

УТВЕРЖДЕН  
643.72410666.00067-07 97 01-ЛУ

ЗАЩИЩЕННАЯ СИСТЕМА УПРАВЛЕНИЯ  
БАЗАМИ ДАННЫХ «ЈАТОВА»

Руководство по безопасности.  
Часть 27.

643.72410666.00067-07 97 01-27

Листов 143

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

## АННОТАЦИЯ

Данный документ представляет собой руководство по установке и настройке целевых и служебных хостов, а также создание SSH и SSL-соединений между компонентами СУБД «Jatoba».

Руководство по установке содержит следующие разделы по настройке SSH и SSL соединений:

В Приложении 1 описана установка и работа службы JDS.PasDoctor.



Все примеры в данном документе приведены для СУБД «Jatoba» версии ядра .x, 6 для других версий все шаги выполняются аналогично, разница состоит в именах директорий.

Например, СУБД «Jatoba» версии 5.x по умолчанию устанавливается в директорию:

- ОС Windows – «C:\Program Files\GIS\Jatoba\5\bin»;
- ОС Linux – «/usr/jatoba-5/bin».

Степени важности примечаний, применяемые в документе:



**Важная информация** – указания, требующие особого внимания



**Дополнительная информация** – указания, позволяющие упростить работу с изделием



**Важная информация**

Для сертифицированной версии СУБД «Jatoba» поддерживается работа только на ОС, указанных в формуляре на поставку!

## СОДЕРЖАНИЕ

1. Этапы развертывания СУБД на хостах .....	6
1.1. Первый этап. Топология хостов СУБД .....	6
1.2. Второй этап. Установка СУБД на хостах .....	7
1.3. Третий этап. Установка JDS .....	7
1.4. Четвертый этап. Подготовка хостов. Настройка SSH соединений .....	7
1.5. Пятый этап. Подготовка SSL подключений .....	7
1.6. Шестой этап. Установка компонентов СУБД .....	7
2. Настройка SSH-соединений JDS .....	8
2.1. Создание пользователя ОС jdscontrol на целевом хосте .....	8
2.2. Копирование скрипта на целевой хост .....	9
2.3. Выполнение скрипта на целевом хосте .....	10
2.4. Создание домашнего каталога пользователя JDS на служебном хосте .....	11
2.5. Создание ключа SSH для пользователя JDS на служебном хосте .....	11
2.6. Передача ключа SSH на целевую СУБД .....	12
2.7. Подключение JDS к целевой СУБД (SSH и PASSWORD) .....	13
2.7.1. Хост .....	14
2.7.2. Настройка конфигурационного файла pg_hba.conf .....	15
2.7.3. СУБД .....	15
3. Настройка SSL-соединений .....	17
3.1. Создание и конвертация сертификатов .....	19
3.2. Пути хранения сертификатов и ключей компонентов СУБД .....	22
4. Подключение JDS к целевой СУБД по SSL .....	26
5. Подключение JDS к служебной СУБД по SSL .....	28
6. Подключение к Web-интерфейсу JDS по SSL .....	30
7. Настройка SSL компонента ja_Dog .....	31
7.1. Настройка SSL между узлами кластера .....	31
7.2. Настройка доступа ja_Dog к СУБД по SSL .....	33
7.3. Подключение кластера «ja_Dog» к JDS разделе «Ландшафт» .....	34
7.3.1. Настройка Rest API .....	35
7.3.2. Проверочные мероприятия .....	37
7.3.3. Добавление существующего кластера ja_Dog .....	39
8. Раздел JDS «Анализ запросов» .....	41
8.1. Настройка на узле, отдельном от JDS .....	41
8.1.1. Подключение Explain в JDS .....	42
8.2. Настройка на одном узле с JDS .....	43
9. Раздел JDS «Мониторинг» .....	45
9.1. Система «Prometheus» .....	45

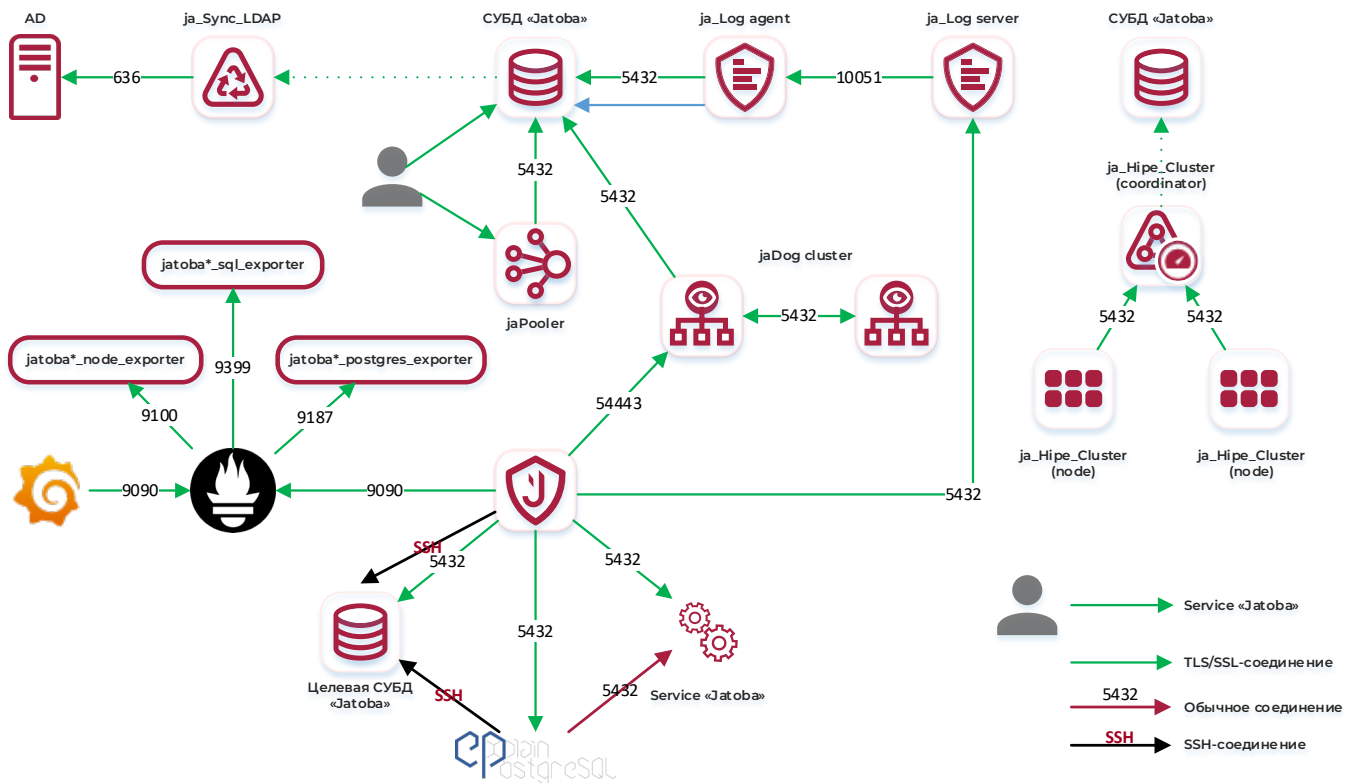
9.2. Экспортер «jatoba*_node_exporter» .....	47
9.3. Экспортер «jatoba*_postgres_exporter» .....	48
9.4. Экспортера «jatoba*_sql_exporter» .....	50
9.5. Настройка подключения в JDS .....	52
9.6. Настройка «Grafana» .....	52
10. ja_log. Централизованный сбор записей событий в СУБД.....	54
10.1. ja_log (сервер).....	54
10.2. ja_log (агент).....	55
10.3. TLS соединение между «jalog_server» и служебной СУБД с БД ja_log .....	57
10.4. JDS. Подключение ja_Log для сбора событий .....	57
11. jaPooler. Балансировщик подключений пользователей к СУБД .....	60
11.1. jaPooler. Подключение по SSL.....	60
11.2. jaPooler. Подключение по SSL с паролем.....	63
12. ja_sync_ldap. Синхронизация учетных записей служб каталогов и СУБД .....	68
13. SSL Аутентификация в контейнере.....	71
13.1. Запуск контейнера через docker compose с SSL аутентификацией .....	71
13.2. Запуск контейнера с SSL аутентификацией .....	73
14. Пример настройки SSL-соединений JDS .....	77
14.1. Требуемое программное обеспечение .....	77
14.2. Пользователи .....	77
14.3. Каталог хранения сертификатов .....	77
14.4. Создание конфигурационных файлов OpenSSL.....	77
14.4.1. Конфигурационный файл OpenSSL корневого ЦС .....	77
14.4.2. Конфигурационный файл OpenSSL промежуточного ЦС .....	78
14.5. Создание самоподписанных сертификатов .....	79
14.5.1. Самоподписанный сертификат корневого ЦС (Root CA) .....	79
14.5.2. Самоподписанный сертификат промежуточного ЦС (Root CA).....	80
14.5.3. Самоподписанный сертификат сервера СУБД (Root CA).....	81
14.6. Создание клиентских сертификатов .....	87
14.6.1. Клиентский сертификат пользователя postgres .....	87
14.6.2. Клиентский сертификат пользователя JDS.....	90
14.6.3. Клиентский сертификат пользователя test_user .....	93
14.7. Структура хранения сертификатов .....	96
14.8. Настройка СУБД для SSL-соединения .....	96
14.9. Создание цели (Target) с SSL-соединением .....	98
14.10. Настройка компонента JDS для SSL-соединений .....	99
15. Подготовка хостов на ОС Windows.....	103
15.1. Подготовка хоста с компонентом JDS.....	103
15.2. Подготовка хоста с СУБД на ОС Windows .....	104

16. Подготовка хоста с СУБД «PostgreSQL» для управления компонентом JDS.....	106
17. Настройки СУБД и ее компонент по умолчанию, которые могут быть использованы для НСД.....	109
17.1. Компоненты хранящие пароли (ключи).....	110
17.2. Перечень настроек авторизации и аутентификации СУБД «Jatoba» и входящих в ее состав компонент, отвечающих за ИБ.....	113
18. Реагирование на инциденты ИБ .....	129
18.1. Нарушение целостности и последующая блокировка пользователей СУБД.....	130
18.2. Превышение попыток количества неудачных попыток входа в СУБД.....	136
Приложение 1 .....	140
Установка службы JDS.PasDoctor .....	140
Перечень сокращений.....	142

## 1. ЭТАПЫ РАЗВЕРТЫВАНИЯ СУБД НА ХОСТАХ

Документ предназначен для администраторов СУБД с целью корректного формирования экосистемы СУБД «Jatoba».

В частности, использование SSH и SSL соединений. Схема подключений представлена на рисунке 1.1.



### Рисунок 1.1 – Схема SSH и SSL соединений

Имеющиеся механизмы развертывания компонентов входящих в состав СУБД «Jatoba» позволяют использовать:

- ручную установку;
- автоматизированную установку.

Ручная установка описана в документе «Руководство по установке».

### 1.1. Первый этап. Топология хостов СУБД

На первом этапе следует определить хосты для СУБД, их назначение и основные параметры.

## 1.2. Второй этап. Установка СУБД на хостах

В зависимости от типа хоста устанавливается СУБД. Как правило, для стандартных хостов СУБД используется инсталлятор

```
jatoba.sh install
```

Описание установки СУБД инсталлятором описана в документе «Руководство по установке».

Для узлов кластера с ролью Slave используется инсталлятор СУБД с

```
jatoba.sh install_server
```

Описание установки СУБД инсталлятором описана в документе

## 1.3. Третий этап. Установка JDS

На данном этапе на хосте с JDS устанавливается служебная СУБД и непосредственно сам компонент. Предпочтительный способ установки – инсталлятор JDS.

## 1.4. Четвертый этап. Подготовка хостов. Настройка SSH соединений

Этап включает в себя:

- подготовку хостов для целевых СУБД;
- настройку SSH соединений.

## 1.5. Пятый этап. Подготовка SSL подключений

В зависимости от внутренних требований, может быть настроена SSL аутентификация как для пользователей СУБД, так и для компонентов СУБД.

## 1.6. Шестой этап. Установка компонентов СУБД

После завершения конфигурирования служебного и целевых хостов СУБД «Jatoba» целесообразно установить требуемые компоненты.

## 2. НАСТРОЙКА SSH-СОЕДИНЕНИЙ JDS

В приведенном ниже описании приведен пример:

- создания SSH-соединения между хостами;
- подключения целевой СУБД к компоненту JDS.

В качестве примера используются 2 сервера с конфигурацией приведенной в таблице 2.1

Таблица 2.1 – Конфигурация сети стенда

№	Имя сервера	IP-адрес	Маска подсети	Компонент	Назначение хоста
1	u602doc-jds01	10.116.102.41/24	255.255.255.0	JDS	Служебная
2	u602doc-pgp01	10.116.102.49/24	255.255.255.0	Pg_profile	Целевая СУБД

Хостах должны быть установлены СУБД и на хосте JDS установлен непосредственно компонент.

### 2.1. Создание пользователя ОС jdscontrol на целевом хосте

Создать пользователя jdscontrol ОС командой в терминале ОС:

```
useradd -b /var/lib -m -s /usr/bin/bash jdscontrol
```

Задать пароль:

```
passwd jdscontrol
```

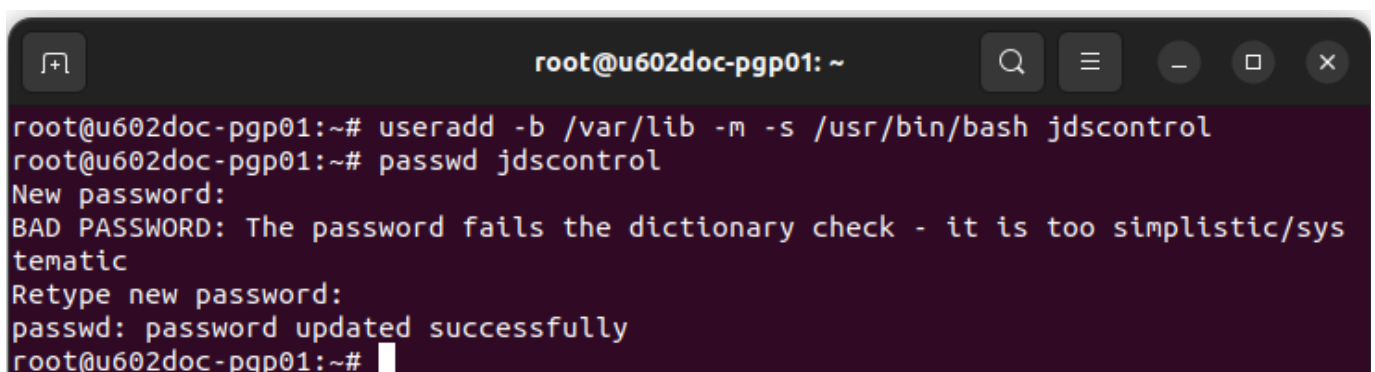


Рисунок 2.1 – Создание пользователя jdscontrol



## 2.2. Копирование скрипта на целевой хост

С после установки компонента JDS на служебном хосте будет находиться скрипт в каталоге `/opt/jds-scripts/assign_control_rights.sh`

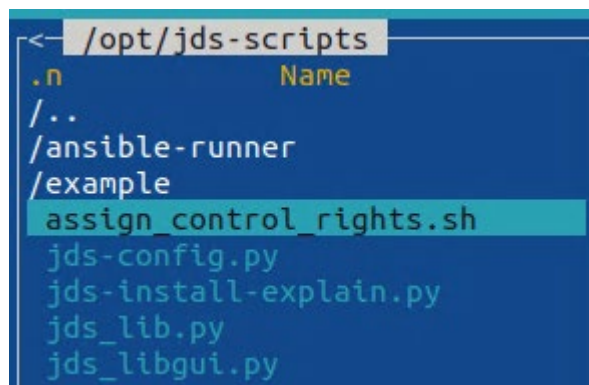


Рисунок 2.2 – Расположение скрипта настройки SSH

Данный скрипт вручную надо скопировать на хост целевой СУБД (10.116.102.49). Поместить в домашний каталог `/root`.

Установить права:

- чтение для владельца;
- запись для владельца;
- запуск/поиск для владельца;
- чтение для группы;
- запись/поиск для группы;
- чтение для других;
- запуск/поиск для других.

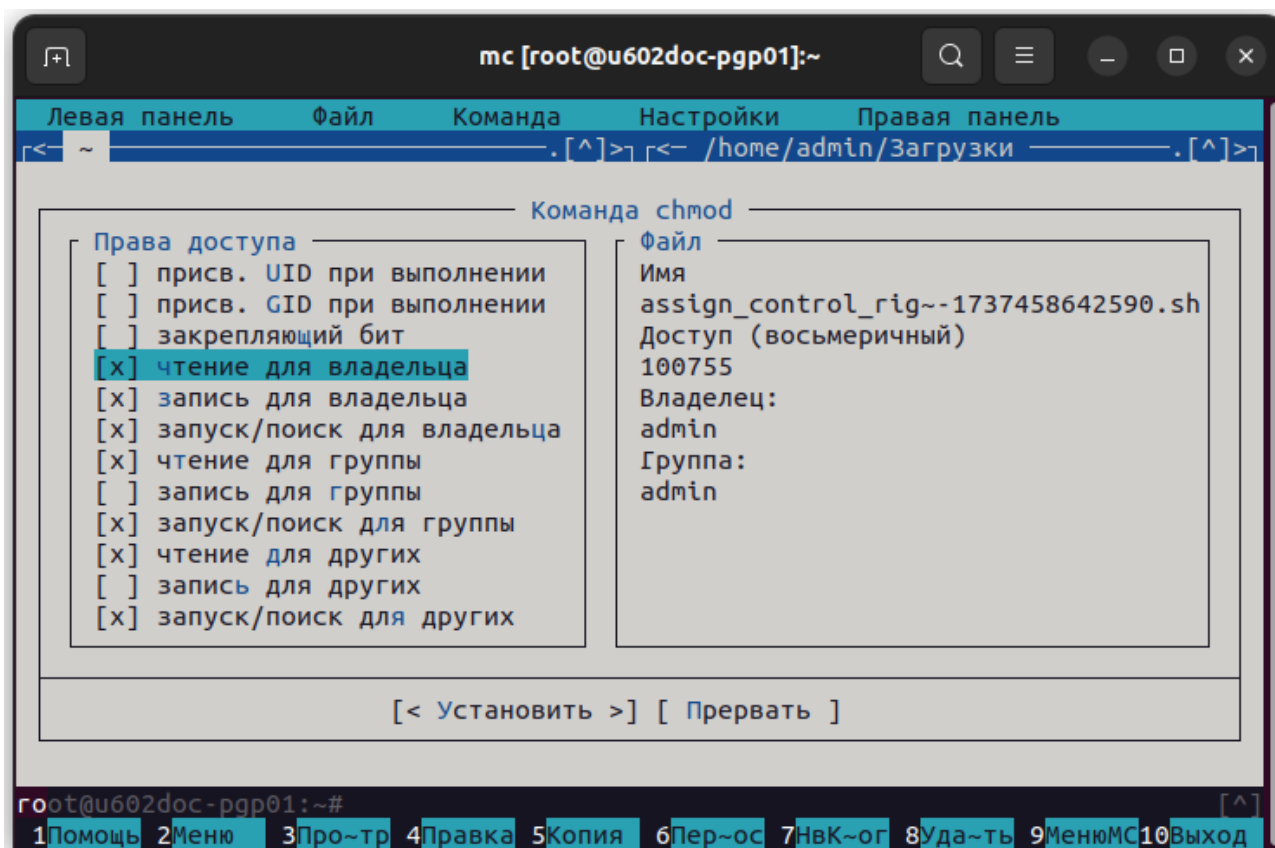


Рисунок 2.3 – Установка прав на скрипт через командер

Либо командой:

```
cd /root
chmod +x assign_control_rights.sh
```

### 2.3. Выполнение скрипта на целевом хосте

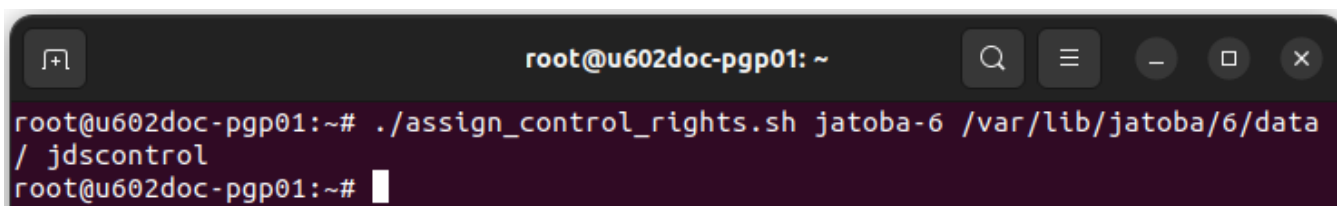
Команда запуска скрипта имеет синтаксис

Скрипт имеет собственный синтаксис запуска:

```
./assign_control_rights.sh <имя сервиса СУБД> <путь к папке  
DATA> <имя пользователя>
```

От имени и с правами привилегированного пользователя root выполнить команду в терминале ОС:

```
./assign_control_rights.sh jatoba-6 /var/lib/jatoba/6/data/  
jdscontrol
```

A terminal window with a dark background. The title bar shows 'root@u602doc-pgp01: ~'. The prompt is 'root@u602doc-pgp01:~#'. The command being executed is './assign\_control\_rights.sh jatoba-6 /var/lib/jatoba/6/data / jdscontrol'. The output is 'root@u602doc-pgp01:~#'.

```
root@u602doc-pgp01:~# ./assign_control_rights.sh jatoba-6 /var/lib/jatoba/6/data / jdscontrol
root@u602doc-pgp01:~#
```

Рисунок 2.4 – Выполнение скрипта

## 2.4. Создание домашнего каталога пользователя JDS на служебном хосте

Пользователь JDS был создан в процессе установки компонента и на данном шаге требуется создание для него домашней директории.

Создать папку пользователя, под которым работает JDS и назначить права, возможно командами:

```
#sudo -s
#mkdir /home/jds
#chown jds /home/jds
```

## 2.5. Создание ключа SSH для пользователя JDS на служебном хосте

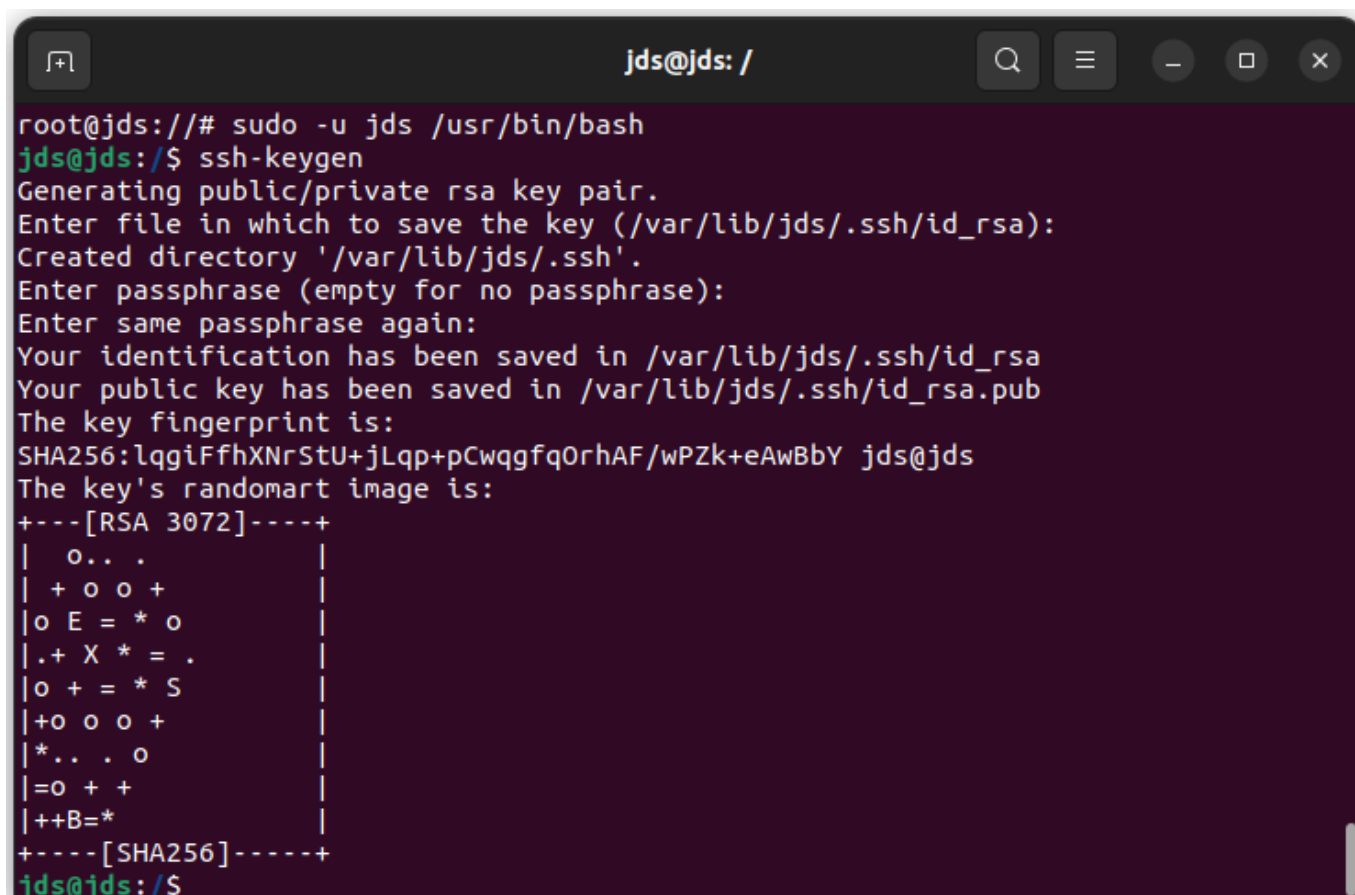
Создание ключей SSH должно проводиться от имени и с правами пользователя ОС jds.

Для этого требуется переключиться на пользователя jds:

```
sudo -u jds /usr/bin/bash
```

Создать ключ SSH для пользователя jds командой:

```
ssh-keygen
```



```
jds@jds: /
root@jds://# sudo -u jds /usr/bin/bash
jds@jds:/$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jds/.ssh/id_rsa):
Created directory '/var/lib/jds/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jds/.ssh/id_rsa
Your public key has been saved in /var/lib/jds/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:lqgiFfhXNrStU+jLqp+pCwqgfqOrhAF/wPZk+eAwBbY jds@jds
The key's randomart image is:
+---[RSA 3072]-----+
|  o.. .                |
| + o o +              |
| o E = * o            |
| .+ X * = .           |
| o + = * S            |
| +o o o +             |
| *.. . o              |
| =o + +               |
| ++B=*                |
+---[SHA256]-----+
jds@jds:/$
```

Рисунок 2.5 – Создание SSH ключей



При создании SSH ключей не устанавливается пароль доступа к ним

## 2.6. Передача ключа SSH на целевую СУБД

Созданный ключ SSH для пользователя jds должны быть переданы на хост целевой СУБД, для формирования SSH-соединения командой:

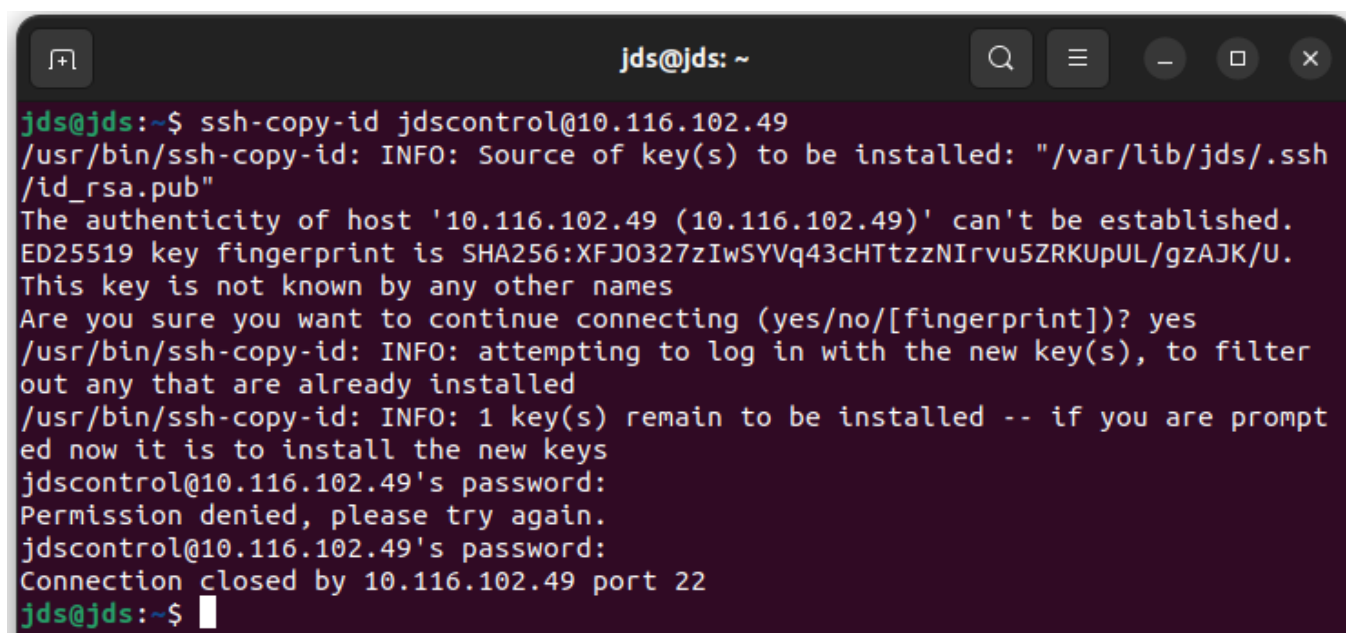
```
ssh-copy-id jdscontrol@10.116.102.49
```

Ответить «yes» прописью на вывод:

```
Are you sure you want to continue connecting
(yes/no/[fingerprint])?
```

Ввести пароль пользователя jdscontrol созданного на целевом хосте СУБД при выводе:

```
jdscontrol@10.116.102.49 password:
```



```
jds@jds: ~  
jds@jds:~$ ssh-copy-id jdscontrol@10.116.102.49  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/var/lib/jds/.ssh/id_rsa.pub"  
The authenticity of host '10.116.102.49 (10.116.102.49)' can't be established.  
ED25519 key fingerprint is SHA256:XFJ0327zIwSYVq43cHTtzzNIrvu5ZRKUUpUL/gzAJK/U.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter  
out any that are already installed  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt  
ed now it is to install the new keys  
jdscontrol@10.116.102.49's password:  
Permission denied, please try again.  
jdscontrol@10.116.102.49's password:  
Connection closed by 10.116.102.49 port 22  
jds@jds:~$
```

Рисунок 2.6 – Копирование SSH-ключа на целевой хост СУБД


Проверить подключение командой

```
ssh jdscontrol@10.116.102.49
```

 При подключении по SSH протоколу не должен запрашиваться пароль

После подключения проверьте операции над службой jatoba<ver>.

## 2.7. Подключение JDS к целевой СУБД (SSH и PASSWORD)

 Данный пункт частично дублирует описание настройки подключения к целевой СУБД данной в руководстве

На данном этапе возможно установить соединение между JDS и целевой СУБД. При этом используются SSH – соединение и метод аутентификации в СУБД «PASSWORD». Это позволит убедиться в корректности настройки и управляемости СУБД. В последствии возможно сменить метод аутентификации и подключаться к целевой СУБД по SSL протоколу.

В разделе «Ландшафт» Должна быть создана иерархическая структура объектов.

– Группа;

- Хост;
- СУБД.

### 2.7.1. Хост

В параметрах хоста целевой СУБД указываются параметры приведенные в таблице 2.2.

Таблица 2.2 – Устанавливаемые параметры для хоста

Параметр	Значение
Тип элемента: *	хост
IP-адрес или FQDN-имя *	10.116.102.49
Имя учётной записи *	jdscontrol
Порт для управления по SSH *	22
Описание	

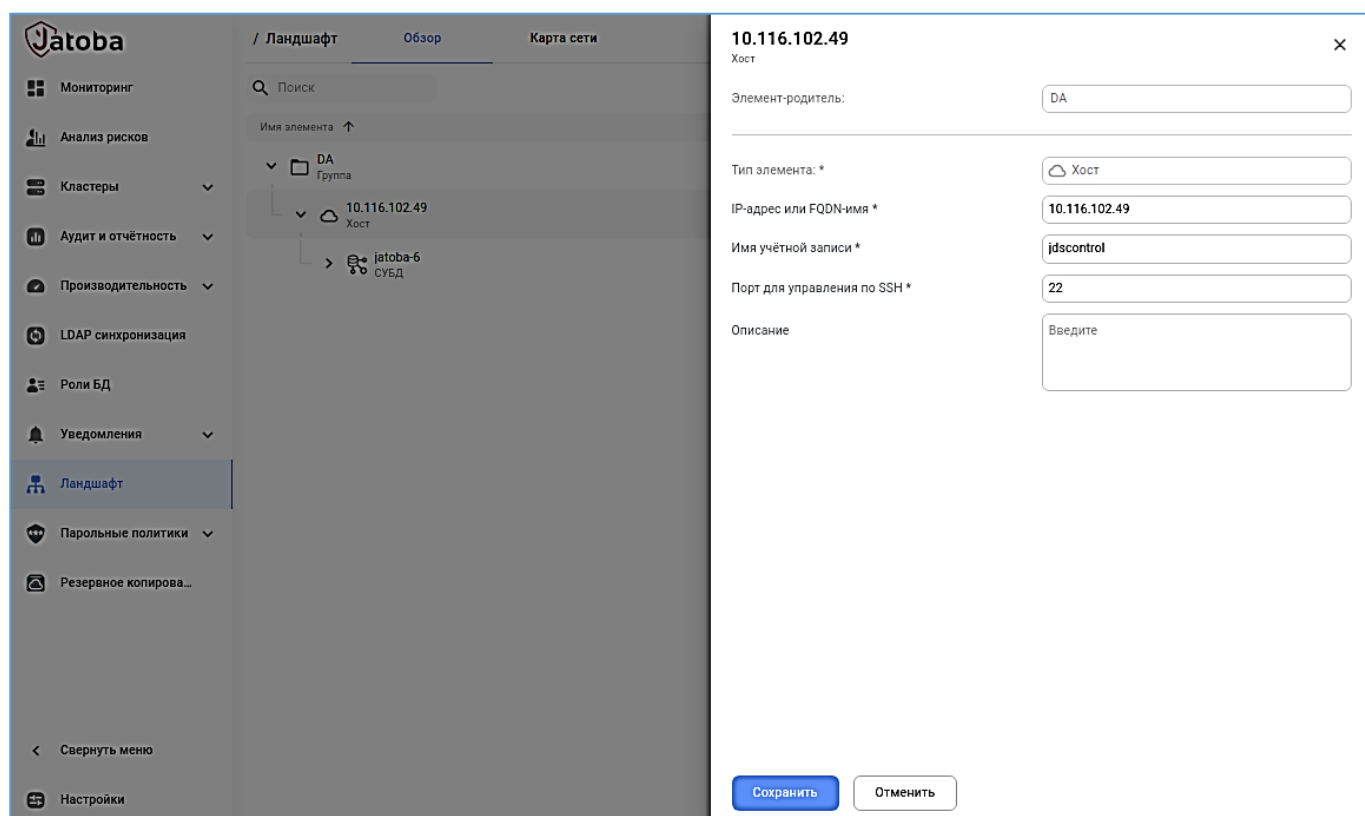


Рисунок 2.7 – Окно настройки хоста целевой СУБД в JDS

### 2.7.2. Настройка конфигурационного файла pg\_hba.conf

Перед настройками СУБД, во вкладке «Правила доступа» должны быть добавлены параметры подключения с удаленного хоста. В рассматриваемом примере это хост JDS IP 10.116.102.41.

В конфигурационном файле pg\_hba.conf должна быть строка, как минимум, разрешающая подключения из подсети.

Например

host	all	all	10.116.102.0/24	md5
------	-----	-----	-----------------	-----

Иначе компонент выведет ошибку и не позволит установить соединение с целевой СУБД.

