Monitoring databases with zabbix

Wonder

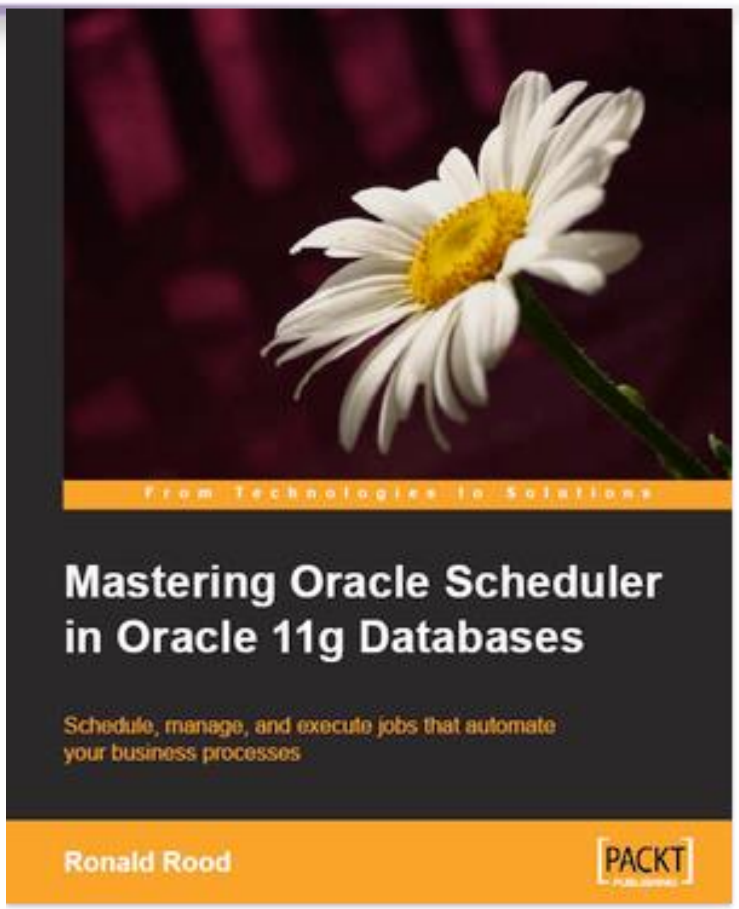
Wonder how it works?

- Ronald Rood @ik_zelf

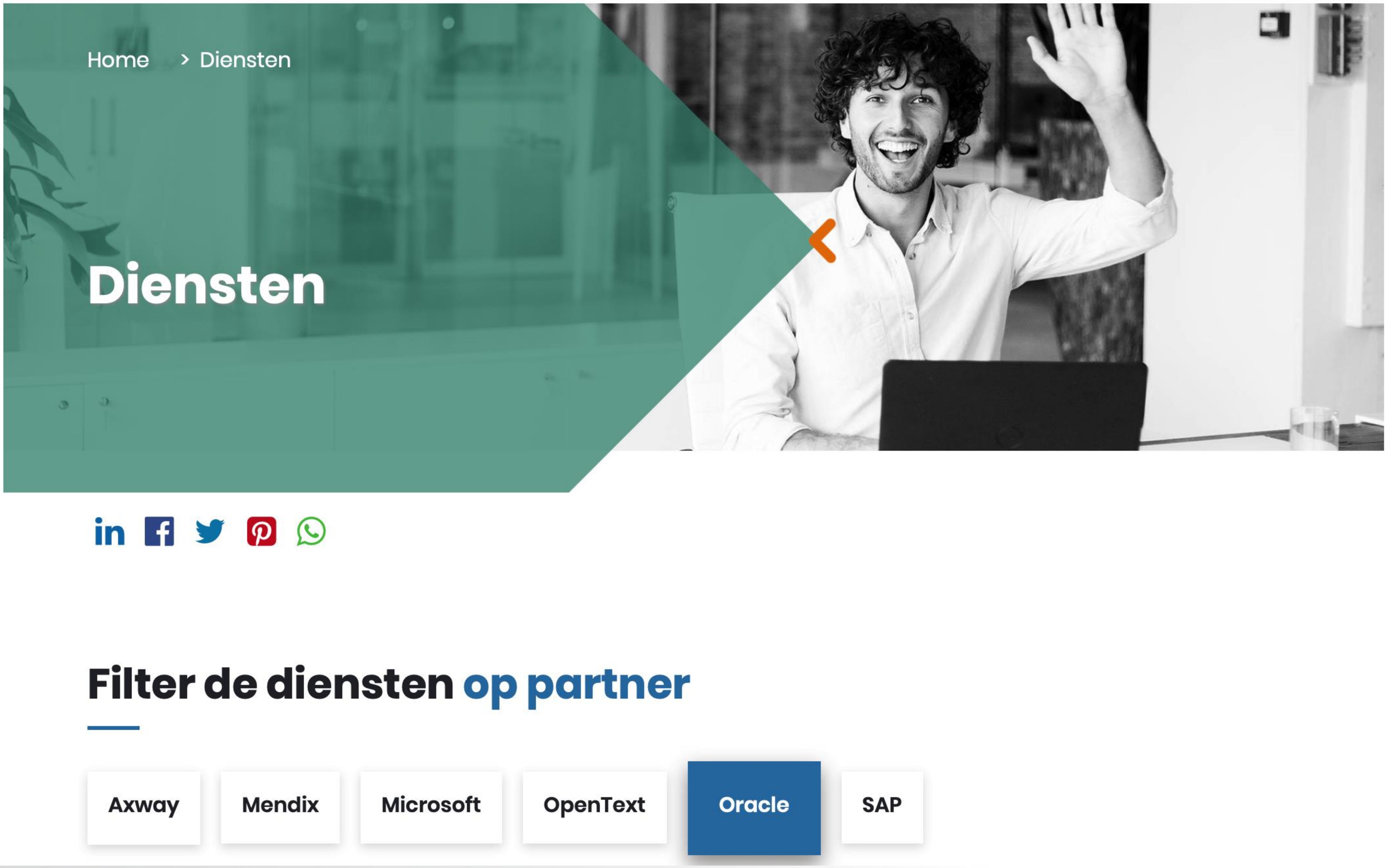
What I do

principal consultant @ Experis Ciber
[Oracle] DBA, also postgres, cockroachDB
Oracle ACE
Oracle Certified Master
Father of 2
Scouting
Skeeler

http://twitter.com/ik_zelf
http://github.com/ikzelf
http://ronr.blogspot.com
https://www.packtpub.com/big-data-and-business-intelligence/mastering-oracle-scheduler-oracle-11g-databases
monitoring enthusiast
IT veteran, does not believe something is impossible


From Technologies to Solutions
Mastering Oracle Scheduler in Oracle 11g Databases
Schedule, manage, and execute jobs that automate your business processes
Ronald Rood
PACKT

