Kim Anthonisen

Team Lead, Senior Consultant and
Zabbix Certified Expert

Miracle42
THE ANSWER TO IT ALL
Part of itm8

ZABBIX
PREMIUM PARTNER

NORDIC Excellence

Powerhouse of databases

Locations

- Skovlunde
- Aarhus
- Stockholm
- Prague

Miracle42
THE ANSWER TO IT ALL

Implementing TimescaleDB without downtime

How we implemented TimescaleDB partitioning and compression on a large Zabbix installation without downtime

Miracle42
THE ANSWER TO IT ALL

Before we start, please vote for this ZBXNEXT:

Our Zabbix

Monitoring our customers databases, servers and applications

We are the designated Database Administrators (DBA's)

Zabbix is our tool

Internally

- Zabbix as our own internal monitoring tool

Zabbix as a Service (MSP)

We offer Zabbix to our customers which they use as their own

Zabbix as a Service

Principles:

We facilitate the Zabbix installation – we administrate users, hostgroups etc, we install and maintain the proxies, but the customers installs the agents, and sets up their own monitoring.

Typically, we help setting up the monitoring together with the customer, and after this, the customer maintains the monitoring them selves, with support from us.

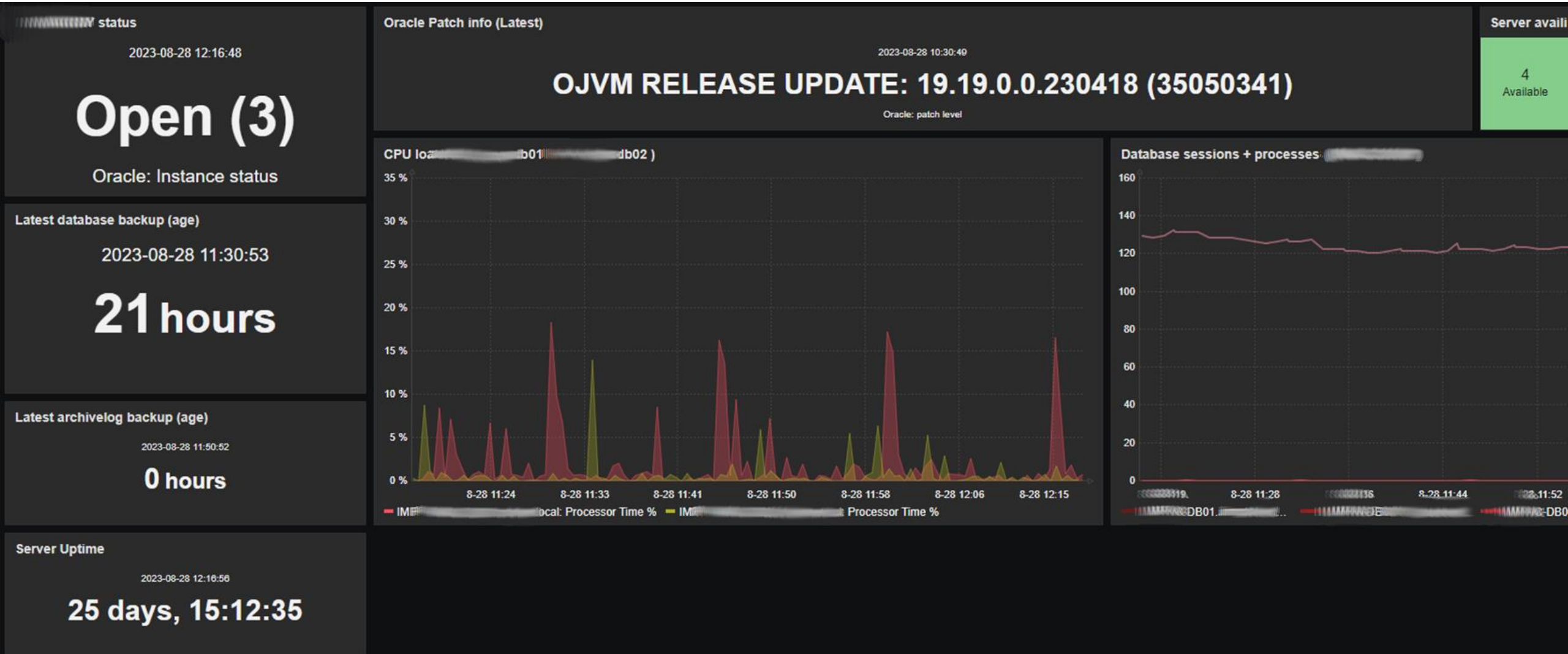
Zabbix Support subscriptions (MSP) are part of this, Miracle42 as 1st level support and Zabbix Support as 2nd level.

Our Zabbix: System information

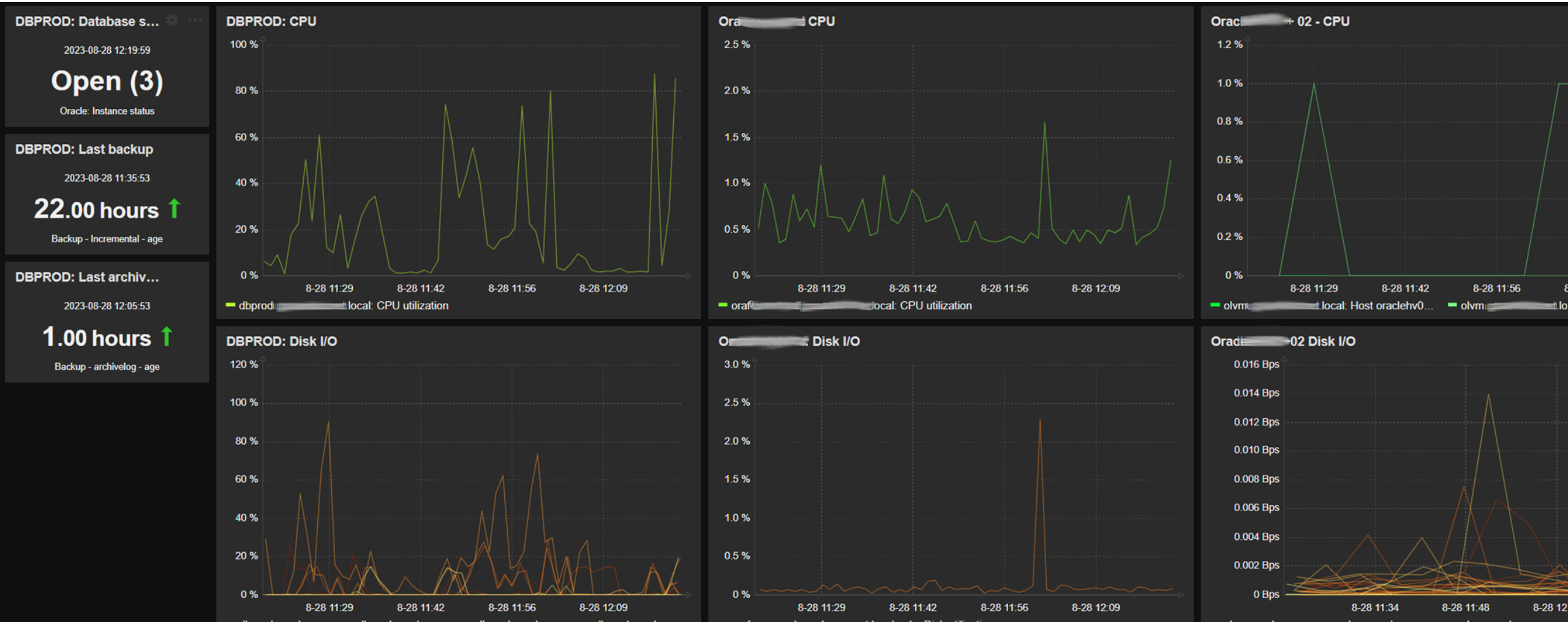
System information

Parameter	Value	Details
Zabbix server is running	Yes	[redacted]0051
Number of hosts (enabled/disabled)	739	644 / 95
Number of templates	159	
Number of items (enabled/disabled/not supported)	89366	71464 / 14967 / 2935
Number of triggers (enabled/disabled [problem/ok])	30606	22096 / 8510 [203 / 21893]
Number of users (online)	95	8
Required server performance, new values per second	641.4	
High availability cluster	Enabled	Fail-over delay: 1 minute

