G3SysAdminDoc

Выпуск 0.1.17

Global System

нояб. 22, 2025

Содержание

1 Описание Global ERP **3**
- 1.1 Назначение и функции системы . . . 3
- 1.2 Компоненты системы . . . 3
- 1.3 Требования к программному и аппаратному обеспечению . . . 4
- 1.4 Структура системы . . . 6
- 1.5 Перечень основных подсистем . . . 6

2 Администрирование Postgres **7**
- 2.1 Установка и настройка СУБД . . . 7
- 2.2 Установка . . . 11
- 2.3 Настройка и запуск . . . 11
- 2.4 Безопасность . . . 14
- 2.5 Тестирование производительности сервера . . . 14
- 2.6 Резервное копирование и восстановление БД . . . 16
- 2.7 Смена локалей для postgresql . . . 24
- 2.8 Структура каталогов и файлов PostgreSQL . . . 26
- 2.9 Инструкция по установке и настройке кластера СУБД Postgresql . . . 28
- 2.10 Инструкция по настройке и работе с расширением pg_stat_kcache . . . 46

3 GlobalServer Автономный режим **49**
- 3.1 Установка сервера приложений . . . 49
- 3.2 Администрирование системы . . . 63
- 3.3 Управление схемой базы данных . . . 71
- 3.4 Настройка HAProxy (standalone) . . . 72
- 3.5 Руководство по установке сервера мониторинга . . . 82

4 GlobalServer кластер в kubernetes **104**
- 4.1 Описание . . . 104
- 4.2 Установка . . . 106
- 4.3 Миграция с предыдущих версий gs-ctk . . . 128
- 4.4 Настройки конфигурации . . . 129
- 4.5 Развертывание одним чартом . . . 137
- 4.6 Хуки . . . 138
- 4.7 Обновление . . . 139
- 4.8 Отладка работы пода . . . 142
- 4.9 Использование с Ingress . . . 142
- 4.10 Соединение между серверами (JGroups) . . . 144

4.11 Отладка прикладного решения . . . 147
4.12 Мониторинг кластера Global ERP . . . 152
4.13 Развертывание с использованием gs-ctk 4 . . . 156

5 Развертывание сервера приложений GS с сервером авторизации 199
5.1 Схема работы сервера авторизации . . . 200
5.2 Подготовка . . . 200
5.3 Развертывание и конфигурация . . . 201
5.4 Обновление . . . 209

6 Процедура обновления Global ERP 209
6.1 1. Общие положения . . . 209
6.2 2. Состав обновления . . . 210
6.3 3. Подход к обновлению . . . 210
6.4 4. Условия и режимы обновления . . . 210
6.5 5. Периодичность и сроки проведения . . . 210
6.6 6. Ответственность и контроль качества . . . 211
6.7 7. Документирование и сопровождение . . . 211
6.8 8. Принятие обновлений . . . 211
6.9 9. Сроки и порядок проведения обновлений . . . 211

7 Дополнительно 212
7.1 Настройка SSO . . . 212
7.2 Синхронизация пользователей (LDAP) . . . 214
7.3 Установка и настройка сервиса конвертации избр. в документы . . . 221
7.4 Генерация ключей шифрования . . . 222
7.5 Конфигурация GlobalScheduler в автономном режиме . . . 225
7.6 Конфигурация GlobalScheduler в Kubernetes . . . 232
7.7 Проверка работоспособности сервиса . . . 233
7.8 Частые проблемы . . . 236
7.9 Руководство по снятию снапшотов GlobalScheduler . . . 237
7.10 Настройка визуализации контуров приложения . . . 244
7.11 Проверка связи улов кластера . . . 247

8 Лицензирование системы 248
8.1 Описание . . . 248
8.2 Первый вход и активация лицензии . . . 248
8.3 Управление лицензиями . . . 250
8.4 Примечания и рекомендации . . . 252

9 Логирование сервера приложений 253
9.1 Общий обзор . . . 253
9.2 Структура конфигурационных файлов . . . 253
9.3 Пример добавления кастомного логгера . . . 254
9.4 Конфигурация логов для HaProxy . . . 254

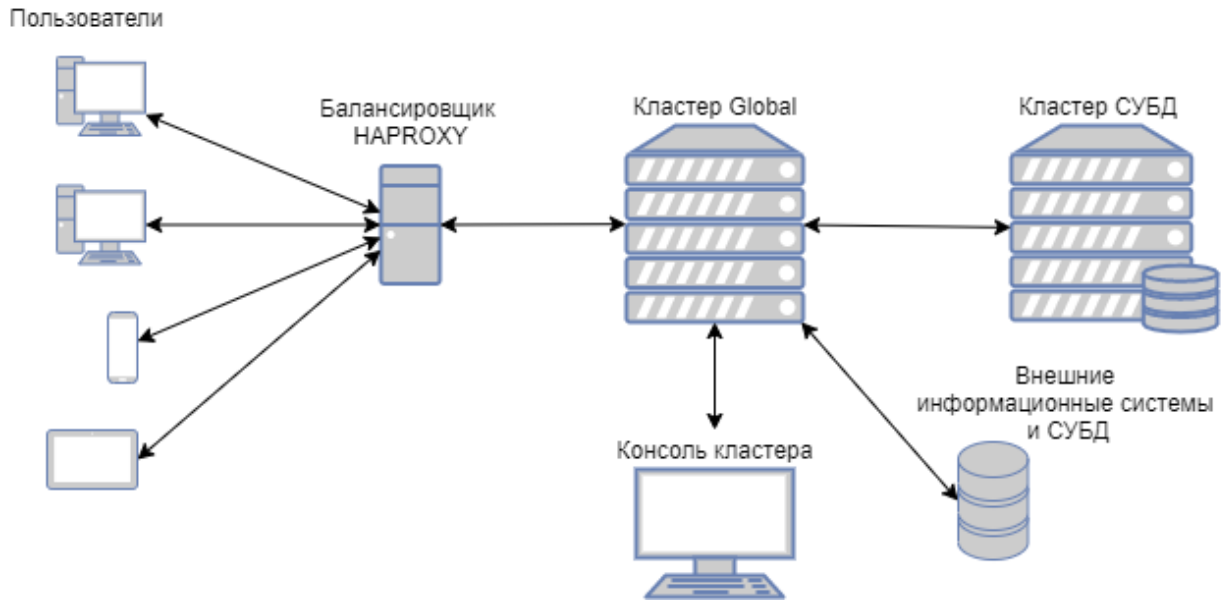
10 Установка клиентского ПО 256
10.1 Одиночная установка . . . 256
10.2 Установка групповыми политиками . . . 257

11 Конфигурация 262
11.1 global3.config.xml . . . 262
11.2 Логирование . . . 264

1 Описание Global ERP

1.1 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.



1.2 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

1.3 Требования к программному и аппаратному обеспечению

Сервер приложений

Аппаратное обеспечение

Характеристика сервера	Рекомендуемые параметры
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 120Gb 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).

Программное обеспечение

Программное обеспечение	Рекомендуемые параметры
ОС	Astra Linux / Debian / РЕД ОС / ALT Linux (в качестве основной используем Astra Linux)
Java для сервера приложений	Axiom JDK (Liberica), OpenJDK
Версия Java	Java 21

Сервер СУБД

Аппаратное обеспечение

Характеристика сервера	Рекомендуемые параметры
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

Программное обеспечение

Программное обеспечение	Рекомендуемые параметры
ОС	Astra Linux / Debian / РЕД ОС / ALT Linux (в качестве основной используем Astra Linux)
СУБД	Tantor, Postgres, Postgres Pro 14 и выше

Рабочие станции

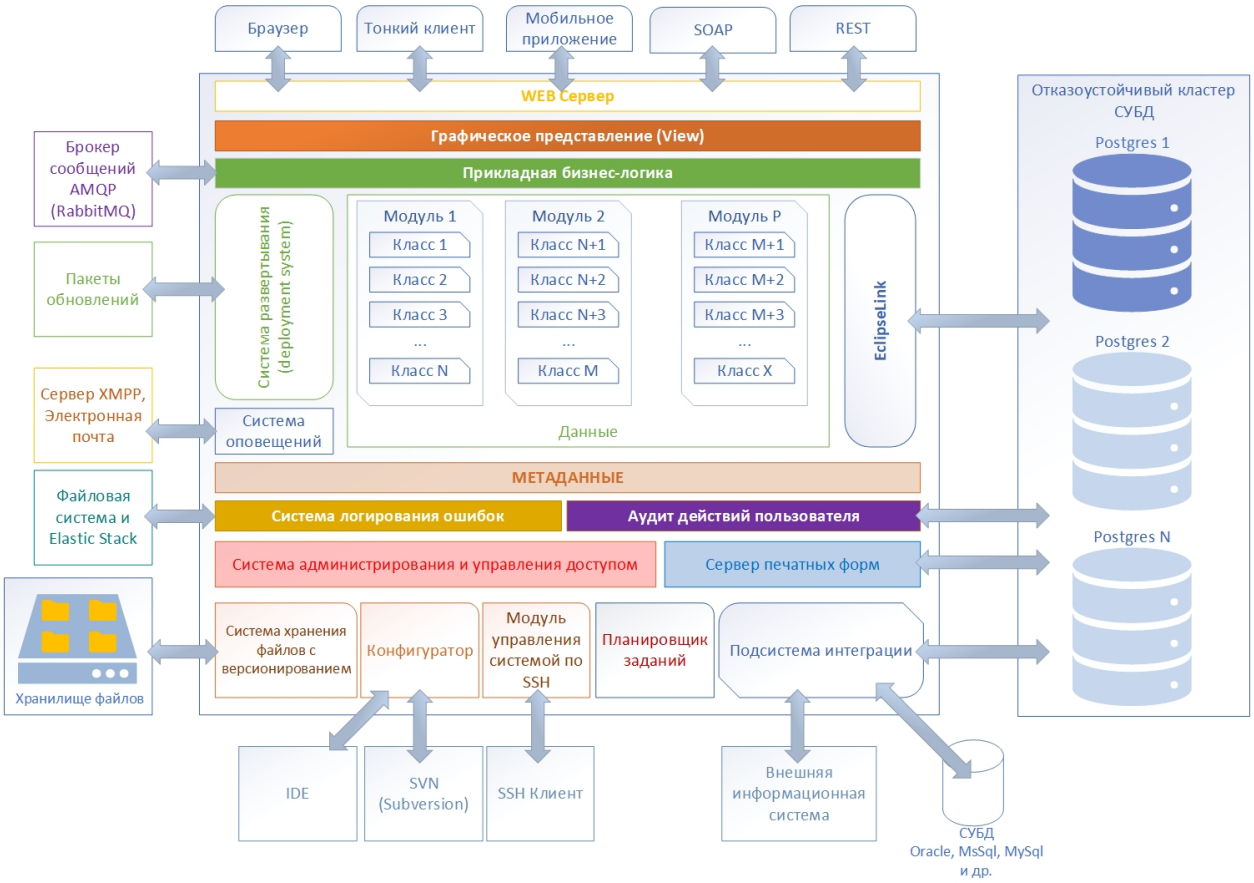
Аппаратное обеспечение

Характеристика	Рекомендуемые параметры
CPU	2 - 8 ядер 2.5 GHz или более
Оперативная память	4 – 32 Gb или более
Память	10 Gb свободного места
Сеть	1 Мбит или более с высоким качеством связи (без потерь)
Монитор	разрешение 1920×1080 (минимально 1140x900)

Программное обеспечение

Характеристика	Рекомендуемые параметры
ОС	Актуальная версия Linux с графической оболочкой или актуальная версия Windows. С поддержкой графического режима и современных браузеров
Клиентское приложение	Не снятые с поддержки браузеры: 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

1.4 Структура системы



1.5 Перечень основных подсистем

В программный комплекс Global входят следующие подсистемы:

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

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

2 Администрирование Postgres

2.1 Установка и настройка СУБД

Необходимое программное обеспечение

ПО под ОС Astra Linux / Debian / Alt Linux:

- Сервер PostgreSQL 14 или выше (Рекомендовано использовать PostgreSQL 17) 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.

Изменение локализации

Если при установке операционной системы выбрана англоязычная локаль, СУБД будет установлена с той же локалью. В этом случае локаль СУБД необходимо изменить на русскую `(ru_RU.UTF-8)` до выполнения любых конфигураций, поскольку смена локали приводит к сбросу всех настроек. Подробную инструкцию см. в документе: *Смена локалей для postgresql*.

Дополнительная настройка

- В конфигурационном файле `/etc/postgresql/*/main/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-ossp";
CREATE EXTENSION if not exists dict_xsyn;
CREATE EXTENSION if not exists ltree;
```

Завершите сессию psql

```
\q
```

Теперь БД готова к работе.

Развертывание поставочного дампа

Важно

Перед развертыванием поставочного дампа необходимо развернуть и зарегистрировать сервер приложений. Полный алгоритм действий:

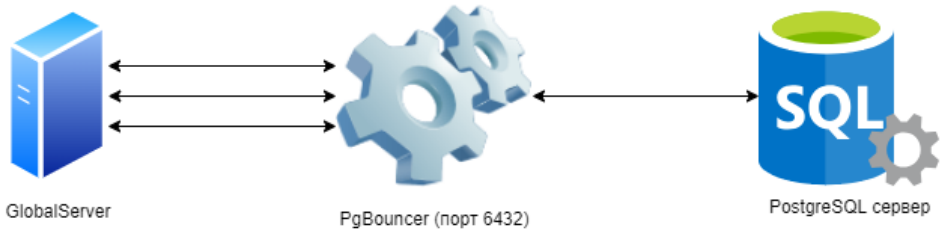
- Создать чистую базу, создать extentions
- Провести полную настройку контура
- Выполнить нагон релиза. Система определит, что производится чистая инициализация и сгенерирует все необходимые схемы и талблицы.
- Зарегистрировать контур лицензией
- Остановить Global (global3.service)
- Нагнать дамп схемы public (предварительно нужно очистить схему public)
- Запустить Global

Подробнее о восстановлении БД в *руководстве*

2.2 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 или 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
$$SELECT usename, passwd FROM pg_shadow WHERE usename = p_user$$;

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 usename = 'pgbouncer';
```

- Завершите сессию `psql`

```
\q
```

Конфигурация PgBouncer

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

- В раздел `[databases]` необходимо добавить адрес и порт, на которых 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
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 описанной выше функции `lookup()`:

```
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 использованием 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 файле службы 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

[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>"
```

Чтобы аутентификация работала с указанными настройками, в файл `userlist.txt` необходимо добавить следующую запись:

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

Запуск PgBouncer

После завершения настройки запустите PgBouncer с помощью службы:

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

По умолчанию PgBouncer будет доступен на порту 6432.

Безопасность

Рекомендуется закрыть основной порт PostgreSQL при использовании PgBouncer.

Для этого измените файл `pg_hba.conf`, разрешив только локальное подключение:

```
host    all             all             127.0.0.1/32            scram-sha-256
host    all             all             ::1/128                 scram-sha-256
```

Если требуется прямое подключение к СУБД с определённых IP-адресов, добавьте их дополнительно:

```
host    all             all             <IP_ADDR>/32    scram-sha-256
```

где `<IP_ADDR>` — IP-адрес хоста, которому разрешено прямое подключение к СУБД.

Пример:

```
host    all             all             192.168.1.12/32         scram-sha-256
```

При использовании брандмауэра задайте следующие правила для запрета прямого подключения:

```
ufw allow 6432/tcp
ufw allow from 127.0.0.1 to any port 5432 proto tcp
ufw deny 5432/tcp
```

Если требуется прямое подключение к СУБД с определённых IP-адресов, добавьте их дополнительно:

```
ufw allow from <IP_ADDR> to any port 5432 proto tcp
```

2.3 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 TEMPLATE template0;
```

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)
```

Интерпретация результатов

- `latency average` – среднее время отклика транзакции (мс).
- `tps` – транзакций в секунду (главный показатель производительности).
- `number of failed transactions` – процент неуспешных транзакций (должен быть близок к 0).
- `initial connection time` – время на установку соединений.

Основные ключи pgbench

- `-i` – инициализация базы данных (создаёт таблицы и заполняет их тестовыми данными).
- `-s N` – масштаб теста (увеличивает объём данных пропорционально N).
- `-c N` – число клиентов (соединений), которые будут одновременно выполнять транзакции.
- `-j N` – число потоков, обслуживающих клиентов. Обычно рекомендуется равным количеству CPU/ядрам.
- `-t N` – число транзакций на клиента. Общее число транзакций = c × t.
- `-T N` – длительность теста в секундах. Вместо количества транзакций тест ограничивается временем.
- `-S` – режим только SELECT (нагрузка только на чтение).
- `-N` – режим без обновлений балансов (исключает UPDATE из теста).
- `-P N` – вывод статистики каждые N секунд.
- `-M {simple|extended|prepared}` – режим выполнения SQL:
  - `simple` – обычные запросы (по умолчанию),
  - `extended` – расширенный протокол,
  - `prepared` – подготовленные выражения (ближе к реальным приложениям).
- `--progress-timestamp` – вывод прогресса с отметкой времени.
- `--protocol {simple|extended}` – протокол обмена с сервером.
- `--rate N` – ограничение нагрузки в N транзакций в секунду (TPS).
- `--aggregate-interval N` – усреднение результатов каждые N секунд (удобно для долгих тестов).
- `--random-seed {time|random|fixed}` – управление генерацией случайных чисел (для воспроизводимости).
- `--max-tries N` – максимальное число повторных попыток транзакции в случае ошибки.

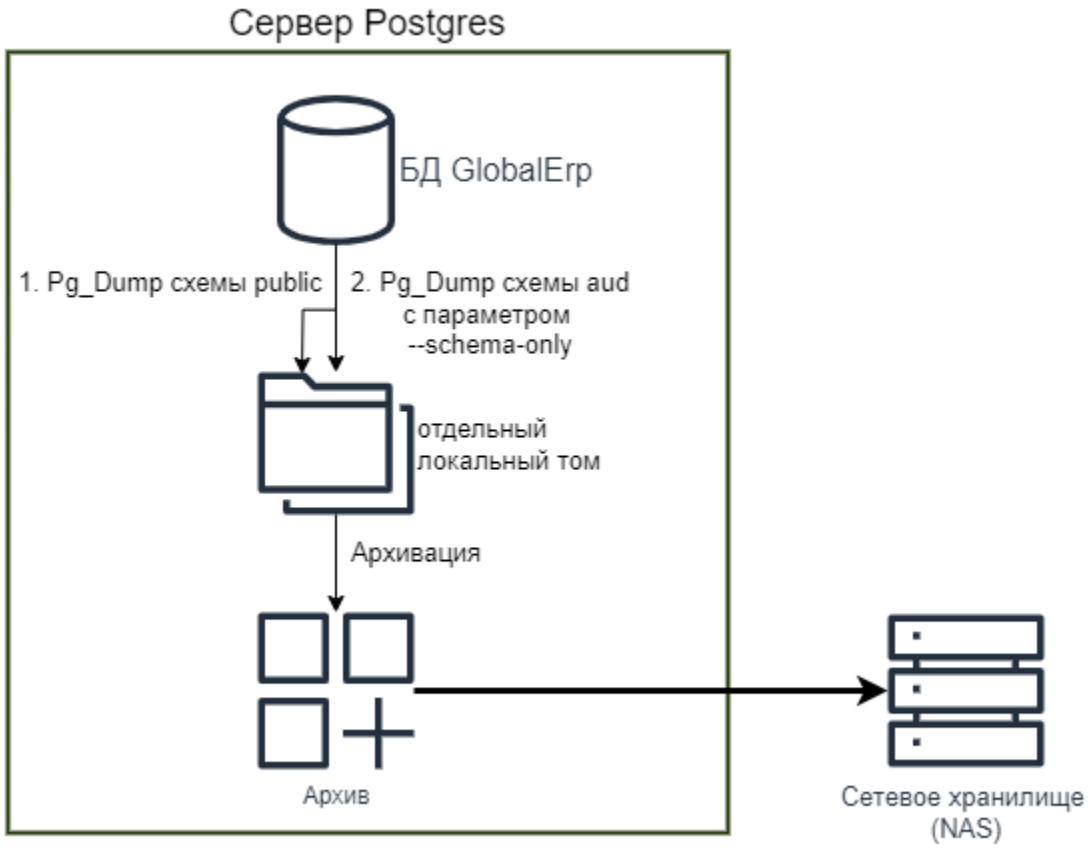
2.4 Резервное копирование и восстановление БД

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 дамп в многопоточном режиме и существенно ускоряет процесс.

Важно

В случае использования пуллеров соединений, таких как Pgbouncer, при снятии и восстановлении дампа к БД необходимо подключаться напрямую, минуя пуллер.

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

Внимание

При установке дампа с одного контура на другой, настройки интеграций и шедулера сохраняются, что может привести к проблемам с клонируемым контуром.

Эти настройки нужно обязательно сбрасывать при установке!

