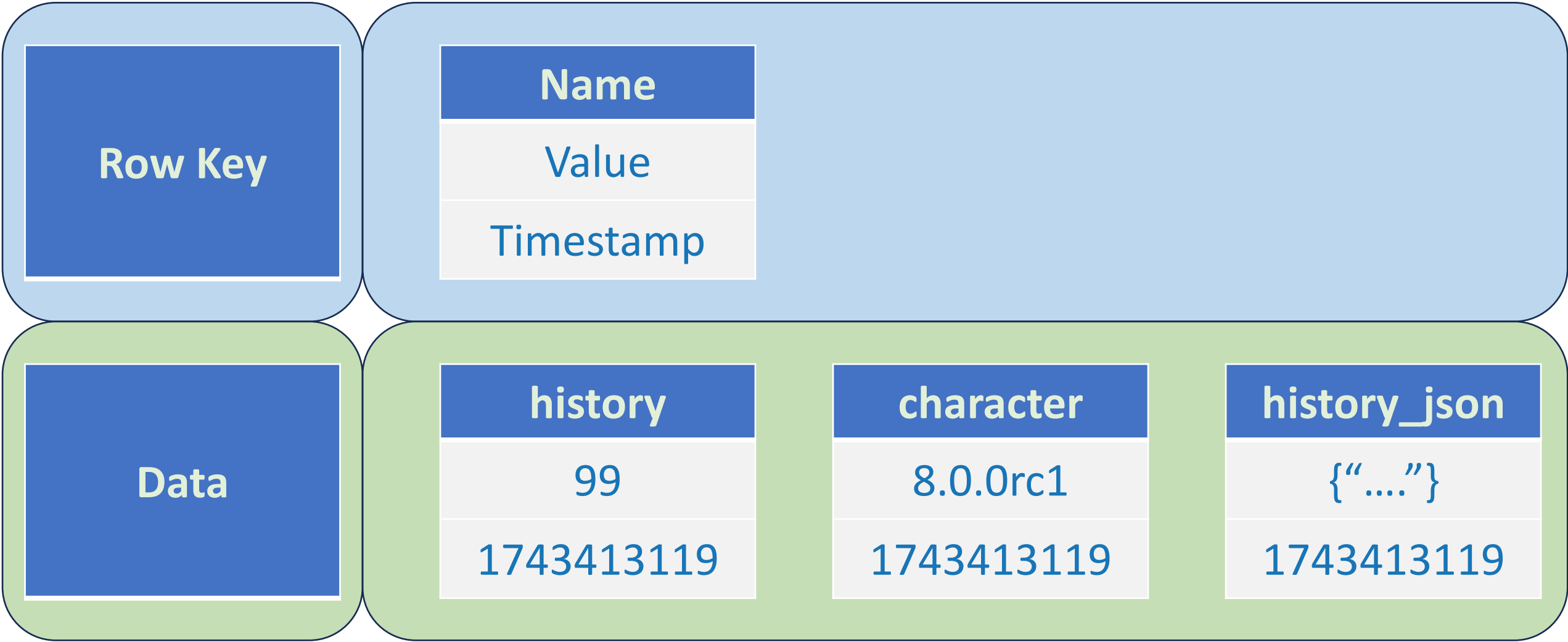
ClickHouse

ClickHouse is column oriented



ClickHouse

What is column oriented?



ClickHouse

What is **sharding**?

– Benefits of sharding

Horizontal scaling: Add nodes to increase storage and query processing capacity.

Parallel processing: Queries can run across multiple shards at the same time.

