G3SysAdminDoc

Выпуск 0.1.7

Global System

сент. 16, 2024

Содержание

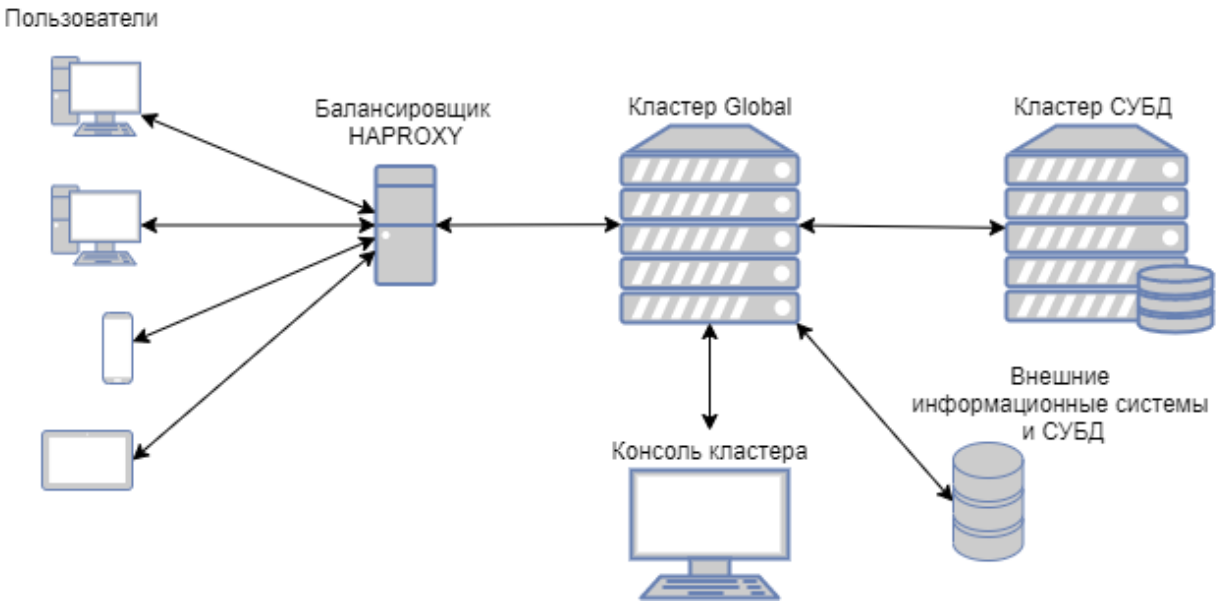
1 РУКОВОДСТВО СИСТЕМНОГО АДМИНИСТРАТОРА

global-system.ru

2 Описание Global ERP

2.1 Назначение и функции системы

Система Global ERP - российская промышленная информационная система для управления предприятием. Является комплексным информационным продуктом, система состоит из набора бизнес-приложений, каждое из которых реализует бизнес-функции и предназначено для использования определенной категорией пользователей.



2.2 Компоненты системы

Для работы Global ERP требуются следующие компоненты:

Компонент	Краткое описание
СУБД PostgreSQL	Система управления базами данных
Сервер приложений Global3 SE	Программный комплекс, выполняющий бизнес логику
Балансировщик нагрузки	Программа, распределяющая пользователей между экземплярами сервера приложений. Используется балансировщик haproxy
Оснастка для управления кластером	Инструментарий для администрирования узлов кластера Global ERP

2.3 Требования к аппаратному обеспечению

Сервер приложений

Характеристика сервера	Рекомендуемые параметры
ОС	Astra Linux / Debian / ALT Linux (в качестве основной используем Astra Linux)
CPU	4 - 32 ядер 2.5 GHz или более
Оперативная память	4 – 128 Gb (напрямую зависит от количества одновременно работающих в системе пользователей)
Память	Твердотельный накопитель SSD 60Gb или больше
Память для файлового хранилища	Если файловое хранилище Global будет располагаться локально на сервере приложений, что обеспечит более быстрый доступ к файлам, то дополнительно потребуется отдельный диск для хранения файлов на 50 Gb или больше (рекомендуется raid любой конфигурации или облачное хранилище).

Сервер СУБД

Характеристика сервера	Рекомендуемые параметры
ОС	Astra Linux / Debian / ALT Linux (в качестве основной используем Astra Linux)
CPU	8 – 16 ядер 2.5 GHz или более
Оперативная память	Минимум 4 Gb (напрямую зависит от количества одновременно работающих в системе пользователей)
Память	SSD raid или гибридный массив минимум 100Gb свободного места для хранения БД
Отдельный раздел для журнала транзакций (опционально)	SSD Диск минимум на 100 Gb или более для журнала транзакций
Отдельный раздел для регистрации аудита Global ERP (опционально)	HDD Диск или RAID минимум на 100 Gb или более для хранения аудита действий в системе

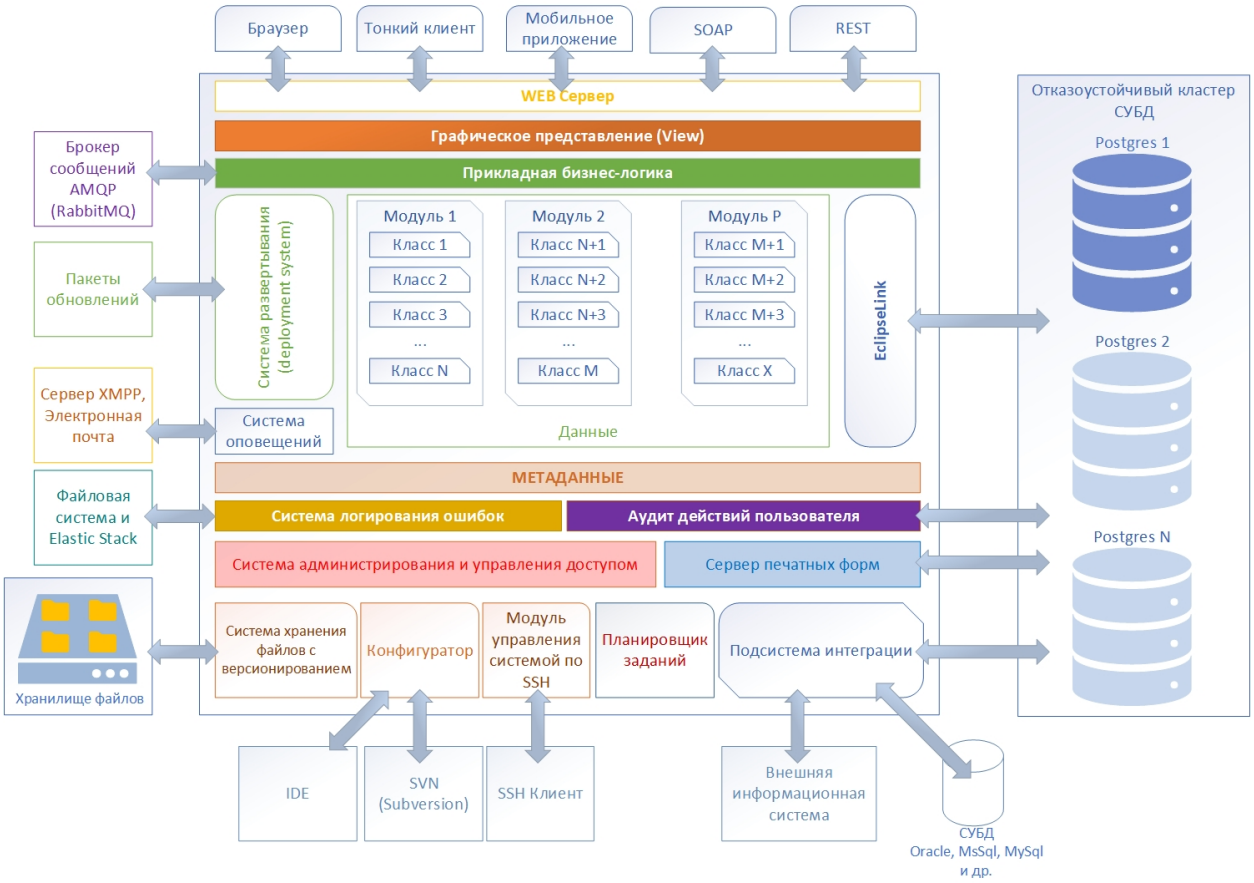
Рабочие станции

Характеристика	Рекомендуемые параметры
ОС	Любая, с поддержкой графического режима и современных браузеров Chrome, Firefox, Яндекс браузер
CPU	2 - 8 ядер 2.5 GHz или более
Оперативная память	4 – 32 Gb или более
Память	10 Gb свободного места
Сеть	100 Мбит или более с высоким качеством связи (без потерь)
Монитор	разрешение 1920×1080 (минимально 1140x900)

2.4 Требования к программному обеспечению

Программное беспечение	Linux
Операционная система сервера СУБД	Astra linux, Debian, Alt Linux
СУБД	Postgres, Postgres Pro 12 и выше
Операционная система сервера приложений	Astra linux, Debian, Alt Linux
Java для сервера приложений	ГосJava, Axiom JDK (Liberica), OpenJDK
Балансировщик нагрузки (для построения кластерного решения)	HAPROXY
Операционная система клиентской машины	Актуальная версия Linux с графической оболочкой или актуальная версия Windows
Клиентское приложение	Браузер: Chrome, Mozilla Firefox, Яндекс браузер, плагин global system с расширением для браузера
Офисное программное обеспечение (для вывода отдельных отчетов и печатных форм)	Libre Office, программа просмотра PDF
Программное обеспечение дизайнера печатных форм	Jasper Studio

2.5 Структура системы



2.6 Перечень основных подсистем

В программный комплекс Global входят следующие подсистемы:

- `Подсистема хранения данных`
  Предназначена для хранения оперативных данных системы, данных для формирования аналитических отчетов.
- `Подсистема управления нормативно-справочной информацией` Предназначена для централизованного ведения классификаторов и справочников, используемых для обеспечения информационной совместимости с другими системами и подсистемами.
- `Подсистема управления правами доступа`
  Предназначена для разграничения прав доступа к функциональности и документам системы.
- `Подсистема аудита`
  Предназначена для ведения и хранения информации о действиях пользователей над документами системы:
  - авторство документов;
  - время и дата изменения атрибутов системы;
  - имена пользователей, вносивших изменения в атрибуты;
  - предыдущее значение измененных атрибутов.

- `Подсистема интеграции`
  Обеспечивает следующие основные виды взаимодействия со смежными системами:
  - прием запросов от смежных систем, обработку полученных запросов и предоставление ответов на запросы;
  - передачу запросов в смежные системы и обработку полученных ответов;
  - ведение журналов учета взаимодействия со смежными системами.
- `Подсистема отчетности`
  Позволяет:
  - проектировать формы регламентированной отчётности в различных форматах на основе данных системы;
  - выводить подготовленные отчёты на печать.
- `Подсистема управления бизнес-приложениями`
  Обеспечивает точную настройку и управление установленными бизнес-приложениями (Склад, документооборот, управление проектами и т.д.).

3 Администрирование Postgres

3.1 Установка и настройка СУБД

Необходимое программное обеспечение под ОС Astra Linux / Debian / Alt Linux

- Сервер PostgreSQL 12 или выше
  PostgreSQL или Postgres Pro
- postgresql-contrib (обычно устанавливаются вместе с сервером Postgres)
- htop
- Iotop
- sysstat
- pgbadger
- ssh (клиент и сервер SSH)
- mc (Midnight Commander)
- tar
- zip

Важно: Не рекомендуется выполнять сборку сервера самостоятельно из исходного кода т.к. нельзя гарантировать стабильность работы таких сборок в продуктивном контуре.

Установка

Скачайте требуемый пакет с сайта https://postgrespro.ru/products/download или https://www.postgresql.org/download. Для установки следуйте инструкциям на сайте.

Дополнительная настройка

- В конфигурационном файле `postgresql.conf` переопределите умолчательные параметры согласно конфигурации железа.
  Пример для сервера с конфигурацией ЦП: 4 ядра; ОЗУ: 8 Гб; Диск: SSD:
  Добавить в конец конфигурационного файла

```
#------------------------------------------------------------------------------
# CUSTOMIZED OPTIONS
#------------------------------------------------------------------------------

# Memory Configuration
shared_buffers = 2GB
effective_cache_size = 6GB
work_mem = 41MB
maintenance_work_mem = 410MB

# Checkpoint Related Configuration
min_wal_size = 2GB
max_wal_size = 3GB
checkpoint_completion_target = 0.9
wal_buffers = -1

# Network Related Configuration
listen_addresses = '*'
max_connections = 100

# Storage Configuration
random_page_cost = 1.1
effective_io_concurrency = 200

# Worker Processes Configuration
max_worker_processes = 8
max_parallel_workers_per_gather = 2
max_parallel_workers = 2

max_locks_per_transaction = 500
```

Примечание: Для подбора параметров можно воспользоваться онлайн-мастером https://www.pgconfig.org

В мастере указать версию постгреса, выбрать профиль `DataWare house and BI Applications` и указать параметры железа сервера: количество ядер ЦП, размер ОЗУ, тип диска.

- Настройте аутентификацию по имени узла в файле `pg_hba.conf`

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
# IPv4 local connections:
host    all             all             all                     md5
```

Примечание: Для получения подробной информации по конфигурации Postgres воспользуйтесь документацией на сайте: https://postgrespro.ru/docs

---

Развертывание новой базы

Поменяйте пароль пользователю ОС с именем `postgres`

```
sudo passwd postgres
```

Подключитесь терминалом к серверу СУБД под административным пользователем

Переключитесь на пользователя «postgres»

```
su postgres
```

Подключитесь локально утилитой psql к СУБД Postgres

```
psql
```

Поменяйте пароль суперпользователя

```
alter user postgres password '<Новый пароль>';
```

Создайте пользователя

```
CREATE ROLE <userName> WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT␣
↪NOREPLICATION CONNECTION LIMIT -1 PASSWORD '<UserPassword>';
GRANT pg_signal_backend TO <userName>;
```

Создайте новую БД, в качестве владельца укажите созданного пользователя

```
CREATE DATABASE "<имяБД>" WITH OWNER = <userName> ENCODING = 'UTF8' LC_COLLATE = 'ru_RU.
↪UTF-8' LC_CTYPE = 'ru_RU.UTF-8' CONNECTION LIMIT = -1;
```

Завершите сессию psql

```
\q
```

Подключитесь к созданной бд

```
psql <имяБД>
```

Подключите, необходимые для работы Global, расширения

```
CREATE EXTENSION if not exists plpgsql;
CREATE EXTENSION if not exists fuzzystrmatch;
CREATE EXTENSION if not exists pg_trgm;
CREATE EXTENSION if not exists pg_stat_statements;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
CREATE EXTENSION if not exists "uuid-ossp";
CREATE EXTENSION if not exists dict_xsyn;
CREATE EXTENSION if not exists ltree;
```

Завершите сессию psql

```
\q
```

Теперь БД готова к работе.

Развертывание поставочного дампа

При получении поставочного дампа нагоните его на созданную БД.

Если БД не пуста и содержит объекты, удалите их

При наличии прав суперпользователя Postgres можно удалить БД

```
DROP DATABASE <имяБД>;
```

и создать заново, выполнив шаги раздела «Развертывание новой базы»

Если прав суперпользователя Postgres нет, воспользуйтесть скриптом удаления

```
SET search_path TO public;
--
DO $$ DECLARE
    r RECORD;
BEGIN
    FOR r IN (SELECT p.oid::regprocedure as sFunctionName
FROM pg_proc p
INNER JOIN pg_namespace ns ON (p.pronamespace = ns.oid)
inner join pg_roles a on p.proowner =a.oid
WHERE ns.nspname = current_schema
  and a.rolname =current_user
  and p.probin is null) LOOP
        EXECUTE 'DROP FUNCTION IF EXISTS ' || r.sFunctionName || ' CASCADE';
    END LOOP;
END $$;
--
DO $$ DECLARE
    r RECORD;
BEGIN
    FOR r IN (SELECT tablename FROM pg_tables WHERE schemaname = current_schema()) LOOP
        EXECUTE 'DROP TABLE IF EXISTS ' || quote_ident(r.tablename) || ' CASCADE';
    END LOOP;
END $$;
--
DO $$ DECLARE
    r RECORD;
BEGIN
    FOR r IN (SELECT  c.relname
FROM pg_class c
```

(continues on next page)

(продолжение с предыдущей страницы)

