Утвержден

РУСБ.10015-37-УД

ОПЕРАЦИОННАЯ СИСТЕМА СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ «ASTRA LINUX SPECIAL EDITION»

Руководство администратора.Часть 2

РУСБ.10015-37 95 01-2

Листов 29

2022

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

АННОТАЦИЯ

Настоящий документ является второй частью руководства администратора программного изделия РУСБ.10015-37 «Операционная система специального назначения «Astra Linux Special Edition» (далее по тексту — ОС).

В документе приведено описание защищенной СУБД, реализованной путем доработки СУБД PostgreSQL версии 11 в части реализации мандатного управления доступом.

Основные сведения по синтаксису языка запросов SQL, поддерживаемым типам данных, встроенным функциям, установке и настройке сервера СУБД, а также созданию отказоустойчивых решений приведены в официальной документации PostgreSQL, которая доступна на информационном ресурсе изготовителя `wiki.astralinux.ru`. Дополнительно особенности использования PostgreSQL детально описаны в официальной документации, поставляемой в составе пакете `postgresql-doc-x.x`.

Документ предназначен для администраторов и разработчиков баз данных.

СОДЕРЖАНИЕ

1. ЗАЩИЩЕННАЯ СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

1.1. Назначение

В качестве защищенной СУБД в составе ОС используется СУБД PostgreSQL, доработанная в соответствии с требованием интеграции с ОС в части мандатного управления доступом к информации и содержащая реализацию ДП-модели управления доступом и информационными потоками. Данная ДП-модель описывает все аспекты дискреционного, мандатного и ролевого управления доступом с учетом безопасности информационных потоков.

СУБД PostgreSQL предназначена для создания и управления реляционными БД и предоставляет многопользовательский доступ к расположенным в них данным.

Данные в реляционной БД хранятся в отношениях (таблицах), состоящих из строк и столбцов. При этом единицей хранения и доступа к данным является строка, состоящая из полей, идентифицируемых именами столбцов. Кроме таблиц, существуют другие объекты БД (виды, процедуры и т. п.), которые предоставляют доступ к данным, хранящимся в таблицах.

Для работы СУБД на диске выделяется область для хранения БД, называемая «кластером БД». Кластер БД является набором БД, управляемых одним экземпляром сервера СУБД. Настройка работы отдельного экземпляра сервера СУБД также определяется в рамках кластера соответствующими конфигурационными файлами.

Корректная работа с СУБД предполагает использование механизма ЕПП.

1.2. Состав

СУБД PostgreSQL состоит из следующих компонентов:

- `postgresql` — сервисная служба, реализующая непосредственно сервер БД;
- `libpq` — клиентская библиотека, предоставляющая доступ к серверу СУБД;
- набор серверных утилит для управления работой сервера и создания кластеров БД;
- набор клиентских утилит для создания и управления БД.

Для единообразного управления кластерами БД, функционирующими под управлением разных версий СУБД, в состав ОС входит набор инструментов администрирования `postgresql-common`, включающий в себя следующие пакеты:

- `postgresql` — метапакет, устанавливающий основную версию СУБД;
- `postgresql_x.x` — метапакет, устанавливающий конкретную версию СУБД;
- `postgresql-client` — метапакет, устанавливающий основную версию клиентских утилит СУБД;

- postgresql-client_x.x — метапакет, устанавливающий конкретную версию клиентских утилит СУБД;

- postgresql-contrib — метапакет, устанавливающий основную версию модулей расширения СУБД;

- postgresql-contrib_x.x — метапакет, устанавливающий конкретную версию модулей расширения СУБД;

- postgresql-doc_x.x — метапакет, устанавливающий конкретную версию документации СУБД;

- postgresql-common — инструменты управления кластерами БД;

ВНИМАНИЕ! В дальнейшем настоятельно рекомендуется управлять СУБД с помощью инструментов управления кластерами БД (см. 1.5), предоставляемыми ОС, а не непосредственно утилитами СУБД.

ВНИМАНИЕ! В состав СУБД входит набор модулей расширения, добавляющих новые типы данных, библиотеки функций, инструменты администрирования и разработки. Модули, предназначенные для администрирования и разработки, включая средства отладки хранимых процедур, исследования страниц данных, изменения конфигурационных файлов и доступа к внешним данным, не должны быть установлены в действующих системах, или доступ к ним непривилигированным пользователям должен быть ограничен.

В состав ОС входит графическая утилита pgadmin3, предназначенная для администрирования БД СУБД PostgreSQL, включая управление мандатным управлением доступом к объектам БД.

1.2.1. Дополнительно поставляемые модули

Список дополнительно поставляемых модулей приведен в таблице 1:

Таблица 1

Модуль	Описание
adminpack	Функции администрирования
auth_delay	Позволяет устанавливать паузы между аутентификациями пользователей
auto_explain	Функции для автоматического EXPLAIN запросов
btree_gin	Операторный класс GIN
btree_gist	Индексный операторный класс GiST
chkpass	Тип данных для хранения зашифрованных паролей
citext	Строковый тип данных, нечувствительный к регистру
cube	Тип многомерного куба
dblink	Фукнции для подключения к другим базам данных из текущей сессии

Продолжение таблицы 1

Модуль	Описание
dict_int	Пример реализации расширяемого шаблона словаря для полнотекстового поиска
dict_xsyn	Пример реализации расширяемого шаблона словаря синонимов для полнотекстового поиска
dummy_seclabel	Пример реализации поставщика меток для объектов баз данных
earthdistance	Функции для расчета расстояния между двумя точками земной поверхности
file_fdw	Обертка внешних данных (FOREIGN DATA WRAPPER) для файлов
fuzzystrmatch	Функции лексического сравнения строк
hstore	Тип данных для хранения пар ключ-значение
intagg	Агрегирующие функции для целого (integer) и перечисляемого (enum) типов данных
intarray	Функции и операторы для работы с непустыми массивами целых чисел
isn	Тип данных для хранения значений различных международных стандартов
lo	Функции для управления большими объектами (LARGE OBJECT)
ltree	Тип данных, позволяющий хранить метки объектов в виде иерархической структуры данных наподобие дерева
orafce	Функции для миграции с СУБД Oracle на PostgreSQL
pageinspect	Функции для получения информации о содержании страниц в базах данных
passwordcheck	Позволяет проверять пользовательские пароли, устанавливаемые командами ALTER ROLE и CREATE ROLE на простоту
pg_buffercache	Функции для получения текущей информации о разделяемом кеше
pg_freespacemap	Функции для получения информации о свободном пространстве отношения
pg_prewarm	Функции для загрузки всего отношения в память
pgrowlocks	Функции для получения информации о блокировках отношений
pg_stat_statements	Функции для получения информации о статистике исполнения запросов сервера
pgstattuple	Функции для получения различной информации о статистике записей
pg_trgm	Функции для сравнения текста
pldebugger	Функции для отладки запросов, написанных на pl/psql
plproxy	Язык для удаленного вызова функций на серверах баз данных
postgres_fdw	Внешняя обертка данных FOREIGN DATA WRAPPER для PostgreSQL
seg	Тип данных для представления интервалов точек
smlar	Функции для определения идентичности массивов
spi	Примеры функций SPI