What Experis Ciber does



Home > Diensten

Diensten

Filter de diensten op partner

Axway Mendix Microsoft OpenText Oracle SAP

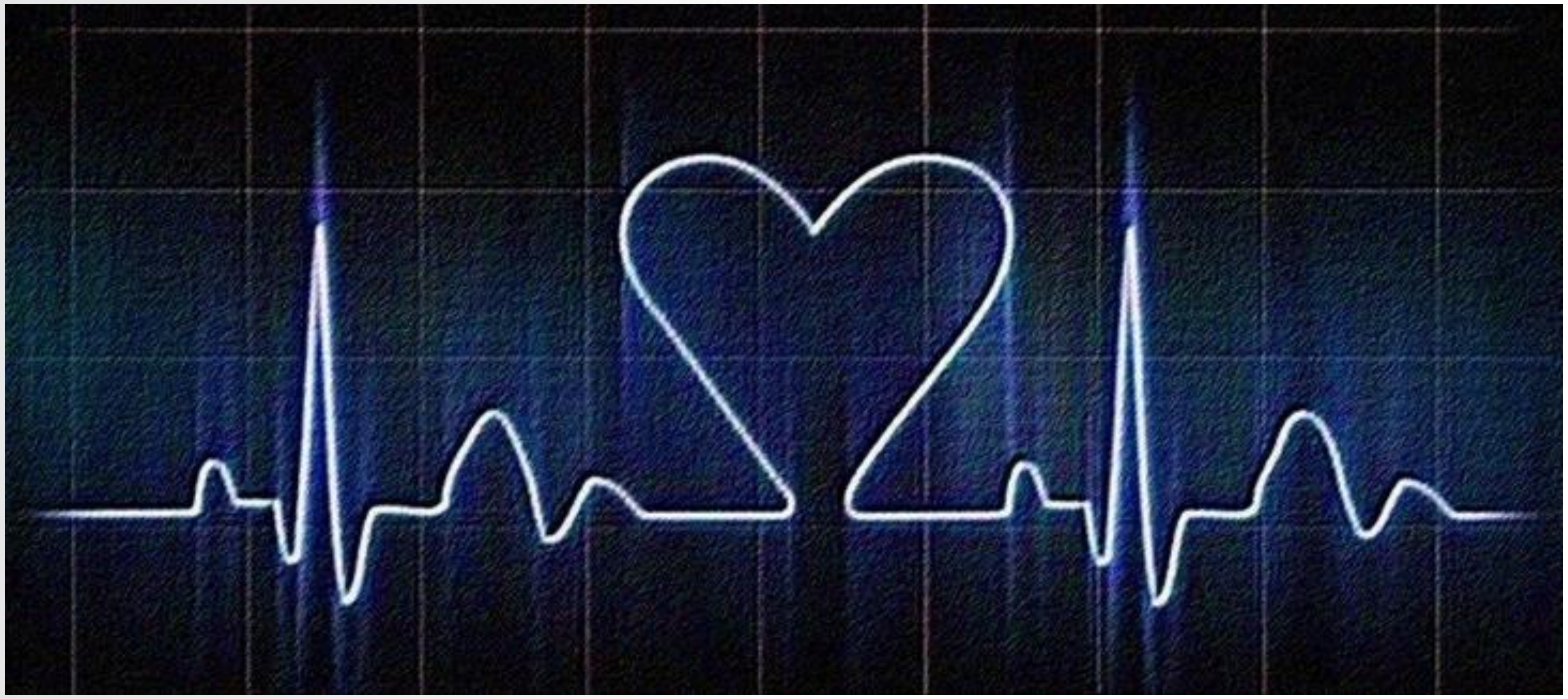
Klik om de tekststijl van het model te bewerken

• Klik om de tekststijl van het model te bewerken

Monitoring solutions

home grown scripts and email
Oracle Enterprise Manager
Oracle Grid Control
Oracle Cloud Control
Nagios
Zabbix

heart beat ….__________…



Why Zabbix?

- The design principle: KISS
- Heartbeat
- Mature
- [Re]active development
- Active community
  - IRC is very active irc://verne.freenode.net/zabbix
  - Zabbix forum https://www.zabbix.com/forum/
- Large installed base
- Very stable
- Good Oracle database citizen (can still be improved)
- easy to extend with plugins

Installation of zabbix - on Oracle

Installation from source

- - name: install instant client /usr/lib/oracle/12.1/client/
- yum: name={{ item }} state=present
- with_items:
- - oracle-instantclient12.1-basic-12.1.0.2.0-1.x86_64.rpm
- - oracle-instantclient12.1-devel-12.1.0.2.0-1.x86_64.rpm
- - oracle-instantclient12.1-precomp-12.1.0.2.0-1.x86_64.rpm
- - oracle-instantclient12.1-sqlplus-12.1.0.2.0-1.x86_64.rpm
- ./configure --enable-server --with-oracle=yes --with-oracle-include=/usr/include/oracle/12.1/client64 --with-oracle-lib=/usr/lib/oracle/12.1/client64/lib --with-net-snmp --with-ssh2 --with-openipmi --with-ldap --with-libcurl --with-jabber --with-unixodbc --with-openssl --with-libxml2"

pre build packages yum

- zabbix offical yum repository at http://repo.zabbix.com/

server and proxy must have same main version

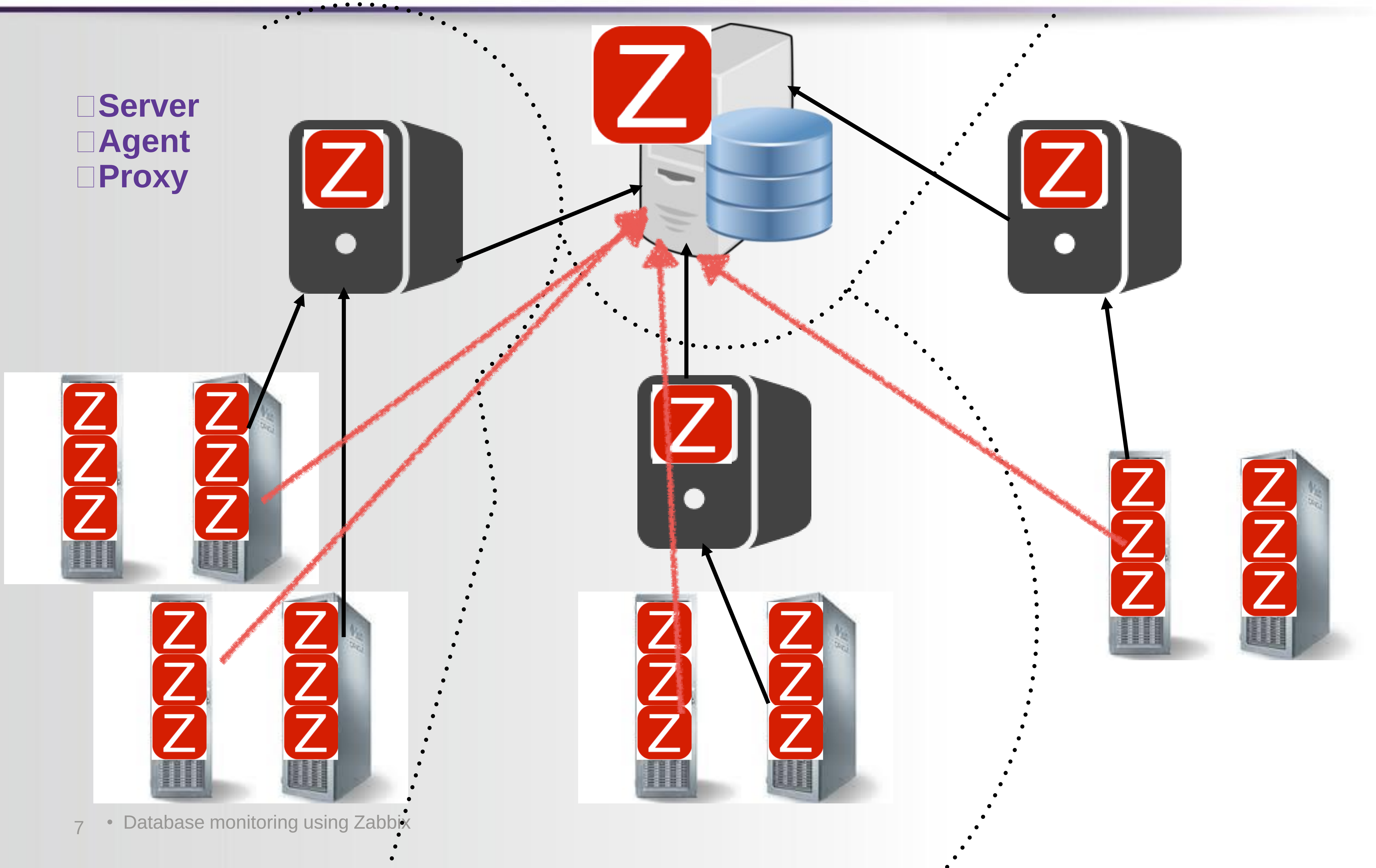
- this is a pity, nice would be to have backward compatibility to make upgrades more manageable

