



# Обновление сервера



- Основные версии и дополнительные выпуски
- Обновление с использованием резервной копии
- Обновление с помощью pg\_upgrade
- Обновление без прерывания обслуживания

## Основные версии (9.5)

изменяется функционал

новая версия не совместима с предыдущей на двоичном уровне

для обновления требуются специальные действия

необходимо изучить Release Notes, раздел «Migration»

## Дополнительные выпуски (9.5.2)

только исправление ошибок

гарантируется двоичная совместимость

достаточно установить новые исполняемые файлы

1. Остановить сервер
2. Установить новую версию
3. Стартовать сервер

## Плюсы

простота

## Минусы

вынужденное прерывание обслуживания

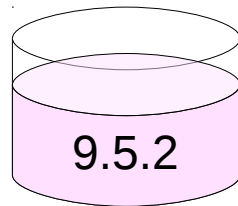


## Физическая репликация

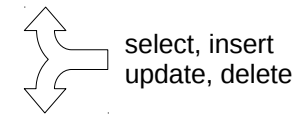
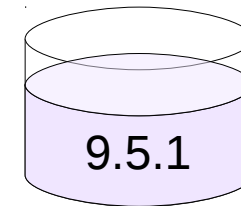
так как имеется двоичная совместимость между версиями

# Доп. выпуск: репликация

на обслуживании  
(обновление)



основной сервер



сегменты WAL

# Доп. выпуск: репликация



# Доп. выпуск: репликация





## ПЛЮСЫ

возможность обновления без прерывания обслуживания

1. Установить новую версию сервера  
при необходимости перенести старый кластер в другой каталог
2. Инициализировать новый кластер
3. Установить расширения  
только файлы; не выполнять `create extension`
4. Изменить конфигурационные файлы  
`postgresql.conf`, `pg_hba.conf`, `pg_ident.conf`  
либо скопировать файлы из старого кластера,  
либо внести изменения в новые файлы
5. Отключить пользователей  
чтобы не потерять изменения

## 6. Логическая резервная копия старого кластера

`pg_dumpall` или `pg_dumpall --globals-only`  
`pg_dump --jobs=N` для каждой БД

имеет смысл использовать утилиты от новой версии

## 7. Остановить старый сервер и стартовать новый

## 8. Восстановить кластер из резервной копии

`psql` или `pg_restore --jobs=N`

## 9. Удалить старые данные

исполняемые файлы

содержимое PGDATA и табличных пространств

## Плюсы

при обновлении можно сменить платформу, разрядность версии, кодировку баз данных и т. п.

## Минусы

большое время обновления  
требуется много дискового пространства

## Условия применимости

в новой версии изменился только формат служебной информации

в остальном сохраняется двоичная совместимость,  
включая представление типов и файлы данных

в базах данных не используются REG-типы

обновление с версии 8.4 или более поздней

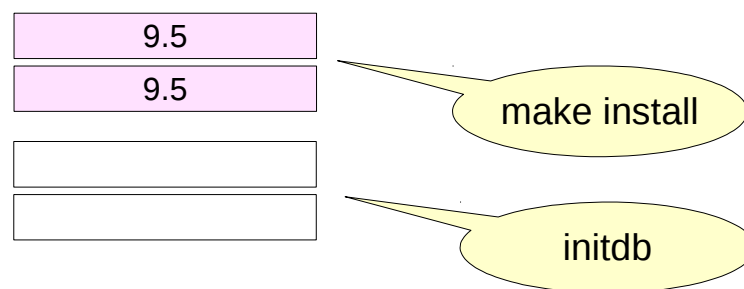
обновление до текущей версии (9.5)

1. Установить новую версию сервера  
при необходимости перенести старый кластер в другой каталог
2. Инициализировать новый кластер  
но не запускать
3. Установить расширения  
только файлы; не выполнять `create extension`
4. Изменить настройки аутентификации  
к обоим кластерам должен быть доступ для суперпользователя

# pg\_upgrade



файловая система  
старого кластера



файловая система  
нового кластера

## 5. Выполнить `pg_upgrade --check`

проверяет совместимость старого и нового кластеров  
(временно запускает новый кластер на порту 50432 или `$PGPORTNEW`)

надо использовать утилиту от новой версии

## 6. Остановить старый сервер

## 7. Выполнить `pg_upgrade`

запускает старый кластер на порту 50432 или `$PGPORTOLD`

делает резервную копию общих объектов старого кластера  
(`pg_dumpall --globals-only --binary-upgrade`)

и резервную копию схемы пользовательских данных для каждой БД  
(`pg_dump --schema-only --binary-upgrade`)

останавливает старый кластер

временно запускает новый кластер на порту 50432 или `$PGPORTNEW`  
и еще раз проверяет совместимость

копирует CLOG и выставляет значение счетчика транзакций  
(для этого сначала замораживает все транзакции нового кластера)

