­­­Министерства науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

*Факультет Программной инженерии и компьютерной техники*

**Отчёт по лабораторной работе №4**

Студенты:

Бондарь Б. М.

Петропавлова Т. В.

Группа:

P33212

Преподаватель:

Шешуков Д. М.

Санкт-Петербург

2024 г.

Оглавление

[Текст задания 3](#_Toc164306771)

[Выполнение 5](#_Toc164306772)

[Этап 1. Настройка 5](#_Toc164306773)

[Этап 2 13](#_Toc164306774)

[Этап 2.1. Подготовка 13](#_Toc164306775)

[Этап 2.2. Сбой 14](#_Toc164306776)

[Этап 2.3. Отработка 14](#_Toc164306777)

[Этап 3. Восстановление 16](#_Toc164306778)

[Вывод 18](#_Toc164306779)

# Текст задания

Отказоустойчивость, балансировка нагрузки и репликация

Требования к выполнению лабораторной работы:

* Работа рассчитана на двух человек.
* В качестве хостов использовать одинаковые виртуальные машины.
* В первую очередь настроить сеть виртуальных машин:
  + Если ВМ запускаются на одном хосте, рекомендуется использовать NAT сеть.
  + Если ВМ запускаются на различных хостах, рекомендуется использовать сетевые интерфейсы в режиме “Bridge”; для связи рекомендуется использовать проводное соединение.
  + Проверить сетевую связность между всеми узлами (ping, ssh).
* Для подключения к СУБД (например, через psql), использовать отдельную виртуальную или физическую машину.
* Перед тем как “сломать” узел на этапе 2, рекомендуется выполнить снапшот виртуальной машины.
* Для демонстрации наполнения базы, а также доступа на запись (см. задание ниже) использовать **не меньше** двух таблиц, трёх столбцов, пяти строк, двух транзакций, двух клиентских сессий. Данные не обязаны быть осмысленными, но должны быть легко отличимы - повторяющиеся строки запрещены.

Этап 1 Настройка (варианты):

* Развернуть postgres на двух узлах в режиме **трансляции логов**. Не использовать дополнительные пакеты. Продемонстрировать доступ в режиме чтение/запись на основном сервере, а также что новые данные синхронизируются на резервный.
* *Развернуть postgres на двух узлах в режиме* ***потоковой репликации****. Не использовать дополнительные пакеты. Продемонстрировать доступ в режиме чтение/запись на основном сервере, а также что новые данные синхронизируются на резервный.*
* Развернуть postgres на двух узлах в режиме **горячего резерва** (Master + Hot Standby). Не использовать дополнительные пакеты. Продемонстрировать доступ в режиме чтение/запись на основном сервере, в режиме чтение на резервном сервере, а также актуальность данных на нём.
* Развернуть postgres на двух узлах в режиме **балансировки нагрузки**. Использовать {одно из двух: pgpool-II или pgbouncer}. Продемонстрировать обработку транзакций обоими серверами.
* Настроить **репликацию** postgres на трёх узлах **в каскадном режиме** A --> B --> C. Для управления использовать pgpool-II. Репликация с A на B синхронная. Репликация с B на C асинхронная. Продемонстрировать, что новые данные реплицируются на B в синхронном режиме, а на C с задержкой.
* Настроить **репликацию** postgres на трёх узлах: **A - основной, B и C - резервные**. Для управления использовать pgpool-II. Репликация с A на B синхронная. Репликация с A на C асинхронная. Продемонстрировать, что новые данные реплицируются на B в синхронном режиме, а на C с задержкой.

Этап 2.1 Подготовка (общая часть):

1. Установить несколько клиентских подключений к СУБД.
2. Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Этап 2.2 Сбой (варианты):

* *Симулировать отказ основного узла - выполнить жесткое выключение виртуальной машины.*
* Симулировать недоступность основного узла - отключить сетевой интерфейс виртуальной машины, переключить его в изолированную подсеть и т.п.
* Симулировать программную ошибку на основном сервере - выполнить pkill -9 postgres.
* Симулировать ошибку диска на основном узле - удалить директорию PGDATA со всем содержимым.
* Симулировать переполнение дискового пространства на основном сервере - заполнить всё свободное пространство раздела с PGDATA “мусорными” файлами.

Этап 2.3 Отработка (общая часть):

1. Найти продемонстрировать в логах релевантные сообщения об ошибках.
2. Выполнить фейловер на резервный сервер.
3. Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Этап 3 Восстановление (общая часть):

1. Восстановить работу основного узла - откатить действие, выполненное с виртуальной машиной на этапе 2.2.
2. Актуализировать состояние базы на основном узле - накатить все изменения данных, выполненные на этапе 2.3.
3. Восстановить работу узлов в исходной конфигурации (в соответствии с этапом 1).
4. Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

# Выполнение

## Этап 1. Настройка

**Выбранный вариант:** развернуть postgres на двух узлах в режиме **потоковой репликации**. Не использовать дополнительные пакеты. Продемонстрировать доступ в режиме чтение/запись на основном сервере, а также что новые данные синхронизируются на резервный.