```
inner join pg_catalog.pg_namespace n on c.relnamespace =n.oid
inner join pg_roles a on c.relowner =a.oid
WHERE (c.relkind = 'S')
  and n.nspname =current_schema
  and a.rolname =current_user) LOOP
        EXECUTE 'drop sequence IF EXISTS ' || quote_ident(r.relname) || ' CASCADE';
    END LOOP;
END $$;
--
DO $$ DECLARE
    r RECORD;
BEGIN
    FOR r IN (SELECT  c.relname
FROM pg_class c
inner join pg_catalog.pg_namespace n on c.relnamespace =n.oid
inner join pg_roles a on c.relowner =a.oid
where c.relkind ='v'
  and n.nspname =current_schema
  and a.rolname =current_user) LOOP
        EXECUTE 'drop view IF EXISTS ' || quote_ident(r.relname) || ' CASCADE';
    END LOOP;
END $$;
--
SELECT lo_unlink(l.oid)
FROM pg_largeobject_metadata l
inner join pg_roles a on l.lomowner =a.oid
WHERE a.rolname =current_user;
```

Поставочный дамп запакован архиватором tar

Имя файла архива имее следующий вид: `<имяБД>_public_<дата>_<ччммсс>.tar` Пример: `demoDb_public_04.11.2022_170406.tar`

Загрузите файл поставочного дампа на сервер Postgres в директорию /usr/dumpstore

Распакуйте архив

```
mkdir -p /tmp/global/Dump
tar -xvf /usr/dumpstore/<имяБД>_public_<дата>_<ччммсс>.tar -C /tmp/global/Dump --strip-
↪components 1
```

Дамп распакуется в каталог `/tmp/global/Dump/<имяБД>_public_<дата>_<ччммсс>`

Восстановите БД из дампа

```
/opt/pgpro/std-12/bin/pg_restore --dbname=postgresql://<sUser>:<sPass>@localhost:5432/
↪<sDbName> -O -x -v --no-tablespaces  --jobs=4 /tmp/global/Dump/<имяБД>_public_<дата>_
↪<ччммсс>
```

/opt/pgpro/std-12/bin/pg_restore - путь до утилиты распаковки дампа (postgres pro 12)

<sUser> - имя пользователя БД

<sPass> - пароль

<sDbName> - имя БД

jobs=4 - количество потоков, указывать по количеству ядер

3.2 Тестирование производительности сервера

Производительность системы зависит от используемого оборудования.

Тестирование проводится с помощью утилиты pgbench, которая входит в комплект поставки СУБД Postgres

Настройка теста pgBench

Подключитесь терминалом к серверу СУБД под административным пользователем

Переключитесь на пользователя «postgres»

```
su postgres
```

Подключитесь локально утилитой psql к СУБД Postgres

```
psql
```

Создайте новую БД, в качестве владельца укажите созданного пользователя

```
CREATE DATABASE "pgbenchDb" WITH OWNER = postgres ENCODING = 'UTF8' LC_COLLATE = 'ru_RU.
↪UTF-8' LC_CTYPE = 'ru_RU.UTF-8' CONNECTION LIMIT = -1;
```

Завершите сессию psql

```
\q
```

Инициализируйте бд для проведения теста

```
pgbench -i -s 50 pgbenchDb
```

Выполните тестирование быстродействия

```
pgbench -c 100 -j 100 -t 10000 pgbenchDb
```

Через несколько минут утилита выдаст результат.

Пример результата теста

```
pgbench (16.1 (Debian 16.1-1.pgdg110+1))
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 50
query mode: simple
number of clients: 100
number of threads: 100
maximum number of tries: 1
number of transactions per client: 10000
number of transactions actually processed: 1000000/1000000
number of failed transactions: 0 (0.000%)
latency average = 1.935 ms
```

(continues on next page)

(продолжение с предыдущей страницы)

```
initial connection time = 56.060 ms
tps = 51677.092175 (without initial connection time)
```

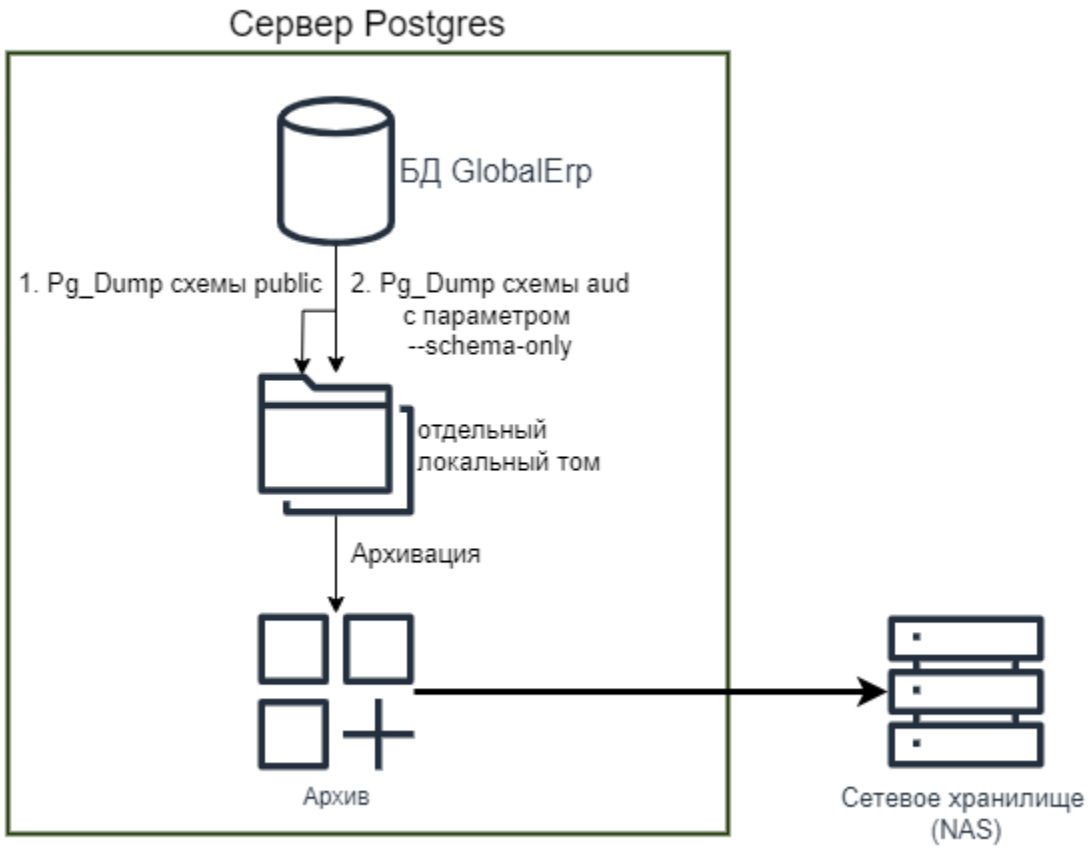
3.3 Резервное копирование и восстановление БД

Для организации бекапа базы мы рекомендуем локальное снятие дампа. Это происходит быстрее, не грузит сеть. Администратор должен следить за свободным местом на диске, свободного места должно хватать для снятия дампа. Лучше всего для сохранения дампа использовать отдельный том. Это позволяет избежать аварийной остановки постгреса при нехватке места на диске.

Локально снятый дамп архивируется и помещается на внешнее NAS хранилище.

Возможно применение сторонних средств резервного копирования, таких как Кибер Бэкап для PostgreSQL

Снятие дампа



Снятие дампа выполняется штатной утилитой pg_dump

Для бекапа снимаем только схему `public`, ее достаточно для сохранения всех данных системы. Отдельно снимаем структуру всех объектов схемы `aud`. Это позволяет при восстановлении не пересоздавать структуры хранения аудита из системы Global.

Если требуется сохранить аудит, то схему `aud` рекомендуется снимать отдельно.

Совет: При включенном аудите в системе Global и высокой интенсивности работы схема aud может содержать большие объемы данных (Несколько терабайт). Создание резервной копии схемы aud может занимать продолжительное время.

Обычно все схемы снимаются для переустановки PostgreSQL или обновления версии PostgreSQL, с последующим нагоном полного дампа.

Для снятия дампа используйте формат «Директория». Это позволяет снимать дамп в многопоточном режиме и существенно ускоряет процесс.

Общий пример запуска утилиты снятия дампа

```
${sDumpUtl} -F d  --dbname=postgresql://${sUser}:${sPass}@${sHost}:5432/${sDbName} --
↪jobs=$nColDbJobs --schema=public  --blobs --verbose
```

где

- ${sDumpUtl} - путь к утилите pg_dump
- ${sUser} - логин
- ${sPass} - пароль
- ${sHost} - хост
- ${sDbName} - имя бд
- $nColDbJobs - количество потоков (выставляется по количеству ядер на сервере postgresql)

Общий пример запуска утилиты для снятия схемы `aud` без данных

```
${sDumpUtl} -F d  --dbname=postgresql://${sUser}:${sPass}@${sHost}:5432/${sDbName} --
↪jobs=$nColDbJobs --schema=aud --schema-only --blobs --verbose --file=$sOutputDumpDir
```

где

- ${sDumpUtl} - путь к утилите pg_dump
- ${sUser} - логин
- ${sPass} - пароль
- ${sHost} - хост
- ${sDbName} - имя бд
- $nColDbJobs - количество потоков (выставляется по количеству ядер на сервере postgresql)

Совет: Параметр `--schema-only` позволяет выгружать только определения объектов (схемы), без данных.

После снятия дампа запаковываем директорию архиватором tar

```
tar -cvf ${sArchiveDir}/${sTarFileName} $sOutputDumpDir
```

где

- ${sArchiveDir} - директория для архива
- ${sTarFileName} - имя архивного файла
- $sOutputDumpDir - директория с дампом

Развертывание дампа

Развертывание дампа выполняет штатная утилита pg_restore

Перед развертыванием дампа обязательно удаляем все объекты схемы `public`

```
$psqlUtl postgresql://${sUser}:${sPass}@${sHost}:5432/${sDbName} -f $SCRIPTPATH/drop.
↪sql
```

где

- ${psqlUtl} - путь к утилите psql
- ${sUser} - логин
- ${sPass} - пароль
- ${sHost} - хост
- ${sDbName} - имя бд
- $SCRIPTPATH - путь до файла drop.sql

файл drop.sql

```
SET search_path TO public;
--
DO $$ DECLARE
    r RECORD;
BEGIN
    FOR r IN (SELECT p.oid::regprocedure as sFunctionName
FROM pg_proc p
INNER JOIN pg_namespace ns ON (p.pronamespace = ns.oid)
inner join pg_roles a on p.proowner =a.oid
WHERE ns.nspname = current_schema
  and a.rolname =current_user
  and p.probin is null) LOOP
        EXECUTE 'DROP FUNCTION IF EXISTS ' || r.sFunctionName || ' CASCADE';
        commit;
    END LOOP;
END $$;
--
DO $$ DECLARE
    r RECORD;
BEGIN
    FOR r IN (SELECT tablename FROM pg_tables WHERE schemaname = current_schema()) LOOP
        EXECUTE 'DROP TABLE IF EXISTS ' || quote_ident(r.tablename) || ' CASCADE';
        commit;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    END LOOP;
END $$;
--
DO $$ DECLARE
    r RECORD;
BEGIN
    FOR r IN (SELECT  c.relname
FROM pg_class c
inner join pg_catalog.pg_namespace n on c.relnamespace =n.oid
inner join pg_roles a on c.relowner =a.oid
WHERE (c.relkind = 'S')
  and n.nspname =current_schema
  and a.rolname =current_user) LOOP
        EXECUTE 'drop sequence IF EXISTS ' || quote_ident(r.relname) || ' CASCADE';
        commit;
    END LOOP;
END $$;
--
DO $$ DECLARE
    r RECORD;
BEGIN
    FOR r IN (SELECT  c.relname
FROM pg_class c
inner join pg_catalog.pg_namespace n on c.relnamespace =n.oid
inner join pg_roles a on c.relowner =a.oid
where c.relkind ='v'
  and n.nspname =current_schema
  and a.rolname =current_user) LOOP
        EXECUTE 'drop view IF EXISTS ' || quote_ident(r.relname) || ' CASCADE';
        commit;
    END LOOP;
END $$;
--
SELECT lo_unlink(l.oid)
FROM pg_largeobject_metadata l
inner join pg_roles a on l.lomowner =a.oid
WHERE a.rolname =current_user;
```

Перед развертыванием дампа его нужно распаковать

```
tar -xvf ${dumpArchiveFile} -C $sTempPath
```

запустите утилиту развертывания дампа

```
$sRestoreUtl --dbname=postgresql://${sUser}:${sPass}@${sHost}:5432/${sDbName} -O -x -
↪v --no-tablespaces  --jobs=$nColDbJobs $sRealDumpDir
```

где

- $sRestoreUtl - путь к утилите pg_restore
- ${sUser} - логин
- ${sPass} - пароль

- ${sHost} - хост
- ${sDbName} - имя бд
- $nColDbJobs - количество потоков (выставляется по количеству ядер на сервере postgresql)
- $sRealDumpDir - путь к каталогу с файлами дампа

4 GlobalServer Автономный режим

4.1 Установка сервера приложений

Автономный режим используется когда нагрузка позволяет использовать один экземпляр сервера приложений.

Необходимое программное обеспечение под ОС Astra Linux / Debian

- postgresql-client (версия клиента должна совпадать с версией сервера)
- jre 8 или jdk 8
  - ГосJava https://lab50.net/gosjava/
  - Axiom JDK (Liberica) https://axiomjdk.ru/pages/downloads/
  - Oracle Java https://java.com/ru/download/
  - другой дистрибутив на основе OpenJDK https://openjdk.java.net/
- expect
- htop
- iotop
- sysstat
- ssh (клиент и сервер SSH)
- mc (Midnight Commander)
- tar
- zip
- unzip
- cifs-utils

Если планируется работать с большим количеством прикрепленных файлов, необходимо подключить дополнительный раздел большого объема. Например сетевой ресурс samba/cifs

Пример подключения через конфигурационный файл `/etc/fstab`

```
# share global attach
//172.16.1.120/globalfiles /mnt/attach cifs _netdev,username=global@testdomain.local,
↪password=123Global321,iocharset=utf8,file_mode=0777,dir_mode=0777 0 0
```

Установка

Скачайте архив дистрибутива и прикладного решения с предоставленного ресурса (Предоставляется через контактное лицо, файлы `globalserver.zip` и `applib.zip`).

Подключитесь терминалом к серверу Global под пользователем root

Создайте временную директорию и загрузите файлы дистрибутива на сервер

```
mkdir -p /tmp/global
```

создайте пользователя и группу `global`

```
sudo groupadd global
sudo useradd -g global global
```

Распакуйте сервер приложений

```
sudo mkdir -p /opt/global/globalserver
sudo unzip /tmp/global/globalserver.zip -d /opt/global/globalserver
```

Распакуйте образ прикладного решения

```
sudo mkdir -p /opt/global/globalserver/application/applib /opt/global/globalserver/
↪application/applibBin
sudo unzip /tmp/global/applib.zip -d /opt/global/globalserver/application/applib
```

Смените владельца

```
sudo chown -R global:global /opt/global/globalserver
```

Выдайте разрешение на запуск

```
sudo chmod ug+x /opt/global/globalserver/start.sh
sudo chmod ug+x /opt/global/globalserver/stop.sh
sudo chmod ug+x /opt/global/globalserver/globalscheduler.sh
```

Настройка сервисов

Сервис нужен для работы системы global в качестве фонового процесса

Сервис global

Для настройки сервиса нужно скопировать файл `/opt/global/globalserver/admin/linux/global3.service.origin` в каталог `/lib/systemd/system` и переименовать `global3.service`