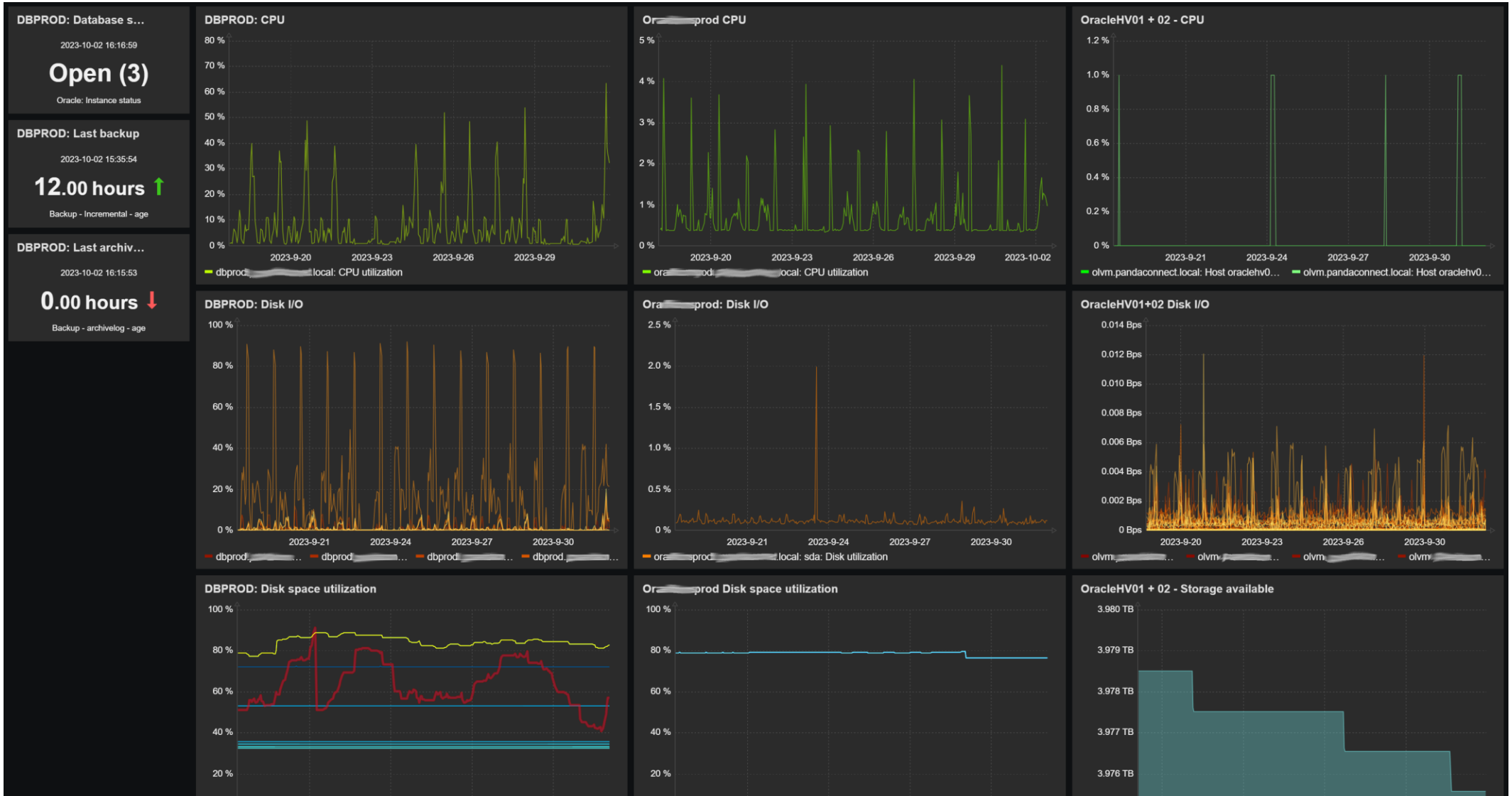
Our Zabbix: Dashboards



Our Zabbix: Dashboards



Our Zabbix: Dashboards



Our Ambition

Our ambition is to offer our customers a state-of-the-art Zabbix with as close to 100 percent uptime as possible.

Downtime ?

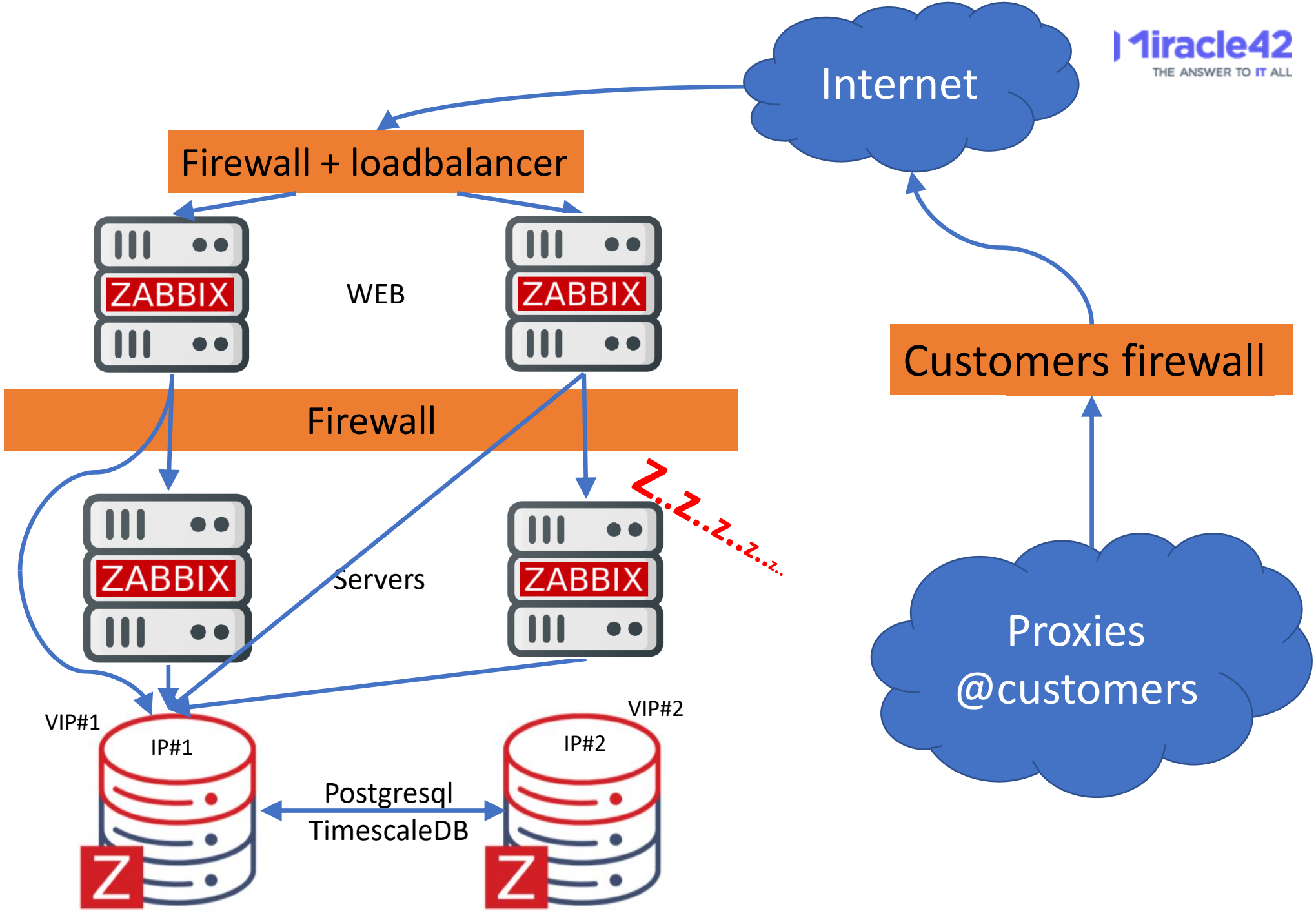
Downtime is difficult for us, as we must agree with many customers about a service window.

Some customers can afford a service window in the daytime when they use the systems, while others prefer downtime in out of office hours.

Downtime ?

Conclusion: downtime is not an option!

M42 Zabbix


Internet
Firewall + loadbalancer
ZABBIX
WEB
ZABBIX
Customers firewall
Firewall
ZABBIX
Servers
ZABBIX
Z..Z..Z..Z..z..
Proxies
@customers
VIP#1
IP#1
VIP#2
IP#2
Postgresql
TimescaleDB
Z
Z

Why TimescaleDB ?

Our Zabbix database has 4 Tb of data and is growing.

House keeper constantly running, cannot keep up.

Partitioning will remove the housekeepers work of deleting data, as old partitions can be dropped instead.

Compression will reduce storage demand from around 4 Tb to a few hundred Gb.

The problem

”The migration of existing history and trend data may take a lot of time. Zabbix server and frontend must be down for the period of migration.”

https://www.zabbix.com/documentation/6.0/en/manual/appendix/install/timescaledb

Zabbix - TimescaleDB

https://www.zabbix.com/documentation/6.0/en/manual/appendix/install/timescaledb

Run /usr/share/zabbix-sql-scripts/postgresql/timescaledb.sql, but without the ”Perform create_hypertable” and ”UPDATE config” lines, as we will do this manually.

