Aladdin 2FA Service.

Руководство администратора для Linux

Версия продукта	1.3.5
Статус	Публичный
Дата	15.04.2026
Листов	69

Оглавление

Руководство администратора для Linux 1

1. О документе 4
   - 1.1 Назначение документа 4
   - 1.2 На кого ориентирован документ 4
   - 1.3 Обозначения и сокращения 4
2. Общие сведения 5
   - 2.1 Назначение продукта 5
   - 2.2 Схема сетевого взаимодействия 5
   - 2.3 Сценарии использования 6
     - 2.3.1 Генерация одноразового пароля 6
     - 2.3.2 PUSH–уведомление 6
     - 2.3.3 Телеграм-аутентификатор 9
   - 2.4 Системные требования 9
   - 2.5 Описание пакетов установки 10
3. Установка 11
4. Настройка 12
   - 4.1 Вход 12
   - 4.2 Настройка СУБД 14
     - 4.2.1 Создание базы данных 15
   - 4.3 Настройка подключения к JAS 16
   - 4.4 Настройка интерфейса для мобильных приложений 17
   - 4.5 Загрузка лицензии 17
   - 4.6 Запуск сервиса 18
     - 4.6.1 Успешный запуск 18
     - 4.6.2 Неуспешный запуск 19
5. Удаление сервиса 20
6. Обновление сервиса 21
7. Сбор логов 22
8. Ручная настройка 23
   - 8.1 Настройка 23
   - 8.2 Формат файла конфигурации 23
   - 8.3 Обязательные ключи 24
     - 8.3.1 Настройки подключения к СУБД 24
     - 8.3.2 Настройки внутреннего сетевого интерфейса для JAS 26
     - 8.3.3 Настройки внешнего сетевого интерфейса для мобильных приложений 27
   - 8.4 Список необязательных ключей 29
     - 8.4.1 Тип обработки PUSH-аутентификаций 29
     - 8.4.2 Настройки журналирования сервера Aladdin 2FA Service 29
   - 8.5 Примеры файлов конфигураций 30
     - 8.5.1 Типовой вариант настройки Aladdin 2FA Service 30
     - 8.5.2 Пример явного указания порта в FQDN 33
     - 8.5.3 Пример настройки сервиса на СУБД PostgreSQL 33
     - 8.5.4 Пример настройки сервиса Aladdin 2FA Service на СУБД MS SQL 34
     - 8.5.5 Настройка на группу доступности MS SQL Always On 35
     - 8.5.6 Пример настройки Aladdin 2FA Service без TLS на внешнем интерфейсе 36

8.6 Создание базы данных ... 37
8.7 Миграция базы данных с MS SQL в PostgreSQL с помощью утилиты aladdin-2fa-magration-tool ... 37
8.8 Настройка подключения сервера Aladdin 2FA Service к PostgreSQL с использованием TLS ... 38
8.9 Повторная настройка ... 39

9. Настройки сервиса для использования Telegram, для передачи второго фактора ... 41

9.1 Введение ... 41
9.2 Конфигурация Telegram-бота для сервера A2FA ... 41
9.2.1 Регистрация бота в Telegram ... 41
9.2.2 Настройка параметров сервера A2FA ... 42
9.3 Варианты использования TLS для подключения к телеграм-боту ... 43
9.3.1 Настройка телеграм-бота с реверс-прокси (на примере NGINX) ... 43
9.3.2 Настройка телеграм-бота без реверс-прокси (без NGINX) ... 43
9.3.3 Создание самоподписанного сертификата ... 44
9.3.4 Вариант с использованием PFX ... 45
9.4 Диагностика ... 45
9.4.1 Проверка соединения ... 45
9.4.2 Проверка настройки TLS ... 46

10. Информация по мобильному приложению «Aladdin 2FA», которую нужно знать при администрировании решения Aladdin 2FA ... 47

Приложение А. Пример конфигурационного файла config.yaml ... 48

Приложение Б. Скрипт для создания базы данных для PostgreSQL ... 51

Приложение В. Пример настройки Aladdin 2FA Service в составе отказоустойчивого кластера pacemaker на ОС RedOS ... 54

Приложение Г. Пример запуска службы A2FA Service от учетной записи без прав root на ОС RedOS ... 61

Приложение Д. Пример установки и настройки PostgresPRO ... 63

Контакты ... 67

Офис (общие вопросы) ... 67
Техническая поддержка ... 67

Список литературы ... 68

Регистрация изменений ... 69

1. О документе

1.1 Назначение документа

Настоящий документ представляет собой описание операций по установке и настройке серверного приложения Aladdin 2FA Service для среды функционирования Linux.

1.2 На кого ориентирован документ

Документ предназначен для администраторов, осуществляющих установку и настройку серверного приложения Aladdin 2FA Service.

1.3 Обозначения и сокращения

В данном документе описан процесс настройки программного обеспечения для создания OTP ключа и его регистрации (с помощью PUSH) в мобильном приложении Aladdin 2FA Service.

На первом этапе работы нужно создать и настроить машину, на которой будет расположен JaCarta Management System (JMS) и JaCarta Authentication System (JAS) в соответствии с инструкциями.

На втором этапе работы нужно установить и настроить программное обеспечение Aladdin 2FA Service. Для этого потребуется установка и настройка следующих компонентов:

- Aladdin 2FA Service – серверное приложение, обрабатывающее запросы на выпуск OTP-токенов и генерацию одноразовых паролей;
- Aladdin 2FA – мобильное приложение, представляющее собой генератор одноразовых паролей (OTP), используемых в качестве второго фактора аутентификации (2FA);
- JAS – сервер усиленной аутентификации пользователей в информационных системах с применением второго фактора аутентификации (2FA);
- JMS – система управления средствами аутентификации, такими как электронные ключи, PUSH-, OTP-, U2F-аутентификаторы и сертификаты пользователей;
- FQDN (Fully Qualified Domain Name – «полное доменное имя») – уникальная последовательность символов, которая идентифицирует конкретный узел в сети интернет. FQDN состоит из имени хоста, домена верхнего уровня (TLD) и всех промежуточных доменов;
- MS SQL Server – СУБД для создания и работы с базами данных, необходимыми для работы приложений JAS, JMS и Aladdin 2FA Service;
- OTP (One-Time Password) – одноразовый пароль. Действителен только для одного сеанса аутентификации;
- PostgreSQL – свободная объектно-реляционная СУБД для создания и работы с базами данных, необходимыми для работы Aladdin 2FA Service;
- PUSH-аутентификация – метод аутентификации с помощью push-уведомлений, используемый на мобильных устройствах.

2. Общие сведения

Aladdin 2FA Service – серверное приложение, предназначенное для автоматического выпуска программного OTP-токена и/или PUSH-токена и дальнейшего их использования с помощью мобильного приложения Aladdin 2FA во взаимодействии с JaCarta Authentication System (JAS).

Для установки и настройки Aladdin 2FA Service необходимо обеспечить наличие готовой инфраструктуры в виде заранее установленных серверов JAS, JMS, а также компонентов, необходимых для их работы в соответствии с актуальной документацией (поставляемой в составе дистрибутива JMS).

Следующим шагом необходимо определиться с СУБД, которую будет использовать сервис Aladdin 2FA (см. Таблица 1), а затем установить и настроить программное обеспечение Aladdin 2FA Service.

2.1 Назначение продукта

Aladdin 2FA – программная платформа, предназначенная для двухфакторной аутентификации, состоящая из мобильного приложения-аутентификатора Aladdin 2FA и серверного приложения Aladdin 2FA Service.

Основным инструментом для пользователя является мобильное приложение Aladdin 2FA, предназначенное для генерации одноразовых паролей и PUSH–уведомлений. Подробное описание про установку и работу мобильного приложения Aladdin 2FA приведено в документе «Aladdin 2FA. Руководство пользователя» [1], которое доступно для загрузки на официальном сайте компания «Аладдин Р. Д.».

2.2 Схема сетевого взаимодействия

Взаимодействие всех компонентов системы показано на рисунке (см. Рисунок 1): для обращения мобильного приложения Aladdin 2FA к серверу Aladdin 2FA Service необходим доступ извне. Типовым вариантом является доступ к Aladdin 2FA через любой прокси-сервер (в документе в примерах используется Nginx).

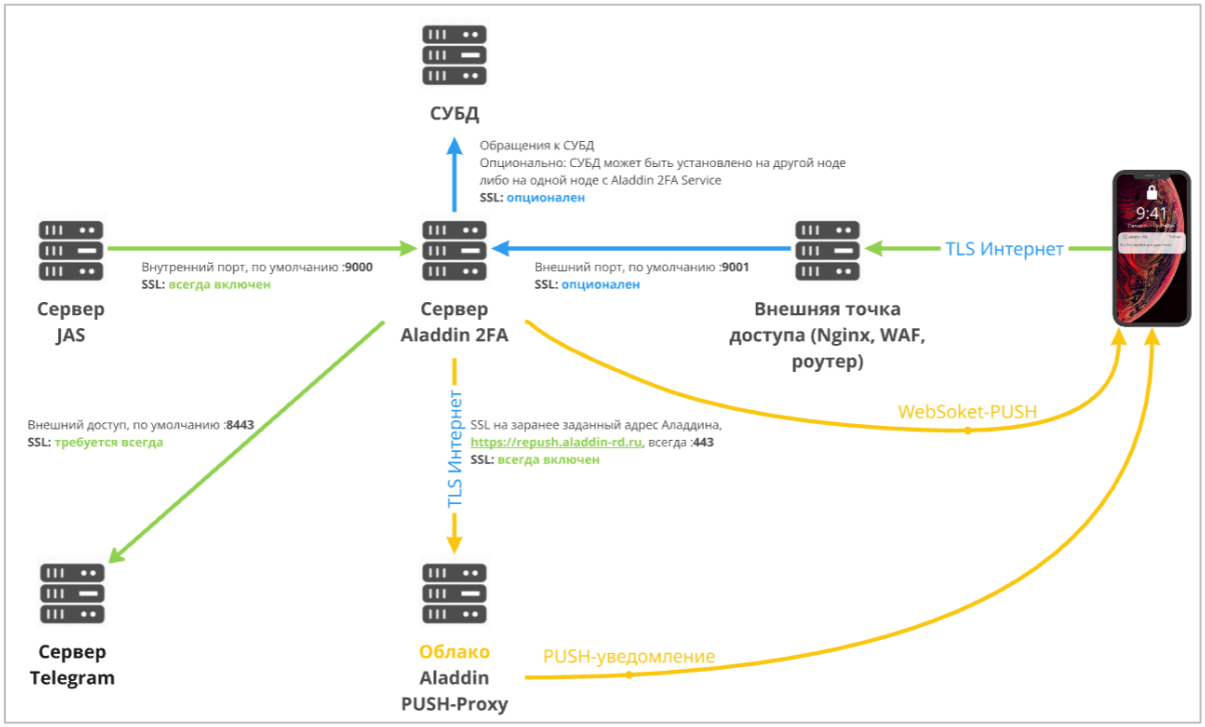


Рисунок 1 – Схема сетевого взаимодействия

2.3 Сценарии использования

Мобильное приложение Aladdin 2FA позволяет осуществлять двухфакторную аутентификацию посредством создания одноразового пароля либо с помощью PUSH–уведомления.

2.3.1 Генерация одноразового пароля

При выборе сценария работы с одноразовым паролем механизм аутентификации схематично будет выглядеть следующим образом (схема процесса приведена ниже (см. Рисунок 2)):

1. Пользователь вводит на своей машине логин, пароль и одноразовый пароль, сгенерированный в мобильном приложении Aladdin 2FA, для проверки на сервере;
2. Сервер прикладной системы отправляет запрос на сервер JAS с целью подтвердить проверку одноразового пароля;
3. На сервере JAS осуществляется проверка. Результат проверки отправляется обратно на запрашиваемый сервер;
4. Сервер прикладной системы возвращает результат на машину пользователя:

- В случае успешной проверки - пользователь, используя аутентификационные данные своей учетной записи, получает доступ к ресурсу (личный кабинет, удаленная сессия и т.д.);
- При отрицательном результате отображается сообщение об ошибке.

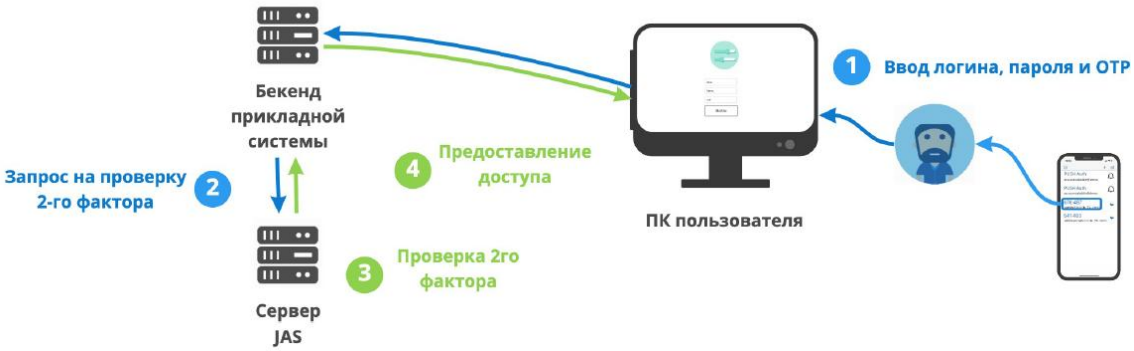


Рисунок 2 – Сценарий аутентификации с использованием одноразового пароля

2.3.2 PUSH–уведомление

При попытке аутентификации в прикладной системе пользователь получает запрос на смартфон (PUSH-нотификацию). Для подтверждения входа в систему или отказа от него необходимо по PUSH-нотификации осуществить вход в мобильное приложение Aladdin 2FA и подтвердить запрос на вход или отказаться, нажав на соответствующую кнопку.

При выборе сценария работы с PUSH-уведомлением механизм аутентификации схематично выглядит следующим образом (схема процесса приведена ниже, см. Рисунок 3):

1. Пользователь на своей машине вводит личные данные для входа в учетную запись;
2. Сформированный запрос проходит цепочку серверов до сервера Aladdin 2FA, который отправляет запрос на PUSH-нотификацию сервер Aladdin PUSH-Proxy и параллельно по каналу WebSocket соединения. Также, сервер Aladdin 2FA отправляет запрос на PUSH-аутентификацию (по каналу аутентификации out-of-band) на смартфон пользователя;
3. Мобильное приложение отображает первое доставленное сообщение PUSH-нотификации доставленное посредством Aladdin PUSH-Proxy или WebSocket соединением. Пользователь принимает запрос на вход. С телефона пользователя запрос отправляется обратно на сервер Aladdin 2FA;

4. На сервере Aladdin 2FA происходит проверка второго фактора для двухфакторной аутентификации;
5. При положительном результате проверки – пользователю предоставляется доступ.

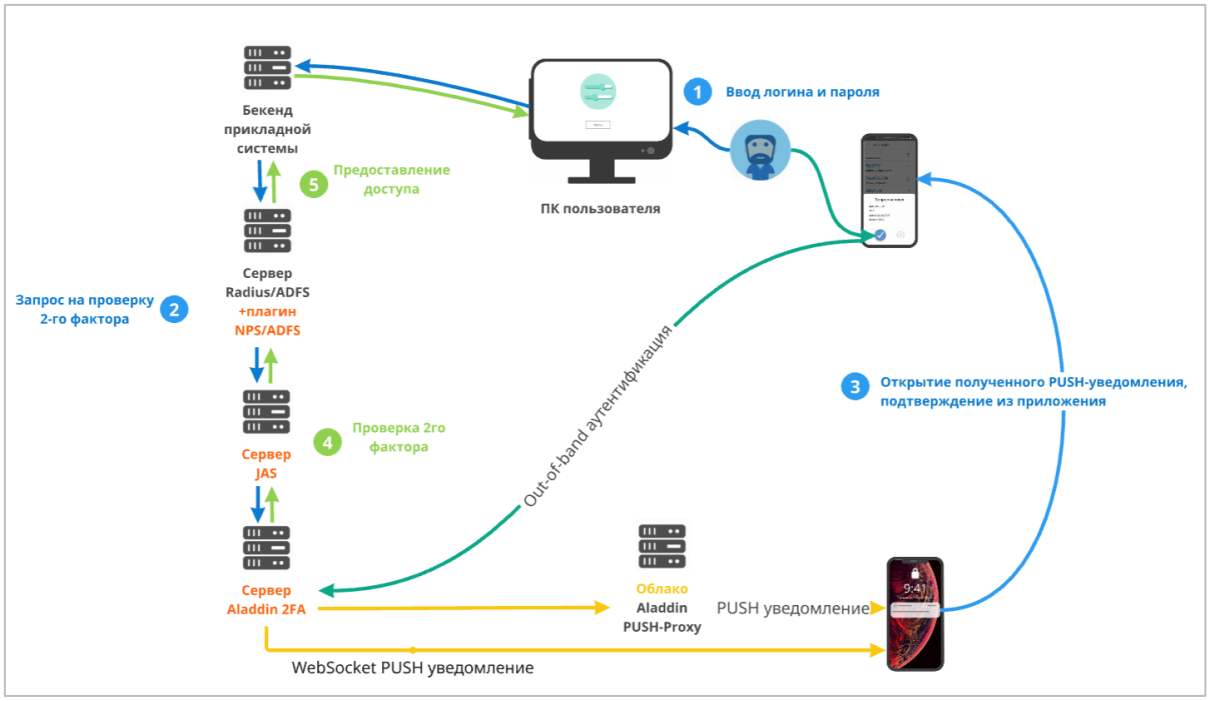


Рисунок 3 - Сценарий аутентификации с использованием PUSH-уведомления

2.3.2.1 WebSocket соединение PUSH-уведомления

Между мобильным приложением и сервером Aladdin 2FA при регистрации первого аутентификатора устанавливается WebSocket соединение, позволяющее минимизировать риски задержек PUSH-уведомлений со стороны PUSH-провайдеров.
В момент формирования запроса на PUSH-уведомление, запрос параллельно отправляется через Aladdin PUSH-Proxy и WebSocket соединение. Отображение первого доставленного PUSH-уведомления и отсутствие дублирования одних и тех же PUSH-уведомлений, регулируется механизмами мобильного приложения. Стабильность WebSocket соединения и поддержание его в фоне, также управляется мобильным приложением.

