



---

Система управления предприятием  
и платформа для цифровизации бизнес-процессов

## Руководство системного администратора

Global-Marine 2.0 Система управления судостроением и  
судоремонтом

---

# G3SysAdminDoc

Выпуск 0.1.9

## Global System

июн. 09, 2025

## Содержание

<b>1</b>	<b>РУКОВОДСТВО СИСТЕМНОГО АДМИНИСТРАТОРА</b>	<b>3</b>
<b>2</b>	<b>Описание Global ERP</b>	<b>3</b>
2.1	Назначение и функции системы . . . . .	3
2.2	Компоненты системы . . . . .	4
2.3	Требования к программному и аппаратному обеспечению . . . . .	4
2.4	Требования к программному обеспечению . . . . .	5
2.5	Структура системы . . . . .	6
2.6	Перечень основных подсистем . . . . .	6
<b>3</b>	<b>Администрирование Postgres</b>	<b>7</b>
3.1	Установка и настройка СУБД . . . . .	7
3.2	Установка и настройка PgBouncer . . . . .	12
3.3	Тестирование производительности сервера . . . . .	15
3.4	Резервное копирование и восстановление БД . . . . .	16
3.5	Смена локалей для postgresql . . . . .	21
<b>4</b>	<b>GlobalServer Автономный режим</b>	<b>23</b>
4.1	Установка сервера приложений . . . . .	23
4.2	Администрирование системы . . . . .	35
4.3	Управление схемой базы данных . . . . .	43
<b>5</b>	<b>GlobalServer кластер в kubernetes</b>	<b>43</b>
5.1	Описание . . . . .	43
5.2	Установка . . . . .	45
5.3	Неуправляемый режим . . . . .	63
5.4	Отладка работы пода . . . . .	72
5.5	Администрирование системы . . . . .	72
5.6	Хуки . . . . .	75
5.7	Отладка прикладного решения . . . . .	77
5.8	Экспорт конфигурации. Восстановление кластера . . . . .	82
5.9	Использование с Ingress . . . . .	84
5.10	Соединение между серверами (JGroups) . . . . .	85
5.11	Обновление . . . . .	87
5.12	Частичное изменение настроек . . . . .	88

<b>6</b>	<b>Развертывание сервера приложений GS с сервером авторизации</b>	<b>88</b>
6.1	Подготовка . . . . .	89
6.2	Развертывание и конфигурация . . . . .	89
6.3	Обновление . . . . .	96
<b>7</b>	<b>Внешняя система мониторинга</b>	<b>97</b>
7.1	Настройка Prometheus . . . . .	98
7.2	Настройка Loki . . . . .	98
7.3	Настройка Tempo . . . . .	99
7.4	Настройка Grafana . . . . .	100
<b>8</b>	<b>Дополнительно</b>	<b>100</b>
8.1	Установка и настройка сервиса конвертации избр. в документы . . . . .	100
8.2	Настройка GlobalScheduler . . . . .	100
<b>9</b>	<b>Первый вход в систему</b>	<b>110</b>
9.1	Получение и установка лицензии . . . . .	110
<b>10</b>	<b>Логирование сервера приложений.</b>	<b>111</b>
10.1	Общий обзор . . . . .	111
10.2	Структура конфигурационных файлов . . . . .	111
10.3	Пример добавления кастомного логгера . . . . .	112

---

# 1 РУКОВОДСТВО СИСТЕМНОГО АДМИНИСТРАТОРА

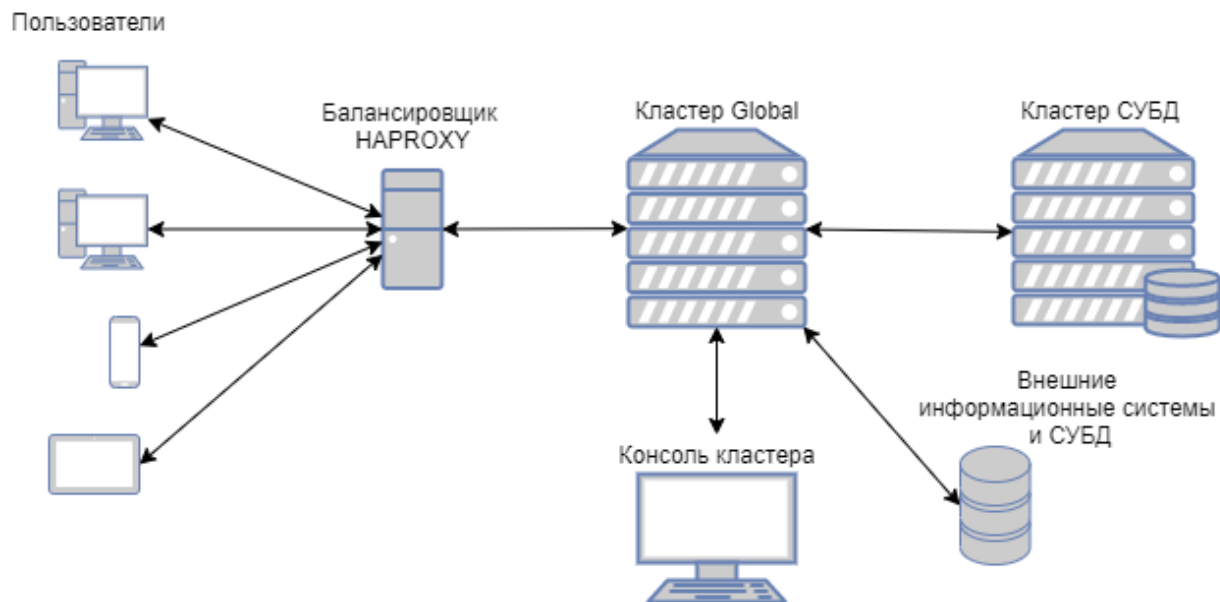


[global-system.ru](http://global-system.ru)

## 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 программному и 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

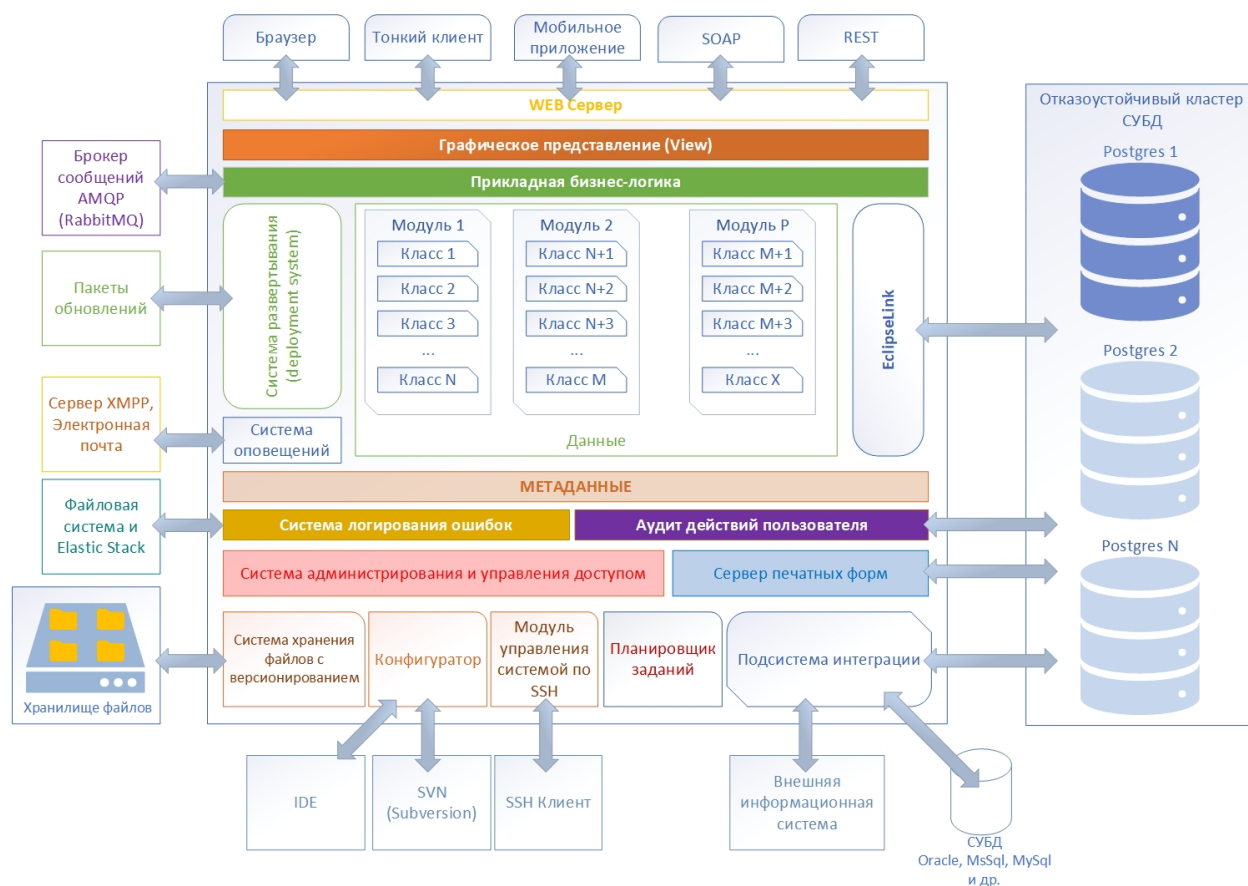
## Рабочие станции

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 1140×900)

## 2.4 Требования к программному обеспечению

Программное обеспечение	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	HotJava, 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 отчетности в различных форматах на основе данных системы;
- выводить подготовленные отчеты 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.

## Добавление локализации

Отредактируйте файл `/etc/locale.gen`, найдите и раскомментируйте строку `ru_RU.UTF-8`

Выполните команду:

```
sudo locale-gen
```

## 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	scram-sha-256

---

**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 TEMPLATE template0;
```

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;
CREATE EXTENSION if not exists "uuid-osspl";
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, воспользуйтесь 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;
```

(continues on next page)

```

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';
    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 имеет 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

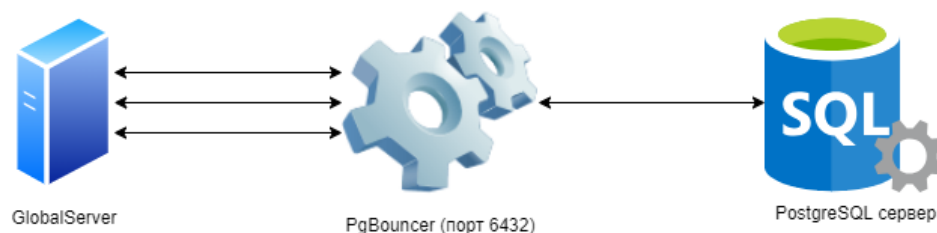
<sDbName> - имя БД

jobs=4 - количество потоков, указывать по количеству ядер

## 3.2 Установка и настройка PgBouncer

PgBouncer — программа, управляющая пулом соединений PostgreSQL. Любое конечное приложение может подключиться к PgBouncer, как если бы это был непосредственно сервер PostgreSQL. PgBouncer создаст подключение к реальному серверу, либо задействует одно из ранее установленных подключений.

Назначение PgBouncer — минимизировать издержки, связанные с установлением новых подключений к PostgreSQL



### Установка

Установите пакет pgbouncer из репозитория дистрибутива при помощи пакетного менеджера.

PgBouncer доступен в официальных APT-репозиториях PostgreSQL. Если репозитории PostgreSQL уже добавлены в вашу систему, вы можете установить PgBouncer напрямую с их использованием.

### Настройка и запуск

#### Предварительная конфигурация PostgreSQL сервера

- Убедитесь, что в файле pg\_hba.conf указан метод аутентификации scram-sha-256:

host	all	all	all	scram-sha-256
------	-----	-----	-----	---------------

- Подключитесь к базе postgres под пользователем postgres:

```
$ su postgres
$ psql
```

- Выполните данный скрипт, который создаст функцию и пользователя, необходимые для аутентификации PgBouncer (измените пароль в поле PASSWORD на более надёжный):

```
CREATE FUNCTION public.lookup (
    INOUT p_user    name,
    OUT  p_password text
) RETURNS record
    LANGUAGE sql SECURITY DEFINER SET search_path = pg_catalog AS
$$SELECT username, passwd FROM pg_shadow WHERE username = p_user$$;

CREATE ROLE pgbouncer WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT
↳NOREPLICATION CONNECTION LIMIT -1 PASSWORD 'pgbouncerpass';
REVOKE EXECUTE ON FUNCTION public.lookup(name) FROM PUBLIC;
GRANT EXECUTE ON FUNCTION public.lookup(name) TO pgbouncer;
```

- Выполните команду, которая выведет хеш пароля для пользователя pgbouncer, и сохраните его. Он понадобится при настройке аутентификации pgbouncer.

```
SELECT passwd FROM pg_shadow WHERE username = 'pgbouncer';
```

- Завершите сессию psql

```
\q
```

## Конфигурация PgBouncer

Конфигурация PgBouncer редактируется в файле /etc/pgbouncer/pgbouncer.ini

- В раздел [databases] нужно добавить адрес и порт, на котором запущен сервер PostgreSQL.

```
[databases]
* = host=localhost port=5432
```

- В разделе [pgbouncer] в значении listen\_addr следует указать, на каких адресах будет слушать PgBouncer, а также при необходимости изменить порт (по умолчанию 6432). При задании значения listen\_addr = \* pgbouncer будет слушать на всех адресах.

```
listen_addr = *
listen_port = 6432
```

- Также в разделе [pgbouncer] следует задать следующие значения:

```
auth_type - метод аутентификации в postgresql
auth_file - путь до файла с данными для аутентификации
auth_user - пользователь, который будет использоваться для аутентификации в postgresql
auth_dbname - база данных, которая используется для аутентификации
auth_query - запрос, который будет выполнен при аутентификации

max_client_conn - максимальное количество клиентских подключений, которые PgBouncer
↳может обслуживать одновременно
```

(continues on next page)

(продолжение с предыдущей страницы)

```
default_pool_size - максимальное количество подключений к базе данных для каждого пула
reserve_pool_size - количество дополнительных подключений, доступных в резервном пуле,
↳ которые используются, если пул исчерпан
reserve_pool_timeout - указывает, сколько секунд клиент может ждать подключения из
↳ резервного пула
pool_mode - определяет, как PgBouncer управляет соединениями. Для использования
↳ PgBouncer с системой Global следует установить режим transaction, чтобы снизить
↳ нагрузку на БД
max_prepared_statements - определяет максимальное количество подготовленных SQL-запросов,
↳ которые PgBouncer может хранить на одно соединение
```

- Пример конфигурации для корректной работы аутентификации с использованием вышеописанной sql-функции:

```
auth_type = scram-sha-256
auth_file = /etc/pgbouncer/userlist.txt
auth_user = pgbouncer
auth_dbname = postgres
auth_query = SELECT p_user, p_password FROM public.lookup($1)
```

- Пример конфигурации для использования с k8s кластером Global из 10 подов (значения следует задать в зависимости от предположительной нагрузки на БД)

```
max_client_conn = 5000
default_pool_size = 20
reserve_pool_size = 10
reserve_pool_timeout = 5
pool_mode = transaction
max_prepared_statements = 200
```

- Для корректной работы pgbouncer с dbeaver и globalserver следует раскомментировать поле «ignore\_startup\_parameters» и задать ему следующее значение:

```
ignore_startup_parameters = extra_float_digits,search_path
```

- Также в сервисе pgbouncer (/lib/systemd/system/pgbouncer.service) следует раскомментировать параметр LimitNOFile, отвечающий за максимальное количество открытых файловых дескрипторов, доступных для процесса, и настроить его в зависимости от количества подключений:

```
[Unit]
Description=connection pooler for PostgreSQL
Documentation=man:pgbouncer(1)
Documentation=https://www.pgbouncer.org/
After=network.target
#Requires=pgbouncer.socket

[Service]
Type=notify
User=postgres
ExecStart=/usr/sbin/pgbouncer /etc/pgbouncer/pgbouncer.ini
ExecReload=/bin/kill -HUP $MAINPID
KillSignal=SIGINT
LimitNOFILE=65535
```

(continues on next page)

(продолжение с предыдущей страницы)

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

```
$ sudo systemctl daemon-reload  
$ sudo systemctl restart pgbouncer
```

### Добавление пользователей

Пользователи добавляются в файл /etc/pgbouncer/userlist.txt

Формат списка пользователей имеет следующий вид:

```
"<ИМЯ_ПОЛЬЗОВАТЕЛЯ>" "<ПАРОЛЬ/ХЕШ ПАРОЛЯ>"
```

Для работы аутентификации с вышеописанными настройками, в userlist.txt требуется установить следующие значения:

```
"pgbouncer" "<scram-sha-256 хеш пароля для пользователя pgbouncer>"
```

### Запуск PgBouncer

После завершения настройки PgBouncer запустите его через сервис:

```
$ sudo systemctl enable --now pgbouncer
```

По умолчанию pgbouncer будет доступен на 6432 порту сервера.

## 3.3 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  
initial connection time = 56.060 ms  
tps = 51677.092175 (without initial connection time)
```

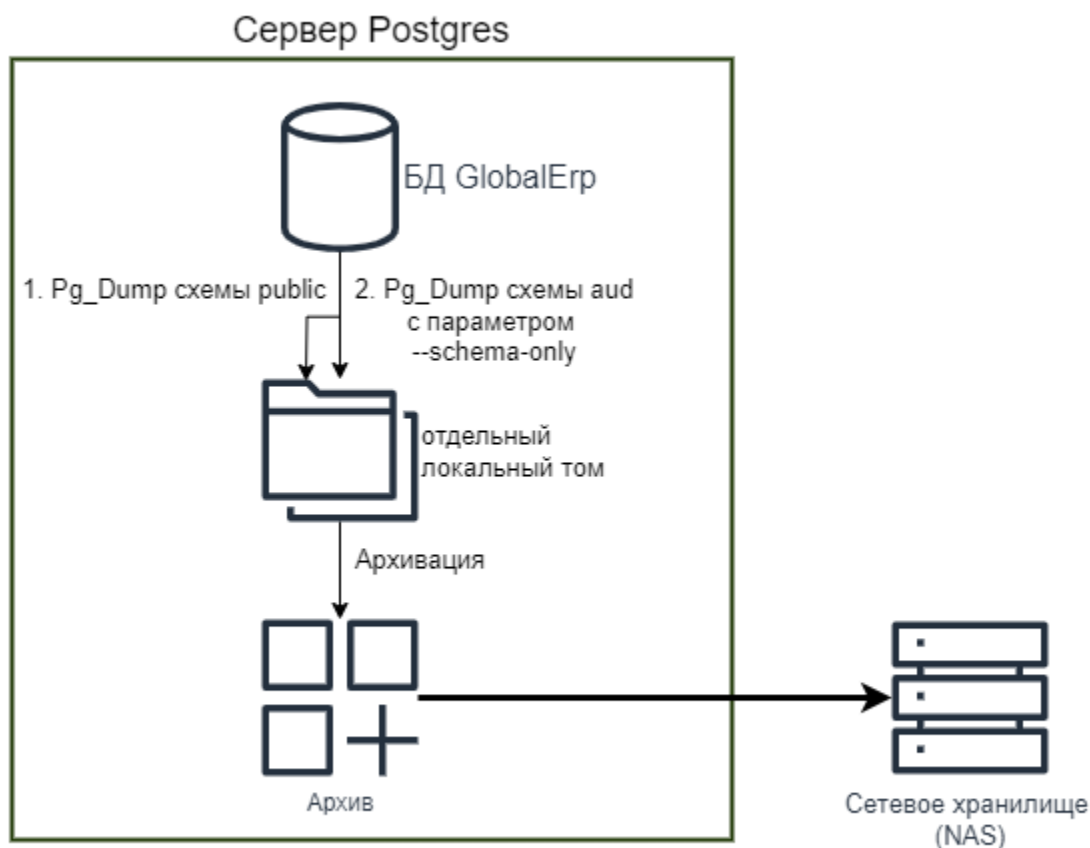
### 3.4 Резервное копирование и восстановление БД

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 дампы 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 $$ DECLARE  
    r RECORD;  
BEGIN  
    FOR r IN (SELECT tablename FROM pg_tables WHERE schemaname = current_schema()) LOOP  
        EXECUTE 'DROP TABLE IF EXISTS ' || quote_ident(r.tablename) || ' CASCADE';  
        commit;  
    END LOOP;  
END $$;  
--  
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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
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

## 3.5 Смена локалей для postgresql

Руководство, как сменить локаль БД Postgresql

Как проверить какая локализация в данный момент

- Выполнить sql-запрос

```
SELECT name, setting FROM pg_settings WHERE name LIKE 'lc_%';
```

- Пример английской локализации

	A-Z name	A-Z setting
1	lc_collate	en_US.UTF-8
2	lc_ctype	en_US.UTF-8
3	lc_messages	en_US.UTF-8
4	lc_monetary	en_US.UTF-8
5	lc_numeric	en_US.UTF-8
6	lc_time	en_US.UTF-8

- Пример русской локализации

	A-Z name	A-Z setting
1	lc_collate	ru_RU.UTF-8
2	lc_ctype	ru_RU.UTF-8
3	lc_messages	ru_RU.UTF-8
4	lc_monetary	ru_RU.UTF-8
5	lc_numeric	ru_RU.UTF-8
6	lc_time	ru_RU.UTF-8

- Через psql

```
psql
\l
```

— Английская локализация

```
postgres=# \l
```

List of databases							
Name	Owner	Encoding	Collate	Ctype	ICU Locale	Locale Provider	Access privileges
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8		libc	
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8		libc	=c/postgres +
							postgres=CTc/postgres
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8		libc	=c/postgres +
							postgres=CTc/postgres

(3 rows)

— Русская локализация

List of databases							
Name	Owner	Encoding	Collate	Ctype	ICU Locale	Locale Provider	Access privileges
postgres	postgres	UTF8	ru_RU.UTF-8	ru_RU.UTF-8		libc	
template0	postgres	UTF8	ru_RU.UTF-8	ru_RU.UTF-8		libc	=c/postgres +
template1	postgres	UTF8	ru_RU.UTF-8	ru_RU.UTF-8		libc	postgres=CtC/postgres +
(3 rows)							postgres=CtC/postgres

## Добавление локализации

Отредактируйте файл `/etc/locale.gen`, найдите и раскомментируйте строку `ru_RU.UTF-8`

Выполните команду:

```
sudo locale-gen
```

## Смена локализации в конфиге postgresql

Отредактируйте файл `/etc/postgresql/<version_number>/main/postgresql.conf`

```
sudo nano /etc/postgresql/<version_number>/main/postgresql.conf
```

Найдите указанные строки, они должны быть с такими значениями:

- `lc_messages = „ru_RU.UTF-8“`
- `lc_monetary = „ru_RU.UTF-8“`
- `lc_numeric = „ru_RU.UTF-8“`
- `lc_time = „ru_RU.UTF-8“`

## Пересоздание кластера базы данных

---

**Примечание:** Если вам нужно сохранить ваши данные, то перед выполнением, снимите дамп с вашей БД

---

Для смены локализации требуется пересоздать кластер базы данных с нужной локалью.

