

УТВЕРЖДЕН  
643.72410666.00067-07 98 01-ЛУ

**ЗАЩИЩЕННАЯ СИСТЕМА УПРАВЛЕНИЯ  
БАЗАМИ ДАННЫХ «JAТОВА»**

**Руководство по настройке. Часть 15.  
Балансировка подключений пользователей к  
СУБД.  
Компонент «jaPooler»**

**643.72410666.00067-07 98 01-15**

Листов 65

Инв. №подл.	Подп. и дата	Взам. инв. №	Инв. №дубл.	Подп. и дата

## АННОТАЦИЯ

В документе приведены сведения необходимые для установки и эксплуатации компонента балансировки подключений к СУБД «jaPooler» (далее по тексту – компонент либо jaPooler).

Настоящее руководство предназначено для администраторов СУБД «Jatoba».

Степени важности примечаний, применяемые в документе:



### Дополнительная информация

Указания, позволяющие упростить работу с изделием.



### Важная информация

Указания, требующие особого внимания.



Все примеры в данном документе приведены для СУБД «Jatoba» версии ядра 4.x, для других версий все шаги выполняются аналогично, разница состоит в именах директорий.

Например, СУБД «Jatoba» версии 5.x по умолчанию устанавливается в директорию:

- ОС Windows – «C:\Program Files\GIS\Jatoba\5\bin»;
- ОС Linux – «/usr/jatoba-5/bin».

Для СУБД «Jatoba» версии ядра 4 и 5 используется версия компонента — 2.1



### Важная информация

Для сертифицированной версии СУБД «Jatoba» поддерживается работа только на ОС, указанных в формуляре на поставку!

## СОДЕРЖАНИЕ

1. Назначение компонента jaPooler .....	7
1.1. Условия применения .....	8
2. Установка и настройка .....	9
2.1. Установка компонента «jaPooler» в ОС GNU/Linux .....	9
2.2. Установка расширения «jarooler» .....	10
2.3. Создание служебного пользователя «jarooler» .....	11
2.4. Файл паролей для служебного пользователя «jarooler» .....	12
2.5. Конфигурирование компонента jaPooler .....	13
2.6. Основные настройки Generic settings .....	13
2.6.1. Logfile – файл журнала .....	13
2.6.2. pidfile .....	14
2.6.3. listen_addr – список адресов .....	14
2.6.4. listen_port – номер принимающего порта .....	14
2.6.5. unix_socket_dir – расположение Unix сокетов .....	14
2.6.6. unix_socket_mode – режим файловой системы для сокета Unix .....	14
2.6.7. unix_socket_group – имя группы для сокета .....	15
2.6.8. User – пользователь Unix .....	15
2.6.9. pool_mode – режим работы .....	15
2.6.10. max_client_conn – максимальное количество клиентских подключений .....	15
2.6.11. default_pool_size – количество подключений в пуле .....	16
2.6.12. min_pool_size – минимальное количество дополнительных соединений .....	16
2.6.13. reserve_pool_size – количество дополнительно разрешенных подключений .....	16
2.6.14. reserve_pool_timeout – резервное время пула .....	16
2.6.15. max_db_connections – максимальное количество подключений .....	16
2.6.16. max_user_connections – максимальное количество подключений для пользователя .....	17
2.6.17. server_round_robin - циклический перебор серверов .....	17
2.6.18. ignore_startup_parameters – игнорирование стартовых параметров .....	17
2.6.19. disable_pqexec – отключение протокола PQexec .....	18
2.6.20. application_name_add_host – имя приложения в адресе хоста .....	18

2.6.21. conffile – расположение файла конфигурации .....	18
2.6.22. service_name – имя службы .....	18
2.6.23. job_name – название задачи .....	19
2.6.24. stats_period – периодичность вывода статистики.....	19
2.7. Аутентификационные параметры.....	19
2.7.1. auth_type – тип аутентификации.....	19
2.7.2. auth_hba_file – файл конфигурации НВА.....	21
2.7.3. auth_file – имя файла аутентификации .....	21
2.7.4. auth_user – имя пользователя для аутентификации в БД.....	21
2.7.5. auth_query – запрос извлечения пароля.....	21
2.8. Настройка соединения по SSL.....	21
2.8.1. Соединение с клиентом.....	21
2.8.2. Соединение с сервером .....	23
2.9. Параметры БД.....	24
2.9.1. dbname – имя целевой базы.....	25
2.9.2. host – адрес подключений к главному серверу .....	25
2.9.3. host_ro – адрес подключений к серверу горячего резерва.....	25
2.9.4. port – порт подключения .....	25
2.9.5. port_ro – порт подключения к серверу горячего резерва.....	26
2.9.6. strategy – стратегия балансировки запросов.....	26
2.9.7. user – имя пользователя.....	26
2.9.8. password – пароль пользователя .....	26
2.9.9. auth_user .....	26
2.9.10. pool_size – размер пулов для БД .....	26
2.9.11. min_pool_size – минимальный размер пулов для БД.....	26
2.9.12. reserve_pool – число дополнительных подключений к БД.....	27
2.9.13. connect_query – выполняемый запрос.....	27
2.9.14. pool_mode – режим пула для БД.....	27
2.9.15. max_db_connections – максимальное количество подключений для БД .....	27
2.9.16. client_encoding – кодировка клиента .....	27
2.9.17. datestyle – стиль даты .....	27

2.9.18. Timezone – временная зона .....	27
2.10. Раздел «пользователь» .....	27
2.10.1. pool_mode – режим работы пользователя .....	27
2.10.2. max_user_connections – максимальное количество подключений пользователя .....	28
2.11. Параметры регистрации событий .....	28
2.11.1. syslog – запись событий в журнал ОС Windows .....	28
2.11.2. syslog_ident – префикс имени событий .....	28
2.11.3. syslog_facility – префикс субъекта в имени событий .....	28
2.11.4. log_connections – регистрация успешных подключений .....	28
2.11.5. log_disconnections – регистрация отключений .....	28
2.11.6. log_pooler_errors – регистрация выводимых ошибок .....	28
2.11.7. log_stats – запись статистики .....	29
2.11.8. verbose – уровень детализации событий .....	29
2.12. Регистрация службы (демона) jaPooler .....	29
2.13. Функциональные возможности компонента «jaPooler» .....	29
2.13.1. Команды подключения .....	30
2.13.2. Административная консоль (Admin console) .....	31
2.13.3. Show commands – команды мониторинга .....	32
2.13.4. Команды управления процессами .....	44
2.13.5. Прочие команды .....	46
2.13.6. Signals .....	47
3. Примеры использования компонента .....	48
3.1. Кластер горячего резерва в режиме стратегии «balancing» .....	48
3.1.1. Установка главного сервера (Master-сервер) .....	49
3.1.2. Установка сервера горячего резерва (Slave-сервер) .....	50
3.1.3. Создание кластера .....	51
3.1.4. Инициализация компонента «jaPooler» .....	51
3.1.5. Подключение пользователя и выполнение запросов RW, RO .....	54
3.1.6. Результат .....	56
3.2. Кластер горячего резерва в режиме стратегии «always_rw» .....	57

3.2.1. Результат.....	60
3.3. Кластер горячего резерва в режиме стратегии «always_go».....	60
3.3.1. Результат.....	63
Перечень сокращений .....	64

## 1. НАЗНАЧЕНИЕ КОМПОНЕНТА jaPooler

Компонент jaPooler предназначен для управления соединениями, позволяет подключиться к СУБД большому числу клиентов без существенного снижения производительности. Между jaPooler и СУБД поддерживаются соединения, которые можно повторно использовать. После отключения клиента соединение возвращается в пул и может быть повторно использовано тем же самым или новым клиентом.

Балансировка в jaPooler представлена в функциях умного распределения RW (ReadWrite)/RO (ReadOnly) запросов.

Компонент имеет функциональную возможность распределять нагрузку при запросах от пользователя (ей) к серверу при их подключении к серверу СУБД на указанный порт, как представлено рисунке 1.1.

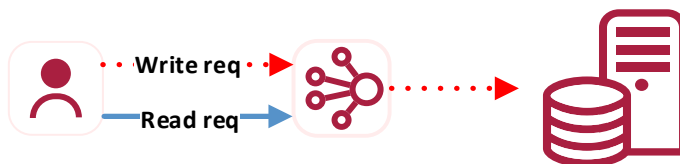


Рисунок 1.1 – Балансировка запросов пользователя

Также компонент обладает функциональной возможностью балансировки подключений множества пользователей к серверам СУБД, как представлено на рисунке 1.2.

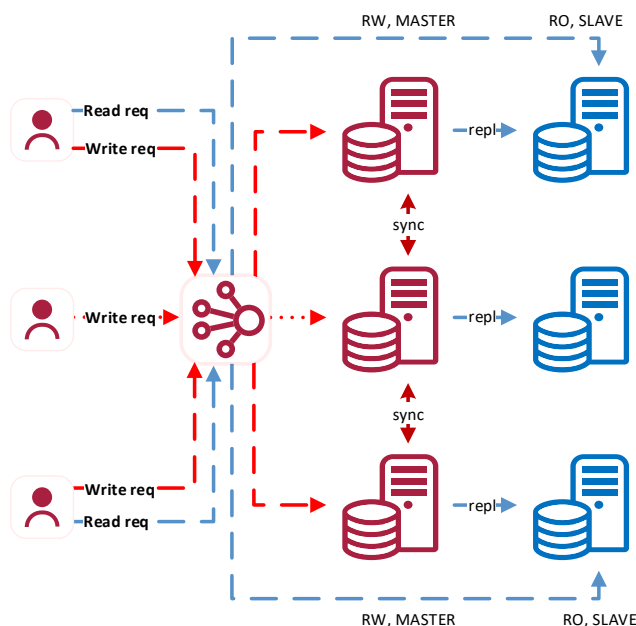


Рисунок 1.2 – Балансировка запросов пользователей к серверам СУБД

Подключения формируются как от пользователей, так и от пользовательских приложений.

Компонент работает по серверной технологии. На серверах СУБД устанавливается компонент работающий на уровне службы (демона) операционной системы. Передача данных осуществляется по протоколу Libpq по порту 6432.

### 1.1. Условия применения

Компонент «jaPooler» может использоваться совместно с СУБД «Jatoba» версии 4.x и выше в ОС, указанных в таблице 1.1.

Таблица 1.1 – Перечень поддерживаемых ОС

№	Наименование ОС
1	Astra Linux 1.7 Special Edition Смоленск (x86-64)
2	Astra Linux 1.8 (x86-64)
3	Astra Linux 2.12 Common Edition Орел (x86-64)
4	Debian 10
5	Debian 11
6	Debian 12
7	Альт 8 СП
8	Альт 9 Server
9	Альт 10 Server
10	Ubuntu 20.04
11	Ubuntu 22.04
12	Ubuntu 24.04
13	ОСНОВА2
14	РЕД ОС 7.3 Муром
15	РЕД ОС 8
16	РОСА 7.9
17	РОСА 12.4
18	RedHat Enterprise Linux 8
19	Oracle Linux 8.4

В текущей реализации не поддерживается управление с помощью пользовательского веб-интерфейса для администраторов «Jatoba Data Safe».

Управление осуществляется через корректировку управляющих параметров и конфигурационных файлов.



## 2. УСТАНОВКА И НАСТРОЙКА

Установка модуля должна производиться от имени пользователя, обладающего административными привилегиями в системе. Компонент штатным образом может быть установлен только с СУБД «Jatoba» (см. документ «Защищенная система управления базами данных «Jatoba». Руководство по установке).

### 2.1. Установка компонента «jaPooler» в ОС GNU/Linux

Установка компонента осуществляется в процессе установки СУБД «Jatoba», также компонент можно установить опционально после основной инсталляции СУБД.

Установку компонента возможно провести двумя способами:

- 1) установка из локального репозитория (CDROM) – производится из файлов, записанных на компакт-диск или скопированных с него;
- 2) установка непосредственно из deb/rpm-файлов – производится опционально, по усмотрению пользователя.

Компонент выполнен в виде отдельного deb или rpm-пакета. Установка компонента осуществляется средствами пакетного менеджера ОС. Для разных типов пакетных менеджеров команда установки немного отличается. Ниже приведены основные типы:

– для систем на основе пакетного менеджера APT (к таким системам относятся все ОС семейства Debian, использующие deb-пакеты) команда установки следующая:

```
apt-get install jatoba<ver>-japooler
```

– для систем на основе пакетных менеджеров YUM/DNF (к таким системам относятся все ОС семейства RedHat и вышедшие из нее, использующие rpm-пакеты) команда установки следующая:

```
yum install jatoba<ver>-japooler
```

Отдельного уточнения требуют операционные системы ALT Linux и openSUSE.

– ALT Linux использует пакетный менеджер APT, но распространяется в виде rpm-пакетов и для нее команда установки выглядит аналогично Debian:

```
apt-get install jatoba<ver>-japooler
```

– openSUSE также распространяется в виде rpm-пакетов, но использует собственный пакетный менеджер zypper, для нее команда установки выглядит следующим образом:

```
zypper install jatoba<ver>-japooler
```

Установка компонента в составе других версий СУБД «Jatoba» осуществляется аналогично. Отличие будет только в номере версии СУБД, в составе которой он распространяется. Например, jatoba4-japooler и т.п.



В процессе установки пакета компонента создается системный пользователь pgbouncer, от имени которого работает системная служба компонента. Также системный пользователь pgbouncer имеет права доступа к каталогам и файлам, используемым компонентом «jaPooler».

Удаление модуля также осуществляется средствами пакетного менеджера ОС. Вместо команды install нужно использовать соответствующую данному пакетному менеджеру команду удаления (remove, purge, erase и т.п.).

Для получения детальной информации по пакетному менеджеру рекомендуется обратиться к документации по ОС.

## 2.2. Установка расширения «japooler»

Расширение СУБД «japooler» служит, для:

- проведения балансировки подключений между узлами кластера;
- установки дополнительных функций, таких как «add\_japooler\_user» и «drop\_japooler\_user».

Расширение «japooler» устанавливается при помощи SQL-команды:

```
create extension japooler;
```



**ВАЖНО!** В случае совместной работы компонента «jaPooler» с SecurityProfile в одной СУБД, расширение «japooler» должно устанавливаться в ту же БД, в которую установлено расширение «securityprofile».

