

УТВЕРЖДЕН  
643.72410666.00067-07 98 01-ЛУ

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

**Руководство по настройке. Часть 34.  
Архивация и восстановление данных.  
Компонент «wal-g»**

**643.72410666.00067-07 98 01-34**

Листов 68

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

## АННОТАЦИЯ

В документе приведены сведения, необходимые для установки и эксплуатации компонента «wal-g» (далее по тексту – «компонент» или wal-g), предназначенного для создания резервных копий (далее – РК) и архивирования WAL (Write-Ahead Logging) и восстановления из архивов в СУБД «Jatoba».

Настоящее руководство предназначено для администраторов СУБД.



Все примеры в данном документе приведены для СУБД «Jatoba» версии ядра 6.x, для других версий все шаги выполняются аналогично, разница состоит в именах директорий.

Версия компонента — 3.0.7.

Степени важности примечаний, применяемые в документе:



**Важная информация** – указания, требующие особого внимания



**Дополнительная информация** – указания, позволяющие упростить работу с изделием



**Важная информация**

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

## СОДЕРЖАНИЕ

1. Назначение компонента.....	5
1.1. Условия применения.....	5
1.2. Ограничения при работе с компонентом.....	5
2. Установка и настройка.....	6
2.1. Установка компонента в ОС GNU/Linux.....	6
2.2. Настройка конфигурационного файла компонента .....	7
2.2.1. Переменные среды окружения.....	9
2.2.2. Настройка каталога сохранения резервных копий.....	9
2.2.3. Настройка метода сжатия данных резервных копий.....	10
2.2.4. Настройка параметров создания инкрементальных резервных копий .....	10
2.2.5. Настройка параметров восстановления .....	13
2.2.6. Настройка ограничений дисковых операций.....	14
2.2.7. Настройка ограничений сетевых операций .....	15
2.2.8. Включение метрик мониторинга .....	19
2.2.9. Настройка параметров подключения к СУБД.....	20
2.2.10. Настройка защищенного соединения .....	23
2.2.11. Параметры журналирования событий .....	24
2.3. Настройка СУБД.....	25
2.3.1. Создание пользователя для выполнения архивации и восстановления данных.....	25
2.3.2. Настройка непрерывного архивирования WAL .....	25
3. Функциональные возможности компонента.....	28
3.1. Создание резервной копии.....	28
3.1.1. Создание полной резервной копии .....	28
3.1.2. Создание постоянной резервной копии .....	31
3.1.3. Создание инкрементальной резервной копии .....	33
3.1.4. Создание инкрементальной резервной копии на основе базовой РК.....	35
3.1.5. Создание резервной копии с удаленного узла.....	36
3.2. Просмотр созданных резервных копий.....	38
3.3. Проверка созданных резервных копий.....	40
3.3.1. Использование аргумента integrity .....	40
3.3.2. Использование аргумента timeline.....	42
3.3.3. Использование обоих аргументов.....	43
3.4. Загрузка резервных копий журналов транзакций (WAL) в хранилище.....	44
3.5. Восстановление данных из резервной копии .....	44
3.5.1. Предварительные условия для выполнения восстановления данных из резервной копии.....	45
3.5.2. Восстановление данных из последней резервной копии .....	45
3.5.3. Восстановление данных из резервной копии на определенный момент времени .....	48

3.5.4. Восстановление данных на момент сразу после создания РК .....	52
3.6. Удаление резервных копий.....	52
3.6.1. Режим retain .....	52
3.6.2. Режим before .....	55
3.6.3. Режим everything .....	57
3.6.4. Режим target .....	59
3.6.5. Режим garbage.....	62
3.7. Перемещение резервных копий.....	63
3.8. Журнал работы компонента.....	63
4. Удаление расширения.....	66
Перечень сокращений.....	67

## 1. НАЗНАЧЕНИЕ КОМПОНЕНТА

Компонент «wal-g» — представляет собой расширение функционала СУБД «Jatoba» и предназначен для управления резервным копированием и восстановлением баз данных СУБД «Jatoba», для регулярного создания резервных копий (РК), позволяющих восстанавливать работу СУБД в случае аварийной ситуации, порчи или потери данных.

Для управления резервными копиями «wal-g» создает каталог, в который сохраняются все файлы резервных копий (РК) с дополнительной метаинформацией, а также архивы журнала транзакций (WAL), необходимые для восстановления на выбранный момент времени.

Используя «wal-g», можно выполнить полное или инкрементальное резервное копирование:

- Полные РК содержат все файлы данных, необходимые для восстановления сервера баз данных с нулевой точки.
- Инкрементальные РК создаются на уровне страниц данных и включают только ту информацию, которая изменилась со времени последнего резервного копирования.

Инкрементальное копирование позволяет экономить место на диске и создавать РК быстрее, чем при полном копировании.

### 1.1. Условия применения

Компонент «wal-g» может использоваться с СУБД «Jatoba» версий 6.x и выше, и только под управлением операционных систем GNU/Linux.

Компонент выполнен в форме расширения СУБД и не имеет ограничений по совместимости с другими компонентами.

### 1.2. Ограничения при работе с компонентом

Имеет ряд особенностей по функциональным возможностям:

- Создание РК в форматах LZ4, LZMA, ZSTD и Brotli. По умолчанию для сжатия РК применяются LZ4;
- Преобразование резервных копий ограничивается алгоритмом AES-256.

## 2. УСТАНОВКА И НАСТРОЙКА

Установка компонента должна производиться от имени пользователя, обладающего административными привилегиями в ОС.

Установка компонента под управлением ОС GNU/Linux приведено ниже.

### 2.1. Установка компонента в ОС GNU/Linux

Компонент «wal-g» устанавливается в составе СУБД «Jatoba». Его возможно установить при первичной установке либо при последующей эксплуатации СУБД.

Установку пакета 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 jatoba6-wal-g
```

— 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 jatoba6-wal_g
```

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

```
# apt-get install jatoba6-wal_g
```

Установка компонента в составе других версий СУБД «Jatoba» осуществляется аналогично. Отличие будет только в номере версии СУБД, в составе которой он распространяется.



### **Важная информация**

В ОС РОСА 7.9 перед началом установки компонента необходимо:

- 1) Выполнить установку репозитория:

```
# dnf install https://archives.fedoraproject.org/pub/archive/epel/7/x86_64/Packages/e/epel-release-7-14.noarch.rpm
```

- 2) Обновить состояние репозитория:

```
# yum makecache
```

- 3) Выполнить обновление:

```
# yum update -y
```

- 4) Выполнить установку компонента для систем на основе пакетных менеджеров YUM/DNF

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

Для удобства работы с компонентом рекомендуется создать символическую ссылку на исполняемый файл при помощи команды:

```
# ln -s /usr/jatoba-6/bin/wal-g /usr/bin/wal-g
```

## **2.2. Настройка конфигурационного файла компонента**

Для использования компонента «wal-g» необходимо создать пользователя СУБД backup с атрибутом Login и придерживаться принципа минимизации назначаемых атрибутов и привилегий. Назначение дополнительных атрибутов роли, системных привилегий и прав осуществляется в соответствии с выбранным типом и режимом резервного копирования.

Для настройки компонента используется конфигурационный файл wal.json.

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

Значения параметров, указанных в конфигурационном файле `walg.json`, переопределяют встроенные значения по умолчанию.

После установки компонента необходимо вручную создать конфигурационный файл `walg.json` в домашнем каталоге пользователя `postgres` при помощи команды:

```
# touch /var/lib/jatoba/.walg.json
```



Если в домашнем каталоге пользователя `postgres` отсутствует конфигурационный файл `walg.json`, то при выполнении команд компонента будет отображаться ошибка «Failed to find any configured storage».

При вызове команд компонента возможно указание другого расположения конфигурационного файла `walg.json` через использование флага «`--config`», например:

```
postgres@node1:~$ wal-g backup-list --config $HOME/.walg.json
```

Пример конфигурационного файла `walg.json` компонента представлен в следующем листинге:

```
{
  "WALG_FILE_PREFIX": "/nfs/walg_archive/server1",
  "WALG_COMPRESSION_METHOD": "lz4",
  "WALG_DELTA_MAX_STEPS": "2",
  "PGDATA": "/var/lib/jatoba/6/data",
  "PGHOST": "/var/run/jatoba/",
  "WALG_LOG_DESTINATION": "/var/lib/jatoba/wal-g.log"
}
```

После создания конфигурационного файла `.walg.json` и определения параметров создания РК необходимо установить для него права доступа для пользователя `postgres`:

```
# chown postgres: /var/lib/jatoba/.walg.json
```

Чтение параметров из конфигурационного файла `.walg.json` производится компонентом при каждом вызове команд.



В случае если конфигурационный файл `.walg.json` располагается в каталоге отличном от используемого пользователем `postgres`, его расположение возможно указать при помощи флага `--config`, например:

```
postgres@node1:~$ wal-g backup-push /var/lib/jatoba/6/data --
config /var/lib/jatoba/.walg.json
```

При отсутствии конфигурационного файла `.walg.json` при выполнении команд компонента будет отображаться ошибка:

```
ERROR: 2025/05/29 15:45:23.531890 Failed to find any configured
storage
```

### 2.2.1. Переменные среды окружения

Переменные среды могут использоваться для определения значений параметров по умолчанию при подключении компонента к СУБД.

Таблица 2.1 – Значения параметров переменных среды окружения в файле `walg.json`

Параметр	Возможные значения	Описание
PGHOST	"/var/run/jatoba", "localhost", "remotehostname.domain", "10.81.250.6:5432"	Название хоста для удаленного подключения. Если имя хоста выглядит как абсолютный путь, то это указывает на использование межпроцессного взаимодействия Unix-домена вместо TCP/IP-соединения. Этот параметр предпочтительный для локальных подключений
PGHOSTADDR	"10.81.250.6:5432"	URI-адрес хоста для подключения. Этот параметр предпочтительный для удаленных подключений
PGDATA	"/var/lib/jatoba/6/data"	Расположение каталога данных СУБД

### 2.2.2. Настройка каталога сохранения резервных копий

Для хранения создаваемых РК необходимо использовать отдельный каталог.

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

WALG\_FILE\_PREFIX определяет для каждого сервера отдельный каталог, что позволяет при выполнении восстановления провести процедуру.

В конфигурационном файле отдельно определяется параметр WALG\_FILE\_PREFIX, который указывает путь к каталогу для хранения резервных копий и журнальных файлов.

Таблица 2.2 – Значения параметра WALG\_FILE\_PREFIX

Параметр	Возможные значения	Описание
WALG_FILE_PREFIX	"/nfs/walg_archive/server1"	Расположение каталога для хранения резервных копий и журнальных файлов

В качестве каталога сохранения РК может выступать подключенный в файловую систему ОС каталог NFS-сервера, в этом случае путь к такому каталогу может иметь вид "/mnt/wal-g/backups/".

### 2.2.3. Настройка метода сжатия данных резервных копий

Для определения метода сжатия, используемого для резервных копий, используется параметр WALG\_COMPRESSION\_METHOD.

Таблица 2.3 – Значения параметра WALG\_COMPRESSION\_METHOD

Параметр	Возможные значения	Описание
WALG_COMPRESSION_METHOD	"lz4", "lzma", "zstd", "brotli"	Выбранный метод сжатия данных РК

Одновременно может быть определен только один метод сжатия данных.

Применяемый в компоненте метод сжатия данных РК по умолчанию: lz4.

### 2.2.4. Настройка параметров создания инкрементальных резервных копий

Инкрементальные РК хранят не полную копию данных, а только список изменений (дельту) между последней полной РК и текущим состоянием СУБД.

Такой подход позволяет значительно экономить дисковое пространство, поскольку вместо создания новых полных копий каждый раз сохраняются лишь те данные, которые были изменены.


Для управления количеством инкрементальных РК, которые могут быть созданы перед следующей полной РК, используется параметр WALG\_DELTA\_MAX\_STEPS.

Параметр WALG\_DELTA\_MAX\_STEPS определяет максимальное количество инкрементальных шагов (дельт) между двумя полными РК, что помогает балансировать между экономией места и необходимостью периодического создания полных бэкапов для обеспечения надежности.

Например, если для параметра WALG\_DELTA\_MAX\_STEPS задано значение "3", то после создания трех инкрементальных РК компонент принудительно создаст новую полную РК.

Таблица 2.4 – Значения параметра WALG\_DELTA\_MAX\_STEPS

Параметр	Возможные значения	Описание
WALG_DELTA_MAX_STEPS	"[число]"	Количество инкрементальных РК между созданием полных РК.

-  По умолчанию значение параметра WALG\_DELTA\_MAX\_STEPS равно 0 в случае если параметр явно не указан в конфигурационном файле walg.json

Параметр WALG\_DELTA\_ORIGIN определяет исходную РК, относительно которой формируются последующие инкрементальные РК.


-  Параметр WALG\_DELTA\_ORIGIN рекомендуется использовать совместно с параметром WALG\_DELTA\_MAX\_STEPS

Таблица 2.5 – Значения параметра WALG\_DELTA\_ORIGIN

Параметр	Возможные значения	Описание
WALG_DELTA_ORIGIN	"LATEST" (по умолчанию) - инкрементальная РК создается относительно последней РК	Определение РК, на основании которой будут созданы инкрементальные РК.

	"LATEST_FULL" – инкрементальная РК создается относительно последней полной РК	
--	---	--

Если параметр WALG\_DELTA\_ORIGIN не определен, компонент автоматически выбирает последнюю РК в качестве основы для создания инкрементальных РК.

Если для параметра WALG\_DELTA\_ORIGIN указано значение LATEST\_FULL, то даже если при создании указано название инкрементальной РК (например с помощью флага --delta-from-name), компонент восстановит цепочку до полной РК, и создаст новую инкрементальную РК на ее основе:

```
list of backups in storage:
base_000000010000000100000040 # полная РК
base_000000010000000100000046_D_000000010000000100000040 # 1я инкрем. РК
base_000000010000000100000061_D_000000010000000100000046 # 2я инкрем. РК
base_000000010000000100000070 # полная РК

export WALG_DELTA_ORIGIN=LATEST_FULL # настройка параметра
WALG_DELTA_ORIGIN в файле walg.json

wal-g backup-push /path --delta-from-name
base_000000010000000100000046_D_000000010000000100000040 # в качестве исходной
указана инкрементальная РК

wal-g logs: # в журнале компонента выводится информация о том, что инкрементальная
РК проигнорирована и выбрана родительская полная РК

INFO: Selecting the backup with name
base_000000010000000100000046_D_000000010000000100000040 as the base for
the current delta backup...

INFO: Delta will be made from full backup.

INFO: Delta backup from base_000000010000000100000040 with LSN 140000060.
```

Параметр WALG\_PREVENT\_WAL\_OVERWRITE управляет режимом защиты WAL-файлов (Write-Ahead Log) от перезаписи при выполнении операций резервного копирования.

Таблица 2.6 – Значения параметра WALG\_PREVENT\_WAL\_OVERWRITE

Параметр	Возможные значения	Описание
WALG_PREVENT_WAL_OVERWRITE	"false" (по умолчанию) – отключение защиты  "true" – включение защиты	Режим защиты РК от перезаписи

Параметр применяется только при использовании компонента для:

- Резервного копирования (команда backup-push);
- Управления РК (команды wal-push, wal-fetch).

При активации параметра WALG\_PREVENT\_WAL\_OVERWRITE компонент проверяет наличие существующих WAL-файлов в целевом хранилище. При обнаружении дубликата операция создания РК завершается с ошибкой. В этом случае гарантируется целостность архивов для PITR (Point-in-Time Recovery).

## 2.2.5. Настройка параметров восстановления

### 2.2.5.1 Параметр WALG\_DOWNLOAD\_FILE\_RETRIES

Параметр WALG\_DOWNLOAD\_FILE\_RETRIES определяет максимальное количество попыток повторной загрузки РК из хранилища при возникновении ошибок в процессе восстановления данных.

Параметр WALG\_DOWNLOAD\_FILE\_RETRIES применяется исключительно для операций:

- Восстановления из резервной копии (backup-fetch);
- Получения WAL-файлов (wal-fetch).

Параметр WALG\_DOWNLOAD\_FILE\_RETRIES действует для всех поддерживаемых типов хранилищ (SSH, файловая система и других).

Таблица 2.7 – Значения параметра WALG\_DOWNLOAD\_FILE\_RETRIES

Параметр	Возможные значения	Описание
----------	--------------------	----------

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

WALG_DOWNLOAD_FILE_RETRIES	"[число]"	Максимальное количество попыток повторной загрузки РК из хранилища
----------------------------	-----------	--

По умолчанию значение параметра WALG\_DOWNLOAD\_FILE\_RETRIES – 15.

Взаимодействие параметра WALG\_DOWNLOAD\_FILE\_RETRIES с другими параметрами:

— WALG\_DOWNLOAD\_CONCURRENCY – влияет на параллельные загрузки РК из хранилища (см. п.п. 2.2.7.1);

— WALG\_SENTINEL\_USER\_DATA – может содержать информацию о попытках загрузки РК.

Каждая попытка загрузки РК использует для установления соединения с сервером хранилища новый TCP-сеанс.

