PostgresPro

Mamonsu 3.0

New Features and Quick Start

Alexandra Kuznetsova
Postgres Professional, 2021

Mamonsu is an active monitoring agent for collecting PostgreSQL and system metrics that can be visualized via Zabbix.

'Monitoring PostgreSQL using Zabbix' by Daria Vilkova

GitHub repo

- Written in Python3
- Data gathering by the *Zabbix trapper*
- Extensibility: you can write your own custom plugins
- ~40 system metrics, ~90 PostgreSQL metrics
- Additional tools for configuring Zabbix server, generating system state reports, etc.

⚠ Mamonsu 3 is incompatible with the previous versions.

What's new

- New template name
- Changed the way some metrics are calculated
- Graphs upgrade
- New PostgreSQL 14 metrics
- Zabbix screens
- *bootstrap* tool improvement
- Zabbix API support improvement
- GitHub Actions features

New template name

The template was difficult to find after loading.

PostgresPro-*OS*  Mamonsu PostgreSQL *OS*

Metrics update: PostgreSQL transactions committed

PostgreSQL transactions total  PostgreSQL transactions committed

```
select
sum(xact_commit) from
pg_catalog.pg_stat_database;
```

Metrics update: PostgreSQL uptime

Name ▲	Last check	Last value
App-PostgresPro-Linux (1 Item)		
PostgreSQL: service uptime	2021-10-19 15:38:32	06:05:34

seconds

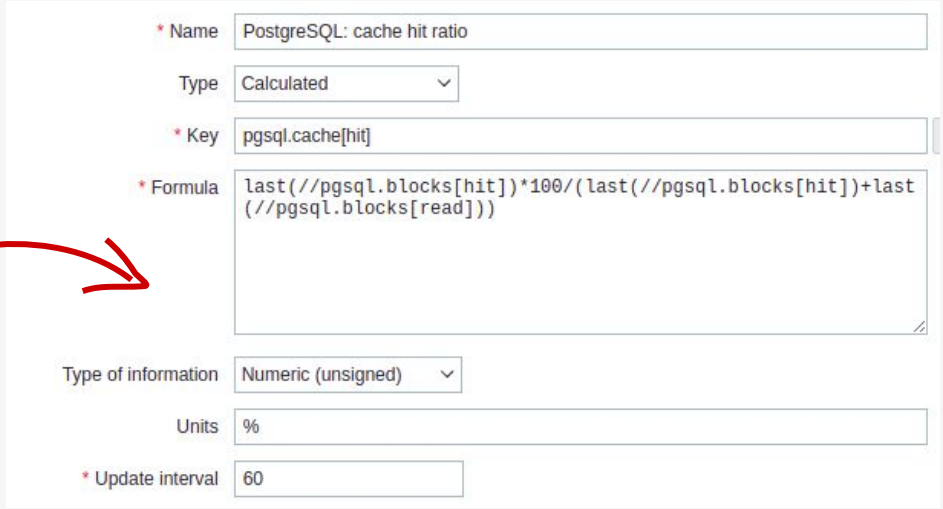
Name ▲	Last check	Last value
Mamonsu PostgreSQL Linux (1 Item)		
PostgreSQL: service uptime	2021-10-19 15:19:02	2021-10-19 09:32:58

unixtime

Metrics update: PostgreSQL cache hit ratio

In the previous version, this metric was calculated based on data for the entire PostgreSQL running time. In Mamonsu 3, it is evaluated via Zabbix Calculated Item using *blocks hit* and *blocks read* deltas.

```
select
round(sum(blks_hit)*100/
sum(blks_hit+blks_read), 2)
from
pg_catalog.pg_stat_database;
```



