АКЦИОНЕРНОЕ ОБЩЕСТВО «СЕЛФ»

АВТОМАТИЗИРОВАННАЯ СИСТЕМА КОНТРОЛЯ КАНАЛОВ ПРОТИВОАВАРИЙНОЙ АВТОМАТИКИ

Инструкция по установке и настройке

Москва, 2025

Взам. инв. №

Подп. и дата

Инв. № подл.

Изм.	Кол.у	Лист	№	Подп.	Дата
Разраб.		Иванов			
Пров.		Иванов			
ГИП		Иванов			

Инструкция по установке и настройке

Стадия	Лист	Листов
Э	1	26

АО «СЭЛФ»

ТЕРМИНЫ, ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

Сокращение	Расшифровка
API	Программный интерфейс приложения (англ. Application Programming Interface)
HDD	Накопитель на жёстких магнитных дисках (англ. Hard Disk Drive)
JSON	Текстовый формат обмена данными, основанный на JavaScript (англ. JavaScript Object Notation)
SNMP	Простой протокол сетевого управления (англ. Simple Network Management Protocol)
URL	Единый указатель ресурса (англ. Uniform Resource Locator)
АРМ	Автоматизированное рабочее место
АС	Автоматизированная система
БД	База данных
СККПА	Автоматизированная система контроля каналов противоаварийной автоматики (Система)
ИА	Исполнительный аппарат АО «СО ЕЭС»
ОЗУ	Оперативное запоминающее устройство
ОС	Операционная система
ПАК	Программно-аппаратный комплекс
ПО	Программное обеспечение
СО ЕЭС	Системный оператор Единой энергетической системы
ЦПУ	Центральное процессорное устройство

1 ВВЕДЕНИЕ

1.1 Состав комплекта поставки

На Рисунке 1 приведена схема развертывания ПО Системы.

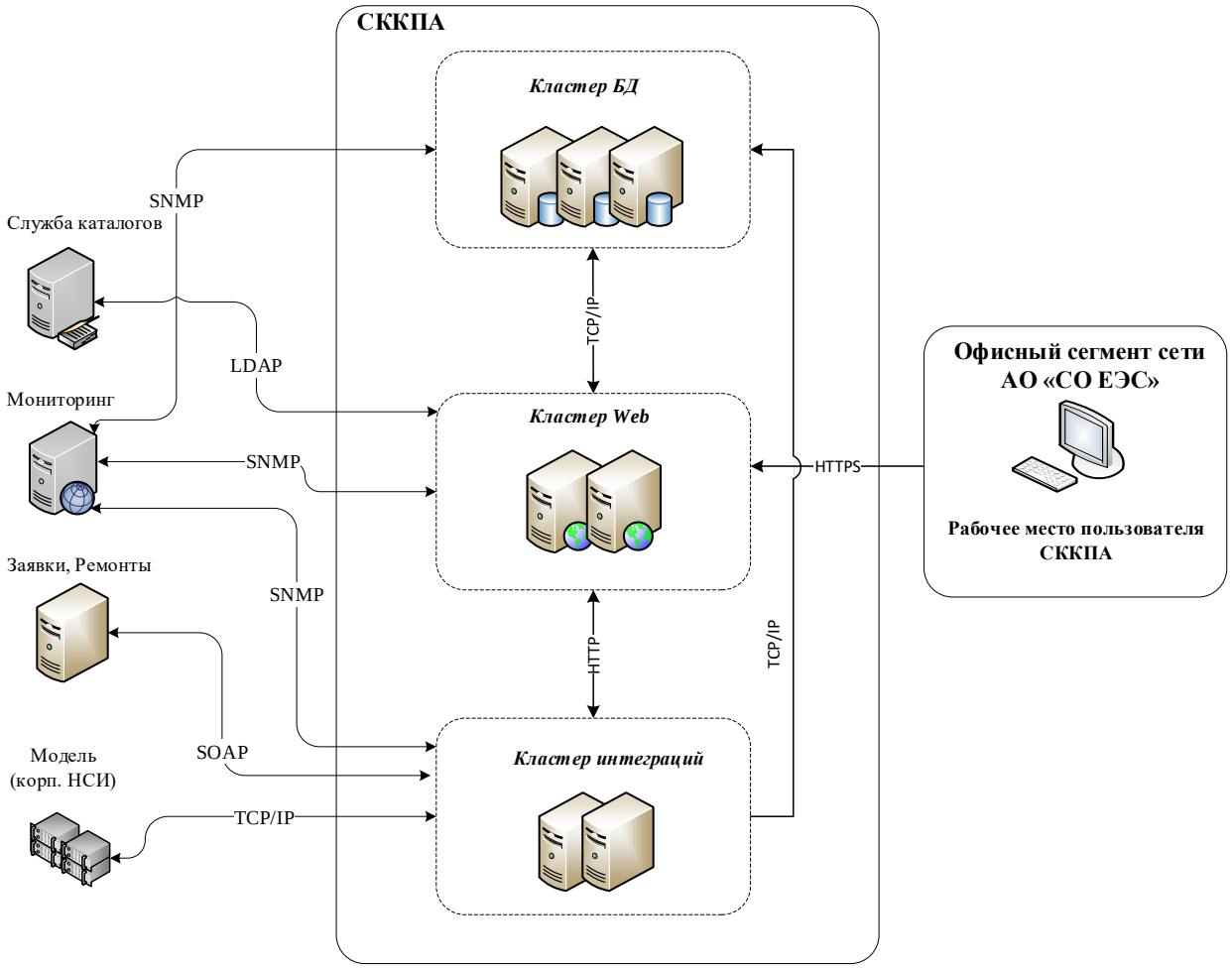

Рисунок 1. Централизованная архитектура СККПА

В комплект поставки ПО Системы для АО «СО ЕЭС» входят инсталляционные пакеты для всех основных компонентов Системы. Перечень этих пакетов представлен в Таблице 1.