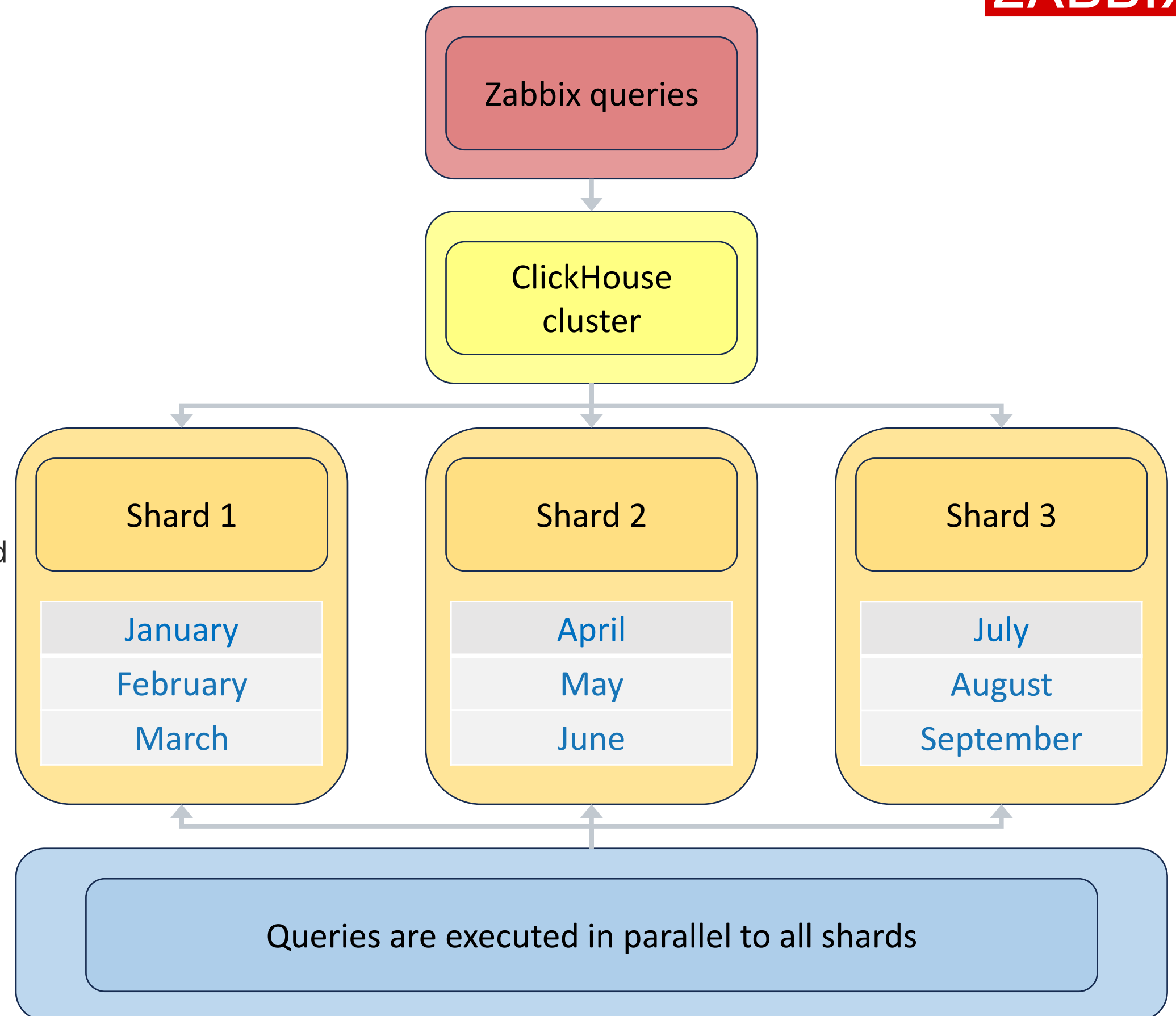
Large data volumes: Distributes data across nodes instead of relying on a single server.