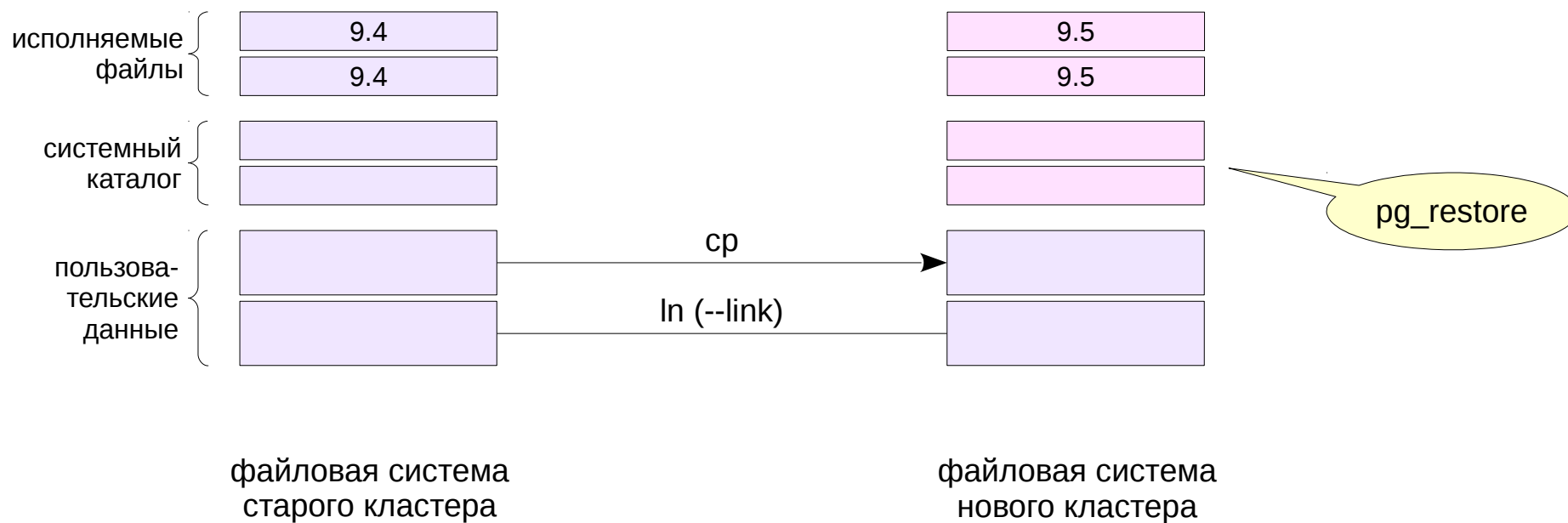
временно запускает новый кластер и восстанавливает  
общие объекты и схему данных из резервной копии  
(`pg_restore`; ссылки на файлы сохраняются)

копирует файлы данных (или ставит ссылки на старые, ключ `--link`)

восстанавливает счетчик OID  
(`pg_resetxlog -o`)

генерирует дополнительные скрипты

# pg\_upgrade



8. Изменить конфигурационные файлы  
`postgresql.conf`, `pg_hba.conf`, `pg_ident.conf`  
либо скопировать файлы из старого кластера,  
либо внести изменения в новые файлы
9. Стартовать новый сервер
10. Выполнить постобработку  
сбор статистики (`analyze_new_cluster.sh`)  
удаление старых данных (`delete_old_cluster.sh`)

## Плюсы

скорость обновления

возможность обойтись без дополнительного места на диске

## Минусы

ограниченное применение

Реплики надо обновлять одновременно с мастером

так как нет двоичной совместимости между версиями

I. Неверно: обновление реплики с помощью pg\_upgrade

нет гарантии, что мастер и реплика останутся идентичными

II. Создание новой реплики из базовой резервной копии

надежно, но долго

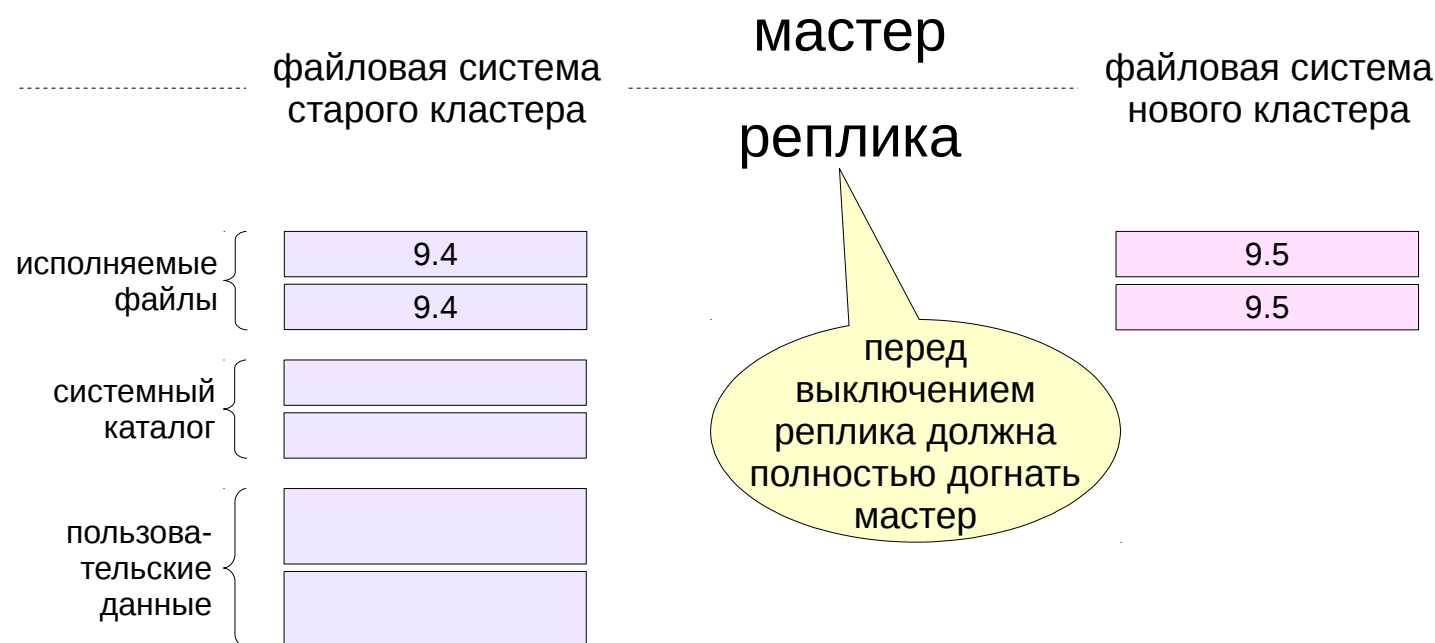
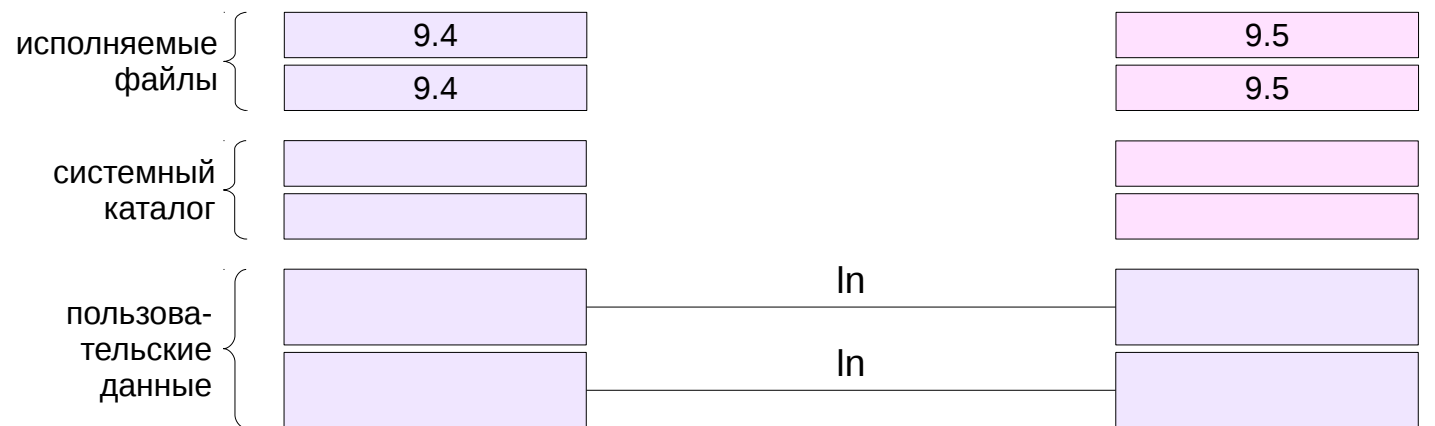
III. Обновление реплики с помощью rsync

