



ООО «Программы учета»

**Программный комплекс «LIGHTING»**  
**Руководство по установке программного комплекса**  
**«LIGHTING»**

## Оглавление

1. Установка комплекса «LIGHTING» .....	2
2. Установка PostgreSQL 15 на ОС Debian.....	0
2.1. Подготовка к установке .....	0
3. Установка необходимых пакетов для PostgreSQL .....	0
4. Проверка установки.....	0
5. Настройка PostgreSQL 15 в Debian 12 .....	0
6. Создание пользователей и базы данных в PostgreSQL.....	0
7. Разрешаем подключение к PostgreSQL по сети .....	0
8. Установка среды выполнения ядра ASP.NET 8 в Debian 12.....	0
9. Установка web-сервера Nginx в Debian 12.....	0
10. Базовая настройка web-сервера Nginx в Debian 12 .....	0
11. Настройка и запуск серверного приложения «LIGHTING».....	0

## **1. Установка комплекса «LIGHTING»**

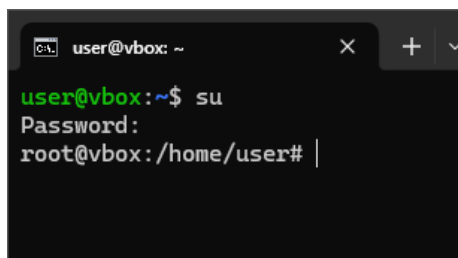
Для работы серверной части комплекса потребуется Linux-подобная операционная система. Ниже представлена инструкция по установке комплекса на ОС Debian 12.

Установка производится удаленно, при помощи программы Windows Terminal.

## 2. Установка PostgreSQL 15 на ОС Debian

### 3. Подготовка к установке

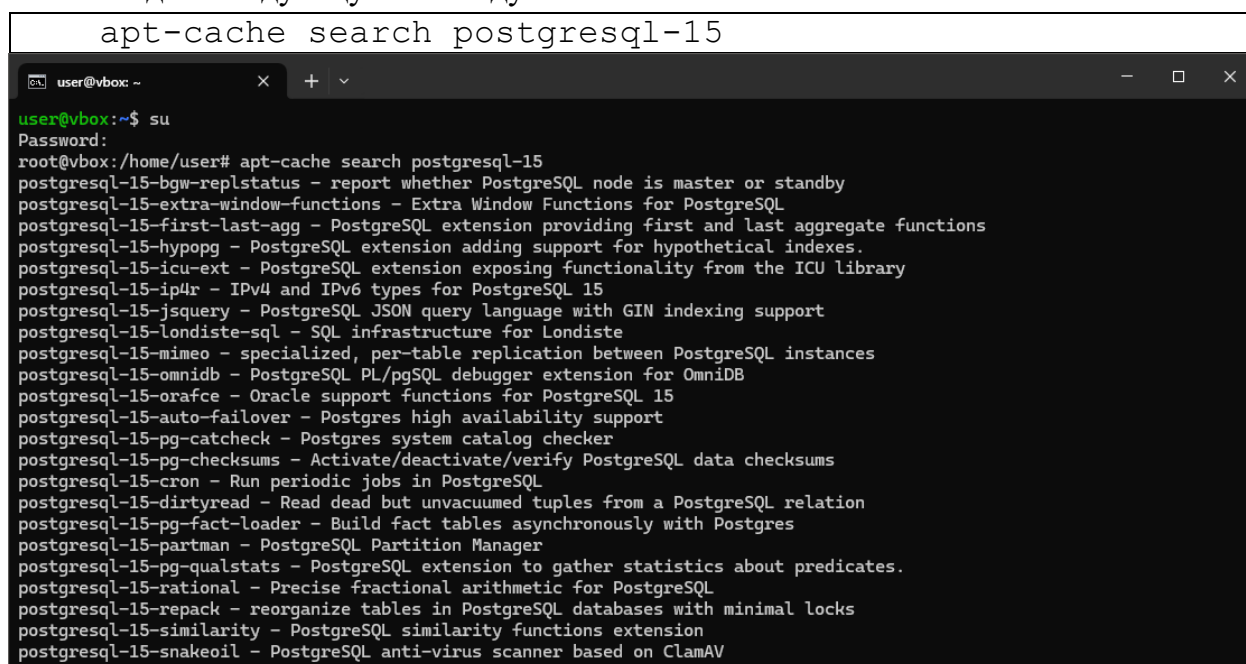
Установку и настройку СУБД PostgreSQL необходимо выполнять с правами пользователя root, поэтому давайте сразу переключимся на root. Для этого пишем команду su и вводим пароль.



```
user@vbox: ~  
user@vbox:~$ su  
Password:  
root@vbox:/home/user# |
```