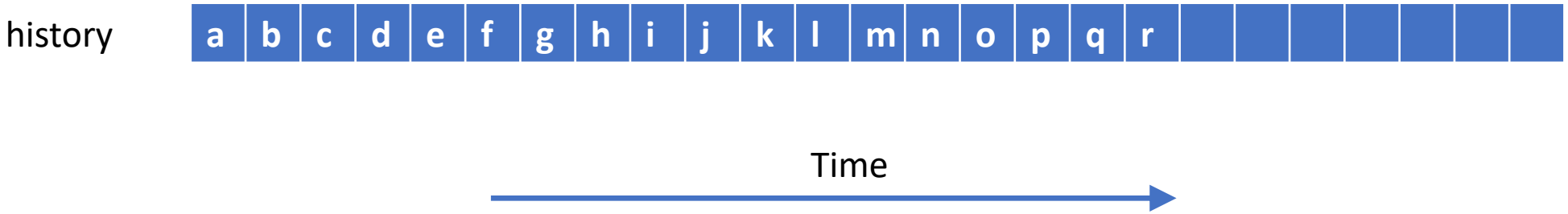
```
PostgreSQL version 14.4 is valid
TimescaleDB extension is detected
TimescaleDB version 2.8.0 is valid
TimescaleDB is configured successfully
```

Migrating to TimescaleDB

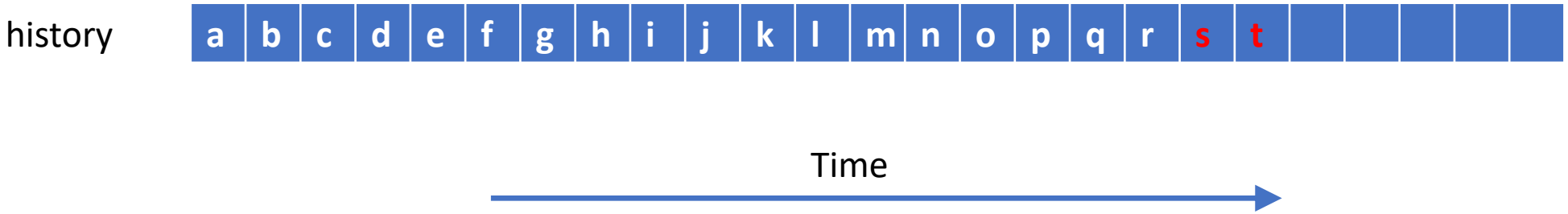
Concept

1. Create new table
2. Register new table with TimescaleDB
3. Create trigger on current table, to populate new table
4. Load data from current table to new table
5. Switch tables

Migrating to TimescaleDB



Migrating to TimescaleDB



Migrating to TimescaleDB

```
SQL> CREATE TABLE history_new (
LIKE history
INCLUDING DEFAULTS
INCLUDING CONSTRAINTS
INCLUDING INDEXES
);
```

history	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t					
history_new																									

Migrating to TimescaleDB

Concept

1. Create new table
2. Register new table with TimescaleDB
3. Create trigger on current table, to populate new table
4. Load data from current table to new table
5. Switch tables

Migrating to TimescaleDB

```
SQL> select create_hypertable
('history_new', 'clock', chunk_time_interval => 86400, migrate_data => true);
```

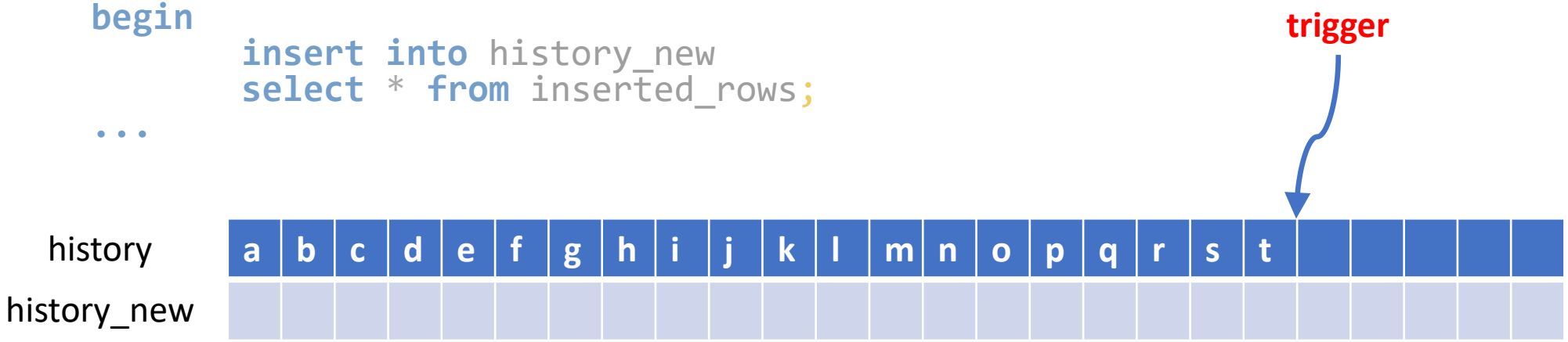
Migrating to TimescaleDB

Concept

1. Create new table
2. Register new table with TimescaleDB
3. Create trigger on current table, to populate new table
4. Load data from current table to new table
5. Switch tables

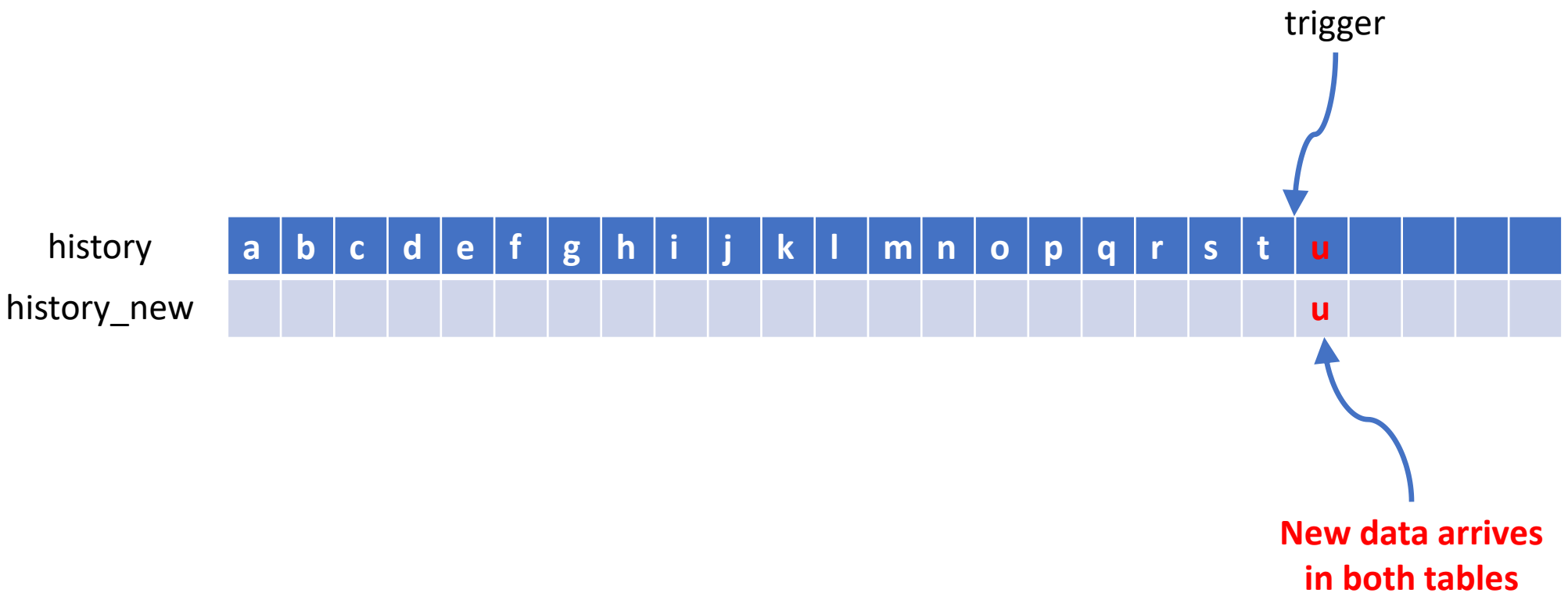
Migrating to TimescaleDB

```
SQL> create or replace function insert_history_m42()
...
begin
        insert into history_new
        select * from inserted_rows;
...
```




```
SQL> create trigger history_ins_trg_m42
after insert on history
...
execute function insert_history_m42();
```

Migrating to TimescaleDB



Migrating to TimescaleDB

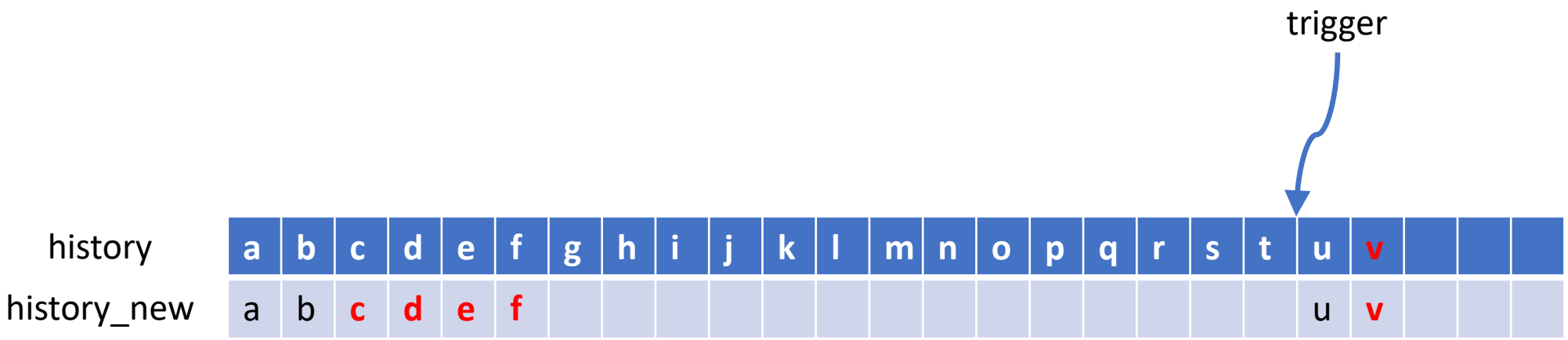
Concept

1. Create new table
2. Register new table with TimescaleDB
3. Create trigger on current table, to populate new table
4. Load data from current table to new table
5. Switch tables