Окончание таблицы 1

Модуль	Описание
sslinfo	Функции для получения информации о SSL сертификатах
tablefunc	Функции, возвращающие таблицы
tcn	Триггерные функции, высылающие асинхронные сообщения слушателям
test_decoding	Пример плагина логического декодирования
test_parser	Пример парсера для полнотекстового поиска
test_shm_mq	Пример использования динамически разделяемой памяти для совместного использования серверов
tsearch2	Функции полнотекстового поиска
unaccent	Функции полнотекстового поиска, удаляющие все диакритические знаки
uuid-ossp	Функции для генерации уникальных идентификаторов (UUID)
xml2	Функции для работы с XPath и XSLT

Примечания:

1. Расширение smlar содержится в пакете postgresql-11.x-smlar.
2. Расширение orafce содержится в пакете postgresql-11.x-orafce.
3. Расширение pldebugger содержится в пакете postgresql-11.x-pldebugger.
4. Дополнительно поставляется поддержка 1С в пакете postgresql-contrib-11.x.
5. Расширение plproxy содержится в пакете postgresql-11.x-plproxy.
6. Все остальные расширения содержатся в пакете postgresql-contib-11.x.

1.3. Настройка

Настройка сервера СУБД осуществляется установкой параметров в конфигурационном файле postgresql.conf. В дополнение к файлу postgresql.conf в PostgreSQL используется еще два конфигурационных файла, которые контролируют аутентификацию клиента. По умолчанию все эти три файла находятся в каталоге данных кластера БД или в соответствующем кластеру конфигурационном каталоге, например /etc/postgresql/x.x/main. За расположение указанных файлов отвечают конфигурационные параметры, приведенные в таблице 2.

Примечание. Некоторые способы использования СУБД (например, организация сервера для 1С) могут требовать дополнительной настройки сервера СУБД и ОС.

Примечание. Особенности настройки аутентификации для работы в текущей ОС приведены в 1.4.

Таблица 2

Параметр	Описание
data_directory	Определяет каталог для хранения данных
config_file	Определяет основной конфигурационный файл сервера (postgresql.conf). Значение этого параметра может быть задано только в командной строке postgres
hba_file	Определяет конфигурационный файл для аутентификации по узлам (pg_hba.conf)
ident_file	Определяет конфигурационный файл для аутентификации по методу ident (pg_ident.conf)
external_pid_file	Определяет имя дополнительного файла с идентификатором процесса, который сервер создает для использования программами администрирования сервера

Для настройки работы сервера с мандатным управлением доступом существует ряд конфигурационных параметров, указываемых в конфигурационном файле postgresql.conf. Описание параметров приведено в документе РУСБ.10015-37 97 01-1 «Операционная система специального назначения «Astra Linux Special Edition». Руководство по КСЗ. Часть 1».

При использовании относительного пути для задания значений этих параметров путь будет отсчитываться от каталога, в котором запущен postgres.

За установку соединений отвечают конфигурационные параметры, приведенные в таблице 3.

Таблица 3

Параметр	Описание
listen_addresses	Определяет TCP/IP-адреса, по которым сервер должен ожидать соединения от клиентских приложений. Значение формируется в виде перечня разделенных запятой имен узлов и/или числовых IP-адресов. Специальный знак * соответствует всем доступным IP-адресам. Если список пуст, сервер не слушает ни один IP-интерфейс. В этом случае установка соединения с сервером возможна только с использованием доменных сокетов UNIX. По умолчанию значением параметра является localhost, которое позволяет создавать только локальные loopback-соединения. **ВНИМАНИЕ!** В настоящее время отсутствует поддержка протокола IPv6. Задаваемые в этом параметре адреса должны соответствовать протоколу IPv4

Окончание таблицы 3

Параметр	Описание
`port`	Определяет TCP-порт, на котором сервер должен ожидать соединения от клиентских приложений (по умолчанию — `5432`). Следует отметить, что для всех IP-адресов, указанных в `listen_addresses (string)`, используется один и тот же порт
`max_connections`	Определяет максимальное число одновременных соединений с сервером СУБД
`superuser_reserved_connections`	Определяет число соединений, зарезервированных для подключения суперпользователей PostgreSQL
`unix_socket_group`	Определяет группу, владеющую доменным сокетом UNIX. Владельцем сокета всегда является пользователь, запускающий сервер. Используется в комбинации с параметром `unix_socket_permissions`, как дополнительный механизм контроля доступа для соединений в домене UNIX
`unix_socket_permissions`	Определяет права доступа для доменного сокета UNIX. Для доменного сокета UNIX имеет значение только разрешение на запись, и следовательно нет необходимости устанавливать или удалять разрешение на чтение или выполнение. Данный механизм контроля доступа не зависит от механизма аутентификации клиента

1.4. Настройки аутентификации

При попытке соединения с сервером СУБД клиентское приложение указывает пользователя СУБД PostgreSQL, от имени которого осуществляется подключение. В пределах окружения SQL активное имя пользователя СУБД определяет права на объекты БД.

PostgreSQL предлагает несколько различных методов аутентификации клиента. Метод, используемый для аутентификации конкретного клиентского соединения, может быть выбран на основе адреса узла сети клиента, БД и пользователя.

Несмотря на то, что имена пользователей СУБД PostgreSQL логически отделены от имен пользователей ОС, в которой запущен сервер, в соответствии с требованиями по защите информации от НСД требуется сопоставление пользователей СУБД пользователям ОС. Таким образом, при настройке аутентификации в СУБД следует использовать только методы аутентификации, в которых осуществляется подобное сопоставление. Для других пользователей осуществляется доступ только к незащищенной информации.

ВНИМАНИЕ! Для обеспечения нормальной работы пользователя с сетевыми сервисами должны быть явно заданы диапазоны его мандатных уровней и категорий с помощью соответствующих утилит, даже если ему не доступны уровни и категории выше 0.