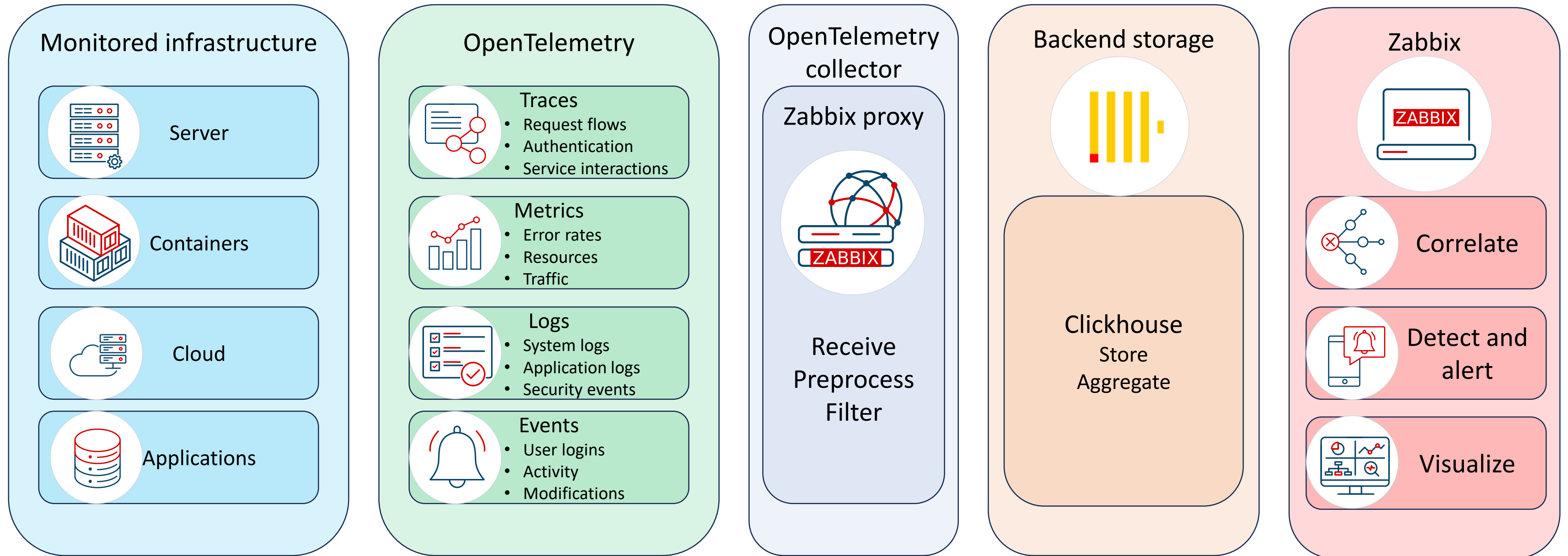
– Benefits for Zabbix

High-volume metrics: Handles large amounts of historical monitoring data as the environment grows.

Faster analytics: Multiple nodes can process Zabbix history and trend queries in parallel.

Scalable monitoring: Add more nodes as the number of monitored hosts and metrics increases.





Extend even further with OTEL

Setting up for success



There are three key elements (information boxes) in the next slides:

- Linux virtual machine CLI (command-line interface)

```
# systemctl enable zabbix-server --now
```

- Text on the screen after a command execution (output may differ)

```
Created symlink /etc/systemd/system/multi-user.target.wants/zabbix-server.service
```

- Linux editor screen – here, change the specified parameters to get required functionality