### 2.7.3. СУБД

В параметрах СУБД указываются параметры приведенные в таблице 2.3

Таблица 2.3 – Устанавливаемые параметры для СУБД

Параметры	Значения
Имя сервиса *	jatoba-6
Порт: *	5432
Папка data*	/var/lib/jatoba/6/data
Папка для резервных копий конфигурации *	/var/lib/jdscontrol/backup
Сертификат УЦ	не используется
Имя служебной БД *	postgres
Режим шифрования	Disable
Проверять сертификат СУБД на отзыв:	не используется
Способ аутентификации	Пароль
Имя роли администратора СУБД *	postgres
Пароль	

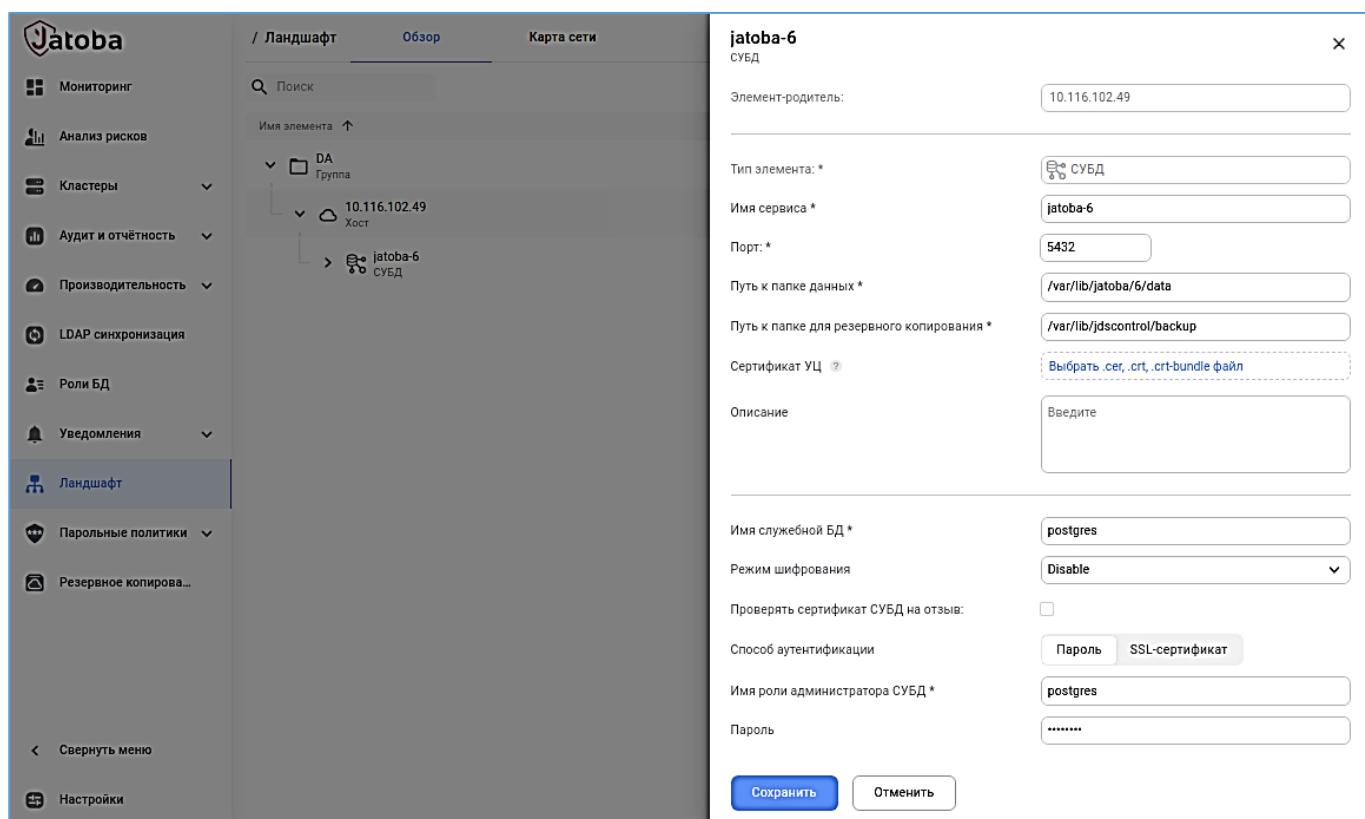


Рисунок 2.8 – Окно настройки целевой СУБД



### 3. НАСТРОЙКА SSL-СОЕДИНЕНИЙ

Настройка SSL соединений, для всех компонентов СУБД, должна выполняться от одно центра сертификации, т.е. на основе одного сертификата «СА».

СУБД «Jatoba» имеет возможность использования шифрованного сетевого трафика и аутентификацию по SSL сертификату. Аутентификация клиента по SSL сертификату позволяет серверу проверить личность подключающегося, подтверждая, что сертификат X.509, представленный клиентом, подписан центром сертификации. Рекомендуется использовать только доверенные центры сертификации для выдачи сертификатов клиенту и серверу

При выпуске серверных сертификатов поле SAN (а при его отсутствии – CN) должно соответствовать доменному имени сервера или его IP адресу.

Список файлов, которые будут использованы в текущем руководстве:

- корневой сертификат удостоверяющего центра (root.crt);
- сертификаты и ключи для каждого целевого узла (в приведенных примерах необходимо заменить {cn} на имя узла);
- сертификаты, ключи и контейнеры для клиентов;
- список отозванных сертификатов (root.crl.pem).

Корневой сертификат Центра сертификации должен быть установлен в целевой системе в «Доверенные корневые центры сертификации» (Windows)

Установка корневого сертификата в GNU/Linux:

- скопировать корневой сертификат удостоверяющего центра (root.crt) в каталог:

```
/usr/share/ca-certificates/
```

- выполнить команду в терминале ОС:

```
dpkg-reconfigure ca-certificates
```

и выбрать нужный сертификат

- выполнить команду в терминале ОС:

update-ca-certificates

Для всех настраиваемых узлов на DNS-сервере или в файлах hosts существуют записи FQDN, соответствующие создаваемым сертификатам

```
127.0.0.1      localhost jadog2.local
127.0.1.1      astra

# The following lines are desirable for
::1      localhost ip6-localhost ip6-loc
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.19.19.16 jds.local
172.19.19.31 prom.local
```

Рисунок 3.1

В названиях пакетов ПО Jatoba вместо символа «X» нужно подставить номер актуальной версии СУБД.

В файле pg\_hba.conf целевых СУБД предварительно надо задать необходимые настройки доступа, к примеру, для локальных подключений по IP или доменному имени, а также по SSL. Данные параметры приведены в качестве примера, для удаленных подключений необходимо указать конкретные адреса, базы и роли. Самая строгая степень проверки SSL сертификата: clientcert=verify-full. При этом типе авторизации проверяется соответствие значения поля CN пользовательского сертификата имени пользователя PostgreSQL.

```
host      all      all      127.0.0.1/32 md5
hostssl   all      all      all          cert clientcert=verify-full
```

В файле postgresql.conf целевых СУБД предварительно надо задать необходимые настройки подключения по SSL. В данном руководстве используются параметры:

```
ssl = on
ssl_ca_file = '/var/lib/jatoba/certs/root.crt'
ssl_cert_file = '/var/lib/jatoba/certs/{cn}.crt'
ssl_key_file = '/var/lib/jatoba/certs/{cn}.key'
ssl_crl_dir = '/var/lib/jatoba/certs/'
```

У данных файлов должны быть корректные права (владелец – postgres, доступ на файл ключа – 600):

```
chown -R postgres /var/lib/jatoba/certs  
chmod 600 /var/lib/jatoba/certs/{cn}.key
```

После добавления CRL-файла в каталог нужно запустить команду:

```
openssl rehash '/var/lib/jatoba/certs'
```



При использовании файла CRL нужно следить за истечением срока его актуальности для своевременной замены

SSL-подключения можно отслеживать в СУБД командой:

```
SELECT username, datname, ssl, client_addr, application_name  
FROM pg_stat_ssl  
JOIN pg_stat_activity ON pg_stat_ssl.pid =  
pg_stat_activity.pid;
```

### 3.1. Создание и конвертация сертификатов

Создание запроса на сертификат пользователя:

```
openssl req -new -nodes -text -out {cn}.csr -keyout {cn}.key -  
subj "/CN={cn}"
```

Создание запроса на сертификат сервера с использованием файла openssl.conf:

```
openssl req -new -nodes -text -out {name}.csr -keyout  
{name}.key -config openssl.conf
```

Содержимое файла openssl.conf

```
[req]  
default_bits = 2048  
distinguished_name = req_distinguished_name  
req_extensions = v3_req  
prompt = no  
  
[req_distinguished_name]  
CN=example.local
```

```
[v3_req]
keyUsage = keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth, clientAuth
subjectAltName = @alt_names

[alt_names]
IP.1 = <IP-адрес>
```

Ключевые поля файла: CN и IP.1 (адрес сервера). Их нужно менять перед генерацией каждого серверного сертификата.

Эти поля нужны, чтобы сертификат мог работать с сервером не только по DNS-имени, но и по адресу (SAN).

При настройке отказоустойчивого кластера также нужно добавлять в alt\_names публичный адрес ja\_Dog.

На основе содержимого полученного файла \*.csr (запроса на сертификат) на сайте Служб сертификации Active Directory запросить сертификат.

При импорте файлов сертификатов из Центра Сертификации MS Active Directory в ОС Linux требуется конвертация в формат PEM. В таком виде с ними может работать СУБД «Jatoba».

Конвертация списка отозванных сертификатов:

```
openssl crl -in {file_name}.crl -inform DER -out root.crl.pem
```

Конвертация корневого, клиентского или серверного сертификата:

```
openssl x509 -inform DER -in {file_name}.cer -out
{file_name}.crt
```

Для подключения JDS к целевым хостам требуется контейнер клиентского сертификата в формате pfx. При наличии промежуточного центра сертификации нужно создать файл root.crt-bundle

```
nano root.crt-bundle
```

Вставить в него содержимое файлов intermediate.crt (сертификат промежуточного ЦС) и root.crt (сертификат корневого ЦС), от begin до end из каждого, включительно, в таком порядке файлов. Данный бандл может быть основой для всех клиентских бандлов.



Пример такой настройки приведен в разделе 14 настоящего документа.

Создать цепочку сертификатов промежуточного и корневого ЦС для клиентского сертификата postgres:

```
cp root.crt-bundle client.postgres.crt-bundle
```

Создать контейнер PFX для пользователя postgres:

```
openssl pkcs12 -inkey client.postgres.key -in  
client.postgres.crt -certfile client.postgres.crt-bundle -  
export -out client.postgres.pfx
```

Пароль оставить пустым.

При отсутствии промежуточного ЦС можно использовать «корневой» сертификат как бандл, если его содержимое – с BEGIN до END. Иначе удалить лишние строки и затем использовать его для создания клиентского бандла теми же командами:

```
cp root.crt root.crt-bundle
```

### 3.2. Пути хранения сертификатов и ключей компонентов СУБД

При формировании сертификатов для SSL-соединения следует стремиться к единообразию имён и мест хранения.

Имена сертификатов и места их хранения представлены в таблице 3.1.

Таблица 3.1 – Имена сертификатов и места их хранения

Назначение/место соединения	Файл	Путь/настройка конфигурации
Хранение корневого сертификата в Linux	root.crt	/usr/share/ca-certificates/
СУБД		<b>postgresql.conf</b>
		/var/lib/jatoba/certs/
		ssl_ca_file = '/var/lib/jatoba/certs/root.crt'
		ssl_cert_file = '/var/lib/jatoba/certs/{cn}.cert'
		ssl_key_file = '/var/lib/jatoba/certs/{cn}.key'
		ssl_crl_dir = '/var/lib/jatoba/certs/'
jaDog		
Между узлами кластера		<b>postgresql.conf</b>
	root.crt	ssl_ca_file = '/var/lib/jatoba/certs/root.crt'
	server.crt	ssl_cert_file = '/var/lib/jatoba/certs/server.crt'
	server.key	ssl_key_file = '/var/lib/jatoba/certs/server.key'
		ssl_crl_dir = '/var/lib/jatoba/certs/'
		<b>В разделе 4 «Security connection settings»</b>
		/var/lib/jatoba/certs/root.crt
		/var/lib/jatoba/certs/server.crt
		/var/lib/jatoba/certs/root.cert.pem
		/var/lib/jatoba/certs/server.key
Доступ ja_Dog к СУБД по SSL		<b>В разделе 5) Database server system account and connection settings</b>
	root.crt	/var/lib/jatoba/certs/root.crt
	root.crt.pem	/var/lib/jatoba/certs/root.crt.pem
	client.jadog_user.crt	/var/lib/jatoba/certs/client.jadog_user.crt
	client.jadog_user.key	/var/lib/jatoba/certs/client.jadog_user.key

№ изменения: \_\_\_\_\_ Подпись отв. лица: \_\_\_\_\_ Дата внесения изм: \_\_\_\_\_

Назначение/место соединения	Файл	Путь/настройка конфигурации
Подключение «jaDog» к JDS разделе «Ландшафт»		<b>11 – Rest API settings</b>
	srv.crt	/var/lib/jatoba/certs/srv.crt
	srv.key	/var/lib/jatoba/certs/srv.key
	root.crt	/var/lib/jatoba/certs/root.crt
	root.crt.pem	/var/lib/jatoba/certs/root.crt.pem
	admin.pfx	/usr/share/jds/certs/clusters
<b>ja_Log (сервер)</b>	ja_log.crt	/var/lib/jatoba/certs/ja_log.crt
	ja_log.key	/var/lib/jatoba/certs/ja_log.key
	root.crt	/var/lib/jatoba/certs/root.crt
	root.crl.pem	/var/lib/jatoba/certs/root.crl.pem
<b>ja_Log (агент)</b>	ja_log_agent.crt	/var/lib/jatoba/certs/ja_log_agent.crt
	ja_log_agent.key	/var/lib/jatoba/certs/ja_log_agent.key
	root.crt	/var/lib/jatoba/certs/root.crt
	root.crl.pem	/var/lib/jatoba/certs/root.crl.pem
<b>JDS – служебная СУБД</b>		/usr/share/jds/certs
	root.crt-bundle	/usr/share/jds/certs/root.crt-bundle
	client.jds.pfx	/usr/share/jds/certs/client.jds.pfx
<b>Web-интерфейс JDS</b>		<b>/opt/jds/appsettings.json</b>
	jds.local.crt	/etc/nginx/ssl/jds.local.crt
	jds.local.key	/etc/nginx/ssl/jds.local.key
<b>JDS. Анализ запросов</b>		
	<b>На отдельном узле</b>	
	prometheus.local.crt prometheus.local.key	/etc/nginx/ssl/prometheus.local.crt
	jds.local.crt jds.local.key	/etc/nginx/ssl/prometheus.local.key
	<b>На одном узле</b>	

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

Назначение/место соединения	Файл	Путь/настройка конфигурации
	jds.local.crt	/etc/nginx/ssl/jds.local.crt
	jds.local.key	/etc/nginx/ssl/jds.local.key
<b>JDS. Мониторинг</b>		
<b>Система «Prometheus»</b>		
	prometheus.local.crt	/var/lib/certs/prometheus.local.crt
	prometheus.local.key	/var/lib/certs/prometheus.local.key
	root.crt	/var/lib/certs/root.crt
<b>jatoba* node_exporter</b>		<b>/usr/jatoba-X/monitoring/default/node_config.yml</b>
	root.crt	client ca file: /var/lib/certs_node/root.crt
	prometheus.local.crt	cert_file: /var/lib/certs_node/prometheus.local.crt
	prometheus.local.key	key_file: /var/lib/certs_node/prometheus.local.key
		<b>/usr/jatoba-X/monitoring/default/prometheus.yml</b>
	root.crt	/var/lib/certs_postgres/root.crt
	postgres_exporter.crt	/var/lib/certs_postgres/postgres_exporter.crt
	postgres_exporter.key	/var/lib/certs_postgres/postgres_exporter.key
<b>jatoba* postgres_exporter</b>		<b>/usr/jatoba-X/monitoring/default/postgres-config.yml</b>
	prometheus.local.crt	cert_file: /var/lib/certs_postgres/prometheus.local.crt
	prometheus.local.key	key_file: /var/lib/certs_postgres/prometheus.local.key
		<b>/usr/jatoba-X/monitoring/default/prometheus.yml</b>
	root.crt	ca_file: /var/lib/certs/root.crt
	prometheus.local.crt	cert_file: /var/lib/certs/prometheus.local.crt
	prometheus.local.key	key_file: /var/lib/certs/prometheus.local.key
		<b>/usr/jatoba-X/monitoring/default/postgres_exporter.yml</b>
	root.crt	sslrootcert=/var/lib/certs_postgres/root.crt
	postgres_exporter.crt	sslcert=/var/lib/certs_postgres/postgres_exporter.crt
	postgres_exporter.key	sslkey=/var/lib/certs_postgres/postgres_exporter.key"
<b>jatoba* sql_exporter</b>		<b>/usr/jatoba-X/monitoring/default/sql-config.yml</b>
	root.crt	client ca file: /var/lib/certs_sql/root.crt

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------



Назначение/место соединения	Файл	Путь/настройка конфигурации
	prometheus.local.crt	cert_file: /var/lib/certs_sql/prometheus.local.crt
	prometheus.local.key	key_file: /var/lib/certs_sql/prometheus.local.key
		<b>/usr/jatoba-X/monitoring/default/prometheus.yml</b>
	root.crt	client_ca_file: /var/lib/certs_sql/root.crt
	prometheus.local.crt	cert_file: /var/lib/certs_sql/prometheus.local.crt
	prometheus.local.key	key_file: /var/lib/certs_sql/prometheus.local.key
		<b>/usr/jatoba-X/monitoring/default/prometheus.yml</b>
	root.crt	ca_file: /var/lib/certs/root.crt
	prometheus.local.crt	cert_file: /var/lib/certs/prometheus.local.crt
	prometheus.local.key	key_file: /var/lib/certs/prometheus.local.key
		<b>/usr/jatoba-X/monitoring/default/sql_exporter.yml</b>
	root.crt	sslrootcert=/var/lib/certs_sql/root.crt
	sql_exporter.crt	sslcert=/var/lib/certs_sql/sql_exporter.crt
	sql_exporter.key	sslkey=/var/lib/certs_sql/sql_exporter.key

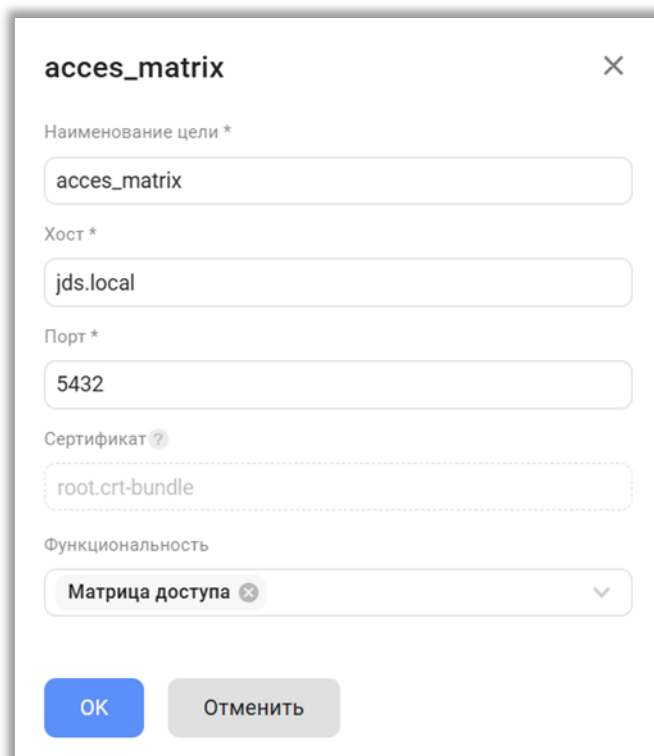
#### 4. ПОДКЛЮЧЕНИЕ JDS К ЦЕЛЕВОЙ СУБД ПО SSL

Для данного раздела понадобится файл бандла корневого и всех промежуточных сертификатов, созданный в разделе п.п. 3.1 «Создание и конвертация сертификатов».

Целевая СУБД должна быть предварительно настроена. Файлы postgresql.conf, pg\_hba.conf, файлы сертификатов хранятся в указанном каталоге.

На примере раздела «Матрица доступа»: в JDS перейти в Настройки - Цели - Добавить

Заполнить настройки для подключения к серверу СУБД. В поле сертификат нужно указать вышеуказанный бандл. В качестве функционала выбрать «Матрица доступа»



The image shows a configuration window titled 'aces\_matrix'. It has a close button (X) in the top right corner. The window contains several input fields and a dropdown menu. The first field is 'Наименование цели \*' with the value 'aces\_matrix'. The second field is 'Хост \*' with the value 'jds.local'. The third field is 'Порт \*' with the value '5432'. The fourth field is 'Сертификат ?' with the value 'root.crt-bundle'. The fifth field is a dropdown menu labeled 'Функциональность' with the selected option 'Матрица доступа'. At the bottom, there are two buttons: 'OK' and 'Отменить'.

Рисунок 4.1 – Создание цели

Нажать на кнопку «Добавить подключение». Заполнить настройки, выбрать режим VerifyFull, способ аутентификации - «SSL-сертификат», в качестве сертификата добавить контейнер rfx указанного выше пользователя

**Редактирование подключения** ✕

admin

Логин пользователя базы данных \*

postgres

Имя базы данных \*

postgres

Режим шифрования

VerifyFull ▾

Проверять сертификат хоста на отзыв

Да Нет

Способ аутентификации

Пароль SSL-сертификат

Сертификат ? \*

client.postgres.pfx

Пароль закрытого ключа

🔑 Тест подключения

ОК Отменить

Рисунок 4.2 – Окно редактирования подключения

Кнопка «Тест подключения» должна отобразить уведомление об успешном подключении.

В разделе Аудит и отчетность - Матрица доступа выбрать созданную цель.

## 5. ПОДКЛЮЧЕНИЕ JDS К СЛУЖЕБНОЙ СУБД ПО SSL

Для данного раздела понадобится файл корневого сертификата УЦ или бандл сертификатов, а также контейнер пользователя .pfx

Создать каталог для сертификатов:

```
mkdir /usr/share/jds/certs
```

Скопировать в каталог файл корневого сертификата (или бандл всех промежуточных ЦС) и контейнер пользователя .pfx

Назначить владельцем каталога пользователя, под именем которого запускается сервис JDS, и ограничить права на файлы:

```
chown -R jds /usr/share/jds/certs  
chmod -R 600 /usr/share/jds/certs
```

Отредактировать файл конфигурации JDS:

```
nano /opt/jds/appsettings.json
```

Заменить строку:

```
"ConnectionStrings": {"DefaultConnection": "User Id=jds;  
Password=sql; Server=localhost; Database=jdsdb; Port=5432;"  
},
```

на

```
"ConnectionStrings": {"DefaultConnection": " User Id=jds;  
Server=jds.local; Port=5432; Database=jdsdb;  
SslMode=VerifyFull"  
},
```

Добавить раздел:

```
},  
"ConnectionSslConfigurator": {  
"Connections": {  
"DefaultConnection": {  
"CAFile": "/usr/share/jds/certs/root.crt-bundle",  
"ClientPfxFile": "/usr/share/jds/certs/client.jds.pfx",
```

```
"ClientPfxPassword":null,  
"CheckServerCertificateRevocation":false  
}  
}  
}  
}
```

Перезапустить службу JDS:

```
systemctl restart jds.service
```

## 6. ПОДКЛЮЧЕНИЕ К WEB-ИНТЕРФЕЙСУ JDS ПО SSL

Для данного раздела понадобится файл корневого сертификата УЦ или бандл сертификатов, а также контейнер пользователя .pfx

По умолчанию JDS устанавливается с самоподписанным сертификатом, его требуется заменить.

Для данного раздела понадобятся файлы jds.local.crt и jds.local.key (сертификат, сгенерированный для сервера с запущенным сервисом JDS)

Для запуска web-интерфейса JDS с сертификатом из ЦС нужно скопировать файлы сертификата и ключа в каталог SSL:

```
cp jds.local.crt /etc/nginx/ssl/  
cp jds.local.key /etc/nginx/ssl/
```

Отредактировать конфигурацию JDS для nginx

```
nano /etc/nginx/conf.d/jds.https.conf
```

Заменить названия файлов сертификата и ключа

```
listen 443 ssl;  
    ssl_certificate          /etc/nginx/ssl/jds.local.crt;  
    ssl_certificate_key      /etc/nginx/ssl/jds.local.key;
```

Перезапустить службу nginx

```
systemctl restart nginx.service
```

## 7. НАСТРОЙКА SSL КОМПОНЕНТА JA\_DOG

### 7.1. Настройка SSL между узлами кластера

В примере рассматривается настройка SSL-подключения между 2 узлами отказоустойчивого кластера

Провести стандартную установку jadog на все узлы кластера, следуя шагам из документа «Компонент jaDog. Управление режимом работы узлов кластера»

Добавить в pg\_hba.conf главного узла строки

hostssl	replication	jadog	<IP другого узла>/32	cert	clientcert=verify-full
hostssl	all	jadog	<IP другого узла>/32	cert	clientcert=verify-full
hostssl	all	jadog	127.0.0.1/32	cert	clientcert=verify-full
hostssl	all	jadog_user	all	cert	clientcert=verify-full

Для всех узлов сформировать серверные сертификаты по инструкциям из раздела «Создание сертификатов» (обязательно с SAN, в дальнейшем будет настройка сервиса по IP), скопировать их, а также корневой сертификат (root.crt) и список отозванных сертификатов (root.crl.pem) в /var/lib/jatoba/certs

Так как конфигурация СУБД с путями к сертификатам будет скопирована в процессе первой репликации, должны быть одинаковыми не только пути, но и имена файлов. Файлы {cn}.crt и {cn}.key переименовать в server.crt и server.key на всех узлах. Параметры CN и SAN не будут затронуты:

```
cd /var/lib/jatoba/certs
mv {cn}.crt server.crt
mv {cn}.key server.key
```

В файле postgresql.conf главного узла нужно внести соответствующие изменения в пути к файлам сертификатов.

```
ssl = on
ssl_ca_file = '/var/lib/jatoba/certs/root.crt'
ssl_cert_file = '/var/lib/jatoba/certs/server.crt'
ssl_key_file = '/var/lib/jatoba/certs/server.key'
ssl_crl_dir = '/var/lib/jatoba/certs/'
```

Рисунок 7.1 – Пример настроек SSL-соединения в файле postgresql.conf

Задать права на каталог и файлы командой:

```
chown -R postgres /var/lib/jatoba/certs  
chmod 600 /var/lib/jatoba/certs/server.key
```

На первом узле перенастроить конфигурацию jalog для использования SSL:

```
cd /usr/jatoba-X/bin
```

#если первоначальная настройка уже была:

```
./jalog setup -C ../etc/jalog/
```

#если это первая настройка:

```
./jalog setup
```

В разделе 2 «Inter-jalog communication setting», пункт 5 «SSL on (param\_ssl:ssl)» переключить в значение true:

```
1) Jalog service name (param_jalog:service_name) [jalog]  
2) Jalog IP address (param_jalog:ip) [172.19.19.31/24]  
3) Jalog PORT number (param_jalog:port) [12345]  
4) Jalog searching protocol port (param_jalog:jalog_search_port) [12346]  
5) SSL on (param_ssl:ssl) [true]  
6) Jalog interconnection user (param_jalog:interconnect_user) [admin]
```

Рисунок 7.2 - Раздел 2 «Inter-jalog communication setting»

В разделе 4 «Security connection settings» заполнить пути к файлам сертификатов и ключей.

```
1) SSL Center Authority certificate (param_ssl:ssl_ca_file) [/var/lib/jatoba/certs/root.crt]  
2) SSL User certificate (param_ssl:ssl_cert_file) [/var/lib/jatoba/certs/server.crt]  
3) SSL Certificate Revocation List (param_ssl:ssl_crl_file) [/var/lib/jatoba/certs/root.crl.pem]  
4) SSL User private key (param_ssl:ssl_key_file) [/var/lib/jatoba/certs/server.key]
```

Рисунок 7.3 - разделе 4 «Security connection settings»

Повторно ввести пароль пользователя кластера в разделе 6.

С помощью раздела 13 сохранить настройки и выйти из конфигуратора.

Повторить аналогичную настройку на втором узле кластера.

В файле /usr/jatoba-X/etc/jalog/jalog\_hba.cfg на обоих узлах добавить строку (если ее там нет)

```
all all ssl
```



Проверить, что служба `jadog` запущена на всех узлах.

```
systemctl status jadog.service
```

С помощью утилиты `jadog_ctl` на главном сервере создать кластер и включить в него узлы согласно документации.

На главном узле можно проверить тип соединений к базе данных (`ssl = «t»`) SQL-командой:

```
sudo -u postgres psql
SELECT username, datname, ssl, client_addr, application_name
FROM pg_stat_ssl
JOIN pg_stat_activity
ON pg_stat_ssl.pid = pg_stat_activity.pid;
```

```
postgres=# SELECT username, datname, ssl, client_addr, application_name FROM pg_stat_ssl
JOIN pg_stat_activity ON pg_stat_ssl.pid = pg_stat_activity.pid;
 username | datname | ssl | client_addr | application_name
-----+-----+----+-----+-----
 jadog_user |         | t   | 172.19.19.31 | slot1
```

Рисунок 7.4 – SQL-команда проверки SSL-соединения между узлами

## 7.2. Настройка доступа `ja_Dog` к СУБД по SSL

Действия проводятся на узлах кластера, настроенного по шагам из прошлого раздела

Создать клиентский сертификат для пользователя `jadog_user` по инструкции, скопировать файлы в каталог `/var/lib/jatoba/certs` всех узлов кластера

```
cp ./client.jadog_user.* /var/lib/jatoba/certs/
chown -R postgres /var/lib/jatoba/certs
chmod 600 /var/lib/jatoba/certs/client.jadog_user.key
```

На всех узлах запустить изменение конфигурации `jadog` командой:

```
cd /usr/jatoba-X/bin
./jadog setup -C ../etc/jadog/
```

Перейти в пункт меню 3 «User / Admin access network setting», в Подменю 1) Public IP address (`param_jadog: public_address`) и заполнить значение:

1) Public address (`param_jadog:public_address`) - общий IP-адрес данного узла (тот же, что указан как SAN в сертификате данного сервера)

Перейти в пункт 5) Database server system account and connection settings и заполнить значения:

- 6) Database auth method (db\_connection\_settings:db\_auth\_method) ssl
- 7) Jadog to database CA file (db\_connection\_settings:ssl\_ca\_file) /var/lib/jatoba/certs/root.crt  
путь к файлу корневого сертификата
- 8) Jadog to database CRL file (db\_connection\_settings:ssl\_crl\_file) /var/lib/jatoba/certs/root.crt.pem  
путь к файлу отозванных сертификатов
- 9) Jadog to database cert file (db\_connection\_settings:ssl\_cert\_file) /var/lib/jatoba/certs/client.jadog\_user.crt  
путь к файлу сертификата jadog\_user
- 10) Jadog to database key file (db\_connection\_settings:ssl\_key\_file) /var/lib/jatoba/certs/client.jadog\_user.key  
путь к файлу ключа сертификата jadog\_user

```

1) Database host (db_connection_settings:host) [172.19.19.31]
2) Database port (db_connection_settings:port) [5432]
3) Database service name (param_postgres:db_service_name) [jatoba-6]
4) Database name (db_connection_settings:database) [postgres]
5) Jadog password file (db_connection_settings:passfile) [/usr/jatoba-6/bin/db_passfile]
6) Database auth method (db_connection_settings:db_auth_method) [ssl]
7) Jadog to database CA file (db_connection_settings:ssl_ca_file) [/var/lib/jatoba/certs/root.crt]
8) Jadog to database CRL file (db_connection_settings:ssl_crl_file) [/var/lib/jatoba/certs/root.crt.pem]
9) Jadog to database cert file (db_connection_settings:ssl_cert_file) [/var/lib/jatoba/certs/client.jadog_user.crt]
10) Jadog to database key file (db_connection_settings:ssl_key_file) [/var/lib/jatoba/certs/client.jadog_user.key]
11) Jadog to database SSL mode (db_connection_settings:ssl_mode) [verify-full]
12) Jadog database user name (db_connection_settings:user) [jadog_user]
13) Jadog database user password (db_connection_settings:user_pass) [HIDDEN VALUE]
14) Database server OS user (param_system:system_user) [postgres]
15) Replication node name (param_replication:replication_slot_name) [slot1]
```

Рисунок 7.5 - пункт 5) Database server system account and connection settings

Ввести пароль пользователя в пункте 13.

Выйти из настройки через общий раздел 13.

При необходимости перезапустить узлы кластера в правильном порядке или в приложении jadog\_ctl воспользоваться командой:

```
reload jadog on cluster
```

### 7.3. Подключение кластера «ja\_Dog» к JDS разделе «Ландшафт»

В приведенном ниже описании приведен пример создания цели (Target) с подключением по SSL/TLS.

Требуются: JDS версии 2.7.0 и новее, jaDog версии 3.2 и новее.

Подключение кластеров осуществляется с помощью Rest API компонента jaDog. Rest API всегда использует SSL подключение.

Для данного раздела понадобится pfx-контейнер сертификата администратора кластера на основе клиентских сертификата и ключа пользователя admin:

```
openssl pkcs12 -inkey admin.key -in admin.crt -certfile  
root.crt -export -out admin.pfx
```

### 7.3.1. Настройка Rest API

Настройка Rest API возможно двумя способами:

#### 1. При первоначальной настройке конфигурации узла кластера:

На всех узлах кластера запустить изменение конфигурации:

```
cd /usr/jatoba-X/bin  
./jadog setup -C ../etc/jadog/
```

Перейти в раздел 11 – Rest API settings, заполнить значения:

- |   |                                    |
|---|------------------------------------|
| 1) REST API use (param_rest_api:rest_api_use)                             | «true»                             |
| 2) REST API listen address (param_rest_api:rest_api_listen_address)       | «0.0.0.0»                          |
| 3) REST API listen port (param_rest_api:rest_api_listen_port)             | 54443                              |
| порт, который будет использовать Rest API, по умолчанию                   |                                    |
| 4) REST API TLS server certificate (param_rest_api:rest_api_cert_file)    | /var/lib/jatoba/certs/srv.crt      |
| путь к файлу открытого ключа сервера                                      |                                    |
| 5) REST API TLS server private key (param_rest_api:rest_api_key_file)     | /var/lib/jatoba/certs/srv.key      |
| путь к файлу закрытого ключа сервера                                      |                                    |
| 6) REST API TLS CA bundle (param_rest_api:rest_api_ca_file)               | /var/lib/jatoba/certs/root.crt     |
| путь к файлу корневого сертификата  |                                    |
| 7) REST API TLS server revocation list (param_rest_api:rest_api_crl_file) | /var/lib/jatoba/certs/root.crt.pem |
| путь к файлу отозванных сертификатов                                      |                                    |

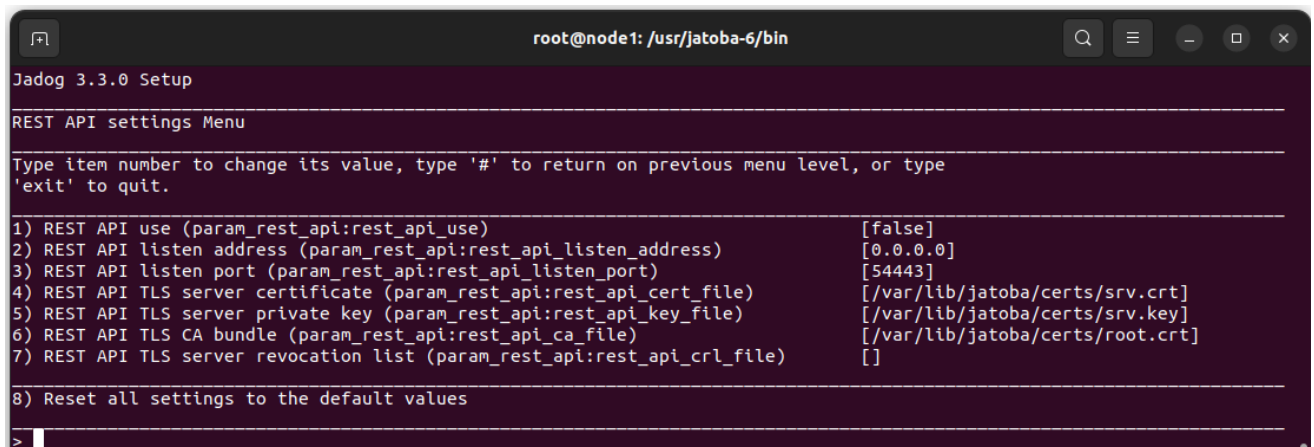


Рисунок 7.6 - Раздел 11 – Rest API settings

#### 2. С помощью изменения файла конфигурации jadog.yml.

Файл конфигурации jadog.yml располагается по адресу:

```
/usr/jatoba-<ver>/etc/jadog
```

Для активации и определения параметров REST API предназначена секция `param_rest_api`. В данную секцию вносятся следующие изменения:

```
param_rest_api:
rest_api_use: true
rest_api_cert_file: /var/lib/jatoba/serts/srv.crt
rest_api_key_file: /var/lib/jatoba/serts/srv.key
rest_api_ca_file: /var/lib/jatoba/serts/root.crt
rest_api_crl_file: /var/lib/jatoba/serts/root.crl.pem
rest_api_listen_address: 0.0.0.0
rest_api_listen_port: 54443
```

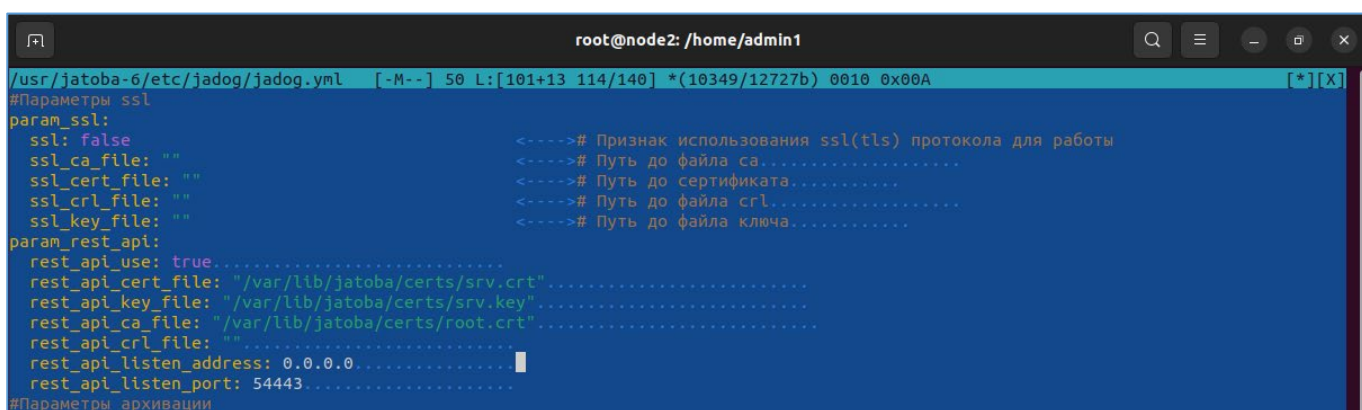


Рисунок 7.7 - Файл конфигурации `jadog.yml`

После внесения изменений необходимо сохранить файл конфигурации «`jadog.yml`»

Работа с файлом конфигурации подробно описана в документе «Руководство по настройке. Часть 1. Управление режимом работы узлов кластера. Компонент «`jaDog`»



Работа с файлом конфигурации подробно описана в документе «Руководство по настройке. Часть 1. Управление режимом работы узлов кластера. Компонент «`jaDog`».

