**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

ПП.09.02.07-1.24.201.03

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

ПМ.07 Соадминистрирование баз данных и серверов

|  |  |  |
| --- | --- | --- |
| Руководитель от предприятия:  М.П. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (С.О. Готовчикова) |
| Руководитель от техникума: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (С.Н. Касьяненко) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Д.А. Бодоев) |

Иркутск 2024

Содержание

[Введение 3](#_Toc164463867)

[1. Настройка мониторинга 4](#_Toc164463868)

[2. Настройка конфигурационного файла 10](#_Toc164463869)

[3. Настройка пользователя и создание базы данных 13](#_Toc164463870)

[4. Настройка PgBouncer 18](#_Toc164463871)

[5. Проверка открытых портов 21](#_Toc164463872)

[Заключение 24](#_Toc164463873)

[Список используемых источников 25](#_Toc164463874)

Введение

Производственная практика является неотъемлемой частью процесса обучения студентов специальности "Администратор баз данных". Она предоставляет возможность студентам применить полученные теоретические знания на практике и ознакомиться с реальными задачами и требованиями, стоящими перед специалистами в области администрирования баз данных и серверов.

Целью данного отчета является систематизация и анализ опыта, накопленного в ходе прохождения производственной практики. В рамках данного исследования рассматриваются основные задачи, выполненные в течение практики, а также примененные методы и технологии, влияющие на эффективность работы с базами данных и серверами.

Отчет основан на конкретных практических навыках, полученных в ходе работы, и направлен на выявление ключевых аспектов, влияющих на процесс администрирования баз данных и серверов. Анализируются проблемы, с которыми сталкиваются специалисты в данной области, и предлагаются практические рекомендации по их решению.

Итак, в данном отчете представлено комплексное изучение производственной практики по соадминистрированию баз данных и серверов, с акцентом на приобретенном опыте, примененных методах и технологиях, а также анализе проблем и поиске путей их решения.

1. Настройка мониторинга

Для настройки мониторинга долгих запросов в PostgreSQL и записи их в отдельный файл я использовал инструмент мониторинга, pg\_stat\_statements, который отслеживает выполнение SQL-запросов и их статистику. Но поскольку по умолчанию pg\_stat\_statements не отслеживает продолжительность запросов, нужно внести некоторые изменения в конфигурацию PostgreSQL. Рисуноки 1 – 2.

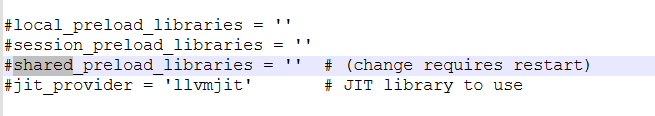


Рисунок 1 – Файл конфигурации до

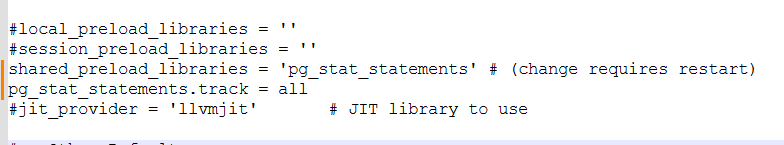


Рисунок 2 – Файл конфигурации после

Также необходимо настроить логирование продолжительности запросов рисунок 3.

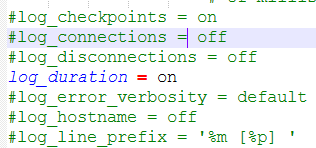


Рисунок 3 – настройка продолжительности запросов

Затем настраиваем логирование длительных запросов рисунок 4.

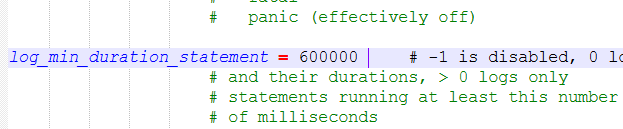


Рисунок 4 – настройка длительных запросов

Настройка записи логов в отдельный файл, указываем путь, где будет храниться файл, и даём ему название рисунок 5.

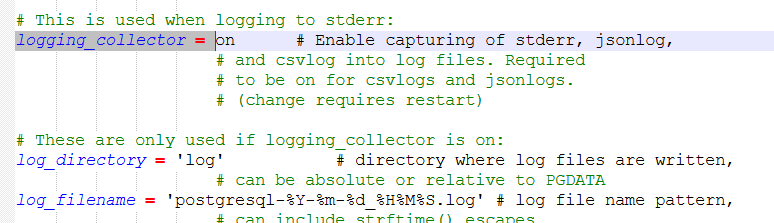


Рисунок 5 – настройка записи логов в отдельный файл

Далее прописываем команду в pgAdmin4 на рисунке 6 она показана.

Это расширение отслеживает статистику использования SQL-запросов, включая их продолжительность.

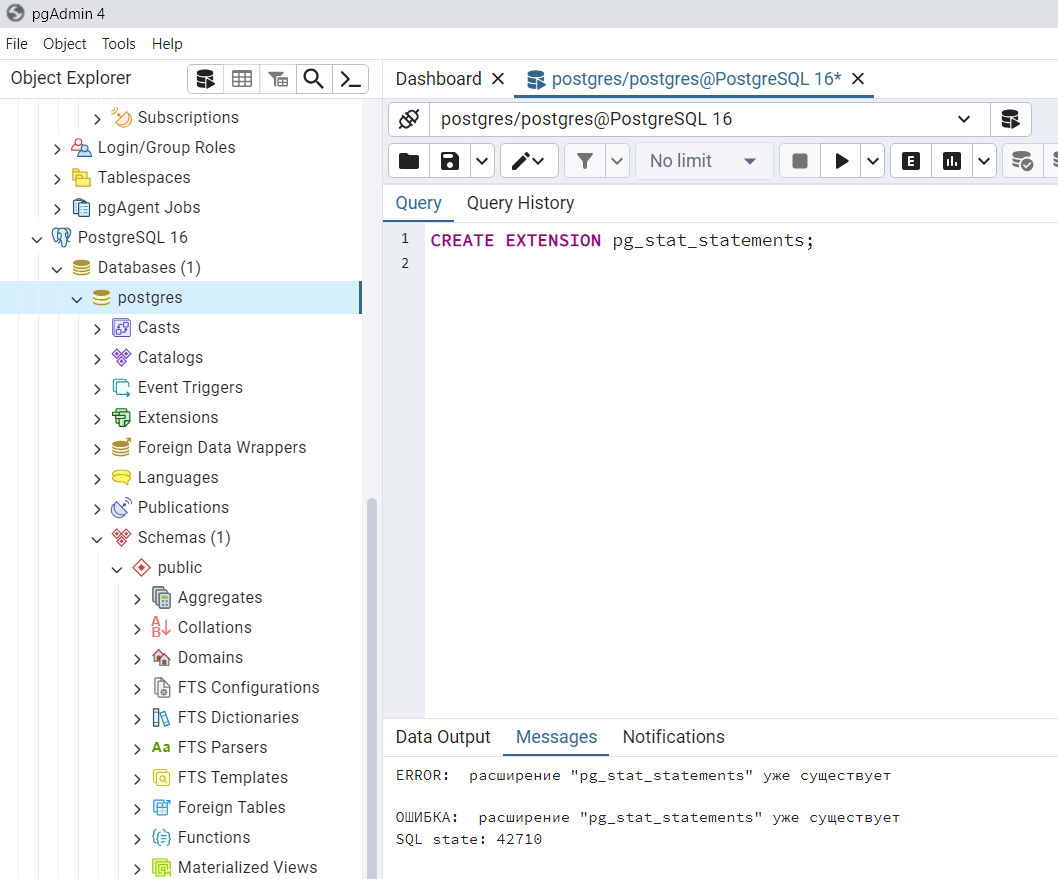


Рисунок 6 – расширение pg\_stat\_statements

На рисунке 7 изображен файл мониторинга запросов продолжительностью более 10 минут.