> *Стабильность работы WebSocket соединения на ОС iOS в фоне ограничена, в связи с архитектурными особенностями этой ОС.*

2.3.2.2 Настройка прокси для использования WebSocket канала PUSH-нотификаций.

При использовании прокси-серера (например Nginx) для использования WebSocket соединения необходимо дополнительно настроить директиву (location) для проксирования пути WebSocket, перед основной директивой (location) для проксирования пути publicServer.

Пример, проксирование настроено с использованием пути /a2fa_public:

```
server {

    listen       80;

    server_name  localhost;
```

```
        # проксирование на publicServer
        location /a2fa_public {
           proxy_pass http://localhost:9001/;
           proxy_set_header Host $host;
        }
    }
```

Добавление проксирования для WebSocket канала:

```
  server {
        listen       80;
        server_name  localhost;

           # Добавляется проксирование на ws - добавляется перед основным
        location ^~ /a2fa_public/ws {
           proxy_pass http://localhost:9001/ws;
           proxy_set_header Host $host;

           # WS
           proxy_http_version 1.1;
           proxy_set_header Upgrade $http_upgrade;
           proxy_set_header Connection "Upgrade";


           # proxy_read_timeout должен быть в 3 раза больше интервала пинга WS - в
конфиге A2FA параметр publicServer.wsPingInterval (по умолчанию 10 минут)
           proxy_read_timeout 30m; # увеличение proxy_read_timeout до 30 минут
        }

        # проксирование на publicServer
        location /a2fa_public {
           proxy_pass http://localhost:9001/;
```

```
            proxy_set_header Host $host;

        }

    }
```

2.3.3 Телеграм-аутентификатор

При выборе сценария работы с телеграм-ботом механизм аутентификации схематично будет выглядеть следующим образом (схема процесса приведена ниже (см. Рисунок 4)):

1. Сервер JAS посылает запрос на создание задачи на регистрацию OTP-аутентификатора, посредством метода /createTGTicket.
2. В ответ Aladdin 2FA Service присылает гиперссылку на телеграм-бот;
3. Сервер JAS пересылает его пользователю;
4. Пользователь сканирует QR-код и переходит в телеграм-бот;
5. Пользователь нажимает кнопку «Старт», связывая свой телеграм профиль с профилем бота;
6. По нажатию кнопки в канале телеграм-бота, TG-бот через сервера TG, отправляет запрос на регистрацию OTP-аутентификатора
7. Aladdin 2FA Service связывает профиль TG с tokenUID. Отправляет ответ телеграмм с каким серийным номером аутентификатора профиль был связан. Через настроенный паблик порт.
8. Профиль телеграм-бота показывает его tokenUID для пользователя;
9. Сервер JAS пересылает в телеграм-бот пользователя значение OTP;
10. Пользователь получает значение OTP в свой телеграм-клиент.

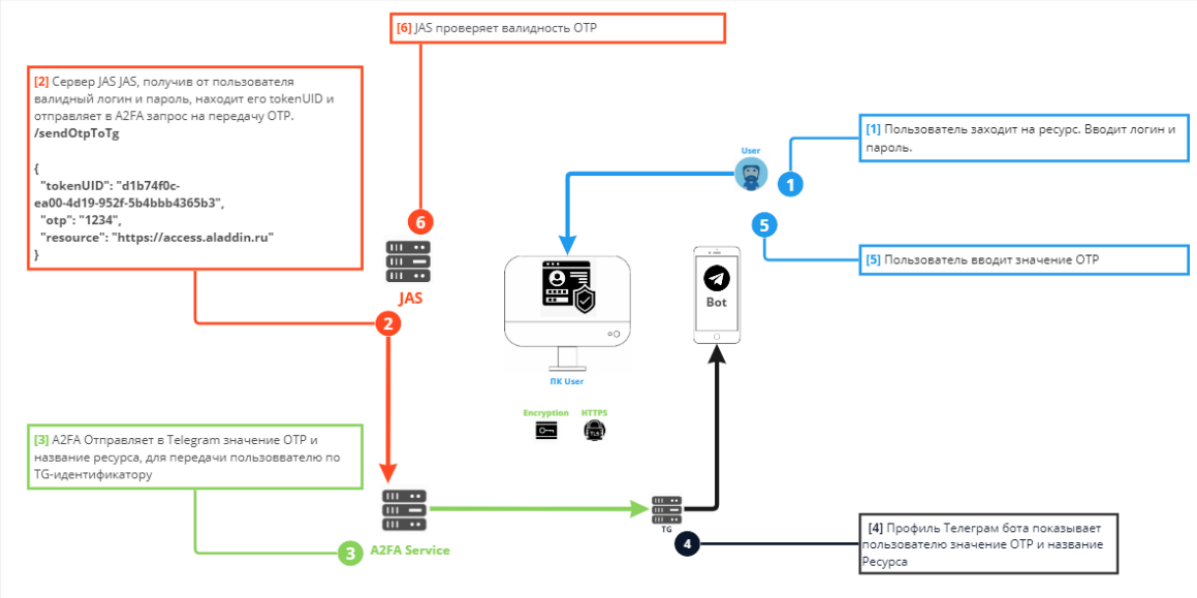


Рисунок 4 - Сценарий аутентификации с использованием телеграм-бота

2.4 Системные требования

Системные требования, необходимые для установки на отдельную машину, приведены ниже (см. Таблица 1).

Таблица 1 – Требования к среде функционирования

Параметр	Значение
Операционная система	• Astra linux SE 1.7.2 и выше; • RedOS 7.3.2 и выше; • CentOS 8 и выше; • Ubuntu 20.04 и выше; • Debian 10 и выше
Сервер СУБД	• PostgreSQL версии 11.0 и выше; • Jatoba 1.9.1-3 и выше
Процессор	4 ядра
Оперативная память (не менее)	8 ГБ
Объём требуемой памяти	80 ГБ

2.5 Описание пакетов установки

Дистрибутив Aladdin 2FA Service включает следующие пакеты установки и обновления (см. Таблица 2):

Таблица 2 - Дистрибутив Aladdin 2FA Service

Файл	Описание
Дистрибутив в форматах `*.deb` и `*.rpm`	`aladdin-2fa-service_x.x.x.xxx_x64.deb`; `aladdin-2fa-service_x.x.x.xxx_x64.rpm`
`cert.pfx`	PFX-контейнер с самозаверенным сертификатом, используемый для быстрой настройки сервиса Aladdin 2FA Service. Пароль от контейнера - `P@ssw0rd!`
`Aladdin 2FA Service. Руководство администратора для Linux.pdf`	Настоящее руководство администратора по настройке Aladdin 2FA Service

3. Установка

> *Для корректной работы серверных компонентов Aladdin 2FA Service, в операционной системе предварительно необходимо настроить синхронизацию времени.*
>
> *Например - синхронизация времени может быть настроена с помощью доменного NTP-сервера, внешнего или с помощью специальных служб, работающих по протоколу NTP (chronyd, timesyncd и т.д.).*

Установка и настройка выполняются от привилегированной учетной записи root. Если учетная запись root отключена, установку необходимо выполнять с использованием sudo перед командой установки или изменения. В зависимости от ОС, выполните установку с используемым пакетным менеджером, например:

`dpkg –i aladdin-2fa-service_x.x.x.xxxx_x64.deb`;

`rpm –i aladdin-2fa-service_x.x.x.xxxx_x64.rpm`.

> *Установить и настроить сервис Aladdin 2FA можно также с использованием Docker-контейнера*

4. Настройка

> *Сервис Aladdin 2FA настраивается с помощью мастера настройки*

4.1 Вход

> *ВНИМАНИЕ! Запускать мастер настройки необходимо через sudo или учетную запись с правами администратора*

Для запуска мастера настройки (см. Рисунок 5) необходимо выполнить следующие команды:

```
cd /sbin/

./aladdin-2fa-wizard
```

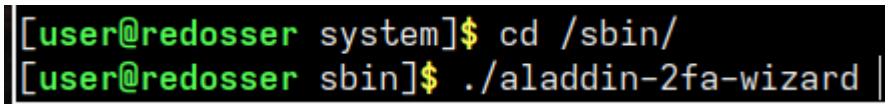


Рисунок 5 – Запуск Мастера настройки

Для одноразового подключения к мастеру настройки с указанным сетевым интерфейсом (см. Рисунок 6) необходимо выполнить следующую команду:

```
./aladdin-2fa-wizard -interface "IP-адрес сетевого адаптера"
```

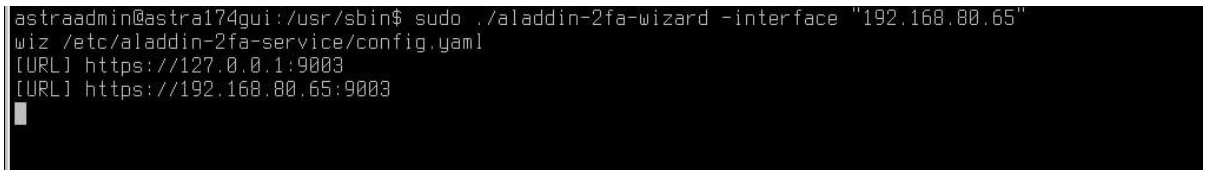


Рисунок 6 – Одноразовое подключение с указанным сетевым интерфейсом

Для того, чтобы использовать сетевой интерфейс для подключения на постоянной основе, необходимо отредактировать файл wizConfig.yaml. Данный файл находится в `/etc/aladdin-2fa-service`.

Для раздела allowedIP, нужно указать адреса сетевых интерфейсов (см. Рисунок 8), на которых будет запускаться мастер настройки, после чего сохранить изменения.

> *Для указания нескольких сетевых интерфесов необходимо, чтобы порт 9003 был не занят*

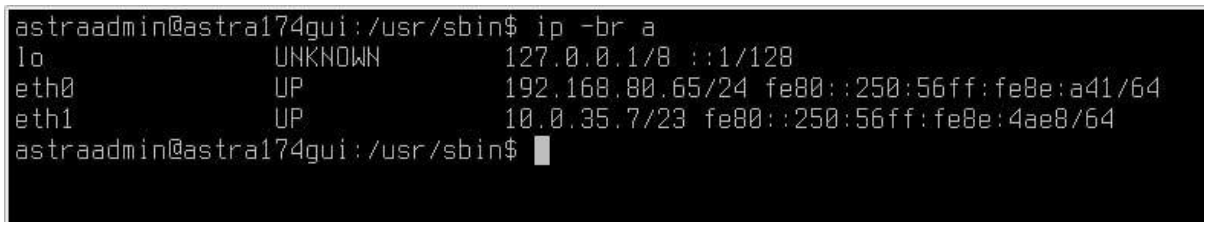


Рисунок 7 – Список доступных интерфейсом на ВМ

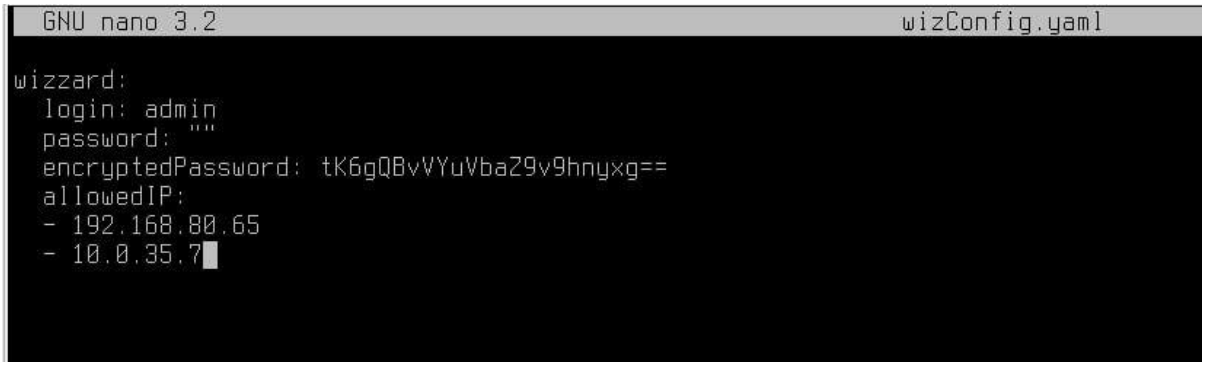


Рисунок 8 – IP-адреса для запуска мастера настройки

После внесения изменений запустить aladdin2-fa-wizard из директории /usr/sbin с помощью команды `sudo ./aladdin-2fa-wizard`.

После запуска отобразятся url адреса по которым можно подключиться к мастеру настройки(см. Рисунок 9).

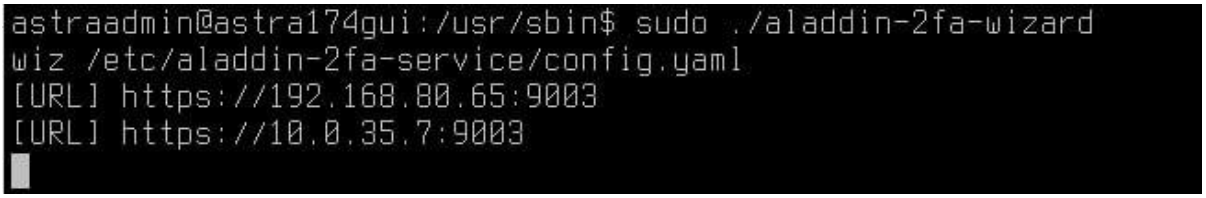


Рисунок 9 – Запуск Мастера настройки через sudo

При первом запуске мастера настройки необходимо придумать логин и пароль для последующих запусков. Заполните поля [Логин], [Пароль], [Повторите пароль] и нажмите кнопку <Сохранить> (см. Рисунок 10). Введенные данные будут сохранены в зашифрованном виде в конфигурационном файле `wizconfig.yaml`.

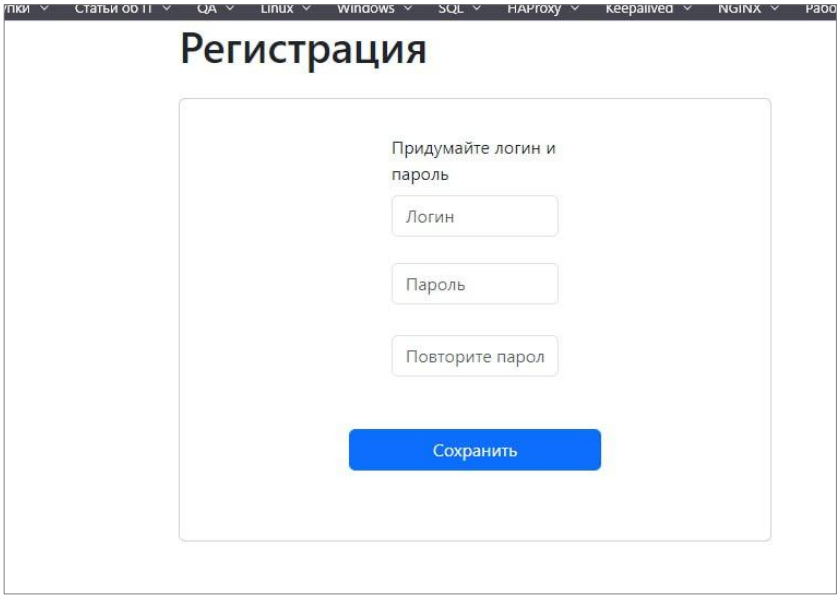


Рисунок 10 – Вход в Мастер настройки при первом запуске

В случае, если пароль и его подтверждение не совпадает, будет отображено соответствующее сообщение (см. Рисунок 11).

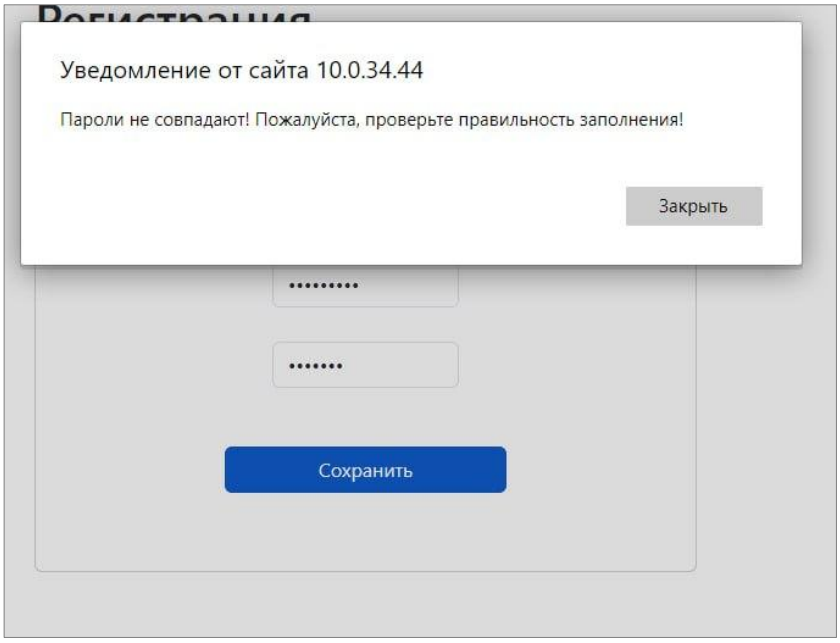


Рисунок 11 – Мастер настройки. Пароль и его подтверждение не совпадают

При последующих входах форма для входа будет выглядеть как на рисунке ниже (см. Рисунок 12).