Metrics update: Archive Command plugin

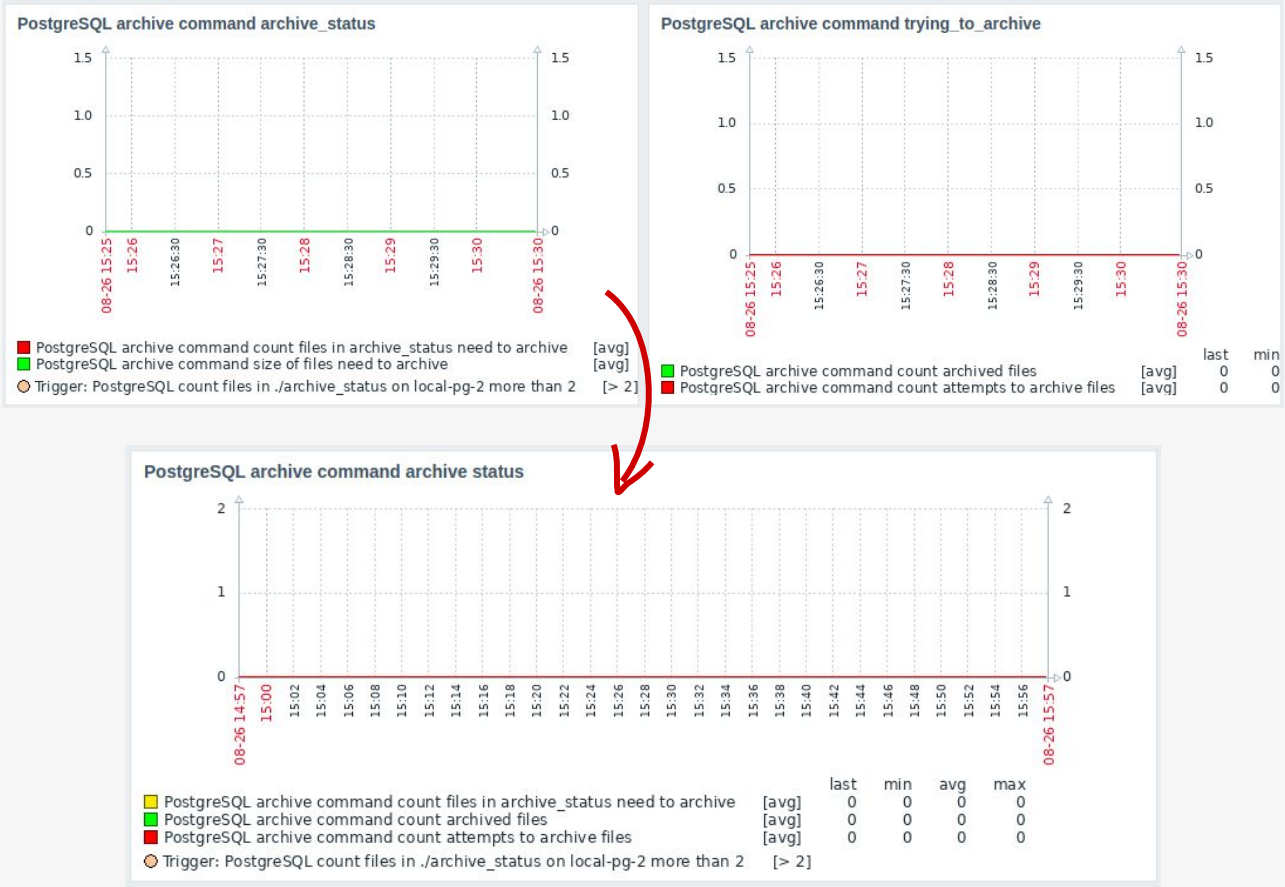
- File system access exclusion
- Evaluates for MASTER only

```
wal_segment_size
+
pg_stat_archiver.last_archived_wal
+
pg_current_wal_lsn()
```