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;
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

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

Перед развертыванием дампа его нужно распаковать:

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

Запустите утилиту развертывания дампа. Первый вариант:

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

Второй вариант:

```
export PGPASSWORD='${sPass}'
$sRestoreUtl -h ${sHost} -U ${sUser} -d ${sDbName} -p 5432 --no-owner --no-privileges -
↪j 20 -v $sRealDumpDir > restore_30_10.log 2>&1
```

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

Отключение задач в шедуллере

После поднятия дампа необходимо отключить активные джобы (исключая системные btk), чтобы они не лазили наружу (интеграция например) Скрипт для отключения джоб

```
update Btk_Job
set benabled = 0
where (ssystemname is null) or (not starts_with(lower(ssystemname), 'btk_'))
```

JEXL-скрипт для сброса настроек шедулера

```
var JObject = function (value) { new ("ru.bitec.app.gtk.lang.json.JObject", value); }

//----------------------------------------
// Параметры шедулера
var soapHost = "localhost"; // Soap хост
var soapPort = 8080B;       // Soap порт (не забудьте 'B' в конце)
var soapReqStr = "/app/sys/soap/sys-service-1.0.0"; // Строка soap запроса
var database = "globalDb";     // База данных
var login = "admin";        // Логин
var pass = "admin";         // Пароль
var publicKey = "";         // Публичный ключ
var isHttpsEnabled = false; // HTTPS включён


//----------------------------------------
// Скрипт
var idvBtkModule = Btk_SettingApi.getModuleIdByMnemoCode("btk");

var jQuartzProps = null;
if (Btk_SettingApi.isExistsSetting(idvBtkModule, "cQuartzProps")) {
  jQuartzProps = JObject(Btk_SettingApi.getSetting(idvBtkModule, "cQuartzProps").
↪asJObject());
} else {
  jQuartzProps = JObject({:});
}

jQuartzProps.set("soapHost", soapHost);
jQuartzProps.set("soapPort", soapPort);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
jQuartzProps.set("soapReqStr", soapReqStr);
jQuartzProps.set("database", database);
jQuartzProps.set("login", login);
jQuartzProps.set("pass", pass);
jQuartzProps.set("publicKey", publicKey);
jQuartzProps.set("isHttpsEnabled", isHttpsEnabled);

Btk_SettingApi.registerJObject(idvBtkModule, "cQuartzProps", jQuartzProps.underlying(), "
↪");
commit();
```

JEXL-скрипт для отключения всех интеграций

```
// Скрипт отключения всех джобов контуров интеграций
for(row: sql(`
  select distinct t.idJob from Rpl_CircuitAgent t
`).asList()) {
  var rvCircuitJob = Btk_JobApi.load(row.idjob);
  Btk_JobApi.setbEnabled(rvCircuitJob, 0B);
}
commit();
```

BASH-скрипт выполнения JEXL-кода через API системы

Bash-скрипт позволяет:

- Загружать скрипты из внешних файлов
- Авторизоваться в системе
- Выполнять код на сервере
- Получать результаты выполнения

Скрипт:

```
#!/bin/bash

# Конфигурация
USERNAME="<username>"
PASSWORD="<password>"
DATABASE="<database>"
URL="<url>"
JEXL_SCRIPT_FILE="<path_to_script>"  # Укажите путь к файлу с JEXL-скриптом

# Проверка существования файла со скриптом
if [ ! -f "$JEXL_SCRIPT_FILE" ]; then
    echo "Ошибка: файл со скриптом '$JEXL_SCRIPT_FILE' не найден"
    exit 1
fi
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
# Чтение содержимого файла со скриптом
JEXL_SCRIPT=$(<"$JEXL_SCRIPT_FILE")

# Кодирование логина и пароля в base64
BASIC_AUTH=$(echo -n "$USERNAME:$PASSWORD" | base64)

# Формирование тела запроса с помощью jq
JSON_BODY=$(jq -n --arg jexl "$JEXL_SCRIPT" '{jexl: $jexl}')

# Отправка POST-запроса
curl -X POST "$URL/app/sys/rest/ss/pkg/Btk_JexlGatePkg/execute" \
  -H "Authorization: Basic $BASIC_AUTH" \
  -H "Database: $DATABASE" \
  -H "Content-Type: application/json" \
  -d "$JSON_BODY"
```

Требования

- **ПО**:
  - `curl` 7.68+
  - `jq` 1.6+
  - `bash` 5.0+
- **Доступ**:
  - К REST API системы
  - Чтение файлов скриптов

Конфигурация

Переменная	Описание	Пример значения
USERNAME	Логин администратора	admin
PASSWORD	Пароль	admin
DATABASE	Целевая БД	globalDb
URL	Базовый URL	http://globalERP.local/globalDb/
JEXL_SCRIPT_FILE	Петь к JEXL скрипту	/scripts/update_settings.jexl

Использование

Исправить переменные в скрипте на свои и запустить его.

```
bash run_jexl_api.sh
```

Подставляйте URL со страницы выбора приложений

Например:

На странице выбора приложений такой URL `http://globalERP.local/globalDb/?redirect-id=8`, следовательно в переменную подставляем `http://globalERP.local/globalDb/`

2.5 Смена локалей для postgresql

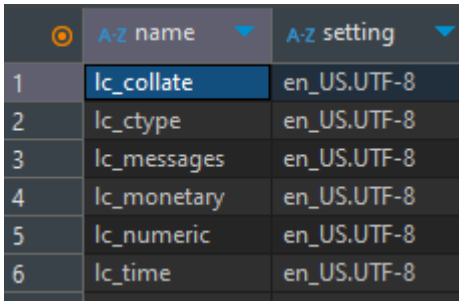
Руководство, как сменить локаль БД Postgresql

Проверка локализации в данный момент

- Выполнить sql-запрос

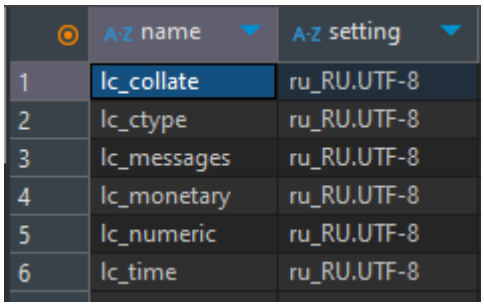
```
SELECT name, setting FROM pg_settings WHERE name LIKE 'lc_%';
```

- Пример английской локализации



	name	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



	name	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          +
           |          |          |             |             |            |                 | postgres=CTc/postgres
 template1 | postgres | UTF8     | ru_RU.UTF-8 | ru_RU.UTF-8 |            | libc            | =c/postgres          +
           |          |          |             |             |            |                 | postgres=CTc/postgres
(3 rows)
```

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';
```

2.6 Структура каталогов и файлов PostgreSQL

СУБД PostgreSQL в разных операционных системах имеет практически идентичную структуру каталогов. В данной статье рассматривается стандартная структура каталогов для ОС Ubuntu 12.04.

По умолчанию PostgreSQL устанавливается в папку /var/lib/postgresql//main. Основной каталог СУБД содержит подкаталоги с пользовательскими данными и служебной информацией.

- postmaster.opts - файл, в котором сдержится командная строка с параметрами, с помощью которой была запущена СУБД. Пример строки из файла postmaster.opts

```
/usr/lib/postgresql/16/bin/postgres "-D" "/var/lib/postgresql/16/main" "-c" "config_
↪file=/etc/postgresql/16/main/postgresql.conf"
```

- PG_VERSION - файл, содержащий основной номер версии СУБД (к примеру, 16)
- base - каталоги внутри base/ соответствуют OID баз данных. Это физические директории, где хранятся файлы таблиц, индексов и т.д. для каждой базы.

  Имена подкаталогов соответствуют OID'ам баз данных.

```
root@server: /var/lib/postgresql/16/main/base# ls
1/  12065/  12070/  16384/  24580/
```

```
user=# select oid, datname from pg_database;
oid   |  datname
-------+-----------
    1 | template1
12065 | template0
12070 | postgres
16384 | user
24580 | t1
(5 rows)
```

- global - Каталог global/ содержит данные, общие для всех БД — например, pg_database, pg_authid и др.
- В PostgreSQL 10 и выше pg_clog переименован в pg_xact и содержит информацию о статусах транзакций (по 2 бита на каждую).. Статус транзакции может иметь следующие значения:
  - транзакция стартовала,
  - транзакция успешно завершена,

– транзакция отменена,

– подтранзакция успешно завершена.

Количество хранимых транзакций ограничено параметром autovacuum_freeze_max_age (максимальное значение ~ 2 миллиарда), который в свою очередь регулирует работу автовакума. Поэтому максимальный размер файла может быть ~ 0.5 Гб максимум.

- pg_multixact - каталог, содержащий информацию, необходимую для координации работы параллельных транзакций (используется для хранения SHARED ROW LOCKS)
- pg_notify - каталог, в котором хранится информация для поддержки работы LISTEN/NOTIFY
- pg_serial - каталог, содержащий информацию о завершенных последовательных (serializable) транзакциях
- pg_snapshots - каталог, в котором хранятся экспортированные снапшоты

  Файлы создаются при вызове процедуры pg_export_snapshot() и существуют до окончания транзакции.

  Ниже представлена последовательность действий, позволяющая понять, как появляются и исчезают файлы в этом каталоге.

  Экспорт снапшота в транзакции

```
BEGIN
user=# SELECT pg_export_snapshot();
pg_export_snapshot
--------------------
00000312-1
(1 row)
```

  В каталоге появился файл, в котором содержится информация о снапшоте

```
root@server: /var/lib/postgresql/16/main/pg_snapshots# ls
00000312-1

root@server: /var/lib/postgresql/16/main/pg_snapshots# cat 00000312-1

xid:786
dbid:16384
iso:1
ro:0
xmin:786
xmax:786
xcnt:0
sof:0
sxcnt:0
rec:0
```

  Отмена транзакции

```
user=# rollback;
ROLLBACK
```

  Файл удален

```
root@server: /var/lib/postgresql/16/main/pg_snapshots# l
----
```

- pg_stat_tmp - каталог, содержащий информацию необходимую для поддержания работы системы статистики. Сборщик статистики использует этот каталог для передачи информации другим процессам СУБД. Этот каталог может быть перемещен в оперативную память для уменьшение нагрузки на жесткий диск.
- pg_subtrans - каталог, в котором хранится информация о подтранзакциях
- pg_tblspc - в каталоге хранятся символьные ссылки на табличные пространства (tablespaces).

  Ниже показана последовательность действий, демонстрирующая, как в этом каталоге появляются файлы.

  Создается табличное пространство

```
user=# CREATE TABLESPACE tmp LOCATION '/home/user/1';
CREATE TABLESPACE
```

  Получается список табличных пространств

```
user=# select oid, spcname  from pg_tablespace;
oid  |  spcname
-------+------------
1663 | pg_default
1664 | pg_global
27936 | tmp
(3 rows)
```

  Для каждого пользовательского табличного пространства в каталоге создается символьная ссылка на заданный каталог, именем этой ссылки является OID этого табличного пространства.

```
/var/lib/postgresql/16/main/pg_tblspc# ll
total 8
drwx------  2 postgres postgres 4096 Nov  4 09:51 ./
drwx------ 15 postgres postgres 4096 Nov  4 07:05 ../
lrwxrwxrwx  1 postgres postgres   13 Nov  4 09:51 27936 -> /home/user/1/
```

- pg_twophase - каталог содержит информацию о подготовленных транзакциях для двухфазного коммита.
- Каталог pg_xlog (в PostgreSQL 10+ — pg_wal) содержит WAL-файлы (журнал предзаписи).

2.7 Инструкция по установке и настройке кластера СУБД Postgresql

Исппользуемые термины

Термин	Значение / расшифровка (рекомендуется добавить в инструкцию)
NTP	Network Time Protocol — протокол синхронизации времени
VIP	Virtual IP — виртуальный IP-адрес
NFS	Network File System — протокол сетевых файловых систем
pgbouncer	Лёгкий пул соединений для PostgreSQL
pcsd, corosync, pacemaker, sbd	Компоненты кластерного ПО (Pacemaker stack)
CRM	Cluster Resource Manager (часть Pacemaker)
STONITH	Shoot The Other Node In The Head — механизм fencing в кластере
HACLUSTER	Системный пользователь для управления кластером
pg_stat_statements	Расширение PostgreSQL для анализа запросов
pg_basebackup	Утилита PostgreSQL для создания резервной копии
pg_probackup	Расширенная утилита для резервного копирования PostgreSQL
WAL	Write Ahead Log — журнал предзаписи PostgreSQL
sysctl	Инструмент управления параметрами ядра Linux
udevadm	Утилита управления правилами устройств (udev)
.pgpass	Файл с паролями для автоматического подключения к PostgreSQL
pg_ctl	Утилита управления экземпляром PostgreSQL
promote/demote	Повышение/понижение роли реплики в кластере
crm_mon	Мониторинг состояния ресурсов кластера
SBD	STONITH Block Device — метод fencing с использованием устройств
HA	High Availability — высокая доступность

Подготовка ОС

Требования для установки кластера СУБД

- Установлена ОС ALT Linux v.10.2, подключен официальный репозиторий вендора ОС.
- Минимальные требования для узла с ролью голосующий – 4 vCPU, 4GB RAM, 50GB Hard Drive
- СУБД – PostgreSQL v.16.4 (или новее).
- Сконфигурирована синхронизация времени по ntp;
- Обеспечено беспрепятственное прохождение трафика между узлами по портам, указанным в таблице.

Таблица узлов

Роль узла	DNS имя	IP адрес	Порты
сервер БД	alt-db-1	IP.alt-db-1	22,2224,5405,5432,6432
сервер БД	alt-db-2	IP.alt-db-2	22,2224,5405,5432,6432
голосующий	alt-voter	IP.alt-voter	22,2224,5405
nfs	alt-nfs-1	IP.alt-nfs-1	111,2049
nfs	alt-nfs-2	IP.alt-nfs-2	111,2049
Виртуальный IP	-	VIP-master	5432
Виртуальный IP	-	VIP-pgbouncer	6432

- Имена узлов, указанных в таблицах, должны разрешаться службой DNS или вручную указаны в /etc/hosts (предпочтительнее)
- Среднее время задержки между узлами не должно превышать 5мс
- Пропускная способность между узлами должна быть не менее 1 Гбит/с
- Скорость записи и чтения на сетевые папки NFS с узлов СУБД должна быть не менее 500 МБ/с
- Смонтированы необходимые NFS разделы:
  - На узле с СУБД на основном ЦОД монтируется по NFS общая сетевая папка backup с СХД локальной площадки как /bkp-local, общая сетевая папка backup с СХД удаленной площадки как /bkp-rmt.
  - На узле с СУБД на резервном ЦОД монтируется по NFS общая сетевая папка backup с СХД локальной площадки как /bkp-local, общая сетевая папка backup с СХД удаленной площадки как /bkp-rmt.

Схема монтирования сетевых папок NFS

Имя сетевой папки NFS	Точка монтирования на узле alt-db-1	Точка монтирования на узле alt-db-2
IP.alt-nfs-1:/backup	/bkp-local	/bkp-rmt
IP.alt-nfs-2:/backup	/bkp-rmt	/bkp-local

Пример: На `alt-db-1`

```
sudo mkdir /bkp-local
sudo mkdir /bkp-rmt
sudo mount -t nfs -o rw,hard,vers=4.2 IP.alt-nfs-1:/mnt/nfs-share/ /bkp-local
sudo mount -t nfs -o rw,hard,vers=4.2 IP.alt-nfs-2:/mnt/nfs-share/ /bkp-rmt
```

На `alt-db-2`

```
sudo mkdir /bkp-local
sudo mkdir /bkp-rmt
sudo mount -t nfs -o rw,hard,vers=4.2 IP.alt-nfs-2:/mnt/nfs-share/ /bkp-local
sudo mount -t nfs -o rw,hard,vers=4.2 IP.alt-nfs-1:/mnt/nfs-share/ /bkp-rmt
```

Создание файлов

Создать файл `/etc/security/limits.d/99-pg.conf` с содержимым

(настройки приведены для случая max_connections=500)

```
postgres softnproc 16384
postgres hardnproc 16384
postgres softnofile 65536
postgres hardnofile 65536
postgres softstack 10240
postgres hardstack 32768
```

Создать файл `/etc/sysctl.d/99-pg.conf` с содержимым

```
kernel.sysrq = 1
kernel.core_pattern = /tmp/core-%e-%s-%u-%g-%p-%t
vm.mmap_min_addr = 65536
vm.swappiness = 1
vm.dirty_ratio = 80
vm.dirty_expire_centisecs = 500
vm.dirty_writeback_centisecs = 100
kernel.shmmax = 4398046511104
kernel.shmall = 1073741824
kernel.shmmni = 4096
fs.file-max = 6815744
fs.aio-max-nr = 1048576
net.ipv4.ip_local_port_range = 9000 65500
net.core.rmem_default = 262144
net.core.rmem_max = 4194304
net.core.wmem_default = 262144
net.core.wmem_max = 1048576
kernel.sched_migration_cost_ns = 5000000
kernel.sched_autogroup_enabled = 0
kernel.sem = 250 512000 100 2048
```

Применить настройки командой

```
sysctl -p /etc/sysctl.d/99-pg.conf
```

Примечание

Примечание. Приведенные настройки могут быть изменены в зависимости от нагрузки и количества подключений к СУБД.

Заполнить файл hosts

Дополните файл `hosts` на каждом узле, пример содержания приведен ниже

```
127.0.0.1       localhost.localdomain localhost
192.168.242.200 alt-db-1
192.168.242.201 alt-db-2
192.168.242.202 alt-voter
192.168.242.203 alt-nfs-1
192.168.242.204 alt-nfs-2
::1     localhost6.localdomain localhost6
```

Выполнить рекомендации по использованию огромных страниц

https://postgrespro.ru/docs/postgresql/13/kernel-resources%23LINUX-HUGE-PAGES

Для узлов в виде виртуальных машин

Необходимо изменить значение в /sys/block/sda(b,c)/queue/rotational с текущего 1 на 0, чтобы ОС определяла диск как ssd

Для проверки значения rotational для дисков в ОС необходимо выполнить команду:

```
lsblk -d -o name,rota
```

Примерный вывод должен быть таким:

```
NAME ROTA
sda 0
sdb 0
...
```

Если параметр будет равен 1 для дисковых устройств, то необходимо изменить его на 0. Для этого необходимо выполнить следующие действия

1) Создать файл 60-ssd.rules

   ```
   touch /etc/udev/rules.d/60-ssd.rules
   ```

   с содержимым

   ```
   ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}="0"
   ```

2) Проверить корректность и срабатывание правила командой (указан пример для диска sda):

   ```
   udevadm test -a add $(udevadm info -q path -n /dev/sda)
   ```

3) Убедиться, что значение rotational =0 командой:

   ```
   lsblk -d -o name,rota
   ```

Установка и настройка PostgreSQL

Установка PostgreSQL

Установить PostgreSQL v16 на `alt-db-1` и `alt-db-2`:

```
sudo apt-get install postgresql16-server postgresql16-contrib
```

Далее инициализировать базу данных. Для этого на `alt-db-1` выполнить команду от имени системного пользователя `postgres`:

```
sudo -u postgres initdb -D /var/lib/pgsql/data/ --lc-ctype=ru_RU.UTF-8 --lc-collate=ru_
↪RU.UTF-8 --auth-host=scram-sha-256 --data-checksums
```

Примечание

/var/lib/pgsql/data/ - директория базы данных, может быть изменена при необходимости.

Настройка PostgreSQL

Настройка СУБД на alt-db-1

Отредактировать файл конфигурации postgresql.conf

Примечание

Файл находиться по пути /var/lib/pgsql/data/postgresql.conf

Рекомендуемое содержимое конфигурационного файла postgresql.conf для конфигурации оборудования (CPU 64 core, RAM 512 GB) приведено ниже:

```
# - Connection Settings -
listen_addresses = '*'
port = 5432
max_connections = 1000
# - Memory -
shared_buffers = 64GB
max_prepared_transactions = 3000
temp_buffers = 32MB
work_mem = 128MB
maintenance_work_mem = 50GB
# - Asynchronous Behavior -
effective_io_concurrency = 2
# - Background Writer -
bgwriter_delay = 20ms
bgwriter_lru_multiplier = 4.0
bgwriter_lru_maxpages = 400
# WRITE AHEAD LOG
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
wal_level = replica
fsync = on
synchronous_commit = on
wal_compression = on
wal_log_hints = on
# - Checkpoints -
checkpoint_timeout = 1800
checkpoint_completion_target = 0.9
# - Archiving -
archive_mode = always
archive_command = 'pg_probackup-16 archive-push -B /bkp-local/backup --instance global --
↪compress --wal-file-path %p --wal-file-name %f'
# REPLICATION
max_wal_senders = 15
wal_keep_size = 20GB
max_wal_size = 20GB
min_wal_size = 5GB
max_replication_slots = 8
hot_standby = on
hot_standby_feedback = on
tcp_keepalives_idle = 60
tcp_keepalives_interval = 5
tcp_keepalives_count = 5
track_commit_timestamp = on
# QUERY TUNING
effective_cache_size = 512GB
# ERROR REPORTING AND LOGGING
logging_collector = on
log_min_duration_statement = 0
# AUTOVACUUM PARAMETERS
autovacuum = on
log_autovacuum_min_duration = 0
autovacuum_max_workers = 32
autovacuum_naptime = 20s
autovacuum_vacuum_scale_factor = 0.01
autovacuum_analyze_scale_factor = 0.001
autovacuum_vacuum_cost_delay = 10
autovacuum_vacuum_cost_limit = 1000
# LOCK MANAGEMENT
max_locks_per_transaction = 150
# ERROR HANDLING
restart_after_crash = off
# CUSTOMIZED OPTIONS
pg_stat_statements.max=10000
pg_stat_statements.track = all
```

Примечание

Указанные выше параметры могут быть изменены при использовании другого оборудования.

Отредактировать файл конфигурации доступов pg_hba.conf

Примечание

Файл находиться по пути /var/lib/pgsql/data/pg_hba.conf

```
local   backupdb    backup                                  scram-sha-256
local   replication backup                                  scram-sha-256
host    template1   postgres        X.X.X.0/24              scram-sha-256
host    all         rewind_user     X.X.X.0/24              scram-sha-256
host    replication repl            X.X.X.0/24              scram-sha-256
host    replication backup          X.X.X.0/24              scram-sha-256
host    backupdb    backup          127.0.0.1/32            scram-sha-256
host    backupdb    backup          X.X.X.0/24              scram-sha-256
```

Запустить сервер PostgreSQL

```
sudo -u postgres pg_ctl -D /var/lib/pgsql/data/ start
```

После запуска рекомендуется проверить отсутствие ошибок в лог-файле postgresql.log.