server (and proxy) supports all versions of agents

- wow!

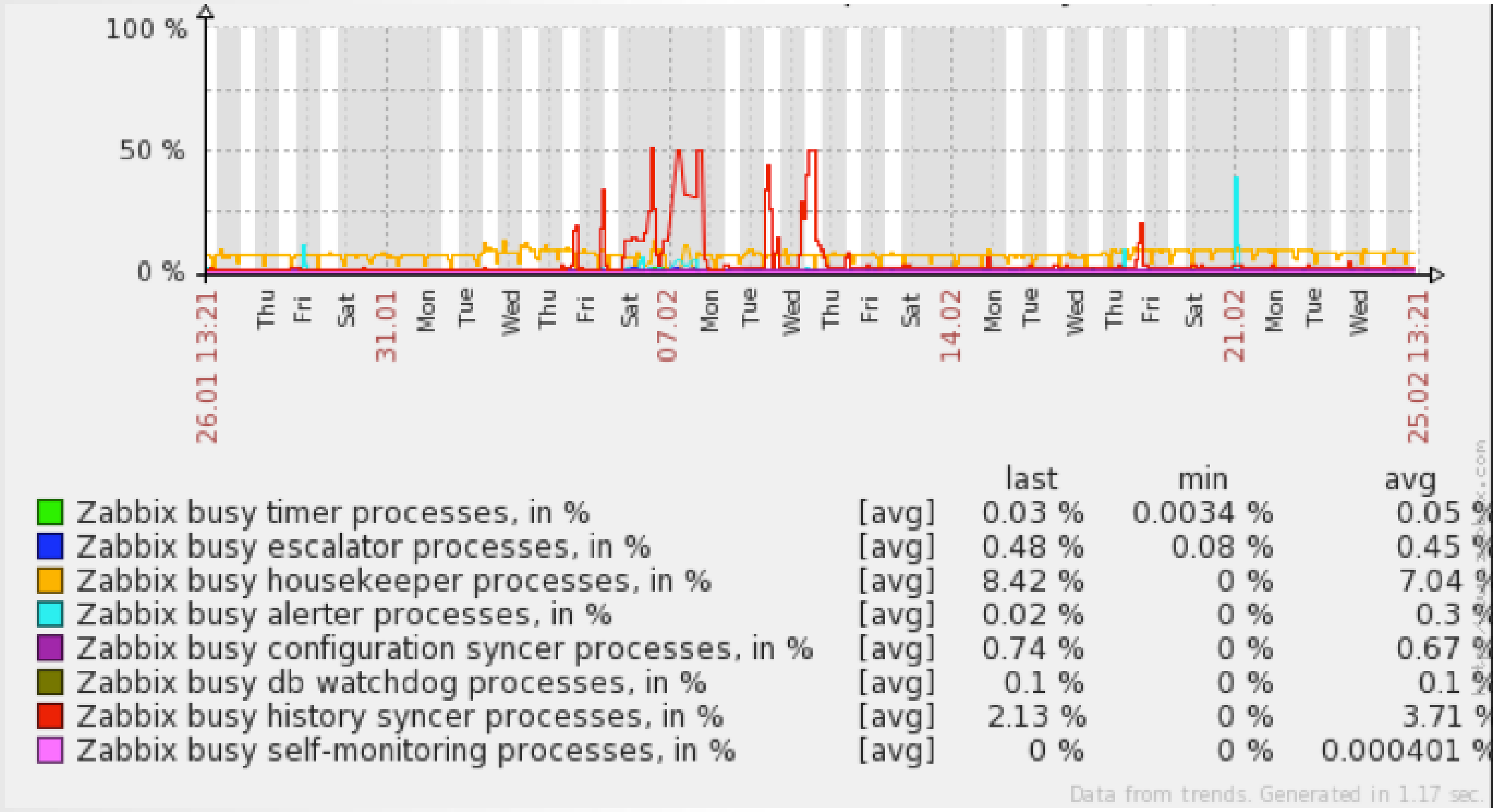
Configuration of zabbix

- Server
- Agent
- Proxy



Configuration of zabbix

- **server does all central functions**
  - update database
- insert new data
- maintain history
  - analyse data
  - trigger alerts
  - activate scripts/actions
- push notifications - Telegram
- ticket creations
  - maintain history



		last	min	avg
🟩 Zabbix busy timer processes, in %	[avg]	0.03 %	0.0034 %	0.05
🟦 Zabbix busy escalator processes, in %	[avg]	0.48 %	0.08 %	0.45
🟧 Zabbix busy housekeeper processes, in %	[avg]	8.42 %	0 %	7.04
🟦 Zabbix busy alerter processes, in %	[avg]	0.02 %	0 %	0.3
🟪 Zabbix busy configuration syncer processes, in %	[avg]	0.74 %	0 %	0.67
🟫 Zabbix busy db watchdog processes, in %	[avg]	0.1 %	0 %	0.1
🟥 Zabbix busy history syncer processes, in %	[avg]	2.13 %	0 %	3.71
🟪 Zabbix busy self-monitoring processes, in %	[avg]	0 %	0 %	0.000401

Data from trends. Generated in 1.17 sec.

Configuration of zabbix - with ansible

- **agent does data collection**
  - active
  - passive
  - auto registration - must be active agent

```
 - name: adjust agent config file
   lineinfile: dest={{ item.file }} regexp="^{{ item.key }} *=" line="{{ item.key }} = {{ item.value
}}" create=yes state=present
   with_items:
   - { file: /etc/zabbix/zabbix_agentd.conf, key: LogFileSize, value: 1 }
   - { file: /etc/zabbix/zabbix_agentd.conf, key: User, value: zabbixa }
   - { file: /etc/zabbix/zabbix_agentd.conf, key: Hostname, value: "{{ zabbix_hostname }}" }
   - { file: /etc/zabbix/zabbix_agentd.conf, key: HostMetadataItem, value: "system.uname" }
   - { file: /etc/zabbix/zabbix_agentd.conf, key: Server, value: "{{ zabbix_agents_server }}" }
   - { file: /etc/zabbix/zabbix_agentd.conf, key: ServerActive, value: "{{ zabbix_agents_server
}}" }
```

Configuration of zabbix

- **proxy way to couple networks to server**
  - active
  - passive
  - does caching when server not reachable
  - since v3 encryption supported
  - think about upgrades!