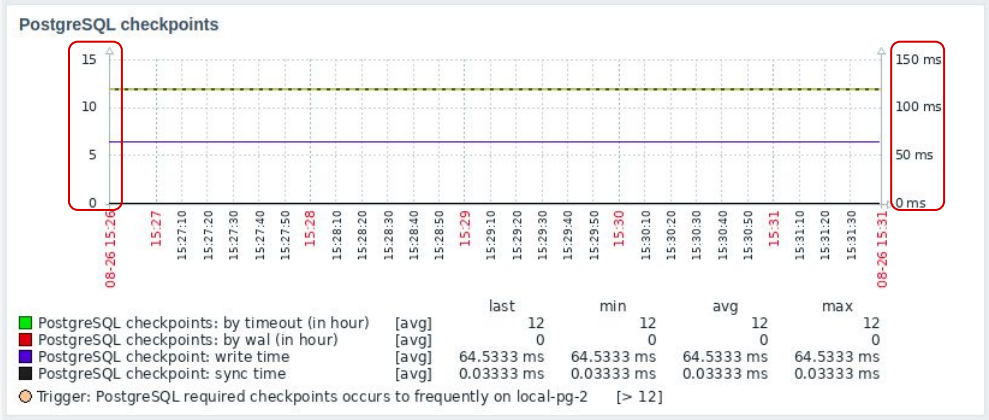
Deleted graphs

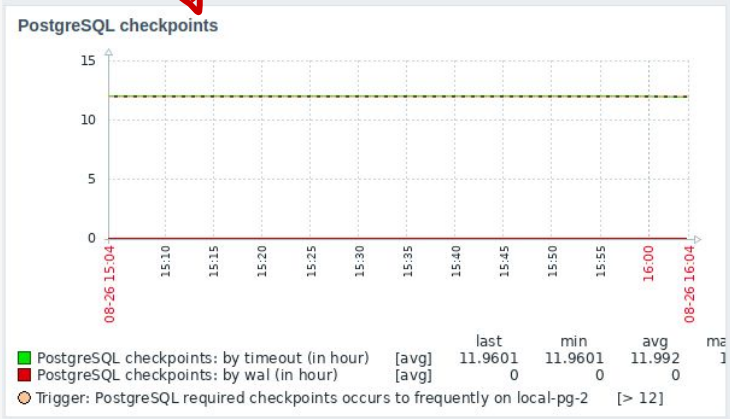
- PostgreSQL uptime
- PostgreSQL cfs compression: current ratio
- PostgreSQL cfs compression: compressed files
- PostgreSQL cfs compression: written bytes
- PostgreSQL cfs compression: total ratio
- PostgreSQL oldest transaction running time
- PostgreSQL age of oldest xid
- PostgreSQL number of parallel queries being executed now
- PostgreSQL write-ahead log generation speed
- PostgreSQL replication lag in second
- PostgreSQL count of xlog files
- System load average
- System: count of opened files
- System up_time

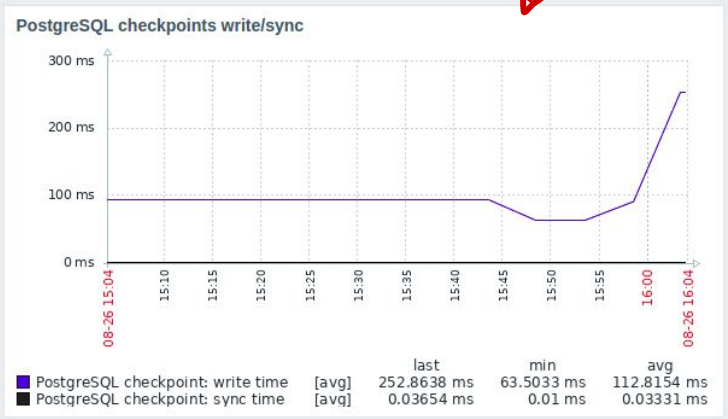
Graphs update: PostgreSQL archive command