- Остановите PostgreSQL

```
sudo systemctl stop postgresql
```

- Переинициализируйте кластер

```
sudo pg_dropcluster <версия> main --stop
sudo pg_createcluster <версия> main --locale=ru_RU.UTF-8 --start
```

- Запустите postgresql

```
sudo systemctl start postgresql
```

Если вам нужно изменить локализацию без пересоздания кластера (например, для одной базы данных), можно создать новую базу с заданной локалью:

```
CREATE DATABASE mydb WITH TEMPLATE = template0 LC_COLLATE = 'ru_RU.UTF-8' LC_CTYPE = 'ru_
↳RU.UTF-8' ENCODING = 'UTF8';
```

## 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
  - GocJava <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,icharset=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

Процесс настройки сервиса описан в *документации 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

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration xmlns="http://www.global-system.ru/xsd/global3.config.1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.global-system.ru/xsd/global3.config.1.0">
  <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>
```

(continues on next page)

```

        <metaManager
            mode="Xml"
            defaultNamespace="ru.bitec.app.btk"
            sbtName="main"
        />
        <eclipseLink
            persistenceUnitName="pgdev"
            autoCommit="false"
        />
    </database>
</databases>

<sbt>
    <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>
</sbt>

<security>
    <!--В секции могут быть указаны значения параметров подключения по
    ↪ умолчанию-->
    <!--user - имя пользователя-->
    <!--password - пароль-->
    <!--dataBase - база данных-->
    <!--application - приложение-->
    <loginDefaults user="admin" password="admin" dataBase="DEVBTK"/>
    <!--Управляет сохранением параметров последнего входа в куках браузера-->
    <loginSavable user="true" password="true"/>
    <!--Ограничения на стойкость паролей.-->
    <!--<passwordStrength-->
    <!--minLength="6"-->
    <!--maxLength="20"-->
    <!--digitCount="1"-->
    <!--specialCharCount="0"-->
    <!--lowercaseCharCount="0"-->
    <!--uppercaseCharCount="0"-->
    <!--rejectUserName="false"-->
    <!--rejectWhitespace="false"-->
    <!--/>-->
    <users>
        <user name="admin" password="admin" roles="ssh,http-monitor"/>
        <!--Роль "system" предоставляет право открытия пользовательских
    ↪ сессий при нахождении сервера в сервисном режиме-->

```

(continues on next page)

```

        <user name="system" password="system" roles="system"/>
    </users>
</security>

<quotas>
    <server maxOldGenMemory="0" maxEdenSpaceMemory="80%"/>
    <session maxUiCellCount="100M" maxFormCount="30"/>
    <transaction maxTxCellCount="100M"/>
</quotas>

<!--minPoolSize - Минимальное число соединений в пуле-->
<!--maxPoolSize - Максимальное число соединений в пуле-->
<!--initialPoolSize - Начальное число соединений в пуле-->
<!--inactiveConnectionTimeout - Время в течении которого сессия остаётся в пуле,
→ после освобождения, в секундах.-->
<!--poolTimeout - Максимальное время ожидания получения соединения из пула, в
→ секундах.-->
<!--maxActiveThreadCount - Максимальное количество потоков работающих с базой
→ данных.-->
<!--activeThreadTimeout - Максимальное время ожидания потока для начала работы
→ с базой данных, в секундах.-->
<connectionPool minPoolSize="2"
                maxPoolSize="200"
                initialPoolSize="2"
                inactiveConnectionTimeout="60"
                poolTimeout="10"
                maxActiveThreadCount="170"
                activeThreadTimeout="10"/>

<!--clientTimeout - Максимальное время бездействия сокета. В секундах.-->
<!--clientPingInterval - Интервал выполнения пинга клиентского приложения, для
→ предотвращения закрытия веб-сокета по таймауту. В секундах.-->
<!--clientPingEnabled - Если true, сервер будет периодически пинговать клиента,
→ что бы сокет не закрывался-->
<!--sessionTimeout - Время жизни сессии после разрыва соединения с браузером.
→ Интервал в секундах.-->
<!--maxSessionCount - Максимальное число запущенных сессий-->
<!--maxSessionsPerUser - Максимальное число запущенных сессий для пользователя-->
<!--
    (лишние будут закрываться при закрытии окна браузера)-->
<!--cookieExpiresTime - Время жизни Cookies в секундах-->
<sessionPool clientTimeout="600"
            clientPingInterval="300"
            clientPingEnabled="true"
            sessionTimeout="900"
            maxSessionCount="200"
            passiveSessionsMax="2"
            cookieExpiresTime="86400"

/>

<!--host - Адрес web-сервера:порт, на котором запущен сервис построения отчётов
→ FastReport.Mono.
    Этот же адрес должен быть указан на прокси-узле -->

```

(continues on next page)

```

    <!--proxyaddress - Корнерой адрес куда клиентское приложение будет посылать
→запрос на получение отчёта.-->
    <!--
    Абсолютный, если необходимо направить запросы напрямую к
→серверу отчётов или на внешний сервис.-->
    <!--
    Относительный, если на текущем веб-сервере настроен проху к
→серверу отчётов.-->
    <fastReportMonoServer host="http://global3:90" proxyaddress="fastreportmono"/>
    <!--host - Адрес веб-сервера:порт, на котором запущен сервис построения отчётов
→FastReport.VCL.
    Этот же адрес должен быть указан на прокси-узле -->
    <!--proxyaddress - Корнерой адрес куда клиентское приложение будет посылать
→запрос на получение отчёта.-->
    <!--
    Абсолютный, если необходимо направить запросы напрямую к
→серверу отчётов или на внешний сервис.-->
    <!--
    Относительный, если на текущем веб-сервере настроен проху к
→серверу отчётов.-->
    <fastReportServer host="http://global3:8010" proxyaddress="fastreport"/>
    <!--<fastReportServer host="http://localhost:8080/fastreport" handler=
→"CreateReport.ashx" proxyaddress="fastreport"/>-->

    <!--enableNetworkPrinting - Включает режим печати отчётных форм на принтерах в
→подсети сервера приложений. -->
    <!-- В противном случае печать осуществляется на клиентской стороне.-->
    <printing enableNetworkPrinting="false"/>

    <!--selectionMetaExpireTimeInSeconds - Время, в течение которого в кэше
→метаданные выборки считаются валидными-->
    <cache selectionMetaExpireTimeInSeconds="300"/>

    <!--operationLevel - Минимальный уровень логов операций отправляемых на клиент-->
    <!--sqlLevel - Минимальный уровень логов SQL отправляемых на клиент-->
    <!--scriptLevel - Минимальный уровень логов скриптовых методов отправляемых на
→клиент-->
    <clientLog operationLevel="INFO" sqlLevel="OFF" scriptLevel="OFF"/>
    <client>
        <automation metaDataAttributes="false"
            jexlExecution="false"
            operExecution="false"/>
    </client>
    <!--enabled - флаг доступности службы оповещения-->
    <instantMessenger enabled="true"/>

    <development>
        <!--enabled - режим отладки. Делает видимым панель отладки в главном
→меню приложения-->
        <debugMode enabled="true"/>
        <!--enabled - Доступность флага "Конфигуратор" в диалоге подключения-->
        <configurator enabled="true"/>
    </development>

    <!--jmxServiceUrl - Пользователь/пароль, под которым создаются временные таблицы
→для olap-кубов-->

```

(continues on next page)

```

    <!--user - пользователь доступа к олап серверу-->
    <!--password - пароль доступа к олап серверу-->
    <olap jmxServiceUrl="service:jmx:rmi:///jndi/rmi://:1999/jmxrmi" user="olap"
    ↪password="olap"/>

    <!-- Можно вручную указать директорию для временных файлов, используемую Global3
    ↪ для своих нужд.
        По умолчанию берется директория для временных файлов вашего пользователя.
        На win7/8 это обычно "C:\Users\<yourUserName>\AppData\Local\Temp\global3"
        Меняйте значение по умолчанию ТОЛЬКО В ТОМ СЛУЧАЕ, если по каким-то причинам вы
    ↪ не имеете доступа к директории по умолчанию.
        Для этого раскомментируйте следующую строку и установите путь к необходимой
    ↪ директории. -->
    <!--<temporaryDir>C:\temp</temporaryDir>-->

    <cluster enabled="false" name="GlobalAppServerCluster">
    <broker>
        <database alias="cluster_broker" driver="org.postgresql.Driver" schema=
    ↪ "global_system"
            url="jdbc:postgresql://v39:5432/test">
            <users>
                <user name="test" password="test"/>
            </users>
        </database>
    </broker>
    <node>
        <balancerTypes>["prod","dev"]</balancerTypes>
    </node>
    </cluster>

    <languages>
    <language name="ru-RU"/>
    <language name="en-US"/>
    </languages>

    <ssh defaultDb="PostgreSql" port="22"/>

</configuration>

```

Где:

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

<host> - адрес сервера postgres

<ИмяБД> - имя базы данных, подготовленной 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

Процесс конфигурации описан в *документации GlobalScheduler*.

## 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
```

(continues on next page)

```
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 параметрами `-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 параметрами `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

Процесс генерации и установки ключей шифрования для планировщика описан в [документации GlobalScheduler](#).

## 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 установлены 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
```

---

**Примечание:** Если при подключении к серверу у вас появляется ошибка

No matching host key type found. Their offer: ssh-rsa

То для её исправления добавьте в файл конфигурации ssh:

```
Host localhost
    PubkeyAcceptedAlgorithms +ssh-rsa
    HostkeyAlgorithms +ssh-rsa
```

---

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 нагона релиза:

```
upgrade
```

## Получение и установка лицензии

Процесс получения и установки лицензии описан в *документации по первому входу в систему*.

## Сброс пароля admin

### Перед сбросом пароля

Перед сбросом пароля проверьте файл `global3.config.xml`

В нем должны быть прописаны пользователи в блоке `security`

```
</security>
    <users>
        <user name="admin" password="admin" roles="ssh,http-monitor"/>
        <!--Роль "system" предоставляет право открытия пользовательских
↪сессий при нахождении сервера в сервисном режиме-->
        <user name="system" password="system" roles="system"/>
    </users>
</security>
```

## Инструкция

- Запустить сервер приложения
- Подключиться к серверу приложения по ssh

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

- Перевести сервер в режим `service`

```
alter server mode service
```

- Зайти под учеткой `system`, `<db_alias>` - алиас БД из `global3.config.xml`

```
login system/system@<db_alias>
```

- Выполнить команду

```
jexl
```

- Вставить содержимое файла `reset_admin_pass.jexl`

```
let sUserName = "admin";
let sRawPassword = "admin";
```

(continues on next page)

(продолжение с предыдущей страницы)

```
let idUser = Btk_UserApi.findByMnemoCode(sUserName);
if (isNull(idUser)) {
    raise("Пользователь с именем '" + sUserName + "' не найден.");
}

let ropUser = Btk_UserApi.load(idUser);
if (isNull(ropUser)) {
    raise("Ошибка загрузки пользователя с ID: " + idUser);
}

let idPassType = Btk_UserPassTypeApi.idTemporaryPassword();
if (isNull(idPassType)) {
    raise("Ошибка получения типа временного пароля.");
}

if (isNull(Btk_UserApi)) {
    raise("Ошибка: Btk_UserApi - null");
}
if (isNull(session)) {
    raise("Ошибка: session - null");
}

Btk_UserApi.setsPass(ropUser, sRawPassword, idPassType);

session.commit();
```

- Написать / и нажать Enter
- Перевести сервер в normal режим

```
alter server server mode normal
```

- Закрыть соединение

```
exit
```

## После выполнения инструкции

После выполнения инструкции, при входе под учетной записью **admin** пароль будет **admin**

После входа нужно будет заменить пароль на новый.

## 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

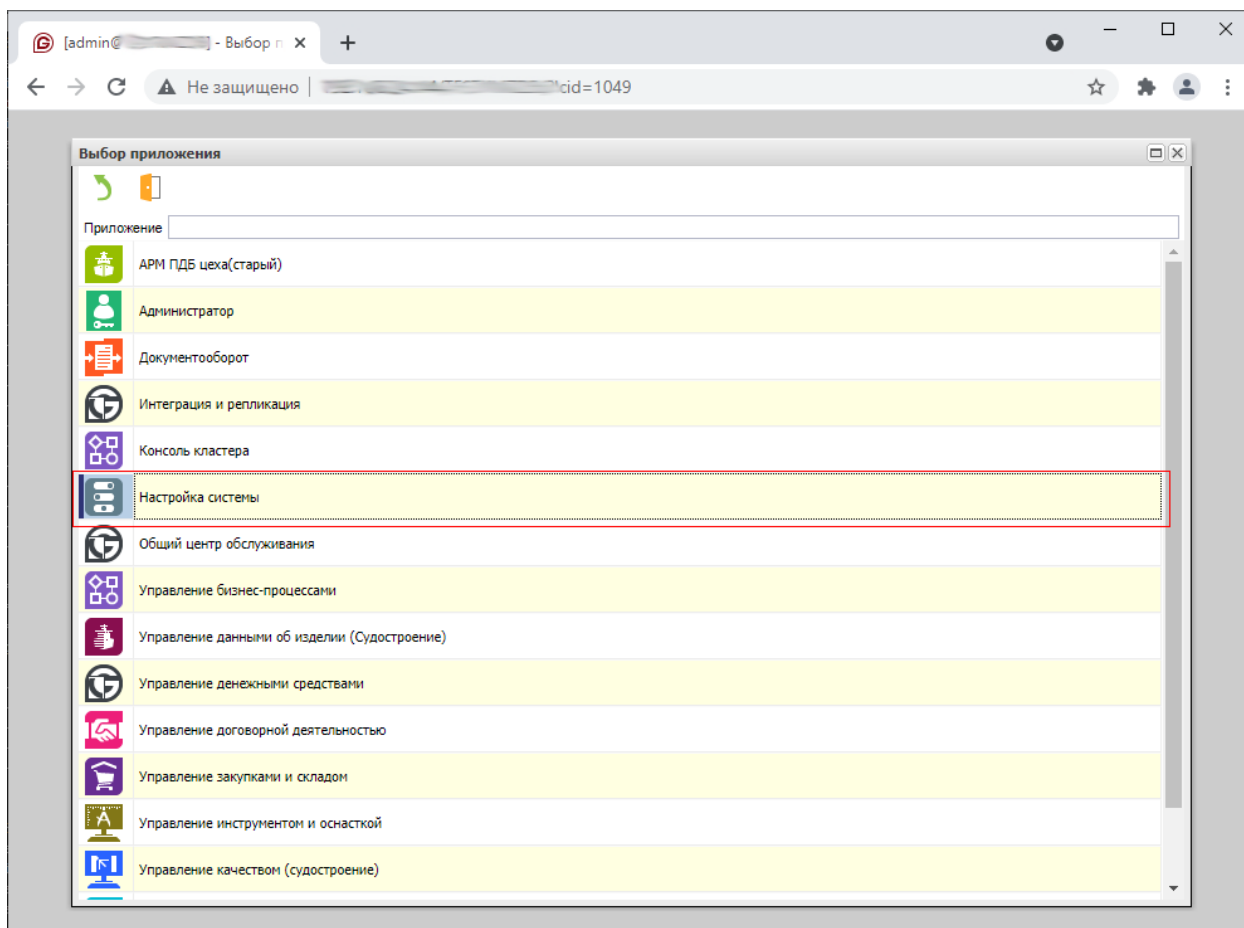
Система версионного хранения файлов может работать в двух режимах:

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

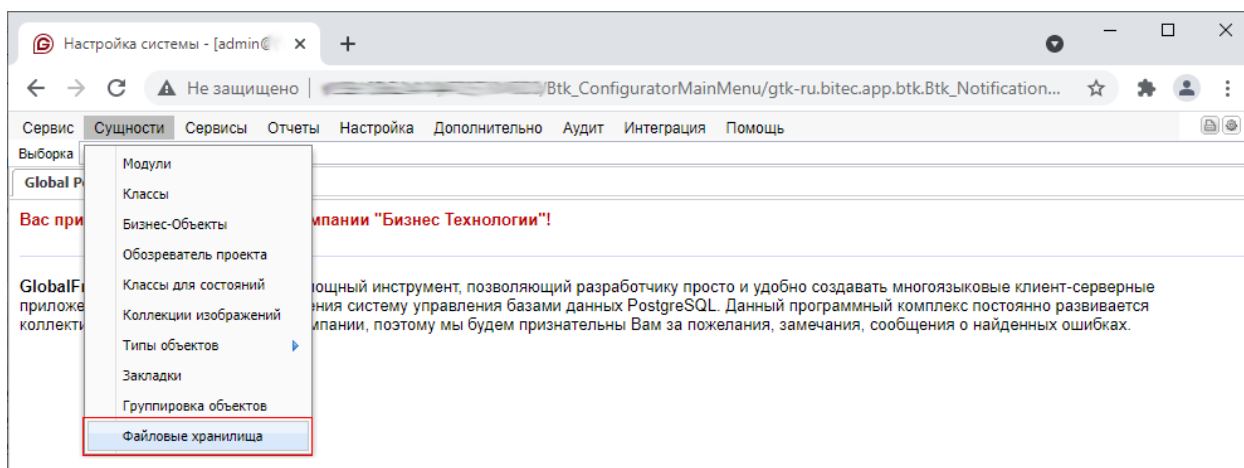
Рекомендуется обеспечить инкрементное резервное копирование директории или сетевого ресурса, который будет использоваться для файлового хранилища Global System.

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.)

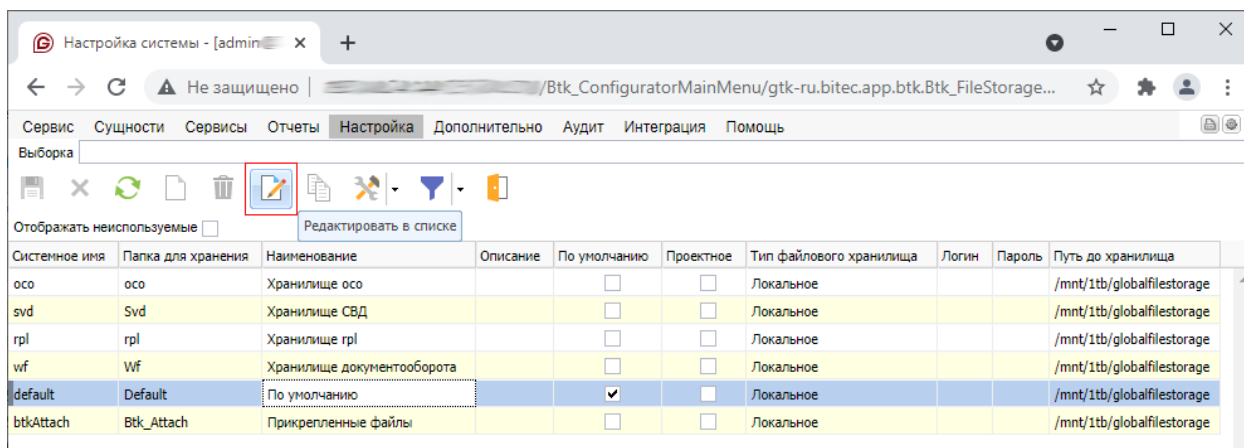
Для настройки файловых хранилищ требуется открыть приложение «Настройка системы»



Открыть меню: Сущности | Файловые хранилища



В списке разблокировать редактирование



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

## Перенос файлового хранилища в системе

Для выполнения операции переноса файлового хранилища внутри системы необходимо воспользоваться специализированным сервисом.

