# Руководство по настройке PostgreSQL для работы через SSL

**Оглавление:**

1. [Заведение OpenSSL сертификатов для работы через SSL](#Заведение_сертификатов).
2. [Настройка PostgreSQL 15 на Linux для работы через SSL](#Настройка_Postgres_SSL_Linux).
3. [Настройка PostgresPro 11 на Astra Linux для работы через SSL](#Настройка_PostgresPro_SSL_Astra_Linux).
4. [Настройка PostgreSQL 15 на Windows для работы через SSL](#Настройка_PostgreSQL_SSL_Windows).
5. [Проверка работоспособности связки PostgreSQL с SSL](#Проверка_работоспособности).
6. [Настройка WebApi для работы с PostgreSQL через SSL.](#Настройка_WebApi)
7. [Полезные ссылки](#Полезные_ссылки).
8. **Заведение OpenSSL сертификатов для работы через SSL.**
   1. Устанавливаем **OpenSSL** либо с официального сайта (<https://www.openssl.org/>), либо используем ту версию, что идет в составе одной из утилит (например можно использовать ту, которая идет в составе **git**).
   2. Для того, чтобы в скриптах каждый раз не использовать полный путь к файлу **openssl.exe**, рекомендуется добавить этот путь в настройки системной переменной **Path** (если же Вы предпочитаете работать через **PowerShell**, а не через командную строку, и не желаете изменять системную переменную **Path**, то можно использовать сеансовые алиасы).
   3. Далее в данном руководстве предполагается, что полный путь прописан в системной переменной **Path**.
   4. Прописать путь в **Path**, если Вы работаете в **Windows** можно таким образом:

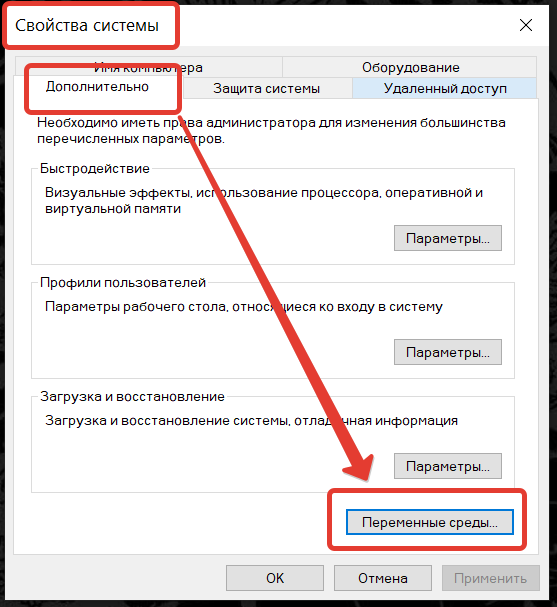
Открываем указанное ниже диалоговое окно (и далее выбираем соответствующие вкладки / нажимаем указанные кнопки / выбираем указанные строки из списков):

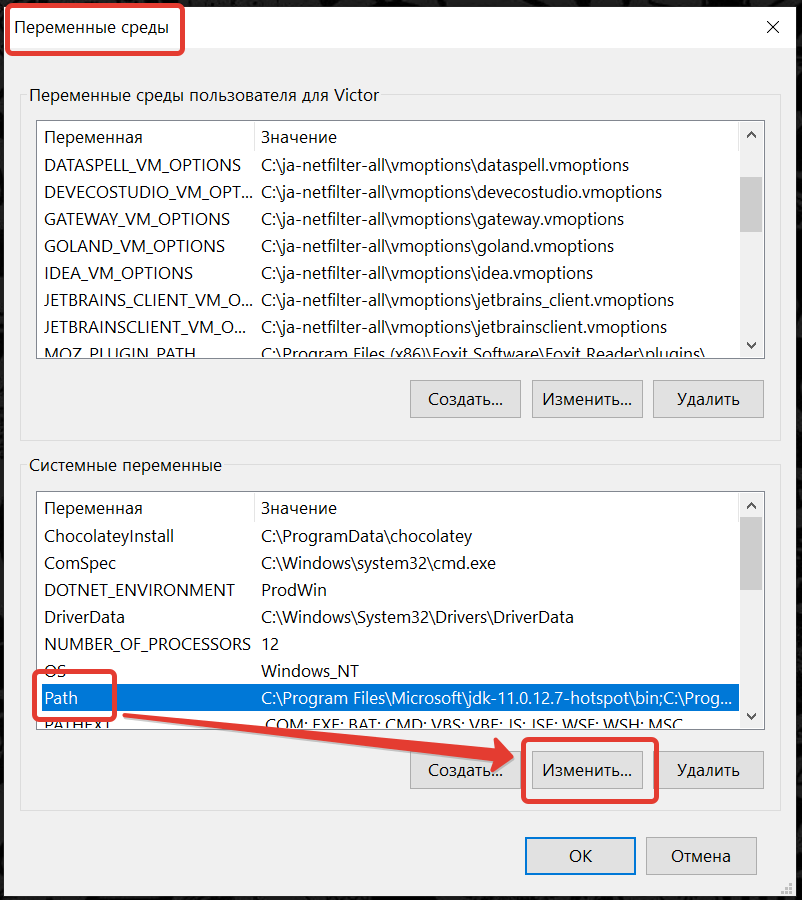
- В нерусифицированной версии Windows:

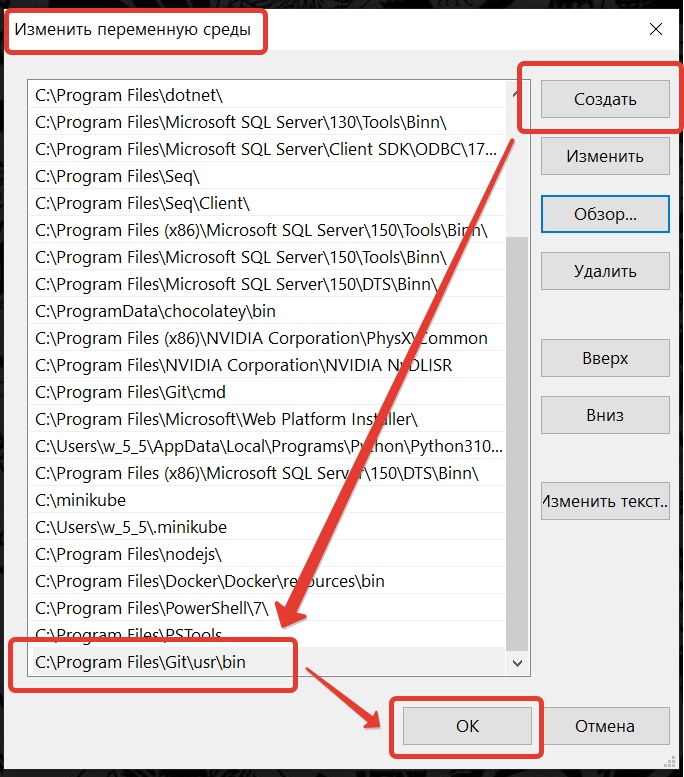
***System Properties* > Advanced > *Environment Variables…* > *System variables* > *Path > Edit… > New > [Тут указываем полный путь к файлу openssl.exe]***:

- В русской версии Windows:

***Свойства системы* > *Дополнительно* > *Переменные среды…* > *Системные переменные* > *Path > Изменить… > Создать > [Тут указываем полный путь к файлу openssl.exe]***:







* 1. Далее создаем каталог, в котором будут размещаться как сами сертификаты, так и необходимые для их создания файлы и переходим в него.
  2. Первым делом создаем **корневой ключ** для CA и затем создаем **корневой сертификат CA**.

|  |
| --- |
| **openssl genrsa -out rootCA.key 2048**  **openssl req -x509 -new -key rootCA.key -days 10000 -out rootCA.crt** |

Открываем командную строку и выполняем в ней такие 2 команды:

Заполняем необходимые поля для сертификата. Имена выбираем произвольно. На этом **корневой сертификат CA** создан.

|  |
| --- |
| **openssl genrsa -out server.key 2048** |

* 1. Генерируем **ключ для сервера**:

|  |
| --- |
| **openssl req -new -key server.key -out server.csr** |

* 1. Теперь создаем запрос на **сертификат сервера**.