Рисунок 1

Далее проверим, присутствуют ли в системе необходимые нам пакеты. Для этого вводим следующую команду:



```
apt-cache search postgresql-15  
user@vbox: ~  
user@vbox:~$ su  
Password:  
root@vbox:/home/user# apt-cache search postgresql-15  
postgresql-15-bgw-replstatus - report whether PostgreSQL node is master or standby  
postgresql-15-extra-window-functions - Extra Window Functions for PostgreSQL  
postgresql-15-first-last-agg - PostgreSQL extension providing first and last aggregate functions  
postgresql-15-hypopg - PostgreSQL extension adding support for hypothetical indexes.  
postgresql-15-icu-ext - PostgreSQL extension exposing functionality from the ICU library  
postgresql-15-ip4r - IPv4 and IPv6 types for PostgreSQL 15  
postgresql-15-jquery - PostgreSQL JSON query language with GIN indexing support  
postgresql-15-londiste-sql - SQL infrastructure for Londiste  
postgresql-15-mimeo - specialized, per-table replication between PostgreSQL instances  
postgresql-15-omnidb - PostgreSQL PL/pgSQL debugger extension for OmniDB  
postgresql-15-orafce - Oracle support functions for PostgreSQL 15  
postgresql-15-auto-failover - Postgres high availability support  
postgresql-15-pg-catcheck - Postgres system catalog checker  
postgresql-15-pg-checksums - Activate/deactivate/verify PostgreSQL data checksums  
postgresql-15-cron - Run periodic jobs in PostgreSQL  
postgresql-15-dirtyread - Read dead but unvacuumed tuples from a PostgreSQL relation  
postgresql-15-pg-fact-loader - Build fact tables asynchronously with Postgres  
postgresql-15-partman - PostgreSQL Partition Manager  
postgresql-15-pg-qualstats - PostgreSQL extension to gather statistics about predicates.  
postgresql-15-rational - Precise fractional arithmetic for PostgreSQL  
postgresql-15-repack - reorganize tables in PostgreSQL databases with minimal locks  
postgresql-15-similarity - PostgreSQL similarity functions extension  
postgresql-15-snakeoil - PostgreSQL anti-virus scanner based on ClamAV
```

Рисунок 2

В данном примере в Debian 12 присутствует нужная нам версия PostgreSQL 15, и можно переходить к разделу «Установка необходимых пакетов для PostgreSQL». Однако, если поиск пакетов не выдал результатов, необходимо добавить стандартные репозитории Debian 12.

```
echo -e "deb http://deb.debian.org/debian bookworm main  
non-free-firmware\n\  
deb-src http://deb.debian.org/debian bookworm main\n\  
deb http://security.debian.org/debian-security  
bookworm-security main\n\  
deb-src http://security.debian.org/debian-security  
bookworm-security main\n\  
"
```

```
deb    http://deb.debian.org/debian    bookworm-updates
main\n\
deb-src http://deb.debian.org/debian    bookworm-updates
main" >> /etc/apt/sources.list
```

*Рисунок 3*

После добавления репозитория необходимо обновить список пакетов, выполнив команду:

```
apt update
```

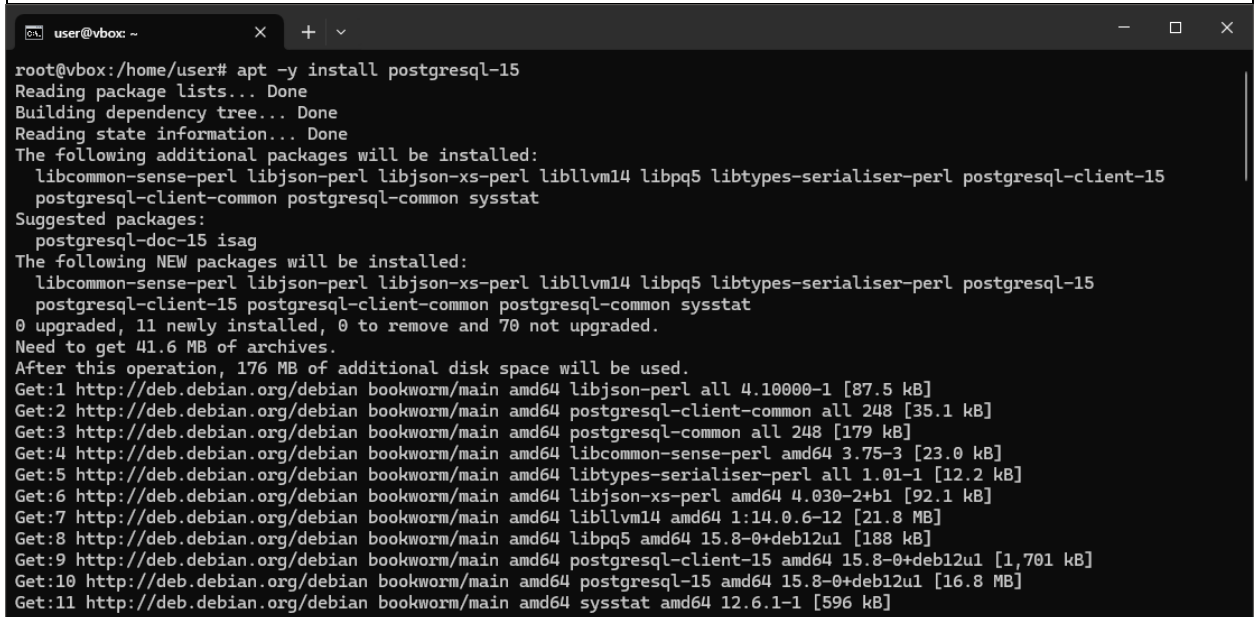
*Рисунок 4*

После обновления списка пакетов должна появиться нужная нам версия PostgreSQL 15, и мы можем перейти к ее установке.

## 4. Установка необходимых пакетов для PostgreSQL

Для установки PostgreSQL и стандартных базовых утилит необходимо установить пакет postgresql-15, это делается следующей командой.

```
apt -y install postgresql-15
```



