

# Руководство пользователя: Средства анализа производительности и мониторинга детальных метрик активности

Центр разработки PostgreSQL

Exported on 05/29/2020

## Table of Contents

1	Список метрик (28.04.2020) .....	4
1.1	Автоматизированное развертывание и интеграция с порталом Динамической Инфраструктуры.....	7
1.2	Ролевая модель .....	9
1.3	Предварительная настройка PostgreSQL SberEditon .....	9
2	Руководство для сотрудника сопровождения.....	10
3	Руководство для администратора безопасности.....	11
4	Руководство для разработчика прикладных сервисов.....	12

PostgreSQL Sber Edition предоставляет средства для анализа производительности и мониторинга метрик своей активности.

## 1 Список метрик (28.04.2020)

Name	Query	Triggers	History
blocks hit per second	select sum(blks_hit) from pg_stat_database;		30d
blocks read per second	select sum(blks_read) from pg_stat_database;		30d
buffers allocated	select buffers_alloc from pg_stat_bgwriter;		30d
buffers written by the bgwriter	select buffers_clean from pg_stat_bgwriter;		30d
buffers written directly by a backend	select buffers_backend_fsync from pg_stat_bgwriter;		30d
buffers written during checkpoints	select buffers_checkpoint from pg_stat_bgwriter;		30d
cache hit ratio	select round(sum(blks_hit)*100/sum(blks_hit+blks_read), 2) from pg_stat_database;	PostgreSQL cache hit ratio too low on {HOSTNAME} ({ITEM.LASTVALUE})	30d
checkpoints by timeout	select checkpoints_timed from pg_stat_bgwriter;		30d
commits per second	select sum(xact_commit) from pg_stat_database;		30d
Current count workers vacuum	select count(*) from pg_catalog.pg_stat_activity where query like '%autovacuum%';	Vacuum worker limit > {\$VACUUM_WORKER_LIMIT}	30d
current max active transaction time	select coalesce(extract(epoch from max(age(now(), query_start))), 0) from pg_stat_activity where state <> 'idle in transaction' and state <> 'idle';	PostgreSQL active transaction too long on {HOSTNAME} (time={ITEM.LASTVALUE})	30d
current max idle transaction time	select coalesce(extract(epoch from max(age(now(), query_start))), 0) from pg_stat_activity where state='idle in transaction';	PostgreSQL idle transaction too long on {HOSTNAME} ({ITEM.LASTVALUE})	30d

Name	Query	Triggers	History
current max prepared transaction time	select coalesce(extract(epoch from max(age(now(), prepared))), 0) from pg_prepared_xacts;		30d
current max waiting transaction time	select coalesce(extract(epoch from max(age(now(), query_start))), 0) from pg_stat_activity where state='active' and wait_event IS NOT NULL;	PostgreSQL waiting transaction too long on {HOSTNAME} ({ITEM.LASTVALUE})	30d
fsync	select current_setting('fsync');		30d
full_page_writes	select current_setting('full_page_writes');		30d
HeartBeat PostgreSQL	select 1;	Cannot connect to DB "{\$PG_NAME_DB}"	30d
Max count workers vacuum	select setting from pg_settings where name='autovacuum_max_workers';	Vacuum worker limit > {\$VACUUM_WORKER_LIMIT}	30d
max written	select maxwritten_clean from pg_stat_bgwriter;		30d
number of active connections	select count(*) from pg_stat_activity where state = 'active';		30d
number of idle connections	select count(*) from pg_stat_activity where state = 'idle in transaction';		30d
number of idle in transaction connections	select count(*) from pg_stat_activity where state = 'idle in transaction';	PostgreSQL idle in transaction connections too high on {HOSTNAME} ({ITEM.LASTVALUE})	30d
number of prepared connections	select count(*) from pg_prepared_xacts;		30d
number of waiting connections	select count (*) from pg_stat_activity where state='active' and wait_event IS NOT NULL;	PostgreSQL number of waiting connections too high on {HOSTNAME} ({ITEM.LASTVALUE})	30d
pg_stat_statements: average query time	select round((sum(total_time) / sum(calls))::numeric,2) from pg_stat_statements;		30d

Name	Query	Triggers	History
ping json	explain (analyze, timing on, format json) select 1;		1d
ping json: ping	explain (analyze, timing on, format json) select 1;	PostgreSQL response too long on {HOSTNAME} ({ITEM.LASTVALUE})	30d
postgresql version	SELECT version() AS pg_version;		30d
recovery state	select pg_is_in_recovery();		30d
registered conflicts	select sum(conflicts) from pg_stat_database;	PostgreSQL recovery conflict occurred on {HOSTNAME}	30d
registered deadlocks	select sum(deadlocks) from pg_stat_database;	PostgreSQL deadlock occurred on {HOSTNAME}	30d
required checkpoints	select checkpoints_req from pg_stat_bgwriter;	PostgreSQL required checkpoints occurs too frequently on {HOSTNAME}	30d
rollbacks per second	select sum(xact_rollback) from pg_stat_database;		30d
service uptime	select date_part('epoch', now() - pg_postmaster_start_time())::int;	PostgreSQL service was restarted on {HOSTNAME} (uptime={ITEM.LASTVALUE})	30d
stand-by count	select count(*) from pg_stat_replication;		30d
sync time	select checkpoint_sync_time from pg_stat_bgwriter;		30d
temp_bytes written	select sum(temp_bytes) from pg_stat_database;		30d
temp_files created	select sum(temp_files) from pg_stat_database;		30d
times a backend had to execute its own fsync	select buffers_backend_fsync from pg_stat_bgwriter;		30d
total connections	select count(*)*100/(select current_setting('max_connections')::int) from pg_stat_activity;		30d