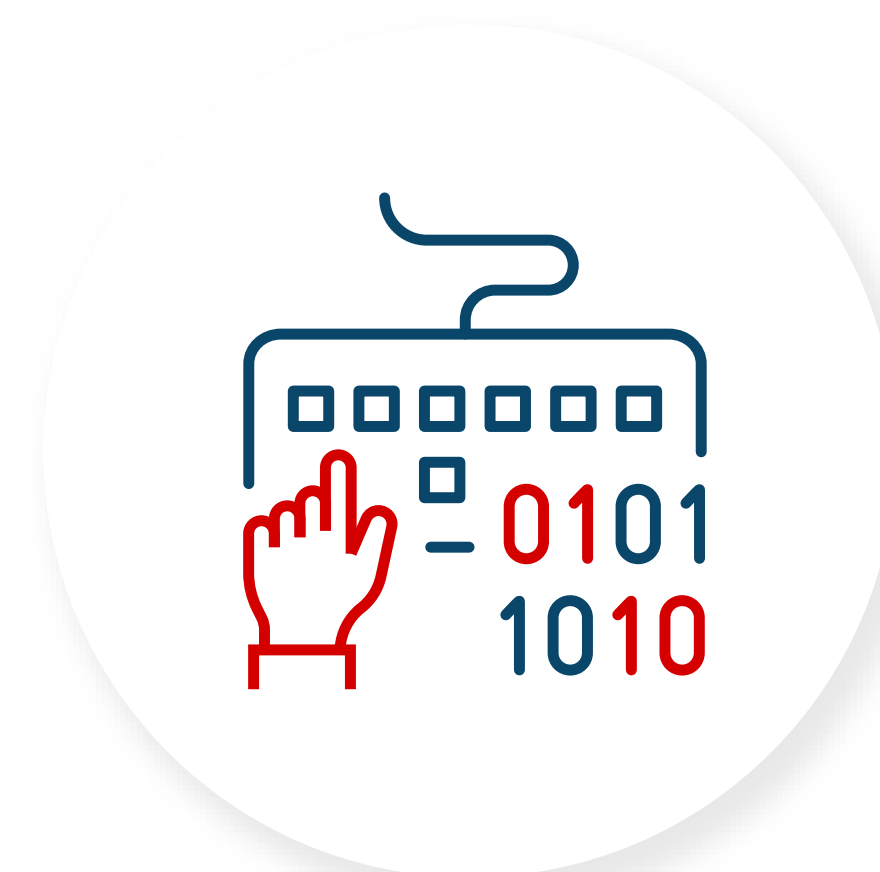
```
### Option: DBPassword
#       Database password.
#       Comment this line if no password is used.
DBPassword=P455w0RD
```

Step 1 - Login

Prepare for practical part:

- Use your favorite SSH client
- Specify the DNS provided on paper
- Use SSH username
- And SSH password
- Follow the slides or the presenter

Zabbix is already installed



Step 2 – Install ClickHouse

Install yum-utils, used by ClickHouse for RPM distributions (each distribution have its own steps)

```
# dnf install -y yum-utils
```

Add the official ClickHouse repository

```
# yum-config-manager --add-repo https://packages.clickhouse.com/rpm/clickhouse.repo
```

Update cache

```
# dnf clean all
```

Install ClickHouse server and client

```
# dnf install -y clickhouse-server clickhouse-client
```

Check the version

```
# clickhouse-client --version
```

```
ClickHouse client version 26.9.10.4 (official build).
```

Step 3 – Start ClickHouse server

Enable and start ClickHouse

```
# systemctl enable clickhouse-server --now
```

Check ClickHouse status

```
# systemctl status clickhouse-server
```

```
clickhouse-server.service - ClickHouse Server (analytic DBMS for big data)
   Loaded: loaded (/usr/lib/systemd/system/clickhouse-server.service; enabled; preset: disabled)
   Active: active (running) since Tue 2026-10-06 07:12:56 UTC; 46s ago
 Invocation: f707581198754fdfb6e49b5f4b083968
   Main PID: 15529 (clickhouse-serv)
    Tasks: 197 (limit: 10498)
   Memory: 481.2M (peak: 484.1M, swap: 10.1M, swap peak: 10.1M)
      CPU: 4.834s
   CGroup: /system.slice/clickhouse-server.service
           └─15500 clickhouse-watchdog "" "" "" "" "" "" "" --config=/etc/clickhouse-
server/config.xml --pid-file=/run/clickhouse-server/clickhouse-server.pid
              └─15529 /usr/bin/clickhouse-server --config=/etc/clickhouse-server/config.xml --pid-
file=/run/clickhouse-server/clickhouse-server.pid
```