Если кластер уже создан, для того чтобы внесенные в файл конфигурации «`jadog.yml`» изменения вступили в силу, необходимо перезагрузить сервис компонента «`jaDog`» при помощи команды:

```
systemctl restart jadog
```

### 7.3.2. Проверочные мероприятия

Перед добавлением существующего кластера в раздел «Ландшафт» JDS требуется перепроверить настройки и собрать необходимую информацию о кластере.

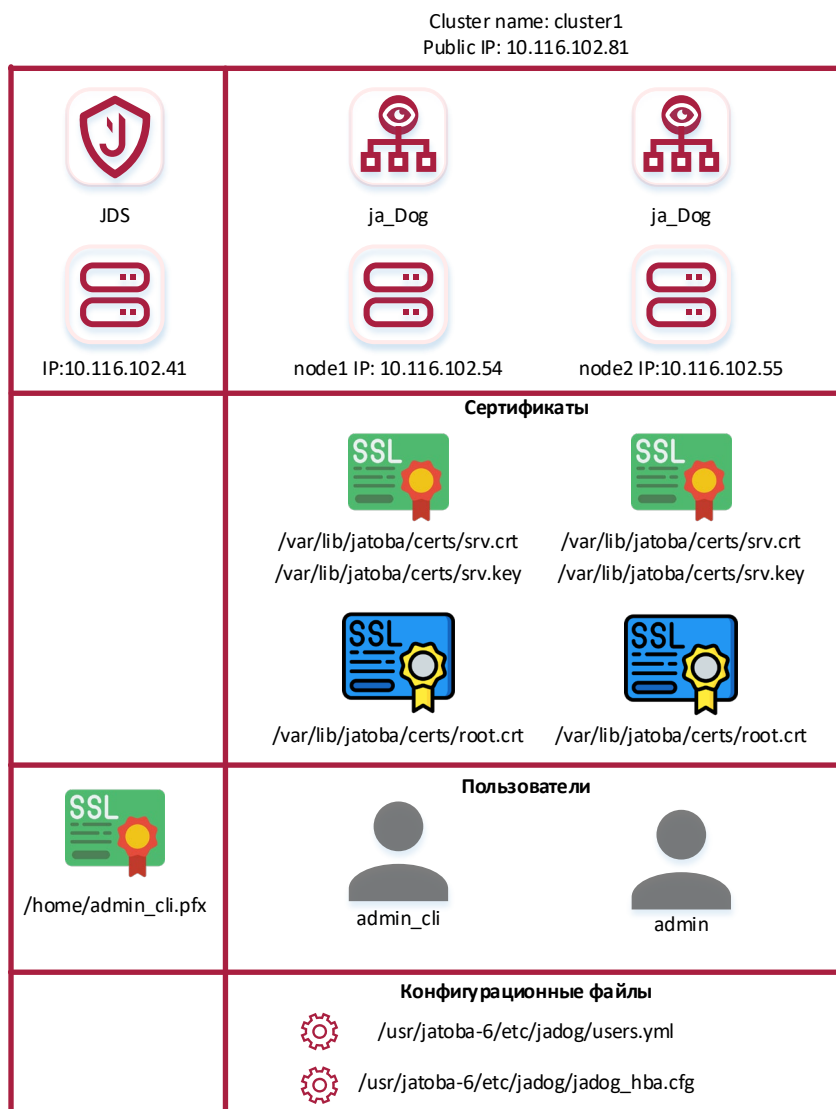


Рисунок 7.8 – Схема настроек подключения кластера

Подключение к публичному адресу кластера со стороны JDS выполняется одной из учетных записей администраторов кластера. Соответственно должен быть сформирован клиентский сертификат в формате \*.PFX для администратора кластера. При этом все сертификаты, как клиентские, как серверные должны быть сформированы от одного ЦС.

Учетная запись администратора кластера должна быть заведена в кластере и отражаться в файле по пути /usr/jatoba-<ver>/etc/jadog/user.yml.

```

GNU nano 6.2 /usr/jatoba-6/etc/jadog/users.yml
jadog_users:
- name: admin
  password: BD831BC3BCB76A812EA8F023C9521E80AA316AC4AA29CDD3636479DABD9B1003
  blocked: false
- name: admin_cli
  password: 66B78CA3A32A99E912D5F8FEE1025F642FF0B0442F4BAEE7BA816AB48280215F
  blocked: false
  
```

Рисунок 7.9 – Список пользователей кластера

В конфигурационном файле аутентификации по пути /usr/jatoba-  
<ver>/etc/jadog/jadog\_hba.cfg, должно быть разрешено подключение по SSL.

```

GNU nano 6.2 /usr/jatoba-6/etc/jadog/jadog_hba.cfg *
# USER ADDRESS METHOD
all all sha-256
admin_cli all ssl
admin all ssl
  
```

Рисунок 7.10 - Конфигурационный файл аутентификации кластера

В файле состояния кластера по пути: /usr/jatoba-<ver>/etc/jadog/jadog\_state.yml  
получить значения параметров приведенных в таблице 7.1 .

Таблица 7.1 – Требуемые параметры для подключения кластера в файле  
jadog\_state.yml

Описание параметра	Наименование параметра	Значение параметра примера	Поле во вкладке «Подключение к кластеру»
Имя кластера	cluster_name	cluster1	Название
Публичный IP-адрес	PublicIP	10.116.102.81	Адрес

В файле параметров созданного узла по пути: /usr/jatoba-<ver>/etc/jadog/jadog.yml,  
сверить и получить значения параметров приведенных в таблице 7.2.

Таблица 7.2 – Требуемые параметры для подключения кластера в файле jadog.yml

Описание параметра	Параметр и значение примера	Поле во вкладке «Подключение к кластеру»
Публичный IP-адрес	public_address: 10.116.102.81/24	Адрес
Параметры REST API	param_rest_api: rest_api_use: true rest_api_cert_file: /var/lib/jatoba/certs/srv.crt	

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

Описание параметра	Параметр и значение примера	Поле во вкладке «Подключение к кластеру»
	rest_api_key_file: /var/lib/jatoba/certs/srv.key rest_api_ca_file: /var/lib/jatoba/certs/root.crt	
Порт REST API	rest_api_listen_port: 54443	Порт REST API

### 7.3.3. Добавление существующего кластера ja\_Dog

Выполнив проверочные мероприятия и собрав требуемую информацию для подключения, возможно подключение кластера к разделу «Ландшафт».

Потребуется перейти в раздел «Ландшафт», во вкладку «Кластеры» и нажать кнопку «Подключиться». В открывшемся окне «Подключение к кластеру» ввести значения параметров.

В «Название» указать имя имеющегося кластера, которое указывалось при создании кластера (поле регистрозависимое), в поле «Адрес» указать IP-адрес или DNS имя любого узла кластера, в том числе Public IP, в поле «Порт Rest API» указать порт, используемый Rest API, в поле сертификат нужно указать pfx-контейнер администратора кластера.

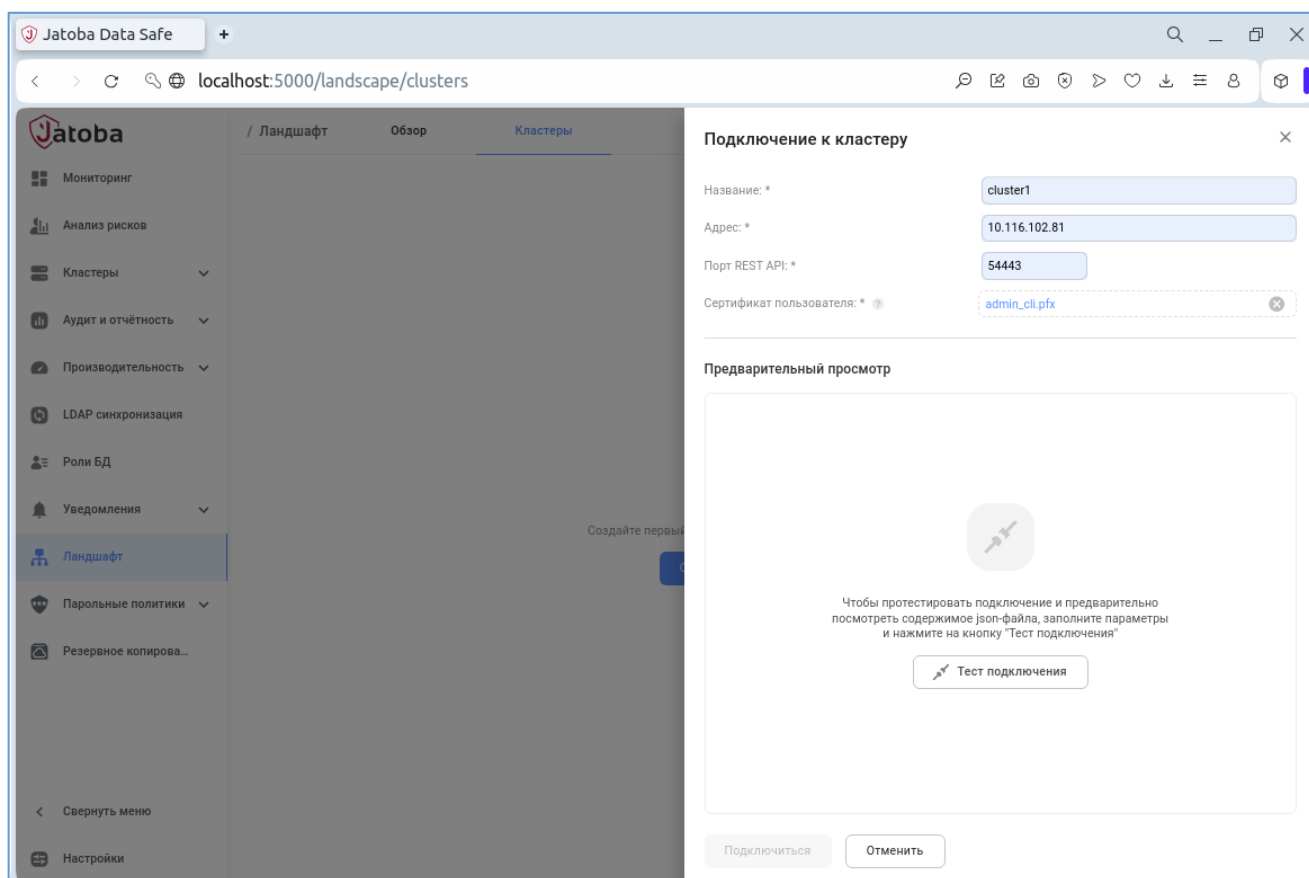


Рисунок 7.11 – Окно подключения к кластеру

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------



Далее нажать «Тест подключения». При корректных указанных параметрах должны отобразиться параметры кластера. После чего нажать «Подключиться».

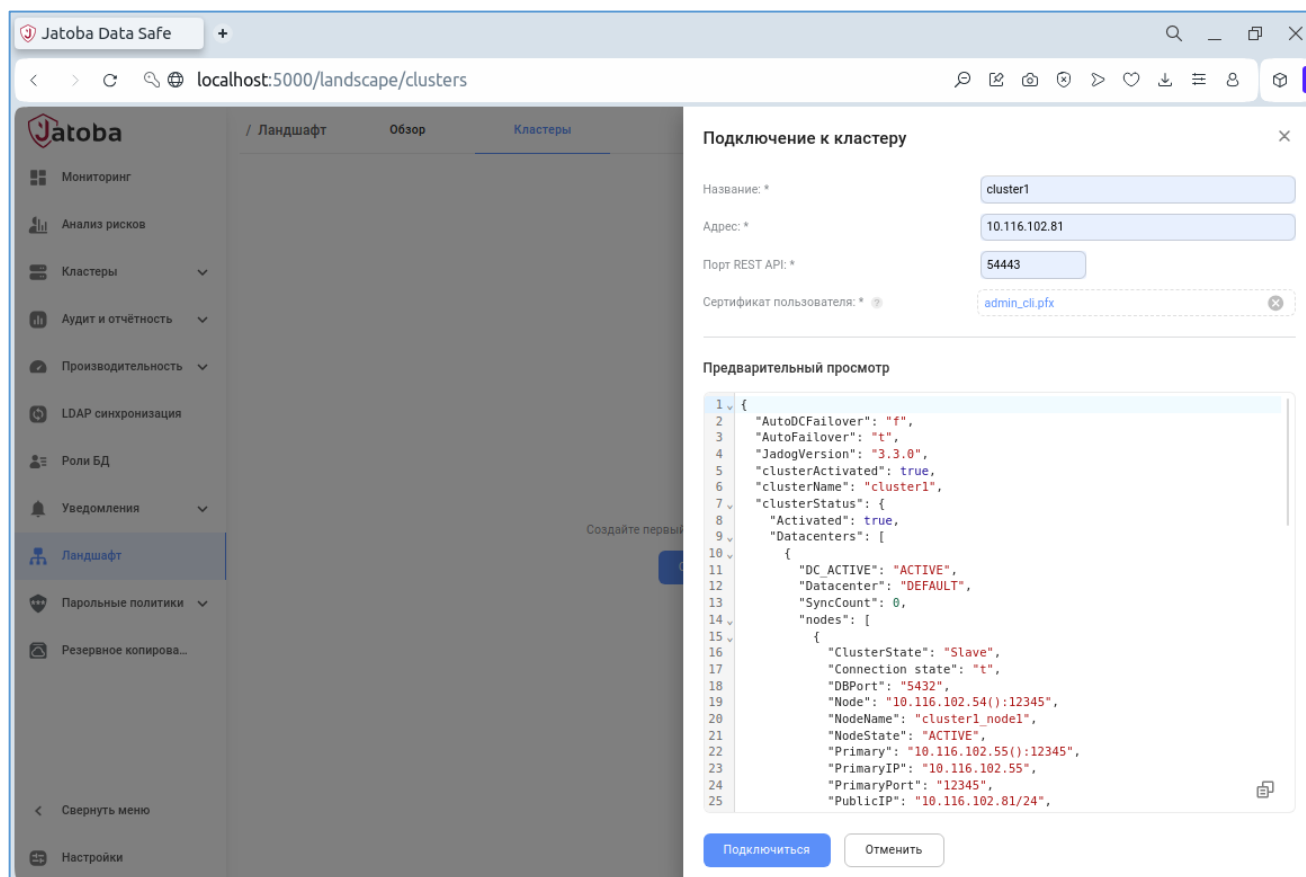


Рисунок 7.12 – Подключение к кластеру

Подключенный кластер отразится в двух разделах JDS:

- Ландшафт;
- Кластеры/Jadog кластеры.

В разделе «Ландшафт», при настроенном SSH-соединении будет доступно управление СУБД.

В разделе Кластеры/Jadog кластеры выполняется непосредственно управление кластером.

Функциональные возможности разделов описаны в документе «Руководство по настройке. Часть 7. Пользовательский веб-интерфейс для администраторов. Компонент «Jatoba data safe».



## 8. РАЗДЕЛ JDS «АНАЛИЗ ЗАПРОСОВ»

Руководство по полной настройке компонента находится в документе «Поддержка мониторинга СУБД в части анализа запросов»

Подключение модуля explain к служебной СУБД по SSL не поддерживается. Подключение к целевой СУБД осуществляется по SSH.

В данном разделе рассматривается настройка системы для доступа по SSL к web-интерфейсу explain.

### 8.1. Настройка на узле, отдельном от JDS

Для данного раздела понадобятся:

- серверные сертификат и ключ;
- установлен полный пакет explain, monitor, nginx;
- компонент explain, установленный и доступный по адресу `http://<адрес сервера explain>:8080`.

Имя узла в данном разделе - `prometheus.local`.

Создать папку для сертификата и ключа командой в терминале ОС:

```
mkdir /etc/nginx/ssl
```

Скопировать в этот каталог сертификат и ключ данного сервера.

Создать файл конфигурации сайта.

```
nano /etc/nginx/conf.d/explain.https.conf
```

Вставить текст и сохранить:

```
server {  
    charset utf-8;  
    access_log /var/log/nginx/explain.access.log;  
    error_log /var/log/nginx/explain.error.log;  
    listen 443 ssl;  
    ssl_certificate  
/etc/nginx/ssl/prometheus.local.crt;  
    ssl_certificate_key  
/etc/nginx/ssl/prometheus.local.key;
```

```
location / {  
    proxy_pass http://localhost:8080;  
}
```

Перезапустить службу nginx командой.

```
systemctl restart nginx
```

Перейти по адресу:

```
https://<адрес сервера explain>
```

### 8.1.1. Подключение Explain в JDS

Взаимодействие JDS с сервисом pg-explain настраивается в конфигурационном файле приложения JDS appsettings.json - в свойстве PgExplainConfig.BaseAddress указать URL по которому доступен https-сервис pg-explain, например:

```
nano /opt/jds/appsettings.json
```

```
...  
"PgExplainConfig": {  
    "BaseAddress": "https://<адрес сервера explain>"  
},  
...
```

Адрес должен быть указан без закрывающего знака «/» - дробная черта.

Зайти в web-интерфейс JDS по логину и паролю (по умолчанию, admin - secret), пункты меню Производительность – Анализ запросов.

Перейти на вкладку Настройки, мини-вкладка Добавить. Ввести IP-адрес узла с наблюдаемой СУБД, порт (если он отличается от стандартного 5432) и отметить чекбоксы собираемой статистики, например, все 4. Сохранить.

На мини-вкладке «все» должен отобразиться добавленный узел.

## 8.2. Настройка на одном узле с JDS

Используется web-сервер nginx из состава JDS. Шаги и параметры немного отличаются.

Для данного раздела понадобятся серверные сертификат и ключ.

Имя узла в данном разделе - jds.local

Создать папку для сертификата и ключа:

```
mkdir /etc/nginx/ssl
```

Скопировать в этот каталог сертификат и ключ данного сервера

Создать файл конфигурации сайта:

```
nano /etc/nginx/conf.d/explain.https.conf
```

Вставить текст и сохранить:

```
server {  
    charset utf-8;  
    access_log /var/log/nginx/explain.access.log;  
    error_log /var/log/nginx/explain.error.log;  
    listen 444 ssl;  
    ssl_certificate /etc/nginx/ssl/jds.local.crt;  
    ssl_certificate_key /etc/nginx/ssl/jds.local.key;  
    location / {  
        proxy_pass http://localhost:8080;  
    }  
}
```

Перезапустить службу nginx:

```
systemctl restart nginx
```

Проверить в браузере работу explain по https:

```
https://<адрес сервера jds-explain>:444
```

Взаимодействие JDS с сервисом pg-explain настраивается в конфигурационном файле приложения JDS appsettings.json - в свойстве PgExplainConfig.BaseAddress указать URL по которому доступен https-сервис pg-explain, например:

```
...  
"PgExplainConfig": {  
  "BaseAddress": "https://<адрес сервера explain>:444"  
},  
...
```

Адрес должен быть указан без закрывающего знака «/» - дробная черта.

Даже при том, что сервис explain работает на том же хосте, что и JDS, то все равно в свойстве BaseAddress нужно указывать внешний IP-адрес (не localhost), т.к. обращение к pg-explain идет не от JDS, а от браузера пользователя.

Зайти в web-интерфейс JDS по логину и паролю (по умолчанию, admin - secret), пункты меню Производительность – Анализ запросов. Перейти на вкладку Настройки, мини-вкладка Добавить. Ввести IP-адрес узла с наблюдаемой СУБД, порт (если он отличается от стандартного 5432) и отметить чекбоксы собираемой статистики, например, все 4. Сохранить.

На мини-вкладке «все» должен отобразиться добавленный узел.

## 9. РАЗДЕЛ JDS «МОНИТОРИНГ»

В разделе описывается подключение по SSL к системе Prometheus и экспортерам.

Сертификат и ключ сервера хранятся в нескольких экземплярах, так как каждый сервис работает под своим пользователем, для которого назначены уникальные права на файл ключа.

### 9.1. Система «Prometheus»

Для данного подключения понадобятся серверные ключ и сертификат, корневого сертификат.

Установить на сервер мониторинга компонент `jatobaX-prometheus` (подробно установка описана в документе «Руководство по настройке. Часть 22. Поддержка мониторинга СУБД».

Создать каталог для хранения сертификата сервера, ключа сервера и корневого сертификата:

```
mkdir /var/lib/certs/
```

Скопировать в этот каталог сертификат сервера, ключ сервера и корневой сертификат

Задать права на каталог

```
chown -R Prometheus /var/lib/certs/  
chmod 600 /var/lib/certs/prometheus.local.key
```

В файл сервиса `jatobaX_prometheus.service` добавить параметры `web.config.file` и `web.external-url`:

```
nano /lib/systemd/system/jatobaX_prometheus.service
```

```
...  
ExecStart=/usr/jatoba-X/bin/Prometheus \  
    --config.file ${CONF_FILE} \  
    --storage.tsdb.path ${STORAGE_TSDB_PATH} \  
    --web.console.templates=${WEB_CONSOLE_TEMPLATES} \  
    --web.console.libraries=${WEB_CONSOLE_LIBRARIES} \  
    --web.external-url=${WEB_EXTERNAL_URL}
```

```
--web.enable-lifecycle \  
--web.config.file=/usr/jatoba-X/monitoring/default/web-  
config.yml \  
--web.external-url=https://prometheus.local/  
...
```

Параметр `web.external-url` содержит доменное имя данного узла, соответствующее полю CN в сертификате (здесь - `prometheus.local`).

Содержимое файла `web-config.yml`:

```
tls_server_config:  
  cert_file: /var/lib/certs/prometheus.local.crt  
  key_file: /var/lib/certs/prometheus.local.key  
  client_ca_file: /var/lib/certs/root.crt  
  client_auth_type: "RequireAndVerifyClientCert"
```

В файл `/usr/jatoba-X/monitoring/default/prometheus.yml` добавить строки:

```
scrape_configs:  
  - job_name: "prometheus"  
    scheme: https  
    tls_config:  
      ca_file: /var/lib/certs/root.crt
```

Перечитать файл службы и перезапустить `prometheus`

```
systemctl daemon-reload  
systemctl restart jatobaX_prometheus.service  
systemctl status jatobaX_prometheus.service
```

В статусе службы (или в выводе команды `journalctl -xe`) должна быть строка «TLS is enabled».

С удаленного узла или локально можно проверить корректность настроек командой:

```
curl --cacert ./prometheus.local.crt  
https://prometheus.local:9090/api/v1/label/job/values
```

Для дальнейшего подключения сервиса JDS к мониторингу необходимо создать `pfх`-контейнер, к примеру, взяв клиентские сертификат и ключ пользователя `postgres`:

```
openssl pkcs12 -inkey postgres.key -in postgres.crt -certfile  
root.crt -export -out client.postgres.pfx
```

## 9.2. Экспортер «jatoba\*\_node\_exporter»

Для данного подключения понадобятся серверные ключ и сертификат, корневой сертификат.

Установить на целевой узел компонент jatobaX-node-exporter (подробно установка описана в документе «Руководство по настройке. Часть 22. Поддержка мониторинга СУБД». В данном примере описана установка на узле с prometheus, используются те же сертификаты.

Создать отдельный каталог для хранения сертификата и ключа:

```
mkdir /var/lib/certs_node/
```

Скопировать в этот каталог сертификат сервера, ключ сервера и корневой сертификат.

Задать права на каталог:

```
chown -R node_exporter_usr /var/lib/certs_node/  
chmod 600 /var/lib/certs_node/prometheus.local.key
```

В файл сервиса jatobaX\_node\_exporter.service добавить параметр web.config.file:

```
nano /lib/systemd/system/jatobaX_node_exporter.service
```

```
...  
ExecStart=/usr/jatoba-X/bin/node_exporter \  
    --web.config.file=/usr/jatoba-  
X/monitoring/default/node-config.yml  
...
```

В каталоге /usr/jatoba-X/monitoring/default создать файл node\_config.yml

```
nano /usr/jatoba-X/monitoring/default/node_config.yml
```

```
tls_server_config:
```

```
client_ca_file: /var/lib/certs_node/root.crt
cert_file: /var/lib/certs_node/prometheus.local.crt
key_file: /var/lib/certs_node/prometheus.local.key
client_auth_type: "RequireAndVerifyClientCert"
```

В файл /usr/jatoba-X/monitoring/default/prometheus.yml в раздел «# экспортер данных для Linux» добавить строки:

```
...
- job_name: "node-exporter"
  scheme: https
  tls_config:
    ca_file: /var/lib/certs/root.crt
    cert_file: /var/lib/certs/prometheus.local.crt
    key_file: /var/lib/certs/prometheus.local.key
    insecure_skip_verify: false
...
```

Перечитать и перезапустить службы:

```
systemctl daemon-reload
systemctl restart jatobaX_node_exporter.service
systemctl restart jatobaX_prometheus.service
```

### 9.3. Экспортер «jatoba\*\_postgres\_exporter»

Для данного подключения понадобятся серверные ключ и сертификат, клиентские ключ и сертификат, корневой сертификат.

Установить на целевой узел компонент jatobaX-postgres-exporter (подробно установка описана в документе «Руководство по настройке. Часть 22. Поддержка мониторинга СУБД»). В данном примере описана установка на узле с prometheus, используются те же сертификаты.

Создать отдельный каталог для хранения сертификата и ключа:

```
mkdir /var/lib/certs_postgres/
```

Скопировать в этот каталог сертификат сервера, ключ сервера, сертификат и ключ пользователя postgres\_exporter, а также корневой сертификат

Задать права на каталог с терминале ОС командой:



```
chown -R postgres_exporter_usr /var/lib/certs_postgres/  
chmod 600 /var/lib/certs_postgres/prometheus.local.key
```

В файл сервиса `jatobaX_postgres_exporter.service` добавить параметр `web.config.file`:

```
nano /lib/systemd/system/jatobaX_postgres_exporter.service
```

```
...  
ExecStart=/usr/jatoba-X/bin/postgres_exporter \  
        --web.config.file=/usr/jatoba-  
X/monitoring/default/postgres-config.yml  
...
```

В каталоге `/usr/jatoba-X/monitoring/default` создать файл `postgres-config.yml` командой:

```
nano /usr/jatoba-X/monitoring/default/postgres-config.yml
```

Установить параметры:

```
tls_server_config:  
  cert_file: /var/lib/certs_postgres/prometheus.local.crt  
  key_file: /var/lib/certs_postgres/prometheus.local.key
```

В файл `/usr/jatoba-X/monitoring/default/prometheus.yml` в раздел «# стандартный экспортер данных для PostgreSQL» добавить строки:

```
...  
- job_name: "postgresql"  
  scheme: https  
  tls_config:  
    ca_file: /var/lib/certs/root.crt  
    cert_file: /var/lib/certs/prometheus.local.crt  
    key_file: /var/lib/certs/prometheus.local.key  
    insecure_skip_verify: false  
...
```

В файле `/usr/jatoba-X/monitoring/default/postgres_exporter.yml` записать вместо IP-адреса доменное имя целевого узла и добавить параметры для подключения к СУБД:

```
DATA_SOURCE_NAME="postgresql://postgres_exporter:Password@prometheus.local:5432/postgres?sslmode=verify-full&\
```

```
sslrootcert=/var/lib/certs_postgres/root.crt&\  
sslcert=/var/lib/certs_postgres/postgres_exporter.crt&\  
sslkey=/var/lib/certs_postgres/postgres_exporter.key"
```

Обратите внимание, что в строке подключения между кавычками не должно быть посторонних символов, в том числе нечитаемых, к примеру символ табуляции \t.

Перечитать и перезапустить службы:

```
systemctl daemon-reload  
systemctl restart jatobaX_postgres_exporter.service  
systemctl restart jatobaX_prometheus.service
```

#### 9.4. Экспортера «jatoba\*\_sql\_exporter»

Для данного подключения понадобятся серверные ключ и сертификат, клиентские ключ и сертификат, корневой сертификат.

Установить на целевой узел компонент jatobaX-sql-exporter (подробно установка описана в документе «Руководство по настройке. Часть 22. Поддержка мониторинга СУБД». В данном примере описана установка на узле с prometheus, используются те же сертификаты.

Создать отдельный каталог для хранения сертификата и ключа:

```
mkdir /var/lib/certs_sql/
```

Скопировать в этот каталог сертификат и ключ сервера

Задать права на каталог

```
chown -R sql_exporter_usr /var/lib/certs_sql/  
chmod 600 /var/lib/certs_sql/prometheus.local.key
```

В файл sql\_exporter добавить значение параметра web\_config\_file:

```
nano /usr/jatoba-X/monitoring/default/sql_exporter
```

...

```
WEB_CONFIG_FILE=/usr/jatoba-X/monitoring/default/sql-config.yml  
...
```

В каталоге /usr/jatoba-X/monitoring/default создать файл sql-config.yml

```
nano /usr/jatoba-X/monitoring/default/sql-config.yml
```

```
tls_server_config:  
  client_ca_file: /var/lib/certs_sql/root.crt  
  cert_file: /var/lib/certs_sql/prometheus.local.crt  
  key_file: /var/lib/certs_sql/prometheus.local.key  
  client_auth_type: "RequireAndVerifyClientCert"
```

В файл /usr/jatoba-X/monitoring/default/prometheus.yml в раздел «# экспортер данных для SQL» добавить строки:

```
...  
- job_name: "sql-exporter"  
  scheme: https  
  tls_config:  
    ca_file: /var/lib/certs/root.crt  
    cert_file: /var/lib/certs/prometheus.local.crt  
    key_file: /var/lib/certs/prometheus.local.key  
    insecure_skip_verify: false  
...  
...
```

В файле /usr/jatoba-X/monitoring/default/sql\_exporter.yml записать вместо IP-адреса доменное имя целевого узла, а также параметры подключения: путь к корневому сертификату, серверному сертификату и серверному ключу:

```
data_source_name:  
'postgres://sql_exporter:Password@prometheus.local:5432/postgres?sslmode=verify-full&sslrootcert=/var/lib/certs_sql/root.crt&sslcert=/var/lib/certs_sql/sql_exporter.crt&sslkey=/var/lib/certs_sql/sql_exporter.key'
```

Перезапустить службы:

```
systemctl restart jatobaX_prometheus.service  
systemctl restart jatobaX_sql_exporter.service
```

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

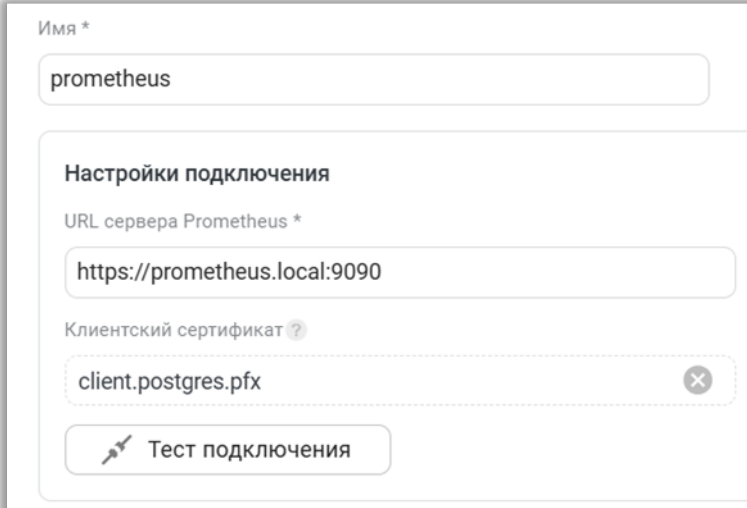
## 9.5. Настройка подключения в JDS

Добавить в JDS новый источник данных: Настройки - Источники данных - Добавить.

Вставить полный адрес сервера prometheus.

Вставить клиентский сертификат (pfx-файл), сформированный в разделе Prometheus

Тест подключения должен пройти успешно.



The screenshot shows a configuration window for a data source in JDS. At the top, there is a field labeled 'Имя \*' (Name \*) with the value 'prometheus'. Below this is a section titled 'Настройки подключения' (Connection settings). Inside this section, there is a field labeled 'URL сервера Prometheus \*' (Prometheus server URL \*) with the value 'https://prometheus.local:9090'. Below that is a field labeled 'Клиентский сертификат ?' (Client certificate ?) with the value 'client.postgres.pfx' and a small 'x' icon to its right. At the bottom of the section is a button labeled 'Тест подключения' (Test connection) with a small icon of a key and a plug.

Рисунок 9.1 – Настройки источника данных в JDS

В разделе «Мониторинг» должны отобразиться графики, автоматически созданные по метрикам из сервиса prometheus.

## 9.6. Настройка «Grafana»

Для подключения источника данных по TLS понадобятся клиентские сертификат и ключ.

В настройках источника данных указать имя сервера, если был указан IP-адрес. Сервер Prometheus должен быть доступен по этому адресу с сервера Grafana.

В категории TLS Settings включить пункт "TLS Server Authentication".

Вписать имя сервера (как указано в CN), вставить содержимое клиентского сертификата и ключа в соответствующие поля.

**TLS settings**  
Additional security measures that can be applied on top of authentication

☐ Add self-signed certificate ⓘ

☒ TLS Client Authentication ⓘ

ServerName \* ⓘ prometheus.local

Client Certificate \* ⓘ  
n2EfJnyapPSyA5gQakd4nB  
8ZI+vg10Hx4OmB8zSJJkEXYU7hHUUw95ZiuhHw  
HXaWfrkrI51SBz0UtljIN6H3WDv  
T50In0YJAuPeJPNTTrbc40RvdAx5Y/QLsTgde2N2d  
e83Jn+OLn2U33yA=  
-----END CERTIFICATE-----

Client Key \* ⓘ  
pykKx2kgAaJeJlbAhuSFsWV6X6QhiTj4uno9Jqt0L  
oqNnEhtgCT6WEWokOgLkWyY  
ZugcPymiNbet3840r0Ej+F5UqV+OY3cSZl6brpow  
EUn9K1SHvvv5ikQWhXFVZYH0  
cNcPgdbT0L9dl+Jrda2O9MiL  
-----END PRIVATE KEY-----

☐ Skip TLS certificate validation ⓘ

Рисунок 9.2 – Настройки TLS – соединения в «Grafana»  
В конце страницы проверить подключение кнопкой «Save & Test».

## 10. JA\_LOG. ЦЕНТРАЛИЗОВАННЫЙ СБОР ЗАПИСЕЙ СОБЫТИЙ В СУБД

Ja\_Log можно настроить 2 вида подключения по SSL: подключение jalog\_agent к jalog\_server и подключение jalog\_server к СУБД.

В данном примере на сервере JDS (jds.local) настраивается серверная часть ja\_Log.

В файлы /etc/hosts сервера и агента добавлены записи с доменными именами узлов (или записи настроены на DNS-сервере).

Генерируются серверные и клиентские сертификаты.

В процессе настройки состояние подключения можно контролировать командами:

```
tail -f /usr/jatoba-X/var/log/jalog/jalog_agent.log  
tail -f /usr/jatoba-X/var/log/jalog/jalog_server.log
```

### 10.1. ja\_log (сервер)

В файлах pg\_hba.conf и postgresql.conf СУБД записать параметры, аналогичные указанным в первом примечании.

Перезапустить СУБД командой:

```
systemctl restart jatoba-X.service
```

В файле /etc/hosts сделать запись для данного сервера:

```
127.0.0.1 localhost jds.local  
<общий IP> jds.local
```

Создать каталог для хранения сертификатов:

```
mkdir /var/lib/jatoba/certs
```

Скопировать в этот каталог все необходимые сертификаты.

Задать права на каталог:

```
chown -R postgres /var/lib/jatoba/certs  
chmod 600 /var/lib/jatoba/certs/ja_log.key
```

Так как сервис `jalog_server` запускается от имени пользователя `postgres`, можно использовать каталог с сертификатами для СУБД. В файл `/usr/jatoba-X/etc/jalog/jalog_server.yml` добавить параметры:

```
# Собственные параметры сервера
server:
  # listen_ip: 0.0.0.0    # IP-адрес, который прослушивает
сервер
  # listen_port: 10051    # Порт, который прослушивает сервер
# Параметры TLS
tls:
  cert_file: /var/lib/jatoba/certs/ja_log.crt
# Путь до сертификата
  key_file: /var/lib/jatoba/certs/ja_log.key
# Путь до файла ключа
  ca_file: /var/lib/jatoba/certs/root.crt
# Путь до файла ca
  crl_file: /var/lib/jatoba/certs/root.crl.pem
# Путь до файла crl
```

Перезапустить службу `jalog_server`:

```
systemctl restart jalog_server.service
```

## 10.2. ja\_log (агент)

Агент устанавливается на целевую СУБД.

В файлах `pg_hba.conf` и `postgresql.conf` целевой СУБД записать параметры, аналогичные указанным в первом примечании (подключение к СУБД по SSL)

Убедиться, что в файле `postgresql.conf` есть параметры:

```
log_destination = 'csvlog'
log_directory = 'log'
logging_collector = on
```

Перезапустить СУБД командой:

```
systemctl restart jatoba-X.service
```

Установить на сервер СУБД компонент `jatobaX-ja-log` (подробно установка описана в документе «Компонент `ja_Log`. Централизованный сбор записей событий СУБД», вариант для агента).

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

Создать каталог для хранения сертификатов:

```
mkdir /var/lib/jatoba/certs
```

Скопировать в этот каталог все необходимые сертификаты.