Таблица 1. Перечень инсталляционных пакетов

Компонент	Название пакета	Описание
Сервер баз данных	-	Не входит в состав поставляемого ПО

Сервер приложений	Архив с исходным кодом микросервисов. Файлы-примеры с ENV переменными для микросервисов.	Предоставляется в zip-архиве
Сервер интеграций	Архив с исходным кодом микросервисов. Файлы-примеры с ENV переменными для микросервисов.	Предоставляется в zip-архиве

В таблице 2 перечислены сервисы, которые необходимо установить на сервере из локального репозитория ИА АО «СО ЕЭС».

Таблица 2. Перечень серверов с запущенными на них сервисами

№	Назначение сервера	Сервисы
1	Сервер БД1	haproxy keepalived etcd patroni
2	Сервер БД2	haproxy keepalived etcd patroni
3	Сервер БД3	haproxy keepalived etcd patroni
4	Сервер приложений 1	docker docker-compose nginx keepalived
5	Сервер приложений 2	docker docker-compose nginx keepalived
6	Интеграционный сервер 1	docker docker-compose keepalived
7	Интеграционный сервер 2	docker docker-compose keepalived

1.2 Последовательность установки

Установка и настройка ПО Системы на основном стенде производится в следующей последовательности:

1) Проверка серверов, предварительно подготовленных в соответствии с требованиями, представленными в Таблице 2;
2) Установка ПО Системы в следующей последовательности:

- Сервера баз данных;
- Сервера приложений;
- Сервера интеграций.

2 ПОДГОТОВКА К УСТАНОВКЕ

ПО Системы устанавливается на виртуальные сервера, подготовленные в соответствии с требованиями, указанными в Таблице 3.

Таблица 3. Требования к серверам

Компонент	Аппаратное обеспечение	Программное обеспечение
Сервер баз данных 1 (Основное хранилище)	ЦПУ - 18 ядер; ОЗУ - 26 GB; HDD – 568 ГБ свободного дискового пространства RAID5	ОС Astra Linux SE 1.7; СУБД Postgres Pro;
Сервер баз данных 2 (Основное хранилище)	ЦПУ - 18 ядер; ОЗУ - 26 GB; HDD – 568 ГБ свободного дискового пространства RAID5	ОС Astra Linux SE 1.7; СУБД Postgres Pro;
Сервер баз данных 3 (Основное хранилище)	ЦПУ - 18 ядер; ОЗУ - 26 GB; HDD – 568 ГБ свободного дискового пространства RAID5	ОС Astra Linux SE 1.7; СУБД Postgres Pro;
Сервис приложений 1	ЦПУ – 16 ядра; ОЗУ – 20 GB; HDD – 50 GB свободного дискового пространства.	ОС Astra Linux SE 1.7;
Сервис приложений 2	ЦПУ – 16 ядра; ОЗУ – 20 GB; HDD – 50 GB свободного дискового пространства.	ОС Astra Linux SE 1.7;
Сервер интеграций 1	ЦПУ - 12 ядер; ОЗУ - не менее 10 GB; HDD - 50 GB свободного дискового пространства	ОС Astra Linux SE 1.7;
Сервер интеграций 2	ЦПУ - 12 ядер; ОЗУ - не менее 10 GB; HDD - 50 GB свободного дискового пространства	ОС Astra Linux SE 1.7;

2.1 Действия персонала перед началом установки ПО

Внимание! Все действия по установке проводятся под системной учетной записью с правами администратора.

Перед началом установки соответствующий инсталляционный пакет должен быть скопирован на диск целевого сервера.

В случае использования виртуальных машин предпочтительнее использование сетевого файлового хранилища, доступ к которому может быть получен с каждой из ОС целевых серверов. Инсталляционный пакет, расположенный на сетевом файловом хранилище, должен быть скопирован на ОС целевого сервера.

Таким образом, в пользовательской папке каждого сервера (например, в /home/<user>/) должен быть размещен соответствующий инсталляционный пакет (см. таблицу 1) и все дополнительные пакеты.

Для установки программного обеспечения поддержки запуска Docker-контейнеров необходимо наличие доступа к локальному репозиторию АО «СО ЕЭС».

3 УСТАНОВКА И НАСТРОЙКА ПО

3.1 Установка и настройка программного обеспечения Docker-контейнеров

3.1.1 Установка и настройка Docker и Docker-Compose

Установку необходимо произвести на сервере приложений и интеграционном сервере.

1. Обновление установщика пакетов

```
# sudo apt-get update
```

2. Создать файл-конфиг **daemon.json** (до установки пакета докера, т.к в противном случае сразу после установки автоматически будет создана основная сеть по умолчанию) файл в папке **/etc/docker/** с текстом:

```
{
        "live-restore": true,
        "log-opts": {"max-size": "200m", "max-file": "5"},
        "insecure-registries" : ["gitlab…:18180"],
        "default-address-pools":
         [
           {"base":"0.0.0.0/16","size":24}
         ]
}
```

live-restore – позволяет контейнерам продолжать работать, если docker-демон становится недоступным. Этот параметр помогает сократить время простоя контейнера из-за сбоев демона, плановых перезапусков или обновлений.

log-opts – ограничивает размер лог-файлов по умолчанию, не позволяя привести к переполнению диска при работе контейнеров с неограниченным логированием.