Step 4 – Test ClickHouse client and server

Use ClickHouse client

```
# clickhouse-client
```

```
Connecting to localhost:9000 as user default.  
Connected to ClickHouse server version 26.9.10.
```

```
Warnings:
```

```
* Available memory at startup is too low (2GiB).....
```

Make a test query

```
student-xx :) select version();
```

```
Query id: b5baa64d-5371-4f13-a195-9d42e8604b54
```

```
1.  ┌version()┐  
    └26.9.10.4┘
```

```
1 row in set. Elapsed: 0.004 sec.
```

Exit the client

```
student-xx :) quit
```

Step 4 – Test and configure ClickHouse frontend

Currently, ClickHouse is listening on localhost only, test by executing

```
# curl 'http://localhost:8123/?query=SELECT%201'
```

```
1
```

Configure it to listen on all interfaces for workshop and save the file

```
# nano /etc/clickhouse-server/config.d/network.xml
```

```
<clickhouse>
  <listen_host>0.0.0.0</listen_host>
</clickhouse>
```

Restart ClickHouse to apply the changes

```
# systemctl restart clickhouse-server
```

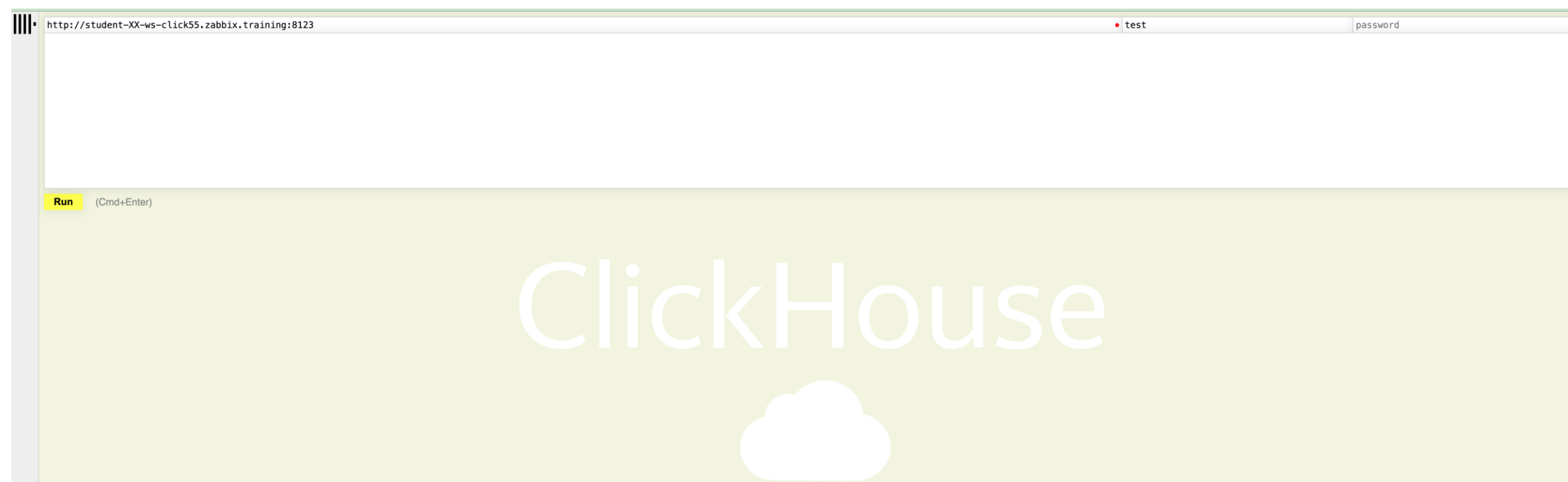
Open the preferred Browser app

Step 5 – Confirm web interface status

In the browser app use your VM URL, replace XX with your number

— <http://student-XX-ws-click.zabbix.training:8123/play>

Check ClickHouse status web interface



Queries can be executed here as well, but we will use backend

Return to the CLI (your SSH client)