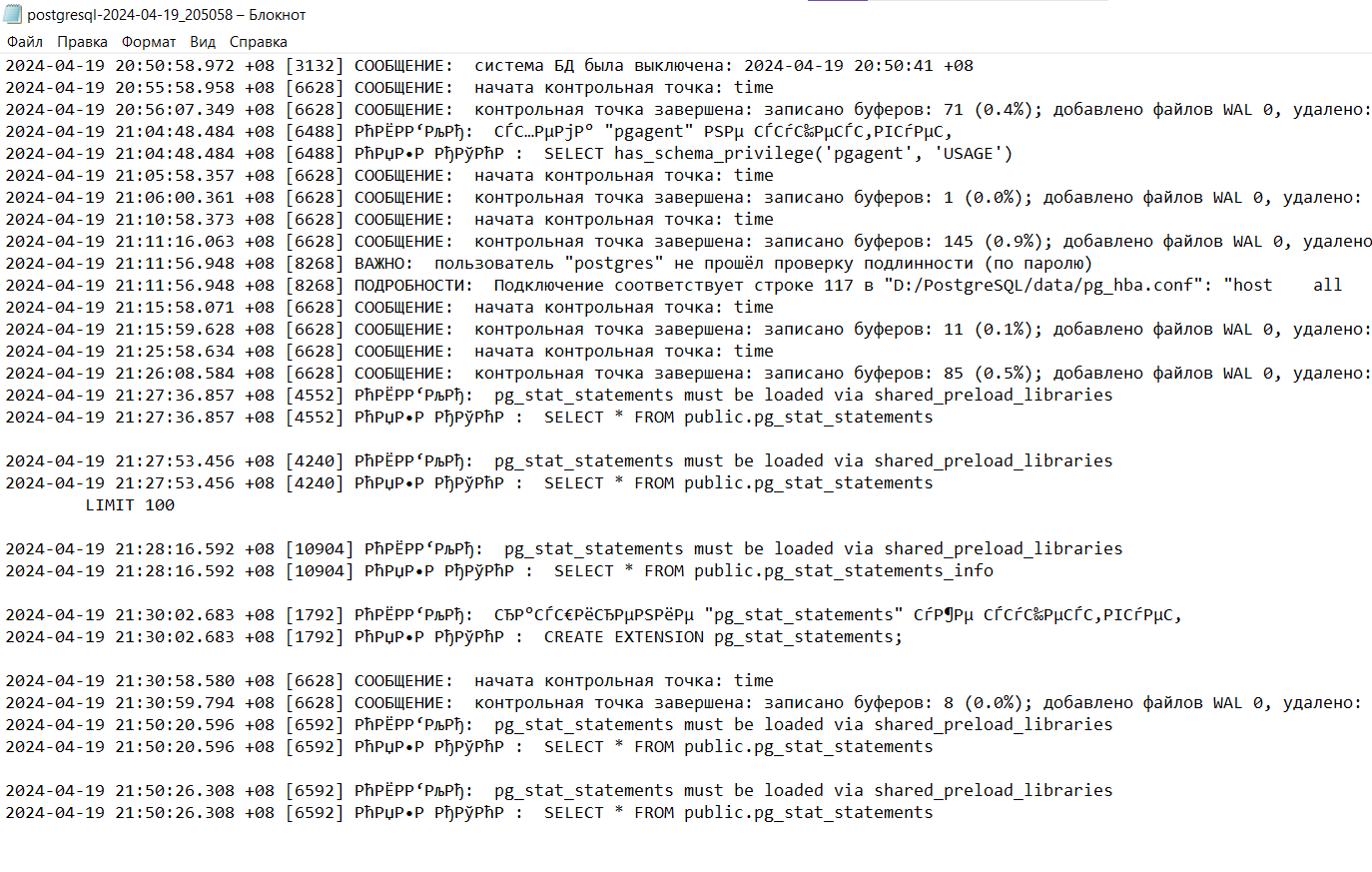


Рисунок 7 – файл мониторинга запросов

Построение на основе Postgrsql Master-Slave репликации. Построение репликации Master-Slave в PostgreSQL обеспечивает отказоустойчивость, масштабируемость и возможность распределения нагрузки на базу данных. В файле pg\_hba.conf настраиваем правила доступа для репликации рисунок 8.

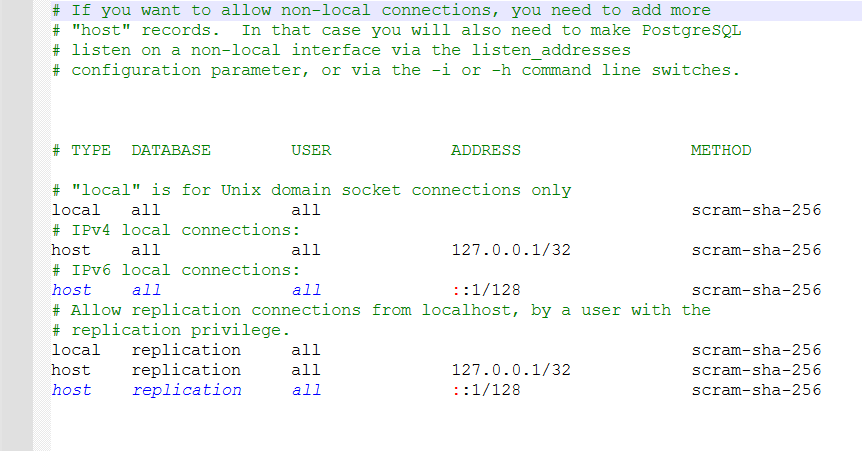


Рисунок 8 – файл pg\_hba.conf

Создаём пользователя для репликации рисунок 9.