Проверить запуск СУБД можно командой:

```
ps auwx | grep postgres
```

Создать пользователей

Подключаемся к БД

```
sudo -u postgres psql
```

Выполняем следующие команды

```
CREATE ROLE repl WITH LOGIN REPLICATION PASSWORD 'password';
CREATE ROLE backup WITH LOGIN REPLICATION PASSWORD 'password';
GRANT USAGE ON SCHEMA pg_catalog TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.current_setting(text) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.set_config(text, text, boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_is_in_recovery() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_backup_start(text, boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_backup_stop(boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_create_restore_point(text) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_switch_wal() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_last_wal_replay_lsn() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current_snapshot() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_snapshot_xmax(txid_snapshot) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_control_checkpoint() TO backup;
COMMIT;
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
CREATE USER rewind_user LOGIN PASSWORD 'password';
GRANT EXECUTE ON function pg_catalog.pg_ls_dir(text, boolean, boolean) TO rewind_user;
GRANT EXECUTE ON function pg_catalog.pg_stat_file(text, boolean) TO rewind_user;
GRANT EXECUTE ON function pg_catalog.pg_read_binary_file(text) TO rewind_user;
GRANT EXECUTE ON function pg_catalog.pg_read_binary_file(text, bigint, bigint, boolean)␣
↪TO rewind_user;
```

Из соображений безопасности для создания бекапов рекомендуется создать и использовать отдельную базу данных `backupdb`.

Необходимо создать БД `backupdb`:

```
CREATE DATABASE backupdb OWNER backup;
```

Изменить при необходимости пароль `postgres`:

```
ALTER USER postgres PASSWORD 'password';
```

Проинициализировать файл паролей

В домашнем каталоге пользователя `postgres` (по умолчанию /var/lib/pgsql), выполнив команду на 2х узлах с СУБД:

```
echo 'VIP-master:5432:replication:repl:pass-for-repl' >> /var/lib/pgsql/.pgpass
echo '*:5432:backupdb:backup:pass-for-backup' >> /var/lib/pgsql/.pgpass
echo '*:5432:*:rewind_user:pass-for-rewind' >> /var/lib/pgsql/.pgpass
echo '*:5432:template1:postgres:pass-for-postgres' >> /var/lib/pgsql/.pgpass
chmod 600 /var/lib/pgsql/.pgpass
chown postgres:postgres /var/lib/pgsql/.pgpass
```

Установка и настройка pg_probackup

Установка pg_probackup на alt-db-1 и alt-db-2

Установить `pg_probackup` на узлах с СУБД (`alt-db-1` и `alt-db-2`):

```
wget https://repo.postgrespro.ru/pg_probackup/rpm/latest/altlinux-p10/x86_64/RPMS.
↪vanilla/pg_probackup-16-2.5.15-2.ac92457c2d1cfe43fced5b1167b5c90ecdc24cbe.x86_64.rpm

sudo rpm -i pg_probackup-16-2.5.15-2.ac92457c2d1cfe43fced5b1167b5c90ecdc24cbe.x86_64.rpm
```

Инициализация pg_probackup на alt-db-1

```
sudo -u postgres pg_probackup-16 init -B /bkp-local/backup
sudo -u postgres pg_probackup-16 add-instance -B /bkp-local/backup -D /var/lib/pgsql/
↪data/ --instance global
sudo -u postgres pg_probackup-16 set-config -B /bkp-local/backup --instance global --log-
↪filename=pg_probackup-global-%d.log
```

Проверка успешности инициализации `pg_probackup`

```
sudo -u postgres pg_probackup-16 backup -B /bkp-local/backup --instance global -b FULL -
↪U backup -d backupdb --compress --stream -j 6
sudo -u postgres pg_probackup-16 show -B /bkp-local/backup --instance global
```

Инициализация pg_probackup на alt-db-2

```
sudo -u postgres pg_probackup-16 init -B /bkp-local/backup
sudo -u postgres pg_probackup-16 add-instance -B /bkp-local/backup -D /var/lib/pgsql/
↪data/ --instance global
sudo -u postgres pg_probackup-16 set-config -B /bkp-local/backup --instance global --log-
↪filename=pg_probackup-global-%d.log
```

Создание копии БД на alt-db-2

С использованием утилиты pg_basebackup.

Выполнить команду:

```
sudo -u postgres pg_basebackup -D /var/lib/pgsql/data/ -h alt-db-1 -U repl -X stream -c␣
↪fast
```

Восстановить из резервной копии с помощью pg_probackup.

Перед восстановлением необходимо узнать идентификатор полного бэкапа (`backup_id`):

```
sudo -u postgres pg_probackup-16 show -B /bkp-rmt/backup --instance global
```

Выполняем восстановление:

```
sudo -u postgres pg_probackup-16 restore -B /bkp-rmt/backup --instance global -i backup_
↪id
```

Скопировать с помощью pg_probackup+catchup.

Выполнить команду (потребуется ввести пароль пользователя postgres):

```
sudo -u postgres pg_probackup-16 catchup
--source-pgdata /var/lib/pgsql/data/ \
--destination-pgdata /var/lib/pgsql/data/ \
-p 5432 -d backupdb -U backup \
--stream \
--backup-mode=FULL \
--remote-host=hostname1 \
--remote-user=postgres \
--remote-path=/usr/bin/ \
-j 6
```

Установка и настройка pgbouncer

Установка

Выполнить на узлах `alt-db-1` и `alt-db-2`:

```
sudo apt-get install pgbouncer
```

Настройка

Необходимо повысить лимиты для сервиса pgbouncer.

```
sudo systemctl edit pgbouncer
```

Дописать в начало файла

```
[Service]
LimitNOFILE=10240
```

Примечание

10240 – максимальное число открытых файлов (при необходимости можно изменить)

Перечитать конфигурацию systemd:

```
sudo systemctl daemon-reload
```

Изменить конфигурационный файл `/etc/pgbouncer/pgbouncer.ini` в соответствии с требуемыми настройками `pgbouncer`.

```
[databases]
* = host=localhost port=5432 auth_user=postgres
[pgbouncer]
listen_addr = *
listen_port = 6432
```

Установка и настройка кластерного ПО

Установка

Выполнить команду

```
sudo apt-get install pcs corosync pacemaker sbd
```

Добавить в файл `/etc/bashrc.d/ruby.sh` если он пуст:

```
export PATH=$PATH:/var/cache/ruby/gemie/bin:/usr/lib/ruby/bin
export GEM_VENDOR=/var/cache/ruby/gemie
export GEM_HOME=/var/cache/ruby/gemie
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
export GEM_PATH=/var/cache/ruby/gemie:/usr/lib/ruby/gemie:/usr/lib64/ruby/gemie:/usr/lib/
↪ruby/gems/2.7.0/:/usr/lib/ruby/gems/2.5.0/
```

Настройка watchdog в кластере

Примечание

Рекомендуется с помощью среды виртуализации эмулировать аппаратный watchdog i6300esb для виртуальных машин кластера СУБД. Если нет возможности эмулировать аппаратный watchdog, то необходимо использовать программный - softdog.

Внимание

Одновременно использовать и аппаратный и программный watchdog нельзя!

Использование аппаратного watchdog

Создать на всех узлах файл `/etc/rc.d/rc.local` с содержимым:

```
#!/bin/sh
/sbin/modprobe i6300esb
```

Сделать файл `/etc/rc.d/rc.local` исполняемым:

```
chmod +x /etc/rc.d/rc.local
```

Затем перезагрузить все узлы и проверить загрузку модуля ядра `i6300esb` командой:

```
lsmod | grep i6300esb
```

Проверить на всех узлах, что в ОС появилось устройство `/dev/watchdog`:

```
ls /dev/watchdog
```

Использование программного watchdog

Создать на всех узлах файл `/etc/rc.d/rc.local` с содержимым:

```
#!/bin/sh
/sbin/modprobe softdog
```

Сделать файл `/etc/rc.d/rc.local` исполняемым:

```
chmod +x /etc/rc.d/rc.local
```

Затем перезагрузить все узлы и проверить загрузку модуля ядра `softdog` командой:

```
lsmod | grep softdog
```

Проверить на всех узлах, что в ОС появилось устройство `/dev/watchdog`:

```
ls /dev/watchdog
```

Настройка основных ресурсов

Подготовка ОС к использованию кластерного ПО

На всех узлах кластера включить автозапуск сервиса `pcsd`

```
sudo systemctl enable pcsd.service
sudo systemctl start pcsd.service
```

На всех узлах кластера выключить автозапуск сервиса `postgresql` (для узлов с СУБД), `corosync` и `pacemaker` (для всех узлов):

```
sudo systemctl disable postgresql
sudo systemctl disable corosync.service pacemaker.service
```

На всех узлах кластера задать пароль пользователя hacluster:

```
sudo passwd hacluster
```

Конфигурация кластера

Примечание

Далее все команды выполняются на 1 узле, к примеру alt-voter

Выполнить команду авторизации узлов в кластере:

```
sudo pcs host auth -u hacluster alt-db-1 alt-db-2 alt-voter
```

Выполнить предварительную настройку кластера.

```
sudo pcs cluster setup global-cluster alt-db-1 alt-db-2 alt-voter
```

Запустить кластер:

```
sudo pcs cluster start --all --wait=60
```

Создать конфигурационный файл кластера:

```
sudo pcs cluster cib cluster.cfg
```

Изменить свойства кластера:

```
sudo pcs -f cluster.cfg resource defaults update resource-stickiness="1000"
sudo pcs -f cluster.cfg resource defaults update migration-threshold="3"
```

Создать ресурсы виртуальных IP:

```
sudo pcs -f cluster.cfg resource create vip-master ocf:heartbeat:IPaddr2 cidr_netmask=24␣
↪ip=<VIP-master>
sudo pcs -f cluster.cfg resource create vip-pgbouncer ocf:heartbeat:IPaddr2 cidr_
↪netmask=24 ip=<VIP-pgbouncer>
```

Создать ресурс pgbouncer:

```
sudo pcs -f cluster.cfg resource create pgbouncer systemd:pgbouncer
```

Создать ресурс pgsql:

```
sudo pcs -f cluster.cfg resource create pgsql \
ocf:heartbeat:pgsql \
master_ip=<VIP-master> \
node_list="alt-db-1 alt-db-2" \
rep_mode=sync \
primary_conninfo_opt="keepalives_idle=60 keepalives_interval=5 keepalives_count=5" \
repuser=repl \
restore_command='pg_probackup-16 archive-get -B /bkp-rmt/backup --instance global --wal-
↪file-path=%p --wal-file-name=%f'
```

Указать ресурсу pgsql параметры клонирования и промоута:

```
sudo pcs -f cluster.cfg resource promotable pgsql promoted-max=1 promoted-node-max=1␣
↪clone-max=2 notify=true clone-node-max=1
```

Запретить ресурсам стартовать на голосующем узле:

```
sudo pcs -f cluster.cfg constraint location pgsql-clone avoids alt-voter
sudo pcs -f cluster.cfg constraint location vip-master avoids alt-voter
sudo pcs -f cluster.cfg constraint location pgbouncer avoids alt-voter
sudo pcs -f cluster.cfg constraint location vip-pgbouncer avoids alt-voter
```

Указать совместное расположение ресурсов pgsql, pgbouncer, vip-pgbouncer и vip-master:

```
sudo pcs -f cluster.cfg constraint colocation add vip-master with master pgsql-clone␣
↪INFINITY
sudo pcs -f cluster.cfg constraint colocation add vip-pgbouncer with master pgsql-clone␣
↪INFINITY
sudo pcs -f cluster.cfg constraint colocation add pgbouncer with master pgsql-clone␣
↪INFINITY
```

Определить порядок запуска ресурсов

```
sudo pcs -f cluster.cfg constraint order promote pgsql-clone then vip-master␣
↪score=INFINITY symmetrical=false
sudo pcs -f cluster.cfg constraint order demote pgsql-clone then stop vip-master score=0␣
↪symmetrical=false
sudo pcs -f cluster.cfg constraint order promote pgsql-clone then vip-pgbouncer␣
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪score=INFINITY symmetrical=false
sudo pcs -f cluster.cfg constraint order demote pgsql-clone then stop vip-pgbouncer␣
↪score=0 symmetrical=false
```

Загрузить конфигурационный файл ресурсов в кластер

```
sudo pcs cluster cib-push cluster.cfg
```

Настройка фенсинга в кластере СУБД

Для настройки фенсинга в кластере необходимо выполнить последовательность команд:

```
sudo pcs stonith sbd enable SBD_STARTMODE=clean
```

Далее потребуется рестарт кластера:

```
sudo pcs cluster stop --all
sudo pcs cluster start --all --wait=60
```

Далее выполнить команды:

```
sudo pcs property set stonith-timeout="12s"
sudo pcs property set stonith-watchdog-timeout="6s"
sudo pcs property set cluster-recheck-interval="1m"
sudo pcs property set fence-reaction="panic"
sudo pcs property set no-quorum-policy=suicide
```

Проверка кластера

Для проверки состояния кластера

```
sudo crm_mon -Afr
```

Примерный вывод должен быть таким:

```
Cluster Summary:
  * Stack: corosync (Pacemaker is running)
  * Current DC: alt-voter (version 2.1.9-alt1-2.1.9) - partition with quorum
  * Last updated: Tue Aug 12 10:38:15 2025 on alt-voter
  * Last change:  Tue Aug 12 10:38:13 2025 by root via root on alt-db-2
  * 3 nodes configured
  * 5 resource instances configured

Node List:
  * Online: [ alt-db-1 alt-db-2 alt-voter ]

Full List of Resources:
  * vip-master  (ocf:heartbeat:IPaddr2):         Started alt-db-1
  * vip-pgbouncer       (ocf:heartbeat:IPaddr2):         Started alt-db-1
  * pgbouncer   (systemd:pgbouncer):     Started alt-db-1
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
  * Clone Set: pgsql-clone [pgsql] (promotable):
    * Promoted: [ alt-db-1 ]
    * Unpromoted: [ alt-db-2 ]

Node Attributes:
  * Node: alt-db-1:
    * master-pgsql                      : 1000
    * pgsql-data-status                 : LATEST
    * pgsql-master-baseline             : 00000000101B4E08
    * pgsql-status                      : PRI

  * Node: alt-db-2:
    * master-pgsql                      : 100
    * pgsql-data-status                 : STREAMING|SYNC
    * pgsql-status                      : HS:sync

Migration Summary:
```

Если у вас статус одной из нод `HS:alone` и `disconected`, то проверьте содержимое файла `/var/lib/pgsql/tmp/recovery.conf` на всех нода с СУБД.

Пример содержания на `alt-db-1`:

```
primary_conninfo = 'host=192.168.242.205 port=5432 user=repl application_name=alt-db-1␣
↪keepalives_idle=60 keepalives_interval=5 keepalives_count=5'
restore_command = 'pg_probackup-16 archive-get -B /bkp-rmt/backup --instance global --
↪wal-file-path=%p --wal-file-name=%f'
recovery_target_timeline = 'latest'
```

Если не хватает данных, то допишите их.

Дополнительная информация

Создание резервных копий виртуальных машин кластера

После успешного запуска кластера необходимо создать резервные копии виртуальных машин кластера в виде образов ВМ на случай катастрофы ДЦ1 или ДЦ2.

1. Для выполнения данной задачи необходимо сначала остановить кластер:

   ```
   sudo pcs cluster stop --all
   ```

2. Остановить виртуальные машины кластера.
3. Выполнить на всех гипервизорах кластера резервное копирование виртуальных машин без хранилищ с БД.
4. Сохранить образы виртуальных машин на всех NFS-серверах.
5. Запустить виртуальные машины кластера.
6. Удалить файл блокировки на мастере:

   ```
   sudo rm /var/lib/pgsql/tmp/PGSQL.lock
   ```

7. Запустить кластер командой:

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

8. В консоли любого узла проконтролировать состояние кластера командой crm_mon:

```
sudo crm_mon -Afr
```

Изменение каких-либо параметров ресурса кластера

При необходимости можно изменить или добавить необходимые параметры ресурса кластера. Например, при необходимости изменить параметр `restore_command` ресурса `pgsql` выполнить следующие шаги и команды.

1. Деактивировать ресурс `pgsql`:

```
sudo pcs resource disable pgsql
```

2. Выполнить команду обновления нужного параметра, например, `restore_command`:

```
sudo pcs resource update pgsql restore_command='новая команда'
```

3. Удалить файл блокировки на мастере:

```
sudo rm /var/lib/pgsql/tmp/PGSQL.lock
```

4. Активировать ресурс `pgsql`:

```
sudo pcs resource enable pgsql
```

5. Убедиться с помощью монитора кластера, что в кластере появился мастер и реплика

```
sudo crm_mon -Afr
```

Удаление (разбор) кластера

Внимание

Внимание. Кластер будет разобран, сервис СУБД будет недоступен.

Для удаления кластерной конфигурации необходимо выполнить следующие шаги.

1. Остановить кластер:

```
sudo pcs cluster stop --all
```

2. Затем удалить кластер:

```
sudo pcs cluster destroy --all
```

Статусы ресурсов кластера

В различных состояниях кластера ресурсы могут иметь разные статусы.

- Ресурс `pgsql`
  - Unpromoted – статус указывает на работу данного узла в режиме реплики.
  - Promoted – статус указывает на работу данного узла в режиме мастера.
  - Disabled – статус указывает на заблокированное (неактивное) состояние ресурса.
  - Stopped – ресурс остановлен.

Также состояние ресурса `pgsql` указано по каждому узлу более подробно в секции `Node Attributes`:

- `master-pgsql`
  - `1000` – score соответствует состоянию мастера
  - `100` – score соответствует синхронной реплике
  - `-INFINITY` – score соответствует асинхронной реплике или реплике без мастера Примечание. Значение -INFINITY («минус бесконечность») у асинхронной реплики не дает кластеру в случае сбоя на мастере переключиться на асинхронную реплику.
- `pgsql-status`
  - `PRI` – соответствует состоянию мастера
  - `HS:sync` – соответствует синхронной реплике
  - `HS:async` – соответствует асинхронной реплике
  - `HS:alone` – соответствует режиму реплики без подключения к мастеру
- `pgsql-data-status`
  - `LATEST` – соответствует состоянию мастера
  - `STREAMING|SYNC` – соответствует синхронной реплике
  - `STREAMING|ASYNC` – соответствует синхронной реплике
  - `DISCONNECT` – реплика не подключена к мастеру или остановлен postgres на узле
- `vip-master vip-pgbouncer pgbouncer`
  - `Started` – ресурс запущен
  - `Stoped` – ресурс остановлен

Вспомогательные команды для кластера

Некоторые вспомогательные команды для кластера, которые помогут при разборе инцидентов:

- Очистка счетчика сбоев для ресурса pgsql-clone на узле hostname1:

```
sudo pcs resource cleanup pgsql node=hostname1
```

- Деактивация ресурса pgsql:

```
sudo pcs resource disable pgsql
```

- Активация ресурса pgsql:

```
sudo pcs resource enable pgsql
```

- Остановка ресурса pgsql:

```
sudo pcs resource stop pgsql
```

- Запуск ресурса pgsql:

```
sudo pcs resource start pgsql
```

- Остановка кластера на текущем узле:

```
sudo pcs cluster stop
```

- Посмотреть конфигурацию кластера:

```
sudo pcs config show
```

- Посмотреть веса (очки score) ресурсов:

```
sudo pcs resource scores
```

- Посмотреть состояние кворума:

```
sudo corosync-quorumtool -s
```

2.8 Инструкция по настройке и работе с расширением pg_stat_kcache

Установка pg_stat_kcache

1. Установить `pg_stat_kcache`

```
sudo apt install postgresql-16-pg-stat-kcache postgresql-contrib
```

2. Зарегистрировать расширение в postgresql

```
su postgres
psql
```

Проверьте какие библиотеки у вас уже занесены в `shared_preload_libraries`

```
show shared_preload_libraries;
```

Пример вывода:

```
postgres=# show shared_preload_libraries;
    shared_preload_libraries
-------------------------------------
<ваши расширения>
(1 строка)
```

Дополнить `shared_preload_libraries` новыми расширениями, сохранив старые

```
alter system set shared_preload_libraries = '<ваши расширения>,pg_stat_statements,
↪pg_stat_kcache';
```

Что-бы применить новые расширения, необходимо перезапустить `postgresql`

```
sudo systemctl restart postgresql
```

3. Создать расширение в нужной БД

   Подключитесь к нужной вам БД

   ```
   \c <ваша бд>
   ```

   Создайте расширения

   ```
   CREATE EXTENSION pg_stat_statements;
   CREATE EXTENSION pg_stat_kcache;
   ```

   Примечание

   Чтобы распространялось на новые БД — выполнить команды в БД template1.