В VirtualBox была создана виртуальная машина, а затем её клон:

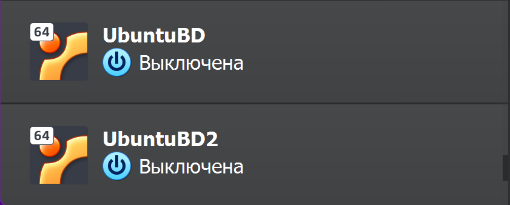


Рисунок 1. Виртуальные машины

При создании клона выставлены следующие настройки:

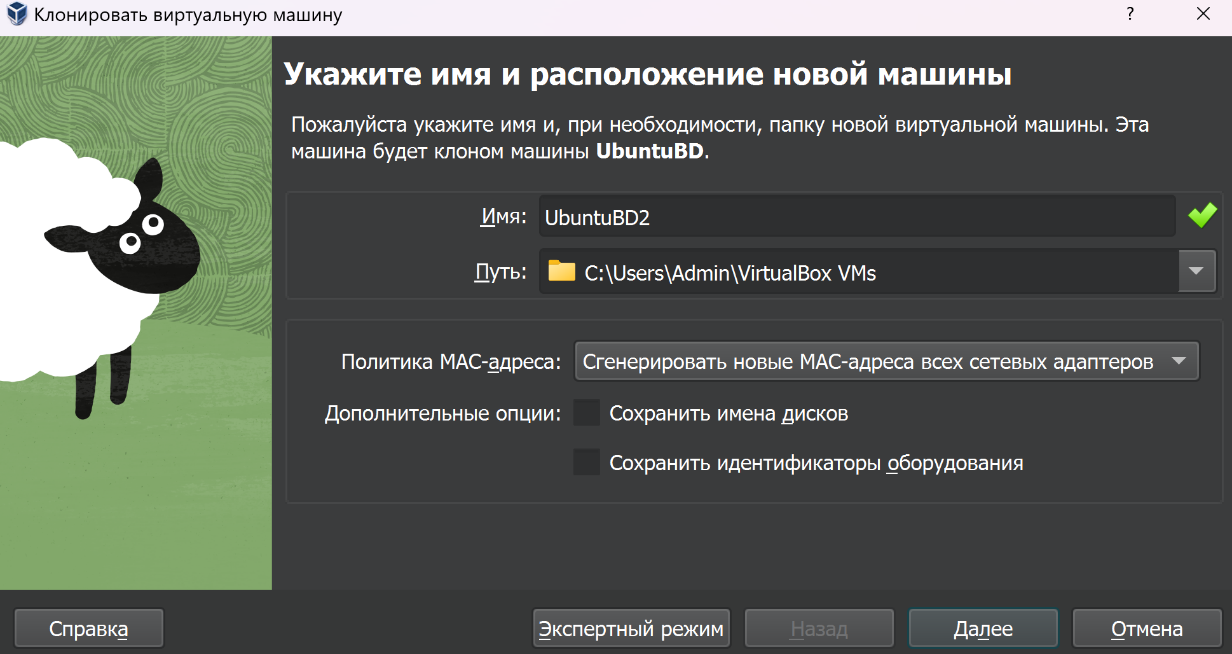


Рисунок 2. Создание клона

Так как машины располагаются на одном хосте, то связь между ними будет осуществляться через сеть NAT:

Создание сети:

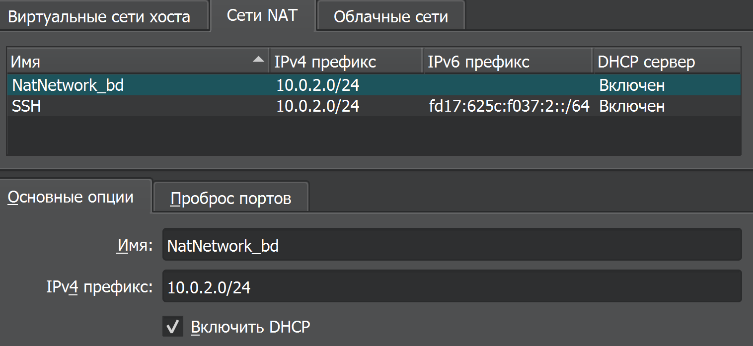


Рисунок 3. Новая сеть NAT

Настройка сети NAT на машинах:

|  |  |
| --- | --- |
| Рисунок 4. Основная машина | Рисунок 5. Реплика |

Для наглядности изменены имена хостов на машинах.

MAC-адреса разные, IP тоже будут разными:

|  |  |
| --- | --- |
| Рисунок 6. IP-адрес основного узла | Рисунок 7. IP-адрес реплики |

Для подключения по ssh с хоста добавляется проброс портов:

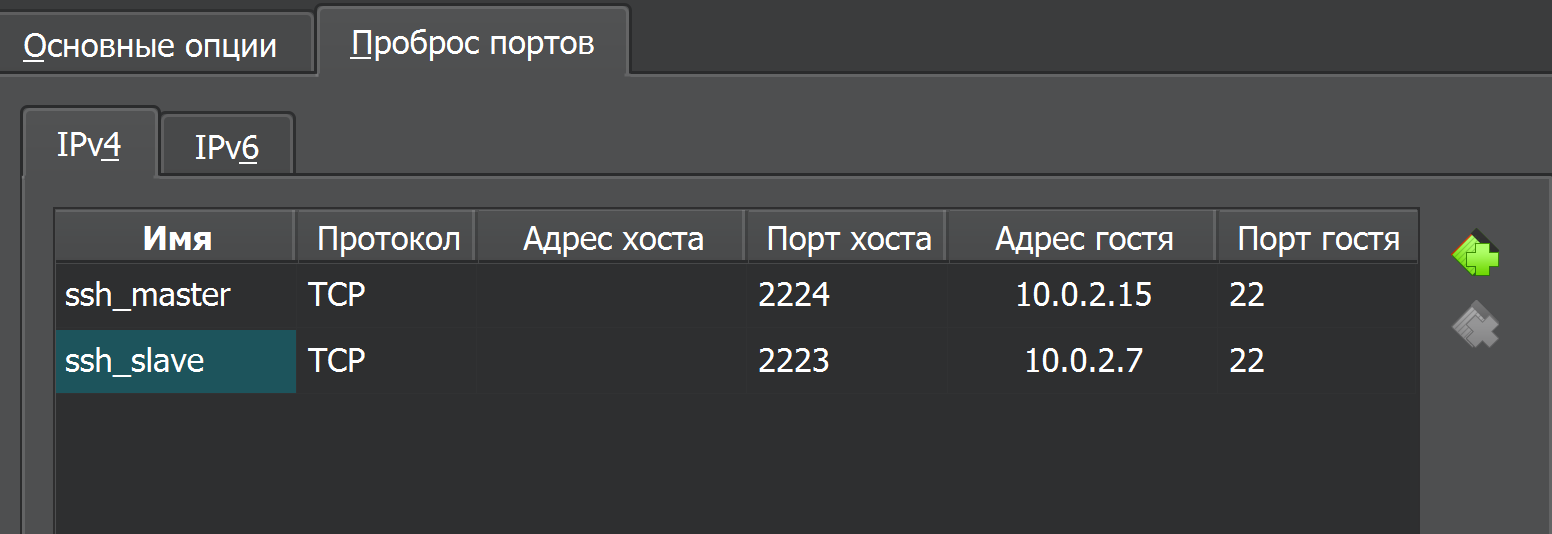


Рисунок 8. Проброс портов

Проверка доступности машин относительно друг друга:

|  |  |
| --- | --- |
| Рисунок 9. Доступность slave с master'а | Рисунок 10. Доступность master'а со slave |

**Установка ssh-сервера:**

На обеих машинах выполняется следующая команда:

sudo apt-get install openssh-server

Проверка подключения по ssh:

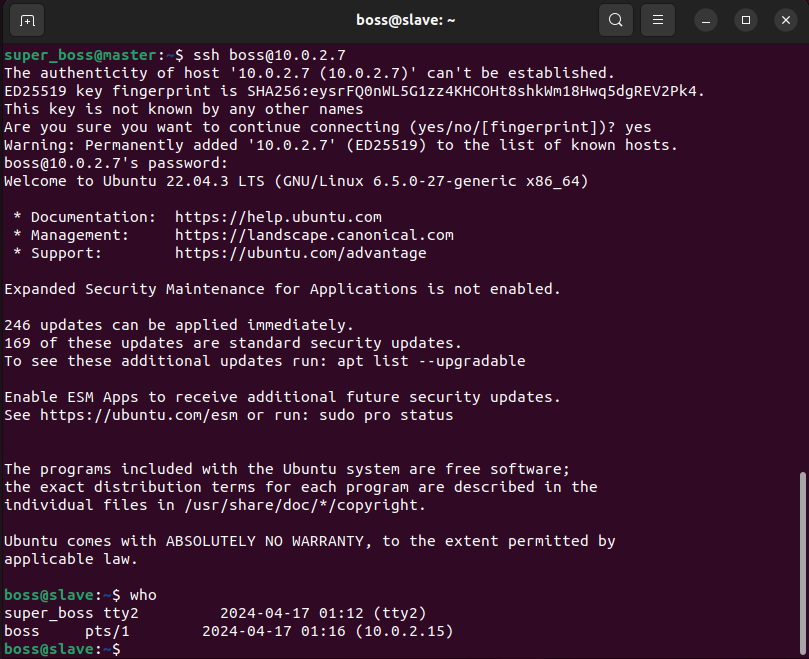


Рисунок 11. Подключение по ssh к slave с master'а

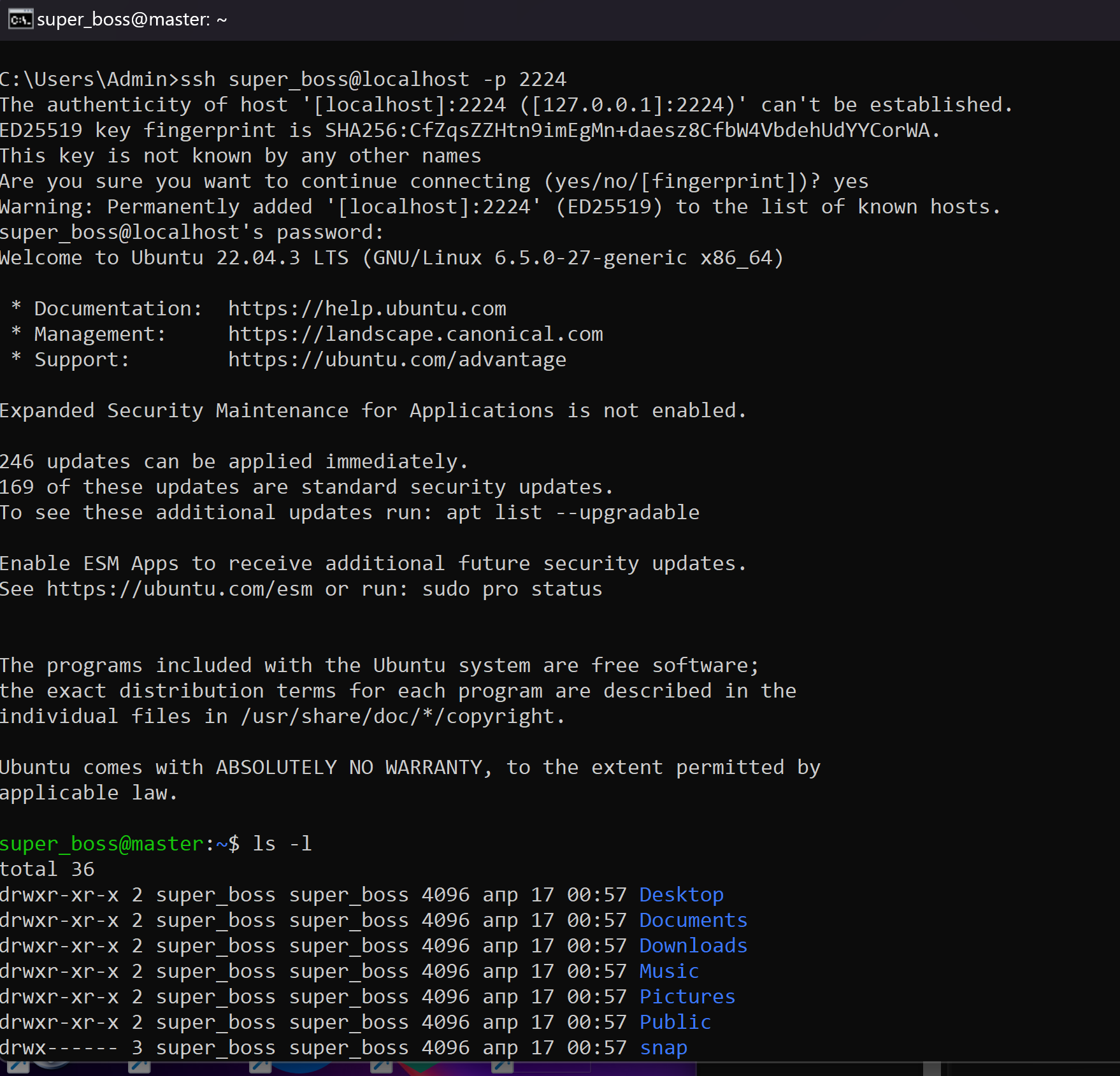


Рисунок 12. Подключение по ssh к master'у через командную строку

**Установка postgres:**

На обеих машинах выполняются следующие команды:

sudo apt-get install postgresql

sudo apt-get install postgresql-client

**Настройка потоковой репликации**

Основной узел (master)

Для возможности создания копий нужно установить связь основного узла с узлом-репликой.

Внесение изменений в файл */etc/postgresql/14/main/pg\_hba.conf:*

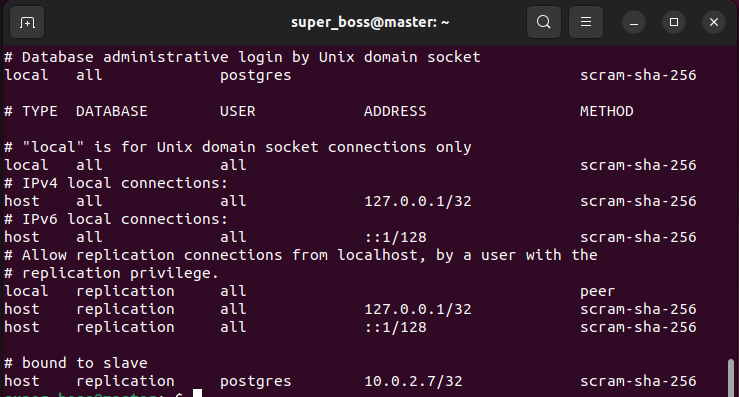


Рисунок 13. Master - pg\_hba.conf

Нужно заменить метод peer на метод входа по зашифрованному паролю для администратора (пользователь postgres) и для соединений через сокет, чтобы не создавать новых пользователей на виртуальной машине при создании пользователей в PostgreSQL. Также было добавлено соединение по TCP-сокету для репликации по IP-адресу 10.0.2.7 (IP slave), репликацией на узел с этим адресом будет заниматься только пользователь postgres (администратор по умолчанию).

Внесение изменений в файл */etc/postgresql/14/main/postgresql.conf*:

listen\_addresses = 'localhost, 10.0.2.15' – настройка IP-адресов для прослушивания подключений, нужно добавить IP-адрес самого master-узла.

wal\_level = replica, так как версия PostgreSQL следующая:



Рисунок 14. Версия PostgreSQL

max\_wal\_senders = 2, на 1 больше, чем реальное число узлов-реплик

Для того, чтобы не настраивать архивирование, нужно задать достаточно большой минимальный размер (в МБ) сегментов файла WAL в pg\_wal для извлечения их резервным сервером:

wal\_keep\_size = 1024

hot\_standby = on – включение возможности делать запросы в момент восстановления.

Резервный узел (slave)

В файле */etc/postgresql/14/main/pg\_hba.conf* выполняются действия, аналогичные действиям с основным узлом, но адрес для связи будет IP-адресом основного узла:

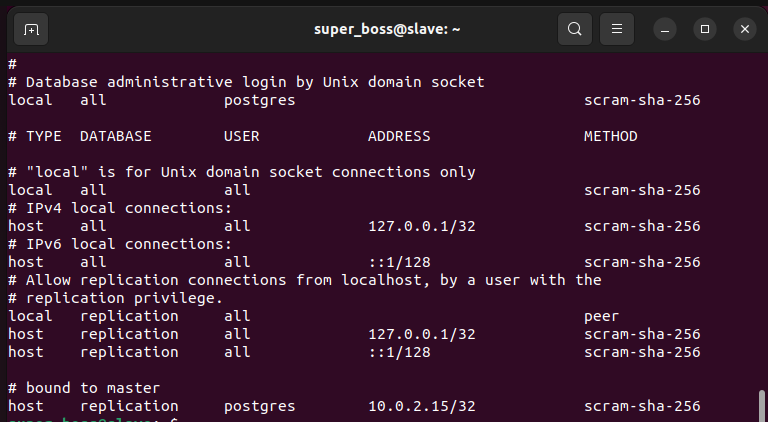


Рисунок 15. Slave - pg\_hba.conf

Затем в файле */etc/postgresql/14/main/postgresql.con*f нужно настроить потоковый прием с основного сервера на реплику:

listen\_addresses = 'localhost, 10.0.2.7'

wal\_level = replica

max\_wal\_senders = 2

wal\_keep\_size = 1024

hot\_standby = on

Настройки такие же, за исключением listen\_addresses, так как в случае сбоя основного узла резервный сможет выступить в качестве основного.