Upgrade from v2 to v3 to v4

1. **Prepare packages**
2. **shutdown proxies**
3. **shutdown server**
4. **when using sqlite as proxy database: remove cache database (no upgrade)**
5. **upgrade proxies**
6. **start proxies - this re-creates the cache database (no upgrade for sqlite)**
7. **upgrade server**
8. **start server**
9. **cached data comes in first (maintenance mode might prevent a few alerts)**

Downtime is a matter of minutes
Database upgrade is automatic

Data collection

- data collection is not limited by agents
- user scripts can collect data and act as extension for active agents
  - key,script -> stdout
  - keep it quick!
- user scripts can collect data and use zabbix_sender to send data to server
- zabbix_sender not only handles measurements, also lld json arrays (on one line)

templates

- the biggest pitfall is creating items on hosts
- almost certainly there will be more hosts with same item[s]
- create templates
- use MACROS for tunables
- use value lists to explain the meaning of values
- use prefixes to visually relate MACROS to templates

lld - Low Level Discovery

- very powerful mechanism to detect variable lists of items (tablespaces,users)
- LLD basically gives lists of items
- passes a json array to the server
- think about what happens when item is no longer discovered
- The discovered items can have all kinds of definitions on them (triggers, graphs, screens)
- In the discovery rule there are the prototypes
- also possible for HOSTS

example lld data Permanent TableSpaces

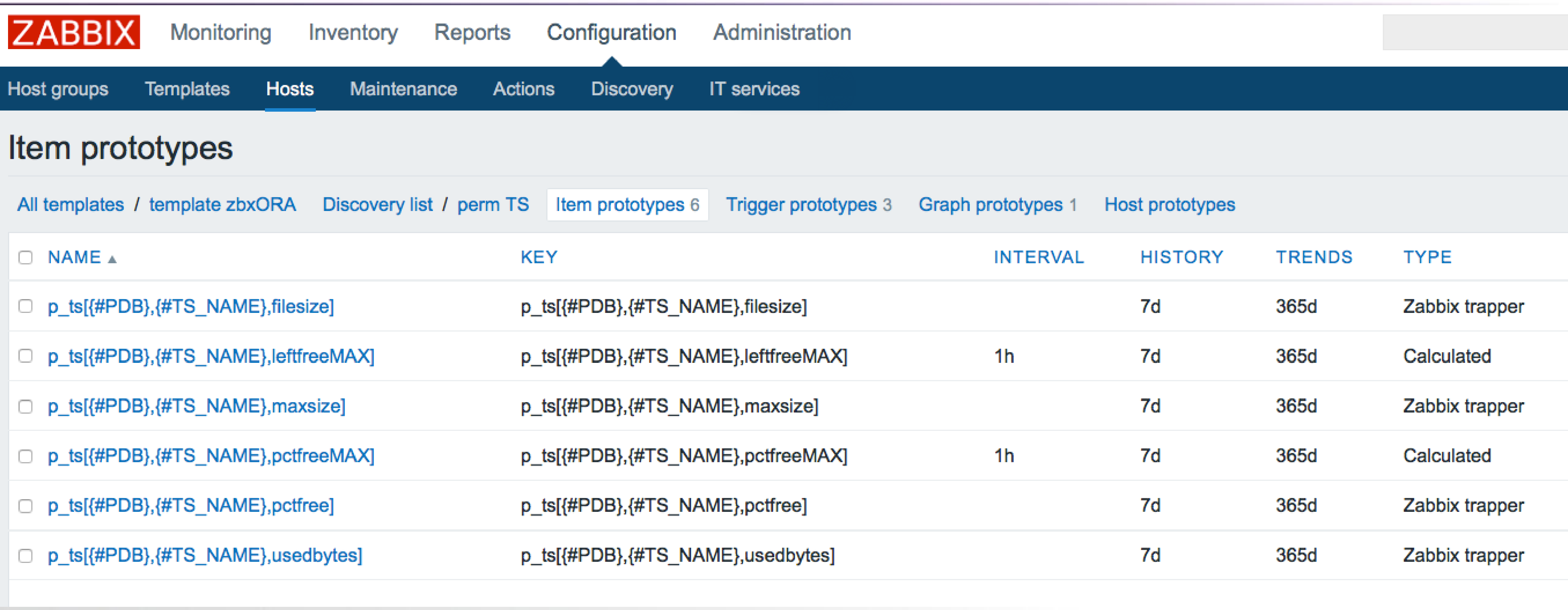
```
host p_ts.lld 1458212406 {“data":[
     {"{#TS_NAME}": "SYSTEM", "{#PDB}": null}
    , {"{#TS_NAME}": "CTXD", "{#PDB}": null}
    , {"{#TS_NAME}": "OWAPUB", "{#PDB}": null}
    , {"{#TS_NAME}": "ODM", "{#PDB}": null}
<snip>
]}
```

host has to be known in zabbix and have the template attached
p_ts.lld has to be a key in the discovery rules for the host
1458212406 unix timestamp when the discovery was done (date “+%s”)
data is the discovered array

There will be item prototypes where {#TS_NAME} and {#PDB} are placeholders for the names in Item Prototypes

When using zabbix_sender, make sure the complete array is on 1 line.