### Порядок действий:

#### 1. Подготовка исходного хранилища

1. Перейдите в раздел «Настройки системы» → «Сущности» → «Файловые хранилища».
2. Включите режим редактирования

**Совет:** Используется как для переименования, так и для создания нового хранилища. Для активации режима редактирования нажмите соответствующую кнопку в интерфейсе списка.

1. Выберите хранилище, подлежащее переносу, и добавьте к его названию суффикс **\_old** (например, storage → storage\_old).

#### 2. Создание нового хранилища

1. **Создайте новую запись** в списке файловых хранилищ.
2. Заполните обязательные поля:
  - **Системное имя** (Наименование старого хранилища, без суффикса \_old)
  - **Папка для хранения**
  - **Наименование**
  - **Тип файлового хранилища**
  - **Путь до хранилища**
3. Сохраните новое хранилище.



### 3. Настройка переноса данных

1. У **исходного хранилища** (с суффиксом `_old`) перейдите на вкладку **«Настройка очистки хранилища от устаревших файлов»**.
2. Создайте новую запись, указав:
  - **Срок хранения** = 0
  - **Действие** = «Переместить»
  - **Целевое хранилище** (выберите только что созданное хранилище).

### Регламент выполнения операции

Перенос файлов выполняется автоматически в рамках ночного задания **«Актуализация расположения файлов в файловых хранилищах»**.

### Запуск вручную

При необходимости немедленного выполнения доступны следующие варианты:

- **Через Менеджер заданий** (внеплановый запуск соответствующей задачи).
- **Через JEXL-скрипт:**

```
Btk_FileStorageCleanSettingApi.procTempFiles()
```

### Настройка ключей шифрования

Процесс генерации и настройки ключей шифрования описан в *документации GlobalScheduler*.

### Обновление системы

Для обновления системы используется специальная утилита `/usr/local/globalserver/update/update.sh`

Перед использованием следует сформировать конфигурационный файл `update/config.sh` (`config.ps1`) на основе `config.sh.origin` и загрузить в указанную папку полученные дистрибутивы.

---

**Совет:** Хранить пароль в конфигурационном файле необязательно - если пароль не указан, он будет запрошен во время выполнения утилиты.

---

**Предупреждение:** Утилита провоцирует необратимые изменения в базе данных. Перед использованием делайте резервные копии БД.

## Режимы 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 `-m`

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

## Режим восстановления

---

**Примечание:** Режим восстановления появился недавно и поэтому может быть недоступен на вашей системе.

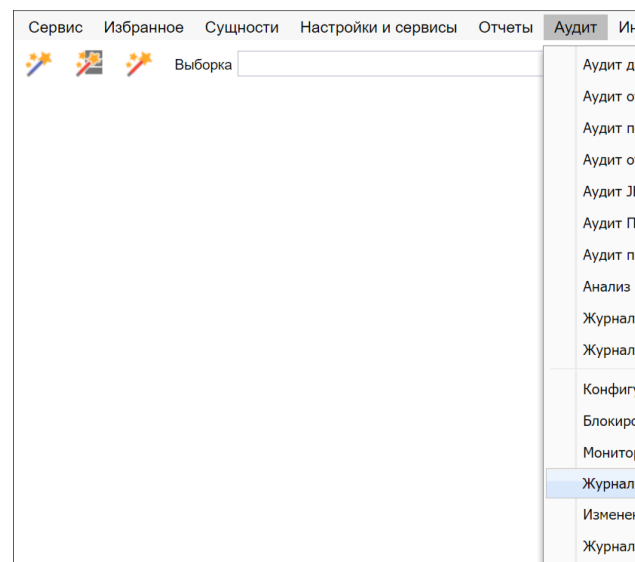
---

Во время обновления утилита автоматически создает резервные копии сервера приложений и прикладного решения (но не БД!). Вы можете откатиться к ним при помощи команды `update.sh -r`. Эта команда начнет диалог, который предложит выбрать резервные копии для восстановления, а затем и само восстановление.

Если интерактивность не требуется, вы можете также добавить аргументы `-s` и `-a`. Укажите после первого путь к резервной копии сервера приложений, а после второго - прикладного решения. Вы можете указать вместо пути слово `skip`, это позволит пропустить обновление соответствующего компонента.

```
./update.sh -r -s skip
# выводит диалоговое окно выбора резервной копии прикладного решения и восстанавливает ее
./update.sh -r -s /tmp/globalupdate/update_yesterday/backup_server.tgz -a /tmp/
↪globalupdate/update_yesterday/backup_applib.tgz
# восстанавливает сервер приложений и прикладное решение из указанных резервных копий
```

## Отображение «Журнал обновления модулей»



Расположение: Настройки системы > Аудит > Журнал обновлений

id сессии обновления	Дата обновления	Обновленные модули	Установленный комплект сборки	Скрипты обновления схемы и данных	Обновление схемы	Скрипты до обновления схемы	Скрипты после обновления схемы	Миграционные задачи
197	21.03.2025 07:50:11	act 1.12.11.249...		✓	✓			✓
196	20.03.2025 14:34:47			✓	✓			
195	20.03.2025 14:14:35	act 1.12.11.245...		✓	✓			
194	20.03.2025 07:50:09	act 1.12.11.247...		✓	✓			
193	19.03.2025 18:51:49	act 1.12.11.245...		✓	✓			
192	19.03.2025 12:19:48				✓			
191	19.03.2025 07:49:53	act 1.12.11.242...		✓	✓			
190	18.03.2025 14:19:29	act 1.12.11.243...		✓	✓			
189	18.03.2025 15:03:36							
188	18.03.2025 07:50:03	act 1.12.11.242...		✓	✓			✓
187	17.03.2025 14:55:25	act 1.12.11.241...		✓	✓			
186	17.03.2025 07:50:01			✓	✓			
185	16.03.2025 07:49:58			✓	✓			
184	15.03.2025 07:49:58	act 1.12.11.240...		✓	✓			
183	14.03.2025 10:03:30			✓	✓			
182	14.03.2025 07:49:57	act 1.21.4.425...		✓	✓			
181	13.03.2025 10:05:15	act 1.12.11.237...		✓	✓			
180	13.03.2025 07:49:41			✓	✓			
179	12.03.2025 18:19:02	rpm 1.43.17.689...		✓	✓			
178	12.03.2025 12:48:02	act 1.12.11.236...		✓	✓			
177	11.03.2025 14:21:46	act 1.21.4.422...		✓	✓			

Путь к журналу

Интерфейс журнала обновлений

Существующие закладки:

1. *Обновление модулей*  
отображение в виде дерева, поделенное на раздел с обновленными и необновленными модулями
2. *Обновление схемы*  
содержит информацию о всех, выполненных при генерации таблиц, DDL-скриптах в виде текста скрипта и возможной ошибки
3. *Скрипты обновления схемы и данных*  
содержит информацию о всех, выполненных при генерации таблиц, скриптах в виде модуля и наименования, версий и текста возможной ошибки
4. *Лог обновления*  
содержит полную информацию отчета генератора
5. *Скрипты, выполненные до и после обновления схемы*
6. *Миграционные задачи upTask и downTask*

- Атрибуты в виде галочек информируют о наличии записей в одноименных закладках. Соответственно, если запись есть - галочка стоит.
- Закрашивание поля красным сигнализирует о том, что в записях одноименных закладок имеются ошибки.

### Механизм построения выборки при генерации таблиц через ssh

Добавление записей в журнал происходит только при генерации таблиц через ssh

### Запись данных о версиях модулей

Срабатывает метод dataUpdate класса DbSchemaUpdater

Этот метод:

- Берет старые и новые версии модулей
- Версии комплектов сборки
- Обновляет в соответствии с этими данными, запись с соответствующим id сессии (который каждый раз увеличивается на 1)

## 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 удалены из кода приложения.

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

### Использование

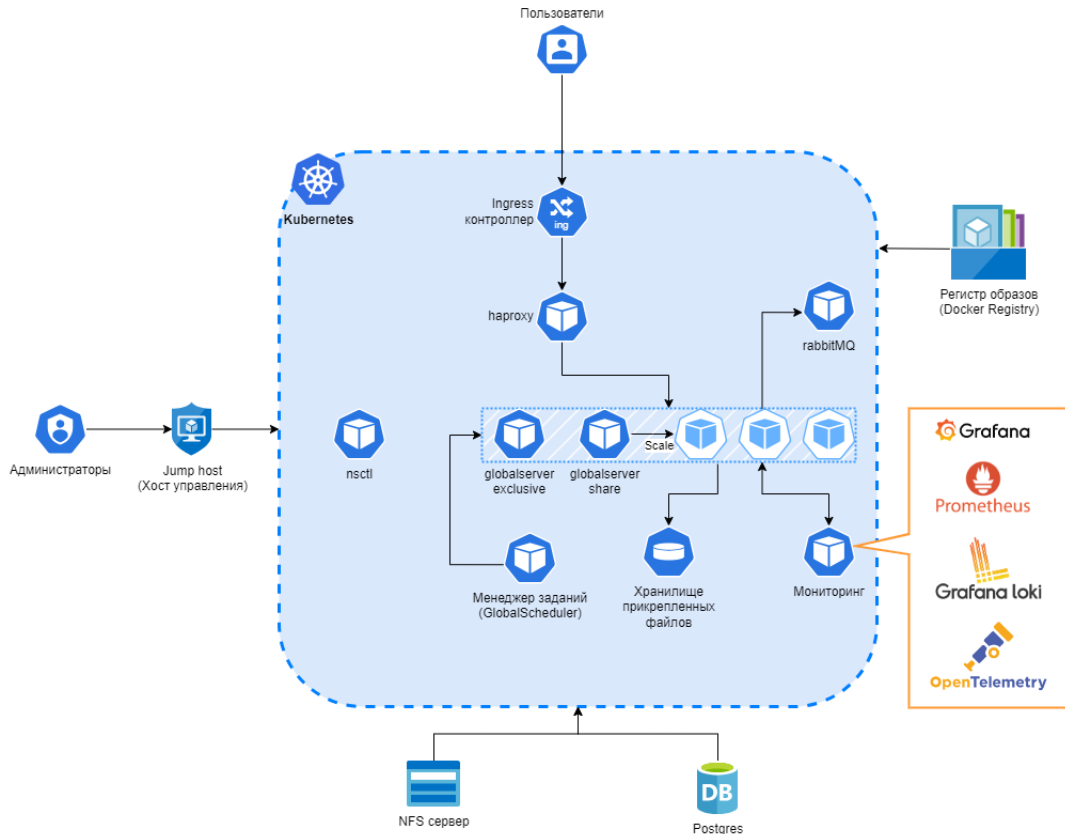
Выборка позволяет искать удаленные объекты, которые остались в базе данных. В ней есть две операции:

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. Работает в закрытой корпоративной сети без доступа в интернет.



## 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

Для работы с Global Server на кластере Kubernetes требуется:

- jump-хост, с которого будет установлен и будет обслуживаться кластер
- готовый к работе кластер Kubernetes, отвечающий вашим требованиям по производительности и отказоустойчивости
- развернутая *PostgreSQL с базой данных для Global ERP*
- NFS-хранилище

Чтобы получить простейший кластер Kubernetes:

1. установите `kubelet`, `kubeadm` и `kubectl`
2. запустите `kubeadm init` на ведущем хосте для инициализации кластера и получите токен для присоединения других нод к кластеру
3. подключите ведомые хосты при помощи команды `kubeadm join` и токена
4. установите сетевой плагин CNI, такой как `Flannel` или `Calico`.

Подробнее читайте в документации Kubernetes.

### Настройка узла администрирования (jump host)

#### Установка необходимых пакетов

На jump-хост необходимо установить следующие пакеты:

```
sudo apt-get install mc http
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. При создании кластера при помощи `kubeadm init`, такой файл сохраняется по пути `/etc/kubernetes/admin.conf`.

Пример файла:

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ01JUkMwZlZrYXhBTE13RFFZSktywklodmNOQVFFFT
    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:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
client-certificate-data:
↳LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSOtLS0tCk1JSURLVENDQWhHZ0F3SUJBZ01JR04ydFhtNkEzc2N3RFFZSktvWklodmNOQVFFT
client-key-data:
↳LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSU1FcEFJQkFBS0NBUEVBc1A1TXlQWlo3Y1poRmo1S1dtSGkzT29MSXpWd
```

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:
- cluster:
  certificate-authority-data:
↳LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSOtLS0tCk1JSURCVENDQWUyZ0F3SUJBZ01JUkMwZFZrYXhBTE13RFFZSktvWklodmNOQVFFT
  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:
↳LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSOtLS0tCk1JSURLVENDQWhHZ0F3SUJBZ01JR04ydFhtNkEzc2N3RFFZSktvWklodmNOQVFFT
    client-key-data:
↳LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSU1FcEFJQkFBS0NBUEVBc1A1TXlQWlo3Y1poRmo1S1dtSGkzT29MSXpWd
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
```

(continues on next page)



(продолжение с предыдущей страницы)

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				

## Требования к ос для nscli

- GNU/Linux, а именно:
  - Debian 11 и выше
- Python 3.9

## Требование к железу для запуска nscli

- 2 ядра
- 200 Мб оперативной памяти
- 30 Гб свободного места

## 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/4.8.1/gs-ctk-nscli-4.8.1.zip"
unzip -o gs-ctk-nscli-4.8.1.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/initvenv.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 ~/nsccli
```

Запустите мастер создания манифеста рабочего пространства

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

---

**Примечание:** В более современных версиях nsccli доступна команда `./namespace.sh create_namespace`

---

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 сервера
- Путь - путь для подключения тома

---

**Примечание:** В NFS-репозитории будет храниться комплект приложений. Для хранения другой информации, в том числе пользовательских файлов, используются прикладное хранилище Appvolume и другие NFS-хранилища, настраиваемые позже с помощью скрипта `resgroup.sh` в составе утилиты `nsccli` (читайте подробнее в документации `gs-ctk`).

---

Пример работы мастера создания пространства имен:

```
k8sadmin@k8s-terminal02:~/nsccli$ ./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:~/nsccli$
```

---

**Примечание:** В новейших версиях доступна команда:

```
./namespace.sh install_namespace
```

Если команда присутствует, вам будет автоматически предложено ей воспользоваться. После успешного выполнения переходите сразу к пункту «Подготовка комплекта приложений appkit».

Рекомендуем пользоваться этой командой. Она не только развернет пространство имен и под nsctl, как предыдущий способ, но также проверит:

- установку сетевого плагина
- готовность нод к развертыванию контейнеров
- наличие связи между подами
- возможность записи в системное хранилище

Вы можете запустить диагностику кластера отдельно по команде `./namespace.sh diagnose`.

---

---

**Примечание:** Если в вашем кластере используются taints и tolerations, тогда для запуска пода nsctl нужно дополнить шаблон `nscli/profile/namespace/template/deploy.yaml`, добавив туда раздел с tolerations.

Пример:

```
...
spec:
  tolerations:
    - key: "node"
      operator: "Equal"
      value: "first"
      effect: "NoSchedule"
  containers:
  ...
```

Для добавления tolerations к подам, которые создает nsctl, смотри раздел «Хуки»

---

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)
- Образ прикладного решения (applib)
- Профиль с шаблонами конфигурационных файлов (profile)

Опционально в комплект приложений также входит:

- Исходный код для отладки прикладного решения (appsrc)

Комплект можно хранить в виде zip-архивов (globalserver.zip, applib.zip, profile.zip) или в упакованном виде в одноименных папках (globalserver/, applib/, profile/). Во втором случае файлы перед загрузкой на сетевое хранилище самой утилитой запаковываются в архив.

В профиле с шаблонами конфигурационных файлов находятся:

- Конфигурация сервера приложений globalserver/template/config/global3.config.xml
- Конфигурация сборщика телеметрии globalserver/template/config/otel-sdk.config.yaml и globalserver/template/config/otel-globalserver.config.yaml
- Конфигурация менеджера заданий globalscheduler/template/config/quartz.properties
- Конфигурация проектных настроек системных логов globalscheduler/template/config/logback-LoggerContext-ext.xml
- Конфигурация проектных настроек логов сессии globalscheduler/template/config/logback-LoggerContext-session-ext.xml

Если в комплекте при загрузке его на сетевое хранилище, профиль отсутствует, то nscli создаст стандартный профиль, который, обычно, не требует изменений.

Как работать и настраивать файлы с логами можно посмотреть в [документации по логам](#)

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

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

Подготовьте и загрузите в этот каталог дистрибутивы.

```
mv globalserver.zip ~/nscli/workspace/appkit/v1
mv applib.zip ~/nscli/workspace/appkit/v1
```

Подготовьте комплект приложений для работы

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

Утилита создаст, при необходимости, стандартный профиль, упакует комплект приложений, а также создаст контрольные суммы

```
Не найден профиль с шаблонами конфигурации, создать профиль по умолчанию?[да,нет]:да
Загружен файл:globalserver.zip
Загружен файл:profile.zip
Загружен файл:applib.zip
```

## Подготовка комплекта группы groupkit

Обычно комплект группы **не требуется**. Однако в нем могут быть важные для работы сервиса компоненты. Если в комплекте поставки был groupkit, загрузите его с помощью команды:

```
./groupkit.sh push --namespace gs-cluster-k8s --source workspace/groupkit/v1 --  
↪ destination groupkits/v1
```

## 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.

```
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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
values:
- k8s-worker01
```

```
cat <<EOF | tee ~/nsccli/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 ~/nsccli/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 ~/nsccli/workspace/haproxy-sercret.yaml
apiVersion: v1
kind: Secret
metadata:
```

(continues on next page)

(продолжение с предыдущей страницы)

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

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

– При необходимости, создайте ssl-сертификат для haproxy и поместите его в секрет  
Создайте ssl-сертификат с помощью openssl, заменив «example» домены на собственные.