Журнал работы компонента содержит информационное поле retry\_attempt о количестве попыток.

При исчерпании, установленных в конфигурационном файле, числа попыток загрузки РК из хранилища журнал работы компонента будет содержать код ошибки WAL-E-RETRY-LIMIT.

### 2.2.6. Настройка ограничений дисковых операций

Параметр WALG\_DISK\_RATE\_LIMIT регулирует максимальную скорость чтения данных с диска при выполнении компонентом операций резервного копирования, предотвращая перегрузку системы ввода-вывода (I/O).



Ограничение параметра WALG\_DISK\_RATE\_LIMIT применяется на уровне **чтения данных с диска**, а не записи в хранилище

Таблица 2.8 – Значения параметра WALG\_DISK\_RATE\_LIMIT

Параметр	Возможные значения	Описание
----------	--------------------	----------

WALG_DISK_RATE_LIMIT	"[число]"	Ограничение скорости чтения диска во время выполнения команды backup-push, в байтах в секунду
----------------------	-----------	---

При значении параметра «WALG\_DISK\_RATE\_LIMIT=0» ограничение скорости чтения диска во время выполнения команды backup-push отключено (максимальная производительность).

При некорректном значении параметра WALG\_DISK\_RATE\_LIMIT (например, отрицательном) – отображается ошибка WAL-E-CONFIG-INVALID.

Взаимодействие параметра WALG\_DISK\_RATE\_LIMIT с другими параметрами:

- WALG\_NETWORK\_RATE\_LIMIT – ограничение скорости передачи данных по сети (дополняет WALG\_DISK\_RATE\_LIMIT);
- WALG\_UPLOAD\_CONCURRENCY – может влиять на общую производительность при ограничении дискового I/O (см. п.п. 2.2.7.2).

Журнал работы компонента содержит информационное поле «disk\_read\_rate» при активированном ограничении параметра WALG\_DISK\_RATE\_LIMIT.

## **2.2.7. Настройка ограничений сетевых операций**

### **2.2.7.1 Параметр WALG\_DOWNLOAD\_CONCURRENCY**

Параметр WALG\_DOWNLOAD\_CONCURRENCY определяет максимальное количество параллельных потоков загрузки данных из удаленного хранилища при выполнении операций восстановления.

Параметр WALG\_DOWNLOAD\_CONCURRENCY применяется исключительно для следующих операций:

- Восстановления из резервной копии (backup-fetch);
- Извлечения WAL-файлов (wal-fetch).

Параметр WALG\_DOWNLOAD\_CONCURRENCY действует для всех поддерживаемых типов хранилищ (SSH, файловая система и других).

Таблица 2.9 – Значения параметра WALG\_DOWNLOAD\_CONCURRENCY

Параметр	Возможные значения	Описание
WALG_DOWNLOAD_CONCURRENCY	"[число потоков]"	Максимальное количество параллельных потоков загрузки РК

Рекомендуемые значения параметра WALG\_DOWNLOAD\_CONCURRENCY: от 1 до 16.

По умолчанию значение параметра WALG\_DOWNLOAD\_CONCURRENCY: 10.

Влияние параметра WALG\_DOWNLOAD\_CONCURRENCY на работу компонента:

— При значениях  $> 1$ :

- Увеличивается скорость восстановления за счет параллельной загрузки РК;
- Возрастает сетевой трафик;
- Возрастает нагрузка на центральный процессор;
- Требуется увеличение доступной оперативной памяти.

— При значении 1:

- Данные РК загружаются из хранилища последовательно;
- Минимальное потребление ресурсов.

Взаимодействие параметра WALG\_DOWNLOAD\_CONCURRENCY с другими параметрами:

— WALG\_UPLOAD\_CONCURRENCY – аналогичный параметр для операций загрузки РК в хранилище (см. п.п. 2.2.7.2);

— WALG\_NETWORK\_RATE\_LIMIT – может ограничивать общую скорость передачи РК из/в хранилище.

Журнал работы компонента содержит информационное поле о количестве активных входящих потоков.



При некорректном значении параметра WALG\_UPLOAD\_CONCURRENCY (например, превышение возможностей хранилища) – отображается ошибка WAL-E-STORAGE-LIMIT.

### 2.2.7.2 Параметр WALG\_UPLOAD\_CONCURRENCY

Параметр WALG\_UPLOAD\_CONCURRENCY определяет максимальное количество параллельных потоков для загрузки данных в удалённое хранилище при выполнении операций резервного копирования.

Параметр WALG\_UPLOAD\_CONCURRENCY применяется исключительно для следующих операций:

- Создания резервных копий (backup-push);
- Отправки WAL-файлов (wal-push).

Параметр WALG\_UPLOAD\_CONCURRENCY действует для всех поддерживаемых типов хранилищ (SSH, файловая система и других).

Таблица 2.10 – Значения параметра WALG\_UPLOAD\_CONCURRENCY

Параметр	Возможные значения	Описание
WALG_UPLOAD_CONCURRENCY	"[число потоков]"	Максимальное количество параллельных потоков передачи данных

Допустимые значения параметра WALG\_UPLOAD\_CONCURRENCY:

- Положительное целое число (рекомендуемый диапазон 1-16);
- 1 (по умолчанию) – последовательная загрузка РК.

Влияние параметра WALG\_UPLOAD\_CONCURRENCY на работу компонента:

- При значениях > 1:
  - Увеличивается скорость загрузки РК в хранилище;
  - Возрастает сетевой трафик;
  - Возрастает нагрузка на центральный процессор;
  - Требуется увеличение доступной оперативной памяти.

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

— При значении 1:

- Данные РК загружаются в хранилище последовательно;
- Минимальное потребление ресурсов.

Взаимодействие параметра WALG\_UPLOAD\_CONCURRENCY с другими параметрами:

— WALG\_DOWNLOAD\_CONCURRENCY – аналогичный параметр для операций восстановления из РК (см. 2.2.7.1);

— WALG\_NETWORK\_RATE\_LIMIT – может ограничивать общую скорость передачи;

— WALG\_DISK\_RATE\_LIMIT - влияет на скорость чтения с диска (см. п.п. 2.2.6).

Журнал работы компонента содержит информационное поле о количестве активных исходящих потоков.

При некорректном значении параметра WALG\_UPLOAD\_CONCURRENCY (например, превышение возможностей хранилища) – отображается ошибка WAL-E-STORAGE-LIMIT.

### 2.2.7.3 Параметр WALG\_NETWORK\_RATE\_LIMIT

Параметр WALG\_NETWORK\_RATE\_LIMIT устанавливает ограничение максимальной скорости передачи данных по сети при выполнении операций резервного копирования и восстановления.

Параметр WALG\_NETWORK\_RATE\_LIMIT применяется для всех сетевых операций:

- Загрузки данных в хранилище (backup-push, wal-push);
- Выгрузки данных из хранилища (backup-fetch, wal-fetch).

Параметр WALG\_NETWORK\_RATE\_LIMIT действует для всех поддерживаемых типов хранилищ (SSH, файловая система и других).

Таблица 2.11 – Значения параметра WALG\_NETWORK\_RATE\_LIMIT

Параметр	Возможные значения	Описание
№ изменения: _____ Подпись отв. лица: _____ Дата внесения изм: _____		

WALG_NETWORK_RATE_LIMIT	"[целое число]" "0" (по умолчанию) – отсутствие ограничения скорости	Ограничение максимальной скорости передачи данных, килобит в секунду
-------------------------	---	--

Влияние параметра WALG\_NETWORK\_RATE\_LIMIT на работу компонента:

- При значениях  $> 0$ :
  - Ограничивает скорость передачи данных;
  - Снижает нагрузку на сетевой интерфейс;
  - Может увеличивать время выполнения операций восстановления из РК.
- При значении 0: используется максимально доступная скорость сети.

Взаимодействие параметра WALG\_NETWORK\_RATE\_LIMIT с другими параметрами:

- WALG\_UPLOAD\_CONCURRENCY – влияет на общую производительность (см. п.п. 2.2.7.2);
- WALG\_DOWNLOAD\_CONCURRENCY – аналогично для операций восстановления (см. п.п. 2.2.7.1);
- WALG\_DISK\_RATE\_LIMIT – может стать узким местом при высоких сетевых ограничениях (см. п.п. 2.2.6).

Журнал работы компонента содержит информационное поле о фактической скорости передачи данных.

При некорректном значении параметра WALG\_NETWORK\_RATE\_LIMIT (например, превышение возможностей локальной вычислительной сети) – отображается ошибка WAL-E-NETWORK-THROTTLE. Сбои сетевого соединения приводят к возникновению ошибки WAL-E-NETWORK-TIMEOUT

### 2.2.8. Включение метрик мониторинга

Для включения и настройки метрик мониторинга компонента применяются параметры:

- WALG\_STATSD\_ADDRESS;

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

— WALG\_STATSD\_EXTRA\_TAGS.

Для сбора метрик производительности рекомендуется к использованию инструмент StatD.

Таблица 2.12 – Значения параметров WALG\_STATSD\_\*

Параметр	Возможные значения	Описание
WALG_STATSD_ADDRESS		Адрес сервиса сбора значений метрик производительности
WALG_STATSD_EXTRA_TAGS	host, operation, database и т.д.	Выбор статических тегов к метрикам

### 2.2.9. Настройка параметров подключения к СУБД

Для определения параметров подключения к СУБД «Jatoba» используются параметры PGDATA и PGHOST.



Для авторизации и доступа компонента к БД должны быть обеспечены соответствующие разрешения в файле pg\_hba.conf.

Для роли пользователя, от лица которого будет выполняться создание РК, должно быть обеспечено разрешение на репликацию данных, например:

```
CREATE ROLE backup WITH REPLICATION LOGIN PASSWORD
'password';
GRANT EXECUTE ON FUNCTION pg_backup_start to backup;
GRANT EXECUTE ON FUNCTION pg_backup_stop to backup;
GRANT EXECUTE ON FUNCTION pg_switch_wal to backup;
GRANT pg_read_all_settings to backup;
```

#### 2.2.9.1 Настройка параметра PGDATA

Параметр PGDATA определяет путь к основному каталогу данных СУБД «Jatoba», содержащему все файлы базы данных, конфигурации и служебную информацию.

Каталог, определенный в параметре PGDATA, используется для операций:

- Резервного копирования через (backup-push);
- Восстановления данных (backup-fetch);
- Управления журналами WAL (wal-push, wal-fetch).

Таблица 2.13 – Значения параметра PGDATA

Параметр	Возможные значения	Описание
PGDATA	"/var/lib/jatoba/6/data/"	Путь к каталогу с СУБД



При отсутствии настроенного доступа к директории, указанной в параметре PGDATA, при создании РК будет отображаться ошибка Invalid PGDATA: directory not exist

Необходимо убедиться в наличии прав доступа к каталогу СУБД и переопределить при необходимости:

```
# chown -R postgres:postgres /var/lib/jatoba/6/data/
# chmod 700 /var/lib/jatoba/6/data/
```

### 2.2.9.2 Настройка параметра PGHOST

PGHOST определяет для подключения сетевой адрес (или сокет) сервера СУБД «Jatoba».

Таблица 2.14 – Значения параметра PGHOST

Параметр	Возможные значения	Описание
PGHOST	"/var/run/jatoba" "10.116.102.54"	IP-адрес, имя узла или сокет для подключения к СУБД

В случае использования параметра PGHOST сетевого имени узла оно должно успешно разрешаться DNS-сервером.

### 2.2.9.3 Настройка параметра PGPORT

Параметр PGPORT определяет TCP-порт для подключения к серверу СУБД «Jatoba» при выполнении операций резервного копирования.

По умолчанию для подключения к серверу СУБД «Jatoba» используется TCP-порт 5432. В случае если для СУБД «Jatoba» определен другой номер TCP-порта, его необходимо указать в параметре PGPORT.

Таблица 2.15 – Значения параметра PGPORT

Параметр	Возможные значения	Описание
PGPORT	"[номер_порта]"	TCP-порт для подключения к серверу СУБД «Jatoba»



Если указанный в параметре PGPORT TCP-порт для подключения к серверу СУБД «Jatoba» недоступен или указан ошибочно, то будет отображена ошибка:

```
ERROR: 2025/05/30 13:26:05.646915 Failed to connect
using provided PGHOST and PGPORT, trying lo-
calhost:5432
```

#### 2.2.9.4 Настройка параметра PGUSER

Параметр PGUSER определяет пользователя для аутентификации при выполнении операций резервного копирования.

Параметр PGUSER используется при:

- Авторизации при выполнении резервного копирования и создания РК;
- Проверки прав доступа к журналам WAL;
- Взаимодействии с системными функциями СУБД.

Таблица 2.16 – Значения параметров PGUSER

Параметр	Возможные значения	Описание
PGUSER	"[имя_пользователя]"	Определяет пользователя для аутентификации и создания РК



Если указанное в параметре PGUSER имя пользователя для подключения к серверу СУБД «Jatoba» указано ошибочно, то будет отображена ошибка:

```
ERROR: 2025/05/30 14:44:35.701169 failed to connect to
`user=postgrest database=`:
/var/run/jatoba/.s.PGSQL.5432 (/var/run/jatoba):
```

```
failed SASL auth: FATAL: password authentication
failed for user "postgres" (SQLSTATE 28P01)
```

### 2.2.9.5 Настройка параметра PGPASSWORD

Параметр PGPASSWORD определяет пароль для аутентификации пользователя при выполнении операций резервного копирования.

Таблица 2.17 – Значения параметров PGPASSWORD

Параметр	Возможные значения	Описание
PGPASSWORD	"[пароль]"	Определяет пароль пользователя для аутентификации и создания РК

### 2.2.10. Настройка защищенного соединения

WALG\_SSH\_PREFIX параметр, который позволяет определить параметры подключения при загрузке РК с использованием протокола SSH.

Таблица 2.18 – Значения параметров WALG\_SSH\_PREFIX

Параметр	Возможные значения	Описание
WALG_SSH_PREFIX	"ssh://[host-name]:/home/[username]/walg_archive/server"	Строка подключения к удаленному серверу с использованием протокола SSH
SSH_PORT	"22"	Номер порта, используемого при установлении соединения по протоколу SSH
SSH_USERNAME	"[username]"	Название пользователя, от имени которого устанавливается защищенное соединение на удаленном сервере
SSH_PASSWORD	"[password]"	Пароль пользователя, от имени которого устанавливается защищенное соединение на удаленном сервере

Параметр	Возможные значения	Описание
SSH_PRIVATE_KEY_PATH	"~/.ssh/private_key"	Путь к каталогу, в котором располагается закрытый ключ для установления защищенного соединения



С целью обеспечения достаточного уровня информационной безопасности и предотвращения передачи паролей в открытом виде рекомендуется к использованию параметр SSH\_PRIVATE\_KEY\_PATH вместо SSH\_PASSWORD.

Доступ пользователя, указанного в параметре SSH\_USERNAME, должен быть предварительно настроен:

- Настроен SSH-ключ без использования пароля;
- Пользователь должен иметь права на запись в целевой каталог, в котором будут сохраняться РК.

Для обеспечения требуемого уровня безопасности работы компонента с хранилищем в параметре WALG\_SSH\_PORT рекомендуется указывать номер сетевого порта, отличного от используемого по умолчанию (порт 22).

### 2.2.11. Параметры журналирования событий

Для определения параметров журналирования событий резервного копирования/восстановления СУБД «Jatoba» используются параметры WALG\_LOG\_LEVEL и S3\_LOG\_LEVEL=DEVEL.

Таблица 2.19 – Значения параметров WALG\_LOG\_LEVEL и S3\_LOG\_LEVEL

Параметр	Возможные значения	Описание
WALG_LOG_LEVEL	DEVEL, ERROR, INFO	Включение режима журналирования
S3_LOG_LEVEL	DEVEL, ERROR, INFO	Включение режима журналирования с S3



## 2.3. Настройка СУБД

### 2.3.1. Создание пользователя для выполнения архивации и восстановления данных

Для использования компонента «wal-g» необходимо создать пользователя СУБД backup с атрибутом Login и придерживаться принципа минимизации назначаемых атрибутов и привилегий.

Назначение дополнительных атрибутов роли, системных привилегий и прав осуществляется в соответствии с выбранным типом и режимом резервного копирования.

### 2.3.2. Настройка непрерывного архивирования WAL

Для настройки непрерывного архивирования WAL необходимо выполнить следующие действия:

- 1) открыть для редактирования конфигурационный файл postgresql.conf;
- 2) в разделе «Write-Ahead Log» – «Settings» задать для параметра wal\_level значение выше replica или logical;

```

root@node1: /home/admin1
GNU nano 6.2 /var/lib/jatoba/6/data/postgresql.conf *
#-----
# WRITE-AHEAD LOG
#-----
# - Settings -
wal_level = replica           # minimal, replica, or logical
                              # (change requires restart)
#fsync = on                  # flush data to disk for crash safety
                              # (turning this off can cause
                              # unrecoverable data corruption)
#synchronous_commit = on    # synchronization level;

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute   ^C Location  M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify   ^/_ Go To Line M-E Redo
  
```

Рисунок 2.1 – Внесение параметра wal\_level в конфигурационный файл postgresql.conf в GNU Linux



При настройке резервного копирования на ведущем сервере, параметр archive\_mode должен иметь значение on или always.