3. Установка Docker (выполняется от имени пользователя, являющегося администратором системы (при включенном МКЦ - пользователя с высоким уровнем целостности).

```
# sudo apt install docker.io docker-compose -y
```

4. После установки Docker рекомендуется предоставить администратору право работать с контейнерами не используя sudo. Для этого пользователя нужно включить в группу docker:

```
# sudo exec su - $USER
```

5. Включить автозапуск и запустить docker-сервис:

```
# sudo systemctl enable --now docker
```

6. Проверка работы Docker

```
# docker ps
```

3.1.2 Ограничение объема памяти, предоставляемой контейнерам Docker

Для того, чтобы работало ограничение объема памяти, предоставляемой контейнерам Docker, следует:

1. К параметрам загрузки ядра в файле /etc/default/grup в строку значений параметра GRUB_CMDLINE_LINUX_DEFAULT добавить параметры:
   cgroup_enable=memory swapaccount=1

   Например:

   GRUB_CMDLINE_LINUX_DEFAULT="quiet net.ifnames=0 parsec.mac=0 parsec.max_ilev=63 cgroup_enable=memory swapaccount=1"
2. Для применения настройки, выполнить команду
   # sudo grub-mkconfig -o /boot/grub/grub.cfg

3.2 Установка и настройка серверов баз данных

Для кластера БД на каждом узле (Сервер БД1, Сервер БД2, Сервер БД3) нужно провести дополнительную конфигурацию:

1. Подключить и настроить репозитории содержащие Postgres Pro.
2. Выполнить обновление списка пакетов и системы:
   # sudo apt update && sudo apt upgrade --enable-upgrade
3. Установить пакеты необходимые пакеты
   # sudo apt install postgrespro-std-13 etcd patroni haproxy keepalived openssl ca-certificates
4. Запрещаем запуск и останавливаем службу postgresql (управлением сервисом PostgreSQL должен заниматься Patroni)
   # sudo systemctl stop postgrespro-std-13
   # sudo systemctl disable postgrespro-std-13
5. Запустите утилиту pg-wrapper с правами администратора системы, чтобы добавить и клиентские, и серверные установленные программы в путь поиска PATH, а также включить страницы man по SQL в файл конфигурации страниц man. Эта утилита входит в состав пакета postgrespro-std-13-client:
   # sudo /opt/pgpro/std-13/bin/pg-wrapper links update
6. Разрешаем пересылку транзитных пакетов и использование процессам виртуального IP
   # sudo sh -c "cat << EOF >> /etc/sysctl.conf
   net.ipv4.ip_forward = 1
   net.ipv4.ip_nonlocal_bind = 1
   EOF"
7. Проверить результат командой
   # sysctl -p
8. Настройка приложения keepalived:
   а. Создать конфигурационный файл по адресу:
   # /etc/keepalived/keepalived.conf
   б. Добавить в конфигурационный файл параметры согласно листингу ниже, заменив данные на свои.
   Замены потребуют параметры:

```
vrrp_instance
interface
auth_pass
virtual_ipaddress
```

б.1 Для мастер ноды

```
vrrp_instance skkpadb {
interface eth0
state MASTER
virtual_router_id 200
priority 100
advert_int 1
authentication {
       auth_type PASS
       auth_pass [AUTH_PASSWORD]
       }
virtual_ipaddress  {
       [VIRTUAL_IP]
       }
}
```

Готовая настройка для хоста 1

```
vrrp_instance sqlservername1 {
  interface eth0
  state MASTER
  virtual_router_id 200
  priority 100
  advert_int 1
  authentication {
    auth_type PASS
    auth_pass ПАРОЛЬ
  }
  virtual_ipaddress  {
    ВИРТУАЛЬНЫЙ.IP-АДРЕС.КЛАСТЕРА
  }
}
```

б.2 Для резервной ноды

```
vrrp_instance skkpadb {
interface eth0
state BACKUP
virtual_router_id 200
priority 99
advert_int 1
authentication {
       auth_type PASS
       auth_pass [AUTH_PASSWORD]
       }
virtual_ipaddress  {
       [VIRTUAL_IP]
       }
}
```

Готовая настройка для хоста 2

```
vrrp_instance sqlservername2 {
  interface eth0
  state BACKUP
  virtual_router_id 200
  priority 99
  advert_int 1
  authentication {
    auth_type PASS
    auth_pass ПАРОЛЬ
  }
  virtual_ipaddress  {
    ВИРТУАЛЬНЫЙ.IP-АДРЕС.КЛАСТЕРА
  }
}
```

Готовая настройка для хоста 3

```
vrrp_instance sqlservername3 {
  interface eth0
  state BACKUP
  virtual_router_id 200
  priority 98
  advert_int 1
  authentication {
    auth_type PASS
    auth_pass ПАРОЛЬ
  }
  virtual_ipaddress  {
    ВИРТУАЛЬНЫЙ.IP-АДРЕС.КЛАСТЕРА
  }
}
```

9. Выполнить команду для перезапуска сервиса с новыми настройками (на каждом сервере)

   # sudo systemctl restart keepalived

10. Настройка etcd задается в файле /etc/default/etcd Необходимые параметры для замены:

    ETCD_NAME – Название ноды, должны быть уникальное

    ETCD_INITIAL_CLUSTER_TOKEN – Токен для авторизации новой ноды кластера, один для всех

    ETCD_INITIAL_CLUSTER – перечисляем ноды, которые будем использовать формат:

    Node1=http://0.0.0.0:2380,Node2=0.0.0.0:2380,http://Node3=0.0.0.0:2380

    ETCD_INITIAL_ADVERTISE_PEER_URLS – список равноправных URL-адресов, по которым его могут найти остальные узлы кластера. Эти адреса используются для передачи данных по кластеру. По крайней мере,

один из этих адресов должен быть маршрутизируемым для всех членов кластера. Могут содержать доменные имена. Используется только при первом запуске нового узла кластера.