Для корректного функционирования авторизации необходимо пользователю, от

которого работает СУБД PostgreSQL (по умолчанию — `postgres`), предоставить права на чтение информации из БД пользователей и сведений о метках безопасности и привилегиях:

```
usermod -a -G shadow postgres
setfacl -d -m u:postgres:r /etc/parsec/macdb
setfacl -R -m u:postgres:r /etc/parsec/macdb
setfacl -m u:postgres:rx /etc/parsec/macdb
setfacl -d -m u:postgres:r /etc/parsec/capdb
setfacl -R -m u:postgres:r /etc/parsec/capdb
setfacl -m u:postgres:rx /etc/parsec/capdb
```

1.4.1. Использование PAM аутентификации

Для настройки PAM в PostgreSQL необходимо создать локальные учетные записи пользователей и назначить им минимальные и максимальные уровни конфиденциальности и категории конфиденциальности. В параметре конфигурации `pg_hba.conf` необходимо указать требуемый метод аутентификации в последнем столбце, например:

```
host all all 0.0.0.0/0 pam
```

1.4.2. Использование сквозной аутентификации в ЕПП

Для обеспечения сквозной аутентификации пользователей ЕПП (см. документ РУСБ.10015-37 95 01-1«Операционная система специального назначения «Astra Linux Special Edition». Руководство администратора. Часть 1») в СУБД необходимо в качестве метода аутентификации указать `gss` и провести соответствующую настройку сервера и клиента СУБД PostgreSQL.

Для работы СУБД PostgreSQL в ЕПП необходимо выполнение следующих условий:

1) наличие в системах, на которых функционируют сервер и клиенты СУБД PostgreSQL, установленного пакета клиента ALD — `ald-client`;

2) разрешение имен должно быть настроено таким образом, чтобы имя системы разрешалось, в первую очередь, как полное имя (например, `myserver.example.ru`);

3) клиент ALD должен быть настроен на используемый ALD домен.

Для проведения операций по настройке ALD и администрированию Kerberos необходимо знание паролей администраторов ALD и Kerberos.

1.4.2.1. Сервер

Для обеспечения совместной работы сервера СУБД PostgreSQL с ALD необходимо, чтобы сервер СУБД PostgreSQL функционировал как сервис Kerberos. Выполнение данного условия требует наличия в БД Kerberos принципала для сервера СУБД PostgreSQL, имя которого задается в формате:

```
servicename/hostname@realm
```

где имя сервиса `servicename` соответствует имени учетной записи пользователя, от ко-

торой осуществляется функционирование сервера СУБД PostgreSQL (по умолчанию — `postgres`), и указывается в конфигурационном файле сервера PostgreSQL как значение параметра `krb_srvname`. В качестве значения `hostname` указывается полное доменное имя системы, на которой функционирует сервер СУБД PostgreSQL, а в качестве значения `realm` — имя домена ALD.

Существует два варианта настройки работы сервера СУБД PostgreSQL с ALD:

1) все кластера используют один принципал Kerberos;

2) кластера используют разные прицепалы Kerberos.

Для настройки сервера СУБД первым способом необходимо выполнить следующие действия:

1) создать в БД ALD с помощью утилиты администрирования ALD принципала, соответствующего устанавливаемому серверу PostgreSQL. Принципал создается с автоматически сгенерированным случайным ключом:

```
ald-admin service-add postgres/server.my_domain.org
```

2) ввести созданного принципала в группу сервисов `mac`, используя следующую команду:

```
ald-admin sgroup-svc-add postgres/server.my_domain.org --sgroup=mac
```

3) создать файл ключа Kerberos для сервера СУБД PostgreSQL с помощью утилиты администрирования ALD `ald-client`, используя следующую команду:

```
ald-client update-svc-keytab postgres/server.my_domain.org
--ktfile="/etc/postgresql-common/krb5.keytab"
```

Права доступа к этому файлу должны позволять читать его пользователю, от имени которого работает сервер СУБД PostgreSQL (как правило, владельцем файла назначается пользователь `postgres`);

4) сменить владельца, полученного на предыдущем шаге, файла `krb5.keytab` на пользователя `postgres`, выполнив следующую команду:

```
chown postgres /etc/postgresql-common/krb5.keytab
```

5) указать для внешних соединений в конфигурационном файле сервера СУБД PostgreSQL `/etc/postgresql/x.x/main/pg_hba.conf` метод аутентификации `gss`.

```
host all all 192.168.32.0/24 gss
```

Для настройки сервера СУБД вторым способом необходимо выполнить следующие действия:

1) создать в БД ALD с помощью утилиты администрирования ALD принципала для кластера СУБД. Принципал создается с автоматически сгенерированным случайным ключом:

```
ald-admin service-add имя_принципала/server.my_domain.org
```

2) ввести созданного принципала в группу сервисов mac, используя следующую команду:

```
ald-admin sgroup-svc-add имя_принципала/server.my_domain.org --sgroup=mac
```

3) создать файл ключа Kerberos для сервера СУБД PostgreSQL с помощью утилиты администрирования ALD ald-client, используя следующую команду (пример приведен для кластера БД по умолчанию):

```
ald-client update-svc-keytab имя_принципала/server.my_domain.org
--ktfile="/etc/postgresql/11/main/krb5.keytab"
```

Полученный файл должен быть доступен серверу СУБД PostgreSQL по пути, указанному в конфигурационном параметре krb_server_keyfile (в данном случае — /etc/postgresql/11/main/krb5.keytab). Права доступа к этому файлу должны позволять читать его пользователю, от имени которого работает сервер СУБД PostgreSQL (как правило, владельцем файла назначается пользователь postgres);

4) сменить владельца, полученного на предыдущем шаге, файла krb5.keytab на пользователя postgres, выполнив следующую команду:

```
chown postgres /etc/postgresql/11/main/krb5.keytab
```

5) задать в конфигурационном файле сервера СУБД PostgreSQL /etc/postgresql/11/main/postgresql.conf путь к ключу сервера СУБД PostgreSQL:

```
krb_server_keyfile = '/etc/postgresql/11/main/krb5.keytab'
```

6) указать для внешних соединений в конфигурационном файле сервера СУБД PostgreSQL /etc/postgreslq/x.x/main/pg_hba.conf метод аутентификации gss.

```
host all all 192.168.32.0/24 gss
```

Примечание. Для подключения к серверу СУБД, имя принципала которого отличается от имени принципала по умолчанию (postgres), используется следующая команда:

```
psql "dbname=db user=dbuser password=password host=astra.test.ru port=port
krbsrvname=имя_принципала"
```

1.4.2.2. Клиент

Общие условия, при которых обеспечивается совместное функционирование клиентов СУБД PostgreSQL с ALD, см. в 1.4.2. Кроме того, сервер СУБД PostgreSQL должен быть также настроен соответствующим образом (см. 1.4.2.1). Для настройки клиента СУБД PostgreSQL необходимо:

1) создать в БД ALD учетную запись пользователя, зарегистрированного в СУБД PostgreSQL (например, pgusername). Дополнительная информация приведена в руководстве man на ALD;

2) задать в качестве значения параметра соединения krbsrvname имя сервиса

servicename, используемое при создании принципала сервера СУБД PostgreSQL. Имя принципала сервера СУБД PostgreSQL задается в формате:

servicename/hostname@realm

где servicename — обычно имя учетной записи сервера СУБД PostgreSQL, используемое при создании принципала сервера СУБД PostgreSQL (по умолчанию — postgres), а hostname — полное доменное имя системы, на которой функционирует сервер СУБД PostgreSQL.