Для выполнения резервного копирования на ведомом сервере параметр archive\_mode должен иметь значение always.

- 3) В разделе «Archiving» установить параметр:

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

```
archive_mode = on
```

- 4) В разделе «Archiving» установить параметр `archive_command`:

```
archive_command = '/usr/jatoba-6/bin/wal-g wal-push %p'
```

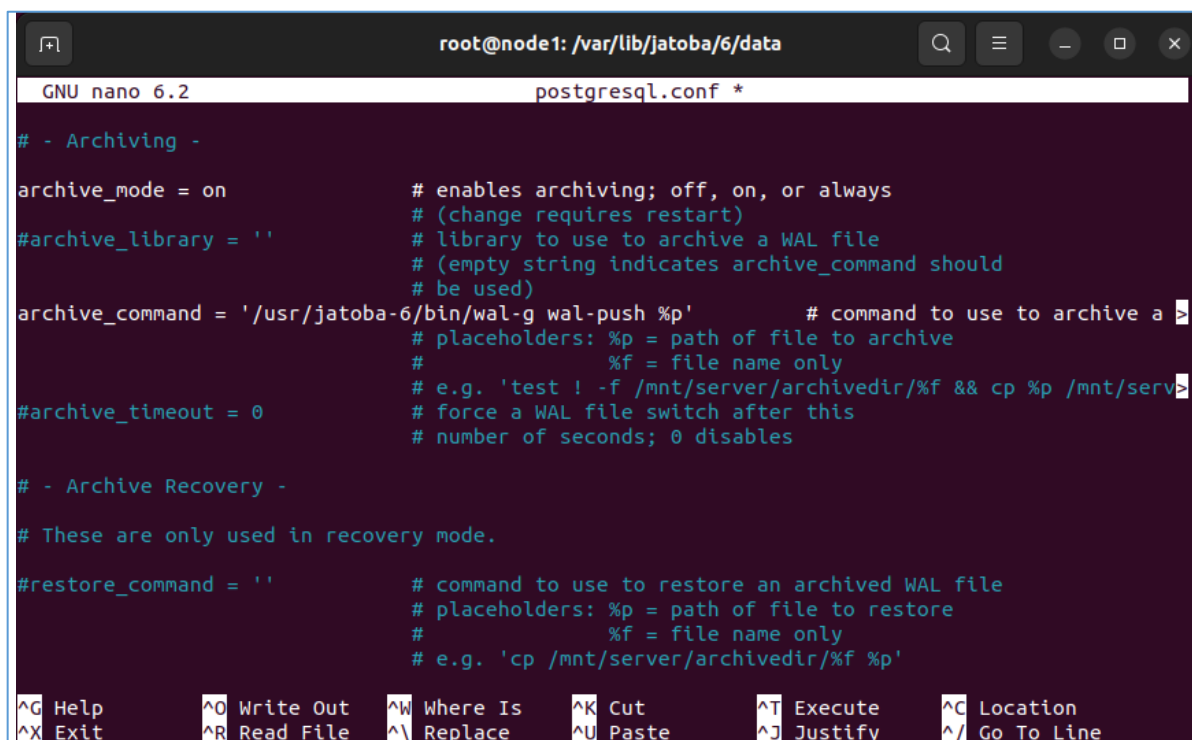


Рисунок 2.2 – Вид конфигурационного файла `postgresql.conf` с внесенной строчкой параметра `archive_command` в GNU Linux

- 5) В разделе «Archive Recovery» установить параметр `restore_command`:

```
restore_command = '/usr/jatoba-6/bin/wal-g wal-fetch %f %p'
```

- 6) В разделе «Recovery Target» установить параметр `recovery_target_action`:

```
recovery_target_action = 'promote'
```

- 7) Для вступления параметров конфигурационного файла `postgresql.conf` выполнить перезагрузку СУБД «Jatoba»:

```
# systemctl restart jatoba-6
```

После выполненных подготовительных действий при помощи компонента возможно создавать РК (см. п.п. 3.1).

Текущее состояние архива WAL можно посмотреть, воспользовавшись командой:

```
postgres@node1:~$ wal-g wal-show
```

### 3. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ КОМПОНЕНТА

Компонент предоставляет следующие функциональные возможности при взаимодействии с РК:

- Создание РК:
  - Полной;
  - Инкрементальной;
  - Сжатие данных РК (LZ4, ZSTD, Brotli).
- Отображение списка доступных РК;
- Восстановление данных из РК:
  - Полное восстановление;
  - Point-in-Time Recovery (PITR);
  - Восстановление WAL-журналов.
- Удаление:
  - Удаление устаревших копий;
  - Точечное удаление;
  - Полная очистка хранилища.
- Журналирование выполняемых операций.



Работа с командами компонента должна выполняться от имени пользователя postgres.

Исполняемый файл компонента расположен в каталоге /usr/jatoba-6/bin

#### 3.1. Создание резервной копии

При выполнении создания РК, в качестве аргумента пользователь должен передать расположение каталога данных СУБД «Jatoba».

Предварительные условия: выполнена настройка СУБД согласно п. 2.3.

##### 3.1.1. Создание полной резервной копии

Для обеспечения отказоустойчивости рекомендуется:

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

- Регулярно проверять целостность полных РК;
- Хранить несколько поколений полных РК.

Синтаксис команды создания новой полной РК выглядит следующим образом:

```
wal-g backup-push --full $PGDATA
```

где `$PGDATA` – путь к рабочему каталогу СУБД «Jatoba».



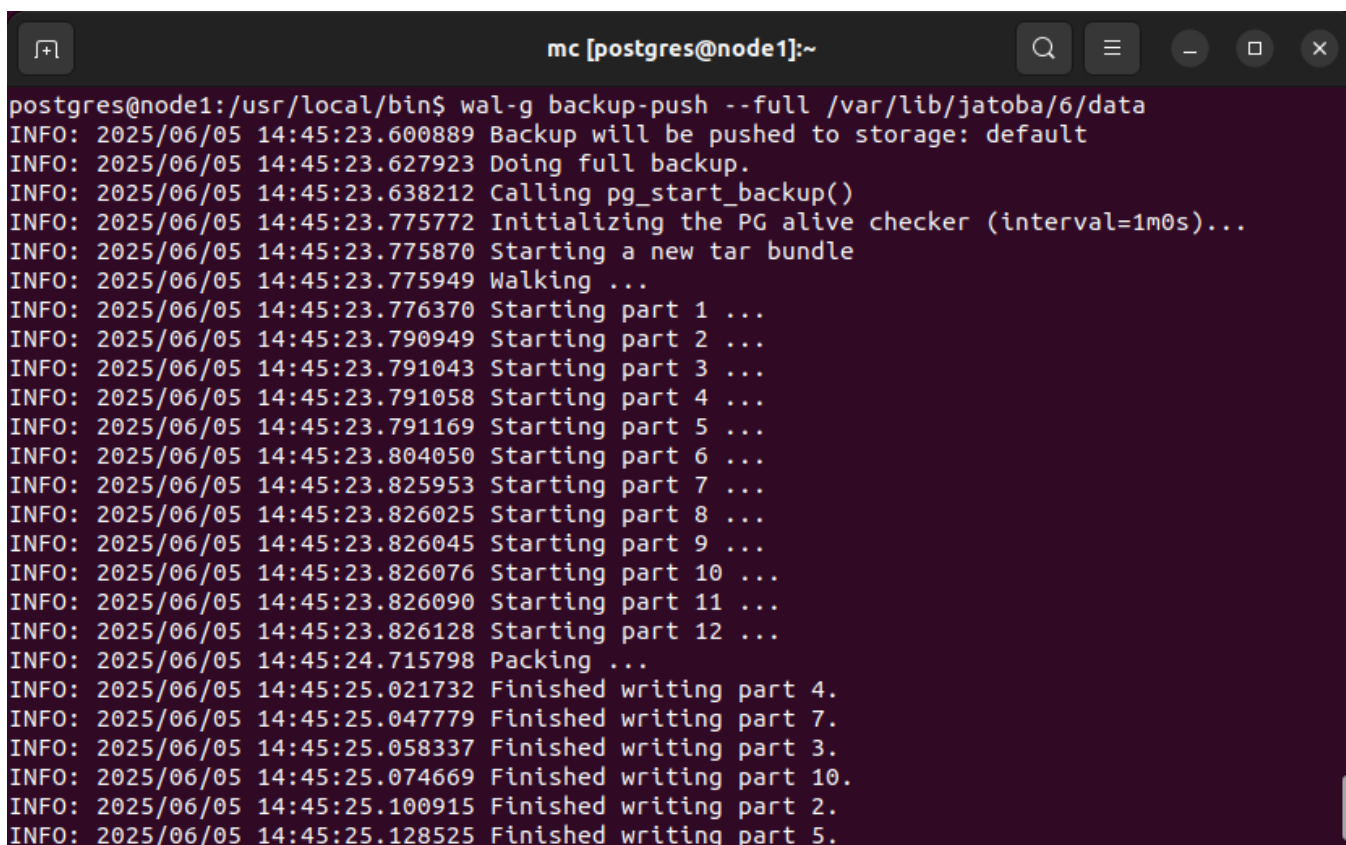
Для дальнейшего использования переменной окружения `$PGDATA` необходимо выполнить команду:

```
export PGDATA=/var/lib/jatoba/6/data
```

Здесь `/var/lib/jatoba/6/data` – путь к каталогу данных СУБД по умолчанию и, в зависимости от настроек СУБД, в некоторых случаях может отличаться. Здесь и далее при описании синтаксиса команд применяется `$PGDATA`, а для практических примеров – путь к местоположению каталога данных СУБД.

### Пример:

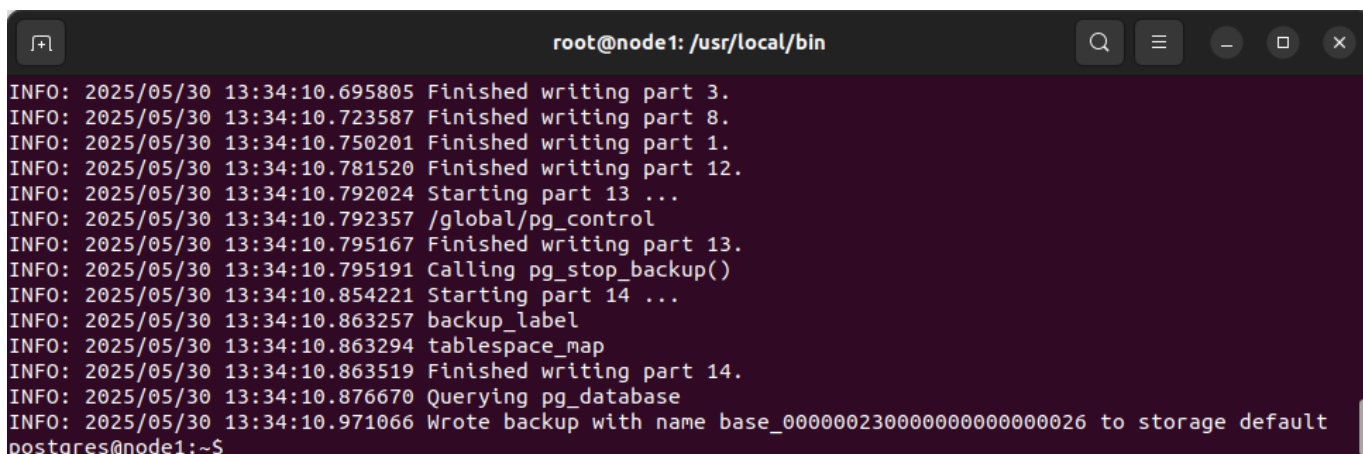
```
postgres@node1:~$ wal-g backup-push --full  
/var/lib/jatoba/6/data
```



```
mc [postgres@node1]:~  
postgres@node1:/usr/local/bin$ wal-g backup-push --full /var/lib/jatoba/6/data  
INFO: 2025/06/05 14:45:23.600889 Backup will be pushed to storage: default  
INFO: 2025/06/05 14:45:23.627923 Doing full backup.  
INFO: 2025/06/05 14:45:23.638212 Calling pg_start_backup()  
INFO: 2025/06/05 14:45:23.775772 Initializing the PG alive checker (interval=1m0s)...  
INFO: 2025/06/05 14:45:23.775870 Starting a new tar bundle  
INFO: 2025/06/05 14:45:23.775949 Walking ...  
INFO: 2025/06/05 14:45:23.776370 Starting part 1 ...  
INFO: 2025/06/05 14:45:23.790949 Starting part 2 ...  
INFO: 2025/06/05 14:45:23.791043 Starting part 3 ...  
INFO: 2025/06/05 14:45:23.791058 Starting part 4 ...  
INFO: 2025/06/05 14:45:23.791169 Starting part 5 ...  
INFO: 2025/06/05 14:45:23.804050 Starting part 6 ...  
INFO: 2025/06/05 14:45:23.825953 Starting part 7 ...  
INFO: 2025/06/05 14:45:23.826025 Starting part 8 ...  
INFO: 2025/06/05 14:45:23.826045 Starting part 9 ...  
INFO: 2025/06/05 14:45:23.826076 Starting part 10 ...  
INFO: 2025/06/05 14:45:23.826090 Starting part 11 ...  
INFO: 2025/06/05 14:45:23.826128 Starting part 12 ...  
INFO: 2025/06/05 14:45:24.715798 Packing ...  
INFO: 2025/06/05 14:45:25.021732 Finished writing part 4.  
INFO: 2025/06/05 14:45:25.047779 Finished writing part 7.  
INFO: 2025/06/05 14:45:25.058337 Finished writing part 3.  
INFO: 2025/06/05 14:45:25.074669 Finished writing part 10.  
INFO: 2025/06/05 14:45:25.100915 Finished writing part 2.  
INFO: 2025/06/05 14:45:25.128525 Finished writing part 5.
```

Рисунок 3.1 – Процесс выполнения команды wal-g backup-push --full

Длительность создания полной РК зависит от объема данных, доступных технических ресурсов, а также ограничений, указанных в конфигурационном файле wal-g.json (см. п.п. 2.2).



```
root@node1: /usr/local/bin  
INFO: 2025/05/30 13:34:10.695805 Finished writing part 3.  
INFO: 2025/05/30 13:34:10.723587 Finished writing part 8.  
INFO: 2025/05/30 13:34:10.750201 Finished writing part 1.  
INFO: 2025/05/30 13:34:10.781520 Finished writing part 12.  
INFO: 2025/05/30 13:34:10.792024 Starting part 13 ...  
INFO: 2025/05/30 13:34:10.792357 /global/pg_control  
INFO: 2025/05/30 13:34:10.795167 Finished writing part 13.  
INFO: 2025/05/30 13:34:10.795191 Calling pg_stop_backup()  
INFO: 2025/05/30 13:34:10.854221 Starting part 14 ...  
INFO: 2025/05/30 13:34:10.863257 backup_label  
INFO: 2025/05/30 13:34:10.863294 tablespace_map  
INFO: 2025/05/30 13:34:10.863519 Finished writing part 14.  
INFO: 2025/05/30 13:34:10.876670 Querying pg_database  
INFO: 2025/05/30 13:34:10.971066 Wrote backup with name base_00000023000000000000000026 to storage default  
postgres@node1:~$
```

Рисунок 3.2 – Созданная полная РК при помощи команды wal-g backup-push --full

При создании РК компонент будет проверять, что аргумент команды, переменная PGDATA в окружении ОС и настройка конфигурации в файле .wal-g.json совпадают. В ином случае создание РК будет прервано, а результат будет записан в журнал выполнения.

### 3.1.2. Создание постоянной резервной копии

Постоянная РК обеспечивает:

- Защиту критических РК от автоматического удаления;
- Ручной контроль за архивацией ключевых точек восстановления;
- Соблюдение регламентов хранения РК.

Постоянная РК не удаляется при использовании команд управления жизненным циклом РК, например, `wal-g delete retain` (см. п.п. 3.6.1) и `wal-g delete before` (см. п.п. 3.6.2) (Исключение использование опции `FORCE` с указанными командами).

Постоянную РК можно создать при помощи:

- Использования флага `--permanent` в команде `wal-g backup-push`;
- Использования команды `backup-mark`.

Синтаксис команды создания новой постоянной РК выглядит следующим образом:

```
wal-g backup-push --full --permanent $PGDATA
```

Где флаг `--permanent` необходим при создании РК с свойством «постоянная»; `$PGDATA` – путь к рабочему каталогу СУБД «Jatoba». Данное свойство предотвратит удаление РК при выполнении команды `delete` в режимах `retain` и `before` (см. п.п. 3.6).

#### Пример:

```
postgres@node1:~$ wal-g backup-push --full --permanent  
/var/lib/jatoba/6/data
```

Команда `backup-mark` изменяет значение признака «`is_permanent`» для существующей РК. Синтаксис команды для изменения свойства «`is_permanent`» РК выглядит следующим образом:

```
wal-g backup-mark [base_name]
```

Где `base_name` – название РК.