Item prototypes for p_ts.lld



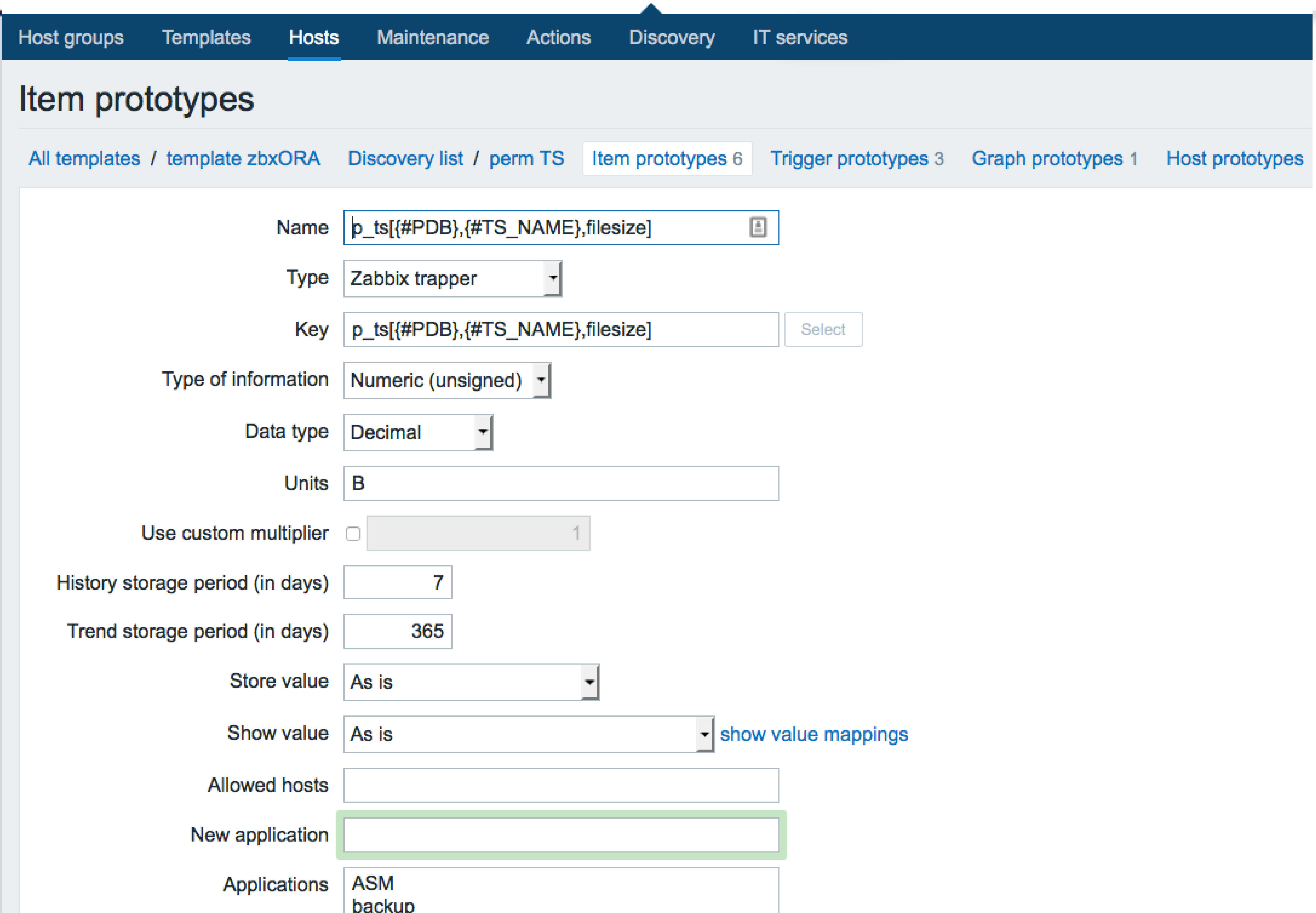
Item prototypes

All templates / template zbxORA   Discovery list / perm TS   Item prototypes 6   Trigger prototypes 3   Graph prototypes 1   Host prototypes

NAME ▲	KEY	INTERVAL	HISTORY	TRENDS	TYPE
p_ts[{#PDB},{#TS_NAME},filesize]	p_ts[{#PDB},{#TS_NAME},filesize]		7d	365d	Zabbix trapper
p_ts[{#PDB},{#TS_NAME},leftfreeMAX]	p_ts[{#PDB},{#TS_NAME},leftfreeMAX]	1h	7d	365d	Calculated
p_ts[{#PDB},{#TS_NAME},maxsize]	p_ts[{#PDB},{#TS_NAME},maxsize]		7d	365d	Zabbix trapper
p_ts[{#PDB},{#TS_NAME},pctfreeMAX]	p_ts[{#PDB},{#TS_NAME},pctfreeMAX]	1h	7d	365d	Calculated
p_ts[{#PDB},{#TS_NAME},pctfree]	p_ts[{#PDB},{#TS_NAME},pctfree]		7d	365d	Zabbix trapper
p_ts[{#PDB},{#TS_NAME},usedbytes]	p_ts[{#PDB},{#TS_NAME},usedbytes]		7d	365d	Zabbix trapper

Zabbix Trapper and Calculated types.

Simple item, from Zabbix trapper

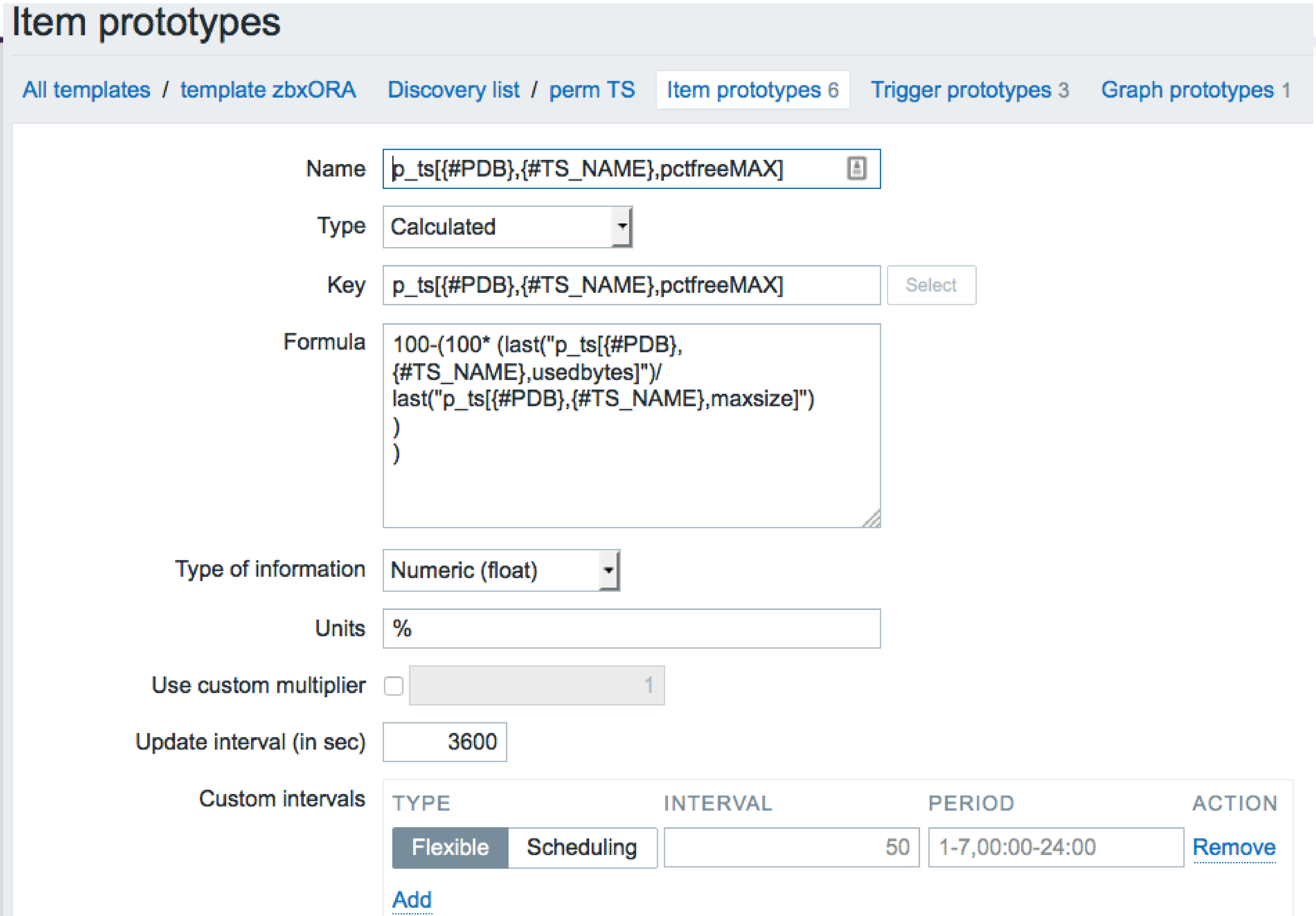


Item prototypes

All templates / template zbxORA   Discovery list / perm TS   Item prototypes 6   Trigger prototypes 3   Graph prototypes 1   Host prototypes