1.5. Управление кластерами СУБД

Несмотря на возможность управления серверами СУБД с помощью штатных утилит PostgreSQL, настоятельно рекомендуется использовать для этого поставляемыми с ОС инструменты управления кластерами СУБД PostgreSQL. Данные утилиты входят в состав пакета postgresql-common и представляют собой скрипты, осуществляющие вызов штатных утилит PostgreSQL, но с учетом специфики ОС и возможности одновременного управления кластерами разных версий СУБД.

Данные утилиты позволяют создать несколько экземпляров кластеров разных версий с собственными конфигурационными файлами. Правкой конфигурационных файлов кластера можно указать порт, IP-адреса, которые будет слушать кластер и т.д.

К описываемым утилитам относятся:

- pg_createcluster — утилита по созданию нового кластера (см. 1.5.1);
- pg_ctlcluster — утилита по управлению кластером (см. 1.5.2);
- pg_dropcluster — утилита по удалению кластера (см. 1.5.3);
- pg_lsclusters — утилита по просмотру состояний существующих кластеров (см. 1.5.4);
- pg_upgradecluster — утилита по обновлению кластера (см. 1.5.5).

Утилиты используют единый формат вызова следующего вида:

pg_<имя утилиты> [опции] версия-кластера имя-кластера <действие>

В зависимости от назначения утилиты некоторые элементы командной строки вызова могут отсутствовать.

Общий подход заключается в физическом разнесении файлов разных кластеров в пределах файловой системы. Как правило, используются следующее расположение:

- /usr/lib/postgresql/версия-СУБД/ — загружаемые модули СУБД определенной версии;
- /usr/share/postgresql/версия-СУБД/ — разделяемые файлы СУБД определенной версии;
- /etc/postgresql/версия-кластера/имя-кластера — конфигурационные файлы конкретного кластера;

- /var/lib/postgresql/версия-кластера/имя-кластера — непосредственно каталог данных конкретного кластера.

Далее приведено описание использования перечисленных утилит. Более подробная информация может быть найдена в руководстве man для каждой из утилит.

1.5.1. Создание кластера - pg_createcluster

Для создания кластера СУБД применяется утилита pg_createcluster. Утилита принимает в качестве аргументов версию и имя требуемого кластера. При этом создается кластер с настройками по умолчанию. В конечном счете вызывается утилита СУБД PostgreSQL initdb, которой передаются принятые в ОС расположения файлов и текущие значения локали. Существует возможность изменения поведения по умолчанию с помощью опций.

Формат вызова:

```
pg_createcluster [опции] версия-кластера имя-кластера [--port PORT]
```

Например, для создания кластера main, работающего на порту 5433, для СУБД PostgreSQL версии 11 необходимо выполнить:

```
pg_createcluster 11 main --port 5433
```

1.5.2. Управление кластером - pg_ctlcluster

Для управления кластером СУБД применяется утилита pg_ctlcluster. Утилита принимает в качестве аргументов версию, имя требуемого кластера и одно из действий:

- start — запуск экземпляра СУБД для указанного кластера;
- stop — останов экземпляра СУБД для указанного кластера;
- restart — перезапуск экземпляра СУБД для указанного кластера;
- reload — повторное чтение конфигурационных файлов указанного кластера сервером СУБД;
- promote — специальная команда перевода резервного сервера из состояния восстановления в состояние приема запросов.

В конечном счете вызывается утилита СУБД PostgreSQL pg_ctl, которой передаются принятые в ОС расположения файлов. Существует возможность изменения поведения по умолчанию с помощью опций.

Формат вызова:

```
pg_ctlcluster [опции] версия-кластера имя-кластера действие -- [опции pg_ctl]
```

Например, для перезапуска кластера main для СУБД PostgreSQL версии 11 необходимо выполнить:

```
pg_ctlcluster 11 main restart
```

1.5.3. Удаление кластера - pg_dropcluster

Для удаления кластера СУБД применяется утилита pg_dropcluster. Утилита принимает в качестве аргументов версию и имя требуемого кластера. При этом удаляются все файлы кластера. Указание опции --stop приводит к остановке сервера СУБД перед удалением файлов кластера.

Формат вызова:

```
pg_dropcluster [--stop] версия-кластера имя-кластера
```

Например, для удаления кластера main для СУБД PostgreSQL версии 11 необходимо выполнить:

```
pg_dropcluster 11 main
```

1.5.4. Просмотр состояние кластеров - pg_lsclusters

Для просмотра состояния существующих кластеров СУБД применяется утилита pg_lsclusters.

Формат вызова:

```
pg_lsclusters
```

Утилита выводит список существующих кластеров в следующем виде:

```
# pg_lsclusters
Ver Cluster Port Status Owner    Data directory              Log file
11  main    5432 online postgres /var/lib/postgresql/11/main /var/log/...
```

1.5.5. Обновление кластера - pg_upgradecluster

pg_upgradecluster обновляет существующий кластер PostgreSQL на новую версию кластера, указанную с помощью параметра newversion (по умолчанию: последняя доступная версия). Конфигурационные файлы копируются со старого кластера на новый.

Формат вызова:

```
/usr/bin/pg_upgradecluster [опции] старая-версия имя-кластера
  [новый-каталог-данных]
```

ВНИМАНИЕ! После завершения процесса обновления необходимо установить параметр ac_auto_adjust_macs в значение false в конфигурационном файле postgresql.conf «нового» кластера.

1.6. Особенности управления разграничением доступа

Подробное описание и расширения языка SQL для управления мандатным управлением доступом приведены в документе РУСБ.10015-37 97 01-1.

1.6.1. Мандатное управление доступом

В основе механизма мандатного управления доступом лежит управление доступом к защищаемым ресурсам БД на основе иерархических и неиерархических меток доступа.