Запуск PostgresSQL на основном сервере:

systemctl start postgresql

На реплике подключение через пользователя postgres:

sudo -u postgres -i

Перенос копии:

Сначала удаляется директория main, которая хранит данные собственного Postgres-сервера узла, который должен стать репликой. Затем директория создается заново с правами rwx. Далее в директорию main переносятся данные с основного узла следующей командой:

pg\_basebackup -P -R -X stream -c fast -h 10.0.2.15 -U postgres -D ./main

Атрибут -R создает в директории кластера файл standby.signal, который указывает на то, что сервер является репликой. А также добавляет параметры подключения в файл postgresql.auto.conf.

Метод stream указывает, что промежуточные файлы не создаются, данные транслируются напрямую с основного сервера на резервный по сети.

Режим контрольных точек fast означает, что время блокировки данных при создании контрольной точки будет минимизированным, а не равномерно одинаковым как при spread

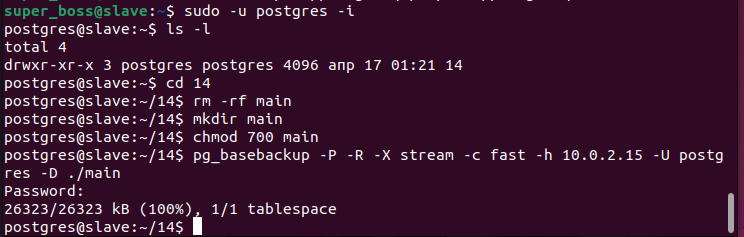


Рисунок 16. Создание копии на резервном узле

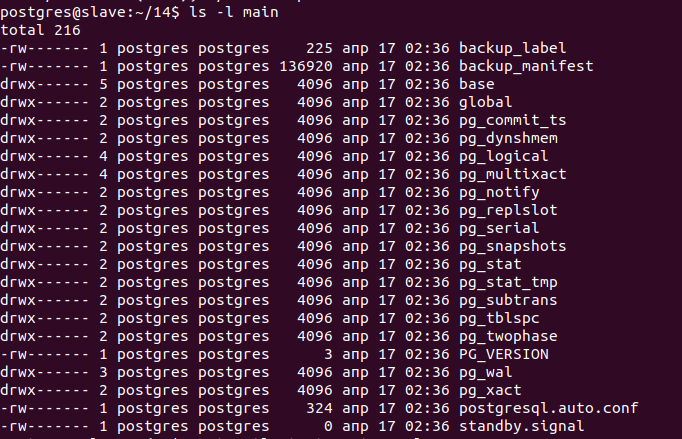


Рисунок 17. Файлы кластера на Slave

**Демонстрация результатов:**

|  |  |
| --- | --- |
| Рисунок 18. Состояние репликации на master | Рисунок 19. Состояние репликации на slave |

PostgreSQL запущен на обеих машинах. На основном сервере создаются таблицы разными пользователями:

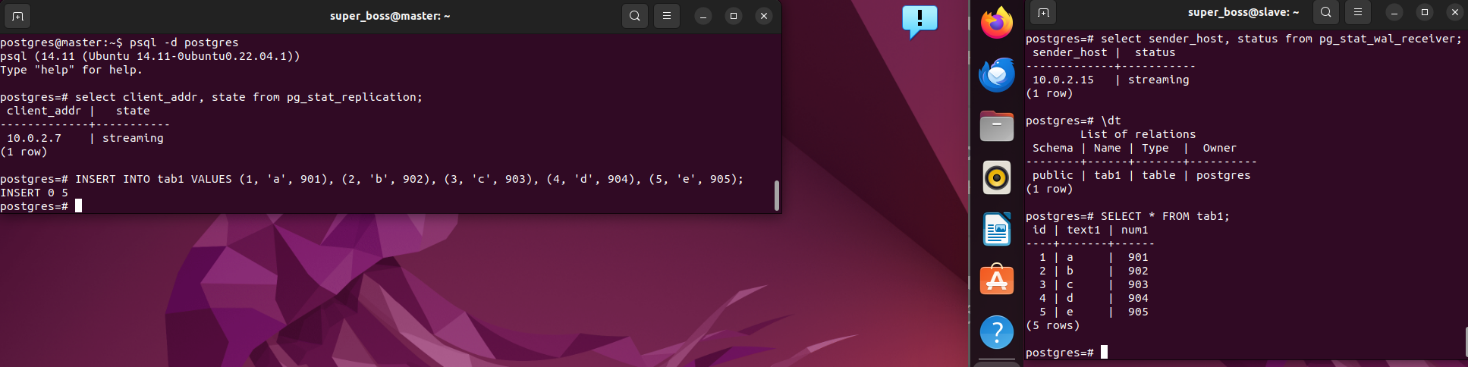


Рисунок 20. Создание таблицы пользователем postgres

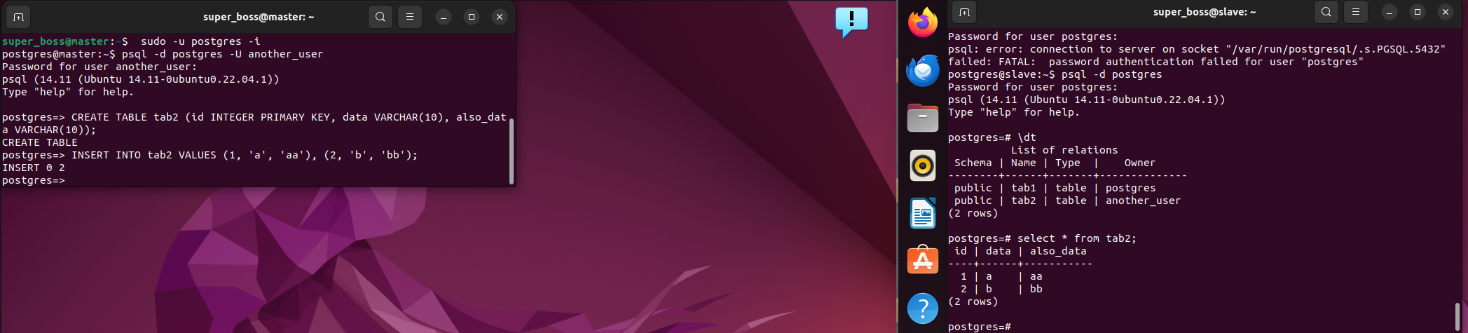


Рисунок 21. Создание таблицы пользователем another\_user

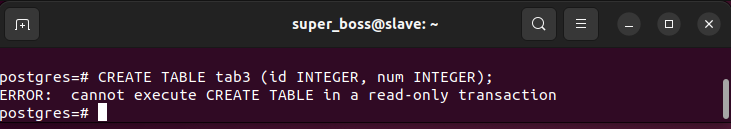


Рисунок 22. Создание таблицы на резервном узле

Основной узел работает в режиме чтения и в режиме записи, резервный узел работает только в режиме чтения, так как кластер на нем является репликой.

## Этап 2

### Этап 2.1. Подготовка

Установка нескольких клиентских подключений и демонстрация результатов:

|  |  |
| --- | --- |
| Рисунок 23. Подключение через postgres | Рисунок 24. Подключение через another\_user |
| Рисунок 25. Узел Slave | |

В данном случае происходят 2 одновременных подключения к базе данных postgres: подключаются пользователи postgres и another\_user. Они вносят изменения в таблицу tab2. На резервном узле к базе данных postgres подключается пользователь postgres. Видно, что после обновления таблицы на основном узле таблица меняется и на резервном.

### Этап 2.2. Сбой

**Выбранный вариант:** симулировать отказ основного узла - выполнить жесткое выключение виртуальной машины.

Для того, чтобы симулировать сбой, нужно выключить виртуальную машину, на которой находится основной узел:

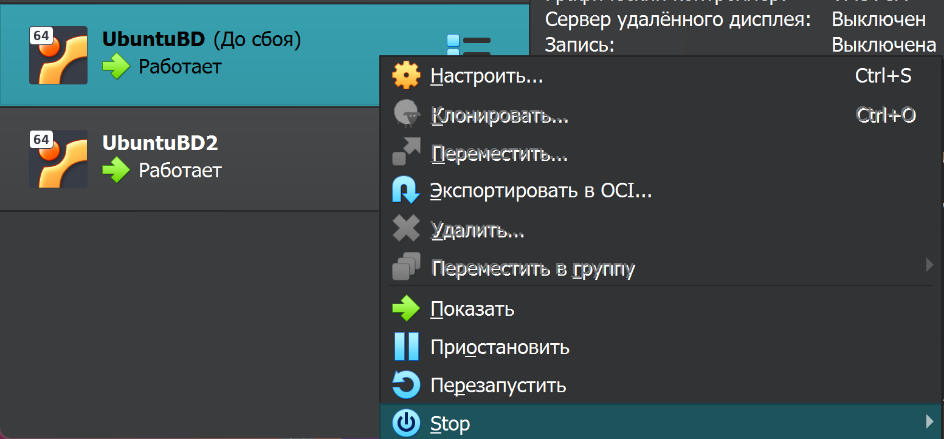


Рисунок 26. Симуляция сбоя

### Этап 2.3. Отработка

Во время «сбоя» виртуальная машина с резервным узлом пишет в лог, что не удается подключиться к primary server (Master):

(команда для просмотра логов: tail -n 20 /var/log/postgresql/postgresql-14-main.log)



Рисунок 27. Журнал логов при сбое

Выполнение failover на резервный узел:

sudo -u postgres -i

/usr/lib/postgresql/14/bin/pg\_ctl promote -D /etc/postgresql/14/main

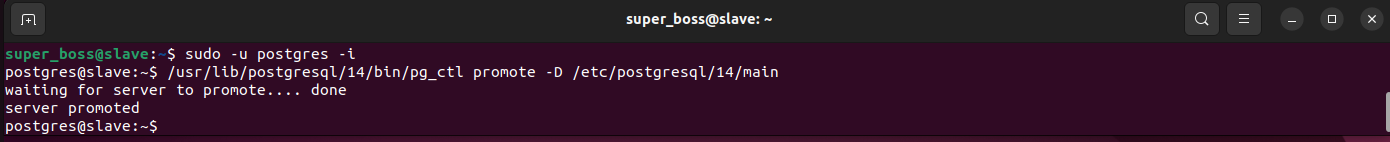


Рисунок 28. Failover

Команда promote выводит резервный сервер из режима резерва, то есть резервный узел может выполнять операции не только чтения, но и записи.

Для демонстрации возможностей режима чтения и записи на резервном узле была создана таблица tab3 и добавлена новая строка в таблицу tab2:

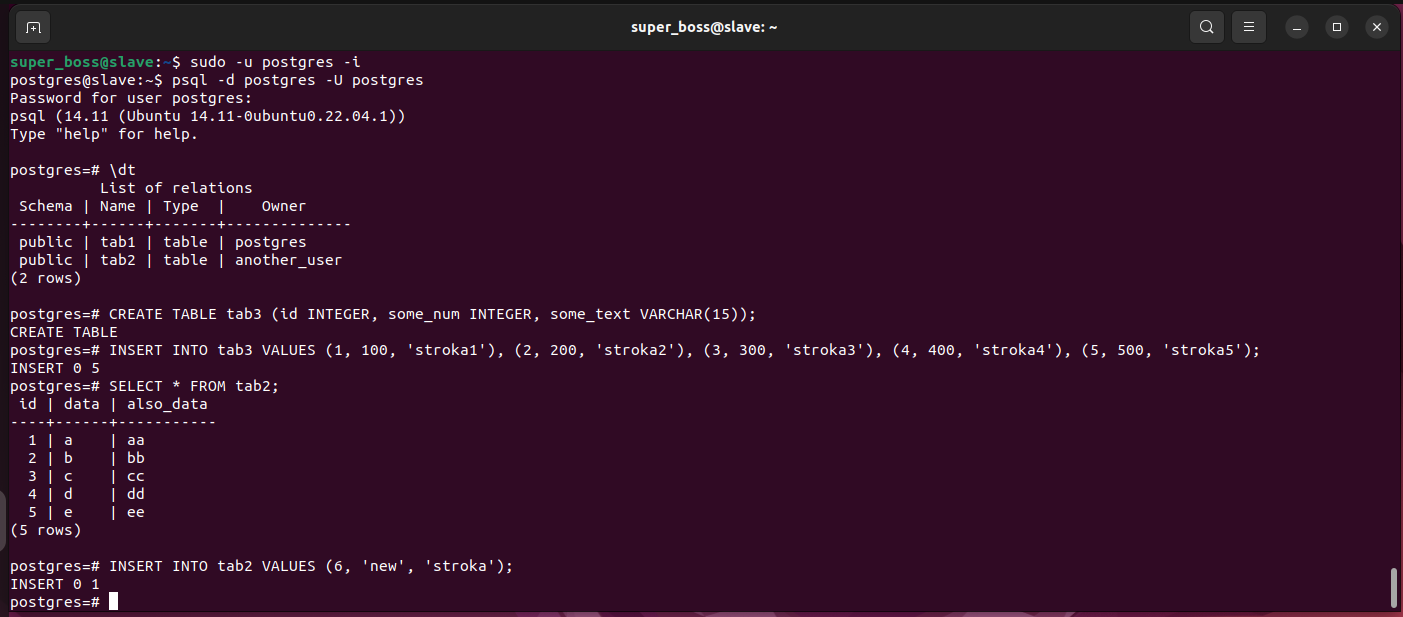


Рисунок 29. Чтение и запись на Slave

## Этап 3. Восстановление

Для начала восстановления конфигурации нужно включить виртуальную машину с основным узлом.

Файлы кластера (в данном случае – main) хранятся в директории */var/lib/postgresql/14/main*. Так как восстановить теперь нужно основной узел, требуется пересоздать эту директорию на основном узле и передать туда данные с резервного:

rsync -avv super\_boss@10.0.2.7:/var/lib/postgresql/14/\* /var/lib/postgresql/14/

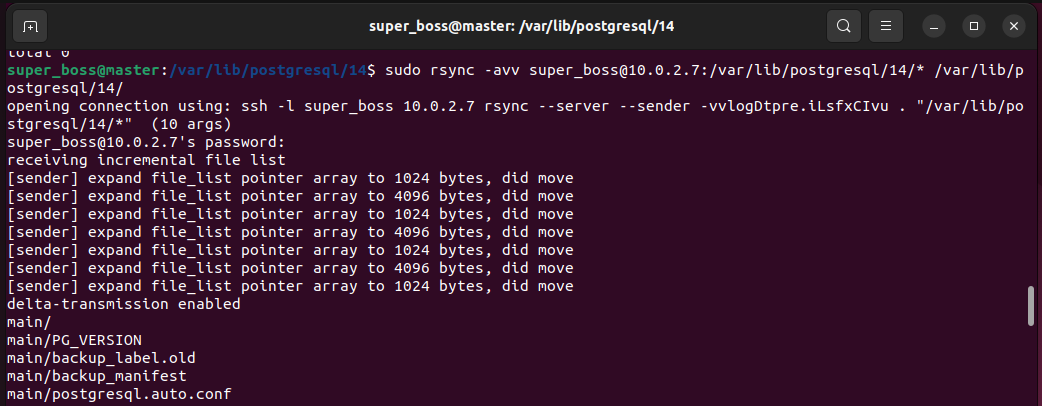


Рисунок 30. Синхронизация Slave с Master

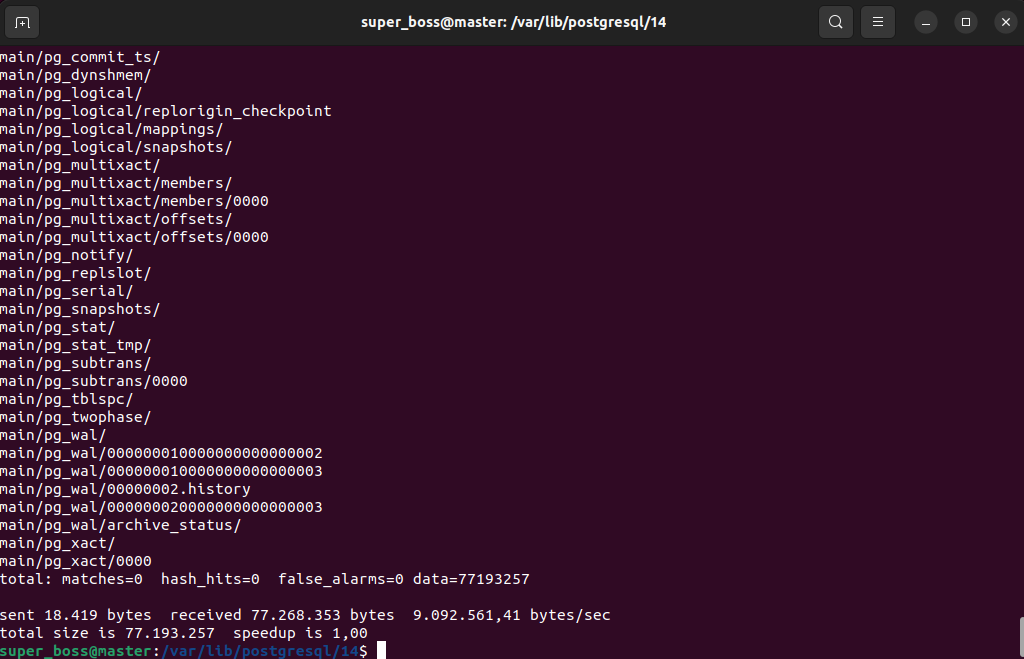


Рисунок 31. Результат

Данные с резервного узла появились на основном.