Задать права на каталог:

```
chown -R postgres /var/lib/jatoba/certs  
chmod 600 /var/lib/jatoba/certs/ja_log.key
```

В файл /usr/jatoba-X/etc/jalog/jalog\_agent.yml параметры SSL

```
jalog_agent:  
  hostname:                # Уникальное имя агента  
  ip:                      # Ip-адрес агента  
  # port: 22345             # Порт агента  
  # task_puller_frequency: 15 # Частота запроса задач у  
сервера, в секундах  
  # task_execution_frequency: 5 # Частота проверки лог-  
файлов, в секундах  
# Параметры сервера, с которым работает агент  
jalog_server:  
  # ip: 127.0.0.1           # Ip-адрес сервера  
  # port: 10051            # Порт сервера  
# Параметры TLS  
tls:  
  cert_file:/var/lib/jatoba/certs/ja_log_agent.crt  
# Путь до сертификата  
  key_file:/var/lib/jatoba/certs/ja_log_agent.key  
# Путь до файла ключа  
  ca_file:/var/lib/jatoba/certs/root.crt  
# Путь до файла ca  
  crl_file:/var/lib/jatoba/certs/root.crl.pem  
# Путь до файла crl
```

Запустить службу и добавить ее в автозагрузку:

```
systemctl start jalog_agent.service  
systemctl enable jalog_agent.service
```



### 10.3. TLS соединение между «jalog\_server» и служебной СУБД с БД ja\_log

Раздел описывает настройку TLS соединения между серверной частью «jalog\_server» компонента «ja\_Log» и служебной СУБД с БД «ja\_log».

TLS соединение между «jalog\_server» и служебной СУБД с БД ja\_log обеспечивается

- настройкой служебной СУБД для SSL/TLS соединения (см. п.3);
- настройкой серверной части «jalog\_server» для SSL/TLS соединения (см. п 10.1);

В этом случае соединение установится автоматически.

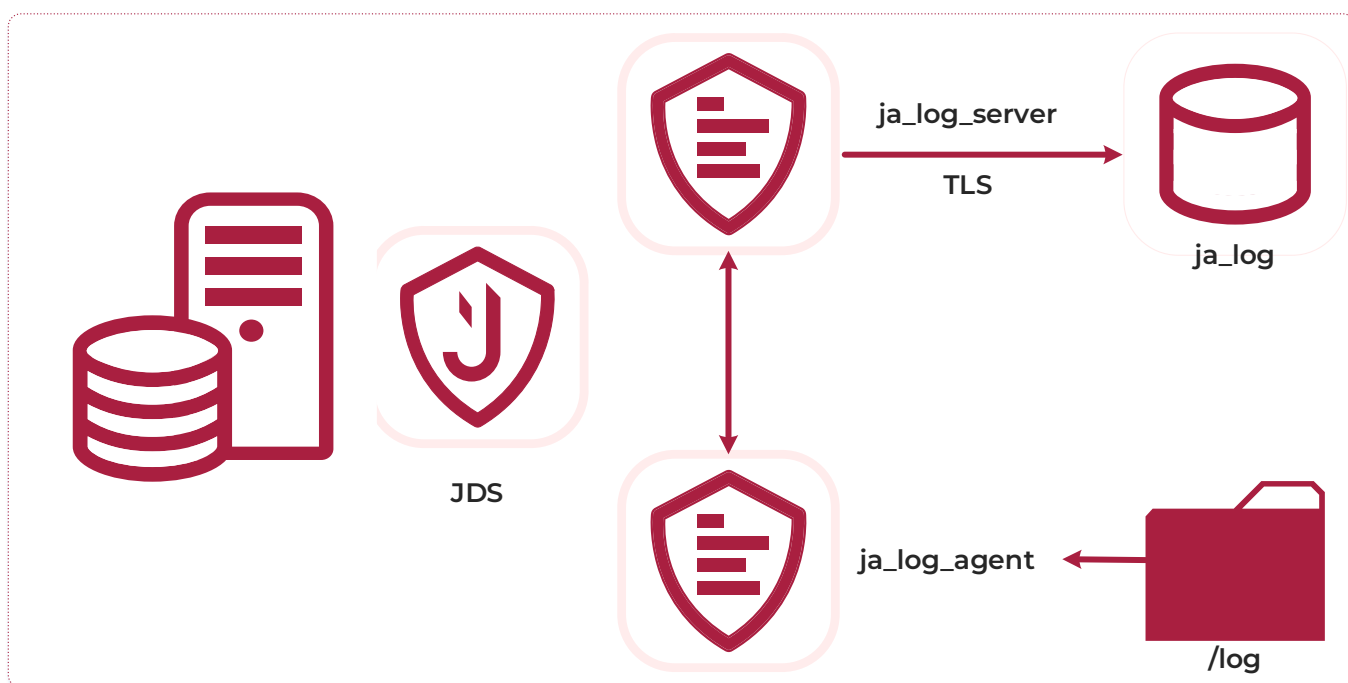


Рисунок 10.1 – Схема TLS соединения

### 10.4. JDS. Подключение ja\_Log для сбора событий

Перейти в Настройки - Цели - Добавить

Заполнить настройки для подключения к серверу ja\_log. В поле сертификат нужно указать бандл корневого и всех промежуточных сертификатов, созданный в разделе Создание и конвертация сертификатов. В качестве функционала выбрать «Список событий»

The image shows a software dialog box titled "jalog\_ssl". It contains the following fields and controls:

- Field: "Наименование цели \*" (Target Name), Value: "jalog\_ssl"
- Field: "Хост \*" (Host), Value: "jds.local"
- Field: "Порт \*" (Port), Value: "5432"
- Field: "Сертификат ?" (Certificate), Value: "root.crt-bundle"
- Field: "Функциональность" (Functionality), Value: "Список событий" (Event List)
- Buttons: "OK" and "Отменить" (Cancel)

Рисунок 10.2- Окно создание цели

Нажать на кнопку с «+» (Добавить подключение). Заполнить настройки, выбрать режим VerifyFull, способ аутентификации - «SSL-сертификат», в качестве сертификата добавить контейнер rfx указанного выше пользователя.

**Редактирование подключения** ✕

admin

Логин пользователя базы данных \*

postgres

Имя базы данных \*

ja\_log

Режим шифрования

VerifyFull ▾

Проверять сертификат хоста на отзыв

☒ Да ☐ Нет

Способ аутентификации

☒ Пароль ☐ SSL-сертификат

Сертификат ? \*

client.postgres.pfx

Пароль закрытого ключа

🔒

Тест подключения

Рисунок 10.3 – Окно редактирования подключения к цели

Кнопка Тест подключения должна отобразить уведомление об успешном подключении

В разделе «Аудит и отчетность» - «Список событий» выбрать созданную цель. Должны отобразиться события за выбранный период, собранные агентом ja\_log на целевом хосте.

## 11. JAPOOLER. БАЛАНСИРОВЩИК ПОДКЛЮЧЕНИЙ ПОЛЬЗОВАТЕЛЕЙ К СУБД

Провести стандартную установку компонента jarooler на все узлы кластера, следуя шагам из документа «Компонент jaPooler. Руководство по установке и эксплуатации».

### 11.1. jaPooler. Подключение по SSL

Данный способ предусматривает подключение всех пользователей СУБД через jarooler ТОЛЬКО ПО СЕРТИФИКАТУ.

Для использования SSL-подключения через JaPooler к СУБД Jatoba должна быть настроена и сама СУБД (см. руководство администратора п. 6.1.2. Настройка SSL).

В pg\_ident.conf для всех СУБД, которые должны работать по сертификатам, необходимо сделать сопоставление с пользователем jarooler - pgbouncer:

```
# MAPNAME          SYSTEM-USERNAME      PG-USERNAME
usermap  pgbouncer      user1
usermap  pgbouncer      user2
```

В pg\_hba.conf для всех пользователей СУБД, которые должны подключаться по сертификатам, должна быть соответствующая запись:

```
# TYPE  DATABASE  USER  ADDRESS  METHOD
hostssl all user1  all cert clientcert=verify-full map=usermap
hostssl all user2  all cert clientcert=verify-full map=usermap
```

Для сервера с jarooler и пользователя pgbouncer сформировать сертификаты по инструкциям из раздела «Создание сертификатов» (сертификат сервера должен содержать CN с именем сервера или его IP-адресом), скопировать их, а также корневой сертификат (root.crt) в каталог /usr/jatoba-X/etc/pgbouncer/

создать файл конфигурации jarooler - /usr/jatoba-X/etc/pgbouncer/pgbouncer.ini по примеру:

```
[databases]
postgres = host=astral.local dbname=postgres port=5432
strategy=always_rw
dbname1 = host=astral.local dbname=dbname1 port=5432
strategy=always_rw
* = host=astral.local port=5432 strategy=always_rw
```

```
[pgbouncer]
listen_port = 6432
listen_addr = *
logfile = /usr/jatoba-X/var/log/pgbouncer/pgbouncer.log
pidfile = /var/run/jatoba/pgbouncer.pid
pool_mode = transaction
max_client_conn = 1024
default_pool_size = 16
ignore_startup_parameters=idle_in_transaction_session_timeout,
extra_float_digits
```

```
client_tls_sslmode = verify-full
client_tls_ca_file = /usr/jatoba-X/etc/pgbouncer/root.crt
client_tls_key_file = /usr/jatoba-
X/etc/pgbouncer/astral.local.key
client_tls_cert_file = /usr/jatoba-
X/etc/pgbouncer/astral.local.crt
```

```
server_tls_sslmode = verify-full
server_tls_ca_file = /usr/jatoba-X/etc/pgbouncer/root.crt
server_tls_key_file = /usr/jatoba-X/etc/pgbouncer/pgbouncer.key
server_tls_cert_file = /usr/jatoba-
X/etc/pgbouncer/pgbouncer.crt
server_tls_ciphers = fast
admin_users = pgbouncer
auth_type = cert
auth_file = /usr/jatoba-X/etc/pgbouncer/userlist.txt
```

Необходимо обратить внимание, что `client_tls_key_file` и `client_tls_cert_file` – это сертификаты сервера, на котором работает `jarooler`, а `server_tls_key_file` и `server_tls_cert_file` – это сертификаты пользователя `pgbouncer`, он же должен быть указать в параметре `admin_user`.

Параметр `client_tls_sslmode` и `server_tls_sslmode` должны быть в значении `verify_full`, а `auth_type` – `cert`.

В файле `/usr/jatoba-X/etc/pgbouncer/userlits.txt` добавить строки со всеми пользователями, которые могут подключаться по сертификату, без указания пароля:

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

```
"user1" ""  
"user2" ""
```

Задать права на файлы конфигурации и сертификатов:

```
chown postgres: /usr/jatoba-X/etc/pgbouncer/*  
chmod 600 /usr/jatoba-X/etc/pgbouncer/*.crt  
chmod 600 /usr/jatoba-X/etc/pgbouncer/*.key
```

Чтобы применить изменения в файлах `pg_hba.conf` и `pg_ident.conf`, необходимо перезапустить СУБД или перечитать конфигурацию, подключившись через `psql`:

```
systemctl restart jatoba-X.service
```

или

```
select pg_reload_conf();
```

Запустить сервис `jaPooler`:

```
systemctl start pgbouncer.service
```

Проверить подключения к СУБД через `jaPooler` можно выполнив команды:

```
psql "dbname=postgres host=astral.local port=6432 user=user1  
sslmode=verify-full sslcert=/usr/jatoba-X/ssl/user1.crt  
sslkey=/usr/jatoba-X/ssl/user1.key sslrootcert=/usr/jatoba-  
X/ssl/root.crt"
```

или

```
psql -p 6432 -h astral.local -U user1 "dbname=postgres  
sslmode=verify-full sslcert=/usr/jatoba-X/ssl/user1.crt  
sslkey=/usr/jatoba-X/ssl/user1.key sslrootcert=/usr/jatoba-  
X/ssl/root.crt"
```

```

root@astral1:~# psql "dbname=postgres host=astral1.local port=6432 user=user1 sslmode=verify-full sslc
ert=/usr/jatoba-6/ssl/user1.crt sslkey=/usr/jatoba-6/ssl/user1.key sslrootcert=/usr/jatoba-6/ssl/root
.crt"
psql (16.6)
SSL-соединение (протокол: TLSv1.3, шифр: TLS_AES_256_GCM_SHA384, сжатие: выкл.)
Введите "help", чтобы получить справку.

postgres=> \q
root@astral1:~# psql -p 6432 -h astral1.local -U user1 "dbname=postgres sslmode=verify-full sslcert=/u
sr/jatoba-6/ssl/user1.crt sslkey=/usr/jatoba-6/ssl/user1.key sslrootcert=/usr/jatoba-6/ssl/root.crt"
psql (16.6)
SSL-соединение (протокол: TLSv1.3, шифр: TLS_AES_256_GCM_SHA384, сжатие: выкл.)
Введите "help", чтобы получить справку.

postgres=> \q

```

Рисунок 11.1 – Проверка SSL-соединения

При попытке подключиться без использования сертификата будет получена ошибка:

```

root@astral1:~# psql -U user1 -d postgres -p 6432 -h astral1.local
psql: ошибка: подключиться к серверу "astral1.local" (127.0.1.1), порту 6432 не удалось: ошибка SSL: t
lsv13 alert certificate required
root@astral1:~#

```

Рисунок 11.2 – Вывод с ошибкой

## 11.2. jaPooler. Подключение по SSL с паролем

Данный способ предусматривает возможность подключение пользователей СУБД через jaPooler по паролю без сертификата, так и по сертификату с паролем. Данный метод так же обеспечивает шифрование передачи данных, но при этом всегда требует пароль, не зависимо от наличия сертификата пользователя.

Для использования SSL-подключения через jaPooler к СУБД Jatoba должна быть настроена и сама СУБД (см. руководство администратора п. 6.1.2. Настройка SSL).

В pg\_ident.conf для всех СУБД, которые должны подключаться по сертификатам, необходимо сделать сопоставление с пользователем jaPooler – pgbouncer. Пользователей, которые будут подключаться только по паролю, указывать не обязательно.

#	MAPNAME	SYSTEM-USERNAME	PG-USERNAME
	usermap	pgbouncer	user1
	usermap	pgbouncer	user2

В pg\_hba.conf для всех пользователей СУБД, которые должны подключаться по сертификатам, должна быть соответствующие записи с методом аутентификации по сертификату и паролю:

```
# TYPE DATABASE USER ADDRESS  
METHOD  
hostssl all user1 all cert clientcert=verify-full map=usermap  
hostssl all user2 all cert clientcert=verify-full map=usermap  
host all user1 all md5  
host all user2 all md5
```

В postgresql.conf СУБД указать метод шифрования пароля md5, возможно также использование SCRAM-SHA-256.

```
password_encryption = md5
```

Для сервера с jarooler и пользователя pgbouncer сформировать сертификаты по инструкциям из раздела «Создание сертификатов» (сертификат сервера должен содержать CN с именем сервера или его IP-адресом), скопировать их, а также корневой сертификат (root.crt) в каталог /usr/jatoba-X/etc/pgbouncer/

Создать файл конфигурации jarooler - /usr/jatoba-X/etc/pgbouncer/pgbouncer.ini по примеру:

```
[databases]  
postgres = host=astral.local dbname=postgres port=5432  
strategy=always_rw  
dbname1 = host=astral.local dbname=dbname1 port=5432  
strategy=always_rw  
* = host=astral.local port=5432 strategy=always_rw
```

```
[pgbouncer]  
listen_port = 6432  
listen_addr = *  
logfile = /usr/jatoba-X/var/log/pgbouncer/pgbouncer.log  
pidfile = /var/run/jatoba/pgbouncer.pid  
pool_mode = transaction  
max_client_conn = 1024  
default_pool_size = 16  
ignore_startup_parameters=idle_in_transaction_session_timeout,  
extra_float_digits
```



```
client_tls_sslmode = require
client_tls_ca_file = /usr/jatoba-X/etc/pgbouncer/root.crt
client_tls_key_file = /usr/jatoba-
X/etc/pgbouncer/astral.local.key
client_tls_cert_file = /usr/jatoba-
X/etc/pgbouncer/astral.local.crt
```

```
server_tls_sslmode = verify-full
server_tls_ca_file = /usr/jatoba-X/etc/pgbouncer/root.crt
server_tls_key_file = /usr/jatoba-X/etc/pgbouncer/pgbouncer.key
server_tls_cert_file = /usr/jatoba-
X/etc/pgbouncer/pgbouncer.crt
server_tls_ciphers = fast
admin_users = pgbouncer
auth_type = md5
auth_file = /usr/jatoba-X/etc/pgbouncer/userlist.txt
```

Необходимо обратить внимание, что `client_tls_key_file` и `client_tls_cert_file` – это сертификаты сервера, на котором работает JaPooler, а `server_tls_key_file` и `server_tls_cert_file` – это сертификаты пользователя pgbouncer, он же должен быть указать в параметре `admin_user`.

Параметры: `client_tls_sslmode` должен быть в значении `require`, `server_tls_sslmode` должен быть в значении `verify_full`, а `auth_type` – `md5`, возможно также использование SCRAM-SHA-256.

В файле `/usr/jatoba-X/etc/pgbouncer/userlist.txt` для всех пользователей СУБД, которые могут подключаться через JaPooler, указать имя и пароль. Пароль можно указывать как в открытом виде, так и в виде хэша md5. Принятый формат пароля, защищённого MD5: "md5" + md5(password + username).

Например, для пользователя `user1` с паролем `sql` хэш пароля будет следующим: `md5d5f86855b37ab02281443ffc4d5070a8`. Так же можно использовать SCRAM-SHA-256.

```
"user1" "md5d5f86855b37ab02281443ffc4d5070a8"  
"user2" "md5e61f39e52ac9b551616f951594f31c0e"  
"user3" "SCRAM-SHA-  
256$4096:70yC1VG7nzPxHiu+JNFIPg==$IXKFrCJDF5cu1GUkJaQ/FYd300MeN  
RAyOqTduA/NzWA=:xqsKCIfpT+qhoZNHGySdp0lsXOvzVg1gAPckHjOYw7s="  
"user4" "123456Aa"  
"pgbouncer" "12345"
```

Сгенерировать MD5 хэш можно на сайте <https://www.md5hashgenerator.com/> или взять из СУБД с помощью запроса, в том числе для паролей с хэш SCRAM:

```
select '||rolname||' '||'||rolpassword||' from pg_authid  
where rolpassword is not NULL;
```

```
postgres=# select '||rolname||' '||'||rolpassword||' from pg_authid where rolpassword is not N  
ULL;  
?column?  
-----  
"postgres" "md501c0ed25a30b871724047a1497b55b44"  
"jds" "md530d759bb0891427a5dae8b65992d6ab3"  
"user1" "md5d5f86855b37ab02281443ffc4d5070a8"  
"japooler" "md59dff0e3641abcf5ec18bd3b454f51e57"  
"user2" "md5e61f39e52ac9b551616f951594f31c0e"  
"user3" "SCRAM-SHA-256$4096:70yC1VG7nzPxHiu+JNFIPg==$IXKFrCJDF5cu1GUkJaQ/FYd300MeNRAyOqTduA/NzWA=:xq  
sKCIfpT+qhoZNHGySdp0lsXOvzVg1gAPckHjOYw7s="  
(6 строк)
```

Рисунок 11.3 – Генерация MD5 хэш

Задать права на файлы конфигурации и сертификатов:

```
chown postgres: /usr/jatoba-X/etc/pgbouncer/*  
chmod 600 /usr/jatoba-X/etc/pgbouncer/*.crt  
chmod 600 /usr/jatoba-X/etc/pgbouncer/*.key
```

Чтобы применить изменения в файлах `pg_hba.conf` и `pg_ident.conf`, необходимо перезапустить СУБД или перечитать конфигурацию, подключившись через `psql`:

```
systemctl restart jatoba-X.service
```

или

```
select pg_reload_conf();
```

Перезапустить сервис `japooler`:

```
systemct restart pgbouncer.service
```

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

Проверить подключения к СУБД через jaPooler можно выполнив команды:

```
# psql "dbname=postgres host=astral.local port=6432 user=user1
sslmode=verify-full sslcert=/usr/jatoba-X/ssl/user1.crt
sslkey=/usr/jatoba-X/ssl/user1.key sslrootcert=/usr/jatoba-
X/ssl/root.crt"
# psql -p 6432 -h astral.local -U user1 "dbname=postgres
sslmode=verify-full sslcert=/usr/jatoba-X/ssl/user1.crt
sslkey=/usr/jatoba-X/ssl/user1.key sslrootcert=/usr/jatoba-
X/ssl/root.crt"
# psql -p 6432 -h astral.local -U user1 -d postgres
```

```
root@astral:~# psql "dbname=postgres host=astral.local port=6432 user=user1 sslmode=verify-full sslc
ert=/usr/jatoba-6/ssl/user1.crt sslkey=/usr/jatoba-6/ssl/user1.key sslrootcert=/usr/jatoba-6/ssl/root
.crt"
Пароль пользователя user1:
psql (16.6)
SSL-соединение (протокол: TLSv1.3, шифр: TLS_AES_256_GCM_SHA384, сжатие: выкл.)
Введите "help", чтобы получить справку.

postgres=> \q
root@astral:~# psql -p 6432 -h astral.local -U user1 "dbname=postgres sslmode=verify-full sslcert=/u
sr/jatoba-6/ssl/user1.crt sslkey=/usr/jatoba-6/ssl/user1.key sslrootcert=/usr/jatoba-6/ssl/root.crt"
Пароль пользователя user1:
psql (16.6)
SSL-соединение (протокол: TLSv1.3, шифр: TLS_AES_256_GCM_SHA384, сжатие: выкл.)
Введите "help", чтобы получить справку.

postgres=> \q
root@astral:~# psql -p 6432 -h astral.local -U user1 -d postgres
Пароль пользователя user1:
psql (16.6)
SSL-соединение (протокол: TLSv1.3, шифр: TLS_AES_256_GCM_SHA384, сжатие: выкл.)
Введите "help", чтобы получить справку.

postgres=> █
```

Рисунок 11.4 – Вывод проверки подключения

При попытке подключиться пользователем, не указанным в userlist.txt, получим ошибку:

```
root@astral:~# psql -p 6432 -h astral.local -U user5 -d postgres
Пароль пользователя user5:
psql: ошибка: подключиться к серверу "astral.local" (127.0.1.1), порту 6432 не удалось: FATAL: password au
thentication failed
подключиться к серверу "astral.local" (127.0.1.1), порту 6432 не удалось: FATAL: SSL required
```

Рисунок 11.5 – Вывод ошибки подключения

## 12. JA\_SYNC\_LDAP. СИНХРОНИЗАЦИЯ УЧЕТНЫХ ЗАПИСЕЙ СЛУЖБ КАТАЛОГОВ И СУБД

Все процедуры раздела будут проводиться на сервере с расширением ja\_sync\_ldap.

Расширение устанавливается и настраивается по инструкции "Руководство по настройке. Часть 8. Синхронизация учетных записей служб каталогов и СУБД. Компонент «ja\_Sync\_LDAP».

Подключение рассматривается на примере службы каталогов Samba.

Для работы расширения по LDAPS требуется корневой сертификат (к примеру, rootCA.crt), сгенерированный в службе сертификатов контроллера домена. Файл необходимо скопировать в каталог на сервере с ja\_sync\_ldap, в данном случае /usr/share/ca-certificates/.

Данный сертификат нужно установить в системе:

```
cp ~/root.crt /usr/share/ca-certificates/  
update-ca-certificates
```

или для РедОС:

```
cp ~/rootCA.crt /etc/pki/ca-trust/source/anchors/  
update-ca-trust
```

В файле /etc/hosts сервера с ja\_sync\_ldap (или в DNS-сервере) должно быть прописано соответствие адреса и FQDN для сервера DC. Синхронизация будет происходить по этому имени.

```
127.0.0.1 localhost  
127.0.1.1 ubuntu  
172.19.19.31 dc.example.local
```

Команда ping dc.example.local должна возвращать успешный ответ.

Чтобы добавить поддержку SSL-соединения в существующий профиль синхронизации, используется команда:

```
SELECT ja_sync_ldap.set_ssl_profile(<Profile_ID>, true);
```

Для указания пути к сертификату используется команда

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

```
SELECT ja_sync_ldap.set_ca_cert_profile(<Profile_ID>, '<
/usr/share/ca-certificates/rootCA.crt>');
```

Просмотреть профили SQL-командой:

```
SELECT ja_sync_ldap.get_sync_profiles();
```

```
postgres=# select ja_sync_ldap.get_sync_profiles();
get_sync_profiles
-----
(1,samba,dc.example.local,636,"cn=administrator,cn=users,dc=example,dc=local",MkBXdhB5UFk=samba,t,/usr/share/ca-certificates/root_dc_local.crt)
(1 строка)
```

Рисунок 12.1 – Вывод профилей синхронизации

Аналогичные настройки можно провести в интерфейсе JDS.

Добавить цель с функциональностью «LDAP-синхронизация» и пользователя для цели.

The screenshot shows a window titled 'localhost' with a close button. It contains the following fields and controls:

- 'Наименование цели \*': Text input with 'LDAP' entered.
- 'Хост \*': Text input with '127.0.0.1' entered.
- 'Порт \*': Text input with '5432' entered.
- 'Сертификат ?': A dashed box with the text 'Выбрать или перетащить файл сюда'.
- 'Функциональность': A dropdown menu with 'LDAP синхронизация' selected.
- Buttons: 'OK' (blue) and 'Отменить' (grey).

Рисунок 12.2 – Окно создания цели

В разделе «LDAP-синхронизация» добавить новый профиль. В данном случае не нужен pfx-бандл.

The screenshot shows a window titled "Редактирование профиля" (Edit Profile) with a close button (X) in the top right corner. Below the title, it says "ID: 1". The main form contains the following fields and controls:

- Наименование профиля \*** (Profile Name): A text input field containing "samba".
- Тип LDAP-сервера \*** (LDAP Server Type): A group of four buttons: "Active Directory", "ALD Pro", "FreelPA", and "Samba". The "Samba" button is currently selected.
- Использовать LDAPS** (Use LDAPS): A toggle switch that is currently turned on (green).
- Путь к сертификату (для Unix-подобных систем)** (Certificate path): A text input field containing "/usr/share/ca-certificates/root\_dc\_local.crt".
- Хост \*** (Host): A text input field containing "dc.example.local".
- Порт \*** (Port): A text input field containing "636".
- Логин \*** (Login): A text input field containing "cn=administrator,cn=users,dc=example,dc=local".
- Пароль \*** (Password): A password input field with masked characters (dots).
- At the bottom, there are two buttons: "OK" (blue) and "Отменить" (grey).

Рисунок 12.3 – Окно редактирования профиля синхронизации  
Для данного профиля добавить маппинг и нажать Синхронизация

The screenshot shows a window titled "Редактирование маппинга" (Edit Mapping) with a close button (X) in the top right corner. Below the title, it says "ID: 1, Профиль: «samba»". The main form contains the following fields and controls:

- Роль БД \*** (DB Role): A text input field containing "db\_users\_smb".
- Группа LDAP \*** (LDAP Group): A text input field containing "cn=guser1,cn=users,dc=example,dc=local".
- Атрибут LDAP \*** (LDAP Attribute): A text input field containing "sAMAccountName".
- Объектный класс (Object class)** (Object class): A text input field containing "person".
- База поиска (Base DN)** (Search Base): A text input field containing "dc=example,dc=local".
- At the bottom, there are two buttons: "OK" (blue) and "Отменить" (grey).

Рисунок 12.4 - Окно редактирования маппинга

### 13. SSL АУТЕНТИФИКАЦИЯ В КОНТЕЙНЕРЕ

Существует два способа настройки и использования SSL аутентификации между клиентом (клиентской частью psql) и СУБД «Jatoba» в контейнерном исполнении:

- Запуск контейнера через docker compose с SSL аутентификацией (п.п. 13.1);
- Запуск контейнера с SSL аутентификацией (п.п. 13.2).

SSL аутентификация используется между хостом с установленной клиентской частью, т.е. с утилитой psql и СУБД «Jatoba» в контейнерном исполнении.

В дальнейшем описании будут использованы обозначения:

- СУБД «Jatoba» в контейнерном исполнении BM1 (docker);
- хост с установленной клиентской частью – BM 2 (psql).

#### 13.1. Запуск контейнера через docker compose с SSL аутентификацией

Настройка и запуск контейнера через docker compose с SSL аутентификацией подразумевает следующие шаги:

##### BM1 (docker)

- Импортировать образ контейнера на BM1 (docker):

```
cd container  
chmod +x *.sh  
./setup.sh
```

- Изменить права доступа на скрипт по созданию сертификатов:

```
chmod a+x generate-ssl-certs.sh
```

- Выполнить скрипт по созданию ssl-сертификатов:

```
./generate-ssl-certs.sh
```

В результате, текущей директории создан каталог «ssl», будут сформированы сертификаты и выведено сообщение:

```
CA Certificate Jatoba Root CA has been successfully generated.  
Certificate server has been successfully generated.
```

```
Certificate client has been successfully generated.
```

— Перейти в каталог standalone и раскомментировать строку в docker-compose.yml:

```
nano docker-compose.yml  
- ${SSL_CERTS_PATH}:<путь к каталогу ssl>/ssl/certs
```

— Открыть файл переменных окружения .env и добавить следующие параметры:

```
SSL_MODE=true  
SSL_CERTS_PATH=<путь к папке ssl>/ssl/server
```

— Выполнить запуск СУБД:

```
docker-compose up -d
```

### **Локально - на VM1 (docker), где запущен docker**

— Добавить SN имя, используемое в server сертификате (по умолчанию jatobadb) в /etc/hosts:

```
sudo bash -c "echo '127.0.0.1 jatobadb' >> /etc/hosts"
```

— Задать следующую переменную:

```
export SSL_DIR=<путь к каталогу ssl>/ssl/client  
chmod 600 $SSL_DIR/client.key
```

— Установить клиент psql и запустить его используя переменные окружения с SSL-аутентификацией:

```
PGSSLMODE=verify-full \  
PGSSLROOTCERT=$SSL_DIR/root.crt \  
PGSSLCERT=$SSL_DIR/client.crt \  
PGSSLKEY=$SSL_DIR/client.key \  
psql -U postgres -p 54321 -h jatobadb
```

Подключение выполнено успешно.

### **Подключение с удаленного хоста VM2 (psql)**



- Скопировать каталог ssl/client на BM2
- Добавить SN имя используемое в server сертификате (по умолчанию jatobadb)

в /etc/hosts:

```
sudo bash -c "echo '<ip адрес VM, где развернут сервис jatobadb  
в докере> jatobadb' >> /etc/hosts";
```

- Задать следующую переменную:

```
export SSL_DIR=<путь к каталогу ssl>/ssl/client  
chmod 600 $SSL_DIR/client.key
```

- Запустить psql, используя переменные окружения с SSL-аутентификацией:

```
PGSSLMODE=verify-full \  
PGSSLROOTCERT=$SSL_DIR/root.crt \  
PGSSLCERT=$SSL_DIR/client.crt \  
PGSSLKEY=$SSL_DIR/client.key \  
psql -U postgres -p 54321 -h jatobadb
```

Подключение выполнено успешно.

- Получить pid номер соединения и проверить тип соединения:

```
SELECT pg_backend_pid();  
SELECT * from pg_stat_ssl where pid=<pid номер из предыдущего  
запроса>;
```

Вывод:

```
ssl=t  
version="TLSv1.3"  
cipher="TLS_AES_256_GCM_SHA384"  
client_dn="/CN=postgres"  
client_serial=< not null >  
issuer_dn="/CN=Jatoba Root CA"
```

### 13.2. Запуск контейнера с SSL аутентификацией

Настройка и запуск контейнера с SSL аутентификацией подразумевает следующие шаги:

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

- Импортировать образ контейнера на ВМ1 (docker):

```
cd container  
chmod +x *.sh  
./setup.sh
```

- Изменить права доступа на скрипт по созданию сертификатов:

```
chmod a+x generate-ssl-certs.sh
```

В результате, текущей директории создан каталог «ssl», будут сформированы сертификаты и выведено сообщение:

```
CA Certificate Jatoba Root CA has been successfully generated.  
Certificate server has been successfully generated.  
Certificate client has been successfully generated.
```

- Открыть файл переменных окружения .env и добавить следующие параметры:

```
SSL_MODE=true  
SSL_CERTS_PATH=<путь к каталогу ssl>/ssl/server
```

- Выполнить:

```
./run.sh; ./log.sh
```

СУБД успешно запущена

**Локально - на ВМ1 (docker), где запущен docker**

- Добавить SN имя используемое в server сертификате (по умолчанию jatobadb) в /etc/hosts:

```
sudo bash -c "echo '127.0.0.1 jatobadb' >> /etc/hosts"
```

- Задать следующую переменную:

```
export SSL_DIR=<путь к каталогу ssl>/ssl/client  
chmod 600 $SSL_DIR/client.key
```

- Установить клиент psql и запустить его используя переменные окружения с SSL-аутентификацией:

```
PGSSLMODE=verify-full \  
PGSSLROOTCERT=$SSL_DIR/root.crt \  
PGSSLCERT=$SSL_DIR/client.crt \  
PGSSLKEY=$SSL_DIR/client.key \  
psql -U postgres -p 54321 -h jatobadb
```

— Получить pid номер соединения и проверить тип соединения:

```
SELECT pg_backend_pid();  
SELECT * from pg_stat_ssl where pid=<pid номер из предыдущего  
запроса>;
```

При правильной настройке будет вывод:

```
ssl=t  
version="TLSv1.3"  
cipher="TLS_AES_256_GCM_SHA384"  
client_dn="/CN=postgres"  
client_serial=< not null >  
issuer_dn="/CN=Jatoba Root CA"
```

### Подключение с удаленного хоста BM2 (psql)

— Скопировать каталог ssl/client на BM2

— Добавить SN имя, используемое в server сертификате (по умолчанию jatobadb)

в /etc/hosts:

```
sudo bash -c "echo '<ip адрес VM, где развернут сервис jatobadb  
в докере> jatobadb' >> /etc/hosts"
```

— Задать следующую переменную:

```
export SSL_DIR=<путь к каталогу ssl>/ssl/client  
chmod 600 $SSL_DIR/client.key
```

— Запустить psql, используя переменные окружения с SSL-аутентификацией:

```
PGSSLMODE=verify-full \  
PGSSLROOTCERT=$SSL_DIR/root.crt \  
PGSSLCERT=$SSL_DIR/client.crt \  
PGSSLKEY=$SSL_DIR/client.key \  
psql -U postgres -p 54321 -h jatobadb
```

Подключение выполнено успешно.

— Получить pid номер соединения и проверить тип соединения:

```
SELECT pg_backend_pid();  
SELECT * from pg_stat_ssl where pid=<pid номер из предыдущего  
запроса>;
```

При корректной настройке SSL-соединения будет следующий вывод:

```
ssl=t  
version="TLSv1.3"  
cipher="TLS_AES_256_GCM_SHA384"  
client_dn="/CN=postgres"  
client_serial=< not null >  
issuer_dn="/CN=Jatoba Root CA"
```

## 14. ПРИМЕР НАСТРОЙКИ SSL-СОЕДИНЕНИЙ JDS

В приведенном ниже описании приведен пример создания цели (Target) с подключением по SSL/TLS.



Информация, приведенная в разделе, носит рекомендательный характер и может использоваться в качестве примера или в учебных целях.

В ИТ - инфраструктуре рекомендуется использовать сертификаты и ключи, выпущенные Удостоверяющими центрами, программное обеспечение которых имеет сертификат соответствия ФСБ России.

### 14.1. Требуемое программное обеспечение

Для формирования сертификатов требуется установить версию OpenSSL не ниже 3.\*.

### 14.2. Пользователи

В целевой СУБД создать тестового пользователя СУБД SQL-командой:

```
CREATE ROLE "test_user" login password 'P@ssword';
```

### 14.3. Каталог хранения сертификатов

Созданные сертификаты и конфигурационные файлы целесообразнее хранить в отдельном каталоге. В рассматриваемом примере используется каталог /usr/share/jds/cert.