Пример создания ssl-сертификата:

```
openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -nodes \
-keyout ca.key -out ca.crt -subj '/CN=example.com'
```

Пример создания мультидоменного ssl-сертификата:

```
openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -nodes \
-keyout ca.key -out ca.crt -subj '/CN=example.com' \
-addext 'subjectAltName=DNS:example.com,DNS:example.net'
```

Создайте секрет, заменив значения шаблона <cert-b64> и <key-b64> на значения, сгенерированные с помощью следующих команд:

```
cat ./ca.crt | base64
cat ./ca.key | base64
```

```
apiVersion: v1
kind: Secret
metadata:
  name: haproxy-tls
  namespace: gs-cluster-k8s
type: Opaque
data:
  tls.crt: |
    <cert-b64>
  tls.key: |
    <key-b64>
```

```
cat <<EOF | tee ~/nsccli/workspace/haproxy-tls.yaml
apiVersion: v1
kind: Secret
metadata:
  name: haproxy-tls
  namespace: gs-cluster-k8s
type: Opaque
data:
  tls.crt: |
    $(cat ./ca.crt | base64 -w 0)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
tls.key: |
    $(cat ./ca.key | base64 -w 0)
EOF
```

```
kubectl apply -f ~/nsccli/workspace/haproxy-tls.yaml
```

- Создайте секрет с admin аккаунтом globalsever-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 ~/nsccli/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 ~/nsccli/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 ~/nsccli/workspace/db-user-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: db-user-secret
  namespace: gs-cluster-k8s
```

(continues on next page)



(продолжение с предыдущей страницы)

```
type: kubernetes.io/basic-auth
stringData:
  username: <username>
  password: <password>
EOF
```

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

- Создайте секрет с аккаунтом клиентского пользователя RabbitMQ

```
apiVersion: v1
kind: Secret
metadata:
  name: rabbitmq-global
  namespace: gs-cluster-k8s
type: kubernetes.io/basic-auth
stringData:
  username: globalrabbitmq
  password: globalrabbitmq
```

```
cat <<EOF | tee ~/nsccli/workspace/rabbitmq-global.yaml
apiVersion: v1
kind: Secret
metadata:
  name: rabbitmq-global
  namespace: gs-cluster-k8s
type: kubernetes.io/basic-auth
stringData:
  username: globalrabbitmq
  password: globalrabbitmq
EOF
```

```
kubectl apply -f ~/nsccli/workspace/rabbitmq-global.yaml
```

- Создайте секрет с аккаунтом администратора RabbitMQ

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

```
cat <<EOF | tee ~/nsccli/workspace/rabbitmq-admin.yaml
apiVersion: v1
kind: Secret
metadata:
  name: rabbitmq-admin
```

(continues on next page)

(продолжение с предыдущей страницы)

```
namespace: gs-cluster-k8s
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: Secret#123#Pass!
EOF
```

```
kubectl apply -f ~/nsccli/workspace/rabbitmq-admin.yaml
```

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

**Внимание:** Токен планировщика должен быть закодирован в Base64, читайте подробнее [здесь](#).

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

```
cat <<EOF | tee ~/nsccli/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 ~/nsccli/workspace/scheduler-token-secret.yaml
```

- Создайте секрет с аккаунтом администратора Графаны.

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

```
cat <<EOF | tee ~/nsccli/workspace/grafana-admin-secret.yaml
apiVersion: v1
kind: Secret
```

(continues on next page)

(продолжение с предыдущей страницы)

```
metadata:
  name: grafana-admin
  namespace: gs-cluster-k8s
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: admin
EOF
```

```
kubectl apply -f ~/nsccli/workspace/grafana-admin-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 ~/nsccli/workspace/cadvisor-sa.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: prometheus-cadvisor
  namespace: gs-cluster-k8s
EOF
```

```
kubectl apply -f ~/nsccli/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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
verbs: ["get", "list", "watch"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]
```

```
cat <<EOF | tee ~/nsccli/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 ~/nsccli/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 ~/nsccli/workspace/cadvisor-role-binding.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: prometheus-cadvisor
roleRef:
  apiGroup: rbac.authorization.k8s.io
```

(continues on next page)

(продолжение с предыдущей страницы)

```
kind: ClusterRole
name: prometheus-cadvisor
subjects:
- kind: ServiceAccount
  name: prometheus-cadvisor
  namespace: gs-cluster-k8s
EOF
```

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

## Настройка параметров кластера

Параметры кластера настраиваются с помощью пода `nsctl`

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

Получите список подов рабочего пространства

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

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

Получите доступ по ssh в под `nsctl`

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

---

**Совет:** В более новых версиях `nsccli` доступна команда `./namespace.sh shell` для подключения к поду `nsctl`.

---

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
# если вы загружали groupkit, то укажите его следующей командой
# ./resgroup.sh switch_groupkit --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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
./resbook.sh create --name grafana --group gs-cluster-1 --class_name grafana
# ресурс RabbitMQ
./resbook.sh create --name rabbitmq --group gs-cluster-1 --class_name rabbitmq
```

---

**Примечание:** О группах и книгах ресурсов читайте в документации [gs-ctk](#).

---

Сконфигурируйте характеристики

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

Введите необходимые характеристики или оставьте значения по умолчанию

```
Инициализация характеристик для группы ресурсов gs-cluster-1
Установка характеристик для книги ресурсов:global-scheduler
Отслеживать метрики:true
Дополнительно отсылать метрики во внешнюю систему: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:1
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
Дополнительно отсылать метрики во внешнюю систему: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:2
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
Дополнительно отсылать метрики во внешнюю систему: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:2
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:grafana
Введите запрос CPU для grafana:1
```

(continues on next page)

(продолжение с предыдущей страницы)

```
Введите запрос MEMORY для grafana:500M
Введите лимиты CPU для grafana:2
Введите лимиты MEMORY для grafana:1Gi
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
```

**Предупреждение:** Если вы захотите использовать Ingress, прочитайте [статью о том, как это сделать](#).

```
Инициализация значений для группы ресурсов gs-cluster-1
Введите timezone подов:Europe/Moscow
Введите поисковые домены для resolv.conf(если их несколько вводите через пробел):

Введите url базы данных:jdbc:postgresql://dbhost:5432/global
Введите alias базы данных:global
Введите имя секрета для пользователя БД:db-user-secret
Введите тип прикладного хранилища:nfs
Введите адрес сервера(server):127.0.0.1
Введите путь(path):/mnt/nfs/gs-cluster-k8s/globalfilestorage
Хотите добавить больше хранилищ?:нет

Установка значений для книги ресурсов:global-scheduler
Введите адрес доступа к prometheus:kube-gs-resgoup-grafana-internal:9090
Введите адрес доступа к loki:kube-gs-resgoup-grafana-internal:3100
Введите адрес доступа к tempo:kube-gs-resgoup-grafana-internal:3201
Введите адрес доступа к внешнему prometheus:external.prometheus.domain:9090
Введите адрес доступа к внешнему loki:external.loki.domain:3100
Введите адрес доступа к внешнему tempo:external.tempo.domain:3201
Введите имя секрета с токеном планировщика:scheduler-token-secret

Установка значений для книги ресурсов:global-server-excl
Введите адрес доступа к prometheus:kube-gs-resgoup-grafana-internal:9090
Введите адрес доступа к loki:kube-gs-resgoup-grafana-internal:3100
Введите адрес доступа к tempo:kube-gs-resgoup-grafana-internal:3201
Введите адрес доступа к внешнему prometheus:external.prometheus.domain:9090
Введите адрес доступа к внешнему loki:external.loki.domain:3100
Введите адрес доступа к внешнему tempo:external.tempo.domain:3201
Введите внешний ip(external_ip):127.0.0.1
Введите имя секрета для администратора:gs-admin-auth

Установка значений для книги ресурсов:global-server-share
Введите количество экземпляров:3
Введите адрес доступа к prometheus:kube-gs-resgoup-grafana-internal:9090
```

(continues on next page)

(продолжение с предыдущей страницы)

```
Введите адрес доступа к loki:kube-gs-resgoup-grafana-internal:3100
Введите адрес доступа к tempo:kube-gs-resgoup-grafana-internal:3201
Введите адрес доступа к внешнему prometheus:external.prometheus.domain:9090
Введите адрес доступа к внешнему loki:external.loki.domain:3100
Введите адрес доступа к внешнему tempo:external.tempo.domain:3201
Введите имя секрета для администратора:gs-admin-auth

Установка значений для книги ресурсов:grafana
Использовать Ingress?[да,нет]:нет
Введите внешний ip(external_ip):127.0.0.1
Введите класс хранилища:nfs-storage
Введите размер хранилища:5Gi
Введите имя системного пользователя с правами на доступ к метрикам cAdvisor:prometheus-
↪cAdvisor
Введите имя секрета с аккаунтом администратора Grafana:grafana-admin

Установка значений для книги ресурсов:haproxy
Использовать Ingress?[да,нет]:нет
Введите внешний ip(external_ip):127.0.0.1
Введите имя секрета basic-auth для авторизации статистики:secret-haproxy-auth
Введите имя tls секрета для доступа по https:

Установка значений для книги ресурсов:rabbitmq
Введите название секрета для доступа к RabbitMQ:rabbitmq-global
Введите название секрета для доступа к консоли RabbitMQ:rabbitmq-admin
Введите название создаваемого виртуального хоста RabbitMQ:globalrabbitmq
Использовать Ingress?[да,нет]:нет
Введите внешний ip(external_ip):10.40.1.100
Введите внешний порт:15672
Подключить RabbitMQ?[да,нет]:да
Введите адрес RabbitMQ:gs-cluster-1-rabbitmq-internal.gs-k8s.svc.cluster.local
Введите порт RabbitMQ:5672
Введите виртуальный хост RabbitMQ:globalrabbitmq
Введите секрет RabbitMQ:db-user-secret
```

---

**Совет:** В последних версиях nsctl доступна команда `./tester.sh test-write --resgroup gs-cluster-1`, которую можно использовать для проверки записи во все используемые группой ресурсов хранилища.

---

При использовании ssl в haproxy укажите имя секрета, содержащего сертификат:

```
Введите имя tls секрета для доступа по https: haproxy-tls
```



## Включение книг и групп ресурсов

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
```

## Установка схемы базы данных

На этом этапе сервер приложений должен быть доступен для открытия через браузер. Однако войти не получится - не проинициализирована база данных.

```
./appkit.sh switch_und_upgrade --namespace gs-cluster-k8s --resgroup gs-cluster-1 --
↪remote_appkit appkits/v1
./invoker.sh run --namespace gs-cluster-k8s --app nsctl --cmd "./resgroup.sh start_
↪appkit --name gs-cluster-1"
```

## Получение и установка лицензии

Процесс получения и установки лицензии описан в *документации по первому входу в систему*.

### Что дальше?

- Подумайте о настройке комплекта группы, если вы используете собственные сертификаты
- Настройте *JGroups*
- Публикуйте через *Ingress*
- Проводите регулярные *обновления* прикладного решения и образов контейнеров

## 5.3 Неуправляемый режим

Утилита `nscli` позволяет развернуть кластер в неуправляемом режиме. Так вместо создания пространства имен и развертывания управляющего пода `nsctl`, `nscli` создаст уже готовый пакет, описывающий все группы и книги ресурсов, который затем уже можно развернуть на кластере при помощи `kubectl`.

Мы рекомендуем пользоваться *управляемым режимом*, но вам может быть необходимо работать с кластером Kubernetes напрямую с минимальным задействованием утилит Global.

## Сравнение управляемого и неуправляемого режима

Критерий	Управляемый режим	Неуправляемый режим
Степень автоматизации	Все основные задачи, необходимые для развертывания и поддержки Global ERP в кластере, автоматизированны	Автоматизировано лишь создание ресурсов, описывающих контейнеры и конфигурацию решения
Сложность администрирования	Работа с нативными средствами K8s сведена к минимуму, используются удобные обертки	Для использования требуется глубокое понимание K8s.
Обновление	Есть налаженные команды обновления решения и работы с сетевыми хранилищами	Утилита в неуправляемом режиме не обеспечивает доставки комплекта приложений, как и обновления БД
Вероятность ошибок	Средства диагностики и другие предохранители сопровождают пользователю во время пользования утилитой, что уменьшает количество ошибок и упрощает их отладку	Предохранители действуют только во время работы с конфигурационными файлами. Встроенных средств диагностики меньше, чем при управляемом режиме

## Развертывание в неуправляемом режиме

Инструкция дана для:

- рабочего кластера Kubernetes из одной control-plane ноды (10.0.1.1) и трех рабочих нод с 6ГБ ОЗУ и установленным дистрибутивом Debian;
- двух NFS-хранилищ 10.0.2.1 с точками монтирования `/nfs/sys` и `/nfs/app`;
- PostgreSQL-сервера по адресу 10.0.3.1 с *подготовленной* БД `global`;
- jump-хоста с доступом к Kubernetes.

---

**Совет:** Доступ к кластеру для хоста, на котором будут генерироваться ресурсы K8s при помощи `nscli`, необязателен.

---

1. Установите утилиту `nscli` в соответствии с *инструкцией*.
2. Подготовьте папку с *комплект приложений (appkit)*. Поместите её на машину с утилитой, например, по пути `~/nscli/workspace/appkit`.
3. Подготовьте *ресурсы секретов и роли для мониторинга*.
4. Выполните в папке `nscli`, чтобы получить (изменить) конфигурацию неуправляемого режима:

```
./namespace.sh configure_unmanaged --config workspace/config.yaml
```

В ходе настройки укажите названия, характеристики и настройки пространства имен, групп и книг ресурсов.

Утилита попросит у вас путь к папке с комплектом приложений (`appkit`), после ввода утилита упакует комплект, пересчитает и поместит в папке его хеш-суммы, после чего сохранит их в конфигурационном файле. Чтобы переупаковать комплект и пересчитать хеш-функции в неинтерактивном режиме используйте команду `./appkit.sh refresh_hash --source`.

---

**Совет:** В некоторых полях, например, при выборе класса книги ресурсов, можно использовать стрелочки «вверх»-«вниз» для выбора из часто встречаемых вариантов.

---

Пример:

```
k8sadmin@k8s-terminal02:~/nsccli$ ./namespace.sh configure_unmanaged --config ↵
↵workspace/config.yaml
Введите имя рабочего пространства:gs-cluster-k8s
Введите адрес докер регистра:dockerhub.global-system.ru
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:10.0.2.1
Введите путь:/nfs/sys
Добавить группу gs-cluster-1?[да,нет]:да
Введите название группы ресурсов 1 (или оставьте пустым, чтобы приступить к ↵
↵настройке групп):gs-cluster-1
Введите название группы ресурсов 2 (или оставьте пустым, чтобы приступить к ↵
↵настройке групп):
Инициализация значений для группы ресурсов gs-cluster-1
Введите timezone подов:Europe/Moscow
Введите url базы данных:jdbc:postgresql://10.0.3.1:5432/global
Введите alias базы данных:global
Введите имя секрета для пользователя БД:db-user-secret
Введите тип прикладного хранилища:nfs
Введите адрес сервера(server):10.0.0.2
Введите путь(path):/nfs/app
Хотите загрузить метаданные appkit из папки?[да,нет]:да
Укажите путь к папке с appkit:workspace/appkit
Метаданные appkit загружены
Укажите хеш globalserver:9828f2ebb9979e9e9765e8fe7591557bd15d0c3a
Укажите хеш applib:56451d21e04b84fad0a87a3867d7bdbe91f464f5
Укажите хеш profile:fb2add3bb36ebd219da754352ccdf0c78143e2fc
Укажите путь к папке с appkit на системном хранилище (NFS):appkit/v1
Внимание! В неуправляемом режиме вы обязаны сами поместить appkit на системное ↵
↵хранилище по указанному пути
Укажите состояние appkit:started
Введите путь к папке с комплектом группы (groupkit) на системном хранилище (NFS); ↵
↵оставьте пустым, если он не требуется:
Введите название книги ресурсов 1 в группе gs-cluster-1 (или оставьте пустым, чтобы ↵
↵закончить настройку книг ресурсов):global-scheduler
Введите класс книги ресурсов global-scheduler в группе gs-cluster-1:global_scheduler
Добавить книгу ресурсов global-scheduler?[да,нет]:да
Настройка книги ресурсов global-scheduler...
Отслеживать метрики:true
Дополнительно отсылать метрики во внешнюю систему:false
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:1
```

(continues on next page)

```

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
Введите адрес доступа к prometheus:gs-cluster-1-grafana-internal:9090
Введите адрес доступа к loki:gs-cluster-1-grafana-internal:3100
Введите адрес доступа к tempo:gs-cluster-1-grafana-internal:3201
Введите имя секрета с токеном планировщика:scheduler-token-secret
Введите название книги ресурсов 2 в группе gs-cluster-1 (или оставьте пустым, чтобы
↪закончить настройку книг ресурсов):global-server-excl
Введите класс книги ресурсов global-server-excl в группе gs-cluster-1:global_server_
↪excl
Отслеживать метрики:true
Дополнительно отсылать метрики во внешнюю систему:false
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:2
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
Введите адрес доступа к prometheus:gs-cluster-1-grafana-internal:9090
Введите адрес доступа к loki:gs-cluster-1-grafana-internal:3100
Введите адрес доступа к tempo:gs-cluster-1-grafana-internal:3201
Введите внешний ip(external_ip):
Введите имя секрета для администратора:gs-admin-auth
Введите тип секрета для администратора:kubernetes.io/basic-auth
Введите название книги ресурсов 3 в группе gs-cluster-1 (или оставьте пустым, чтобы
↪закончить настройку книг ресурсов):global-server-share
Введите класс книги ресурсов global-server-share в группе gs-cluster-1:global_
↪server_share
Отслеживать метрики:true
Дополнительно отсылать метрики во внешнюю систему:false
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:2
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
Введите количество экземпляров:2
Введите адрес доступа к prometheus:gs-cluster-1-grafana-internal:9090
Введите адрес доступа к loki:gs-cluster-1-grafana-internal:3100
Введите адрес доступа к tempo:gs-cluster-1-grafana-internal:3201
Введите имя секрета для администратора:gs-admin-auth
Введите тип секрета для администратора:kubernetes.io/basic-auth
Введите название книги ресурсов 4 в группе gs-cluster-1 (или оставьте пустым, чтобы
↪закончить настройку книг ресурсов):grafana

```

(continues on next page)

```

Введите класс книги ресурсов grafana в группе gs-cluster-1:grafana
Настройка книги ресурсов grafana...
Введите запрос CPU для grafana:1
Введите запрос MEMORY для grafana:500M
Введите лимиты CPU для grafana:2
Введите лимиты MEMORY для grafana:1Gi
Введите внешний ip(external_ip):10.0.1.1
Введите класс хранилища:local-storage
Введите размер хранилища:1Gi
Введите имя системного пользователя с правами на доступ к метрикам
↪cAdvisor:prometheus-cadvisor
Введите имя секрета с аккаунтом администратора Grafana:grafana-admin
Введите название книги ресурсов 5 в группе gs-cluster-1 (или оставьте пустым, чтобы
↪закончить настройку книг ресурсов):haproxy
Введите класс книги ресурсов haproxy в группе gs-cluster-1:haproxy
Настройка 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
Введите внешний ip(external_ip):10.0.1.1
Введите внешний порт(external_port):8081
Введите имя секрета basic-auth для авторизации статистики:secret-haproxy-auth
Введите имя tls секрета для доступа по https:
Добавить книгу ресурсов rabbitmq?[да,нет]:да
Введите название книги ресурсов 6 в группе gs-cluster-1 (или оставьте пустым, чтобы
↪закончить настройку книг ресурсов):rabbitmq
Введите класс книги ресурсов rabbitmq в группе gs-cluster-1:rabbitmq
Настройка книги ресурсов rabbitmq...
Введите запрос CPU для rabbitmq:1
Введите запрос MEMORY для rabbitmq:2Gi
Введите лимиты CPU для rabbitmq:2
Введите лимиты MEMORY для rabbitmq:4Gi
Введите название секрета для доступа к RabbitMQ:rabbitmq-global
Введите название секрета для доступа к консоли RabbitMQ:rabbitmq-admin
Введите название создаваемого виртуального хоста RabbitMQ:globalrabbitmq
Введите внешний ip(external_ip):10.0.1.1
Введите внешний порт:15672
Введите название книги ресурсов 7 в группе gs-cluster-1 (или оставьте пустым, чтобы
↪закончить настройку книг ресурсов):
Подключить RabbitMQ?[да,нет]:да
Введите адрес RabbitMQ:gs-cluster-1-rabbitmq-internal.gs-cluster-k8s.svc.cluster.
↪local
Введите порт RabbitMQ:5672
Введите виртуальный хост RabbitMQ:globalrabbitmq
Введите секрет RabbitMQ:rabbitmq-global

```

5. Поместите упакованный комплект приложений (находится по тому же пути ~/nsccli/workspace/appkit) на системное (NFS-)хранилище, по указанному на предыдущем этапе пути (в нашем случае /nfs/sys/appkit/v1/).
6. Создайте ресурсы с пространством имен, группами и книгами:

```
./namespace.sh create_unmanaged_namespace --config workspace/config.yaml --  
↪ namespace-deploy workspace/namespace.yaml # пространство имен  
./namespace.sh create_unmanaged_resources --config workspace/config.yaml --  
↪ resources workspace/resources.yaml # группы и книги ресурсов
```

7. Последовательно примените ресурс пространства имен, секреты, роли и, наконец, группы и книги ресурсов:

```
kubectl apply -f workspace/namespace.yaml  
# kubectl apply -f secrets.yaml  
# kubectl apply -f roles.yaml  
kubectl apply -f workspace/resources.yaml
```

8. Проверьте, что кластер исправно работает:

- просмотрите список развернутых подов при помощи `kubectl get -A pods` на наличие созданных ресурсов и проверьте, что они работают;
- попробуйте подключиться к веб-интерфейсу через браузер по указанному для HAProxy адресу.

В случае неисправностей проанализируйте логи и метрики при помощи Grafana.

---

**Совет:** Поды автоматически перезапускаются при критических ошибках, что мешает диагностике неисправностей. Чтобы остановить перезапуск включите *режим отладки*.

---

**Внимание:** Вы можете делать правки в конфигурации, а затем генерировать и применять новые ресурсы с этими правками. Однако обратите внимание, что в неуправляемом режиме при использовании `kubectl` удаление уже развернутых ресурсов при помощи `yaml`-манифеста почти невозможно. Следовательно, такие действия, как выключение Ingress в конфигурации (что требует удаление ресурса типа Ingress в Kubernetes), не будут отображены на кластере при применении новых ресурсов. Поэтому после получения верной конфигурации рекомендуется удалить пространство имен и применить ресурсы снова.

## Шифрование настроек учетной записи от реестра образов Docker

Поскольку реквизиты от реестра образов - это чувствительная информация, она хранится в конфигурации в зашифрованном виде. Следовательно, чтобы работать с ними нужен мастер-ключ, хранящийся по пути `~/gs-ctk.priv`, и настройки шифрователя по пути `(~/nsccli/)workspace/cryptor_state.json`.

Обратите внимание, что мастер-ключ не требуется для работы с группами и книгами ресурсов. Вы можете сказать утилите не расшифровывать реквизиты докер при помощи параметра `--no-cryptor`.