Это позволяет реализовать многоуровневую защиту с обеспечением разграничения доступа пользователей к защищаемым ресурсам БД и управление потоками информации. В качестве иерархических и неиерархических меток доступа при использовании СУБД в ОС используются метки безопасности ОС.

Для хранения метки объекта БД введено служебное поле `maclabel`. При создании любого объекта БД, он маркируется меткой безопасности пользователя текущей сессии БД. В дальнейшем по этой метке производится разграничение доступа к созданному объекту.

СУБД PostgreSQL не имеет собственного механизма назначения, хранения и модификации меток пользователей и использует для этого механизмы ОС.

1.6.2. Ограничения при использовании ролевого управления доступом

СУБД PostgreSQL была доработана для обеспечения соответствия требованиям по защите информации от несанкционированного доступа. При реализации названных требований была установлена необходимость обеспечения соответствия пользователей СУБД учетным записям в ОС.

Введены дополнительные параметры, ограничивающие распространение прав доступа (см. РУСБ.10015-37 97 01-1).

1.6.3. Особенности обновления кластера при использовании `pg_upgrade`

`pg_upgrade` — штатная утилита по обновлению кластеров PostgreSQL, входящая в состав пакета `postgresql-contrib-11.x`.

Из-за различий моделей мандатного управления доступом при выполнении обновления с помощью `pg_upgrade` на версию `11` необходимо включить флаг автоматического подъема метки безопасности контейнерных объектов баз данных (параметр `ac_alow_auto_adjust_mac = on`). При штатной эксплуатации СУБД данный параметр должен быть **отключен**.

Для обновления кластера со следующими обозначениями:

- `NEW_VERSION` — «новая» версия кластера;
- `NEW_CLUSTER_NAME` — имя «нового» кластера;
- `OLD_BIN` — путь к «старому» каталогу с исполняемыми файлами PostgreSQL;
- `NEW_BIN` — путь к «новому» каталогу с исполняемыми файлами PostgreSQL;
- `OLD_CONF` — путь к каталогу конфигурационных файлов «старого» кластера;
- `NEW_CONF` — путь к каталогу конфигурационных файлов «нового» кластера;
- `OLD_PORT` — порт «старого» кластера PostgreSQL;
- `NEW_PORT` — порт «нового» кластера PostgreSQL

предусмотрен следующий типичный сценарий:

1) создать новый кластер:

```
$ sudo pg_createcluster PG_NEW_VERSION PG_NEW_CLUSTER_NAME
```

2) в конфигурационном файле `postgresql.conf` нового кластера указать

```
ac_allow_auto_adjust_mac = on
```

3) убедиться, что «старый» и «новый» кластера выключены. Если это не так, выключить их с помощью утилиты `pg_ctlcluster`;

4) используя утилиту `pg_upgrade`, выполнить процесс переноса данных со «старого» на «новый» кластер (команду необходимо выполнить от имени пользователя `postgres`):

```
$ /usr/lib/postgresql/11/bin/pg_upgrade -b PG_OLD_BIN -B
PG_NEW_BIN -d PG_OLD_CONF -D PG_NEW_CONF -p PG_OLD_PORT -P
PG_NEW_PORT}
```

ВНИМАНИЕ! После завершения процесса обновления необходимо установить параметр `ac_auto_adjust_macs` в значение `false` в конфигурационном файле `postgresql.conf` «нового» кластера.

1.7. Средства администрирования и обеспечения надежности

1.7.1. Настройка репликации

Репликация — механизм синхронизации содержимого нескольких копий баз данных.

PostgreSQL предоставляет несколько способов репликации кластеров:

- пофайловая — основана на покомандном копировании WAL-файлов (1.7.1.1);
- потоковая — основана на транслировании потока WAL-файлов ведущего сервера на ведомый (1.7.1.2);
- слоты репликации — модификация потоковой репликации, позволяющая реплицировать содержимое одного ведущего сервера на несколько ведомых. В этом режиме ведущий сервер не удаляет журналы транзакций до тех пор, пока ведомый (ведомые) сервер (серверы) не отчитается (отчитаются) об их применении; (1.7.1.3)
- логическая — основана на пересылке изменений состояния данных ведущего сервера и применении их на ведомом;

Кроме того, возможна репликация таблиц посредством асинхронной системы репликации `Slony-I` (1.7.1.4).

При описании различных способов настройки репликации используются следующие обозначения:

- `VERSION` — версия СУБД PostgreSQL;
- `MASTER` — имя кластера, являющегося ведущим сервером;
- `MASTER_IP` — IP-адрес ведущего сервера;
- `MASTER_PORT` — порт работы ведущего сервера;
- `SLAVE` — имя кластера, являющегося ведомым сервером;

- SLAVE_IP — IP-адрес ведомого сервера;
- SLAVE_PORT — порт работы ведомого сервера;
- REPL — имя пользователя, обладающего правами репликации (REPLICATION), от имени которого реализуется репликация;
- WAL_DIR — каталог для хранения WAL-файлов ведущего сервера.
- REPL_PASSWORD — пароль пользователя, от имени которого реализуется репликация;
- MASTER_DB — база данных, из которой реплицируются данные;
- SLAVE_DB — база данных, в которую реплицируются данные;

Для выполнения команд переноса WAL-файлов с ведущего на ведомый сервер требуется утилита rsync.

1.7.1.1. Настройка пофайловой (Log Shipping) репликации

Для настройки пофайловой репликации необходимо выполнить следующие действия:

1) остановить ведомый кластер баз данных:

```
$ sudo pg_ctlcluster $VERSION $SLAVE stop
```

2) создать каталог для хранения WAL файлов:

```
$ sudo mkdir $WAL_DIR
```

3) назначить владельцем этого каталога пользователя postgres:

```
$ sudo postgres.postgres $WAL_DIR
```

4) установить следующие параметры в postgresql.conf ведущего сервера:

```
wal_level = replica
archive_mod = on
archive_command = 'cp %p $WAL_DIR/%f < /dev/null'
archive_timeout = 1
```

5) удалить все файлы в каталоге ведомого сервера за исключением pg_audit.conf:

```
$ sudo cd /var/lib/postgresql/$VERSION/$SLAVE/ &&
    sudo ls grep -v pg_audit.conf | xargs rm -rf
```