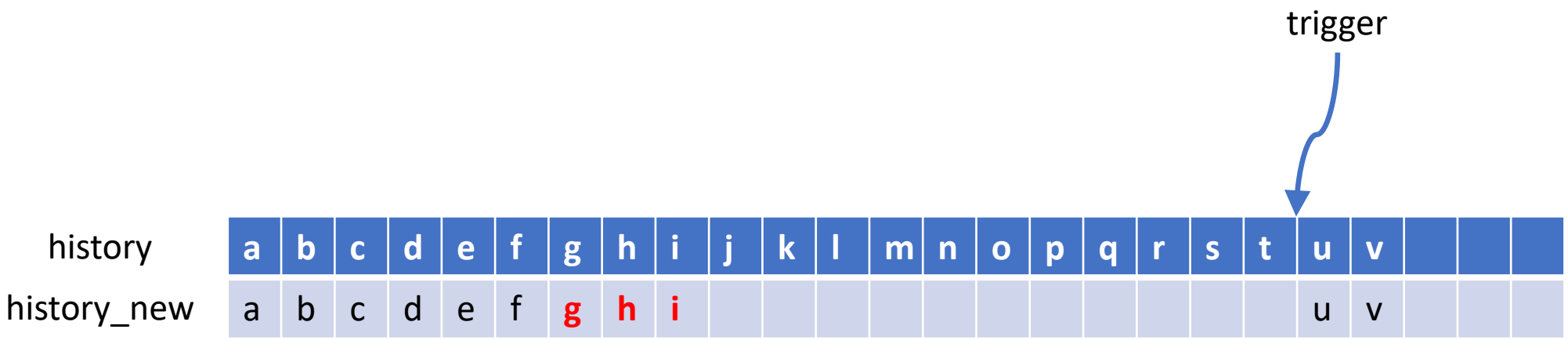
Migrating to TimescaleDB

```
SQL>
        for r in select itemid...
        loop
                insert into history_new
                select * from history as t where...
        end loop;
```

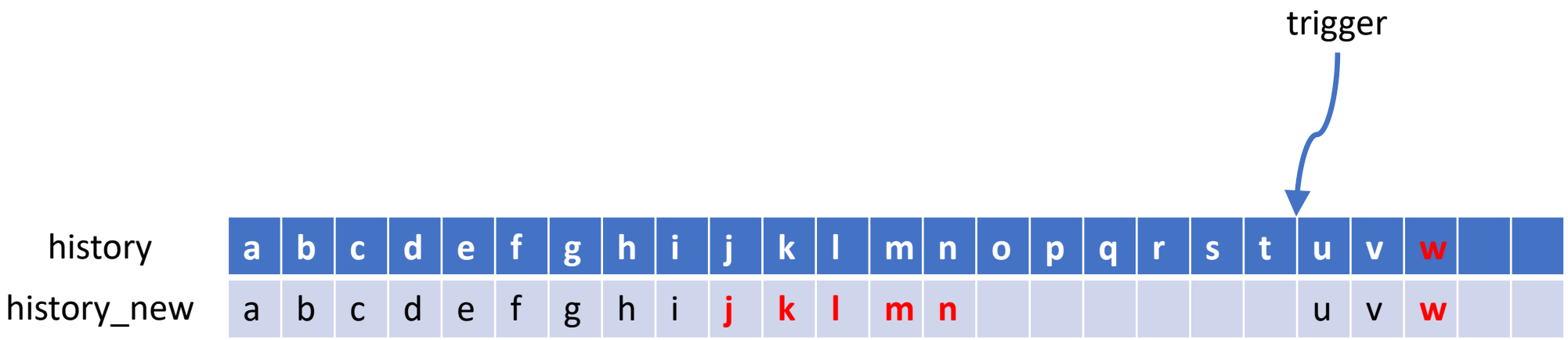
Migrating to TimescaleDB



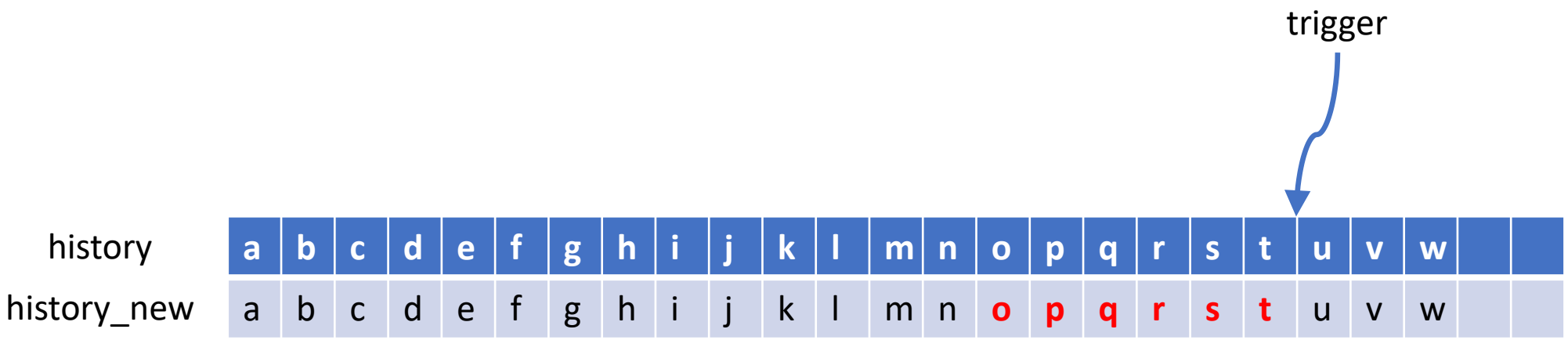
Migrating to TimescaleDB



Migrating to TimescaleDB



Migrating to TimescaleDB



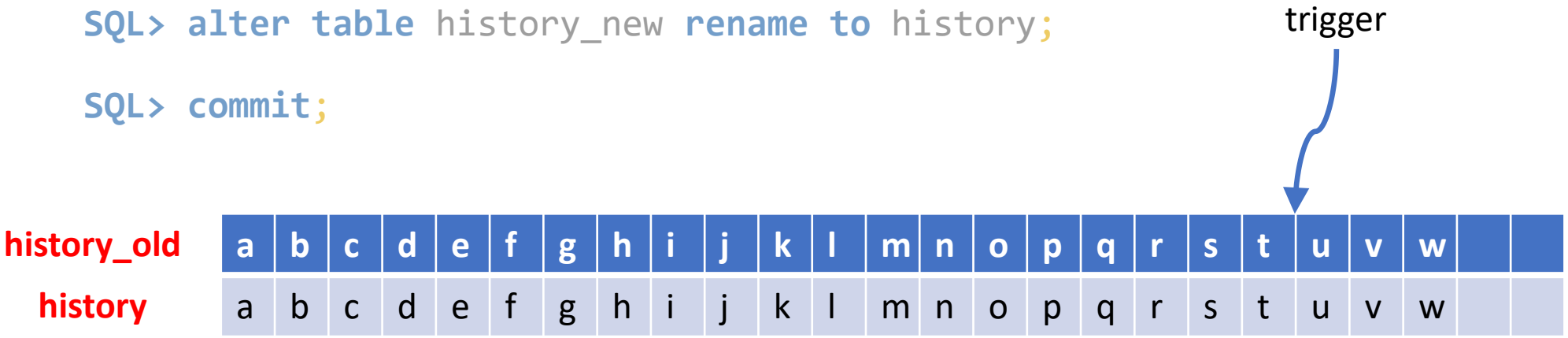
Migrating to TimescaleDB

Concept

1. Create new table
2. Register new table with TimescaleDB
3. Create trigger on current table, to populate new table
4. Load data from current table to new table
5. Switch tables

Migrating to TimescaleDB

```
SQL> alter table history rename to history_old;

SQL> alter table history_new rename to history;

SQL> commit;
```