Graphs update: PostgreSQL checkpoints







New metrics: pg_stat_wal

pg_stat_wal
wal_records
wal_fpi
wal_bytes
wal_buffers_full
wal_write
wal_sync
wal_write_time
wal_sync_time
stats_reset

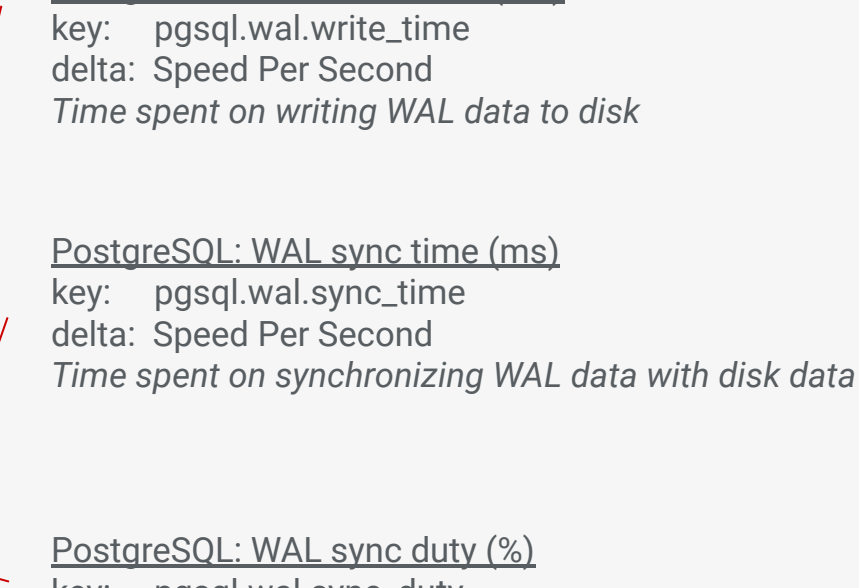
PostgreSQL: WAL records generated
key: pgsql.wal.records.count[]
delta: Speed Per Second
Number of WAL records generated per second

PostgreSQL: WAL full page images generated
key: pgsql.wal.fpi.count[]
delta: Speed Per Second
Number of WAL full page images generated per second
(full_page_writes = on)

PostgreSQL: WAL buffers full
key: pgsql.wal.buffers_full
delta: Speed Per Second
Number of times when WAL data was forcibly written to disk due to buffer overflows -> wal_buffers adjustment

New metrics: pg_stat_wal

pg_stat_wal
wal_records
wal_fpi
wal_bytes
wal_buffers_full
wal_write
wal_sync
wal_write_time
wal_sync_time
stats_reset

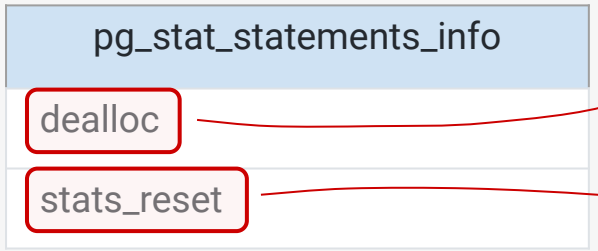


PostgreSQL: WAL write time (ms)
key: pgsql.wal.write_time
delta: Speed Per Second
Time spent on writing WAL data to disk

PostgreSQL: WAL sync time (ms)
key: pgsql.wal.sync_time
delta: Speed Per Second
Time spent on synchronizing WAL data with disk data

PostgreSQL: WAL sync duty (%)
key: pgsql.wal.sync_duty
delta: Speed Per Second
Percentage of time spent on synchronizing WAL data to disk during the metric collection period
Source: github.com/zubkov-andrei/pg_profile