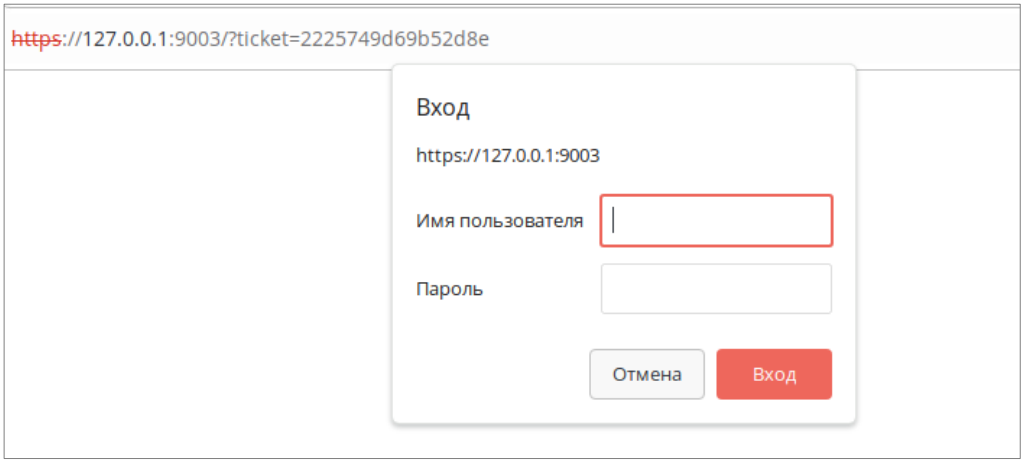


Рисунок 12 – Мастер настройки. Форма для последующего входа

4.2 Настройка СУБД

На данном этапе настраивается тип СУБД, учетная запись для доступа к СУБД и другие данные. Если сервис Aladdin 2FA Service ранее настраивался - то настройки на вкладке будут автоматически заполнены (см. Рисунок 13).

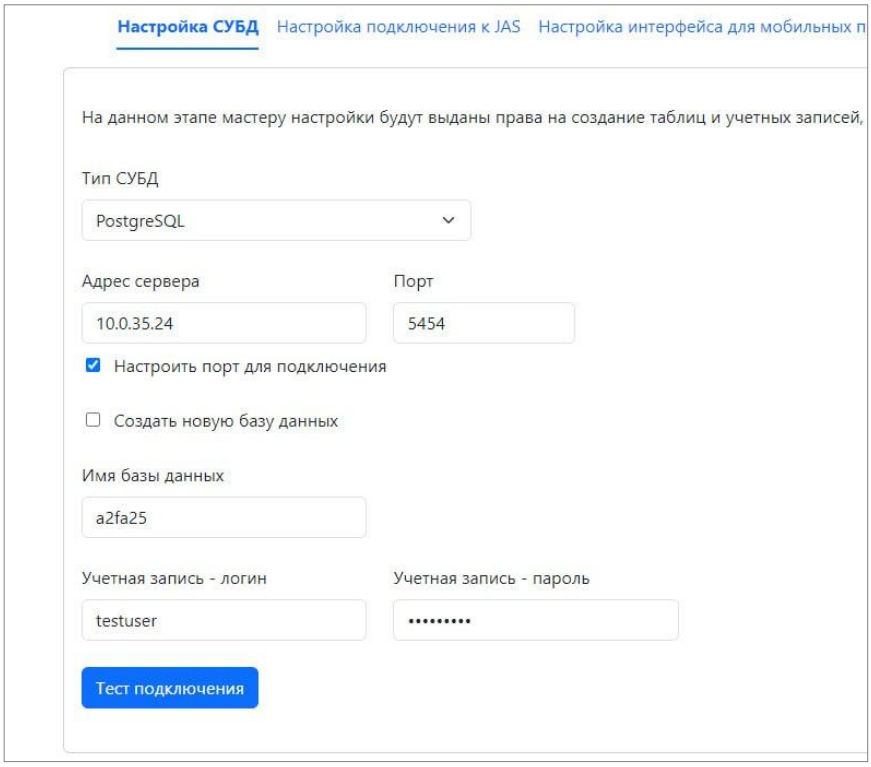


Рисунок 13 – Мастер настройки. Вкладка [Настройка СУБД]. Заполненные поля из конфигурационного файла

4.2.1 Создание базы данных

При активации чекбокса «Создать новую базу данных» на вкладке появятся следующие настройки (см. Рисунок 14):

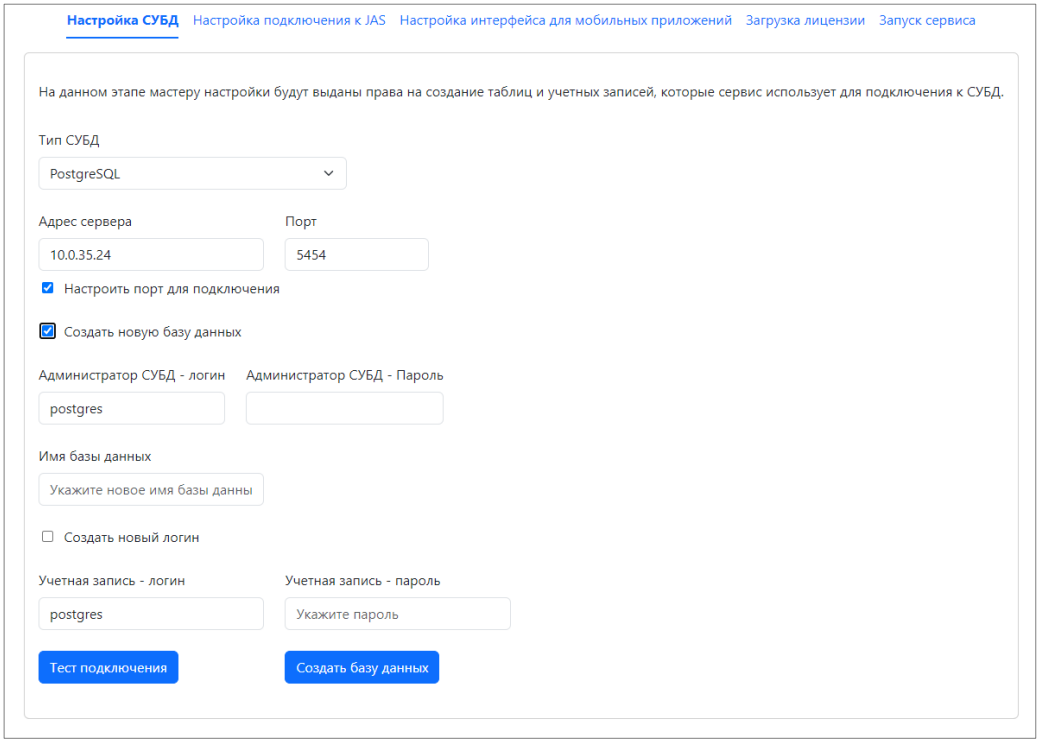


Рисунок 14 – Мастер настройки. Вкладка [Настройка СУБД]. Создание новой базы данных

- [Администратор СУБД – логин] – поле предназначено для указания логина администратора СУБД, с правами которого мастер настройки создаст новую базу данных. Имя учетной записи администратора СУБД зависит от типа СУБД. По умолчанию для PostgreSQL – postgres, для MS SQL – sa;
- [Администратор СУБД – пароль] – поле предназначено для указания пароля администратора базы данных. Пароль используется в сочетании с логином администратора для аутентификации и обеспечения безопасности доступа к базе данных;
- [Имя базы данных] – поле предназначено для указания имени создаваемой базы данных. Указанная база данных будет использоваться сервисом A2FA.
- <Создать новый логин> – если данный чекбокс активирован, на последнем шаге мастера настройки будет создан новый пользователь с правами доступа к базе данных A2FA. Имя и пароль от учетной записи устанавливаются в соответствующих полях – [Учетная запись – логин], [Учетная запись – пароль];
- <Тест подключения> – кнопка, при нажатии на которую происходит тестовое подключение к базе данных с использованием указанных на вкладке данных и сообщает об успешности (см. Рисунок 15) или неуспешности подключения;
- <Создать базу данных> – кнопка, при нажатии на которую будет создана новая база данных в соответствии с указанными на странице настройками.

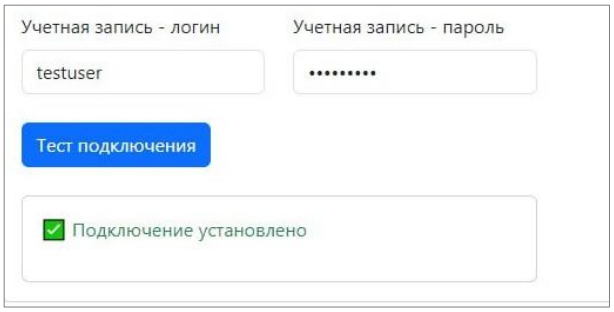


Рисунок 15 – Мастер настройки. Вкладка [Настройка СУБД]. Уведомление об успешном подключении к базе данных

4.3 Настройка подключения к JAS

На вкладке «Настройка подключения к JAS» задаются настройки сетевого интерфейса для сервера JAS. При подключении используется протокол HTTPS.

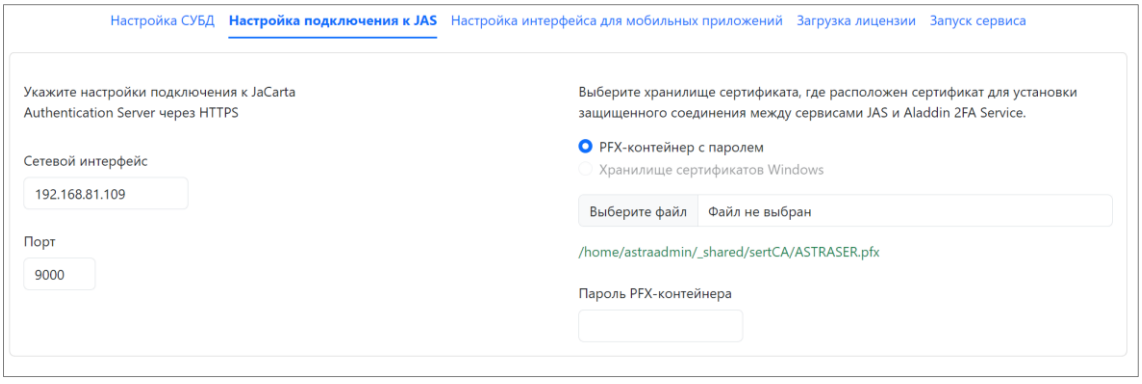


Рисунок 16 – Мастер настройки. Вкладка [Настройка подключения к JAS]. Загруженные настройки

Задайте параметры подключения в соответствии с следующими настройками (см. Рисунок 16):

- [Сетевой интерфейс] – в поле отображен сетевой интерфейс, по которому будет подключаться сервер JAS. Если поле пустое, необходимо ввести значение вручную;

- [Порт] – в поле указать порт для подключения 9000, если он не задан;
- [Выберите хранилище сертификата, где расположен сертификат для установки защищенного соединения между сервисами JAS и Aladdin 2FA Service] – выбрать значение:
  - <PFX-контейнер с паролем> – если не указан контейнер (файл вида `*.pfx`), можно загрузить его с помощью кнопки <Выберите файл>. Указать пароль от pfx-контейнера необходимо в поле [Пароль PFX-контейнера].

4.4 Настройка интерфейса для мобильных приложений

На вкладке «Настройка интерфейса для мобильных приложений» задаются настройки внешнего интерфейса Aladdin 2FA Service.

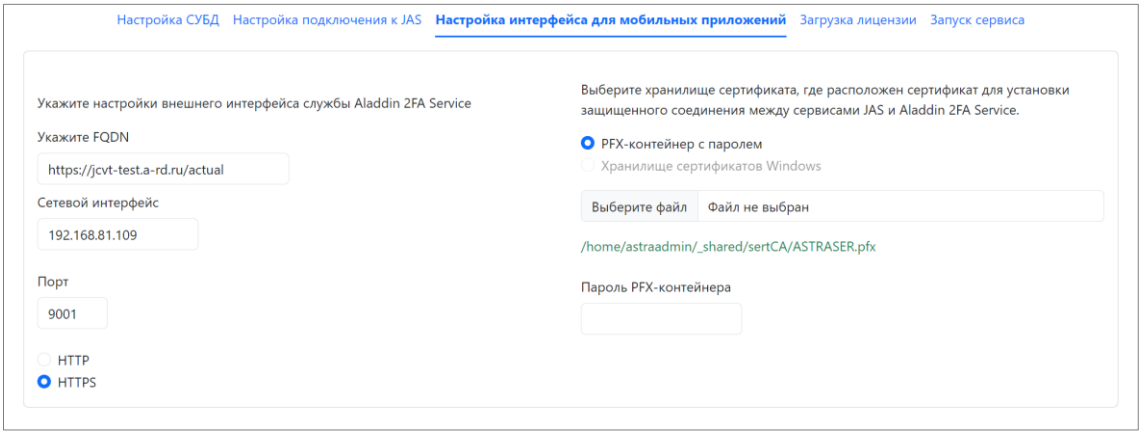


Рисунок 17 – Мастер настройки. Вкладка [Настройка интерфейса для мобильных приложений]

На вкладке заданы следующие настройки (см. Рисунок 17):

- [Укажите FQDN] – указан внешний адрес, обратившись по которому мобильные приложения пользователей смогут обмениваться данными с сервером Aladdin 2FA Service. Этот домен – точка обращений мобильных устройств для запроса безопасной передачи секрета, операций синхронизации и т.д.;
- [Сетевой интерфейс] – указан адрес, соответствующий сетевому интерфейсу, по которому будут инициироваться TCP-подключения со стороны мобильных приложений;
- [Порт] – указан внешний порт. По умолчанию 9001;
- <HTTP>/<HTTPS> – выбран тип соединения;
- <PFX-контейнер с паролем> – если не указан контейнер (файл вида `*.pfx`), можно загрузить его с помощью кнопки <Выберите файл>. Указать пароль от pfx-контейнера необходимо в поле [Пароль PFX-контейнера].

4.5 Загрузка лицензии

Если лицензия уже была загружена на вкладке «Настройка интерфейса для мобильных приложений», то она будет отображаться на вкладке «Загрузка лицензии» (см. Рисунок 18). В противном случае вы можете загрузить файл лицензии с помощью кнопки «Выберите файл».

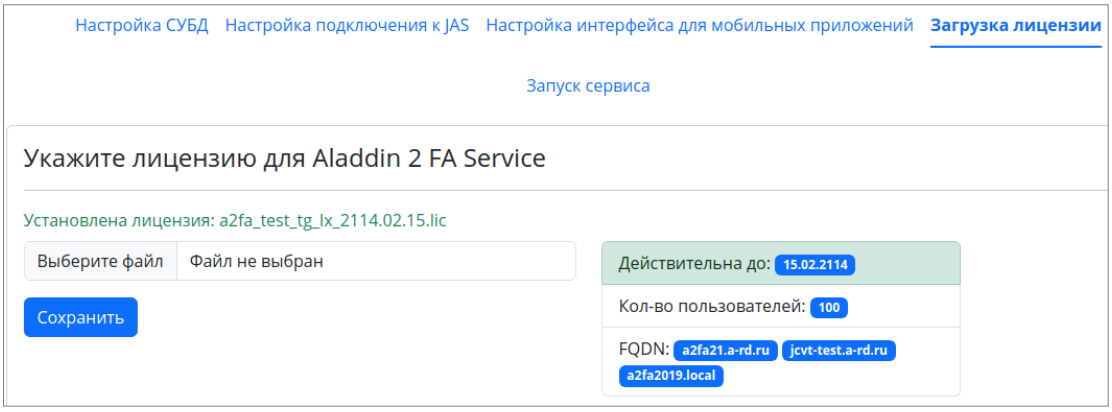


Рисунок 18 – Мастер настройки. Вкладка [Загрузка лицензии]. Отображение загруженной лицензии

На вкладке отображены следующие настройки:

- [Установлена лицензия] – отображается информация по уже загруженной лицензии;
- <Выберите файл> – при нажатии на кнопку будет открыто диалоговое окно для выбора файла лицензии;
- <Сохранить> – при нажатии на кнопку происходит загрузка лицензии;

В случае ошибки при загрузке отображается соответствующее сообщение (см. Рисунок 19).

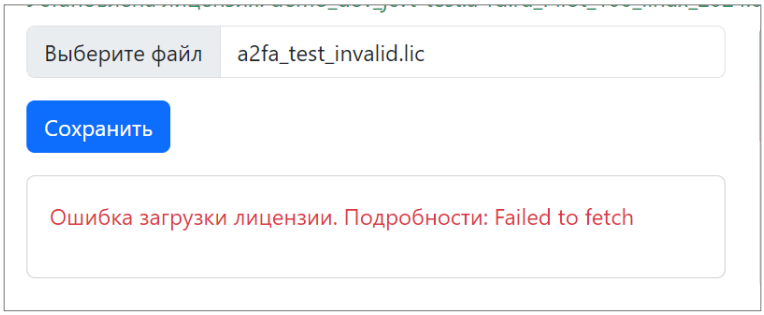


Рисунок 19 – Мастер настройки. Вкладка [Загрузка лицензии]. Ошибка при загрузке лицензии

- [Действительна до] – отображается дата окончания лицензии. Когда срок лицензии будет подходить к концу то цвет на поле «Действительна до» сменится с зеленого на красный;
- [Количество пользователей] – отображается количество пользователей, на которое выпущена лицензия;
- [FQDN] – отображаются домены публикаций, для которых валидна данная лицензия.

4.6 Запуск сервиса

4.6.1 Успешный запуск

Если настройки заданы корректно – на последнем шаге мастера настройки станет доступна кнопка «Сохранить». Успешный запуск показан на рисунке (см. Рисунок 20).