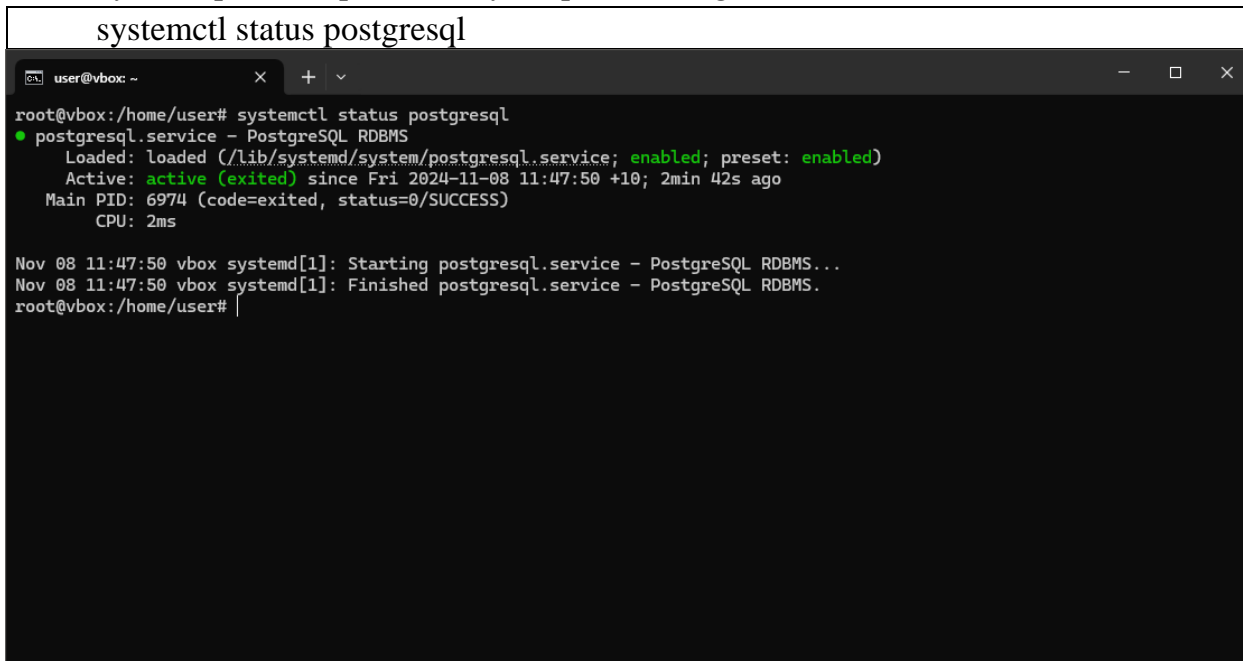
```
root@vbox: /home/user# apt -y install postgresql-15
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcommon-sense-perl libjson-perl libjson-xs-perl libllvm14 libpq5 libtypes-serialiser-perl postgresql-client-15
  postgresql-client-common postgresql-common sysstat
Suggested packages:
  postgresql-doc-15 isag
The following NEW packages will be installed:
  libcommon-sense-perl libjson-perl libjson-xs-perl libllvm14 libpq5 libtypes-serialiser-perl postgresql-15
  postgresql-client-15 postgresql-client-common postgresql-common sysstat
0 upgraded, 11 newly installed, 0 to remove and 70 not upgraded.
Need to get 41.6 MB of archives.
After this operation, 176 MB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main amd64 libjson-perl all 4.10000-1 [87.5 kB]
Get:2 http://deb.debian.org/debian bookworm/main amd64 postgresql-client-common all 248 [35.1 kB]
Get:3 http://deb.debian.org/debian bookworm/main amd64 postgresql-common all 248 [179 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 libcommon-sense-perl amd64 3.75-3 [23.0 kB]
Get:5 http://deb.debian.org/debian bookworm/main amd64 libtypes-serialiser-perl all 1.01-1 [12.2 kB]
Get:6 http://deb.debian.org/debian bookworm/main amd64 libjson-xs-perl amd64 4.030-2+b1 [92.1 kB]
Get:7 http://deb.debian.org/debian bookworm/main amd64 libllvm14 amd64 1:14.0.6-12 [21.8 MB]
Get:8 http://deb.debian.org/debian bookworm/main amd64 libpq5 amd64 15.8-0+deb12u1 [188 kB]
Get:9 http://deb.debian.org/debian bookworm/main amd64 postgresql-client-15 amd64 15.8-0+deb12u1 [1,701 kB]
Get:10 http://deb.debian.org/debian bookworm/main amd64 postgresql-15 amd64 15.8-0+deb12u1 [16.8 MB]
Get:11 http://deb.debian.org/debian bookworm/main amd64 sysstat amd64 12.6.1-1 [596 kB]
```

Рисунок 5

## 5. Проверка установки

Чтобы убедиться, что Postgres установлен и запущен, выполним следующую команду, которая отобразит статус сервиса PostgreSQL.

```
systemctl status postgresql
```



```
root@vbox:/home/user# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; preset: enabled)
   Active: active (exited) since Fri 2024-11-08 11:47:50 +10; 2min 42s ago
     Main PID: 6974 (code=exited, status=0/SUCCESS)
        CPU: 2ms

Nov 08 11:47:50 vbox systemd[1]: Starting postgresql.service - PostgreSQL RDBMS...
Nov 08 11:47:50 vbox systemd[1]: Finished postgresql.service - PostgreSQL RDBMS.
root@vbox:/home/user#
```

Рисунок 6

PostgreSQL 15 установлен и запущен.

## **6. Настройка PostgreSQL 15 в Debian 12**

PostgreSQL установлен, теперь нужно выполнить настройку. В частности, необходимо создать пользователя, определить, какие сетевые интерфейсы будет прослушивать сервер, а также разрешить подключения по сети.



## 7. Создание пользователей и базы данных в PostgreSQL

После установки к серверу PostgreSQL мы можем подключиться только с помощью системного пользователя postgres, причем без пароля.