При заполнении полей, в поле **Common Name** важно указать имя сервера: **домен** или **IP** адрес **сервера** (например домен: **db.example.com** или IP-адрес: **192.168.0.116**):

* 1. Подписываем запрос на сертификат нашим корневым сертификатом:

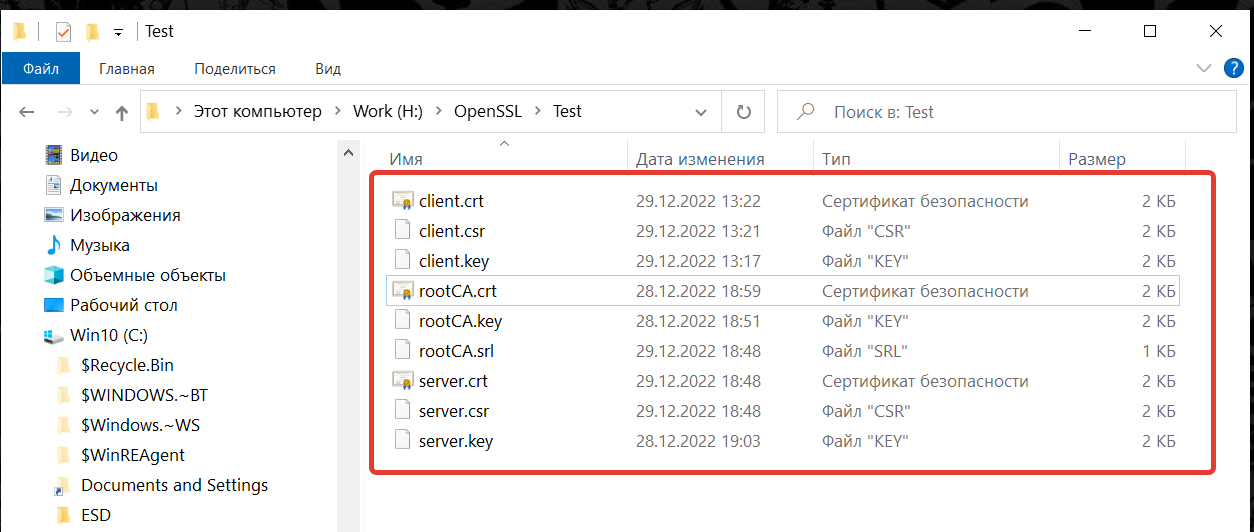
|  |
| --- |
| **openssl x509 -req -in server.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out server.crt -days 5000** |

* 1. Теперь таким же образом (как описано в трех предыдущих пунктах) создаем связку ключ-сертификат для клиента. Имена файлов соответственно меняем (например: **client.key, client.crt, client.csr**) :

|  |
| --- |
| **openssl genrsa -out client.key 2048**  **openssl req -new -key client.key -out client.csr**  **openssl x509 -req -in client.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out client.crt -days 5000** |

При заполнении полей сертификата клиента, в поле **Common Name** указываем **логин** под которым будет выполняться подключение к БД (у меня это имя: **postgres** – далее во всех скриптах в этом руководстве подразумевается, что было задано именно такое имя для этого параметра).

* 1. В итоге в нашей рабочей папке должно быть 9 таких файлов:



* 1. Теперь, когда с ключами покончено, переходим к настройке PostgreSQL, как описано далее в настоящем руководстве.

1. **Настройка PostgreSQL 15 на Linux для работы через SSL.**
   1. Копируем **корневой сертификат CA**, **ключ** и **сертификат сервера** (созданные ранее в п.1. настоящего руководства) в **каталог БД** (допустим он расположен по такому пути: **/var/lib/postgresql/data**, при этом мы сейчас находимся в папке с сертификатами, созданными ранее в п.1 ). В командной строке выполняем такие 3 команды:

|  |
| --- |
| **cp server.key /var/lib/postgresql/data**  **cp server.crt /var/lib/postgresql/data**  **cp rootCA.crt /var/lib/postgresql/data** |

*Пример*: Если БД развернута в докер-контейнере с именем **pg**, а мы сейчас находимся в папке с сертификатами, созданными ранее в п.1, то указанные выше команды будут выглядеть таким образом:

|  |
| --- |
| **docker cp server.key pg:/var/lib/postgresql/data**  **docker cp server.crt pg:/var/lib/postgresql/data**  **docker cp rootCA.crt pg:/var/lib/postgresql/data** |

* 1. Меняем **имя владельца** для указанных выше файлов на то имя от которого мы будем коннектится к серверу и которое мы ранее (в п.1.9.) указали при создании сертификата клиента в поле **Common Name** (у меня это имя: **postgres**):

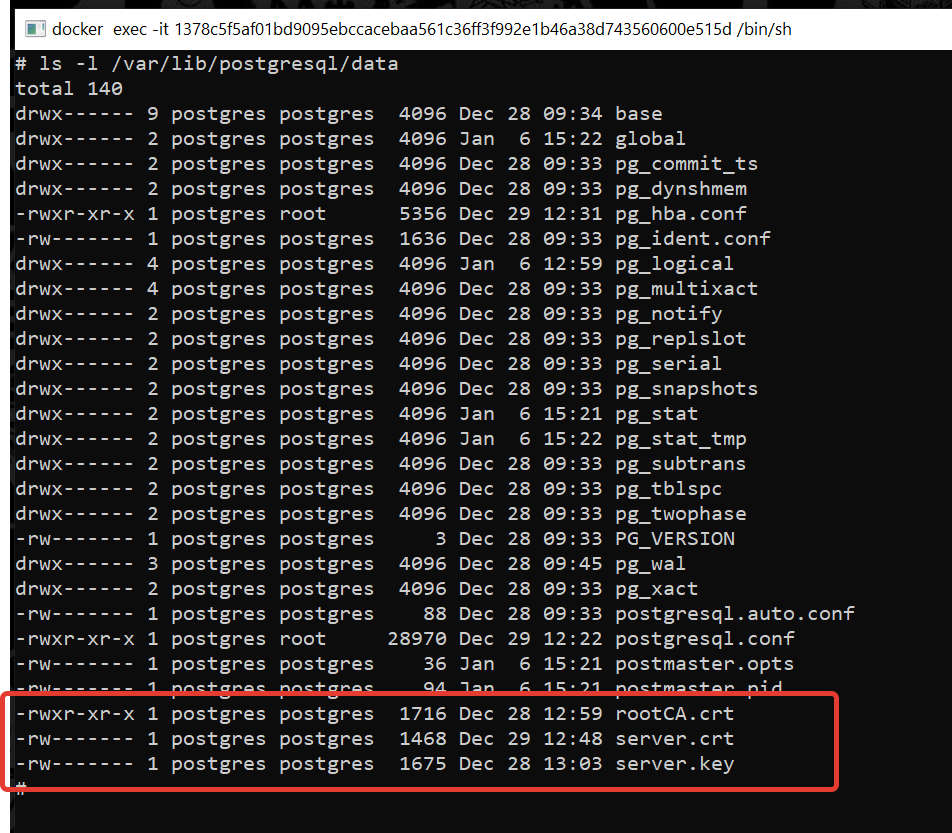
Для этого переходим в каталог БД и выполняем такую команду:

|  |
| --- |
| **chown postgres: server.crt server.key rootCA.crt** |

* 1. Изменяем **права доступа** для файлов **server.crt** и **server.key** (мы еще находимся в каталоге БД):