```
sudo cp /opt/global/globalserver/admin/linux/global3.service.origin /lib/systemd/system/
↪global3.service
```

или создать новый файл

```
sudo touch /usr/lib/systemd/system/global3.service
```

Содержимое

```
[Unit]
Description=Система Global
After=multi-user.target

[Service]
User=global
Group=global
AmbientCapabilities=CAP_NET_BIND_SERVICE

# Env Vars
Environment=JAVA_OPTS=-Dfile.encoding=UTF-8
Type=idle
WorkingDirectory=/opt/global/globalserver
ExecStart=/opt/global/globalserver/start.sh
ExecStop=/opt/global/globalserver/stop.sh
TimeoutStopSec=110

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

Разрешите автозагрузку сервиса

```
sudo systemctl daemon-reload
sudo systemctl enable global3
```

Сервис globalscheduler

Для настройки сервиса нужно скопировать файл /opt/global/globalserver/admin/linux/scheduler/globalscheduler.service.origin в каталог /lib/systemd/system и переименовать globalscheduler.service

```
sudo cp /opt/global/globalserver/admin/linux/scheduler/globalscheduler.service.origin /
↪lib/systemd/system/globalscheduler.service
```

или создать новый файл

```
sudo touch /usr/lib/systemd/system/globalscheduler.service
```

Содержимое

```
[Unit]
Description=Менеджер заданий Global SE Postgres
After=multi-user.target

[Service]
Type=idle
WorkingDirectory=/opt/global/globalserver
ExecStart=/opt/global/globalserver/globalscheduler.sh

TimeoutStopSec=110

```

(continues on next page)

(продолжение с предыдущей страницы)

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

Разрешите автозагрузку сервиса

```
sudo systemctl daemon-reload
sudo systemctl enable globalscheduler
```

Настройка сервера приложений

Конфигурация Global

Основной конфигурационный файл системы Global расположен в каталоге `/opt/global/globalserver/application/config/global3.config.xml`

Пропишите бд в основную конфигурацию

```
<databases>
 <database alias="<ПсевдонимБД>" driver="org.postgresql.Driver" schema="PUBLIC"
     url="jdbc:postgresql://<host>:5432/<ИмяБД>" connectionType="proxyShared"␣
↪authenticationType="btk">
  <users>
   <user name="<ИмяПользователяБД>" password="<ПарольПользователяБД>"/>
  </users>
  <metaManager
    mode="Xml"
    defaultNamespace="ru.bitec.app.btk"
    sbtName="main"
  />
  <eclipseLink
    persistenceUnitName="pgdev"
    autoCommit="false"
  />
 </database>

</databases>

<sbts>
 <sbt name="main"
  sourceMode="Jar"
  jarFolder="/opt/global/globalserver/application/applib"
  binaryFolder="/opt/global/globalserver/application/applibBin"
  source="/opt/global/globalserver/application/applib"
  dirWatcherEnabled="false"
 >
  <fileStorages>
   <fileStorage name="Default"
    path="/mnt/attach/"/>
  </fileStorages>
 </sbt>

</sbts>
```

Где:

<ПсевдонимБД> - сокращенное название организации, или доменное имя.

<host> - адрес сервера postgres

<ИмяБД> - имя базы данных, подготовленоой для работы Global System

<ИмяПользователяБД> - пользователь БД

<ПарольПользователяБД> - пароль пользователя БД

Конфигурация запуска

Параметры запуска расположены в файле `/opt/global/globalserver/default.sh`, их можно переопределить в файле `/opt/global/globalserver/parameters.sh`

```
#!/bin/bash
set -x
# Настройки портов
export JETTY_HTTP_PORT=8080

#память: 1000M - в мегабайтах 1G - в гигабайтах
export globalMemory=3G
```

Параметры запуска назначают порты для http сервера, максимальный размер оперативной памяти, а также порты для вспомогательных служб.

Конфигурация планировщика заданий

Расположение основного файла конфигурации: `/opt/global/globalserver/application/config/tools/scheduler/quartz.properties`

Создайте файл конфигурации планировщика

```
sudo mkdir -p /opt/global/globalserver/application/config/tools/scheduler
sudo touch /opt/global/globalserver/application/config/tools/scheduler/quartz.properties
```

Вставьте содержимое

```
org.quartz.scheduler.instanceName = PostgresScheduler
org.quartz.scheduler.instanceId = AUTO

org.quartz.threadPool.class = org.quartz.simpl.SimpleThreadPool
org.quartz.threadPool.threadCount = 500

org.quartz.jobStore.class = org.quartz.impl.jdbcjobstore.JobStoreTX
org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
org.quartz.jobStore.dataSource = quartzDS
org.quartz.jobStore.dontSetAutoCommitFalse=false

org.quartz.dataSource.quartzDS.driver = org.postgresql.Driver
org.quartz.dataSource.quartzDS.URL = jdbc:postgresql://<DBHOST>:5432/<DBNAME>?
↪ApplicationName=Global-Scheduler
org.quartz.dataSource.quartzDS.user = <DBUSER>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
org.quartz.dataSource.quartzDS.password = <DBPASS>

org.quartz.jobStore.tablePrefix=btk_qrtz_
org.terracotta.quartz.skipUpdateCheck=true
```

Где

<DBHOST> - адрес сервера postgres

<DBNAME> - имя БД

<DBUSER> - пользователь БД

<DBPASS> - пароль пользователя БД

Создайте файл конфигурации лога планировщика

```
sudo touch /opt/global/globalserver/application/config/tools/scheduler/logback.xml
```

Вставьте содержимое

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>

    <appender name="STDOUT_DEFAULT" class="ch.qos.logback.core.ConsoleAppender">
        <!-- encoders are assigned the type
        ch.qos.logback.classic.encoder.PatternLayoutEncoder by default -->
        <encoder>
   <pattern>%d{dd.MM.yyyy HH:mm:ss.SSS} | [%-5level] | [%thread] | %logger | %msg%n</
↪pattern>
        </encoder>
    </appender>

    <appender name="FILEOUT_SESSION" class="ch.qos.logback.core.rolling.
↪RollingFileAppender">
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>${LOG_DIR}/jobscheduler.%d{yyyy-MM-dd_HH}.log</
↪fileNamePattern>
            <!--<maxHistory>720</maxHistory>-->
            <!--<totalSizeCap>3GB</totalSizeCap>-->
        </rollingPolicy>
        <encoder>
            <pattern>%d{dd.MM.yyyy HH:mm:ss.SSS} | [%-5level] | [%thread] | %logger |
↪%msg%n</pattern>
        </encoder>
    </appender>

    <!--
        Возможные значения параметра "level" в порядке приоритета важноости
        OFF
        ERROR
        WARN
        INFO
        DEBUG
        TRACE
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        ALL
    -->

    <root level="WARN">
        <appender-ref ref="STDOUT_DEFAULT"/>
        <appender-ref ref="FILEOUT_SESSION"/>
    </root>

</configuration>
```

Конфигурация утилиты обновления

В дистрибутив сервера приложений Global входит утилита обновления, которая обновляет сам сервер приложений или прикладной код образа проектного решения.
Для настройки утилиты обновления скопируйте и переименуйте поставочный конфигурационный файл /opt/global/globalserver/update/config.sh.origin

```
sudo cp /opt/global/globalserver/update/config.sh.origin /opt/global/globalserver/update/
↪config.sh
```

```
#!/bin/sh

#
# Global Postgres update
# параметры утилиты обновления системы
#

#ssh сервера приложений
sSshHost="localhost"
#ssh порт
sSshPort="2299"
#ssh логин
sSshLogin="admin"
#ssh пароль
sSshPass="admin"
#наименование бд
sDbName="<ПсевдонимБД>"
#наименование SBT
sSbtName="main"

# имя сервиса сервера приложений
sGlobalService="global3"

# имя сервиса менеджера заданий
sSchedulerService="globalscheduler"

# временный каталог обновлений
sTmp="/opt/global/globalupdate"
```

Где:

sSshPort - порт внутреннего ssh сервера Global

sSshLogin - логин из `global3.config.xml` в разделе security\users

sSshPass - пароль из `global3.config.xml` в разделе security\users

sDbName - Псевдоним БД из `global3.config.xml` в разделе databases\database:alias

sSbtName - Имя SBT из `global3.config.xml` в разделе databases\database\metaManager:sbtName

sGlobalService - Имя сервиса сервера приложений

sSchedulerService - Имя сервиса менеджера заданий

Выдайте права на запуск

```
sudo find /opt/global/globalserver/update -type f -name '*.sh' -exec chmod a+x {} \;
```

Шифрование паролей в конфигурационных файлах

В конфигурационных файлах системы пароли по умолчанию хранятся в открытом виде. Для шифрования паролей используется генератор шифрованных паролей, интегрированный в сервер приложений. Шифрованные пароли указываются в отдельных тэгах конфигурационных файлов. Пароли могут указываться в следующих файлах:

- global3.config.xml, описан в разделе `Конфигурация Global`
- quartz.properties, описан в разделе `Конфигурация планировщика заданий`
- config.sh, описан в разделе `Конфигурация утилиты обновления`

Общие принципы шифрования

Для шифрования пароля необходимо запустить Global 3 Server с парамерами `-encryptPassword` и `-masterPassword`

```
start.sh -encryptPassword password -masterPassword masterpassword
```

Где:

- `-encryptPassword`
  Пароль который необходимо зашифровать
- `-masterPassword`
  Ключ шифрования, если не указан используется секретный ключ по умолчанию.

Результатом выполнения будет вывод в консоль строки, полученной в результате шифрования пароля переданного параметром `-encryptPassword`.

Использование шифрованных паролей

Шифрованные пароли можно использовать во всех местах файла global3.config.xml где требуется указание пароля пользователя. Вместо тэга `password` зашифрованный пароль необходимо указывать в тэге `encryptedPassword`.

Пример:

```
<databases>
 <database alias="<ПсевдонимБД>" driver="org.postgresql.Driver" schema="PUBLIC"
     url="jdbc:postgresql://<host>:5432/<ИмяБД>" connectionType="proxyShared"␣
↪authenticationType="btk">
  <users>
   <user name="<ИмяПользователяБД>" encryptedPassword="
↪<ЗашифрованныйПарольПользователяБД>"/>
  </users>
 </database>
</databases>
```

Мастер-пароль может быть указан следующими способами:

- Параметром запуска сервера приложений `-masterPassword`

```
start.sh -masterPassword masterpassword
```

- Параметром конфигурационного файла `global3.config.xml` с указанием файла, хранящего мастер пароль.

```
<security>
  <masterPassword file="<ПутьКФайлуСПаролем>">
</security>
```

Шифрование паролей планировщика заданий

В файле quartz.properties зашифрованный пароль может быть указан в отдельном параметре

```
ru.bitec.jobscheduler.quartz.dataSource.quartzDS.encryptedpassword=encryptedPassword
```

Ключ шифрования может быть указан следующими способами:

- Параметром запуска планировщика `-masterPassword`
- Путь до файла с ключом шифрования указывается в конфигурационном файле в параметре

  ```
  ru.bitec.jobscheduler.masterpass.path=/some/path/to/file
  ```

  Примечание: Если в пути до файла используются символ \ (обратный слэш), то его требуется экранировать вторым слэшем. Пример пути:

  ```
  ru.bitec.jobscheduler.masterpass.path=C:\\some\\path\\to\\file
  ```

Для шифрования пароля требуется запустить планировщик заданий с парамерами encryptPassword и masterPassword

```
globalscheduler.sh -encryptPassword password -masterPassword masterpassword
```

Где:

- `-encryptPassword`
  Пароль который необходимо зашифровать

- `-masterPassword`
  Ключ шифрования, если не указан то используется стандартная логика определения мастер-пароля, так же как и при обычном запуске планировщика.

шифрование паролей утилиты обновления

В файле config.sh хранится пароль доступа к ssh сервису сервера приложений в открытом виде и не может быть зашифрован. Если в этом файле не указывать параметры sSshLogin и sSshPass, то они будут запрошены у запускающего пользователя.

Ключи шифрования для токенов

Закрытый ключ

В основном файле конфигурации планировщика `quartz.properties` можно указать путь до файла с закрытым ключом шифрования в параметре

```
ru.bitec.jobscheduler.privatekey.path = /some/path/to/file
```

Примечание: Если в пути до файла используются символ \ (обратный слеш), то его требуется экранировать вторым слешем. Пример пути:

```
ru.bitec.jobscheduler.privatekey.path=C:\\some\\path\\to\\file
```

Этот ключ будет использоваться для подписания токена авторизации для пользователя, который является исполнителем задачи планировщика.

Файл, до которого указан путь, должен содержать в себе только массив байт закрытого ключа, созданного по алгоритму `RSA`, с размером 2048 бит, и закодированный в `Base64`.

Пример содержимого файла с закрытым ключом:

```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAst2dVRFXdMJOSvpz0ryXtGa0mxMiPZiXC/5rCzX7Zl/
↪kUnct16mqGbkqVe0+B04g0p0kSgyE1bnhpNbLxt4Rnywc6EAMKrF6pxaux592zi8QF0CjT2jWfjU+ps9HBo3WH9lRRRMCnNtf+11U
↪KeweQVKDRAJ/mkNXKyHQDPoAuB/
↪0e8tQilzKMk4FspgjJPI0cyn2j88vNVKVIWES5Uq+7ENuzhbwGau78i2nEbw5Eh5MSZFsmGGcjBxtC2FKMDtQVfod+5qjr3YcAWt4l
```

Открытый ключ

Открытый ключ используется для проверки подписанного закрытым ключом токена авторизации. Данный ключ хранится в системе Global, поэтому его задание необходимо производить на этапе `Администрирование системы`.

Первичная инициализация БД

Перед началом работы или установки поставочного дампа пустую базу данных системы требуется инициализировать. При инициализации БД будут созданы все необходимые схемы, таблицы, функции, а так же будут установленны первоначальные данные в таблицах.

Для инициализации выполните следующие действия:

Подключитесь терминалом ssh к серверу приложений. По умолчанию порт 2299, логин admin, пароль admin

```
ssh admin@localhost -p 2299
```

Где `admin` - это имя пользователя из конфигурационного файла global3.config.xml, `localhost` - адрес сервера Global

Подключитесь к системе в режиме `sys`

```
attach db Global as sys
```

Где `Global` - это псевдоним БД из конфигурационного файла global3.config.xml

Выполните команду инициализации схемы