Создается каталог командой:

```
sudo mkdir /usr/share/jds/cert
```

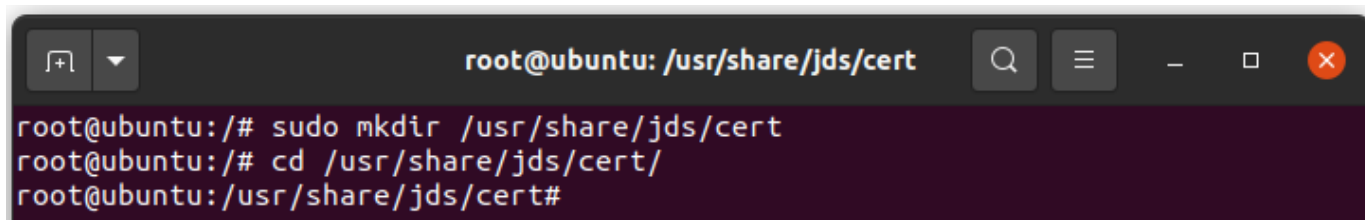


Рисунок 14.1 – Команда создания каталога

### 14.4. Создание конфигурационных файлов OpenSSL

#### 14.4.1. Конфигурационный файл OpenSSL корневого ЦС

Создать конфигурационный файл OpenSSL корневого ЦС командой:

```
gedit _openssl.root.conf
```

Вставить в него следующие параметры:

```
[req]
distinguished_name=dn
[ dn ]
[ v3_ca ]
basicConstraints=CA:TRUE,pathlen:1
```

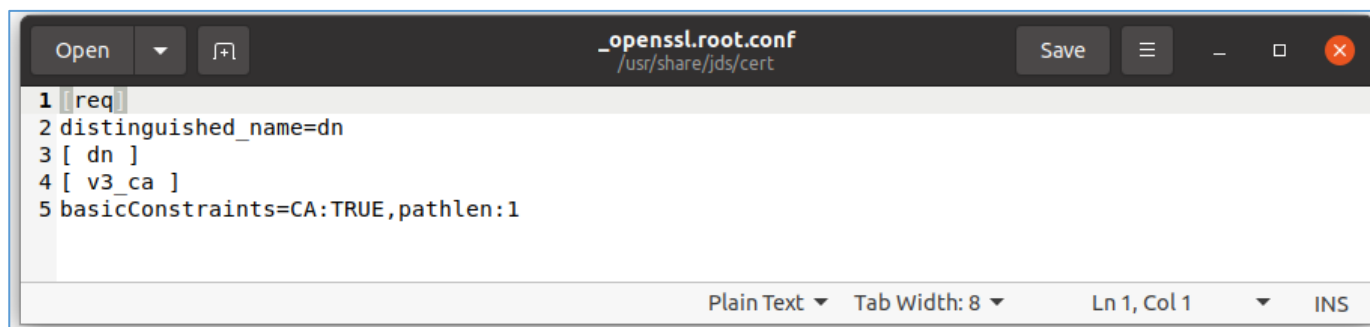


Рисунок 14.2 – Содержание конфигурационного файла OpenSSL корневого ЦС

#### 14.4.2. Конфигурационный файл OpenSSL промежуточного ЦС

Создать конфигурационный файл OpenSSL промежуточного ЦС командой:

```
gedit _openssl.intermediate.conf
```

Вставить в него следующие параметры:

```
[req]
distinguished_name=dn
[ dn ]
[ v3_ca ]
basicConstraints=CA:TRUE,pathlen:0
```

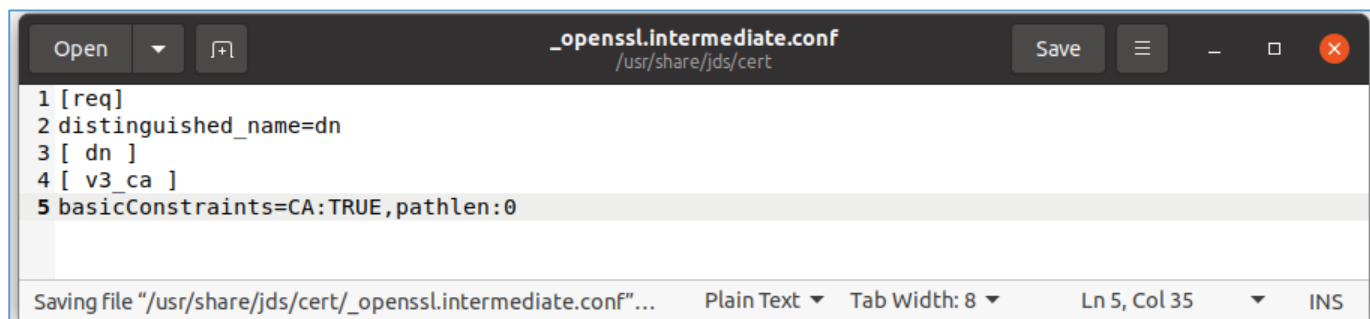


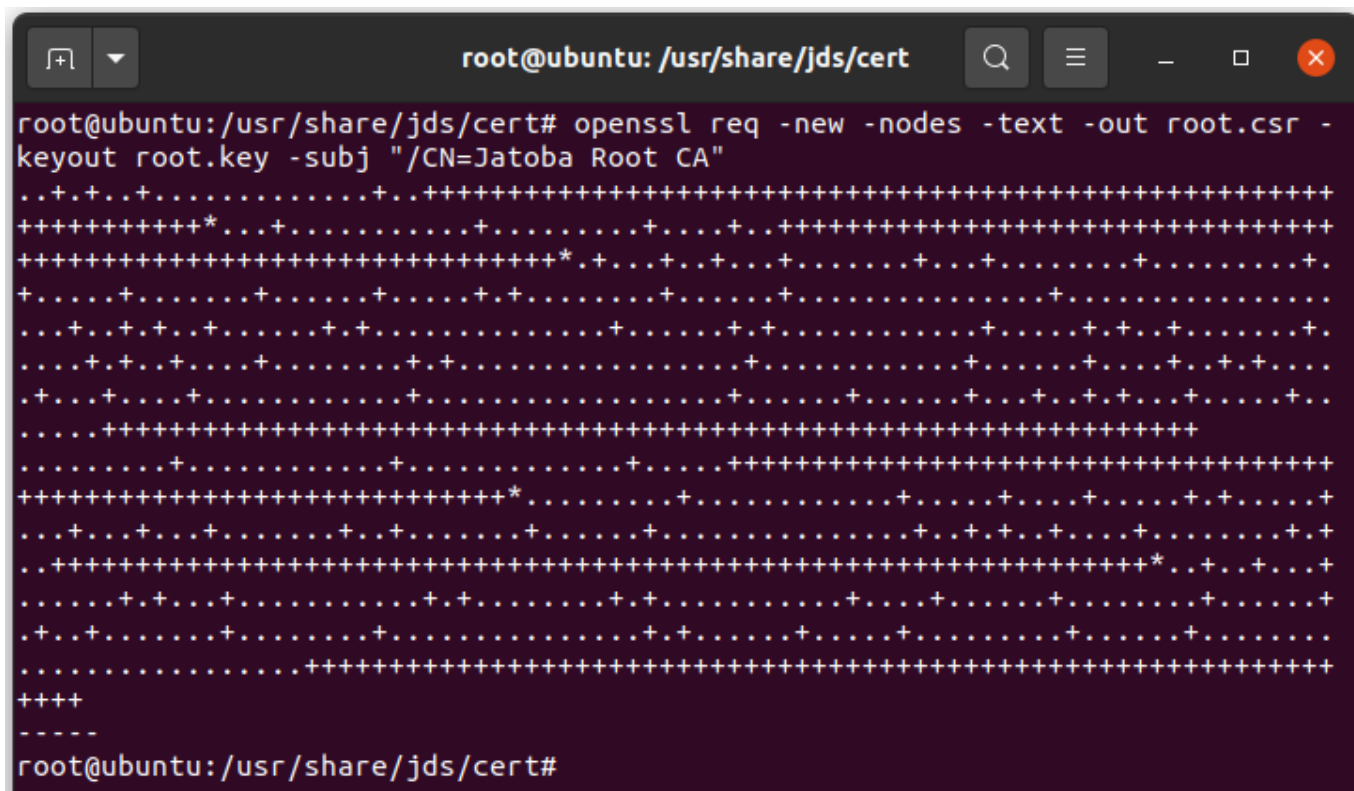
Рисунок 14.3 - Содержание конфигурационного файла OpenSSL промежуточного ЦС

## 14.5. Создание самоподписанных сертификатов

### 14.5.1. Самоподписанный сертификат корневого ЦС (Root CA)

Создать ключ корневого ЦС командой:

```
#openssl req -new -nodes -text -out root.csr -keyout root.key -  
subj "/CN=Jatoba Root CA"
```

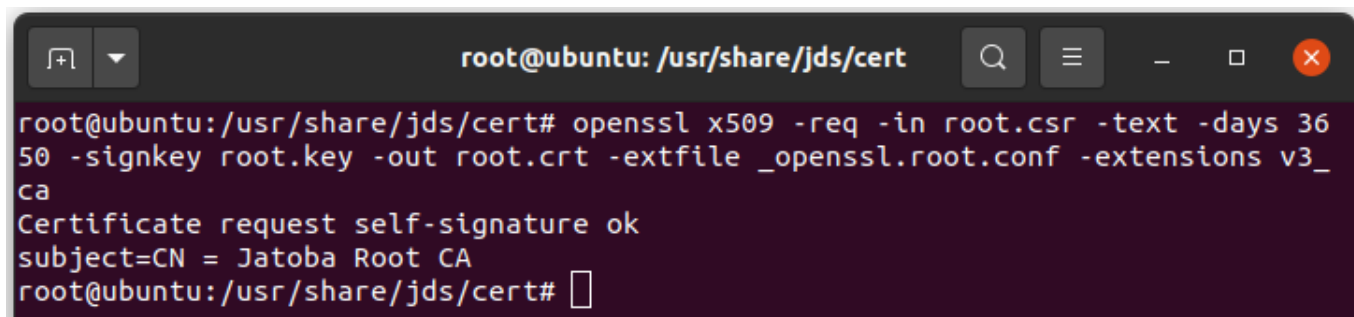
A screenshot of a terminal window with a dark background. The title bar shows 'root@ubuntu: /usr/share/jds/cert'. The terminal text shows the command 'openssl req -new -nodes -text -out root.csr -keyout root.key -subj "/CN=Jatoba Root CA"' being executed. The output consists of a series of '+' characters forming a decorative border around the command prompt, followed by the prompt 'root@ubuntu: /usr/share/jds/cert#'.

```
root@ubuntu: /usr/share/jds/cert# openssl req -new -nodes -text -out root.csr -  
keyout root.key -subj "/CN=Jatoba Root CA"  
.....+.....  
+++++*.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
+++++*.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
...+...+...+...+...+...+...+...+...+...+...+...+...+...+...+...+...  
...+...+...+...+...+...+...+...+...+...+...+...+...+...+...+...+...  
+...+...+...+...+...+...+...+...+...+...+...+...+...+...+...+...  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
+++++*.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
...+...+...+...+...+...+...+...+...+...+...+...+...+...+...+...+...  
..+++++*.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
.....+...+...+...+...+...+...+...+...+...+...+...+...+...+...+...+...  
+...+...+...+...+...+...+...+...+...+...+...+...+...+...+...+...  
.....+.....+.....+.....+.....+.....+.....+.....+.....+.....  
++++  
-----  
root@ubuntu: /usr/share/jds/cert#
```

Рисунок 14.4 – Команда создания ключа корневого ЦС

Создать сертификат корневого ЦС командой:

```
#openssl x509 -req -in root.csr -text -days 3650 -signkey  
root.key -out root.crt -extfile _openssl.root.conf -extensions  
v3_ca
```

A screenshot of a terminal window with a dark background. The title bar shows 'root@ubuntu: /usr/share/jds/cert'. The terminal text shows the command 'openssl x509 -req -in root.csr -text -days 3650 -signkey root.key -out root.crt -extfile \_openssl.root.conf -extensions v3\_ca' being executed. The output shows 'Certificate request self-signature ok' and 'subject=CN = Jatoba Root CA', followed by the prompt 'root@ubuntu: /usr/share/jds/cert#'.

```
root@ubuntu: /usr/share/jds/cert# openssl x509 -req -in root.csr -text -days 36  
50 -signkey root.key -out root.crt -extfile _openssl.root.conf -extensions v3_  
ca  
Certificate request self-signature ok  
subject=CN = Jatoba Root CA  
root@ubuntu: /usr/share/jds/cert#
```

Рисунок 14.5 – Команда создания сертификата корневого ЦС

### 14.5.2. Самоподписанный сертификат промежуточного ЦС (Root CA)

Создать ключ промежуточного ЦС командой:

```
#openssl req -new -nodes -text -out intermediate.csr -keyout  
intermediate.key -subj "/CN=Jatoba Intermediate CA"
```

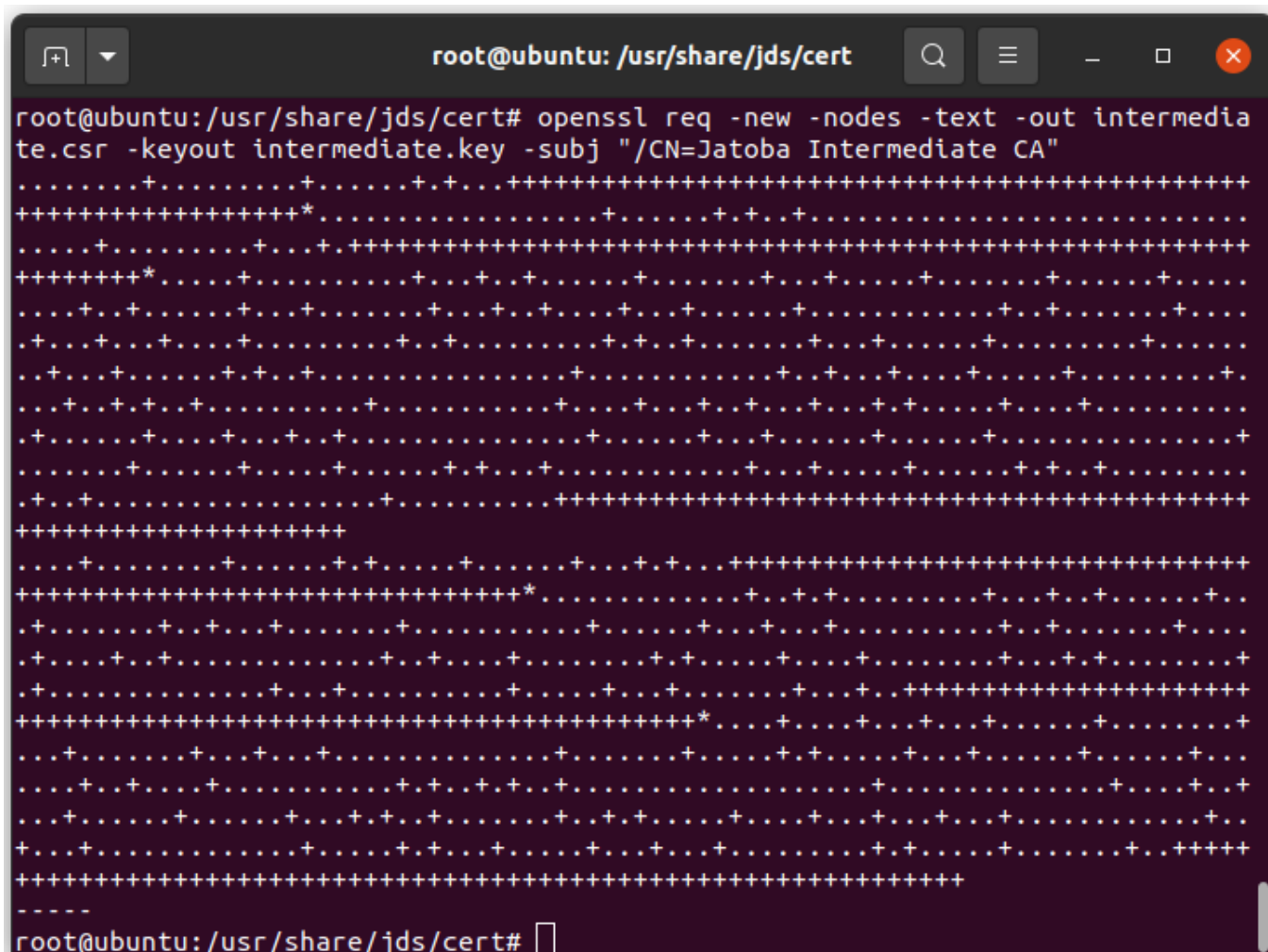
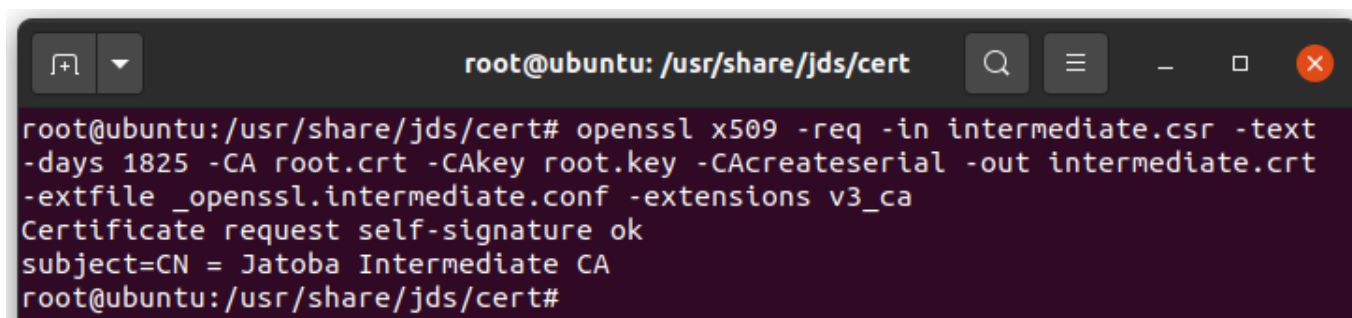


Рисунок 14.6 – Команда создания ключа промежуточного ЦС

Создать сертификат промежуточного ЦС командой:

```
#openssl x509 -req -in intermediate.csr -text -days 1825 -CA  
root.crt -CAkey root.key -CAcreateserial -out intermediate.crt  
-extfile _openssl.intermediate.conf -extensions v3_ca
```





```
root@ubuntu: /usr/share/jds/cert
root@ubuntu:/usr/share/jds/cert# openssl x509 -req -in intermediate.csr -text -days 1825 -CA root.crt -CAkey root.key -CAcreateserial -out intermediate.crt -extfile _openssl.intermediate.conf -extensions v3_ca
Certificate request self-signature ok
subject=CN = Jatoba Intermediate CA
root@ubuntu:/usr/share/jds/cert#
```

Рисунок 14.7 – Команда создания сертификата промежуточного ЦС командой:

### 14.5.3. Самоподписанный сертификат сервера СУБД (Root CA)

Создать ключ сервера СУБД командой:

```
#openssl req -new -nodes -text -out server.csr -keyout server.key -subj "/CN=<hostname>"
```



В значении CN=<hostname> должно указываться имя хоста, на котором установлена СУБД. Получить данное имя возможно выводом команды в терминале ОС.

```
# hostname
```

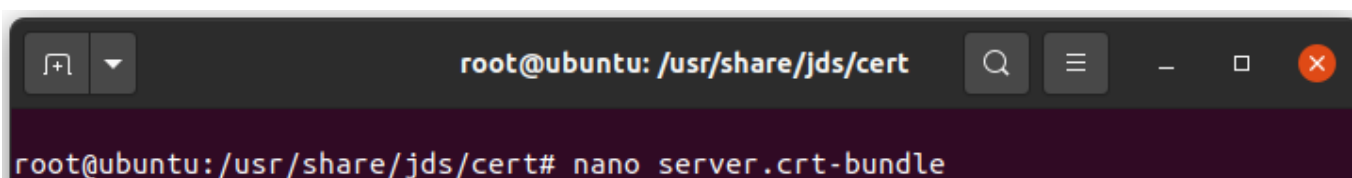
Создать сертификат сервера СУБД командой:

```
#openssl x509 -req -in server.csr -text -days 365 -CA intermediate.crt -CAkey intermediate.key -CAcreateserial -out server.crt
```

Создать цепочку сертификатов ЦС (CA Bundle).

Создать файл server.crt-bundle командой:

```
nano server.crt-bundle
```



```
root@ubuntu: /usr/share/jds/cert
root@ubuntu:/usr/share/jds/cert# nano server.crt-bundle
```

Рисунок 14.8 – Команда создания файла server.crt-bundle

В открывшийся файл последовательно вставить содержание сертификатов:

- server;

- intermediate;
- root.

Последовательность добавления сертификатов нарушать нельзя.

В параллельной сессии терминала ОС открыть файл сертификата server.crt через команду:

```
gedit server.crt
```

Выделить и скопировать содержание сертификата от значения «BEGIN CERTIFICATE» до «END CERTIFICATE», как показано на рисунке 14.9.



Рисунок 14.9 – Копируемая область сертификата server.crt

Вставить из буфера обмена в файл server.crt-bundle.



Рисунок 14.10 – Вставленное содержимое сертификата server.crt

В параллельной сессии терминала ОС открыть файл сертификата промежуточного ЦС intermediate.crt через команду:

```
gedit intermediate.crt
```

Выделить и скопировать содержание сертификата от значения «BEGIN CERTIFICATE» до «END CERTIFICATE», как показано на рисунке 14.11.

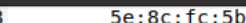


Рисунок 14.11 - Копируемая область сертификата intermediate.crt

Вставить из буфера обмена в файл server.crt-bundle.

```

GNU nano 4.8 server.crt-bundle
yCQt7BkK5z9KmuAbzMpuEHccKfjqHs9j7PvJU5vYAD/FWE514csnuIgrX1+Bpv+7
MiCXIFFmuv6kP7A5oo1pJj2RC2nrRRgbnBBjCE0WsVe629MRnG+DEyt/6jsFUPHV
naE/jSylC3uILTCRsY2o5pC06bE+B2LW39gdjVqI7a0K
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIDGzCCAgOgAwIBAgIUN6XsOD9T+w/HER2Z0osEe2aqa04wDQYJKoZIhvcNAQEL
BQAwGTEXMBUGA1UEAwOSMf0b2JhIFJvb3QgQ0EwHhcNMjQwOTA1MTQwOTU1WhcN
MjkWOTA0MTQwOTU1WjAhMR8wHQYDVQQDDBZKYXRvYmEgSW50ZXJtZWRpYXRlIENB
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAs03PjNqRmBjPxnbn+nZ8T
2y+yZzljsX9i8WL4UXAeAkB0ZF/Ref/T90n2YPzwb8V6M/sPbxQATVctbrYUpTmy
2uJpkT6zLV856QcovIvuJRj4SweA0rZtTnqrRvdCERsBJip30yVqqH5Ey0ZFmkqi
Z1qgzY5njD56DsadbIBGBKGtoq6UmUwiGu0vtS10bhGwQ8UMiB+ec/NoHK7j8v4z
nb5jLhWpQDht0C3uYFHZXS2YyFEXU1dxF/SFbHj6kX96F4ra+zEXXBWNkNzbu5Cm
3V7H5cQY4n1fILZucFtcY/vtdbdM4GFFF/UjDbzgxdkAH/Y+Q7VVVbmQjARPO2NG
rQIDAQAB01MwUTAPBgNVHRMECDAGAQH/AgEAMB0GA1UdDgQWBBTBELX0QJXVMA8x
/9fUJlDBrcjAsTAFBgNVHSMEGDAWgBRCHMCvMk30WL0Jdc0sVrWHRZ3WPzANBgkq
hkiG9w0BAQsFAAOCAQEUAUPfr13pv/HKi8ncptKTj8Zkdb+ZY8Y08xHwdppkugxuy
jwf00Igbav57TDuM80gm4cF6/jSzLcG3vPSnv009bdHbgV1yaRVzJO308+ONy3U3
h+Yc6ugVFZE1j+AEbFW3rg1up8Kbx/R6FEsXbB/bv3LJYDoxplwT/ZwGdH+3VEwE
aijuvSGYuYlriaZni25XivH/Xn0W+5L0kv56fT14SM9cID3Xw9CTCnhZHlocmKFL
5Yska9vHFNNqKwyoX2Hvf1wd8KrJ+A/+5fHkN+hVRimX/xvLtExnKBsci/+N06rQ
+/y9Ax+7cp7SFNNkROZyJzJ31/TIAMqNfOfIXoz8Ww==
-----END CERTIFICATE-----
^G Get Help      ^O Write Out     ^W Where Is      ^K Cut Text      ^J Justify
^X Exit          ^R Read File     ^\ Replace        ^U Paste Text    ^T To Spell

```

Рисунок 14.12 – Вставленное содержимое сертификата intermediate.crt

В параллельной сессии терминала ОС открыть файл сертификата root.crt через команду:

```
gedit root.crt
```

Выделить и скопировать содержание сертификата от значения «BEGIN CERTIFICATE» до «END CERTIFICATE», как показано на рисунке 14.13.



```

55      a8:f4:ec:01:b3:88:9e:75:41:ba:ba:5f:2e:00:b9:1a:18:e9:
56      29:3d:70:ed
57      -----BEGIN CERTIFICATE-----
58      MIIC8jCCAdqgAwIBAgIUcay3wMMqKLJzJmJ7DoIKgEXsvlowDQYJKoZIhvcNAQEL
59      BQAwGTEXMBUGA1UEAwOSmF0b2JhIFJvb3QgQ0EwHhcNMjQwOTA1MTM0MTM3WhcN
60      MzQwOTAzMTM0MTM3WjAZMRcwFQYDVQDDA5KYXRvYmEgUm9vdCBDQTCASiwdQYJ
61      KoZiHvcNAQEBBQADggEPADCCAQoCggEBAM1Zg2cRdQ+TGyMIuR63dSGjxcJ0RdGF
62      FTMwao/imSqUupMBIcjiUPFGmZ+Lq9PCRu+h0g3W+C6KIEqhlFJDBhXMTXN2zbSK
63      roi354ERfmLsyxZurRkz3mWKgGCBs8Q8zA5ChFbjR3PzEZt9RTtLP7BDpH+TqqdZ
64      o/QKcLtw5LqvorRg6inlQ5A5HPk0oz/DFvvcA7FB/jGsb9wBABVORmz8kVy2UnMz
65      kB8un73gYF5Bxc7/Zp7HrBnu/FuQmEEMPNS0BVpi6Gmgm4Sf5RFb5YXrvpo1D82E
66      DOgGVueIiF7UZSkFmkz6m8IbzQuZIZWY3yqAcvmoN63bF5SsQ0x0knUCAwEAAaMy
67      MDAwDwYDVROTBAGwBgEB/wIBATAdBgNVHQ4EFgQUQhZArzJNzli9CXXNLFa1h0Wd
68      1j8wDQYJKoZIhvcNAQELBQADggEBABX6mT2boSKWzKsDBZYXp19UKHv8SuQ84I5I
69      vZXUQH/rWd4VWxUNK+pXNxx3V7wYadEaE9ZAXGvk6xJh8TzJj6a8VtWm4+tRZLx4
70      vC1dLPMQSQBPqrdGkDnb40SLXEAhdjbiGL5yMWlu0PFvPPZpVrZbisWSl2Fwki8
71      y8wuF3lX/XDbPGFR8/zVvF1z0JmDUfvginprwsse9XQ6cXUNJf6P7DBrkoghXBp
72      sR5EJF0v8yyH5oKfT4uhrysdYeufujyVCGexCFQWpPp0fGFCvhrBGnrX0ApMsLXo
73      Iz7tshoEn7iwqqVheIA+lqj07AGziJ51Qbq6Xy4AuRoY6Sk9c00=
74      -----END CERTIFICATE-----

```

Рисунок 14.13 - Копируемая область сертификата root.crt

Вставить из буфера обмена в файл server.crt-bundle.

```

GNU nano 4.8      server.crt-bundle
a1juvSGYuYlriaZni25XivH/Xn0W+5L0kv56fT14SM9cID3Xw9CTCnhZHlocmKFL
5Yska9vHFNNqKwoyX2Hvf1wd8KrJ+A/+5fHKN+hVrImX/xvLTExnKBsci/+N06rQ
+/y9Ax+7cp7SFNNkROZyJzJ31/TIAMqNfOfIXoz8Ww==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIC8jCCAdqgAwIBAgIUcay3wMMqKLJzJmJ7DoIKgEXsvlowDQYJKoZIhvcNAQEL
BQAwGTEXMBUGA1UEAwOSmF0b2JhIFJvb3QgQ0EwHhcNMjQwOTA1MTM0MTM3WhcN
MzQwOTAzMTM0MTM3WjAZMRcwFQYDVQDDA5KYXRvYmEgUm9vdCBDQTCASiwdQYJ
KoZiHvcNAQEBBQADggEPADCCAQoCggEBAM1Zg2cRdQ+TGyMIuR63dSGjxcJ0RdGF
FTMwao/imSqUupMBIcjiUPFGmZ+Lq9PCRu+h0g3W+C6KIEqhlFJDBhXMTXN2zbSK
roi354ERfmLsyxZurRkz3mWKgGCBs8Q8zA5ChFbjR3PzEZt9RTtLP7BDpH+TqqdZ
o/QKcLtw5LqvorRg6inlQ5A5HPk0oz/DFvvcA7FB/jGsb9wBABVORmz8kVy2UnMz
kB8un73gYF5Bxc7/Zp7HrBnu/FuQmEEMPNS0BVpi6Gmgm4Sf5RFb5YXrvpo1D82E
DOgGVueIiF7UZSkFmkz6m8IbzQuZIZWY3yqAcvmoN63bF5SsQ0x0knUCAwEAAaMy
MDAwDwYDVROTBAGwBgEB/wIBATAdBgNVHQ4EFgQUQhZArzJNzli9CXXNLFa1h0Wd
1j8wDQYJKoZIhvcNAQELBQADggEBABX6mT2boSKWzKsDBZYXp19UKHv8SuQ84I5I
vZXUQH/rWd4VWxUNK+pXNxx3V7wYadEaE9ZAXGvk6xJh8TzJj6a8VtWm4+tRZLx4
vC1dLPMQSQBPqrdGkDnb40SLXEAhdjbiGL5yMWlu0PFvPPZpVrZbisWSl2Fwki8
y8wuF3lX/XDbPGFR8/zVvF1z0JmDUfvginprwsse9XQ6cXUNJf6P7DBrkoghXBp
sR5EJF0v8yyH5oKfT4uhrysdYeufujyVCGexCFQWpPp0fGFCvhrBGnrX0ApMsLXo
Iz7tshoEn7iwqqVheIA+lqj07AGziJ51Qbq6Xy4AuRoY6Sk9c00=
-----END CERTIFICATE-----

```

Рисунок 14.14 – Вставленное содержимое сертификата root.crt

Сохранить и закрыть файл.

Создать файл root.crt-bundle командой:

```
nano root.crt-bundle
```

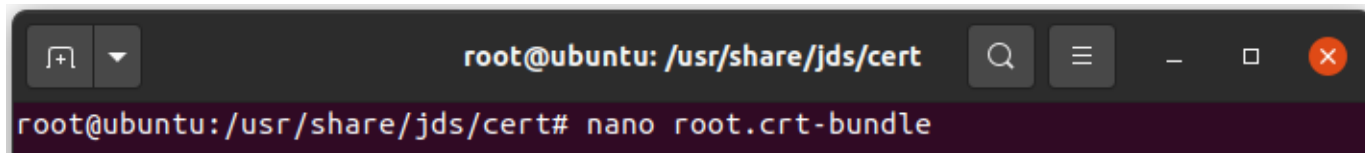


Рисунок 14.15 – Команда создания файла root.crt-bundle

В открывшийся файл последовательно вставить содержание сертификатов:

- intermediate;
- root.

Последовательность добавления сертификатов нарушать нельзя.

В параллельной сессии терминала ОС открыть файл сертификата промежуточного ЦС intermediate.crt через команду:

```
gedit intermediate.crt
```

Выделить и скопировать содержание сертификата от значения «BEGIN CERTIFICATE» до «END CERTIFICATE», как показано на рисунке 14.11.

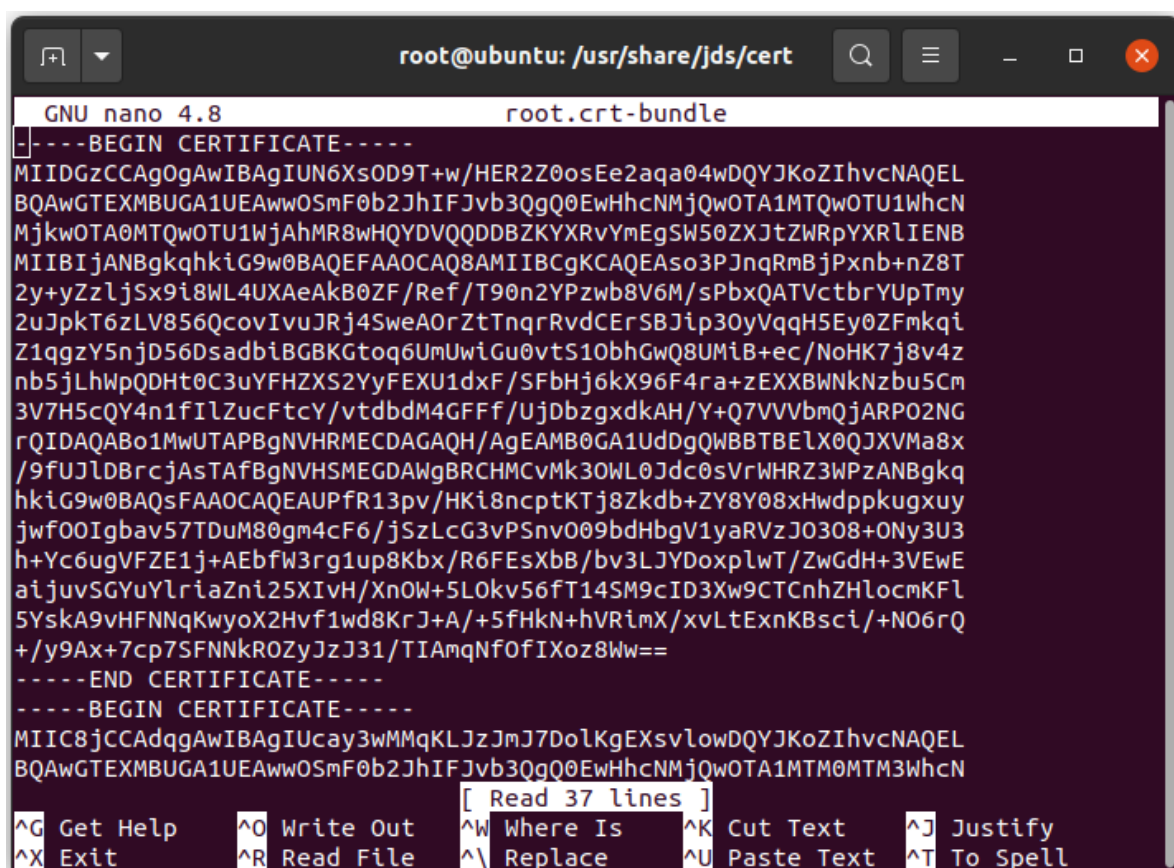
Вставить из буфера обмена в файл root.crt-bundle.

В параллельной сессии терминала ОС открыть файл сертификата root.crt через команду:

```
gedit root.crt
```

Выделить и скопировать содержание сертификата от значения «BEGIN CERTIFICATE» до «END CERTIFICATE», как показано на рисунке 14.13.

Вставить из буфера обмена в файл root.crt-bundle.



```

GNU nano 4.8 root.crt-bundle
-----BEGIN CERTIFICATE-----
MIIDGzCCAqOgAwIBAgIU6Xs0D9T+w/HER2Z0osEe2aqa04wDQYJKoZIhvcNAQEL
BQAwGTEXMBUGA1UEAwOSmF0b2JhIFJvb3QgQ0EwHhcNMjQwOTA1MTQwOTU1WhcN
MjkwOTA0MTQwOTU1WjAhMR8wHQYDVQDDDBZKYXRvYmEgSW50ZXJtZWRpYXRlIENB
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAs03PJnqRmBjPxnbn+nZ8T
2y+yZzljSx9i8WL4UXAeAkB0ZF/Ref/T90n2YPzwb8V6M/sPbxQATVctbrYUpTmy
2uJpKt6zLV856QcovIvuJRj4SweA0rZtTnqrRvdCERsBJip30yVqqH5Ey0ZFmkqi
Z1qgzY5njD56DsadbIBGBKGtoq6UmUwiGu0vtS10bhGwQ8UMiB+ec/NoHK7j8v4z
nb5jLhWpQDht0C3uYFHZXS2YyFEXU1dxF/SFbHj6kX96F4ra+zEXXBWNkNzbu5Cm
3V7H5cQY4n1fILZucFtcY/vtdbdM4GFFF/UjDbzgxdkAH/Y+Q7VVVbmQjARPO2NG
rQIDAQABo1MwUTAPBgNVHRMECDAGAQH/AgEAMBoGA1UdDgQWBWBTBELX0QJXVMA8x
/9fUJlDBrcjAsTAFBgnVHSMEGDAWgBRCHMCvMk3OWL0Jdc0sVrWHRZ3WPzANBgkq
hkiG9w0BAQsFAAOCAQEAPfR13pv/HKi8ncptKTj8Zkdb+ZY8Y08xHwdppkugxuy
jwf00Igbav57TDuM80gm4cF6/jSzLcG3vPSnv009bdHbgV1yaRVzJ0308+ONy3U3
h+Yc6ugVFZE1j+AEbfW3rg1up8Kbx/R6FESxbB/bv3LJYDoxplwT/ZwGdH+3VEwE
aijuvSGYuYlriaZni25XivH/XnOW+5L0kv56FT14SM9cID3Xw9CTCnhZHlocmKFL
5Yska9vHFNNqKwyoX2Hvf1wd8KrJ+A/+5fHkN+hVRimX/xvLtExnKBsci/+NO6RQ
+/y9Ax+7cp7SFNNkROZyJzJ31/TIAmqNfofIXoz8Ww==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIC8jCCAdqgAwIBAgIUcay3wMMqKLJzJmJ7DolKgEXsvlowDQYJKoZIhvcNAQEL
BQAwGTEXMBUGA1UEAwOSmF0b2JhIFJvb3QgQ0EwHhcNMjQwOTA1MTQwOTU1WhcN
[ Read 37 lines ]
^G Get Help      ^O Write Out     ^W Where Is      ^K Cut Text       ^J Justify
^X Exit          ^R Read File     ^\ Replace        ^U Paste Text    ^T To Spell

```

Рисунок 14.16 – Содержание файла root.crt-bundle

Сохранить и закрыть файл.