ETCD_ADVERTISE_CLIENT_URLS – Список равноправных URL-адресов, по которым его могут найти остальные узлы кластера. Эти адреса используются для передачи данных по кластеру. По крайней мере, один из этих адресов должен быть маршрутизируемым для всех членов кластера. Могут содержать доменные имена.

ETCD_LISTEN_PEER_URLS – задаёт схему и точку подключения для остальных узлов кластера, по шаблону scheme://IP:port. Схема может быть http, https. Альтернатива, unix:// или unixs:// для юникс сокетов. Если в качестве IP адреса указано 0.0.0.0, то указанный порт будет прослушиваться на всех интерфейсах

ETCD_LISTEN_CLIENT_URLS – задаёт схему и точку подключения для клиентов кластера. В остальном совпадает с ETCD_LISTEN_PEER_URLS.

По окончании настройки, после первого запуска и успешного соединения нод etcd, необходимо параметр ETCD_INITIAL_CLUSTER_STATE изменить с **new** на **existing.**

a.1 Пример настройки MASTER ноды:

```
ETCD_NAME="skkpadb1"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster"
ETCD_INITIAL_CLUSTER="skkpadb1=http://[NODE_1_IP]:2380,skkpadb2=http://
[NODE_2_IP]:2380,skkpadb3=http://[NODE_3_IP]:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://[NODE_1_IP]:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://[NODE_1_IP]:2379,http://127.0.0.1:2379"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS=http://0.0.0.0:2379
```

Готовая (после первого запуска) настройка для хоста 1

```
ETCD_NAME="host1"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_INITIAL_CLUSTER_STATE="existing"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster"
ETCD_INITIAL_CLUSTER=" host1=http://0.0.0.0:2380,
host2=http://0.0.0.0:2380,host3=http://0.0.0.0:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://0.0.0.0:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://0.0.0.0:2379"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS="http://0.0.0.0:2379"
```

a.2 Пример настройки BACKUP нод:

```
ETCD_NAME="skkpadb2
ETCD_DATA_DIR="/var/lib/etcd"
# ВНИМАНИЕ! Параметр ETCD_INITIAL_CLUSTER_STATE="new"
# после первого запуска и успешного соединения ETCD-кластера, необходимо
# сменить на ETCD_INITIAL_CLUSTER_STATE="existing" (см. пример готовой конфигурации)
# После смены параметра требуется перезапуск службы на нодах (service etcd restart)
```

```
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster"
ETCD_INITIAL_CLUSTER="skkpadb1=http://[NODE_1_IP]:2380,skkpadb2=http://
[NODE_2_IP]:2380,skkpadb3=http://[NODE_3_IP]:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://[NODE_1_IP]:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://[NODE_1_IP]:2379,http://127.0.0.1:2379"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS=http://0.0.0.0:2379
```

Готовая (после первого запуска) настройка для хоста 2

```
ETCD_NAME="host2"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_INITIAL_CLUSTER_STATE="existing"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster"
ETCD_INITIAL_CLUSTER="host1=http://0.0.0.0:2380,
host2=http://0.0.0.0:2380,host3=http://0.0.0.0:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://0.0.0.0:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://0.0.0.0:2379"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS="http://0.0.0.0:2379"
```

11.Перезапускаем службу etcd на всех нодах

```
# sudo systemctl restart etcd
```

12. Проверяем работу:

```
# etcdctl member list -w table
# etcdctl endpoint status --cluster -w table
```

13. Настройка patroni

Перед запуском кластера на patroni, желательно удалить все данные из DATA-каталога постгреса на всех нодах (остановив PG и заблокировав автозапуск службы – см. пункт 3.2).

Пример конфигурации задается в файле /etc/patroni/config.yml

```
name: skkpadb1
scope: pgcluster
restapi: # Настройки для мониторинга HAproxy
listen: [NODE_1_IP]:8008
connect_address: [NODE_1_IP]:8008
etcd:
  hosts: [NODE_1_IP]:2379, [NODE_2_IP]:2379, [NODE_3_IP]:2379 # перечисляем адреса и
порты нод ETCD
bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout : 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: true
      use_slots: true
      parameters:
        wal_keep_segments: 100
  initdb:
  - encoding: UTF8
  - data-checksums
  pg_hba:
  - host replication replicator 0.0.0.0/0 md5
```

```
    - host all all 0.0.0.0/0 md5

    users: # создаем пользователей с нужными правами
      postgres:
        password: [PASSWORD]
        options:
          - createrole
          - createdb
      replicator:
        password: [PASSWORD]
        options:
          - replication
  postgresql:
    listen: 0.0.0.0:5432
    connect_address: [NODE_1_IP]:5432
    data_dir: /var/lib/pgpro/std-13/data/
    bin_dir: /opt/pgpro/std-13/bin/
    config_dir: /var/lib/pgpro/std-13/data/

    authentication:
      replication:
        username: replicator
        password: [PASSWORD]
      superuser:
        username: postgres
        password: [PASSWORD]
  parameters:
    unix_socket_directories: '/var/run/postgresql/'
    stats_temp_directory: /var/lib/pgpro/pgsql_stats_tmp
```