6) выполнить операции по созданию резервной копии ведущего сервера и перенести ее на ведомый (выполняются от имени пользователя postgres):

```
$ psql -h $MASTER_IP -p $MASTER_PORT -U $REPL -d template1 -c "SELECT
pg_start_backup('initial');"
$ rsync -a /var/lib/postgresql/$VERSION/$MASTER/*
postgres@$SLAVE_IP:/var/lib/postgresql/$VERSION/$SLAVE/
--exclude pg_log/* --exclude pg_xlog/* --exclude postmaster.* --exclude
pg_audit.conf >> /dev/null
```

```
$ psql -h $MASTER_IP -p $MASTER_PORT -U $REPL -d template1 -c "SELECT
pg_stop_backup();"
```

Примечание. Если сервера находятся на одном хосте, то можно использовать следующий вариант команды `rsync`:

```
$ rsync -a /var/lib/postgresql/$VERSION/$MASTER/*
/var/lib/postgresql/$VERSION/$SLAVE/ --exclude
pg_log/* --exclude pg_xlog/* --exclude postmaster.*
--exlude pg_audit.conf >> /dev/null
```

7) установить следующие параметры в `postgresql.conf` ведомого сервера:

```
hot_standby = on
```

8) создать конфигурационный файл `recovery.conf` на ведомом сервере (выполняется от имени пользователя `postgres`):

```
$ touch /var/lib/postgresql/$VERSION/$SLAVE/recovery.conf
```

9) в созданный файл `recovery.conf` добавить следующие строки:

```
restore_command = '/usr/lib/postgresql/$VERSION/bin/pg_standby -d -t
/tmp/trigger.$VERSION.$SLAVE_PORT $WAL_DIR %f %p %r 2'
recovery_end_command = 'rm -f /tmp/trigger.$VERSION.$SLAVE_PORT'
trigger_file = '/tmp/trigger_$SLAVE.$SLAVE_PORT'
```

10) выполнить запуск ведомого сервера:

```
$ sudo pg_ctlcluster $VERSION $SLAVE start
```

1.7.1.2. Настройка потоковой (Streaming) репликации

Для настройки потоковой репликации необходимо выполнить следующие действия:

1) остановить ведомый кластер баз данных:

```
$ sudo pg_ctlcluster $VERSION $SLAVE stop
```

2) установить следующие параметры в `postgresql.conf` ведущего сервера:

```
listen_addresses = '*'
wal_level = replica
max_wal_senders = 1
wal_keep_segments = 32
```

3) настроить аутентификацию пользователя `$REPL`, разрешив ему подключаться к базе данных `replication` с IP-адреса `$SLAVE_IP`, добавив следующую строку в `pg_hba.conf` ведущего сервера:

```
host replication $REPL $SLAVE_IP trust
```

Примечание. Вместо `trust` должен быть использован более надежный метод аутентификации.

4) удалить все файлы в каталоге ведомого сервера за исключением `pg_audit.conf`:

```
$ sudo -s
```

```
ls /var/lib/postgresql/$VERSION/$SLAVE/ | grep -v pg_audit.conf |
     xargs rm -rf
```

5) выполнить операции по созданию резервной копии ведущего сервера и перенести ее на ведомый (выполняются от имени пользователя `postgres`):

```
$ psql -h $MASTER_IP -p $MASTER_PORT -U $REPL -d template1 -c "SELECT
pg_start_backup('initial');"
$ rsync -a /var/lib/postgresql/$VERSION/$MASTER/*
postgres@$SLAVE_IP:/var/lib/postgresql/$VERSION/$SLAVE/
--exclude pg_log/* --exclude pg_xlog/* --exclude postmaster.* --exclude
pg_audit.conf >> /dev/null
$ psql -h $MASTER_IP -p $MASTER_PORT -U $REPL -d template1 -c "SELECT
pg_stop_backup();"
```

Примечание. Если сервера находятся на одном хосте, то можно использовать следующий вариант команды `rsync`:

```
$ rsync -a /var/lib/postgresql/$VERSION/$MASTER/*
/var/lib/postgresql/$VERSION/$SLAVE/ --exclude
pg_log/* --exclude pg_xlog/* --exclude postmaster.*
--exlude pg_audit.conf >> /dev/null
```

6) установить следующие параметры в `postgresql.conf` ведомого сервера:

```
hot_standby = on
```

7) создать конфигурационный файл `recovery.conf` на ведомом сервере (выполняется от имени пользователя `postgres`):

```
$ touch /var/lib/postgresql/$VERSION/$SLAVE/recovery.conf
```

8) в созданный файл `recovery.conf` добавить следующие строки:

```
standby_mode = 'on'
primary_conninfo = 'host=$MASTER_IP port=$MASTER_PORT user=$REPL'
trigger_file = '/tmp/trigger_$SLAVE.$SLAVE_PORT'
```

9) выполнить запуск ведомого сервера:

```
$ sudo pg_ctlcluster $VERSION $SLAVE start
```

1.7.1.3. Настройка репликации с помощью слотов репликации

Для настройки репликации с помощью слотов репликации необходимо выполнить следующие действия:

1) остановить ведомый кластер баз данных:

```
$ sudo pg_ctlcluster $VERSION $SLAVE stop
```

2) установить следующие параметры в `postgresql.conf` ведущего сервера:

```
listen_addresses = '*'
wal_level = replica
max_wal_senders = 1
```

```
max_replication_slots = 1
```

3) настроить аутентификацию пользователя $REPL, разрешив ему подключаться к базе данных replication с IP-адреса $SLAVE_IP, добавив следующую строку в pg_hba.conf ведущего сервера:

```
host replication $REPL $SLAVE_IP trust
```

Примечание. Вместо trust должен быть использован более надежный метод аутентификации.

4) удалить все файлы в каталоге ведомого сервера за исключением pg_audit.conf:

```
$ sudo cd /var/lib/postgresql/$VERSION/$SLAVE/ &&
    sudo ls grep -v pg_audit.conf | xargs rm -rf
```

5) выполнить операции по созданию резервной копии ведущего сервера и перенести ее на ведомый (выполняются от имени пользователя postgres):

```
$ psql -h $MASTER_IP -p $MASTER_PORT -U $REPL -d template1 -c "SELECT
pg_start_backup('initial');"
$ rsync -a /var/lib/postgresql/$VERSION/$MASTER/*
postgres@$SLAVE_IP:/var/lib/postgresql/$VERSION/$SLAVE/
--exclude pg_log/* --exclude pg_xlog/* --exclude postmaster.* --exclude
pg_audit.conf >> /dev/null
$ psql -h $MASTER_IP -p $MASTER_PORT -U $REPL -d template1 -c "SELECT
pg_stop_backup();"
```