```
init schema
```

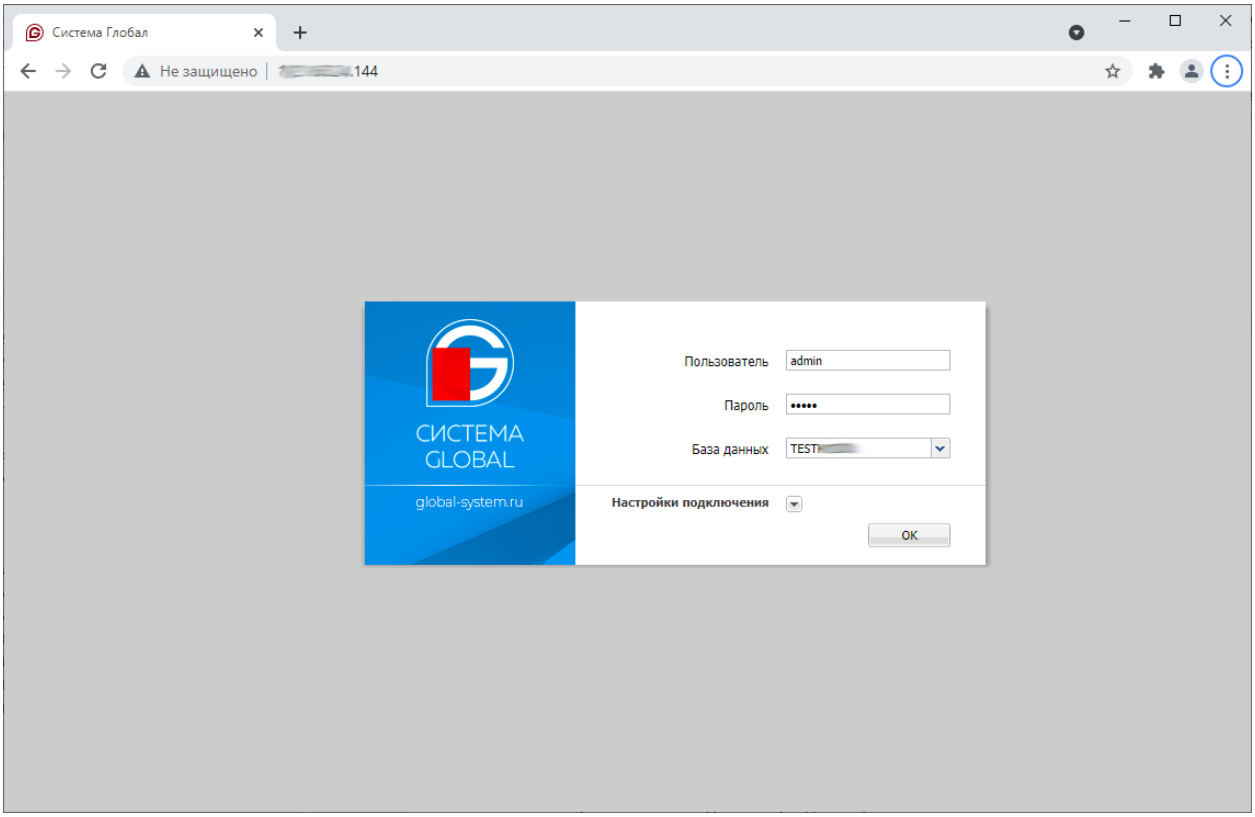
Выполните команду инициализации первоначальных данных

```
init data
```

Получение и установка лицензии

Лицензирование системы привязано к базе данных. Уникальный код базы формируется с учетом аппаратно-программного окружения СУБД и привязано к конкретной БД. Любые изменения программно-аппаратного окружения (Тип процессора, материнская плата, мак-адрес сетевой карты, версия СУБД, имя базы данных и т.д.) приведут к потере лицензии.
Лицензия устанавливается при первой авторизации в системе. Позже лицензия может быть изменена или дополнена. Для регистрации необходимо зайти в систему Global с помощью браузера

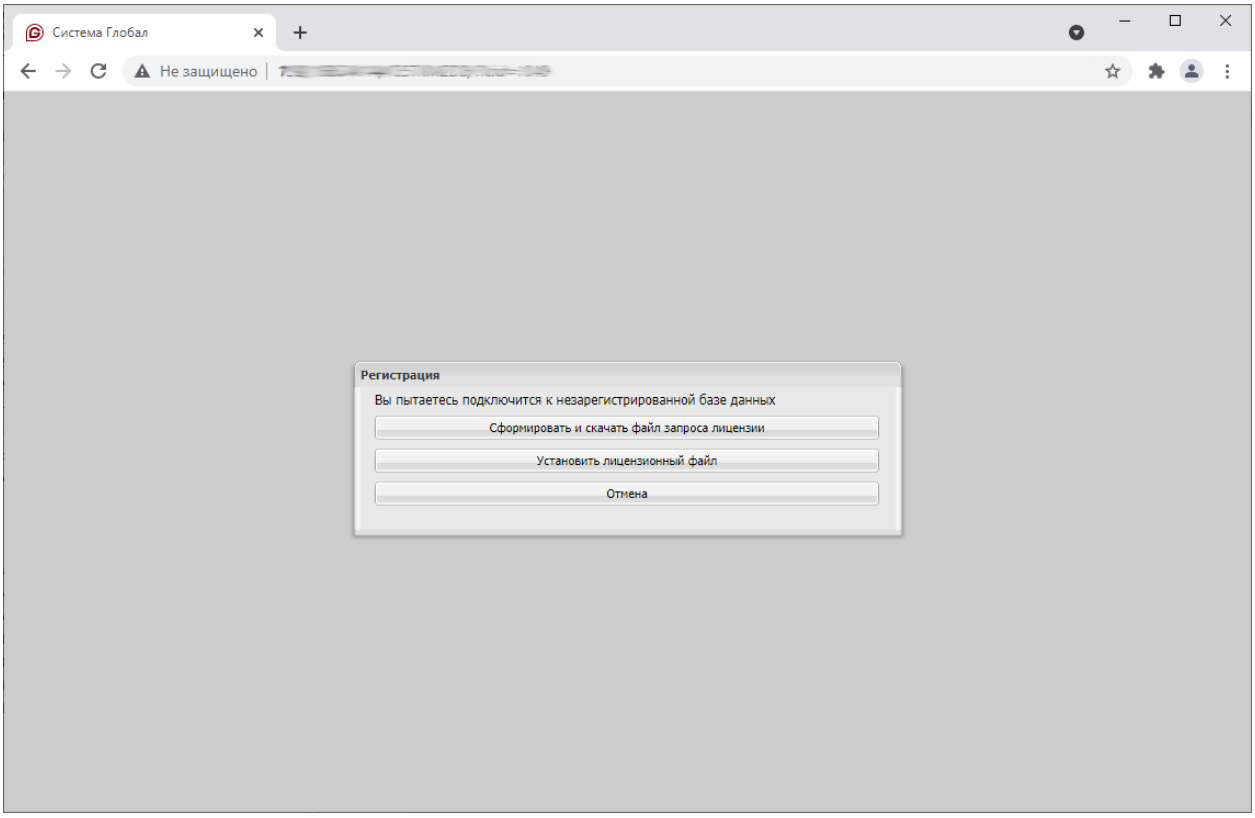
```
http://Адрес_хоста_системы_Global/
```



Для входа используются учетные данные:
Логин: admin
Пароль: admin

Примечание: Если используется поставочный дамп, контактное лицо передает логин и пароль для входа в систему.

После входа будет предложено зарегистрировать базу данных:



Система предложит сформировать файл запроса лицензии.
Полученный файл запроса нужно отправить контактному лицу в ООО «Бизнес-Технологии» и получить от него файл с лицензией.

4.2 Администрирование системы

Обзор компонентов администрирования

Конфигурационные файл

global3.config.xml

Основной конфигурационный файл сервера приложения. Данная конфигурация задается до старта сервера. Основные параметры:

- База данных
- Квоты
- Параметры доступа к ssh консоли сервера

Ssh консоль сервера приложения

Основные задачи консоли, ручное управление миграцией данных:

- синхронизация схемы базы данных
- добавление лицензии
- список сессий
- отключение сессий

Пример подключения к консоли:

```
ssh admin@127.0.0.1 -p 2299
```

Совет: Документацию по командам смотрите командой `help` в консоли

Настройка системы

После старта системы и разворачивания схемы базы данных доступен веб интерфейс из которого можно сконфигурировать административные параметры системы.

- `Настройка системы > Сущности > Настройка файлового хранилища`
  Позволяет задать правила хранения файлов в системе.

Запуск и остановка сервера Global

Команда запуска сервера

```
systemctl start global3
```

Перезапуск сервера

```
systemctl restart global3
```

Остановка сервера

```
systemctl stop global3
```

Запуск и остановка планировщика заданий

Команда запуска