воспользоваться тем, что файлы данных не изменяются

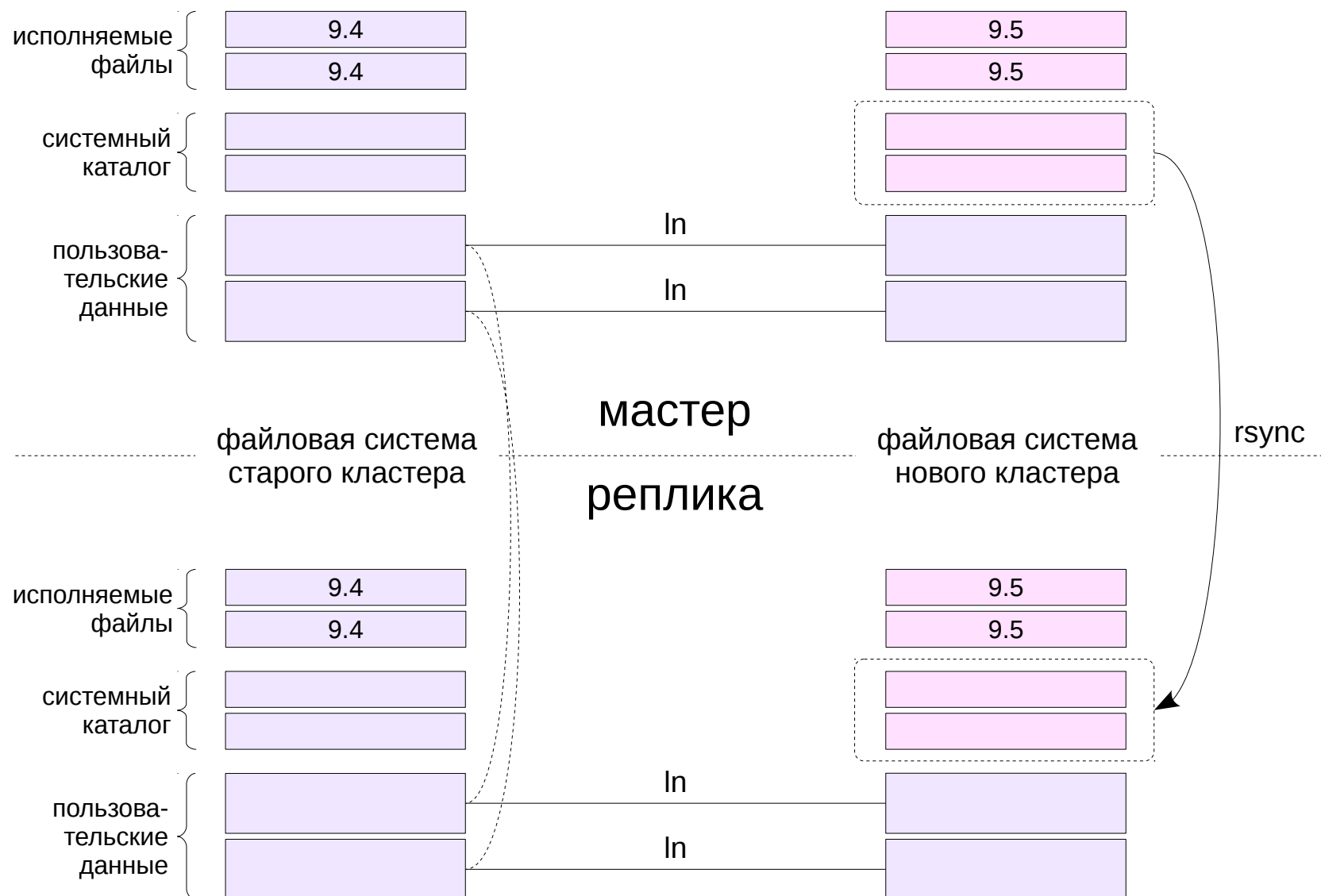
применимо только для Unix, дает эффект только при pg\_upgrade --link