4. Проверка

   Чтоб проверить успешность, выполните команду

   ```
   \dx
   ```

   Пример вывода:

```
pg16=# \dx
                                    Список установленных расширений
        Имя          | Версия |   Схема    |                                Описание
---------------------+--------+------------+---------------------------------------
↪-------------------------------
...
pg_stat_kcache      | 2.3.0  | public     | Kernel statistics gathering
pg_stat_statements | 1.10   | public     | track planning and execution statistics␣
↪of all SQL statements executed
...
(N строк)
```

Пример запросов

Топ-20 по IO (exec_reads + exec_writes) с текстом запроса

```
WITH k AS (
  SELECT *
  FROM pg_stat_kcache_detail
  WHERE top IS TRUE
),
s AS (
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
  SELECT dbid, userid, toplevel, query,
         calls
  FROM pg_stat_statements
)
SELECT
  s.calls,
  (k.exec_reads + k.exec_writes)                  AS io_ops,
  k.exec_reads, k.exec_writes,
  round(k.exec_user_time::numeric, 3)   AS user_sec,
  round(k.exec_system_time::numeric, 3) AS sys_sec,
  left(k.query, 200)                    AS query_sample
FROM k
JOIN pg_database d ON d.datname = k.datname
JOIN pg_roles    u ON u.rolname = k.rolname
JOIN s ON s.dbid = d.oid
       AND s.userid = u.oid
       AND s.toplevel = k.top
       -- нормализуем пробелы в тексте запроса на обоих сторонах
       AND md5(regexp_replace(s.query, '\s+', ' ', 'g'))
           = md5(regexp_replace(k.query, '\s+', ' ', 'g'))
ORDER BY (k.exec_reads + k.exec_writes) DESC
LIMIT 20;
```

Топ-20 по CPU (exec_user_time + exec_system_time)

```
WITH k AS (
  SELECT *
  FROM pg_stat_kcache_detail
  WHERE top IS TRUE
),
s AS (
  SELECT dbid, userid, toplevel, query,
         calls
  FROM pg_stat_statements
)
SELECT
  s.calls,
  round((k.exec_user_time + k.exec_system_time)::numeric, 3) AS cpu_sec,
  round(k.exec_user_time::numeric, 3)   AS user_sec,
  round(k.exec_system_time::numeric, 3) AS sys_sec,
  left(k.query, 200)                    AS query_sample
FROM k
JOIN pg_database d ON d.datname = k.datname
JOIN pg_roles    u ON u.rolname = k.rolname
JOIN s ON s.dbid = d.oid
       AND s.userid = u.oid
       AND s.toplevel = k.top
       AND md5(regexp_replace(s.query, '\s+', ' ', 'g'))
           = md5(regexp_replace(k.query, '\s+', ' ', 'g'))
ORDER BY (k.exec_user_time + k.exec_system_time) DESC
LIMIT 20;
```

Быстрая сводка по БД (без джойнов)

```
SELECT
  datname,
  (exec_reads + exec_writes) AS io_ops,
  exec_reads, exec_writes,
  round(exec_user_time::numeric, 3)   AS user_sec,
  round(exec_system_time::numeric, 3) AS sys_sec,
  stats_since
FROM pg_stat_kcache
ORDER BY io_ops DESC;
```

3 GlobalServer Автономный режим

3.1 Установка сервера приложений

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

Необходимое программное обеспечение

ПО под ОС Astra Linux / Debian:

- postgresql-client (версия клиента должна совпадать с версией сервера)
- jre 21 или jdk21
  - Axiom JDK (Liberica) https://axiomjdk.ru/pages/downloads/#/java-21-lts
  - другой дистрибутив на основе OpenJDK https://jdk.java.net/java-se-ri/21
- expect
- htop
- iotop
- sysstat
- ssh (клиент и сервер SSH)
- mc (Midnight Commander)
- tar
- zip
- unzip
- cifs-utils
- sudo

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

Пример подключения через конфигурационный файл `/etc/fstab`

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

Установка

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

Подключитесь терминалом к серверу Global

Создайте временную директорию и загрузите файлы дистрибутива на сервер

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

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

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        <security>
                <!--В секции могут быть указаны значения парметров 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-->
                        <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"
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                                        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 -->
        <!--proxyaddress - Корнерой адрес куда клиентское приложение будет полысать␣
↪запрос на получение отчёта.-->
        <!--                Абсолютный, если необходимо направить запросы напрямую к␣
↪серверу отчётов или на внешний сервис.-->
        <!--                Относительный, если на текущем web-сервере настроен proxy к␣
↪серверу отчётов.-->
        <fastReportMonoServer host="http://global3:90" proxyaddress="fastreportmono"/>
        <!--host - Адрес web-сервера:порт, на котором запущен сервис построения отчётов␣
↪FastReport.VCL.
                                Этот же адрес должен быть указан на прокси-узле -->
        <!--proxyaddress - Корнерой адрес куда клинетское приложение будет посылать␣
↪запрос на получение отчёта.-->
        <!--                Абсолютный, если необходимо направить запросы напрямую к␣
↪серверу отчётов или на внешний сервис.-->
        <!--                Относительный, если на текущем web-сервере настроен proxy 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␣
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪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-->
        <!--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>
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                <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
sGlobalService="global3"

# имя сервиса менеджера заданий
sSchedulerService="globalscheduler"

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

Где:

sSshPort - порт внутреннего ssh сервера Global

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

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

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

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

sGlobalService - Имя сервиса сервера приложений

sSchedulerService - Имя сервиса менеджера заданий

Выдайте права на запуск

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

Шифрование паролей в конфигурационных файлах

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

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

Общие принципы шифрования

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

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

Где:

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

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

Использование шифрованных паролей

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

Пример:

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

Мастер-пароль может быть указан следующими способами:

- Параметром запуска сервера приложений `-masterPassword`

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

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

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

Шифрование паролей планировщика заданий

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

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

Ключ шифрования может быть указан следующими способами:

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

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

  Примечание

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

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

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

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

Где:

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

Шифрование паролей утилиты обновления

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

Ключи шифрования для токенов

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 установленны первоначальные данные в таблицах.

Для инициализации выполните следующие действия:

Запустите сервер приложений

```
sudo systemctl start global3
```

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 добавте 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";

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)) {
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
  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 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.

Логирование сервера приложений

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

Подробнее о конфигурации в разделе *Логирование сервера приложений*.

3.2 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 сервера через скрипт

Раздел описывает процедуру ручного запуска сервера Global 3 с использованием встроенного скрипта `start.sh`.

Скриптовый запуск предназначен для отладки и диагностики работы приложения Global 3. Он позволяет наблюдать вывод логов в реальном времени и быстро выявлять причины ошибок при старте сервера или во время его работы.

Такой способ рекомендуется использовать только в следующих случаях:

- Отладка приложения - проверка корректности конфигурации, подключения к СУБД, загрузки плагинов и т.д.
- Проверка после обновления - убедиться, что новая версия успешно запускается вручную перед активацией системного сервиса.
- Диагностика проблем с systemd - если служба global3 не стартует или завершается с ошибкой.

Важно

В рабочем (продуктивном) режиме сервер **должен запускаться через systemd** как сервис `global3`.
Ручной запуск через скрипт применяется только для временной диагностики и не используется в постоянной эксплуатации.

Перед запуском убедитесь, что служба global3 остановлена (если она существует):

```
systemctl stop global3
```

Перейдите в каталог приложения. По умолчанию сервер установлен в директорию:

```
cd /opt/global/globalserver
```

Если используется нестандартный путь, замените его на актуальный каталог расположения исполняемых файлов приложения.

Выполните запуск bash-скрипта с правами root:

```
sudo ./start.sh
```

После успешного запуска приложение станет доступно по сетевому порту, указанному в конфигурации. По умолчанию используется порт: `8080`

Работа с активным процессом:

- Не закрывайте окно терминала, в котором выполняется сервер — это приведёт к его остановке.
- Для выполнения дополнительных команд или взаимодействия с системой откройте второе SSH-подключение или новую вкладку терминала.
- Для корректного завершения работы сервера активируйте окно с запущенным процессом и нажмите комбинацию клавиш: `Ctrl + C`

Запуск и остановка планировщика заданий

Команда запуска

```
systemctl start globalscheduler
```

Перезапуск

```
systemctl restart globalscheduler
```

Остановка

```
systemctl stop globalscheduler
```

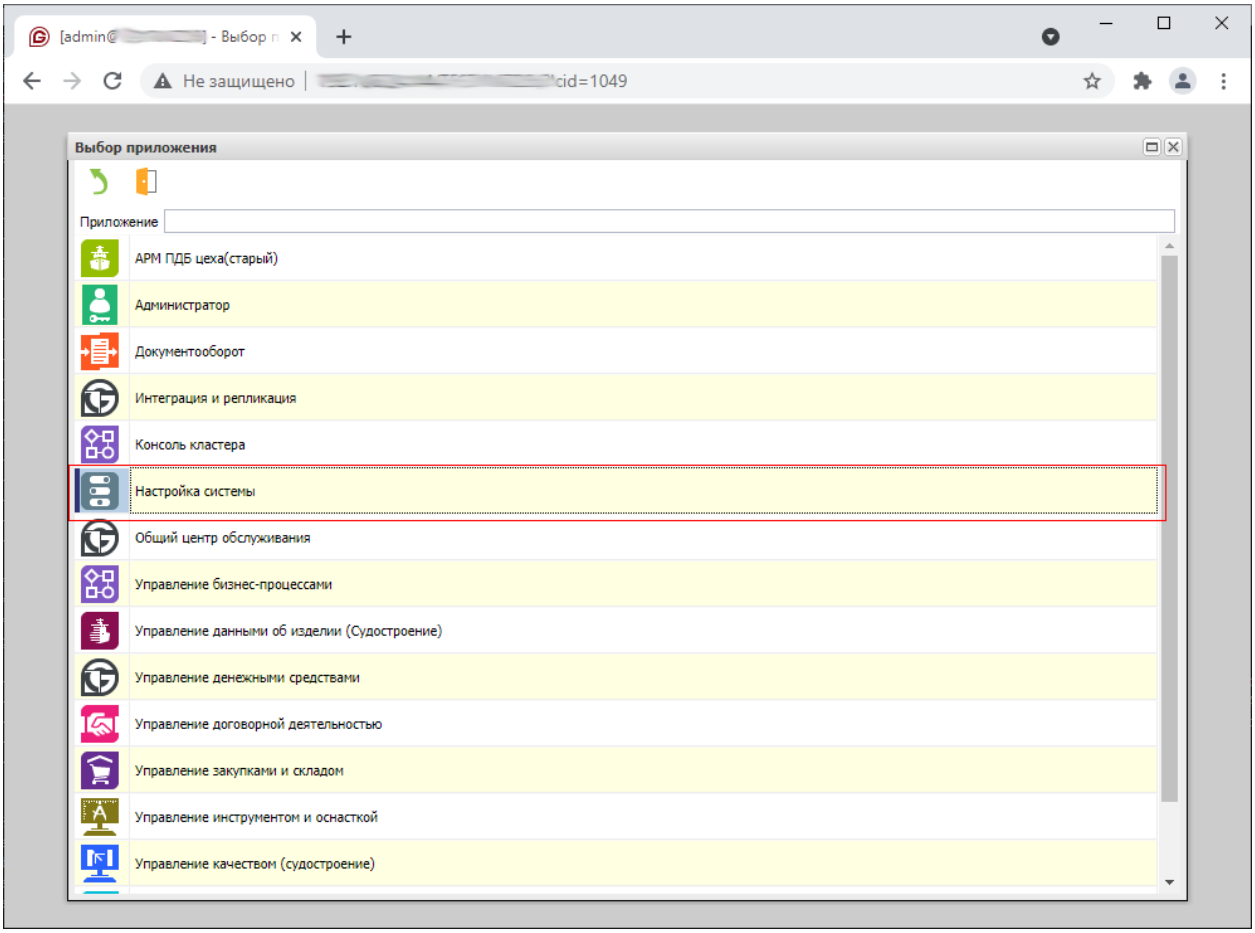
Настройка файлового хранилища

Система версионного хранения файлов может работать в двух режимах:

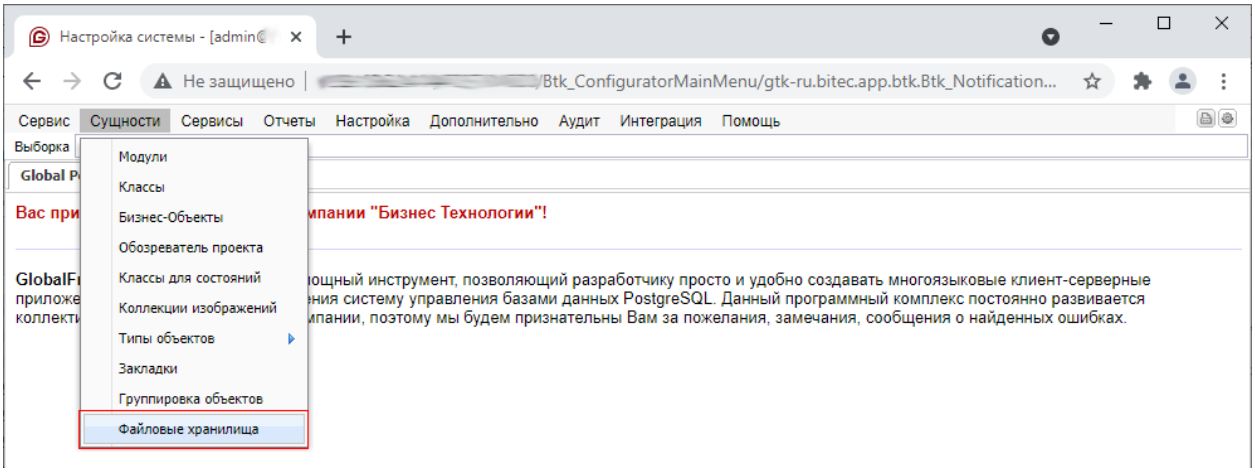
- SMB/ CIFS - медленный режим. Подключается к сетевому ресурсу каждый раз, когда требуется доступ файлу.
- Локальное хранилище – быстрый режим. Работает с локальной директорией на сервере.

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.)

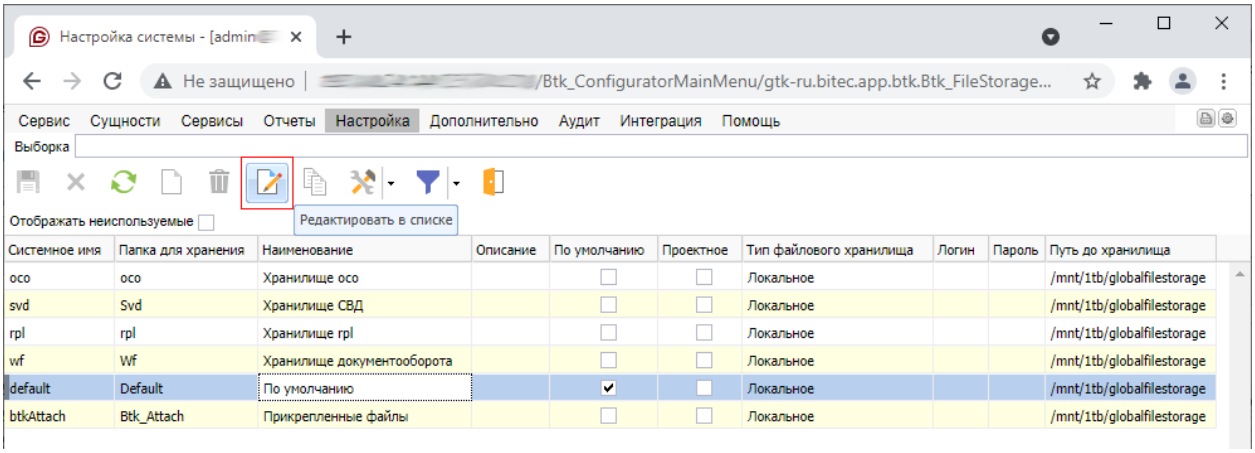
Для настройки файловых хранилищ требуется открыть приложение «Настройка системы»



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

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.

3. Выберите хранилище, подлежащее переносу, и добавьте к его названию суффикс **_old** (например, `storage` → `storage_old`).

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.

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 `/opt/global/globalserver/update`

Перед использованием следует сформировать конфигурационный файл `update/config.sh` (`config.ps1`) на основе `config.sh.origin`. После формирования файла загрузите полученные дистрибутивы в каталог, который вы указали в настройках файла в переменной `sTmp=`.

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:

- upgrade - обновляет jar-файлы прикладных модулей. Устанавливает релиз (без рестарта).
- 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`

```
/opt/global/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.

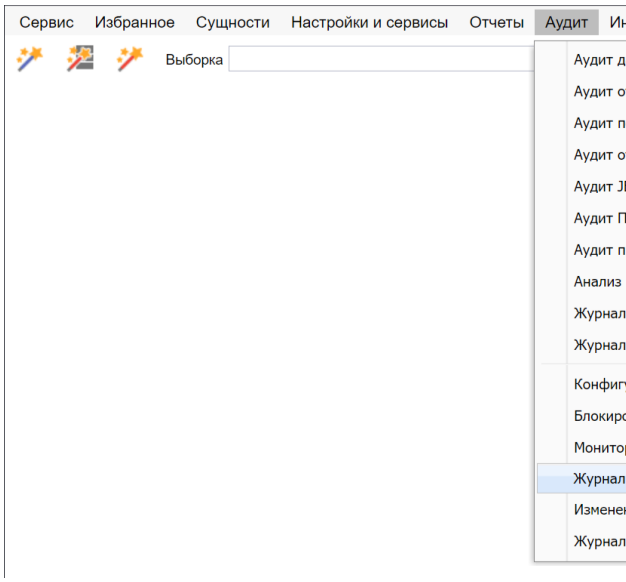
```
./update.sh -r -s skip
# выводит диалоговое окно выбора резервной копии прикладного решения и восстанавливает ее
./update.sh -r -s /tmp/globalupdate/update_yesterday/backup_server.tgz -a /tmp/
```

(продолжается на следующей странице)

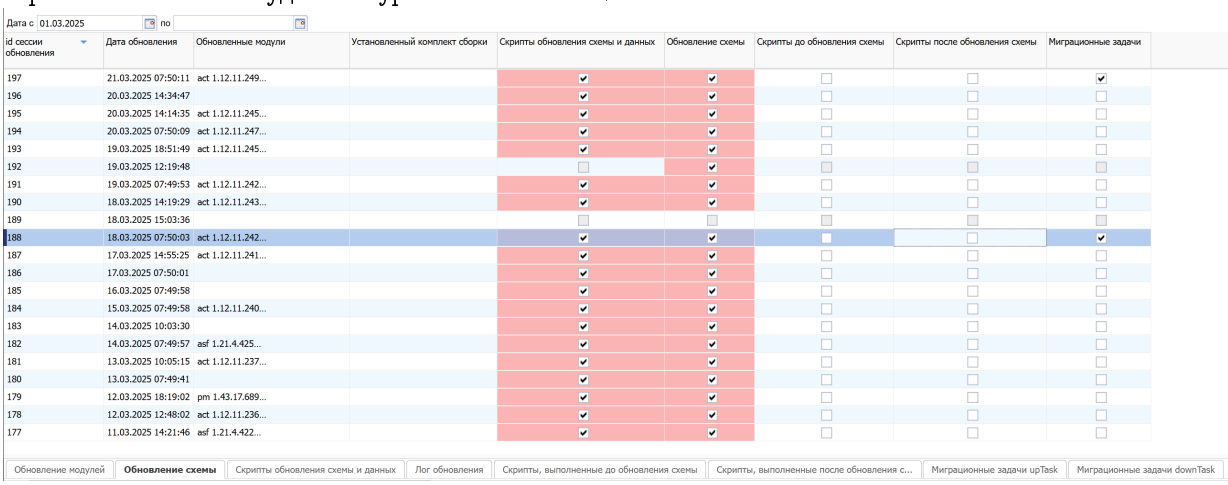
(продолжение с предыдущей страницы)

```
↪globalupdate/update_yesterday/backup_applib.tgz
# восстанавливает сервер приложений и прикладной решение из указанных резервных копий
```

Отображение «Журнал обновления модулей»



Расположение: `Настройки системы > Аудит > Журнал обновлений`



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)

3.3 Управление схемой базы данных

Корзина удаленных объектов схемы

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

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

Использование

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

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

3.4 Настройка HAProxy (standalone)

Общая информация HAProxy

HAProxy — это высокопроизводительный прокси-сервер и балансировщик нагрузки для протоколов TCP (L4) и HTTP/HTTPS (L7). Он используется для:

- распределения трафика между несколькими приложениями (балансировка нагрузки);
- проверки работоспособности целевых серверов (health checks);
- добавления служебных заголовков (например, X-Forwarded-For, X-Forwarded-Proto);
- ограничения скорости, защиты от атак и пр.

В режиме standalone HAProxy работает на одном сервере и проксирует трафик к локальным или удалённым backend-службам, повышая отказоустойчивость и безопасность.

Перед началом работ

- Получите права администратора (root) или возможность использовать sudo.
- Запустите приложение на порту 8080.

Подготовка системы и брандмауэра (UFW)

Обновление индексов пакетов и базовой системы

```
sudo apt update && sudo apt upgrade
```

Установка и включение Firewall

```
sudo apt install ufw
sudo systemctl enable ufw --now
```

Базовая политика firewall

```
sudo ufw default deny incoming
sudo ufw allow ssh
sudo ufw enable
```

- `default deny incoming` - по умолчанию запрещаем все входящие подключения (подход «deny by default»).
- `allow ssh` - разрешаем доступ по SSH (обычно порт 22), чтобы не потерять удалённый доступ.
- `enable` - активируем правила файрвола.

Разрешение нужных портов

```
sudo ufw allow 443/tcp
sudo ufw allow from 127.0.0.1 to any port 8080
sudo ufw allow 5000/tcp
sudo ufw allow 8405/tcp
```

- 443/tcp - входящий HTTPS-трафик к HAProxy.
- Разрешение доступа к локальному backend на 8080 (только с 127.0.0.1).
- 5000/tcp - входящий трафик для статистики HAProxy (включить при необходимости).
- 8405/tcp - входящий трафик для статистики Prometheus (включить при необходимости).

Важно

Показан пример для тестового стенда. В продакшене рекомендуется не открывать порты мониторинга для любых подключений.

Проверка правил

```
sudo ufw status verbose
```

Установка HAProxy

```
sudo apt install haproxy -y
sudo systemctl enable haproxy --now
```

Инструкции для других версий/ОС размещены на официальном сайте HAProxy для Debian: https://haproxy.debian.net/

TLS/SSL: самоподписанный сертификат для теста

Важно

В продакшене используйте сертификаты от доверенного CA. Самоподписанный сертификат подходит для тестов/внутренних стендов.