```
./namespace.sh configure_unmanaged --no-cryptor --config cfg.yaml  
./namespace.sh create_unmanaged_resources --no-cryptor --config cfg.yaml --resources m.  
↪ yaml  
./namespace.sh create_unmanaged_namespace --config cfg.yaml --namespace-deploy ns.yaml #  
↪ невозможно сформировать пространство имен без правильного мастер-ключа
```

## Перенос конфигурации

В последних версиях утилит, *экспорт/импорт конфигурации* управляемого режима полностью совместим с конфигурацией неуправляемого режима.

Чтобы перенести конфигурацию, выполните:

```
./configmgr.sh export --namespace gs-cluster-1 --file workspace/config.yaml
# конфигурация gs-cluster-1 сохранена в файл workspace/config.yaml

./namespace.sh configure_unmanaged --config workspace/config.yaml
# укажите настройки реестра образов и внесите другие необходимые изменения, например,
↪ поменяйте название пространства имен
```

Теперь вы можете использовать (`~/nscli/`)`workspace/config.yaml`, как конфигурацию неуправляемого режима.

## Обновление комплекта приложений

Поскольку в неуправляемом режиме утилита `nscli` не имеет доступа к кластеру, вам придется своими руками выполнить процедуру, которую обычно при обновлении выполняет утилита:

1. Поместите новый комплект приложений в папку (`~/nscli/`)`workspace/appkit`.
2. Переведите комплект приложений в состояние `stopped`. Для этого, измените в конфиге, который вы получили при помощи команды `./namespace.sh configure_unmanaged` (в нашем примере (`~/nscli/`)`workspace/config.yaml`) значение состояния (`spec.appkit.state`) у необходимой группы ресурсов на указанное.

```
config_version: 1
resgroups:
- name: gs-cluster-1
  spec:
    appkit:
      path: appkit/v1
      state: stopped
      applib_sha1: 56451d21e04b84fad0a87a3867d7bdbe91f464f5
      ...
  values:
    ...
```

Создайте и примените новые ресурсы:

```
./namespace.sh create_unmanaged_resources --config workspace/config.yaml --
↪resources workspace/resources.yaml
kubectl apply -f workspace/resources.yaml
```

3. Запустите редактор конфигурации:

```
./namespace.sh configure_unmanaged --config workspace/config.yaml
```

Обновите хеши комплекта приложений и установите ему состояние `drained`, когда пользователи не подключается, но сервер приложений работает.

---

**Совет:** Чтобы утилита сама подсчитала хеши, укажите в поле «Хотите загрузить метаданные appkit из папки?» значение «да», затем укажите папку с новым копмлектом приложений (~/.nscli/)workspace/config.yaml.

---

```
k8sadmin@k8s-terminal02:~/nscli$ ./namespace.sh configure_unmanaged --config_
↪workspace/config.yaml
...
Хотите загрузить метаданные appkit из папки?[да,нет]:да
Укажите путь к папке с appkit:workspace/appkit
Метаданные appkit загружены
Укажите хеш globalserver:de014bfd3604100ea19b4b5a6104e789fe136692
Укажите хеш applib:ca4118be45efc4509e80f9cb97279399d697ca01
Укажите хеш profile:18ddc5f7762ae21b3789692f304b4a4e35c8f115
Укажите путь к папке с appkit на системном хранилище (NFS):appkit/v2
Внимание! В неуправляемом режиме вы обязаны сами поместить appkit на системное_
↪хранилище по указанному пути
Укажите состояние appkit:drained
...
```

---

**Совет:** Обратите внимание, что новый appkit хранится по пути appkit/v2, а не appkit/v1.

---

4. Скопируйте новый комплект приложений с хешами (workspace/appkit) на системное NFS-хранилище по указанному пути (appkit/v2).
5. Примените новую конфигурацию.

```
./namespace.sh create_unmanaged_resources --config workspace/config.yaml --
↪resources workspace/resources.yaml
kubectl apply -f workspace/resources.yaml
```

6. Обновите базу данных. Для этого *подключитесь к поду* с эксклюзивным экземпляром сервера приложений (global-server-excl) и выполните на нем команду:

```
./migrator.sh upgrade
```

---

**Совет:** Вы также можете воспользоваться командой ./invoke.sh в составе nscli, если у утилиты есть доступ к кластеру.

```
./invoke.sh --namespace [namespace] --app [resgroup]-[resbook] --cmd './migrator.
↪sh upgrade'
```

, где [namespace] - пространство имен, [resgroup] - группа ресурсов, [resbook] - книга ресурсов класса global-server-excl

---

Проверьте, что вывод генератора таблиц не сообщает об ошибках.

7. Переведите новый комплект приложений в состояние **started** аналогично шагу 2.

Обновление должно быть завершено, а сервер приложений должен принимать пользователей.



## Обновление на новую версию gs-ctk

Обновление до новой версии gs-ctk не отличается от развертывания в первый раз за тем исключением, что начинается оно с редактирования уже имеющегося у вас конфигурационного файла. Ниже мы рассмотрим самый простой вариант обновлением лишь с использованием nscli и kubectl.

---

**Совет:** При необходимости адаптируйте этот вариант обновления согласно вашим требованиям по развертыванию. Читайте подробнее в документации Kubernetes по поводу существующих средств [управления ресурсами](#) и [обновления контейнеров](#).

---

1. Запустите редактор конфигурации и внесите нужные изменения в уже имеющийся конфигурационный файл:

```
./namespace.sh configure_unmanaged workspace/config.yaml
```

Скорее всего, изменения не потребуются, но данный шаг гарантирует, что будут указаны все необходимые для развертывания настройки.

2. Удалите пространство имен в Kubernetes:

```
kubectl delete namespace gs-cluster-k8s
```

**Внимание:** Простого пересоздания ресурсов без удаления старого пространства имен будет недостаточно, так как от предыдущей версии могут остаться ресурсы, признанные устаревшими в новой версии.

---

**Совет:** Если вы не можете удалить пространство имен (например, из-за отсутствия прав), но у вас есть yaml-манифест с развернутой версией ресурсов Kubernetes, вы можете использовать команду `kubectl delete -f` для удаления ресурсов. Тогда вам не придется удалять пространство имен, секреты и роли при обновлении.

```
kubectl delete -f workspace/resources.yaml
```

По причинам выше рекомендуется все же удалить пространство имен, если такая возможность есть.

---

3. Сгенерируйте ресурсы заново:

```
./namespace.sh create_unmanaged_namespace --config workspace/config.yaml --  
↪ namespace-deploy workspace/namespace.yaml  
./namespace.sh create_unmanaged_resources --config workspace/config.yaml --  
↪ resources workspace/resources.yaml
```

И примените их:

```
kubectl apply -f workspace/namespace.yaml  
# kubectl apply -f secrets.yaml  
# kubectl apply -f roles.yaml  
kubectl apply -f workspace/resources.yaml
```

## 5.4 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 в управляемом режиме:

```
./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
```

Чтобы включить режим отладки в неуправляемом режиме, вручную измените в ресурсе соответствующего пода значение переменной окружения `IS_DEBUG_ENABLED` на `true`.

## 5.5 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 кubernetes
- `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/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"
```

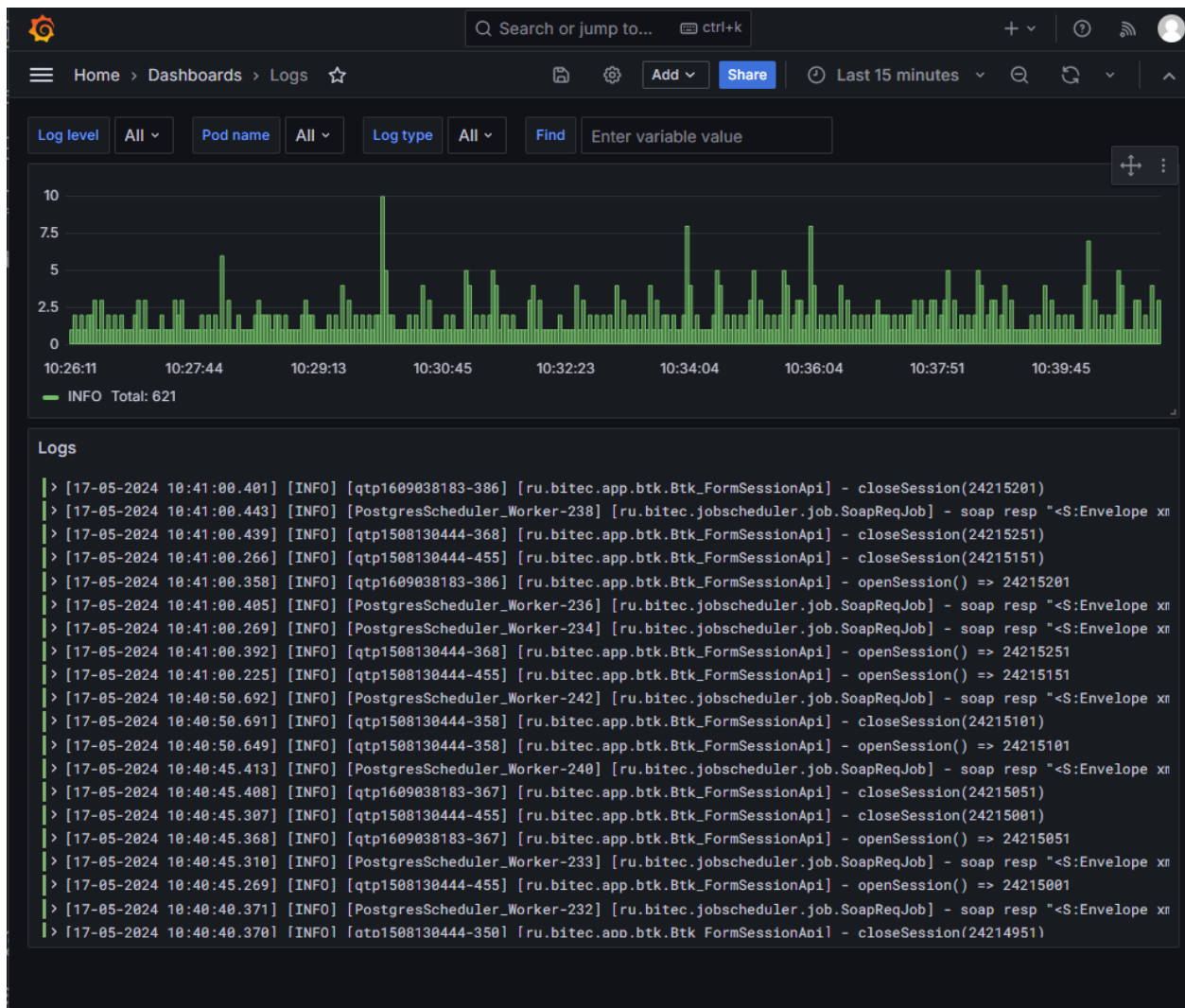
## Логи кластера

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



## 5.6 Хуки

Для каждого пода, создаваемого с помощью `nsctl`, есть список доступных хуков.

Список хуков:

- `tolerations` - добавление `tolerations` для пода

### Управление хуками книг в управляемом режиме

#### Описание аргументов:

- `--name` название хука
- `--group` название ресгруппы
- `--book` название ресбука
- `--hook_file` название файла с содержанием хука

## Добавить хук

Для добавление хука к книге, необходимо выполнить следующие действия

- Создать файл с содержанием хука, например:

```
cat <<EOF | tee ./tolerations.yaml
tolerations:
  - key: "node"
    operator: "Equal"
    value: "first"
    effect: "NoSchedule"
EOF
```

- Добавить хук к нужной вам книге:

```
./resbook.sh set_hook --name tolerations --group gs-cluster-1 --book global-server-
↪excl --hook_file ./tolerations.yaml
```

## Удалить хук

```
./resbook.sh unset_hook --name tolerations --group gs-cluster-1 --book global-server-excl
```

## Посмотреть список хуков

```
./resbook.sh list_hooks --group gs-cluster-1 --book global-server-excl
```

## Управление хуками книг в неуправляемом режиме

### Описание аргументов:

- --name - название хука
- --group - название ресгруппы
- --book - название ресбука
- --hook\_file - название файла с содержанием хука
- --config - название файла конфига

## Добавить хук

Для добавление хука к книге, необходимо выполнить следующие действия

- Создать файл с содержанием хука, например:

```
cat <<EOF | tee ./tolerations.yaml
tolerations:
  - key: "node"
```

(continues on next page)

(продолжение с предыдущей страницы)

```
operator: "Equal"
value: "first"
effect: "NoSchedule"
EOF
```

- Добавить хук к нужной вам книге:

```
./namespace.sh set_unmanaged_hook --name tolerations --group gs-cluster-1 --book_
↪global-server-sxcl --hook_file tolerations.yaml --config ./config.yaml
```

### Удалить хук

```
./namespace.sh unset_unmanaged_hook --name tolerations --group gs-cluster-1 --book_
↪global-server-excl --config ./config.yaml
```

### Посмотреть список хуков

```
./namespace.sh list_unmanaged_hooks --group gs-cluster-1 --book global-server-excl --
↪config ./config.yaml
```

## 5.7 Отладка прикладного решения

**Предупреждение:** Экспериментальная функция, работает только на Java 21

Иногда программную ошибку не получается воспроизвести на тестовых стендах. В таких случаях, gs-ctk позволяет создать среду для отладки прикладного решения в условиях, максимально приближенных к тем, в которых работает основной кластер.

Для этого создается под, в котором стартует тот же сервер приложений, что и обычно, но с включенной подсистемой отладки Java. В поде также есть дополнительный сопровождающий контейнер со средой OpenVSCode Server, подключенной к данной подсистеме. Обычные пользователи не могут подключиться к данному поду без особой ссылки.

### Инструкция по отладке

Требования:

- gs-ctk версии  $\geq 5.0$
- настроенная работающая группа ресурсов с включенными книгами ресурсов `global_server_share` и `haproxy`
- наличие в комплекте приложений архива `appsrc.zip`, содержащего исходные файлы проекта.

1. Администратор включает отладку при помощи команды `nsctl`:

```
~/nsctl $ ./cloud_debugger.sh start --group [имя группы ресурсов]
```

```
Выберите книгу ресурсов, которую необходимо взять за основу облачного
↪отладчика:global-server-share
Введите запрос CPU для debugger:2
Введите запрос MEMORY для debugger:2G
Введите лимиты CPU для debugger:4
Введите лимиты MEMORY для debugger:12G
Желаєте посмотреть или изменить характеристики и значения книги ресурсов отладчика?
↪[да,нет]:нет
Укажите максимальную длительность работы облачного отладчика в секундах:3600
Запускаем облачный отладчик!
Облачный отладчик будет доступен до 10:25:54 UTC 19.03.2025
Ожидаем готовности облачного отладчика...
Ожидаем готовности облачного отладчика...
Ожидаем готовности облачного отладчика...
Облачный отладчик готов к использованию!

Для доступа к отладчику перейдите по ссылке:
(http://example.ru)/debugger/open/gs-cluster-1-debugger-1b0e4084?debugger%2Fvscode
↪%2F%3Ftkn%3Db0211cf-cd8a-4ba5-b2a2-a5d1791e587b%26folder%3D%2Fuser%2Fapplication

Чтобы подключиться к серверу приложений, связанному с отладчиком, перейдите по
↪ссылке:
(http://example.ru)/debugger/open/gs-cluster-1-debugger-1b0e4084?login%2Flogin.html
```

---

**Совет:** Команду можно запустить в неинтерактивном режиме, читайте подробнее справку `./cloud_debugger.sh start --help`

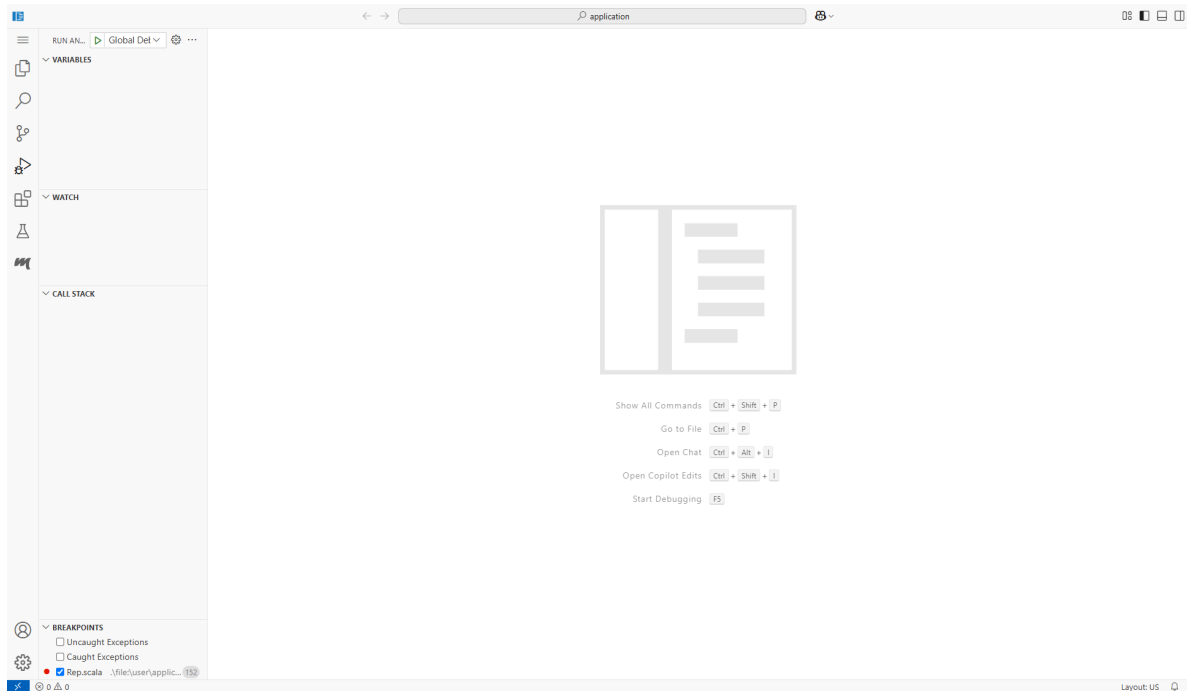
---

Подстроку (`https://example.ru`) надо подменить на имя хоста, к которому обращается пользователь для подключения к HAProxy. Например:

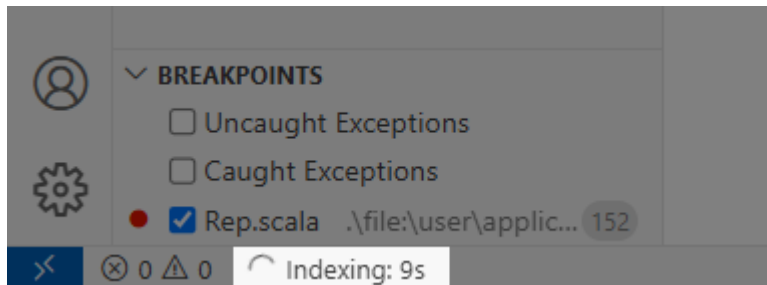
```
https://globalerp.mycompany.ru/debugger/open/gs-cluster-1-debugger-1b0e4084?debugger
↪%2Fvscode%2F%3Ftkn%3Db0211cf-cd8a-4ba5-b2a2-a5d1791e587b%26folder%3D%2Fuser
↪%2Fapplication
```

- Администратор передает ссылки лицам, проводящим отладку (далее - разработчикам).
- Разработчик переходит в отладчик и на сервер приложений по ссылкам.