Давайте переключимся на пользователя postgres (данная учетная запись была создана автоматически во время установки PostgreSQL).

```
su - postgres
```

Рисунок 7

Затем запускаем утилиту psql – это консоль для PostgreSQL.

```
psql
```

Рисунок 8

Первым делом нам нужно задать пароль для пользователя postgres.

```
\password postgres
```

Рисунок 9

Затем создаем новых пользователей на сервере PostgreSQL, так как работать от имени postgres крайне не рекомендуется.

```
CREATE USER lighting WITH PASSWORD 'd14MrHN4DKPFdJP'  
CREATEDB CREATEROLE;  
CREATE USER master WITH PASSWORD '441XkTObxpH5Zyy'  
CREATEDB;  
CREATE USER editor WITH PASSWORD 'h125RMnI1aXYBjS';  
CREATE USER manager WITH PASSWORD '0At825q4a0qjLWz';  
CREATE USER viewer WITH PASSWORD 'QG9aS58N2opK3vc';
```

Рисунок 10

Далее создадим базу данных.

```
CREATE DATABASE ownlight WITH OWNER = lighting;
```

Рисунок 11

Все готово, выходим из консоли.

```
\q
```

Рисунок 12

Теперь создадим схему ownlight в базе данных ownlight. Для этого зайдём в psql под пользователем lighting.

```
psql -h localhost -U lighting -d ownlight
```

Рисунок 13

Создаем основную схему.

```
CREATE SCHEMA ownlight;
```

Рисунок 14

Укажем привилегии для других созданных пользователей.

```
GRANT ALL PRIVILEGES ON DATABASE ownlight TO master;  
GRANT CONNECT ON DATABASE ownlight TO editor;  
GRANT SELECT ON ALL TABLES IN SCHEMA ownlight TO editor;  
GRANT CONNECT ON DATABASE ownlight TO manager;
```

```
GRANT CONNECT ON DATABASE ownlight TO viewer;
```

*Рисунок 15*

Все готово, выходим из консоли.

```
\q
```

*Рисунок 16*

Для того чтобы переключиться обратно на пользователя root, используем команду.

```
exit
```

*Рисунок 17*

## 8. Разрешаем подключение к PostgreSQL по сети

По умолчанию PostgreSQL прослушивает только адрес localhost, поэтому для того, чтобы была возможность подключаться по сети, нужно указать, какие сетевые интерфейсы будет прослушивать PostgreSQL. Для примера укажем, что прослушивать нужно все доступные интерфейсы. Если у Вас несколько сетевых интерфейсов, и Вы хотите, чтобы PostgreSQL использовал только один конкретный, то Вы его можете указать именно здесь.

Открываем файл postgresql.conf, например, в редакторе nano.

```
nano /etc/postgresql/15/main/postgresql.conf
```

Рисунок 18

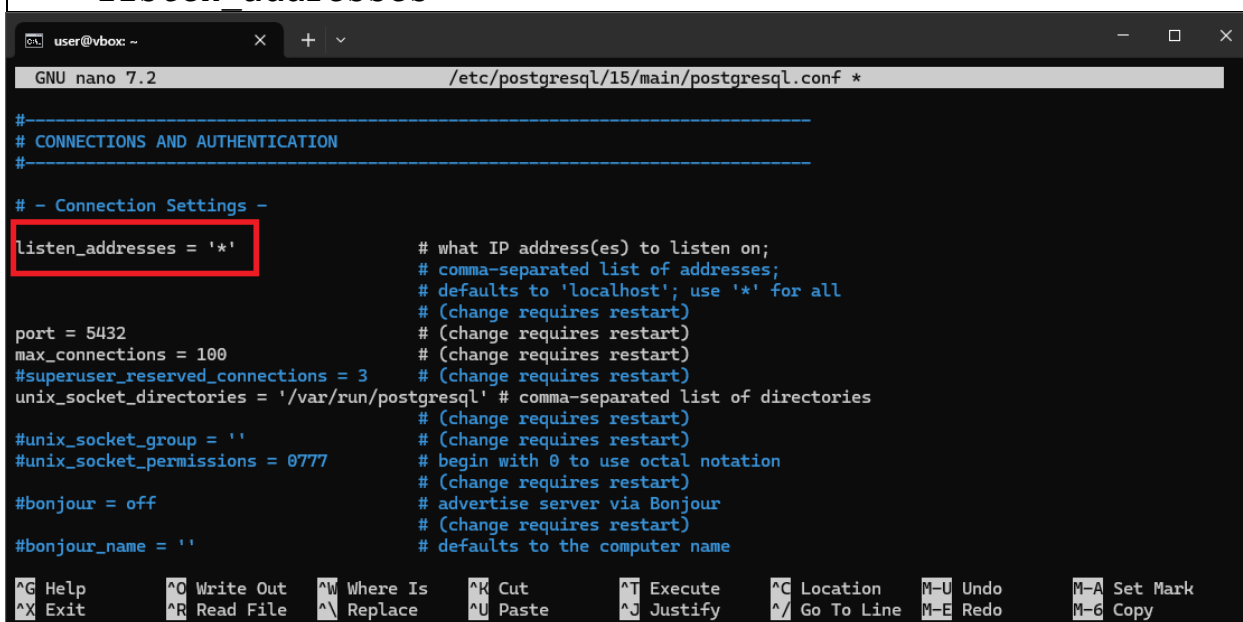
Находим следующую строку.

```
#listen_addresses = 'localhost'
```

Рисунок 19

и вносим следующие изменения (вместо звездочки Вы в случае необходимости указываете IP адрес нужного интерфейса).

```
listen_addresses = '*'
```



```
GNU nano 7.2 /etc/postgresql/15/main/postgresql.conf *
#-----
# CONNECTIONS AND AUTHENTICATION
#-----
# - Connection Settings -
listen_addresses = '*'          # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost'; use '*' for all
                                # (change requires restart)
port = 5432                    # (change requires restart)
max_connections = 100         # (change requires restart)
#superuser_reserved_connections = 3 # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
                                                # (change requires restart)
#unix_socket_group = ''      # (change requires restart)
#unix_socket_permissions = 0777 # begin with 0 to use octal notation
                                # (change requires restart)
#bonjour = off              # advertise server via Bonjour
                                # (change requires restart)
#bonjour_name = ''         # defaults to the computer name

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

Рисунок 20

Сохраняем изменения сочетанием клавиш Ctrl + O и подтверждаем нажатием Enter, затем закрываем редактор nano, нажав Ctrl + X.

Теперь разрешим подключение из сети, для примера разрешим подключаться из сети 192.168.1.0/24. Для этого открываем файл pg\_hba.conf.