Справочная информация по настройке SecurityProfile приведена в документе «Руководство администратора СУБД Jatoba» 643.72410666.00067-07 95 01

```

root@shared-node: /usr/jatoba-5/bin
postgres=# create extension japooler;
CREATE EXTENSION
postgres=# \dx

```

Name	Version	Schema	Description
japooler	1.0	public	Adds japooler integration features
plpgsql	1.0	pg_catalog	PL/pgSQL procedural language

```

(2 rows)
postgres=#

```

Рисунок 2.1 – Установка расширения

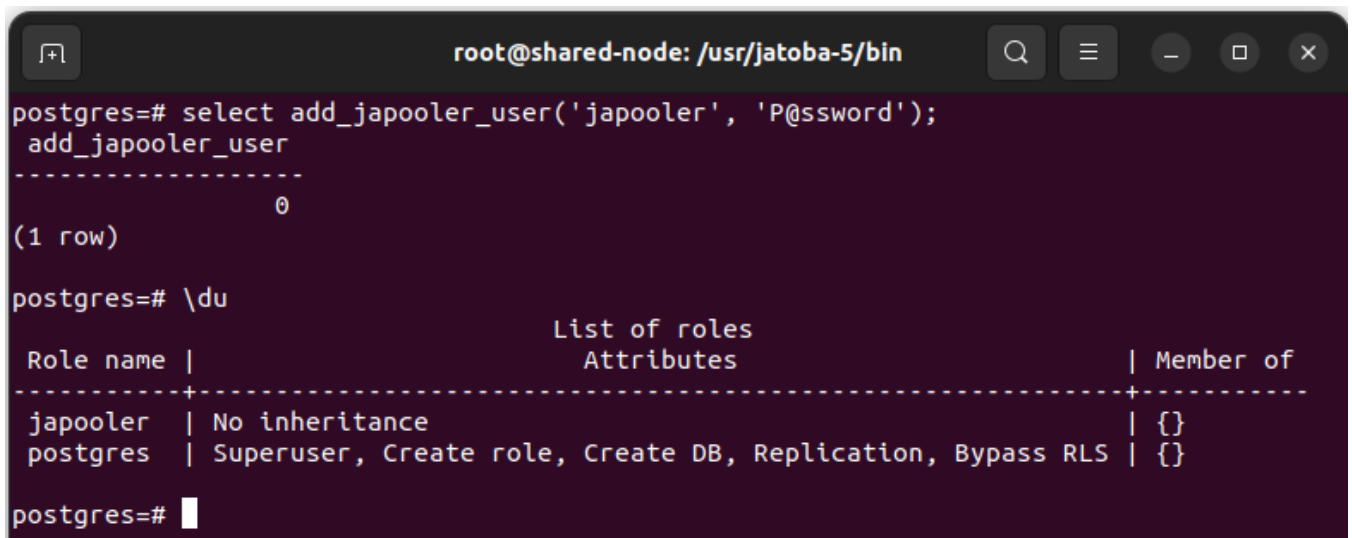
### 2.3. Создание служебного пользователя «japooler»

Служебный пользователь «japooler» создаётся от имени и с правами привилегированного пользователя. SQL-команда имеет синтаксис:

```
select add_japooler_user('japooler', '[password]');
```

В рассматриваемом примере служебный пользователь «japooler» создаётся SQL-командой:

```
select add_japooler_user('japooler', 'P@ssword');
```



```
root@shared-node: /usr/jatoba-5/bin
postgres=# select add_japooler_user('japooler', 'P@ssword');
add_japooler_user
-----
0
(1 row)

postgres=# \du
                                List of roles
Role name |                               Attributes                               | Member of
-----+-----+-----
japooler  | No inheritance                                     | {}
postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}

postgres=#
```

Рисунок 2.2 – Создание пользователя

#### 2.4. Файл паролей для служебного пользователя «japooler»

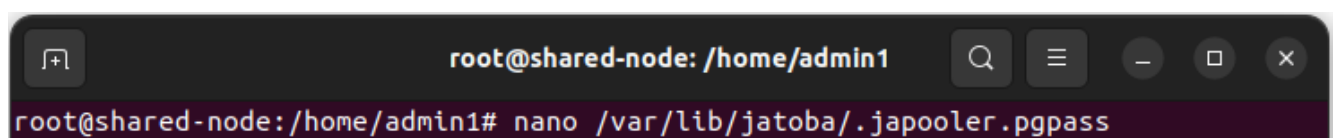
Установленный пароль для служебного пользователя требуется сохранить в файле паролей, который будет использоваться для подключений.

Файл паролей .japooler.pgpss создается по пути:

```
/var/lib/jatoba
```

командой в терминале ОС:

```
# nano /var/lib/jatoba/.japooler.pgpss
```



```
root@shared-node: /home/admin1
root@shared-node:/home/admin1# nano /var/lib/jatoba/.japooler.pgpss
```

Рисунок 2.3 – Команда создания файла паролей

В созданном файле добавить строку:

```
127.0.0.1:5432:postgres:japooler:P@ssword
```

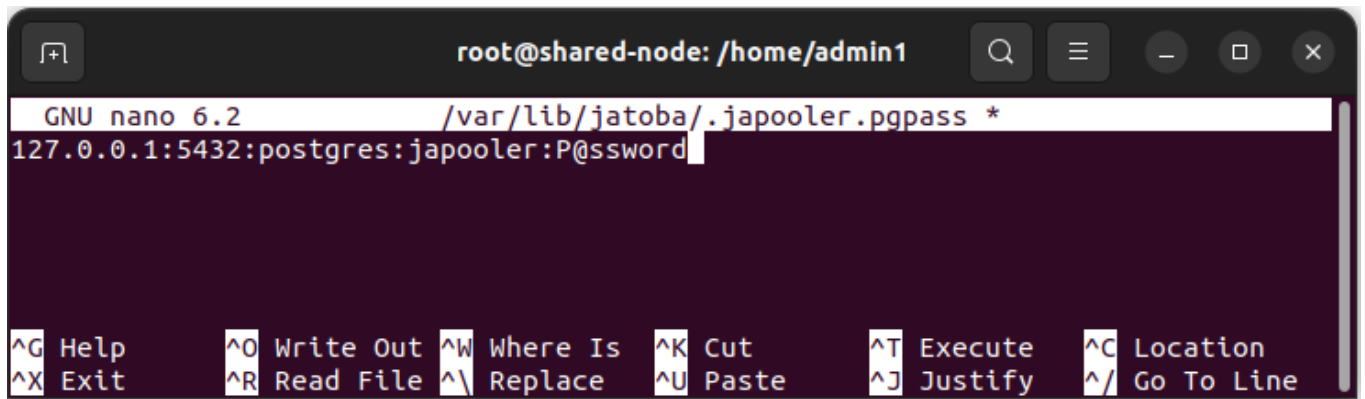


Рисунок 2.4 – Строка подключения в файле паролей

Сохранить внесённые изменения.

В GNU/Linux разрешения для файла паролей должны запрещать любой доступ к нему всем и группе. Для этого владельцем файла назначается пользователь ОС postgres и командой:

```
# chown postgres: /var/lib/jatoba/.japooler.pgpass
# chmod 600 /var/lib/jatoba/.japooler.pgpass
```

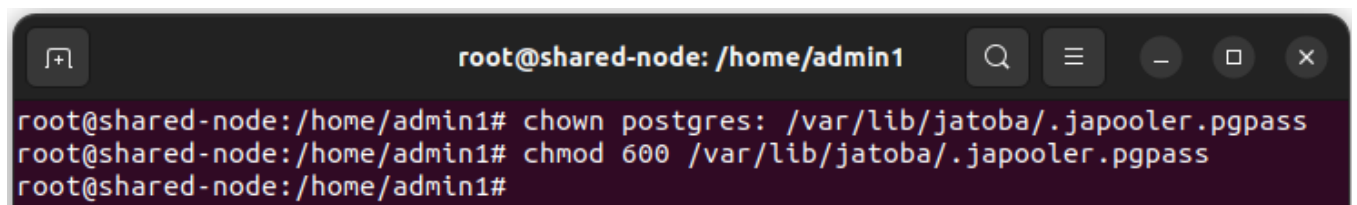


Рисунок 2.5 – Установка разрешений на файл паролей

## 2.5. Конфигурирование компонента jaPooler

Основные настройки компонента jaPooler проводятся в файле /usr/jatoba-<ver>/etc/pgbouncer.ini. Неактивные параметры маркируются символами «;» «#» в начале строки.

## 2.6. Основные настройки Generic settings

### 2.6.1. Logfile – файл журнала

Указывает имя файла журнала. Для работы в режиме демона (-d) должен быть задан либо этот параметр, либо syslog. Файл журнала остается открытым, так что после смены имени для прокрутки следует выполнить kill -HUP или «[RELOAD](#)» в консоли. В ОС Windows необходимо остановить и вновь запустить службу.

Обратите внимание, что параметр logfile сам по себе не отключает вывод сообщений в stderr. Отключить его можно, воспользовавшись параметром командной строки -q или -d.

Установить рекомендуемое значение:

```
/usr/jatoba-6/var/log/pgbouncer/pgbouncer.log
```

### 2.6.2. pidfile

Указывает имя файла PID. Без этого указания работа в режиме демона (-d) не допускается.

Установить рекомендуемое значение:

```
/usr/jatoba-6/var/run/pgbouncer/pgbouncer.pid
```

### 2.6.3. listen\_addr – список адресов

Указывает список адресов, по которым должны приниматься TCP-подключения. При указании \* будет означать «принимать по всем адресам». Когда этот параметр не задан, принимаются только подключения через Unix-сокеты.

Адреса могут задаваться числами (в формате IPv4/IPv6) или именами.

Значение по умолчанию: не установлено.

### 2.6.4. listen\_port – номер принимающего порта

Номер принимающего порта. Задается и для сокетов TCP, и для Unix-сокетов.

По умолчанию: 6432.

### 2.6.5. unix\_socket\_dir – расположение Unix сокетов

Указывает расположение Unix-сокетов. Действует и для принимающего сокета, и для подключений к серверу. Если задана пустая строка, сокеты Unix отключаются. При указании значения, начинающегося с @, будет создан сокет Unix в абстрактном пространстве имен (в настоящее время поддерживается в ОС Linux и ОС Windows).

Чтобы работала перезагрузка «на лету» (-R), необходимо настроить сокет Unix, который должен находиться в пространстве имен файловой системы.

По умолчанию: /tmp (в Windows — пустая строка).

### 2.6.6. unix\_socket\_mode – режим файловой системы для сокета Unix

Режим файловой системы для сокета Unix. Игнорируется для сокетов в абстрактном пространстве имен.

Не поддерживается в ОС Windows.

По умолчанию: 0777.

### 2.6.7. **unix\_socket\_group** – имя группы для сокета

Имя группы для сокета Unix. Игнорируется для сокетов в абстрактном пространстве имен.

Не поддерживается в ОС Windows.

Значение по умолчанию: не установлено.

### 2.6.8. **User** – пользователь Unix

Данный параметр определяет, на какого пользователя Unix нужно переключиться после запуска. Работает при условии, если компонент запускается от имени root или уже запущен от имени заданного пользователя.

Не поддерживается в ОС Windows.

Значение по умолчанию: не установлено.

### 2.6.9. **pool\_mode** – режим работы

Указывает, когда подключение к серверу могут повторно использовать другие клиенты. Режим работы может быть:

- **session** – серверное подключение возвращается в пул после отключения клиента. Это вариант по умолчанию;
- **transaction** – сервер возвращается в пул после завершения транзакции;
- **statement** – сервер возвращается в пул после завершения запроса. В этом режиме не допускаются транзакции, охватывающие несколько операторов.

### 2.6.10. **max\_client\_conn** – максимальное количество клиентских подключений

Максимально допустимое число клиентских подключений. При увеличении должно также увеличиваться ограничение на число файловых дескрипторов. Фактическое число занятых файловых дескрипторов будет больше чем `max_client_conn`. Если каждый пользователь подключается к серверу под своим именем, теоретически возможный максимум равен:

```
max_client_conn + (max_pool_size * total_databases * total_users)
```

Если пользователь задан в строке соединения (все пользователи подключаются под одним именем), теоретический максимум равен:

```
max_client_conn + (max_pool_size * total_databases)
```

Теоретический максимум не должен достигаться никогда, если только намеренно не предпринимаются специальные меры для этого. Тем не менее это значит, что число файловых дескрипторов должно ограничиваться довольно большим числом.

Значение по умолчанию: 100.

#### **2.6.11. default\_pool\_size – количество подключений в пуле**

Максимально возможное количество подключений к серверу для пары пользователь/база. Может быть переопределено в конфигурации базы данных.

Значение по умолчанию: 20.

#### **2.6.12. min\_pool\_size – минимальное количество дополнительных соединений**

Число серверных подключений для пары пользователь/база, которые доступны в пуле со старта сервера. Дает положительный эффект, когда нагрузка появляется внезапно после периода простоя. Это значение будет ограничено размером пула.

Значение по умолчанию: 0 (отключено).

#### **2.6.13. reserve\_pool\_size – количество дополнительно разрешенных подключений**

Число дополнительно разрешенных подключений в пуле (см. `reserve_pool_timeout`). При 0 резерв отсутствует.

Значение по умолчанию: 0 (отключено).

#### **2.6.14. reserve\_pool\_timeout – резервное время пула**

Если клиент не обслуживается заданное число секунд, `pgbouncer` задействует дополнительные подключения из резервного пула. При 0 это не происходит.

Значение по умолчанию: 5.0

#### **2.6.15. max\_db\_connections – максимальное количество подключений**

Не допускать больше заданного числа серверных подключений к базе данных, независимо от пользователя. При этом учитывается база данных компонента `jaPooler`, к которой подключен клиент, а не база данных СУБД «Jatoba» исходящего подключения. Данный предел можно установить для каждой базы данных в разделе `[databases]`.



Обратите внимание, при достижении предела, закрытие подключения клиента в одном пуле не позволяет немедленно установить другое подключение к



серверу через другой пул, так как подключение первого пула по-прежнему открыто. Когда сервер закроет его по тайм-ауту неактивности, новое подключение будет немедленно установлено для ожидающего пула.

Значение по умолчанию: 0 (нет ограничения).

#### **2.6.16. max\_user\_connections – максимальное количество подключений для пользователя**

Не допускать больше заданного числа подключений к серверу для пользователя (независимо от базы данных). При этом учитывается пользователь pgbouncer, связанный с пулом. В данной роли может выступать либо пользователь, указанный для подключения к серверу, либо, в отсутствие этого указания, пользователь, от имени которого подключился клиент. Этот предел можно установить для каждого пользователя в разделе [users].



Обратите внимание, при достижении предела, закрытие подключения клиента в одном пуле не позволяет немедленно установить другое подключение к серверу через другой пул, так как подключение первого пула по-прежнему открыто. Когда сервер закроет его по тайм-ауту неактивности, новое подключение будет немедленно установлено для ожидающего пула.

Значение по умолчанию: 0 (нет ограничения).

#### **2.6.17. server\_round\_robin - циклический перебор серверов**

По умолчанию компонент повторно использует подключения сервера в порядке LIFO (последнее пришло, первое ушло), так что основная загрузка распределяется по нескольким последним соединениям. Это дает наибольшую производительность, если базу данных обслуживает один сервер. Но если за IP-адресом базы данных скрывается балансировщик TCP-соединений, лучше, если компонент будет использовать подключения в данном режиме, обеспечивая таким образом равномерную нагрузку.

Значение по умолчанию: 0.

#### **2.6.18. ignore\_startup\_parameters – игнорирование стартовых параметров**

По умолчанию компонент принимает только параметры, которые он может отслеживать в стартовых пакетах:

- client\_encoding;
- datestyle;

- `timezone`;
- `standard_conforming_strings`.