Migrating to TimescaleDB

```
SQL> drop table history_old;
SQL> commit;
```

history	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

5. Switch tables

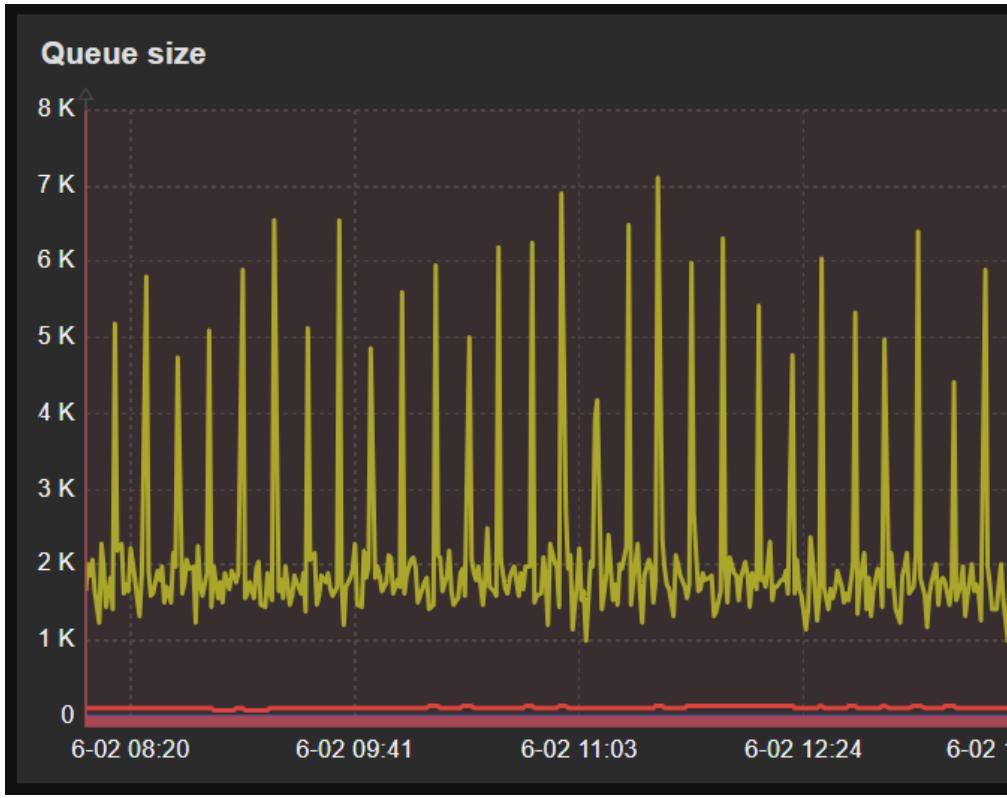
Warning: Be careful with database locks (!)

5 seconds	10 seconds	30 seconds	1 minute	5 minutes
0	0	0	0	0
0	0	2	6	0
7	48	3615	12	0
8	54	161	0	0

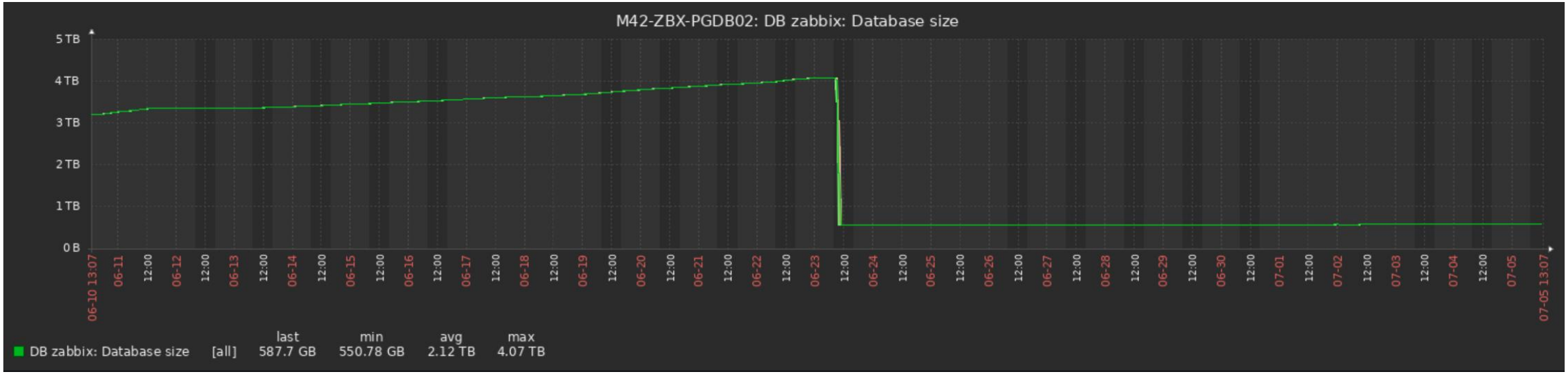
5. Switch tables

Warning: Be careful with database locks (!)

If House keeper is running, or if Zabbix is loading a large amount of data, the rename commands can hang for some time, resulting in more locks in the database and Zabbix unable to get new data.



Result



Surprisingly, the size of the database dropped from around 4 TB to 550 GB soon after the switch to TimescaleDB had completed (!)

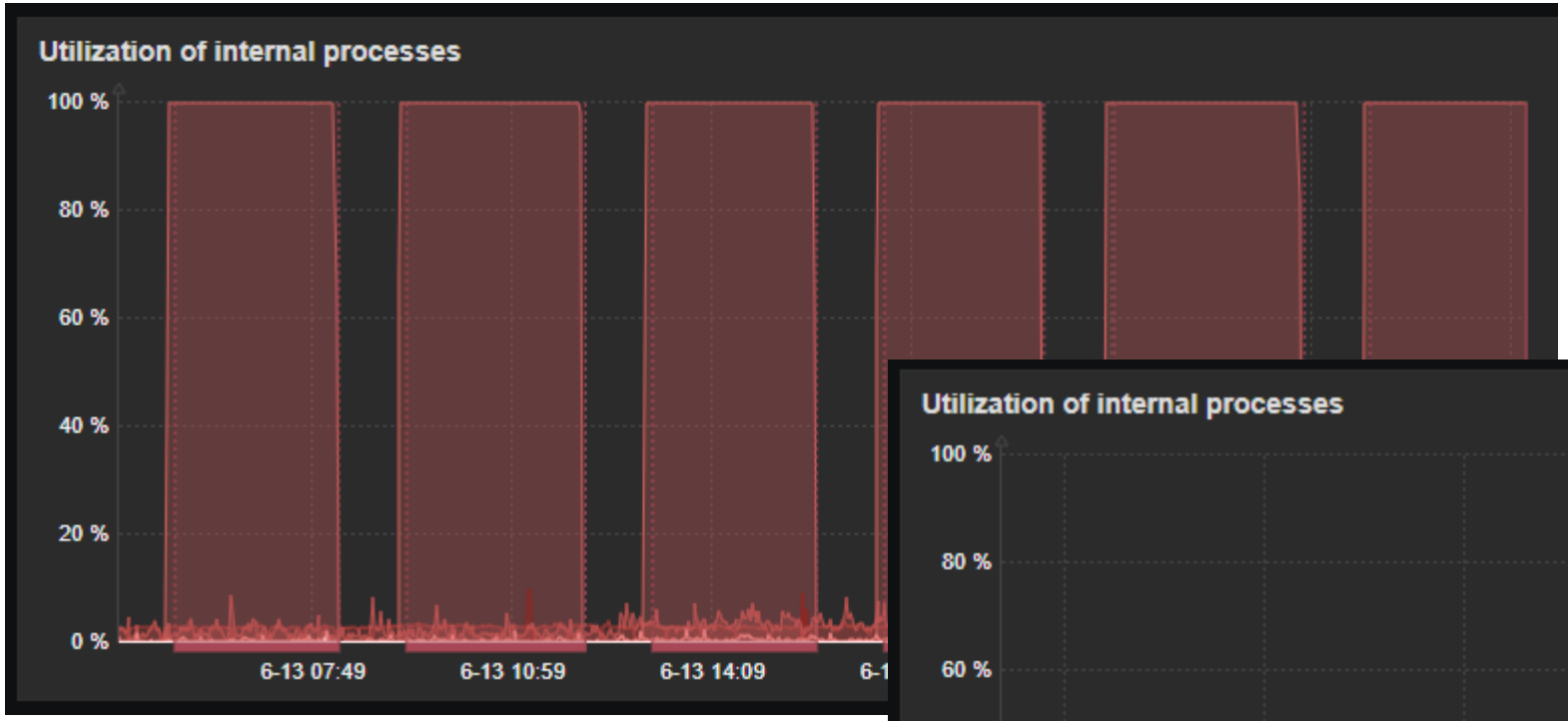
Result

The reason for this significant drop in size was not compression (which has not yet been enabled), but that old data was dropped, when their partitions was dropped.

The cause for this was, that the housekeeper had not been able to keep up with the data coming in, and for this reason, our database had grown a lot more than it was supposed to.

Result

Before



After

Lessons learned

1) Although the Housekeeper process does not run all the time, it can be trailing behind in the work it should do.

2) Make sure You have plenty of space for archive logs.
   - A lot of archive will be generated.

3) The load can run for quite some time.
   - Our history_uint was completed over the course of 2 weeks (!)

4) Be careful with locks when switching tables

Compression

Next, we enable the compression, which is performed by the housekeeper.