```
openssl genrsa -out cert.key 2048
openssl req -new -key cert.key -out cert.csr
openssl x509 -req -days 365 -in cert.csr -signkey cert.key -out cert.crt
mkdir /etc/ssl/globalerp/
cat cert.key cert.crt > /etc/ssl/globalerp/haproxy_ssl.pem
```

- Генерируем приватный ключ RSA 2048 бит.
- Формируем CSR.

- Выпускаем самоподписанный сертификат на 365 дней.
- Сливаем ключ и сертификат в единый PEM-файл haproxy_ssl.pem, который удобно указывать в haproxy.cfg.

Рекомендуемое расположение PEM: `/etc/ssl/globalerp/haproxy_ssl.pem`
Права: `chmod 640`, владелец/группа: `haproxy:haproxy`, чтобы процесс `haproxy` мог читать файл.

Базовая конфигурация HAProxy

Откройте конфигурацию:

```
sudo nano /etc/haproxy/haproxy.cfg
```

```
global
        # Логи отправляются в системный syslog (rsyslog/journald)
        log /dev/log    local0 info
        log /dev/log    local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin
        stats timeout 300s
        user haproxy
        group haproxy
        daemon

        # Default SSL material locations
        ca-base /etc/ssl/certs
        crt-base /etc/ssl/private

        # See: https://ssl-config.mozilla.org/#server=haproxy&server-version=2.0.3&
↪config=intermediate
        ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-
↪SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
↪POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-
↪SHA384
        ssl-default-bind-ciphersuites TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_
↪CHACHA20_POLY1305_SHA256
        ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

frontend stats
        mode http
        bind :5000
        stats enable
        stats refresh 10s
        stats uri /
        stats show-modules
        stats hide-version
        stats admin if TRUE
        stats auth <Login>:<Password>
        timeout client 10s

defaults
        log     global
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        mode    http
        option  httplog
        option  dontlognull
        option  log-health-checks
        option  abortonclose
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        # Кастомные страницы ошибок
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http

frontend haproxynode
        bind *:80
        redirect scheme https code 301 if !{ ssl_fc }
        bind *:443 ssl crt /etc/ssl/release-info.dev.local/release-info.dev.local.pem
        mode http
        default_backend backendnodes

backend backendnodes
        balance leastconn #roundrobin
        option forwardfor # Включает X-Forwarded-For для передачи IP-адреса клиента␣
↪обслуживающему серверу.
        cookie GS_BALANCER_SERVER_NAME insert #  добавлять куки. Для режима липких␣
↪сессий, когда один ip подключается к одному backend-ду

        acl has_x_forwarded_port req.hdr(X-Forwarded-Port) -m found
        http-request set-header X-Forwarded-Port %[req.hdr(X-Forwarded-Port)] if has_x_
↪forwarded_port
        http-request set-header X-Forwarded-Port %[dst_port] unless has_x_forwarded_port
        http-request add-header X-Forwarded-Proto https if { ssl_fc }
        option httpchk HEAD / HTTP/1.1\r\nHost:localhost

        server global-server-01 localhost:8080 check cookie global-server-01
```

Разбор конфигурации

Уровни логирования

Строки `log /dev/log local0 info` и `log /dev/log local1 notice` задают отправку логов в системный syslog с различным уровнем. Это позволит разнести рабочие логи и, например, административные сообщения.

```
global
        log /dev/log    local0 info
        log /dev/log    local1 notice
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin
        stats timeout 300s
        user haproxy
        group haproxy
        daemon
```

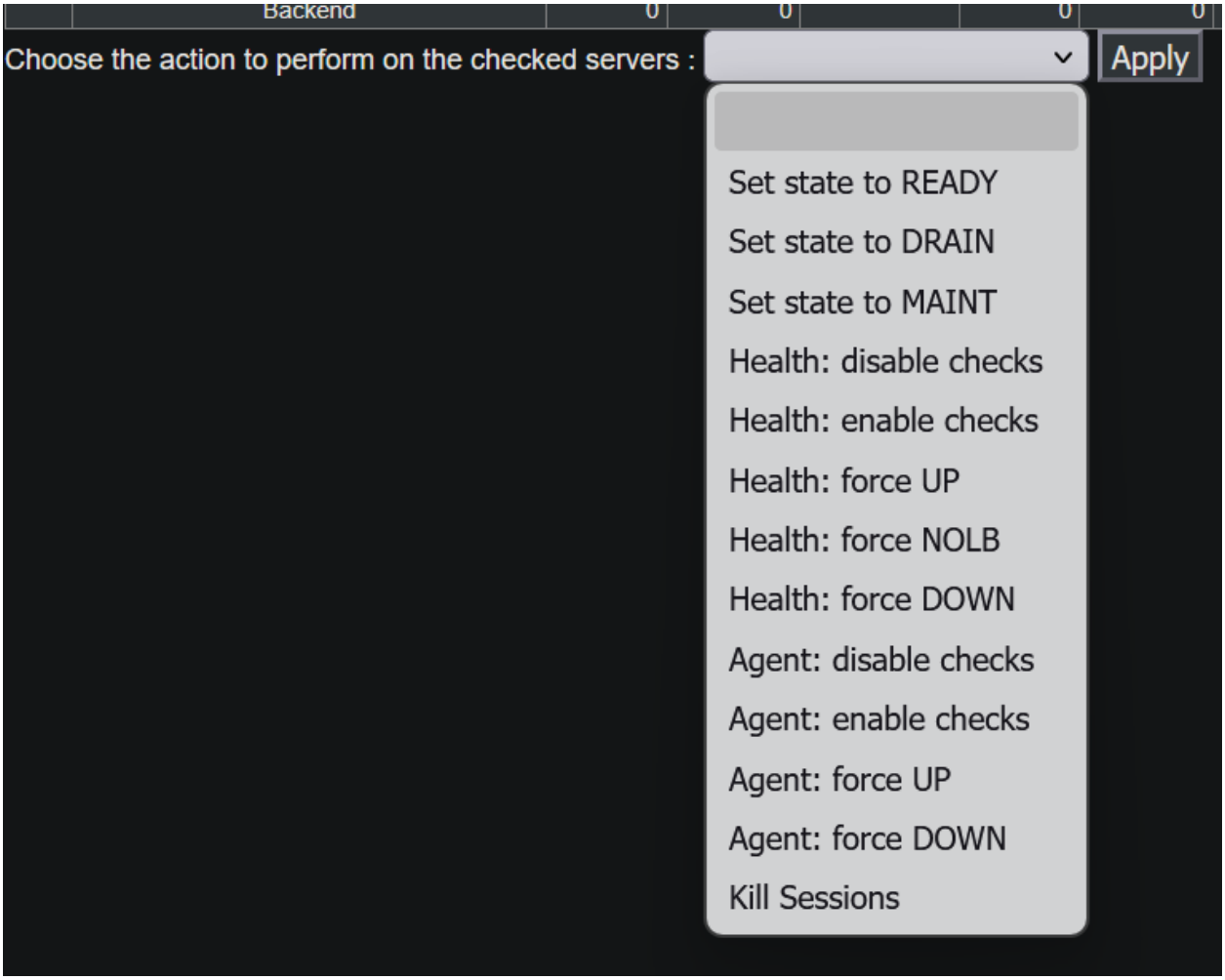
Уровень лога	Описание
emerg	Ошибки, такие как исчерпание файловых дескрипторов операционной системы.
alert	Некоторые редкие случаи непредвиденных событий, например, невозможность кэширования ответа.
err	Ошибки, такие как невозможность анализа файла карты, невозможность анализа файла конфигурации HAProxy и сбой операции с таблицей Stick.
warni	Некоторые важные, но некритические ошибки, такие как невозможность установить заголовок запроса или невозможность подключения к DNS-серверу имён.
notific	Изменения состояния сервера, например, «UP» (включен) или «DOWN» (отключён). Также учитываются другие события при запуске, такие как запуск прокси-серверов и загрузка модулей. Журнал проверки работоспособности, если он включен, также использует этот уровень.
info	Подробности и ошибки TCP-подключения и HTTP-запросов.
debug	Вы можете написать собственный код Lua, который будет регистрировать отладочные сообщения.

Статистика

Доступ к странице мониторинга: http://IP_ADDRESS:5000/stats (логин/пароль из stats auth).

IP_ADDRESS - ip адрес вашего сервера с HAProxy

После авторизации на странице мониторинга, открывается доступ к различным действиям. Подробнее о каждом, в таблице ниже.


Backend
0
0
0
0
Choose the action to perform on the checked servers :
Apply
Set state to READY
Set state to DRAIN
Set state to MAINT
Health: disable checks
Health: enable checks
Health: force UP
Health: force NOLB
Health: force DOWN
Agent: disable checks
Agent: enable checks
Agent: force UP
Agent: force DOWN
Kill Sessions

Действие	Описание
Set state to READY	Выбранный сервер будет получать трафик. Окончательное состояние сервера будет определено на основе настроенной проверки работоспособности.
Set state to DRAIN	Выбранный сервер не будет принимать никаких новых подключений, кроме тех, которые приняты через сохранение сеанса.
Set state to MAINT	Выбранный сервер не будет принимать никаких новых подключений, а проверки работоспособности будут остановлены.
Health: disable checks	Отключить проверки работоспособности сервера.
Health: enable checks	Включить проверки работоспособности сервера.
Health: force UP	Немедленно перевести проверку работоспособности сервера в состояние UP.
Health: force NOLB	Принудительно перевести проверку работоспособности сервера в состояние NOLB. Это останавливает сервер от приема новых непостоянных подключений.
Health: force DOWN	Немедленно перевести проверку работоспособности сервера в состояние DOWN.
Agent: disable checks	Отключить проверки агента сервера.
Agent: enable checks	Включить проверки агента сервера.

Frontend часть

Принимает HTTP на `:80` и HTTPS на `:443`. HTTP трафик перенаправляет на HTTPS.

```
frontend haproxynode
        bind *:80
        redirect scheme https code 301 if !{ ssl_fc }
    bind *:443 ssl crt /etc/ssl/release-info.dev.local/release-info.dev.local.pem
    mode http
    default_backend backendnodes
```

- `bind :80` - приём HTTP (для редиректа).
- `bind :443 ssl crt ...` - приём HTTPS с указанным PEM-сертификатом.
- `http-request redirect ... unless { ssl_fc }` - перевод всего трафика на HTTPS.
- `default_backend backendnodes` - выбор backend по умолчанию.

Backend часть

Перенаправляет запросы от frontend части на сервера приложений.

```
backend backendnodes
    balance leastconn #roundrobin
    option forwardfor # Включает X-Forwarded-For для передачи IP-адреса клиента␣
↪обслуживающему серверу.
        cookie GS_BALANCER_SERVER_NAME insert
    acl has_x_forwarded_port req.hdr(X-Forwarded-Port) -m found
    http-request set-header X-Forwarded-Port %[req.hdr(X-Forwarded-Port)] if has_x_
↪forwarded_port
    http-request set-header X-Forwarded-Port %[dst_port] unless has_x_forwarded_port
    http-request add-header X-Forwarded-Proto https if { ssl_fc }
    option httpchk HEAD / HTTP/1.1\r\nHost:localhost

    server global-server-01 localhost:8080 check cookie global-server-01
```

- `balance roundrobin` — равномерная балансировка по списку `server`.
- `option forwardfor` — добавляет `X-Forwarded-For` с IP клиента.
- `cookie ... insert` + `server ... cookie ...` — липкие сессии (sticky) для привязки клиента к одному серверу.
- `server <name> <ip:port> check` — сервер backend; `check` включает health-check. Добавляйте столько строк `server`, сколько у вас целевых инстансов.

Проверка конфигурации и перезапуск

```
sudo haproxy -f /etc/haproxy/haproxy.cfg -c
sudo systemctl restart haproxy
```

- `-c` - проверяет синтаксис и базовую корректность конфигурации без запуска.
- `restart` - перезапускает службу с разрывом соединений клиентов.

Проверка статуса работы:

```
sudo systemctl status haproxy
journalctl -u haproxy -e
```

Включение логирования HAProxy

По умолчанию HAProxy пишет лог в `/dev/log`. Настроим rsyslog для работы с логами HAProxy.

Rsyslog — это быстрый, расширяемый сервис для управления логами.

Установка Syslog server:

```
sudo apt install -y rsyslog
```

Конфигурация rsyslog для HAProxy:

```
sudo nano /etc/logrotate.d/haproxy
```

Пример конфигурации файла

```
/var/log/haproxy/haproxy.log {
    daily
    rotate 7
    compress
    delaycompress
    missingok
    notifempty
    create 0640 haproxy adm
    postrotate
        invoke-rc.d rsyslog reload > /dev/null
    endscript
}
```

Безопасная аутентификация к странице статистики

В избежании хранения пароля для входа на страницу в открытом виде, используйте `userlist` с хэшированным паролем.

Подробнее в документации

Для шифрования паролей в HAProxy используется функция **crypt(3)**. Это библиотечная функция, которая используется для создания и проверки хешей паролей. Она генерирует специальную строку, содержащую хеш, «соль» (случайная строка, добавленная для усиления безопасности) и информацию об алгоритме, которая хранится в файлах пользователей, например, /etc/shadow. Эта строка затем сравнивается с введённым паролем при входе пользователя для аутентификации

Наиболее простой способ использовать данную функцию, воспользоваться утилитой mkpasswd, входящей в состав whois.

Установите утилиту и сгенерируйте хеш:

```
sudo apt install -y whois
mkpasswd -m sha-256
# Введите пароль → получите строку $...
```

Замените секцию `frontend stats` и добавьте `userlist`:

```
userlist stats_users
    user admin password <hash> # Вставьте сгенерируемый ранее хэш

frontend stats
    mode http
    bind :5000
    stats enable
    stats refresh 10s
    stats uri /stats
    stats show-modules
    stats admin if TRUE
    http-request auth unless { http_auth(stats_users) }
```

11. Подключение HAProxy к Prometheus

Предполагается что у вас развернута ВМ с Prometheus и Grafana. Подробнее об установке в *руководстве*.

Включим встроенный Prometheus-экспортёр HAProxy и настроим сбор метрик Prometheus. Подробнее о конфигурировании в документации

Включить экспорт метрик в HAProxy

Добавьте во frontend отдельный блок для выдачи метрик:

```
frontend prometheus
    bind :8405
    mode http
    http-request use-service prometheus-exporter
```

Конфигурирование Prometheus

Отредактируйте /opt/prometheus/prometheus.yml и добавьте job с HAProxy:

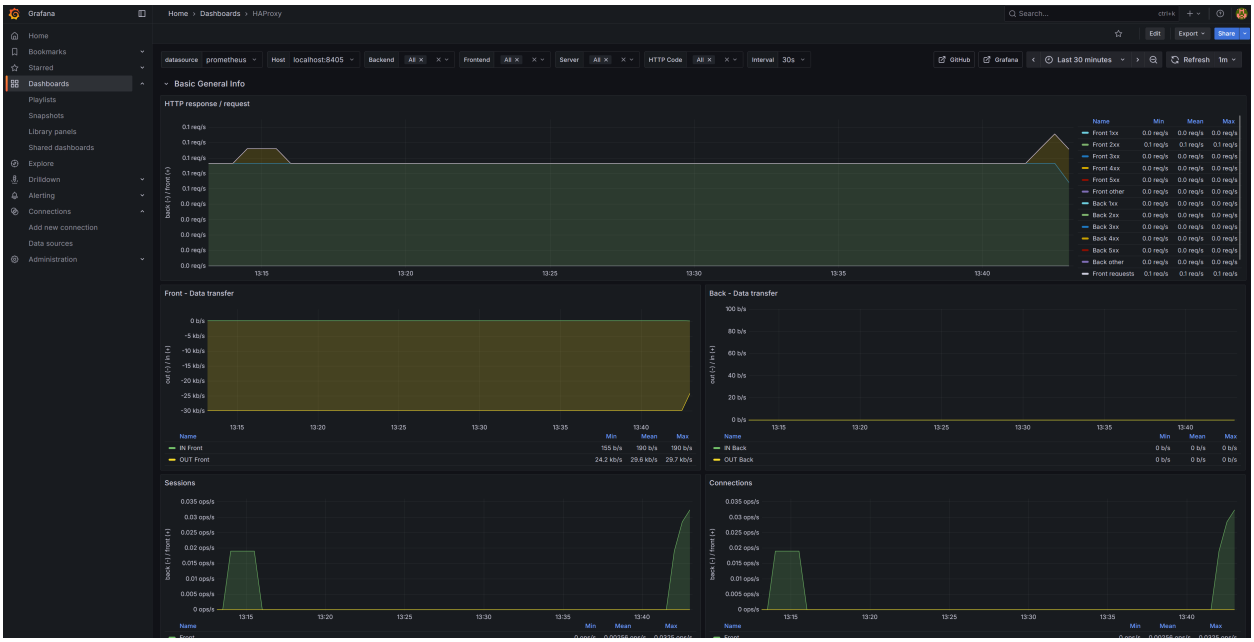
```
scrape_configs:
- job_name: "haproxy"
  static_configs:
    - targets: ["<IP_ADDRESS HAPROXY>:8405"]
```

Импорт готового дашборда HAProxy в Grafana

Существуют готовые дашборды. В этом примере используем JSON из репозитория, рекомендованного Grafana: https://github.com/rfmoz/grafana-dashboards/blob/master/prometheus/haproxy.json

Шаги импорта:

1. В Grafana: Dashboards → New → Import.
2. Загрузите файл haproxy.json (предварительно скачайте его с GitHub).
3. В поле Select a data source выберите ранее добавленный источник Prometheus.
4. Нажмите Import — дашборд появится в списке и станет доступен к просмотру.



3.5 Руководство по установке сервера мониторинга

1. Общие сведения

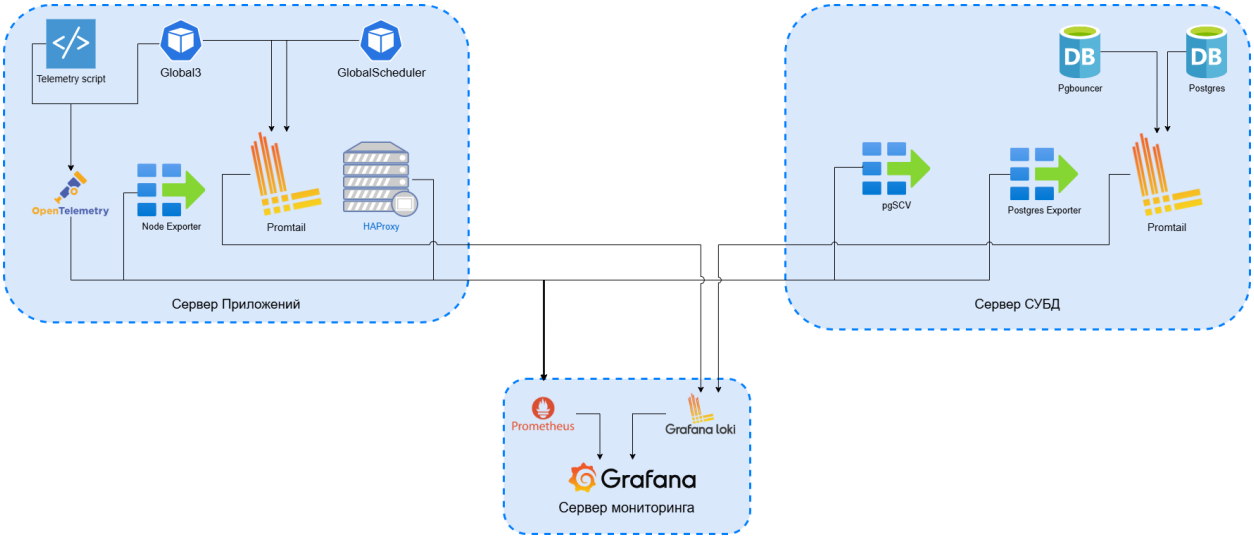
Для предотвращения перегрузки основного сервера приложений все инструменты телеметрии рекомендуется размещать на отдельной виртуальной машине.
Руководство описывает процесс установки и базовой настройки следующих компонентов:

1. **Grafana** - платформа визуализации данных;
2. **Prometheus** - система сбора и хранения метрик;
3. **Loki** - система централизованного хранения логов;
4. **Promtail** - агент для отправки логов в Loki;
5. **OpenTelemetry Collector** - компонент для сбора и экспорта телеметрических данных.
6. **Настройки сервисов:**
   - HAProxy
   - NFS
   - Global3
   - GlobalScheduler
7. **Настройки экспортеров:**
   - Node Exporter
   - pgSCV
   - Postgres exporter
   - Скрипт снятия дополнительных метрик

Пример архитектуры серверов

1. Сервер СУБД (PostgreSQL)
2. Сервер приложений (Global ERP) + HAProxy (Load Balancer)
3. Сервер мониторинга (Prometheus, Grafana, Loki)

Принципиальная схема работы системы мониторинга:



---

2. Установка Prometheus

2.1. Общие сведения

Внимание

Сервис устанавливается на сервер мониторинга.

Подробная документация: Prometheus - First Steps
Загрузить релиз можно с официального сайта: prometheus.io/download

Важно

Для систем на базе Debian рекомендуется использовать установку через apt, а не скачивать последнюю версию вручную - это повышает стабильность и предсказуемость обновлений.

2.2. Установка

```
sudo apt update
sudo apt install prometheus
```

В сервисе prometheus.service приведите стартовый параметр к следующему виду:

```
ExecStart=/usr/bin/prometheus $ARGS --web.enable-remote-write-receiver
```

2.3. Конфигурация

Отредактируйте основной файл конфигурации `/etc/prometheus/prometheus.yml`

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s
  external_labels:
    monitor: 'example'

alerting:
  alertmanagers:
    - static_configs:
        - targets: [ 'localhost:9093' ]

rule_files:

scrape_configs:
  - job_name: 'prometheus'
    scrape_interval: 5s
    scrape_timeout: 5s
    static_configs:
      - targets: [ 'localhost:9090' ]
    relabel_configs:
      - replacement: '{{project_name}}'
        target_label: project_name

  - job_name: 'haproxy'
    static_configs:
      - targets: [ '<IP адрес сервера haproxy>:8405' ]
        labels: { host: '<IP адрес сервера haproxy>' }
    params:
      extra-counters: [ "on" ]

  - job_name: 'otelcol'
    static_configs:
      - targets: [ '<IP адрес сервера приложений>:8888' ]
        labels: { host: '<IP адрес сервера приложений>' }
    relabel_configs:
      - replacement: '{{project_name}}'
        target_label: project_name
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
  - job_name: 'node-exporter-global'
    static_configs:
      - targets: [ '<IP адрес сервера приложений>:9100' ]
        labels: { host: '<IP адрес сервера приложений>' }

  - job_name: 'node-exporter-postgres'
    static_configs:
      - targets: [ '<IP адрес сервера СУБД>:9100' ]
        labels: { host: '<IP адрес сервера СУБД>' }

  - job_name: 'postgres'
    static_configs:
      - targets: [ "<IP адрес сервера СУБД>:9187" ]
        labels: { host: '<IP адрес сервера СУБД>' }

  - job_name: 'pgbouncer'
    static_configs:
      - targets: [ "<IP адрес сервера СУБД>:9890" ]
        labels: { host: '<IP адрес сервера СУБД>' }
```

2.4. Веб-интерфейс

Prometheus доступен по адресу:

```
http://<IP_ADDRESS>:9090
```

где <IP_ADDRESS> - IP-адрес сервера мониторинга.