Все другие параметры вызывают ошибку. Чтобы принимались и другие параметры, их нужно указать здесь, чтобы `pgbouncer` знал, что они обрабатываются администратором и их можно игнорировать.

Значение по умолчанию: пустая строка.

#### **2.6.19. `disable_pqexes` – отключение протокола PQexes**

Отключает протокол простых запросов (PQexes). В отличие от протокола расширенных запросов, этот протокол допускает указание нескольких запросов в одном пакете, что оставляет место для атак с SQL-инъекцией. Отключение этого протокола может улучшить безопасность. При этом означает, что смогут работать только клиенты, которые используют исключительно протокол расширенных запросов.

Значение по умолчанию: 0.

#### **2.6.20. `application_name_add_host` – имя приложения в адресе хоста**

Добавляет адрес компьютера и порт клиента к имени приложения, задаваемого при установлении подключения. Это помогает идентифицировать источник плохих запросов и т.п. Это выполняется, только когда подключение устанавливается. Если свойство `application_name` будет изменено позднее командой SET, компонент его уже не поменяет.

Значение по умолчанию: 0.

#### **2.6.21. `conffile` – расположение файла конфигурации**

Показывает расположение текущего файла конфигурации. При изменении этого параметра `pgbouncer` будет использовать другой файл конфигурации после команд «[RELOAD](#)» или «[SIGHUP](#)».

Значение по умолчанию: файл, заданный в командной строке.

#### **2.6.22. `service_name` – имя службы**

Используется при регистрации службы win32.



Параметр используется в ОС семейства Windows

Значение по умолчанию: `japooler`.

### 2.6.23. **job\_name** – название задачи

Аналог параметра `service_name` (см. п. 2.6.22).



Параметр используется в GNU Linux

### 2.6.24. **stats\_period** – периодичность вывода статистики

Определяет, с какой периодичностью (в секундах) будут пересчитываться средние значения, выводимые различными командами `SHOW`, и как часто агрегированная статистика будет записываться в журнал (см. `log_stats`).

Значение по умолчанию: 60.

## 2.7. Аутентификационные параметры

Компонент поддерживает аутентификацию пользователей по собственной базе по методам, указываемым в параметре «[auth\\_type](#)».

### 2.7.1. **auth\_type** – тип аутентификации

Устанавливаются следующие типы аутентификации:

— `cert`

Клиент должен подключаться по соединению TLS с действительным клиентским сертификатом. Имя пользователя берется из поля «`CommonName`» (Общее имя) сертификата.

— `md5`

Применять проверку пароля по хеш MD5. Этот метод аутентификации выбирается по умолчанию. Файл [auth\\_file](#) может содержать как зашифрованные MD5, так и открытые пароли. Даже при выборе `md5`, если пароль пользователя задан для метода SCRAM, автоматически будет применяться проверка по алгоритму SCRAM.

— `scram-sha-256`

Применять проверку пароля по алгоритму SCRAM-SHA-256. Заданный параметром [auth\\_file](#) файл должен содержать зашифрованные SCRAM или открытые пароли. Зашифрованные SCRAM пароли могут использоваться только для проверки пароли клиентов, но не для входа на сервер. Чтобы использовать SCRAM для серверных подключений, пароли необходимо задать открытым текстом.



В СУБД «Jatoba» версии 4 по умолчанию поддерживается шифрование пароля по алгоритму scram-sha-256

— plain

По каналу передается пароль в открытом тексте. Устаревший вариант.



Использования метода аутентификации «plain» не рекомендуется

— trust

Аутентификация не выполняется. Тем не менее имя пользователя должно присутствовать в auth\_file.



Использования метода аутентификации «trust» не рекомендуется

— any

Подобен методу «trust», но переданное имя пользователя игнорируется. Требуется, чтобы для всех баз данных было настроено подключение заданного пользователя. Кроме того, база данных консоли допускает подключение любого пользователя в качестве администратора.



Использования метода аутентификации «any» не рекомендуется

— hba

Фактический тип аутентификации загружается из [auth\\_hba\\_file](#). Это позволяет применять разные методы аутентификации для разных вариантов доступа.

Например: для подключений через сокет Unix применять метод peer, а для TCP — TLS.

— pam

Для проверки подлинности пользователей используется инфраструктура PAM (Pluggable Authentication Modules – подключаемые модули аутентификации). Файл [auth\\_file](#) игнорируется. Этот метод несовместим с базами данных, для которых используется auth\_user. PAM в файле конфигурации HBA не поддерживается.

### 2.7.2. **auth\_hba\_file** – файл конфигурации HBA

Файл конфигурации HBA, который используется, когда параметр `auth_type` равен `hba`.

Значение по умолчанию: не задано.

### 2.7.3. **auth\_file** – имя файла аутентификации

Имя файла, из которого будут загружаться имена и пароли пользователей.

Значение по умолчанию: не задано.

### 2.7.4. **auth\_user** – имя пользователя для аутентификации в БД

Если установлен параметр `auth_user`, то любой пользователь, не указанный в [auth\\_file](#), будет запрошен с помощью запроса `auth_query` из `pg_shadow` в базе данных с использованием `auth_user`. Пароль `auth_user` будет взят из `auth_file`. Если `auth_user` не требует пароля, то его не нужно определять в [auth\\_file](#).

Для прямого доступа к `pg_shadow` требуются права администратора.

Значение по умолчанию: не установлено.

### 2.7.5. **auth\_query** – запрос извлечения пароля

Запрос для извлечения пароля пользователя из базы данных.

Для прямого доступа к `pg_shadow` требуются права администратора. Поэтому рекомендуется, чтобы обычный пользователь обращался к ней, вызывая функцию `SECURITY DEFINER` (с контекстом безопасности определившего).

Заметьте, что этот запрос выполняется в целевой базе данных, так что, если в нем используются функции, они должны быть установлены в каждой базе.

По умолчанию:

```
SELECT username, passwd FROM pg_shadow WHERE username=$1
```

## 2.8. Настройка соединения по SSL

В случае необходимости настройки SSL соединения, необходимо произвести:

- создать серверный сертификат и ключ к нему (подключение к серверу БД);
- создать клиентский сертификат и ключ к нему (подключение к клиентскому ПО).

### 2.8.1. Соединение с клиентом

Настройка соединения с клиентом выполняется внесением параметров в конфигурационный файл `pgbouncer.ini`.

```
client_tls_sslmode = verify-full  
client_tls_ca_file = /usr/jatoba-6/etc/pgbouncer/root.crt  
client_tls_key_file = /usr/jatoba-6/etc/pgbouncer/pgbouncer.key  
client_tls_cert_file = /usr/jatoba-6/etc/pgbouncer/pgbouncer.crt  
client_tls_crl_file = /usr/jatoba-6/etc/pgbouncer/root.crl
```

### Параметр `client_tls_sslmode`

Параметр `client_tls_sslmode` должен иметь значение `verify-full`.

```
client_tls_sslmode = verify-full
```

При создании клиентского сертификата, значение поля CN (Common Name) в сертификате (как пример `pgbouncer.crt`) должно соответствовать доменному имени сервера или его IP адреса, на котором расположен продукт. В случае, если используется система виртуальных IP в системах «failover», таких как «keepalived», то необходимо прописать виртуальный IP адрес или виртуальный DNS сервер.

### Параметр `client_tls_ca_file`

Параметр `client_tls_ca_file` определяет полный путь до корневого сертификата. Все сертификаты можно хранить в текущей директории настроек:

```
client_tls_ca_file = /usr/jatoba-6/etc/pgbouncer/root.crt
```

### Параметр `client_tls_key_file`

Параметр `client_tls_key_file` определяет полный путь до файла ключа сертификата X-509.

```
client_tls_key_file = /usr/jatoba-6/etc/pgbouncer/pgbouncer.key
```

### Параметр `client_tls_cert_file`

Параметр `client_tls_cert_file` определяет полный путь до файла сертификата X-509.

```
client_tls_cert_file = /usr/jatoba-6/etc/pgbouncer/pgbouncer.crt
```

Так же необходимо в файле `userlist.txt` () указать имена пользователей, которые будут подключаться к базе данных, в качестве пароля – указать пустую строку:

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм.: _____
--------------------	--------------------------	---------------------------

```
"user1" ""  
"user2" ""
```

Полный путь до файла определяется в параметре `auth_file`, как описано в п. 2.7.3.

### 2.8.2. Соединение с сервером

Настройка соединения с сервером выполняется внесением параметров в конфигурационный файл `pgbouncer.ini`.

```
server_tls_sslmode = verify-full  
server_tls_ca_file = /usr/jatoba-6/etc/pgbouncer/root.crt  
server_tls_key_file = /usr/jatoba-6/etc/pgbouncer/reporter.key  
server_tls_cert_file = /usr/jatoba-6/etc/pgbouncer/reporter.crt  
server_tls_crl_file = /usr/jatoba-6/etc/pgbouncer/root.crl  
server_tls_ciphers = fast
```

Значение поля CN (Common Name) в клиентском сертификате (`user.crt`) должно соответствовать имени пользователя, от имени которого клиентское программное обеспечение подключается к СУБД «Jatoba».

#### Параметр `server_tls_sslmode`

Параметр `server_tls_sslmode` должен иметь значение `verify-full`.

```
server_tls_sslmode = verify-full
```

#### Параметр `server_tls_ca_file`

Параметр `server_tls_ca_file` полный путь до корневого сертификата. Все сертификаты можно хранить в текущей директории настроек:

```
server_tls_ca_file = /usr/jatoba-6/etc/pgbouncer/root.crt
```

#### Параметр `server_tls_key_file`

Параметр `server_tls_key_file` указывает полный путь до файла ключа сертификата X-509.

```
server_tls_key_file = /usr/jatoba-6/etc/pgbouncer/reporter.key
```

### Параметр `server_tls_cert_file`

Параметр `server_tls_cert_file` определяет полный путь до файла сертификата X-509.

```
server_tls_cert_file = /usr/jatoba-6/etc/pgbouncer/reporter.crt
```

### Параметр `server_tls_ciphers`

Параметр `server_tls_ciphers` определяет использование алгоритмов шифрования.

```
server_tls_ciphers = fast
```

Параметр может иметь значения:

- `default`;
- `secure`
- `fast`;
- `normal`.

## 2.9. Параметры БД

Этот раздел содержит пары ключ=значение, где в качестве ключа принимается имя базы данных, а в качестве значения — строка подключения для `libpq` в виде пар ключ=значение. Так как сама `libpq` не используется, в этой строке можно задать не все свойства, которые понимает `libpq` (`service=`, `.pgpass`).

Имя базы данных может содержать символы `_0-9A-Za-z` без кавычек. Имена, содержащие другие символы, должны заключаться в двойные кавычки по правилам для идентификаторов SQL (две кавычки (`""`) воспринимаются внутри строки как одна).

Имя базы данных `pgbouncer` зарезервировано для административной консоли и не может использоваться здесь в качестве ключа.

«\*» воспринимается как имя всех остальных баз: если точного соответствия имени для запрошенной базы данных не находится, в качестве строки подключения выбирается данное значение. Например, если есть следующая запись и нет других переопределяющих записей:



```
* = host=foo
```

В этом случае подключение к pgbouncer с указанием базы данных bar будет работать так, как если бы существовала следующая запись. В качестве dbname по умолчанию используется имя базы, полученное от клиента:

```
bar = host=foo dbname=bar
```

Такие автоматически создаваемые записи баз данных очищаются, если они простаивают больше времени, задаваемого параметром autodb\_idle\_timeout.

### 2.9.1. dbname – имя целевой базы

dbname

Имя целевой базы данных.

По умолчанию: имя базы данных, полученное от клиента.

### 2.9.2. host – адрес подключений к главному серверу

host

Имя или IP-адрес сервера, к которому нужно подключиться. Имена компьютеров разрешаются в момент подключения, и результат кешируется в течение времени, заданного параметром dns\_max\_ttl. Если результат разрешения имени меняется, существующие подключения к серверу автоматически закрываются при их освобождении в соответствии с режимом пула, и изменение немедленно отражается на новых подключениях. Если DNS возвращает несколько записей, они используются по очереди.

Если значение начинается с /, используется сокет Unix в пространстве имен файловой системы. Если значение начинается с @, используется сокет Unix в абстрактном пространстве имен.

По умолчанию: не задан, что подразумевает использование сокетов Unix.

### 2.9.3. host\_ro – адрес подключений к серверу горячего резерва

host\_ro

Имя или IP-адрес сервера горячего резерва.

### 2.9.4. port – порт подключения

По умолчанию: 5432.

### 2.9.5. port\_ro – порт подключения к серверу горячего резерва

По умолчанию: 5432.

### 2.9.6. strategy – стратегия балансировки запросов

strategy

Отвечает за стратегию балансировки запросов. Параметр может принимать 3 значения:

- always\_rw – все запросы всегда отправляются на главный сервер;
- always\_ro – все запросы отправляются на сервер горячего резерва;
- balancing – определяется тип запроса и в зависимости от него отправляется на нужный сервер, как представлено на рисунке 1.2. RW – Master (главный сервер), RO – Slave (сервер горячего резерва).

### 2.9.7. user – имя пользователя

Если задано user=, все подключения к целевой базе данных будут выполняться с заданным именем пользователя, что означает, что для этой базы данных будет всего один пул.

В противном случае pgbouncer подключается к целевой базе данных с именем пользователя, переданным клиентом, что означает, что для каждого пользователя будет отдельный пул.

### 2.9.8. password – пароль пользователя

Если пароль здесь не задается, будет использован пароль из [auth\\_file](#) или [auth\\_query](#).

### 2.9.9. auth\_user

Переопределяет глобальную переменную auth\_user, если она задана.

### 2.9.10. pool\_size – размер пулов для БД

Задаёт максимальный размер пулов для этой базы данных. Если не задано, применяется значение default\_pool\_size.

### 2.9.11. min\_pool\_size – минимальный размер пулов для БД

Задаёт минимальный размер пулов для этой базы данных. Если не задано, применяется глобальное значение min\_pool\_size.

**2.9.12. reserve\_pool – число дополнительных подключений к БД**

Задает число дополнительных подключений для этой базы данных. Если не задано, применяется значение reserve\_pool\_size.

**2.9.13. connect\_query – выполняемый запрос**

Запрос, который будет выполняться сразу после установления соединения, но до того, как его смогут использовать какие-либо клиенты. Если при запросе возникают ошибки, они только фиксируются в журнале, другой реакции не следует.

**2.9.14. pool\_mode – режим пула для БД**

Задает режим пула для данной базы данных. Если этот параметр не задается, применяется значение pool\_mode по умолчанию.

**2.9.15. max\_db\_connections – максимальное количество подключений для БД**

Задает максимум подключений для базы данных, то есть используя все пулы этой базы данных, нельзя будет установить больше этого числа подключений к серверу.

**2.9.16. client\_encoding – кодировка клиента**

Запрашивает у сервера использование указанной клиентской кодировки client\_encoding.

**2.9.17. datestyle – стиль даты**

Запрашивает у сервера использование указанного стиля даты datestyle.