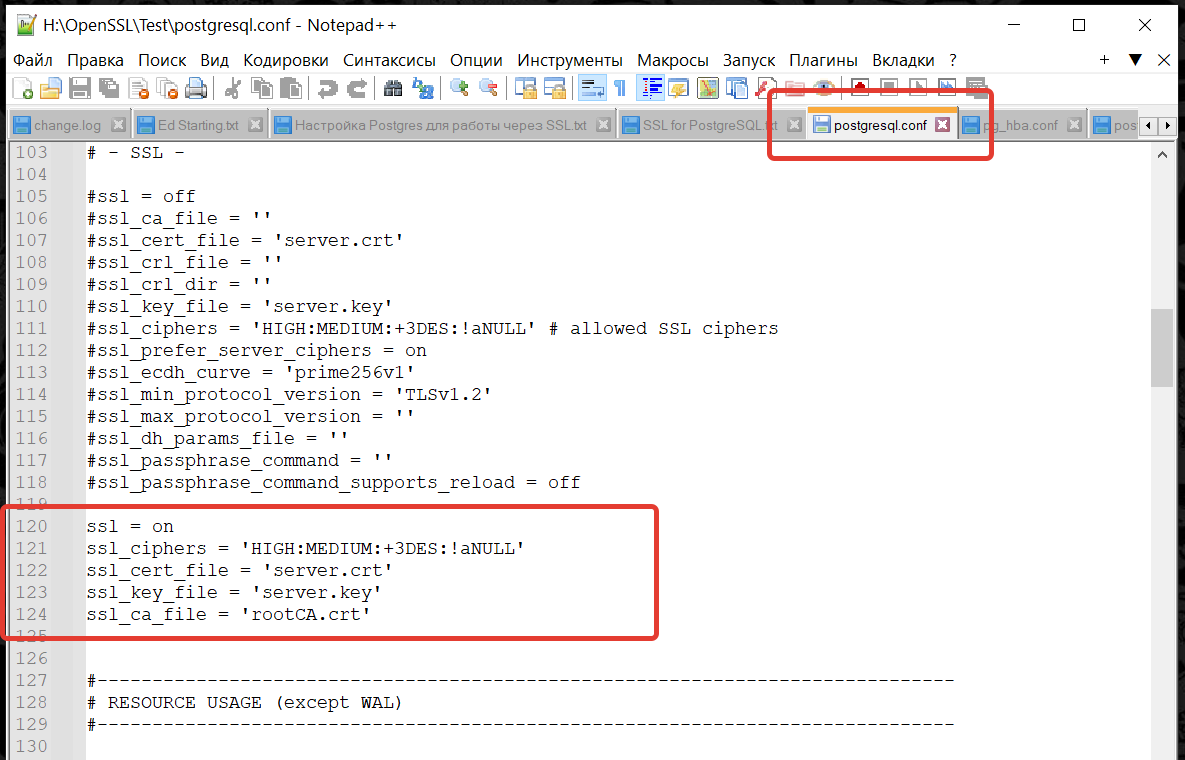
|  |
| --- |
| **chmod 600 server.crt**  **chmod 600 server.key** |

* 1. В итоге в каталоге БД должны быть установлены указанные выше файлы с такими правами:



* 1. Теперь переходим к правке конфигурации **PostgreSQL**. **PostgreSQL** должен быть собран с поддержкой **SSL**. Для этого необходимо внести изменения в файлы **postgresql.conf** и **pg\_hba.conf**.
  2. В файл **postgresql.conf** добавляем такие строки:

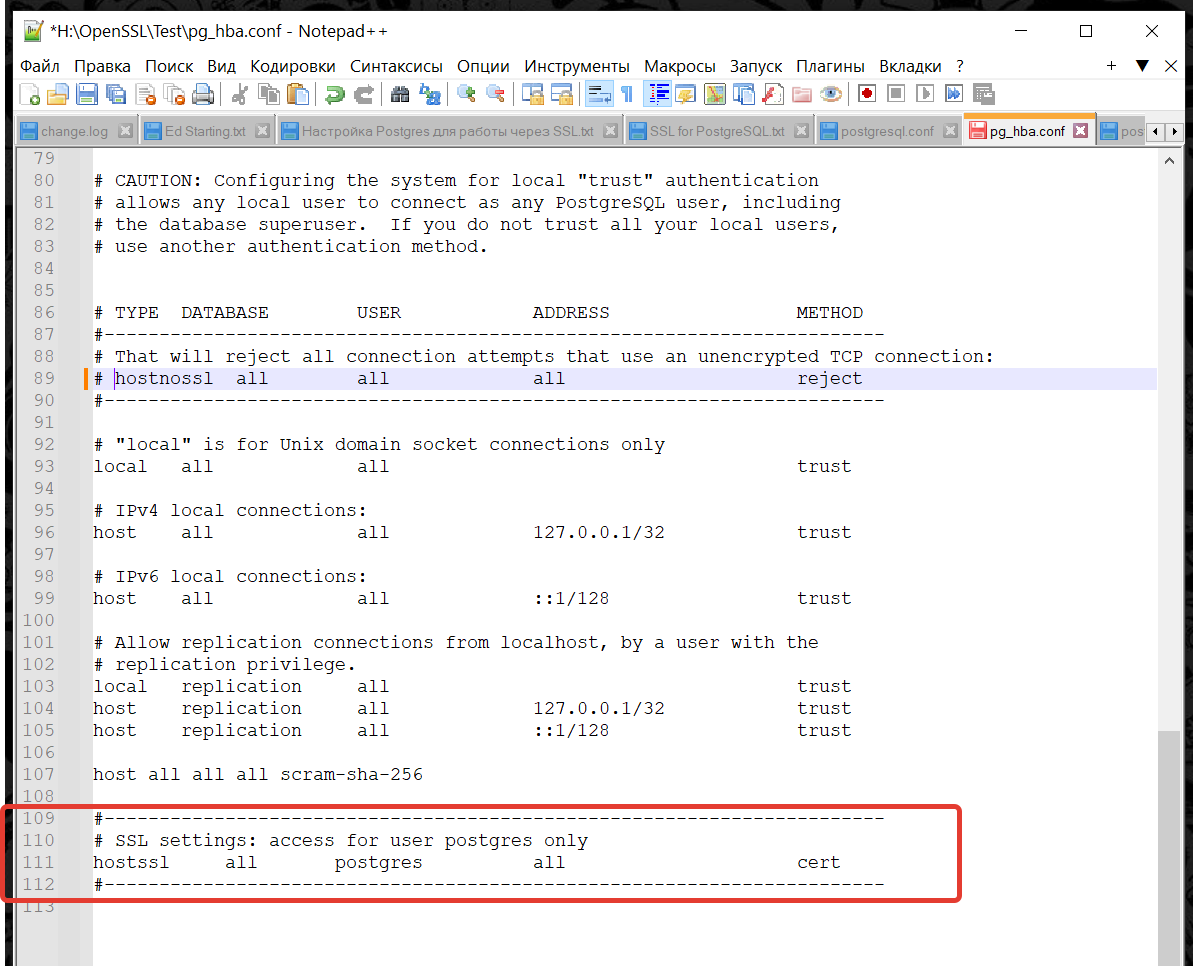
|  |
| --- |
| **ssl = on**  **ssl\_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'**  **ssl\_cert\_file = 'server.crt'**  **ssl\_key\_file = 'server.key'**  **ssl\_ca\_file = 'rootCA.crt'** |



* 1. В файл **pg\_hba.conf** добавляем такую строку:

|  |
| --- |
| **hostssl all postgres all cert** |

Тут предполагается, что мы ранее указали при создании сертификата клиента в поле **Common Name** имя: **postgres**.