---

3. Установка Grafana

3.1. Установка

Внимание

Сервис устанавливается на сервер мониторинга.

Проверить последнюю версию Grafana можно в документации

```
sudo apt-get install -y adduser libfontconfig1 musl
wget https://dl.grafana.com/grafana-enterprise/release/<VERSION>/grafana-enterprise_
↪<VERSION>_<OS>_<ARCH>.deb
sudo dpkg -i grafana-enterprise_<VERSION>_<OS>_<ARCH>.deb
sudo systemctl enable --now grafana-server
```

3.2. Доступ

Интерфейс доступен по адресу:

```
http://<IP_ADDRESS>:3000
```

Стандартные учетные данные:
`admin / admin`
После первого входа система предложит изменить пароль - задайте надёжный.

---

4. Установка Loki

> **Внимание**
>
> Сервис устанавливается на сервер мониторинга.

4.1. Загрузка и установка

Проверить последний релиз можно в репозитории

```
wget https://github.com/grafana/loki/releases/download/v<VERSION>/loki_<VERSION>_<ARCH>.
↪deb
sudo apt install loki_<VERSION>_<ARCH>.deb
```

В сервисе loki.service приведите стартовый параметр к следующему виду:

```
ExecStart=/usr/bin/loki -config.file /etc/loki/config.yml -config.expand-env=true
```

4.2. Конфигурация

Создайте файл конфигурации `/etc/loki/config.yml`

```
auth_enabled: false

server:
  http_listen_port: 3100
  grpc_listen_port: 9096
  log_level: debug
  grpc_server_max_concurrent_streams: 1000
  grpc_server_max_recv_msg_size: 209715200
  grpc_server_max_send_msg_size: 209715200

common:
  instance_addr: 127.0.0.1
  path_prefix: /tmp/loki
  storage:
    filesystem:
      chunks_directory: /tmp/loki/chunks
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
      rules_directory: /tmp/loki/rules
  replication_factor: 1
  ring:
    kvstore:
      store: inmemory

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

limits_config:
  metric_aggregation_enabled: true
  max_entries_limit_per_query: 1000000
  query_timeout: 10m
  max_global_streams_per_user: 10000

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

pattern_ingester:
  enabled: true
  metric_aggregation:
    loki_address: localhost:3100

ruler:
  alertmanager_url: http://localhost:9093

frontend:
  encoding: protobuf


analytics:
  reporting_enabled: false
```

4.3. Запуск и автозагрузка

```
sudo systemctl daemon-reload
sudo systemctl start loki
sudo systemctl enable loki.service
sudo systemctl status loki
```

---

5. Установка Promtail

Внимание

Сервис устанавливается как на сервер приложений, так и на сервер СУБД

5.1. Загрузка и установка

Проверить последний релиз можно в репозитории

```
https://github.com/grafana/loki/releases/download/v3.5.7/promtail_3.5.7_amd64.deb
wget "https://github.com/grafana/loki/releases/download/v<VERSION>/promtail_<VERSION>_
↪<ARCH>.zip"
sudo apt install promtail_<VERSION>_<ARCH>.deb
```

5.2. Запуск и проверка

```
sudo systemctl start promtail
sudo systemctl status promtail
```

5.3. Конфигурация Promtail

Отредактируйте основной файл конфигурации `/etc/promtail/config.yml`:

```
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://<IP Адрес сервера метрик>:3100/loki/api/v1/push

scrape_configs:
  - job_name: haproxy
    static_configs:
      - targets:
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        - localhost
      labels:
        job: "haproxy"
        host: "<IP Адрес сервера приложений>"
        service: "HAProxy"
        __path__: /var/log/haproxy.*
  pipeline_stages:
    - regex:
        expression: '^(?P<timestamp>\S+\s+\d+\s+\S+)\s+(?P<hostname>\S+)\s+haproxy\[(?P
↪<pid>\d+)\]:\s+(?P<client_ip>\S+):(?P<client_port>\d+)\s+\[(?P<accept_time>[^\]]+)\]\
↪s+(?P<frontend>\S+)\s+(?P<backend>\S+)\/(?P<server>\S+)\s+(?P<tq>\d+)\/(?P<tw>\d+)\/(?P
↪<tc>\d+)\/(?P<tr>\d+)\/(?P<tt>\d+)\s+(?P<status_code>\d+)\s+(?P<bytes_read>\d+)\s+(?P
↪<captured_request_headers>\S+)\s+(?P<captured_response_headers>\S+)\s+(?P<termination_
↪state>\S+)\s+(?P<actconn>\d+)\/(?P<feconn>\d+)\/(?P<beconn>\d+)\/(?P<srvconn>\d+)\/(?P
↪<retries>\d+)\s+(?P<srv_queue>\d+)\/(?P<backend_queue>\d+)\s+\{(?P<request_headers>[^}
↪]+)\}\s+\{(?P<response_headers>[^}]+)\}\s+"(?P<http_request>[^"]+)"'
    - labels:
        client_ip:
        frontend:
        backend:
        server:
        status_code:
        termination_state:
- job_name: global3
  static_configs:
    - targets:
        - localhost
      labels:
        job: "Global"
        host: "<IP Адрес сервера приложений>"
        service: "Global_3"
        __path__: /opt/global/globalserver/logs/*
  pipeline_stages:
    - multiline:
        firstline: '\['
    - regex:
        expression:
          '^(?s)\[(?P<time>\d{2}-\d{2}-\d{4} \d{2}:\d{2}:\d{2}\.\d{3})\] \[(?P<level>
↪[A-Z ]*)\] \[(?P<thread>.*)\] \[(?P<logger>[^ ]*)\] \[(?P<user>[^ ]*)\] - (?P<msg>.*)$'
    - labels:
        level:
        thread:
        logger:
        user:
    - timestamp:
        source: time
        format: 01-01-2025 13:30:30.000
        location: Europe/Moscow
    - labeldrop:
        - filename
        - time
    - output:
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        source: msg
  - job_name: globalscheduler
    static_configs:
      - targets:
          - localhost
        labels:
          job: "globalscheduler"
          host: "<IP Адрес сервера приложений>"
          service: "Global_Scheduler"
          __path__: /opt/global/globalserver/logs/scheduler/jobscheduler.*
    pipeline_stages:
      - multiline:
          firstline: '\['
      - regex:
          expression:
            '^(?s)\[(?P<time>\d{2}-\d{2}-\d{4} \d{2}:\d{2}:\d{2}\.\d{3})\] \[(?P<level>
↪[A-Z ]*)\] \[(?P<thread>.*)\] \[(?P<logger>[^ ]*)\] - (?P<msg>.*)$'
      - labels:
          level:
          thread:
          logger:
      - timestamp:
          source: time
          format: 01-01-2025 13:30:30.000
          location: Europe/Moscow
      - labeldrop:
          - filename
          - time
      - output:
          source: msg
```

Выдайте права на чтение логов:

```
sudo setfacl -R -m u:promtail:rX /var/log/
sudo chown promtail /var/log/haproxy.log
```

```
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://<IP Адрес сервера метрик>:3100/loki/api/v1/push

scrape_configs:
  - job_name: postgresql
    static_configs:
      - targets:
          - localhost
        labels:
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
          job: postgresql
          service: "PostgreSQL"
          __path__: /var/log/postgresql/postgresql-17-main.*
    pipeline_stages:
      - regex:
          expression: '^(?P<timestamp>\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}\.\d{3}) \[\d+\
↪] (?P<user_database>\w+@\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}|\w+@local)?:\w+ \[\d+\] (?P
↪<level>\w+):  (?P<message>.*)$'
      - labels:
          level:
          user_database:
          timestamp:

  - job_name: pgbouncer
    static_configs:
      - targets:
          - localhost
        labels:
          job: pgbouncer
          service: "PgBouncer"
          __path__: /var/log/postgresql/pgbouncer.*
    pipeline_stages:
      - regex:
          expression: '^(?P<timestamp>\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}\.\d{3}) \[\d+\
↪] (?P<user_database>\w+@\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}|\w+@local)?:\w+ \[\d+\] (?P
↪<level>\w+):  (?P<message>.*)$'
      - labels:
          level:
          user_database:
          timestamp:
```

Выдайте права на чтение логов:

```
sudo setfacl -R -m u:promtail:rX /var/log/
sudo usermod -a -G systemd-journal promtail
```

Раскомментируйте и настройте следующие строки в `/etc/postgresql/*/main/postgresql.conf`:

```
log_destination = 'stderr'
log_rotation_age = 3d
log_min_messages = warning
log_min_error_statement = error
```

---

6. Установка OpenTelemetry Collector

Подробности: opentelemetry.io/docs/collector

6.1 Настройка Otelcol сервиса

Внимание

Сервис устанавливается на сервер приложений

Проверить последний релиз можно в репозитории

```
sudo apt-get update
sudo apt-get install -y wget
wget https://github.com/open-telemetry/opentelemetry-collector-releases/releases/
↪download/v<VERSION>/otelcol_<VERSION>_<OS>_<ARCH>.deb
sudo apt install otelcol_<VERSION>_<OS>_<ARCH>.deb
```

Отредактируйте конфигурационный файл `/etc/otelcol-contrib/config.yaml`

```
extensions:
  health_check:
  pprof:
    endpoint: 0.0.0.0:1777
  zpages:
    endpoint: 0.0.0.0:55679

receivers:
  otlp:
    protocols:
      grpc:
        endpoint: 0.0.0.0:4317
      http:
        endpoint: 0.0.0.0:4318

  # Collect own metrics
  prometheus:
    config:
      scrape_configs:
        - job_name: 'otel-collector'
          scrape_interval: 10s
          static_configs:
            - targets: [ '0.0.0.0:8888' ]

  jaeger:
    protocols:
      grpc:
        endpoint: 0.0.0.0:14250
      thrift_binary:
        endpoint: 0.0.0.0:6832
      thrift_compact:
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        endpoint: 0.0.0.0:6831
      thrift_http:
        endpoint: 0.0.0.0:14268

  zipkin:
    endpoint: 0.0.0.0:9411

processors:
  batch:

exporters:
  debug:
    verbosity: detailed
  prometheusremotewrite:
    endpoint: http://<IP Адрес сервера метрик>:9090/api/v1/write
    retry_on_failure:
      enabled: true
      initial_interval: 5s
      max_interval: 60s
      max_elapsed_time: 120s

service:

  pipelines:

    traces:
      receivers: [ otlp, jaeger, zipkin ]
      processors: [ batch ]
      exporters: [ debug ]

    metrics:
      receivers: [ otlp, prometheus ]
      processors: [ batch ]
      exporters: [ debug, prometheusremotewrite ]

    logs:
      receivers: [ otlp ]
      processors: [ batch ]
      exporters: [ debug ]

  extensions: [ health_check, pprof, zpages ]
```

Запустите сервис:

```
sudo systemctl stop otelcol-contrib
sudo systemctl start otelcol-contrib
sudo systemctl status otelcol-contrib
```

6.2 Настройка сервера приложений

В папку с дистрибутивом проекта `global/globalserver/application/config/` добавить файл `otel-sdk.config.yaml`

```
file_format: "0.1"

disabled: false

resource:
  attributes:
    service.name: Global
    #service.name: ${env:OTEL_SERVICE_NAME}

exporters:
  otlp: &otlp-exporter
    timeout: 10000
    # https://opentelemetry.io/docs/specs/otel/protocol/exporter/#specify-protocol
    protocol: grpc
    endpoint: http://localhost:4318

logger_provider:
  processors:
    - batch:
        exporter:
          otlp: *otlp-exporter

meter_provider:
  readers:
    - periodic:
        interval: 5000
        timeout: 30000
        exporter:
          otlp:
            timeout: 10000
            protocol: grpc
            endpoint: http://localhost:4317

tracer_provider:
  processors:
    - batch:
        exporter:
          otlp:
            timeout: 10000
            protocol: http/protobuf
            endpoint: http://localhost:4318
```

И файл `otel-globalserver.config.yaml` с содержимым

```
instrumentation:
  common:
    default-enabled: true
  runtime-telemetry:
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    enabled: true
    emit-experimental-telemetry:
      enabled: true
  logback-appender:
    enabled: true
```

---

7. Настройка HAProxy

Подробнее о настройке выдачи метрик HAProxy в Prometheus в *Руководстве по настройке HAProxy*.

8. Установка Node Exporter

Внимание

Сервис устанавливается на сервер приложений и, при необходимости, на сервер СУБД

Проверить последний релиз можно в репозитории

```
wget https://github.com/prometheus/node_exporter/releases/download/v<VERSION>/node_
↪exporter-<VERSION>.<OS>-<ARCH>.tar.gz
tar xvfz node_exporter-*.tar.gz
sudo mv node_exporter-<VERSION>.<OS>-<ARCH>/node_exporter /usr/local/bin/
sudo useradd --no-create-home --shell /bin/false node_exporter
sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter
```

Создайте сервис `/etc/systemd/system/node_exporter.service`

```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter --web.listen-address=:9100 --collector.systemd --
↪collector.processes --collector.nfs

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

Запустите сервис:

```
sudo systemctl daemon-reload
sudo systemctl enable node_exporter
sudo systemctl start node_exporter
sudo systemctl status node_exporter
```

9. Установка Postgres Exporter

Внимание

Сервис устанавливается на сервер СУБД.

Проверить последний релиз можно в репозитории

```
wget https://github.com/prometheus-community/postgres_exporter/releases/download/v
↪<VERSION>/postgres_exporter-<VERSION>.<OS>-<ARCH>.tar.gz
tar -xf postgres_exporter-*.tar.gz
sudo cp postgres_exporter /usr/local/bin/
sudo chown -R postgres:postgres /usr/local/bin/postgres_exporter
```

Создайте сервис /etc/systemd/system/postgres_exporter.service

```
[Unit]
Description=Prometheus PostgreSQL Exporter
After=network.target

[Service]
Type=simple
Restart=always
User=postgres
Group=postgres
Environment=DATA_SOURCE_NAME="user=postgres host=/var/run/postgresql/ sslmode=disable"
ExecStart=/usr/local/bin/postgres_exporter

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

Запустите сервис:

```
sudo systemctl daemon-reload
sudo systemctl start postgres_exporter.service
sudo systemctl enable postgres_exporter.service
```

10. Установка pgSCV

Внимание

Сервис устанавливается на сервер СУБД. Используется для снятия метрик с Postgres, Pgbouncer.

Установка сервиса:

Проверить последний релиз можно в репозитории

```
curl -s -L https://github.com/cherts/pgscv/releases/download/v<VERSION>/pgscv_<VERSION>_
↪linux_<ARCH>.tar.gz -o - | tar xzf - -C /tmp && \
mv /tmp/pgscv.yaml /etc && \
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
mv /tmp/pgscv.service /etc/systemd/system &&  \
mv /tmp/pgscv.default /etc/default/pgscv && \
mv /tmp/pgscv /usr/sbin && \
chown postgres:postgres /etc/pgscv.yaml && \
chmod 640 /etc/pgscv.yaml && \
systemctl daemon-reload && \
systemctl enable pgscv --now
```

Приведите сервис `/etc/systemd/system/pgscv.service` к следующему виду:

```
[Unit]
Description=pgSCV - PostgreSQL ecosystem metrics collector
Documentation=https://github.com/cherts/pgscv/wiki
Requires=network-online.target
After=postgresql.service
Wants=postgresql.service

[Service]
Type=simple
User=postgres
Group=postgres
EnvironmentFile=-/etc/default/pgscv
# Start the agent process
ExecStart=/usr/sbin/pgscv $ARGS
# Kill all processes in the cgroup
KillMode=control-group
# Wait reasonable amount of time for agent up/down
TimeoutSec=5
# Restart agent if it crashes
Restart=on-failure
RestartSec=10
# if agent leaks during long period of time, let him to be the first person for eviction
OOMScoreAdjust=1000

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

- Создайте нового пользователя в postgres:

```
CREATE ROLE pgscv WITH LOGIN PASSWORD 'SUPERSECRETPASSWORD';
GRANT pg_read_server_files, pg_monitor TO pgscv;
GRANT EXECUTE on FUNCTION pg_current_logfile() TO pgscv;
```

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

```
SELECT passwd FROM pg_shadow WHERE usename = 'pgscv';
```

- Отредактируйте файл `/etc/pgbouncer/pgbouncer.ini`. Добавьте строку: `stats_users = pgscv`.
- Для работы аутентификации с вышеописанными настройками, в userlist.txt требуется установить следующие значения:

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

- Убедитесь, что в файле pg_hba.conf указан метод аутентификации scram-sha-256:

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

- Перезапустите PgBouncer:

```
sudo systemctl restart pgbouncer
```

- Удалите дефолтный `pgscv.yaml` и создайте новый:

```
cat << EOF > /etc/pgscv.yaml
listen_address: 0.0.0.0:9890
services:
  "postgres:5432":
    service_type: "postgres"
    conninfo: "postgres://pgscv:SUPERSECRETPASSWORD@127.0.0.1:<db_port>/<global_db_name>"
  "pgbouncer:6432":
    service_type: "pgbouncer"
    conninfo: "postgres://pgscv:SUPERSECRETPASSWORD@127.0.0.1:<pgbouncer_port>/pgbouncer"
EOF
```

- Выдайте необходимые права:

```
sudo chown postgres:postgres /etc/pgscv.yaml
sudo chmod 640 /etc/pgscv.yaml
```

- Запустите сервис:

```
sudo systemctl daemon-reload
sudo systemctl enable pgscv.service
sudo systemctl start pgscv.service
sudo systemctl status pgscv.service
```

Проверьте работу сервиса. Значения должны быть больше нуля.