#### Пример:

```
postgres@node1:~$ wal-g backup-mark
base_00000023000000000000000026
```

```
mc [postgres@node1:~$ wal-g backup-mark base_00000023000000000000000026
INFO: 2025/06/04 18:52:18.411765 Retrieving previous related backups to be marked: toPermanent=true
INFO: 2025/06/04 18:52:18.414166 Retrieved backups to be marked, marking: [base_00000023000000000000000026]
postgres@node1:~$
```

Рисунок 3.3 – Создание постоянной РК при помощи команды wal-g backup-mark

После этого необходимо проверить значение признака «is\_permanent» при помощи команды backup-list с флагом --detail:

```
postgres@node1:~$ wal-g backup-list --detail
```

```
mc [postgres@node1:~$ wal-g backup-list --detail
INFO: 2025/06/04 18:57:02.405440 List backups from storages: [default]
backup_name      finish_time      hostname data_dir      modified      pg_version start_lsn finish_lsn is permanent storage_name start_time
base_00000023000000000000000026 2025-05-30T13:34:10+03:00 00000023000000000000000026 default 2025-05-30 00000023 0/26000178 true default 2025-05-30
T10:33:57Z 2025-05-30T10:34:10Z node1 /var/lib/jatoba/6/data 160006 0/26000028 0/26000178 false default 2025-06-02
T15:05:31Z 2025-06-02T15:05:33Z node1 /var/lib/jatoba/6/data 160006 0/2D000028 0/2D000178 false default 2025-06-03
base_00000025000000000000000035_D_0000002500000000000000002D 2025-06-03T11:52:13+03:00 000000250000000000000035 default 2025-06-03
T08:52:12Z 2025-06-03T08:52:13Z node1 /var/lib/jatoba/6/data 160006 0/35000028 0/35000130 false default 2025-06-04
base_00000025000000000000000038_D_00000025000000000000000038 2025-06-04T17:46:06+03:00 000000250000000000000038 default 2025-06-04
T14:46:05Z 2025-06-04T14:46:06Z node1 /var/lib/jatoba/6/data 160006 0/38000028 0/38000178 false default 2025-06-04
base_0000002500000000000000003A 2025-06-04T17:46:44+03:00 00000025000000000000003A default 2025-06-04
```

Рисунок 3.4 – Список созданных РК (постоянная РК отмечена признаком is\_permanent = true)

Узнать текущее количество постоянных РК можно при помощи следующей команды:

```
postgres@node1:~$ wal-g backup-list --detail | grep -c 'is_per-
manent'
```

```
root@node1: /home/admin1
postgres@node1:/usr/jatoba-6/bin$ ./wal-g backup-list --detail | grep -c 'is_permanent'
1
postgres@node1:/usr/jatoba-6/bin$
```

Рисунок 3.5 – Просмотр количества постоянных РК

Для того чтобы убрать признак is\_permanent, которым отмечаются постоянные РК, необходимо выполнить команду с флагом -i:

```
postgres@node1:~$ wal-g backup-mark -i
base_00000023000000000000000026
```



### 3.1.3. Создание инкрементальной резервной копии

Инкрементальные РК содержат изменения относительно полной или другой инкрементальной РК, поэтому:

- Являются производными от выбранной РК;
- Не содержат полного набора данных;
- Требуют доступа к неповреждённой РК для корректного восстановления данных.



Если значение параметра `WALG_DELTA_MAX_STEPS > 0`, то в этом случае при создании инкрементальной РК используется последняя РК в качестве базы. Это поведение можно изменить с помощью следующих флагов:

— Флаг «--delta-from-name» или переменная окружения `WALG_DELTA_FROM_NAME` для выбора РК с указанным именем в качестве основы для инкрементальной РК;

— Флаг «--delta-from-user-data» переменная окружения `WALG_DELTA_FROM_USER_DATA` для выбора РК с указанными пользовательскими данными в качестве основы для инкрементальной РК.

Создание инкрементальной РК выполняется в следующей последовательности:

- 1) В первую очередь необходимо предварительно создать полную РК с флагом «-full» (см. п.п. 3.1.1).
- 2) Затем выполнить создание инкрементальной РК при помощи команды:

```
wal-g backup-push $PGDATA
```

**Пример:**

```
postgres@node1:~$ wal-g backup-push /var/lib/jatoba/6/data
```

```
mc [postgres@node1]:~
INFO: 2025/06/02 18:05:33.375037 Finished writing part 7.
INFO: 2025/06/02 18:05:33.399091 Finished writing part 12.
INFO: 2025/06/02 18:05:33.558557 Finished writing part 10.
INFO: 2025/06/02 18:05:33.593340 Finished writing part 8.
INFO: 2025/06/02 18:05:33.609907 Finished writing part 1.
INFO: 2025/06/02 18:05:33.621444 Starting part 13 ...
INFO: 2025/06/02 18:05:33.621514 /global/pg_control
INFO: 2025/06/02 18:05:33.624399 Finished writing part 13.
INFO: 2025/06/02 18:05:33.624434 Calling pg_stop_backup()
INFO: 2025/06/02 18:05:33.678952 Starting part 14 ...
INFO: 2025/06/02 18:05:33.686739 backup_label
INFO: 2025/06/02 18:05:33.686776 tablespace_map
INFO: 2025/06/02 18:05:33.686836 Finished writing part 14.
INFO: 2025/06/02 18:05:33.698111 Querying pg_database
INFO: 2025/06/02 18:05:33.835217 Wrote backup with name base_0000002300000000000000002D_D_000000230000000000000026
to storage default
postgres@node1:/usr/local/bin$
```

Рисунок 3.6 – Созданная инкрементальная РК при помощи команды wal-g backup-push

3) Проверить наличие созданной инкрементальной РК (см. п.п. 0).



Если выполнить команду backup-push без использования флагов, то будет создана полная РК, в случае если:

- Это первая созданная РК;
- Количество инкрементальных РК, созданных для последней полной, равно WALG\_DELTA\_MAX\_STEPS (см. п.п. 2.2.4).

Название инкрементальной РК состоит из двух частей:

- первая часть указывает на идентификатор РК, на основании которой создана инкрементальная РК. В качестве родительской РК при этом может выступать как полная, так и другая инкрементальная РК;
- вторая часть – это уникальное название созданной РК.

base\_00000023000000000000000026\_D\_00000023000000000000004C

	идентификатор		идентификатор	
	родительской РК		созданной РК	

При использовании инкрементальных РК имеются следующие функциональные ограничения:

- Процесс восстановления из инкрементальной РК требует сборки цепочки данных из инкрементальных РК, что может быть медленнее, чем восстановление из полной РК.

Чем большее число указано для параметра WALG\_DELTA\_MAX\_STEPS (см. п.п. 2.2.4), тем дольше выполняется процесс восстановления данных;

— Целостность инкрементальных резервных копий напрямую зависит от сохранности, соответствующей полной РК. В случае повреждения или утраты базовой полной копии все связанные с ней инкрементальные резервные копии теряют функциональность и не могут быть использованы для восстановления данных.

### 3.1.4. Создание инкрементальной резервной копии на основе базовой РК

При создании дельта бэкапа (при значении WALG\_DELTA\_MAX\_STEPS > 0), компонент по умолчанию использует последнюю РК как базовую.

Это поведение может быть изменено с помощью следующих флагов:

— `--delta-from-name` (или `WALG_DELTA_FROM_NAME` в конфигурационном файле `walg.json`) – определяет название РК, которая будет использована в качестве базовой для инкрементальной РК;

— `--delta-from-user-data` (или `WALG_DELTA_FROM_USER_DATA` в конфигурационном файле `walg.json`) – определяет `user-data` РК, которая будет использована в качестве базовой для инкрементальной РК.

### Примеры:

```
postgres@node1:~$ wal-g backup-push /var/lib/jatoba/6/data --  
delta-from-name  
base_0000000100000000000000006_D_000000010000000000000004  
postgres@node1:~$ wal-g backup-push /var/lib/jatoba/6/data --  
delta-from-user-data "{ \"x\": [3], \"y\": 4 }"
```

```

root@node1: /home/admin1
INFO: 2025/06/17 15:44:38.254266 List backups from storages: [default]
INFO: 2025/06/17 15:45:34.292260 Backup will be pushed to storage: default
INFO: 2025/06/17 15:45:34.292521 Selecting the backup with name base_000000010000000000000006_D_000000010000000000000004 as the
base for the current delta backup...
INFO: 2025/06/17 15:45:34.323737 Delta backup from base_000000010000000000000006_D_000000010000000000000004 with LSN 0/6000028.
INFO: 2025/06/17 15:45:34.350301 Calling pg_start_backup()
INFO: 2025/06/17 15:45:34.487419 Initializing the PG alive checker (interval=1m0s)...
INFO: 2025/06/17 15:45:34.487620 Delta backup enabled
INFO: 2025/06/17 15:45:34.488340 Starting a new tar bundle
INFO: 2025/06/17 15:45:34.488574 Walking ...
INFO: 2025/06/17 15:45:34.489340 Starting part 1 ...
INFO: 2025/06/17 15:45:34.515067 Packing ...
INFO: 2025/06/17 15:45:34.515260 Finished writing part 1.
INFO: 2025/06/17 15:45:34.518825 Starting part 2 ...
INFO: 2025/06/17 15:45:34.518873 /global/pg_control
INFO: 2025/06/17 15:45:34.526025 Finished writing part 2.
INFO: 2025/06/17 15:45:34.526059 Calling pg_stop_backup()
INFO: 2025/06/17 15:45:34.542288 Starting part 3 ...
INFO: 2025/06/17 15:45:34.545352 backup_label
INFO: 2025/06/17 15:45:34.545373 tablespace_map
INFO: 2025/06/17 15:45:34.545566 Finished writing part 3.
INFO: 2025/06/17 15:45:34.549539 Querying pg_database
INFO: 2025/06/17 15:45:34.627107 Wrote backup with name base_00000001000000000000000A_D_000000010000000000000006 to storage default
postgres@node1: /usr/jatoba-6/bin$

```

Рисунок 3.7 – Результат выполнения wal-g backup-push с флагом --delta-from-name

### 3.1.5. Создание резервной копии с удаленного узла

Компонент wal-g поддерживает создание резервных копий (РК) с удаленных узлов СУБД.

Параметры подключения к удаленному узлу могут быть указаны в конфигурационном файле walg.json (см. п. 2.2) или как дополнительные параметры в командной строке:

1) В конфигурационном файле walg.json указываются следующие параметры подключения к СУБД, расположенной на удаленном узле:

- PGHOST – обязательный параметр;
- PGPASSWORD, PGPORT, PGUSER, PGDATABASE – опциональные параметры.



Пользователь postgres должен иметь права доступа к каталогу, который указан в параметре WALG\_FILE\_PREFIX:

```
chown -R postgres:postgres [/path/to/backups]
```

Для создания РК с использованием конфигурационного файла выполните:

```
wal-g backup-push --config /var/lib/jatoba/.walg.json
```

Для каждого из удаленных узлов необходимо создать свой конфигурационный файл walg.json с параметрами подключения, например:

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

```
wal-g backup-push --config  
/var/lib/jatoba/.walg_10_116_102_54.json
```

2) Также можно задать напрямую в команде backup-push:

```
wal-g backup-push --pghost [IP-адрес] --pgdatabase [db_name] --  
pgport 5432 --pguser postgres --pgpassword [password] --walg-  
file-prefix [/path/to/backups]
```

**Например:**

```
wal-g backup-push --pghost 192.168.223.161 --pgdatabase post-  
gres --pgport 5432 --pguser postgres --pgpassword 123456Aa --  
walg-file-prefix /nfs/wal/backups
```

Создание РК с удаленных узлов также поддерживается через защищенные соединения, например, по протоколу SSH.

Для этого на узле, на котором установлен компонент wal-g, выполняется создание ключа SSH, с помощью которого будет выполняться подключение к удаленному узлу:

```
su - postgres  
ssh-keygen -t rsa -b 4096 -f /var/lib/jatoba/.ssh/id_rsa
```

Далее необходимо скопировать созданный ключ SSH на удаленный узел при помощи команды с синтаксисом:

```
ssh-copy-id [имя пользователя]@[ip-адрес]
```

**Пример:**

```
ssh-copy-id postgres@192.168.223.161
```

После этого в конфигурационном файле компонента wal-g необходимо определить следующие параметры:

```
"WALG_SSH_PREFIX" : "ssh://postgres@<IP-адрес>/nfs/wal/back-  
ups/",  
"SSH_USERNAME" : "postgres",
```

```
"SSH_PASSWORD" : "[password]",  
"SSH_PRIVATE_KEY_PATH" : "/var/lib/jatoba/.ssh/id_rsa"
```

Восстановление из РК выполняется также как это указано в п. 3.5.

### 3.2. Просмотр созданных резервных копий

Для отображения списка названий и времени создания РК необходимо выполнить команду:

```
wal-g backup-list [--pretty] [--json] [--detail]
```

#### Пример:

```
postgres@node1:~$ wal-g backup-list
```

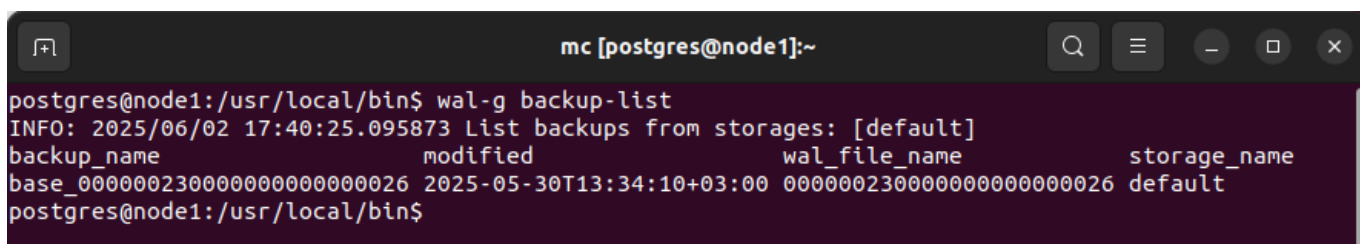


Рисунок 3.8 – Список созданных РК

Список созданных РК по умолчанию содержит следующую информацию:

- backup\_name – название файла РК;
- modified – дата и время изменения РК;
- wal\_file\_name – название файла WAL;
- storage\_name – название хранилища РК.

Таблица 3.1 – Дополнительные ключи команды backup-list

Ключ	Описание
--pretty	Отображение списка РК в табличном виде
--json	Отображение списка РК в формате JSON
--detail	Отображение дополнительной информации о созданных РК

```
mc [postgres@node1]:~
postgres@node1:/usr/local/bin$ wal-g backup-list --pretty
INFO: 2025/06/02 17:44:38.994402 List backups from storages: [default]
+-----+-----+-----+-----+
| # | BACKUP NAME | MODIFIED | WAL FILE NAME | STORAGE NAME |
+-----+-----+-----+-----+
| 0 | base_0000002300000000000000026 | Friday, 30-May-25 13:34:10 MSK | 0000002300000000000000026 | default |
+-----+-----+-----+-----+
```

Рисунок 3.9 – Список созданных РК в табличном виде при использовании флага --pretty

При использовании флага --detail список РК содержит следующую информацию:

- backup\_name – название файла РК;
- modified – дата и время изменения РК;
- wal\_file\_name – название файла WAL;
- storage\_name – название хранилища РК;
- start\_time – дата и время начала создания РК;
- finish\_time – дата и время завершения создания РК;
- hostname – название сервера, на котором создана РК;
- data\_dir – путь к каталогу БД;
- pg\_version – версия СУБД;
- start\_lsn – позиция «головы» файла WAL;
- finish\_lsn – позиция «хвоста» файла WAL;
- is\_permanent – признак постоянной РК (true, false).

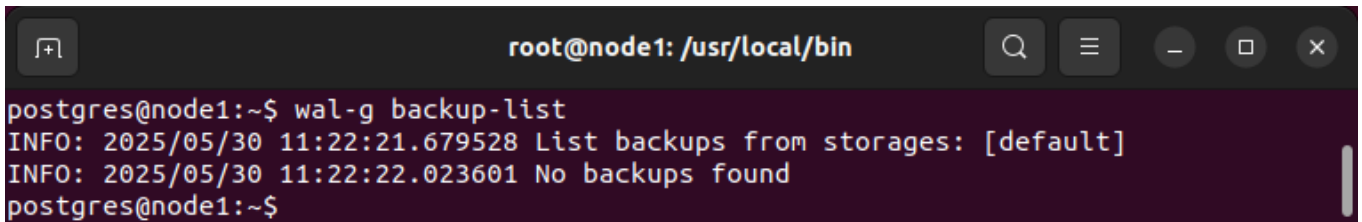
Список созданных РК может быть представлен в формате данных JSON при помощи флага --json.

```
postgres@node1:~$ wal-g backup-list --json
```

```
mc [postgres@node1]:~
postgres@node1:/usr/local/bin$ wal-g backup-list --json
INFO: 2025/06/02 18:18:20.952115 List backups from storages: [default]
[{"backup_name": "base_0000002300000000000000026", "time": "2025-05-30T13:34:10.965780666+03:00", "wal_file_name": "0000002300000000000000026", "storage_name": "default"}, {"backup_name": "base_000000250000000000000002D_D_0000002300000000000000026", "time": "2025-06-02T18:05:33.818890419+03:00", "wal_file_name": "000000250000000000000002D", "storage_name": "default"}]
```

Рисунок 3.10 – Список созданных РК в виде данных JSON при использовании флага --json

При отсутствии в назначенном каталоге (см. п.п. 2.2.2) созданных РК (полных или инкрементальных) будет выведено сообщение No backups found.



```
root@node1: /usr/local/bin
postgres@node1:~$ wal-g backup-list
INFO: 2025/05/30 11:22:21.679528 List backups from storages: [default]
INFO: 2025/05/30 11:22:22.023601 No backups found
postgres@node1:~$
```

Рисунок 3.11 – Сообщение компонента при отсутствии в хранилище созданных РК

### 3.3. Проверка созданных резервных копий

Функционал компонента позволяет выполнять проверку корректности созданных РК для того, чтобы убедиться в том, что хранилище сегментов WAL находится в рабочем состоянии.