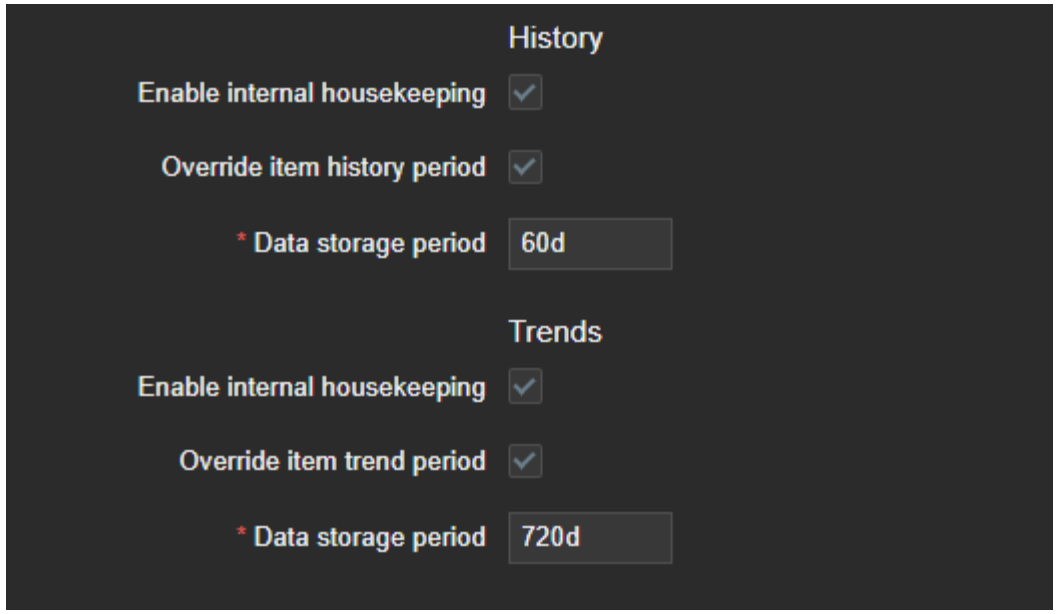
First step is to run the two update statements from the file /usr/share/zabbix-sql-scripts/postgresql/timescaledb.sql

```
SQL> UPDATE config SET
     db_extension='timescaledb’,
     hk_history_global=1,
     hk_trends_global=1;

SQL> UPDATE config SET
     compression_status=1,
     compress_older='7d’;

SQL> commit;
```

Compression

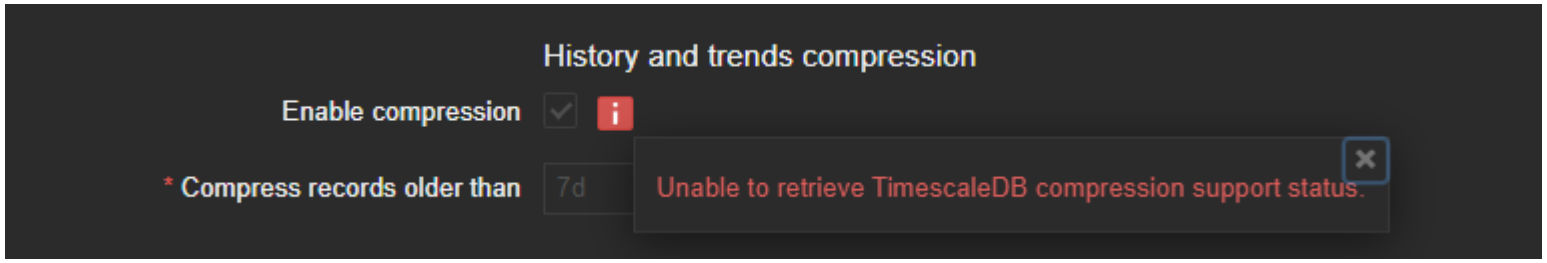
Once the data has been loaded and the tables switched, we should set global housekeeper options, overriding both item and trends storage period.

This will effectively disable the DELETION of data, and make Housekeeper DROP partitions instead.



Compression

For some reason, Zabbix might not detect and verify the TimescaleDB compression support, and for this reason, no compression is taking place.



Compression

Found that someone else has faced same issue previously

https://support.zabbix.com/browse/ZBX-21420

Solution seems to be restart Zabbix (my interpretation)

Compression

The issue seems to be with the logic by which the Zabbix server decides, if it can compress the partitions.

For some reason, Zabbix really did not want to compress our partitions.

Compression

I downloaded the Zabbix source code, found the place where compression of chunks are set, and backtraced the code to housekeeper's main loop.

Outline of the logic:

```
housekeeper main loop                          (housekeeper.c)
└ hk_history_compression_init/update           (history_compress.c)
  └ hk_history_enable_compression              (history_compress.c)
    └ hk_check_table_segmentation              (history_compress.c)
```

Compression

Running house keeper with debug level = 5, revealed where the issue should be found

```
# zabbix_server -R log_level_increase=452428
# zabbix_server -R log_level_increase=452428

# zabbix_server -R housekeeper_execute
```

Compression

/var/log/zabbix/zabbix_server.log:

452428:... executing housekeeper
...
452428:... In DBconnect() flag:0
...
452428:... End of DBconnect():0
...
452428:... In hk_history_compression_update()
452428:... End of hk_history_compression_update()

Note: the hk_history_enable_compression sub process was NOT executed!

Compression

With inspiration from the source code, I disabled compression

```
SQL> update config set compression_status=0;
SQL> commit;
```

I verified in the front end that compression was disabled and restarted the Zabbix server. And then switched on compression again:

```
SQL> update config set compression_status=1;
SQL> commit;
```

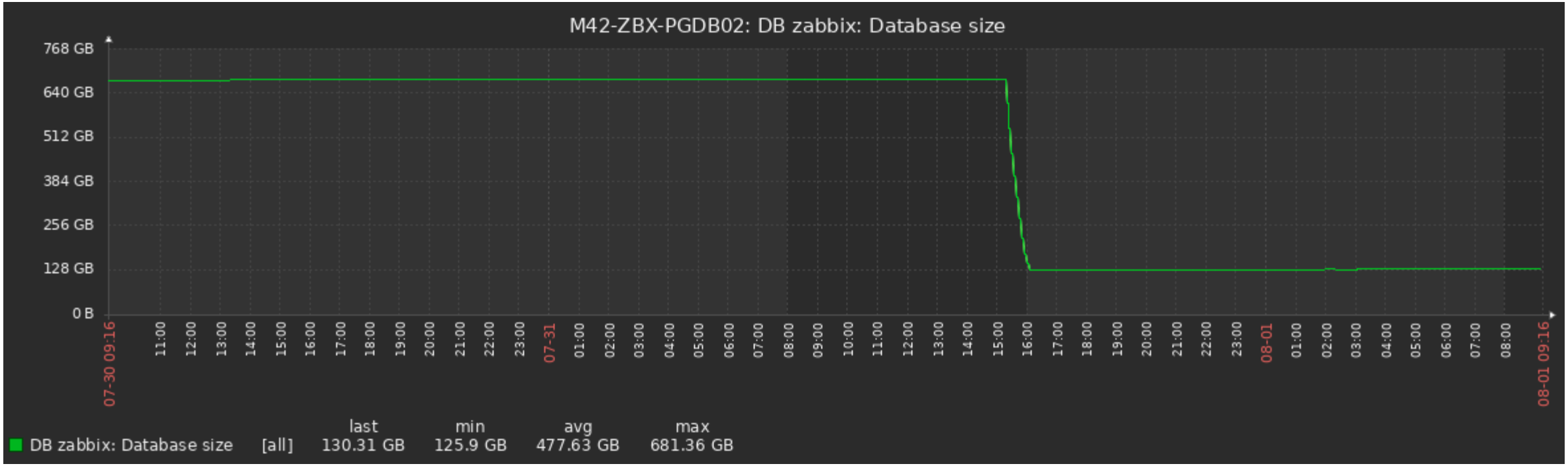
Compression

846073:… executing housekeeper
…
846073:… In hk_history_enable_compression()
…
846073:… In hk_check_table_segmentation(): table: history
…
846073:… query [txnlev:0] [alter table history set (timescaledb.compress,timescaledb.compress_segmentby='itemid',timescaledb.compress_orderby='clock,ns’)]
…
(repeats for all history and trends tables)

Compression

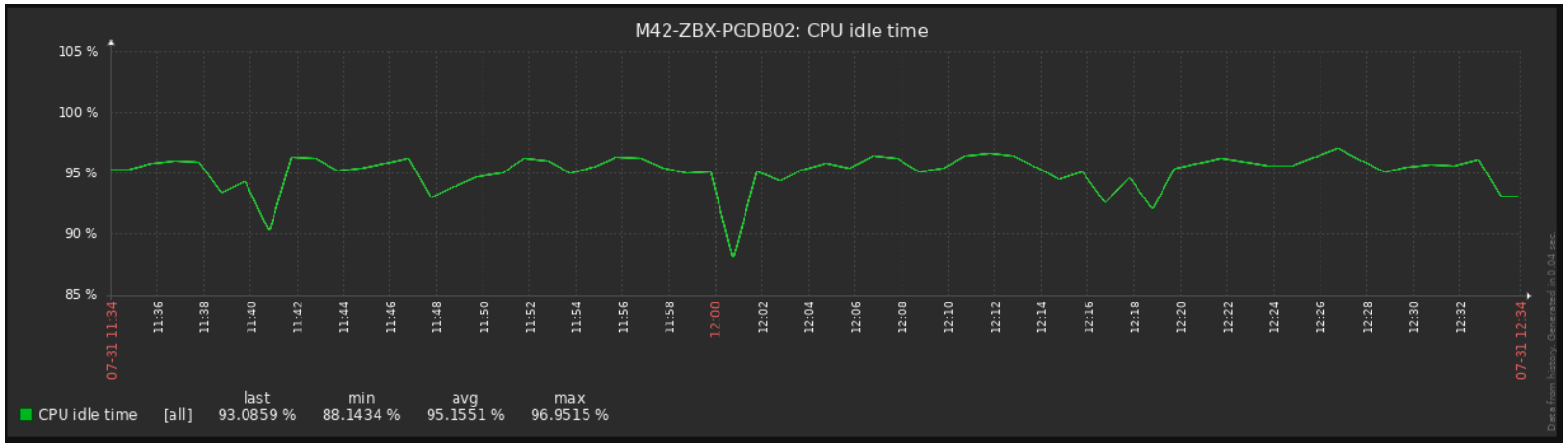
Succes !