```
curl -s http://127.0.0.1:9890/metrics | grep -c ^postgres
curl -s http://127.0.0.1:9890/metrics | grep -c ^pgbouncer
curl -s http://127.0.0.1:9890/metrics | grep -c ^node
curl -s http://127.0.0.1:9890/metrics | grep -c ^go
```

11. Настройка Global3 и GlobalScheduler

Необходимо включить вывод логов в указанную в конфигурационном файле Promtail директорию.

По умолчанию для Global3: `/opt/global/globalserver/logs/`.

Подробнее в *руководстве по Global3*

По умолчанию для GlobalScheduler: `/opt/global/globalserver/logs/scheduler/`.

Подробнее в *руководстве по GlobalScheduler*

12. Запуск скрипта снятия дополнительных метрик

Внимание

Сервис устанавливается на сервер приложений.

Необходимо загрузить и распаковать архив:

```
wget https://git.global-system.ru/devops/monitoring...
tar -czvf /path/to/archive.tar.gz /path/to/directory
```

Запустите скрипт на выполнение:

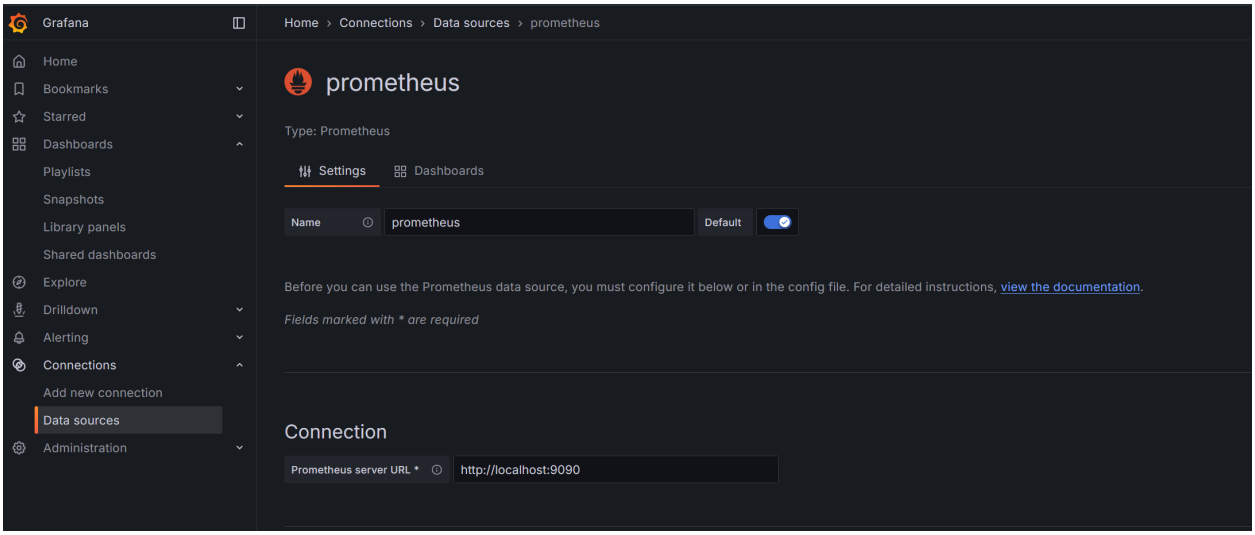
```
chmod +x install_service.sh
sudo ./install_service.sh global_telemetry GsTelemetryAgent.py
```

13. Добавление дашбордов в Grafana

После настройки сбора всех метрик, необходимо перейти на страницу grafana для добавления визуальных дашбордов (досок) для мониторинга состояния системы.

13.1 Подключение источников данных

1. В Grafana откройте меню Connections → Data sources → Add new data source.
2. Выберите Prometheus.
3. В поле Prometheus server URL укажите адрес сервера мониторинга (http://<IP_ADDRESS>:9090).
4. Нажмите Save & test.



Таким же образом добавьте Loki как источник данных, указав адрес сервера мониторинга (http://<IP_ADDRESS>:3100)

Примечание

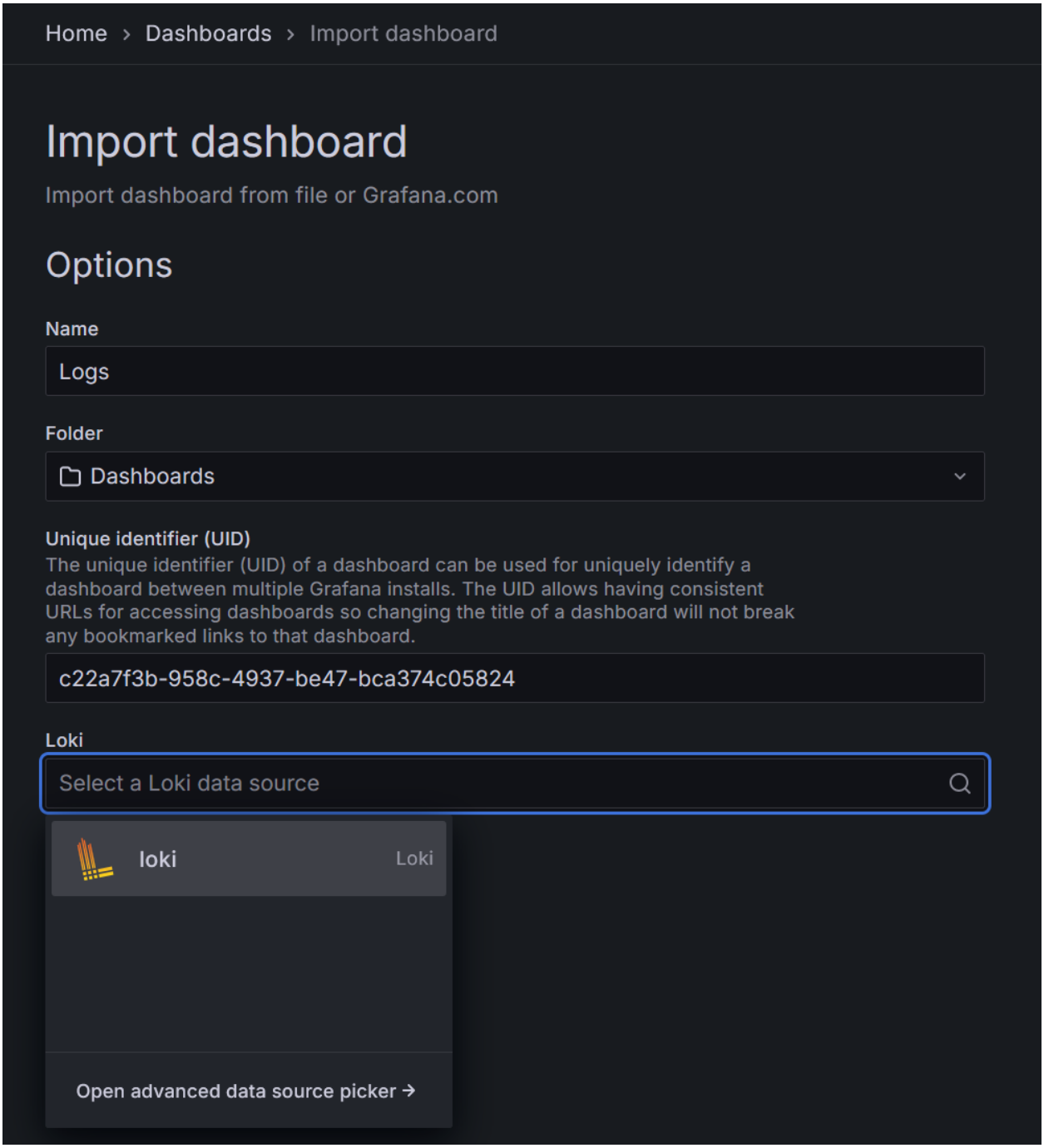
Если вы устанавливали Grafana, Prometheus, Loki на один сервер, в url источника данных можно прописывать localhost

13.2 Импорт готовых дашбордов

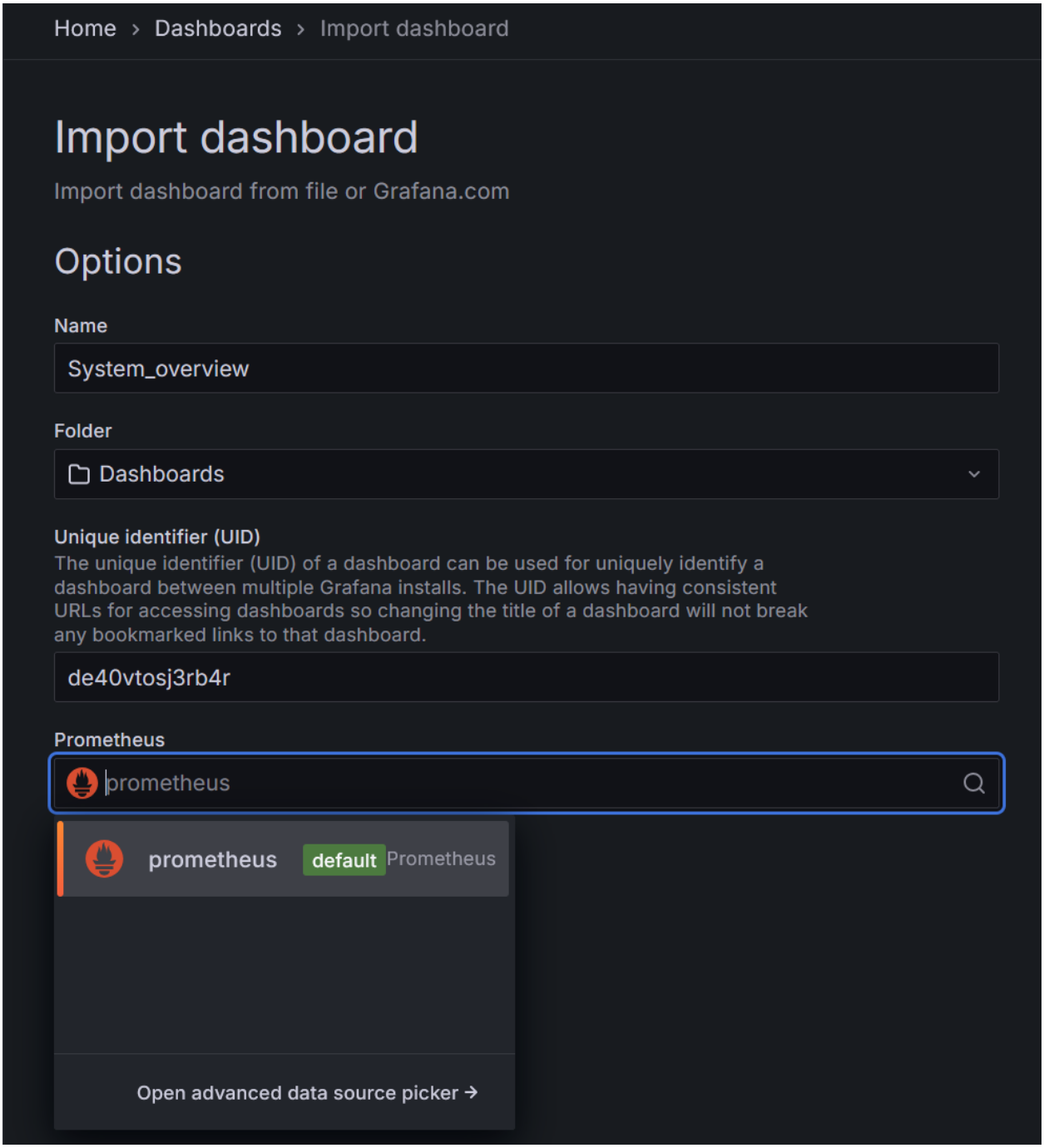
Скачайте все необходимые дашборды по ссылке.

Перейдите на вкладку `Dashboards`. Справа сверху кнопка `New - Import`. Загрузите по одному дашборды.

Если дашборд относится к отображению логов, в источнике данных выберете Loki:



Иначе - Prometheus:


Home › Dashboards › Import dashboard
Import dashboard
Import dashboard from file or Grafana.com
Options
Name
System_overview
Folder
Dashboards
Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.
de40vtosj3rb4r
Prometheus
prometheus
prometheus default Prometheus
Open advanced data source picker →

14. Настройка Firewall (UFW)

14.1. Установка и запуск

```
sudo apt install ufw
sudo systemctl enable ufw --now
```

14.2. Базовая политика безопасности

Применяем данные настройки на всех серверах:

```
sudo ufw default deny incoming
sudo ufw allow ssh
sudo ufw enable
```

- `default deny incoming` - запрет всех входящих соединений по умолчанию.
- `allow ssh` - разрешение SSH-доступа.
- `enable` - активация правил.

14.3. Разрешённые порты

Сервер мониторинга

```
sudo ufw allow 9090/tcp   # Prometheus
sudo ufw allow 3000/tcp   # Grafana
sudo ufw allow 3100/tcp   # Loki
```

Сервер СУБД

```
sudo ufw allow 9080/tcp   # Promtail
sudo ufw allow 9100/tcp   # Node exporter
sudo ufw allow 9187/tcp   # Postgres exporter
sudo ufw allow 9890/tcp   # pgSCV
```

Сервер приложений

```
sudo ufw allow 9080/tcp   # Promtail
sudo ufw allow 8405/tcp   # HAProxy telemetry
sudo ufw allow 8888/tcp   # Otelcol
sudo ufw allow 9100/tcp   # Node exporter
```

Важно

Пример приведён для тестовой среды. Также необходимо открыть порты для СУБД, сервера приложений (балансировщика). На продакшене рекомендуется ограничить доступ из вне ко всем портам мониторинга. При необходимости настроить proxy и другие меры безопасности.

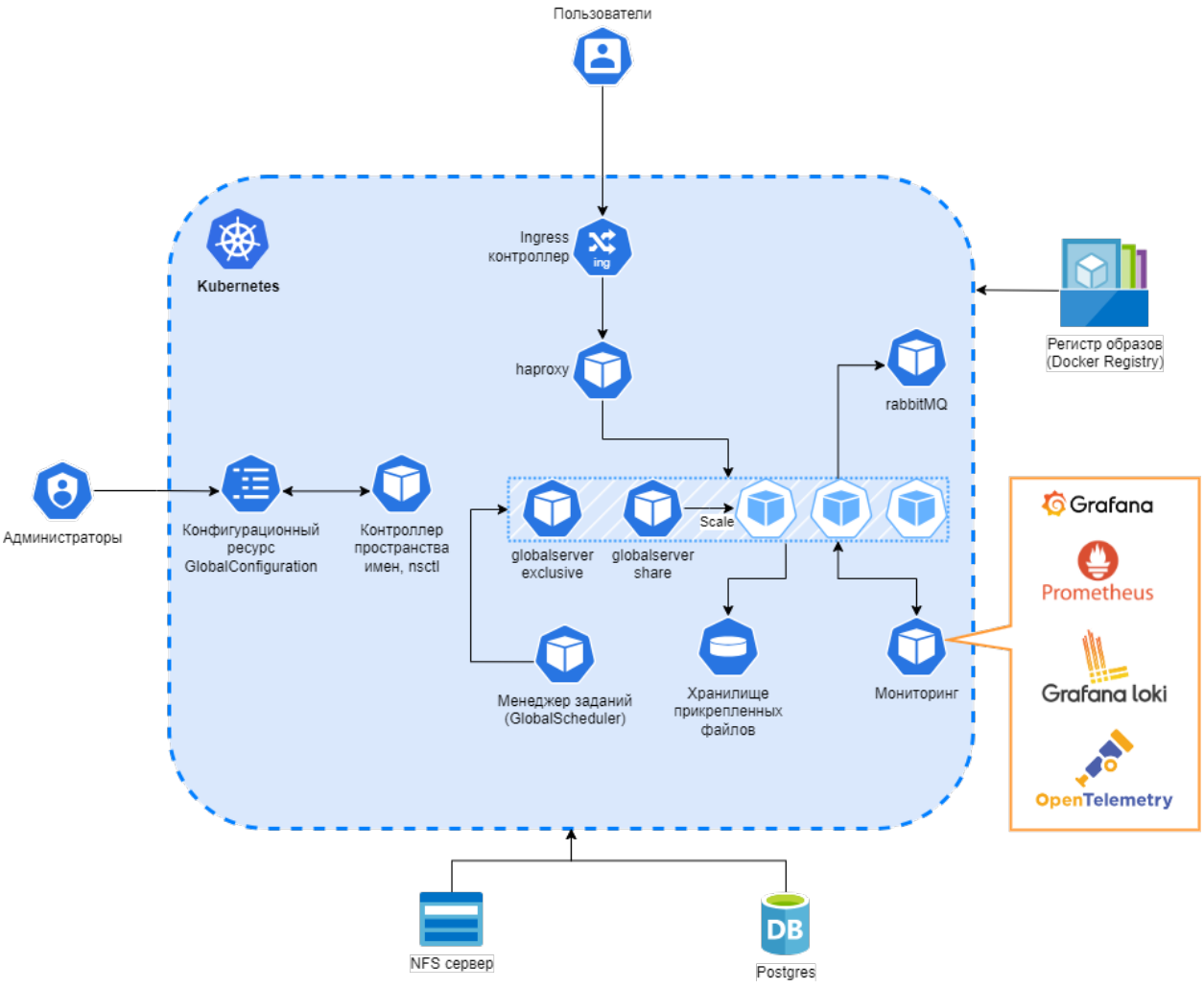
14.4. Проверка состояния

```
sudo ufw status verbose
```

4 GlobalServer кластер в kubernetes

4.1 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.
  - Шрифты
- Комплект приложения `appkit`
  - Дистрибутив сервера приложений `globalserver.zip`
  - Дистрибутив прикладного решения `applib.zip`
  - Пакет конфигурационных файлов («профиль») для элементов кластера (`globalserver`, `globalscheduler`, `haproxy`)
- NFS сервер Используется для хранения комплектов группы и комплектов приложений

- Сервер СУБД Postgres
- Реестр образов docker или доступ к https://dockerhub.global-system.ru/ Если кластер разворачивается в закрытой среде, то потребуется развернуть собственный реестр и загрузить в него необходимые образы.

4.2 Установка

Примечание

Эта инструкция - о развертывании Global ERP, используя gs-ctk версии 5.0 и выше.

Инструкция по установке, используя ранние версии gs-ctk, доступна *здесь*.

Для работы с Global ERP на кластере Kubernetes требуется:

- готовый к работе кластер Kubernetes, отвечающий вашим требованиям по производительности и отказоустойчивости,
- развернутая *PostgreSQL с базой данных для Global ERP*,
- NFS-хранилище,
- рабочее место, с которого производиться развертывание, представляющее собой командную оболочку ОС Debian 11 или выше.

В этой инструкции мы также предполагаем, что у вас есть *Ingress-контроллер*, который мы будем использовать в качестве прокси для подключения к серверу приложений.

Чтобы получить простейший кластер Kubernetes:

1. установите на хостах 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.
5. Установите пакет `nfs-common` на каждую ноду, чтобы Kubernetes мог смонтировать NFS.

Подробнее читайте в документации Kubernetes.

Примечание

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

Подготовка рабочего места

1. Установите kubectl.
2. Установите Helm.
3. Поместите файл kubeconfig, необходимый для авторизации пользователя перед кластером, по пути `~/.kube/config`. При создании кластера при помощи `kubeadm init`, такой файл для суперпользователя сохраняется по пути `/etc/kubernetes/admin.conf`.

Проверяем доступ:

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

При успешном подключении должны получить вывод:

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

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

Шаг 1. Скачайте nscli

Примечание

Временно Helm-чарт поставляется только вместе с nscli. Установка и использование nscli для развертывания при этом необязательны.

Скачайте и распакуйте nscli из нашего репозитория:

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

Мы рекомендуем также установить нужные пакеты и развернуть виртуальное окружение Python для запуска скриптов nscli:

```
chmod +x nscli/bin/*.sh
./nscli/bin/install.sh
```

Шаг 2. Установите контроллер nsctl

Установка контроллера nsctl может быть выполнена двумя способами в зависимости от ваших прав доступа к Kubernetes-кластеру:

- ***Автоматизированная установка***.

  Требует прав администратора. Helm-чарт самостоятельно создаст все необходимые ресурсы Kubernetes, включая пространство имен для Global ERP и ресурсы RBAC. *Выберите этот путь, если у вас есть права администратора кластера или вы работаете на тестовом стенде.*

- ***Установка с ручной подготовкой окружения***.

  Требует участия администратора кластера. Пространство имён и роли с правами доступа должны быть созданы вручную, а оставшиеся ресурсы, такие как ресурс контроллера nsctl и ConfigMap с параметрами, разворачиваются из Helm-чарта. *Выберите этот путь, если развертывание производится на кластере с ограниченными правами пользователя Kubernetes, например, на общекорпоративном коммунальном кластере.*

Автоматизированная установка

1. Подготовьте values.yaml:

   Важно

   Не забудьте изменить параметры доступа к системному файловому хранилищу

```
cat <<EOF | tee values.yaml
# Реестр образов
imageRepository: docker-registry.net

# Системное файловое хранилище для хранения комплектов групп и приложений
systemVolume:
    nfs:
        server: "nfs.server" # Путь к NFS-серверу
        path: /export # Путь к публикуемой на NFS-сервере папке

# Секрет загрузки с репозитория
imagePullSecret: "" # или "docker-secret", если Docker-реестр требует авторизации

# Селекторы нод для подов
#nodeSelector: {}
# Tolerations для подов
#tolerations: []
# Affinity для подов
#affinity: {}
EOF
```

   Примечание

   Docker-секрет будет создан позже, после развертывания пространства имен Helm-
   ↪чартом.

2. Разверните Helm-чарт вместе с пространством имен.

```
helm upgrade \
     --install \
     gs-ctk \
     ~/nscli/helm_chart \
     -f values.yaml
```

```
NAME: gs-ctk
LAST DEPLOYED: Tue Jan 19 03:14:08 2038
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
======================================

GS-CTK
Версия SNAPSHOT-5.1.3
Copyright © 2000-2025
Компания «Бизнес Технологии»
Все права защищены.

======================================
Развернут в пространстве имен `gs-ctk`.
(venv)
```

3. Создайте и примените Docker-секрет

```
docker login docker-registry.net # укажите здесь ваш репозиторий

cat <<EOF | tee docker-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: docker-secret
  namespace: gs-ctk
data:
  .dockerconfigjson: $(cat ~/.docker/config.json | base64 | tr -d '\n')
type: kubernetes.io/dockerconfigjson
EOF

kubectl apply -f docker-secret.yaml
```

4. Проверьте, что под nsctl находится в состоянии `Running`.

   На развертывание может потребоваться некоторое время, так как требуется скачать образ контейнера и запустить его. Обычно это занимает не больше пяти минут.

```
kubectl get pods --namespace gs-ctk
```

```
NAME                      READY   STATUS    RESTARTS   AGE
nsctl-84cb5578fd-b2c5x    1/1     Running   0          1h
```

Если nsctl запустился, у вас должно быть все готово к переходу *к следующему шагу - созданию секретов*.

Установка с ручной подготовкой окружения

1. Создайте от имени администратора Kubernetes пространство имен, роли с требуемыми полномочиями, сервисного пользователя и его привязки к ролям:

```
cat <<EOF | tee superuser-resources.yaml
# Пространство имен
apiVersion: v1
kind: Namespace
metadata:
  name: gs-ctk
---
# Роли
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: gs-ctk-role
  namespace: gs-ctk
rules:
  - apiGroups:
      - ""
      - apps
      - policy
      - rbac.authorization.k8s.io
      - coordination.k8s.io
      - networking.k8s.io
    resources:
      - pods
      - componentstatuses
      - configmaps
      - daemonsets
      - deployments
      - events
      - ingresses
      - pods
      - persistentvolumes
      - persistentvolumeclaims
      - poddisruptionbudgets
      - replicasets
      - services
      - statefulsets
      - secrets
      - "pods/log"
      - "pods/exec"
      - leases
    verbs: [ "*" ]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
metadata:
  name: gs-ctk-cluster-role
rules:
  - apiGroups:
    - global-system.ru
    - events.k8s.io
    resources:
    - "*"
    verbs:
    - "*"
---
# Сервисный пользователь и его привязки к ролям
apiVersion: v1
kind: ServiceAccount
metadata:
  name: gs-ctk-sa
  namespace: gs-ctk
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: gs-ctk-role-bind
  namespace: gs-ctk
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: gs-ctk-role
subjects:
- kind: ServiceAccount
  name: gs-ctk-sa
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: gs-ctk-cluster-role-bind
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: gs-ctk-cluster-role
subjects:
- kind: ServiceAccount
  name: gs-ctk-sa
EOF

kubectl apply -f superuser-resources.yaml
```

2. При использовании Docker с авторизацией, необходимо подготовить ресурс с секретом от репозитория.