Примечание. Если сервера находятся на одном хосте, то можно использовать следующий вариант команды rsync:

```
$ rsync -a /var/lib/postgresql/$VERSION/$MASTER/*
/var/lib/postgresql/$VERSION/$SLAVE/ --exclude
pg_log/* --exclude pg_xlog/* --exclude postmaster.*
--exlude pg_audit.conf >> /dev/null
```

6) создать слот репликации на ведущем сервере:

```
$ psql -h $MASTER_IP -p $MASTER_PORT -U $REPL -d template1 -c "SELECT
pg_create_physical_replication_slot('slave');"
```

7) установить следующие параметры в postgresql.conf ведомого сервера:

```
hot_standby = on
```

8) создать конфигурационный файл recovery.conf на ведомом сервере (выполняется от имени пользователя postgres):

```
$ touch /var/lib/postgresql/$VERSION/$SLAVE/recovery.conf
```

9) в созданный файл recovery.conf добавить следующие строки:

```
standby_mode = 'on'
primary_conninfo = 'host=$MASTER_IP port=$MASTER_PORT user=$REPL'
```

```
trigger_file = '/tmp/trigger_$SLAVE.$SLAVE_PORT'
primary_slot_name = 'slave'
```

10) выполнить запуск ведомого сервера:

```
$ sudo pg_ctlcluster $VERSION $SLAVE start
```

1.7.1.4. Настройка репликации с помощью Slony-I

Slony-I — система асинхронной «Ведущий-ведомый» репликации для СУБД PostgreSQL. Поддерживает следующие возможности:

- поддержка нескольких ведомых узлов;
- поддержка частичной репликации, то есть репликации только определенного набора таблиц;
- поддержка репликации разных таблиц на разные ведомые узлы;
- поддержка репликации разных таблиц с разных мастеров на один ведомый узел.

Для настройки репликации с помощью Slony-I необходим предустановленный пакет slony1-2-bin, а также пакет postgresql-11.x-slony1-2 на ведущем и ведомом серверах, затем выполнить действия:

1) на ведущем и ведомом сервере создать пользователя, от имени которого будет происходить репликация с правом SUPERUSER;
2) на ведущем и ведомом серверам создать базы данных $MASTER_DB, $SLAVE_DB, а также таблицы, из которой и в которую будут реплицироваться данные.

Примечание. Slony-I поддерживает репликацию таблиц при условии, если они содержат первичный ключ (PRIMARY KEY), а также если таблицы имеют одинаковую структуру (одинаковое количество столбцов и совпадающие типы данных в них);

3) назначить мандатные атрибуты на объекты баз данных (если необходимо);
4) на ведущем сервере запустить службу slonik от имени пользователя postgres:

```
slonik <<_EOF_
  cluster name = slony_example;
  node 1 admin conninfo = 'dbname=$MASTER_DB host=$MASTER_IP
    port=$MASTER_PORT user=$REPL password=$REPL_PASSWORD';
  node 2 admin conninfo = 'dbname=$SLAVE_DB host=SLABE_IP
    port=$SLAVE_PORT user=$REPL password=$REPL_PASSWORD';
  init cluster ( id=1, comment = 'Master Node');
  create set (id=1, origin=1, comment='All tables');

  # Здесь указываются таблицы,
  # которые будут реплицироваться на ведомый сервер
  set add table (set id=1, origin=1,
    fully qualified name = 'public.test', comment='test');
```

```
  store node (id=2, comment = 'Slave node', event node=1);
  store path (server = 1, client = 2, conninfo='dbname=$MASTER_DB
    host=$MASTER_IP port=$MASTER_PORT user=$REPL
    password=REPL_PASSWORD');
  store path (server = 2, client = 1, conninfo='dbname=$SLAVE_DB
    host=SLABE_IP port=$SLAVE_PORT user=$REPL
    password=REPL_PASSWORD');

  store listen ( origin = 1, provider = 1, receiver = 2);
  store listen ( origin = 2, provider = 2, receiver = 1);
_EOF_
```

5) на ведомом сервере запустить службу `slonik` от имени пользователя `postgres`:

```
slonik <<_EOF_
  cluster name = slony_example;

 node 1 admin conninfo = 'dbname=$MASTER_DB host=$MASTER_IP
    port=$MASTER_PORT user=$REPL password=REPL_PASSWORD';
  node 2 admin conninfo = 'dbname=$SLAVE_DB host=SLABE_IP
    port=$SLAVE_PORT user=$REPL password=REPL_PASSWORD';

  subscribe set ( id = 1, provider = 1, receiver = 2, forward = no);
_EOF_
```

6) на ведущем сервере запустить службу репликации `slon` от имени пользователя `postgres`:

```
slon slony_example "dbname=$MASTER_DB port=$MASTER_IP
  host=$MASTER_IP user=$REPL password=$REPL_PASSWORD"
```

7) на ведомом сервере запустить службу репликации `slon` от имени пользователя `postgres`:

```
slon slony_example "dbname=$SLAVE_DB port=$SLAVE_IP
  host=$SLAVE_IP user=$REPL password=$REPL_PASSWORD"
```

ВНИМАНИЕ! Если метка безопасности таблиц, из которой реплицируются и в которую реплицируются, не совпадает, то процесс репликации данных таблицы будет остановлен. При этом в журнал службы репликации `slon` ведомого сервера будет добавлено следующее сообщение:

```
ERROR remoteWorkerThread_1: Table "public"."test" on provider does not match
on subscriber
```

В этом случае необходимо:

1) удалить схему кластера slony-I на ведомом сервере:

```
DROP SCHEMA _slony_example CASCADE;
```

2) скорректировать мандатные атрибуты реплицируемых таблиц;

3) перезапустить службы репликации slon на ведомом сервере.

1.7.2. Pgpool-II

Pgpool-II представляет собой прокси-прослойку между сервером и клиентами, прозрачную для обоих, и может организовывать пул для ограничения соединений. Реализует следующие возможности:

- создание высокопроизводительной сетевой структуры между узлами, кластерами и пользователями;
- синхронная репликация данных на множество серверов без остановки;
- балансировщик нагрузки между узлами кластера;
- Failover — обнаружение отказа и переключения нагрузки.

Настройка Pgpool-II заключается в изменении конфигурационных параметров файла /etc/pgpool2/pgpool.conf. Этот файл по структуре схож с конфигурационным файлом postgresql.conf СУБД PostreSQL и имеет синтаксис ключ = значение. Символом комментария является #.

Для настройки аутентификации используется конфигурационный файл /etc/pgpool2/pool_hba.conf, который имеет следующий синтаксис:

```
local      DATABASE  USER  METHOD  [OPTION]
host       DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTION]
```

Описание основных параметров конфигурационного файла приведено в 4.