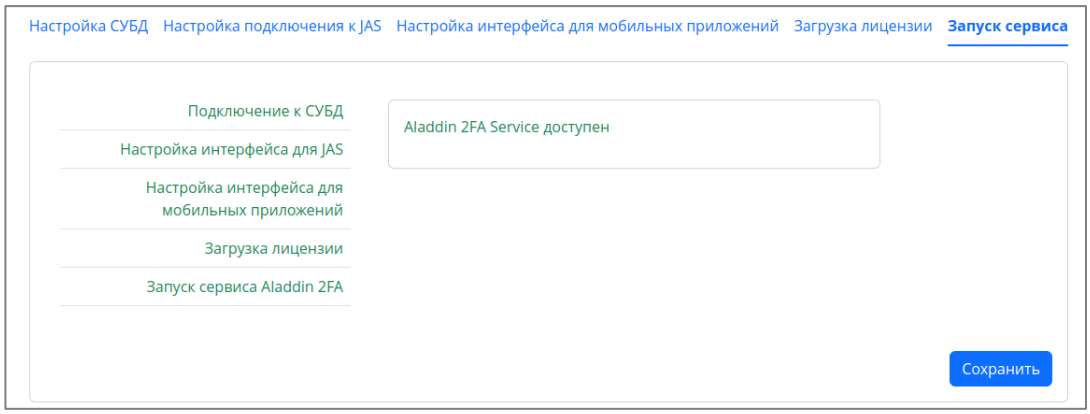


Рисунок 20 – Мастер настройки. Вкладка [Запуск сервиса]. Успешный запуск сервиса

4.6.2 Неуспешный запуск

В случае, если на каком-то шаге заданы некорректные настройки, после нажатия кнопки [Сохранить], на странице красным будет отображаться этап настройки сервиса, на котором была допущена ошибка (см. Рисунок 21).

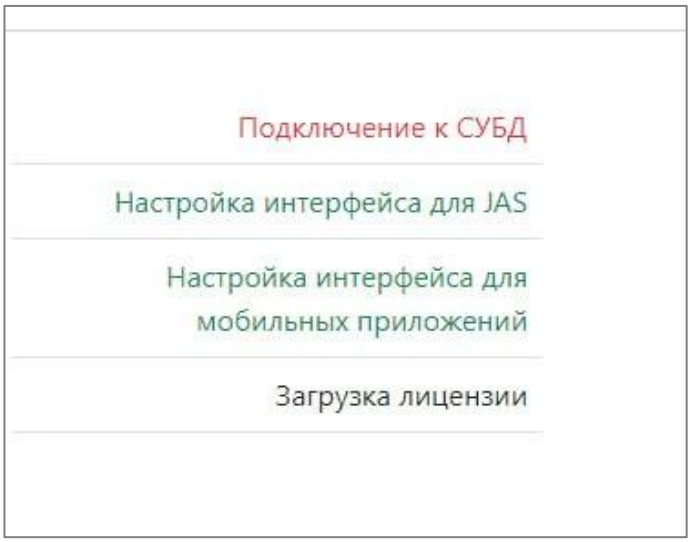


Рисунок 21 – Мастер настройки. Вкладка [Запуск сервиса]. Ошибка при подключении к СУБД

5. Удаление сервиса

Команды для удаления сервиса Aladdin 2FA Service зависит от ОС:

- **RedOS**. Команда для удаления сервиса:

```
dnf remove aladdin-2fa-service
```

После использования этой команды, будет удален только сам пакет сервиса, без удаления файлов конфигурации и зависимостей, связанных с ним. В результате останутся файлы конфигурации (находящиеся в папке `/etc/aladdin-2fa-service`), лицензии сервиса (находящиеся в папке `/etc/aladdin-2fa-service/license`), чтобы при установке новой версии сервиса Aladdin 2FA Service не пришлось настраивать конфигурационный файл повторно и загружать лицензию;

- **AstraLinux**. Команда для удаления сервиса:

```
sudo apt remove aladdin-2fa-service
```

Аналогично пункту выше, после использования данной команды, будет удален только сам пакет сервиса, без удаления файла конфигурации (расположенный по пути `/etc/aladdin-2fa-service`) и лицензий (расположенных по пути `/etc/aladdin-2fa-service/license`).
Если требуется удалить все связанные файлы и зависимости, то воспользоваться командой полного удаления:

```
sudo apt purge aladdin-2fa-service
```

6. Обновление сервиса

Для обновления Aladdin 2FA Service на новую версию необходимо:

1. Остановить работу сервиса командой:

```
sudo systemctl stop aladdin-2fa-service
```

2. Установите новую версию сервиса, выполнив команду в зависимости от операционной системы:
   - AltLinux - `sudo apt-get install *.rpm`
   - RedOS - `sudo yum install *.rpm`
   - AstraLinux - `sudo dpkg -i *.deb`

   где * - полное название пакета;
3. Проверить версию сервиса в терминале командой:

```
sudo aladdin-2fa-service -version
```

7. Сбор логов

Для сбора логов скопируйте из директории `/var/log/aladdin-2fa-service` следующие файлы:

- `main.log`;
- `privateServer.log`;
- `publicServer.log`.

Затем поместите эти файлы в архив и отправьте в техподдержку компании Аладдин.

8. Ручная настройка

8.1 Настройка

Для настройки выполните следующие действия:

1. При установке расширения `*.deb` в папке `/etc/aladdin-2fa-service` будет создана копия конфигурационного файла, работа с которым будет проводится на следующих шагах (см. Рисунок 22);

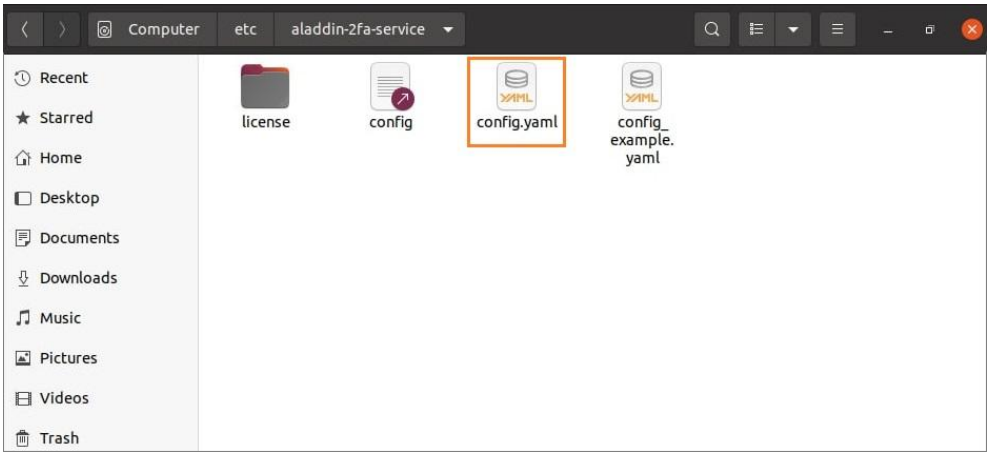


Рисунок 22 – Созданный конфигурационный файл

2. Изменить конфигурационный файл, указав необходимые значения. Описание полей файла приведено в п. 8.3 и п. 8.4;
3. Разместить лицензию в папку `/etc/aladdin-2fa-service/license`;
4. Запустить (или перезапустить) Aladdin 2FA Service одной из следующих команд:

```
sudo service aladdin-2fa-service start

sudo systemctl start aladdin-2fa-service
```

5. Проверить корректность запуска, открыв папку с логами:

   `/var/log/aladdin-2fa-service`:

- В папке должно быть три файла: `main.log`, `privateServer.log`, `publicServer.log`;
- В содержимом файлов не должно быть сообщений об ошибках;
- Убедиться, что пароль от учетной записи СУБД и пароли от PFX-контейнеров при запуске сервиса Aladdin 2FA Service зашифрованы. Примеры шифрования значений параметров в файле конфигурации см. в п. 8.3.

Также проверить корректность запуска Aladdin 2FA Service можно, выполнив одну из следующих команд:

```
sudo systemctl status aladdin-2fa-service

sudo service aladdin-2fa-service status
```

8.2 Формат файла конфигурации

Файл конфигурации представляет собой файл с расширением `*.yaml`. Ключи в файле конфигурации могут представлять собой как имена отдельных параметров, так и группы параметров.

> *Внимание! Формат `*.yaml` чувствителен к отступам. Особое внимание следует уделить вложенным структурам:*

```
publicServer:
  address: 192.168.81.106:9001
  timeout: 90
  tls:
    pfx:

pfxContainer:/home/astraadmin/_shared/Astralinux172se/a2fatest.a2fa2019.local
.pfx
      pwdContainer: +8.9)4~e:,P%G!w8791W
```

В рассмотренном примере присутствуют несколько уровней вложенности (цифра соответствует количеству пробелов перед ключом):

0 – для ключа, `publicServer`;

2 – для ключей `address`, `timeout` и `TLS`;

4 – для ключа `PFX`;

6 – для ключей `pfxContainer`, `pwdContainer`.

8.3 Обязательные ключи

В данном разделе перечислены параметры и группы параметров, которые должны обязательно присутствовать в файле конфигурации.

Типовые варианты настроек на различные СУБД можно посмотреть в разделе 8.5. Примеры файлов конфигураций.

8.3.1 Настройки подключения к СУБД

Данная группа параметров – `database` - позволяет настроить подключение к СУБД, где будет располагаться база данных, необходимая для работы сервиса Aladdin 2FA Service (см. в Таблица 3).

Таблица 3 – Описание параметров группы [database]

Ключевое слово	Уровень вложенности	Описание параметра
database	0	Обязательная корневая группа параметров
type	2	Тип СУБД Допустимые значения: `mssql` – для СУБД MS SQL; `postgresql` – для СУБД PostgreSQL
user	2	Имя аутентификатора с правами на доступ к базе данных Aladdin 2FA Service
password	2	Пароль в открытом виде. После старта сервиса Aladdin 2FA Service ключ `password` изменится на `encryptedPassword`, а значение будет зашифровано
encryptedPassword	2	Параметр добавится автоматически в ходе запуска сервиса Aladdin 2FA Service путём замены поля

		`password`. Ручной настройки не требуется. Содержит пароль от учетной записи СУБД в зашифрованном виде
host	2	Адрес сервера СУБД в формате `host:port`
instance	2	По умолчанию поле пустое. Используется только для СУБД MS SQL, представляет собой имя инстанса MS SQL
parameters	2	Дополнительные параметры подключения. Параметр должен содержать имя базы данных в формате `database=DBNAME`
options	4	Необязательная группа параметров. Используется для указания специфичных настроек для конкретной СУБД.
idleConns	6	Необязательный параметр. Используется только для MS SQL и предназначен для оптимизации нагрузки. Настраивает количество кэшируемых соединений. По умолчанию кэшируются два соединения

Пример группы параметров `database` перед запуском сервиса Aladdin 2FA Service:

```
database:
  type: postgresql
  user: postgre_admin
  password: o!g>*J3^_39f>X)oRY6m
  host: postgres.aladdin-rd.ru:5432
  instance: ""
  parameters: database=a2fa
```

Ниже в примере представлено, как будет выглядеть группа параметров `database` в файле настроек `config.yaml` после запуска сервиса Aladdin 2FA Service. Произойдёт шифрование пароля от учетной записи базы данных: параметр `password` будет заменен параметром `encryptedPassword`, значение будет зашифровано.

Пример изменения файла `config.yaml` после запуска сервиса Aladdin 2FA Service:

```
database:
  type: postgresql
  user: postgres
  encryptedPassword: OMWW9XVDj88xm2dEPqIzZQ==
  host: postgres.aladdin-rd.ru:5432
  instance: ""
  parameters: database=a2fa
```

Пример группы параметров `database` для настройки сервера Aladdin 2FA Service на СУБД PostgreSQL:

```
database:
  type: postgresql
```

```
  user: postgres
  password: o!g>*J3^_39f>X)oRY6m
  host: postgres.aladdin-rd.ru:5432
  instance: ""
  parameters: database=2faservice
```

Пример группы параметров `database` для настройки сервера Aladdin 2FA Service на СУБД MS SQL:

```
database:
  type: mssql
  user: sa
  password: o!g>*J3^_39f>X)oRY6m
  host: a2famssql.aladdin-rd.ru
  instance: "SQLEXPRESS"
  parameters: database=SQLA2FA
```

8.3.2 Настройки внутреннего сетевого интерфейса для JAS

В данной группе параметров настраивается адрес, порт и параметры защищённого соединения для обращения из сервера JAS. Так же блок содержит параметр `externalAddress`. Описание параметров см. в Таблица 4.

Таблица 4 - Описание параметров группы [privateServer]

Ключевое слово	Уровень вложенности	Описание параметра
privateServer	0	Обязательная корневая группа параметров
externalAddress	2	Внешний адрес (FQDN), по которому будут обращаться мобильные приложения к серверу Aladdin 2FA Service для регистрации аутентификаторов, обновления их статусов и так далее
address	2	Адрес, соответствующий сетевому интерфейсу, по которому будут инициироваться TCP-подключения из сервера JAS
timeout	2	Таймаут запросов. Рекомендуемое значение – 90 секунд
tls	2	Обязательная группа параметров. Предназначена для настройки TLS между серверами JAS и Aladdin 2FA Service
pfx	4	Группа настроек PFX-контейнера

pfxContainer	6	Путь к PFX-контейнеру
pwdContainer	6	Пароль от PFX-контейнера в открытом виде
encryptedPwdContainer	6	Параметр добавится автоматически в ходе запуска сервиса Aladdin 2FA Service путём замены параметра `pwdContainer`. Ручной настройки не требуется. Содержит пароль от контейнера PFX в зашифрованном виде

Пример настройки группы параметров `privateServer`:

```
privateServer:
  externalAddress: https://aladdin-rd.ru
  address: 192.168.81.106:9000
  timeout: 90
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/private_container.pfx
      pwdContainer: +8.9)4~e:,P%G!w8791W
```

Ниже в примере представлено, как будет выглядеть группа параметров privateServer в файле настроек `config.yaml` после запуска сервиса Aladdin 2FA Service. Произойдёт шифрование пароля PFX-контейнера `public_container.pfx`.

Пример изменения файла `config.yaml` после запуска сервиса Aladdin 2FA Service:

```
privateServer:
  externalAddress: https://aladdin-rd.ru
  address: 192.168.81.106:9000
  timeout: 90
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/private_container.pfx
      encryptedPwdContainer: C+NEJxsDgoLNAO3HaS44Cg==
```

8.3.3 Настройки внешнего сетевого интерфейса для мобильных приложений

В данной группе параметров настраивается адрес, порт и параметры защищённого соединения для обращения со стороны мобильных приложений. Подробное описание параметров см. в Таблица 5.

Таблица 5 - Описание параметров группы [publicServer]

Ключевое слово	**Уровень вложенности**	**Описание параметра**

publicServer	0	Обязательная корневая группа параметров
address	2	Адрес, соответствующий сетевому интерфейсу, по которому будут инициироваться TCP-подключения со стороны мобильных приложений
timeout	2	Таймаут запросов. Рекомендуемое значение – 60 секунд
tls	2	Группа параметров для настройки TLS между мобильными приложениями и Aladdin 2FA Service
pfx	4	Группа настроек PFX-контейнера
pfxContainer	6	Путь к PFX-контейнеру
pwdContainer	6	Пароль от PFX-контейнера в открытом виде
encryptedPwdContainer	6	Параметр добавится автоматически в ходе запуска сервиса Aladdin 2FA Service путём замены параметра `pwdContainer`. Ручной настройки не требуется. Содержит пароль от контейнера PFX в зашифрованном виде
wsPingInterval	2	Интервал пинга по websocket-соединению (в секундах). По-умолчанию 10 мин.

Пример настройки Внешнего интерфейса для мобильных приложений перед запуском сервиса:

```
publicServer:
  address: 192.168.81.106:9001
  timeout: 90
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/public_container.pfx
      pwdContainer: HTztb_GN8Vbf*Z0>YU7q
 wsPingInterval: 600
```

Ниже в примере представлено, как будет выглядеть группа параметров `publicServer` в файле настроек `config.yaml` после запуска сервиса Aladdin 2FA Service. Произойдёт шифрование пароля PFX-контейнера `public_container.pfx`.

Пример изменения файла `config.yaml` после запуска сервиса Aladdin 2FA Service:

```
publicServer:
```

```
address: 192.168.81.106:9001
timeout: 90
tls:
  pfx:
    pfxContainer: /home/administratora2fa/public_container.pfx
    encryptedPwdContainer: tK6gQBvVYuVbaZ9v9hnyxg==
```

8.4 Список необязательных ключей

8.4.1 Тип обработки PUSH-аутентификаций

Настройка обработки PUSH-аутентификаций используется для горизонтального масштабирования решения. По умолчанию сервер Aladdin 2FA Service настроен на обработку аутентификаций с помощью оперативной памяти. Описание параметра приведено в таблице (см. Таблица 6).

Таблица 6 - Описание параметра [pushDB]

Ключевое слово	Уровень вложенности	Описание параметра
pushDB	0	Способ обработки аутентификаций. Допустимые значения для параметра: • `mem` - рекомендовано для серверов с небольшим количеством аутентификаций. В этом случае сервер Aladdin 2FA Service для обработки аутентификаций будет использовать оперативную память; • `sql` - рекомендовано для серверов в составе NLB-кластера, от которых требуется высокая производительность и отказоустойчивость. Обработка аутентификаций на сервере Aladdin 2FA Service будет происходить с помощью базы данных SQL, что увеличит нагрузку на сервер базы данных, однако даст возможность горизонтального масштабирования Aladdin 2FA Service

Пример настройки параметра `pushDB` на использование оперативной памяти:

```
pushDB: mem
```

Пример параметра `pushDB` на использование SQL-сервера:

```
pushDB: sql
```

8.4.2 Настройки журналирования сервера Aladdin 2FA Service

Журналирование – запись информации о PUSH-аутентификациях в базе данных Aladdin 2FA Service. Параметр отключен по умолчанию для оптимизации производительности. Описание параметров приведено в таблице (см. Таблица 7).