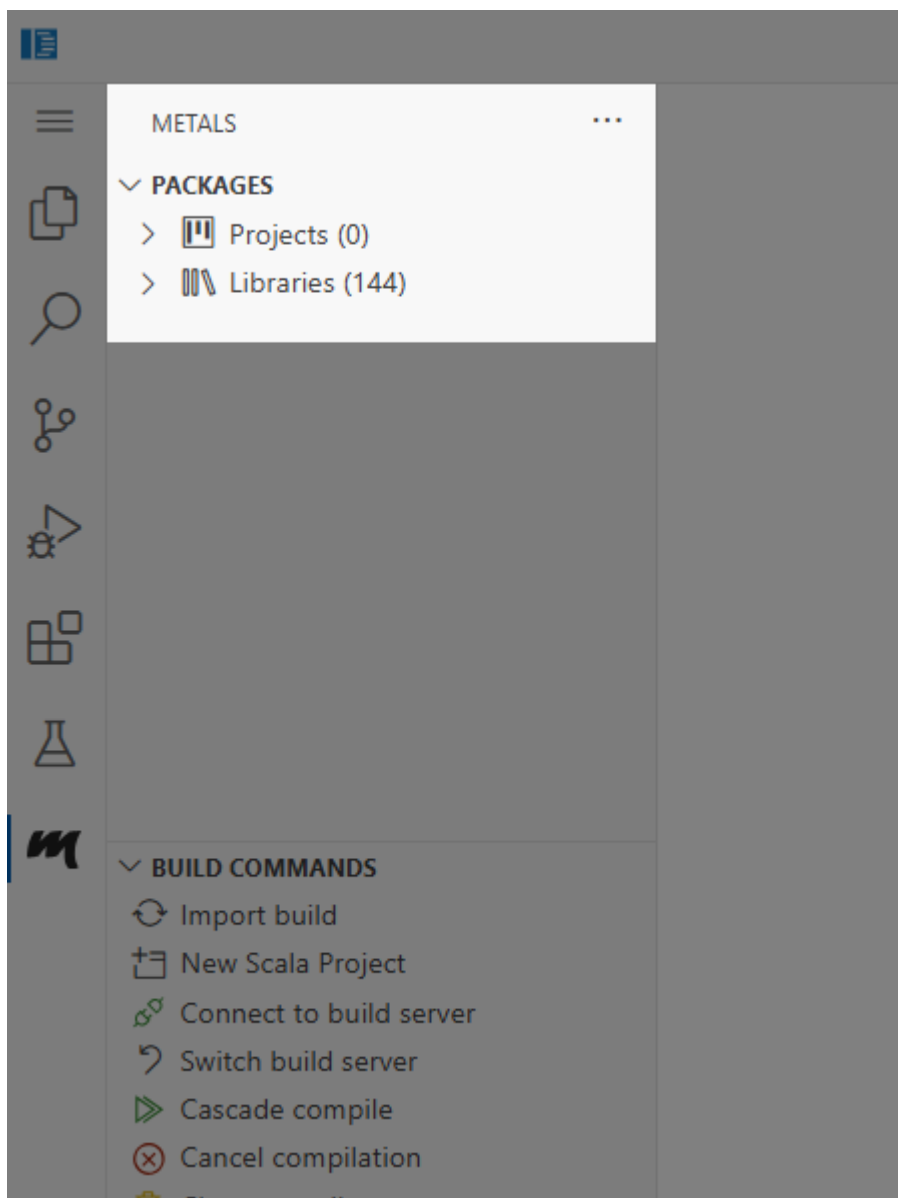




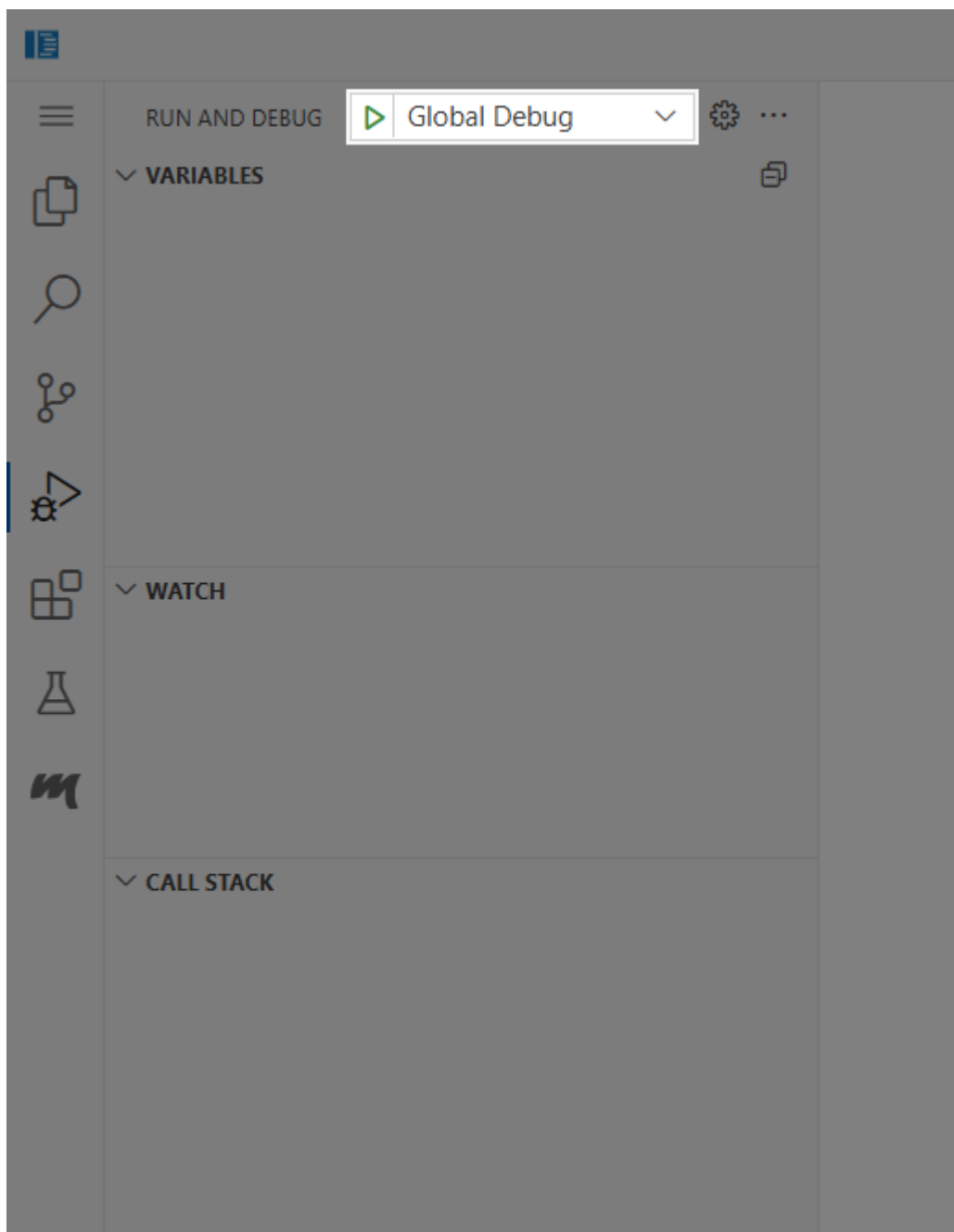
4. Разработчик подтверждает, что он находится в доверенном каталоге `/user/application`, и переходит во вкладку Metals на боковой панели.
5. В течении 30 секунд должна начаться индексация проекта. Дождитесь завершения. После этого в верхней части вкладки Metals появится список используемых модулей и библиотек.



Если индексация не началась, или после ее завершения модули не появились, нажмите «Import build» в секции «Build Commands» вкладки Metals.



6. Разработчик открывает нужные модули или библиотеки, устанавливает точки останова, кликая слева от номера нужной строки.
7. Разработчик переходит на вкладку Run and Debug боковой панели, нажимает кнопку «Start Debugging».



Отладчик включен и подключен к серверу приложений, интерфейс которого доступен через вторую ссылку. Лог сервера приложений доступен по адресу `/user/application/globalserver.log`.

## Неуправляемый режим

Облачный отладчик доступен как класс книги ресурсов `debugger`. Добавьте эту книгу в конфигурацию неуправляемого режима при помощи `./namespace.sh configure_unmanaged` (читайте подробнее в [статье о неуправляемом режиме](#)), затем создайте и примените новые ресурсы. Ссылки для подключения, можно получить, выполнив в контейнере отладчика команду `./service.sh get_urls`.

## Безопасность

В ссылку для отладчика вписан токен для подключения, который используется для авторизации в OpenVSCode Server. Следовательно, без этой ссылки подключиться к отладчику невозможно.

В будущих версиях, способ авторизации может измениться.

OpenVSCode Server работает в отдельном контейнере от имени непривилегированного пользователя, не имеет прямого доступа к секретам и файловым системам других контейнеров.

Доступ к логам и подсистеме отладки предоставляет пользователю OpenVSCode Server широкие возможности по взаимодействию с системой. В будущем они будут ограничены специальным прокси-сервером протокола отладки.

## 5.8 Экспорт конфигурации. Восстановление кластера

Скрипт `configmgr.sh` в составе `pscli` предоставляет возможность экспорта и импорта конфигурации в `yaml`-файл на `jump`-хосте, что может быть полезно, если по какой-либо причине кластер потребовалось установить заново.

Такой файл содержит лишь значения, указанные пользователем при помощи `psctl`, например, объемы вычислительных мощностей, настройки мониторинга и базы данных. Секреты, комплект приложений, постоянные тома не копируются, однако *названия* используемых секретов и постоянных томов экспортируются.

### Экспорт

Экспорт конфигурации осуществляется при помощи команды:

```
./configmgr.sh export
--namespace NAMESPACE # пространство имен, в котором развернут GlobalServer
--resgroup GROUP       # группа ресурсов, которую необходимо экспортировать
--file FILE            # путь к файлу, в который будет записана конфигурация кластера
```

### Импорт

Импорт конфигурации осуществляется при помощи команды:

```
./configmgr.sh import
--namespace NAMESPACE # пространство имен, в котором (будет) развернут GlobalServer
--file FILE            # путь к YAML-файлу, из которого будет загружена конфигурация
↳ кластера
--keep-appkit          # исключает комплект приложений из переносимых значений, если
↳ производится перезапись группы ресурсов (комплект остается тот, что был изначально на
↳ кластере)
```

## Пример восстановления кластера

Предположим, что у нас был кластер GlobalServer, развернутый в группе ресурсов **gs-cluster** пространства имен **gs-k8s**, который более недоступен. Однако, есть YAML-описания ресурсов Kubernetes (секреты, постоянные тома, сервисные пользователи и роли - все, что было создано на этапе установки), а также есть конфигурация сервера, заранее экспортированная при помощи команды:

```
cd ~/ncli
./configmgr.sh export \
  --namespace gs-k8s \
  --resgroup gs-cluster \
  --file config.yaml
```

Тогда процесс восстановления на чистый K8s с подготовленного jump-хоста (следуйте *инструкции по установке кластера* до раздела «Установка утилиты Ncli» включительно) выглядит следующим образом:

1. Воссоздаем пространство имен при помощи команд:

```
./namespace.sh create_install_scripts
chmod +x ~/ncli/workspace/install_scripts/gs-k8s/install.sh
~/ncli/workspace/install_scripts/gs-k8s/install.sh
```

Подробнее читайте в разделе «Настройка рабочего пространства в kubernetes» инструкции по установке.

2. Если в NFS-хранилище не осталось комплекта приложений, то загружаем его в соответствии с «Подготовка комплекта приложений appkit».
3. Устанавливаем ресурсы K8s, например, при помощи команды:

```
cd k8s-resources # директория с ресурсами k8s
kubectl apply -f *
```

4. Импортируем конфигурацию:

```
cd ~/ncli
./configmgr.sh import \
  --namespace gs-k8s \
  --file config.yaml
```

Скрипт сообщит об успешном завершении:

```
Конфигурация импортирована
```

5. Запускаем кластер:

```
cd ~/ncli
./invoker.sh run --namespace gs-k8s --app nsctl --cmd "./resgroup.sh start_appkit --
↪name gs-cluster"
./invoker.sh run --namespace gs-k8s --app nsctl --cmd "./resbook.sh enable_all --
↪group gs-cluster"
./invoker.sh run --namespace gs-k8s --app nsctl --cmd "./resgroup.sh enable --name_
↪gs-cluster"
```

## 5.9 Использование с Ingress

Вы можете использовать Global Server через прокси-сервер, поднятый подсистемой Ingress кластера Kubernetes.

В основе такой подсистемы лежит Ingress-контроллер, который поставляется отдельно от Kubernetes или gs-ctlk.

Схема отображает типичную схему, которая может отличаться для выбранного вами Ingress-контроллера. Пользователи при подключении к кластеру распределяются балансировщиком нагрузки на поды кластера на один из подов с прокси-сервером контроллера (на уровне протокола TCP). Тот в свою очередь распределяет запросы на нужные сервисы на уровне протокола HTTP(S) на основе заголовка Host HTTP и пути ресурса (работа Global гарантируется лишь на отдельном домене/IP-адресе, поэтому используется только поле Host). Запросы к Global ERP дополнительно распределяются внутренним HAProxy, тесно связанным с работой серверов.

Чтобы начать использовать Ingress, выберите и установите Ingress-контроллер, укажите при настройке значений группы ресурсов для книги HAProxy:

```
Использовать Ingress?[да,нет]:да
```

Затем введите настройки для ресурса Ingress:

```
Введите класс Ingress:nginx
Введите хост Ingress:globalerp.example.ru
Введите дополнительную аннотацию Ingress (или пропустите поле): ingress.appscode.com/
↳default-timeout: '{"connect": "28800s"}'
Введите дополнительную аннотацию Ingress (или пропустите поле): haproxy-ingress.github.
↳io/timeout-connect: "28800s"
Введите дополнительную аннотацию Ingress (или пропустите поле):
```

Название используемого класса Ingress должно быть ясно из документации и настроек Ingress-контроллера.

Хост соответствует полю Host в протоколе HTTP. По нему прокси-сервер определяет, к какой службе запрашивается подключения. Если указывается пустое значение, то доступ осуществляется с любого домена.

Аннотации указываются те, что нужны для полной настройки вашего Ingress-контроллера. Если вы не знаете нужны ли вам дополнительные аннотации, то пропускайте поле. Вам скорее всего не нужно его заполнять.

Аналогично заполняются и Ingress-ресурсы для панелей администратора Grafana и RabbitMQ.

### Таймауты

Пользователи при работе с GlobalERP могут встретить сообщение о разрыве соединения. Помимо обычных причин (физический разрыв соединения, перебои сети, переход оборудования в энергосберегающий режим), такое сообщение может возникать в результате закрытия прокси-сервером соединения из-за таймаутов.

У многих Ingress-контроллеров есть аннотации, которые позволяют указать таймауты. Из коробки уже указаны аннотации для следующих контроллеров:

- ingress-nginx (предлагается в документации VK Cloud)
- NGINX Ingress Controller (предлагается в документации Yandex Cloud)

- Voyager
- haproxy-ingress

Вот пример аннотаций, используемых в ingress-nginx:

```
nginx.ingress.kubernetes.io/proxy-connect-timeout: "28800"
nginx.ingress.kubernetes.io/proxy-send-timeout: "28800"
nginx.ingress.kubernetes.io/proxy-read-timeout: "28800"
```

Вы можете переопределить эти настройки указав их при конфигурации группы ресурсов.

Обращаем внимание, что не все Ingress-контроллеры позволяют изменить таймауты при помощи аннотаций. В таком случае, необходимо изменить таймауты при помощи конфигурации контроллера, или отказаться от использования Ingress для GlobalERP, или начать использовать другой контроллер.

## 5.10 Соединение между серверами (JGroups)

Сервера приложений связываются друг с другом через библиотеку JGroups. В gs-ctk есть два способа установления связи:

1. **JGroups DNS**: добавляется книга ресурсов из одного ресурса - сервиса, при помощи которого сервера приложений могут отправлять запросы на раскрутку соединения.
2. **GossipRouter**: добавляется книга ресурсов, поднимающая дополнительный роутер сообщений между серверами. Прямое соединения между серверами не происходит.

Первый способ не требует поднятия дополнительного пода (запросы обрабатываются любым доступным сервером приложений), но менее надежен и требует поддержки соединения между подами серверов приложений, а значит не подойдет, если нужно создать единый логический кластер GlobalERP на нескольких Kubernetes-кластерах с межсетевым экраном между ними.

### Настройка JGroups DNS

Достаточно добавить книгу ресурсов „jgroups\_dns“.

```
./resbook.sh create --name jgroups-dns --group gs-cluster-1 --class_name jgroups_dns
./resbook.sh enable --name jgroups-dns --group gs-cluster-1
```

Никаких дополнительных настроек не требуется.

### Настройка GossipRouter

Вот схема подключения при использовании GossipRouter и двух кластеров:

Как можно увидеть, кластера друг с другом не общаются, а подключаются к GossipRouter'ам по их публичным IP/доменам, что упрощает настройку сети.

**Внимание:** Механизм авторизации соединений не предусмотрен, следовательно вы должны обеспечить безопасность подключения к GossipRouter. Например:

- настроить фильтры на межсетевом экране, чтобы не допустить неуполномоченного подключения к GossipRouter

- шифровать соединение (к примеру, с помощью VPN) при передаче по публичным каналам связи

1. Добавьте книгу ресурсов „gossiprouter“.

```
./resbook.sh create --name gossiprouter --group gs-cluster-1 --class_name_
↪gossiprouter
```

2. Выполните `./resgroup.sh init_spec` и `./resgroup.sh init_values`:

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

```
... # оставьте значения по умолчанию
```

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

```
...
```

Введите список GossipRouter, если такие требуются, или пропустите поле:gossipa.

```
↪example.ru[12001],gossipb.example.ru[12001]
```

```
... # установка значений для других книг ресурсов
```

Установка значений для книги ресурсов: gossiprouter

```
...
```

Введите порт gossip\_router:12001

Введите внешний ip(external\_ip):10.10.0.1

```
...
```

В списке GossipRouter в примере указано два хоста, подключение к которым осуществляется через порт 12001. Предполагается, что по одному из хостов можно подключиться к роутеру в локальном кластере, а по другому - в удаленном.

---

**Совет:** Вы можете использовать IP-адреса вместо доменов и использовать любое число роутеров, в том числе всего один (помните, что кластер может разорваться при отказе единственного роутера).

---

Также мы указали для книги ресурсов порт (12001) и внешний IP (10.10.0.1), при помощи которых можно подключиться к роутеру.

**Внимание:** Сервера приложений должны свободно подключаться к поднятым gs-ctk роутерам по именам и портам из списка, используя протокол TCP.

3. Включите книгу ресурсов.

```
./resbook.sh enable --name gossiprouter --group gs-cluster-1
```



## 5.11 Обновление

### Обновление комплекта приложения

Для обновления *комплекта приложения* (appkit: сервера приложений, прикладного решения) подготовьте его и загрузите на NFS-хранилище (далее `nscli $` означает, что код выполняется на jump-хосте в папке утилиты `nscli`, `nsctl $` - следовательно, на служебном поде `nsctl`):

```
nscli $ ./appkit.sh push --namespace gs-cluster-k8s --source workspace/appkit/v2 --  
↪ destination appkits/v2
```

Где `workspace/appkit/v2` - путь к папке нового appkit на jump-хосте, а `appkits/v2` - путь к папке NFS-хранилища, в которую следует загрузить appkit.

Запустите процесс обновления командой:

```
nscli $ ./appkit.sh switch_und_upgrade --namespace gs-cluster-k8s --resgroup gs-cluster-  
↪ 1 --remote_appkit appkits/v2
```

Помимо `switch_and_upgrade` есть также команда `switch`. Первая, помимо собственно переключения appkit, также обновляет базу данных. Подробнее о необходимости обновлять БД узнавайте у контактного лица.

После обновления комплекта приложения, ваша группа ресурсов останется в опустошенном состоянии (*сервисном режиме*), когда войти могут только администраторы. Чтобы перейти в нормальный режим, воспользуйтесь командой:

```
nsctl $ ./resgroup.sh start_appkit --name gs-cluster-1
```

### Обновление кластерных утилит

С версии `nscli 4.0.0` доступна команда:

```
./namespace.sh upgrade_namespace
```

Она позволяет обновить образа `gs-ctk` без пересоздания пространства имен. Для использования:

1. *Загрузите и установите* на jump-хост новую версию `nscli`.
2. Запустите:

```
./namespace.sh create_namespace
```

Вам будет представлен диалог для введения настроек пространства имен (реестр Docker, NFS). Проверьте, что данные верны.

**Предупреждение:** Не запускайте `./namespace.sh install_namespace`

3. Если вы используете версию `gs-ctk 1.0`, то *экспортируйте конфигурацию*. Если на кластере развернута `gs-ctk 2.0` и выше, резервная копия будет сделана автоматически.
4. Запустите:

```
./namespace.sh upgrade_namespace
```

После выполнения команды, кластерные утилиты будут обновлены до версии `nscli`.  
Для использования команды отключать книги и группы ресурсов необязательно, но рекомендуется.

## 5.12 Частичное изменение настроек

Иногда появляется острая необходимость изменить настройки отдельного приложения, оставив остальные работать. Для этого вы можете отключить одну книгу ресурсов, сделать необходимые изменения и включить книгу ресурсов обратно.

В случае *неуправляемого режима* перезапись ресурса идентичным не должна вызывать изменений в Kubernetes, поэтому просто сделайте нужные изменения в конфигурации, сгенерируйте новые ресурсы и примените их.

### Пример

*Данные:* встроенный под grafana упал под нагрузкой из-за ошибки OutOfMemory. Поскольку это вызывает задержку в обработке телеметрии, объем необработанных данных увеличивается, а следовательно нагрузка на стек мониторинга только растет, из-за чего он начинает регулярно падать. Единственное решение в таком случае - остановить приложение и запустить его заново (что затруднительно сделать на продуктивном контуре) или увеличить ресурсы у пода.

*Решение:* увеличим ресурсы у пода.