Step 6 – Create Zabbix history dedicated database and user

Login to ClickHouse

```
# clickhouse-client
```

Create the database for Zabbix data

```
student-xx :) CREATE DATABASE IF NOT EXISTS zabbix;
```

Create dedicated database user

```
student-xx :) CREATE USER IF NOT EXISTS zabbix IDENTIFIED WITH sha256_password BY 'ZabbixWS_2026!';
```

Give user the necessary permissions to the created user

```
student-xx :) GRANT CREATE, ALTER, DROP, INSERT, SELECT, UPDATE, OPTIMIZE ON zabbix.* TO zabbix;
```

Exit the ClickHouse CLI

```
student-xx :) quit
```

Step 7 – Import the Zabbix schema for ClickHouse

Confirm the database scripts are in place

```
# ls -l /usr/share/zabbix/sql-scripts/clickhouse/
```

```
total 32
-rw-r--r--. 1 root root 1560 Oct  1 00:00 clickhouse.sh
-rwxr-xr-x. 1 root root  239 Oct  1 00:00 history_all.sh
-rwxr-xr-x. 1 root root  462 Oct  1 00:00 history_json_schema.sh
-rwxr-xr-x. 1 root root  513 Oct  1 00:00 history_log_schema.sh
-rwxr-xr-x. 1 root root  436 Oct  1 00:00 history_schema.sh
-rwxr-xr-x. 1 root root  443 Oct  1 00:00 history_str_schema.sh
-rwxr-xr-x. 1 root root  445 Oct  1 00:00 history_text_schema.sh
-rwxr-xr-x. 1 root root  445 Oct  1 00:00 history_uint_schema.sh
```

The clickhouse.sh is a helper script for connection

history_all.sh creates the actual tables in ClickHouse

Step 7-1 – Import the Zabbix schema for ClickHouse

Execute the script with required ClickHouse connection parameters

```
# /usr/share/zabbix/sql-scripts/clickhouse/history_all.sh \  
--user zabbix \  
--password 'ZabbixWS_2026!' \  
--db zabbix \  
--server http://localhost:8123 \  
--ttl 604800
```

Verify the schema (enter the password when prompted)

```
# clickhouse-client --user zabbix --password --query "SHOW TABLES FROM zabbix"
```

```
Password for user (zabbix): ZabbixWS_2026!  
history  
history_json  
history_log  
history_str  
history_text  
history_uint
```

Step 7-2– Import the Zabbix schema for ClickHouse

Verify one of the tables in ClickHouse (enter the password when prompted)

```
# clickhouse-client --user zabbix --password --query "SHOW CREATE TABLE zabbix.history_uint"
```

```
Password for user (zabbix): ZabbixWS_2026!  
CREATE TABLE zabbix.history_uint\  
(\  
    `itemid` UInt64,\  
    `clock_ns` DateTime64(9),\  
    `value` UInt64\  
\  
ENGINE = MergeTree\  
PARTITION BY toDate(clock_ns)\  
PRIMARY KEY (itemid, clock_ns)\  
ORDER BY (itemid, clock_ns)\  
TTL clock_ns + toIntervalSecond(604800)\  
SETTINGS index_granularity = 8192
```

This particular definition creates a **time-partitioned, item/time-sorted ClickHouse table for Zabbix JSON history, with nanosecond timestamps, sparse indexing, and automatic retention of approximately one hour.**

Step 8 – Configure Zabbix server to work with ClickHouse

Edit the Zabbix server configuration file

```
# vi /etc/zabbix/zabbix_server.conf
```

```
# Default:
# HistoryCacheSize=16M
HistoryCacheSize=512M
# Default:
# HistoryIndexCacheSize=4M
HistoryIndexCacheSize=64M
### Option: HistoryProvider
HistoryProvider=clickhouse;value_types="uint,dbl,str,log,text,json",url=http://localhost:8123,db=zabbix,username=zabbix,password=ZabbixWS_2026!
```