Для перевода резервного узла обратно в режим резерва нужно создать файл standby.signal в директории кластера на резервном узле:

sudo -u postgres -i

touch 14/main/standby.signal

Резервный узел нужно перезапустить (systemctl restart postgresql).

Демонстрация исходной конфигурации:

При запуске основного узла видно следующие результаты:

Изменения, внесенные в таблицы во время сбоя, применены к таблицам на основном узле. С основного узла можно вносить изменения в таблицы.

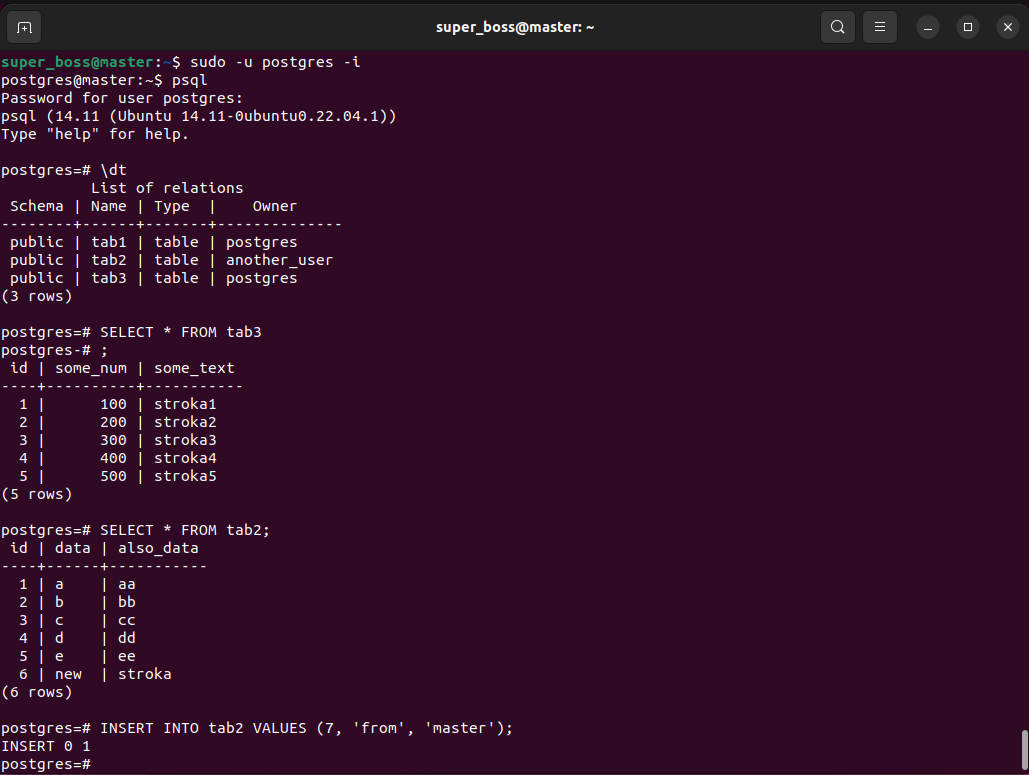


Рисунок 32. Master

Резервный узел снова имеет права только на чтение, на нем уже появляются изменения, внесенные на основном:

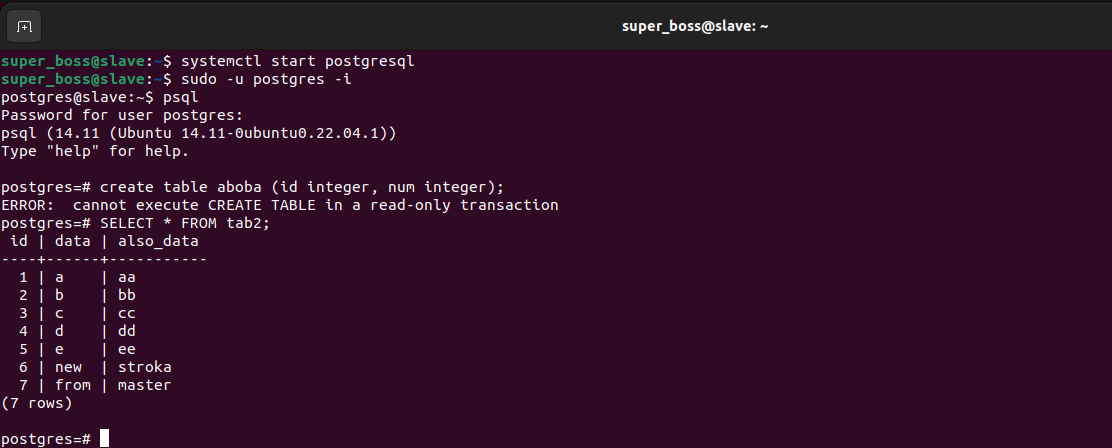


Рисунок 33. Slave

Таким образом, после сбоя первоначальная конфигурация узлов была полностью восстановлена.

# Вывод

В результате выполнения лабораторной работы был получен опыт настройки потоковой репликации между основным и резервным узлами. Также было проведено восстановление узлов после сбоя на основном сервере путем использования фейловера, в результате чего резервный узел некоторое время исполнял роль основного.