при работе rsync мастер еще должен быть выключен

# pg\_upgrade и реплики



# pg\_upgrade и реплики



Мастер и логические реплики можно обновлять по очереди

так как есть совместимость на уровне команд SQL

процесс такой же, как при обновлении на дополнительный выпуск с использованием физической репликации

## Плюсы

возможность обновления без прерывания обслуживания

## Минусы

штатная логическая репликация пока отсутствует

# Демонстрация



Обновление на дополнительный выпуск —  
простая замена исполняемых файлов

физическая репликация, чтобы не прерывать обслуживание

Обновление на новую основную версию:

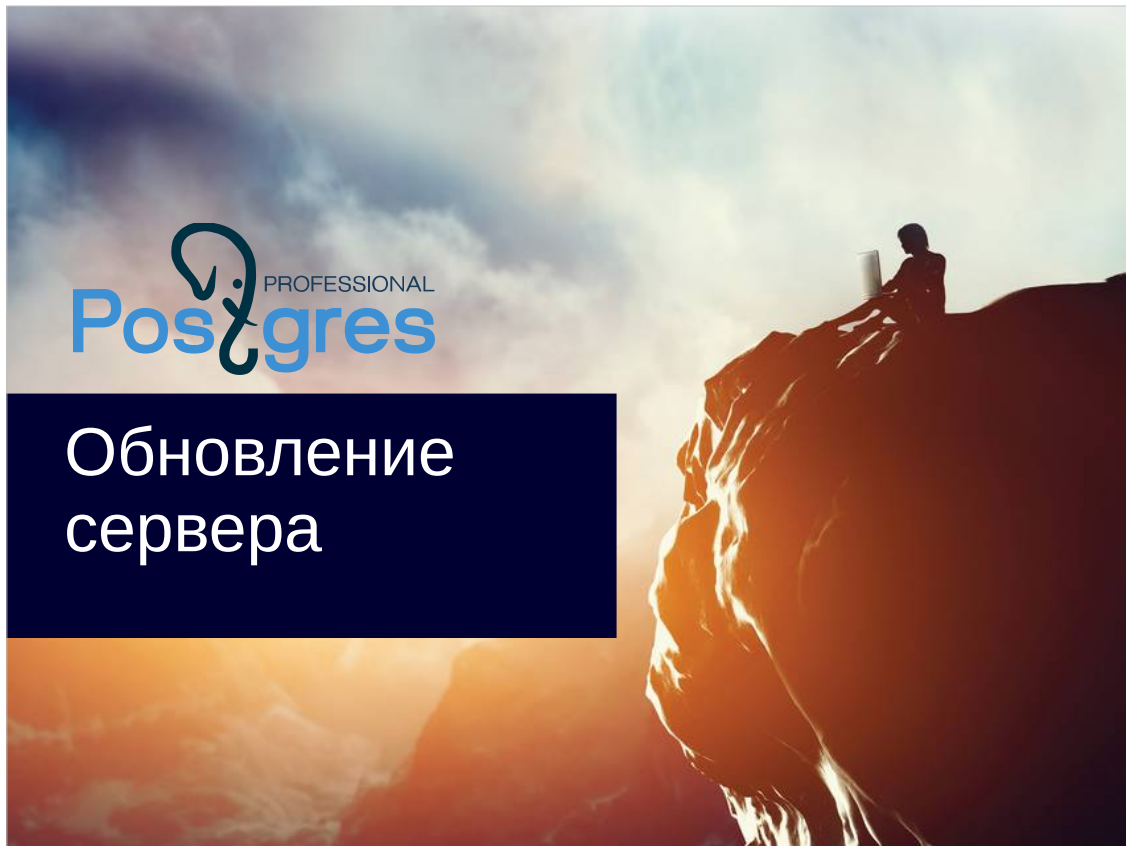
резервная копия (`pg_dumpall`)

программа обновления (`pg_upgrade`)

физическая репликация усложняет процедуру

логическая репликация, чтобы не прерывать обслуживание

1. Под пользователем ОС postgres4 установлен PostgreSQL версии 9.4. Создайте базу данных DB23 и пользователя usr с аутентификацией по паролю.
2. Установите расширение hstore. Под пользователем usr создайте таблицу, включающую столбец типа hstore. Наполните таблицу данными.
3. Выполните обновление сервера на версию 9.5 с помощью логической резервной копии. Новая версия собрана в каталоге /home/student/postgres-9.5.0, установка будет производиться в каталог /usr/local/pgsql4.  
При обновлении сделайте так, чтобы данные были размещены в отдельном табличном пространстве.
4. Проверьте корректность обновления: доступность сервера, аутентификацию пользователя usr, содержимое таблицы.



### **Авторские права**

Курс «Администрирование PostgreSQL 9.5. Расширенный курс»

© Postgres Professional, 2016 год.

Авторы: Егор Рогов, Павел Лузанов

### **Использование материалов курса**

Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

### **Обратная связь**

Отзывы, замечания и предложения направляйте по адресу:

[edu@postgrespro.ru](mailto:edu@postgrespro.ru)

### **Отказ от ответственности**

Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или косвенным, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Основные версии и дополнительные выпуски

Обновление с использованием резервной копии

Обновление с помощью pg\_upgrade

Обновление без прерывания обслуживания

## Основные версии (9.5)

изменяется функционал

новая версия не совместима с предыдущей на двоичном уровне

для обновления требуются специальные действия

необходимо изучить Release Notes, раздел «Migration»

## Дополнительные выпуски (9.5.2)

только исправление ошибок

гарантируется двоичная совместимость

достаточно установить новые исполняемые файлы

Номер версии PostgreSQL состоит из двух частей: номер основной версии (major release) — первые два числа, например, 9.5, — и номер дополнительного выпуска (minor release) — последнее число, например, 2 в 9.5.2.

Дополнительные выпуски служат только и исключительно для исправления ошибок, найденных в основной версии. Для них гарантируется сохранение двоичной совместимости (на одной платформе). В частности, две разные версии, отличающиеся только последней цифрой, будут нормально работать как мастер и реплика.