Таблица 7 - Описание параметров журналирования

Ключевое слово	Уровень вложенности	Описание параметра
backupJournals	0	Группа параметров, предназначенных для настройки журналирования
enable	2	Включение/отключение журналирования событий
path	2	Расположение, куда будут сохраняться файлы резервного копирования. Необходимо создать вручную
scheduler	2	Группа параметров для настройки периода архивации
periodInDays	4	Периодичность архивации журналов (в днях)
hours	4	Время в часах, когда будет выполняться архивация
minutes	4	Время в минутах, когда будет выполняться архивация

Пример настройки бэкапирования журналов (группы параметров `backupJournals`) раз в неделю, в 03:00:

```
backupJournals:
  enable: true
  path: /home/administratora2fa/journals_backup_a2fa/
  scheduler:
    periodInDays: 7
    hours: 03
    minutes: 00
```

8.5 Примеры файлов конфигураций

8.5.1 Типовой вариант настройки Aladdin 2FA Service

Сервер настроен для следующей инфраструктуры (см. Таблица 8):

Таблица 8 - Описание параметров типового варианта настройки Aladdin 2FA Service

Часть инфраструктуры	Значение	Комментарий
Используемая СУБД	PostgreSQL	Сервис Aladdin 2FA Service будет настроен на использование СУБД PostgreSQL.

		Учетная запись для доступа к СУБД (должна быть предварительно создана) - `postgre_admin`; Пароль от учетной записи - `o!g>*J3^_39f>X)oRY6m`; Адрес сервера PostgreSQL - `postgres.aladdin-rd.ru:5432`; Имя базы данных (должна быть предварительно создана) - `database=a2fa`
Адрес для подключения из JAS	192.168.81.106:9000	Параметр `externalAddress` используется для связи мобильных приложений с сервером Aladdin 2FA Service либо с промежуточными узлами (см. п.2. Общие сведения). Адрес может быть указан явно или в виде `:PORT`, позволив, таким образом, слушать выбранный порт на всех доступных сетевых интерфейсах. Файл с PFX-контейнером расположен по пути: `/home/administratora2fa/private_container.pfx` Рекомендуемое значение параметра `timeout` – 90 секунд
Адрес внешнего интерфейса	192.168.81.106:9001	Адрес может быть указан явно или в виде `:PORT`, позволив, таким образом, слушать выбранный порт на всех доступных сетевых интерфейсах. Файл с PFX-контейнером расположен по пути: `/home/administratora2fa/public_container.pfx` Рекомендуемое значение параметра `timeout` – 60 секунд

Пример:

```
database:
  type: postgresql
  user: postgre_admin
  password: o!g>*J3^_39f>X)oRY6m
  host: postgres.aladdin-rd.ru:5432
  instance: ""
  parameters: database=a2fa
privateServer:
  externalAddress: https://aladdin-rd.ru
  address: 192.168.81.106:9000
```

```
  timeout: 90
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/private_container.pfx
      pwdContainer: +8.9)4~e:,P%G!w8791W
publicServer:
  address: 192.168.81.106:9001
  timeout: 60
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/public_container.pfx
      pwdContainer: HTztb_GN8Vbf*Z0>YU7q
```

Файл конфигурации изменится после запуска сервиса Aladdin 2FA Service.

Параметр `password` в группе параметров `database` будет заменён на `encryptedPassword`, который теперь содержит пароль от учетной записи `postgre_admin` в зашифрованном виде.

Параметр `pfxContainer` в группе параметров `privateServer->tls->pfx` будет заменён на `encryptedPwdContainer`, который теперь содержит пароль от PFX-контейнера в зашифрованном виде.

Параметр `pfxContainer` в группе параметров `publicServer ->tls->pfx` будет заменён на `encryptedPwdContainer`, который теперь содержит пароль от PFX-контейнера в зашифрованном виде.

Пример:

```
database:
  type: postgresql
  user: postgre_admin
  encryptedPassword: OMWW9XVDj88xm2dEPqIzZQ==
  host: postgres.aladdin-rd.ru:5432
  instance: ""
  parameters: database=a2fa
privateServer:
  externalAddress: https://aladdin-rd.ru
  address: 192.168.81.106:9000
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/private_container.pfx
      encryptedPwdContainer: C+NEJxsDgoLNAO3HaS44Cg==
  timeout: 90
```

```
publicServer:
  address: 192.168.81.106:9001
  timeout: 60
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/public_container.pfx
      encryptedPwdContainer: tK6gQBvVYuVbaZ9v9hnyxg==
```

8.5.2 Пример явного указания порта в FQDN

Пример настройки сервиса Aladdin 2FA Service с явным указанием порта после FQDN в поле `externalAddress` для использования в инфраструктуре без использования reverse-proxy:

```
database:
  type: postgresql
  user: postgre_admin
  password: o!g>*J3^_39f>X)oRY6m
  host: postgres.aladdin-rd.ru:5432
  instance: ""
  parameters: database=2faservice
privateServer:
  externalAddress: https://aladdin-rd.ru:9000
  address: :9000
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/private_container.pfx
      pwdContainer: +8.9)4~e:,P%G!w8791W
  timeout: 90
publicServer:
  address: :9001
  timeout: 60
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/public_container.pfx
      pwdContainer: HTztb_GN8Vbf*Z0>YU7q
```

8.5.3 Пример настройки сервиса на СУБД PostgreSQL

```
database:
```

```
  type: postgresql
  user: postgre_admin
  password: o!g>*J3^_39f>X)oRY6m
  host: postgres.aladdin-rd.ru:5432
  instance: ""
  parameters: database=2faservice
privateServer:
  externalAddress: https://aladdin-rd.ru:9000
  address: :9000
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/private_container.pfx
      pwdContainer: +8.9)4~e:,P%G!w8791W
  timeout: 90
publicServer:
  address: :9001
  timeout: 60
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/public_container.pfx
      pwdContainer: HTztb_GN8Vbf*Z0>YU7q
```

8.5.4 Пример настройки сервиса Aladdin 2FA Service на СУБД MS SQL

Сервер MS SQL 2017 редакции Express с именем инстанса `SQLEXPRESS`.

```
database:
  type: mssql
  user: sa
  password: o!g>*J3^_39f>X)oRY6m
  host: a2famssql.aladdin-rd.ru
  instance: "SQLEXPRESS"
  parameters: database=SQLA2FA
privateServer:
  externalAddress: https://aladdin-rd.ru
  address: 192.168.81.112:9000
  tls:
```

```
    pfx:
      pfxContainer: /home/administratora2fa/private_container.pfx
      pwdContainer: +8.9)4~e:,P%G!w8791W
  timeout: 90
publicServer:
  address: 192.168.81.112:9001
  timeout: 60
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/public_container.pfx
      pwdContainer: HTztb_GN8Vbf*Z0>YU7q
```

8.5.5 Настройка на группу доступности MS SQL Always On

Пример настройки сервиса Aladdin 2FA Service на группу доступности SQL Always On с использованием DNS-имени `2fasqllistener.aladdin-rd.ru` с явным указанием порта:

```
database:
  type: mssql
  user: sa
  password: o!g>*J3^_39f>X)oRY6m
  host: 2fasqllistener.aladdin-rd.ru:1433
  instance:
  parameters: database=SQLA2FA
privateServer:
  externalAddress: https://aladdin-rd.ru
  address: 192.168.81.112:9000
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/private_container.pfx
      pwdContainer: +8.9)4~e:,P%G!w8791W
  timeout: 90
publicServer:
  address: 192.168.81.112:9001
  timeout: 60
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/public_container.pfx
```

```
      pwdContainer: HTztb_GN8Vbf*Z0>YU7q
```

8.5.6 Пример настройки Aladdin 2FA Service без TLS на внешнем интерфейсе

Иногда в инфраструктуре между пользователями, которые используют мобильное приложение Aladdin 2FA, и сервисом Aladdin 2FA Service используется промежуточный узел. Этот узел может быть представлен, например, в виде Nginx, который настроен как Reverse Proxy (см. Рисунок 23).

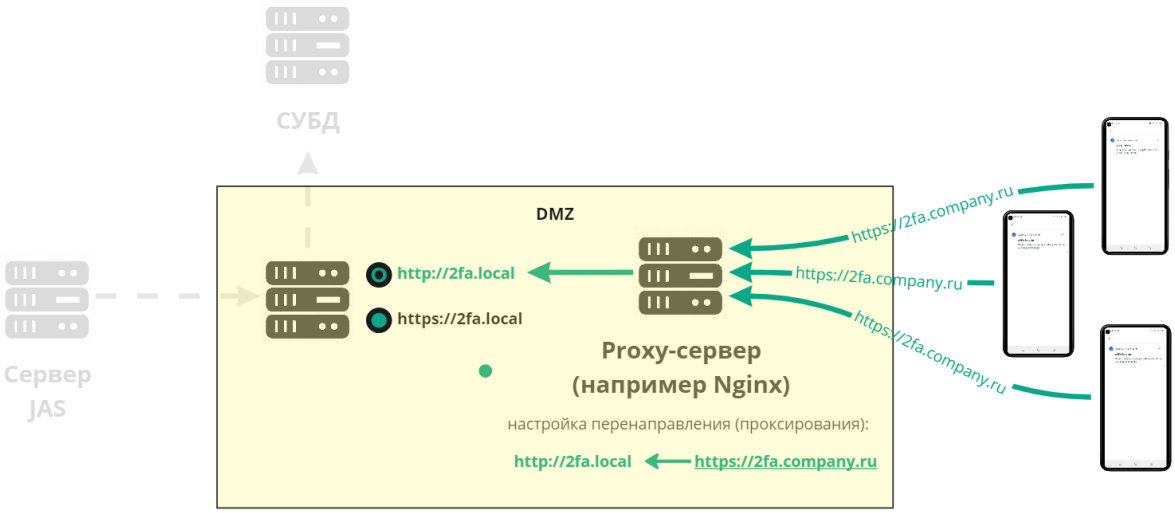


Рисунок 23 – Схематичное представление настройки DMZ

В случае полноценной защиты и тщательного мониторинга зоны DMZ возможен вариант, когда настройкой TLS между промежуточным узлом и сервисом Aladdin 2FA можно пренебречь.

Пример настройки Aladdin 2FA Service:

```
database:
  type: postgresql
  user: postgres
  password: o!g>*J3^_39f>X)oRY6m
  host: postgres.aladdin-rd.ru:5432
  instance: ""
  parameters: database=2faservice
privateServer:
  externalAddress: https://aladdin-rd.ru:9000
  address: :9000
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/private_container.pfx
      pwdContainer: +8.9)4~e:,P%G!w8791W
  timeout: 90
```

```
publicServer:

  address: :9001

  timeout: 60
```

Для корректного запуска Aladdin 2FA Service необходимо предварительно сконфигурировать сервер СУБД, создать базу данных или перенести существующую с MS SQL.

8.6 Создание базы данных

Для создания базы данных необходимо выполнить следующие шаги:

1. Создать текстовый файл, скопировав в него скрипт из пункта «Сбор логов», сохранить файл со скриптом (в примере имя файла `a2fa_create_postgres_db.sh`), положить его в папку с PostgreSQL;
2. Перед тем, как выполнить скрипт в терминале, нужно установить для него флаг исполняемости с помощью команды:

```
# chmod ugo+x a2fa_create_postgres_db.sh
```

   После этого название файла в терминале будет выделено зеленым цветом, а не белым (как было ранее), что говорит о том, что для него был установлен флаг исполняемости;

3. Указать полный путь до расположения файла со скриптом:

```
# cd /var/lib/pgsql/13/
```

4. Запустить скрипт:

```
#bash ./SCRIPT.SH
```

5. Далее, нажимая клавишу <Enter> на клавиатуре, ввести в терминале название создаваемой базы данных, имя пользователя, пароль (см. Рисунок 24).

```
[root@REDOSPOSTGRESQL732 13]# ./bd_test.sh
Введите имя администратора базы данных (или нажмите Enter для использования имени по умолчанию 'postgres'):

Введите название базы данных:
a2fa_test
Введите имя пользователя для базы данных a2fa_test:
postgres
Введите пароль для пользователя postgres:
CREATE DATABASE
ОШИБКА:  роль "postgres" уже существует
GRANT
База данных a2fa_test и пользователь postgres созданы, права на базу данных назначены.
```

Рисунок 24 – пример запуска в терминале файла со скриптом

В терминале будет отображено информационное сообщение о создании базы данных, пользователя и правах (см. Рисунок 24).

8.7 Миграция базы данных с MS SQL в PostgreSQL с помощью утилиты aladdin-2fa-magration-tool

> *Перед процедурой переноса базы данных из одной СУБД в другую, рекомендуется создать резервную копию баз данных MS SQL. Это необходимо в случае, если что-то пойдет не так в процессе миграции*

Миграции базы данных с MS SQL на PostgreSQL осуществляется с помощью утилиты `aladdin-2fa-magration-tool`, которая позволяют перенести данные из одной системы в другую без потери информации и сохранения структуры данных.

Для переноса базы данных из С MS SQL Server в PostgreSQL с помощью утилиты `aladdin-2fa-magration-tool` необходимо выполнить следующие шаги:

1. Подготовить PostgreSQL. Убедиться, что на машине с ОС Linux установлен и запущен PostgreSQL, есть необходимые права доступа;
2. Создать новую базу данных в PostgreSQL с помощью команды (подробнее процесс создания базы данных приведен в п. 8.1):

```
create database
```

3. Открыть PowerSheel и перейти в папку с утилитой `aladdin-2fa-magration-tool.exe`:

```
PS C:\_shared>
```

4. Далее указать название утилиты, имя БД, которую необходимо скопировать и местоположение куда ее необходимо перенести. Пример запуска утилиты:

```
PS C:\_shared> .\aladdin-2fa-migration-tool.exe -
mssql="sqlserver://sa:SA!!dbQq1234567890,,..@192.168.81.4:49151?database=2FAS
erviceDB&connection+timeout=30" -
postgres="postgres://postgres:admin1q2W@192.168.81.108:5432?database=2fa"
```

5. Дождаться завершения процесса миграции. После успешного завершения утилита `aladdin-2fa-magration-tool` создаст таблицы в PostgreSQL, заполненные данными из базы данных MS SQL Server.

> *Хоть описанная процедура и позволяет успешно мигрировать данные с MS SQL на PostgreSQ, однако не гарантирует полностью идентичного соответствие данных между СУБД (из-за различий между ними), поэтому после переноса рекомендуется тщательно протестировать новую базу данных*

8.8 Настройка подключения сервера Aladdin 2FA Service к PostgreSQL с использованием TLS

Для настройки подключения Aladdin 2FA Service к PostgreSQL с использованием TLS требуется выполнить действия, приведенные ниже.

Со стороны PostgreSQL:

1. Выпустить сертификат и ключ для PostgreSQL, подписанные вашим УЦ. Это необходимо, чтобы обеспечить безопасность и защиту данных, хранящихся в базе данных PostgreSQL;
2. Поместить выпущенный сертификат с ключом и корневым сертификатом УЦ в директорию tls:

```
 /etc/postgresql/14/main/tls/
```

3. Обязательно изменить права `0600` для ключа. Это необходимо, чтобы обеспечить безопасность и защиту ключа, предотвратив доступ к нему для неавторизованных пользователей.

```
-rw------- 1 postgres postgres 1704 Apr 12 22:44 server.key
```

4. Открыть файл конфигурации `postgresql.conf` и разрешить подключения, изменив параметры `listen_addresses = '*'` и `TLS`:

```
#--SSL--

ssl = on

ssl_ca_file = '/etc/postgresql/14/main/certs/root.crt' # Сертификат УЦ

ssl_cert_file = '/etc/postgresql/14/main/certs/server.crt' # Сертификат
сервера

ssl_key_file = '/etc/postgresql/14/main/certs/server.key' # Ключ
```

Далее необходимо добавить правила для подключения с TLS в файл `pg_hba.conf`. Для этого необходимо выполнить следующие:

1. Открыть файл `pg_hba.conf` с помощью текстового редактора или командной строки;
2. Найти раздел, который соответствует IP-адресу и пользователю, который будет использоваться для подключения;
3. Добавить новое правило в этот раздел, указав параметры для TLS-подключения:

```
# TYPE      DATABASE       USER        ADDRESS          METHOD

hostssl     all            all         0.0.0.0/0       md5 clientcert=verify-ca
```

4. Сохранить изменения в файле `pg_hba.conf` и перезагрузить сервер PostgreSQL, чтобы изменения вступили в силу.

Со стороны Aladdin 2FA Service:

1. Выпустить сертификат и ключ для Aladdin 2FA Service, подписанные вашим УЦ;
2. Поместить выпущенные сертификат, ключ и корневой сертификат УЦ в директорию `certs`:

```
/etc/aladdin-2fa-service/certs/
```

3. Указать в файле конфигурации `config.yaml` Aladdin 2FA Service:

```
database:

  type: postgresql # Тип СУБД

  user: user # Имя пользователя БД

  password: password # Пароль пользователя БД

  host: localhost:5432 # Адрес СУБД PostgreSQL

  parameters: database=a2fa&sslmode=verify-ca&sslcert=/etc/aladdin-2fa-
service/certs/client.crt&sslkey=/etc/aladdin-2fa-
service/certs/client.key&sslrootcert=/etc/aladdin-2fa-service/certs/root.crt
# Параметры с указанием имени БД, тип проверки сертификата, путей к
сертификатам и ключу
```

8.9 Повторная настройка

Повторная настройка применяется для изменения или обновления параметров, которые уже неактуальны: добавить или обновить лицензию, обновить устаревшие сертификаты, перенастроить базу данных при смене СУБД.