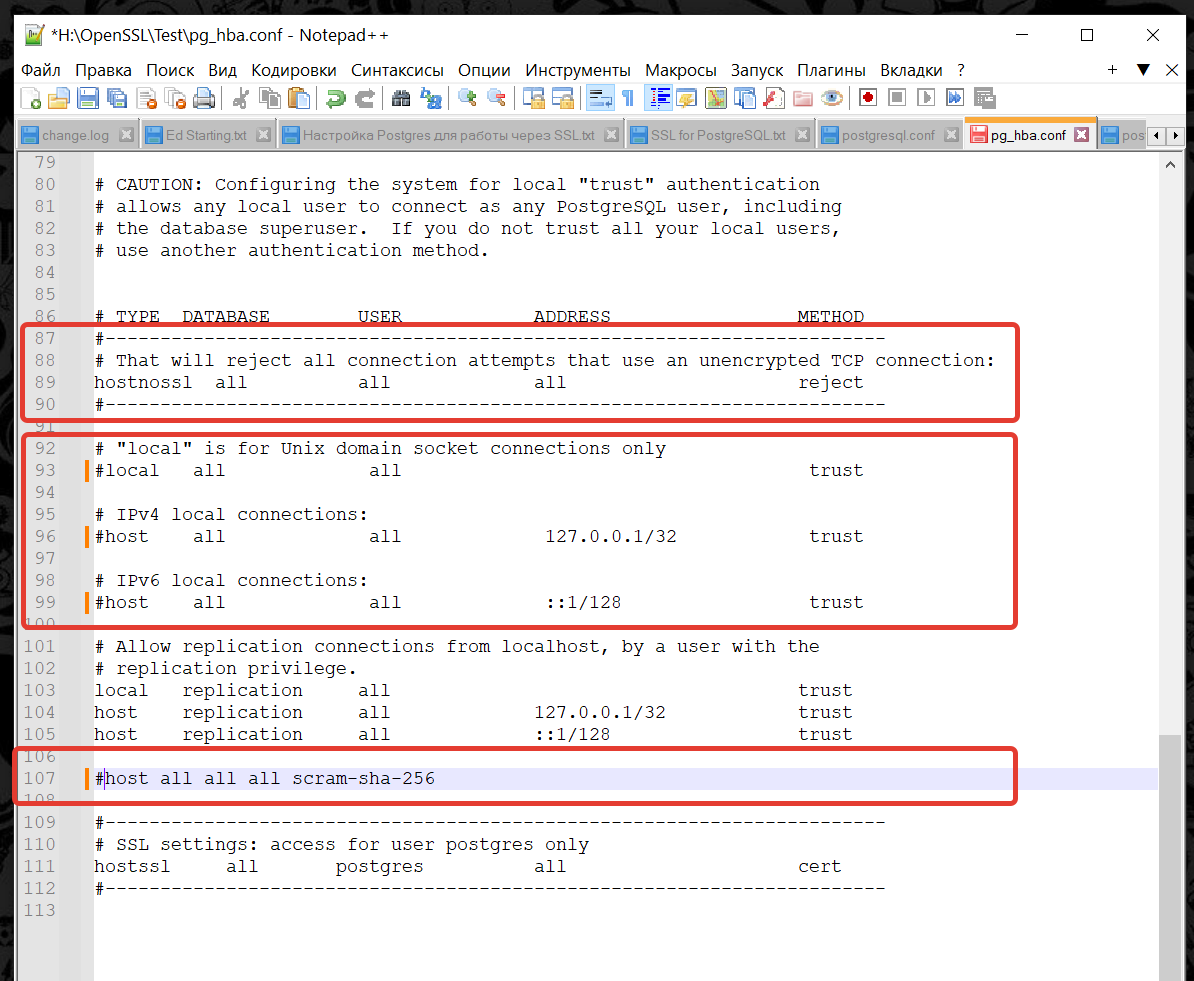


* 1. Если нам требуется, чтобы коннект с сервером БД происходил **только через** **SSL**, то в дополнение к тому, что было сделано в п.2.7., в файл **pg\_hba.conf** требуется добавить выше остальных конфигурационных строк такую строку:

|  |
| --- |
| **hostnossl all all all reject** |

и удалить (или закомментировать) строки, которые начинаются таким образом:

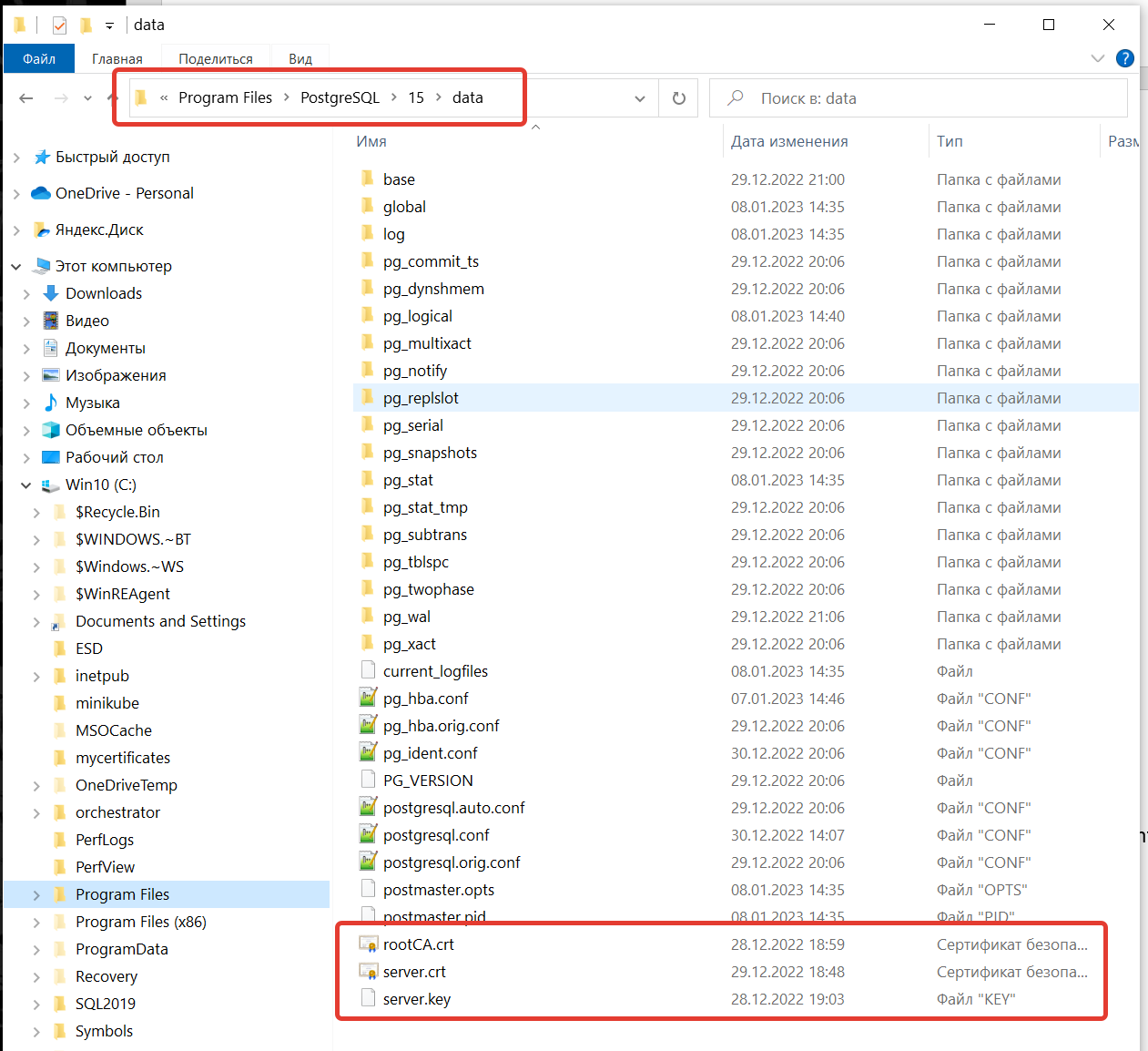
|  |
| --- |
| **local all**  **host all** |