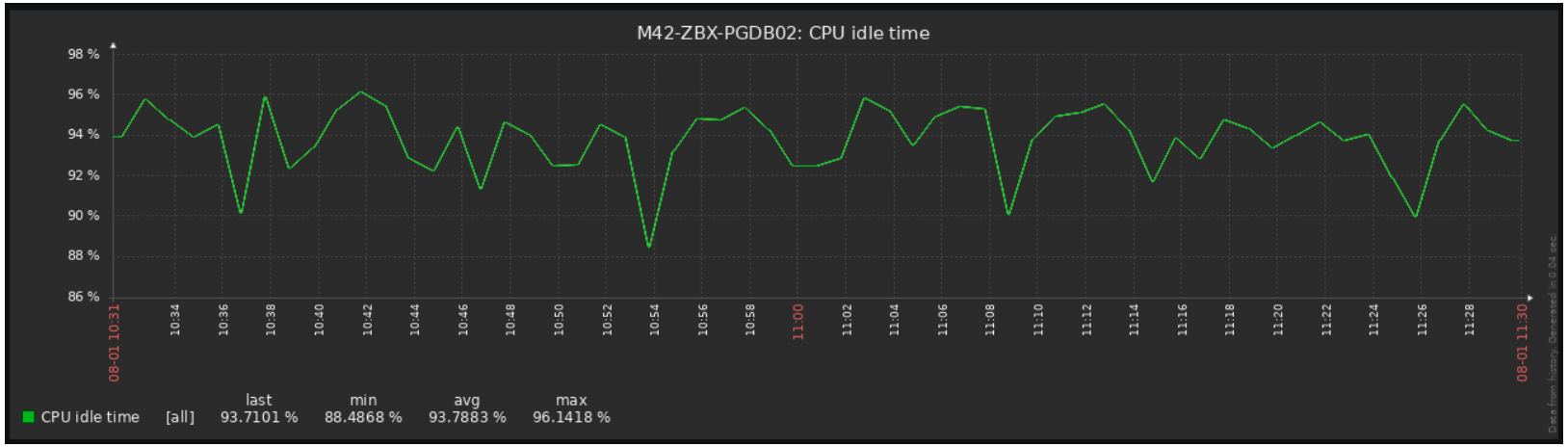
Database was compressed from 681 GB to 130 GB (81% reduction) !!!



Compression

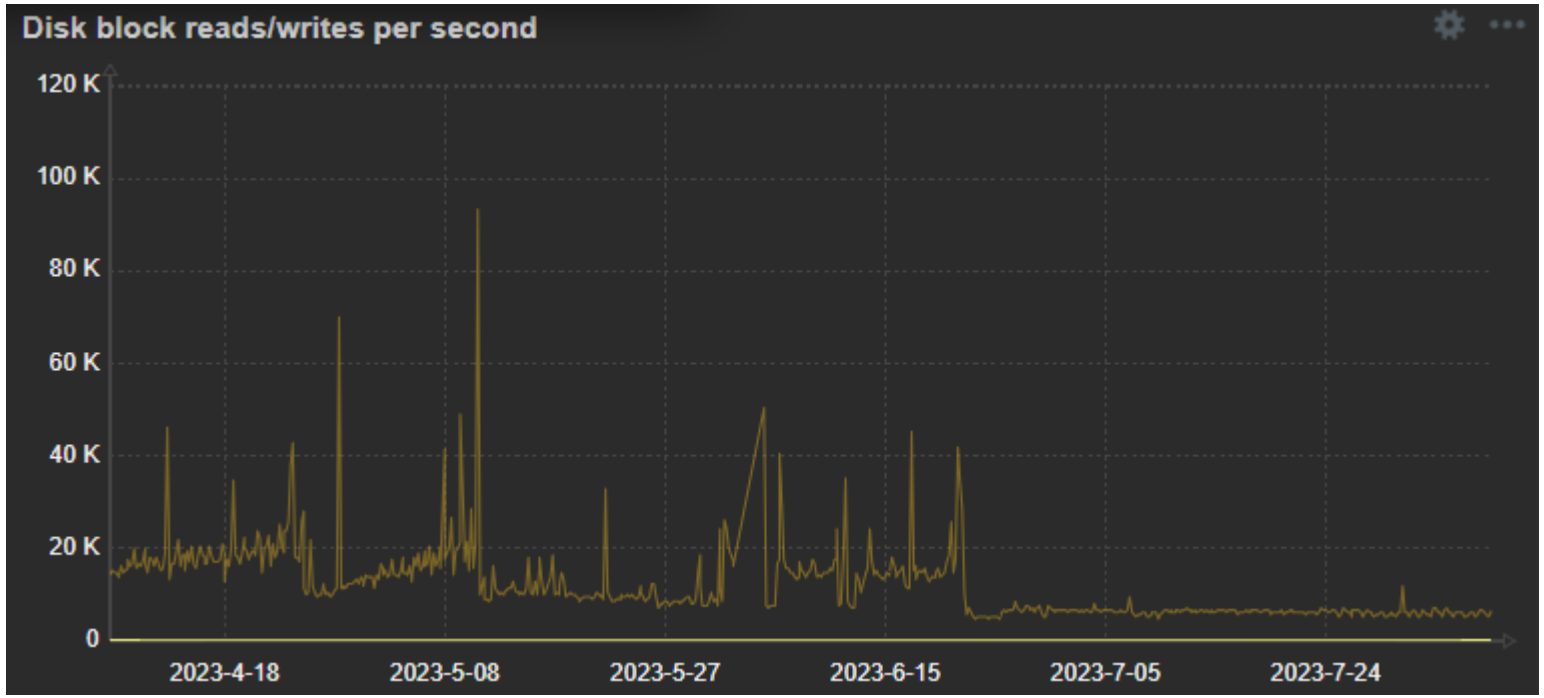
(only) a small increase in cpu usage (1-2 %)





Compression

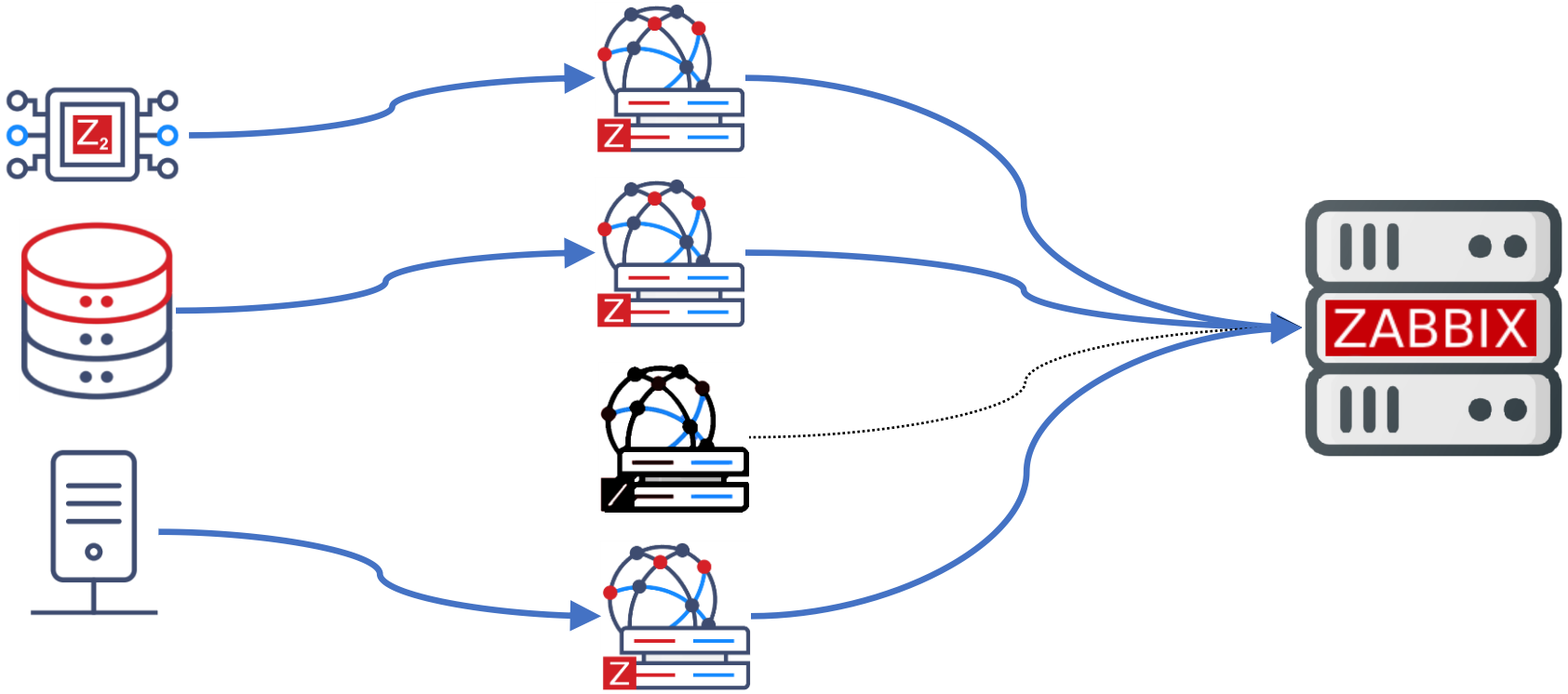
I/O has dropped significantly (by a factor of 3-4)