В зависимости от выбранного сценария необходимо внести изменение в конфигурационный файл в соответствующей секции:

1. **Перенастроить базу данных при смене СУБД**. При подключении к другой БД в секции `database` необходимо заново ввести значение в поле `encryptedPassword`: значение `encryptedPassword` заменить на `password` и задать пароль (подробнее описание параметров секции `database` см. п. 8.3.1);

Такая ситуация может возникнуть, если администратор сделал резервную копию базы данных и решил заменить сервер СУБД. Рекомендуется не злоупотреблять данной процедурой, так как возникнут трудности в работе JMS: записей о токенах и пользователях в новой базе не будет

2. **Замена сертификатов или смена места их расположения**. В секциях `privateServer` (внутренней сертификат) и `publicServer` (внешний сертификат) для переменных `certificate` и `privateKey` указать пути до сертификата (файл `*.pem`) и файла закрытого ключа (`*key.pem`). При использовании PFX-контейнера (`*.pfx)`, изменить пути директорий необходимо для переменных `pfxContainer` и `pwdContainer`.:

```
database:
  type: postgresql
  user: postgres
  password: o!g>*J3^_39f>X)oRY6m
  host: postgres.aladdin-rd.ru:5432
  instance: ""
  parameters: database=2faservice
privateServer:
  externalAddress: https://aladdin-rd.ru:9000
  address: :9000
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/private_container.pfx
      pwdContainer: +8.9)4~e:,P%G!w8791W
  timeout: 90
publicServer:
  address: :9001
  timeout: 60
  tls:
    pfx:
      pfxContainer: /home/administratora2fa/public_container.pfx
      pwdContainer: HTztb_GN8Vbf*Z0>YU7q
```

3. **Добавить или обновить лицензию**. Для успешного обновления/добавления лицензии необходимо разместить файла с расширением `*.lic` в каталог `/etc/aladdin-2fa-service/license`.

9. Настройки сервиса для использования Telegram, для передачи второго фактора

9.1 Введение

Сервис Aladdin 2FA поддерживает двухфакторную аутентификацию через сервер Telegram.

Сервер Telegram использует технологию webhook для работы с сервисом Aladdin 2FA.

Webhook – это механизм, который позволяет внешним сервисам принимать уведомления от Telegram и отправлять запросы к нему. Когда пользователь отправляет сообщение в канал или группу, подключенную к определенному вебхуку, сервер Telegram отправляет запрос на этот вебхук. Таким образом можно получать сообщения из Telegram и обрабатывать их в реальном времени.

Техническое окружение:

- установленный дистрибутив A2FA Service на ОС Linux (Astra, redOS);
- установленная и настроенная БД (PostgreSQL или MSSQL);
- установленное приложение Telegram на мобильном устройстве.

Для установки и настройки необходимы:

- включенная в лицензию опция телеграм-бота;
- сертификаты для Private Server и сервера телеграм-бота.

Для подключения к Telegram нужны сертификаты, выпущенные либо доверенным УЦ (например - let's encrypt), либо самоподписанным УЦ (например, доменным - MS CA).

Сетевое окружение

Для корректной работы необходимо:

1. Использовать протокол IPv4;
2. Разрешить входящий трафик на сетевом оборудовании из подсетей 149.154.160.0/20 и 91.108.4.0/22 на один из портов (443, 80, 88, 8443), который будет использоваться в сервисе Aladdin 2FA;
3. Зарегистрировать доменное имя для Webhook;
4. Использовать доменный или самоподписанный сертификат;
5. В сертификате должно быть указано зарегистрированное доменное имя в параметре «Common Name»;
6. В сертификате должны быть указаны промежуточные сертификаты для проверки цепочки сертификатов.

9.2 Конфигурация Telegram-бота для сервера A2FA

9.2.1 Регистрация бота в Telegram

Для регистрации бота в Telegram необходимо выполнить следующие действия:

1. Открыть приложение Telegram и найти аккаунт «@BotFather»;
2. Запустить диалог с BotFather, нажав на кнопку «Start»;
3. Ввести команду `/newbot`, чтобы начать процесс создания нового бота;
4. Придумать уникальное имя для бота, которое должно заканчиваться на «_bot»;

5. Выбрать «username» для бота, который будет использоваться как ссылка для доступа к нему. Необходимо убедиться, что оно уникальное и нажать кнопку «Отправить»;
6. После этого в диалоге с BotFather отображается «токен» созданного бота;

> *«Токен» это набор символов, чисел и специальных символов*

7. Добавить описание бота, указав его назначение и основные функции;
8. Добавить логотип для бота;
9. Выбрать тип бота «приватный». Для изменения типа необходимо:
   - ввести команду `/mybots`;
   - выбрать созданного бота;
   - нажать на кнопку «Bot Settings» -> «Group Privacy» -> «Turn on». Текст параметра должен измениться на «Privacy mod is [enadled](https://core.telegram.org/bots/features#privacy-mode) for …»;

> *Данная настройка позволяет обращаться к боту только по ссылке*

10. Запретить добавление бота в группы. Для этого необходимо:
    - ввести команду `/mybots`;
    - выбрать созданного бота;
    - нажать кнопку «Allow Groups?». Текст параметра должен измениться на «Groups are currently **disabled** for …»;
11. Сохранить «токен» бота, который будет использоваться для его аутентификации при взаимодействии сервиса Aladdin 2FA с API Telegram.

> ***Важно!*** *Полученный токен нужно хранить в секрете. Не передавать его 3-м лицам и не хранить в открытом виде*

9.2.2 Настройка параметров сервера A2FA

Настройка параметров сервиса Aladdin 2FA для взаимодействия с серверами Telegram.

Для включения телеграм-бота Aladdin 2FA в конфигурационном файле необходимо добавить следующий блок параметров:

```
tgBotServer:

  externalAddress: https://somednsname.ru #Указывается зарегистрированное доменное имя для Webhook.

address: 192.168.1.10:8443 # Указывается адрес сетевого интерфейса, на котором установлен сервис Aladdin 2FA и порт (443, 80, 88, или 8443) из перечисленных. Если сервис находится за пограничным устройством (сетевым экраном, сетевым балансировщиком и т.д.), допускается указание любого порта, но на пограничном устройстве обязательно должен быть открыт любой из перечисленных портов т.к. сервера Telegram работают только с этими номерами портов.

Tls # Указывается сертификат или контейнер сертификата, который выпускался для зарегистрированного доменного имени. Если сервис Aladdin 2FA находится за пограничным сетевым устройством и это устройство поддерживает создание TLS (TLS v1.2), данный пункт можно не включать, а вся настройка TLS проводится на пограничном устройстве.

pfx:

  pfxContainer: tls/tgBot/localhost_8443.pfx

  pwdContainer: 1234567890

token: 6279835370:TAGMd6JuC8Ts5FAJcB55vNifIbJdxQMEdd4 # Указывается токен, который сгенерировал **BotFather**. После запуска сервиса данное поле будет заменено полем encryptedToken с зашифрованным токеном.
```

```
timeout: 60

debug: true # Отладочный режим Telegram-бота. При включенном режиме отладки(--verbose) будет записываться отладочная информация бота
```

Для подключения без использования NGINX указать `externalAddress` в разделе `tgBotServer` с портами (test.ru:8443). Порт допустимо указывать в одном из следующих форматах:

- `externalAddress: https://test.ru:8443`;
- `address: 1.1.1.1:8443`.

> *В поле `address` допустимо указывать параметр в виде `address: :8443`*

9.3 Варианты использования TLS для подключения к телеграм-боту

Для подключения к телеграм-боту с использованием TLS доступны следующие варианты настройки:

- с использованием реверс-прокси (см. п. 9.3.1);
- без использования реверс-прокси (см. п. 9.3.2);

9.3.1 Настройка телеграм-бота с реверс-прокси (на примере NGINX)

В случае подключения через NGINX, для защиты соединения будет использован tls сертификат, используемый NGINX. В этом случае, в конфигурационном файле в разделе `tgBotServer` указывать сертификат не нужно.

На рисунке (см. Рисунок 25) приведен пример подключения с имеющимися локациями, необходимо указать не занятые порты и открыть их в системе с помощью команды:

```
sudo ufw allow 6786/tcp
```

Локация (location) в Nginx — структура в конфигурационном файле, определяющая, какие правила будут применяться к URL и запросам.

```
tgBotServer:
  externalAddress: https://jcvt-test.a-rd.ru/tg_bot
  address: :9002
  encryptedToken: e494ijuUNqOZSE9pd20RLVqDUknSR6RA/7aU7vZwTBBHP2Sn48Q5MBTn9eAPvBWn
  timeout: 60
  debug: true
```

Рисунок 25 – Пример конфигурационного файла с использованием локаций NGINX

9.3.2 Настройка телеграм-бота без реверс-прокси (без NGINX)

В случае подключения без реверс-прокси, для защиты соединения будет использован tls сертификат, указанный в разделе `tgBotServer`. Имеется два способа указания сертификатов:

- с помощью параметра cert: в этом случае, указать путь до сертификата (certificate) и ключа (privateKey) (см. Рисунок 26);
- с помощью pfx: в этом случае, указать путь до pfx-контейнера (pfxContainer) и пароль от него (pwdContainer) (см. Рисунок 27).

> *Доменное имя должно быть зарегистрированным*

Для корректной работы телеграм-бота с использованием самоподписанных сертификатов или выпущенных доверенным УЦ необходимо в конфигурационный файл добавить параметр `externalRootCertificate` с указанием пути до сертификата, который является корневым для сертификата, указанного в параметрах TLS блока tgBotServer.

```
tgBotServer:
  externalAddress: https://a2fa-test.a-rd.ru:8443
  address: :8443
  encryptedToken: e494ijuUNqOZSE9pd20RLVqDUknSR6RA/7aU7vZwTBBHP2Sn48Q5MBTn9eAPvBWn
  externalRootCertificate: /home/administratorjcvt/certs/a2fa.pem
  tls:
   cert:
     certificate: /home/administratorjcvt/certs/a2fa.pem
     privateKey: /home/administratorjcvt/certs/a2fakey.pem
  timeout: 60
  debug: true
```

Рисунок 26 – Пример использования сертификата TLS

```
tgBotServer:
  externalAddress: https://a2fa-test.a-rd.ru:8443
  address: :8443
  encryptedToken: e494ijuUNqOZSE9pd20RLVqDUknSR6RA/7aU7vZwTBBHP2Sn48Q5MBTn9eAPvBWn
  externalRootCertificate: /home/administratorjcvt/certs/a2fa.pem
  tls:
   pfx:
     pfxContainer: /home/administratorjcvt/certs/a2fa.pfx
     encryptedPwdContainer: tK6gQBvVYuVbaZ9v9hnyxg==
  timeout: 60
  debug: true
```

Рисунок 27 – Пример использования pfx-контейнера

9.3.3 Создание самоподписанного сертификата

Для создания самоподписанного сертификата необходимо выполнить следующие действия:

1. Создать закрытый ключ:

```
openssl genrsa -out tgkey.key 2048
```

2. Создать запрос на подпись:

```
openssl req -key tgkey.key -new -out tgcsr.csr
```

3. После выполнения запроса необходимо заполнить поле Common Name с указанием FQDN:

```
Common Name (e.g. server FQDN or YOUR name) []:a2fadebug.aladdin-rd.ru
```

4. Подписать сертификат самим собой:

```
openssl x509 -signkey tgkey.key -in tgcsr.csr -req -days 365 -out tgcert.crt
```

5. При успешном создании выводятся следующие параметры:

```
user@docker:~/test$ sudo openssl x509 -signkey tgkey.key -in tgcsr.csr -req -days 365 -out tgcert.crt
Certificate request self-signature ok
subject=C = RU, ST = Moscow, L = Moscow, O = Aladdin, CN = a2fadebug.aladdin-rd.ru
```

Далее необходимо добавить в конфигурационный файл в раздел настройки телеграм-бота следующие параметры:

```
tgBotServer:
  externalAddress: https://a2fadebug.aladdin-rd.ru:8443
  address: :8443
  token: klkmhhmkhmfgk;mhtk;rklty5i6j54khr065
  externalRootCertificate: /home/test/tgcert.crt
  tls:
    cert:
      certificate: /home/test/tgcert.crt
      privateKey: /home/test/tgkey.key
  timeout: 60
  debug: true
```

9.3.4 Вариант с использованием PFX

Самоподписанный сертификат для телеграмм-бота также возможно использовать в pfx-контейнере, для этого необходимо:

1. Сгенерировать контейнер с указанием его имени и что в него входит:

```
openssl pkcs12 -export -out tgpfx.pfx -inkey tgkey.key -in tgcert.crt
```

2. Задать и подтвердить пароль для контейнера;
3. При успешном задании пароля pfx-контейнер будет создан.

Для того, чтобы пользоваться pfx-контейнером необходимо в конфигурационном файле в разделе настройки телеграм-бота указать корневой сертификат:

```
tgBotServer:
  externalAddress: https://a2fadebug.aladdin-rd.ru:8443
  address: :8443
  token: klkmhhmkhmfgk;mhtk;rklty5i6j54khr065
  externalRootCertificate: /home/test/tgcert.crt
  tls:
    pfx:
      pfxContainer: /home/test/tgpfx.pfx
      pwdContainer: admin1q2W
  timeout: 60
  debug: true
```

9.4 Диагностика

Корректность настроек можно проверить способами, описанными ниже.

9.4.1 Проверка соединения

При возникновении ряда ошибок необходимо выполнить диагностику телеграм-бота для этого необходимо:

1. Перейти на тестовую страницу с помощью ссылки, которая указана в разделе `tgBotServer` в параметре `externalAddress`, к которой добавляется /test;
2. В результате могут быть отражены следующие ошибки:
   - "Не удается получить доступ к сайту" – при отсутствии соединения, неправильных настройках сети и т.д.;
   - "200, OK" – соединение установлено корректно;
   - "404, not found" – неверно указана ссылка на сервис телеграм-бота;

- "Предупреждение: Вероятная угроза безопасности" – при использовании самоподписанного или недоверенного сертификата правильно ли указан путь до этого сертификата. При подтверждении перехода должен вернуться ответ "OK";
- "525 – SSL Handshake Failed" – при использовании невалидного tls-сертификата.

Для корректной работы телеграм-бота с использованием самоподписанных сертификатов или выпущенных доверенным УЦ необходимо в конфигурационный файл добавить параметр `externalRootCertificate` с указанием пути до сертификата, который является корневым для сертификата, указанного в параметрах TLS блока tgBotServer.

9.4.2 Проверка настройки TLS

Для того что бы проверить что настройки TLS Telegram-бота необходимо выполнить следующие шаги:

1. Включить в настройках «tgBotServer» параметр «debug: true»;
2. Запустить сервис с параметром –verbose в консоли и убедиться, что сервис успешно запущен;
3. В консоли найти строчку:

```
Endpoint: setWebhook, params: map[allowed_updates:null
url:https://somednsname.ru/1c48863afe7f7c6ea22724f23c5f5a0fed2aa86ec3459189af
962a1faca45b63]`
```

где `https://somednsname.ru/1c48863afe7f7c6ea22724f23c5f5a0fed2aa86ec3459189af962a1faca45b6 адрес Webhook.`

4. Скопировать адрес Webhook
5. Открыть бот разработчиков Telegram, расположенный по адресу: https://t.me/CanOfWormsBot;
6. Ввести команду `/start` и следовать инструкциям (бот проверит ваш сертифкат).

Подробное описание запуска бота в мобильном приложении Telegram приведено в документе «Aladdin 2FA. Руководство пользователя» [1], которое доступно для загрузки на официальном сайте компания «Аладдин Р. Д.».

10. Информация по мобильному приложению «Aladdin 2FA», которую нужно знать при администрировании решения Aladdin 2FA

> *В мобильном приложении «Aladdin 2FA» реализован запрет запуска на эмуляторе Android*

Пример интерфейса настроек приведен в документе «Aladdin 2FA. Руководство пользователя» [1], которое доступно для загрузки на официальном сайте компания «Аладдин Р. Д.».