* 1. После этого **перезапускаем сервер** и проверяем коннект к нему через **SSL** (например, как описано в п.4. далее).

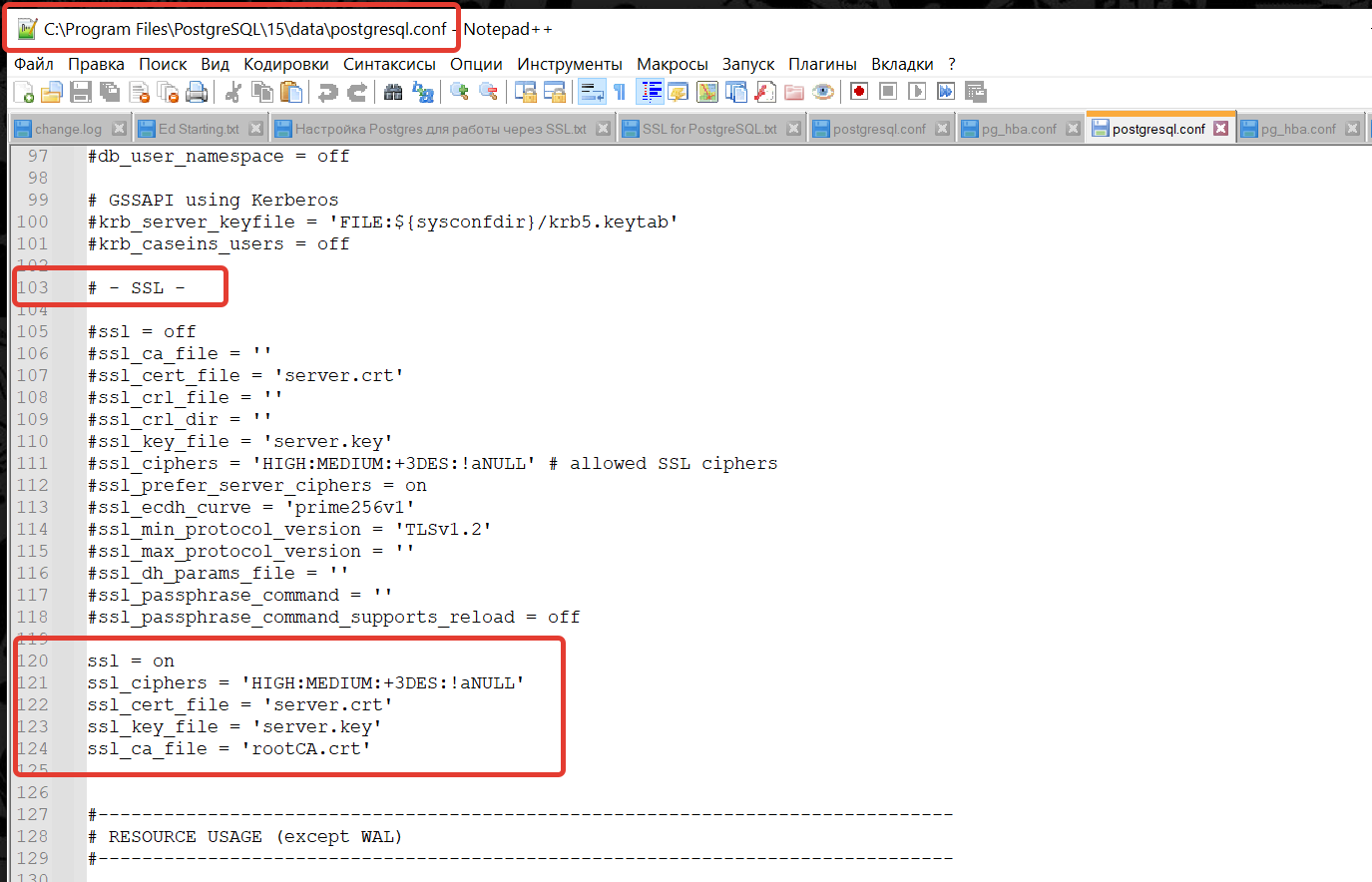
1. **Настройка PostgrePro 11 на Astra Linux 1.7 для работы через SSL.**
   1. Настройка аналогична таковой для **PostgreSQL 15 на Linux** (см. [п.2. настоящего руководства](#Настройка_Postgres_SSL_Linux)), за исключением того, что теперь все серверные конфиги и сертификаты будут расположены по такому пути: **/var/lib/pgpro/std-11/data**.
2. **Настройка PostgreSQL 15 на Windows для работы через SSL.**
   1. Копируем **корневой сертификат CA**, **ключ** и **сертификат сервера** (созданные ранее в п.1. настоящего руководства) в каталог БД (для **PostgreSQL** **15** он по умолчанию расположен по такому пути: **"C:\Program Files\PostgreSQL\15\data"**).

Т.е. надо в этот каталог добавить файлы: **server.key, server.crt, rootCA.crt:**

****

* 1. Теперь переходим к правке конфигурации **PostgreSQL**. **PostgreSQL** должен быть собран с поддержкой **SSL**. Для этого необходимо внести изменения в файлы **postgresql.conf** и **pg\_hba.conf**.
  2. В файл **postgresql.conf** добавляем такие строки:

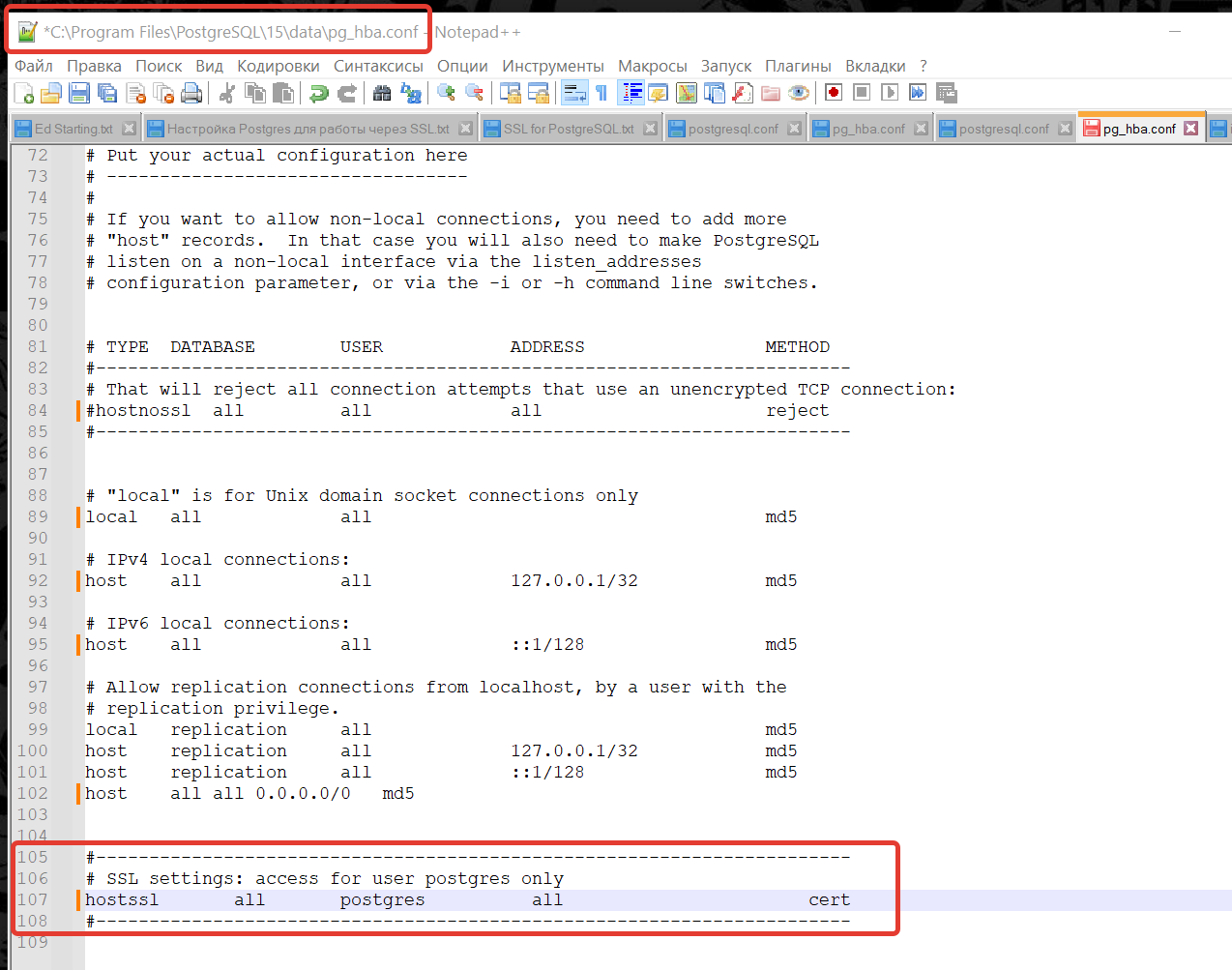
|  |
| --- |
| **ssl = on**  **ssl\_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'**  **ssl\_cert\_file = 'server.crt'**  **ssl\_key\_file = 'server.key'**  **ssl\_ca\_file = 'rootCA.crt'** |



* 1. В файл **pg\_hba.conf** добавляем такую строку:

|  |
| --- |
| **hostssl all postgres all cert** |

Тут предполагается, что мы ранее указали при создании сертификата клиента в поле **Common Name** имя: **postgres**.

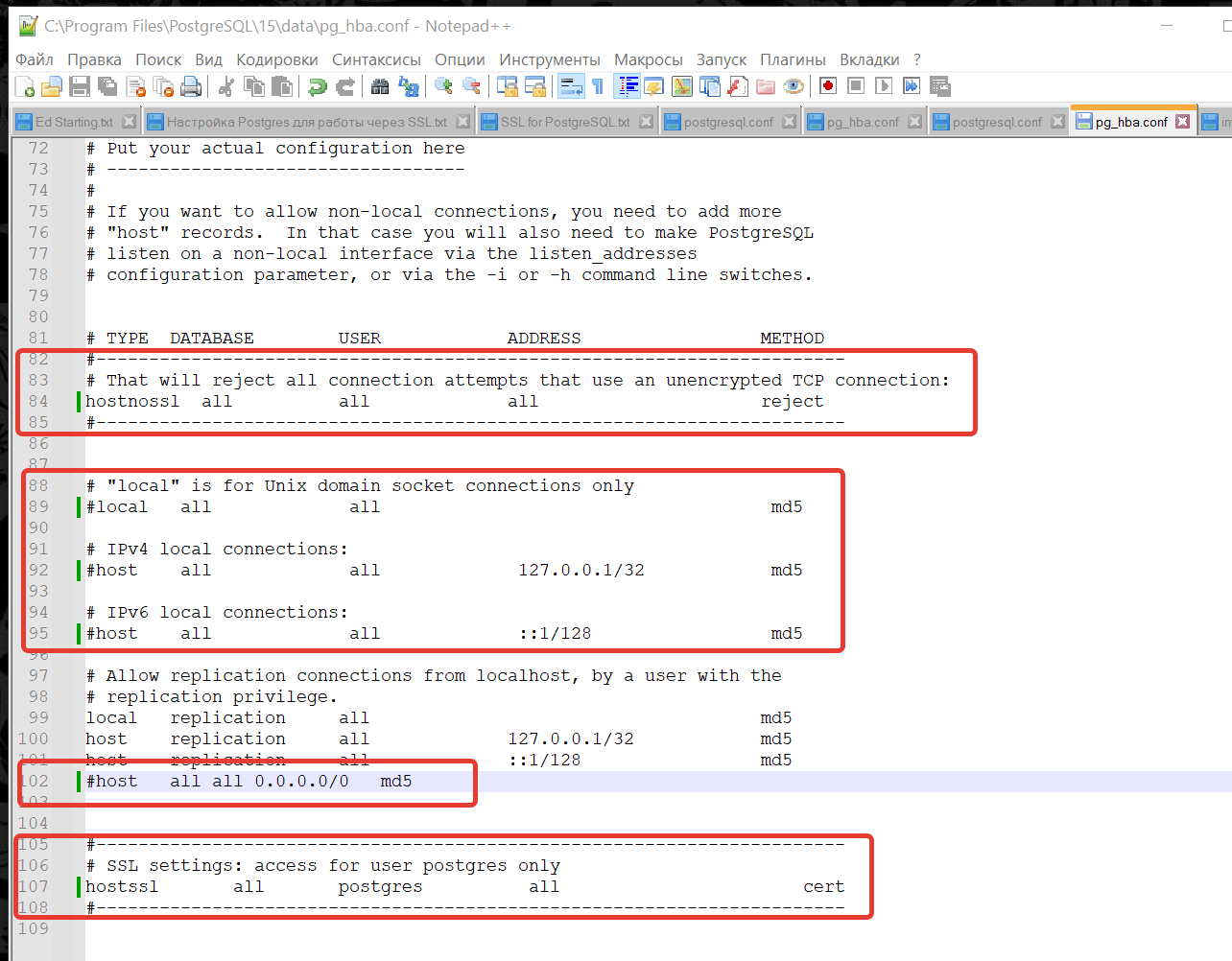


* 1. Если нам требуется, чтобы коннект с сервером БД происходил **только через** **SSL**, то в дополнение к тому, что было сделано в п.3.4., в файл **pg\_hba.conf** требуется добавить выше остальных конфигурационных строк такую строку:

|  |
| --- |
| **hostnossl all all all reject** |

и удалить (или закомментировать) строки, которые начинаются таким образом:

|  |
| --- |
| **local all**  **host all** |

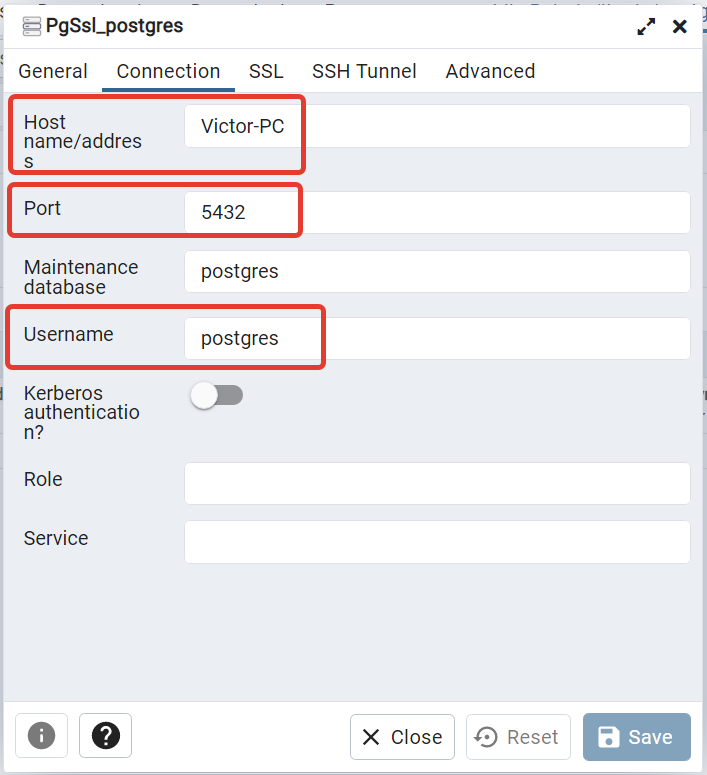


* 1. После этого **перезапускаем сервер** и проверяем коннект к нему через **SSL** (например, как описано в п.4. далее).

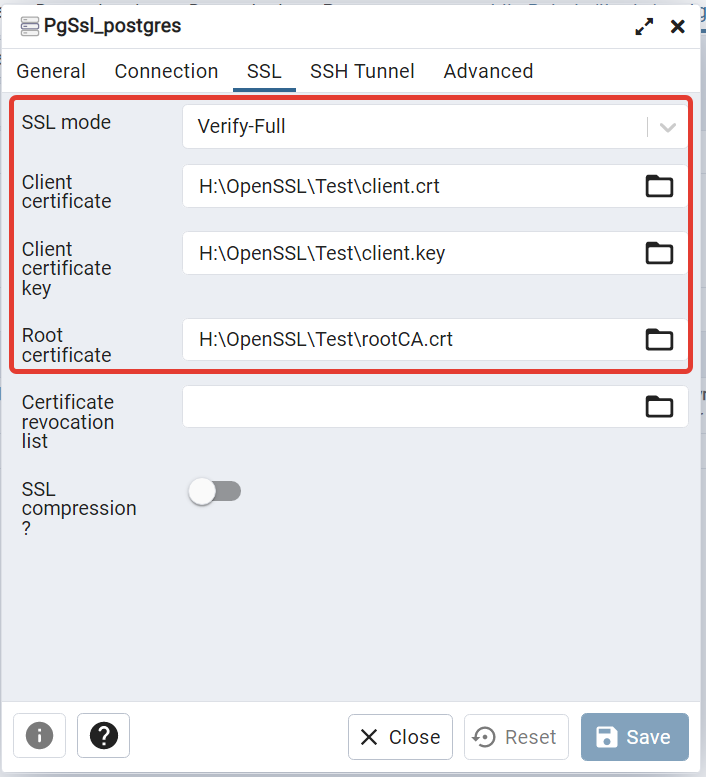
1. **Проверка работоспособность связки PostgreSQL с SSL.**

Для этого будем использовать утилиту **pgAdmin4**. Создаем в ней новый коннект с такими параметрами:

* 1. Вкладка **Connection**:
* В поле **Host name/address** вводим то имя **Common Name**, которое было для него указано при заведении серверного сертификата **server.crt**
* В поле **Port** указываем тот же самый порт, что используется для обычных **tcp**-соединений с сервером
* В поле **Username** вводим то имя **Common Name**, на которое был оформлен клиентский сертификат **client.crt** (в данном руководстве мы ранее задавали его как **postgres**)**:**

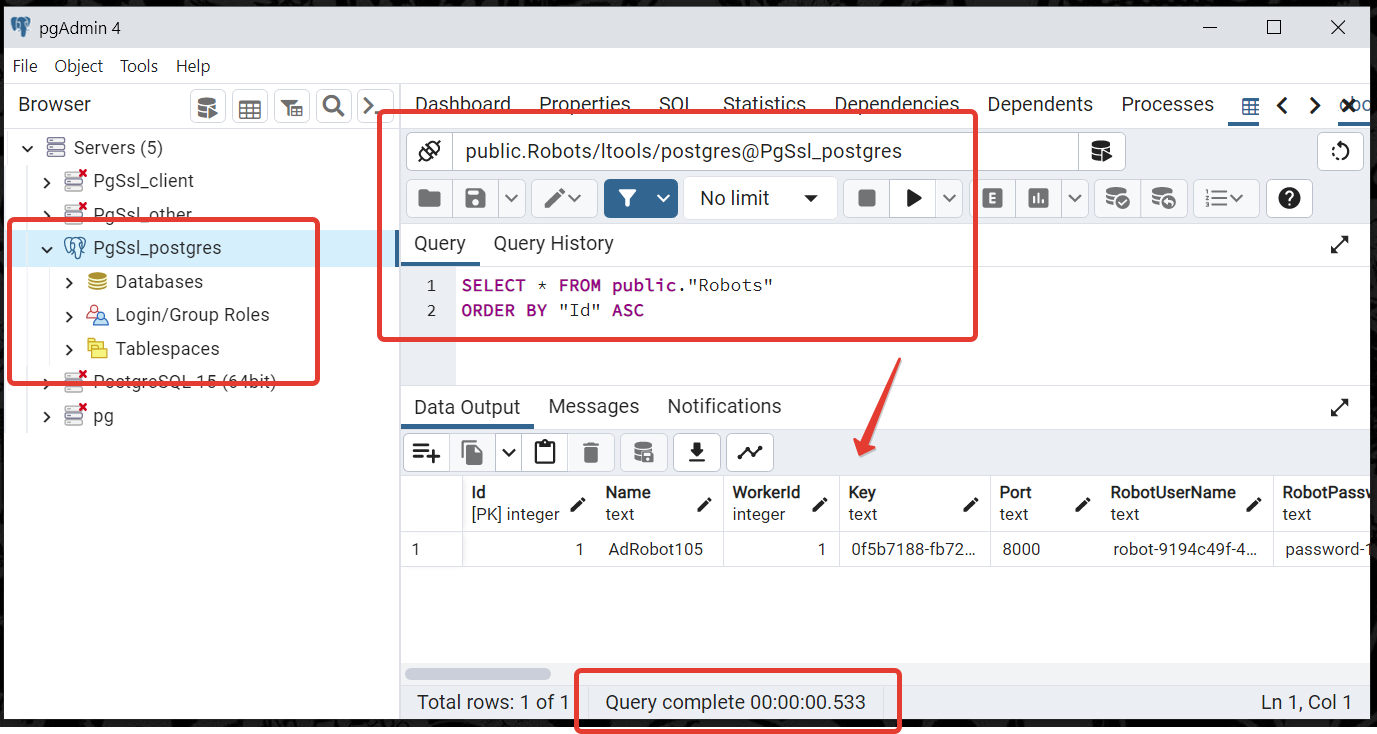


* 1. Вкладка **SSL**:
* В поле **SSL mode** выбираем из выпадающего списка **Verify-Full**
* В поле **Client certificate** указываем полный путь к папке с сертификатом клиента (т.е. к файлу **client.crt**) и имя этого сертификата
* В поле **Client certificate key** указываем полный путь к папке с ключом сертификата клиента (т.е. к файлу **client.key**) и имя этого ключа сертификата
* В поле **Root certificate** указываем полный путь к папке с сертификатом CA (т.е. к файлу **rootCA.crt**) и имя этого сертификата:



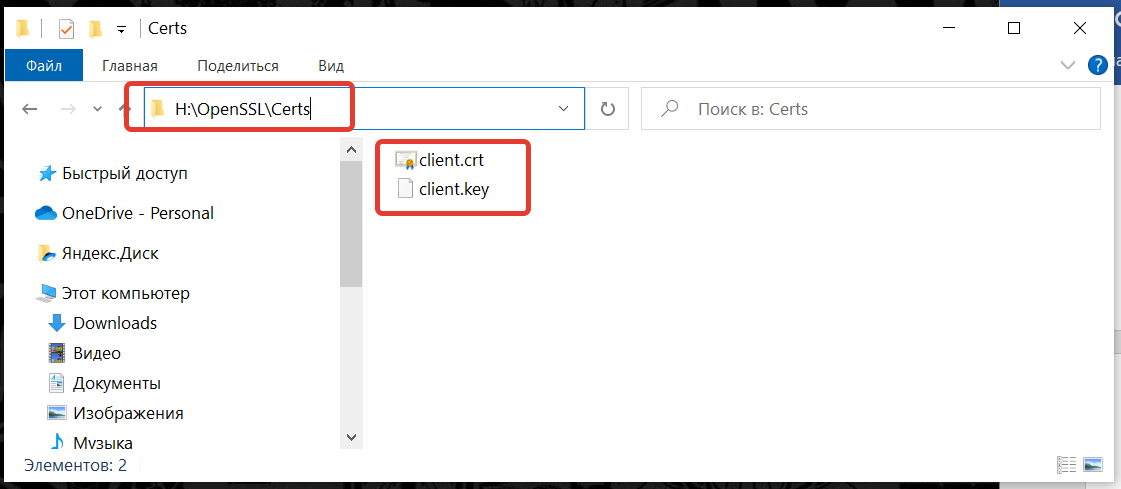
* 1. Далее **сохраняем** настройки этого коннекта.
  2. Далее дважды кликаем по имени только что созданного коннекта в дереве **Servers** слева. При условии корректного заполнения указанных выше полей и соответствия введенных данных тем, что были указаны при заведении **SSL**-сертификатов и их ключей, мы должны без проблем установить **SSL**-коннект к серверу и выполнить любой допустимый **SQL**-запрос.

Например, как показано тут:

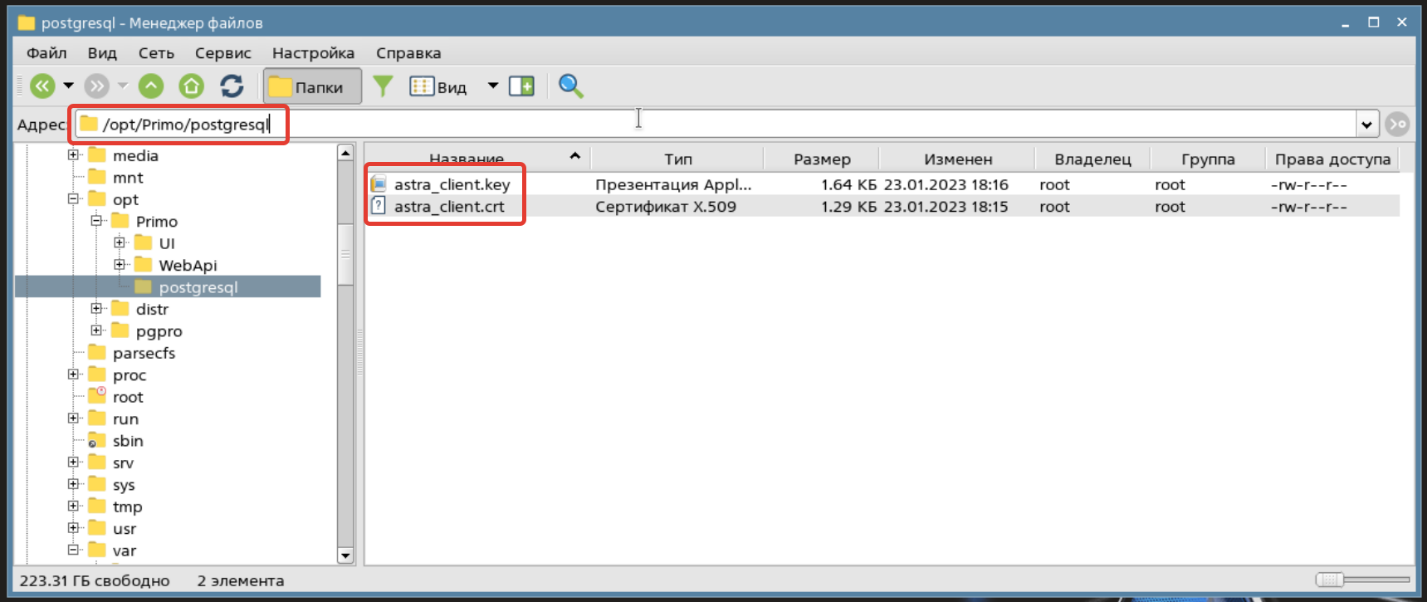


1. **Настройка WebApi для работы с PostgreSQL через SSL.**
   1. Копируем на машину с **WebAPI** в удобное для Вас место клиентский сертификат и его ключ (файлы **client.crt** и **client.key**).

Например, для машин с ОС **Windows** это может выглядеть так:

****

Или для машин под ОС из семейства **Linux:**

****

* 1. Далее меняем конфигурационный файл **WebApi** (файл **appsettings.ProdWin.json** для машин семейства **Windows**, или **appsettings.ProdLinux.json** для машин семейства **Linux**):
     1. В секции **PostgresWithSslOptions** для параметра **ClientCertPath у**казываем полный путь и имя клиентского сертификата.
     2. В той же секции для параметра **ClientCertKeyPath у**казываем полный путь и имя ключа клиентского сертификата.
     3. В той же секции для параметра **UseSsl** **у**казываем значение **true**.
     4. В секции **ConnectionStrings** для каждой из срок подключения добавляем в ее конец такую подстроку: “**SSLMode=Require;TrustServerCertificate=true;**” .

Например, эти секции конфига для машин с ОС **Windows** могут выглядеть так:



Или для машин под ОС из семейства **Linux:**

****

* 1. После сохранения изменений в конфиге перезапускаем сервис **WebApi**.

1. **Полезные ссылки:**
   1. Сайт OpenSSL: <https://www.openssl.org/>
   2. Документация по настройкам PostgresPro с SSL (на русском языке): <https://postgrespro.ru/docs/postgresql/9.5/ssl-tcp>
   3. Документация по настройкам PostgreSQL с SSL (на английском языке): <https://www.postgresql.org/docs/current/ssl-tcp.html>
   4. Описание параметров postgresql.conf (на английском языке): <https://github.com/CrunchyData/os-pg-testing/blob/master/conf/postgresql.conf>
   5. Описание параметров pg\_hba.conf (на английском языке): <https://github.com/CrunchyData/os-pg-testing/blob/master/conf/pg_hba.conf>
   6. Руководство по настройка связки PostgreSQL для работы через SSL на Windows (на английском языке): <https://methoddev.com/blog/postgresql-windows-encrypted-connection-ssl>