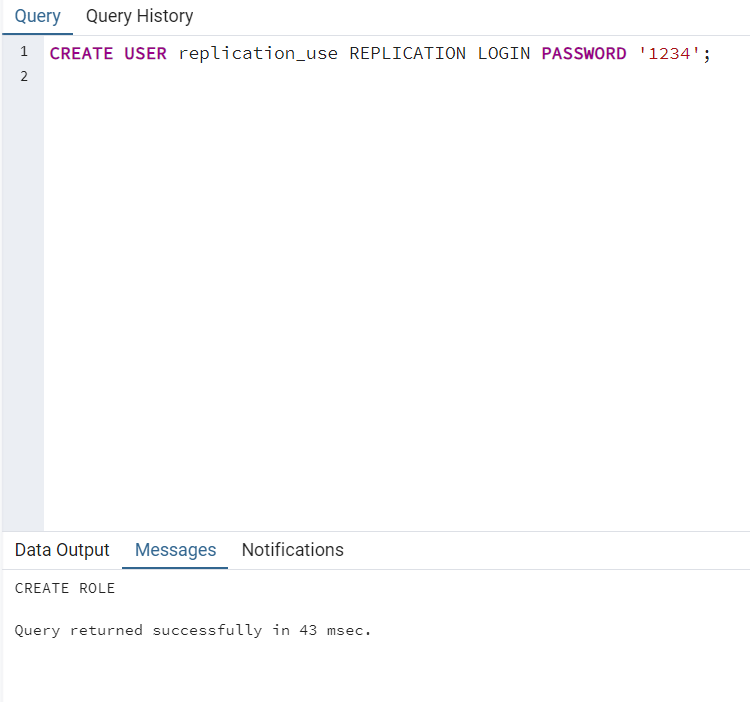


Рисунок 9 – создание пользователя для репликации

Настраиваем параметры репликации в файле postgresql.conf рисуноки 10 – 11.

1. wal\_level = replica

* minimal:
* Это самый базовый уровень WAL.
* PostgreSQL записывает в журнал только минимальное количество информации, необходимое для восстановления базы данных после сбоя.
* В режиме minimal PostgreSQL не поддерживает репликацию.
* replica:
* Этот уровень включает в себя все, что включает уровень minimal, плюс дополнительную информацию, необходимую для поддержки репликации.
* PostgreSQL записывает в WAL достаточно информации, чтобы поддерживать чтение реплик (слейвов) и восстановление на основе точки восстановления.
* Это наиболее распространенный и рекомендуемый уровень для репликации Master-Slave.
* logical:
* Этот уровень включает в себя все, что включает уровень replica, плюс дополнительную информацию, необходимую для поддержки логической репликации.
* PostgreSQL записывает в WAL дополнительную информацию, которая позволяет логической репликации воспроизводить изменения на уровне строк, а не на уровне блоков.
* Этот уровень подходит для сценариев, когда требуется более гибкая и настраиваемая репликация, например, для репликации в сторонние системы или между разными версиями PostgreSQL.
* Этот параметр определяет, как много информации будет записано в WAL (Write-Ahead Logging) журнал.
* Значение "replica" указывает, что PostgreSQL будет записывать в WAL достаточно информации, чтобы поддерживать репликацию и чтение реплики. Это обеспечивает полную поддержку репликации и позволяет слейвам читать данные.
* В режиме "replica" PostgreSQL записывает данные, необходимые для восстановления слейвов и поддержания целостности данных.

1. max\_wal\_senders = 10

* Этот параметр устанавливает максимальное количество процессов отправки WAL (WAL sender processes).
* WAL sender процессы отправляют журналы WAL с мастера на слейвы во время репликации.
* Значение этого параметра определяет, сколько слейвов можно подключить к мастеру для репликации.



Рисунок 10 – команда wal\_level



Рисунок 11 – команда max\_wal\_senders

Создаём каталог для хранения данных PostgreSQL.

Инициализируем слейв с помощью резервной копии мастера рисунок 12.



Рисунок 12 – инициализация слейва

1. Настройка конфигурационного файла

Для того чтобы настроить конфигурационный файл для оптимизации работы при различных аппаратных ресурсах, необходимо перейти в корневую папку, рисунок 13.

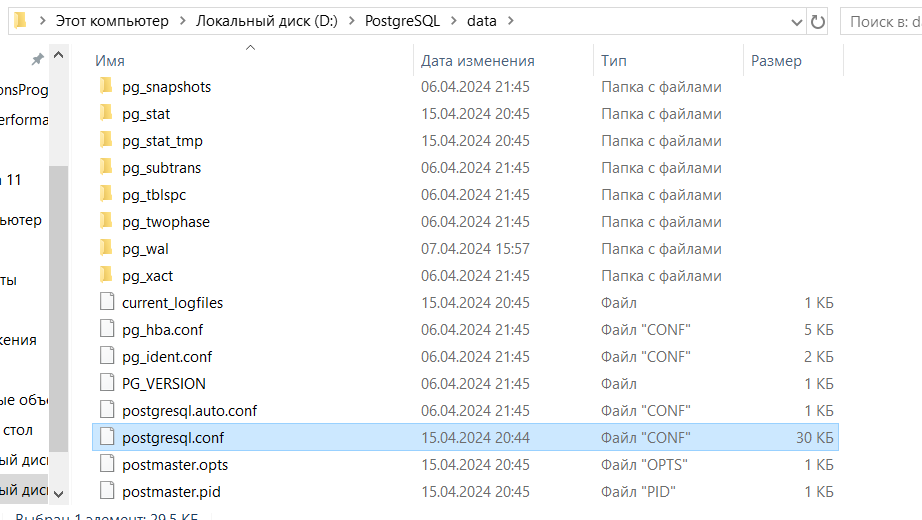


Рисунок 13 – Файл конфигурации после

Открываем файл postgresql.conf и ищем такие команды

1. shared\_buffers: Этот параметр определяет количество памяти, которая выделена для кеширования данных. Он должен быть установлен на основе объема доступной памяти на сервере. Для больших серверов рекомендуется увеличить этот параметр. Например, для сервера с 8GB оперативной памяти, можно установить значение около 2GB.
2. effective\_cache\_size: Этот параметр указывает PostgreSQL оценочный размер кэша операционной системы. Он также должен быть установлен на основе доступной памяти.
3. work\_mem: Этот параметр определяет объем памяти, выделенный для сортировки и других операций. Установите его на уровне, который позволяет выполнить наиболее требовательные запросы.
4. maintenance\_work\_mem: Этот параметр определяет объем памяти, выделенный для операций обслуживания, таких как создание индексов. Увеличьте его для выполнения операций обслуживания на больших таблицах.
5. checkpoint\_completion\_target: Этот параметр определяет, какую часть записей в журнале транзакций должен завершить каждый проход контрольной точки. Это помогает управлять скоростью записи на диск и может быть настроено для оптимального баланса между производительностью и надежностью.
6. max\_connections: Этот параметр определяет максимальное количество одновременных соединений с сервером PostgreSQL.
7. random\_page\_cost: Этот параметр определяет стоимость произвольного доступа к данным на диске по сравнению с последовательным доступом.

На рисунках 14 – 20 показаны команды в файле postgresql.conf.