Validate configuration

```
# zabbix_server -T
```

Restart Zabbix server to apply the changes

```
# systemctl restart zabbix-server
```

Step 9 – Configure Zabbix frontend to work with ClickHouse

Edit the Zabbix server configuration file

```
# nano /etc/zabbix/web/zabbix.conf.php
```

```
// ClickHouse and Elasticsearch:
$HISTORY_PROVIDERS[] = [
    'types' => ['uint', 'dbl', 'str', 'log', 'text', 'json'],
    'provider' => 'clickhouse',
    'url' => 'http://localhost:8123',
    'db' => 'zabbix',
    'username' => 'zabbix',
    'password' => 'ZabbixWS_2026!'
];
```

Open the Web interface and confirm no errors are visible (replace XX with your number):

– URL: <https://student-XX-ws-click.zabbix.training>

Clickhouse setup is finished!



Data loader test



Step 1 – Download Zabbix DataLoader

First install the prerequisites

```
# dnf install -y unzip php8.4-cli
```

Change directory to /opt and download the DataLoader

```
# cd /opt  
# wget -O data-loader.zip 'https://space.zabbix.com/public.php/dav/files/D65GcXG8m5fRZEE/?accept=zip'
```

Unzip the DataLoader and enter the created directory

```
# mkdir -p data-loader && unzip data-loader.zip -d data-loader  
# cd /opt/data-loader
```

Confirm it works by executing

```
./generate
```

```
Data loader v0.9  
Usage:  
./generate -options  
.....
```

Step 2 – Create API token

– Login into your Zabbix web interface (<https://student-XX-ws-click.zabbix.training>)

– Go to 'User settings' > 'API Tokens' and press

Create API token

- Name: ClickHouse Workshop

- Description: Token for ClickHouse

– Press Add

– **Copy and save the API token!**

- It will be never displayed again

- Only the token hash is stored in database

– Press Close

New API token

* Name

Description

Set expiration date and time ☐

Enabled ☒

Add Cancel

API token

✓ API token added

Name: ClickHouse Workshop

Auth token: 37ee0b3e508d253cc71509251b234b4d0acb4deb44c73789a249b749a4bdc55 ⓘ 📄

Expires at: -

Description: Token for ClickHouse

Enabled: ☒

Close

Step 3 – Export API access

Export the Frontend URL and API token (replace XX with your number)

```
# export API_URL="http://student-XX-ws-click.zabbix.training/"  
# export API_TOKEN="37ee0b3e508d253cc71509251b234b4d0acb4debf44c73789a249b749a4bdc55"
```

Confirm export

```
# echo "$API_URL"  
# echo "$API_TOKEN"
```

<http://student-XX-ws-click.zabbix.training>

37ee0b3e508d253cc71509251b234b4d0acb4debf44c73789a249b749a4bdc55

Now we will create

- 1 host group
- 500 hosts
- 100 items per host

Step 4 – Load the Data

To do that in DataLoader directory (/opt/data-loader) execute

```
# time ./generate --api-url="$API_URL" --api-token="$API_TOKEN" -H1 -h2 -i100 --data=10000 \  
--from="-1 hour" --to="now" --min=0 --max=100 --server="127.0.0.1" --port=10051 --bulk=10000
```