```
systemctl start globalscheduler
```

Перезапуск

```
systemctl restart globalscheduler
```

Остановка

```
systemctl stop globalscheduler
```

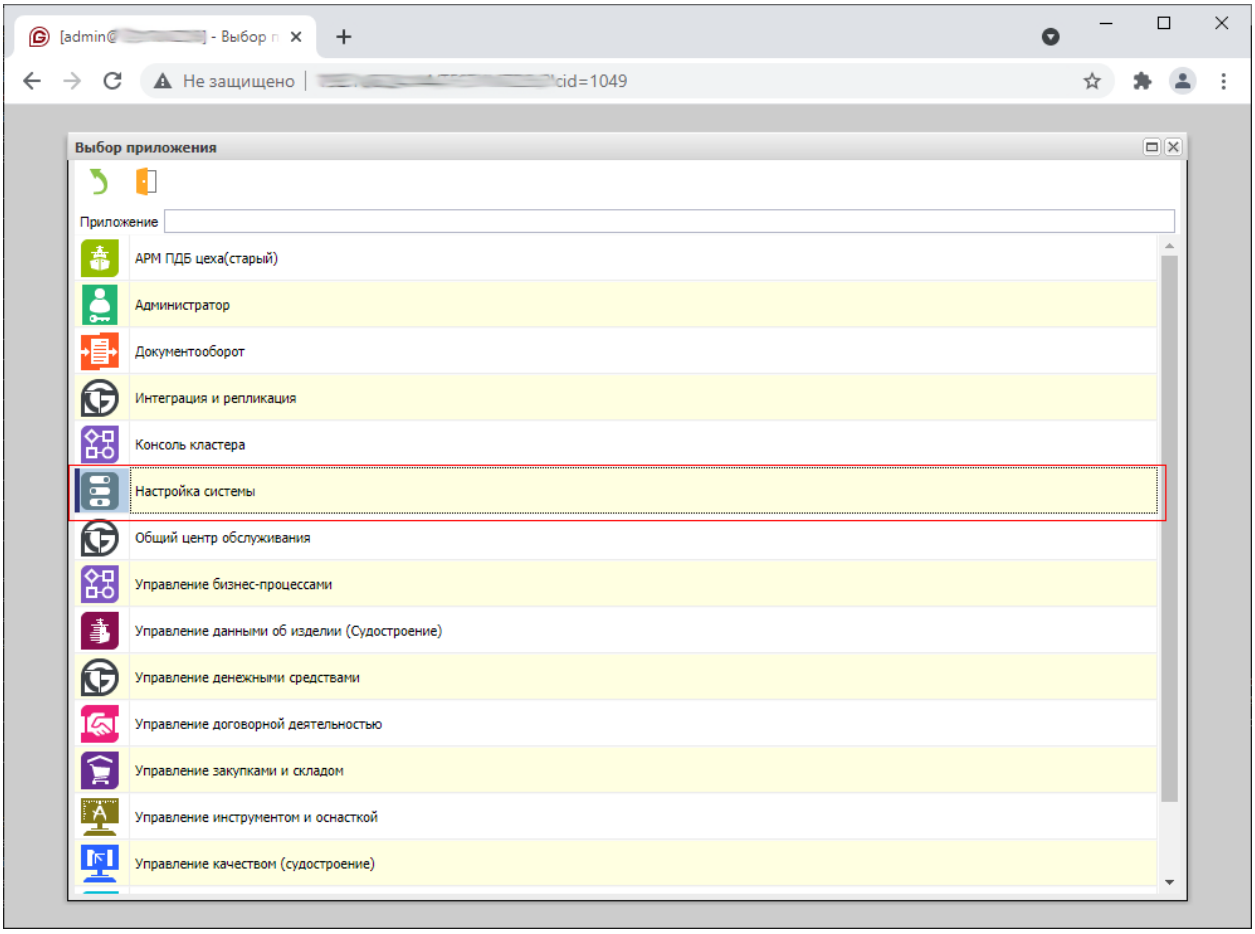
Настройка файлового хранилища

Система версионного хранения файлов может работать в двух режимах:

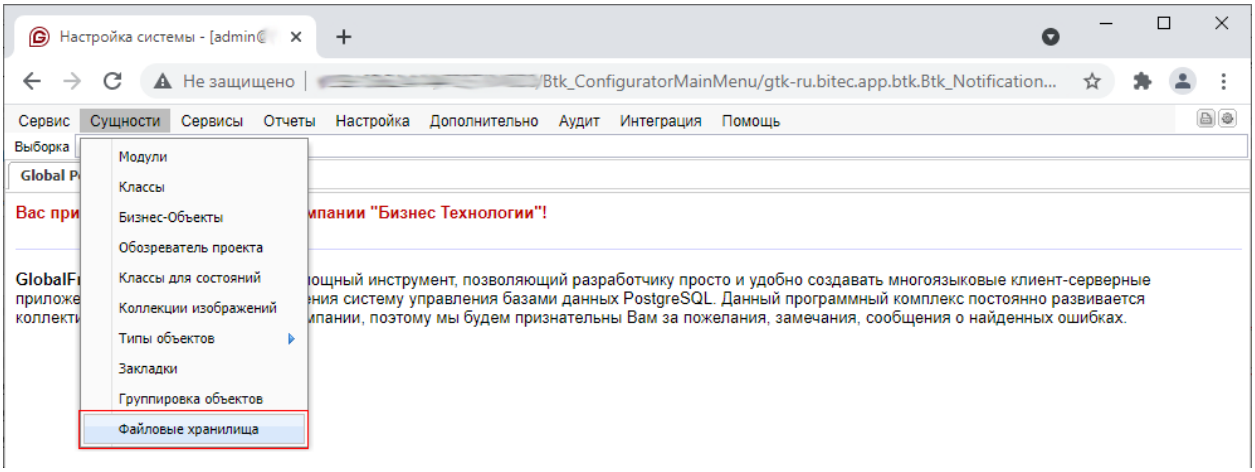
- SMB/ CIFS - медленный режим. Подключается к сетевому ресурсу каждый раз, когда требуется доступ файлу.
- Локальное хранилище – быстрый режим. Работает с локальной директорией на сервере.

Рекомендуется обеспечить инкрементное резервное копирование директории или сетевого ресурса, который будет использоваться для файлового хранилища Global System.
Количество файловых хранилищ в системе Global не ограничено. Обычно хранилища создаются на каждую функциональную подсистему (Например: система документооборота, система прикрепленных файлов, система интеграции и репликации и т.д.)

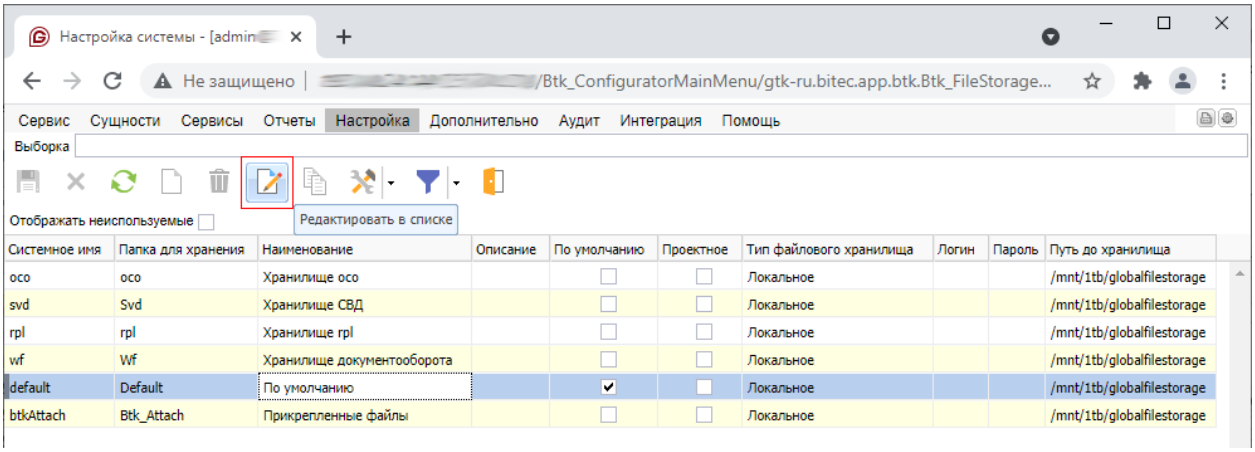
Для настройки файловых хранилищ требуется открыть приложение «Настройка системы»



Открыть меню: Сущности | Файловые хранилища



В списке разблокировать редактирование

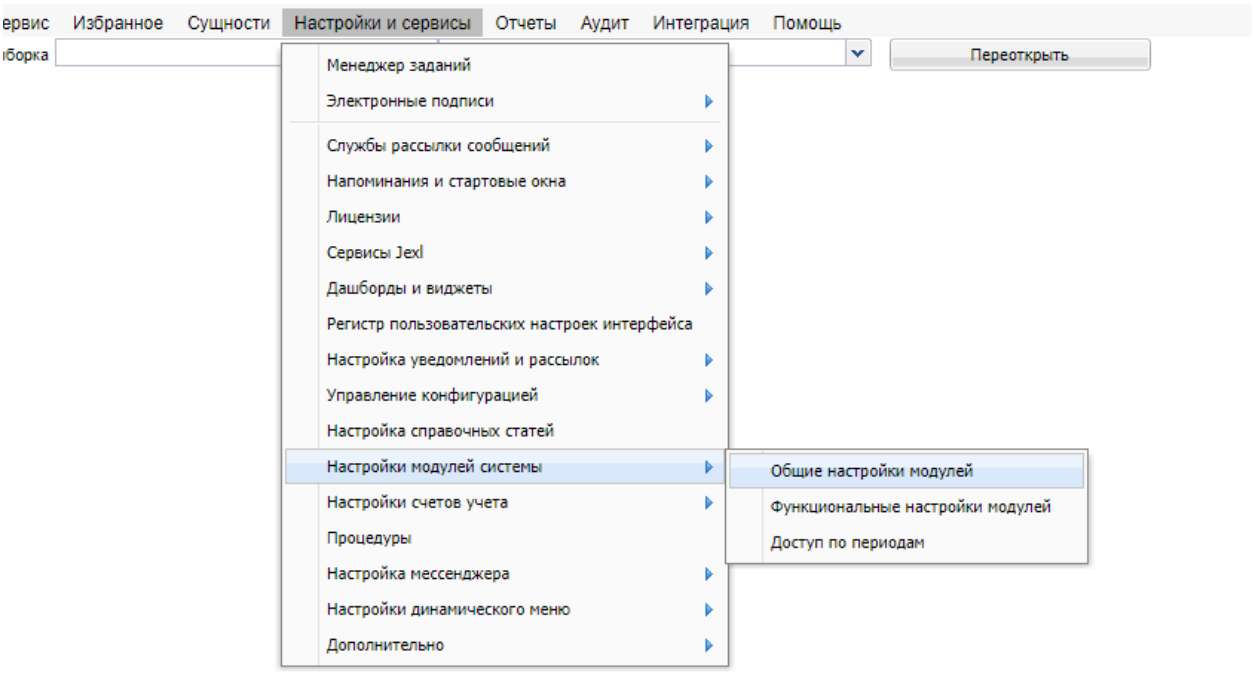


Указать для всех файловых хранилищ тип «локальное» и «Путь до хранилища» (Например: /mnt/1tb/globalfilestorage/)

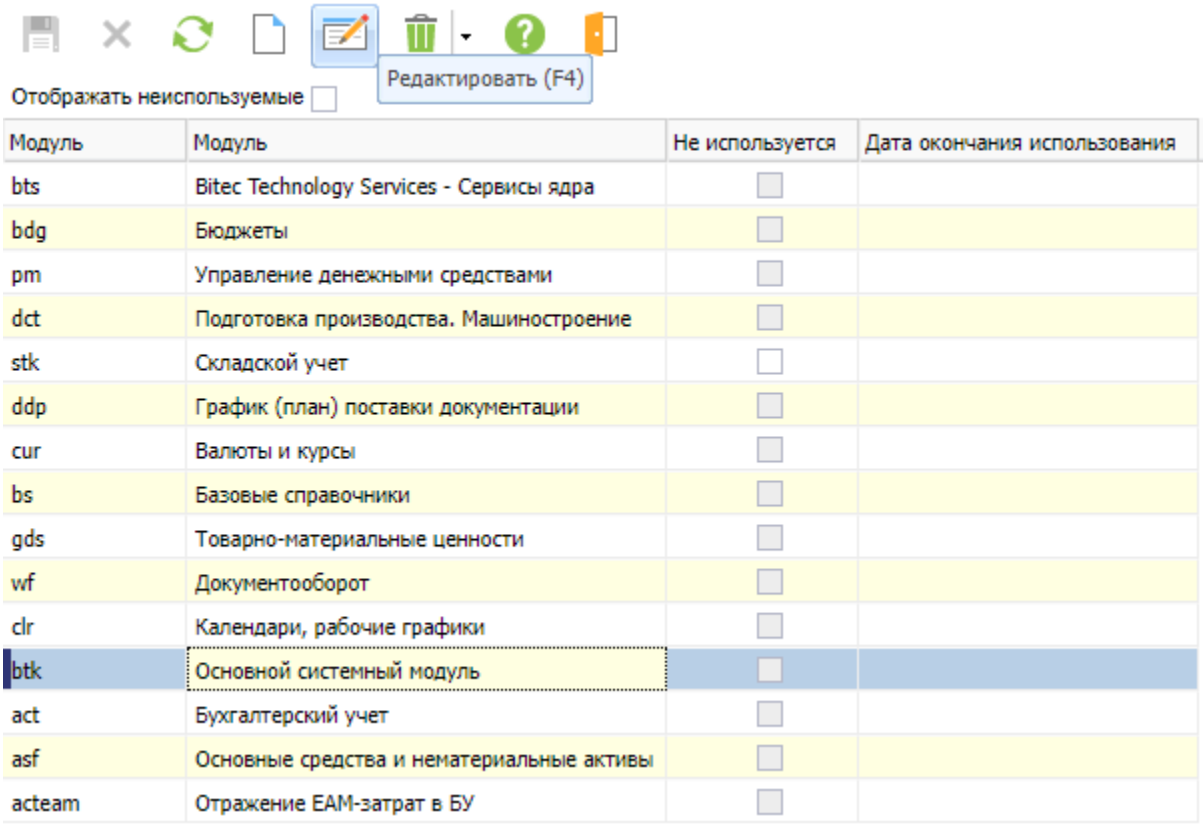
Настройка ключей шифрования

Для указания открытого ключа для проверки токена авторизации планировщика заданий необходимо открыть приложение «Настройка системы».

Открыть меню: Настройки и сервисы | Настройки модулей системы | Общие настройки модулей

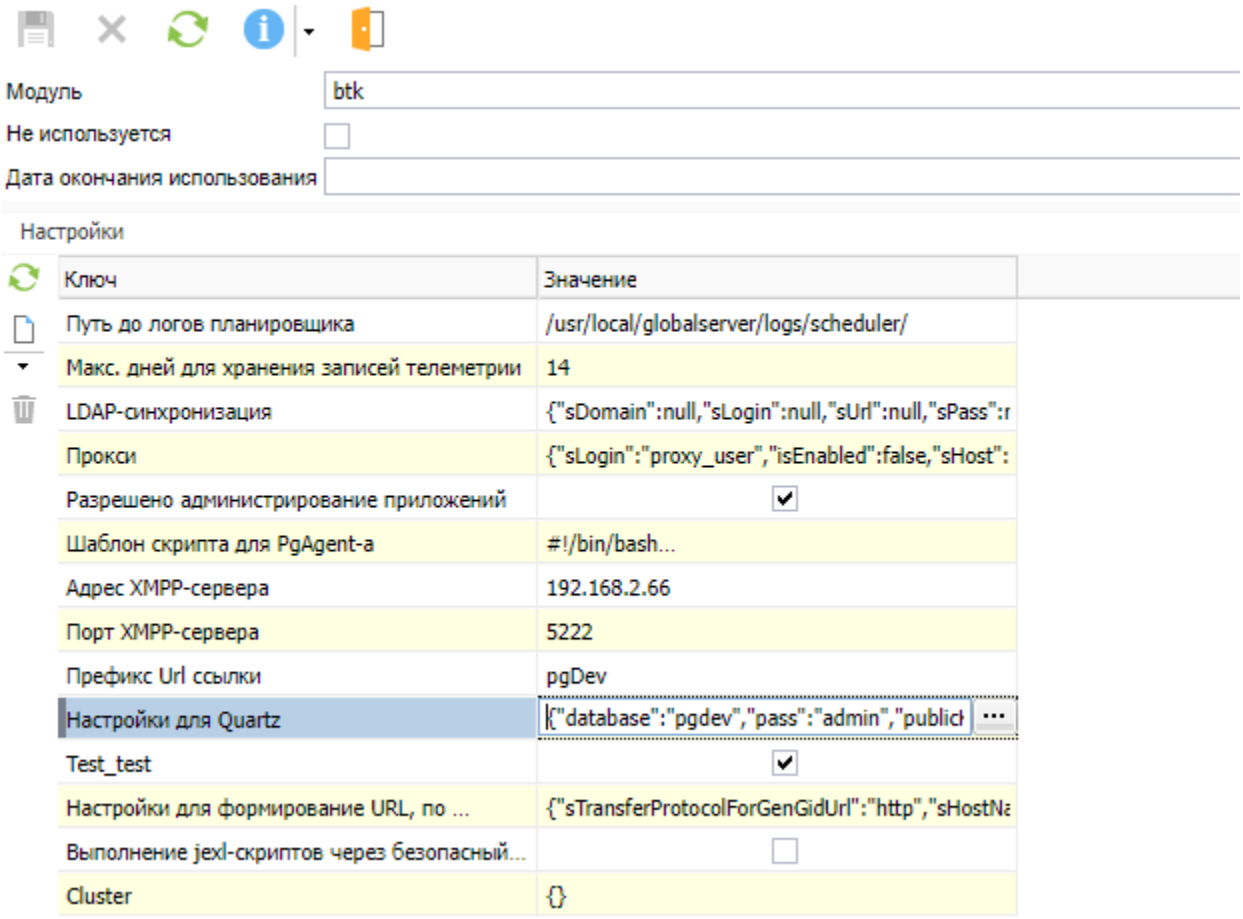


В списке выбрать модуль btk и открыть на редактирование



Модуль	Модуль	Не используется	Дата окончания использования
bts	Bitec Technology Services - Сервисы ядра	☐	
bdg	Бюджеты	☐	
pm	Управление денежными средствами	☐	
dct	Подготовка производства. Машиностроение	☐	
stk	Складской учет	☐	
ddp	График (план) поставки документации	☐	
cur	Валюты и курсы	☐	
bs	Базовые справочники	☐	
gds	Товарно-материальные ценности	☐	
wf	Документооборот	☐	
clr	Календари, рабочие графики	☐	
btk	Основной системный модуль	☐	
act	Бухгалтерский учет	☐	
asf	Основные средства и нематериальные активы	☐	
acteam	Отражение EAM-затрат в БУ	☐	

В открывшемся окне выбрать «Настройки для Quartz» и нажать на кнопку с тремя точками

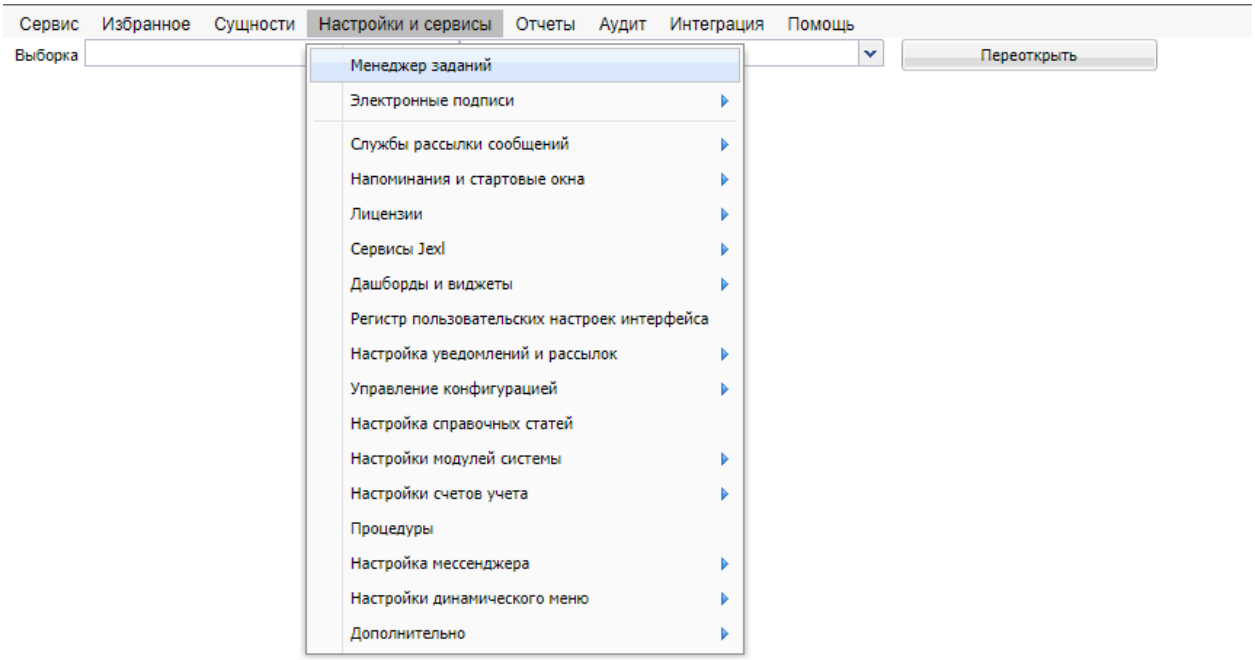


В открывшемся выбрать «Открытый ключ» и вставить значение, после чего нажать зеленую стрелку, а затем операцию «Сохранить» в окне с настройками модуля btk.

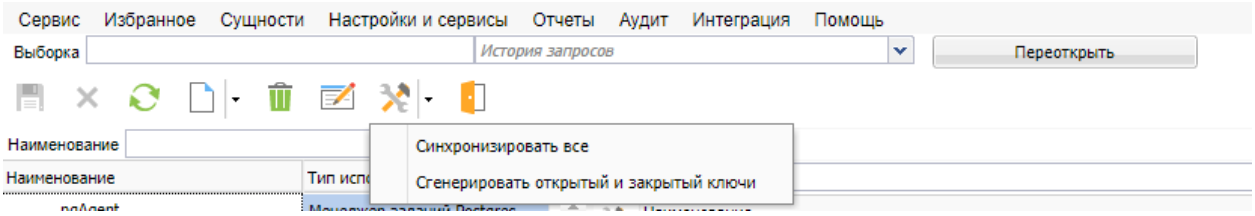
Генерация ключей шифрования

Генерацию ключей можно выполнить с помощью системы Global.

Для этого в приложении «Настройки системы» необходимо открыть меню Настройки и сервисы | Менеджер заданий



В открывшемся окне выполнить операцию «Сгенерировать открытый и закрытый ключ»



В результате выполнения операции на компьютер пользователя будут скачены два файла: один - с открытым ключом, а другой - с закрытым.

Генерация ключей сторонними утилитами

Также, генерация ключей допускается сторонними утилитами.

Внимание: Открытый ключ, как и закрытый, должен содержать в себе только массив байт, созданный по алгоритму `RSA`, с размером 2048 бит, и закодированный в `Base64`.

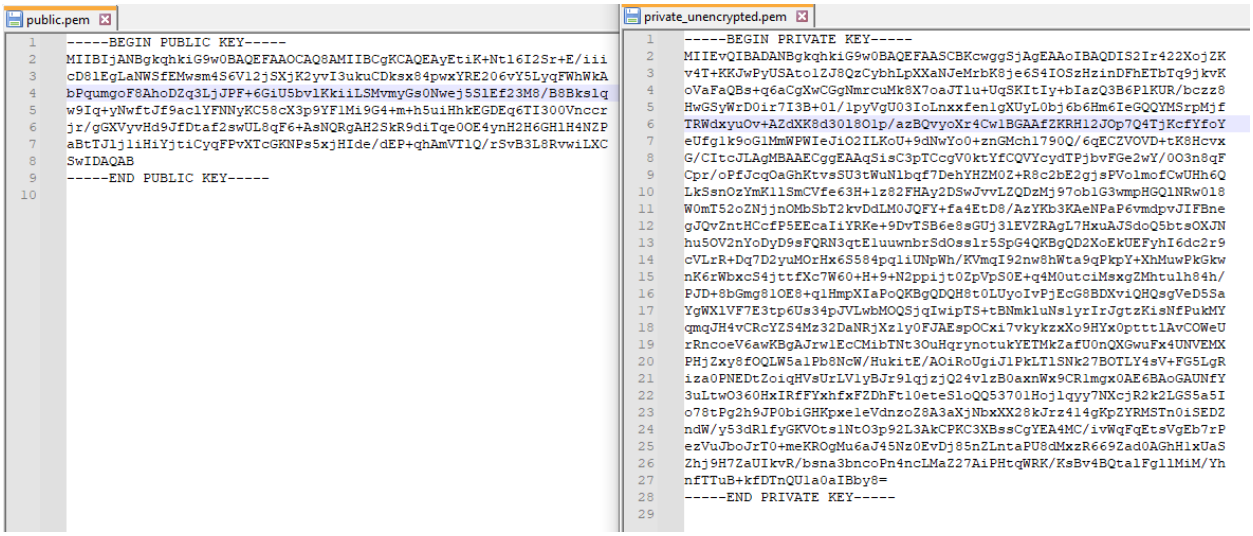
Пример создания ключей с помощью криптографической библиотеки OpenSSL:

```
openssl genrsa -des3 -out private.pem 2048

openssl rsa -in private.pem -outform PEM -pubout -out public.pem

openssl rsa -in private.pem -out private_unencrypted.pem -outform PEM
```

После выполнения данных комманд появится два файла с ключами, которые будут следующего вида:



Для корректной работы этих ключей в нашей системе необходимо убрать записи типа: `-----BEGIN PUBLIC KEY-----`, `-----END PRIVATE KEY-----` и т.д., а также убрать все переносы строк. После этого ключи будут готовы для работы в системе Global.

Обновление системы

Для обновления системы используется специальная утилита `/usr/local/globalserver/update/update.sh`
По умолчанию утилита проверяет наличие архивов с обновлениями по адресу /opt/global/globalupdate. Перед запуском утилиты администратор должен поместить в этот каталог полученные архивы с обновлениями. Архивы обновлений бывают двух типов:

- globalserver.zip – дистрибутив сервера приложений
- applib.zip – образ прикладного решения

Режимы работы утилиты обновления

Предусмотрены следующие режимы обновления:

- jarOnly - обновление только jar файлов прикладного решения (без рестарта), используется по умолчанию
- dbGen - обновление jar файлов прикладного решения с запуском генератора схемы Сервер переводится в сервисный режим, у всех пользователей автоматически выполняется выход из системы. Вход в систему разрешается после установки обновления.
- dbGenAc – тоже самое что и dbGen плюс синхронизация прав доступа и объектных привилегий. Полная синхронизация прав доступа может занимать продолжительное время, поэтому не рекомендуется запускать этот режим в рабочее время.
- server - обновление сервера приложений (без обновления прикладного решения). Сервер останавливается, выполняется обновление исполняемых файлов и ресурсов. После обновления сервер автоматически запускается.
- full - режим полного обновления, при котором сначала обновляется сервер, а потом образ прикладного решения.

Режим обновления передается в утилиту параметром

```
/usr/local/globalserver/update/update.sh -m DbGen
```

4.3 Управление схемой базы данных

Корзина удаленных объектов схемы

Корзина удаленных объектов схемы показывает те объекты схемы, которые были когда-либо сформированы генератором схемы, но затем были удаленны из кода приложения.

Открывается она из «Настройка системы» -> «Сущности» -> «Обозреватель проекта» -> Под операцией с шестерней -> «Корзина удаленных объектов схемы».

Использование

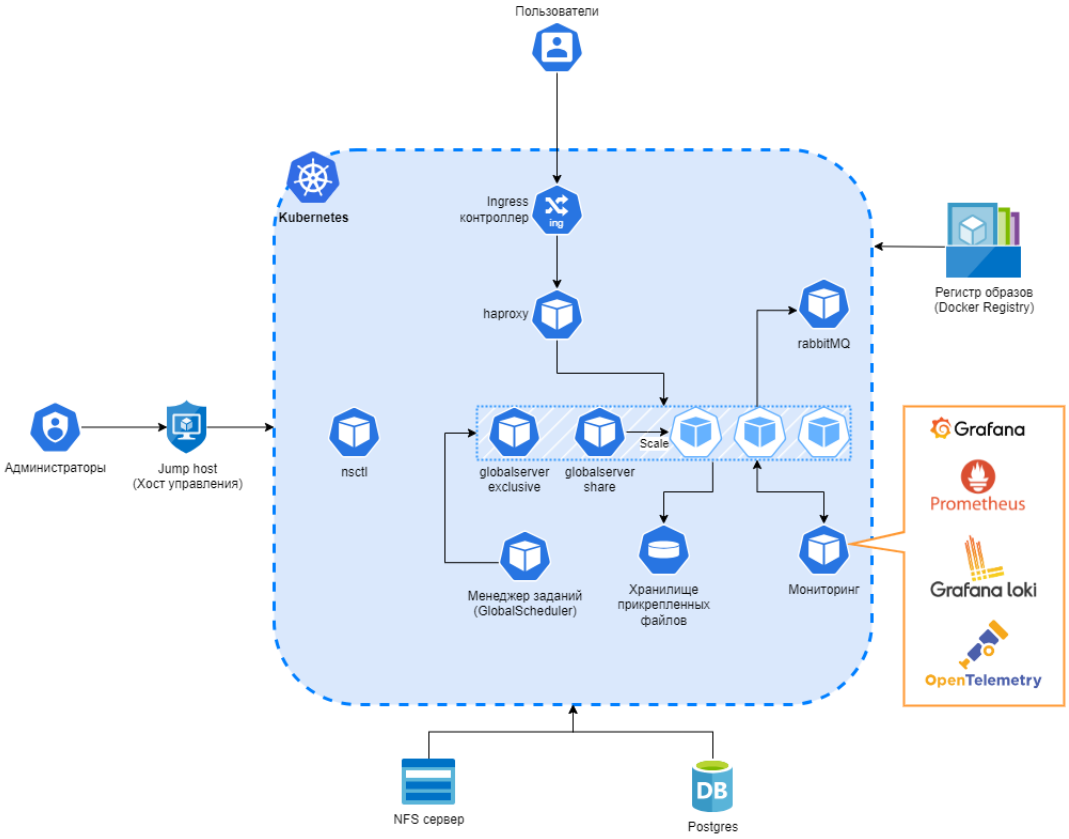
Выборка позволяет искать удаленные объекты, которые остались в базе данных. В ней есть две операции:

1. «Удалить» - Удаляет объект из схемы базы данных
2. «Удалить без удаление объекта БД» - удаляет запись об объекте в классе соответствующем типу объекта, но оставляет объект в схеме

5 GlobalServer кластер в kubernetes

5.1 Описание

Кластер GlobalServer может работать в режиме высокой доступности и легко масштабироваться горизонтально под высокие нагрузки, используя среду Kubernetes. Запускается в облачных средах, таких как VK Cloud или любых других, имеющих поддержку Kubernetes.



Структура кластера Global ERP

Кластер состоит из следующих элементов

- Кластер kubernetes
- Комплект группы `groupkit`
  - внешние jar библиотеки (если требуются на проекте)
  - системное хранилище сертификатов для java (cacerts)
    требуется для добавления корневых сертификатов заказчика. Используется для SSL соединений.
- Комплект приложения `appkit`
  - Дистрибутив сервера приложений `globalserver.zip`
  - Дистрибутив прикладного решения `applib.zip`
  - Пакет конфигурационных файлов для элементов кластера (`globalserver`, `globalscheduler`, `haproxy`)
- NFS сервер
  Используется для хранения комплектов группы и комплектов приложений
- Jump хост для администрирования с утилитой `nscli`
- Сервер СУБД Postgres

- Регистр образов docker или доступ к https://dockerhub.global-system.ru/
  Если кластер разворачивается в закрытой среде, то потребуется развернуть персональный регистр и загрузить в него необходимые образы.

5.2 Установка

Нам потребуется виртуальная машина для административных задач - jump host. С нее будем выполнять все операции по администрированию кластера. Пользователь, под которым будет проводится настройка должен иметь привилегии администратора и разрешения для запуска sudo.

Настройка узла администрирования (jump host)

Установка необходимых пакетов

На jump хост необходимо установить следующие пакеты:

```
sudo apt-get install mc htop
sudo apt-get install -y kubectl
sudo apt-mark hold kubectl
sudo apt install nfs-common
```

Совет: Версия kubectl должна соответствовать версии в платформе kubernetes или быть новее

Настройка работы с docker образами

Если требуется работа с образами из docker регистра

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
↪compose-plugin
sudo usermod -aG docker $USER
```

Insecure Docker Registry

Если кластер запускается в закрытой сети, docker регистр может не использовать SSL (Insecure Docker Registry). Запуск Insecure Docker Registry для хранения своих docker-образов не самый лучший вариант с точки зрения безопасности, но порой это самое простое и разумное решение в закрытых сетях.

Для настройки нужно изменить (или создать, если такового нет) конфигурационный файл /etc/docker/daemon.json, добавив в него следующие строки:

```
{
    "insecure-registries" : ["myregistry.example.local:1234"]
}
```

, где myregistry.example.local:1234 - адрес и порт локального докер регистра

скрипт для создания файла:

```
cat <<EOF | sudo tee /etc/docker/daemon.json
{
    "insecure-registries" : ["myregistry.example.local:1234"]
}
EOF
```

После чего выполнить перезапуск сервиса с помощью:

```
sudo systemctl restart docker
```

Настройка авторизации kubernetes кластера

Получите конфигурационный файл авторизации для кластера Kubernetes, который создается при разворачивании kubernetes

Пример файла:

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:␣
↪LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJUkMwZFZrYXhBTE13RFFZSktvWklodmNOQVFFT
    server: https://127.0.0.1:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data:␣
↪LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURLVENDQWhHZ0F3SUJBZ0lJR04ydFhtNkEzc2N3RFFZSktvWklodmNOQVFFT
    client-key-data:␣
↪LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBclA1TXlQWlo3YlpoRmo1SldtSGkzT29MSXpWd
```

Настройте авторизацию kubernetes

```
# создаем каталог с конфигурацией
mkdir ~/.kube && \
chmod -R 0700 ~/.kube && \
cd ~/.kube

# сохраняем файл
cat <<EOF | tee ~/.kube/config
apiVersion: v1
apiVersion: v1
clusters:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
- cluster:
    certificate-authority-data:␣
↪LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJUkMwZFZrYXhBTE13RFFZSktvWklodmNOQVFFT
    server: https://127.0.0.1:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data:␣
↪LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURLVENDQWhHZ0F3SUJBZ0lJR04ydFhtNkEzc2N3RFFZSktvWklodmNOQVFFT
    client-key-data:␣
↪LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBclA1TXlQWlo3YlpoRmo1SldtSGkzT29MSXpWd
EOF
```

Проверяем доступ

```
kubectl cluster-info
kubectl get nodes -o wide
```

При успешном подключении должны получить вывод:

```
Kubernetes control plane is running at https://0.0.0.0:6443
CoreDNS is running at https://0.0.0.0:6443/api/v1/namespaces/kube-system/services/kube-
↪dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
NAME           STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-
↪IMAGE                         KERNEL-VERSION    CONTAINER-RUNTIME
k8s-master01   Ready    control-plane   46d   v1.29.1   0.0.0.0         <none>        ␣
↪Debian GNU/Linux 11 (bullseye)   5.10.0-27-amd64   containerd://1.6.27
k8s-worker01   Ready    <none>          46d   v1.29.1   0.0.0.0         <none>        ␣
↪Debian GNU/Linux 11 (bullseye)   5.10.0-27-amd64   containerd://1.6.27
k8s-worker02   Ready    <none>          46d   v1.29.1   0.0.0.0         <none>        ␣
↪Debian GNU/Linux 11 (bullseye)   5.10.0-27-amd64   containerd://1.6.27
k8s-worker03   Ready    <none>          46d   v1.29.1   0.0.0.0         <none>        ␣
↪Debian GNU/Linux 11 (bullseye)   5.10.0-27-amd64   containerd://1.6.27
```

Установка утилиты Nscli

`Nscli` автоматизирует работу по развертыванию кластера Global.

Установка:

```
#Скачиваем из репозитория
cd ~
wget --backups=1 --user=<пользователь> --ask-password "https://repo.global-system.ru/
↪artifactory/general/ru/bitec/gs-ctk-nscli/SNAPSHOT/gs-ctk-nscli-SNAPSHOT.zip"
unzip -o gs-ctk-nscli-SNAPSHOT.zip -d nscli
```

где <пользователь> - учетная запись, полученная через контактное лицо технической поддержки.

Совет: В закрытой среде требуется установить все пакеты, указанные в скрипте ~/nscli/bin/installpkg.sh вручную

Перейдите в каталог с утилитой и выполняем первичную установку

```
cd ~/nscli
# выполняем скрипты установки
./bin/install.sh
```

Jump хост готов к работе.

Настройка рабочего пространства в kubernetes

Для настройки подключаемся по ssh к jump хосту

Перейдите в каталог с утилитой

```
cd ~/nscli
```

Запустите мастер создания манифеста рабочего пространства

```
./namespace.sh create_install_scripts
```

В режиме диалога введите параметры рабочего пространства:

- Имя рабочего пространства - имя воркспейса, который будет создан в kubernetes
- Адрес докер регистра - адрес ресурса с хранилищем образов (Публичный докер регистр Global: dockerhub.global-system.ru)
- Имя файла для хранения мастер ключа - полный путь с именем файла, в котором будет храниться мастер ключ для шифрования паролей
- Пользователь для авторизации докера - имя пользователя для авторизации, при анонимном доступе поле можно оставить пустым.
- Пароль для авторизации докера - пароль для авторизации в регистре (вводится два раза)
- Тип репозитория - тип репозитория, по умолчанию nfs
- Имя сервера nfs - указываем ip адрес или доменное имя заранее настроенного NFS сервеа
- Путь - путь для подключения тома

Пример работы мастера создания манифеста:

```
k8sadmin@k8s-terminal02:~/nscli$ ./namespace.sh create_install_scripts
Введите имя рабочего пространства:gs-cluster-k8s
Введите адрес докер регистра:dockerhub.global-system.ru
Для безопасного хранения паролей необходимо сгенерировать приватный ключ.
Укажите имя файла для хранения мастер ключа:/home/k8sadmin/.gs-ctk.priv
Введите пользователя для авторизации докера:userk8s
Введите пароль для авторизации докера:**********
Введите пароль для авторизации докера:**********
Выберите тип репозитория:nfs
Необходимо задать параметры для Network File System
Введите имя сервера:0.0.0.0
Введите путь:/mnt/nfs/gs-cluster-k8s
k8sadmin@k8s-terminal02:~/nscli$
```

Разрешите запуск скрипта установки рабочего пространства

```
chmod +x ~/nscli/workspace/install_scripts/gs-cluster-k8s/install.sh
```

Создайте рабочее пространство

```
~/nscli/workspace/install_scripts/gs-cluster-k8s/install.sh
```

После выполнения скрипта будет создано рабочее пространство и будет запущен под управления кластером `nsctl`

```
namespace/gs-cluster-k8s created
secret/docker-registry-secret created
configmap/values created
role.rbac.authorization.k8s.io/worker created
rolebinding.rbac.authorization.k8s.io/worker-pods created
deployment.apps/nsctl created
```

Подготовка комплекта приложений appkit

Комплект приложений определяет перечень артефактов, необходимых для разворачивания кластера системы Global ERP:

- Дистрибутив сервера приложений `globalserver.zip`
- Образ прикладного решения `applib.zip`
- Конфигурация сервера приложений
- Конфигурация менеджера заданий
- Конфигурация балансировщика haproxy

Для создания комплекта приложений используйте каталог `~/nscli/workspace/appkit/v1`

```
mkdir -p ~/nscli/workspace/appkit/v1/
```

Подготовте и загрузите в каталог дистрибутивы и конфигурационные файлы.

Подготовте комплект приложений для работы

```
./appkit.sh push --namespace gs-cluster-k8s --source workspace/appkit/v1 --destination␣
↪appkits/v1
```

Утилита автоматически упакует комплект приложений и создаст контрольные суммы