1. Отключим книгу ресурсов, запустив команду на `nsctl`:

```
./resbook.sh disable --group gs-cluster-1 --name grafana
```

2. Сделаем изменения в лимитах ОЗУ:

```
./resbook.sh init_spec --group gs-cluster-1 --name grafana  
#./resbook.sh init_values --group gs-cluster-1 --name grafana
```

3. Включим книгу ресурсов обратно:

```
./resbook.sh enable --group gs-cluster-1 --name grafana
```

## 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

Склонировать [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
```

Создайте свою пару ключей для подписи токенов (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"}`
- Обратите внимание на флаг `bUsernameEnglishLettersOnly`. Если он установлен, то системные имена пользователей не могут содержать символы национального алфавита, только английские буквы.
- Добавьте публичный ключ подписи в текстовом виде без заголовка и подвала под ключом `extUserPublicKey`. Например, для тестового ключа `deploy/templates/private_key.pem`:

```
MIIBIjANBgkqhkiG9w0BAQEFAA0CAQ8AMIIBGKCAQEAAnY1eq7C1PMhnXdvrlM5EcC6B4VgkLheotvPIiLf5vV2ZS+VPDhc2ZyO
↪i4zoFcs8qUwHfCwsTRmdl828YRlzf0rHA/
↪jiT21J1AzkkgMSQmIH9XRxlSS9HmkP6Hgvx7Fe+ir86kU6Pw50LaCeBnlh0RUrolSnyLNhjPuCqIQ54pfz8YzR/
↪T2jMgxU+hvVjD2vC80JLNDWv7gOrNzynV4zIWnt6gkiHzkvwIDAQAB
```

## 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 меню приложений)
- настройки Django
- имена для генерации токена
- путь к файлу с логами
- сведения об организации
- время жизни кода подтверждения
- путь для доступа к cookie
- включение тихого режима (True/False) (По-умолчанию приватный ключ хранится в `~/gs-authproxy.priv`)

Пример заполненного `config.yml`:

```
database:
# Имя БД
name: authproxy
# хост на котором будет запущена БД
host: localhost
# порт на котором будет запущена БД
port: 5432
```

(continues on next page)

```

django:
# Url для записи секретного ключа в credential manager. Секретный ключ Django играет
↳ роль в обеспечении безопасности вашего веб-приложения.
  url: 'django_secret_key'
# Режим разработки.
  debug: true

endpoints:
# url на который будет отправляться запросы
  request_url: 'http://127.0.0.1:80/app/sys/rest/ss/pkg/Btk_JexlGatePkg/execute'
# url на который будет отправляться пользователь для авторизации
  authorize_url: '/PGDEV/Btk_ConfiguratorMainMenu/gtk-ru.bitec.app.btk.Btk_Notification
↳ %23Head/'
  database_name: PGDEV
  debug_register_url: '/PGDEV/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.example.com'
# порт SMTP-сервера. Значение 587 часто используется для подключения к серверу через TLS
↳ (Transport Layer Security)
  email_port: 587
# булево 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
  email_use_tls: True
  email_use_ssl: False
# Тема письма.
  subject: 'Код подтверждения для портала поставщиков'
# Текст сообщения в письме. Обязательно сохранять переменную "{code}"
  message: '''
  Добрый день!

Ваш код подтверждения {code} - введите его на сайте для продолжения.
Если Вы не запрашивали данный код, пожалуйста, проигнорируйте это письмо.

С уважением

----

Данное сообщение сформировано автоматически и ответ на этот адрес не будет прочитан.'''

gs_tokens:
# Ключ для подписи токена сервисного пользователя
# Должен располагаться в каталоге gs-authproxy/mail_gate_pass/security
  service_private_key: 'security/service_private_key.pem'

# Ключ для подписи токена пользователя под которым идет регистрация учетных данных

```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Должен располагаться в каталоге gs-authproxy/mail_gate_pass/security
register_private_key: 'security/register_private_key.pem'

# Ключ для подписи простых пользователей.
# Должен располагаться в каталоге gs-authproxy/mail_gate_pass/security
user_private_key: 'security/user_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://127.0.0.1'

# Укажите путь, по которому куки будут доступны
cookie:
  path: 'PGDEV'

# Укажите путь для хранения логов
log:
  path: /tmp/gs-authproxy-error.log

# Укажите время жизни кода подтверждения в секундах.
verification_code:
  time: 300

# Тихий режим. При включении не запрашивает место для хранения ключа от паролей,
↳использует значение по-умолчанию.
# Нужен для работы в режиме CI/CD. Допустимые значения: True / False. По-умолчанию
↳приватный ключ хранится в ~/.gs-authproxy.priv
silent_mode:
  enabled: {{ silent_mode_enabled }}

# Данные организации. Для замены логотипа и favicon. Замените соответствующие файлы в
↳директории "mail_gate_pass/static_dev/img"
organization:
  name: "Бизнес Технологии"
  phone_number: '+7 (777) 77-78-90'
  email: 'support@gmail.com'
  background_color: '#33a93d'
  url: 'https://global-system.ru/'
```

**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 сервер авторизации пересылает по статичной ссылке).

Затем инициализируйте БД и соберите статические файлы.

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

Задайте пароли для базы данных, секретного ключа Django и почтового сервера. Для этого перейдите в каталог `gs-authproxy` и выполните скрипт `set_credentials.sh`

```
./set_credentials.sh set --url your_url --user your_user --password your_password`
```

- для БД:
  - в качестве параметра `--url` используйте имя БД, которое вы указали в файле с конфигурациями проекта.
  - для параметра `--user` имя пользователя, которое вы использовали при создании БД.
  - для параметра `--password` пароль от БД, который вы использовали при создании БД.
- для секретного ключа Django:
  - в качестве параметра `--url` используйте `django:url`, который вы указали в файле с конфигурациями проекта.
  - для параметра `--user` «django».
  - для параметра `--password` можно использовать любой достаточно длинный случайный набор символов.
- для сервера почты:
  - в качестве параметра `--url` используйте `email_host`, который вы указали в файле с конфигурациями проекта.
  - для параметра `--user` имя пользователя (адрес электронной почты) для аутентификации на SMTP-сервере
  - для параметра `--password` пароль для аутентификации на SMTP-сервере` ``

## 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 соответствующие 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
    # Задаёт 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
```

Перезапустите Apache:



```
sudo systemctl restart apache2
```

## 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
    option forwardfor
    # 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
    # добавляет заголовок X-Forwarded-Port
    http-request set-header X-Forwarded-Port %[dst_port]
    # определяет globalserver.
    server globalserver 127.0.0.1:8080
```

### Пример конфигурации haproxy.cfg с использованием https:

```
frontend http-in
    mode http
    # указывает HaProxy, какие адреса и порты прослушивать
    bind :443 ssl crt /etc/haproxy/certs/cert.pem # путь до ssl-сертификата
    http-request add-header X-Forwarded-Proto https if { ssl_fc }
    option forwardfor
    # 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
    # добавляет заголовок X-Forwarded-Port
    http-request set-header X-Forwarded-Port %[dst_port]
    # определяет globalserver.
    server globalserver 127.0.0.1:8080
```

Перезапустите сервис haproxy:

```
sudo systemctl restart haproxy
```

## 6.3 Обновление

Перед обновлением, не забудьте сделать резервные копии!

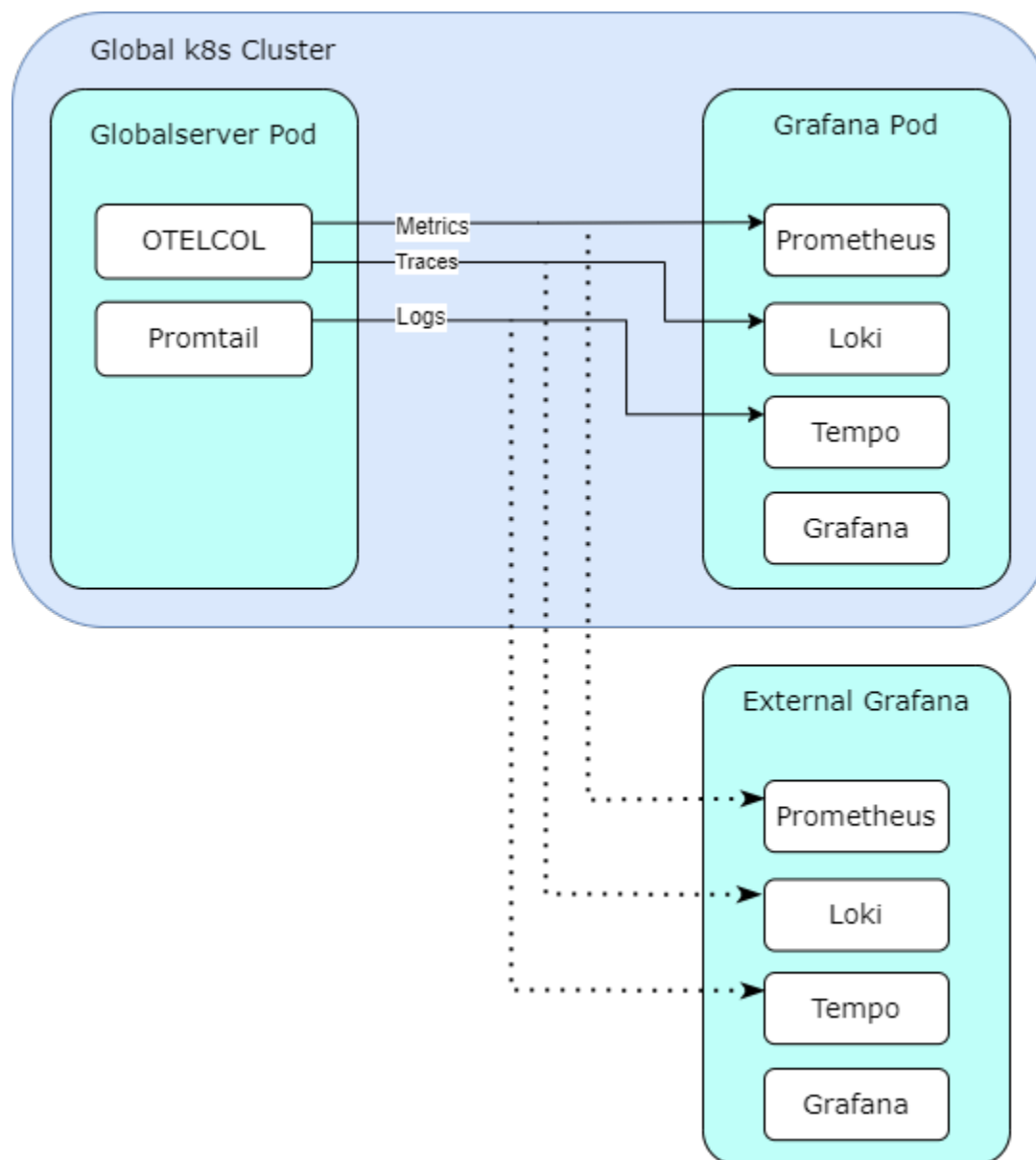
Если вы клонировали репозиторий при помощи средств git, то обновите его до последней версии. Если же вы получили сервер авторизации в виде архива, то:

1. Скопируйте mail\_gate\_pass/config.yml в надежное место. Если вы создавали другие конфигурационные файлы в папке gs-authproxy (например, ключ подписи), то также скопируйте его в другое место.
2. Удалите содержимое папки gs-authproxy и распакуйте в нее новый дистрибутив.
3. Верните файлы конфигурации на их места.

Независимо от способа обновления файлов, выполните следующие команды для успешной миграции:

```
# cd gs-authproxy
bin/initvenv.sh
source venv/bin/activate
cd mail_gate_pass
python manage.py migrate
python manage.py collectstatic
```

## 7 Внешняя система мониторинга



Система мониторинга состоит из Prometheus, Loki, Tempo и Grafana.

Сервер Global, развёрнутый в кластере Kubernetes, отправляет метрики, трейсы и логи во внутреннюю систему мониторинга. Также возможно настроить кластер Global так, чтобы он отсылал телеметрию

во внешнюю систему. Для этого, при [конфигурировании](#) кластера Global требуется:

- В визарде установки характеристик (`./resgroup.sh init_spec`) задать значение **true** для полей «Отслеживать метрики» и «Дополнительно отсылать метрики во внешнюю систему».
- В визарде установки значений (`./resgroup.sh init_values`) задать параметры для полей «Введите адрес доступа к внешнему prometheus», «Введите адрес доступа к внешнему loki» и «Введите адрес доступа к внешнему tempo» для каждой книги ресурсов.

## 7.1 Настройка Prometheus

Prometheus устанавливается по [официальной документации](#). Для получения метрик сервера Global дополнительные настройки не требуются.

Пример `config.yaml`

```
global:
  scrape_interval:    5s
  evaluation_interval: 5s
```

## 7.2 Настройка Loki

Loki устанавливается по [официальной документации](#).

Простейший конфиг приведён ниже. При его использовании требуется задать корректные значения параметров `path_prefix`, `chunks_directory` и `rules_directory`.

Пример `config.yaml`

```
auth_enabled: false

server:
  http_listen_port: 3100
  grpc_listen_port: 9096

common:
  instance_addr: 127.0.0.1
  path_prefix: {path}          #/mnt/data/loki
  storage:
    filesystem:
      chunks_directory: {path} #/mnt/data/loki/chunks
      rules_directory: {path}  #/mnt/data/loki/rules
  replication_factor: 1
  ring:
    kvstore:
      store: inmemory

query_range:
  results_cache:
    cache:
      embedded_cache:
        enabled: true
        max_size_mb: 100
```

(continues on next page)

```

schema_config:
  configs:
    - from: 2020-10-24
      store: tsdb
      object_store: filesystem
      schema: v13
      index:
        prefix: index_
        period: 24h

ruler:
  alertmanager_url: http://localhost:9093

analytics:
  reporting_enabled: false

```

### 7.3 Настройка Темпо

Темпо устанавливается по [официальной документации](#). Простейший конфиг приведён ниже. При его использовании требуется задать корректные значения параметров *path*.

Пример `config.yaml`

```

stream_over_http_enabled: true
server:
  http_listen_address: 0.0.0.0
  http_listen_port: 3200

distributor:
  receivers:
    otlp:
      protocols:
        http:
          endpoint: 0.0.0.0:3201

storage:
  trace:
    backend: local
    wal:
      path: {path} #/mnt/data/tempo/wal
    local:
      path: {path} #/mnt/data/tempo/blocks

```

## 7.4 Настройка Grafana

Grafana устанавливается и настраивается по официальной документации.

## 8 Дополнительно

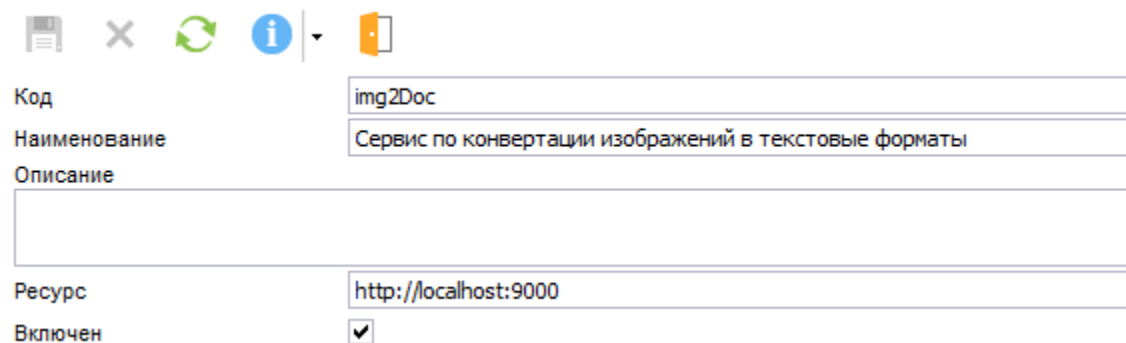
### 8.1 Установка и настройка сервиса конвертации избр. в документы

#### Установка

Скачайте релиз сервиса gs-img2doc с предоставленного ресурса.  
(Предоставляется через контактное лицо, файл gs-img2doc-X.X.X.zip).  
Актуальная инструкция по установке находится в архиве /doc/install.md  
Требования к железу 3 сру 3Gb ram (~4 одновременных пользователя).

#### Интеграция с Global

Указать URL сервиса и проставить галочку включено в приложении «Настройки системы» («Настройки и сервисы» -> «Дополнительно» -> «Сервисы»)



Код	img2Doc
Наименование	Сервис по конвертации изображений в текстовые форматы
Описание	
Ресурс	http://localhost:9000
Включен	<input checked="" type="checkbox"/>

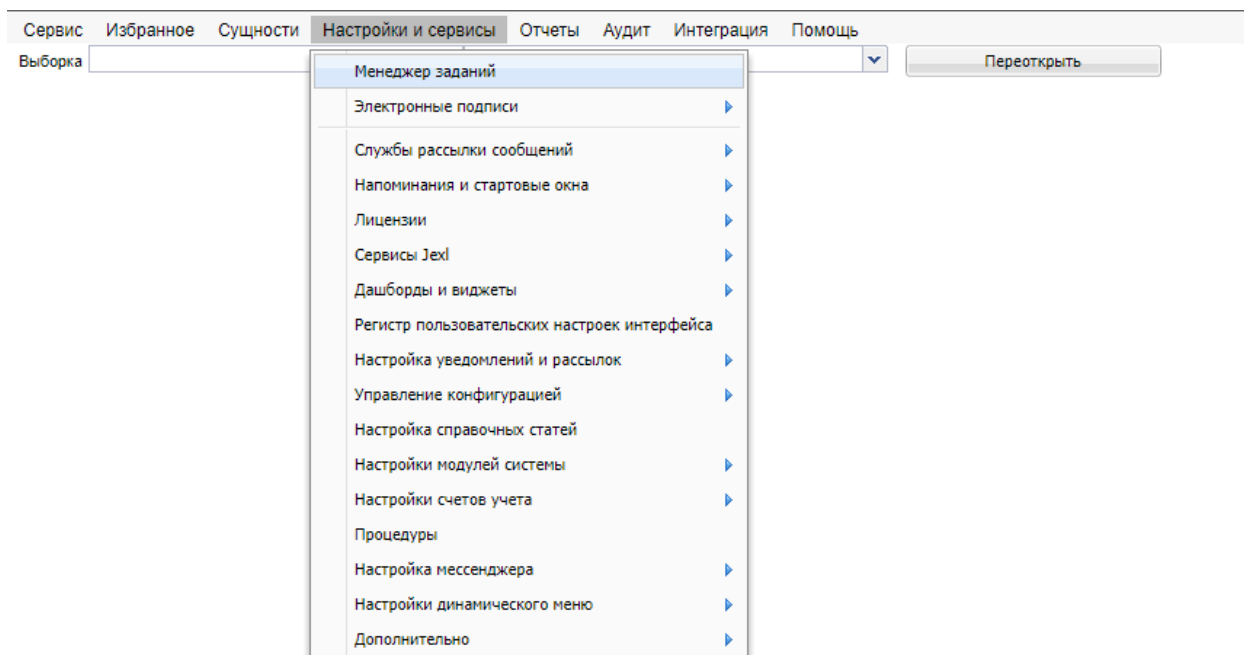
### 8.2 Настройка GlobalScheduler

#### Генерация ключей шифрования

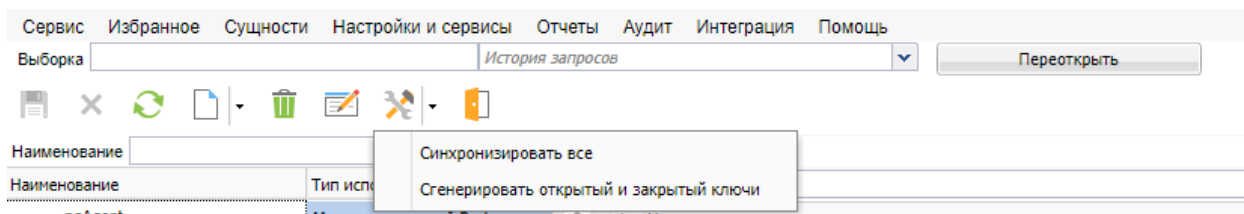
#### Генерация ключей в системе Global

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 скачаны два файла: quartz.publickey.txt (открытый ключ) и quartz.privatekey.txt (закрытый ключ).

## Генерация ключей сторонними утилитами

Допускается генерация ключей с помощью сторонних утилит.