Logging in by token

Running HostGroupGenerator

Running HostGenerator

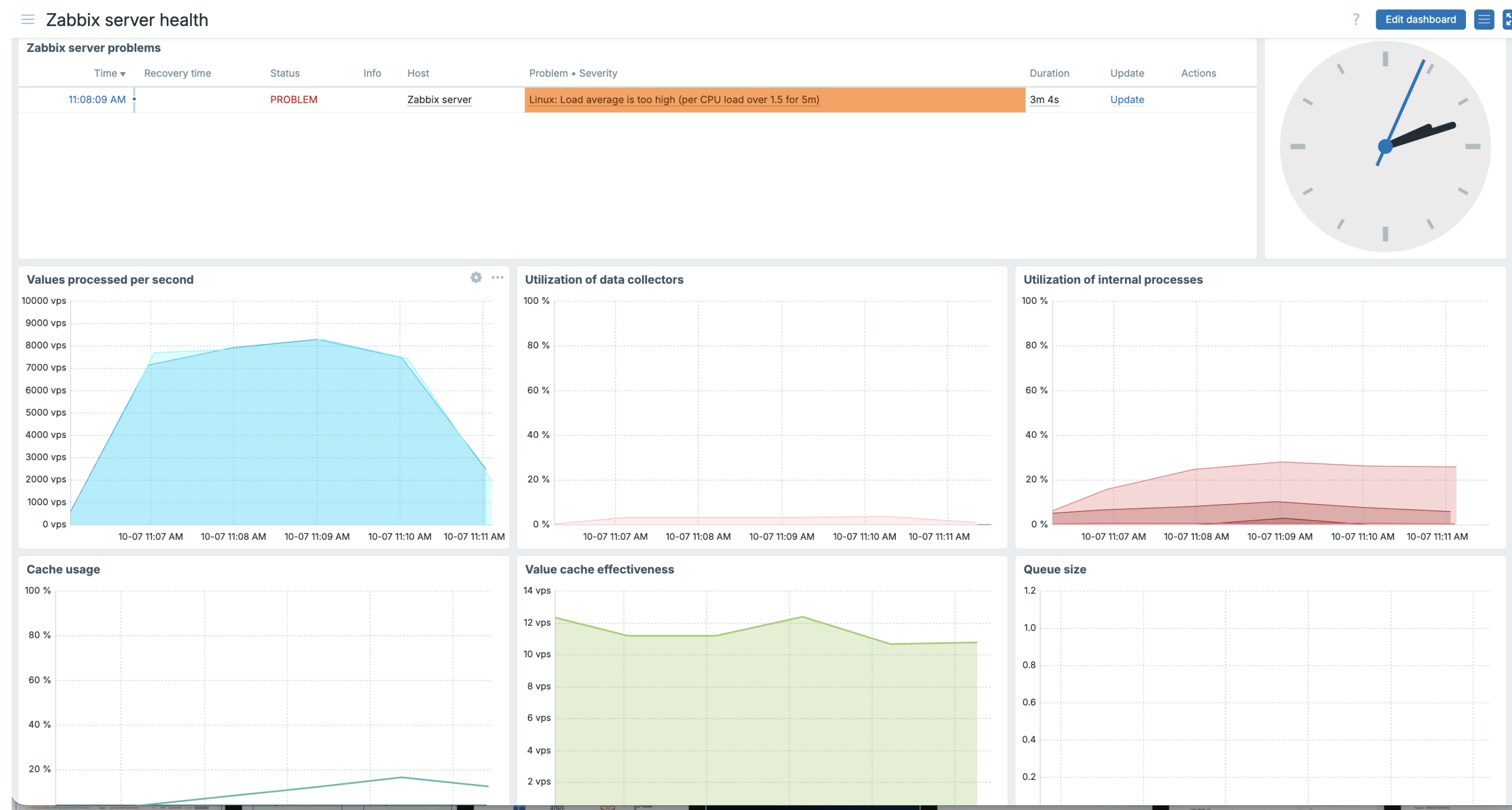
Running HostItemGenerator

Running HostItemDataGenerator

Before data generation please reload server cache with "zabbix_server -R config_cache_reload" and press any key.

Wait for 15 seconds and press any key (i.e. enter)

– Open the [Zabbix server health](#) Dashboard, check the Values Processed per seconds



Thank you!