```
Загружен файл:globalserver.zip
Загружен файл:profile.zip
Загружен файл:applib.zip
```

Работа с постоянными томами

Данные в кластере Kubernetes могут храниться несколькими способами: непосредственно в контейнере или на томах (volumes). При хранении данных в контейнере возникают проблемы:

- При сбое или остановке контейнера данные теряются.
- Данные контейнера недоступны для других контейнеров, даже если все контейнеры находятся в одном поде.

Чтобы решить эти проблемы, используются тома Kubernetes. Тома имеют разный жизненный цикл в зависимости от сценария использования:

- У временных томов (ephemeral volume, EV) жизненный цикл совпадает с жизненным циклом пода. Когда под, использующий такой том, прекращает свое существование, том тоже удаляется. Временные тома могут использоваться только одним подом, поэтому объявление томов происходит непосредственно в манифесте пода.
- У постоянных томов (persistent volume, PV) свой жизненный цикл, не зависящий от жизненного цикла пода. Благодаря разделению жизненных циклов такие тома можно переиспользовать позднее с другими подами. Для работы с постоянными томами поды и другие рабочие нагрузки используют Persistent Volume Claim (PVC).

Кластеру Глобал ERP потребуется постоянный том для хранения метрик.

Kubernetes поддерживает разные типы хранилищ, ниже представлен пример хранилища на основе локального каталога.

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-pv-50
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-storage
  local:
    path: /mnt/disks/disk1
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        - key: kubernetes.io/hostname
          operator: In
          values:
          - k8s-worker01
```

```
cat <<EOF | tee ~/nscli/workspace/local-pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-pv-50
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-storage
  local:
    path: /mnt/disks/disk1
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - k8s-worker01
EOF
```

```
kubectl apply -f ~/nscli/workspace/local-pv.yaml
```

Создание секретов

Секреты используются для безопасного хранения учетных данных

- Создайте секрет для доступа к статистике haproxy

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-haproxy-auth
  namespace: gs-cluster-k8s
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: t0p-Secret
```

```
cat <<EOF | tee ~/nscli/workspace/haproxy-sercret.yaml
apiVersion: v1
```

(continues on next page)

(продолжение с предыдущей страницы)

```
kind: Secret
metadata:
  name: secret-haproxy-auth
  namespace: gs-cluster-k8s
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: tOp-Secret
EOF
```

```
kubectl apply -f ~/nscli/workspace/haproxy-sercret.yaml
```

- Создайте секрет с admin аккаунтом globalserver-a

```
apiVersion: v1
kind: Secret
metadata:
  name: gs-admin-auth
  namespace: gs-cluster-k8s
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: Secret#123#Pass!
```

```
cat <<EOF | tee ~/nscli/workspace/admin-auth-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: gs-admin-auth
  namespace: gs-cluster-k8s
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: Secret#123#Pass!
EOF
```

```
kubectl apply -f ~/nscli/workspace/admin-auth-secret.yaml
```

- Создайте секрет с аккаунтом пользователя БД, заменив значения шаблона <username> и <password> на корректные

```
apiVersion: v1
kind: Secret
metadata:
  name: db-user-secret
  namespace: gs-cluster-k8s
type: kubernetes.io/basic-auth
stringData:
  username: <username>
  password: <password>
```

```
cat <<EOF | tee ~/nscli/workspace/db-user-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: db-user-secret
  namespace: gs-cluster-k8s
type: kubernetes.io/basic-auth
stringData:
  username: <username>
  password: <password>
EOF
```

```
kubectl apply -f ~/nscli/workspace/db-user-secret.yaml
```

- Создайте секрет с токеном планировщика, заменив значение шаблона <key> на корректное

```
apiVersion: v1
kind: Secret
metadata:
  name: scheduler-token-secret
  namespace: gs-cluster-k8s
data:
  private.key: <key>
```

```
cat <<EOF | tee ~/nscli/workspace/scheduler-token-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: scheduler-token-secret
  namespace: gs-cluster-k8s
data:
  private.key: <key>
EOF
```

```
kubectl apply -f ~/nscli/workspace/scheduler-token-secret.yaml
```

Сервис аккаунт для получения метрик cAdvisor

- Создайте сервис аккаунт

```
apiVersion: v1
kind: ServiceAccount
metadata:
name: prometheus-cadvisor
namespace: gs-cluster-k8s
```

```
cat <<EOF | tee ~/nscli/workspace/cadvisor-sa.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
name: prometheus-cadvisor
```

(continues on next page)

(продолжение с предыдущей страницы)

```
namespace: gs-cluster-k8s
EOF
```

```
kubectl apply -f ~/nscli/workspace/cadvisor-sa.yaml
```

- Создайте роль

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus-cadvisor
rules:
- apiGroups: [""]
  resources:
  - nodes
  - nodes/proxy
  - services
  - endpoints
  - pods
  verbs: ["get", "list", "watch"]
- apiGroups:
  - extensions
  resources:
  - ingresses
  verbs: ["get", "list", "watch"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]
```

```
cat <<EOF | tee ~/nscli/workspace/cadvisor-role.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus-cadvisor
rules:
- apiGroups: [""]
  resources:
  - nodes
  - nodes/proxy
  - services
  - endpoints
  - pods
  verbs: ["get", "list", "watch"]
- apiGroups:
  - extensions
  resources:
  - ingresses
  verbs: ["get", "list", "watch"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]
EOF
```

```
kubectl apply -f ~/nscli/workspace/cadvisor-role.yaml
```

- Создайте биндинг роли

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: prometheus-cadvisor
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus-cadvisor
subjects:
- kind: ServiceAccount
  name: prometheus-cadvisor
  namespace: gs-cluster-k8s
```

```
cat <<EOF | tee ~/nscli/workspace/cadvisor-role-binding.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: prometheus-cadvisor
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus-cadvisor
subjects:
- kind: ServiceAccount
  name: prometheus-cadvisor
  namespace: gs-cluster-k8s
EOF
```

```
kubectl apply -f ~/nscli/workspace/cadvisor-role-binding.yaml
```

Настройка параметров кластера

Параметры кластера настраиваются с помощью пода `nsctl`

Для настройки нужно подключится к поду `nsctl`, который был запущен при создании рабочего пространства.

Получите список подов рабочего простнанства

```
kubectl get pods --namespace gs-cluster-k8s
```

```
NAME                      READY   STATUS    RESTARTS   AGE
nsctl-7f97cb6df4-8kjkc    1/1     Running   0          57m
```

Получите доступ по ssh в под `nsctl`

```
kubectl -n gs-cluster-k8s exec -it nsctl-7f97cb6df4-8kjkc -- /bin/bash
```

Выполните первоначальную настройку

```
# создаем группу
./resgroup.sh create --name gs-cluster-1
# указываем актуальный appkit
./resgroup.sh switch_appkit --name gs-cluster-1 --path appkits/v1
# создаем эксклюзивный экземпляр
./resbook.sh create --name global-server-excl --group gs-cluster-1 --class_name global_
↪server_excl
# создаем клонируемые экземпляры
./resbook.sh create --name global-server-share --group gs-cluster-1 --class_name global_
↪server_share
#экземпляр шедулера
./resbook.sh create --name global-scheduler --group gs-cluster-1 --class_name global_
↪scheduler
# ресурсы балансировщика и мониторинга
./resbook.sh create --name haproxy --group gs-cluster-1 --class_name haproxy
./resbook.sh create --name grafana --group gs-cluster-1 --class_name grafana
```

Сконфигурируйте характеристики

```
./resgroup.sh init_spec --name gs-cluster-1
```

Введите необходимые характеристики или оставте значения по умолчанию

```
Инициализация характеристик для группы ресурсов gs-cluster-1
Установка характеристик для книги ресурсов:global-scheduler
Отслеживать метрики:true
Введите максимальный размер(java -Xmx) для globalscheduler:800M
Введите запрос CPU для globalscheduler:1
Введите запрос MEMORY для globalscheduler:1G
Введите лимиты CPU для globalscheduler:4
Введите лимиты MEMORY для globalscheduler:1G
Введите запрос CPU для systemagent:1
Введите запрос MEMORY для systemagent:250M
Введите лимиты CPU для systemagent:1
Введите лимиты MEMORY для systemagent:500M
Установка характеристик для книги ресурсов:global-server-excl
Отслеживать метрики:true
Введите максимальный размер(java -Xmx) для globalserver:3500M
Введите запрос CPU для globalserver:2
Введите запрос MEMORY для globalserver:4G
Введите лимиты CPU для globalserver:4
Введите лимиты MEMORY для globalserver:4G
Введите запрос CPU для systemagent:1
Введите запрос MEMORY для systemagent:250M
Введите лимиты CPU для systemagent:1
Введите лимиты MEMORY для systemagent:500M
Установка характеристик для книги ресурсов:global-server-share
Отслеживать метрики:true
Введите максимальный размер(java -Xmx) для globalserver:3500M
Введите запрос CPU для globalserver:2
Введите запрос MEMORY для globalserver:4G
Введите лимиты CPU для globalserver:4
Введите лимиты MEMORY для globalserver:4G
```

(continues on next page)

(продолжение с предыдущей страницы)

```
Введите запрос CPU для systemagent:1
Введите запрос MEMORY для systemagent:250M
Введите лимиты CPU для systemagent:1
Введите лимиты MEMORY для systemagent:500M
Установка характеристик для книги ресурсов:grafana
Введите запрос CPU для grafana:1
Введите запрос MEMORY для grafana:500M
Введите лимиты CPU для grafana:4
Введите лимиты MEMORY для grafana:4Gi
Установка характеристик для книги ресурсов:haproxy
Введите запрос CPU для haproxy:1
Введите запрос MEMORY для haproxy:500M
Введите лимиты CPU для haproxy:2
Введите лимиты MEMORY для haproxy:1G
```

Сконфигурируйте параметры кластера

```
./resgroup.sh init_values --name gs-cluster-1
```

```
Инициализация значений для группы ресурсов gs-cluster-1
Введите url базы данных:jdbc:postgresql://pgProject2:5432/nordsy
Введите тип прикладного хранилища:nfs
Введите адрес сервера(server):127.0.0.1
Введите путь(path):/mnt/nfs/gs-cluster-k8s/globalfilestorage
Установка значений для книги ресурсов:global-scheduler
Введите адрес доступа к prometheus:gs-cluster-1-grafana-internal:9090
Установка значений для книги ресурсов:global-server-excl
Введите адрес доступа к prometheus:gs-cluster-1-grafana-internal:9090
Введите внешний ip(external_ip):127.0.0.1
Установка значений для книги ресурсов:global-server-share
Введите количество экземпляров:3
Введите адрес доступа к prometheus:gs-cluster-1-grafana-internal:9090
Установка значений для книги ресурсов:grafana
Введите внешний ip(external_ip):127.0.0.1
Введите класс хранилища:local-storage
Введите размер хранилища:5Gi
Установка значений для книги ресурсов:haproxy
Введите внешний ip(external_ip):127.0.0.1
Введите внешний порт(external_port):8080
Введите имя секрета basic-auth для авторизации статистики:secret-haproxy-auth
Введите имя tls секрета для доступа по https:
```

Запустите кластер

```
./resgroup.sh start_appkit --name gs-cluster-1
./resbook.sh enable_all --group gs-cluster-1
./resgroup.sh enable --name gs-cluster-1
```

5.3 Отладка работы пода

При проблемах конфигурации или любых других проблемах возникающих с подом kubernetes автоматически перезапускает его. Такой режим позволяет поддерживать высокую доступность, но при ошибках трудно выявить причины.

Для отладки работы пода предусмотрен debug режим. Поды в режиме отладки не перезапускаются автоматически, это позволяет подключиться администраторам напрямую в под и отладить работу приложения.

Включение режима отладки

```
./resbook.sh disable --name global-server-excl --group gs-cluster-1
./resbook.sh enable_debug --name global-server-excl --group gs-cluster-1
./resbook.sh enable --name global-server-excl --group gs-cluster-1
```

5.4 Администрирование системы

Команды администрирования кластера выполняются в консоли специального пода `nsctl`.

Для упрощения операций администрирования можно использовать использовать `invoker`, который позволяет выполнять команды на подах.

Внимание: invoker.sh производит удаленное выполнение команд, которые не имеют диалога с пользователем.

Запуск и остановка комплекта приложений

Команды запуска и остановки комплекта приложений останавливают сервера приложений в подах

Остановка комплекта

Остановить комплект приложений

```
./invoker.sh run --namespace gs-cls-debug --app nsctl --cmd "./resgroup.sh stop_appkit --
↪name rg-debug"
```

, где

- gs-cls-debug - неймспейс кластера в кубернетес
- rg-debug - группа ресурсов

Запуск комплекта

```
./invoker.sh run --namespace gs-cls-debug --app nsctl --cmd "./resgroup.sh start_appkit -
↪-name rg-debug"
```

, где

- gs-cls-debug - неймспейс кластера в кубернетес
- rg-debug - группа ресурсов

Обновление комплекта группы

На Jump хосте подготовить элементы комплекта группы:

- ~/nscli/workspace/groupkit/v1/java/jre/lib/security/cacerts
- ~/nscli/workspace/groupkit/v1/libs

Передать `groupkit` в nfs хранилище

```
./groupkit.sh push --namespace gs-cls-debug --source workspace/groupkit/v1 --destination␣
↪groupkits/v1
```

, где

- gs-cls-debug - неймспейс кластера в кубернетес
- workspace/groupkit/v1 - путь расположения версии `groupkit` на jump хосте
- groupkits/v1 - путь хранения на nfs сервере

Обновление `groupkit`

```
# запретить группу ресурсов
./invoker.sh run --namespace gs-cls-debug --app nsctl --cmd "./resgroup.sh disable --
↪name rg-debug"
# переключить groupkit
./invoker.sh run --namespace gs-cls-debug --app nsctl --cmd "./resgroup.sh switch_
↪groupkit --name rg-debug --path groupkits/v1"
# разрешить группу ресурсов
./invoker.sh run --namespace gs-cls-debug --app nsctl --cmd "./resgroup.sh enable --name␣
↪rg-debug"
# запустить комплект приложений
./invoker.sh run --namespace gs-cls-debug --app nsctl --cmd "./resgroup.sh start_appkit -
↪-name rg-debug"
```

, где

- gs-cls-debug - неймспейс кластера в кубернетес
- rg-debug - группа ресурсов
- groupkits/v1 - путь хранения на nfs сервере

Обновление комплекта приложений

На Jump хосте подготовить элементы комплекта приложений:

- ~/nscli/workspace/appkit/gs-cls-debug/v1/applib.zip
- ~/nscli/workspace/appkit/gs-cls-debug/v1/globalserver.zip
- ~/nscli/workspace/appkit/gs-cls-debug/v1/profile/globalscheduler/template/config/quartz.properties
- ~/nscli/workspace/appkit/gs-cls-debug/v1/profile/globalserver/template/config/global3.config.xml

Передать `appkit` в nfs хранилище

```
./appkit.sh push --namespace gs-cls-debug --source workspace/appkit/gs-cls-debug/v1 --
↪destination appkits/v1
```

, где

- gs-cls-debug - неймспейс кластера в кубернетес
- workspace/appkit/gs-cls-debug/v1 - путь расположения версии `appkit` на jump хосте
- appkits/v1 - путь хранения на nfs сервере

Обновление `appkit`