Таблица 4 – Основные параметры конфигурационного файла pgpool2.conf

Параметр	Описание
listen_addresses	Определяет TCP/IP-адреса, по которым сервер должен ожидать соединения от клиентских приложений. Значение формируется в виде перечня разделенных запятой имен узлов и/или числовых IP-адресов. Специальный знак * соответствует всем доступным IP-адресам. Если список пуст, сервер не слушает ни один IP-интерфейс. В этом случае установка соединения с сервером возможна только с использованием доменных сокетов UNIX. По умолчанию значением параметра является *.
port	Определяет TCP-порт, на котором сервер должен ожидать соединения от клиентских приложений. Следует отметить, что для всех IP-адресов, указанных в listen_addresses (string), используется один и тот же порт.
backend_hostnameN	Определяет TCP/IP адрес N-го сервера PostgreSQL.
backend_portN	Определяет TCP-порт N-го сервера PostgreSQL.
backend_weightN	Определяет долю нагрузки на N-ый сервер PostgreSQL.

Окончание таблицы 4

Параметр	Описание
backend_data_directoryN	Определяет путь к каталогу PGDATA N-го сервера PostgreSQL.
enable_pool_hba	Если этот параметр установлен в on, то Pgpool-II будет использовать конфигурационный файл pool_hba.conf для аутентификации клиентов.
log_destination	Определяет, куда выводить сообщения. По умолчанию stderr, также поддерживается режим вывода в syslog.
load_balance_mode	Если этот параметр установлен в on, то Pgpool-II работает в режиме балансировки нагрузки серверов. По умолчанию off.
master_slave_mode	Если этот параметр установлен в on, то Pgpool-II будет работать в режиме Ведущий-ведомый. По умолчанию off.
master_slave_sub_mode	Определяет режим репликации серверов. Может принимать значения slony и streaming.
replication_mode	Если этот параметр установлен в on, то Pgpool-II будет отправлять копии запросов на все сервера. По умолчанию off.
replicate_select	Если этот параметр установлен в on, то Pgpool-II будет отправлять команды по выбору данных (SELECT) на все узлы. По умолчанию off.
sr_check_user	Определяет имя пользователя, проверяющего статус репликации. По умолчанию nobody.
sr_check_password	Определяет пароль пользователя, проверяющего статус репликации. По умолчанию используется пустое значение.

Для отображения различной информации Pgpool-II могут быть использованы следующие команды:

- SHOW pool_status; — получить информацию о конфигурации Pgpool-II;
- SHOW pool_nodes; — получить информацию об узлах;
- SHOW pool_processes; — получить информацию о процессах Pgpool-II;
- SHOW pool_pools; — получить информацию о пуле процессов;
- SHOW pool_version; — получить информацию о версии Pgpool-II.

1.7.2.1. Настройка аутентификации

Pgpool-II поддерживает следующие методы аутентификации: trust, reject, md5 и pam.

Для настройки аутентификации по методу md5 требуется выполнить следующие действия:

1) создать пароль для пользователя:

```
$ sudo pg_md5 --md5auth ПАРОЛЬ --username ИМЯ_ПОЛЬЗОВАТЕЛЯ
```

2) установить метод аутентификации md5 в pool_hba.conf;

3) перезапустить Pgpool-II:

```
$ sudo service pgpool2 restart
```

ВНИМАНИЕ! Если в процессе работы службы Pgpool-II был сменен пароль, то необходимо выполнить перезапуск службы.

Для настройки аутентификации по методу pam требуется скопировать файл-шаблон /usr/share/doc/pgpool2/examples/pgpool.pam в каталог /etc/pam.d, если необходимо, дополнить сценариями, после чего установить метод аутентификации pam в pool_hba.conf.

1.7.2.2. Настройка протоколирования

Для того, чтобы настроить протоколирование событий в файл необходимо выполнить следующие действия:

1) в конфигурационном файле /etc/pgpool2/pgpool.conf установить:

```
log_destination = 'syslog'
```

2) в конфигурационный файл /etc/rsyslog.conf службы протоколирования rsyslog добавить строку:

```
local0.* /var/log/pgpool.log
```

3) перезапустить службу протоколирования rsyslog:

```
$ sudo service rsyslog restart
```

4) перезапустить Pgpool-II:

```
$ sudo service pgpool2 restart
```

Теперь все сообщения от службы Pgpool-II будут заноситься в журнал /var/log/pgpool.log.

1.7.2.3. Настройка Pgpool-II в режиме Ведущий-ведомый

ВНИМАНИЕ! Настройка Pgpool-II в режиме «Ведущий-ведомый» может быть произведена только для кластеров одной версии.

Для настройки Pgpool-II в режиме «Ведущий-ведомый» необходимо:

1) выполнить настройку протоколирования согласно 1.7.2.2;

2) выполнить настройку репликации согласно 1.7.1.2 или 1.7.1.4;

3) в файл /etc/pgppol2/pgpool.conf добавить данные об узлах, например:

```
backend_hostname0 = '192.168.1.101'
backend_port0 = 5432
backend_weight0 = 1
backend_data_directory0 = '/var/lib/postgresql/11/master'

backend_hostname1 = '192.168.1.102'
backend_port1 = 5433
backend_weight1 = 1
backend_data_directory1 = '/var/lib/postgresql/11/slave'
```

4) в файле `/etc/pgpool2/pgpool.conf` изменить значение следующих параметров:

```
master_slave_mode = on
```

Также установить используемый режим репликации (параметр `master_slave_sub_mode`;

5) в файле `/etc/pgpool2/pgpool.conf` указать имя пользователя, проверяющего активность репликации, и его пароль (параметры `sr_check_user` и `sr_check_password` соответственно);

6) настроить аутентификацию согласно 1.7.2.1.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БД — база данных

ЕПП — единое пространство пользователей

КСЗ — комплекс средств защиты

НСД — несанкционированный доступ

ОС — операционная система специального назначения «Astra Linux Special Edition»

СУБД — система управления базами данных

ALD — Astra Linux Directory (единое пространство пользователей)

HBA — Host-based Authentication (аутентификация на основе адресов узлов сети)

IP — Internet Protocol (межсетевой протокол)

PAM — Pluggable Authentication Modules (подключаемые модули аутентификации)

SQL — Structured Query Language (язык структурированных запросов)

SSL — Secure Sockets Layer (протокол защищенных сокетов)

TCP — Transmission Control Protocol (протокол управления передачей данных)

UID — User Identifier (идентификатор пользователя)

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	Номер документа	Входящий номер сопроводительного документа и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					