Name	Query	Triggers	History
total connections (%)	<code>select count(*)*100/(select current_setting('max_connections')::int) from pg_stat_activity;</code>	PostgreSQL total number of connections too high on {HOSTNAME} ({ITEM.LASTVALUE})	30d
transactions per second	<code>select sum(xact_commit)+sum(xact_rollback) from pg_stat_database;</code>		30d
tuples deleted per second	<code>select sum(tup_deleted) from pg_stat_database;</code>		30d
tuples fetched per second	<code>select sum(tup_fetched) from pg_stat_database;</code>		30d
tuples inserted per second	<code>select sum(tup_inserted) from pg_stat_database;</code>		30d
tuples returned per second	<code>select sum(tup_returned) from pg_stat_database;</code>		30d
tuples updated per second	<code>select sum(tup_updated) from pg_stat_database;</code>		30d
WAL segments count	<code>select count(*) from pg_ls_dir('pg_wal');</code>		30d
WAL write	<code>select pg_wal_lsn_diff(pg_current_wal_lsn(),'0/00000000');</code>		30d
write time	<code>select checkpoint_write_time from pg_stat_bgwriter;</code>		30d

**Примечание:** Средство мониторинга и анализа производительности обязательно подключается только для промышленной эксплуатации. ИТ|НТ|ПСИ – по требованию админов АС.

## 1.1 Автоматизированное развертывание и интеграция с порталом Динамической Инфраструктуры

Список параметров PostgreSQL SE необходимых для указания при развертывании:

- `shared_preload_libraries: 'pg_stat_statements';`
- ТУЗ мониторинга с именем: `zabbix_oasubd/<password>`

Для постановки сервера на мониторинг требуется вызвать API метод POST

(Alpha) [api.zabbix.ca.sbrf.ru<sup>1</sup>/api/monitoring/create\\_host](http://api.zabbix.ca.sbrf.ru/api/monitoring/create_host)  
(Sigma) [api.zabbix.sigma.sbrf.ru<sup>2</sup>/api/monitoring/create\\_host](http://api.zabbix.sigma.sbrf.ru/api/monitoring/create_host)


Со следующими параметрами:

Обязательные параметры			
env	строка	test или prom	Среда в которой работает хост
tag	строка	pg10	Тип приложения которое ставится на мониторинг. Например, это хост ОС то значение = 'os'
os	строка	linuxdi или windowsdi	ОС узла
diuid	строка		diuid хоста в ДИ. Добавляется в макросы узла сети как \${DIUID}
hostname	строка		Имя хоста
ip	строка		ip-адрес узла сети.
Необязательные параметры			
macro	массив	{'macro': ' ', 'value': ' '}	Макросы. Добавляется в макросы узла сетихоста как есть. каждый элемент массива должен быть словарем определенного в допустимых значениях вида.
port	строка		Порт узла сети. Необязательный параметр, если не указан используется 10050
postfix	строка	для кластерного решения PG_SE_Patroni_dbname  Для standalone: PG_SE_dbname	В случае узла сети для приложения используется для составления имени узла сети в заббиксе по принципу: <hostname>.<tag>.<postfix>. В случае узла сети ОС не используется и может быть не указан.
interfa cetypeid	число		

<sup>1</sup> <http://api.zabbix.ca.sbrf.ru/>

<sup>2</sup> <http://api.zabbix.sigma.sbrf.ru>



 **Внимание:** для работы с API должен быть получен предварительно токен авторизации.

Пример для Ansible:

```
- name: add host to monitoring
  uri:
    url: "api.zabbix.ca.sbrf.ru/api/monitoring/create_host3"
  method: POST
  body: '{"env": "prom", "tag": "pg10" , "os": "linuxdi", "diuid":
"rsefr31342","hostname": "tvli0514", "ip":"1.2.3.4" }'
  body_format: json
  headers:
    Accept: "application/json"
    Content-Type: "application/json"
    Authorization: "Bearer
yrtuiweyrtiuewyriuweyuirweuiyriweyiruyweiyuryweiryweiyriuweyriewrywueiryiewruyiwueyriwueyr"
```

## 1.2 Ролевая модель

Достаточный уровень привилегий для пользователя: NOSUPERUSER, NOCREATEDB, NOCREATEROLE, LOGIN, EXECUTE на исполнение функции pg\_catalog.pg\_ls\_dir(text).

## 1.3 Предварительная настройка PostgreSQL SberEditon

Должна присутствовать роль пользователя БД с именем **zabbix\_oasubd** с привилегиями уровня: NOSUPERUSER, NOCREATEDB, NOCREATEROLE, LOGIN, EXECUTE на исполнение функции pg\_catalog.pg\_ls\_dir(text).

---

<sup>3</sup> [http://api.zabbix.ca.sbrf.ru/api/monitoring/create\\_host](http://api.zabbix.ca.sbrf.ru/api/monitoring/create_host)

## 2 Руководство для сотрудника сопровождения

Реализованная функциональность не содержит специальных инструкций для сотрудника сопровождения.

### 3 Руководство для администратора безопасности

Реализованная функциональность не содержит специальных инструкций для администратора безопасности.

## 4 Руководство для разработчика прикладных сервисов

Реализованная функциональность не содержит специальных инструкций для разработчика прикладных сервисов.