Готовая настройка /etc/patroni/config.yml для хоста 1

```
name: host1
scope: pg13cluster

restapi: # Настройки для мониторинга HAproxy
  listen: 0.0.0.0:8008
  connect_address: 0.0.0.0:8008

etcd:
  hosts: 0.0.0.0:2379, 0.0.0.0:2379, 0.0.0.0:2379

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout : 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: true
      use_slots: true
      parameters:
        wal_keep_segments: 100
  initdb:
  - encoding: UTF8
  - data-checksums
  pg_hba:
  - host replication replicator 0.0.0.0/0 md5
  - host all all 0.0.0.0/0 md5

  users: # создаем пользователей с нужными правами
    postgres:
      password: ПАРОЛЬ-СУПЕРПОЛЬЗОВАТЕЛЯ
      options:
        - createrole
        - createdb
    replicator:
      password: ПАРОЛЬ-РЕПЛИКА-ПОЛЬЗОВАТЕЛЯ
      options:
        - replication

postgresql:
  listen: 0.0.0.0:5432
  connect_address: 0.0.0.0:5432
  data_dir: /var/lib/pgpro/std-13/data/
  bin_dir: /opt/pgpro/std-13/bin/
  config_dir: /var/lib/pgpro/std-13/data/
  authentication:
    replication:
      username: replicator
      password: ПАРОЛЬ-РЕПЛИКА-ПОЛЬЗОВАТЕЛЯ
    superuser:
      username: postgres
      password: ПАРОЛЬ-СУПЕРПОЛЬЗОВАТЕЛЯ

parameters:
  unix_socket_directories: '/var/run/postgresql/'
  stats_temp_directory: /var/lib/pgpro/pgsql_stats_tmp
```

Готовая настройка /etc/patroni/config.yml для хоста «2

```
name: host2
scope: pg13cluster

restapi: # Настройки для мониторинга HAproxy
  listen: 0.0.0.0:8008
  connect_address: 0.0.0.0:8008

etcd:
  hosts: 0.0.0.0:2379, 0.0.0.0:2379, 0.0.0.0:2379

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout : 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: true
      use_slots: true
      parameters:
        wal_keep_segments: 100
  initdb:
  - encoding: UTF8
  - data-checksums
  pg_hba:
  - host replication replicator 0.0.0.0/0 md5
  - host all all 0.0.0.0/0 md5

  users: # создаем пользователей с нужными правами
    postgres:
      password: ПАРОЛЬ-СУПЕРПОЛЬЗОВАТЕЛЯ
      options:
        - createrole
        - createdb
    replicator:
      password: ПАРОЛЬ-РЕПЛИКА-ПОЛЬЗОВАТЕЛЯ
      options:
        - replication

postgresql:
  listen: 0.0.0.0:5432
  connect_address: 0.0.0.0:5432
  data_dir: /var/lib/pgpro/std-13/data/
  bin_dir: /opt/pgpro/std-13/bin/
  config_dir: /var/lib/pgpro/std-13/data/
  authentication:
    replication:
      username: replicator
      password: ПАРОЛЬ-РЕПЛИКА-ПОЛЬЗОВАТЕЛЯ
    superuser:
      username: postgres
      password: ПАРОЛЬ-СУПЕРПОЛЬЗОВАТЕЛЯ

parameters:
  unix_socket_directories: '/var/run/postgresql/'
  stats_temp_directory: /var/lib/pgpro/pgsql_stats_tmp
```

14. *В случае очистки DATA-каталога постгреса перед запуском кластера, этот пункт пропускается.*
Разрешаем подключение пользователям postgres по сети, необходимо выполнить на всех нодах:

```
# sudo sh -c "cat << EOF >> /var/lib/pgpro/std-13/data/pg_hba.conf
host replication replicator 0.0.0.0/0 md5
host all all 0.0.0.0/0 md5
EOF"
```

15. Перезапускаем сервис patroni на каждой ноде

```
# sudo systemctl restart patroni
```

16.Настройка haproxy

Добавить в конфигурационный файл по пути /etc/haproxy/haproxy.cfg
Необходимо будет заменить строку вида:

```
skkpadb1 [NODE_1_IP]:5432
```

на:

```
Название_ноды IP_Ноды:Порт_Ноды

global
    maxconn 100
    log 127.0.0.1 local0
defaults
   log global
    mode tcp
    retries 2
    timeout client 30m
    timeout connect 4s
    timeout server 30m
    timeout check 5s
listen stats
    mode http
    bind *:7000
    stats enable
    stats uri /
listen postgres
    bind *:5000
    option httpchk
    http-check expect status 200
    default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
    server skkpadb1 [NODE_1_IP]:5432 maxconn 100 check port 8008 # указываем 1 ноду
    server skkpadb2 [NODE_2_IP]:5432 maxconn 100 check port 8008 # указываем 2 ноду
    server skkpadb3 [NODE_3_IP]:5432 maxconn 100 check port 8008 # указываем 3 ноду
```

17. После чего необходимо перечитать конфигурационные файлы демонов и перезапустить службы на нодах

```
# sudo systemctl daemon-reload && sudo systemctl restart haproxy patroni
```

18. Для проверки работоспособности можно перейти по адресу:

```
# http://виртуальный_ip:7000
```

или

```
# http://Нода_IP:7000
```

с соблюдением очередности:
1. Сервер БД1

2. Сервер БД2
3. Сервер БД3

3.2.1 Первичная настройка базы данных

Выполнить только на Сервере БД1 (мастер-нода):