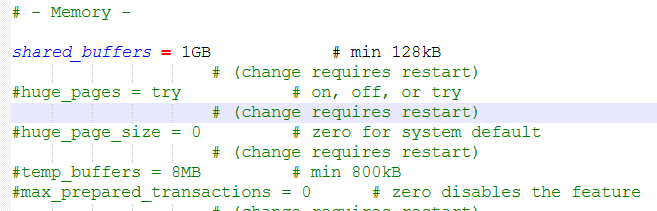


Рисунок 14 – Команда shared\_buffers

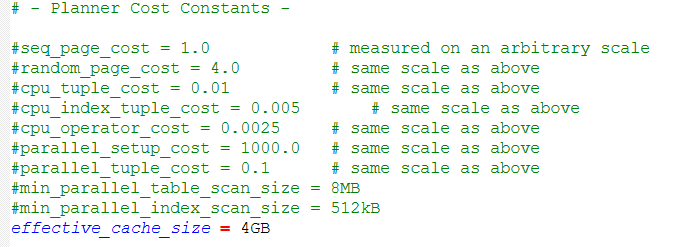


Рисунок 15 – Команда effective\_cache\_size

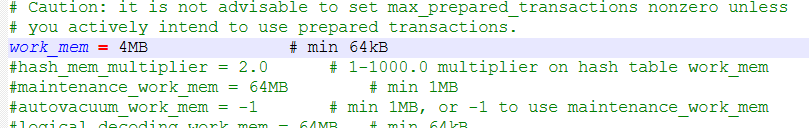


Рисунок 16 – Команда work\_mem

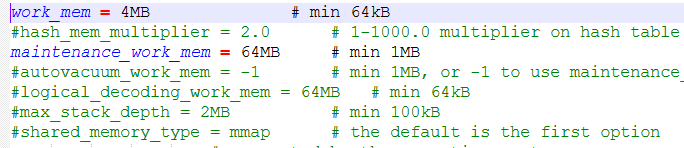


Рисунок 17 – Команда maintenance\_work\_mem



Рисунок 18 – Команда checkpoint\_completion\_target

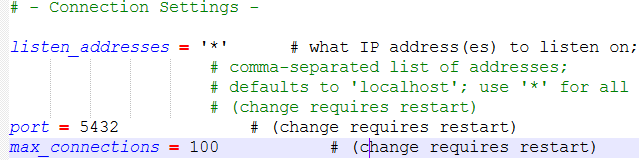


Рисунок 19 – Команда max\_connections

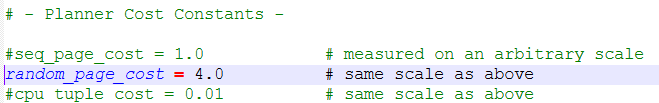


Рисунок 20 – Команда random\_page\_cost

1. Настройка пользователя и создание базы данных

На рисунках 21 – 29 показаны этапы создания базы данных и настройки пользователя в pgAdmin4.

Сначала создаем базу данных, для учета посещения студентами учебного заведения.

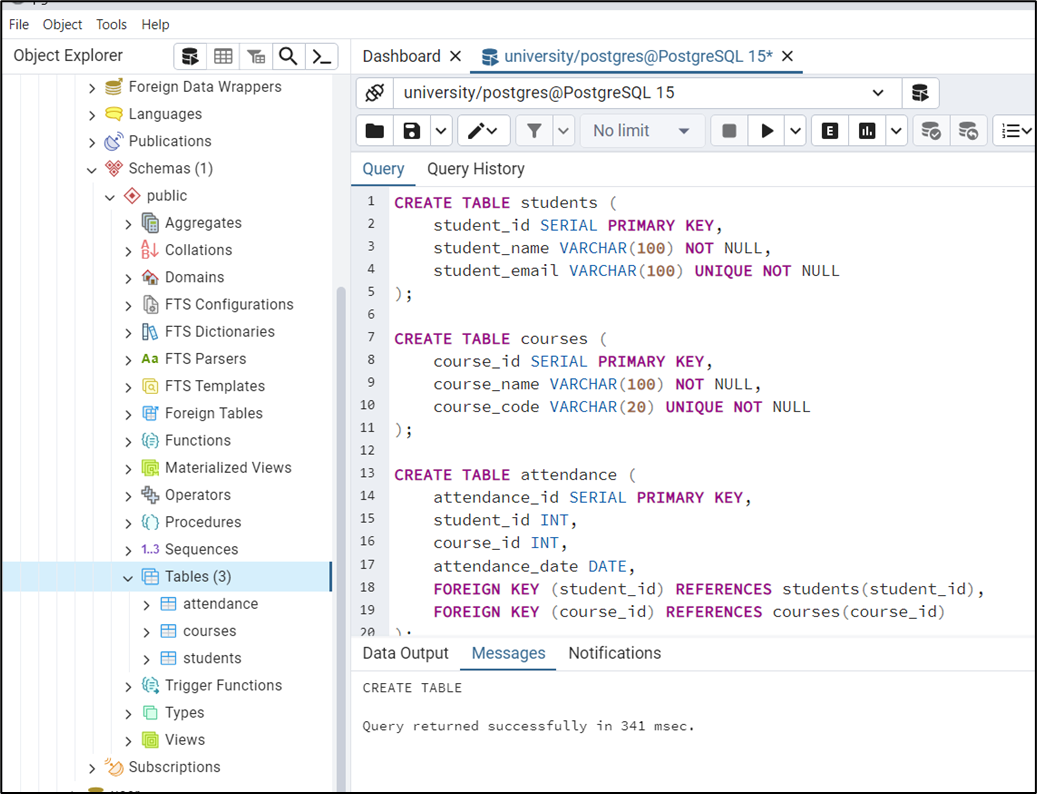


Рисунок 21 – создание базы данных university

Далее создаём пользователя и выдаём права доступа.

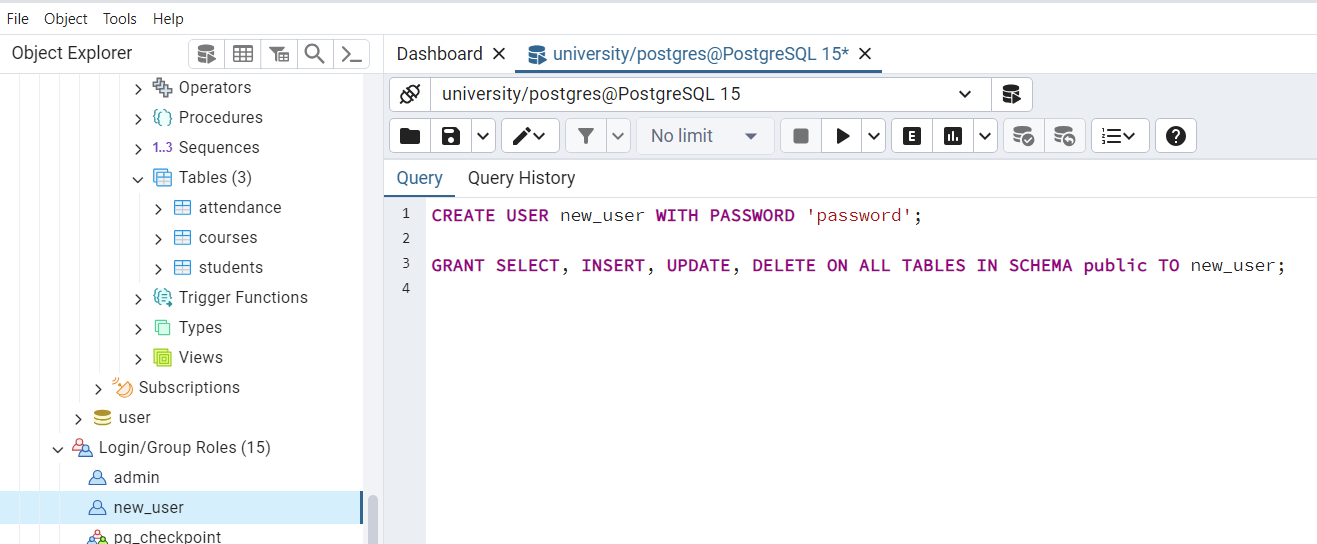


Рисунок 22 – создание пользователя и выдача прав доступа

Настройка пользователя без использования SQL – запросов.