Name	p_ts[{#PDB},{#TS_NAME},filesize]
Type	Zabbix trapper
Key	p_ts[{#PDB},{#TS_NAME},filesize]
Type of information	Numeric (unsigned)
Data type	Decimal
Units	B
Use custom multiplier	☐ 1
History storage period (in days)	7
Trend storage period (in days)	365
Store value	As is
Show value	As is
Allowed hosts	
New application	
Applications	ASM backup

Calculated Item prototype



Item prototypes

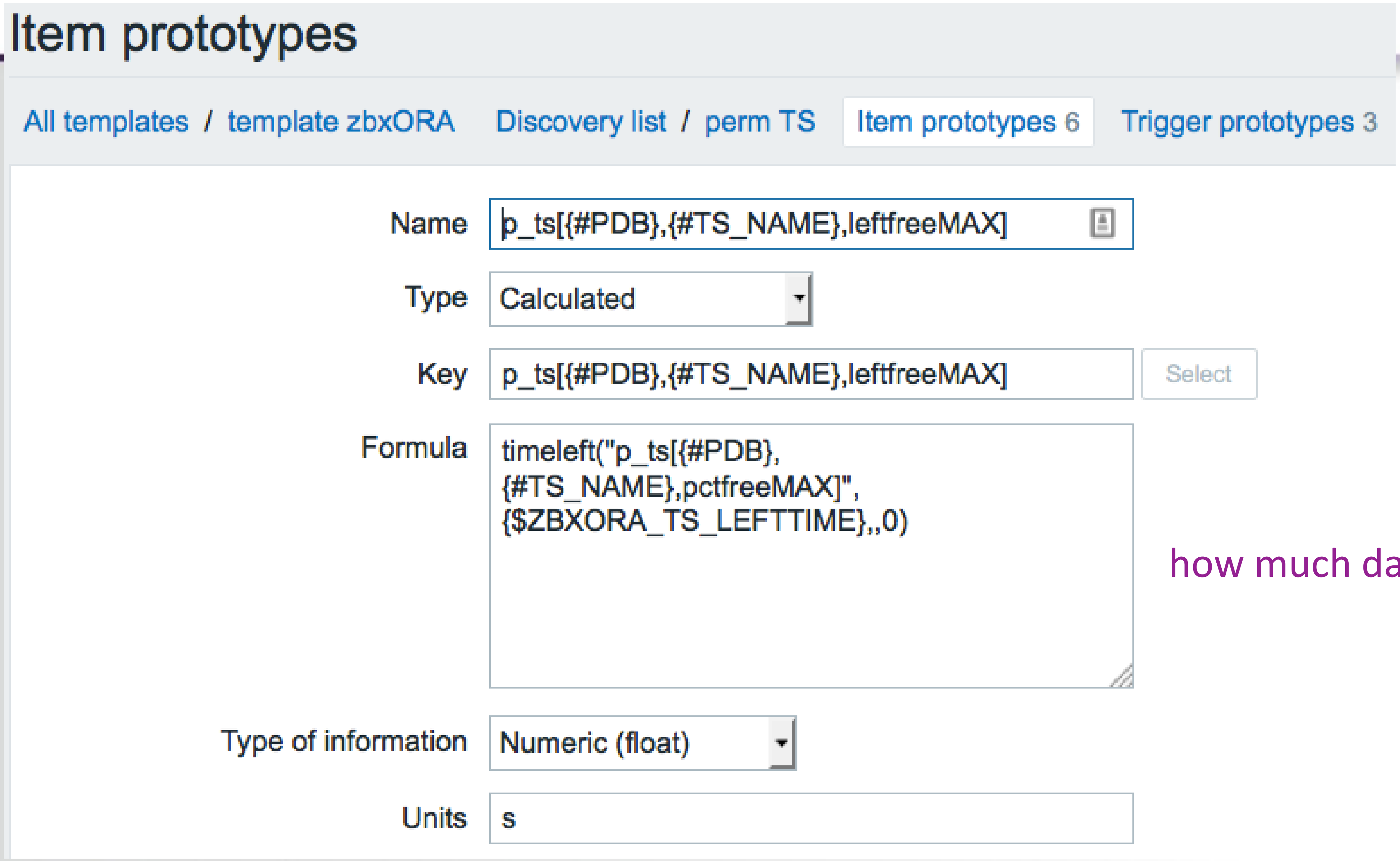
Name	p_ts[{#PDB},{#TS_NAME},pctfreeMAX]
Type	Calculated
Key	p_ts[{#PDB},{#TS_NAME},pctfreeMAX]
Formula	see below
Type of information	Numeric (float)
Units	%
Use custom multiplier	☐ 1
Update interval (in sec)	3600

```
100-(100* (last("p_ts[{#PDB},
{#TS_NAME},usedbytes]")/
last("p_ts[{#PDB},{#TS_NAME},maxsize]")
)
)
```

Custom intervals

TYPE	INTERVAL	PERIOD	ACTION
Flexible / Scheduling	50	1-7,00:00-24:00	Remove

Predictive Item Prototype



Item prototypes

All templates / template zbxORA   Discovery list / perm TS   Item prototypes 6   Trigger prototypes 3

Name	p_ts[{#PDB},{#TS_NAME},leftfreeMAX]
Type	Calculated
Key	p_ts[{#PDB},{#TS_NAME},leftfreeMAX]
Formula	timeleft("p_ts[{#PDB},{#TS_NAME},pctfreeMAX]",{$ZBXORA_TS_LEFTTIME},,0)
Type of information	Numeric (float)
Units	s

how much data to analyze?

Example data for zabbix_sender

```
host p_ts[,USERS,maxsize] 1458212417 524288000
host p_ts[,SYSTEM,maxsize] 1458212417 18027118592
host p_ts[,APPS_TS_INTERFACE,maxsize] 1458212417 18874368000
host p_ts[,ODM,maxsize] 1458212417 104857600
```

Think about quoting!
Space is column delimiter, if space can be in key, quote the key
Missing value? -> null

Host discovery

- **agent can register itself to the server**
- **server can scan for new hosts in the network**
- **with zabbix_sender we can auto define hosts using templates**
- a host is owner of discovered items … (also discovered hosts)

Database monitoring integration

- there are several tools to monitor databases and pass data to zabbix
- Zabbix since v3 also has internal odbc support
- tools like dbforbix Java based and a bit hard to grasp (for me)
  - http://www.smartmarmot.com/product/dbforbix/
- zbxora.py is born - Oracle only
  - https://github.com/ikzelf/zbxora
- zbxdb added as refactored copy of zbxora but database agnostic
  - https://share.zabbix.com/databases/multi-databases/zbxdb-generic-database-plugin
  - https://github.com/ikzelf/zbxdb

Database monitoring integration - zbxdb

zbxdb is a zabbix plugin consisting of

- zbxdb.py
- database query files for primary/standby/asm instances
- zabbix template
  - Low Level Discovery rules (lld)
  - items
  - triggers
  - graphs
- queries per vendor per version of database
- zbxdb_starter
- zbxdb_sender
- zbx_alertlog.sh
- zbx_discover_oradbs

works from zabbix v2 (never used v1)
Database versions depend on their python driver availability and capabilities