Доступные методы проверки определяются с помощью аргументов:

- integrity – на основании последовательности сегментов;
- timeline – на основании временной шкалы.

Аргументы можно использовать вместе или по отдельности.

#### 3.3.1. Использование аргумента integrity

Аргумент integrity проверяет, что существует последовательная история сегментов WAL в диапазоне [старейший начальный сегмент - текущий сегмент] доступны в хранилище WAL. Это условие требуется для успешного выполнения PITR.

Команда проверки с использованием аргумента integrity:

```
postgres@node1:~$ wal-g wal-verify integrity
```



```

root@node1: /usr/local/bin
postgres@node1:~$ wal-g wal-verify integrity
INFO: 2025/08/13 15:59:28.347806 Current WAL segment: 00000025000000000000007A
INFO: 2025/08/13 15:59:28.347967 Building check runner: integrity
INFO: 2025/08/13 15:59:28.348385 Detected earliest available backup: base_000000250000000000000077
INFO: 2025/08/13 15:59:28.348396 Running the check: integrity
[wal-verify] integrity check status: WARNING
[wal-verify] integrity check details:
+-----+-----+-----+-----+-----+
| TLI | START | END | SEGMENTS COUNT | STATUS |
+-----+-----+-----+-----+-----+
| 37 | 000000250000000000000077 | 000000250000000000000079 | 3 | MISSING_UPLOADING |
+-----+-----+-----+-----+-----+
postgres@node1:~$

```

Рисунок 3.12 – Проверка созданных РК при помощи wal-verify integrity

Вывод команды wal-verify integrity состоит из:

1) Статуса проверки РК:

— OK – нет отсутствующих сегментов;

— WARNING – есть некоторые отсутствующие сегменты, но они не отмечены статусом MISSING\_LOST;

— FAILURE – есть некоторые сегменты, отмеченные статусом MISSING\_LOST.

2) Список, который содержит сегменты WAL в хронологическом порядке, сгруппированные по временной шкале и статусу.

В выводе проверки целостности, существуют четыре статуса сегментов WAL:

— FOUND – сегменты присутствуют в хранилище WAL;

— MISSING\_DELAYED – сегменты отсутствуют в хранилище WAL, но в СУБД не выполнялась их архивация с помощью archive\_command (см. п.п. 2.3);

— MISSING\_UPLOADING – сегменты, которые отсутствуют в хранилище WAL, но находятся в процессе загрузки в хранилище;

— MISSING\_LOST – некоторые сегменты отсутствуют в хранилище WAL и не соответствуют другим статусам проверки.

Ограничение на размер диапазона сегментов (ProbablyUploading) определяется параметром WALG\_UPLOAD\_CONCURRENCY.

Размер диапазона сегментов (ProbablyDelayed) регулируется с помощью параметра WALG\_INTEGRITY\_MAX\_DELAYED\_WALS.

### 3.3.2. Использование аргумента `timeline`

Аргумент `timeline` проверяет, что текущая временная шкала кластера больше или равна любой из временных шкал сегментов WAL хранилища.



Стоит обратить внимание на то, что проверка с использованием аргумента `timeline` работает корректно только в том случае, если создано новое хранилище WAL или существующее хранилище очищено при восстановлении из РК или выполнении `pg_upgrade`.

Команда проверки с использованием аргумента `timeline`:

```
postgres@node1:~$ wal-g wal-verify timeline
```

```
root@node1: /usr/local/bin
postgres@node1:~$ wal-g wal-verify timeline
INFO: 2025/08/13 15:21:59.030544 Current WAL segment: 0000002500000000000000007A
INFO: 2025/08/13 15:21:59.030793 Building check runner: timeline
INFO: 2025/08/13 15:21:59.030813 Running the check: timeline
[wal-verify] timeline check status: WARNING
[wal-verify] timeline check details:
Highest timeline found in storage: 0
Current cluster timeline: 37
postgres@node1:~$
```

Рисунок 3.13 – Проверка созданных РК при помощи `wal-verify timeline`

Вывод команды `wal-verify timeline` состоит из:

1) Статус проверки:

— OK – если текущий идентификатор временной шкалы совпадает с наивысшим идентификатором временной шкалы, найденным в хранилище WAL;

— WARNING – если не удалось определить, совпадает ли текущая временная шкала с самой высокой в хранилище WAL;

— FAILURE – если текущий идентификатор временной шкалы не равен наивысшему идентификатору временной шкалы, найденному в хранилище WAL.

2) Текущий идентификатор временной шкалы.

3) Самый высокий идентификатор временной шкалы, найденный в каталоге хранения WAL.

По умолчанию вывод команды wal-verify осуществляется обычным текстом. Чтобы включить вывод в формате JSON, необходимо использовать флаг «--json»:

```
postgres@node1:~$ wal-g wal-verify timeline --json
```

```
root@node1: /usr/local/bin
postgres@node1:~$ wal-g wal-verify timeline --json
INFO: 2025/08/13 16:03:56.491870 Current WAL segment: 0000002500000000000000007A
INFO: 2025/08/13 16:03:56.492042 Building check runner: timeline
INFO: 2025/08/13 16:03:56.493455 Running the check: timeline
{"timeline":{"status":"WARNING","details":{"current_timeline_id":37,"highest_storage_timeline_id":0}}}pop
postgres@node1:~$
```

Рисунок 3.14 – Проверка созданных РК при помощи wal-verify timeline в формате данных JSON

### 3.3.3. Использование обоих аргументов

Использование обоих аргументов позволяет отобразить одновременно всю сводную информацию о состоянии созданных РК.

В этом случае команда выглядит следующим образом:

```
postgres@node1:~$ wal-g wal-verify integrity timeline
```

```
root@node1: /usr/local/bin
postgres@node1:~$ wal-g wal-verify integrity timeline
INFO: 2025/08/13 18:31:42.080357 Current WAL segment: 0000002500000000000000007F
INFO: 2025/08/13 18:31:42.081025 Building check runner: integrity
INFO: 2025/08/13 18:31:42.082846 Detected earliest available backup: base_00000025000000000000000077
INFO: 2025/08/13 18:31:42.082962 Running the check: integrity
INFO: 2025/08/13 18:31:42.083304 Building check runner: timeline
INFO: 2025/08/13 18:31:42.083432 Running the check: timeline
WARNING: 2025/08/13 18:31:42.083589 Could not parse the timeline Id from 00000025000000000000000077.0000002
8.backup.lz4. Skipping...
WARNING: 2025/08/13 18:31:42.083876 Could not parse the timeline Id from 00000025000000000000000079.0000002
8.backup.lz4. Skipping...
WARNING: 2025/08/13 18:31:42.083987 Could not parse the timeline Id from 0000002500000000000000007B.0000002
8.backup.lz4. Skipping...
WARNING: 2025/08/13 18:31:42.084087 Could not parse the timeline Id from 0000002500000000000000007E.0000002
8.backup.lz4. Skipping...
WARNING: 2025/08/13 18:31:42.084202 Could not parse the timeline Id from base_00000025000000000000000077.lz
4. Skipping...
WARNING: 2025/08/13 18:31:42.084278 Could not parse the timeline Id from basebackups_005.lz4. Skipping...
WARNING: 2025/08/13 18:31:42.084361 Could not parse the timeline Id from walg.lz4. Skipping...
[wal-verify] integrity check status: OK
[wal-verify] integrity check details:
+-----+-----+-----+-----+-----+
| TLI | START | END | SEGMENTS COUNT | STATUS |
+-----+-----+-----+-----+-----+
| 37 | 00000025000000000000000077 | 0000002500000000000000007E | 8 | FOUND |
+-----+-----+-----+-----+-----+
[wal-verify] timeline check status: OK
[wal-verify] timeline check details:
Highest timeline found in storage: 37
Current cluster timeline: 37
postgres@node1:~$
```

Рисунок 3.15 – Проверка созданных РК при помощи аргументов integrity и timeline

### 3.4. Загрузка резервных копий журналов транзакций (WAL) в хранилище

После создания и проверки РК журналов транзакций (WAL) необходимо загрузить в хранилище. В большинстве случаев данная процедура выполняется автоматически СУБД.

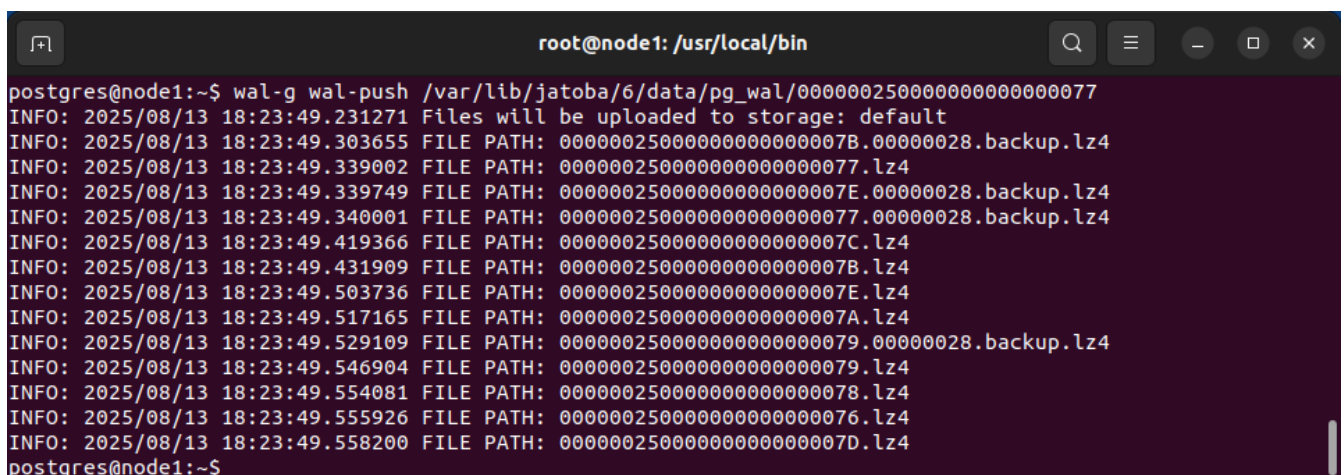
Созданные, но не загруженные в хранилище РК журналы транзакций (WAL) отмечаются статусом `MISSING_UPLOADING` (см. пример в п.п. 3.3.1).

В случае если необходимо загрузить РК журналы транзакций (WAL) вручную применяется команда с следующим синтаксисом:

```
wal-g wal-push $PGDATA/pg_wal/[backup_name]
```

#### Пример:

```
postgres@node1:~$ wal-g wal-push  
/var/lib/jatoba/6/data/pg_wal/00000025000000000000000077
```



```
root@node1: /usr/local/bin  
postgres@node1:~$ wal-g wal-push /var/lib/jatoba/6/data/pg_wal/00000025000000000000000077  
INFO: 2025/08/13 18:23:49.231271 Files will be uploaded to storage: default  
INFO: 2025/08/13 18:23:49.303655 FILE PATH: 0000002500000000000000007B.00000028.backup.lz4  
INFO: 2025/08/13 18:23:49.339002 FILE PATH: 00000025000000000000000077.lz4  
INFO: 2025/08/13 18:23:49.339749 FILE PATH: 0000002500000000000000007E.00000028.backup.lz4  
INFO: 2025/08/13 18:23:49.340001 FILE PATH: 00000025000000000000000077.00000028.backup.lz4  
INFO: 2025/08/13 18:23:49.419366 FILE PATH: 0000002500000000000000007C.lz4  
INFO: 2025/08/13 18:23:49.431909 FILE PATH: 0000002500000000000000007B.lz4  
INFO: 2025/08/13 18:23:49.503736 FILE PATH: 0000002500000000000000007E.lz4  
INFO: 2025/08/13 18:23:49.517165 FILE PATH: 0000002500000000000000007A.lz4  
INFO: 2025/08/13 18:23:49.529109 FILE PATH: 00000025000000000000000079.00000028.backup.lz4  
INFO: 2025/08/13 18:23:49.546904 FILE PATH: 00000025000000000000000079.lz4  
INFO: 2025/08/13 18:23:49.554081 FILE PATH: 00000025000000000000000078.lz4  
INFO: 2025/08/13 18:23:49.555926 FILE PATH: 00000025000000000000000076.lz4  
INFO: 2025/08/13 18:23:49.558200 FILE PATH: 0000002500000000000000007D.lz4  
postgres@node1:~$
```

Рисунок 3.16 – Результат выполнения команды wal-push

В случае, если загрузка РК журналов транзакций (WAL) уже выполнялась ранее, в ответ будет выведено сообщение `Files will be uploaded to storage: [storage_name]`. В качестве значения `[storage_name]` по умолчанию используется «default».

### 3.5. Восстановление данных из резервной копии

Использование компонента позволяет восстановить исходные данные СУБД на момент создания РК.

### 3.5.1. Предварительные условия для выполнения восстановления данных из резервной копии

Предварительными условиями выполнения компонентом процедур восстановления являются:

- настроенное непрерывное архивирование WAL (см. п.п. 2.3.2);
- настроено резервное копирование СУБД с помощью backup-push (см. п.п. 3.1);
- с целью обеспечения безопасности созданных ранее РК рекомендуется выпол-

нять восстановление данных в новый каталог, из содержимого которого будут в дальнейшем созданы новый РК. Далее в примерах восстановление данных СУБД выполняется в каталог `/var/lib/jatoba/6/data_new`



В случае использования для восстановления каталога СУБД (по умолчанию `/var/lib/jatoba/6/data`) повторно его необходимо удалить при помощи команды:

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

**ВАЖНО!** Данное действие приведет к удалению текущего экземпляра СУБД.

### 3.5.2. Восстановление данных из последней резервной копии

В качестве последней РК может использоваться как полная, так и инкрементальная РК.

Восстановление данных из последней РК выполняется в несколько шагов:

- 1) Выполнить предварительные условия (см. 3.5.1);
- 2) Остановить СУБД при помощи команды:

```
# systemctl stop jatoba-6
```

- 3) Создать новый каталог СУБД для восстановления:

```
postgres@node1:~$ mkdir /var/lib/jatoba/6/data_new/
```

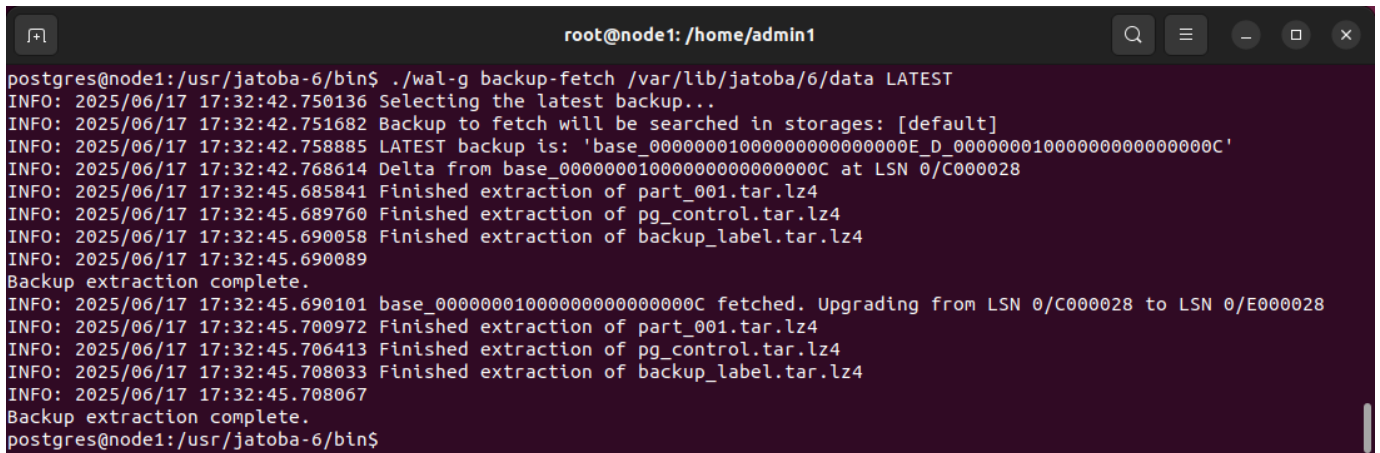
- 4) Выполнить загрузку данных СУБД из последней полной РК. Общий синтаксис команды выглядит следующим образом:

```
wal-g backup-fetch $PGDATA LATEST
```

Где \$PGDATA – путь к новому каталогу распаковки данных СУБД. Если каталог для распаковки не существует, то в таком случае компонент создаст его и все промежуточные подкаталоги; LATEST – указание использовать в качестве источника последнюю полную РК.