В обозревателе объектов выбираем роли и группы входа, кликаем правой кнопкой мыши и откроется диалоговое окно, выбираем создать роли и группы входа.

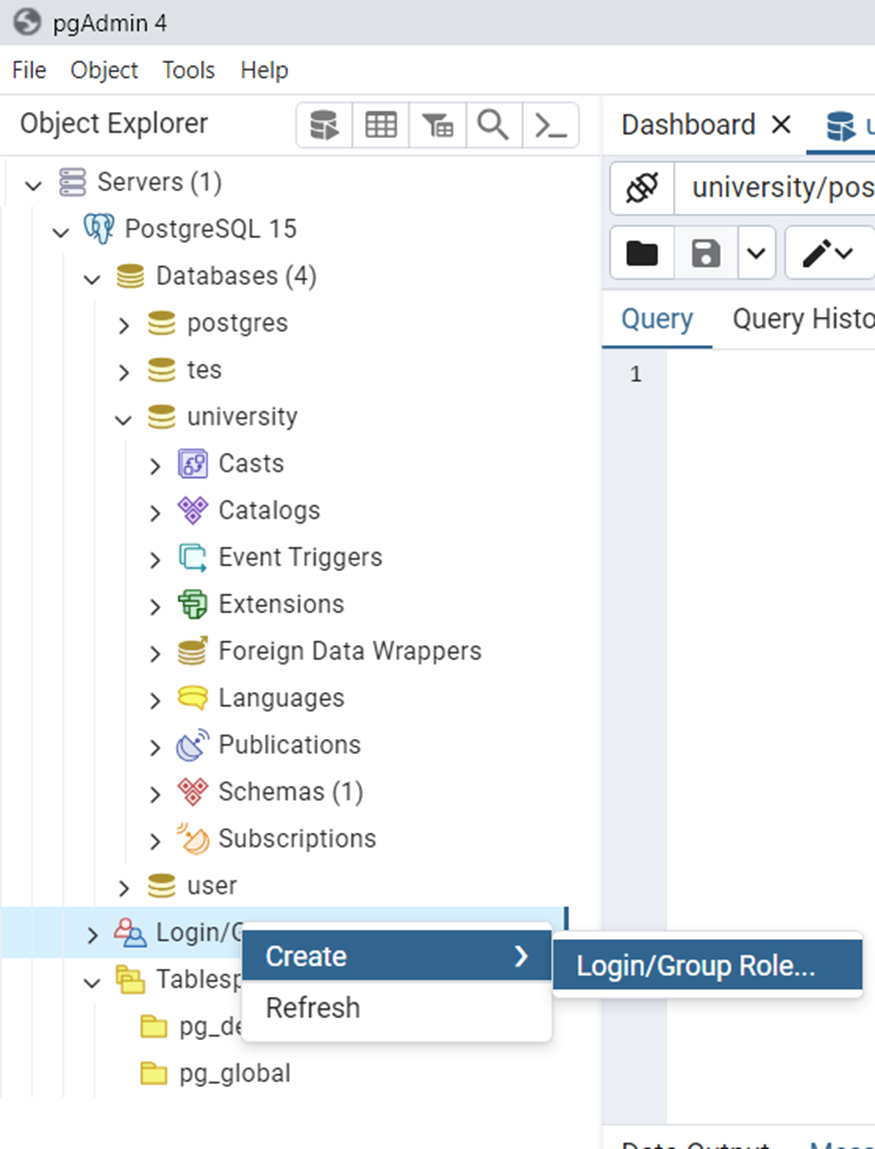


Рисунок 23 – Обозреватель объектов

Открылось окно, в котором указываем имя пользователя.

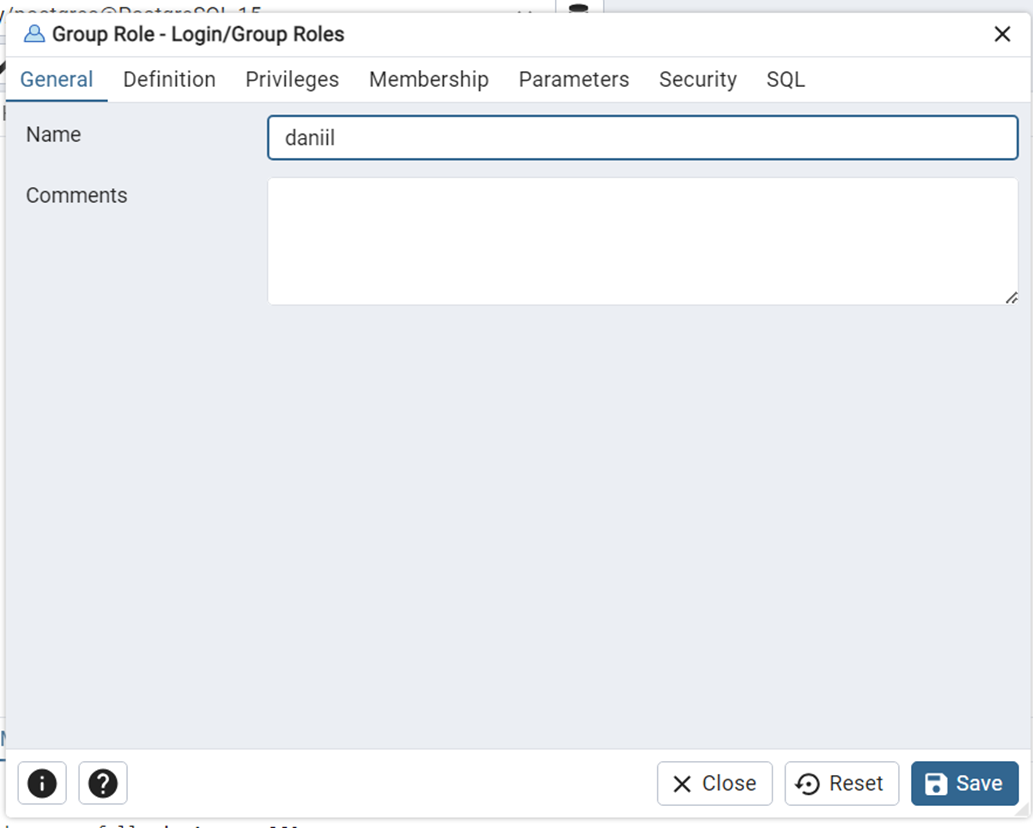


Рисунок 24 – Окно ввода имени пользователя

Во вкладке definition указываем пароль пользователя.