```
nano /etc/postgresql/15/main/pg_hba.conf
```

Рисунок 21

Ищем следующие строки.

```
GNU nano 7.2 /etc/postgresql/15/main/pg_hba.conf
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local all postgres peer
# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all peer
# IPv4 local connections:
host all all 127.0.0.1/32 scram-sha-256
# IPv6 local connections:
host all all ::1/128 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
host replication all 127.0.0.1/32 scram-sha-256
host replication all ::1/128 scram-sha-256
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/_ Go To Line M-E Redo M-G Copy
```

Рисунок 22

И указываем нужную нам сеть (если IPv6 Вы не будете использовать, то можете закомментировать соответствующие строки знаком #).

```
GNU nano 7.2 /etc/postgresql/15/main/pg_hba.conf
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local all postgres peer
# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all peer
# IPv4 local connections:
host all all 192.168.1.0/24 scram-sha-256
# IPv6 local connections:
host all all ::1/128 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
host replication all 127.0.0.1/32 scram-sha-256
host replication all ::1/128 scram-sha-256
[ Wrote 104 lines ]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/_ Go To Line M-E Redo M-G Copy
```

Рисунок 23

Сохраняем изменения сочетанием клавиш Ctrl + O и подтверждаем нажатием Enter, затем закрываем редактор nano, нажав Ctrl + X.

Перезапускаем PostgreSQL, чтобы изменения вступили в силу.

```
systemctl restart postgresql
```

Рисунок 24

## 9. Установка среды выполнения ядра ASP.NET 8 в Debian 12

Для входа под учетной записью суперпользователя используем команду и вводим пароль.

```
su -
```

Рисунок 25

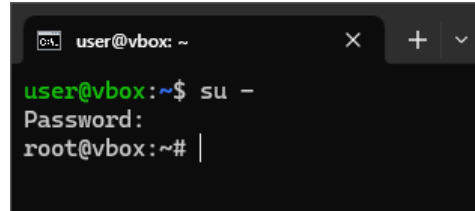


Рисунок 26

Для установки версии с помощью apt потребуется выполнить несколько команд: добавить ключ подписи пакета Microsoft в список доверенных ключей и добавить репозиторий пакетов.

```
wget
https://packages.microsoft.com/config/debian/12/packages-
microsoft-prod.deb -O packages-microsoft-prod.deb

dpkg -i packages-microsoft-prod.deb

rm packages-microsoft-prod.deb
```

Рисунок 27

Выполним установку среды выполнения ядра ASP.NET 8, используя следующую команду.

```
apt update && apt install -y aspnetcore-runtime-8.0
```

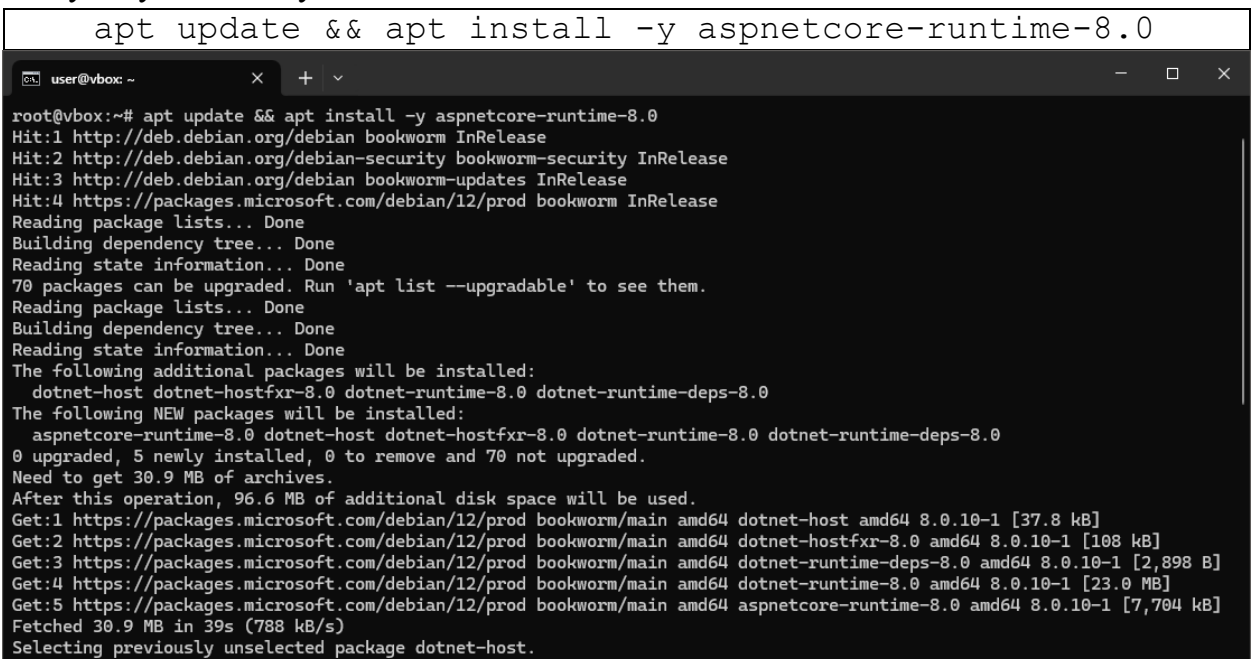


Рисунок 28

Чтобы убедиться, что ASP.NET 8 установлен, выполним следующую команду, которая отобразит информацию об установленном пакете.