**Пример:**

```
postgres@node1:~$ wal-g backup-fetch /var/lib/jatoba/6/data_new  
LATEST
```



```
root@node1: /home/admin1  
postgres@node1:/usr/jatoba-6/bin$ ./wal-g backup-fetch /var/lib/jatoba/6/data LATEST  
INFO: 2025/06/17 17:32:42.750136 Selecting the latest backup...  
INFO: 2025/06/17 17:32:42.751682 Backup to fetch will be searched in storages: [default]  
INFO: 2025/06/17 17:32:42.758885 LATEST backup is: 'base_000000010000000000000000E_D_0000000100000000000000C'  
INFO: 2025/06/17 17:32:42.768614 Delta from base_000000010000000000000000C at LSN 0/C000028  
INFO: 2025/06/17 17:32:45.685841 Finished extraction of part_001.tar.lz4  
INFO: 2025/06/17 17:32:45.689760 Finished extraction of pg_control.tar.lz4  
INFO: 2025/06/17 17:32:45.690058 Finished extraction of backup_label.tar.lz4  
INFO: 2025/06/17 17:32:45.690089 Backup extraction complete.  
INFO: 2025/06/17 17:32:45.690101 base_000000010000000000000000C fetched. Upgrading from LSN 0/C000028 to LSN 0/E000028  
INFO: 2025/06/17 17:32:45.700972 Finished extraction of part_001.tar.lz4  
INFO: 2025/06/17 17:32:45.706413 Finished extraction of pg_control.tar.lz4  
INFO: 2025/06/17 17:32:45.708033 Finished extraction of backup_label.tar.lz4  
INFO: 2025/06/17 17:32:45.708067 Backup extraction complete.  
postgres@node1:/usr/jatoba-6/bin$
```

Рисунок 3.17 – Запуск и успешное восстановление СУБД из РК

5) В конфигурационный файл СУБД «Jatoba» postgresql.conf, расположенный в каталоге /var/lib/jatoba/6/data\_new, внести необходимые изменения:

а) Настроить параметры для выполнения восстановления:

```
data_directory = '/var/lib/jatoba/6/data_new'
```

б) Отключить непрерывное архивирование WAL с помощью изменения следующих параметров:

```
archive_mode = off
```

6) Создать в каталоге БД специальный файл-сигнал для восстановления:

```
postgres@node1:~$ touch /var/lib/jatoba/6/data_new/recov-  
ery.signal
```

7) Запустить сервисы СУБД «Jatoba» и убедиться в доступе к восстановленным данным:

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

```
# /usr/jatoba-6/bin/pg_ctl start -D /var/lib/jatoba/6/data_new  
-l /var/lib/jatoba/6/data_new/startup.log
```

8) Остановить СУБД после завершения восстановления при помощи команды:

```
# /usr/jatoba-6/bin/pg_ctl stop -D /var/lib/jatoba/6/data_new
```



Если необходимо сохранить каталог данных, то его необходимо переименовать:

```
mv /var/lib/jatoba/6/data /var/lib/jatoba/6/data.old
```

9) При восстановлении СУБД в другой каталог данных, переместить его в исходный при помощи команды:

```
# mv -f /var/lib/jatoba/6/data_new /var/lib/jatoba/6/data
```

В этом случае в конфигурационном файле postgresql.conf необходимо изменить параметры:

```
data_directory = '/var/lib/jatoba/6/data'  
archive_mode = 'on'
```

10) Изменить в конфигурационном файле walg.json (см. п.п. 2.2) параметр "WALG\_FILE\_PREFIX" указав новое расположение создаваемых РК:

```
"WALG_FILE_PREFIX": "/nfs/walg_archive/server2"
```

11) В конфигурационном файле postgresql.conf повторно настроить параметры непрерывного архивирования WAL (см. п.п. 2.3.2) и выполнить перезагрузку сервиса СУБД:

```
# systemctl stop jatoba-6  
# systemctl start jatoba-6
```

После выполнения восстановления данных СУБД из последней полной РК необходимо убедиться в доступе и в их корректности.



### 3.5.3. Восстановление данных из резервной копии на определенный момент времени

Восстановление данных из РК на определенный момент времени доступно при использовании:

- названия РК, созданной в требуемый период времени;
- требуемой даты и времени.

#### 3.5.3.1 С использованием названия резервной копии

Восстановление данных из РК с использованием названия резервной копии, созданной в требуемый период времени, выполняется в несколько шагов:

- 1) Выполнить предварительные условия (см. п. 3.5.1);
- 2) Остановить СУБД при помощи команды:

```
# systemctl stop jatoba-6
```

- 3) Вывести детализированный список доступных в хранилище РК при помощи команды:

```
postgres@node1:~$ wal-g backup-list --detail
```

- 4) Определить наименование полной РК, дата создания которой (по столбцу "finish\_time" ) раньше момента возникновения проблем в СУБД.

Например, если требуется выполнить восстановление на дату и время 09:00 17.06.2025 года, то в списке определяется РК, с наиболее близкими датой и временем к требуемой. На рисунке 3.18 РК с названием base\_00000001000000000000000004 и значением finish\_time 2025-06-17T08:53:52Z.



```

root@node1: /home/admin1
postgres@node1:~$ wal-g backup-list --detail
INFO: 2025/06/30 11:06:07.860049 List backups from storages: [default]
backup_name      modified          wal_file_name      storage_name start_time      fini
sh_time          hostname data_dir      pg_version start_lsn finish_lsn is_permanent
base_00000001000000000000000000000004 2025-06-17T11:53:52+03:00 00000001000000000000000000000004 default 2025-06-17T08:53:51Z 2025
-06-17T08:53:52Z node1 /var/lib/jatoba/6/data 160009 0/4000028 0/4000178 true
base_00000001000000000000000000000006_D_00000001000000000000000000000004 2025-06-17T14:16:05+03:00 00000001000000000000000000000006 default 2025-06-17T11:16:04Z 2025
-06-17T11:16:05Z node1 /var/lib/jatoba/6/data 160009 0/6000028 0/6000130 false
base_00000001000000000000000000000008 2025-06-17T15:44:30+03:00 00000001000000000000000000000008 default 2025-06-17T12:44:29Z 2025
-06-17T12:44:30Z node1 /var/lib/jatoba/6/data 160009 0/8000028 0/8000130 false
base_0000000100000000000000000000000A_D_00000001000000000000000000000006 2025-06-17T15:45:34+03:00 0000000100000000000000000000000A default 2025-06-17T12:45:34Z 2025
-06-17T12:45:34Z node1 /var/lib/jatoba/6/data 160009 0/A000028 0/A000130 false
base_0000000100000000000000000000000C 2025-06-17T16:14:09+03:00 0000000100000000000000000000000C default 2025-06-17T13:14:07Z 2025
-06-17T13:14:09Z node1 /var/lib/jatoba/6/data 160009 0/C000028 0/C000130 false
base_0000000100000000000000000000000E_D_0000000100000000000000000000000C 2025-06-17T17:14:45+03:00 0000000100000000000000000000000E default 2025-06-17T14:14:45Z 2025
-06-17T14:14:45Z node1 /var/lib/jatoba/6/data 160009 0/E000028 0/E000130 false
postgres@node1:~$

```

Рисунок 3.18 – Просмотр детальной информации о РК

Стоит также учитывать столбец `hostname`, который отображает название сервера, на котором создана РК.

5) Выполнить загрузку данных СУБД из последней полной РК. Общий синтаксис команды выглядит следующим образом:

```
wal-g backup-fetch $PGDATANEW [backup_name]
```

Где `$PGDATANEW` – путь к новому каталогу данных для распаковки. Если каталог для распаковки не существует, то в таком случае компонент создаст его и все промежуточные подкаталоги; `backup_name` – название РК.

### Пример:

```
postgres@node1:~$ wal-g backup-fetch /var/lib/jatoba/6/data_new
base_0000000100000000000000000000000C
```

6) В конфигурационный файл СУБД «Jatoba» `postgersql.conf`, расположенный в каталоге `/var/lib/jatoba/6/data_new`, внести необходимые изменения:

а) Настроить параметры для выполнения восстановления:

```
recovery_target_name = 'rp1'
recovery_target_action = 'promote'
data_directory = '/var/lib/jatoba/6/data_new'
```

б) Отключить непрерывное архивирование WAL с помощью изменения следующих параметров:

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

```
archive_mode = off
```

- 7) Создать в каталоге БД специальный файл-сигнал для восстановления:

```
postgres@node1:~$ touch /var/lib/jatoba/6/data_new/recovery.signal
```

- 8) Запустить сервисы СУБД «Jatoba» и убедиться в доступе к восстановленным данным:

```
# /usr/jatoba-6/bin/pg_ctl start -D /var/lib/jatoba/6/data_new  
-l /var/lib/jatoba/6/data_new/startup.log
```

- 9) Изменить в конфигурационном файле walg.json (см. п.п. 2.2) параметр "WALG\_FILE\_PREFIX" указав новое расположение создаваемых РК в хранилище:

```
"WALG_FILE_PREFIX": "/nfs/walg_archive/server2"
```

- 10) В конфигурационном файле postgresql.conf повторно настроить параметры непрерывного архивирования WAL (см. п.п. 2.3.2) и выполнить перезагрузку и проверить статус сервиса СУБД:

```
# systemctl stop jatoba-6  
# systemctl start jatoba-6  
# systemctl status jatoba-6
```

После выполнения восстановления данных СУБД из последней полной РК необходимо убедиться в доступе к ним и в их корректности.

### 3.5.3.2 С использованием даты и времени резервной копии (PITR)

Восстановление данных из РК на определенный момент времени с указанием даты и времени выполняется в следующей последовательности:

- 1) Выполнить предварительные условия (см. п. 3.5.1);
- 2) Остановить СУБД при помощи команды:

```
# systemctl stop jatoba-6
```

3) Выполнить загрузку данных СУБД с использованием указанием даты и времени. Общий синтаксис команды выглядит следующим образом:

```
wal-g backup-fetch $PGDATANEW --pitr-timestamp=["YYYY-MM-DDThh:mm:ss+hh:mm"]
```

Где \$PGDATANEW – путь к новому каталогу данных для распаковки. Если каталог для распаковки не существует, то в таком случае компонент создаст его и все промежуточные подкаталоги; YYYY-MM-DDThh:mm:ss+hh:mm – дата и время, на которое необходимо выполнить восстановление СУБД.

### Пример:

```
postgres@node1:~$ wal-g backup-fetch /var/lib/jatoba/6/data_new  
--pitr-timestamp= "2025-09-09T14:02:08+00:00"
```



В случае, если в параметре --pitr-timestamp указывается момент времени меньший, чем момент проведения последней транзакции, то выполнение последующего запуска СУБД с восстановленным каталогом данных (шаг 5 перечня данного раздела) завершится с ошибкой.

4) Создать в каталоге БД специальный файл-сигнал для восстановления:

```
postgres@node1:~$ touch /var/lib/jatoba/6/data_new/recovery.signal
```

5) Запустить сервисы СУБД «Jatoba» и убедиться в доступе к восстановленным данным:

```
# /usr/jatoba-6/bin/pg_ctl start -D /var/lib/jatoba/6/data_new  
-l /var/lib/jatoba/6/data_new/startup.log
```

6) Изменить в конфигурационном файле walg.json (см. п.п. 2.2) параметр "WALG\_FILE\_PREFIX" указав новое расположение создаваемых РК в хранилище:

```
"WALG_FILE_PREFIX": "/nfs/walg_archive/server2"
```

7) Выполнить перезагрузку и проверить статус службы СУБД:

```
# systemctl stop jatoba-6
```

```
# systemctl start jatoba-6  
# systemctl status jatoba-6
```

После выполнения восстановления данных СУБД из последней полной РК необходимо убедиться в доступе к ним и в их корректности.

### 3.5.4. Восстановление данных на момент сразу после создания РК

Восстановление данных на момент сразу после создания РК выполняется в той же последовательности, что и в п. 3.5.3.1 кроме шага ба. Здесь при настройке параметров для выполнения восстановления необходимо вместо параметра `recovery_target_name` указать:

```
recovery_target = 'immediate'
```

## 3.6. Удаление резервных копий

Для того чтобы удалить ранее созданные РК применяется команда, имеющая следующий синтаксис:

```
wal-g delete [retain|before|everything|target] --confirm
```

По умолчанию при вызове команды `wal-g delete` фактическое удаление РК не производится. Вместо этого выводится список архивов с данными, содержащихся в РК, для оценки и дальнейшего принятия решения.

Для окончательного удаления РК при вызове команды `wal-g delete` применяется флаг `--confirm`.



Созданные РК, для которых установлен признак «постоянная» при создании или при помощи команды `backup-mark` с флагом `-i` (см. 3.1.2), будут сохранены в хранилище.

Для РК с признаком «постоянная» игнорируются задаваемые условия удаления.

Команда `wal-g delete` выполняется с специальными режимами, описание которых представлено далее в руководстве.

### 3.6.1. Режим `retain`

Режим `retain` предоставляет функционал удаления РК из хранилища, но с сохранением определенного пользователем количества РК по заданным условиям.

Режим `retain` может использоваться в случае необходимости автоматизированного контроля количества РК в хранилище.

Синтаксис команды `wal-g delete` в режиме `retain` выглядит следующим образом:

```
wal-g delete retain [FULL|FIND_FULL] [number] [--after  
[backup_name|timestamp] [--confirm]
```

Аргумент `number` определяет количество РК, которые необходимо сохранить в хранилище в режиме `retain`.

Если для команды `wal-g delete` в режиме `retain` указан флаг `FULL`, то при выполнении удаления в хранилище сохраняется число полных РК указанных в параметре `number` и все зависимые от них инкрементальные РК. Например при запуске `wal-g delete retain FULL 3` после выполнения команды в хранилище останутся 3 последних полных РК, а также инкрементальные РК, созданные на их основе.



Промежуточные инкрементальные РК при выполнении команды `wal-g delete retain FULL` будут удалены. Для сохранения инкрементальных РК, сделанных между полными РК, необходимо использовать параметр `FIND_FULL`.

Если для команды `wal-g delete` в режиме `retain` используется флаг `--after`, то в этом случае сохраняются:

— Количество `number` последних полных РК (в том числе инкрементальные РК, созданные на их основе полных РК) созданных после РК с названием `backup_name`. В качестве `backup_name` рекомендуется использовать названия полных РК;

— Количество `number` последних полных РК созданных после `timestamp` (дата и время в формате RFC3339: `yyyy-MM-ddTHH:mm:ss.SSSZ`).



В случае когда в команде `wal-g delete` в режиме `retain` не используется параметр `FULL`, выполнение команды может быть прервано при проверке условий. Например, имеется следующая конфигурация РК (лишние столбцы команды `wal-g backup-list` удалены для удобства представления):

```
postgres@node1:~$ wal-g backup-list  
INFO: 2025/06/05 15:52:17.530839 List backups from storages: [de-  
fault]  
backup_name
```

```
base_00000025000000000000000046
base_00000025000000000000000047
base_00000025000000000000000049_D_00000025000000000000000047
base_0000002500000000000000004B_D_00000025000000000000000049
base_0000002500000000000000004D
postgres@node1:~$ wal-g delete retain 2 --after
base_0000002500000000000000004D --confirm
INFO: 2025/06/05 15:53:02.136911 Backup to delete will be searched in
storages: [default]
INFO: 2025/06/05 15:53:02.137122 retrieving permanent objects
ERROR: 2025/06/05 15:53:02.154943
base_0000002500000000000000004B_D_00000025000000000000000049_backup_stop_
sentinel.json is incremental and it's predecessors cannot be deleted.
Consider FIND_FULL option
```

В данном случае команда wal-g delete retain прерывается по причине того, что заданные условия приведут к сохранению в хранилище в качестве последней инкрементальной РК (выделена полужирным начертанием). Но так как инкрементальная РК связана с удаляемыми полными РК, то нарушится их связность.