1. Зайти под пользователем postgres в psql:

```
# psql -U postgres -h [ip_адрес_мастер-ноды] -p 5000 -W
```

2. Создать пользователя skkpa:

```
# create user skkpa;
```

3. Создать базу данных skkpa с владельцем skkpa:

```
# create database skkpa with owner skkpa;
```

4. Передать пользователю skkpa права postgres:

```
# grant postgres to skkpa;
```

5. Добавить права на подключение пользователя skkpa к базе данных skkpa:

```
# grant connect on database skkpa to skkpa;
```

6. Назначить пользователю skkpa пароль:

```
# alter user skkpa with password '';
```

Заполнение базы данных из дампа

Для запуска системы с предзагруженными данными можно воспользоваться заполнением БД из дампа на сервере БД1:

1. Поместить дамп на сервер
2. Выполнить команду:

```
# psql -U postgres -h [ip_адрес_мастер-ноды] -p 5000 -W -d skkpa < имя_файла_дампа
```

3. Дождаться завершения и выйти из контейнера

3.3 Распаковка SSL ключей

На сервере приложений:

1. Перенести pfx сертификат на сервер.
2. Распаковать сертификат:

```
# openssl pkcs12 -in certificate.pfx -clcerts -nokeys -out certificate.crt
```

3. Распаковать ключ:

```
# openssl pkcs12 -in certificate.pfx -nodes -nocerts -out key-encrypted.key
```

3.4 Создание spnego.keytab

1. Создать нового пользователя в корпоративной службе каталогов. От имени этого пользователя будет выполняться запросы в службу каталогов при сквозной аутентификации. Путь учетной записи должен содержать только латинские символы (кириллические символы не

поддерживаются spnego). Рекомендуется в настройках пользователя запретить смену пароля пользователем и срок жизни пароля сделать неограниченным. При изменении пароля учетной записи потребуется повторно создать keytab файл.

2. Добавить SPN в пользователя:
   # setspn -A HTTP/appserver @[].[] <УЗ>
3. Проверить, что SPN записи добавлены:
   # setspn -L <УЗ>
4. Сгенерировать C:\temp\spnego:
   # ktpass /princ HTTP/appserver @[].[] /mapuser <УЗ>@[].[] /crypto ALL /ptype KRB5_NT_PRINCIPAL /target dc1.[].[] /out C:\temp\spnego.keytab /pass <>

3.5 Настройка Keepalived на серверах приложений и интеграционных серверах

19. Выполнить обновление списка пакетов и системы:
    # sudo apt update && sudo apt upgrade --enable-upgrade
20. Установить пакеты необходимые пакеты
    # sudo apt install keepalived

21. Разрешаем пересылку транзитных пакетов и использование процессам виртуального IP
    # sudo sh -c "cat << EOF >> /etc/sysctl.conf
    net.ipv4.ip_forward = 1
    net.ipv4.ip_nonlocal_bind = 1
    EOF"

22. Проверить результат командой
    # sysctl -p

23. Настройка приложения keepalived:

    а. Создать конфигурационный файл по адресу:
    # /etc/keepalived/keepalived.conf

    б. Добавить в конфигурационный файл параметры согласно листингу ниже, заменив данные на свои.

    Замены потребуют параметры:

```
vrrp_instance
interface
virtual_router_id
auth_pass
virtual_ipaddress
```

    б.1 Для мастер ноды

```
vrrp_instance [NODE_NAME] {
interface eth0
state MASTER
virtual_router_id [ROUTER_ID]
priority 100
advert_int 1
authentication {
```

```
        auth_type PASS
        auth_pass [AUTH_PASSWORD]
        }
virtual_ipaddress  {
        [VIRTUAL_IP]
        }
}
```

б.2 Для резервной ноды

```
vrrp_instance [NODE_NAME] {
interface eth0
state BACKUP
virtual_router_id [ROUTER_ID]
priority 99
advert_int 1
authentication {
        auth_type PASS
        auth_pass [AUTH_PASSWORD]
}
virtual_ipaddress  {
        [VIRTUAL_IP]
        }
}
```

24. Выполнить команду для перезапуска сервиса с новыми настройками (на каждом сервере)

```
# sudo systemctl restart keepalived
```

25. Для удобства последующей настройки рекомендуется для [VIRTUAL_IP] прописать в DNS домен.

3.6 Установка и настройка сервера приложений

1. Обновление установщика пакетов
   ```
   # sudo apt-get update
   ```
1. Распаковать установочный архив (skkpa-install.zip) в /home/<username>/install/.
2. Установка дополнительных пакетов:
   ```
   # sudo apt install -y libkrb5support0 libkrb5-3 build-essential wget tar gcc make libssl-dev libpcre3-dev krb5-user libpam-krb5 krb5-multidev libkrb5-dev zlib1g-dev libcap2-bin libgd-dev libpcre++-dev libxslt-dev git
   ```
3. Скопировать исходный кода nginx и модуля spnego (nginx-1.23.1.tar.gz и spnego-http-auth-nginx-module.zip). Переместить их в директорию /tmp. Распаковать. Перейти в каталог с исходным кодом nginx:
   ```
   # cd /tmp
   # tar -xzvf nginx-1.23.1.tar.gz
   # unzip spnego-http-auth-nginx-module.zip -d ./nginx-1.23.1/spnego-http-auth-nginx-module
   # cd /tmp/nginx-1.23.1
   ```
4. Установка пакетов для сборки nginx:
   ```
   # sudo apt install -y libssl1.1 libssl-dev libxml2-dev libxslt-dev libxpm4 libxpm-dev libtiff5 libexpat1 libexpat1-dev libfontconfig-dev libgd-dev libkrb5support0 libkrb5-3 libkrb5-3 libk5crypto3 libgssapi-krb5-2 libgssrpc4 libkadm5srv-mit12 libkadm5clnt-mit12 libcom-err2 comerr-dev krb5-multidev libkrb5-dev
   ```
5. Убедиться, что в системе отсутствует штатный nginx:

```
# sudo apt remove nginx -y && rm -rf /usr/share/nginx/modules
```