**2.9.18. Timezone – временная зона**

Запрашивает у сервера использование указанного часового пояса timezone.

**2.10. Раздел «пользователь»**

Этот раздел содержит пары ключ=значение, где в качестве ключа принимается имя пользователя, а в качестве значения – переопределяемые для него параметры конфигурации в виде пар ключ=значение в формате строк подключения libpq. Таким образом переопределить можно лишь немногие параметры.

```
user1 = settings
user1 = pool_mode=session
```

**2.10.1. pool\_mode – режим работы пользователя**

Задает режим пула для всех подключений данного пользователя. Если этот параметр не задается, применяется значение [pool\\_mode](#) по умолчанию или заданное для базы данных.

### **2.10.2. max\_user\_connections – максимальное количество подключений пользователя**

Задаёт максимум подключений для пользователя, то есть, используя все пулы, нельзя будет установить больше этого числа подключений к серверу.

## **2.11. Параметры регистрации событий**

### **2.11.1. syslog – запись событий в журнал ОС Windows**

Включает/отключает запись в syslog. В ОС Windows вместо этого используется журнал событий.

Значение по умолчанию: 0.

### **2.11.2. syslog\_ident – префикс имени событий**

Имя, с которым события передаются в syslog.

Значение по умолчанию: pgbouncer (имя программы).

### **2.11.3. syslog\_facility – префикс субъекта в имени событий**

Субъект, который будет указываться в событиях, отправляемых в syslog.

Возможные варианты:

- auth;
- authpriv;
- daemon;
- user;
- local0 -7.

Значение по умолчанию: daemon.

### **2.11.4. log\_connections – регистрация успешных подключений**

Фиксировать в журнале успешные подключения.

Значение по умолчанию: 1.

### **2.11.5. log\_disconnections – регистрация отключений**

Фиксировать отключения с указаниями их причин.

Значение по умолчанию: 1.

### **2.11.6. log\_pooler\_errors – регистрация выводимых ошибок**

Фиксировать сообщения об ошибках, которые компонент передает клиентам.

Значение по умолчанию: 1.

### 2.11.7. log\_stats – запись статистики

Записывать агрегированную статистику в журнал, с периодичностью по параметру «[stats\\_period](#)». Это может быть лишним, если ту же статистику запрашивают внешние средства мониторинга, вызывая команды [SHOW](#).

Значение по умолчанию: 1.

### 2.11.8. verbose – уровень детализации событий

Увеличивает уровень детализации. Соответствует ключу `-v` в командной строке. Указание `-v -v` в командной строке равносильно записи `verbose=2` в конфигурации.

Значение по умолчанию: 0.

## 2.12. Регистрация службы (демона) jaPooler

Регистрация службы jaPooler возможно только после предварительной настройки конфигурационного файла «config.ini».

```
$ japoole -regservice config.ini
```

Для удаления службы используется команда:

```
$ japoole r -unregservice config.ini
```

Чтобы использовать журнал событий ОС Windows, установите `syslog = 1` в файле конфигурации. Но перед этим нужно зарегистрировать `pgbevent.dll`:

```
$ regsvr32 pgbevent.dll
```

Для отмены регистрации используется команда:

```
$ regsvr32 /u pgbevent.dll
```

## 2.13. Функциональные возможности компонента «jaPooler»

Компонент работает в трех основных режимах:

— Режим сеансов «Session pooling»

Метод используется по умолчанию и является самым мягким для пользователя. Установленное соединение сохраняется на протяжении всего времени работы и возвращается в пул после отключения клиента.

— Режим транзакций «Transaction pooling»

Соединение с сервером назначается клиенту только во время транзакции. Когда pgbouncer заметит, что транзакция завершена, соединение с сервером будет возвращено в пул.

— Режим операторов «Statement pooling»

Самый агрессивный метод. Соединение с сервером будет возвращено в пул сразу после завершения запроса. Транзакции с несколькими операторами в этом режиме запрещены, поскольку они могут быть нарушены.

### 2.13.1. Команды подключения

Переключатели командной строки:

```
-d, --daemon
```

Запуск в фоновом режиме, в режиме демона в ОС GNU Linux. Потребуется настройка pid-файла, а также файла журнала или системного журнала. Никакие сообщения журнала не будут записываться в stderr после перехода в фоновый режим.



Команда не выполняется в операционных системах семейства ОС Windows, так как в них компонент используется в режиме службы операционной системы

```
-R, --reboot
```

Выполнение перезагрузки в режиме онлайн. Это означает подключение к запущенному процессу, загрузку из него открытых сокетов и последующее их использование.



Команда поддерживается в ОС GNU Linux, если ОС поддерживает сокет Unix и unix\_socket\_dir не отключен в конфигурации

А также не работает с подключениями TLS, они отбрасываются.

```
-u USERNAME, --user=USERNAME
```

Подключение от имени и с правами пользователя.

```
-v, --verbose
```

Подробное протоколирование.

```
-q, --quiet
```

Тихий режим. Не влияет на детализацию ведения журнала

```
-V, --version
```

Вызов версии компонента.

```
-h, --help
```

Вызов справки по компоненту.

```
--regservice
```

Инсталляция сервиса.



Команда используется только для операционных систем семейства ОС Windows.

```
--unregservice
```

Деинсталляция сервиса.



Команда используется только для операционных систем семейства ОС Windows.

### 2.13.2. Административная консоль (Admin console)

Для подключения к административной консоли потребуется использовать команду:

```
$ psql -p 6432 pgbouncer
```

Только пользователи, указанные в параметрах конфигурации `admin_users` или `stats_users`, могут входить в консоль. За исключением случая `auth_type=any`, тогда любой пользователь может войти в качестве `stats_user`.

Кроме того, пользователю с именем `pgbouncer` разрешается входить в систему без пароля, если вход осуществляется через сокет Unix, а клиент имеет тот же UID пользователя Unix, что и запущенный процесс.

Консоль администратора в настоящее время поддерживает только простой протокол запросов.

### 2.13.3. Show commands – команды мониторинга

Команда SHOW предназначена для вывода информации. Каждая команда описана ниже.

#### 2.13.3.1. Статистические команды (SHOW STATS)

Показывает статистику. В этой и связанных с ней командах общие показатели указаны с момента запуска процесса, средние значения обновляются каждый stats\_period.

```
database
```

Представление статистика по каждой БД.

```
total_xact_count
```

Общее количество SQL-транзакций, объединенных компонентом.

```
total_query_count
```

Общее количество SQL-запросов, объединенных компонентом.

```
total_received
```

Общий объем в байтах сетевого трафика, полученного компонентом.

```
total_sent
```

Общий объем в байтах сетевого трафика, отправленного компонентом.

```
total_xact_time
```

Общее количество микросекунд, затраченных компонентом при подключении к СУБД в транзакции, либо в режиме ожидания в транзакции, либо при выполнении запросов.

```
total_query_time
```

Общее количество микросекунд, затраченных компонентом при активном подключении к СУБД на выполнение запросов.

```
total_wait_time
```

Время, затраченное клиентами на ожидание сервера, в микросекундах.



avg\_xact\_count

Среднее количество транзакций в секунду за последний период статистики.

avg\_query\_count

Среднее количество запросов в секунду за последний период статистики.

avg\_recv

Среднее количество полученных (от клиентов) байт в секунду.

avg\_sent

Среднее количество отправленных (клиентам) байт в секунду.

avg\_xact\_time

Средняя продолжительность транзакции в микросекундах.

avg\_query\_time

Средняя продолжительность запроса в микросекундах.

avg\_wait\_time

Время, затраченное клиентами на ожидание сервера, в микросекундах (в среднем в секунду).

#### **2.13.3.2. SHOW STATS\_TOTALS**

Команда показывает общие значения.

#### **2.13.3.3. SHOW STATS\_AVERAGES**

Команда показывает средние значения.

#### **2.13.3.4. SHOW TOTALS**

Команда показывает статистику по всем базам данных.

#### **2.13.3.5. SHOW SERVERS - просмотр состояния сервера**

Команда показывает состояние сервера со следующими параметрами:

type

«S», для сервера.

user

Имя пользователя, используемое для подключения к серверу.

database

База данных используемая для подключения.

state

Показ состояния соединения с сервером, которое может иметь значения:

- **Active** – активное;
- **Idle** – бездействующее;
- **Used** – используемое;
- **Tested** – проверяемое;
- **New** – новое.

addr

IP-адрес сервера СУБД.

port

Порт подключения к серверу СУБД.

local\_addr

Начальный адрес подключения на локальной машине.

local\_port

Порт начала подключения на локальной машине.

connect\_time

Время начала подключения.

request\_time

Время, когда был отправлен запрос.

wait

Текущее время ожидания в секундах.

wait\_us

Микросекундная часть текущего времени ожидания.

close\_needed

Немедленное закрытие соединения, т.к. требуется перезагрузка конфигурационного файла или требуется обновление DNS или был выполнена команда **RECONNECT**

ptr

Адрес внутреннего объекта для этого соединения. Используется как уникальный идентификатор.

link

Адрес клиентского подключения, с которым связан сервер.

remote\_pid

PID серверного процесса. В случае, если соединение осуществляется через сокет Unix, а ОС поддерживает получение информации об идентификаторе процесса, его PID ОС. В противном случае он извлекается из пакета отмены, отправленного сервером, который должен быть PID, если сервер является СУБД, но это случайное число, если сервер является другим компонентом.

tls

Строка с информацией о соединении TLS или пустая, если TLS не используется.

### 2.13.3.6. SHOW CLIENTS - просмотр состояние клиента

Команда показывает состояние клиента с следующими параметрами:

type

Тип клиента (пользователя).

user

Имя подключенного пользователя.

database

Имя базы данных.

state

Состояние клиентского соединения, которое может иметь значение:

- **active** – активное;
- **waiting** – ожидание.

addr

IP-адрес клиента.

port

Исходящий порт клиента.

local\_addr

Конечный адрес соединения на локальной машине.

local\_port

Конечный порт подключения на локальной машине.

connect\_time

Отметка времени подключения.

request\_time

Метка времени последнего запроса клиента.

wait

Текущее время ожидания в секундах.

wait\_us

Микросекундная часть текущего времени ожидания.

ptr

Адрес внутреннего объекта для этого соединения. Используется как уникальный идентификатор.

link

Адрес подключения к серверу, с которым связан клиент.

remote\_pid

Идентификатор процесса, если клиент подключается через сокет Unix и ОС поддерживает его получение.

tls

Строка с информацией о соединении TLS или пустая, если TLS не используется.

#### **2.13.3.7. SHOW POOLS - просмотр пула**

Для каждой пары (база данных, пользователь) создается новая запись в пуле.

database

Имя базы данных.

user

Имя пользователя.

cl\_active

Клиентские соединения, которые связаны с соединением с сервером и могут обрабатывать запросы.

cl\_waiting

Клиентские соединения, которые отправили запросы, но еще не получили соединение с сервером.

cl\_cancel\_req

Клиентские соединения, которые еще не передавали отмены запросов на сервер.

sv\_active

Соединения с сервером, которые связаны с клиентом.

sv\_idle

Соединения с сервером, которые не используются и могут быть немедленно использованы для клиентских запросов.

sv\_used

Подключения к серверу, которые простаивали более server\_check\_delay, поэтому их необходимо server\_check\_query запустить, прежде чем их можно будет использовать снова.

sv\_tested

Соединения с сервером, которые в настоящее время работают либо, server\_reset\_query либо server\_check\_query.

sv\_login

Соединения с сервером в настоящее время находятся в процессе входа в систему.

maxwait

Время ожидания первого (самого старого) клиента в очереди в секундах. Если это число начнет увеличиваться, значит, текущий пул серверов недостаточно быстро обрабатывает запросы. Причиной может быть либо перегруженный сервер, либо слишком маленькое значение параметра **pool\_size**.

maxwait\_us

Микросекундная часть максимального времени ожидания.

pool\_mode

Используемый режим объединения.

### 2.13.3.8. Просмотр внутренней информации SHOW LISTS

Показать следующую внутреннюю информацию в столбцах (не в строках)

databases

Количество баз данных.

users

Количество пользователей.

pools

Количество пулов.

free\_clients

Количество свободных клиентов.

used\_clients

Количество активных клиентов.

login\_clients

Количество клиентов в состоянии **входа** в систему.

free\_servers

Количество свободных серверов.

used\_servers

Количество используемых серверов.

dns\_names

Количество DNS-имен в кеше.

dns\_zones

Количество зон DNS в кеше.

dns\_queries

Количество незавершенных DNS-запросов.

### **2.13.3.9. SHOW USERS – просмотр пользователей**

Команда используется для просмотра состояния пользователей с параметрами:

№ изменения: \_\_\_\_\_

Подпись отв. лица: \_\_\_\_\_

Дата внесения изм.: \_\_\_\_\_

name

Имя пользователя.

pool\_mode

Режим переопределения пользователя pool\_mode или NULL, если вместо него будет использоваться значение по умолчанию.

### **2.13.3.10. SHOW DATABASES - просмотр БД**

Команда используется для просмотра состояния БД с параметрами:

name

Имя сконфигурированной БД.

host

Хост, к которому подключается компонент.

port

Порт, к которому подключается компонент.

database

Актуальное имя БД, к которой подключен компонент.

force\_user

Когда пользователь является частью строки подключения, соединение между компонентом и СУБД принудительно устанавливается для данного пользователя, каким бы ни был клиентский пользователь.

pool\_size

Максимальное количество подключений к серверу.

min\_pool\_size

Минимальное количество подключений к серверу.

reserve\_pool



Максимальное количество дополнительных соединений для этой базы данных.

`pool_mode`

Максимальное количество дополнительных соединений для этой базы данных.

`max_connections`

Максимальное количество разрешенных соединений для этой базы данных, заданное параметром **max\_db\_connections**, либо глобально, либо для каждой базы данных.

`current_connections`

Текущее количество соединений для этой базы данных.

`paused`

1, если эта база данных в настоящее время приостановлена, иначе 0.

`disabled`

1, если эта база данных в настоящее время отключена, иначе 0.

### 2.13.3.11. SHOW FDS – просмотр файловых дескрипторов

Внутренняя команда — показывает список используемых файловых дескрипторов с прикрепленным к ним внутренним состоянием.

Когда подключенный пользователь имеет имя пользователя «pgbouncer», подключается через сокет Unix и имеет тот же UID, что и запущенный процесс, фактические FD передаются по соединению. Этот механизм используется для онлайн-перезагрузки.



Команда не выполняется в операционных системах семейства ОС Windows

Эта команда также блокирует внутренний цикл обработки событий, поэтому ее нельзя использовать, пока используется компонент.

`fd`

Числовое значение дескриптора файла.

task

Один из пулеров, клиент или сервер.

user

Пользователь соединения с использованием FD.

database

База данных соединения с использованием FD.

addr

IP-адрес соединения с использованием FD, unix, если используется сокет Unix.

port

Порт, используемый соединением с помощью FD.

cancel

Отмена соединения.

link

FD для соответствующего сервера/клиента. NULL, если бездействует.