Приложение А. Пример конфигурационного файла config.yaml

```
# Блок конфигурации базы данных. Более подробно каждое поле описано в Руководстве администратора Aladdin 2FA Service.

# Тип СУБД, Учетная запись, пароль, адрес СУБД, порт и параметры являются демонстрационными!

# Перед использованием сервера настройте одну из поддерживаемых A2FA СУБД, создайте в ней базу данных для A2FA.

# Создайте пользователя и выдайте ему полные права для созданной базы.

database:

 # Тип БД

 type: mssql

 # Имя учетной записи для доступа к СУБД

 user: user

 # Пароль от учетной записи

 password: test     //после первого успешного запуска сервиса пароль отображается в зашифрованном виде в поле encryptedPassword

 # Адрес СУБД, по которому будет подключаться сервис

 host: JASJMS:1433

 # Экземпляр базы данных

 instance: SQLEXPRESS

 #Параметр - укажите базу данных, предварительно созданную из

 parameters: database=smoke131&connection+timeout=30&TrustServerCertificate=true

# Конфигурация private-сервера

privateServer:

 # Адрес сервера, который будет отправляться пользователям в e-mail. Это может быть как адрес сервера, так и адрес прокси-сервера.

 externalAddress: https://jcvt-test.a-rd.ru/jmstest

 # Адрес запуска сервера. Если не указан - сервер не будет запускаться

 # address: "localhost:6787" # полный адрес

 address: 192.168.80.54:9000

 # Отключение опции «Статистика»

 turnOffStatisticsPage: true

 # Настройки TLS

 tls:

  # TLS с использованием PFX контейнера
```

```
  pfx:

    # Путь до PFX контейнера

    pfxContainer: C:\_shared\A2FAFOONEW.pfx

    # Пароль от PFX контейнера в зашифрованном виде

    encryptedPwdContainer:
AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAAedf6W4MF0k60sLs34EtepQAAAAACAAAAAAAQZgAAAAEAACAAAACQtJ
JodFnBJ8PNuZMTC/8VXOcdXXVWFq1MB68BqfXF4AAAAAAOgAAAAIAACAAAAAGoXA4Yv7bGDVWL9j4ZydXbEJG
xHSch3abCED5Js+J9hAAAADBY2zuCl2nXQk4+q7y0YeKQAAAANBjNceaHn7unGqjImSeCsoEqV2lU3OnPQAx3z/HicR
GeBgoFLbK0uW78H74HuVHMjOHjrXsv1awisbWPj68UzI=

 # Таймаут запросов

 timeout: 90

# Конфигурация private-сервера

publicServer:

 address: 192.168.80.54:9001

 # Таймаут запросов

 timeout: 60

 # Настройки TLS

 tls:

   # TLS с использованием PFX контейнера

   pfx:

     # Путь до PFX контейнера

     pfxContainer: C:\_shared\A2FAFOONEW.pfx

     # Пароль от PFX контейнера в зашифрованном виде

     encryptedPwdContainer:
AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAAedf6W4MF0k60sLs34EtepQAAAAACAAAAAAAQZgAAAAEAACAAAABxd
mzrxz8AZku1lHSX0vJpYa7Mxh7qzER4EaWH7K99iQAAAAAOgAAAAIAACAAAAAK2UoZwsj36KQwgBIQ5kx8zm1YQ
AqnLttVwgxNKfgrPxAAAAB3qnuNFSMUMEt3KJuQLDI5QAAAAA3EHYL1VukkOQrsBCiFamykfY1jL03YNYnodaw1Nz0
UmkZko51Q4WNuHYq7KNHgvbiMlB0S+MpOC2RAJVtuJ+o=



 # Интервал пинга по websocket-соединению (в секундах). По-умолчанию 10 мин.

 # wsPingInterval: 600



 tgBotServer:

   externalAddress: https://somednsname.ru #Указывается зарегистрированное доменное имя для Webhook.

 address: 192.168.1.10:8443 # Указывается адрес сетевого интерфейса, на котором установлен сервис
Aladdin 2FA и порт (443, 80, 88, или 8443) из перечисленных. Если сервис находится за пограничным
устройством (сетевым экраном, сетевым балансировщиком и т.д.), допускается указание любого порта, но
на пограничном устройстве обязательно должен быть открыт любой из перечисленных портов т.к. сервера
Telegram работают только с этими номерами портов.
```

Tls # Указывается сертификат или контейнер сертификата, который выпускался для зарегистрированного доменного имени. Если сервис Aladdin 2FA находится за пограничным сетевым устройством и это устройство поддерживает создание TLS (TLS v1.2), данный пункт можно не включать, а вся настройка TLS проводится на пограничном устройстве.

pfx:

pfxContainer: tls/tgBot/localhost_8443.pfx

pwdContainer: 1234567890

token: 6279835370:TAGMd6JuC8Ts5FAJcB55vNiflbJdxQMEdd4 # Указывается токен, который сгенерировал **BotFather**. после запуска сервиса данное поле будет заменено полем encryptedToken с зашифрованным токеном.

timeout: 60

debug: true # Отладочный режим Telegram-бота. При включенном режиме отладки(--verbose) будет записываться отладочная информация бота

Приложение Б. Скрипт для создания базы данных для PostgreSQL

Файл `a2fa_create_postgres_db.sh`:

```
#!/bin/bash



# Проверка прав пользователя

if [ "$EUID" -ne 0 ]

  then echo "Этот скрипт должен быть запущен от пользователя root"

  exit

fi



# Запрос имени администратора базы данных

echo "Введите имя администратора базы данных (или нажмите Enter для
использования имени по умолчанию 'postgres'):"

read -r DB_ADMIN

DB_ADMIN=${DB_ADMIN:-postgres}



# Получение полного пути до скрипта

SCRIPT=$(readlink -f "$0")

SCRIPT_PATH=$(dirname "$SCRIPT")



# Проверка наличия пользователя postgres и его прав

if ! sudo -u $DB_ADMIN psql -c "SELECT 1 FROM pg_roles WHERE
rolname='postgres' AND rolsuper AND rolcreaterole AND rolcreatedb" | grep -q
1

then

    echo "Пользователь \"$DB_ADMIN\" не существует или у него нет необходимых
прав"

    exit

fi



# Проверка прав доступа для пользователя postgres

if ! sudo -u $DB_ADMIN test -r "$SCRIPT_PATH"

then

    echo "Пользователь \"$DB_ADMIN\" не имеет прав на чтение каталога
$SCRIPT_PATH"
```

```
    echo "Переместите скрипт и перейдите в домашний каталог пользователя
\"$DB_ADMIN\" или в каталог /tmp"

    exit

fi



# Запрос названия базы данных

echo "Введите название базы данных:"

read DB_NAME



# Запрос имени пользователя

echo "Введите имя пользователя для базы данных $DB_NAME:"

read DB_USER



# Запрос пароля пользователя

echo "Введите пароль для пользователя $DB_USER:"

read -s DB_PASS







# Проверка установки PostgreSQL

if ! command -v psql &> /dev/null

then

    echo "PostgreSQL не установлен на этой системе"

    exit

fi







# Создание базы данных

sudo -u $DB_ADMIN psql -c "CREATE DATABASE \"$DB_NAME\";"



# Создание пользователя и назначение пароля

sudo -u $DB_ADMIN psql -c "CREATE USER \"$DB_USER\" WITH PASSWORD
'$DB_PASS';"



# Назначение прав на базу данных
```

```
sudo -u $DB_ADMIN psql -c "GRANT ALL PRIVILEGES ON DATABASE \"$DB_NAME\" TO
\"$DB_USER\";"



echo "База данных $DB_NAME и пользователь $DB_USER созданы, права на базу
данных назначены."
```

Приложение В. Пример настройки Aladdin 2FA Service в составе отказоустойчивого кластера pacemaker на ОС RedOS

Предварительные условия:

- имеются минимум две ВМ с установленной ОС RedOS (узел 1 и узел 2);
- имеется ВМ с установленным и настроенным сервисом NGINX;
- имеется ВМ с установленной и настроенной БД (PostgreSQL или MSSQL);
- на каждом узле установлен дистрибутив A2FA Service (оба узла настраиваются одинаково и имеют подключение к одной БД.

Пример конфигурационного файла с узла 1 приведен ниже (см. Рисунок 28).

```
database:
  type: postgresql
  user: jcvt
  encryptedPassword: tK6gQBvVYuVbaZ9v9hnyxg==
  host: 192.168.100.125:5432
  instance: ""
  parameters: database=pacemaker
privateServer:
  externalAddress: https://jcvt-test.a-rd.ru/cluster9551_mob
  address: :9550
  tls:
    cert:
      certificate: /home/user/_shared/cert/fullchain.pem
      privateKey: /home/user/_shared/cert/privkey.pem
  timeout: 150
publicServer:
  address: :9551
  timeout: 60
  tls:
    cert:
      certificate: /home/user/_shared/cert/fullchain.pem
      privateKey: /home/user/_shared/cert/privkey.pem
```

Рисунок 28 – Пример конфигурационного файла с узла 1

1. Установка и настройка двухузлового кластера pacemaker

Параметр стенда:

- сеть 192.168.100.0/24;
- адрес маршрутизатора (шлюза) сети 192.168.100.5;
- предполагается, что в сети отсутствует служба DNS, поэтому адреса узлов задаются с помощью файла /etc/hosts.

Структура стенда

Для стенда используется два компьютера:

1. В качестве IP-адреса кластера использован адрес 192.168.100.133;
2. Для развертывания кластера используются два подсоединенных к сети компьютера с установленной ОС RedOS. Каждый из этих компьютеров будет выполнять роль узла кластера, поэтому далее они будут называться Узел 1 и Узел 2:

- 192.168.100.134 nod1;
- 192.168.100.135 nod2.

Настройка сетевых подключений

Узлам рекомендуется присвоить статические адреса. Если на узлах кластера для управления сетевыми подключениями используется `Network Manager`, то настройку статических адресов можно выполнить с помощью графического инструмента или из командной строки.

```
sudo nano/etc/NetworkManager/system-connections
```

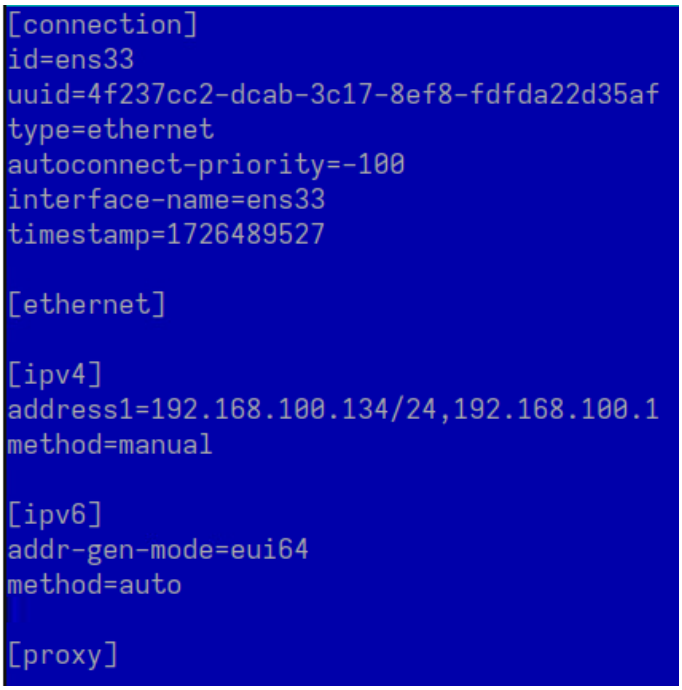


Рисунок 4 – Сетевые настройки на узле 1

```
[connection]
id=ens32
uuid=09f8af38-4eef-3530-8cc2-1e5e5b20f678
type=ethernet
autoconnect-priority=-100
interface-name=ens32
timestamp=1726493364

[ethernet]

[ipv4]
address1=192.168.100.135/24,192.168.100.1
method=manual

[ipv6]
addr-gen-mode=eui64
method=auto

[proxy]
```

Рисунок 5 – Сетевые настройки на узле 2

Настройка разрешения имен узлов с помощью /etc/hosts

На всех узлах кластера внести изменения в файл sudo nano etc/hosts:

- удалить строки, начинающиеся с 127.0.1.1
- внести строки с IP адресами и именами узлов.

Данные настройки делаются на всех узлах (см. Рисунок 29).

```
 GNU nano 4.3                                /etc/hosts
192.168.100.134 node1
192.168.100.135 node2
::1         localhost localhost.localdomain localhost6
```

Рисунок 29 - Настройка файла /etc/host на узле 1

2. Установка пакетов кластерного ПО

Установка пакетов кластерного ПО (выполняется на каждом узле кластера)

```
sudo apt install pacemaker pc
```

После установки пакета назначить пользователю hacluster пароль (для примера admin1q2W):

```
sudo passwd hacluster
```

Инициализация кластера (выполняется однократно на любом из узлов кластера)

Далее удалить все существующие файлы конфигурации кластера:

```
sudo pcs cluster destroy
```

После чего собрать кластер (для примера используется имя кластера astracluster):

```
sudo pcs host auth nod1 nod2 -u hacluster -p admin1q2W

sudo pcs cluster setup astracluster nod1 nod2 --force
```

Запустить инсатнсы кластера и добавить их в автостарт:

```
sudo pcs cluster enable --all
```

В случае отсутствия файла необходимо перейти в папку etc/corosync и создать файл:

```
sudo touh corosync.conf
```

С помощью текстового редактора добавить в файл следующую структуру:

```
totem {
    version: 2
    cluster_name: astracluster
    transport: knet
    crypto_cipher: aes256
    crypto_hash: sha256
}

nodelist {
    node {
        ring0_addr: astra1
        name: astra1
        nodeid: 1
    }

    node {
        ring0_addr: astra2
        name: astra2
        nodeid: 2
    }
}

quorum {
    provider: corosync_votequorum
    two_node: 1
}
```

```
logging {

    to_logfile: yes

    logfile: /var/log/corosync/corosync.log

    to_syslog: yes

    timestamp: on

}
```

Добавить инстансы в автостарт:

```
sudo pcs cluster start --all
```

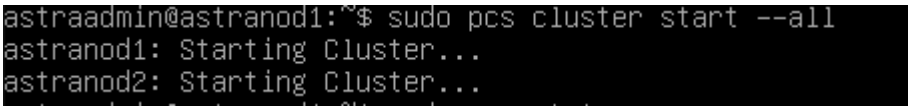


Рисунок 30 – Запуск кластера

Отключить механизм добивания так как он не актуален для 2-ух узлового варианта кластера (для трех и более требуется дополнительная настройки логики):

```
sudo pcs property set stonith-enabled=false
```

Проверить состояние узлов кластера:

```
sudo pcs status
```

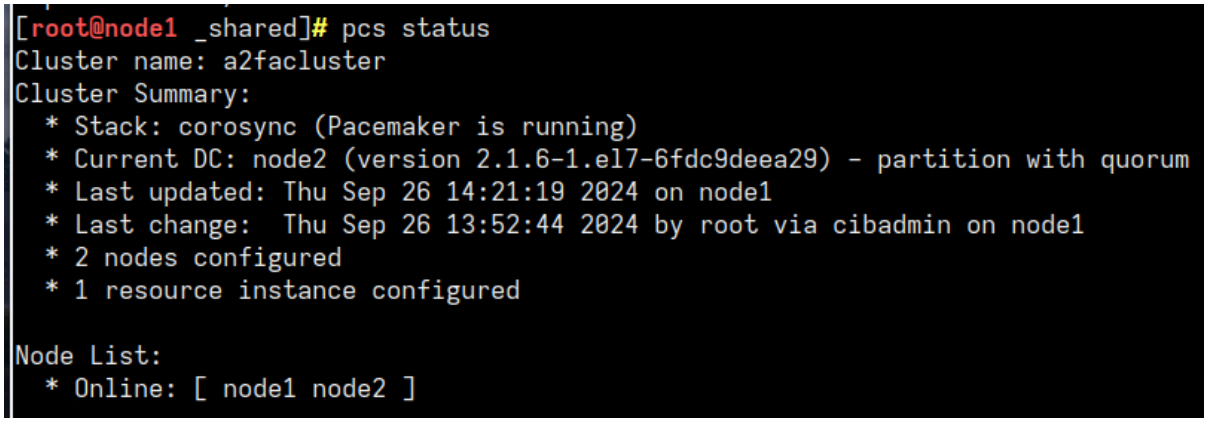


Рисунок 31 – Состояние узлов кластера

Создание активного/пассивного режима работы кластера

- **Добавление ресурса**

Создать уникальный IP-адрес для нашей сети (192.168.100.130), который кластер может использовать на любом из узлов:

```
sudo pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=192.168.100.130
cidr_netmask=32
```

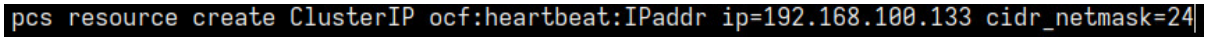


Рисунок 32 – Создание уникального IP-адреса

Перезагрузить кластер:

```
sudo pcs cluster start
```

Рисунок 33 – Запуск кластера с настроенными узлами IP-адресом кластера

- **Запрет перемещения ресурсов**

В большинстве случаев крайне желательно предотвратить перемещение исправных ресурсов по кластеру. Для это в Pacemaker используется концепция «липкости» ресурсов.

```
sudo pcs resource defaults resource-stickiness=100
```

Создание самоподъёмного A2FA на двух нодах

Создание ресурса (выполняется однократно на любом из узлов кластера)

Создать ресурс без параметров отказа:

```
sudo pcs resource create a2fa-service systemd:aladdin-2fa-service.service op
monitor interval=10s timeout=30s
```

Настройка метаданных (выполняется однократно на любом из узлов кластера)

Настройка метаданных производится через pcs resource meta:

```
sudo pcs resource meta a2fa-service migration-threshold=1

sudo pcs resource meta a2fa-service failure-timeout=30s
```

Так же failure-timeout, устанавливал 10 и 60 секунд, разнится в том что при падении aladdin-2fa-service.service поднимается быстрее.

```
sudo pcs status
```

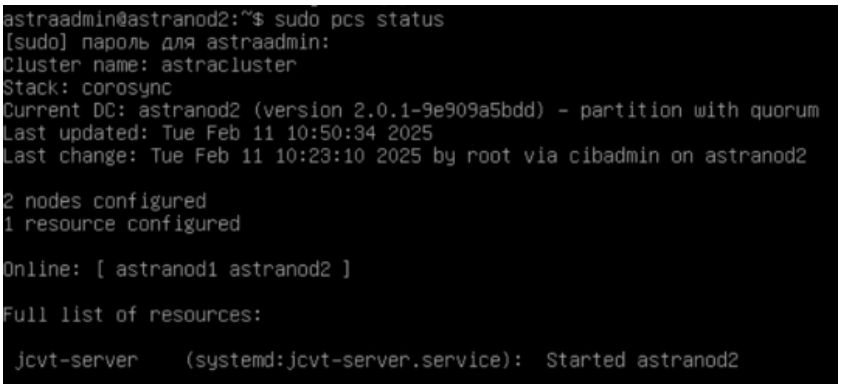


Рисунок 34 - Отслеживание a2fa сервера

Проверка результатов

На одной из нод (где на данный момент запущен aladdin-2fa-service.service), необходимо остановить работу приложения:

```
sudo systemctl stop aladdin-2fa-service.service
```

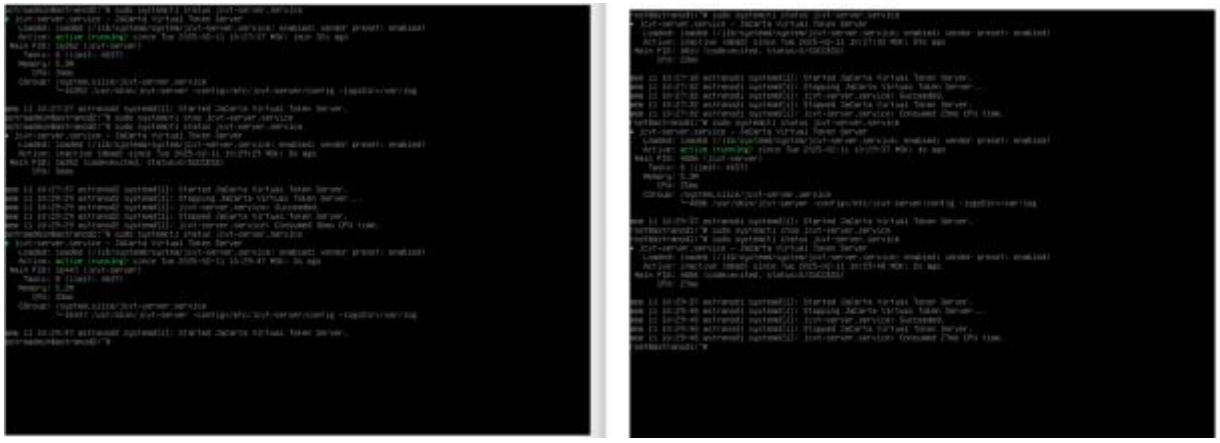


Рисунок 35 - Работа a2fa сервера на разных нодах

Проверить работу приложения можно командой:

```
sudo systemctl status aladdin-2fa-service.service
```

Аналогичную операцию можно повторить на второй ноде также остановив работу приложения и проверив статус приложения на другой. Ожидаемое поведение будет следующим: происходит остановка aladdin-2fa-service.service на node1 и оно же начинает работу на node2, при остановке на node2 приложение поднимается на node1.