New metrics: pg_stat_statements_info

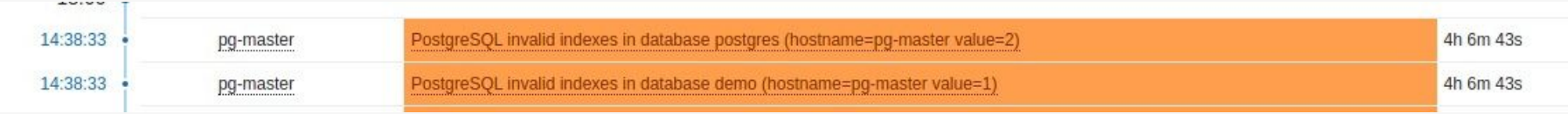


PostgreSQL: statements dealloc
key: pgsql.stat_info[dealloc]
delta: Simple Change
Number of times records were cleared due to exceeding pg_stat_statements.max

PostgreSQL: statements last reset
key: pgsql.stat_info[stats_reset]
delta: As Is
Time of last pg_stat_statements statistics reset

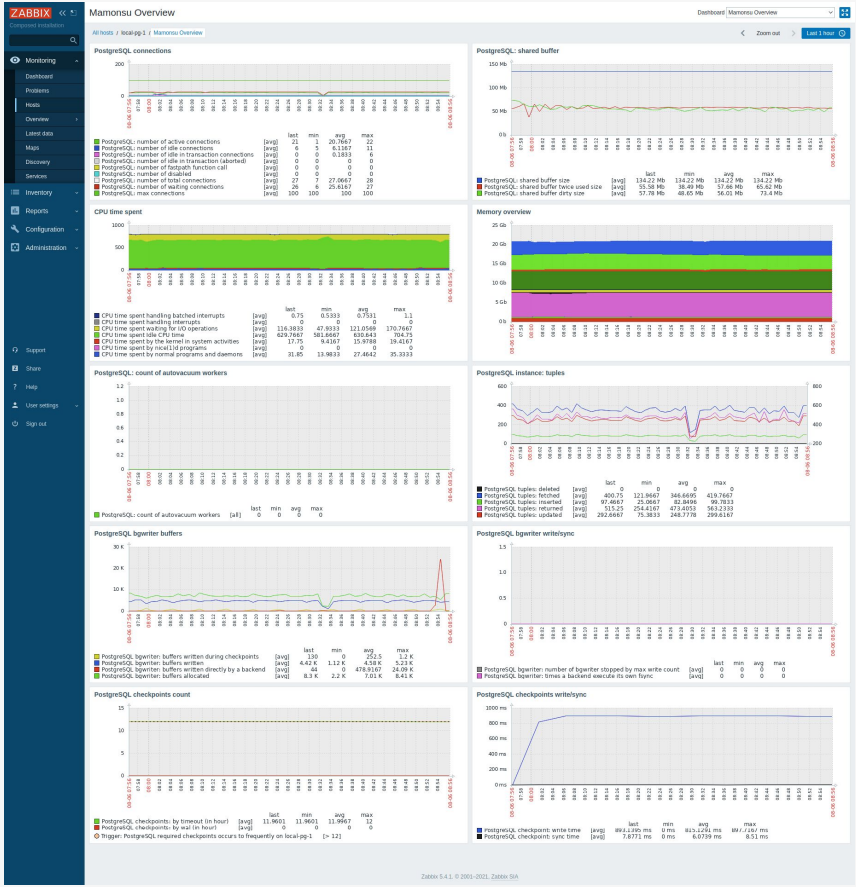
New metrics: Count of invalid indexes

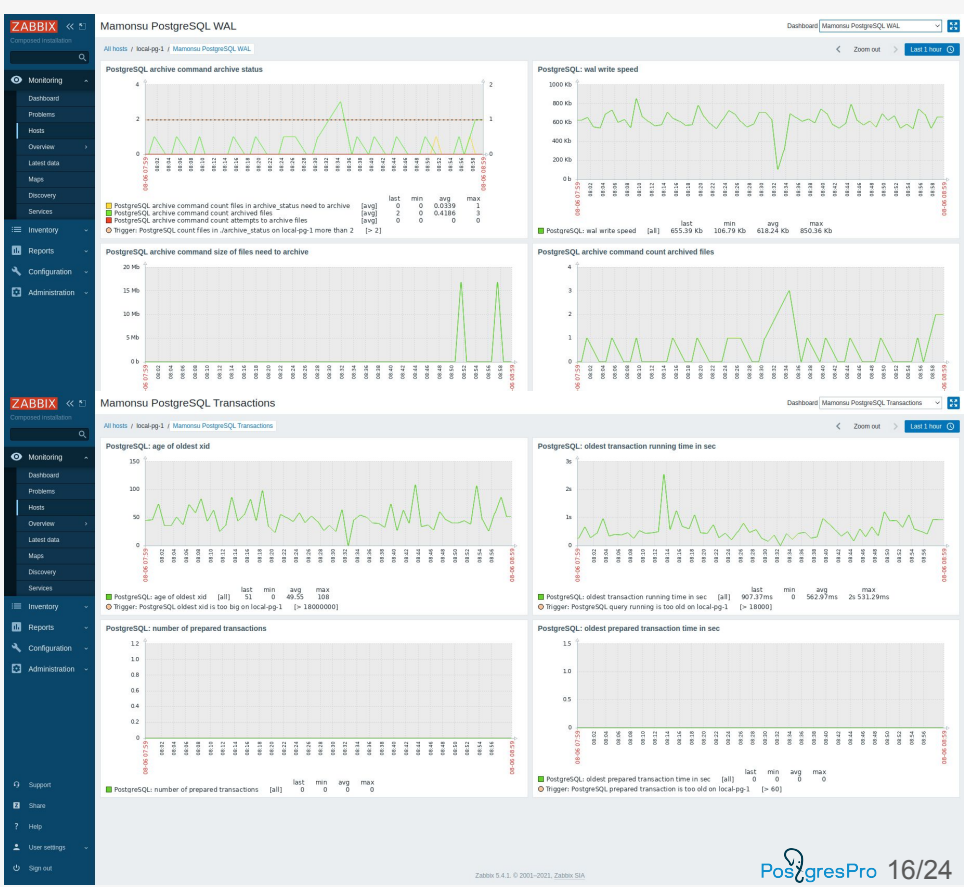
> *If a problem arises while scanning the table, such as a deadlock or a uniqueness violation in a unique index, the* `CREATE INDEX` *command will fail but leave behind an "invalid" index.*




```
SELECT count (*)
FROM pg_catalog.pg_index i LEFT JOIN pg_catalog.pg_locks l
ON (i.indexrelid = l.relation)
WHERE i.indisvalid = false AND l.relation IS NULL;
```

New features: Zabbix screens





Plugin structure update

A new *dashboard* argument in the template generating functions *items()*, *discovery_rules()*, *graphs()*, *triggers()*

Parameters:
`'name'` - item key/graph name
`'page'` - screen name
`'size'` - widget size
`'position'` - screen position (optional)

```
def items(self, template, dashboard=False):
        result = ''
        if self.Type == "mamonsu":
            delay = self.plugin_config('interval')
            value_type = Plugin.VALUE_TYPE.numeric_unsigned
        else:
            delay = 5  # TODO check delay
            value_type = Plugin.VALUE_TYPE.numeric_float
        result += template.item({
            'name': 'PostgreSQL: ping',
            'key': self.right_type(self.key_ping),
            'value_type': Plugin.VALUE_TYPE.numeric_float,
            'units': Plugin.UNITS.ms,
            'delay': delay
        })
        if not dashboard:
            return result
        else:
            return [{'dashboard': {'name': self.right_type(self.key_ping),
                                   'page':
ZbxTemplate.dashboard_page_instance['name'],
                                   'size':
ZbxTemplate.dashboard_widget_size_medium,
                                   'position': 1}}]
```

documentation:
https://github.com/postgrespro/mamonsu/blob/master/documentation/adding_custom_plugins.md