#### **2.13.3.12. SHOW SOCKETS, SHOW ACTIVE\_SOCKETS**

Показывает низкоуровневую информацию о сокетах или только об активных сокетах. Сюда входит информация, отображаемая в разделах «[SHOW CLIENTS](#)» и «[SHOW SERVERS](#)», а также другая более низкоуровневая информация.

#### **2.13.3.13. SHOW CONFIG – просмотр параметров конфигурации**

Показать текущие параметры конфигурации, по одному в строке, со следующими столбцами:

key

Имя переменной конфигурации.

value

Значение конфигурации.

default

Значение конфигурации по умолчанию.

changeable

Либо «YES», либо «NO», показывает, возможность изменения переменной во время работы. Если «NO», переменная может быть изменена только во время загрузки. Для изменения переменной во время выполнения необходимо использовать SET.

#### **2.13.3.14. SHOW MEM – просмотр памяти**

Показывает низкоуровневую информацию о текущих размерах различных выделений внутренней памяти. Представленная информация может быть изменена.

#### **2.13.3.15. SHOW DNS\_HOSTS – просмотр имен хостов**

Команда используется для просмотра имен хостов в кеше DNS.

hostname

Имя хоста.

ttd

Количество секунд до следующего поиска.

addrs

Список адресов, разделенных запятыми.

#### **2.13.3.16. SHOW DNS\_ZONES – просмотр зон DNS**

Команда используется для просмотра зон DNS, находящихся в памяти.

zonename

Имя зоны DNS.

serial

Актуальный порядковый номер.

count

Имена хостов, принадлежащих этой зоне.

### 2.13.3.17. SHOW VERSION – просмотр версии компонента

SHOW VERSION

Команда покажет версию компонента.

### 2.13.4. Команды управления процессами

#### 2.13.4.1. PAUSE [db] - Пауза

PAUSE [db]

При выполнении команды компонент попытается отключиться от всех серверов, сначала ожидая завершения всех запросов. Команда не будет закончена, пока все запросы не будут завершены. Команда используется во время перезапуска базы данных.

Если указано имя базы данных, будет приостановлена только эта база данных.

Новые клиентские подключения, к приостановленной БД, не будут производиться пока не будет выполнена команда «[RESUME](#)».

#### 2.13.4.2. DISABLE db – отключение подключений к БД

DISABLE db

При использовании команды все новые подключения к БД будут отклоняться.

#### 2.13.4.3. ENABLE db – разрешение подключений к БД

ENABLE db

Команда разрешает новые клиентские подключения к БД после предыдущей команды «[DISABLE](#)».

#### 2.13.4.4. RECONNECT [db] – переподключение к БД

RECONNECT [db]

При выполнении команды закроются все активные соединения к БД или всем БД. Новые подключения будут созданы немедленно, и они будут подключаться по мере необходимости в соответствии с настройками размера пула.

Эта команда полезна, когда настройки подключения к серверу изменились, например, для постепенного переключения на новый сервер.

Нет необходимости запускать эту команду, когда строка подключения в pgbouncer.ini была изменена и перезагружена (см. RELOAD) или когда изменилось разрешение DNS, потому что тогда эквивалент этой команды будет запущен автоматически.

Эта команда необходима только в том случае, если что-то ниже уровнем компонента маршрутизирует соединения. После запуска команды может наступить длительный период, когда некоторые соединения с сервером переходят к старому назначению, а некоторые соединения с сервером переходят к новому назначению.

Применение команды целесообразно при переключении трафика только для чтения между репликами только для чтения или при переключении между узлами настройки репликации с несколькими мастерами. Если все соединения необходимо переключить одновременно, вместо этого рекомендуется использовать команду «[PAUSE](#)». Чтобы закрыть соединения с сервером без ожидания (например, при аварийном переходе на другой ресурс, а не в сценариях постепенного переключения), рекомендуется использовать команду «[KILL](#)».

#### **2.13.4.5. KILL db – сброс клиентских подключений**

KILL db

Команда немедленно сбрасывает все клиентские подключения. Все новые подключения будут находиться в режиме ожидания до выполнения команды «[RESUME](#)».

#### **2.13.4.6. SUSPEND – приостановка**

SUSPEND

При выполнении команды все буферы сокетов очищаются, и компонент перестает прослушивать данные в них. Команда будет повторяться до тех пор, пока все буферы не будут пусты.

Новые клиентские подключения будут находиться в режиме ожидания до вызова команды «[RESUME](#)».

### 2.13.4.7. RESUME [db] – возобновление работы с БД

RESUME [db]

Команда возобновляет работу с БД после команд «[KILL](#)», «[PAUSE](#)» или «[SUSPEND](#)».

### 2.13.4.8. SHUTDOWN – закрытие процесса

SHUTDOWN

Команда закрывает процесс компонента в ОС.

### 2.13.4.9. RELOAD – перезапуск настроек компонента

RELOAD

Команда перезапускает процесс компонента, перезагружает конфигурационные файлы и обновляет свои настройки. Перезагружаются настройки из основного конфигурационного файла и из файлов «[auth file](#)» и «[auth hba file](#)».

Текущие подключения будут закрыты и восстановлены с новыми параметрами.

### 2.13.4.10. WAIT\_CLOSE [db] – ожидание закрытие подключения

WAIT\_CLOSE [db]

Команда заставляет компонент ожидать, когда все подключения к БД или всем БД перейдут в состояние “close\_needed”.

Команда может применяться после выполнения команд «[RECONNECT](#)» или «[RELOAD](#)» для полного применения внесенных изменений в конфигурацию компонента.

## 2.13.5. Прочие команды

### 2.13.5.1. SET key = arg – изменение параметра конфигурации в административной консоли

Команда в административной консоли изменяет параметры конфигурационного файла.

Например:

```
SET log_connections = 1;
SET server_check_query = 'select 2';
```

### 2.13.6. Signals

SIGHUP

Перезагрузить конфигурацию. Аналогично вводу команды **RELOAD** на консоли.

SIGINT

Безопасное отключение. Аналогично, что и выдача **PAUSE** и **SHUTDOWN** на консоли.

SIGTERM

Немедленное отключение. Аналогично, что и выдача **SHUTDOWN** на консоли.

SIGUSR1

Аналогично, что и команда **PAUSE** на консоли.

SIGUSR2

Аналогично выдаче **RESUME** на консоли.

### 3. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ КОМПОНЕНТА

В качестве примера использования компонента, будут использованы 2 сервера под управлением Ubuntu 18.04, параметры которых приведены в таблице 3.1.

Таблица 3.1 – Параметры серверов стенда

Наименование	Назначение	IP- адрес	Имя
Master-сервер	Главный сервер	192.168.23.133/24	j4@j4-ubuntu-1
Slave-сервер	Сервер горячего резерва	192.168.23.134/24	j4@j4-ubuntu-2

Компонент устанавливается на Master-сервер, с него же будет осуществляться подключение пользователя к кластеру.

В примере п. 3.1 «Кластер горячего резерва в режиме стратегии «balancing» рассматриваются подготовительные действия для использования компонента. Последующие примеры строятся на основе сформированного стенда и для демонстрации режимов стратегии работы компонента будут меняться настройки компонента.

#### 3.1. Кластер горячего резерва в режиме стратегии «balancing»

Функционирование компонента jaPooler при установленной стратегии

```
strategy = balancing
```

в конфигурационном файле `rgbouncer.ini`, определяется как направление запросов типа:

- RW – в адрес Master-сервера (IP 192.168.23.133);
- RO – в адрес Slave-сервера (IP 192.168.23.134).

Компонент слушает входящий трафик запросов, разбирает его и перенаправляет на сервера. Данный режим является основным в практическом применении.

Принципиальная схема работы компонента при установленной стратегии «balancing» представлена на рисунке 3.1



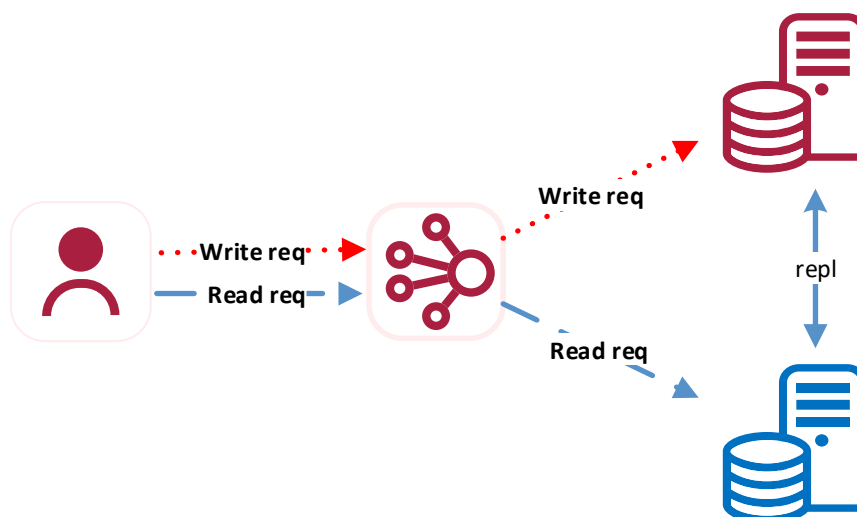


Рисунок 3.1 – Схема работы при установленной стратегии «balancing»

### 3.1.1. Установка главного сервера (Master-сервер)

Установка осуществляется от имени и с правами привилегированного пользователя ОС при помощи команды:

```
sudo apt install jatoba6-server jatoba6-client jatoba6-japooler
jatoba6-libs jatoba6-contrib
```

Последующие действия приведены в документе 643.72410666.00067-07 97 01 «Руководство по установке».

#### 3.1.1.1. Параметры конфигурационного файла postgresql.conf

В конфигурационном файле «postgresql.conf» Master-сервера требуется установить приведенные ниже параметры в соответствующих разделах.

```
#-----
# CONNECTIONS AND AUTHENTICATION
#-----
# - Connection Settings -
listen_addresses = '*'
#-----
# WRITE-AHEAD LOG
#-----
# - Settings -
wal_level = replica
#-----

# REPORTING AND LOGGING
#-----
```

```
# - Where to Log -
log_destination = 'stderr'
logging_collector = on
# - Authentication -
password_encryption = md5
```

После изменения метода шифрования паролей в postgresql.conf, необходимо переназначить пароль для пользователя, от имени которого будет происходить подключение к СУБД по порту jaPooler. Это необходимо сделать из-за того, что по умолчанию для шифрования паролей СУБД использует алгоритм SCRAM-SHA-256, а после его смены на md5 аутентификация пользователей со старыми паролями будет происходить с ошибкой.

### 3.1.1.2. Параметры конфигурационного файла pg\_hba

В конфигурационном файле «pg\_hba» Master-сервера требуется установить приведенные ниже параметры.

TYPE	DATABASE	USER	ADDRESS	METHOD
# Allow replication connections from localhost, by a user with the				
host	all	all	192.168.23.0/24	md5
host	replication	all	192.168.23.0/24	md5

### 3.1.2. Установка сервера горячего резерва (Slave-сервер)

Установка осуществляется от имени и с правами привилегированного пользователя ОС при помощи команды:

```
sudo apt install jatoba6-server jatoba6-client jatoba6-japooler
jatoba6-libs jatoba6-contrib
```



Для сервера горячего резерва установка компонента jaPooler не требуется

Последующие действия приведены в документе 643.72410666.00067-07 97 01 «Руководство по установке» до этапа инициализации БД.



Инициализация БД не требуется

Если случайно была выполнена операция по инициализации БД, то удалить каталог БД возможно при помощи команды:

```
rm -rf /var/lib/jatoba/6/data
```

### 3.1.3. Создание кластера

После основных действий по установке СУБД на серверах, выполняется процедура резервного копирования Master-сервера на Slave-сервер и автоматическое создание репликации между ними.

Перейдя на Slave-сервер от имени и с правами привилегированного пользователя ОС, выполняется создание кластера серверов командой:

```
sudo -u postgres ./pg_basebackup -h 192.168.23.133 -U postgres  
-D /var/lib/jatoba/6/data -R
```

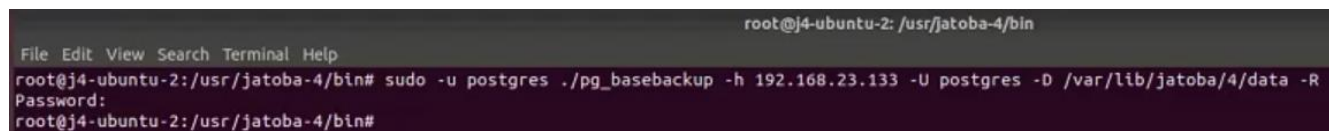


Рисунок 3.2 – Создание кластера горячего резерва

При успешном выполнении операции конфигурационные файлы «postgresql.conf» и «pg\_hba» на серверах будут идентичными.

Далее потребуется:

- запустить демон «Jatoba-6», командой:

```
systemctl start jatoba-6
```

- проверить статус демона «Jatoba-6», командой:

```
systemctl status jatoba-6
```

Демон должен иметь статус «active (running)».

- добавить демон «Jatoba-6» в автозагрузку ОС командой:

```
systemctl enable jatoba-6
```

На этом создание кластера горячего резерва завершено.

### 3.1.4. Инициализация компонента «jaPooler»

#### 3.1.4.1. Конфигурирование pgbouncer.ini

В разделе «configuration file» указать параметры:

```
postgres = host=192.168.23.133 dbname=postgres port=5432
host_ro=192.168.23.134 port_ro=5432 user=postgres
strategy=balancing
```

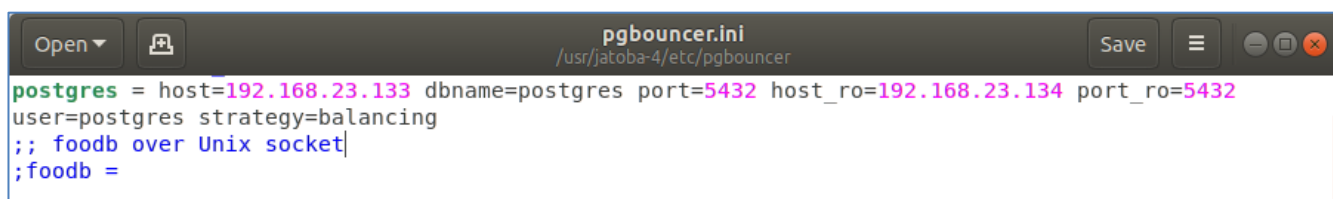


Рисунок 3.3 – Параметры конфигурационного файла pgbouncer.ini

В строке указываются параметры, приведенные в таблице 3.2

Таблица 3.2 – Параметры

Наименование	Параметр	Ссылка
Адрес Master-сервера	host=192.168.23.133	2.9.2
Имя БД для подключения	dbname=postgres	2.9.1
Порт подключения	port=5432	2.9.4
Адрес Slave-сервера	host_ro=192.168.23.134	2.9.3
Порт подключения к Slave – серверу (серверу горячего резерва)	port_ro=5432	2.9.5
Пользователь	user=postgres	2.9.7
Стратегия балансировки запросов	strategy=balancing	2.9.6

Таким образом компонент jaPooler, в режиме определения типа запроса (strategy=balancing), запросы, приходящие от пользователя «postgres» будет разбирать и запросы типа:

- RW направлять на Master-сервер;
- RO направлять на Slave-сервер.