6. Сборка nginx с модулем spnego:

```
# sudo ./configure \
        --user=nginx \
        --prefix=/usr/share/nginx \
        --sbin-path=/usr/sbin/nginx \
        --conf-path=/etc/nginx/nginx.conf \
        --http-log-path=/var/log/nginx/access.log \
        --pid-path=/var/run/nginx.pid \
        --with-mail=dynamic \
        --with-mail_ssl_module \
        --with-stream=dynamic \
        --with-stream_ssl_module \
        --with-stream_realip_module \
        --with-stream_ssl_preread_module \
        --with-http_auth_request_module \
        --with-http_secure_link_module \
        --with-http_gzip_static_module \
        --with-http_stub_status_module \
        --with-http_ssl_module \
        --with-compat \
        --with-pcre \
        --with-pcre-jit \
        --with-http_gunzip_module \
        --with-http_gzip_static_module \
        --with-http_image_filter_module=dynamic \
        --with-http_sub_module \
        --with-http_xslt_module=dynamic \
        --with-stream=dynamic \
        --with-stream_ssl_module \
        --with-mail=dynamic \
        --with-mail_ssl_module\
        --add-module=spnego-http-auth-nginx-module && \
    sudo make && \
    sudo make install
```

7. Создание дополнительных папок:

```
#  sudo mkdir -p /usr/lib/nginx/modules && \
    sudo mkdir -p /var/lib/nginx/body && \
    sudo mkdir -p /var/lib/nginx/fastcgi && \
    sudo mkdir -p /var/lib/nginx/proxy && \
    sudo mkdir -p /var/lib/nginx/scgi && \
    sudo mkdir -p /var/lib/nginx/uwsgi && \
    sudo mkdir -p /etc/nginx/conf.d
```

8. Настройка пользователя nginx:

```
# sudo adduser --system --no-create-home --group --disabled-login --disabled-password
 nginx && \
   sudo setcap cap_net_bind_service=ep /usr/sbin/nginx
# sudo chown -R nginx:nginx /var/lib/nginx
```