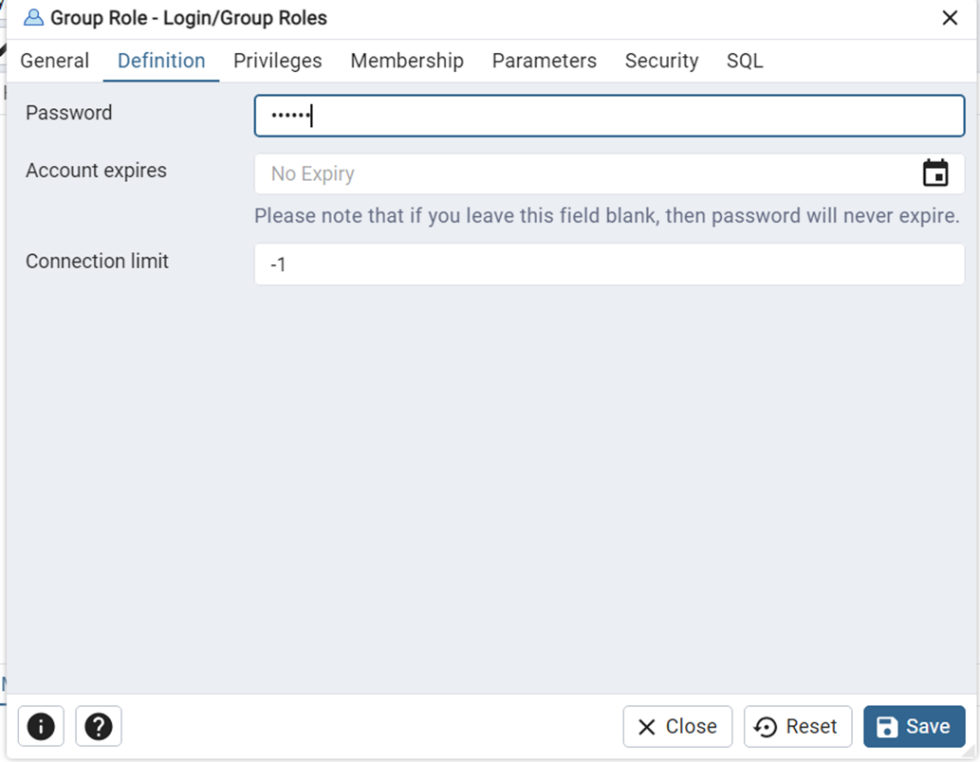


Рисунок 25 – Окно ввода пароля пользователя

Пользователь создан, но не обладает правами доступа, для того чтобы выдать ему права доступа, необходимо кликнуть правой кнопкой мыши по пользователю.

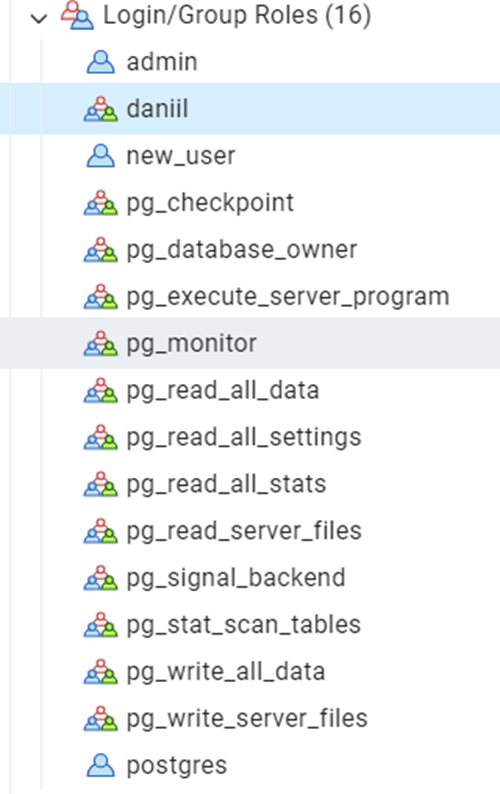


Рисунок 26 – Созданный пользователь

Открылось диалоговое окно, в котором выбираем properties.

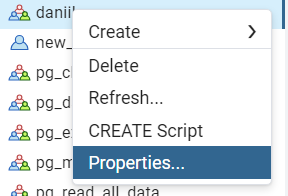


Рисунок 27 – Окно свойства пользователя

Переходим во вкладку привилегии и выбираем: can login, superuser, create roles, create database и inherit rights from the parent roles нажимаем save для сохранения изменений.

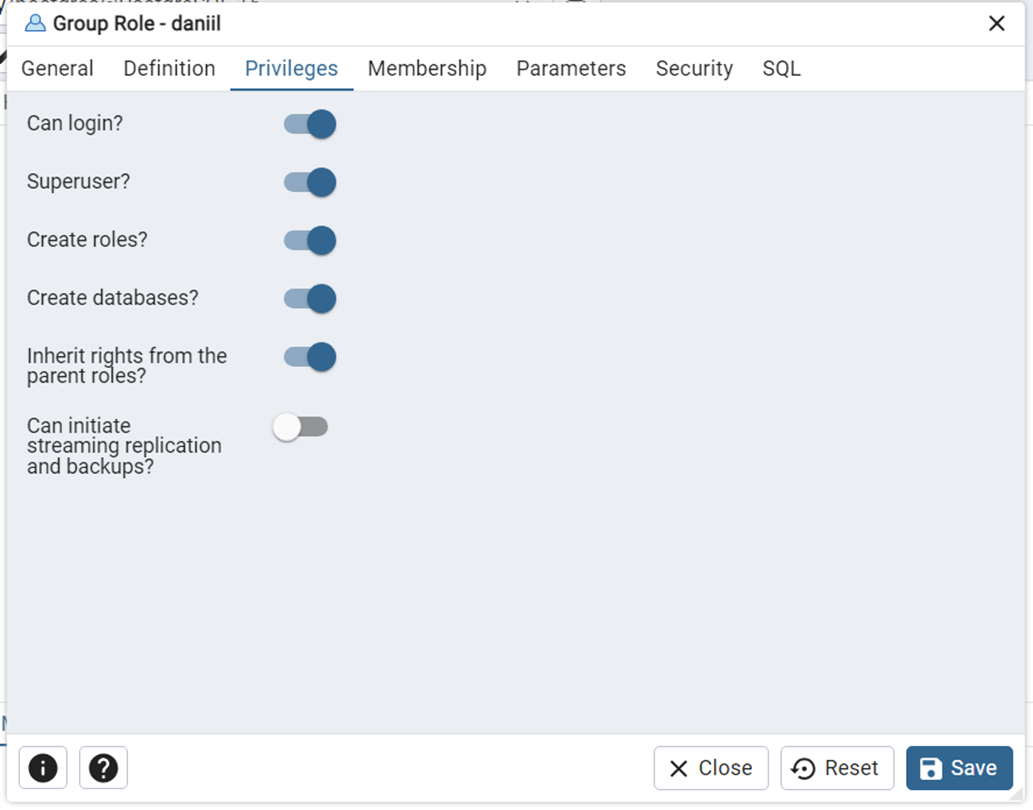


Рисунок 28 – Окно привилегии

Пользователь создан и обладает правами доступа.

