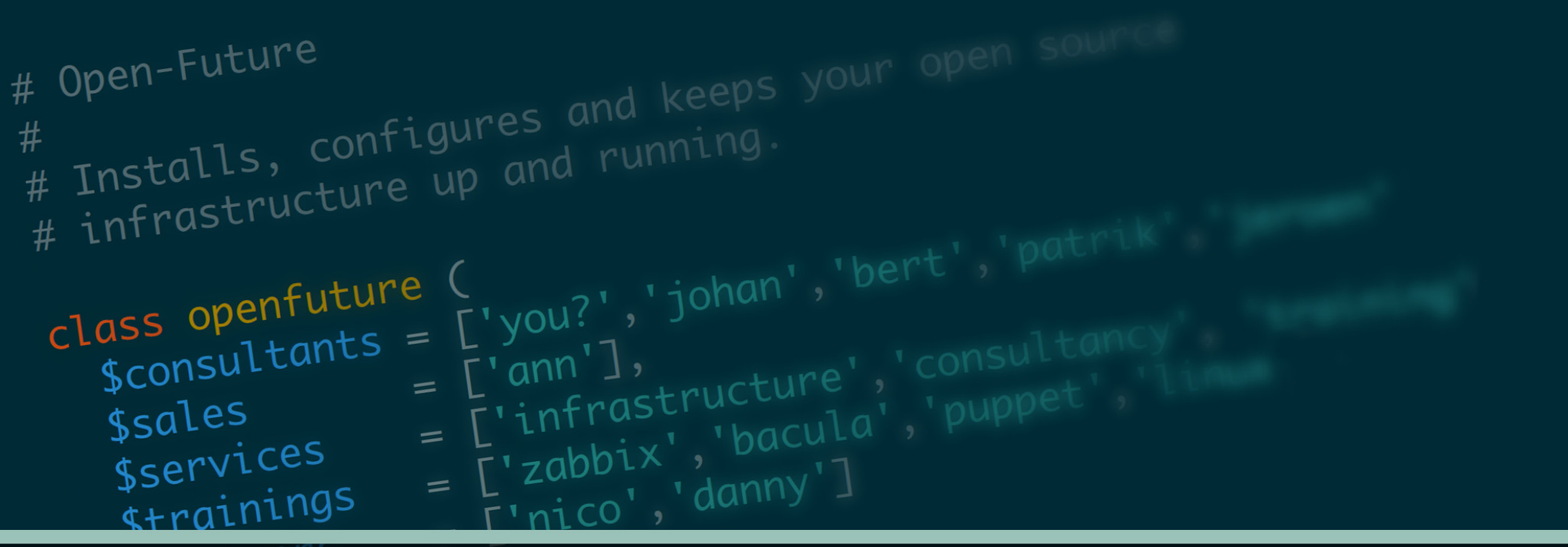


Upgrade Zabbix and migrate from PostgreSQL table partitioning to PostgreSQL with TimescaleDB

open-future

Author: Patrik Uytterhoeven

open-future

Making open source work for you

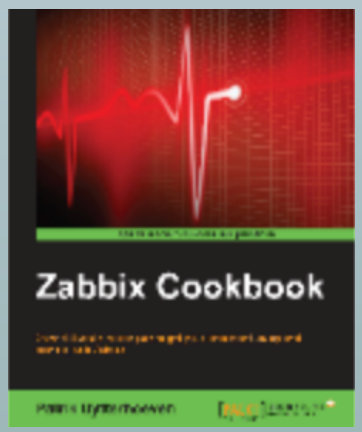
- ✓ Privately owned company based in Nossegem (Belgium)
- ✓ Open-Future was found in 2009 by Danny Herreman and Nico Vercammen
- ✓ Focus is exclusively on the open source market
- ✓ Focus is on Partnerships but not limited to …
- ✓ Official trainingspartner: Zabbix, Bacula, Puppet, …

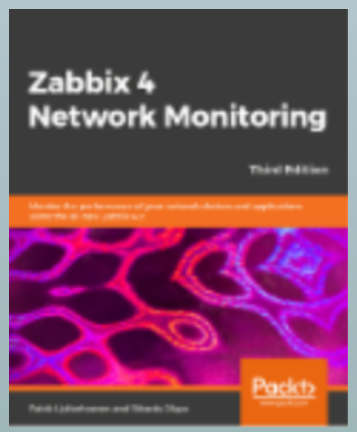
open-future

Making open source work for you

Who Am I ?