## 14.6. Создание клиентских сертификатов

### 14.6.1. Клиентский сертификат пользователя postgres

Создать ключ для клиентского сертификата пользователя postgres:

```
#openssl req -new -nodes -text -out client.postgres.csr -keyout
client.postgres.key -subj "/CN=postgres"
```

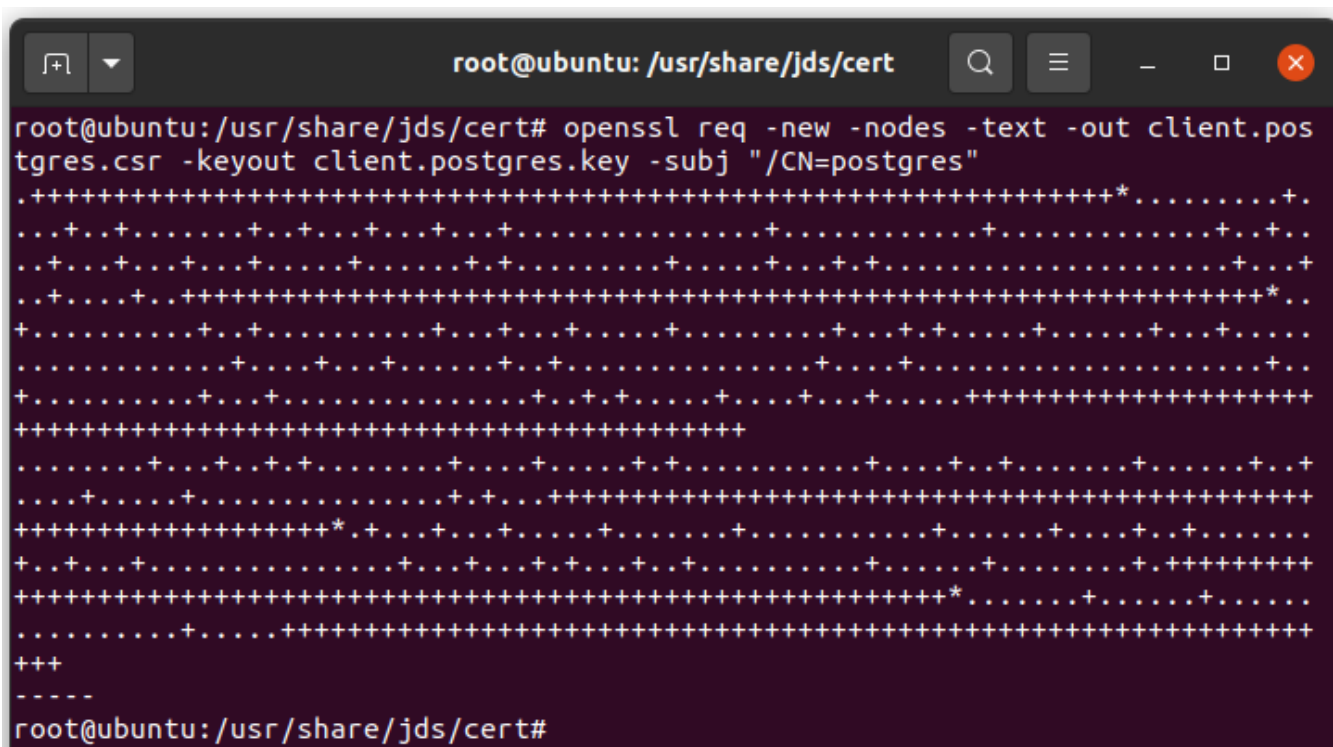


Рисунок 14.17 – Команда создания ключа для клиентского сертификата postgres

## Создать клиентский сертификат для пользователя postgres:

```
#openssl x509 -req -in client.postgres.csr -text -days 365 -CA
intermediate.crt -CAkey intermediate.key -CAcreateserial -out
client.postgres.crt
```

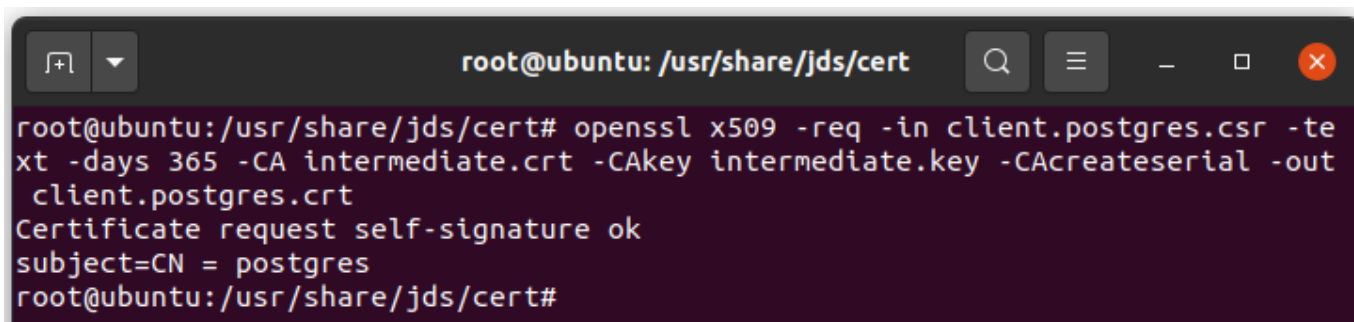


Рисунок 14.18 – Создание клиентского сертификата client.postgres.crt

#### 14.6.1.1 Конвертация клиентского сертификата postgres в PKCS#12

Сформировать цепочку сертификатов промежуточного и корневого ЦС.

```
#type intermediate.crt > client.postgres.crt-bundle
#type root.crt >> client.postgres.crt-bundle
```



Затем из нее, клиентского сертификата и соответствующего ему закрытого ключа создадим контейнер PFX. Пароль защиты закрытого ключа оставим пустым.

#### 14.6.1.2 Создание цепочки сертификатов промежуточного и корневого ЦС для клиентского сертификата postgres

Создать файл client.postgres.crt-bundle командой:

```
nano client.postgres.crt-bundle
```

В открывшийся файл последовательно вставить содержание сертификатов:

- intermediate;
- root.

Последовательность добавления сертификатов нарушать нельзя.

В параллельной сессии терминала ОС открыть файл сертификата промежуточного ЦС intermediate.crt через команду:

```
gedit intermediate.crt
```

Выделить и скопировать содержание сертификата от значения «BEGIN CERTIFICATE» до «END CERTIFICATE», как показано на рисунке 14.11.

Вставить из буфера обмена в файл client.postgres.crt-bundle.

В параллельной сессии терминала ОС открыть файл сертификата root.crt через команду:

```
gedit root.crt
```

Выделить и скопировать содержание сертификата от значения «BEGIN CERTIFICATE» до «END CERTIFICATE», как показано на рисунке 14.13.

Вставить из буфера обмена в файл client.postgres.crt-bundle.

Сохранить и закрыть файл.

### 14.6.1.3 Создание контейнера PFX

Из сформированной цепочки сертификатов промежуточного и корневого ЦС, клиентского сертификата postgres и соответствующего ему закрытого ключа создайте контейнер PFX командой:

```
#openssl pkcs12 -inkey client.postgres.key -in  
client.postgres.crt -certfile client.postgres.crt-bundle -  
export -out client.postgres.pfx
```

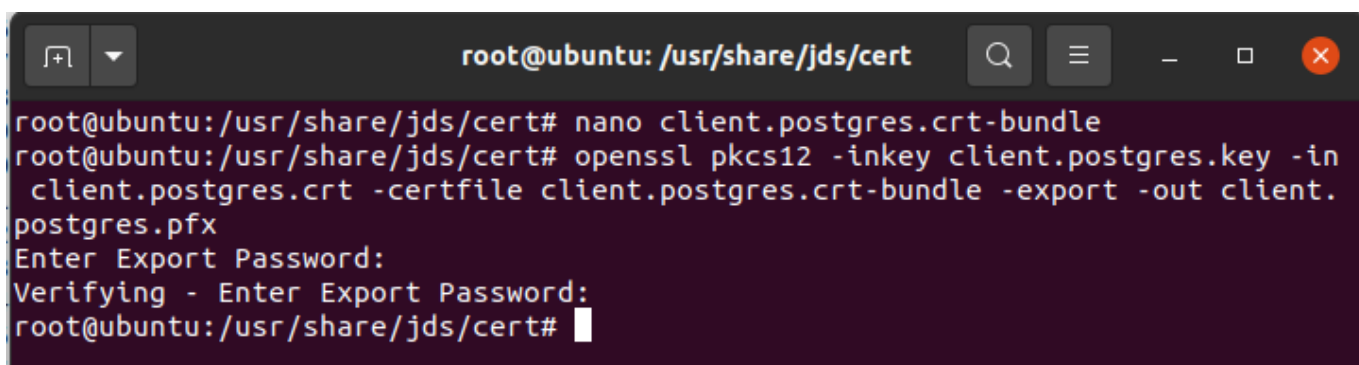


Рисунок 14.19 – Команда создания контейнера client.postgres.pfx

Пароль защиты закрытого ключа оставим пустым.

### 14.6.2. Клиентский сертификат пользователя JDS

Создать ключ для клиентского сертификата пользователя JDS:

```
#openssl req -new -nodes -text -out client.jds.csr -keyout  
client.jds.key -subj "/CN=jds"
```

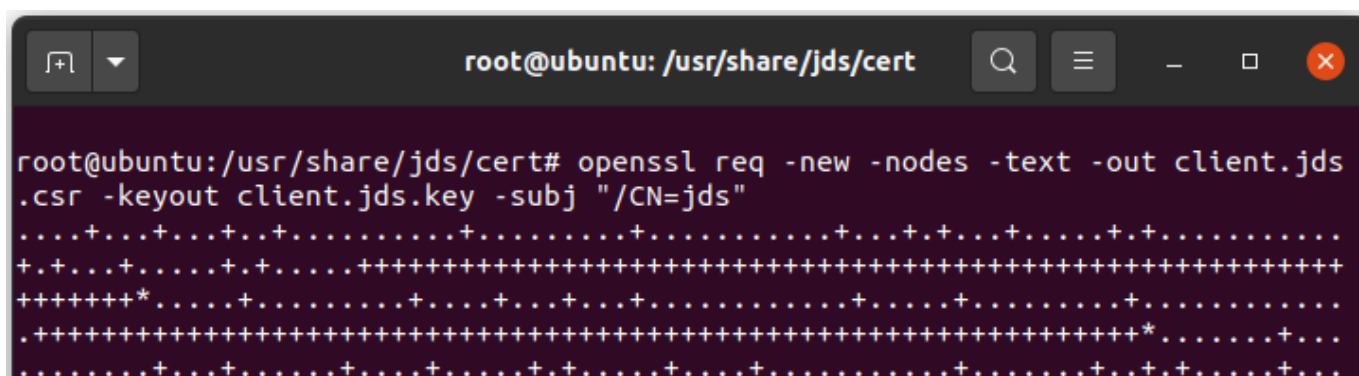
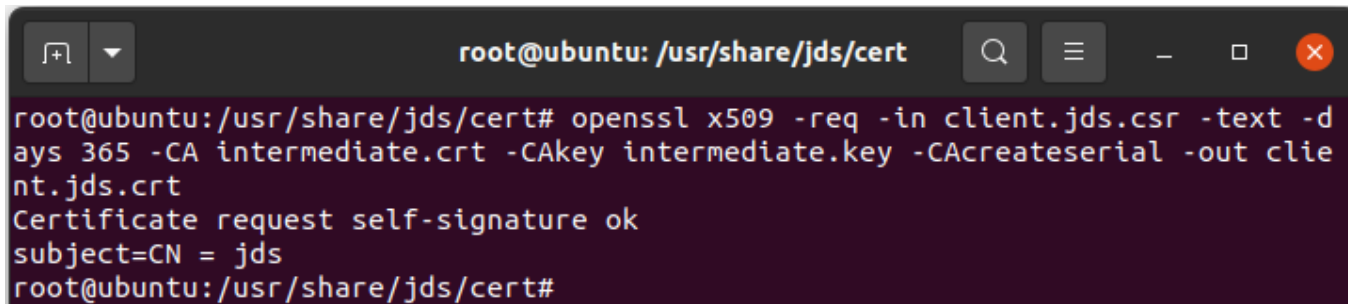


Рисунок 14.20 - Команда создания ключа для клиентского сертификата JDS

Создать клиентский сертификат для пользователя JDS:

```
#openssl x509 -req -in client.jds.csr -text -days 365 -CA  
intermediate.crt -CAkey intermediate.key -CAcreateserial -out  
client.jds.crt
```



```
root@ubuntu: /usr/share/jds/cert  
root@ubuntu:/usr/share/jds/cert# openssl x509 -req -in client.jds.csr -text -d  
ays 365 -CA intermediate.crt -CAkey intermediate.key -CAcreateserial -out clie  
nt.jds.crt  
Certificate request self-signature ok  
subject=CN = jds  
root@ubuntu:/usr/share/jds/cert#
```

Рисунок 14.21 – Создание клиентского сертификата client.jds.crt

#### 14.6.2.1 Конвертация клиентского сертификата JDS в PKCS#12

Сформировать цепочку сертификатов промежуточного и корневого ЦС.

```
type intermediate.crt > client.jds.crt-bundle  
type root.crt >> client.jds.crt-bundle
```

Затем из нее, клиентского сертификата и соответствующего ему закрытого ключа создадим контейнер PFX. Пароль защиты закрытого ключа оставим пустым.

#### 14.6.2.2 Создание цепочки сертификатов промежуточного и корневого ЦС для клиентского сертификата JDS

Создать файл client.jds.crt-bundle командой:

```
nano client.jds.crt-bundle
```

В открывшийся файл последовательно вставить содержание сертификатов:

- intermediate;
- root.

Последовательность добавления сертификатов нарушать нельзя.

В параллельной сессии терминала ОС открыть файл сертификата промежуточного ЦС intermediate.crt через команду:

```
gedit intermediate.crt
```

Выделить и скопировать содержание сертификата от значения «BEGIN CERTIFICATE» до «END CERTIFICATE», как показано на рисунке 14.11.

Вставить из буфера обмена в файл client.jds.crt-bundle.

В параллельной сессии терминала ОС открыть файл сертификата root.crt через команду:

```
gedit root.crt
```

Выделить и скопировать содержание сертификата от значения «BEGIN CERTIFICATE» до «END CERTIFICATE», как показано на рисунке 14.13.

Вставить из буфера обмена в файл client.jds.crt-bundle.

Сохранить и закрыть файл.

#### 14.6.2.3 Создание контейнера PFX

Из сформированной цепочки сертификатов промежуточного и корневого ЦС, клиентского сертификата postgres и соответствующего ему закрытого ключа создайте контейнер PFX командой:

```
#openssl pkcs12 -inkey client.jds.key -in client.jds.crt -  
certfile client.jds.crt-bundle -export -out client.jds.pfx
```

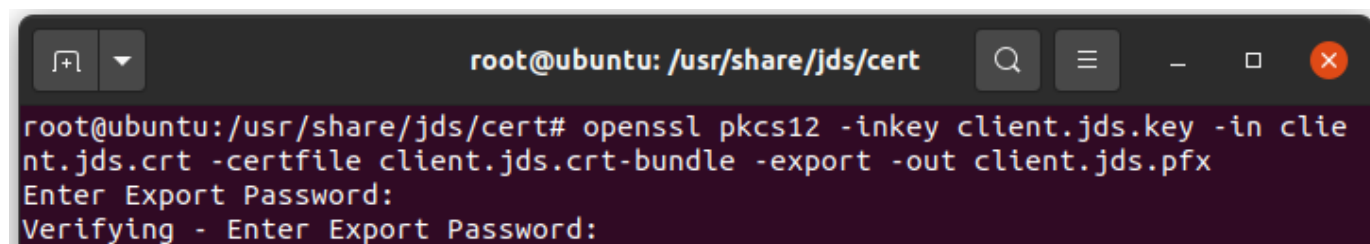


Рисунок 14.22 - Команда создания контейнера client.jds.pfx

Пароль защиты закрытого ключа оставим пустым.

Создать ключ для клиентского сертификата пользователя test user:

[illegible]

Рисунок 14.23 - Команда создания ключа для клиентского сертификата test user

Создать клиентский сертификат для пользователя test user:

```
#openssl x509 -req -in client.test_user.csr -text -days 365 -CA
intermediate.crt -CAkey intermediate.key -CAcreateserial -out
client.test user.crt
```

```
root@ubuntu:/usr/share/jds/cert# openssl x509 -req -in client.test_user.csr -t
ext -days 365 -CA intermediate.crt -CAkey intermediate.key -CAcreateserial -out
client.test_user.crt
Certificate request self-signature ok
subject=CN = jds
root@ubuntu:/usr/share/jds/cert#
```

Рисунок 14.24 – Создание клиентского сертификата client.test user.crt

#### 14.6.3.1 Конвертация клиентского сертификата test\_user в PKCS#12

Сформировать цепочку сертификатов промежуточного и корневого ЦС.

```
#type intermediate.crt > client.postgres.crt-bundle  
#type root.crt >> client.postgres.crt-bundle
```

Затем из нее, клиентского сертификата и соответствующего ему закрытого ключа создадим контейнер PFX. Пароль защиты закрытого ключа оставим пустым.

#### 14.6.3.2 Создание цепочки сертификатов промежуточного и корневого ЦС для клиентского сертификата test\_user

Создать файл client.test\_user.crt-bundle командой:

```
nano client.test_user.crt-bundle
```

В открывшийся файл последовательно вставить содержание сертификатов:

- intermediate;
- root.

Последовательность добавления сертификатов нарушать нельзя.

В параллельной сессии терминала ОС открыть файл сертификата промежуточного ЦС intermediate.crt через команду:

```
gedit intermediate.crt
```

Выделить и скопировать содержание сертификата от значения «BEGIN CERTIFICATE» до «END CERTIFICATE», как показано на рисунке 14.11.

Вставить из буфера обмена в файл client.postgres.crt-bundle.

В параллельной сессии терминала ОС открыть файл сертификата root.crt через команду:

```
gedit root.crt
```

Выделить и скопировать содержание сертификата от значения «BEGIN CERTIFICATE» до «END CERTIFICATE», как показано на рисунке 14.13.

Вставить из буфера обмена в файл client.test\_user.crt-bundle.

Сохранить и закрыть файл.

```

root@ubuntu: /usr/share/jds/cert
GNU nano 4.8 client.test_user.crt-bundle Modified
aijuvSGYuYlriaZni25XivH/Xn0W+5L0kv56fT14SM9cID3Xw9CTCnhZHlocmKFL
5Yska9vHFNNqKwoX2Hvf1wd8KrJ+A/+5fHkN+hVRimX/xvLtExnKBsci/+N06rQ
+/y9Ax+7cp7SFNNkROZyJzJ31/TIAMqNfOfIXoz8Ww==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIC8jCCAdqgAwIBAgIUcay3wMMqKLJzJmJ7DoIKgEXsvlowDQYJKoZIhvcNAQEL
BQAwGTEXMBUGA1UEAwOSMf0b2JhIFJvb3QgQ0EwHhcNMjQwOTA1MTM0MTM3WhcN
MzQwOTAzMTM0MTM3WjAZMRcwFQYDVQQDDA5KYXRvYmEgUm9vdCBDQTCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAM1Zg2cRdQ+TGyMIUR63dSGjxcJ0RdGF
FTMwao/imSqUupMBIcjiUPFGmZ+Lq9PCRu+h0g3W+C6KIEqhLFJDBhXMTXN2zbSK
roi354ERfmLsyxZurRkz3mWKgGCBs8Q8zA5ChFbjR3PzEZt9RTtLP7BDpH+TqqdZ
o/QKcLtw5LqvorRg6inLQ5A5HPk0oz/DFvvcA7FB/jGsb9wBABVORmz8kVy2UnMz
kB8un73gYF5Bxc7/Zp7HrBnu/FuQmEEMPNS0BVpi6Gmgm4Sf5RFb5YXrvpo1D82E
DOgGVueIiF7UZSkFmkz6m8IbzQuzIZWY3yqAcvmON63bF5SsQ0x0knUCAwEAAaMy
MDAwDwYDVR0TBAGwBgEB/wIBATAdBgNVHQ4EFgQUQhZArzJNzli9CXXNLFa1h0wd
1j8wDQYJKoZIhvcNAQELBQADggEBABX6mT2boSKWzKsDBZYXp19UKHv8SuQ84I5I
vZXUQH/rWD4VWxUNK+pXNxx3V7wYadEaE9ZAXGvk6xJh8TzJj6a8VtWm4+tRZLx4
vC1dLPMQSQBPqrdGkDnb40SLXEAhdjbiGL5yMWlu0PFvPPZpVrZbisWSl2Fwki8
y8wuF3lX/XDbPGFR8/zVvF1z0JmiDufvginprwsse9XQ6cXUNJf6P7DBrkoghXBp
sR5EJF0v8yyH5oKfT4uhrysdYeufujyVCGeXCFQWpPpOfGFCvhrBGnrXOApMsLXo
Iz7tshoEn7iwqqVheIA+lqj07AGziJ51Qbq6Xy4AuRoY65k9c00=
-----END CERTIFICATE-----
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell

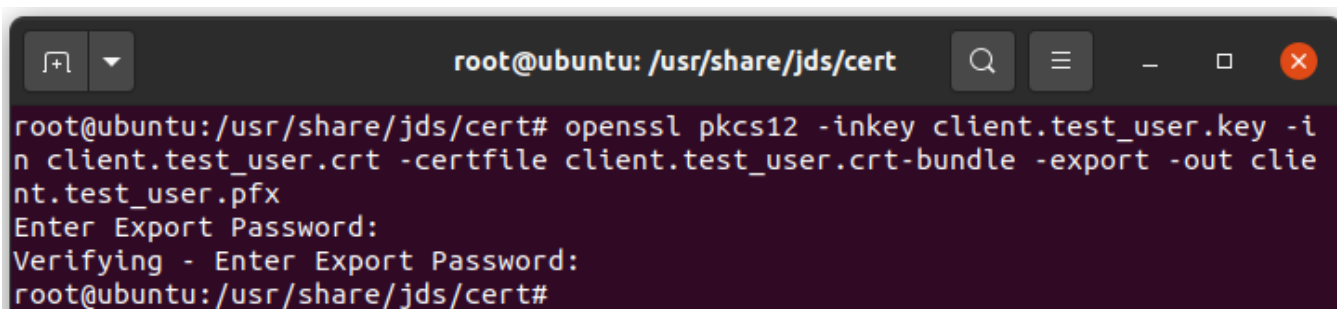
```

Рисунок 14.25 – Содержание файла client.test\_user.crt-bundle

### 14.6.3.3 Создание контейнера PFX

Из сформированной цепочки сертификатов промежуточного и корневого ЦС, клиентского сертификата test\_user и соответствующего ему закрытого ключа создайте контейнер PFX командой:

```
#openssl pkcs12 -inkey client.test_user.key -in
client.test_user.crt -certfile client.test_user.crt-bundle -
export -out client.test_user.pfx
```



```
root@ubuntu: /usr/share/jds/cert
root@ubuntu:/usr/share/jds/cert# openssl pkcs12 -inkey client.test_user.key -in client.test_user.crt -certfile client.test_user.crt-bundle -export -out client.test_user.pfx
Enter Export Password:
Verifying - Enter Export Password:
root@ubuntu:/usr/share/jds/cert#
```

Рисунок 14.26 – Команда создания контейнера client.test\_user.pfx

### 14.7. Структура хранения сертификатов

В силу особенностей архитектуры установки компонента JDS, служебной и целевых СУБД, возникает необходимость в структурированном хранении сертификатов.

Сертификаты СУБД предлагается хранить в каталоге:

```
/var/lib/jatoba/certs
```

Для создания SSL-соединения компонента JDS с целевой и/или служебной СУБД сертификат центра сертификации (CA) root.crt-bundle, должен храниться локально, например, в корневом каталоге:

```
/certs
```

Далее данный сертификат будет использован для создания «Target».

Сертификаты компонента JDS должны храниться в каталоге:

```
/opt/jds-cert
```

### 14.8. Настройка СУБД для SSL-соединения

Целевая и служебная СУБД настраивается для SSL-соединения путем внесения изменений в конфигурационные файлы:

- /var/lib/jatoba/<ver>/data/postgresql.conf;
- /var/lib/jatoba/<ver>/data/pg\_hba.conf.

В конфигурационный файл postgresql.conf внести следующие изменения:

```
ssl = on
```



```
ssl_ca_file = '/var/lib/jatoba/certs root.crt-bundle'  
ssl_cert_file = '/var/lib/jatoba/certs server.crt-bundle'  
ssl_key_file = '/var/lib/jatoba/certs server.key'
```

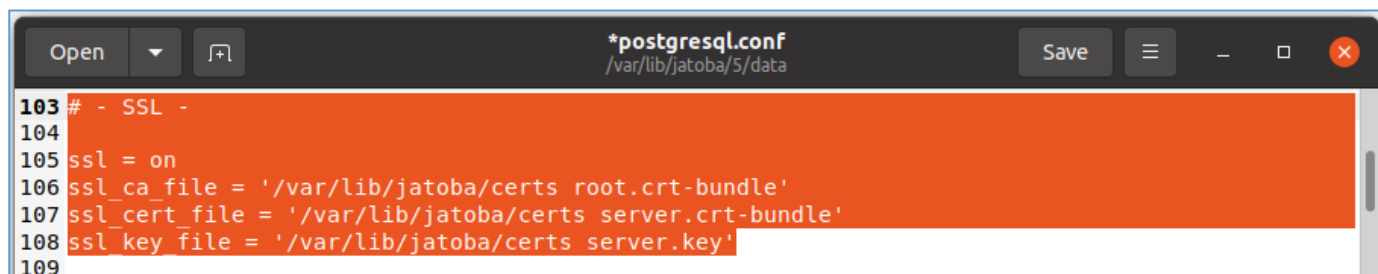


Рисунок 14.27 – Параметры SSL-соединения в СУБД

В конфигурационный файл pg\_hba.conf внести следующие изменения:

```
# TYPE DATABASE USER ADDRESS METHOD OPTIONS  
hostssl all all <ip адрес>/CIDR cert clientcert=verify-full  
hostssl all all 127.0.0.1/CIDR cert clientcert=verify-full
```

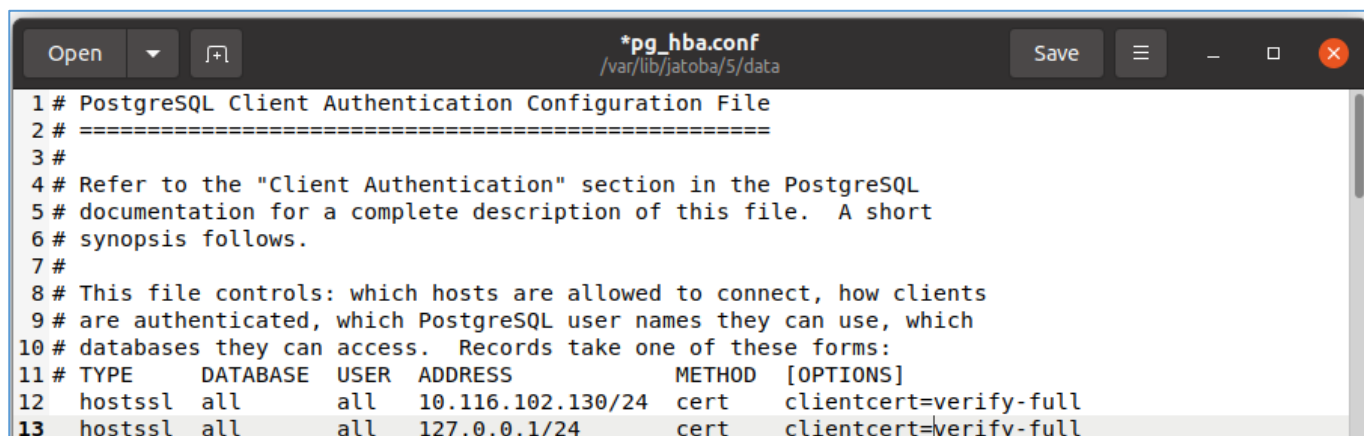


Рисунок 14.28 – Содержание конфигурационного файла pg\_hba.conf

Скопировать в каталог /var/lib/jatoba/ файлы:

- root.crt-bundle;
- server.crt-bundle;
- server.key.

Назначить владельца postgres для сертификатов:

```
chown postgres:postgres /var/lib/jatoba/certs server.key
```

```
root@ubuntu: /var/lib/jatoba/certs
root@ubuntu:/var/lib/jatoba/certs# chown postgres:postgres /var/lib/jatoba/certs/server.key
root@ubuntu:/var/lib/jatoba/certs# ls -l
total 12
-rw-r--r-- 1 root      root      2221 Sep  9 02:46 root.crt-bundle
-rw-r--r-- 1 root      root      3230 Sep  9 02:44 server.crt-bundle
-rw----- 1 postgres postgres 1704 Sep  5 07:17 server.key
root@ubuntu:/var/lib/jatoba/certs#
```

Рисунок 14.29 – Установка прав для сертификатов

Перезапустить службу jatoba:

```
systemctl restart jatoba-5
```

```
root@ubuntu: //
root@ubuntu://# systemctl restart jatoba-5
root@ubuntu://# systemctl status jatoba-5
● jatoba-5.service - Jatoba 5 database server
   Loaded: loaded (/lib/systemd/system/jatoba-5.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2024-10-20 03:52:27 PDT; 9s ago
```

Рисунок 14.30 – Перезапуск и статус службы jatoba<ver>

На данном шаге настройка СУБД «Jatoba» для SSL-соединения закончена.

### 14.9. Создание цели (Target) с SSL-соединением

Создание цели (Target) с SSL-соединением состоит из следующих шагов:

- Перейти в раздел «Настройки» – «Цели»;
- Нажать «Добавить»;
- Ввести название цели в поле «Наименование цели»;
- Ввести адрес существующего хоста в поле «Хост» (если целевая СУБД находится на том же хосте, что и JDS - указать localhost);
- В поле «Сертификат» вставить сертификат root.crt-bundle (см. п. 14.5.3 «Самоподписанный сертификат сервера СУБД (Root CA)»)

Как описывалось в п. 14.7, настоящего документа, целесообразнее выбрать сертификат по пути:

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

```
/cert/root.crt-bundle
```

- В поле "Функциональность" выбрать все функциональности.
- Нажать "ОК".

#### 14.10. Настройка компонента JDS для SSL-соединений

При ручной настройке компонента JDS для SSL-соединений потребуется создать каталог:

```
/opt/jds-cert/
```

Скопировать файлы сертификатов:

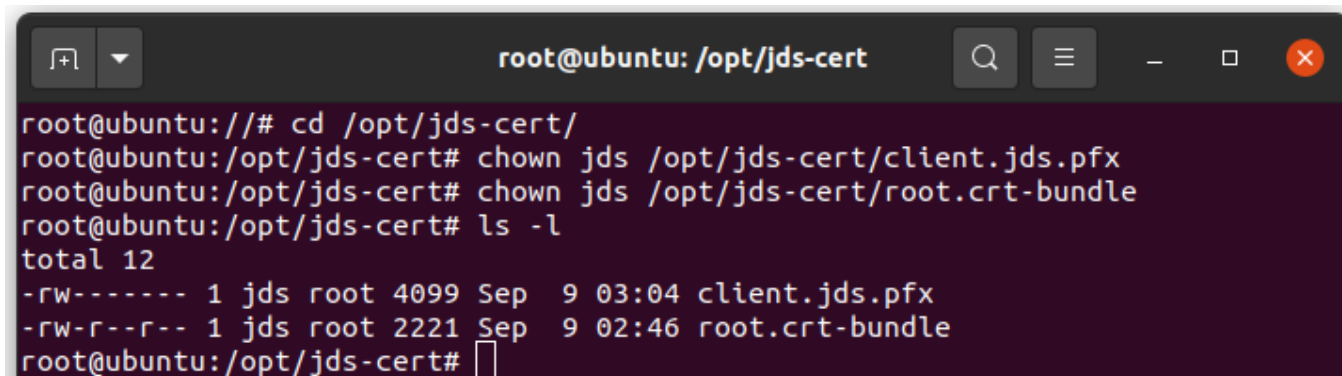
- /opt/jds-cert/client.jds.pfx;
- /opt/jds-cert/root.crt-bundle.

В рассматриваемом примере, как описано в п. 14.3, настоящего документа сформированные сертификаты храниться в каталоге:

```
/usr/share/jds/cert
```

Назначить владельца jds для сертификата и ключа:

```
# chown jds /opt/jds-cert/client.jds.pfx
# chown jds /opt/jds-cert/root.crt-bundle
```



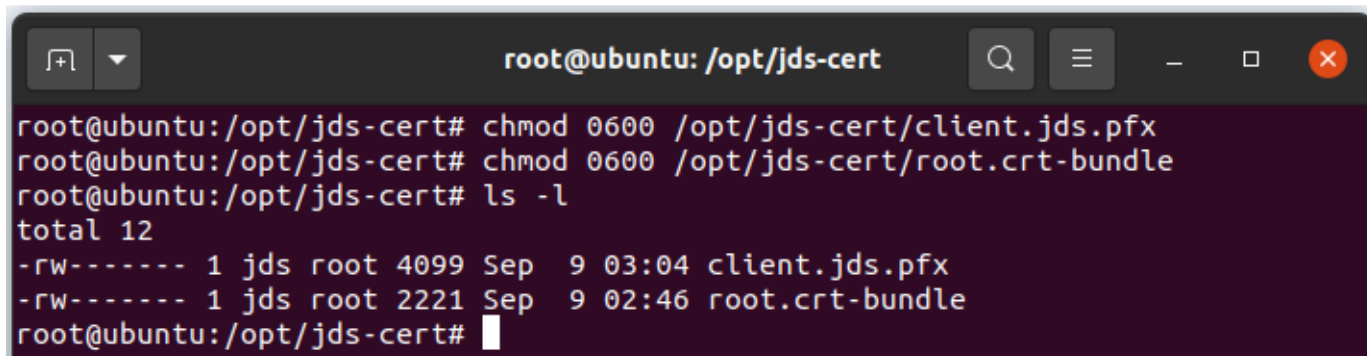
```
root@ubuntu: /opt/jds-cert
root@ubuntu:~# cd /opt/jds-cert/
root@ubuntu:/opt/jds-cert# chown jds /opt/jds-cert/client.jds.pfx
root@ubuntu:/opt/jds-cert# chown jds /opt/jds-cert/root.crt-bundle
root@ubuntu:/opt/jds-cert# ls -l
total 12
-rw----- 1 jds root 4099 Sep  9 03:04 client.jds.pfx
-rw-r--r-- 1 jds root 2221 Sep  9 02:46 root.crt-bundle
root@ubuntu:/opt/jds-cert#
```

Рисунок 14.31 – Назначение владельца jds для сертификата и ключа

Назначить минимальные права для сертификата и ключа:

```
# chmod 0600 /opt/jds-cert/client.jds.pfx
```

```
# chmod 0600 /opt/jds-cert/root.crt-bundle
```



```
root@ubuntu: /opt/jds-cert
root@ubuntu:/opt/jds-cert# chmod 0600 /opt/jds-cert/client.jds.pfx
root@ubuntu:/opt/jds-cert# chmod 0600 /opt/jds-cert/root.crt-bundle
root@ubuntu:/opt/jds-cert# ls -l
total 12
-rw----- 1 jds root 4099 Sep  9 03:04 client.jds.pfx
-rw----- 1 jds root 2221 Sep  9 02:46 root.crt-bundle
root@ubuntu:/opt/jds-cert#
```

Рисунок 14.32 – Установка прав на сертификат и ключ

Выполнить рестарт службы jds:

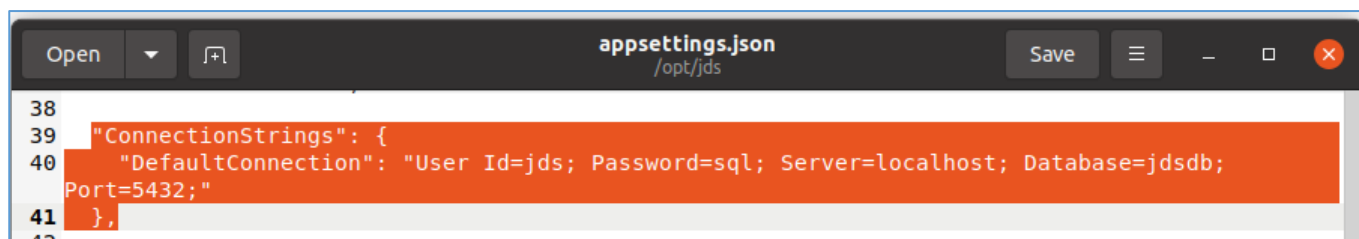
```
systemctl restart jds
```

Открыть файл appsettings.json:

```
gedit /opt/jds/appsettings.json
```

Исследовать строку DefaultConnection:

```
"ConnectionStrings": {
  "DefaultConnection": "User Id=jds; Password=sql;
Server=localhost; Database=jdsdb; Port=5432;"
},
```



```
appsettings.json
/opt/jds
Save
38
39 "ConnectionStrings": {
40   "DefaultConnection": "User Id=jds; Password=sql; Server=localhost; Database=jdsdb;
Port=5432;"
41 },
42
```

Рисунок 14.33 – Параметры по умолчанию в файле appsettings.json

Имеющиеся параметры по умолчанию, т.е. строку подключения к БД, изменить и установить параметры подключения пользователя JDS по SSL.

```
"ConnectionStrings": {  
  "DefaultConnection": "Server=ubuntu; Port=5432;  
Database=jdsdb; User Id=jds; SslMode=VerifyFull"  
},
```

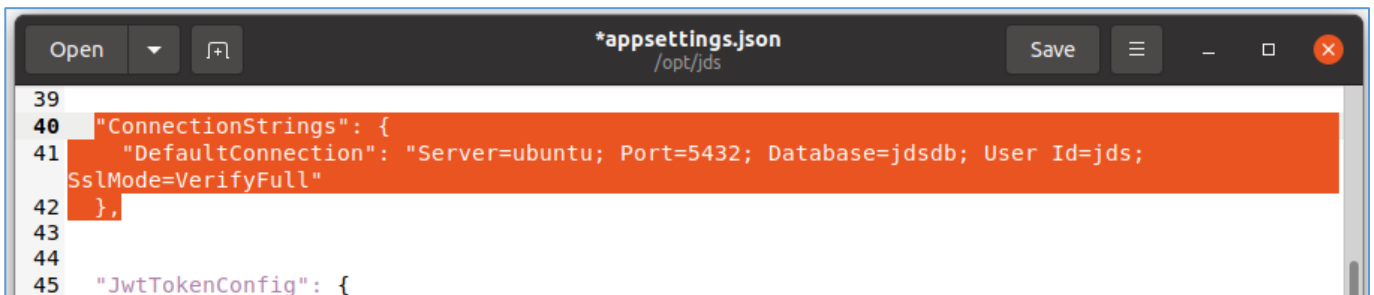


Рисунок 14.34 – Строка подключения пользователя jds по SSL

Вручную добавить раздел ConnectionSslConfigurator с следующим содержанием:

```
},  
"ConnectionSslConfigurator": {  
  "Connections": {  
    "DefaultConnection": {  
      "CAFile": "/opt/jds-cert/root.crt-bundle",  
      "ClientPfxFile": "/opt/jds-cert/client.jds.pfx",  
      "ClientPfxPassword": null,  
      "CheckServerCertificateRevocation": false  
    }  
  }  
}
```



Рисунок 14.35 - Раздел ConnectionSslConfigurator

Выполнить рестарт службы jds:

```
# systemctl restart jds
# systemctl status jds
```

На этом шаге настройка SSL-соединения закончено.

