Setting up Zabbix Agent 2 for PostgreSQL monitoring

Daria Vilkova
Software Engineer (Postgres Professional)

Setting Up Zabbix Agent 2 for PostgreSQL Monitoring and Revealing How It Works

6:20 PM CET
Who am I?

- Software Engineer at Postgres Professional for 2+ years
- Co-author of PostgreSQL monitoring plugin for Zabbix Agent 2
- Co-maintainer of mamonsu, an open-source active monitoring agent for PostgreSQL and OS based on Zabbix
- MSc Degree in Applied Mathematics and Computer Science from Lomonosov Moscow State University
What will we discuss today?

- Plugin implementation and basic capabilities
- Connection levels
- How to get a simple metric?
- How to set up custom metrics?
- Q&A
How does Zabbix work?

- **Database server**
  - Zabbix Agent 2

- **Zabbix Server**
  - template (*.xml)

- **Data Visualization**
Implementation

- [github.com/jackc/pgx](https://github.com/jackc/pgx) - PG driver and toolkit for Go
- A handler for each metric or a group of metrics
- Some metrics are generated in JSON and grouped as dependency items and discovery rules.
Zabbix Agent 2: Basic Features

- Keeps permanent connection with PostgreSQL.
- Provides flexible polling intervals.
- Is compatible with PostgreSQL version 10+, and Zabbix Server version 4.4+.
- Is able to monitor several PostgreSQL instances by one Agent.
Levels of Connection Parameters

- Global
- Macros
- Sessions
Connection Parameters - Macros Level

- Fill in the template

$PG.URI$: tcp://localhost:5433

$PG.USER$: my_user

$PG.DATABASE$: my_database
# Connection Parameters - Macros Level

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th>PostgreSQL: Get connections</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Zabbix agent</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td><code>pgsql.connections&quot;{PG.URI}&quot; , &quot;{PG.USER}&quot; , &quot;{PG.PASSWORD}&quot;</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Type of information</strong></th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Update interval</strong></td>
<td>1m</td>
</tr>
</tbody>
</table>

### Custom intervals

<table>
<thead>
<tr>
<th>Type</th>
<th>Interval</th>
<th>Period</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexible</td>
<td>Scheduling</td>
<td>50s</td>
<td>1-7,00:00-24:00</td>
</tr>
</tbody>
</table>

**History storage period**: Do not keep history

**New application**: 

---

*PostgresPro*
Connection Parameters - Sessions Level

# Mandatory: no
# Default:

### Option: Plugins.Postgres.Sessions.*.User
# Username for session connection. "*" should be replaced with a session name.
# Mandatory: no
# Range: Must matches PostgreSQL user name.
# Default:

### Option: Plugins.Postgres.Sessions.*.Password
# Password for session connection. "*" should be replaced with a session name.
# Mandatory: no
# Range: Must matches the Password format.
# Default:
# Plugins.Postgres.Sessions.Test.Password=TestPassword

### Option: Plugins.Postgres.Sessions.*.Database
# Database for session connection. "*" should be replaced with a session name.
# Mandatory: no
# Default:
# Plugins.Postgres.Sessions.Test.Database=TestDatabase
## Connection Parameters - Sessions Level

<table>
<thead>
<tr>
<th>Macro</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>${PG.CONFLICTS.MAX.WARN}</td>
<td>0</td>
</tr>
<tr>
<td>${PGCONN_TOTAL_PCT.MAX.WARN}</td>
<td>90</td>
</tr>
<tr>
<td>${PG.DEADLOCKS.MAX.WARN}</td>
<td>0</td>
</tr>
<tr>
<td>${PG.LLD.FILTER.DBNAME}</td>
<td>(.+)</td>
</tr>
<tr>
<td>${PG.SESSION}</td>
<td>Test</td>
</tr>
</tbody>
</table>
Connection Parameters - Sessions Level

Preprocessing

* Name: Autovacuum: Count of autovacuum workers
Type: Zabbix agent
* Key: pgsql.autovacuum.count["${PG.SESSION}"]
Type of information: Numeric (float)
Units:
* Update interval: 1m

Custom intervals:
- Type: Flexible
  - Scheduling
  - Interval: 50s
  - Period: 1-7,00:00-24:00
  - Action: Remove

Add
Monitoring plugin in numbers

> 98 metrics
Metrics

- number of connections
- database size
- info about archive files
- number of “bloating” tables
- replication status
- background writer processes activity
- ...
## Metrics

<table>
<thead>
<tr>
<th>Host</th>
<th>Name</th>
<th>Interval</th>
<th>History</th>
</tr>
</thead>
<tbody>
<tr>
<td>go-agent-demo</td>
<td>CPU</td>
<td>(15 Items)</td>
<td></td>
</tr>
<tr>
<td>go-agent-demo</td>
<td>General</td>
<td>(5 Items)</td>
<td></td>
</tr>
<tr>
<td>go-agent-demo</td>
<td>Memory</td>
<td>(5 Items)</td>
<td></td>
</tr>
<tr>
<td>go-agent-demo</td>
<td>OS</td>
<td>(8 Items)</td>
<td></td>
</tr>
<tr>
<td>go-agent-demo</td>
<td>Performance</td>
<td>(15 Items)</td>
<td></td>
</tr>
<tr>
<td>go-agent-demo</td>
<td>PostgreSQL</td>
<td>(55 Items)</td>
<td></td>
</tr>
<tr>
<td>go-agent-demo</td>
<td>PostgreSQL: DB</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pgbench_1gb</td>
<td>(3 Items)</td>
<td></td>
</tr>
<tr>
<td>go-agent-demo</td>
<td>PostgreSQL: DB</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pgbench_3gb</td>
<td>(3 Items)</td>
<td></td>
</tr>
<tr>
<td>go-agent-demo</td>
<td>PostgreSQL: DB</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pgbench_10gb</td>
<td>(3 Items)</td>
<td></td>
</tr>
</tbody>
</table>
Example: How to Get a Simple Metric?