New in interaction with Zabbix: different versions support

Supported Zabbix versions: 3.0 - 5.4

What were the inaccuracies?

```
$ mamonsu zabbix export template template.xml
```

- Some Zabbix API compatibility errors
- *Zabbix CLI* tool enhancements
- *'deleteMissing'* argument related issues

New in *bootstrap*: automatic user and schema creation

```
$ mamonsu bootstrap -U postgres -d mamonsu_db -M mamonsu_user
```

Default user: *mamonsu/mamonsu*, schema *mamonsu*

```
mamonsu_db=# \dn+ mamonsu
                   List of schemas
  Name   |  Owner  | Access privileges   | Description
---------+---------+---------------------+-------------
 mamonsu | mamonsu | mamonsu=UC/mamonsu |

mamonsu_db=# \df mamonsu.*
       List of functions
 Schema  |          Name
 mamonsu | archive_command_files
 mamonsu | archive_stat
 mamonsu | buffer_cache
 mamonsu | count_autovacuum
 mamonsu | count_wal_files
<...>
```

New in *bootstrap*: -x/--create-extensions

```
$ mamonsu bootstrap -U postgres -d mamonsu_db


mamonsu_db=# \dx
List of installed extensions
-[ RECORD 1 ]------------------------------
Name        | plpgsql
Version     | 1.0
Schema      | pg_catalog
Description | PL/pgSQL procedural language

mamonsu_db=# \df mamonsu.buffer*
(0 rows)
```

```
$ mamonsu bootstrap -U postgres -d mamonsu_db -x


List of installed extensions
-[ RECORD 1 ]------------------------------
Name        | pg_buffercache
Version     | 1.3
Schema      | mamonsu
Description | examine the shared buffer cache
-[ RECORD 2 ]------------------------------
Name        | plpgsql
Version     | 1.0
Schema      | pg_catalog
Description | PL/pgSQL procedural language

mamonsu_db=# \df mamonsu.buffer*
List of functions
-[ RECORD 1 ]-------+------------------------
Schema              | mamonsu
Name                | buffer_cache
Type                | func
```

Frequent *pg_buffercache* function calls can affect performance, so it was necessary to organize the optional extensions installation.

Documentation

Table of Contents

Configuration

1. **Optionally, bootstrap Mamonsu**

   If you omit this step, metrics can only be collected on behalf of a superuser, which is not recommended.

   Create a non-privileged database user for Mamonsu. For example:

   ```
   CREATE USER mamonsu_user WITH PASSWORD 'mamonsu_password';
   ```

   Create a database that will be used for connection to PostgreSQL. For example:

   ```
   CREATE DATABASE mamonsu_database OWNER mamonsu_user;
   ```

   Run the following command to bootstrap Mamonsu:

   ```
   mamonsu bootstrap [-M mamonsu_user] [-x | --create-extensions] [connection_options]
   ```

   For details of usage, see "Tools".

   As the result of this operation, monitoring functions are created in the mamonsu_database in *mamonsu* schema, and the right to execute them is granted to the mamonsu_user. Thus, a superuser connection is no longer required. Mamonsu also creates several tables in the specified database. Do not delete these tables as they are required for Mamonsu to work.

2. **Configure Mamonsu**

   Edit the agent.conf configuration file.

   Configure Zabbix-related settings. The address field must point to the running Zabbix server, while the client field must provide the name of the Zabbix host. You can find the list of hosts available for your account in the Zabbix web interface under Configuration > Hosts.

   ```
   [zabbix]
   ; enabled by default
   enabled = True
   client = zabbix_host_name
   address = zabbix_server
   ```

   By default, Mamonsu will collect both PostgreSQL and system metrics. If required, you can disable metrics collection of either type by setting the enabled parameter to False in the [postgres] or [system] section of the agent.conf file, respectively.