## 15. ПОДГОТОВКА ХОСТОВ НА ОС WINDOWS

В данном разделе приведено описание установки SSH-соединения между хостами с компонентом JDS под управлением ОС Windows и хостами с СУБД под управлением GNU/Linux.

Основной целью является, чтобы локальная учётная запись, под которой работает компонент JDS, имела настроенные ssh-ключи для беспарольного входа на все целевые хосты под учётной записью jdscontrol (или другой, назначенной на целевом хосте как учётная запись для удалённого администрирования).

### 15.1. Подготовка хоста с компонентом JDS

На хосте с компонентом JDS выполняются следующие действия:

- на хосте с JDS нужно установить поддержку SSH;



Инструкции по установке SSH находятся по ссылкам:

<https://learn.microsoft.com/ru-ru/windows/terminal/tutorials/ssh>

<https://winitpro.ru/index.php/2020/01/22/vstroennyj-ssh-klient-windows/>

- создать локальную учётную «jdsuser» запись с правами обычного пользователя, установить ей пароль, согласно принятой политике сложности паролей;
- выполнить вход под учётной записью jdsuser
- создать ssh-ключи для учётной записи jdsuser, закрытый ключ без пароля командой:

```
ssh-keygen
```

- запомнить путь к файлу ключа;
- для каждого целевого хоста (включая localhost, если JDS управляет СУБД на собственном хосте):

```
type %USERPROFILE%\.ssh\id_XXXX.pub | ssh  
jdscontrol@target_host "cat >> .ssh/authorized_keys"
```

- вместо id\_XXXX указать актуальное имя файла публичного ключа;
- расширение должно быть .pub;

- ответить, т.е. ввести "yes" на предложение принять ключ целевого хоста;
- выйти (exit).

В результате, локальная учётная запись jdsuser на Windows-хосте будет иметь возможность беспарольного входа на целевые хосты.

В случае, когда компонент JDS развёрнут как IIS-приложение выполняются следующие действия:

- создать новый AppPool IIS, далее – JdsAppPool;
- настроить JdsAppPool на работы под локальной учётной записью jdsuser в качестве Identity, включить опцию загрузки профиля пользователя в настройке JdsAppPool;
- указать JdsAppPool в качестве пула приложений для сайта с JDS.



Инструкции по настройке находятся по ссылке:

<https://learn.microsoft.com/en-us/iis/manage/configuring-security/application-pool-identities#configuring-iis-application-pool-identities>

## 15.2. Подготовка хоста с СУБД на ОС Windows

Подготовка хоста с СУБД на ОС Windows требует выполнения следующих действий:

- установить поддержку SSH;



Инструкции по установке SSH находятся по ссылкам:

<https://learn.microsoft.com/ru-ru/windows/terminal/tutorials/ssh>

<https://winitpro.ru/index.php/2020/01/22/vstroennyj-ssh-klient-windows/>

- создать локальную учётную запись «jdscontrol» с правами обычного пользователя, установить ей пароль, согласно принятой политике сложности паролей;
- убедиться в возможности входа на Windows-хост с СУБД по SSH под учётной записью jdscontrol командой:

```
ssh jdscontrol@windows-host-with-jatoba
```

- после входа выполнить команду:



dir

- достаточно получить любой вывод
- выйти командой:

exit

- предоставить учётной записи jdscontrol следующие права:
  - право чтения списка файлов в каталоге DATA;
  - право чтения списка файлов в каталоге DATA\log;
  - право чтения файлов в каталоге DATA\log;
  - право чтения и записи ограниченного перечня файлов в папке DATA: postgres.conf, pg\_hba.conf, pg\_ident.conf;
  - право чтения и записи для дополнительных файлов в папке DATA, относящихся к настройке СУБД и созданных администратором вручную пример - файлы в директивах include\_file, include\_dir.

## 16. ПОДГОТОВКА ХОСТА С СУБД «POSTGRESQL» ДЛЯ УПРАВЛЕНИЯ КОМПОНЕНТОМ JDS

Действия по подготовке хоста с СУБД «PostgreSQL» для управления компонентом JDS отличаются от действия по установке СУБД «Jatoba».

Отличие следует из изменений путей к файлам конфигурации СУБД:

- для СУБД «Jatoba» файлы конфигурации «postgrtesql.conf», «pg\_hba.conf» и «pg\_ident.conf» располагаются в папке DATA, независимо от версии СУБД;
- для СУБД PostgreSQL выше указанные файлы конфигурации находятся в папке /etc/postgresql и далее, в зависимости от особенностей установки СУБД;
- кроме того, Jatoba может быть установлена вручную, с нестандартным расположением файлов конфигурации и предыдущий пункт будет относиться и к Jatoba.

Сложность состоит в том, что для управления конфигурацией СУБД требуется доступ на запись к определённым файлам, а для выполнения этих действий на хосте с целевой СУБД используется учётная запись с ограниченными правами, предугадать где расположены файлы конфигурации СУБД нельзя.

Для СУБД PostgreSQL, а также для нестандартных установок СУБД «Jatoba», требуется выполнить настройку вручную:

- требуется выбрать локальную учётную запись, которая будет использоваться для управления целевым хостом и службой СУБД (далее принимается имя - jdscontrol); эта учётная запись должна иметь домашнюю папку и разрешение использовать bash как оболочку;
- имя этой учётной записи указывается в соответствующем поле настройки хоста в JDS;
- определить название группы, которой принадлежат файлы конфигурации СУБД и добавить локальную учётную запись «jdscontrol» в эту группу;
- файл «postgresql.conf» целевой СУБД должен иметь права на запись для группы;
- файл «pg\_hba.conf» целевой СУБД должен иметь права на запись для группы;

- файл `pg_ident.conf` целевого инстанса СУБД должен иметь права на запись для группы;
- папки, в которых находятся указанные файлы, должны иметь разрешение на чтение списка файлов для группы, членом которой является «jdscontrol», на всех уровнях вложенности до файлов конфигурации;
- вышесказанное, также, должно выполняться ко всем дополнительным файлам конфигурации, которые используются как "включаемые" (`include`, `include_if_exists`, `include_dir`);
- должна существовать и быть доступной на чтение и запись для учётной записи «jdscontrol» папка для хранения резервных копий конфигураций СУБД, указанная в соответствующем поле настройки хоста в JDS;
- учётная запись «jdscontrol» должна иметь право запуска, остановки, перезапуска, разрешения и блокирования автозапуска для службы СУБД, название которой указано в соответствующем поле настройки СУБД в JDS, без запроса пароля - средствами настройки механизма `sudoers`.

### Например

Действия выполняются от привилегированного пользователя ОС «root», учётная запись «jdscontrol» уже существует, она имеет домашнюю папку)

Операции по настройке прав учётной записи `jdscontrol` для СУБД `Postgresql 16`, установленной штатным образом на ОС `Debian 12`, работающей как служба `systemd` с именем `postgresql@16-main`, будут следующими:

- создать локальную учётную запись для управления;
- учётную запись включить в группу «postgres», т.к. этой группе принадлежат файлы конфигурации СУБД, командой:

```
useradd -r -b /var/lib -m -G postgres -s $(which bash)
jdscontrol
```

- файлы конфигурации службы находятся в папке `/etc/postgresql/16/main`

```
chmod g+w /etc/postgresql/16/main/postgresql.conf  
chmod g+w /etc/postgresql/16/main/pg_hba.conf  
chmod g+w /etc/postgresql/16/main/pg_ident.conf
```

- создать список команд для управления сервисом СУБД;
- создать файл /etc/sudoers.d/jds\_commands\_postgresql16main следующего содержания:

```
Cmnd_Alias POSTGRESQL16MAIN = \  
/usr/bin/systemctl start postgresql@16-main.service,  
/usr/bin/systemctl start postgresql@16-main, \  
/usr/bin/systemctl stop postgresql@16-main.service,  
/usr/bin/systemctl stop postgresql@16-main, \  
/usr/bin/systemctl restart postgresql@16-main.service,  
/usr/bin/systemctl restart postgresql@16-main, \  
/usr/bin/systemctl enable postgresql@16-main.service,  
/usr/bin/systemctl enable postgresql@16-main, \  
/usr/bin/systemctl disable postgresql@16-main.service,  
/usr/bin/systemctl disable postgresql@16-main
```

- предоставить право учётной записи «jdscontrol» выполнять список команд под именем POSTGRESQL16MAIN через sudo без запроса пароля:
- создать файл /etc/sudoers.d/jdscontrol\_postgresql16main следующего содержания:

```
jdscontrol ALL = NOPASSWD: POSTGRESQL16MAIN
```

- создать в домашней папке пользователя jdscontrol папку для резервных копий конфигураций СУБД:

```
su jdscontrol -c "mkdir -p ~/backup"
```

На данном шаге настройка закончена.

## **17. НАСТРОЙКИ СУБД И ЕЕ КОМПОНЕНТ ПО УМОЛЧАНИЮ, КОТОРЫЕ МОГУТ БЫТЬ ИСПОЛЬЗОВАНЫ ДЛЯ НСД**

В данном разделе приводятся параметры компонент, хранящие пароли (ключи), и перечень настроек авторизации и аутентификации СУБД «Jatoba» и входящих в ее состав компонент, отвечающих за ИБ.

### 17.1. Компоненты хранящие пароли (ключи)

Таблица 17.1 – Перечень компонент СУБД «Jatoba» хранящие пароли

Название компонента	Конфигурационный файл	Название параметра	Значение по умолчанию	Описание
fasttrun	postgresql.conf	—		—
fulleq	postgresql.conf	—		—
ja_csum	postgresql.conf	—		—
ja_sync_ldap	служебная таблица ja_sync_ldap.profile	поле таблицы: pswd	Значение по умолчанию отсутствует. Всегда явно задается Администратором СУБД	Пароль маскируется через BASE64
activator/validator	postgresql.conf	—		—
jcs	postgresql.conf	jcs.key jcs.iv	Значение по умолчанию отсутствует. Всегда явно задается Администратором СУБД	Хранит общий ключ шифрования записей для всех таблиц / баз данных
jdv	postgresql.conf	—		—
BTreeKNN	postgresql.conf	—		—
mchar	postgresql.conf	—		—
online_analyze	postgresql.conf	—		—
pg_cryogen	postgresql.conf	—		—
pg_hint_plan	postgresql.conf	—		—
pg_store_plans	postgresql.conf	—		—
pg-ulid	postgresql.conf	—		—
plantuner	postgresql.conf	—		—
plspgsql	хранилище сертификатов (Crypto API / КриптоПро)	хранилище сертификатов	Значение по умолчанию отсутствует.	Для обфускации/деобфускации кода хранимых процедур расширение берет ключ из локального хранилища сертификатов

Название компонента	Конфигурационный файл	Название параметра	Значение по умолчанию	Описание
			Всегда явно задается Администратором СУБД/БД; Разработчиком БД	
securityprofile	служебная таблица securityprofile.password_history	поле таблицы: passhistpassword	Значение по умолчанию отсутствует. Значения появляются по мере накопления истории паролей пользователей	Для нужд выполнения политик, связанных с подсчетом количества различных символов в пароле и подсчетом отличных от предыдущего пароля символов, расширение требует хранения паролей в служебной таблице в открытом виде
sql_firewall	postgresql.conf	—	—	—
fasttrun	postgresql.conf	—	—	—
fulleq	postgresql.conf	—	—	—
ja_csum	postgresql.conf	ja_csum.db_name	postgres	Параметр используется для фоновых процессов компонента, проверяющих контрольные суммы файлов и объектов. Фоновый процесс производит подключение к СУБД по внутренним механизмам. (Для взаимодействия с внешней средой не используется)
ja_sync_ldap	конфигурация внешних подключений хранится в таблице ja_sync_ldap.profile	поля таблицы: – host_ip – port – login – pswd	значение по умолчанию отсутствует; всегда явно задается Администратором СУБД	Данные из указанных полей используются для установления <b>исходящего</b> соединения из СУБД в службу каталогов по протоколам LDAP/LDAPS
jcs	postgresql.conf	—	—	—
jdv	postgresql.conf	—	—	—
BTreeKNN	postgresql.conf	—	—	—
mchar	postgresql.conf	—	—	—
online_analyze	postgresql.conf	—	—	—
pg_cryogen	postgresql.conf	—	—	—
pg_hint_plan	postgresql.conf	—	—	—

№ изменения: \_\_\_\_\_ Подпись отв. лица: \_\_\_\_\_ Дата внесения изм: \_\_\_\_\_

Название компонента	Конфигурационный файл	Название параметра	Значение по умолчанию	Описание
pg_store_plans	postgresql.conf	—	—	—
pg-ulid	postgresql.conf	—	—	—
plantuner	postgresql.conf	—	—	—
plspgsql	postgresql.conf	Утилита wrp1pgsql. Опции командной строки -h -P -U -W/-w	значение по умолчанию отсутствует; всегда явно задается Администратором СУБД/БД; Разработчиком БД	Данный компонент может использоваться удаленно от СУБД (например, на стороне разработчика БД) и устанавливает <b>исходящее</b> соединение по протоколу libpq. <b>SSL соединение не поддерживается</b>
securityprofile	postgresql.conf	—	—	—
sql_firewall	postgresql.conf	—	—	—



## 17.2. Перечень настроек авторизации и аутентификации СУБД «Jatoba» и входящих в ее состав компонент, отвечающих за ИБ

Таблица 17.2 – Перечень настроек авторизации и аутентификации СУБД «Jatoba» и входящих в ее состав компонент, отвечающих за ИБ

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
Jatoba	postgresql.conf	hba_file	pg_hba.conf	Задаёт файл конфигурации для аутентификации по сетевым узлам. <b>Параметр задаётся при старте сервера и нельзя изменить</b>
Jatoba	postgresql.conf	config_file	postgresql.conf	Задаёт основной файл конфигурации сервера. <b>Параметр задаётся при старте сервера и нельзя изменить</b>
Jatoba	postgresql.conf	ident_file	pg_ident.conf	Задаёт файл конфигурации для сопоставлений имён пользователей. <b>Параметр задаётся при старте сервера и нельзя изменить</b>
Jatoba	postgresql.conf	authentication_timeout	1m	Максимальное время, за которое должна произойти аутентификация. Если это значение задаётся без единиц измерения, оно считается заданным в миллисекундах
Jatoba	postgresql.conf	password_encryption	scram-sha-256	Алгоритм шифрования пароля (scram-sha-256 или md5), пароль сохраняется в виде хеша
Jatoba	postgresql.conf	krb_server_keyfile	/usr/local/pgsql/etc/krb5.keytab	Задаёт расположение файла ключей Kerberos для данного сервера
Jatoba	postgresql.conf	krb_caseins user	Off	Обработка имен пользователей GSSAPI с/без учёта регистра
Jatoba	postgresql.conf	db_user_namespace	Off	Соотносит имена пользователей к базам данных. <b>Параметр задаётся при старте сервера и нельзя изменить</b>
Jatoba	postgresql.conf	SSL	Off	Задаёт тип подключения SSL. Этот параметр можно задать только в postgresql.conf или в командной строке при запуске сервера
Jatoba	postgresql.conf	ssl_ca_file	—	Задаёт имя файла, содержащего сертификаты центров сертификации (ЦС) для SSL-сервера. Этот параметр можно задать только в postgresql.conf или в командной строке при запуске сервера
Jatoba	postgresql.conf	ssl_cert_file	server.crt	Задаёт имя файла, содержащего сертификат этого SSL-сервера.

№ изменения: \_\_\_\_\_ Подпись отв. лица: \_\_\_\_\_ Дата внесения изм: \_\_\_\_\_

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
				Этот параметр можно задать только в <code>postgresql.conf</code> или в командной строке при запуске сервера
Jatoba	<code>postgresql.conf</code>	<code>ssl_key_file</code>	<code>server.key</code>	Задаёт имя файла, содержащего закрытый ключ SSL-сервера. Этот параметр можно задать только в <code>postgresql.conf</code> или в командной строке при запуске сервера
Jatoba	<code>postgresql.conf</code>	<code>ssl_crl_file</code>	—	Задаёт имя файла, содержащего список отзыва клиентских сертификатов (CRL, Certificate Revocation List) для SSL. Этот параметр можно задать только в <code>postgresql.conf</code> или в командной строке при запуске сервера
Jatoba	<code>postgresql.conf</code>	<code>ssl_ciphers</code>	<code>HIGH:MEDIUM:+3DES:!aNULL</code>	Задаёт список наборов шифров SSL, которые могут применяться для SSL-соединений. Этот параметр действует только для подключений TLS версии 1.2 и ниже. Этот параметр можно задать только в <code>postgresql.conf</code> или в командной строке при запуске сервера
Jatoba	<code>postgresql.conf</code>	<code>ssl_crl_dir</code>	—	Задаёт директорию, содержащего списки отзыва клиентских сертификатов. Этот параметр можно задать только в <code>postgresql.conf</code> или в командной строке при запуске сервера
Jatoba	<code>postgresql.conf</code>	<code>ssl_prefer_server_ciphers</code>	<code>On</code>	Определяет, должны ли шифры SSL сервера предпочитаться клиентским. Этот параметр можно задать только в <code>postgresql.conf</code> или в командной строке при запуске сервера
Jatoba	<code>postgresql.conf</code>	<code>ssl_ecdh_curve</code>	—	Задаёт имя кривой для использования при обмене ключами ECDH. Эту кривую должны поддерживать все подключающиеся клиенты. Этот параметр можно задать только в <code>postgresql.conf</code> или в командной строке при запуске сервера
Jatoba	<code>postgresql.conf</code>	<code>ssl_min_protocol_version</code>	<code>TLSv1.2</code>	Задаёт минимальную версию протокола SSL/TLS. Этот параметр можно задать только в <code>postgresql.conf</code> или в командной строке при запуске сервера
Jatoba	<code>postgresql.conf</code>	<code>ssl_max_protocol_version</code>	—	Задаёт максимальную версию протокола SSL/TLS. Этот параметр можно задать только в <code>postgresql.conf</code> или в командной строке при запуске сервера

№ изменения: \_\_\_\_\_ Подпись отв. лица: \_\_\_\_\_ Дата внесения изм: \_\_\_\_\_

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
Jatoba	postgresql.conf	ssl_dh_params_file	—	Задаёт имя файла с параметрами алгоритма Диффи-Хеллмана. Использование нестандартных параметров DH защищает от атаки, рассчитанной на взлом хорошо известных встроенных параметров DH. Этот параметр можно задать только в <code>postgresql.conf</code> или в командной строке при запуске сервера
Jatoba	postgresql.conf	ssl_passphrase_command_supports_reload	Off	Этот параметр определяет, будет ли заданная параметром <code>ssl_passphrase_command</code> команда, запрашивающая пароль, также вызываться при перезагрузке конфигурации, если для файла ключа требуется пароль. Этот параметр можно задать только в <code>postgresql.conf</code> или в командной строке при запуске сервера
Jatoba	pg_hba.conf	hostssl all all <ip/mask> cert verify-full	—	Указывает, что все внешние соединения должны быть по SSL из конкретной подсети
Jatoba	pg_hba.conf	hostssl replication jalog_user <ip/mask> cert verify-full	—	Указывает, что репликация для пользователя <code>jalog_user</code> должна быть по SSL с конкретной подсети
Jatoba	pg_hba.conf	local all postgres scram-sha-256	—	
jaPooler	pgbouncer.ini	confdir	—	Показывает расположение текущего файла конфигурации. При изменении этого параметра компонент будет использовать другой файл конфигурации после команды RELOAD
jaPooler	pgbouncer.ini	auth_type	scram-sha-256	Тип аутентификации: <ul style="list-style-type: none"> <li>— <code>cert</code> – Клиент должен подключаться по соединению TLS с действительным клиентским сертификатом;</li> <li>— <code>md5</code> – Применять проверку пароля по хеш MD5.</li> </ul> Этот метод аутентификации выбирается по умолчанию. При установке <code>md5</code> , если пароль пользователя задан для метода SCRAM, то применяется проверка по алгоритму SCRAM; <ul style="list-style-type: none"> <li>— <code>scram-sha-256</code> - Применять проверку пароля по алгоритму SCRAM-SHA-256;</li> </ul> Зашифрованные SCRAM пароли могут использоваться только для проверки пароли клиентов, но не для входа на сервер.

№ изменения: \_\_\_\_\_ Подпись отв. лица: \_\_\_\_\_ Дата внесения изм: \_\_\_\_\_

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
				<p>Чтобы использовать SCRAM для серверных подключений, пароли необходимо задать открытым текстом.</p> <ul style="list-style-type: none"> <li>– plain – Пароли хранятся в открытом виде;</li> <li>– trust – Аутентификация не производится;</li> <li>– any – Аутентификация не производится, имя настраивается в строке подключения;</li> <li>– hba – Аутентификация по файлу pg_hba.conf;</li> <li>– ram - Для проверки подлинности пользователей используется инфраструктура РАМ. Не совместим с использованием директивы auth_user</li> </ul>
jaPooler	pgbouncer.ini	auth_hba_file	—	Файл конфигурации HBA (аналог pg_hba.conf в Jatoba), который используется, когда параметр auth_type равен hba
jaPooler	pgbouncer.ini	auth_file	—	Имя файла аутентификации, из которого будут загружаться имена и пароли пользователей.
jaPooler	pgbouncer.ini	auth_user	—	Имя пользователя для аутентификации в БД. Если установлен параметр auth_user, то любой пользователь, не указанный в auth_file, будет запрошен с помощью запроса auth_query из системного каталога pg_shadow с использованием auth_user. Для прямого доступа к pg_shadow требуются права администратора
jaPooler	pgbouncer.ini	auth_query	SELECT username, passwd FROM pg_shadow WHERE username=\$1	Запрос для извлечения пароля пользователя из базы данных. Для прямого доступа к pg_shadow требуются права администратора.
jadog	jadog.yml	param_connection: conn_string. В рамках него passfile	—	Параметр используется для передачи пути файла с паролем
jadog	jadog.yml	param_connection: conn_string. В рамках него sslrootcert	—	Параметр используется для настройки SSL соединения между jadog и СУБД и отвечает за путь до СА сертификата

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
jadog	jadog.yml	param_connection: conn_string. В рамках него sslcert	—	Параметр используется для настройки SSL соединения между jadog и СУБД и отвечает за путь до сертификата клиента
jadog	jadog.yml	param_connection: conn_string. В рамках него sslkey	—	Параметр используется для настройки SSL соединения между jadog и СУБД и отвечает за путь до закрытого ключа клиента
jadog	jadog.yml	param_connection: conn_string. В рамках него sslmode	—	Параметр используется для настройки SSL соединения между jadog и СУБД и отвечает за режим работы SSL
jadog	jadog.yml	param_connection: conn_string. В рамках него sslcr1	—	Параметр используется для настройки SSL соединения между jadog и СУБД и отвечает за путь до списка отзывов сертификатов
jadog	jadog.yml	param_connection: conn_string. В рамках него user	—	Определяет имя пользователя при работе jadog с СУБД
jadog	jadog.yml	param_connection: conn_string. В рамках него dbname	—	Определяет имя базы данных при работе jadog с СУБД
jadog	jadog.yml	param_ssl:ssl	false	Включает режим защищенных соединений jadog-jadog и jadog – jadog_ctl
jadog	jadog.yml	param_ssl:ssl_ca_file	—	Параметр используется для настройки SSL соединения jadog-jadog и jadog – jadog_ctl и отвечает за путь до СА сертификата
jadog	jadog.yml	param_ssl:ssl_cert_file	—	Параметр используется для настройки SSL соединения jadog-jadog и jadog – jadog_ctl и отвечает за путь до сертификата клиента
jadog	jadog.yml	param_ssl:ssl_crl_file	—	Параметр используется для настройки SSL соединения jadog-jadog и jadog – jadog_ctl и отвечает за путь до списка отзыва
jadog	jadog.yml	param_ssl:ssl_key_file	—	Параметр используется для настройки SSL соединения jadog-jadog и jadog – jadog_ctl и отвечает за путь до закрытого ключа клиента
jadog	jadog.yml	param_jadog: interconnect user	admin	Определяет имя пользователя для подключения jadog к jadog
JDS	appsettings.json	DefaultConnection	Нет	Строка подключения к служебной БД в формате «ключ=значение», которая может содержать пароль, в том

№ изменения: \_\_\_\_\_ Подпись отв. лица: \_\_\_\_\_ Дата внесения изм: \_\_\_\_\_

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
				случае, если выбран способ аутентификации по логину и паролю. Находится в группе «ConnectionStrings». Настраивается скриптом jds-config.py
JDS	appsettings.json	CAFile	Нет	Содержит полный путь к «CA Bundle» в формате X.509 (CRT), содержащем сертификаты, используемые для подписи серверного сертификата СУБД при настройке аутентификации по SSL-сертификату:  <ul style="list-style-type: none"> <li>– сертификат корневого УЦ;</li> <li>– сертификат(ы) промежуточного (промежуточных) УЦ.</li> </ul> Находится в группе «ConnectionSslConfigurator \ Connections \ DefaultConnection» Настраивается скриптом jds-config.py
JDS	appsettings.json	ClientPfxFile	Нет	Содержит полный путь к контейнеру в формате PKCS#12 (PFX), содержащем:  <ul style="list-style-type: none"> <li>– клиентский сертификат;</li> <li>– закрытый ключ к клиентскому сертификату;</li> <li>– сертификаты, используемые для подписи клиентского сертификата (вся цепочка, включая сертификат корневого УЦ).</li> </ul> Находится в группе «ConnectionSslConfigurator \ Connections \ DefaultConnection» Настраивается скриптом jds-config.py
JDS	appsettings.json	ClientPfxPassword	Нет	Содержит пароль, в том случае, если сгенерированный клиентский сертификат, указанный в настройке «ClientPfxFile», защищён паролем. Находится в группе «ConnectionSslConfigurator \ Connections \ DefaultConnection» Настраивается скриптом jds-config.py
JDS	appsettings.json	CheckServerCertificateRevocation	True	Признак «Проверять серверный сертификат по списку отозванных сертификатов» Находится в группе «ConnectionSslConfigurator \ Connections \ DefaultConnection»

№ изменения: \_\_\_\_\_ Подпись отв. лица: \_\_\_\_\_ Дата внесения изм: \_\_\_\_\_

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
				Настраивается скриптом jds-config.py
JDS	appsettings.json	Key	UUID-значение	Начальное значение токена безопасности Находится в группе «JwtTokenConfig»
JDS	appsettings.json	Issuer	URL сайта JDS в инфраструктуре, с указанием протокола доступа (HTTPS)	Параметр «Издатель» токена безопасности Находится в группе «JwtTokenConfig»
JDS	appsettings.json	Audience	URL сайта JDS в инфраструктуре, с указанием протокола доступа (HTTPS)	Параметр «Получатель» токена безопасности Находится в группе «JwtTokenConfig»
JDS	appsettings.json	AccessTokenExpiration	480	Время действия токена безопасности, в секундах. После истечения этого времени токен автоматически обновляется. Находится в группе «JwtTokenConfig»
JDS	appsettings.json	RefreshTokenExpiration	960	Время действия токена безопасности, в секундах. После истечения этого времени токен автоматически обновляется Находится в группе «JwtTokenConfig»
JDS	appsettings.json	BaseAddress	Нет	URL сервиса Explain, развернутого в инфраструктуре, включая протокол
JDS	appsettings.json	UseSsl	False	Признак «Использовать SSL-сертификат для аутентификации на jaDog»
JDS	appsettings.json	CAFile	Нет	Путь к файлу корневого CA-сертификата для подключения к jaDog
JDS	appsettings.json	CertFile	Нет	Путь к файлу клиентского сертификата для подключения к jaDog
JDS	appsettings.json	CrlFile	Нет	Путь к файлу списка отозванных сертификатов, используемых для проверки при к jaDog

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
JDS	appsettings.json	KeyFile	Нет	Путь к файлу закрытого ключа клиентского сертификата для подключения к jaDog
JDS	appsettings.json	SslEngine	Нет	Имя криптографического модуля
JDS doctor	appsettings.json	DefaultConnection	Нет	Строка подключения к служебной БД в формате «ключ=значение», которая может содержать пароль, в том случае, если выбран способ аутентификации по логину и паролю. Находится в группе «ConnectionStrings». Настраивается скриптом jds-config.py
JDS doctor	appsettings.json	CAFile	Нет	Содержит полный путь к «CA Bundle» в формате X.509 (CRT), содержащем сертификаты, используемые для подписи серверного сертификата СУБД при настройке аутентификации по SSL-сертификату: <ul style="list-style-type: none"> <li>– сертификат корневого УЦ;</li> <li>– сертификат(ы) промежуточного (промежуточных) УЦ.</li> </ul> Находится в группе «ConnectionSslConfigurator \ Connections \ DefaultConnection» Настраивается скриптом jds-config.py
JDS doctor	appsettings.json	ClientPfxFile	Нет	Содержит полный путь к контейнеру в формате PKCS#12 (PFX), содержащем: <ul style="list-style-type: none"> <li>– клиентский сертификат;</li> <li>– закрытый ключ к клиентскому сертификату;</li> <li>– сертификаты, используемые для подписи клиентского сертификата (вся цепочка, включая сертификат корневого УЦ).</li> </ul> Находится в группе «ConnectionSslConfigurator \ Connections \ DefaultConnection» Настраивается скриптом jds-config.py
JDS doctor	appsettings.json	ClientPfxPassword	Нет	Содержит пароль, в том случае, если сгенерированный клиентский сертификат, указанный в настройке «ClientPfxFile», защищён паролем. Находится в группе «ConnectionSslConfigurator \ Connections \ DefaultConnection»

№ изменения: \_\_\_\_\_ Подпись отв. лица: \_\_\_\_\_ Дата внесения изм: \_\_\_\_\_



Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
				Настраивается скриптом jds-config.py
JDS doctor	appsettings.json	CheckServerCertificateRevocation	True	Признак «Проверять серверный сертификат по списку отозванных сертификатов» Находится в группе «ConnectionSslConfigurator \ Connections \ DefaultConnection» Настраивается скриптом jds-config.py
keepalived	keepalived.conf	script	—	в секции vrrp_script pg_check указан путь для файла проверки смотрим строку подключения в этом скрипте
ja_Hipe_Cluster	postgresql.conf	citushostname	—	Устанавливает hostname при соединении самим с собой (необходимо при настройке кластера по SSL)
ja_Hipe_Cluster	SQL	pg_dist_authinfo	—	Смотрим строку подключения в таблице для каждого узла
pgAudit	—	—	—	Авторизация и аутентификация не требуется
pgSQL-HTTP	—	—	—	Авторизация и аутентификация не требуется
PostGIS	—	—	—	Авторизация и аутентификация не требуется
TDS_FDW	SQL параметры	servername	—	IP Адрес или DNS имя MS SQL сервера
TDS_FDW	SQL параметры	port	—	Порт MS SQL сервера
TDS_FDW	SQL параметры	Database	—	Имя БД MS SQL сервера
TDS_FDW	SQL параметры	Dbuse	0	0 – подключение идет к базе данных, указанной в параметр database 1 – подключение идет к базе данных, полученной вызовом dbuse()
grafana	Аутентификация осуществляется посредством JWT токена через Pomerium	GF_AUTH_SIGNOUT_REDIRECT_URL	—	Подписывает пользователей из Pomerium, когда они выходят из Grafana
grafana	переменная окружения	GF_AUTH_JWT_ENABLED	disable	Включает JWT аутентификацию
	переменная окружения	GF_AUTH_JWT_HEADER_NAME	—	Указывает на имя HTTP заголовка, в котором расположен JWT токен
grafana	переменная окружения	GF_AUTH_JWT_EMAIL_CLAIM	—	связывает email_claim в JWT с электронной почтой пользователя Grafana

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
grafana	переменная окружения	GF_AUTH_JWT_JWK_SET_URL	—	Указывает на URL-адрес с ключом подписи для проверки
grafana	переменная окружения	GF_AUTH_JWT_CACHE_TTL	60m	Время жизни JWT токена
grafana	defaults.ini	disable initial admin creation	false	Отключает создание пользователя admin при первом запуске
grafana	defaults.ini	admin_user	Admin	Административный пользователь
grafana	defaults.ini	admin_password	admin	Административный пароль
grafana	defaults.ini	secret_key	—	Для подписи некоторых параметров источника данных, таких как секреты и пароли, используется формат шифрования AES-256 в режиме CFB
grafana	defaults.ini	disable_gravatar	false	Использование Gravatar для картинок (аватарок) профилей
grafana	defaults.ini	data_source_proxy_whitelist	—	Определяет белый список разрешенных IP-адресов или доменов с портами, которые будут использоваться в URL-адресах источников данных с помощью прокси-сервера Grafana Data Source
grafana	defaults.ini [auth.generic_oauth]	enabled = true	—	Активирует авторизацию по токену
grafana	defaults.ini [auth.generic_oauth]	allow_sign_up = true	—	Указывает, что всегда должна быть подпись токеном
grafana	defaults.ini [auth.generic_oauth]	name = Auth0	—	Задаёт имя авторизирующего сервера
grafana	defaults.ini [auth.generic_oauth]	client_id = <client id>	—	Идентификатор клиента
grafana	defaults.ini [auth.generic_oauth]	client_secret = <client secret>	—	Секретный ключ, выданный авторизационным центром
grafana	defaults.ini [auth.generic_oauth]	auth_url = https://<domain>/authorize	—	URL адрес запроса на авторизацию
grafana	defaults.ini [auth.generic_oauth]	token_url = https://<domain>/oauth/token	—	URL адрес проверки токена
grafana	defaults.ini [auth.generic_oauth]	api_url = https://<domain>/userinfo	—	URL адрес доступа в API
prometheus	web.yml	tls_server_config: cert_file: prometheus.crt key_file: prometheus.key	—	Закрытый ключ и сертификат для SSL

№ изменения: \_\_\_\_\_ Подпись отв. лица: \_\_\_\_\_ Дата внесения изм: \_\_\_\_\_

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
prometheus	web.yml	client_auth_type : RequireAndVerifyClientCert	NoClientCert	Показывает режим аутентификации
prometheus	web.yml	min_version	TLS12	Минимальная версия TLS протокола
prometheus	web.yml	max_version	TLS13	Максимальная версия TLS протокола
prometheus	web.yml	prefer_server_cipher_suites: true	true	Выбор предпочтении шифров: <ul style="list-style-type: none"> <li>– true – используются шифры сервера;</li> <li>– false – клиента</li> </ul>
prometheus	web.yml	curve_preferences	—	Наименование ECDHE кривой
prometheus	web.yml	basic_auth_users: <string>: <secret>	—	Базовая HTTP авторизация, Имя пользователя : base64(hash)
node_exporter	config.yml	cert_file: /etc/node_exporter/ssl/node_ex porter.crt key_file: /etc/node_exporter/ssl/node_ex porter.key	—	Закрытый ключ и сертификат для SSL
node_exporter	config.yml	basic_auth_users: prometheus: "хеш пароля"	—	Настройка базовой HTTP авторизации
windows_exporte r	web.yml tls_server_config:	cert_file: <полный путь до файла сертификата>	—	Сертификат для TLS соединения
windows_exporte r	tls_server_config:	key_file: <полный путь до файла ключа>	—	Закрытый ключ для TLS соединения
windows_exporte r	tls_server_config:	cert: строка сертификата в формате pem	—	Сертификат для TLS соединения
windows_exporte r	tls_server_config:	key: строка ключа в формате pem	—	Закрытый ключ для TLS соединения
windows_exporte r	tls_server_config:	client_ca: строка сертификата в формате pem	—	Сертификат центра авторизации
windows_exporte r	tls_server_config:	client_ca_file: <полный путь до файла сертификата>	—	
windows_exporte r	tls_server_config:	client_auth_type: RequireAndVerifyClientCert	NoClientCert	Если нужна аутентификация по сертификату, то указать RequireAndVerifyClientCert

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
windows_exporter	tls_server_config:	prefer_server_cipher_suites: true	true	Выбор предпочтении шифров: – true – используются шифры сервера; – false – клиента
sql_exporter	web.yml tls_server_config:	cert_file: <полный путь до файла сертификата>	—	Сертификат для TLS соединения
sql_exporter	tls_server_config:	key_file: <полный путь до файла ключа>	—	Закрытый ключ для TLS соединения
sql_exporter	tls_server_config:	cert: строка сертификата в формате pem	—	Сертификат для TLS соединения
sql_exporter	tls_server_config:	key: строка ключа в формате pem	—	Закрытый ключ для TLS соединения
sql_exporter	tls_server_config:	client_ca: строка сертификата в формате pem	—	Сертификат центра авторизации
sql_exporter	tls_server_config:	client_ca_file: <полный путь до файла сертификата>	—	
sql_exporter	tls_server_config:	client_auth_type: RequireAndVerifyClientCert	NoClientCert	Если нужна аутентификация по сертификату, то указать RequireAndVerifyClientCert
sql_exporter	tls_server_config:	prefer_server_cipher_suites: true	true	Выбор предпочтении шифров: – true – используются шифры сервера; – false – клиента
postgre_exporter	web.yml tls_server_config:	cert_file: <полный путь до файла сертификата>	—	Сертификат для TLS соединения
postgre_exporter	tls_server_config:	key_file: <полный путь до файла ключа>	—	Закрытый ключ для TLS соединения
postgre_exporter	tls_server_config:	cert: строка сертификата в формате pem	—	Сертификат для TLS соединения
postgre_exporter	tls_server_config:	key: строка ключа в формате pem	—	Закрытый ключ для TLS соединения
postgre_exporter	tls_server_config:	client_ca: строка сертификата в формате pem	—	Сертификат центра авторизации
postgre_exporter	tls_server_config:	client_ca_file: <полный путь до файла сертификата>	—	