Create a file to get a new metric:

```
zabbix/src/go/plugins/postgres/handler_uptime.go
```

Import package and specify key for new metric:

```
package postgres

const (    keyPostgresUptime = "pgsql.uptime"
)
```
Example: How to Get a Simple Metric?

Create a handler with the following query:

```go
func uptimeHandler(ctx context.Context, conn PostgresClient, _ string, _ map[string]string, _ ...string) (interface{}, error) {

    var uptime float64

    query := `SELECT date_part('epoch', now() - pg_postmaster_start_time());`
```
Example: How to Get a Simple Metric?

Run the following query:

```go
row, err := conn.QueryRow(ctx, query)
if err != nil {
    ...
}
err = row.Scan(&uptime)
if err != nil {
    ...
}
return uptime, nil
```
Example: How to Get a Simple Metric?

Register the key of your new metric in metrics.go:

```go
var metrics = metric.MetricSet{
    ....,
    keyPostgresUptime: metric.New("Returns uptime.",
        []*metric.Param{paramURI, paramUsername, paramPassword, paramDatabase}, false),
}
```
Example: How to Get a Simple Metric?

Recompile the agent!
Coming soon: custom metrics feature
Custom Metrics

- Create a sql file with the query.
- In `zabbix_agent2.conf` specify the path to the directory with the sql files named `Plugins.Postgres.CustomQueriesPath`.
- In the template, provide the name for the sql file as the 5th parameter for the new key - `pgsql.query.custom` and specify the additional parameters for this query if needed.
Custom Metrics

- # CREATE table example (phrase text, year int );
- # SELECT * FROM example;

<table>
<thead>
<tr>
<th>phrase</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>new 2020 year is coming</td>
<td>2020</td>
</tr>
<tr>
<td>new 2021 year is coming</td>
<td>2021</td>
</tr>
<tr>
<td>new 2022 year is coming</td>
<td>2022</td>
</tr>
</tbody>
</table>

(3 rows)
Custom Metrics

- $touch custom2.sql
- $echo “SELECT * FROM example;” > custom2.sql

- $touch custom1.sql
- $echo “SELECT phrase FROM example WHERE year=$1;” > custom1.sql
Custom Metrics

- Add path to the sql files in `zabbix_agent2.conf`

Plugins.Postgres.CustomQueriesPath=/path/to/file
Custom Metrics

- `pgsql.query.custom - key`

* Name: Custom queries
* Type: Zabbix agent
* Key: `y"{"PG.URI"","PG.USER"","PG.PASSWORD"","PG.DATABASE"","custom2"}`
Type of information: Text
* Update interval: 30s

Custom intervals:
- Type: Flexible Scheduling
- Interval: 50s
- Period: 1-7,00:00-24:00
- Action: Remove
Add

* History storage period: Do not keep history, Storage period 90d
**Custom Metrics**

- `pgsql.query.custom - key`

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Custom queries with parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Zabbix agent</td>
</tr>
<tr>
<td>* Key</td>
<td><code>.URL&quot;,&quot;{PG.USER}&quot;&quot;,&quot;{PG.PASSWORD}&quot;&quot;,&quot;{PG.DATABASE}&quot;&quot;,&quot;custom1&quot;,&quot;2021&quot;]</code></td>
</tr>
<tr>
<td>Type of information</td>
<td>Text</td>
</tr>
<tr>
<td>* Update interval</td>
<td>30s</td>
</tr>
<tr>
<td>Custom intervals</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Interval</td>
</tr>
<tr>
<td>Flexible</td>
<td>Scheduling</td>
</tr>
<tr>
<td>Add</td>
<td></td>
</tr>
<tr>
<td>* History storage period</td>
<td>Do not keep history</td>
</tr>
</tbody>
</table>
```
## Custom Metrics

The result for each query will appear in text format.

<table>
<thead>
<tr>
<th>Custom queries with parameters</th>
<th>30s</th>
<th>90d</th>
<th>Zabbix agent</th>
<th>2020-11-29 20:53:52</th>
<th>&quot;new 2021 year is ...&quot;</th>
</tr>
</thead>
</table>

| Custom queries | 30s | 90d | Zabbix agent | 2020-11-29 20:53:53 | "new 2020 year is ..." |
Plugin Availability

The new version with custom metrics will become available with the next Zabbix Server release:

https://www.zabbix.com/download
Useful Links

- Zabbix Git:

- Official template of the plugin:
  https://www.zabbix.com/ru/integrations/postgresql#tab:official2

- PostgreSQL monitoring plugin article:
  https://postgrespro.com/blog/pgsql/5967895
Thank you for your attention!

info@postgrespro.com

postgrespro.com

d.vilkova@postgrespro.com
Q&A