Поэтому обновление на следующий дополнительный выпуск делается максимально просто и рекомендуется, как только дополнительный выпуск появляется.

Новая основная версия приносит изменение функционала: какие-то возможности добавляются, изменяются, реже — удаляются. В этом случае двоичная совместимость отсутствует. Если попробовать подключить новую версию исполняемых файлов к старой версии кластера БД, PostgreSQL откажется с ним работать.

Для обновления основной версии требуется предпринимать специальные шаги. Также надо обратить пристальное внимание на раздел документации, содержащий замечания к выпуску (Release Notes), в частности на раздел Migration, в котором описаны ключевые несовместимости. Не исключено, что помимо обновления сервера БД, потребуется внесение изменений и в приложение.

<http://www.postgresql.org/docs/9.5/static/upgrading.html>

1. Остановить сервер
2. Установить новую версию
3. Стартовать сервер

### Плюсы

простота

### Минусы

вынужденное прерывание обслуживания

Для того, чтобы обновить сервер до очередного дополнительного выпуска, требуется остановить сервер и установить новые исполняемые файлы. После этого сервер можно запускать в обычном режиме.

Единственный минус этой процедуры состоит в том, что требуется кратковременное прерывание обслуживания пользователей.

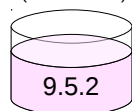


## Физическая репликация

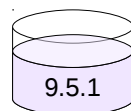
так как имеется двоичная совместимость между версиями

Если же в системе используется репликация и предусмотрен механизм переключения пользователей между серверами, процедуру обновления можно провести и без прерывания обслуживания. Речь здесь идет об обычной физической репликации, будь то потоковая или с трансляцией файлов журналов.

на обслуживании  
(обновление)



основной сервер



select, insert  
update, delete



сегменты WAL

Для этого отключают мастер и повышают реплику. Пока бывшая реплика обслуживает соединения, другой сервер обновляют до новой версии.



Затем бывший мастер вводят в строй в качестве новой реплики. Как это делать, было подробно рассмотрено в теме «Переключение на реплику».

Поскольку двоичная совместимость сохраняется, два сервера с разными версиями PostgreSQL нормально работают друг с другом.



Дальше та же процедура применяется к другому серверу.



## Плюсы

возможность обновления без прерывания обслуживания

В итоге получаем два обновленных сервера, и при этом обслуживание не прерывается.

1. Установить новую версию сервера  
при необходимости перенести старый кластер в другой каталог
2. Инициализировать новый кластер
3. Установить расширения  
только файлы; не выполнять `create extension`
4. Изменить конфигурационные файлы  
`postgresql.conf`, `pg_hba.conf`, `pg_ident.conf`  
либо скопировать файлы из старого кластера,  
либо внести изменения в новые файлы
5. Отключить пользователей  
чтобы не потерять изменения

Теперь рассмотрим обновление основной версии.

Самый простой способ обновления состоит в том, чтобы сделать *логическую резервную копию*, которую использовать для того, чтобы восстановить кластер БД на новой версии сервера. Напомним, что логическая резервная копия — это команды SQL, необходимые для развертывания базы данных (подробнее см. курс DBA1).

Сначала удобно выполнить подготовительные действия: установить новый сервер и инициализировать кластер. Если серверы устанавливаются в каталоги, содержащие номера версий (например, `/var/lib/pgsql/9.4/` и `/var/lib/pgsql/9.5/`), то они не пересекутся.

В противном случае может потребоваться переместить старый сервер в другой каталог.

Помимо установки исполняемых файлов СУБД, надо установить и все расширения, которые присутствовали в старом кластере. Иначе команда `create extension` из резервной копии не отработает.

Потребуется изменить конфигурационные файлы нового кластера. Можно попробовать скопировать эти файлы из старого кластера, но, поскольку от версии к версии в них бывают изменения, может оказаться более правильным внести необходимые изменения в уже новые файлы.

Резервную копию надо делать при работающем сервере, но требуется отключить всех пользователей (кроме только читающих), поскольку изменения, сделанные после запуска резервного копирования, не попадут в копию.



## Плюсы

при обновлении можно сменить платформу, разрядность версии, кодировку баз данных и т. п.

## Минусы

большое время обновления  
требуется много дискового пространства

К плюсам обновления с помощью `pg_dumpall` можно отнести определенную свободу маневра: при обновлении можно сменить платформу, разрядность версии, кодировку базы данных и т. п., так как двоичная совместимость между версиями не требуется.

Из минусов стоит отметить большое время обновления (тем большее, чем больше объем данных) и большое потребление дискового пространства: на сервере надо держать старую версию кластера, новую версию кластера и саму резервную копию.

Заметим, что на резервной копии можно сэкономить, запустив старый и новый сервера одновременно и направив вывод команды `pg_dumpall` на вход `psql` нового сервера.

## Условия применимости

- в новой версии изменился только формат служебной информации
- в остальном сохраняется двоичная совместимость, включая представление типов и файлы данных
- в базах данных не используются REG-типы
- обновление с версии 8.4 или более поздней
- обновление до текущей версии (9.5)

Второй способ обновления — утилита [pg\\_upgrade](#), которая умеет изменять файлы кластера так, чтобы привести их к виду, совместимому с новой версией. Никакой резервной копии в этом случае не требуется.

Возможность применения утилиты обусловлена тем, что при обновлении основной версии как правило меняются служебные таблицы, относящихся к системному каталогу, но формат файлов данных и индексов остаются без изменений. Поэтому достаточно поправить только небольшую часть данных, оставив основной массив как есть, без изменений.

При этом, конечно, новый кластер должен быть совместим со старым по разрядности, по кодировке и локалям.

В некоторых версиях изменяется двоичное представление типов данных. Например, тип JSONB изменился по сравнению с бета-версией 9.4. В таких случаях утилита не сможет помочь, но перечислит, где используется проблемный тип, чтобы помочь избавиться от него вручную перед обновлением.