### Примеры:

```
postgres@node1:~$ wal-g delete retain FULL 5 --after 2019-12-
12T12:12:12 --confirm
```

При выполнении данной команды компонент сохранит в хранилище 5 последних РК (включая зависимые от них инкрементальные РК), а также РК созданные начиная с даты и времени 2019-12-12 12:12:12 включительно.

```
postgres@node1:~$ wal-g delete retain FULL 3 --after
base_0000002500000000000000003D --confirm
```

При выполнении данной команды компонент сохранит в хранилище три последних РК, созданных после РК base\_0000002500000000000000003D включительно.

```
mc [postgres@node1]:~
postgres@node1:/usr/local/bin$ wal-g delete retain FULL 3 --after base_000000250000000000000003D --confirm
INFO: 2025/06/05 11:46:04.480069 Backup to delete will be searched in storages: [default]
INFO: 2025/06/05 11:46:04.480185 retrieving permanent objects
INFO: 2025/06/05 11:46:04.490514 Found permanent objects: backups=map[{base_0000002300000000000000026 default}:true], wals=map[{0000
00230000000000000000026 default}:true]
INFO: 2025/06/05 11:46:04.494919 Start delete
INFO: 2025/06/05 11:46:04.545799 Objects in folder:
INFO: 2025/06/05 11:46:04.545868 will be deleted: basebackups_005/base_000000250000000000000002D_D_0000002300000000000000026_b
ackup_stop_sentinel.json, from storage: default
INFO: 2025/06/05 11:46:04.545889 will be deleted: basebackups_005/base_0000002500000000000000035_D_000000250000000000000002D_b
ackup_stop_sentinel.json, from storage: default
INFO: 2025/06/05 11:46:04.545901 will be deleted: basebackups_005/base_0000002500000000000000038_D_0000002500000000000000035_b
ackup_stop_sentinel.json, from storage: default
INFO: 2025/06/05 11:46:04.545991 will be deleted: basebackups_005/base_000000250000000000000002D_D_0000002300000000000000026/f
iles_metadata.json, from storage: default
INFO: 2025/06/05 11:46:04.546011 will be deleted: basebackups_005/base_000000250000000000000002D_D_0000002300000000000000026/m
etadata.json, from storage: default
```

Рисунок 3.19 – Удаление РК с использованием режима retain


### 3.6.2. Режим before

Режим before предоставляет функционал удаления РК, но с сохранением в хранилище определенного количества РК по заданным в команде wal-g delete условиям. Режим before может использоваться в случае необходимости автоматизированного контроля количества РК в хранилище.

Синтаксис команды wal-g delete в режиме before выглядит следующим образом:

```
wal-g delete before [FIND_FULL] [backup_name|timestamp] [--con-
firm]
```

Где [backup\_name] – название РК, до которой должно выполняться удаление РК из хранилища; [timestamp] – дата и время в формате RFC3339: yyyy-MM-ddTHH:mm:ss.SSSZ.

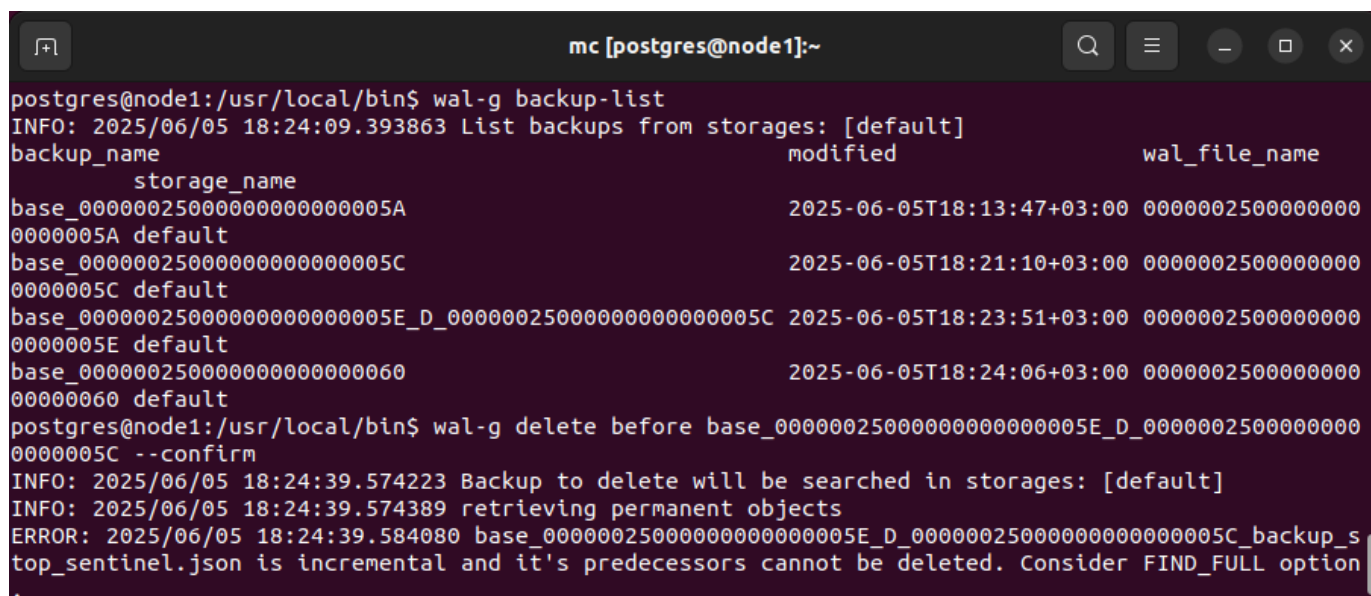
 Если флаг FIND\_FULL не указан, а выполнение удаления РК (полной или инкрементальной) может привести к появлению осиротевших инкрементальных РК, то выполнение команды wal-g delete в режиме before завершится с выводом сообщения об ошибке «[backup\_name] is incremental and it's predecessors cannot be deleted. Consider FIND\_FULL option».

Например, имеется следующая конфигурация РК (лишние столбцы команды wal-g backup-list удалены для удобства представления):

```
postgres@node1:~$ wal-g backup-list
INFO: 2025/06/05 18:24:09.393863 List backups from storages: [de-
fault]
backup_name
base_00000025000000000000000005A
base_00000025000000000000000005C
base_00000025000000000000000005E_D_000000250000000000000005C
base_000000250000000000000000060
```

```
postgres@node1:~$ wal-g delete before
base_0000002500000000000000005E_D_0000002500000000000000005C --confirm
INFO: 2025/06/05 18:24:39.574223 Backup to delete will be searched in
storages: [default]
INFO: 2025/06/05 18:24:39.574389 retrieving permanent objects
ERROR: 2025/06/05 18:24:39.584080
base_0000002500000000000000005E_D_0000002500000000000000005C_backup_stop_
sentinel.json is incremental and it's predecessors cannot be deleted.
Consider FIND_FULL option.
```

Здесь при выполнении команды `wal-g delete before` в качестве РК до которой будет выполняться удаление указана инкрементальная (выделена полужирным начертанием). В случае выполнения данного условия указанная инкрементальная РК потеряет связность, поэтому компонент прерывает выполнение команды (см. рисунок 3.20).



```
mc [postgres@node1]:~
postgres@node1:/usr/local/bin$ wal-g backup-list
INFO: 2025/06/05 18:24:09.393863 List backups from storages: [default]
backup_name      modified      wal_file_name
  storage_name
base_0000002500000000000000005A 2025-06-05T18:13:47+03:00 0000002500000000
0000005A default
base_0000002500000000000000005C 2025-06-05T18:21:10+03:00 0000002500000000
0000005C default
base_0000002500000000000000005E_D_0000002500000000000000005C 2025-06-05T18:23:51+03:00 0000002500000000
0000005E default
base_00000025000000000000000060 2025-06-05T18:24:06+03:00 0000002500000000
00000060 default
postgres@node1:/usr/local/bin$ wal-g delete before base_0000002500000000000000005E_D_0000002500000000
0000005C --confirm
INFO: 2025/06/05 18:24:39.574223 Backup to delete will be searched in storages: [default]
INFO: 2025/06/05 18:24:39.574389 retrieving permanent objects
ERROR: 2025/06/05 18:24:39.584080 base_0000002500000000000000005E_D_0000002500000000000000005C_backup_s
top_sentinel.json is incremental and it's predecessors cannot be deleted. Consider FIND_FULL option
```

Рисунок 3.20 – Прерывание выполнения команды `wal-g delete before` при возникновении ошибки удаления РК

Если для команды `wal-g delete` в режиме `before` указан флаг `FIND_FULL`, то при выполнении удаления РК из хранилища будет определена полная РК, необходимая для сохранения всех зависимых от нее инкрементальных РК.

```
postgres@node1:~$ wal-g delete before FIND_FULL
base_0000002500000000000000005E_D_0000002500000000000000005C --con-
firm
```



```
mc [postgres@node1]:~
postgres@node1:/usr/local/bin$ wal-g delete before FIND_FULL base_000000250000000000000005E_D_0000002500
00000000000005C --confirm
INFO: 2025/06/05 18:25:01.775585 Backup to delete will be searched in storages: [default]
INFO: 2025/06/05 18:25:01.775704 retrieving permanent objects
INFO: 2025/06/05 18:25:01.778121 Start delete
INFO: 2025/06/05 18:25:01.815957 Objects in folder:
INFO: 2025/06/05 18:25:01.816033 will be deleted: basebackups_005/base_000000250000000000000005A_
backup_stop_sentinel.json, from storage: default
INFO: 2025/06/05 18:25:01.816078 will be deleted: basebackups_005/base_000000250000000000000005A/
files_metadata.json, from storage: default
INFO: 2025/06/05 18:25:01.816086 will be deleted: basebackups_005/base_000000250000000000000005A/
metadata.json, from storage: default
INFO: 2025/06/05 18:25:01.816126 will be deleted: basebackups_005/base_000000250000000000000005A/
tar_partitions/backup_label.tar.lz4, from storage: default
INFO: 2025/06/05 18:25:01.816138 will be deleted: basebackups_005/base_000000250000000000000005A/
tar_partitions/part_001.tar.lz4, from storage: default
INFO: 2025/06/05 18:25:01.816149 will be deleted: basebackups_005/base_000000250000000000000005A/
tar_partitions/part_002.tar.lz4, from storage: default
INFO: 2025/06/05 18:25:01.816159 will be deleted: basebackups_005/base_000000250000000000000005A/
```

Рисунок 3.21 – Удаление РК командой wal-g delete в режиме before с использованием флага FIND\_FULL

В данном случае, несмотря на то, что для удаления указана инкрементальная РК base\_000000250000000000000005E\_D\_000000250000000000000005C компонент оставляет в хранилище ее родительскую постоянную РК с названием base\_000000250000000000000005C.

```
mc [postgres@node1]:~
postgres@node1:/usr/local/bin$ wal-g backup-list
INFO: 2025/06/05 18:25:06.342514 List backups from storages: [default]
backup_name                                modified                                wal_file_name
storage_name
base_00000025000000000000000005C          2025-06-05T18:21:10+03:00 0000002500000000
0000005C default
base_000000250000000000000005E_D_000000250000000000000005C 2025-06-05T18:23:51+03:00 0000002500000000
0000005E default
base_000000250000000000000000060          2025-06-05T18:24:06+03:00 0000002500000000
00000060 default
postgres@node1:/usr/local/bin$
```

Рисунок 3.22 – Результат выполнения команды wal-g delete в режиме before с использованием флага FIND\_FULL

### 3.6.3. Режим everything

Режим everything выполняет полную очистку всех резервных копий в хранилище в соответствии с задаваемыми условиями.

Синтаксис команды wal-g delete в режиме everything выглядит следующим образом:

```
postgres@node1:~$ wal-g delete everything [FORCE] [--confirm]
```

Таблица 3.2 – Флаги команды wal-g delete в режиме everything

Флаг	Описание	Пример
Без флага	Удаляет все непостоянные РК (полные и инкрементальные)	everything
FORCE	Принудительное удаление всех РК (включая постоянные)	everything FORCE

```
mc [postgres@node1]:~
postgres@node1:/usr/local/bin$ wal-g delete everything --confirm
INFO: 2025/06/06 09:49:53.863419 Backup to delete will be searched in storages: [default]
INFO: 2025/06/06 09:49:53.863566 retrieving permanent objects
INFO: 2025/06/06 09:49:53.964097 Objects in folder:
INFO: 2025/06/06 09:49:53.964124 will be deleted: basebackups_005/base_000000250000000000000005C_
backup_stop_sentinel.json, from storage: default
INFO: 2025/06/06 09:49:53.964136 will be deleted: basebackups_005/base_000000250000000000000005E_
D_000000250000000000000005C_backup_stop_sentinel.json, from storage: default
INFO: 2025/06/06 09:49:53.964145 will be deleted: basebackups_005/base_0000002500000000000000060_
backup_stop_sentinel.json, from storage: default
INFO: 2025/06/06 09:49:53.964148 will be deleted: basebackups_005/base_000000250000000000000005C/
files_metadata.json, from storage: default
INFO: 2025/06/06 09:49:53.964177 will be deleted: basebackups_005/base_000000250000000000000005C/
metadata.json, from storage: default
INFO: 2025/06/06 09:49:53.964180 will be deleted: basebackups_005/base_000000250000000000000005E_
D_000000250000000000000005C/files_metadata.json, from storage: default
INFO: 2025/06/06 09:49:53.964183 will be deleted: basebackups_005/base_000000250000000000000005E_
D_000000250000000000000005C/metadata.json, from storage: default
INFO: 2025/06/06 09:49:53.964191 will be deleted: basebackups_005/base_0000002500000000000000060/
files_metadata.json, from storage: default
INFO: 2025/06/06 09:49:53.964194 will be deleted: basebackups_005/base_0000002500000000000000060/
metadata.json, from storage: default
```

Рисунок 3.23 – Удаление РК с использованием режима everything

```
mc [postgres@node1]:~
tar_partitions/part_011.tar.lz4, from storage: default
INFO: 2025/06/06 09:49:53.964373 will be deleted: basebackups_005/base_0000002500000000000000060/
tar_partitions/part_012.tar.lz4, from storage: default
INFO: 2025/06/06 09:49:53.964375 will be deleted: basebackups_005/base_0000002500000000000000060/
tar_partitions/pg_control.tar.lz4, from storage: default
postgres@node1:/usr/local/bin$ wal-g backup-list
INFO: 2025/06/06 09:52:40.770541 List backups from storages: [default]
INFO: 2025/06/06 09:52:40.774395 No backups found
postgres@node1:/usr/local/bin$
```

Рисунок 3.24 – Отсутствие РК в хранилище в результате выполнения команды wal-g delete в режиме everything

При использовании флага FORCE будут удалены постоянные РК с признаком permanent (см. п.п. 3.1).

### 3.6.4. Режим target

Режим target позволяет выполнять удаление РК с возможностью выбора стратегии обработки зависимых (родительских или дочерних) РК.

```
wal-g delete target [FIND_FULL] [backup_name] --confirm
```

Таблица 3.3 – Флаги команды wal-g delete в режиме target

Флаг	Описание	Пример
Без флага	Удаляет указанную РК и зависимые инкрементальные РК	target [backup_name]
FIND_FULL	Удаляет все зависимые полные РК от указанной инкрементальной РК	target FIND_FULL [backup_name]

В режиме target выполнение команды delete по умолчанию приведет к удалению РК с указанным названием.

В отличие от других режимов удаления РК, команда wal-g delete target позволяет удалить РК, отмеченные признаком «permanent», то есть постоянная РК (см. п.п. 3.1.2).