Mamonsu upgrade

⚠ Mamonsu 3 is incompatible with the previous versions.

Do not save old data

1. Generate a new template for the Zabbix server
2. If you performed a *bootstrap* using the previous version of mamonsu, run the *bootstrap* command again
3. Upload the new template to the Zabbix server
4. Link the new template to the host instead of the old one

Save old data

1. Generate a new template for the Zabbix server
2. If you performed a *bootstrap* using the previous version of mamonsu, run the *bootstrap* command again
3. Upload the new template to the Zabbix server
4. Rename the host for which you want to retain the collected data and leave the old template linked to that host
5. Create a new host for the same system and link the new template to it

Mamonsu upgrade

Scripts: https://github.com/postgrespro/mamonsu#best-practices

```
zabbix=# SELECT host, name FROM hosts
zabbix-# WHERE host LIKE '%local-pg%';
-[ RECORD 1 ]----
host | local-pg-2
name | local-pg-2
-[ RECORD 2 ]----
host | local-pg-3
name | local-pg-3

zabbix=# UPDATE hosts
zabbix=# SET host = host || ' OLD-MAMONSU',
zabbix=#     name = name || ' OLD-MAMONSU'
zabbix=# WHERE host LIKE '%local-pg%';
UPDATE 2
zabbix=# SELECT host, name FROM hosts
zabbix=# WHERE host LIKE '%local-pg%';
-[ RECORD 1 ]----------------
host | local-pg-2 OLD-MAMONSU
name | local-pg-2 OLD-MAMONSU
-[ RECORD 2 ]----------------
host | local-pg-3 OLD-MAMONSU
name | local-pg-3 OLD-MAMONSU
```

```
curl -H "Content-type: application/json-rpc" -X POST
http://zabbix/api_jsonrpc.php -d'
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "HOST_ID",
        "host": "local-pg-3 OLD-MAMONSU",
        "name": "local-pg-3 OLD-MAMONSU"
    },
    "auth": "AUTH_TOKEN",
    "id": 1
}'

$ rename_zabbix_hosts.sh --url=http://localzabbix/
--pattern="local-pg" --suffix="OLD-MAMONSU"
```

Mamonsu testing

Build example: https://github.com/postgrespro/mamonsu/actions/runs/1303672795

MASTER

- OS: Ubuntu 20.04, CentOS 7, CentOS 8;
- Zabbix: 4.0, 5.0, 5.4;
- PostgreSQL: 9.6, 10, 11, 12, 13, 14;

DEV

- OS: CentOS 7;
- Zabbix: 5.4;
- PostgreSQL: 9.6, 10, 11, 12, 13, 14;


```
14  ------> mamonsu export config mamonsu.conf --add-plugins
15  Configuration file for mamonsu has been saved as mamonsu.conf
16  mamonsu.conf: ASCII text
17
18  ------> mamonsu export template template.xml --add-plugins
19  Template for mamonsu has been saved as template.xml
20  template.xml: XML 1.0 document, ASCII text, with very long lines
21
22  ------> mamonsu export zabbix-parameters zabbix.conf --add-
    plugins=/etc/mamonsu/plugins --config=/etc/mamonsu/agent.conf --pg-version=13
23  Configuration file for zabbix-agent has been saved as zabbix.conf
24  Bash scripts for native zabbix-agent have been saved to //scripts
25  zabbix.conf: ASCII text, with very long lines
```

PostgresPro

> Save the date: *Data Compression in PostgreSQL* - January 25, 2022, 18:00 CET.

Follow us on Eventbrite: https://www.eventbrite.co.uk/o/stacy-from-postgres-pro-30819802948

PostgresPro

Q&A

a.kuznetsova@postgrespro.ru

https://github.com/postgrespro/mamonsu

https://postgrespro.com/products/extensions/mamonsu