Есть и дополнительные ограничения. Например, утилита не сможет обновить кластер, если в одной из баз данных используются REG-типы (кроме regtype, regclass и regenum).

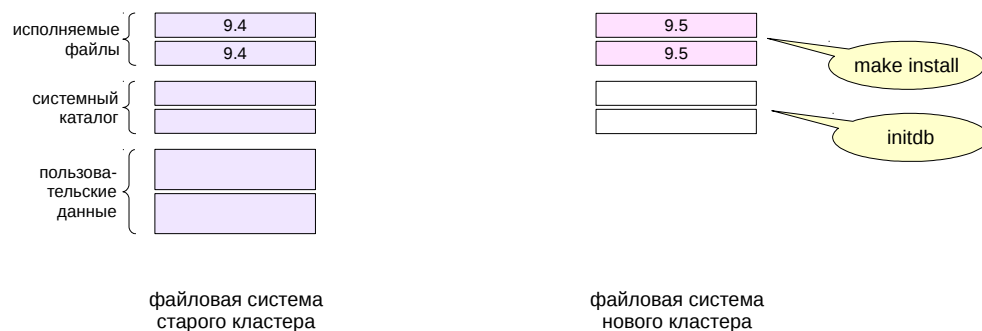
Утилитой поддерживается обновление любой версии, начиная с 8.4, на текущую версию. Таким образом, pg\_upgrade от 9.5 может обновить старый сервер только до 9.5, но не до 9.4.

1. Установить новую версию сервера  
при необходимости перенести старый кластер в другой каталог
2. Инициализировать новый кластер  
но не запускать
3. Установить расширения  
только файлы; не выполнять create extension
4. Изменить настройки аутентификации  
к обоим кластерам должен быть доступ для суперпользователя

Чтобы минимизировать время прерывания обслуживания, можно придерживаться следующего порядка действий.

Сначала установить новую версию сервера, включая расширения, и инициализировать кластер. Запускать новый кластер не надо.

Убедиться, что настройки аутентификации позволят подключиться к обоим кластерам суперпользователем.



После этих шагов картина должна соответствовать изображенной на иллюстрации:

- Старый сервер (например, версии 9.4) продолжает работать.
- Новый сервер (версии 9.5) выключен; кластер инициализирован, но не содержит никаких пользовательских объектов, только стандартные базы данных и системный каталог.

## 5. Выполнить `pg_upgrade --check`

проверяет совместимость старого и нового кластеров  
(временно запускает новый кластер на порту 50432 или `$PGPORTNEW`)  
надо использовать утилиту от новой версии

## 6. Остановить старый сервер

## 7. Выполнить `pg_upgrade`

запускает старый кластер на порту 50432 или `$PGPORTOLD`  
делает резервную копию общих объектов старого кластера  
(`pg_dumpall --globals-only --binary-upgrade`)  
и резервную копию схемы пользовательских данных для каждой БД  
(`pg_dump --schema-only --binary-upgrade`)  
останавливает старый кластер

Далее имеет смысл выполнить проверку совместимости кластеров. Это можно сделать, не выключая старый сервер. Хотя проверка и будет проведена еще раз при обновлении, лучше заранее убедиться, что все готово. Например, может оказаться, что в новом кластере не установлено какое-нибудь расширение.

Если проверка прошла успешно, можно выключать старый сервер и запускать утилиту `pg_upgrade`.

Утилита еще раз проверяет совместимость и делает резервную копию глобальных объектов старого кластера и схемы пользовательских данных. Для этого она временно запускает старый сервер (на порту 50432, чтобы избежать случайных подключений).

временно запускает новый кластер на порту 50432 или \$PGPORTNEW  
и еще раз проверяет совместимость

копирует CLOG и выставляет значение счетчика транзакций  
(для этого сначала замораживает все транзакции нового кластера)

временно запускает новый кластер и восстанавливает  
общие объекты и схему данных из резервной копии  
(pg\_restore; ссылки на файлы сохраняются)

копирует файлы данных (или ставит ссылки на старые, ключ --link)

восстанавливает счетчик OID  
(pg\_resetxlog -o)

генерирует дополнительные скрипты

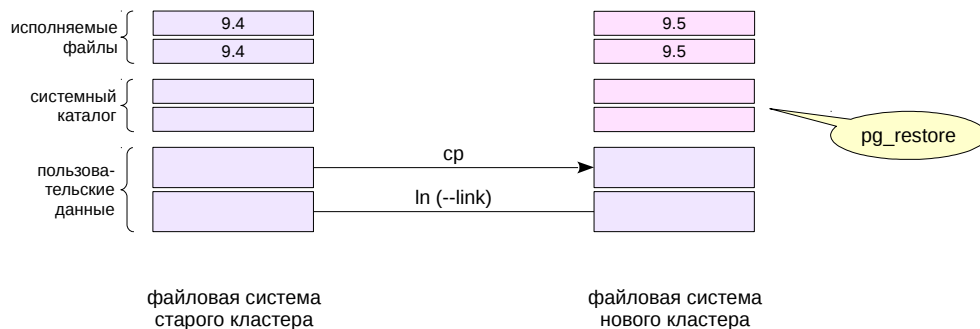
Далее утилита замораживает все транзакции нового кластера (чтобы не зависеть от текущего номера транзакции), выставляет счетчик транзакций со старого кластера и копирует статусы транзакций (CLOG).

Затем в новом кластере восстанавливаются из резервной копии глобальные объекты (такие, как роли и табличные пространства). Это не занимает много времени, поскольку таблицы только создаются, но не заполняются данными.

После этого копируются файлы данных (либо вместо копирования можно использовать жесткие ссылки; эта операция выполняется практически мгновенно и не требует дополнительного места на диске). Напомним, что при жесткой ссылке данные как таковые остаются в единственном экземпляре, но получают несколько независимых имен. Данные удаляются только в том случае, когда удалены все файлы, ссылающиеся на них.

Отметим, что при обычном использовании pg\_dump/pg\_restore такой «фокус» не пройдет, поскольку нарушится связь системного каталога с именами файлов. Утилита использует специальный режим --binary-upgrade, который официально не поддерживается.