#### Примеры:

Выполнить удаление полной РК, а также всех зависимых от нее инкрементальных РК:

```
postgres@node1:~$ wal-g delete target  
base_00000025000000000000000062 --confirm
```

```
mc [postgres@node1]:~
postgres@node1:/usr/local/bin$ wal-g backup-list
INFO: 2025/06/06 16:19:38.667187 List backups from storages: [default]
backup_name                               modified                               wal_file_name
  storage_name
base_00000025000000000000000000000062    2025-06-06T16:19:12+03:00 000000250000000000000000
0062 default
base_00000025000000000000000000000064_D_00000025000000000000000000000062 2025-06-06T16:19:20+03:00 000000250000000000000000
0064 default
base_00000025000000000000000000000065_D_00000025000000000000000000000064 2025-06-06T16:19:23+03:00 000000250000000000000000
0065 default
base_00000025000000000000000000000066    2025-06-06T16:19:33+03:00 000000250000000000000000
0066 default
postgres@node1:/usr/local/bin$ wal-g delete target base_00000025000000000000000000000062 --confirm
INFO: 2025/06/06 16:20:31.779992 Backup to delete will be searched in storages: [default]
INFO: 2025/06/06 16:20:31.780121 retrieving permanent objects
INFO: 2025/06/06 16:20:31.790283 Selecting the backup with name base_00000025000000000000000000000062...
```

Рисунок 3.25 – Удаление полной РК с использованием режима target

```
mc [postgres@node1]:~
0000000064/tar_partitions/part_008.tar.lz4, from storage: default
INFO: 2025/06/06 16:20:31.833702 will be deleted: base_00000025000000000000000000000065_D_00000025000000
0000000064/tar_partitions/part_009.tar.lz4, from storage: default
INFO: 2025/06/06 16:20:31.833705 will be deleted: base_00000025000000000000000000000065_D_00000025000000
0000000064/tar_partitions/part_010.tar.lz4, from storage: default
INFO: 2025/06/06 16:20:31.833709 will be deleted: base_00000025000000000000000000000065_D_00000025000000
0000000064/tar_partitions/part_011.tar.lz4, from storage: default
INFO: 2025/06/06 16:20:31.833712 will be deleted: base_00000025000000000000000000000065_D_00000025000000
0000000064/tar_partitions/part_012.tar.lz4, from storage: default
INFO: 2025/06/06 16:20:31.833716 will be deleted: base_00000025000000000000000000000065_D_00000025000000
0000000064/tar_partitions/pg_control.tar.lz4, from storage: default
postgres@node1:/usr/local/bin$ wal-g backup-list
INFO: 2025/06/06 16:20:37.497695 List backups from storages: [default]
backup_name                               modified                               wal_file_name                               storage_name
base_00000025000000000000000000000066    2025-06-06T16:19:33+03:00 00000025000000000000000000000066 default
postgres@node1:/usr/local/bin$
```

Рисунок 3.26 – Результат удаления полной РК с использованием режима target

Удаление инкрементальной РК и зависимых от нее инкрементальных РК:

```
postgres@node1:~$ wal-g delete target
base_00000002500000000000000000000068_D_00000002500000000000000000000066 --con-
firm
```

```
mc [postgres@node1]:~
postgres@node1:/usr/local/bin$ wal-g backup-list
INFO: 2025/06/06 16:36:53.174861 List backups from storages: [default]
backup_name                               modified                               wal_file_name
storage_name
base_00000025000000000000000000000066 2025-06-06T16:19:33+03:00 000000250000000000000000
0066 default
base_00000025000000000000000000000068_D_00000025000000000000000000000066 2025-06-06T16:36:39+03:00 000000250000000000000000
0068 default
base_00000025000000000000000000000069_D_00000025000000000000000000000068 2025-06-06T16:36:43+03:00 000000250000000000000000
0069 default
base_0000002500000000000000000000006B 2025-06-06T16:36:50+03:00 000000250000000000000000
006B default
postgres@node1:/usr/local/bin$ wal-g delete target base_00000025000000000000000000000068_D_00000025000000000000000000000066 --confirm
INFO: 2025/06/06 16:37:28.191130 Backup to delete will be searched in storages: [default]
INFO: 2025/06/06 16:37:28.191281 retrieving permanent objects
INFO: 2025/06/06 16:37:28.197059 Selecting the backup with name base_00000025000000000000000000000068_D_00000025000000000000000000000066
```

Рисунок 3.27 – Удаление инкрементальной РК и зависимых от нее инкрементальных РК с использованием режима target

```
mc [postgres@node1]:~
0000000068/tar_partitions/part_008.tar.lz4, from storage: default
INFO: 2025/06/06 16:37:28.243587 will be deleted: base_00000025000000000000000000000069_D_00000025000000000000000000000068/tar_partitions/part_009.tar.lz4, from storage: default
INFO: 2025/06/06 16:37:28.243594 will be deleted: base_00000025000000000000000000000069_D_00000025000000000000000000000068/tar_partitions/part_010.tar.lz4, from storage: default
INFO: 2025/06/06 16:37:28.243597 will be deleted: base_00000025000000000000000000000069_D_00000025000000000000000000000068/tar_partitions/part_011.tar.lz4, from storage: default
INFO: 2025/06/06 16:37:28.243599 will be deleted: base_00000025000000000000000000000069_D_00000025000000000000000000000068/tar_partitions/part_012.tar.lz4, from storage: default
INFO: 2025/06/06 16:37:28.243603 will be deleted: base_00000025000000000000000000000069_D_00000025000000000000000000000068/tar_partitions/pg_control.tar.lz4, from storage: default
postgres@node1:/usr/local/bin$ wal-g backup-list
INFO: 2025/06/06 16:37:31.347076 List backups from storages: [default]
backup_name                               modified                               wal_file_name                               storage_name
base_00000025000000000000000000000066 2025-06-06T16:19:33+03:00 00000025000000000000000000000066 default
base_0000002500000000000000000000006B 2025-06-06T16:36:50+03:00 0000002500000000000000000000006B default
postgres@node1:/usr/local/bin$
```

Рисунок 3.28 – Результат удаления инкрементальной РК и зависимых от нее инкрементальных РК с использованием режима target

Удаление инкрементальной РК, а также ее родительской РК, и зависимых инкрементальных РК, выполняется следующей командой:

```
postgres@node1:~$ wal-g delete target FIND_FULL
base_00000025000000000000000000000080_D_0000002500000000000000000000007E --confirm
```



```

root@node1: /usr/local/bin

postgres@node1:~$ wal-g backup-list
INFO: 2025/08/14 13:10:41.162434 List backups from storages: [default]
backup_name                                modified                                wal_file_name                            storage_name
base_00000025000000000000000000000077    2025-08-13T15:18:33+03:00 00000025000000000000000000000077 default
base_00000025000000000000000000000079_D_00000025000000000000000000000077 2025-08-13T15:20:22+03:00 00000025000000000000000000000079 default
base_0000002500000000000000000000007B    2025-08-13T16:15:35+03:00 0000002500000000000000000000007B default
base_0000002500000000000000000000007E    2025-08-13T17:10:19+03:00 0000002500000000000000000000007E default
base_00000025000000000000000000000080_D_0000002500000000000000000000007E 2025-08-14T11:56:16+03:00 00000025000000000000000000000080 default
base_00000025000000000000000000000082_D_00000025000000000000000000000080 2025-08-14T12:19:44+03:00 00000025000000000000000000000082 default
base_00000025000000000000000000000084    2025-08-14T12:20:17+03:00 00000025000000000000000000000084 default
postgres@node1:~$ wal-g delete target FIND_FULL base_00000025000000000000000000000080_D_0000002500000000000000000000007E --confirm
INFO: 2025/08/14 13:12:41.912630 Backup to delete will be searched in storages: [default]
INFO: 2025/08/14 13:12:41.912779 retrieving permanent objects
INFO: 2025/08/14 13:12:41.918981 Selecting the backup with name base_00000025000000000000000000000080_D_0000002500000000000000000000007E..
INFO: 2025/08/14 13:12:41.928527 Objects in folder:
INFO: 2025/08/14 13:12:41.928564 will be deleted: base_0000002500000000000000000000007E_backup_stop_sentinel.json, from storage: default
INFO: 2025/08/14 13:12:41.928587 will be deleted: base_00000025000000000000000000000080_D_0000002500000000000000000000007E_backup_stop_sentinel.json, from storage: default
INFO: 2025/08/14 13:12:41.928600 will be deleted: base_00000025000000000000000000000082_D_00000025000000000000000000000080_backup_stop_sentinel.json, from storage: default
INFO: 2025/08/14 13:12:41.928648 will be deleted: base_0000002500000000000000000000007E/files_metadata.json, from storage: default

```

Рисунок 3.29 – Удаление инкрементальной РК, а также ее родительской РК, и зависимых инкрементальных РК с использованием режима target

В случае если удаление с флагом FIND\_FULL в режиме target выполняется для крайней инкрементальной РК, то будет выполнено удаление по цепочке РК вплоть до полной РК, которая является родительской по отношению ко всем дочерним.

```

root@node1: /usr/local/bin

om storage: default
INFO: 2025/08/14 13:12:41.928730 will be deleted: base_0000002500000000000000000000007E/tar_partitions/pg_control.tar.lz4, from storage: default
INFO: 2025/08/14 13:12:41.928738 will be deleted: base_00000025000000000000000000000080_D_0000002500000000000000000000007E/tar_partitions/backup_label.tar.lz4, from storage: default
INFO: 2025/08/14 13:12:41.928744 will be deleted: base_00000025000000000000000000000080_D_0000002500000000000000000000007E/tar_partitions/part_001.tar.lz4, from storage: default
INFO: 2025/08/14 13:12:41.928750 will be deleted: base_00000025000000000000000000000080_D_0000002500000000000000000000007E/tar_partitions/pg_control.tar.lz4, from storage: default
INFO: 2025/08/14 13:12:41.928755 will be deleted: base_00000025000000000000000000000082_D_00000025000000000000000000000080/tar_partitions/backup_label.tar.lz4, from storage: default
INFO: 2025/08/14 13:12:41.928759 will be deleted: base_00000025000000000000000000000082_D_00000025000000000000000000000080/tar_partitions/part_001.tar.lz4, from storage: default
INFO: 2025/08/14 13:12:41.928767 will be deleted: base_00000025000000000000000000000082_D_00000025000000000000000000000080/tar_partitions/pg_control.tar.lz4, from storage: default
postgres@node1:~$ wal-g backup-list
INFO: 2025/08/14 13:13:03.013133 List backups from storages: [default]
backup_name                                modified                                wal_file_name                            storage_name
base_00000025000000000000000000000077    2025-08-13T15:18:33+03:00 00000025000000000000000000000077 default
base_00000025000000000000000000000079_D_00000025000000000000000000000077 2025-08-13T15:20:22+03:00 00000025000000000000000000000079 default
base_0000002500000000000000000000007B    2025-08-13T16:15:35+03:00 0000002500000000000000000000007B default
base_00000025000000000000000000000084    2025-08-14T12:20:17+03:00 00000025000000000000000000000084 default
postgres@node1:~$

```

Рисунок 3.30 – Результат удаления инкрементальной РК, а также ее родительской РК, и зависимых инкрементальных РК с использованием режима target

### 3.6.5. Режим garbage

Режим garbage позволяет выполнять удаление устаревших архивов WAL и остаточных файлов РК из хранилища, например, неудачные РК или частично удаленные.

В режиме garbage удаляются все не постоянные объекты (см. п.п. 3.1.2) до самой ранней не постоянной РК.

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

Этот режим полезен в случаях, когда РК удаляются командой `delete target` (см. 3.6.4), после которой возможно появление инкрементальных РК без привязки к полным РК.

Синтаксис команды `wal-g delete` в режиме `garbage` выглядит следующим образом:

```
wal-g delete garbage [ARCHIVES | BACKUPS] [--confirm]
```

Где флаг `ARCHIVES` – удаляет из хранилища только устаревшие архивы WAL; `BACKUPS` – удаляет из хранилища только оставшиеся (частично удаленные или неудачные) файлы РК.



Если запустить команду `wal-g delete` в режиме `garbage` без флагов, то выполняются и `ARCHIVES`, и `BACKUPS`.

### Пример:

```
postgres@node1:~$ wal-g delete garbage --confirm
```

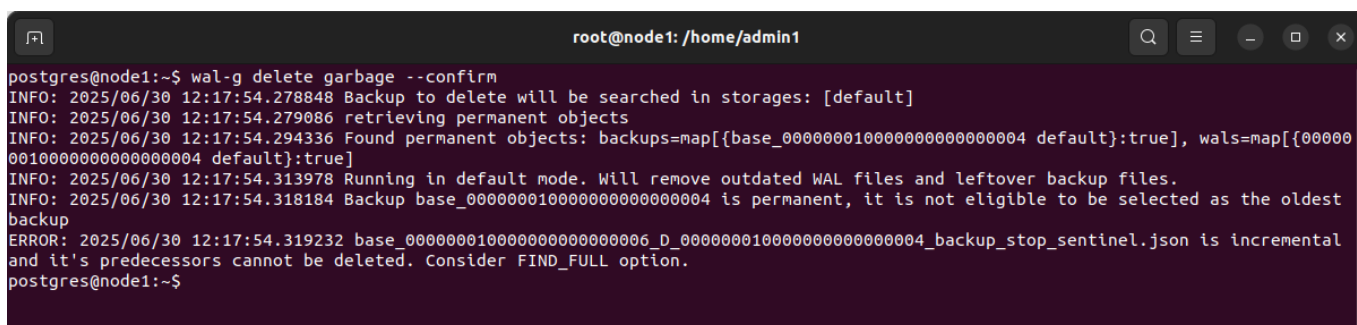


Рисунок 3.31 – Удаление устаревших РК с использованием режима `garbage`

## 3.7. Перемещение резервных копий

При выполнении перемещения РК в хранилище необходимо выполнить изменение в конфигурационном файле `walg.json` параметра `"WALG_FILE_PREFIX"` указав актуальное местонахождение РК в хранилище.

Дополнительных действий после изменения параметра `"WALG_FILE_PREFIX"` не требуется.

## 3.8. Журнал работы компонента

Система ведения журналов работы компонента обеспечивает фиксацию событий и операций резервного копирования/восстановления.

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

При установке компонента журнал работы не будет создан. Необходимо в конфигурационном файле (см. п.п. 2.2) указать расположение файла журнала работы при помощи параметра WALG\_LOG\_DESTINATION, например:

```
"WALG_LOG_DESTINATION": "/var/lib/jatoba/wal-g.log"
```

После указания в конфигурационном файле расположения файла журнала он будет создан автоматически при первом вызове команды компонента.

```
mc [root@node1]:/var/lib/jatoba
/var/lib/jatoba/wal-g.log 2154/2154 100%
INFO: 2025/06/17 11:47:57.845429 List backups from storages: [default]
INFO: 2025/06/17 11:47:57.852410 No backups found
INFO: 2025/06/17 11:48:57.560809 List backups from storages: [default]
INFO: 2025/06/17 11:48:57.561731 No backups found
INFO: 2025/06/17 11:49:16.108324 Backup will be pushed to storage: default
ERROR: 2025/06/17 11:49:16.110563 failed to connect to `user=postgres database=`: /var/run/jatoba/.s.PGSQL.5432/.s.PGSQL.5432 (/var/run/jatoba/.s.PGSQL.5432): dial error: dial unix /var/run/jatoba/.s.PGSQL.5432/.s.PGSQL.5432: connect: not a directory
ERROR: 2025/06/17 11:49:16.110585 Failed to connect using provided PGHOST and PGPORT, trying localhost:5432
ERROR: 2025/06/17 11:49:16.128813 Connect: postgres connection failed: failed to connect to `user=postgres database=`: 127.0.0.1:5432 (localhost): server error: FATAL: unrecognized configuration parameter "gp_session_role" (SQLSTATE 42704)
INFO: 2025/06/17 11:53:40.773589 List backups from storages: [default]
INFO: 2025/06/17 11:53:40.775097 No backups found
INFO: 2025/06/17 11:53:51.595031 Backup will be pushed to storage: default
INFO: 2025/06/17 11:53:51.654757 Doing full backup.
INFO: 2025/06/17 11:53:51.666961 Calling pg_start_backup()
INFO: 2025/06/17 11:53:51.860942 Initializing the PG alive checker (interval=1m0s)...
INFO: 2025/06/17 11:53:51.861240 Starting a new tar bundle
INFO: 2025/06/17 11:53:51.861401 Walking ...
INFO: 2025/06/17 11:53:51.862049 Starting part 1 ...
INFO: 2025/06/17 11:53:52.672790 Packing ...
INFO: 2025/06/17 11:53:52.684504 Finished writing part 1.
INFO: 2025/06/17 11:53:52.730786 Starting part 2 ...
INFO: 2025/06/17 11:53:52.731056 /global/pg_control
INFO: 2025/06/17 11:53:52.734714 Finished writing part 2.
INFO: 2025/06/17 11:53:52.734762 Calling pg_stop_backup()
INFO: 2025/06/17 11:53:52.754202 Starting part 3 ...
INFO: 2025/06/17 11:53:52.758710 backup_label
INFO: 2025/06/17 11:53:52.758755 tablespace_map
INFO: 2025/06/17 11:53:52.759190 Finished writing part 3.
INFO: 2025/06/17 11:53:52.766748 Querying pg_database
INFO: 2025/06/17 11:53:52.863471 Wrote backup with name base_00000001000000000000000004 to storage default
1Help 2Unkrap 3Quit 4Hex 5Goto 6 7Search 8Raw 9Format 10Quit
```

Рисунок 3.32 – Фрагмент содержимого журнала работы компонента



При активации записи событий в файл журнал работы компонента прекращается вывод результатов выполнения команд в системной консоли ОС. Компонент не поддерживает одновременный вывод информации в журнал работы и системную консоль ОС.

Иначе можно в конфигурационном файле walg.json указать в качестве получателя сообщений стандартный системный вывод:

```
"WALG_LOG_DESTINATION": "stderr"
```



Журнал поддерживает следующие уровни детализации событий:

- ERROR – только критические сбои и события;
- NORMAL – основные этапы операций резервного копирования/восстановления;
- DEVEL – подробная отладочная информация.

Уровень детализации журнала работы компонента указывается при помощи параметров конфигурационного файла (см. п.п. 2.2.11), например:

```
"WALG_LOG_LEVEL": "NORMAL"
```



Если параметр не WALG\_LOG\_LEVEL не определен в конфигурационном файле используется внутреннее значение NORMAL.

#### 4. УДАЛЕНИЕ РАСШИРЕНИЯ

Удаление компонента должна производиться от имени пользователя, обладающего административными привилегиями в ОС.

Удаление компонента вместе с установленными конфигурационными файлами выполняется при помощи стандартных средств управления программным обеспечением ОС GNU/Linux.

В качестве примера будет приведена процедура удаления компонента в ОС Ubuntu.

Для удаления компонента необходимо выполнить следующую команду:

```
# apt-get purge jatoba6-wal-g
```

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

После удаления необходимо удалить конфигурационный файл `walg.json` при помощи следующей команды:

```
# rm /var/lib/jatoba/.walg.json
```

После удаления компонента «wal-g» необходимо удалить пользователя СУБД backup.

## **ПЕРЕЧЕНЬ СОКРАЩЕНИЙ**

NFS	–	Network File System
SSH	–	Secure Shell
WAL	–	Write-Ahead Logging
БД	–	База данных
ОС	–	Операционная система
РК	–	Резервная копия
СУБД	–	Система управления базами данных

[illegible]

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