- ✓Patrik Uytterhoeven
- ✓Open-Source consultant
- ✓Certified Zabbix Trainer
- ✓Interests:
  - Monitoring
  - Ansible
  - SeLinux
  - PostgreSQL
- ✓Some books that I have written over the years





NOTES

- ✓ There are different solutions to solve this problem
- ✓ The solution that worked for us is maybe not the best solution for you

Making open source work for you

Our Needs

- ✓ We want to preserve history
- ✓ Trends is nice but we hardly ever need it so we don’t have to keep it
- ✓ We would like to upgrade Zabbix from 4.0 to 4.4
- ✓ We would also love to upgrade our Postgres 10 to 11 as version 11 is now our standard.
- ✓ Limit migration time as our infrastructure is highly dependent on Zabbix

How we upgraded Zabbix ?

- ✓ Upgrading Zabbix is easy, you replace the repos and upgrade the binaries and you are good to go as configuration is preserved on CentOS.
- ✓ This we tested before on a test machine with a copy of our DB to test it out and see if it would break things.

How did we move to TimescaleDB ?

- ✓ As there was no need to preserve trending we decided to drop the trends tables as this would speed up the migration process a lot.
- ✓ 2 weeks before the migration we stopped all DB scripts needed for our DB partitioning this way our historical data was already in the correct place.
- ✓ The day of the migration we only had to drop the history data in the partitioned tables together with our trending tabes.

Migration Tips

- There are other solutions to migrate the data. TimescaleDB has tools like timescaledb-parallell-copy. This is a command line program for parallelizing PostgreSQL's built-in COPY functionality for bulk inserting data into TimescaleDB.
- You could write your own script to move the data to the correct location
- ....

Making open source work for you

TimescaleDB Tips

- TimescaleDB has it own tools to tune the database: timescaledb-tune.
- Be careful as this tool will try to change your original postgresql.conf file
- Also it will not look in files you have included in your original potgresql.conf file (it will not see you have loaded its library if it is in the include file)
- The timescaledb-tuner will also thing that your DB is on a dedicated machine. So don’t just copy or apply its suggestions.

Making open source work for you

TimescaleDB Tips

- There is a chance that you will encounter errors in your postgresql logs like this:

psql: FATAL: out of shared memory

HINT: You might need to increase max_locks_per_transaction.

- TimescaleDB relies heavily on table partitioning for scaling time-series workloads, which has implications for lock management. A hypertable needs to acquire locks on many sub-tables during queries, which can exhaust the default limits for the number of allowed locks held.

Example

- /usr/bin/timescaledb-tune --pg-config=/usr/pgsql-11/bin/pg_config

Writing backup to:

/tmp/timescaledb_tune.backup202002270952

Memory settings recommendations

Current:

shared_buffers = 128MB

#effective_cache_size = 4GB

#maintenance_work_mem = 64MB

#work_mem = 4MB

Recommended:

shared_buffers = 16036MB

effective_cache_size = 48108MB

maintenance_work_mem = 2047MB

work_mem = 10263kB

Is this okay? [(y)es/(s)kip/(q)uit]: y

Making open source work for you

Upgrading to PG11

- ✓ Also here it can be done in many different ways like replicating the DB, creating a dump, …
- ✓ However we wanted to do it as fast as possible,
- ✓ Some downtime was acceptable as we where down already for the Zabbix migration
- ✓ It had to be done on the same hardware

Making open source work for you

Upgrading to PG11

- Our solution to this was to do an upgrade with pg_upgrade
- This has the advantage that you can do it without having to dump the DB and upload it again. This saves lots of time and space.
- We only had to install a new version (binaries) of PostgreSQL next to the old version.
- With pg_upgrade you can point to the old and new data folder and pg_upgrade will do the rest.
- Since we already had dropped our trending data we only had to migrate the history … also a huge win in time.

Making open source work for you

Upgrading to PG11

- /usr/pgsql-11/bin/pg_upgrade -k -b /usr/pgsql-10/bin/ -B /usr/pgsql-11/bin/ -d /data/data/ -D /data/11/ —check
- The -b option is to specify the old data folder
- The -B option is to specify the new data folder
- The -k option allows PostgreSQL to use hard links this will speed up the migration a lot as we don’t need to copy the data over.
- The -c or —check option will check the cluster for you without migrating the data to see if there are any errors.

TIPS

- TimescaleDB is an extension so remember that you need to upgrade it with your database.
- Don’t forget to add the extension in your postgresql.conf file
- The parameters for tuning can be kept in a config file like timescaledb.conf that you include in you postgresql.conf file this makes it easy to see what settings you change or add.

Making open source work for you

OUR SPECS

- ✓ 2000 Hosts
- ✓ 270.000 Items
- ✓ 96 000 Triggers
- ✓ 1700 VPS
- ✓ 1 Machine : 2x 4 Core CPU + BBC Raid 10 with 4x SSD for the Database + 2x SAS for the OS

Questions ?

Making open source work for you

Thank You !