```
dotnet --info

user@vbox: ~
root@vbox:~# dotnet --info

Host:
  Version:      8.0.10
  Architecture: x64
  Commit:       81cabf2857
  RID:          linux-x64

.NET SDKs installed:
  No SDKs were found.

.NET runtimes installed:
  Microsoft.AspNetCore.App 8.0.10 [/usr/share/dotnet/shared/Microsoft.AspNetCore.App]
  Microsoft.NETCore.App 8.0.10 [/usr/share/dotnet/shared/Microsoft.NETCore.App]

Other architectures found:
  None

Environment variables:
  Not set

global.json file:
  Not found

Learn more:
  https://aka.ms/dotnet/info
```

Рисунок 29

ASP.NET 8 установлен.

## 10. Установка web-сервера Nginx в Debian 12

Для входа под учетной записью суперпользователя используем команду и вводим пароль.

```
su
```

Рисунок 30

Скачиваем и добавляем ключ Nginx Inc. в нашу систему.

```
wget --quiet -O -  
http://nginx.org/keys/nginx_signing.key | apt-key add -
```

Рисунок 31

Устанавливаем Nginx из ветки Stable.

```
echo "deb http://nginx.org/packages/debian/  
$(lsb_release -sc)  
nginx">/etc/apt/sources.list.d/nginx.list  
  
echo "deb-src http://nginx.org/packages/debian/  
$(lsb_release -sc)  
nginx">>/etc/apt/sources.list.d/nginx.list
```

Рисунок 32

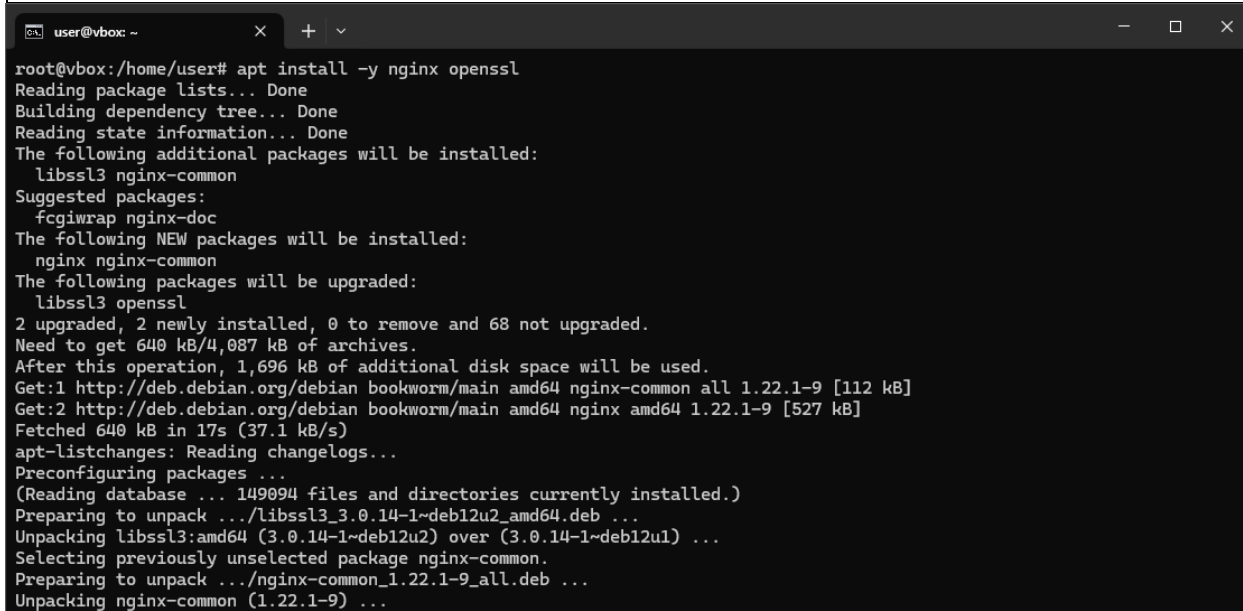
Обновляем список пакетов.

```
apt update
```

Рисунок 33

Устанавливаем Nginx и OpenSSL.