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
postgre_exporter	tls_server_config:	client_auth_type: RequireAndVerifyClientCert	NoClientCert	Если нужна аутентификация по сертификату, то указать RequireAndVerifyClientCert
postgre_exporter	tls_server_config:	prefer_server_cipher_suites: true	true	Выбор предпочтении шифров: – true – используются шифры сервера; – false – клиента
jalog	Jalog_server.conf	DBAuthMethod	password	Определяет способ подключения jalog_server к СУБД (ssl или password)
jalog	Jalog_server.conf	DBTLSCertFile	—	Параметр используется для настройки SSL соединения между jalog_server и СУБД и отвечает за путь до сертификата клиента
jalog	Jalog_server.conf	DBTLSKeyFile	—	Параметр используется для настройки SSL соединения между jalog_server и СУБД и отвечает за путь до закрытого ключа клиента
jalog	Jalog_server.conf	DBTLSCAFile	—	Параметр используется для настройки SSL соединения между jalog_server и СУБД и отвечает за путь до CA сертификата
jalog	Jalog_server.conf	DBTLSCRLFile	—	Параметр используется для настройки SSL соединения между jalog_server и СУБД и отвечает за путь до списка отзывов сертификатов
jalog	Jalog_server.conf	DBTLSMode	verify-full	Параметр используется для настройки SSL соединения между jalog_server и СУБД и отвечает за режим проверки SSL сертификатов
jalog	Jalog_server.conf	UseSchannel	—	Только для Windows. Параметр используется для настройки ssl соединения между jalog_server и jalog_agent и отвечает за использование пакета безопасности из ОС (Secure Channel)
jalog	Jalog_server.conf	EngineName	—	Только для Linux. Параметр используется для настройки SSL соединения между jalog_server и jalog_agent и отвечает за название криптографического OpenSSL движка
jalog	Jalog_server.conf	TLSCAFile	—	Параметр используется для настройки SSL соединения между jalog_server и jalog_agent и отвечает за путь до CA сертификата
jalog	Jalog_server.conf	TLSCRLFile	—	Параметр используется для настройки SSL соединения между jalog_server и jalog_agent и отвечает за путь до списка отзывов сертификатов

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
jalog	Jalog_server.conf	TLSCertFile	—	Параметр используется для настройки SSL соединения между jalog_server и jalog_agent и отвечает за путь до сертификата клиента
jalog	Jalog_server.conf	TLSKeyFile	—	Параметр используется для настройки SSL соединения между jalog_server и jalog_agent и отвечает за путь до закрытого ключа клиента
jalog	Jalog_agent.conf	TLSConnect	unencrypted	Параметр используется для настройки SSL соединения между jalog_agent и jalog_server и отвечает за способ подключения jalog_agent к jalog_server (unencrypted или cert)
jalog	Jalog_agent.conf	TLSAccept	unencrypted	Параметр используется для настройки SSL соединения между jalog_agent и jalog_server и отвечает за способ приёма входящих подключений (unencrypted или cert)
jalog	Jalog_agent.conf	UseSchannel	—	Только для Windows. Параметр используется для настройки SSL соединения между jalog_agent и jalog_server и отвечает за использование пакета безопасности из ОС (Secure Channel)
jalog	Jalog_agent.conf	EngineName	—	Только для Linux. Параметр используется для настройки SSL соединения между jalog_agent и jalog_server и отвечает за название криптографического OpenSSL движка
jalog	Jalog_agent.conf	TLSCAFile	—	Параметр используется для настройки SSL соединения между jalog_agent и jalog_server и отвечает за путь до CA сертификата
jalog	Jalog_agent.conf	TLSCRLFile	—	Параметр используется для настройки SSL соединения между jalog_agent и jalog_server и отвечает за путь до списка отзывает сертификатов
jalog	Jalog_agent.conf	TLSCertFile	—	Параметр используется для настройки SSL соединения между jalog_agent и jalog_server и отвечает за путь до сертификата клиента
jalog	Jalog_agent.conf	TLSKeyFile	—	Параметр используется для настройки SSL соединения между jalog_agent и jalog_server и отвечает за путь до закрытого ключа клиента
pgBadger	-	--ssh-program ssh Указывает путь к используемому SSH-клиенту. По умолчанию: ssh.	—	Данный компонент может использоваться для анализа удаленных журналов

№ изменения: \_\_\_\_\_ Подпись отв. лица: \_\_\_\_\_ Дата внесения изм: \_\_\_\_\_

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
		--ssh-port порт Указывает порт SSH для подключения. По умолчанию: 22. --ssh-user имя_пользователя Указывает имя пользователя для подключения. По умолчанию: имя пользователя, запускающего pgbadger. --ssh-identity имя_файла Указывает путь к файлу идентификации. --ssh-timeout секунды Задаёт тайм-аут в секундах на случай сбоя SSH-соединения. По умолчанию: 10. --ssh-option параметры Задаёт список параметров для SSH-соединения.		
fasttrun	postgresql.conf	—	—	—
fulleq	postgresql.conf	—	—	—
ja_csum	postgresql.conf	ja_csum.db_name	postgres	Параметр используется для фоновых процессов компонента, проверяющих контрольные суммы файлов и объектов. Фоновый процесс производит подключение к СУБД по внутренним механизмам. <b>Для взаимодействия с внешней средой не используется</b>
ja_sync_ldap	конфигурация внешних подключений хранится в таблице ja_sync_ldap.profile	поля таблицы: host_ip port login pswd	значение по умолчанию отсутствует; всегда явно задается	Данные из указанных полей используются для установления <b>исходящего</b> соединения из СУБД в службу каталогов по протоколам LDAP/LDAPS

Компонент	Конфигурационный файл	Настройка	Значение по умолчанию	Описание
			Администратором СУБД	
jcs	postgresql.conf	—	—	—
jdv	postgresql.conf	—	—	—
BTreeKNN	postgresql.conf	—	—	—
mchar	postgresql.conf	—	—	—
online_analyze	postgresql.conf	—	—	—
pg_cryogen	postgresql.conf	—	—	—
pg_hint_plan	postgresql.conf	—	—	—
pg_store_plans	postgresql.conf	—	—	—
pg-ulid	postgresql.conf	—	—	—
plantuner	postgresql.conf	—	—	—
plspgsql	postgresql.conf	Утилита wplpgsql. Опции командной строки -h -P -U -W/-w	значение по умолчанию отсутствует; всегда явно задается Администратором СУБД/БД; Разработчиком БД	Данный компонент может использоваться удаленно от СУБД (например, на стороне разработчика БД) и устанавливает <b>исходящее</b> соединение по протоколу libpq <b>SSL соединение не поддерживается</b>
securityprofile	postgresql.conf	—	—	—
sql_firewall	postgresql.conf	—	—	—



## 18. РЕАГИРОВАНИЕ НА ИНЦИДЕНТЫ ИБ

СУБД «Jatoba» обеспечивает противодействие угрозам, представленным в «Банке данных угроз безопасности информации» на официальном сайте <https://bdu.fstec.ru/threat> ФСТЭК России и в частности, представленным в таблице 18.1.

Таблица 18.1 – Сопоставление угроз безопасности с мерами безопасности

Угроза	Описание	Меры ГИС
<u>УБИ. 031</u>	Угроза использования механизмов авторизации для повышения привилегий	УПД.1, УПД.2(1), УПД.4, УПД.5, УПД.6, УПД.6(1)
<u>УБИ. 086</u>	Угроза несанкционированного изменения аутентификационной информации	УПД.1(1, 2)
<u>УБИ. 088</u>	Угроза несанкционированного копирования защищаемой информации	ИАФ.1, ИАФ.4, УПД.1, УПД.1(1, 2), УПД.2, УПД.2(1), УПД.4, УПД.9, РСБ.3, РСБ.6, РСБ.7, РСБ.8
<u>УБИ. 090</u>	Угроза несанкционированного создания учетной записи пользователя	УПД.1, УПД.1 (1, 2), УПД.2, УПД.4, УПД.9
<u>УБИ. 091</u>	Угроза несанкционированного удаления защищаемой информации	УПД.2
<u>УБИ. 100</u>	Угроза обхода некорректно настроенных механизмов аутентификации	ИАФ.1, ИАФ.4
<u>УБИ. 122</u>	Угроза повышения привилегий	УПД.1, УПД.2(1), УПД.4
<u>УБИ. 124</u>	Угроза подделки записей журнала регистрации событий	РСБ.6, РСБ.7
<u>УБИ. 037</u>	Угроза исследования приложения через отчеты об ошибках	РСБ.7
<u>УБИ. 114</u>	Угроза переполнения целочисленных переменных	ОЦЛ.7
<u>УБИ. 158</u>	Угроза форматирования носителей информации	ОДТ.4, ОДТ.5

Организация менеджмента инцидентов информационной безопасности должна соответствовать семейству стандартов СМИБ состоящих из взаимосвязанных стандартов, опубликованных или разрабатываемых, и содержащих несколько ключевых структурных компонентов. К числу этих компонентов относятся:

- нормативные стандарты, устанавливающие требования к СМИБ (ИСО/МЭК 27001);
- требования к органам по сертификации, осуществляющим сертификацию на соответствие ИСО/МЭК 27001 (ИСО/МЭК 27006);
- дополнительные требования, связанные с внедрением СМИБ в конкретных отраслях (ИСО/МЭК 27009).

Противодействие угрозам ИБ состоит из совокупности технических и организационных мероприятий.

К основным, критичным инцидентам ИБ, связанных с эксплуатацией СУБД «Jatoba» относятся:

- нарушение целостности и последующая блокировка пользователей СУБД;
- неудачные попытки входа в СУБД.

Для контроля над работой компонентов «ja\_CSum», «SecurityProfile» и в целом СУБД, целесообразно воспользоваться функциональными возможностями раздела «Уведомления» компонента пользовательского веб-интерфейса для администраторов «Jatoba data safe».

### **18.1. Нарушение целостности и последующая блокировка пользователей СУБД**

Нарушение целостности СУБД будет зафиксировано в журнале аудита СУБД:

- событием компонента «ja\_CSum» с идентификатором 115182106 и сообщением «Целостность объекта нарушена»;
- событием компонента «SecurityProfile» с идентификатором 103118105 и сообщением «Блокирование учетной записи».

В зависимости от требований внутренних регламентов оповестить должностных лиц, участие которых предусмотрено в расследовании и устранении последствий инцидента ИБ.

В журналах аудита СУБД, SIEM и в прочих доступных источниках, установить причину нарушения целостности СУБД.

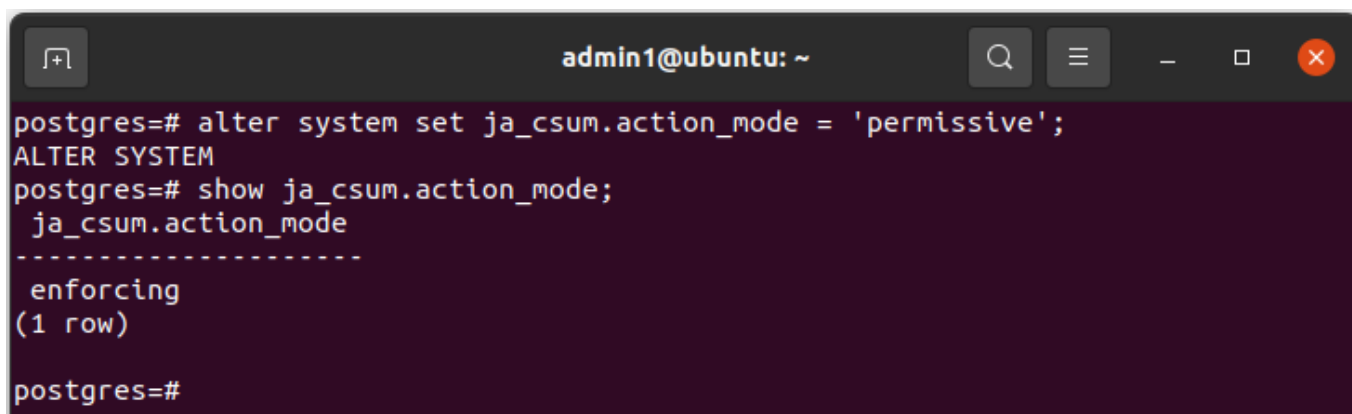
Установив причину, следует восстановить исходное состояние СУБД.

При восстановлении работоспособности СУБД, порядок действий должен быть следующим:

- 1) Войти в СУБД от имени и с правами привилегированного пользователя «postgres» или пользователя, имеющего атрибут «Superuser».
- 2) Перевести компонент «ja\_CSum» в режим информирования «permissive» (см. п. 3.7.1 Руководства по настройке. Часть 14. Контроль целостности. Компонент «ja\_CSum»).

Включение режима информирования выполняется SQL-командой:

```
ALTER SYSTEM set ja_csum.action_mode = 'permissive';
```



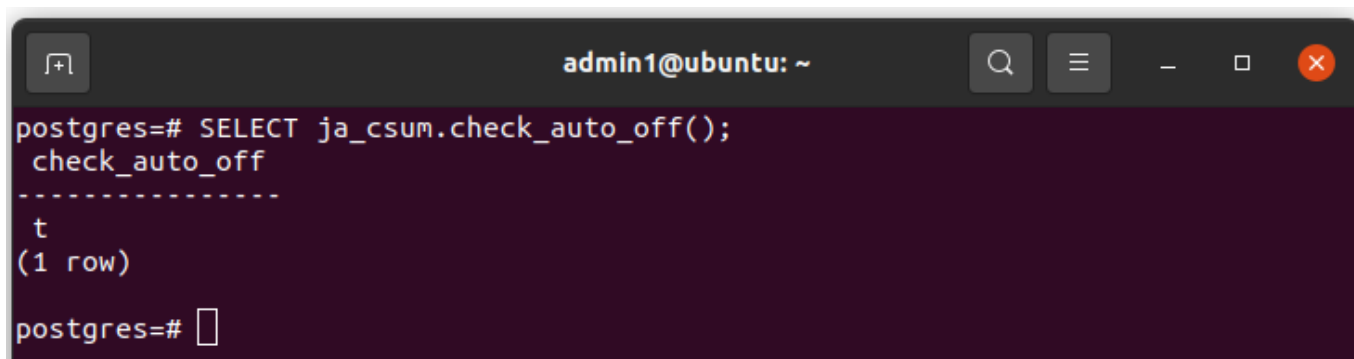
```
admin1@ubuntu: ~  
postgres=# alter system set ja_csum.action_mode = 'permissive';  
ALTER SYSTEM  
postgres=# show ja_csum.action_mode;  
ja_csum.action_mode  
-----  
enforcing  
(1 row)  
postgres=#
```

Рисунок 18.1 – Включение режима информирования «permissive»

- 3) Отключить режим периодической проверки (см. п. 3.5 Руководства по настройке. Часть 14. Контроль целостности. Компонент «ja\_CSum»).

Отключение режима периодической проверки выполняется SQL-командой:

```
SELECT ja_csum.check_auto_off();
```



```
admin1@ubuntu: ~  
postgres=# SELECT ja_csum.check_auto_off();  
check_auto_off  
-----  
t  
(1 row)  
postgres=#
```

Рисунок 18.2 – SQL-команда отключения режима периодической проверки

4) Внести требуемые изменения для приведения СУБД в первоначальное состояние.

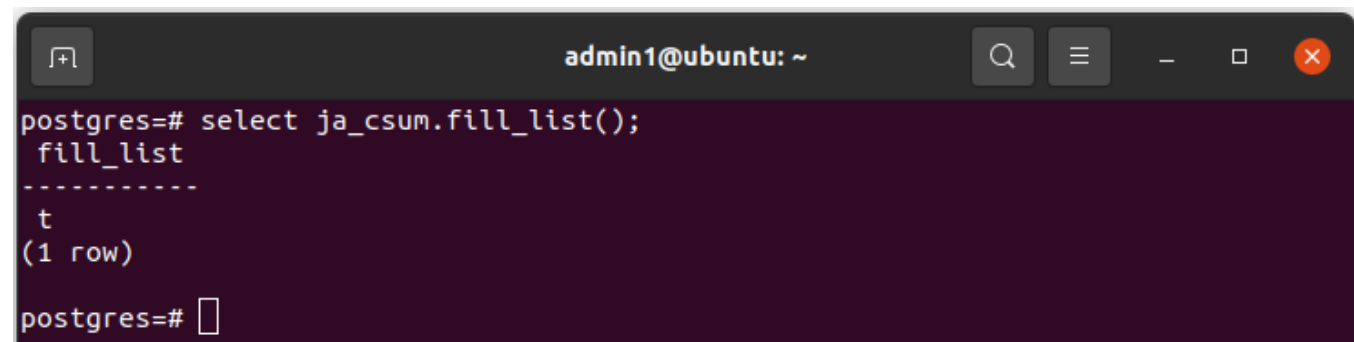
5) При необходимости перезагрузить СУБД.

6) Войти в СУБД от имени и с правами привилегированного пользователя «postgres» или пользователя, имеющего атрибут «Superuser».

7) Обновить файлы с контрольными суммами (см. п. 3.2 Руководства по настройке. Часть 14. Контроль целостности. Компонент «ja\_CSum»).

Список контролируемых файлов создается SQL-командой:

```
SELECT ja_csum.fill_list();
```



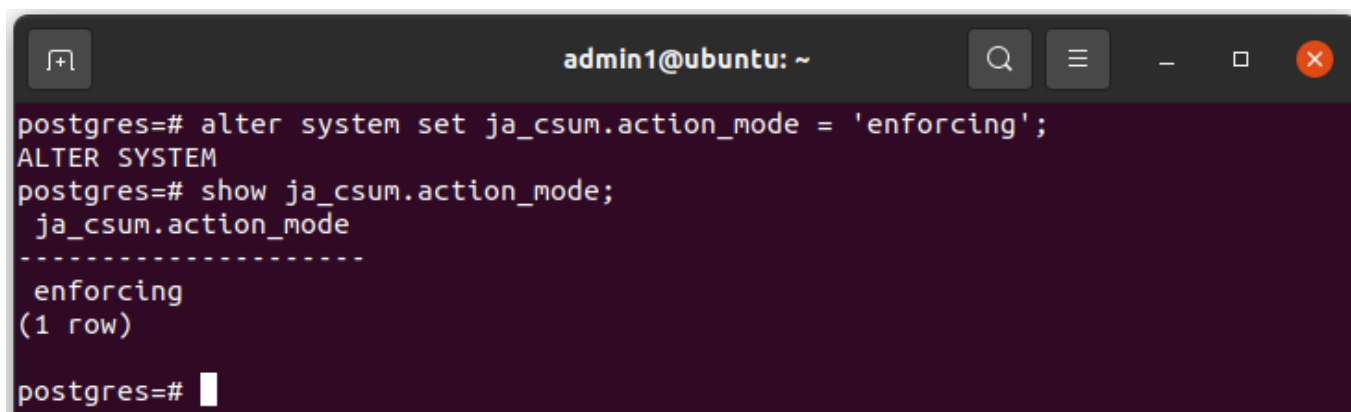
```
admin1@ubuntu: ~  
postgres=# select ja_csum.fill_list();  
fill_list  
-----  
t  
(1 row)  
postgres=#
```

Рисунок 18.3 – Команда создания файла с контрольными суммами

8) Перевести компонент «ja\_CSum» в режим блокирования «enforcing» (см. п. 3.7.2 Руководства по настройке. Часть 14. Контроль целостности. Компонент «ja\_CSum»).

Установка режима блокирования «enforcing» выполняется SQL-командой:

```
ALTER SYSTEM set ja_csum.action_mode = 'enforcing';
```



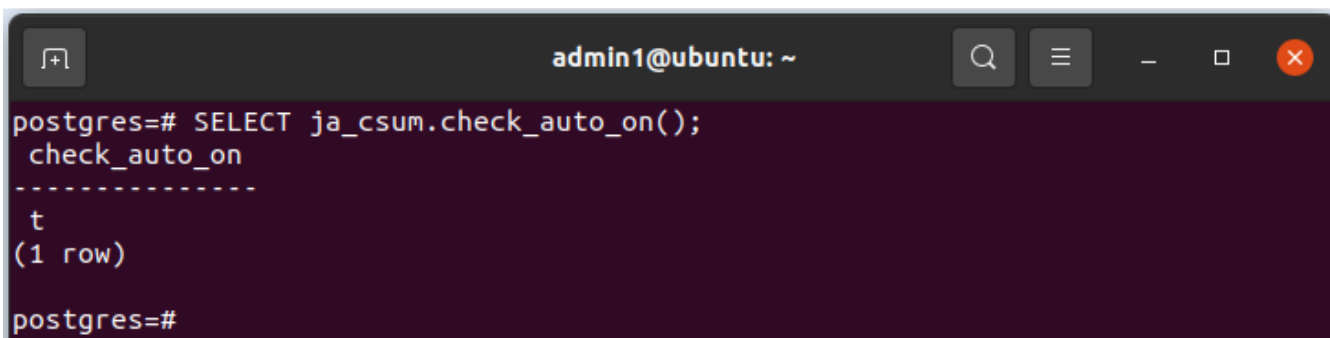
```
admin1@ubuntu: ~  
postgres=# alter system set ja_csum.action_mode = 'enforcing';  
ALTER SYSTEM  
postgres=# show ja_csum.action_mode;  
ja_csum.action_mode  
-----  
enforcing  
(1 row)  
postgres=#
```

Рисунок 18.4 – Включение режима блокирования «enforcing»

9) Включить режим периодической проверки (см. п. 3.5 Руководства по настройке. Часть 14. Контроль целостности. Компонент «ja\_CSum»).

Включение режима периодической проверки выполняется SQL-командой:

```
SELECT ja_csum.check_auto_on();
```



```
admin1@ubuntu: ~  
postgres=# SELECT ja_csum.check_auto_on();  
check_auto_on  
-----  
t  
(1 row)  
postgres=#
```

Рисунок 18.5 – SQL-команда включения режима периодической проверки

10) Установить запрет на создание пользовательских функций.

Запрет на создание пользовательских функций устанавливается SQL-командой:

```
ALTER SYSTEM SET securityprofile.user_function_creation=off;
```

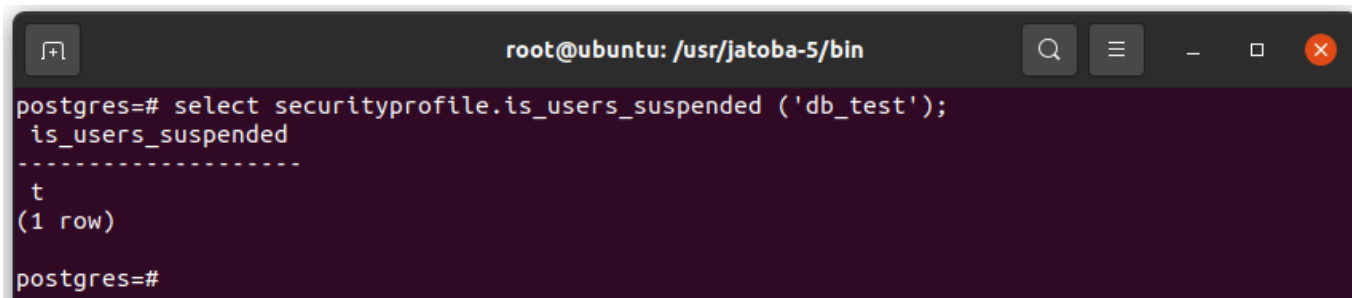
Затем выполнить перезагрузку конфигурации СУБД:

```
select pg_reload_conf();
```

11) Выполнить проверку блокировки пользователей (см. п. 3.5 Руководства администратора).

Вывод наличия блокировки пользователей выполняется SQL-командой:

```
SELECT securityprofile.is_users_suspended ('db_name');
```



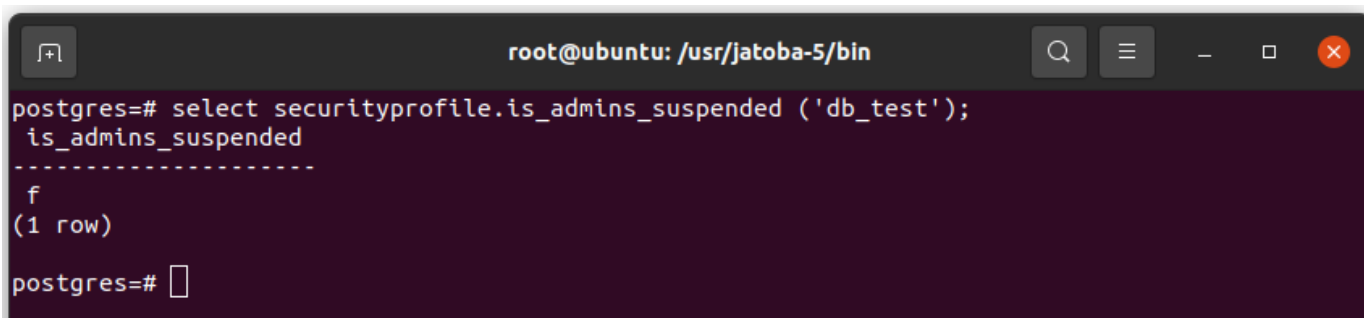
```
root@ubuntu: /usr/jatoba-5/bin
postgres=# select securityprofile.is_users_suspended ('db_test');
 is_users_suspended 
-----
 t
(1 row)
postgres=#
```

Рисунок 18.6 – Вывод состояния блокировки пользователей в БД

12) Выполнить проверку блокировки пользователей (см. п. 6.2.2.2 Руководства администратора).

Вывод наличия блокировки администраторов БД выполняется SQL-командой:

```
SELECT securityprofile.is_admins_suspended ('db_name');
```



```
root@ubuntu: /usr/jatoba-5/bin
postgres=# select securityprofile.is_admins_suspended ('db_test');
 is_admins_suspended 
-----
 f
(1 row)
postgres=#
```

Рисунок 18.7 – Вывод состояния блокировки администраторов БД

13) Вывести информацию о статусе блокировок всех пользователей СУБД (см. п.п. 6.2.1.1, 6.2.1.3.1 Руководства администратора).

Вывод информации о статусе блокировок всех пользователей СУБД выполняется SQL-командой:

```
SELECT * from securityprofile.is_locked ('');
```

```

root@ubuntu: /home/admin1

postgres=# SELECT * from securityprofile.is_locked ('');
role_name | database_name | is_locked |          until_date          | unlock_hint
-----+-----+-----+-----+-----
postgres | *all*         | f         |                               | 
test     | *all*         | t         | 2024-03-25 07:59:51.796105-07 | unlock_account
(2 rows)

postgres=# 

```

Рисунок 18.8 – Вывод списка состояния пользователей

14) Принять решение о разблокировке пользователей или групп пользователей и администраторов СУБД.

15) Разблокировать группу пользователей с игнорированием ошибки (см. п. 6.2.1.2.6 Руководства администратора).

Разблокировка пользователей СУБД, вне зависимости от имеющихся ошибок, выполняется SQL-командой:

```
SELECT securityprofile.resume_users_noerror ('db_name', 0);
```

```

root@ubuntu: /usr/jatoba-5/bin

postgres=# select securityprofile.suspend_users_noerror ('db_test', 0);
suspend_users_noerror
-----
(1 row)

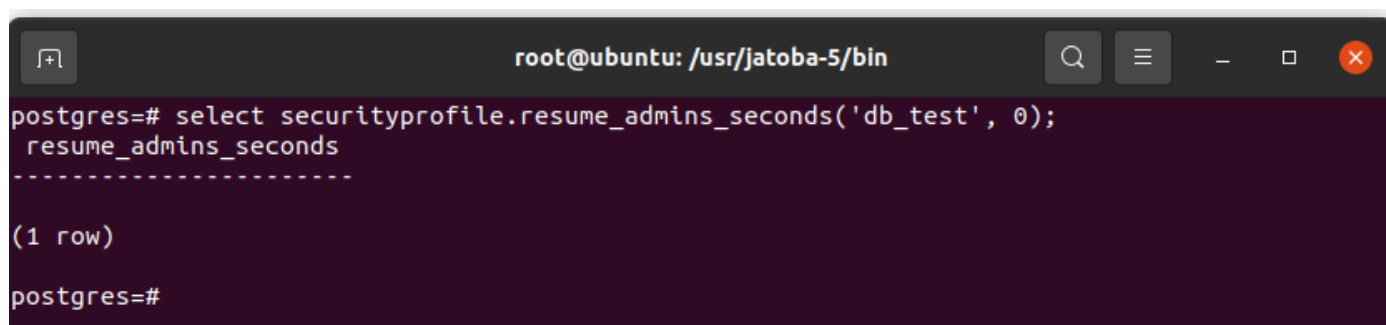
```

Рисунок 18.9 – Выполнение команды разблокировки пользователей

16) Разблокировать группу администраторов БД с игнорированием ошибки (см. п. 6.2.1.3.6 Руководства администратора).

Разблокировка администраторов БД, вне зависимости от имеющихся ошибок, выполняется SQL-командой:

```
SELECT securityprofile.resume_admins_seconds ('db_name', 0);
```



```
root@ubuntu: /usr/jatoba-5/bin
postgres=# select securityprofile.resume_admins_seconds('db_test', 0);
 resume_admins_seconds
-----
(1 row)
postgres=#
```

Рисунок 18.10 – Выполнение команды разблокировки пользователей  
На данном шаге восстановление работоспособности СУБД закончено.

## 18.2. Превышение попыток количества неудачных попыток входа в СУБД

Инцидент ИБ блокировки пользователя СУБД в связи превышение попыток количества неудачных попыток входа в СУБД, может быть следствием:

- попытки перебора паролей для взлома СУБД;
- неаккуратности пользователя СУБД.

В зависимости от требований внутренних регламентов необходимо оповестить должностные лица, участие которых предусмотрено в расследовании и устранении последствий инцидента ИБ.

В журналах аудита СУБД, SIEM и в прочих доступных источниках, установить причину блокировки пользователя СУБД.

Блокирование пользователя СУБД будет зафиксировано в журнале аудита событием компонента «SecurityProfile» с идентификатором 103118105 и сообщением «Блокирование учетной записи».

В ходе проверки важно обратить внимание на следующие аспекты:

- является учетная запись технической (служебной) или пользовательской;
- в случае пользовательской УЗ необходимо установить кому присвоена и статус заявки на разблокирование УЗ в ServiceDesk;
- совпадает имя заявителя и присвоенный ему IP-адрес с IP-адресом компьютера, с которого велось подключение к СУБД;
- какие привилегии, атрибуты предоставлены учетной записи и в какие группы входит.

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------



Когда анализ выясненных обстоятельств покажет отсутствие попыток взлома СУБД, для восстановления УЗ порядок действий должен быть следующим:

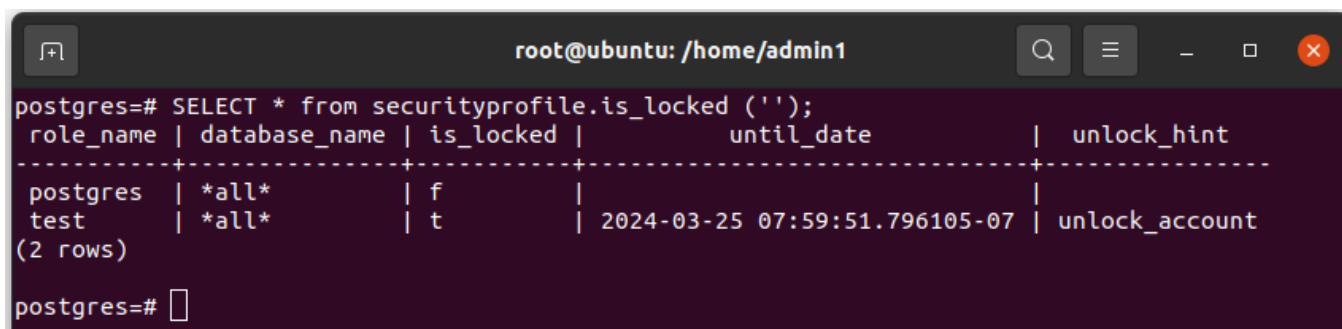
- 1) Проверить установленные блокировки УЗ.

Для проверки факта блокировки и времени, в течение которого она будет действовать, администратору СУБД необходимо выполнить следующую команду:

```
SELECT * from securityprofile.is_locked('имя_пользователя');
```

Вывод информации о всех пользователях выполняется SQL-командой:

```
SELECT * from securityprofile.is_locked ('');
```



```
root@ubuntu: /home/admin1
postgres=# SELECT * from securityprofile.is_locked ('');
 role_name | database_name | is_locked |          until_date          | unlock_hint
-----+-----+-----+-----+-----
 postgres | *all*         | f         |                               |
 test     | *all*         | t         | 2024-03-25 07:59:51.796105-07 | unlock_account
(2 rows)
postgres=#
```

Рисунок 18.11 – Вывод списка состояния пользователей

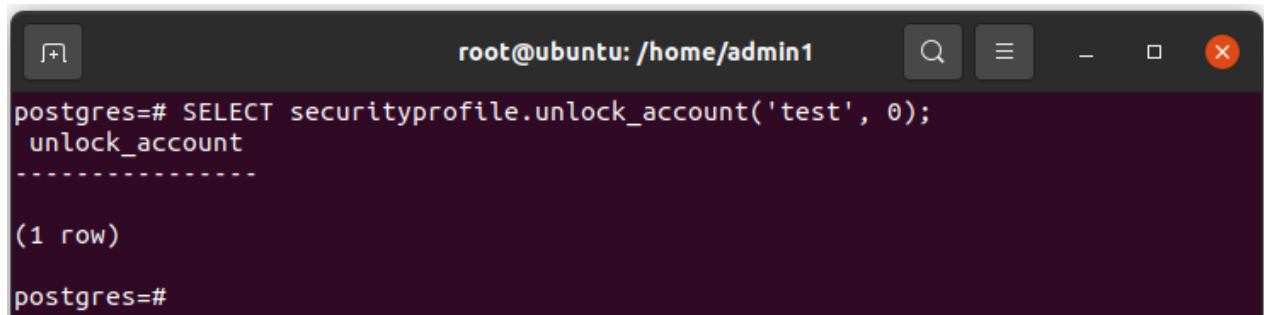
- 2) Разблокировать УЗ.

Для разблокировки учетных записей пользователей администратору СУБД необходимо выполнить следующую команду:

```
SELECT securityprofile.unlock_account ('имя_пользователя',
bigint);
```

Примечание: bigint – задержка, с которой будет выполнено снятие блокировки в днях.

```
SELECT securityprofile.unlock_account('test', 0);
```



A terminal window titled 'root@ubuntu: /home/admin1' with standard window controls. The terminal shows a PostgreSQL session where the command 'SELECT securityprofile.unlock\_account('test', 0);' is entered. The output shows the function name 'unlock\_account' followed by a separator line '-----' and the result '(1 row)'. The prompt 'postgres=#' is visible at the bottom.

```
root@ubuntu: /home/admin1
postgres=# SELECT securityprofile.unlock_account('test', 0);
unlock_account
-----
(1 row)
postgres=#
```

Рисунок 18.12 – SQL-команда блокирования пользователя



## ПРИЛОЖЕНИЕ 1

### Установка службы JDS.PasDoctor

#### Строка подключения к служебной БД JDS

Для корректной работы службы ей требуется подключение к служебной БД «JDS». Строка подключения находится в файле «appsettings.json» и определяется ключом «ConnectionStrings:DefaultConnection».

#### Установка в Windows

Для установки службы Windows рекомендуется использовать утилиту «sc.exe». Следует запускать службу под учетной записью «NetworkService».

Пример вызова «sc.exe» для установки службы с параметрами:

- имя службы «JDS.Doctor»;
- автоматический запуск;
- учетная запись «Network Service»;
- отображаемое имя «JDS Doctor».

```
sc.exe create JDS.Doctor start= auto binpath= C:\Full\Path\To\JDS.PasDoctor.exe obj= "NT AUTHORITY\NetworkService" DisplayName = "JDS Doctor"
```

Вторая команда устанавливает описание для службы.

```
sc.exe description JDS.Doctor "Служба для поиска и исправления проблем с производительностью и безопасностью СУБД. Для управления службой используйте раздел Jatoba Data Safe 'Производительность - Проблемы и решения'."
```

#### Файлы журналов в ОС Windows

По умолчанию служба сохраняет журналы в папке C:\ProgramData\JDS\logs. Никаких действий по созданию папки или назначению прав доступа не требуется.

#### Установка в ОС Linux

Для примера используется Ubuntu 23.04. В других дистрибутивах процедура установки может отличаться.

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

- создать пользователя, под которым будет работать служба:

```
sudo useradd -s /usr/sbin/nologin jds
```

- создать папку для журналов (логов), назначить ей владельца и права:

```
sudo mkdir /var/log/jds  
sudo chown jds:jds /var/log/jds  
sudo chmod 744 /var/log/jds
```

В некоторых дистрибутивах при создании пользователя «jds» группа «jds» не создаётся.

Для просмотра группы по умолчанию служит команда «groups jds».

- создать сервис-файл /etc/systemd/system/jds-doctor.service со следующим содержимым:

```
[Unit]  
Description=JATOBA DATA SAFE Doctor  
  
[Service]  
WorkingDirectory=/opt/jds-doctor  
ExecStart=/opt/jds-doctor/JDS.PasDoctor  
Restart=always  
RestartSec=10  
SyslogIdentifier=jds-doctor  
User=jds  
  
[Install]  
WantedBy=multi-user.target
```

- разрешить и запустить сервис:

```
sudo systemctl daemon-reload  
sudo systemctl enable jds-doctor  
sudo systemctl start jds-doctor
```

## Файлы журналов в Linux

По умолчанию служба сохраняет журналы (логи) в папке /var/log/jds/. Папка должна быть предварительно создана, должен быть изменен владелец и назначены права (см. секцию «Установка в Linux»).

## ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

DDL	–	Data Definition Language — язык описания данных
DML	–	Data Manipulation Language — язык манипулирования данными
SQL	–	Structured Query Language — язык структурированных запросов
БД	–	База данных
ОЗУ	–	Оперативное запоминающее устройство
ОС	–	Операционная система
СУБД	–	Система управления базами данных
ЭВМ		Электронно-вычислительная машина
ЗПС	–	Замкнутая программная среда в ОС Astra Linux Special Edition 1.6 Смоленск — это механизм авторизации на основании контроля целостности файлов с использованием проверки ЭЦП, реализованный в модуле ядра ОС disgsig_verif

[illegible]

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------