В разделе «Administrative settings» указать параметры:

```
logfile = /usr/jatoba-6/var/log/pgbouncer/pgbouncer.log
pidfile = /usr/jatoba-6/var/run/pgbouncer/pgbouncer.pid
```

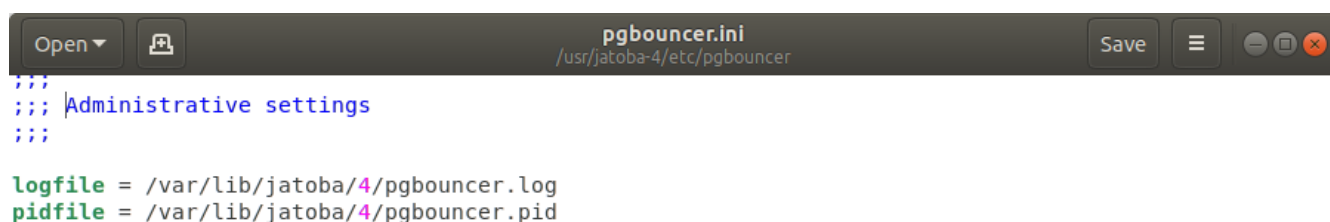


Рисунок 3.4 – Раздел «Administrative settings»

В разделе «Where to wait for clients» указать параметры:

```
listen_addr = localhost
listen_port = 6432
```

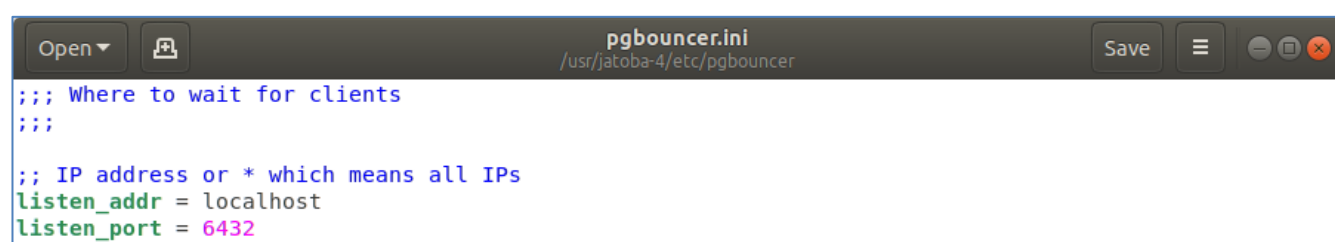


Рисунок 3.5 – Раздел «Where to wait for clients»

Параметры listen\_addr (см. п. 2.6.3) и listen\_port (см. п. 2.6.4) частично формируют строку подключения пользователя к компоненту jaPooler.

В разделе «Authentication settings» указать параметры:

```
auth_type = md5
auth_file = /usr/jatoba-6/etc/pgbouncer/userlist.txt
```

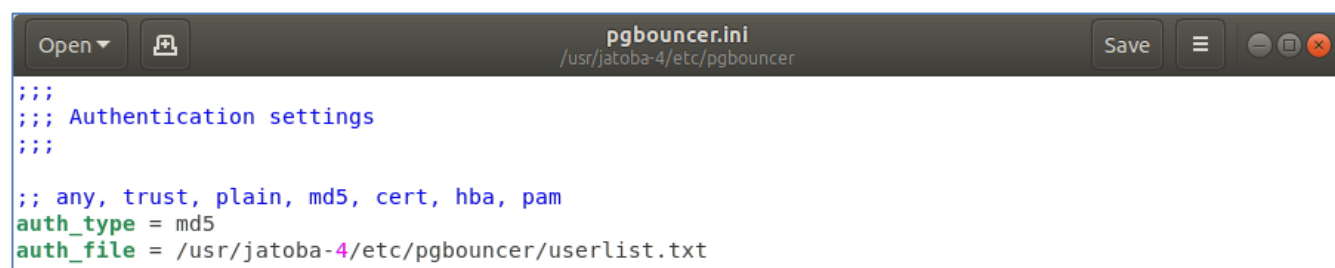


Рисунок 3.6 – Раздел «Authentication settings»

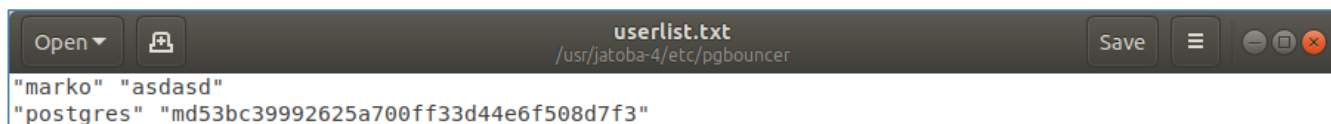
Параметр auth\_type (см. п. 2.7.1) указывает на используемый метод аутентификации при доступе к компоненту, а в параметре auth\_file (см. п. 2.7.3) приводится полный путь к файлу userlist.txt. Эти параметры влияют на формирование строки пользователя.

### 3.1.4.2. Формирование userlist.txt

В конфигурационном файле «userlist.txt» указываются пользователи, имеющие право доступа к компоненту.

Строка пользователя представляет собой указание имени учетной записи и хеш пароля пользователя.

В представленном примере представлен хеш пароля по 128-битному алгоритму хеширования MD5.



При этом указывается, алгоритм хеширования и без пробелов хеш пароля.

### 3.1.4.3. Запуск компонента

Компонент запускается от имени и с правами привилегированного пользователя ОС при помощи команды:

```
sudo systemctl start pgbouncer.service
```

При этом указывается полный путь к конфигурационному файлу pgbouncer.ini для применения установленных параметров.

После запуска компонента станет доступно подключение пользователя на порт 6432.

### 3.1.5. Подключение пользователя и выполнение запросов RW, RO

Для выполнения примера будут выполнены SQL-запросы RO и RW с выводом результатов.

Первоначально устанавливается соединения пользователем командой:

```
psql -h localhost -p 6432 -U postgres -d postgres
```

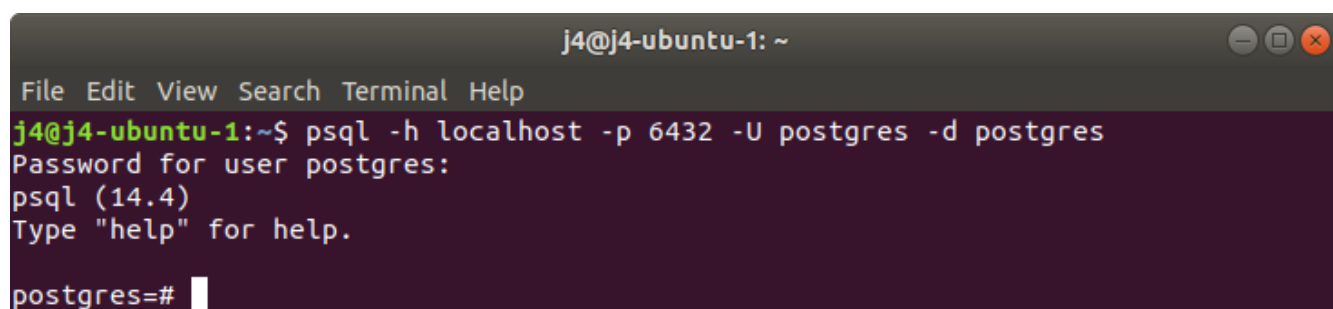


Рисунок 3.7 – Установление соединения с компонентом

После ввода пароля пользователь получает доступ в СУБД.

От имени и с правами пользователя, SQL-запросом:

```
SELECT inet_server_addr();
```

выводится IP-адрес сервера, с которым установлено соединение пользователем.

```

j4@j4-ubuntu-1: ~
File Edit View Search Terminal Help
j4@j4-ubuntu-1:~$ psql -h localhost -p 6432 -U postgres -d postgres
Password for user postgres:
psql (14.4)
Type "help" for help.

postgres=# SELECT inet_server_addr();
 inet_server_addr
-----
 192.168.23.134
(1 row)

postgres=#

```

Рисунок 3.8 – Вывод адреса Slave-сервера

Это означает, что компонент разобрал SQL-запрос, определил его тип как RO и установил соединение с Slave-сервером j4@j4-ubuntu-2, IP: 192.168.23.134/24.

Затем необходимо создать таблицу «t1» с помощью команды:

```
CREATE table t1(g int);
```

В информация компонента об обработке SQL-запроса CREATE видно, что запрос был определен как тип RW и направлен на Master-сервер j4@j4-ubuntu-1, IP: 192.168.23.133/24.

```

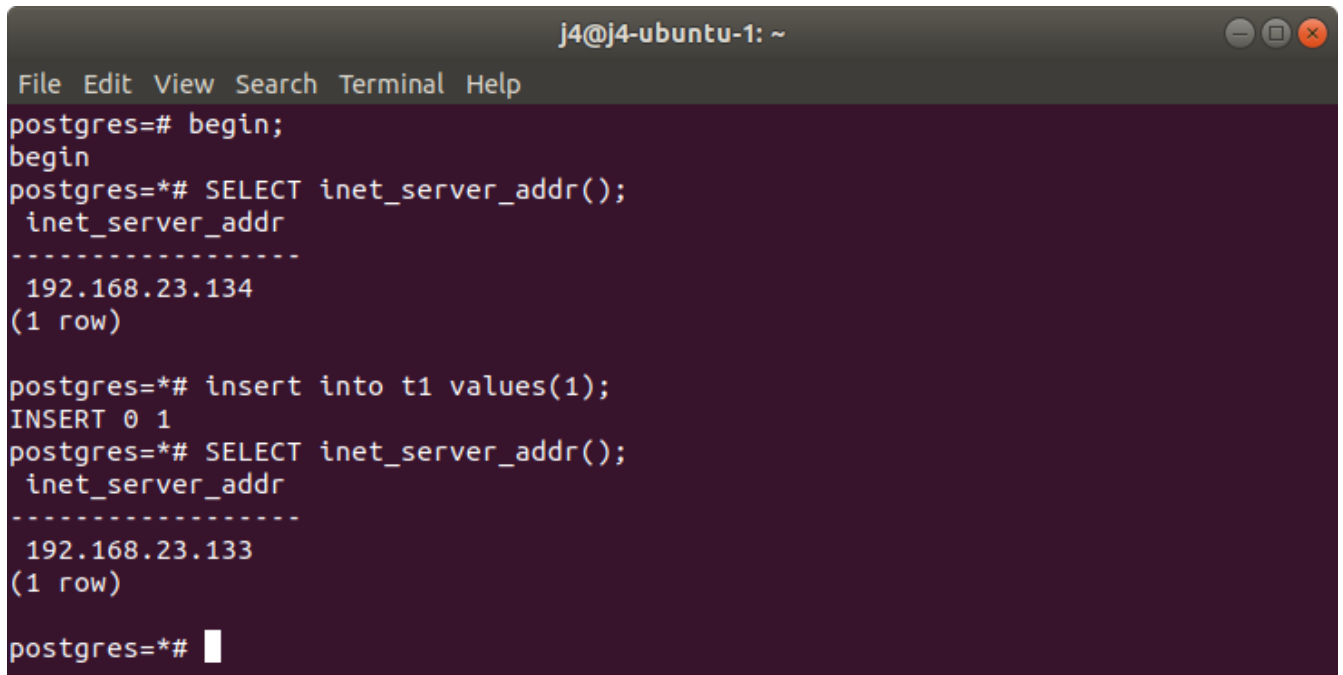
root@j4-ubuntu-1: /usr/jatoba-4/bin
File Edit View Search Terminal Help
2022-07-13 00:22:53.629 PDT [26167] DEBUG query: create table t1(g int); 0
2022-07-13 00:22:53.629 PDT [26167] DEBUG query create table t1(g int); 0
2022-07-13 00:22:53.629 PDT [26167] DEBUG setClientRwParameters
2022-07-13 00:22:53.629 PDT [26167] DEBUG find pool 2
2022-07-13 00:22:53.629 PDT [26167] DEBUG C-0x1153b20: postgres/postgres@127.0.0.1:43512 pause_client
2022-07-13 00:22:53.629 PDT [26167] NOISE S-0x115b058: postgres/postgres@192.168.23.133:5432 launching new connection to server
2022-07-13 00:22:53.629 PDT [26167] NOISE connect(15, 192.168.23.133:5432) = Operation now in progress
2022-07-13 00:22:53.630 PDT [26167] NOISE S-0x115b058: postgres/postgres@192.168.23.133:5432 S: connect ok
2022-07-13 00:22:53.630 PDT [26167] LOG S-0x115b058: postgres/postgres@192.168.23.133:5432 new connection to server (from 192.168.23.133:51818)
2022-07-13 00:22:53.630 PDT [26167] NOISE S-0x115b058: postgres/postgres@192.168.23.133:5432 P: startup
2022-07-13 00:22:53.630 PDT [26167] NOISE safe_send(15, 41) = 41
2022-07-13 00:22:53.630 PDT [26167] DEBUG launch_new_connection: already progress
2022-07-13 00:22:53.631 PDT [26167] NOISE resync(15): done=0, parse=0, recv=0
2022-07-13 00:22:53.631 PDT [26167] NOISE safe_recv(15, 4096) = 13
2022-07-13 00:22:53.631 PDT [26167] NOISE S-0x115b058: postgres/postgres@192.168.23.133:5432 read pkt='R', len=13
2022-07-13 00:22:53.631 PDT [26167] DEBUG S-0x115b058: postgres/postgres@192.168.23.133:5432 calling login_answer
2022-07-13 00:22:53.631 PDT [26167] DEBUG S-0x115b058: postgres/postgres@192.168.23.133:5432 S: req md5-crypted psw
2022-07-13 00:22:53.632 PDT [26167] DEBUG S-0x115b058: postgres/postgres@192.168.23.133:5432 P: send md5 password

```

Рисунок 3.9 – Информация компонента об обработке SQL-запроса CREATE

Следующим действием помещаем данные в ранее созданную таблицу «t1» и определяем сервер, к которому подключены:

```
INSERT into t1 values(1);
SELECT inet_server_addr();
```



```
j4@j4-ubuntu-1: ~  
File Edit View Search Terminal Help  
postgres=# begin;  
begin  
postgres=# SELECT inet_server_addr();  
inet_server_addr  
-----  
192.168.23.134  
(1 row)  
  
postgres=# insert into t1 values(1);  
INSERT 0 1  
postgres=# SELECT inet_server_addr();  
inet_server_addr  
-----  
192.168.23.133  
(1 row)  
  
postgres=#
```

Рисунок 3.10 – Последовательное выполнение SQL-запросов INSERT SELECT

В этом случае при попадании первого SQL-запроса на запись (RW), остальные SQL-запросы автоматически перенаправляются на Master-сервер, чтобы имелся доступ к одним и тем же данным.