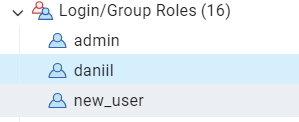


Рисунок 29 – Окно пользователей

1. Настройка PgBouncer

Устанавливаем pgBouncer. Я использовал Stack Builder. Выбираем наш сервер, нажимаем следующий рисунок 30 – 31.

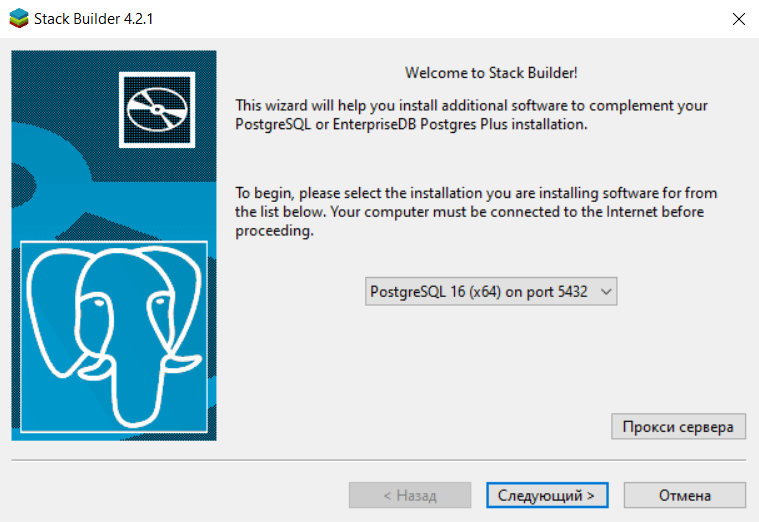


Рисунок 30 – Выбор сервера

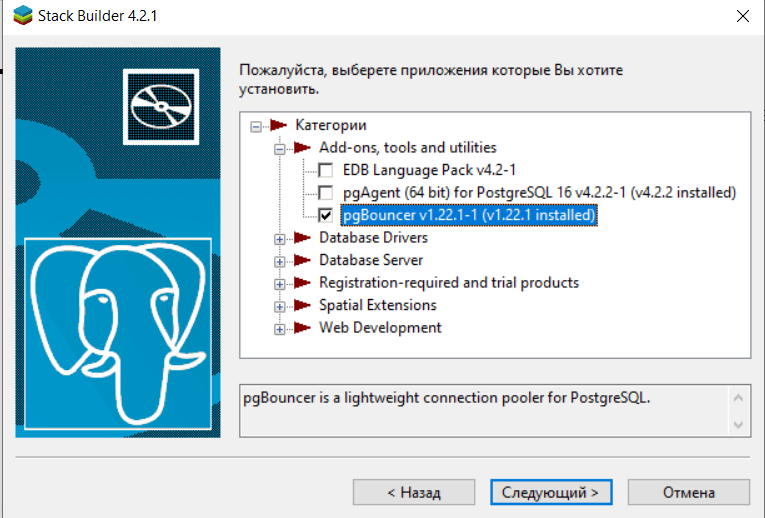


Рисунок 31 – Установка pgBouncer

Настройка файла pgbouncer.ini, указываем хост и порт рисунок 32.

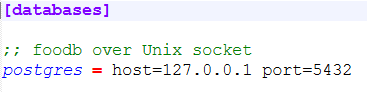


Рисунок 32 – Файл pgbouncer.ini

Создаём файл userlist.txt, в котором будут храниться пользователи и зашифрованные пароли. Формат файла следующий: username = password.

Используем команду оболочки операционной системы для добавления информации о пользователях и их зашифрованных паролях в файл userlist.txt.

Используем утилиту pgbouncer для шифрования паролей и добавления их в этот файл рисунки 33 – 34:

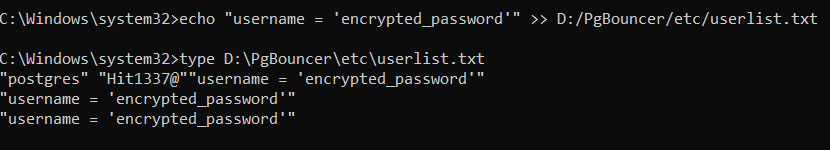


Рисунок 33 – командная строка

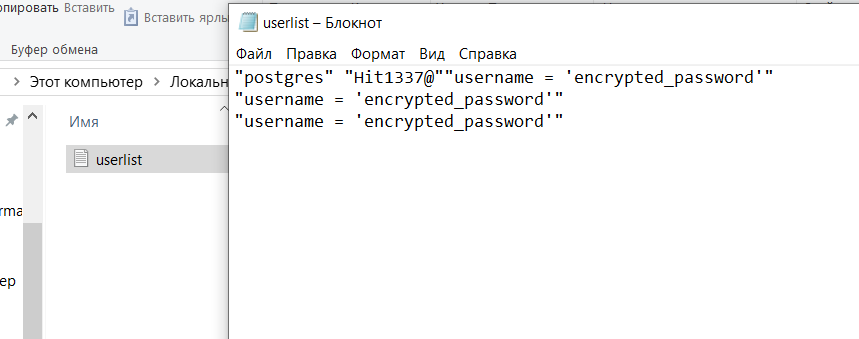


Рисунок 34 – файл userlist.txt

Настройка правил iptables обычно выполняется на самом сервере, на котором работает PostgreSQL.

Открываем терминал и подключаемся к серверу удаленно через SSH.

Вводим следующие команды рисунок 35.

Эти команды добавят правила iptables, разрешающие входящие TCP-соединения на порт 5432 только с IP-адреса 192.168.1.100, а также отклоняющие все остальные входящие соединения на этот порт.

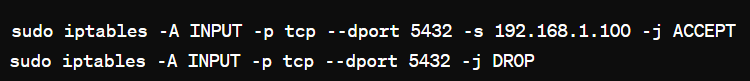


Рисунок 35 – команды

Проверяем, что правила применяются и работают, проверив текущие правила iptables рисунок 36.



Рисунок 36 – команда для проверки

Мы увидим список текущих правил iptables.

1. Проверка открытых портов

Nmap используется для активного зондирования целевой сети на наличие активных хостов (обнаружение хостов), сканирования портов, обнаружения ОС, сведений о версии и активных служб, запущенных на хостах, которые включены.

Сканирование портов является одной из функций Nmap, в которой инструмент определяет состояние портов на активных хостах в сети. Статус портов может быть открытым, фильтрованным или закрытым.

Сканирование TCP SYN (-sS): сканирование SYN часто называют “полуоткрытым” или “скрытым” сканированием. Получает ПЕРВЫЙ пакет для закрытого порта и не отвечает для отфильтрованных портов. Единственное различие заключается в том, как они обрабатывают открытые порты. SYN scan отправляет ответный пакет на сервер с установленным ФЛАГОМ СБРОСА (но не ACK, который обычно используется по умолчанию при фактическом трехстороннем квитировании) после получения SYN / ACK от целевого сервера. Это делается для того, чтобы сервер не мог постоянно отправлять запросы на установление соединения и тем самым сократить время сканирования.