```
# запуск обновления appkit и нагона релиза
./appkit.sh switch_and_upgrade --namespace gs-cls-debug --resgroup rg-debug  --remote_
↪appkit appkits/v1
# старт кластера
./invoker.sh run --namespace gs-cls-debug --app nsctl --cmd "./resgroup.sh start_appkit -
↪-name rg-debug"
```

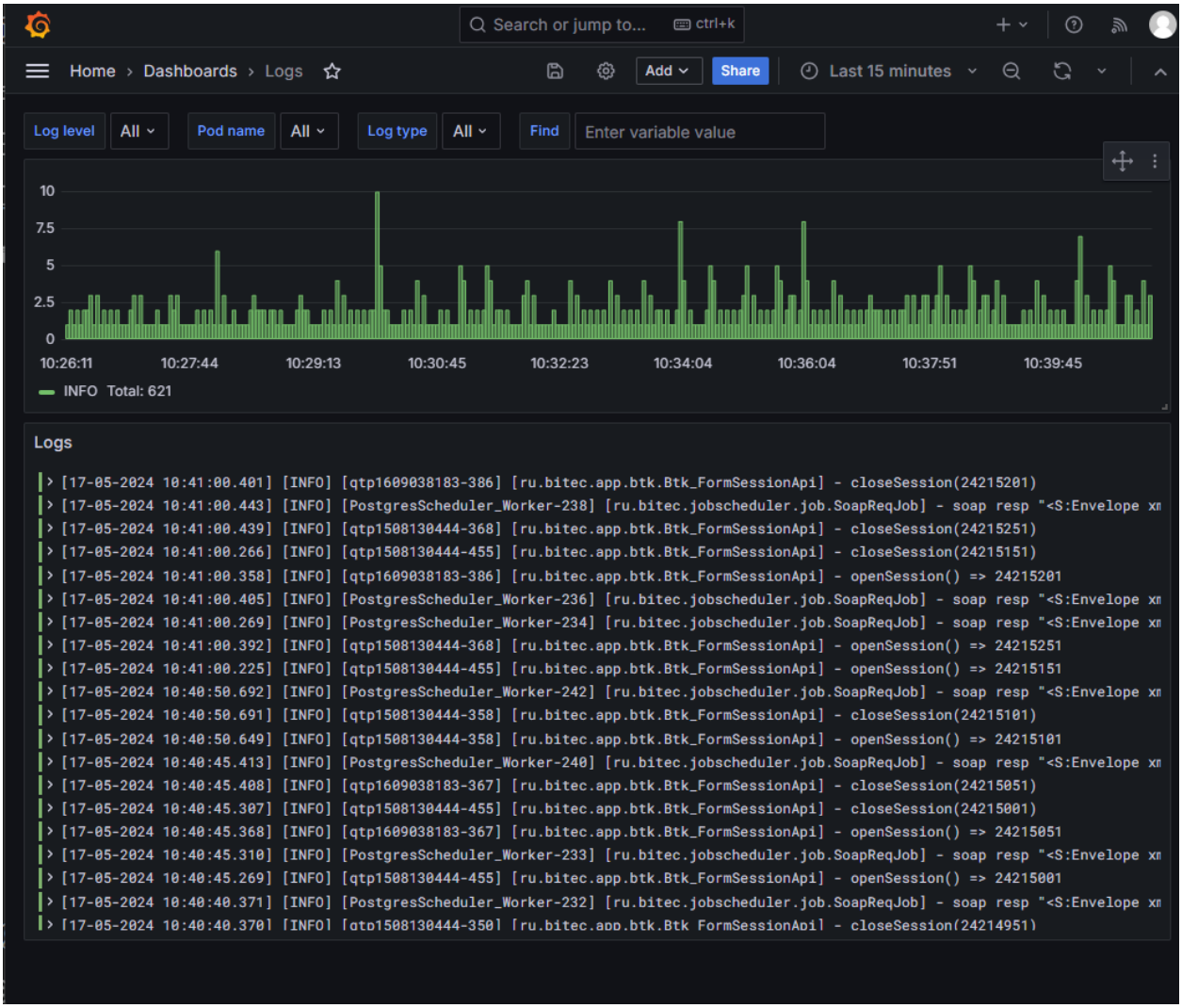
Логи кластера

Логи собираются со всех подов и сохраняются в grafana Loki

Доступ к grafana: http://0.0.0.0:3000/

, где 0.0.0.0 - адрес публикации grafana

В grafana развернут дашбоард с логами: Dashboards > Logs



6 Развертывание сервера приложений GS с сервером авторизации

Инструкция описывает установку сервера (далее также - прокси, gs-authproxy) регистрации и авторизации пользователей, представляющих внешние организации (например, для возможности завести личный кабинет поставщика)

6.1 Подготовка

Для работы прокси необходимо следующее ПО:

1. Debian 11 (с настроенным sudo)
2. Global Server
3. Apache HTTP Server в качестве web-сервера для сервера авторизации
4. HAProxy в качестве форвард-прокси (и, если требуется, распределителя нагрузки) для сервера приложений и сервера авторизации. Возможно заменить на nginx
5. PostgreSQL для хранения сессионной информации.

Global Server установите в соответствии с документацией

Установите пакеты:

```
sudo apt install apache2 libapache2-mod-wsgi-py3
sudo a2enmod wsgi
sudo apt install haproxy
```

В файле `/etc/apache2/ports.conf` измените директиву `Listen 80` на `Listen 8000`, или укажите другой порт.

Создайте PostgreSQL базу данных:

```
create user worker with password 'worker';
create database authproxy;
grant all privileges on database authproxy to worker;
```

6.2 Развертывание и конфигурация

Склонируйте репозиторий в удобную вам папку.

Установите нужные серверу авторизации пакеты и настройте окружение:

```
chmod +x bin/*
bin/installpkg.sh
bin/initvenv.sh
source venv/bin/activate
```

Вы можете использовать тестовые ключи для подписи токенов. Если вы хотите создать свою пару ключей для токенов подписи (настоятельно рекомендуется), то сгенерируйте их следующими командами:

```
openssl genrsa -out privateKey.pem 2048
openssl rsa -in privateKey.pem -pubout -out publicKey.pem
```

Настройка GlobalServer

Откройте в Global Server «Настройка системы» - «Настройки и сервисы» - «Настройки модулей системы» - «Общие настройки модулей» - «btk»:

- Укажите значение `jSettingForGenGidUrl`, например, `{"sTransferProtocolForGenGidUrl":"http", "sHostNameForGenGidUrl":"192.168.24.89:9000"}`
- Проверьте, что флаг `bUsernameEnglishLettersOnly` снят или отсутствует, иначе создание новых пользователей будет невозможно из-за наличия цифр в логинах
- Создайте и укажите значение `sUrlGenPrefix`, равное `sHostNameForGenGidUrl` в `jSettingForGenGidUrl`, например, `192.168.24.89:9000`. Должно быть необязательно, начиная с btk 1.0.700.
- Добавьте публичный ключ подписи в текстовом виде без заголовка и подвала под ключом `extUserPublicKey`. Например, для тестового ключа:

```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAnY1eq7C1PMhnXdvrlM5EcC6B4VgkLheotvPIiLf5vV2ZS+VPDhc2Zy0
↪i4zoFcs8qUwHfCwsTRmdl828YRlzf0rHA/
↪jiT21J1AzkkgMSQmIH9XRxlSS9HmkP6Hgvx7Fe+ir86kU6Pw50LaCeBnlh0RUrolSnyLNhjPuCqIQ54pfz8YzR/
↪T2jMgxU+hvVjD2vC8OJLNDWv7g0rNzynV4zIWNt6gkiHzkvwIDAQAB
```

Настройка gs-authproxy

Заполните шаблон `config.yml` из `deploy/templates` и поместите файл в директорию `mail_gate_pass`:

- Укажите учетные данные для доступа к БД и SMTP-серверу
- Укажите доступ к закрытому ключу подписи (`deploy/templates/private_key.pem`, если используете тестовые значения)
- Укажите доменное имя сервера или IP-адрес, название БД, поправьте ссылки на корректные в соответствии с именем БД и прикладными задачами (например, укажите authorize_url, как просто название бд (`/PGTEST/`), чтобы при входе открывалось меня приложений)

Пример заполненного `config.yml`:

```yaml
database:
  # Имя БД выделенной для работы с проектом
  name: authproxy
  # Имя пользователя
  user: worker
  # Пароль
  password: worker
  # хост на котором будет запущена БД
  host: 10.100.0.1
  # порт на котором будет запущена БД
  port: 5432

django:
  # Секретный ключ Django играет роль в обеспечении безопасности вашего веб-приложения.
  secret_key: 'django-insecure-gf35bki&j&=ftvc!(j92kx+=cu^q(3*54_66ue&e9_6f*ay5ii'
  # Режим разработки.
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  debug: False

endpoints:
  # url на который будет отправляться запросы
  request_url: 'http://127.0.0.1:80/app/sys/rest/ss/pkg/Btk_JexlGatePkg/execute'
  # название базы данных GS
  database_name: pgTest
  # url на который будет отправляться пользователь для аутентификации
  authorize_url: '/PGTEST/'
  # url который используется в режиме разработки для регистрации пользователя
  debug_register_url: '/PGTEST/Bs_RegOrgMainMenu/gtk-ru.bitec.app.bs.organization.Bs_
↪Organization%23CardForRegOrganization/'

email:
  # адрес SMTP-сервера, через который будут отправляться электронные письма. Замените
↪'smtp.example.com' на реальный адрес вашего SMTP-сервера.
  email_host: 'smtp.mail-server.net'
  # порт SMTP-сервера. Значение 587 часто используется для подключения к серверу через␣
↪TLS (Transport Layer Security)
  email_port: 465
  # имя пользователя (адрес электронной почты) для аутентификации на SMTP-сервере.␣
↪Укажите здесь ваш реальный адрес электронной почты.
  email_host_user: 'user@mail-server.net'
  # пароль для аутентификации на SMTP-сервере. Укажите здесь ваш реальный пароль от␣
↪почтового ящика.
  email_host_password: 'password'
  # булевое значение, указывающее, следует ли использовать TLS (Transport Layer␣
↪Security) для защищенного соединения с SMTP-сервером. Установите True, если ваш SMTP-
↪сервер поддерживает TLS, и False в противном случае. Скорее всего True, если порт - 587
  email_use_tls: 'False'
  # булевое значение, указывающее, следует ли использовать SSL для защищенного␣
↪соединения с SMTP-сервером. Установите True, если ваш SMTP-сервер поддерживает SSL, и␣
↪False в противном случае. Скорее всего True, если порт - 465
  email_use_ssl: 'True'
  # email_use_tls и email_use_ssl не могут одновременно быть True

#Для получения реквизитов учетных данных обратитесь в отдел прикладной разработки
#Авторизация пользователей происходит по алгоритму деашт
#При необходимости создайте дерикторию gs-authproxy/mail_gate_pass/security для
#Хранения приватных ключей
# Типы учетных записей используемых в системе:
# - Сервисный пользователь
#   Используется для обращения к rest api сервера приложения
# - Пользователь для регистрации учетных данных
#   Под этим пользователем идет регистрация учетнных данных для работы в сервере␣
↪приложения
# - Простой пользователь
#   Под данной учетной записью происходит штатная работа внешних пользователей
gs_tokens:
  # Ключ для подписи токена сервисного пользователя
  # Должен располагаться в каталоге gs-authproxy/mail_gate_pass/security
  service_private_key: '../deploy/templates/private_key.pem'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  # Ключ для подписи токена пользователя под которым идет регистрация учетных данных
  # Должен располагаться в каталоге gs-authproxy/mail_gate_pass/security
  register_private_key: '../deploy/templates/private_key.pem'

  # Ключ для подписи простых пользователей.
  # Должен располагаться в каталоге gs-authproxy/mail_gate_pass/security
  user_private_key: '../deploy/templates/private_key.pem'
  # имя сервисного пользователя
  service_user_name: 'admin'
  # имя пользователя под которым идет регистрация учетных данных.
  register_user_name: 'admin'


# Используется для указания доверенных источников запросов, которые могут обходить␣
↪защиту от атак CSRF.
# В неё нужно прописывать домены или IP-адреса, с которых разрешены такие запросы.
csrf:
  domain: 'http://192.168.24.89:9000'


# Укажите путь, по которому куки будут доступны
cookie:
  path: '/PGTEST/'
```

Примечание: Не включайте режим debug: на актуальной версии он не работает, так как неправильно перенаправляет на карточку регистрации (начиная с btk 1.0.734 открытие карточки регистрации возможно только по ссылке, сгенерированной сервером, в режиме отладки серевер авторизации пересылает по статичной ссылке).

Затем инициализируйте БД и соберите статические файлы.

```
python manage.py migrate
python manage.py collectstatic
```

Настройка Apache2

Заполните шаблон `001-mail_gate.conf` из `deploy/templates` и поместите файл в директорию `/etc/apache2/sites-available/`:

- Укажите адрес и порт прослушивания (Virtual Host)
- Укажите доменное имя сервера или IP-адрес (ServerName)
- В директивах DocumentRoot, WSGIDaemonProcess, WSGIScriptAlias, Directory, Alias поправьте пути так, чтобы они вели на соответсвующие файлы и папки из репозитория.

Пример заполненного Apache2:

```
# Укажите полные пути в соответствии с вашим проектом.
<VirtualHost *:8000>
    # Устанавливает основное имя сервера для данного виртуального хоста. Здесь указан IP-
↪адрес сервера, можно указать доменное имя.
    ServerName 192.168.24.89:9000
    # Задает каталог, в котором располагаются файлы, обслуживаемые этим виртуальным␣
↪хостом (каталог проекта).
    DocumentRoot /opt/gs-ap/mail_gate_pass

    # Определяет процесс WSGI, используемый для обработки запросов к Python-приложению.
    # project1 - имя процесса.
    # python-home - путь к виртуальной среде Python.
    # python-path - путь к каталогу Django приложения.
    WSGIDaemonProcess project1 python-home=/opt/gs-ap/venv python-path=/opt/gs-ap/mail_
↪gate_pass
    WSGIProcessGroup project1
    # путь к wsgi файлу Django приложения
    WSGIScriptAlias / /opt/gs-ap/mail_gate_pass/mail_gate_pass/wsgi.py

    #Определяет каталог на файловой системе и его настройки доступа. Устанавливает␣
↪правила доступа к файлу с именем wsgi.py и разрешает доступ.
    <Directory /opt/gs-ap/mail_gate_pass/mail_gate_pass>
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>

    # Создает псевдоним URL для статических файлов и определяет каталог в системе где␣
↪хранятся статические файлы и настройки доступа.
    Alias /gs-authproxy/static /opt/gs-ap/mail_gate_pass/static
    <Directory /opt/gs-ap/mail_gate_pass/static>
       Require all granted
    </Directory>

    # Создает псевдоним URL для медиа файлов и определяет каталог в системе где хранятся␣
↪медиа файлы и настройки доступа.
    Alias /gs-authproxy/media /opt/gs-ap/mail_gate_pass/media
    <Directory /opt/gs-ap/mail_gate_pass/media>
        Require all granted
    </Directory>

    # Устанавливает файл, куда будут записываться сообщения об ошибках сервера.
    ErrorLog /var/log/mail_gate_pass-error.log
    # Устанавливает файл, куда будут записываться запросы к серверу.
    CustomLog /var/log/mail_gate_pass-access.log combined
</VirtualHost>
```

Включите сайт:

```
sudo a2ensite 001-mail_gate.conf
```

Настройка HAProxy

Заполните шаблон `haproxy.cfg` из `deploy/templates` и поместите файл в директорию `/etc/haproxy/haproxy.cfg`:

- Укажите адрес и порт прослушивания (директива bind в секции http-in)
- Укажите адрес доступа к Apache (директива server в секции gs_authproxy_server)
- Укажите адрес доступа к Global Server (директива server в секции globalservers)

Пример заполненного `haproxy.cfg`:

```
# определяет, что это для обработки входящих HTTP запросов
frontend http-in
    mode http
    # указывает HaProxy, какие адреса и порты прослушивать
    bind :80
    # создает условие, проверяющее, является ли запрос корнем ("/").
    acl is_root path -m str /
    # создает условие, проверяющее, начинается ли путь запроса с "/gs-authproxy".
    acl is_gs_authproxy path_beg /gs-authproxy
    # указывает HaProxy использовать бэкенд gs_authproxy_servers для запросов,␣
↪удовлетворяющих условию.
    use_backend gs_authproxy_servers if is_gs_authproxy || is_root
    # определяет бэкенд по умолчанию для всех остальных запросов.
    use_backend globalservers unless is_gs_authproxy || is_root

# определяет бэкенд для обработки запросов с префиксом "/gs-authproxy".
backend gs_authproxy_servers
    mode http
    #определяет Django сервер.
    server django_server 127.0.0.1:8000

#определяет бэкенд для всех остальных запросов.
backend globalservers
    mode http
    # определяет  globalserver.
    server globalserver 127.0.0.1:8080
```