Создание группы ресурсов

Для того что бы ресурсы переезжали с ноды на ноду вместе необходимо создать группу:

```
sudo pcs resource group add a2fa ClusterIP a2fa-service
```

где:

- a2fa – название создаваемой группы с ресурсами;
- ClusterIP и a2fas-service это название созданных нами ранее групп.

> *Важно соблюдать порядок, ClusterIP должен быть указанным первым*

3. Настройка NGINX

Для корректной работы всех компонентов A2FA Service у них должен быть сетевой доступ друг к другу. В данном примере показано, как это реализовано с помощью NGINX (созданы локации).

```
 location /cluster9550_jas/ {
        proxy_buffering off;
        proxy_pass https://192.168.100.136:9550/;
}
 location /cluster9551_mob/ {
        proxy_buffering off;
        proxy_pass https://192.168.100.136:9551/;
}
```

Рисунок 36

Одна локации для сервиса JAS, а вторая для exyternallAdress (publicserver).

Локации названы по аналогии с названием разделов в файле конфигурации a2fa server.

Для ClusterIP использовался IP-адрес 192.168.100.130, для которого в NGINX были созданы локации.

Приложение Г. Пример запуска службы A2FA Service от учетной записи без прав root на ОС RedOS

Для настройки запуска службы от учетной записи без права root нужно выполнить следующие шаги:

1. На компьютере или ВМ создать группу и учетную запись:

- через командную строку с помощью команды:

```
groupadd a2fa_user

useradd a2fa_user
```

> *Имя учетной записи пользователя дожно отличаться от имени группы*

- через графический интерфейс (см. Рисунок 37);

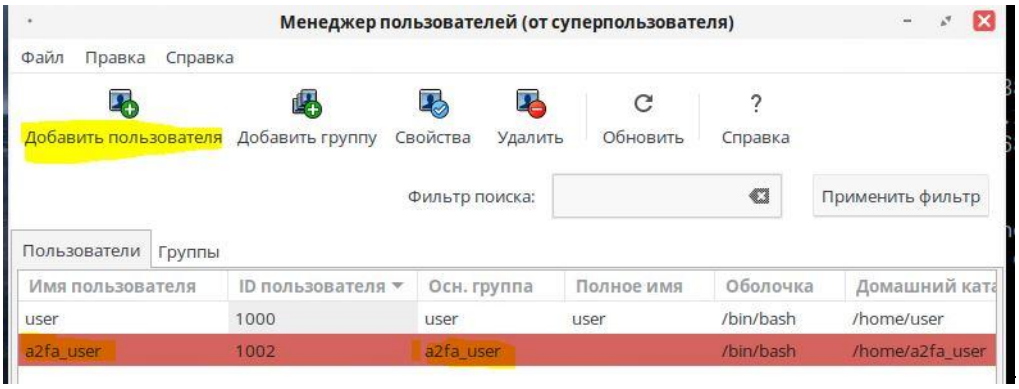


Рисунок 37 – Создание пользователя

2. Добавить пользователя в группу:

```
usermod -g «имя группы» «имя пользователя»

прим: usermod -g a2fa_user a2fa_usertest
```

3. Дальше нужно для юнит-файла создать Drop-in файл. Что бы создать drop-in файл для `/usr/lib/systemd/system/юнит`, создаем каталог `/etc/systemd/system/юнит.d` и помещаем в него файл .conf с добавленными или изменёнными опциями. System будет анализировать эти файлы и применять их поверх оригинального юнита.

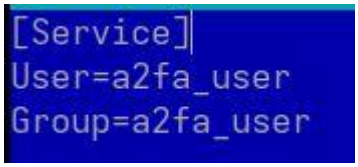
> *Drop-in файл для юнит-файла — это альтернативный способ редактирования юнит-файлов в системе systemd. Вместо прямого редактирования оригинальных файлов, создаются новые файлы с расширением .conf в каталоге /etc/systemd/system/<юнит>.d/. Эти файлы содержат изменённые параметры и опции, которые будут применены поверх оригинального юнит-файла после перезагрузки системы.*
>
> *Юнит-файл — это файл в стиле ini, который содержит информацию о сервисе, сокете, устройстве, точке монтирования, точке автомонтирования, файле или разделе подкачки, цели запуска, пути к файловой системе, таймере, управляемом systemd, срезе управления ресурсами или группе внешне созданных процессов.*

Самый простой и быстрый способ - это команда:

```
systemctl edit «название службы»

пример: systemctl edit aladdin-2fa-service.service
```

Команда откроет `/etc/systemd/system/юнит.d/override.conf` в текстовом редакторе, ввости следующее:

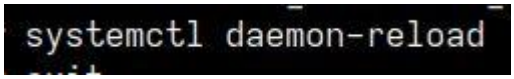


Все остальные строки оставляем закоммиченными или удаляем. После завершения редактирования юнит перезапустится автоматически.

4. Необходимо поменять владельца:

```
# chown -R a2fa_user:a2fa_user /etc/aladdin-2fa-service/ /var/log/aladdin-2fa-service/
```

5. Перезапустить службу демонов:



6. Проверить, каким пользователем запущена текущая служба:

```
[root@redosser system]# ps aux | grep aladdin-2fa-service
root       65241  0.0  0.5 1325976 23784 ?       Ssl  ноя11  10:51 /usr/sbin/aladdin-2fa-service -config=/etc/aladdin-2fa-service/conf
ig -logsDir=/var/log
```

7. Перезапустить службу:

```
[root@redosser system]# systemctl status aladdin-2fa-service.service
● aladdin-2fa-service.service - Aladdin 2FA Service
     Loaded: loaded (/usr/lib/systemd/system/aladdin-2fa-service.service; enabled; vendor preset: disable>
    Drop-In: /etc/systemd/system/aladdin-2fa-service.service.d
             └─override.conf
     Active: active (running) since Tue 2024-11-19 10:25:28 MSK; 6s ago
   Main PID: 3620273 (aladdin-2fa-ser)
      Tasks: 8 (limit: 4642)
     Memory: 8.0M
        CPU: 109ms
     CGroup: /system.slice/aladdin-2fa-service.service
             └─ 3620273 /usr/sbin/aladdin-2fa-service -config=/etc/aladdin-2fa-service/config -logsDir=/v>

ноя 19 10:25:28 redosser.A2FA2019.local systemd[1]: Started Aladdin 2FA Service.
```

И проверить, от какого пользователя стартует служба после смены владельца:

```
[root@redosser system]# ps aux | grep aladdin-2fa
a2fa_us+ 3620273  0.2  0.4 1250900 18780 ?       Ssl  10:25   0:00 /usr/sbin/aladdin-2fa-service -config=/etc/aladdin-2fa-service/conf
ig -logsDir=/var/log
```

Все операции A2FA агента (aladdin-2fa-service.serivce) выполняются УЗ a2fa_user.

8. В ходе запуска сервиса может возникнуть ошибка:

Для файлов с расширением pfx/pem нужно выдать права на чтение пользователем из-под которого запущен демон (a2fa_user).
Так же можно дать доступ всем пользователям на чтение файлов с расширением pem/pfx:

```
sudo chmod +r название_сертификата.pem/pfx
```

Приложение Д. Пример установки и настройки PostgresPRO

Данный тип ПО является платным. Необходимо зарегистрироваться в ЛК и купить лицензию. После чего выдают ключ, который пригодится для скачивания дистрибутива.

Postgres Pro — это коммерческая версия СУБД PostgreSQL, разработанная компанией Postgres Professional. Она предлагает расширенные функции и улучшенную производительность по сравнению с открытым исходным кодом PostgreSQL. Postgres Pro включает в себя дополнительные модули и инструменты для управления базами данных, обеспечения безопасности, аудита и высокой доступности.

BiHA (Built-in High Availability) — это расширение для Postgres Pro, которое обеспечивает отказоустойчивость и автоматическое восстановление после сбоев. Основные особенности BiHA включают:

- **Физическая репликация**: синхронизация данных между узлами кластера через потоковую репликацию файлов WAL.
- **Встроенное аварийное переключение**: автоматическое выявление отказа узла и переключение на резервный узел без потери данных.
- **Выделенный узел-лидер**: узел, доступный для чтения и записи, и узлы-последователи, доступные только для чтения.
- **Синхронная и асинхронная репликация**: возможность выбора режима репликации в зависимости от требований к производительности и надежности.
- **Отсутствие необходимости в стороннем ПО**: все функции отказоустойчивости реализованы внутри Postgres Pro, что упрощает настройку и управление кластером.

BiHA позволяет создавать отказоустойчивые кластеры с высокой доступностью и минимальными простоями, что делает его идеальным решением для критически важных приложений.

Предварительные условия:

- имеются минимум две ВМ с установленной ОС RedOS;
- на каждой ВМ установлен дистрибутив A2FA Service.

Далее необходимо выполнить действия на ВМ, которая будет **мастером**:

```
sudo wget --user ввести ключ из личного кабинета:

--ask-password https://repo.postgrespro.ru/ent/ent-17/keys/pgpro-repo-add.sh
```

1. Запустить скрипт:

```
sudo bash -xe pgpro-repo-add.sh
```

2. Установить PostgreSQL Pro:

```
sudo yum install postgrespro-ent-17 -y
```

3. Остановить службу:

```
sudo systemctl stop postgrespro-ent-17.service
```

4. Зайти под пользователем Postgres:

```
su - postgres
```

5. Выполнить команду на подключение и создание кластера

```
bihactl init --convert --biha-node-id=1 --host=<IP ADDRESS> --port=5432 --
nquorum=2 --minnodes=2 --pgdata=/var/lib/pgpro/ent-17/data **//set password
for biha_replication_user save "magic-string" //IP LEADER host**
```

6. Задать пароль для пользователя BiHA и прописать его в файл:

```
echo "*:*:*:biha_replication_user:<PASSWORD>" > /var/lib/pgsql/.pgpass
```

7. Выдать права на созданный файл:

```
chmod 600 /var/lib/pgsql/.pgpass
```

8. Выйти из-под пользователя Postgres:

```
exit
```

9. Запустить службу Postgres:

```
sudo systemctl start postgrespro-ent-17.service
```

10. Зайти под пользователем Postgres:

```
su - postgres
```

11. Проверить статус сервиса кластеризация BiHA:

```
bihactl status --host=localhost --port=5432
```

Далее необходимо выполнить действия на ВМ, которая будет **workerom**:

```
sudo wget --user ввести ключ из личного кабинета:

--ask-password https://repo.postgrespro.ru/ent/ent-17/keys/pgpro-repo-add.sh
```

1. Запустить скрипт:

```
sudo bash -xe pgpro-repo-add.sh
```

2. Установить PostgreSQL Pro:

```
sudo yum install postgrespro-ent-17 -y
```

3. Остановить службу:

```
sudo systemctl stop postgrespro-ent-17.service
```

4. Удалить папку:

```
sudo rm -rf /var/lib/pgpro/ent-17/data
```

5. Зайти под пользователем Postgres:

```
su – postgres
```

6. Зашифровать секретный ключ в переменную:

```
magicstring="key from LEADER steps #6"
```

7. Задать пароль для пользователя BiHA от ВМ с **мастером**:

```
    echo "*:*:*:biha_replication_user:<PASSWORD>" > /var/lib/pgsql/.pgpass
//PASSWORD   from LEADER steps
```

8. Выдать права на созданный файл:

```
chmod 600 /var/lib/pgsql/.pgpass
```

9. Выполнить команду на создание **workera** и подключение к кластеру:

```
bihactl add --biha-node-id=<2++> --host=<IP ADDRESS> --port=5432 --biha-
port=5433 --magic-string="$magicstring" --pgdata=/var/lib/pgpro/ent-17/data
**//IP WORKER host**
```

10. Выйти из-под пользователя Postgres:

```
exit
```

11. Запустить службу Postgres:

```
sudo systemctl start postgrespro-ent-17.service
```

Далее необходимо выполнить действия на ВМ с ролью **мастера**:

```
su - postgres
psql biha_db
```

Настройки BiHa:

```
select * from biha.config();
```

Изменить настройки кластера:

```
select biha.set_nquorum_and_minnodes(1,1);
```

Например, можно изменить параметры nquorum и minnodes.

Список узлов в кластере:

```
select * from biha.nodes_v;
```

Статус кластера:

```
select * from biha.status_v;
```

Удалить ведомый узел из кластера:

```
select biha.remove_node(2);
```

Принудительно указать лидера кластера:

```
select biha.set_leader(1);
```

По итогу имеется три ВМ: 1 это мастер и 2 worker.

```
[postgres@biha-master-host ~]$ psql biha_db
psql (17.2)
Type "help" for help.

biha_db=# select * from biha.nodes_v;
 id |      host      | port | state  | since_conn_start | conn_count
----+----------------+------+--------+------------------+------------
  1 | 192.168.81.170 | 5433 |        |                  |
  2 | 192.168.81.171 | 5433 | ACTIVE | 18:04:50.250684  |          1
  3 | 192.168.81.172 | 5433 | ACTIVE | 16:52:48.250684  |          2
(3 rows)

biha_db=# select * from biha.status_v;
 id | leader_id | term | online |   state   | last_known_state | since_last_hb
----+-----------+------+--------+-----------+------------------+---------------
  1 |         1 |    2 | t      | LEADER_RW | LEADER_RW        |
  2 |         1 |    2 | t      | FOLLOWER  | FOLLOWER         | 00:00:01.22763
  3 |         1 |    2 | t      | FOLLOWER  | FOLLOWER         | 00:00:01.22763
(3 rows)
```

Рисунок 38

```
database:
  type: postgresql
  user: postgres
  password: PASSWORD
  host: ipadress_nod_1:5432,ipadress_nod_2:5432,ipadress_nod_3:5432
  instance: ""
  parameters: database=testbiha&target_session_attrs=read-write
backupJournals:
```

Рисунок 39 - Пример файл конфигурации A2FA Service

Параметр read-write означает что подключение будет только к лидеру.

Пример:

Текущий лидер находится на ipadress_nod_1, произошёл сбой, и лидер переехал на ipadress_nod_3. Службу A2FA Servcie в таком случае перезагружать не придется, так как ipadress_nod_3 мы указали как один из адресов для подключения, а параметр read-write переподключился к новому лидеру автоматически.

Контакты

Офис (общие вопросы)

Адрес: 129226, Москва, ул. Докукина, д. 16, стр. 1, 7 этаж, компания "Аладдин Р.Д."

Телефон: +7 (495) 223-00-01 (секретарь)

E-mail: aladdin@aladdin.ru (общий)

Web: https://www.aladdin.ru

Время работы: ежедневно с 10:00 до 19:00, кроме выходных и праздничных дней.

Техническая поддержка

Контакты службы техподдержки:

Телефон: +7 (499) 702-39-68
Web: www.aladdin.ru/support/

Список литературы

1 Aladdin 2FA. Руководство пользователя

2 JaCarta Management System v3.7. Руководство администратора. Часть 1. Установка и настройка,
JaCarta Management System 4LX. Руководство администратора. Часть 1. Установка и настройка

3 JaCarta Management System v3.7. Руководство администратора. Часть 2. Функции управления,
JaCarta Management System 4LX. Руководство администратора. Часть 2. Функции управления

4 JaCarta Management System v3.7. Руководство администратора. Часть 3. Установка и настройка сервера аутентификации (JAS),
JaCarta Management System 4LX. Руководство администратора. Часть 3. Установка и настройка сервера аутентификации (JAS),

5 JaCarta Management System v3.7. Руководство пользователя,
JaCarta Management System 4LX. Руководство пользователя

Регистрация изменений

Версия документа	Изменения
1.3	Добавлены: • Пункт 2.3.2.1 WebSocket соединение PUSH-уведомления; Изменено: • Схема сетевого взаимодействия – добавлен WebSocket и TG-сервис; • Описание сценария PUSH-уведомление, в связи с добавлением WebSocket соединения; • Пример конфигурационного файла, добавлены новые параметры;
1.2	Добавлены: • В Приложение п. Сбор логов; п. «Веб-визард»
1.1	Добавление разделов: «3.2 Миграция базы данных с MS SQL в PostgreSQL с помощью утилиты aladdin-2fa-magration-tool», «3.3 Настройка подключения сервера Aladdin 2FA Service к PostgreSQL с использованием TLS»
1.0	Исходная версия документа по настройке сервера Aladdin 2FA Service для ОС Linux