Затем выставляется счетчик OID со старого кластера, и генерируются дополнительные скрипты.



После этих шагов получаем следующую картину:

- Старый кластер без изменений;
- Системный каталог нового кластера соответствует системному каталогу старого кластера, с точностью до изменения формата данных (именно поэтому для него и используется логическая резервная копия);
- Файлы данных нового кластера скопированы со старого, или на них проставлены жесткие ссылки.

Заметим, что пока старый кластер не изменился, можно в любой момент вернуться к нему, если при обновлении что-то пошло не так. Но при использовании жестких ссылок файлы данных изменятся, как только будет запущен *новый* кластер — так как и старый, и новый кластеры работают фактически с одними и теми же файлами.

## 8. Изменить конфигурационные файлы

postgresql.conf, pg\_hba.conf, pg\_ident.conf

либо скопировать файлы из старого кластера,  
либо внести изменения в новые файлы

## 9. Стартовать новый сервер

## 10. Выполнить постобработку

сбор статистики (analyze\_new\_cluster.sh)

удаление старых данных (delete\_old\_cluster.sh)

Заключительные действия: надо изменить нужным образом конфигурационные файлы нового кластера, стартовать его и выполнить скрипты, сгенерированные pg\_upgrade:

- сбор статистики (так как статистика не переносится при обновлении),
- удаление старых данных (конечно, после проверки корректности обновления).

## Плюсы

- скорость обновления
- возможность обойтись без дополнительного места на диске

## Минусы

- ограниченное применение

Огромный плюс `pg_upgrade` состоит в скорости обновления. Особенно это справедливо при использовании жестких ссылок вместо копирования. В этом случае обновление совершается без расходования дополнительного места. И то, и другое может быть критичным для больших баз данных.

Из минусов отметим то, что утилита применима не всегда.

Реплики надо обновлять одновременно с мастером

так как нет двоичной совместимости между версиями

I. Неверно: обновление реплики с помощью pg\_upgrade

нет гарантии, что мастер и реплика останутся идентичными

II. Создание новой реплики из базовой резервной копии

надежно, но долго

III. Обновление реплики с помощью rsync

воспользоваться тем, что файлы данных не изменяются

применимо только для Unix, дает эффект только при pg\_upgrade --link

при работе rsync мастер еще должен быть выключен

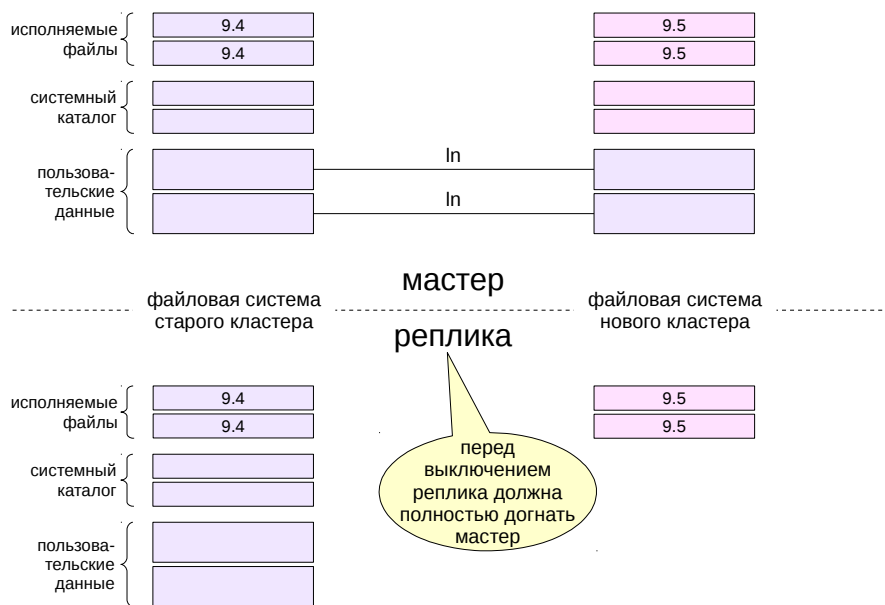
Отдельные трудности возникают при использовании физической репликации. Дело в том, что и мастер, и реплики необходимо обновлять одновременно. Соединение мастера одной версии с репликой другой версии скорее всего приведет к потере данных.

Еще одна тонкость: реплику нельзя обновлять с помощью pg\_upgrade. Такое обновление не дает 100% гарантии того, что два идентичных экземпляра после обновления также получатся идентичными.