### 3.1.6. Результат

В результате запросы типа:

- RW – CREATE table t1(g int); были направлены компонентом в адрес Master-сервера (IP 192.168.23.133);
- RO - SELECT inet\_server\_addr() были направлены компонентом в адрес Slave-сервера (IP 192.168.23.134), что соответствует стратегии балансировки «balancing».

Схема работы стенда с IP-адресами серверов представлена на рисунке 3.11.



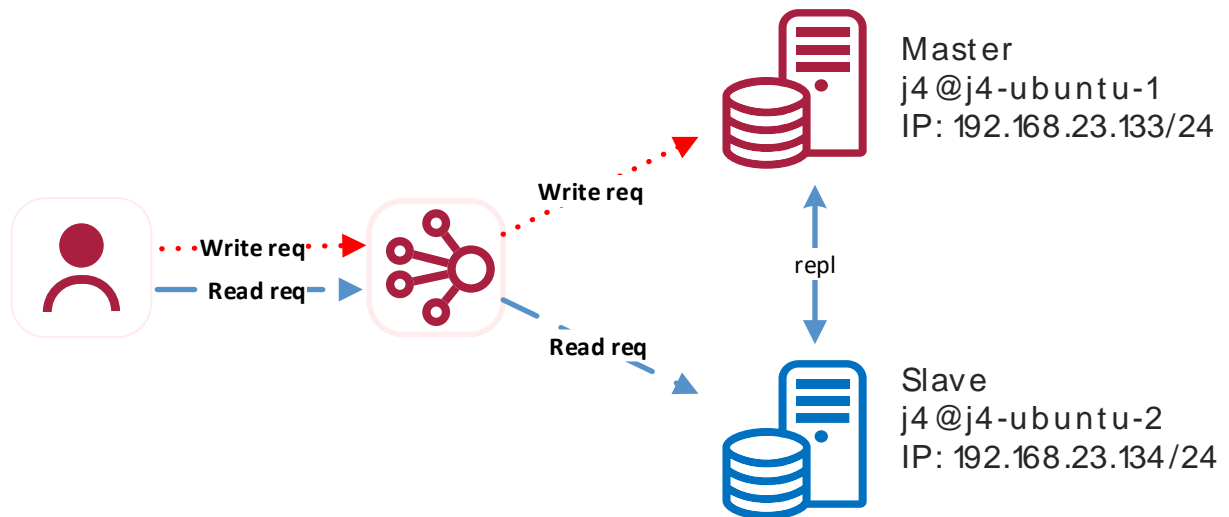


Рисунок 3.11 – Схема работы стенда при установленной стратегии «balancing».

### 3.2. Кластер горячего резерва в режиме стратегии «always\_rw»

Функционирование компонента jaPooler при установленной стратегии

```
strategy=always_rw
```

в конфигурационном файле pgbouncer.ini, определяется как направление всех типов SQL-запросов пользователя и RO и RW на Master-сервер.

Принципиальная схема работы компонента при установленной стратегии «always\_rw» представлена на рисунке 3.12.

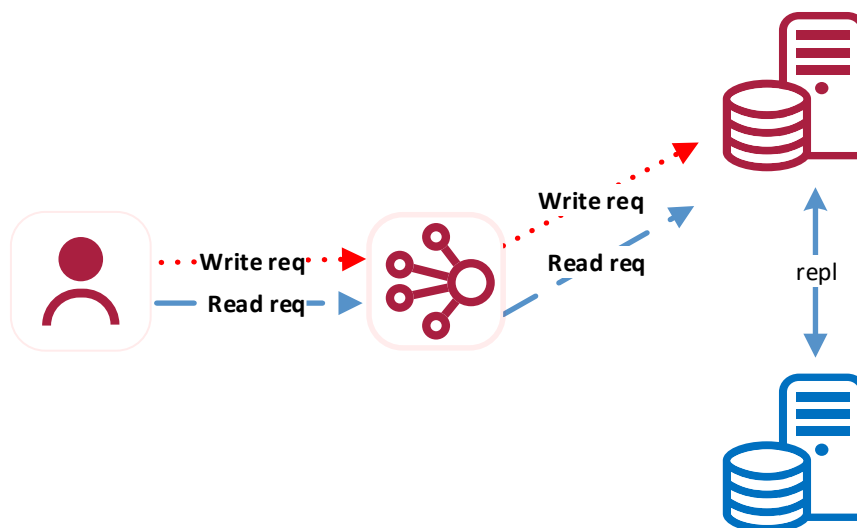


Рисунок 3.12 – Схема работы при установленной стратегии «always\_rw»

Конфигурирование компонента в режим «always\_rw» осуществляется через конфигурационный файл pgbouncer.ini внесением параметров:

```
postgres = host=192.168.23.133 dbname=postgres port=5432
host_ro=192.168.23.134 port_ro=5432 user=postgres
strategy=always_rw
```

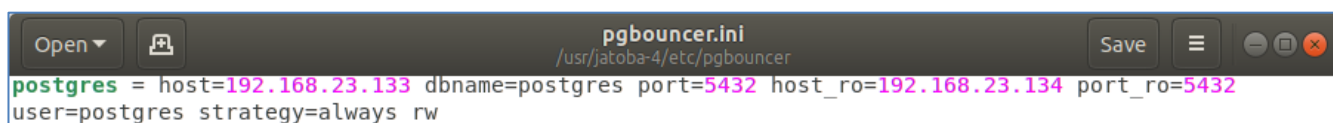


Рисунок 3.13 – Параметры конфигурационного файла pgbouncer.ini

Для применения параметров следует перезагрузить компонент сочетанием клавиш CTRL+C и последующей командой запуска компонента:

```
sudo systemctl start pgbouncer.service
```

Для выполнения примера будут выполнены SQL-запросы RO и RW с выводом результатов.

Первоначально устанавливается соединения пользователем командой:

```
psql -h localhost -p 6432 -U postgres -d postgres
```

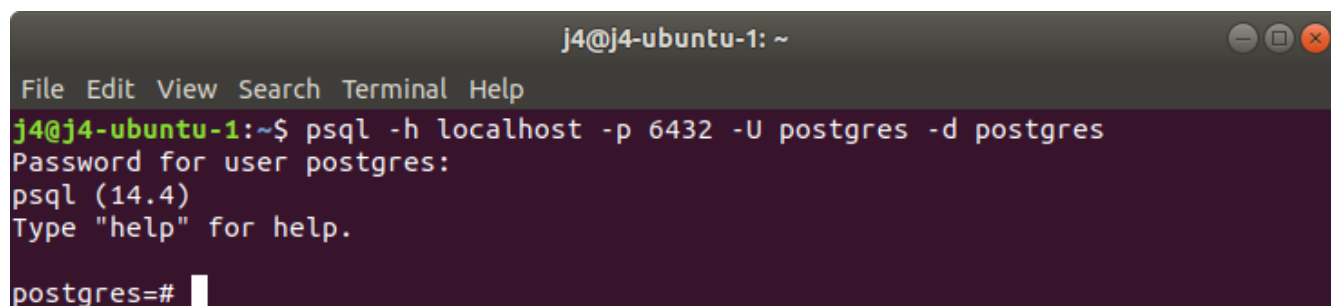


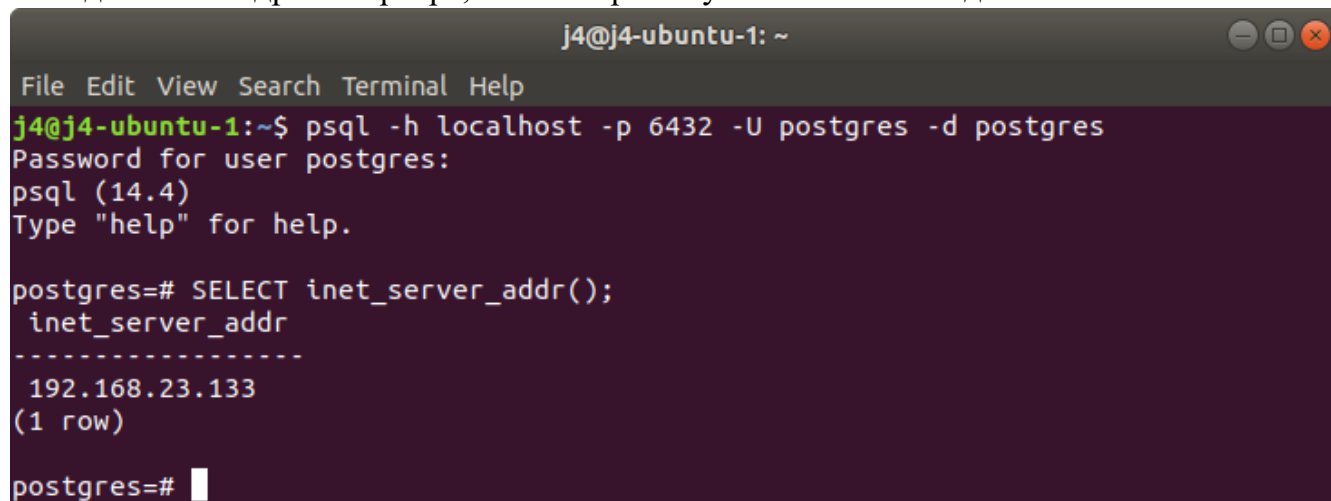
Рисунок 3.14 – Установление соединения с компонентом

После ввода пароля пользователь получает доступ в СУБД.

От имени и с правами пользователя SQL-запросом:

```
SELECT inet_server_addr();
```

выводится IP-адрес сервера, с которым установлено соединение пользователем.



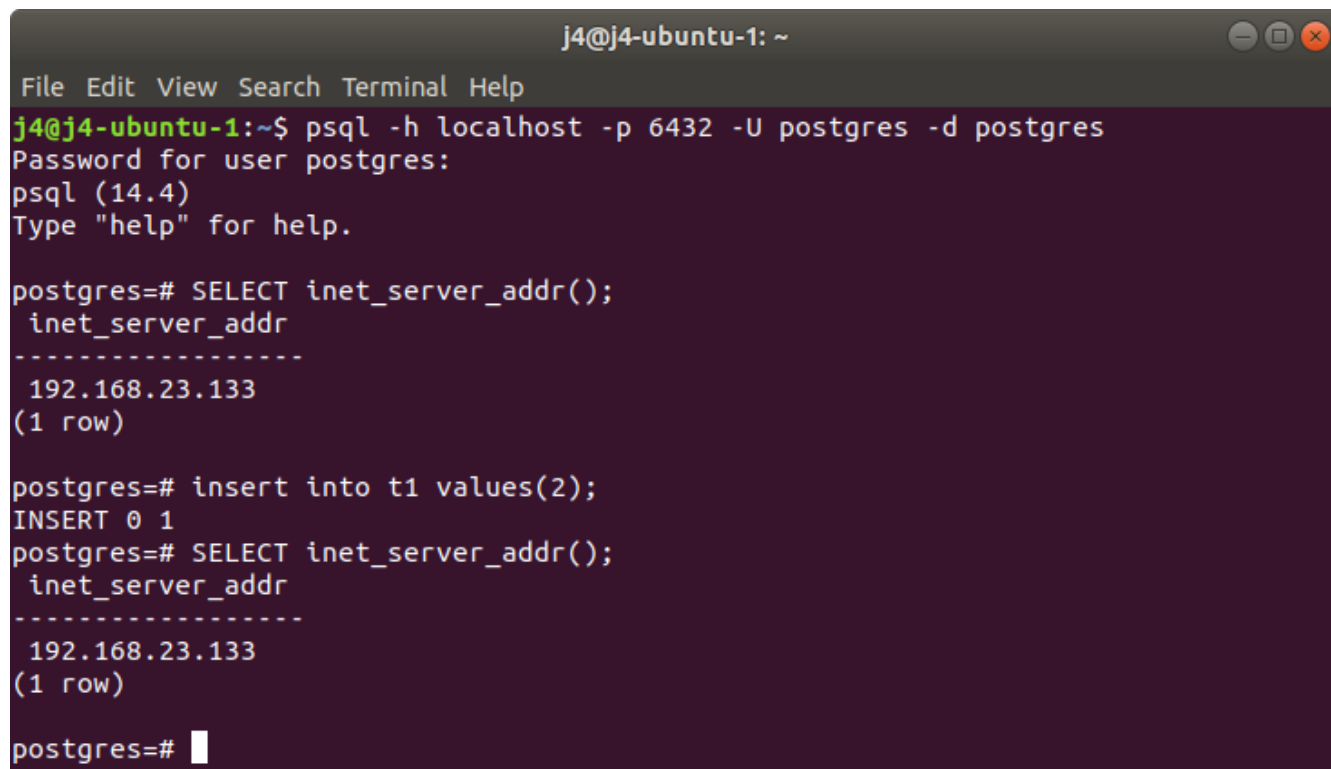
```
j4@j4-ubuntu-1: ~  
File Edit View Search Terminal Help  
j4@j4-ubuntu-1:~$ psql -h localhost -p 6432 -U postgres -d postgres  
Password for user postgres:  
psql (14.4)  
Type "help" for help.  
  
postgres=# SELECT inet_server_addr();  
inet_server_addr  
-----  
192.168.23.133  
(1 row)  
  
postgres=#
```

Рисунок 3.15 – Вывод адреса Master-сервера

В данном случае установлено, соединение с Master – сервером j4@j4-ubuntu-1, IP: 192.168.23.133/24.

Следующим действием помещаем данные в ранее созданную таблицу «t1»:

```
INSERT into t1 values (2);
```



```
j4@j4-ubuntu-1: ~  
File Edit View Search Terminal Help  
j4@j4-ubuntu-1:~$ psql -h localhost -p 6432 -U postgres -d postgres  
Password for user postgres:  
psql (14.4)  
Type "help" for help.  
  
postgres=# SELECT inet_server_addr();  
inet_server_addr  
-----  
192.168.23.133  
(1 row)  
  
postgres=# insert into t1 values (2);  
INSERT 0 1  
postgres=# SELECT inet_server_addr();  
inet_server_addr  
-----  
192.168.23.133  
(1 row)  
  
postgres=#
```

Рисунок 3.16 – Вставка данных в таблицу «t1»

### 3.2.1. Результат

В результате запросы типа:

- RO – SELECT inet\_server\_addr();
- RW – INSERT into t1 values(2);

были направлены компонентом в адрес Master-сервера (IP 192.168.23.133), что соответствует стратегии балансировки «always\_rw».

Схема работы стенда с IP-адресами серверов представлена на рисунке 3.17.