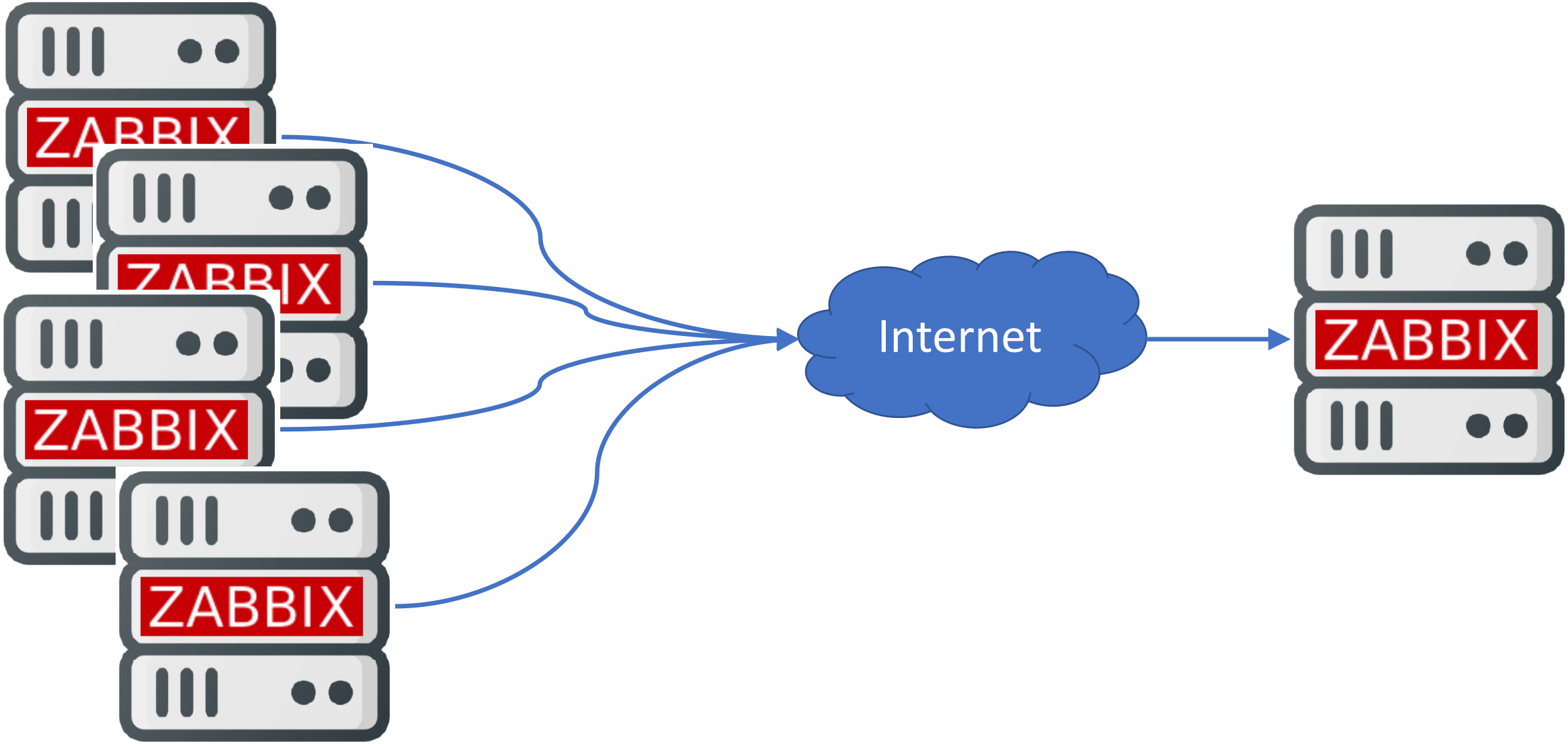
Conclusion

- Partitioning of history and trends removes a large load from the housekeeper
- Compression reduces database size a lot, enabling a much longer retention period and reducing I/O considerately, without increasing cpu usage noticeably
- Both partitioning and compression can be implemented without downtime for Zabbix

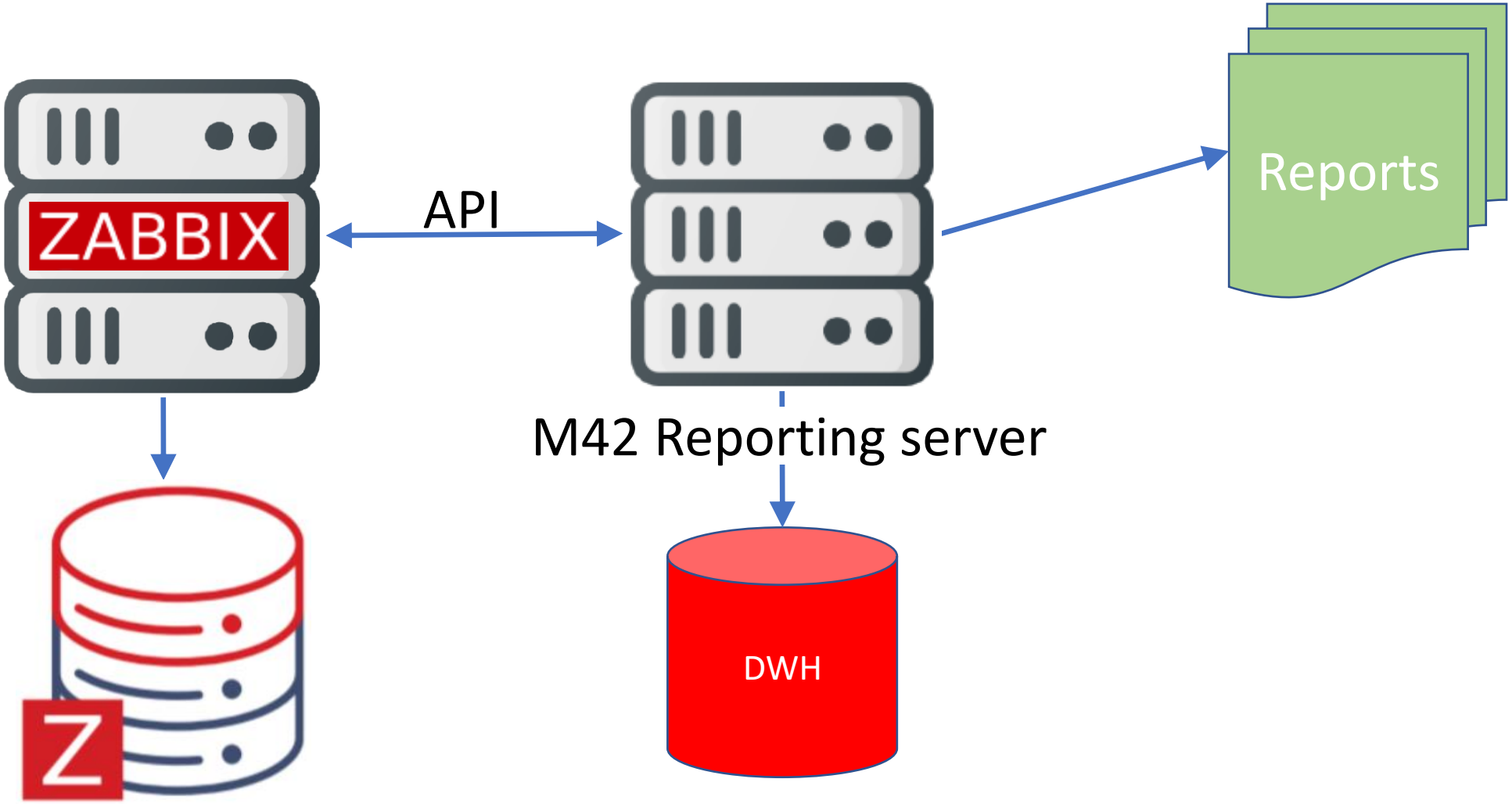
Proxy HA and load balancing



Zabbix-to-Zabbix project



M42 Zabbix: Reporting project



Miracle42
THE ANSWER TO IT ALL

Thank You for Your time!

Questions ?

Feel free to contact me at

zabbix@miracle42.dk