```
apt install -y nginx openssl
```



```
user@vbox: ~  
root@vbox:/home/user# apt install -y nginx openssl  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  libssl3 nginx-common  
Suggested packages:  
  fcgiwrap nginx-doc  
The following NEW packages will be installed:  
  nginx nginx-common  
The following packages will be upgraded:  
  libssl3 openssl  
2 upgraded, 2 newly installed, 0 to remove and 68 not upgraded.  
Need to get 640 kB/4,087 kB of archives.  
After this operation, 1,696 kB of additional disk space will be used.  
Get:1 http://deb.debian.org/debian bookworm/main amd64 nginx-common all 1.22.1-9 [112 kB]  
Get:2 http://deb.debian.org/debian bookworm/main amd64 nginx amd64 1.22.1-9 [527 kB]  
Fetched 640 kB in 17s (37.1 kB/s)  
apt-listchanges: Reading changeLogs...  
Preconfiguring packages ...  
(Reading database ... 149094 files and directories currently installed.)  
Preparing to unpack ../libssl3_3.0.14-1~deb12u2_amd64.deb ...  
Unpacking libssl3:amd64 (3.0.14-1~deb12u2) over (3.0.14-1~deb12u1) ...  
Selecting previously unselected package nginx-common.  
Preparing to unpack ../nginx-common_1.22.1-9_all.deb ...  
Unpacking nginx-common (1.22.1-9) ...
```

Рисунок 34

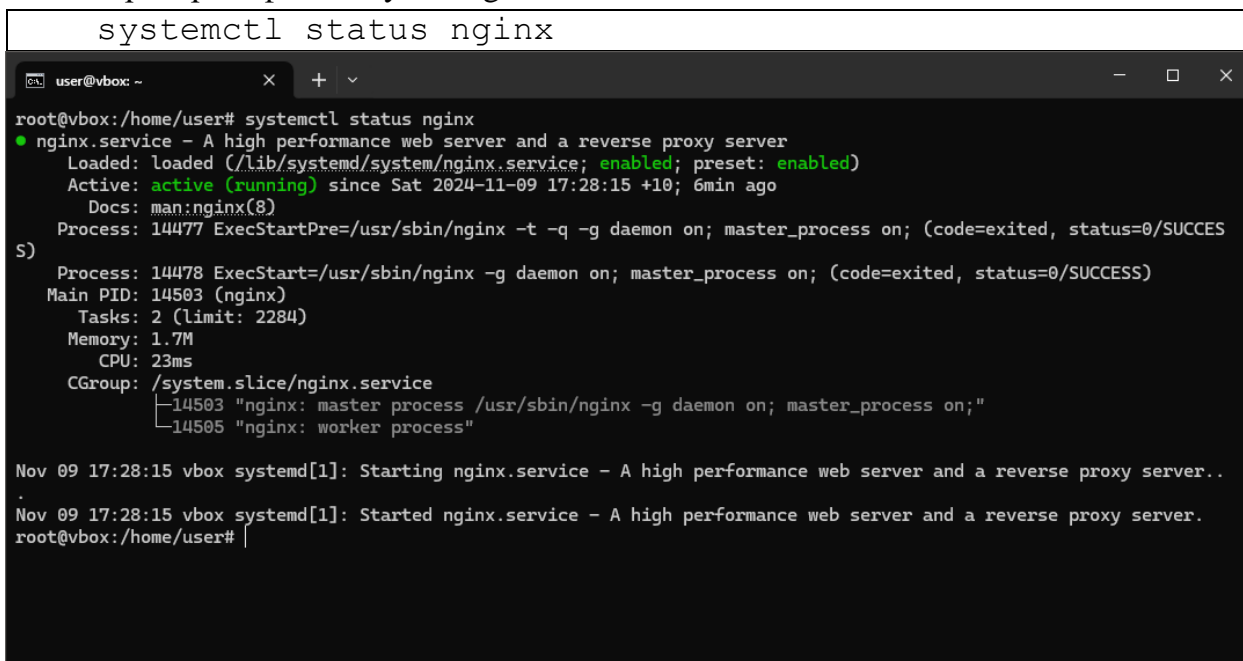
После установки запускаем Nginx.

```
systemctl start nginx
```

Рисунок 35

Проверим факт запуска Nginx.

```
systemctl status nginx
```



```
root@vbox:/home/user# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Sat 2024-11-09 17:28:15 +10; 6min ago
     Docs: man:nginx(8)
   Process: 14477 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 14478 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 14503 (nginx)
    Tasks: 2 (limit: 2284)
   Memory: 1.7M
      CPU: 23ms
   CGroup: /system.slice/nginx.service
           └─14503 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─14505 "nginx: worker process"

Nov 09 17:28:15 vbox systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server..
Nov 09 17:28:15 vbox systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
root@vbox:/home/user#
```

Рисунок 36

Nginx запущен.

Проверим открытые порты.

```
ss -ltupn | grep nginx
```



```
root@vbox:/home/user# ss -ltupn | grep nginx
tcp LISTEN 0      511      0.0.0.0:80      0.0.0.0:*      users:(("nginx",pid=14505,fd=5),("nginx",pid=14503,fd=5))
tcp LISTEN 0      511      ::::80          ::::*          users:(("nginx",pid=14505,fd=6),("nginx",pid=14503,fd=6))
root@vbox:/home/user#
```

Рисунок 37

Видим, что Nginx ожидает соединения на 80 порту на всех сетевых интерфейсах.



## 11. Базовая настройка web-сервера Nginx в Debian 12

Создадим директорию для хранения SSL сертификатов и DH-ключей, а также создаем файл с параметрами для DHE-шифров.

```
mkdir /etc/nginx/ssl

openssl dhparam -out /etc/nginx/ssl/dhparams.pem 2048
```

Рисунок 38

Скачаем готовый файл настроек приложения Nginx.

```
wget
https://gist.githubusercontent.com/CHERTS/8e9ecf4fbfb76555
6311a88e5106174b/raw/nginx.conf -O /etc/nginx/nginx.conf
```

Рисунок 39

Создадим файл с конфигурацией для подключения к нашему приложению.

```
echo 'server {
    listen 5050;
    server_name _;