9. Создание службы nginx:

```
# sudo nano /lib/systemd/system/nginx.service
```

10. В файл необходимо вписать:

```
[Unit]
Description=The NGINX HTTP and reverse proxy server
After=syslog.target network-online.target remote-fs.target nss-lookup.target
Wants=network-online.target

[Service]
Type=forking
PIDFile=/var/run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t
ExecStart=/usr/sbin/nginx
ExecReload=/usr/sbin/nginx -s reload
ExecStop=/bin/kill -s QUIT $MAINPID
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

11. Запуск службы nginx:
    # systemctl start nginx
12. Проверка службы:
    # systemctl status nginx
13. Создание рабочих каталогов:
    # mkdir -p /etc/ssl/<название сервера>/
    # mkdir -p /etc/nginx/ldap/
14. Положить сертификаты веб-сервера:
    # sudo cp certificate.crt /etc/ssl/<название сервера>/certificate.crt
    # sudo cp key-encrypted.key /etc/ssl/<название сервера>/key-encrypted.key
15. Переложить конфигурацию nginx (предварительно изменить в файле путь к сертификатам из п.15):
    #sudo cp /home/<username>/install/etc_nginx_conf/nginx-skkpa.conf /etc/nginx/conf.d/nginx-skkpa.conf
16. Отредактировать файл nginx-skkpa.conf указав правильный путь к сертификатам из п.15:
    # sudo nano /etc/nginx/conf.d/nginx-skkpa.conf
17. Переложить новый spnego.keytab файл:
    # sudo cp /home/<username>/spnego.keytab /etc/nginx/ldap/spnego.keytab
18. Изменить krb5.conf:
    # sudo nano /etc/krb5.conf

    В начало файла добавить две строки:
    [libdefaults]
        default_realm = domain.domain
19. Перезагрузить nginx:
    # systemctl restart nginx
20. Удалить все временные файлы для сборки:
    # rm -rf /tmp/*

3.7 Установка микросервисов на сервере приложений

2. Создать директорию проекта:

sudo mkdir -p /opt/energo/apps

3. Распаковать установочный архив:
   # unzip skkpa-install.zip -d ./install
4. Скопировать конфиги в директорию проекта:
   # cp /home/<username>/install/_env_shared /opt/energo/.env.shared
5. Создать Docker сеть:
   # sudo docker network create -d bridge energo_bridge
6. Отредактировать .env файлы и прописать правильные пароли или адреса серверов:
   # nano /opt/energo/.env.shared
   # nano /opt/energo/apps/.env.web
   # nano /opt/energo/apps/.env.butler
   # nano /opt/energo/apps/.env.sesame
   # nano /opt/energo/apps/.env.signal-loss

3.8 Запуск микросервисов на сервере приложений

1. Подгрузить Docker образы нужной версии (заменить номер версии):
   # docker load -i web-1.0.0.tar.gz
   # docker load -i butler-1.0.0.tar.gz
   # docker load -i sesame-1.0.0.tar.gz
   # docker load -i signal-loss-1.0.0.tar.gz
2. Сгенерировать Docker конфиги (заменить номер версии):
   # cd /opt/energo/apps/
   # SERVICE_NAME=web IMAGE_NAME=web IMAGE_VERSION=1.0.0 EXPOSE_PORT=8080 docker-compose config > docker-compose.web.yml
   # SERVICE_NAME=sesame IMAGE_NAME=sesame IMAGE_VERSION=1.0.0 EXPOSE_PORT=8000 docker-compose config > docker-compose.sesame.yml
   # SERVICE_NAME=butler IMAGE_NAME=butler IMAGE_VERSION=1.0.0 EXPOSE_PORT=8001 docker-compose config > docker-compose.butler.yml
   # SERVICE_NAME=signal-loss IMAGE_NAME=signal-loss IMAGE_VERSION=1.0.0 EXPOSE_PORT=8002 docker-compose config > docker-compose.signal-loss.yml

3. Запустить микросервисы:
   # cd /opt/energo/apps/
   # docker-compose -f docker-compose.web.yml -p web up -d
   # docker-compose -f docker-compose.sesame.yml -p sesame up -d
   # docker-compose -f docker-compose.butler.yml -p butler up -d
   # docker-compose -f docker-compose.signal-loss.yml -p signal-loss up -d
4. За состоянием сервисов можно наблюдать с помощью команд:
   # sudo docker ps
   Контейнеры web, sesame, butler, signal-loss должны быть в статусе “Up”.
   # sudo docker logs -f <имя контейнера>
   # sudo docker logs –tail=100 -f <имя контейнера>

3.9 Установка микросервисов на интеграционном сервере

5. Создать директорию проекта:
   # sudo mkdir -p /opt/energo/apps
6. Распаковать установочный архив:
   # unzip skkpa-install.zip -d ./install

7. Скопировать конфиги в директорию проекта:
   # cp /home/<username>/install/_env_shared /opt/energo/.env.shared
8. Создать Docker сеть:
   # sudo docker network create -d bridge energo_bridge
9. Отредактировать .env файлы и прописать правильные пароли или адреса серверов:
   # nano /opt/energo/.env.shared
   # nano /opt/energo/apps/.env.model
   # nano /opt/energo/apps/.env.repair-request

3.10 Запуск микросервисов на интеграционном сервере

1. Подгрузить Docker образы нужной версии:
   # docker load -i model-1.0.0.tar.gz
   # docker load -i repair-request-1.0.0.tar.gz
2. Сгенерировать Docker конфиги:
   # cd /opt/energo/apps/
   # SERVICE_NAME=model IMAGE_NAME=model IMAGE_VERSION=1.0.0 EXPOSE_PORT=9001 docker-compose config > docker-compose.model.yml
   # SERVICE_NAME=repair-request IMAGE_NAME=repair-request IMAGE_VERSION=1.0.0 EXPOSE_PORT=9002 docker-compose config > docker-compose.repair-request.yml
3. Запустить микросервисы:
   # cd /opt/energo/apps/
   # docker-compose -f docker-compose.model.yml -p model up -d
   # docker-compose -f docker-compose.repair-request.yml -p repair-request up -d
4. За состоянием сервисов можно наблюдать с помощью команд:
   # sudo docker ps
   Контейнры model, repair-request должны быть в статусе “Up”.
   # sudo docker logs -f <имя контейнера>

4 ОБНОВЛЕНИЕ ПО

Обновление ПО необходимо выполнять в следующем порядке:

1. Сервер приложений
2. Сервер интеграций

4.1 Сервер приложений

1. Остановить запущенные микросервисы

```
# cd /opt/energo/apps/
# sudo docker-compose -f docker-compose.web.yml -p web down
# sudo docker-compose -f docker-compose.sesame.yml -p sesame down
# sudo docker-compose -f docker-compose.butler.yml -p butler down
# sudo docker-compose -f docker-compose.signal-loss.yml -p signal-loss down
```

2. Запустить новую версию микросерисов (п.п. 3.7)

4.2 Сервер интеграций

1. Остановить запущенные микросервисы

```
# cd /opt/energo/apps/
# sudo docker-compose -f docker-compose.model.yml -p model down
# sudo docker-compose -f docker-compose.repair-request.yml -p repair-request down
```

2. Запустить новую версию микросерисов (п.п. 3.9)

5 УДАЛЕНИЕ ПО

5.1 Сервер приложений

Удаление ПО Системы должно производиться пользователем, имеющим права администратора сервера.

Для удаления ПО Системы следует:

1) Остановить докер-контейнеры:

```
# cd /opt/energo/apps/
# sudo docker-compose -f docker-compose.web.yml -p web down
# sudo docker-compose -f docker-compose.sesame.yml -p sesame down
# sudo docker-compose -f docker-compose.butler.yml -p butler down
# sudo docker-compose -f docker-compose.signal-loss.yml -p signal-loss down
```

2) Удалить созданную Docker сеть:

```
# sudo docker network rm energo_bridge
```

3) Удалить содержимое директории /opt/energo.

5.2 Сервер интеграций

Удаление ПО Системы должно производиться пользователем, имеющим права администратора сервера.

Для удаления ПО Системы следует:

4) Остановить докер-контейнеры:

```
# cd /opt/energo/apps/
# sudo docker-compose -f docker-compose.model.yml -p model down
# sudo docker-compose -f docker-compose.repair-request.yml -p repair-request down
```

5) Удалить созданную Docker сеть:

```
# sudo docker network rm energo_bridge
```

6) Удалить содержимое директории /opt/energo.