Database monitoring integration - zbxdb

- Very user extensible
- http://ronr.blogspot.com/2015/10/how-to-create-new-metrics-for-zbxora.html
- Very open
- Very simple to use
- runs from a client (the machine running the proxy is a good candidate)
- needs a regular Oracle client installation (instant client is OK) if monitoring oracle
- requires python 3 or newer
- requires database driver[s]
- monitors itself
- collects data in files per connection
- zbxdb_sender collects the zbxora output and sends them to the server(crontab)
- zbxdb_sender keeps a little history for debugging purposes
- zbxdb_starter is meant to guarantee your monitors are running (crontab)
- do NOT run as root or any database owner
- does NOT need any special OS privilege
- runs as a regular database client with monitoring privileges in the database
- uses 1 session per database and tries to keep that forever

Configuration of zbxdb

```
[zbxdb]
db_url = //IP-ADDRESS/ORAPROD1
username = cistats
password =
db_type = oracle
db_driver = cx_Oracle
instance_type = rdbms
role = normal
out_dir = $HOME/zbxora_out
hostname = OracleDB1
checks_dir = etc/zbxdb_checks
site_checks = sap,ebs
password_enc = Z2xhQUMzYTdi
```

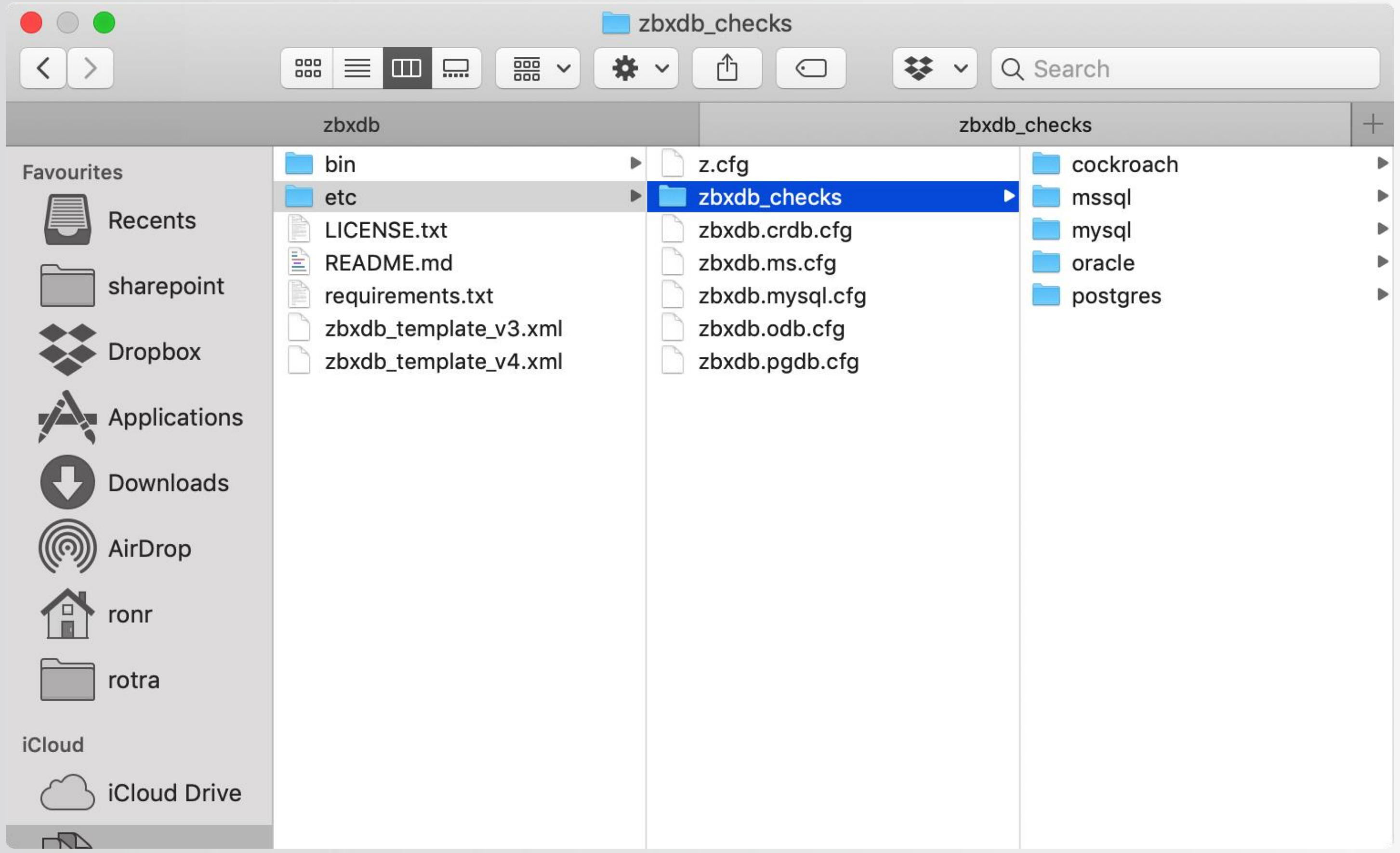
Note password_enc
Initially enter password and leave password_enc empty.
Upon first start zbxdb will fill password_enc with an ‘encrypted’ version of password and clear password in the config file.

other zbxdb config example

```
[zbxdb]
db_url: //localhost/fsdb01
username: cistats
password: knowoneknows
db_type: msssql
# db_type: postgres
# db_type: mysql
# db_type: mssql
# db_type: db2
server: hostname.domain
server_port: 1433
db_name: master
db_driver: pytds
# db_driver: psycopg2
# db_driver: mysql.connector
# db_driver: ibm_db_dbi
role: normal
# for ASM instance role should be SYSDBA
out_dir: $HOME/zbxora_out
hostname: testhost
checks_dir: etc/zbxdb_checks
site_checks: NONE
instance_type: rdbms
```

db_type

Can be anything.
db_type is used to find the SQL files in {checks_dir}
db_type should have it's own directory in {checks_dir}