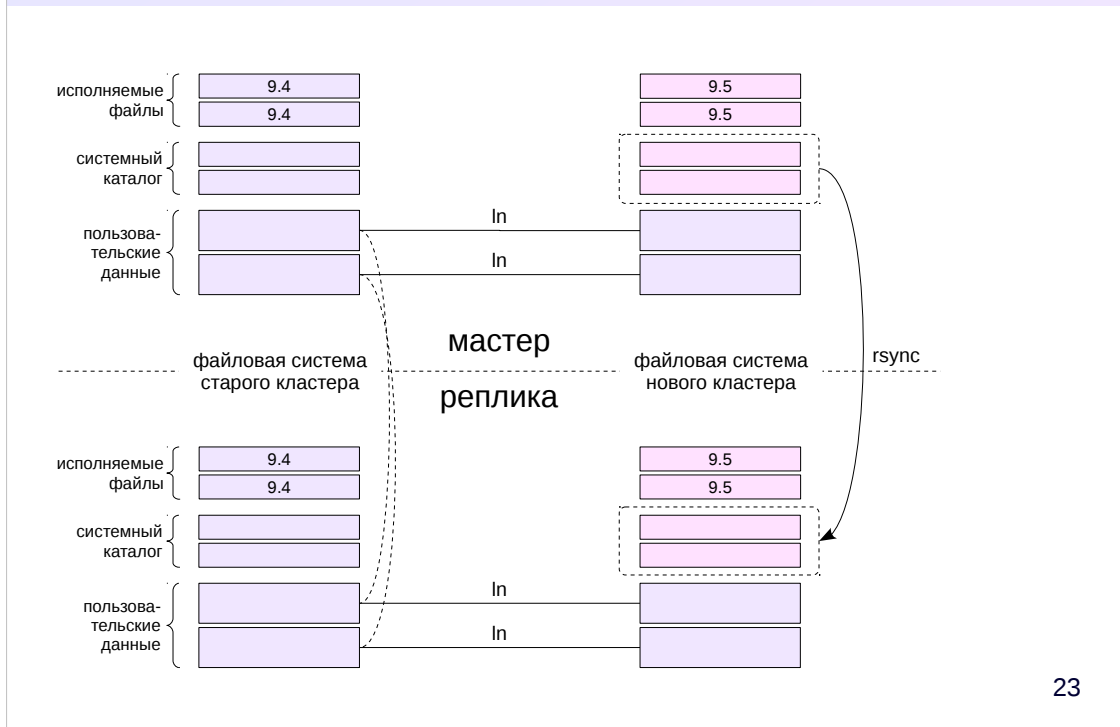
Обычный способ состоит в том, чтобы забыть про существующие реплики и создать их заново. Для этого надо выполнить базовую резервную копию мастера и развернуть из нее новую реплику (см. темы «Файловая репликация» и «Потоковая репликация»). Понятно, что это достаточно длительная процедура.

Если речь идет об операционной системе Unix, и при обновлении мастера использовался режим жестких ссылок, то время развертывания новой реплики можно существенно ускорить, получив примерно такой же выигрыш по времени, который дает pg\_upgrade для мастера. Для этого надо правильным образом воспользоваться утилитой rsync.

# pg\_upgrade и реплики



Исходная позиция: мастер обновлен, но новый сервер еще не запускался. Реплика выключена, причем перед выключением она полностью догнала мастер (таким образом файлы данных совпадают).  
На реплике устанавливается новая версия сервера, но кластер не инициализируется.



Дальше запускается `rsync` так, чтобы в одной команде скопировать с мастера *оба кластера*, старый и новый, на реплику.

В таком режиме `rsync` понимает, что:

- старый и новый кластер разделяют общие файлы данных;
- файлы данных старого кластера на мастере и на реплике совпадают (для этого сравниваются только размеры файлов, а не содержимое);
- следовательно, файлы данных не надо физически копировать на реплику, а достаточно проставить жесткие ссылки.

И `rsync` остается скопировать только файлы, относящиеся к системному словарию, а это небольшой объем.

Этот процесс описан в документации к [pg\\_upgrade](#), в том числе и конкретные команды, которые необходимо выполнить. К сожалению, документация не объясняет, почему необходимы именно такие команды, видимо, из-за того, что объяснение было бы слишком пространным.

Про `rsync` мы уже рассказали, а еще один тонкий момент связан с тем, что при установке счетчика `OID` используется утилита `pg_resetxlog`, которая выставляет в управляющем файле `pg_control` минимальный уровень журнала. Поэтому документация рекомендует запустить и тут же остановить новый мастер перед тем, как выполнять `rsync` (при этом `pg_control` исправляется).

Для того, чтобы разобраться в этом вопросе детально, полезно ознакомиться с [перепиской разработчиков](#).

## Мастер и логические реплики можно обновлять по очереди

так как есть совместимость на уровне команд SQL

процесс такой же, как при обновлении на дополнительный выпуск с использованием физической репликации

### Плюсы

возможность обновления без прерывания обслуживания

### Минусы

штатная логическая репликация пока отсутствует

Можно ли выполнить обновление основной версии без прерывания обслуживания?

Можно, если использовать не физическую, а логическую репликацию. В этом случае от мастера и реплики не требуется двоичной совместимости, поэтому можно применить процедуру, описанную в начале этой темы (применительно к обновлению на дополнительный выпуск).

К сожалению, штатная логическая репликация в PostgreSQL пока отсутствует (в том числе ее не будет и в версии 9.6), хотя в этом направлении ведутся активные работы. В настоящее время можно пользоваться только сторонними расширениями.



Обновление на дополнительный выпуск —  
простая замена исполняемых файлов

физическая репликация, чтобы не прерывать обслуживание

Обновление на новую основную версию:

резервная копия (`pg_dumpall`)

программа обновления (`pg_upgrade`)

физическая репликация усложняет процедуру

логическая репликация, чтобы не прерывать обслуживание

1. Под пользователем ОС postgres4 установлен PostgreSQL версии 9.4. Создайте базу данных DB23 и пользователя usr с аутентификацией по паролю.
2. Установите расширение hstore. Под пользователем usr создайте таблицу, включающую столбец типа hstore. Наполните таблицу данными.
3. Выполните обновление сервера на версию 9.5 с помощью логической резервной копии. Новая версия собрана в каталоге /home/student/postgres-9.5.0, установка будет производиться в каталог /usr/local/pgsql4.  
При обновлении сделайте так, чтобы данные были размещены в отдельном табличном пространстве.
4. Проверьте корректность обновления: доступность сервера, аутентификацию пользователя usr, содержимое таблицы.

1. Для аутентификации по паролю укажите пароль при создании пользователя:

```
create user usr password 'mypassword';
```

А в начало файла pg\_hba.conf добавьте строку:

```
local all usr md5
```

3. Не забудьте перенести старую версию сервера из /usr/local/pgsql4 в другой каталог.

При установке новой версии убедитесь, что установлено расширение hstore (либо `sudo make install-world`, либо отдельно `sudo make install`; `sudo make -C contrib/hstore install`).

Чтобы разместить данные в отдельном табличном пространстве, предварительно создайте его, и отредактируйте файл с резервной копией (добавьте табличное пространство по умолчанию для БД).

4. Чтобы сохранить аутентификацию, скопируйте файл pg\_hba.conf из старого кластера.