**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:

```
# Генерация приватного ключа в стандартном формате
openssl genpkey -algorithm RSA -out private.pem -pkeyopt rsa_keygen_bits:2048; \
# Генерация публичного ключа в стандартном формате
openssl rsa -in private.pem -outform PEM -pubout -out public.pem; \
# Форматирование ключей для работы с Global
sed '/-----.*-----/d' ./private.pem | tr -d '\n' > quartz.private.pem; \
```

(continues on next page)

(продолжение с предыдущей страницы)

```
sed '/-----.*-----/d' ./public.pem | tr -d '\n' > quartz.public.pem; \  
# Конвертация ключа в base64 для использования в секрете kubernetes (не требуется при \  
→ использовании Global в автономном режиме)  
base64 -w 0 ./quartz.private.pem > ./b64.private.pem
```

Убедитесь, что сгенерированный вами ключ был создан в подходящем для Globalscheduler формате можно при помощи следующей команды:

```
base64 -d ./quartz.private.pem | openssl asn1parse -inform DER | grep -q rsaEncryption &&  
→ echo "Ключ верного формата" || echo "Ключ неподходящего формата"
```

## Конфигурация GlobalScheduler и установка приватного ключа

### Автономный режим

### Настройка конфигурационных файлов

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>  
org.quartz.dataSource.quartzDS.password = <DBPASS>  
  
org.quartz.jobStore.tablePrefix=bt_k_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

```
<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>[%-5level] %d{dd-MM-yyyy HH:mm:ss.SSS} [%thread]
↪%logger - %msg%n</pattern>
    </encoder>
  </appender>

  <appender name="OUT_SSH" class="ru.bitec.engine.core.logging.SshConsoleAppender">
    <encoder>
      <!--<pattern>[%-5level] %d{HH:mm:ss.SSS} [%thread]:%X{USER} -
↪%msg - %logger%n</pattern>-->
      <pattern>[%-5level] %d{dd-MM-yyyy HH:mm:ss.SSS} - %msg</pattern>
    </encoder>
  </appender>

  <appender name="FILEOUT" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy"
↪">
      <fileNamePattern>${G3_HOME}\logs\global3.%d{yyyy-MM-dd_HH}.log</
↪fileNamePattern>
    </rollingPolicy>
    <encoder>
      <pattern>%d{HH:mm:ss.SSS} [%thread] %logger - %msg%n</pattern>
    </encoder>
  </appender>

  <!--
  Возможные значения параметра "level" в порядке приоритета важности
  OFF
  ERROR
  WARN
  INFO
  DEBUG
  TRACE
  ALL
  -->

  <root level="INFO">
    <appender-ref ref="STDOUT_DEFAULT"/>
    <appender-ref ref="OUT_SSH"/>
```

(continues on next page)

```

        <appender-ref ref="FILEOUT"/>
    </root>

    <!-- Для логирования SQL вызовов переключите в режим INFO или ниже-->
    <!-- Текущие значения не менять, тк: -->
    <!-- Установка уровня логирования на сервере задается через фильтр в начале
    ↪ файла.-->
    <!-- Установка уровня логирования на клиенте задается через общий файл
    ↪ конфигурации и\или выставляется клиентом.-->
    <logger name="jdbc" level="off" additivity="false"/>

    <!-- Выводит SQL-текст, значения IN\OUT параметров -->
    <!--<logger name="jdbc.audit" level="info"/>-->

    <!-- Выводит текст SQL - вызова.
    Переведено в режим OFF, что бы не дублировать вывод SQL-текста-->
    <logger name="jdbc.sqlonly" level="OFF"/>

    <!-- Выводит текст SQL - вызова с временем выполнения.
    Переведено в режим OFF, что бы не дублировать вывод SQL-текста -->
    <logger name="jdbc.sqltiming" level="OFF"/>

    <logger name="net.sf.log4jdbc" level="warn">
        <appender-ref ref="STDOUT_DEFAULT"/>
    </logger>

    <!-- Sbt -->
    <!-- info - общие сообщения о действиях менеджера-->
    <!-- debug - "+" весь вывод SBT, обновляемые файлы-->
    <!-- trace - "+" все события DirWatcher'a-->
    <logger name="ru.bitec.engine.sbt" level="info"/>

    <!--<logger name="ru.bitec.engine.session" level="trace"/>-->
    <!--<logger name="ru.bitec.common.rpc" level="trace"/>-->

</configuration>

```

## Настройка сервиса 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

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

Разрешите автозагрузку сервиса

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

## Установка приватного ключа

В основном файле конфигурации планировщика `quartz.properties` требуется указать путь до файла с закрытым ключом шифрования (`quartz.privatekey.txt/quartz.private.pem`) в параметре

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

Пример конфигурации:

```
ru.bitec.jobscheduler.privatekey.path = /opt/global/globalserver/application/config/
↪tools/scheduler/quartz.private.pem
```

**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.

Для применения конфигурации перезапустите сервис `globalscheduler`:

```
sudo systemctl restart globalscheduler.service
```

## Kubernetes

В случае, если вы сгенерировали ключи при помощи системы Global, необходимо дополнительно зашифровать закрытый ключ в base64. Для этого можно воспользоваться следующей командой, находясь в директории с ключом:

```
base64 -w 0 ./quartz.private.key > ./b64.private.pem
```

Создайте секрет с токеном планировщика, заменив значение namespace на название пространства имен Global, и значение шаблона <b64\_key> на содержимое файла b64.private.pem

```
apiVersion: v1
kind: Secret
metadata:
  name: scheduler-token-secret
  namespace: gs-cluster-k8s
data:
  private.key: <b64_key>
```

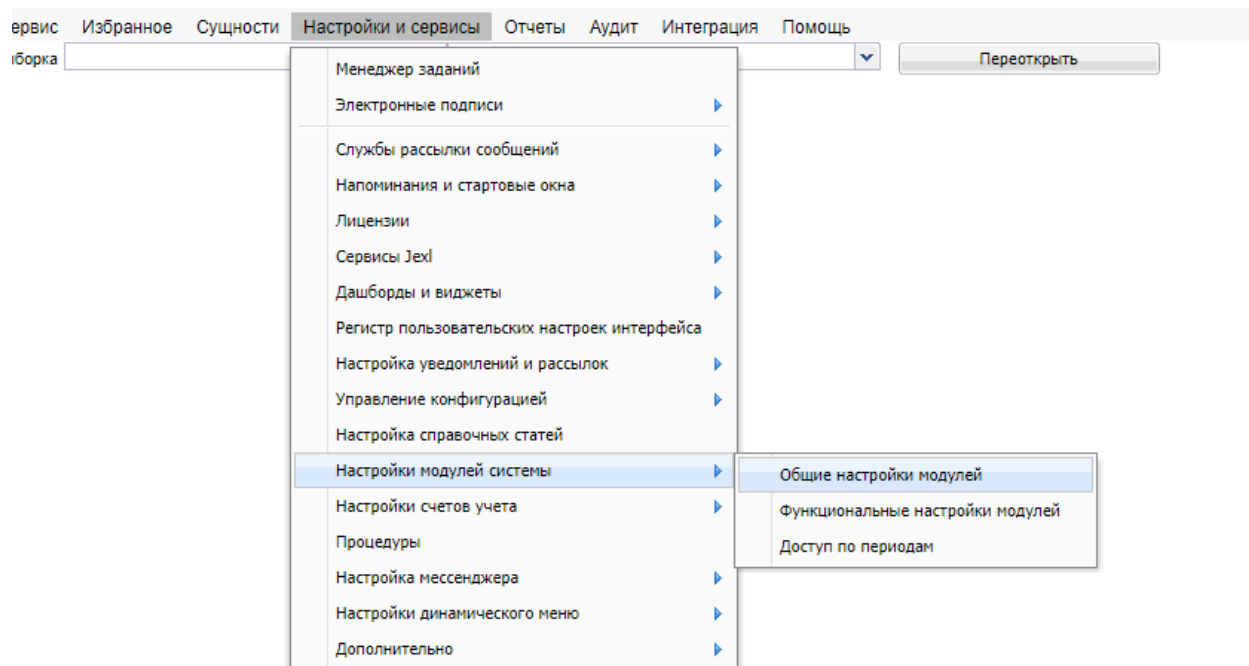
```
cat <<EOF | tee ~/nsccli/workspace/scheduler-token-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: scheduler-token-secret
  namespace: gs-cluster-k8s
data:
  private.key: <b64_key>
EOF
```

```
kubect1 apply -f ~/nsccli/workspace/scheduler-token-secret.yaml
```









## Установка публичного ключа и настроек Quartz

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

Отображать неиспользуемые ☐ Редактировать (F4)

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

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

Модуль

Не используется ☐

Дата окончания использования

Настройки

Ключ	Значение
Путь до логов планировщика	/usr/local/globalserver/logs/scheduler/
Макс. дней для хранения записей телеметрии	14
LDAP-синхронизация	{"sDomain":null,"sLogin":null,"sUrl":null,"sPass":r
Прокси	{"sLogin":"proxy_user","isEnabled":false,"sHost":
Разрешено администрирование приложений	<input checked="" type="checkbox"/>
Шаблон скрипта для PgAgent-a	#!/bin/bash...
Адрес XMPP-сервера	192.168.2.66
Порт XMPP-сервера	5222
Префикс Url ссылки	pgDev
Настройки для Quartz	{"database":"pgdev","pass":"admin","public
Test_test	<input checked="" type="checkbox"/>
Настройки для формирования URL, по ...	{"sTransferProtocolForGenGidUrl":"http","sHostN
Выполнение jexl-скриптов через безопасный...	<input type="checkbox"/>
Cluster	{}

В открывшемся окне задать следующие значения:

База данных: alias базы данных из файла global3.config.xml  
 HTTPS включён: в зависимости от того, используется ли https в Global, включить данную опцию  
 Логин: логин провозвателя scheduler в Global (scheduler по умолчанию)  
 Пароль: пароль пользователя scheduler в Global  
 Публичный ключ: открытый ключ (содержимое файла quartz.publickey.txt/quartz.public.pem)  
 Soap хост: домен/IP адрес, на котором доступен Global  
 Soap порт: порт, на котором доступен Global (8080 по умолчанию)

Пример конфигурации:

По завершению конфигурации нажать зеленую стрелку, а затем операцию «Сохранить» в окне с настройками модуля btk.

## Частые проблемы

### algid parse error, not a sequence

При запуске globalscheduler в логах может возникнуть ошибка следующего вида:

```
Job Default.22 threw a JobExecutionException: org.quartz.JobExecutionException: java.  
↳ security.spec.InvalidKeySpecException: java.security.InvalidKeyException: IOException  
↳ : algid parse error, not a sequence
```

Чаще всего данная проблема связана с некорректным форматом приватного ключа. Для работы с Globalscheduler требуется 2048-битный ключ RSA формата PKCS#8. Ключи формата PKCS#1 не подходят для работы с Globalscheduler.

Чтобы убедиться, что ваш ключ верного формата, проверьте, что сгенерированный вами неотформатированный приватный ключ (quartz.privatekey.txt/quartz.private.pem) начинается со следующей строки:

```
-----BEGIN PRIVATE KEY-----
```

В случае, если ваш ключ неверного формата (PKCS#1), он будет начинаться со следующей строки:

```
-----BEGIN RSA PRIVATE KEY-----
```

Также проверить формат ключа можно при помощи openssl, что может быть полезно, если ключ доступен только в отформатированном для Globalscheduler виде (quartz.private.txt/quartz.private.pem), и не содержит заголовков

Для этого используйте следующую команду, заменив значение <key\_path> на путь до приватного ключа:

```
base64 -d <key_path> | openssl asn1parse -inform DER | grep -q rsaEncryption && echo  
↳ "Ключ верного формата" || echo "Ключ неподходящего формата"
```

Чтобы сгенерировать ключ подходящего формата, воспользуйтесь генерацией пары ключей в системе Global, или следующей командой:

```
openssl genpkey -algorithm RSA -out private.pem -pkeyopt rsa_keygen_bits:2048
```

### Illegal base64 character

В случае возникновения ошибки «Illegal base64 character» убедитесь, что ключи верно отформатированы. Ключи, используемые Globalscheduler, не должны содержать заголовков (—BEGIN PUBLIC KEY—, —END PUBLIC KEY— и пр.) и знаков переноса строки.

Если вы используете kubernetes, убедитесь, что в секрете используется ключ, дополнительно закодированный в base64 (b64.private.pem).

Для этого проверьте, что файл ~/globalserver/workspace/mnt/secret/scheduler/private.key внутри контейнера Globalscheduler содержит приватный ключ в читаемом формате.

**Error 401 JWT signature does not match locally computed signature. JWT validity cannot be asserted and should not be trusted.**

Данная ошибка связана с тем, что установленный публичный ключ не подходит к установленному приватному ключу. Убедитесь, что установленные в Globalscheduler ключи из одной пары, либо сгенерируйте новую пару ключей (*Генерация ключей шифрования*)

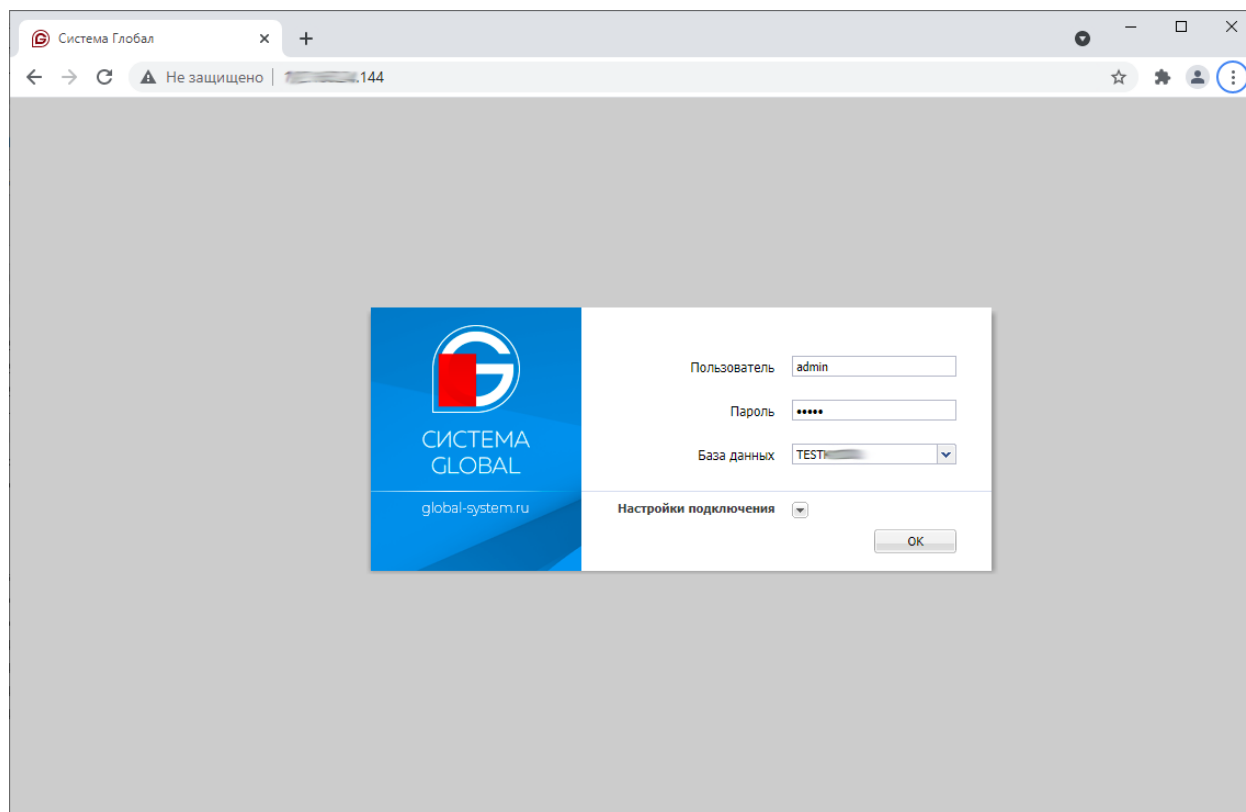
## 9 Первый вход в систему

### 9.1 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

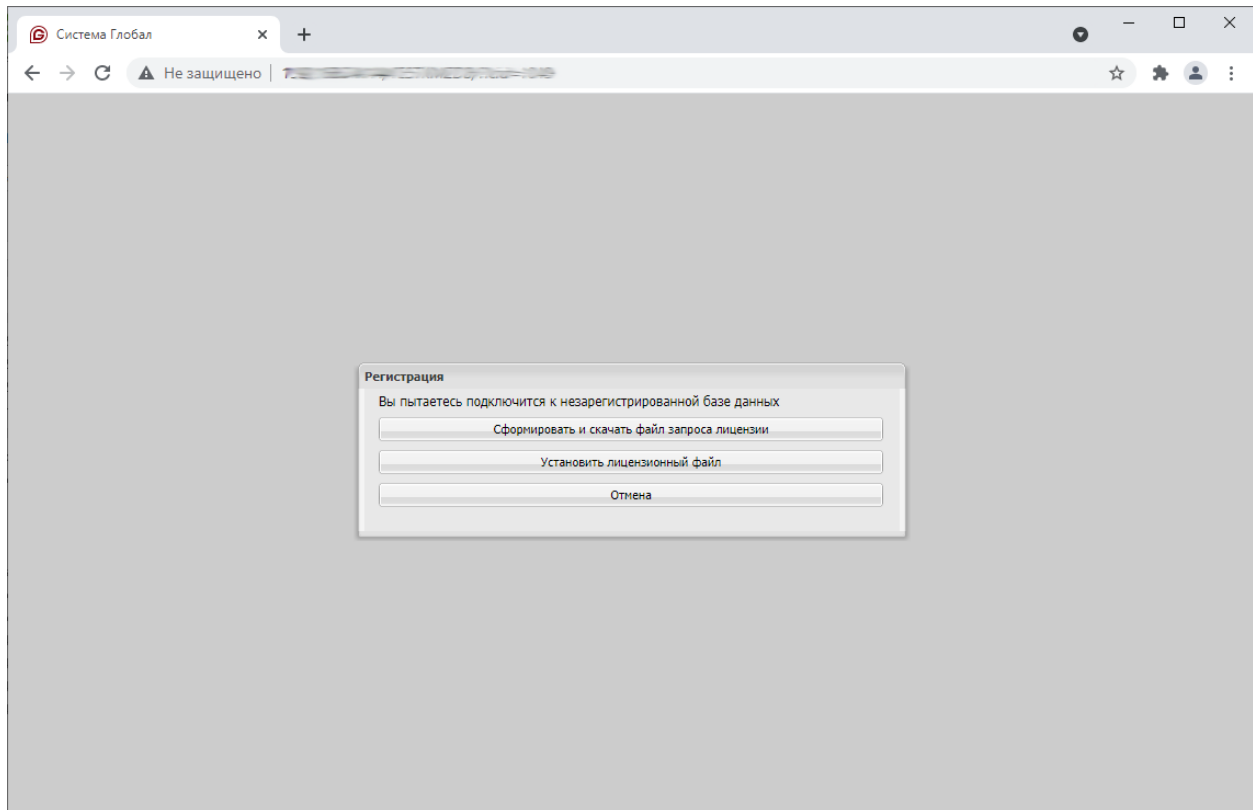
Пароль: admin

**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.

## 10 Логирование сервера приложений.

### 10.1 Общий обзор

Логирование в проекте осуществляется с использованием **Logback**. Конфигурация логирования задается в XML-файлах, расположенных в каталоге `{{workspace}}/application/config/`.

### 10.2 Структура конфигурационных файлов

```
{{workspace}}/  
└─ application/  
    └─ config/  
        └─ logback-LoggerContext.xml # Основной файл конфигурации системных логов  
        └─ logback-LoggerContext-ext.xml # Проектные настройки системных логов  
        └─ logback-LoggerContext-session.xml # Основной файл конфигурации логов сессии  
        └─ logback-LoggerContext-session-ext.xml # Проектные настройки логов сессии
```

### ! Внимание!

Логирование в проекте настроено с учетом системных и сессионных логов.

**Изменение основных файлов конфигурации запрещено**, но пользователь может вносить свои изменения через файлы `logback-LoggerContext-ext.xml` и `logback-LoggerContext-session-ext.xml`.

## Основные конфигурационные файлы

### 1. `logback-LoggerContext.xml`

- Основной файл конфигурации логирования.
- Отвечает за запись системных логов.
- Фиксирует сообщения начиная с уровня **INFO** и выше.
- Изменение этого файла **не допускается**.

### 2. `logback-LoggerContext-session.xml`

- Конфигурация логирования текущей сессии.
- Фиксирует сообщения начиная с уровня **WARN** и выше.
- Изменение этого файла **не допускается**.

## Расширяемые файлы конфигурации

Если требуется изменить настройки логирования, пользователь может использовать специальные файлы расширений:

### 1. `logback-LoggerContext-ext.xml`

- Подключается к `logback-LoggerContext.xml`.
- Позволяет добавить или изменить настройки логирования системных событий.

### 2. `logback-LoggerContext-session-ext.xml`

- Подключается к `logback-LoggerContext-session.xml`.
- Позволяет добавить или изменить настройки логирования текущей сессии.

## 10.3 Пример добавления кастомного логгера

Чтобы добавить кастомный логгер в файл `logback-LoggerContext-ext.xml`, можно использовать следующий шаблон:

```
<included>
  <appender name="FILEOUT-EXT" class="ch.qos.logback.core.rolling.RollingFileAppender
  <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <fileNamePattern>/opt/global/globalserver/logs/global3.%d{yyyy-MM-dd_HH}.log
  </fileNamePattern>
  </rollingPolicy>
  <encoder>
    <pattern>%d{HH:mm:ss.SSS} [%thread] %logger - %msg%n</pattern>
  </encoder>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
</appender>

    <root level="INFO">
    <appender-ref ref="FILEOUT-EXT"/>
    </root>
</included>
```