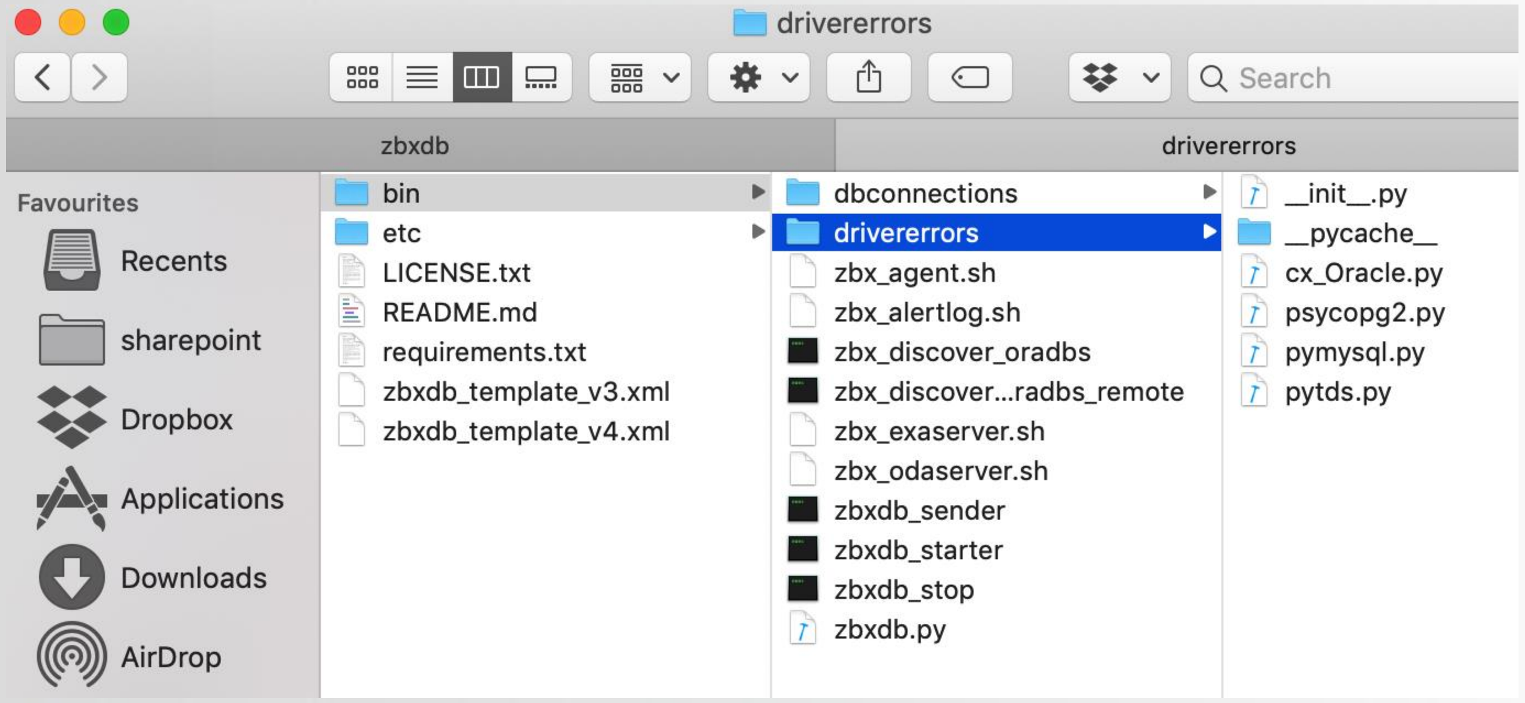


```
bin                      z.cfg              cockroach
etc                      zbxdb_checks       mssql
LICENSE.txt              zbxdb.crdb.cfg     mysql
README.md                zbxdb.ms.cfg       oracle
requirements.txt         zbxdb.mysql.cfg    postgres
zbxdb_template_v3.xml    zbxdb.odb.cfg
zbxdb_template_v4.xml    zbxdb.pgdb.cfg
```

db_type is also used to load the corresponding module from dbconnections

db_driver

zbxdb uses this driver to connect to the database.
It needs to be installed separately.
Since the driver raises the errors and since the drivers have different ways to report errors, there is also a drivererrors module


drivererrors

zbxdb: bin, etc, LICENSE.txt, README.md, requirements.txt, zbxdb_template_v3.xml, zbxdb_template_v4.xml

drivererrors: dbconnections, drivererrors, zbx_agent.sh, zbx_alertlog.sh, zbx_discover_oradbs, zbx_discover...radbs_remote, zbx_exaserver.sh, zbx_odaserver.sh, zbxdb_sender, zbxdb_starter, zbxdb_stop, zbxdb.py

__init__.py, __pycache__, cx_Oracle.py, psycopg2.py, pymysql.py, pytds.py

if you want to use a different driver, just create the corresponding script in drivererrors/ so it can be loaded by zbxdb

site_checks

The intention is to have your site or application specific checks here. In the git code there are only generic SQL’s aiming mostly on availability and capacity.

If no site_checks, just remove the parameter or make it empty.

instance_type

In Oracle we have RDBMS, ASM for instance types. For Oracle, the dbconnection module detects this byself. Others can do the same but for now, it is input - and mostly ‘rdbms’

Usage TIPs for zbxdb

- use zabbix server or zabbix proxy server as monitoring host
- use a separate Linux account to run zbxdb
- no special OS privileges needed
- do NOT run as root or a database owner
- does need zabbix_sender
- zabbix_sender needs access to zabbix_server or zabbix_proxy
- zbxdb hardly uses any CPU and is most of the time sleeping
- since zbxdb runs a separate process for every database, use zbxdb_starter
- zbxdb_starter launches all configuration that it finds with a second sleep between 2 starts, making sure there are no CPU spikes on the server
- if zbxdb wakes up on the 13th second, it will always try to wake up on a 13th second.
- zbxdb also monitors itself, if the script changes, it will relaunch itself
- zbxdb also monitors the checks_files. If they change, they will be reloaded
- zbxdb also monitors it's configuration file. If it is changed and zbxdb is not connected to a database, it will reload the config file.
- zbxdb uses about 24KB memory per instance.

Oracle Alertlog monitoring - lld

Alertlog discovery done by zbx_alertlog.sh and should be used as a user parameter for zabbix agent.
If used for Oracle, the agent's OS account should also have the Oracle dba group membership because

- alertog.sh will try to connect to each running instance to find the log.xml location that is passed to zabbix.
- needs to be able to read the log.xml

We also send lines with 'time=' to the server so eventually alerts can be cleared. For that we make sure that our databases perform a log switch at least every hour causing some lines to be written.

zbx_discover_oradbs - host discovery

zbx_discover_oradbs can be used to dynamically discover databases for zabbix.

It should be run from a monitoring host that can reach all databases for that site.
Use the zabbix_server or a zabbix_proxy as monitoring host.

the process tries to connect to the specified hosts
tries to connect to the remote listeners (after jumping to the host for local access)
finds the instances that the listener serves
tries to generate a databases list from that.

This is tested on exadata with RAC clusters and single instance db's
I consider this as a manual activity but it could be done in crontab.

configfile example:

```
# site_prefix (clustername|"") host[s]
cust1 dm01 dm01db01 dm01db02
cust1 dm02 dm02db01 dm02db02
cust1 "" srv-dbs-001
```

```
zbx_discover_oradb your_host [(zabbix|proxy)_server]
```

zbx_discover_oradbs - output

```
>zbxdb/bin/zbx_discover_oradbs your_host 2>/dev/null
reads etc/zbx_discover_oradata.cfg

your_host oradb.lld 1547653101 { "data":[
 "{#DB_NAME}":"cust1_dm01_ASM"
,"{#DB_NAME}":"cust1_dm01_DBS1"
,"{#DB_NAME}":"cust1_dm01_DBS2"
,"{#DB_NAME}":"cust1_dm01_DBS3"
,”{#DB_NAME}”:”cust1_dm01_DBS3"
<snip>
,"{#DB_NAME}":"cust1_dm02_OTA1"
,"{#DB_NAME}":"cust1_dm02_OTA2"
,"{#DB_NAME}":"cust1_CC12"
]}
```

join list to 1 single line, prefix it with the host and discovery key before sending with zabbix_sender (zbx_discover_oradbs does it when sending to zabbix)

In discovery rules add the template[s]

my requests for zabbix

make new server compatible with previous version of proxy.
make remote tasks possible for agents behind proxy - will be done.
make more use of bulk operations when inserting in the database.
make use of an install and of a runtime user in the database.
make use of read connections to the database, for read only access
make use of write connection to the database, when read only does not fit.
make a nice mobile version of the web app.

Questions?

danke
謝謝
ngiyabonga
teşekkür ederim
спасибо
dank je
tapadh leat
thank you
gracias
hvala
mauruuru
bedankt
dziękuję
mochchakkeram
go raibh maith agat
obrigado
sagolun
sukriya
kop khun krap
grazie
arigatô
takk
dakujem
мерси
terima kasih
감사합니다
ευχαριστώ
merci