Этот тип сканирования называется скрытым сканированием из-за следующих преимуществ:

* Быстрее, потому что для этого не нужно выполнять полное трехстороннее рукопожатие.
* Некоторые приложения часто регистрируют только те соединения, которые полностью установлены. Таким образом, приложения, прослушивающие открытые порты, не регистрируют эти соединения, что делает SYN scan “скрытым”.

На рисунке 37 показан метод TCP SYN (-sS).

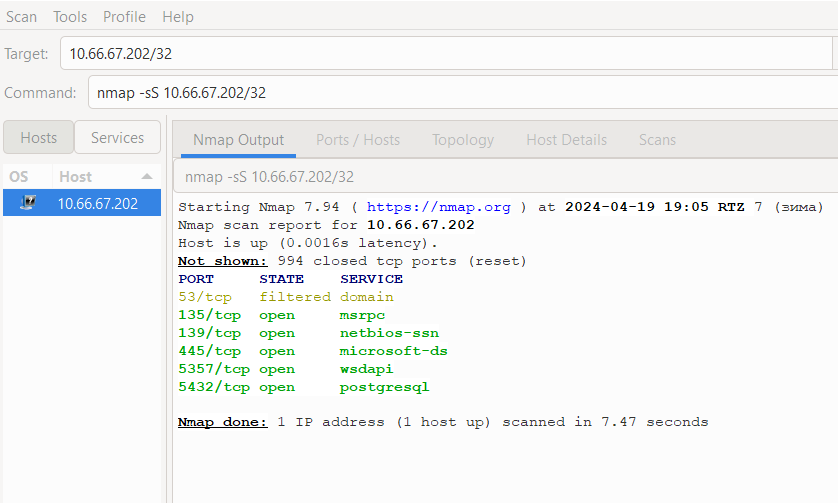


Рисунок 37 – TCP SYN (-sS)

UDP-сканирование (-sU): UDP в отличие от TCP не выполняет квитирование для установления соединения перед отправкой пакетов данных на целевой порт, а скорее отправляет пакеты в надежде, что пакеты будут получены целевым портом. Вот почему UDP-соединения часто называют “без состояния”. Этот тип соединения более эффективен, когда скорость превосходит качество, например, при совместном использовании видео. Поскольку от целевого порта не будет подтверждения о том, получил ли он пакет, сканирование UDP становится более сложным и намного медленнее.

Когда после отправки UDP-пакета нет ответа от целевого порта, это часто означает, что порт либо “открыт”, либо работает за брандмауэром, “фильтруется” и в этом случае сервер просто отбросит пакет без ответа.

UDP-сканирование может эффективно идентифицировать закрытые порты, поскольку целевой UDP-порт отвечает ICMP-пакетом с сообщением о том, что порт недоступен.

На рисунке 38 показан метод UDP-сканирование (-sU).

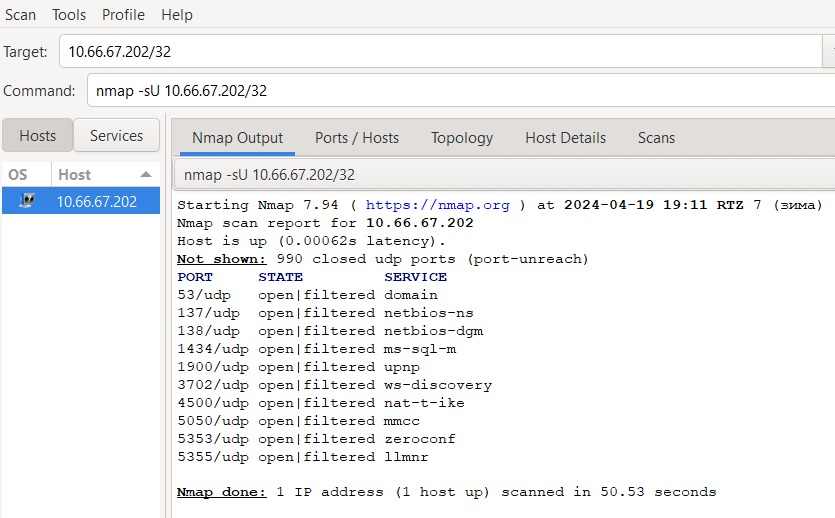


Рисунок 38 – UDP-сканирование (-sU)

На рисунке 39 изображены порты и хосты.

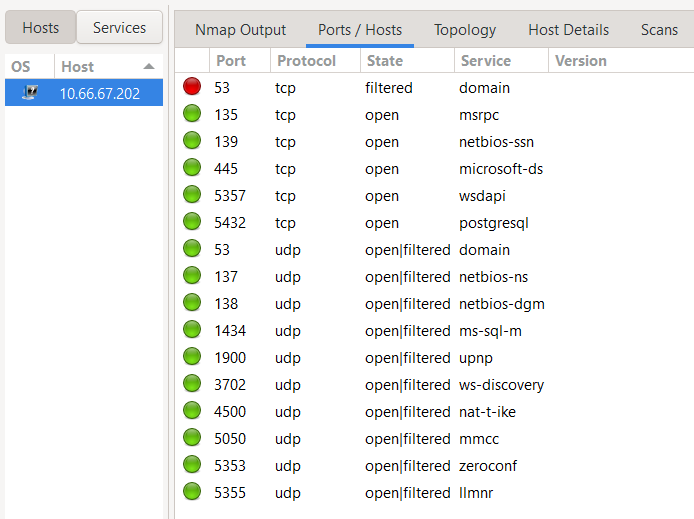


Рисунок 39 – порты и хосты

# Заключение

В ходе производственной практики я приобрел ценный опыт, который оказался важным этапом в моем профессиональном росте. Полученные знания и навыки стали надежным фундаментом для дальнейшего развития в области информационных технологий и программирования. Практика не только позволила мне применить теоретические концепции на практике, но и укрепила мою способность адаптироваться к различным задачам и ситуациям.

Благодаря практике я освоил методы эффективного планирования и организации работы, что позволило мне успешно справляться с намеченными задачами. Я научился грамотно распределять время и ресурсы, что имело ключевое значение для достижения поставленных целей в установленные сроки.

Список используемых источников

1. "Документация PgBouncer": https://www.pgbouncer.org/ - (Дата обращения: 04.04.24) - Текст электронный.
2. "Руководство по настройке PgBouncer": https://pgbouncer.github.io/config.html - (Дата обращения: 28.03.24) - Текст электронный.
3. "Официальная документация PostgreSQL Wiki для PgBouncer": https://wiki.postgresql.org/wiki/PgBouncer - (Дата обращения: 03.04.24) - Текст электронный.
4. "Статья о настройке PgBouncer для PostgreSQL": https://habr.com/ru/company/mailru/blog/258673/ - (Дата обращения: 03.04.24) - Текст электронный.