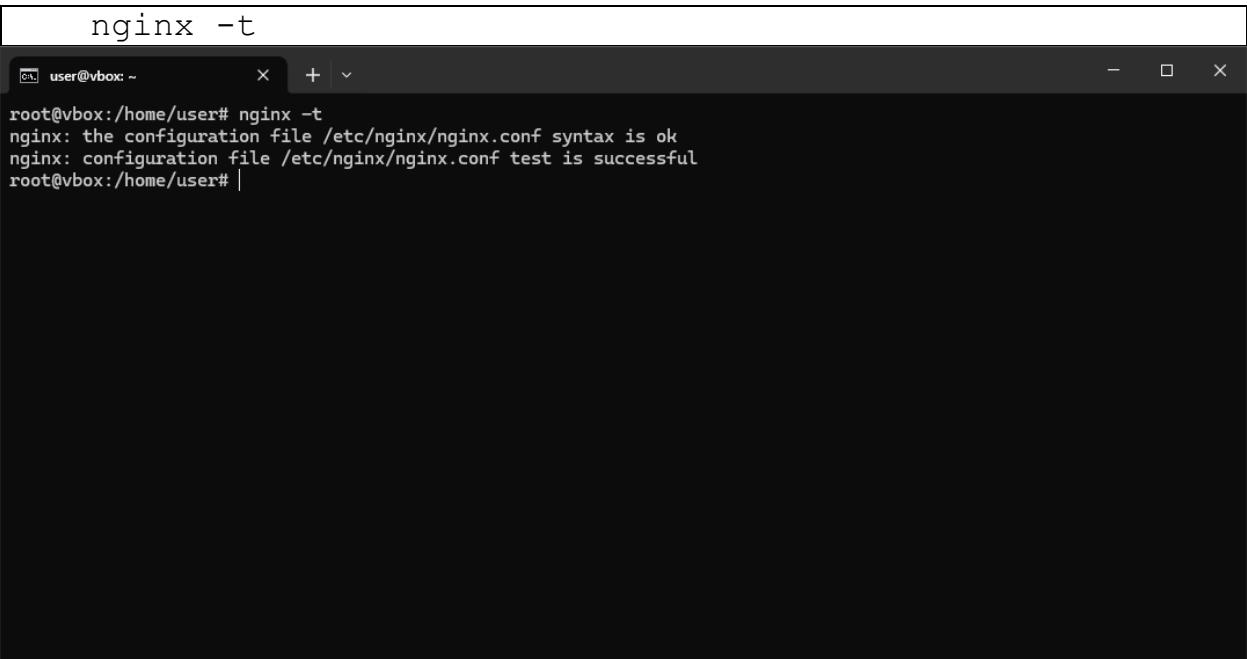
    location / {
        proxy_pass http://localhost:5000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}' >> /etc/nginx/conf.d/LightingDispatcher.conf
```

Рисунок 40

Запускаем команду **echo \$PATH** и проверяем содержится ли в путях /usr/local/sbin. Если нет, тогда вручную добавим путь: **PATН=/usr/sbin:\$PATH**.

После этого проверяем конфигурацию Nginx.

```
nginx -t
```



```
root@vbox:/home/user# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@vbox:/home/user# |
```

*Рисунок 41*

Ошибок нет, теперь перезагрузим Nginx.

```
nginx -s reload
```

*Рисунок 42*

## 12. Настройка и запуск серверного приложения «LIGHTING»

Создадим пользователя dispatcher, под которым будем запускать приложение.

```
sudo adduser dispatcher
```

Рисунок 43

Переключимся на пользователя dispatcher.

```
su dispatcher
```

Рисунок 44

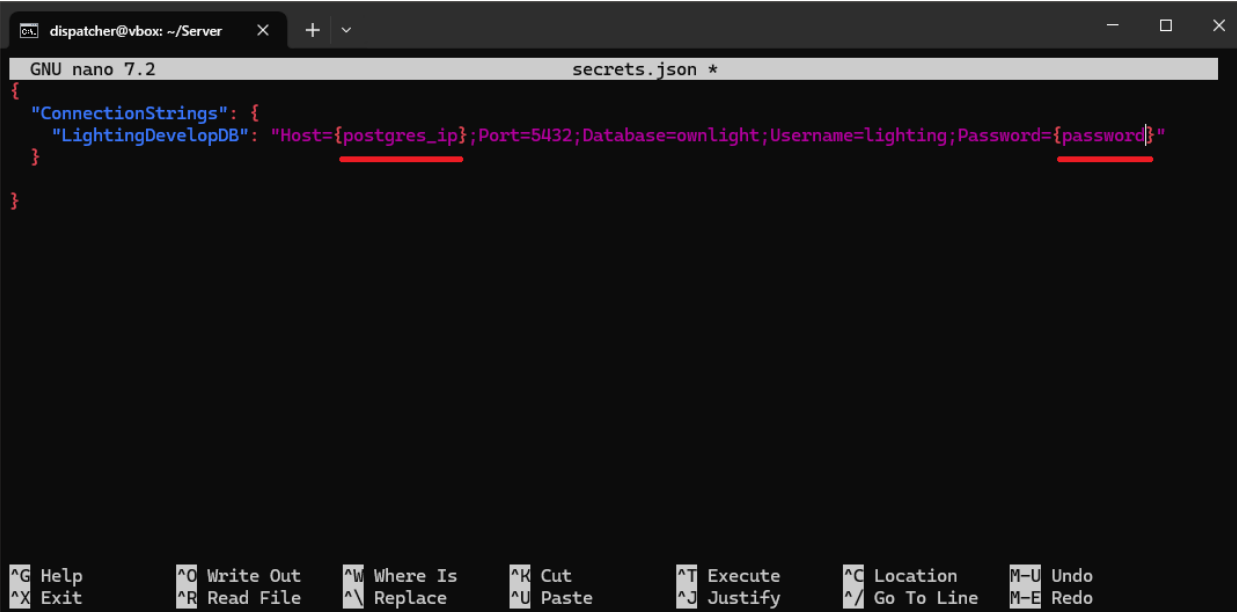
Создадим каталог для приложения.

```
mkdir /home/dispatcher/Server
```

Рисунок 45

Переносим файлы из полученного вами архива в каталог /home/dispatcher/Server.

Замените содержимое файла secrets.json ({postgres\_ip} и {password}) на ваши установленные значения.



```
dispatcher@vbox: ~/Server
GNU nano 7.2 secrets.json *
{
  "ConnectionStrings": {
    "LightingDevelopDB": "Host={postgres_ip};Port=5432;Database=ownlight;Username=lighting;Password={password};"
  }
}
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo
```

Рисунок 46

Запустим приложение.

```
cd /home/dispatcher/Server
```

```
dotnet LightingDispatcher.Server.dll
```

Рисунок 47

Теперь по адресу <http://IP-адрес-вашего-сервера:5050> Вы сможете увидеть запущенное приложение.