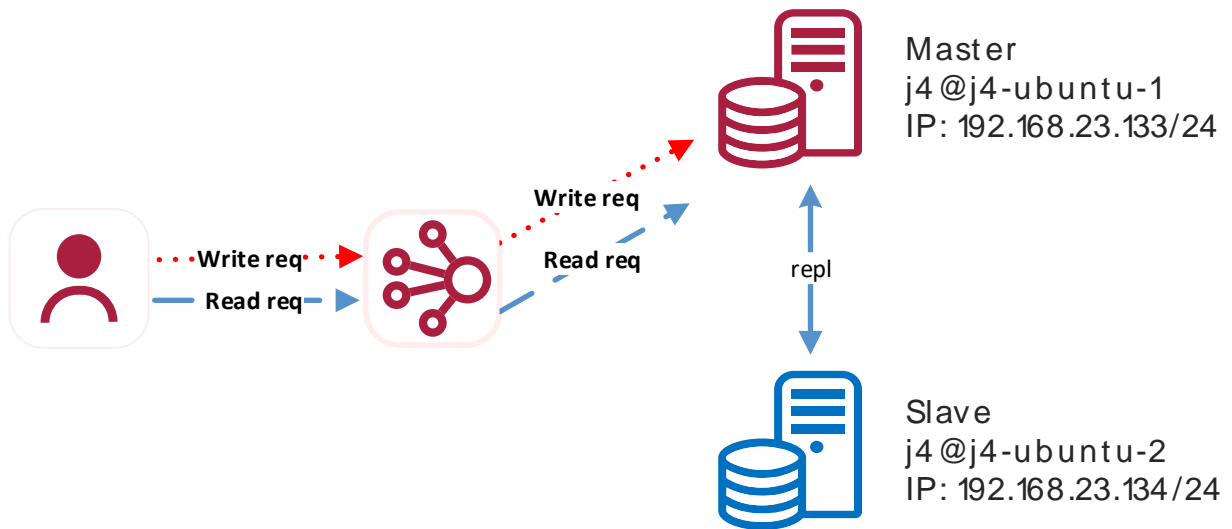


Рисунок 3.17 – Схема работы стенда при установленной стратегии «always\_rw»

### 3.3. Кластер горячего резерва в режиме стратегии «always\_ro»

Функционирование компонента jaPooler при установленной стратегии

```
strategy=always_ro
```

в конфигурационном файле pgbouncer.ini, определяется как направление всех типов SQL-запросов пользователя и RO и RW, на Slave-сервер.

Принципиальная схема работы компонента при установленной стратегии «always\_ro» представлена на рисунке 3.18.

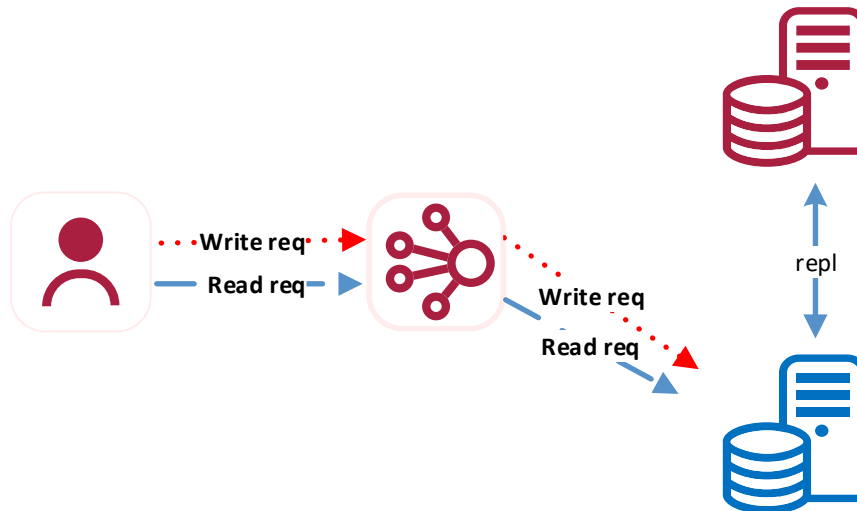


Рисунок 3.18 – Схема работы при установленной стратегии «always\_ro»

Конфигурирование компонента в режим «always\_ro» осуществляется через конфигурационный файл `pgbouncer.ini` внесением параметров:

```
postgres = host=192.168.23.133 dbname=postgres port=5432
host_ro=192.168.23.134 port_ro=5432 user=postgres
strategy=always_ro
```

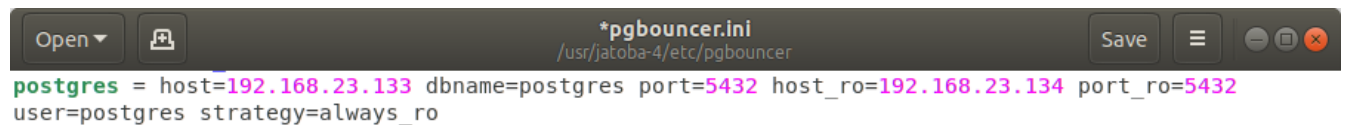


Рисунок 3.19 – Параметры конфигурационного файла `pgbouncer.ini`

Устанавливается соединение пользователем командой:

```
psql -h localhost -p 6432 -U postgres -d postgres
```

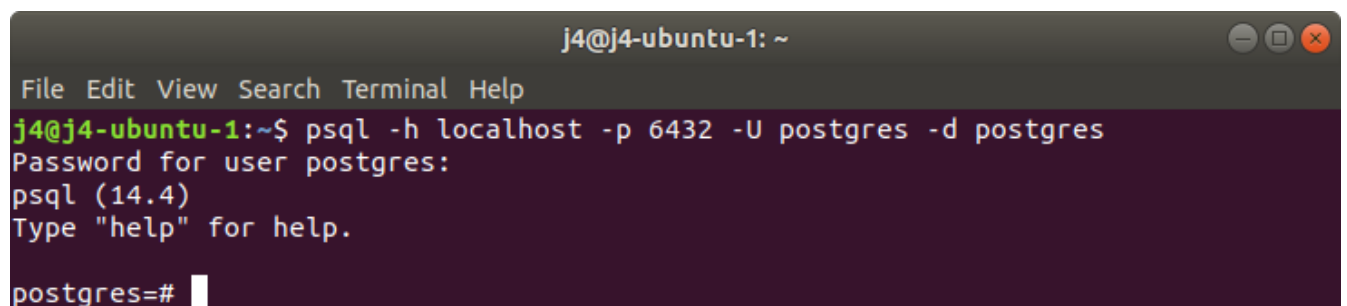


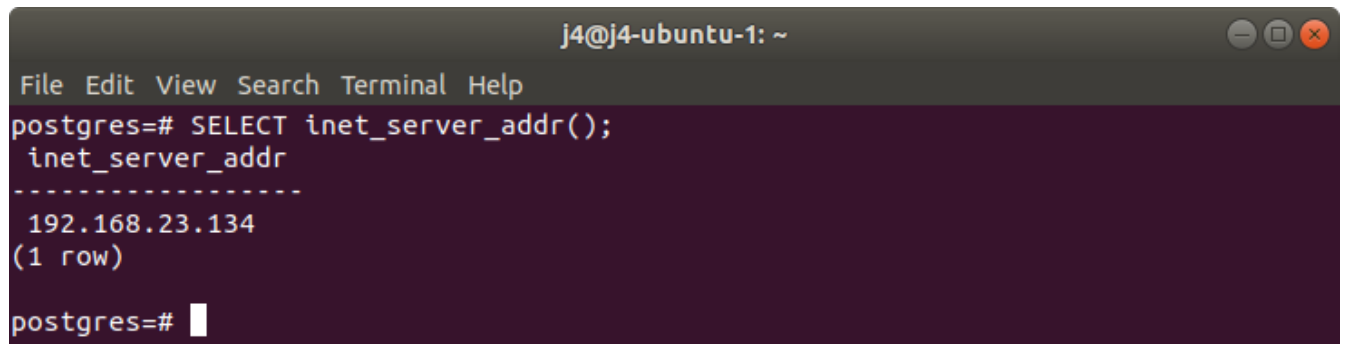
Рисунок 3.20 – Установление соединения с компонентом

После ввода пароля пользователь получает доступ в СУБД.

От имени и с правами пользователя SQL-запросом:

```
SELECT inet_server_addr();
```

выводится IP-адрес сервера, с которым установлено соединение пользователем.



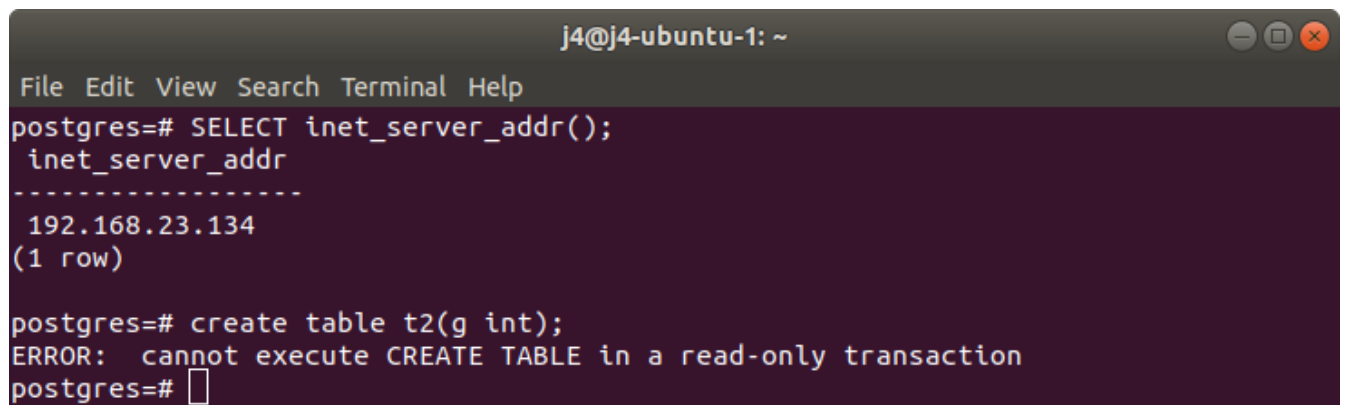
```
j4@j4-ubuntu-1: ~  
File Edit View Search Terminal Help  
postgres=# SELECT inet_server_addr();  
inet_server_addr  
-----  
192.168.23.134  
(1 row)  
postgres=#
```

Рисунок 3.21 – Вывод адреса Slave-сервера

Соединение установлено с Slave – сервером j4@j4-ubuntu-2, IP: 192.168.23.134/24.

От имени и с правами пользователя необходимо выполнить SQL-команду создания таблицы «t2»:

```
CREATE table t2(g int);
```

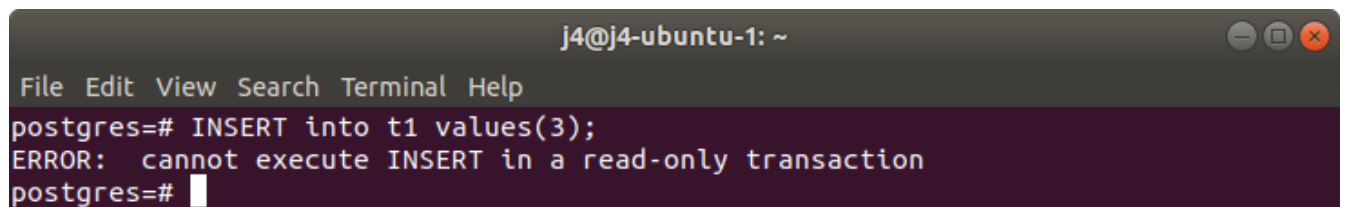


```
j4@j4-ubuntu-1: ~  
File Edit View Search Terminal Help  
postgres=# SELECT inet_server_addr();  
inet_server_addr  
-----  
192.168.23.134  
(1 row)  
postgres=# create table t2(g int);  
ERROR: cannot execute CREATE TABLE in a read-only transaction  
postgres=#
```

Рисунок 3.22 – SQL-команды создания таблицы «t2»

SQL-командой пытаемся внести значение в ранее созданную таблицу «t1»:

```
INSERT into t1 values(3);
```



```
j4@j4-ubuntu-1: ~  
File Edit View Search Terminal Help  
postgres=# INSERT into t1 values(3);  
ERROR: cannot execute INSERT in a read-only transaction  
postgres=#
```

Рисунок 3.23 – Вставка значения в таблицу «t1»

В результате выполнить SQL - команды, связанные с записью в БД невозможно, т.к. транзакции доступны только на чтение.

СУБД выведет ошибку:

```
cannot execute [operation] in a read-only transaction
```

### 3.3.1. Результат

В результате запросы типа:

- RO – SELECT inet\_server\_addr();
- RW – INSERT into t1 values(2);

были направлены компонентом в адрес Slave-сервера (IP 192.168.23.134), что соответствует стратегии балансировки «always\_rw».



Запросы типа RW будут отвергнуты, т.к. Slave-сервер работает только на чтение и операции записи не выполняются.

Схема работы стенда с IP-адресами серверов представлена на рисунке 3.24

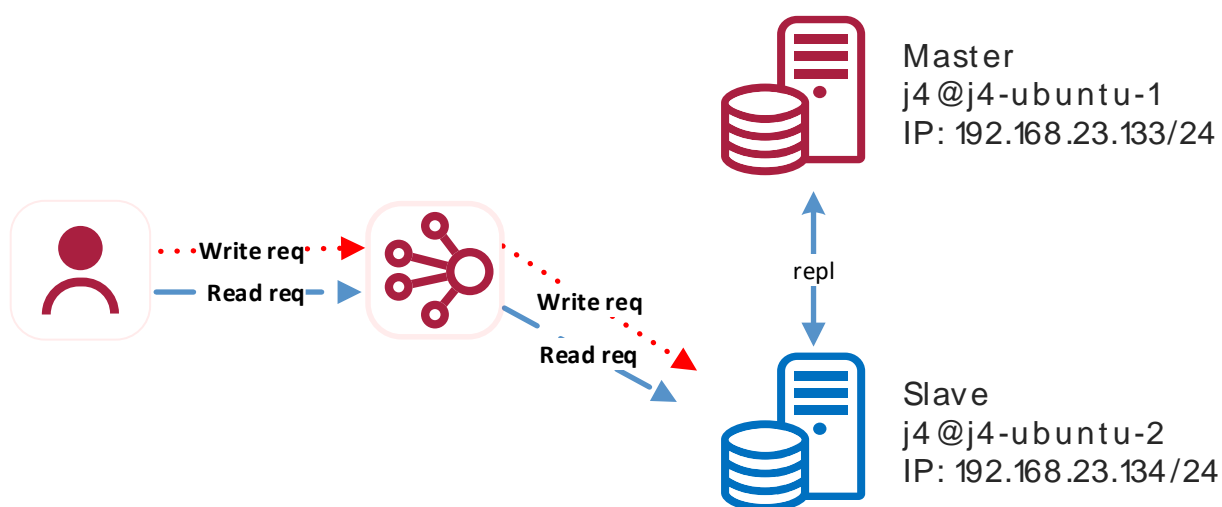


Рисунок 3.24 – Схема работы при установленной стратегии «always\_ro»

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ**

SQL	–	Structured Query Language – язык структурированных запросов
БД	–	База данных
ОС	–	Операционная система
СУБД	–	Система управления базами данных
ЭВМ	–	Электронно-вычислительная машина



№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм.: _____
--------------------	--------------------------	---------------------------