```
docker login docker-registry.net # укажите здесь ваш репозиторий

cat <<EOF | tee docker-secret.yaml
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
apiVersion: v1
kind: Secret
metadata:
  name: docker-secret
  namespace: gs-ctk
data:
  .dockerconfigjson: $(cat ~/.docker/config.json | base64 | tr -d '\n')
type: kubernetes.io/dockerconfigjson
EOF

kubectl apply -f docker-secret.yaml
```

3. Примените от имени администратора CRD из `~/nscli/helm_chart/crds/crd.yaml`:

```
kubectl apply -f ~/nscli/helm_chart/crds/crd.yaml
```

Примечание

На этом этапе мы развернули все ресурсы, которые могут потребовать прав администратора. *Перед продолжением убедитесь, что вы переключились с контекста администратора на контекст непривилегированного пользователя.*

4. Подготовьте values.yaml по обычной инструкции развертывания. Обратите внимание, что при использовании Docker с авторизацией, требуется указать название секрета из 2 пункта в `imagePullSecret`.

```
# Реестр образов
imageRepository: "docker-registry.net"
# Тег образов, по-умолчанию равен версии чарта
imageTag: ""
# Секрет загрузки с репозитория
imagePullSecret: "" # "docker-secret" - ваш секрет с учетной записью докер-реестра

# Тип развертывания:
#  "cluster" - будут сгенерированы только ресурсы уровня кластера (пространство
↪имен, роли)
#  "namespace" - будут сгенерированы только ресурсы уровня пространства имен
↪(ConfigMap, ресурс с контроллером)
#  "both" - все ресурсы, значение по умолчанию. Обычно не требует изменения.
deploymentType: namespace

# Часовой пояс
timezone: "Europe/Moscow"

# Режим отладки для nsctl
nsctlDebugEnabled: false

# Суффикс, добавляемый к названию релиза, чтобы получить название пространства имен
namespaceSuffix: ""

# Системное файловое хранилище для хранения комплектов групп и приложений
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
systemVolume:
    nfs:
        server: "nfs.server"
        path: /export

# Субъекты RoleBinding/ClusterRoleBinding, которым стоит предоставить права по
# управлению GlobalConfiguration, ресурсами в пространстве имен. Можно
# использовать для предоставления прав администраторам GlobalERP. Например:
#privilegedSubjects:
#- kind: User
#  name: gs_admin
privilegedSubjects: []

# Переопределение ServiceAccount для установки nsctl и другим подам.
# По умолчанию используются пользователи, созданные этим Helm-чартом на уровне
# кластера.
overrideNsctlServiceAccount: gs-ctk-sa
overrideWorkerServiceAccount: gs-ctk-sa

# Аннотации подов
podAnnotations: {}
# Лейблы подов
podLabels: {}
# Селекторы нод для подов
nodeSelector: {}
# Tolerations для подов
tolerations: []
# Affinity для подов
affinity: {}
```

Обратите внимание, что должны быть установлены следующие параметры:

```
deploymentType: namespace # указывает, что разворачивать пространство имен и роли␣
↪не требуется
overrideNsctlServiceAccount: gs-ctk-sa # созданный ранее ServiceAccount
overrideWorkerServiceAccount: gs-ctk-sa # созданный ранее ServiceAccount
```

Примечание

Если для работы gs-ctk требуются Tolerations или nodeSelector, укажите их здесь, в values.yaml

5. Разверните ресурсы уровня пространства имен (ресурс контроллера nsctl и ConfigMap с параметрами) с правами непривилегированного пользователя:

```
helm upgrade \
     --install \
     gs-ctk \
     ~/nscli/helm-chart \
     -f values.yaml \
     --namespace gs-ctk \
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    --skip-crds
```

Примечание

Если не указать `--skip-crds`, то высока вероятность, что у вашего пользователя не хватит прав на развертывание.

```
NAME: gs-ctk
LAST DEPLOYED: Tue Jan 19 03:14:08 2038
NAMESPACE: gs-ctk
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
====================================

GS-CTK
Версия SNAPSHOT-5.1.3
Copyright © 2000-2025
Компания «Бизнес Технологии»
Все права защищены.

====================================
Развернут в пространстве имен `gs-ctk`.
(venv)
```

6. Проверьте, что под nsctl находится в состоянии `Running`.

   На развертывание может потребоваться некоторое время, так как требуется скачать образ контейнера и запустить его. Обычно это занимает не больше пяти минут.

```
kubectl get pods --namespace gs-ctk
```

```
NAME                       READY   STATUS    RESTARTS   AGE
nsctl-84cb5578fd-b2c5x   1/1     Running   0          1h
```

Если nsctl запустился, у вас должно быть все готово к переходу *к следующему шагу - созданию секретов*.

Шаг 3. Создайте секреты

Секреты используются для безопасного хранения учетных данных. Для работы Global ERP потребуются следующие секреты:

- *логин/пароль администратора Global ERP*
- *логин/пароль для подключения к базе данных PostgreSQL*
- *токен авторизации планировщика перед сервером приложений*
- *секреты HAProxy*:
  - логин/пароль для доступа к статистике HAProxy

  - если клиенты будут подключаться к балансировщику напрямую (без Ingress) по TLS, то также сертификат и приватный ключ
- *логин/пароль для доступа к мониторингу встроенной Grafana*
- *секреты встроенного RabbitMQ*:
  - логин/пароль от панели администратора RabbitMQ
  - логин/пароль для взаимодействия RabbitMQ и сервера приложений

Учетная запись администратора GlobalERP

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

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

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

Учетная запись для подключения к базе данных

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

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

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

Токен планировщика

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-ctk
data:
  private.key: <key>
```

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

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

Секреты HAProxy

Создайте секрет для доступа к статистике HAProxy

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

```
cat <<EOF | tee ~/nscli/workspace/haproxy-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: secret-haproxy-auth
  namespace: gs-ctk
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: t0p-Secret
EOF
```

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

При необходимости, создайте TLS-сертификат для HAProxy и поместите его в секрет

Создайте TLS-сертификат с помощью OpenSSL, заменив «example» домены на собственные.

Пример создания TLS-сертификата:

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

Пример создания мультидоменного TLS-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-ctk
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
type: Opaque
data:
tls.crt: |
    <cert-b64>
tls.key: |
    <key-b64>
```

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

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

Учетная запись администратора Grafana

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

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

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

Секреты RabbitMQ

- Создайте секрет с аккаунтом RabbitMQ для подключения к нему сервера приложений

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

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

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

- Создайте секрет с аккаунтом администратора RabbitMQ

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

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

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

Шаг 4. Настройте группы ресурсов

Примечание

Если вы *мигрировали с кластера предыдущей версии* переходите к шагу 5 - у вас уже есть конфигурационный ресурс.

Совет

Здесь конфигурационный ресурс будет формироваться при помощи мастеров утилиты nscli.

Вы можете сформировать его самостоятельно без использования утилиты. Подробнее читайте *здесь*.

1. Создайте конфигурационный ресурс и группу ресурсов в нем следующей командой:

```
./configmgr.sh configure_resgroup --config-path config.yaml --resgroup gs-cluster-1
```

```
Настройка группы ресурсов: gs-cluster-1
=======================================

Введите имя проекта (контура):
Введите timezone подов:Europe/Moscow
Введите поисковые домены для resolv.conf (если их несколько, вводите через пробел):
Введите url базы данных:jdbc:postgresql://host:5432/database
Введите alias базы данных:db_alias
Введите имя секрета для пользователя БД:db-user-secret
Укажите версию Java:8
Введите адрес к дополнительному GossipRouter или пропустите поле:
Введите тип прикладного хранилища:nfs
Введите адрес сервера(server):10.10.0.1
Введите путь(path):/export
Дополнительно отсылать метрики во внешнюю систему[да,нет]:нет

Группа ресурсов настроена!

Следующие действия:
- Добавьте книгу ресурсов: `./configmgr.sh configure_resbook --config config.yaml --
↪resgroup gs-cluster-1 --resbook [имя книги ресурсов] --classname [класс книги␣
↪ресурсов]`
- Список классов ресурсов можно получить при помощи команды `./configmgr.sh show_
↪resbook_classes`
- Список групп ресурсов можно получить при помощи команды `./configmgr.sh list_
↪resgroups --config config.yaml`
- Удалить группу ресурсов: `./configmgr.sh remove_resgroup --config config.yaml --
↪resgroup gs-cluster-1`
```

2. Создайте книгу ресурсов `global-server-excl`, описывающую эксклюзивный под с сервером приложений:

```
./configmgr.sh configure_resbook --config-path config.yaml --resgroup gs-cluster-1 -
↪-resbook global-server-excl --classname global_server_excl
```

```
Настройка книги ресурсов: global-server-excl (класс global_server_excl)
=======================================================================

Введите внешний ip(external_ip):
Введите максимальный размер кучи Java (java -Xmx):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
Дополнительно отсылать метрики во внешнюю систему[да,нет]:нет
Введите имя секрета для администратора:gs-admin-auth

Книга ресурсов настроена!

Следующие действия:
- Список книг ресурсов можно получить при помощи команды `./configmgr.sh list_
↪resbooks --config config.yaml --resgroup gs-cluster-1`
- Удалить книгу ресурсов: `./configmgr.sh remove_resbook --config config.yaml --
↪resgroup gs-cluster-1 --resbook global-server-excl`
```

3. Создайте книгу ресурсов `global-server-share`, описывающую группу подов с сервером приложений:

```
./configmgr.sh configure_resbook --config-path config.yaml --resgroup gs-cluster-1 -
↪-resbook global-server-share --classname global_server_share
```

```
Настройка книги ресурсов: global-server-share (класс global_server_share)
=========================================================================

Введите количество экземпляров:2
Введите длительность вежливого отключения в секундах (0 для немедленного␣
↪выключения):0
Введите максимальный размер кучи Java (java -Xmx):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
Дополнительно отсылать метрики во внешнюю систему[да,нет]:нет
Введите имя секрета для администратора:gs-admin-auth

Книга ресурсов настроена!
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
Следующие действия:
- Список книг ресурсов можно получить при помощи команды `./configmgr.sh list_
↪resbooks --config config.yaml --resgroup gs-cluster-1`
- Удалить книгу ресурсов: `./configmgr.sh remove_resbook --config config.yaml --
↪resgroup gs-cluster-1 --resbook global-server-share`
```

4. Создайте книгу ресурсов `global-scheduler`, описывающую планировщик задач:

```
./configmgr.sh configure_resbook --config-path config.yaml --resgroup gs-cluster-1 -
↪-resbook global-scheduler --classname global_scheduler
```

```
Настройка книги ресурсов: global-scheduler (класс global_scheduler)
======================================================================

Введите максимальный размер кучи Java (java -Xmx):800M
Введите запрос CPU для globalscheduler:2
Введите запрос MEMORY для globalscheduler:4G
Введите лимиты CPU для globalscheduler:2
Введите лимиты MEMORY для globalscheduler: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
Дополнительно отсылать метрики во внешнюю систему[да,нет]:нет
Введите имя секрета с токеном планировщика:scheduler-token-secret

Книга ресурсов настроена!

Следующие действия:
- Список книг ресурсов можно получить при помощи команды `./configmgr.sh list_
↪resbooks --config config.yaml --resgroup gs-cluster-1`
- Удалить книгу ресурсов: `./configmgr.sh remove_resbook --config config.yaml --
↪resgroup gs-cluster-1 --resbook global-scheduler`
```

5. Создайте книгу ресурсов `haproxy`, описывающую балансирощик:

```
./configmgr.sh configure_resbook --config-path config.yaml --resgroup gs-cluster-1 -
↪-resbook haproxy --classname haproxy
```

```
Настройка книги ресурсов: haproxy (класс haproxy)
================================================

Использовать Ingress?[да,нет]:да
Введите класс Ingress:nginx
Введите хост Ingress:
Введите дополнительную аннотацию Ingress (или пропустите поле):
Введите имя секрета basic-auth для авторизации статистики:secret-haproxy-auth
Введите имя tls секрета для доступа по https:
Введите запрос CPU для haproxy:1
Введите запрос MEMORY для haproxy:500M
Введите лимиты CPU для haproxy:2
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
Введите лимиты MEMORY для haproxy:1Gi

Книга ресурсов настроена!

Следующие действия:
- Список книг ресурсов можно получить при помощи команды `./configmgr.sh list_
↪resbooks --config config.yaml --resgroup gs-cluster-1`
- Удалить книгу ресурсов: `./configmgr.sh remove_resbook --config config.yaml --
↪resgroup gs-cluster-1 --resbook haproxy`
```

6. Создайте книгу ресурсов `grafana`, описывающую внутренний стек мониторинга:

Внимание

Данная книга ресурсов требует адрес сервиса kube-prometheus-stack. Он задаётся в формате „{{service_name}}.{{namepace}}.svc.cluster.local“

Для этого:

- Найдите неймспейс kube-prometheus-stack-а (обычно monitoring, prometheus-monitoring)

```
kubectl get namespaces
```

- Найдите имя сервиса прометея (сервис с портами 9090:web/TCP,8080:reloader-web/TCP, обычно prometheus-k8s, prometheus-prometheus)

```
kubectl get services -n {{namespace}}
```

Создайте книгу ресурсов

```
./configmgr.sh configure_resbook --config-path config.yaml --resgroup gs-cluster-1 -
↪-resbook grafana --classname grafana
```

```
Настройка книги ресурсов: grafana (класс grafana)
=================================================

Использовать Ingress?[да,нет]:нет
Введите внешний ip(external_ip):10.10.0.1
Введите класс хранилища:
Введите размер хранилища:
Введите адрес сервиса kube-prometheus-stack:prometheus-k8s.monitoring.svc.cluster.
↪local
Введите имя секрета с аккаунтом администратора Grafana:grafana-admin
Введите запрос CPU для grafana:1
Введите запрос MEMORY для grafana:500M
Введите лимиты CPU для grafana:2
Введите лимиты MEMORY для grafana:1Gi

Книга ресурсов настроена!

Следующие действия:
- Список книг ресурсов можно получить при помощи команды `./configmgr.sh list_
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪resbooks --config config.yaml --resgroup gs-cluster-1`
- Удалить книгу ресурсов: `./configmgr.sh remove_resbook --config config.yaml --
↪resgroup gs-cluster-1 --resbook grafana`
```

7. Создайте книгу ресурсов `rabbitmq`, описывающую встроенный экземпляр RabbitMQ:

```
./configmgr.sh configure_resbook --config-path config.yaml --resgroup gs-cluster-1 -
↪-resbook rabbitmq --classname rabbitmq
```

```
Настройка книги ресурсов: rabbitmq (класс rabbitmq)
==================================================

Введите название секрета для доступа к RabbitMQ:rabbitmq-global
Введите название секрета для доступа к консоли RabbitMQ:rabbitmq-admin
Введите название виртуального хоста RabbitMQ:globalrabbitmq
Введите внешний порт:15672
Использовать Ingress?[да,нет]:нет
Введите внешний ip(external_ip):

Книга ресурсов настроена!

Следующие действия:
- Список книг ресурсов можно получить при помощи команды `./configmgr.sh list_
↪resbooks --config config.yaml --resgroup gs-cluster-1`
- Удалить книгу ресурсов: `./configmgr.sh remove_resbook --config config.yaml --
↪resgroup gs-cluster-1 --resbook rabbitmq`
```

8. (Опционально) Создайте книгу ресурсов `img2doc`, описывающую встроенный экземпляр img2doc:

```
   ./configmgr.sh configure_resbook --config-path config.yaml --resgroup gs-cluster-1 --
↪resbook img2doc --classname img2doc
```

```
 Настройка книги ресурсов: img2doc (класс img2doc)
 ==================================================

 Введите порт:7000
 Введите кол-во реплик:1
 Введите запрос CPU для grafana:1
 Введите запрос MEMORY для grafana:500M
 Введите лимиты CPU для grafana:2
 Введите лимиты MEMORY для grafana:1Gi

 Книга ресурсов настроена!

 Следующие действия:
 - Список книг ресурсов можно получить при помощи команды `./configmgr.sh list_resbooks␣
↪--config config.yaml --resgroup gs-cluster-1`
 - Удалить книгу ресурсов: `./configmgr.sh remove_resbook --config config.yaml --
↪resgroup gs-cluster-1 --resbook img2doc`
```

Шаг 5. Подготовьте комплект приложений

Комплект приложений - это артефакты, необходимые 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 облачной отладки прикладного решения (appsrc)

Комплект хранится в виде zip-архивов (`globalserver.zip`, `applib.zip`, `profile.zip`). Сервер приложений и образ прикладного решения должен быть предоставлен вам контактным лицом. Содержимое `profile.zip` лежит в папке (`~/nscli/`)`default/profile` в дистрибутиве nscli.

Утилита nscli позволяет хранить комплект в распакованном виде в одноименных папках (`globalserver/`, `applib/`, `profile/`), при этом если `profile/` нет при запуске утилиты, она будет создана из (`~/nscli/`)`default/profile`. В таком случае файлы перед загрузкой на сетевое хранилище самой утилитой запаковываются в архив.

Совет

Мы здесь будем использовать утилиту nscli, однако это необязательно. Читайте подробнее *здесь*.

1. Загрузите комплект приложений на системное NFS-хранилище:

```
./appkit.sh push --namespace gs-ctk --source workspace/appkit --destination appkit/
↪v1
```

   - `--namespace gs-ctk` - пространство имен, в котором развертан gs-ctk
   - `--source workspace/appkit` - путь к папке с комплектом на локальной ФС относительно текущей папки
   - `--destination appkit/v1` - путь к папке, в которую следует поместить комплект, относительно *точки монтирования системного хранилища*

2. Обновите хеши и пути к комплекту приложения (appkit) в конфигурации при помощи одной из следующих команд:

   - Эта команда не требует подключения к кластеру, хеш-суммы будут считаны с комплекта приложений на локальном диске

```
./appkit.sh switch_local --config-path ./config.yaml --resgroup gs-cluster-1 --
↪local-appkit workspace/appkit --remote-appkit appkit/v1
```

     - `--config-path ./config.yaml` - путь к конфигурационном ресурсу
     - `--resgroup gs-cluster-1` - название группы ресурсов
     - `--local-appkit workspace/appkit` - путь к папке с комплектом на локальной ФС относительно текущей папки (`--source` из предыдущего пункта)
     - `--remote-appkit appkit/v1` - путь к папке на системном хранилище, в котором находится комплект (`--destination` из предыдущего пункта)

   ИЛИ

- При выполнении этой команды хеш-суммы читаются с комплекта приложений на хранилище в кластере, следовательно требует подключения к кластеру

```
# хеши читаются с комплекта приложений на хранилище в кластере, следовательно␣
↪требует подключения к кластеру
./appkit.sh switch_remote --config-path ./config.yaml --resgroup gs-cluster-1 --
↪namespace gs-ctk --remote-appkit appkit/v1
```

  - `--config-path ./config.yaml` - путь к конфигурационном ресурсу
  - `--resgroup gs-cluster-1` - название группы ресурсов
  - `--namespace gs-ctk` - пространство имен, в котором развертан gs-ctk
  - `--remote-appkit appkit/v1` - путь к папке на системном хранилище, в котором находится комплект (`--destination` из предыдущего пункта)

3. Переведите комплект приложения в состояние `started`:

```
./appkit.sh start --config-path ./config.yaml --resgroup gs-cluster-1
```

Шаг 6. Подготовьте комплект группы

Комплект группы (groupkit) - это дополнительные и редко изменяемые артефакты, используемые в развертывании Global ERP. В их число входят:

- Архив с дополнительными библиотеками (libs)
- Архив с дополнительными шрифтами (fonts)
- Файлы JDK для переопределения (java), например cacerts.

Наличие всех компонентов необязательно, и обычно **комплект группы не требуется, поэтому вы можете пропустить этот шаг**.

1. Загрузите комплект группы на системное NFS-хранилище аналогично комплекту приложений:

```
./groupkit.sh push --namespace gs-ctk --source workspace/groupkit --destination␣
↪groupkit/v1
```

2. Добавьте в файл (`config.yaml`) параметр группы ресурсов `groupkit`:

```
apiVersion: global-system.ru/v1
kind: GlobalConfiguration
metadata:
  name: config
spec:
  type: advanced
  resgroups:
  - name: gs-cluster-1
    groupkit: groupkit/v1
    ...
```

Шаг 7. Примените конфигурацию при помощи kubectl

К этому шагу у вас должен был получиться конфигурационный файл приблизительно следующего содержания:

```
apiVersion: global-system.ru/v1
kind: GlobalConfiguration
metadata:
  name: config
spec:
  type: advanced
  resgroups:
  - name: gs-cluster-1
    #groupkit: groupkit/v1
    appkit:
      applib_sha1: e1ffb3becd7e9315e371adcc7615a29f4df59e3e
      globalserver_sha1: 76416b8e09ec36438ad84aa6cb89c2a699496719
      profile_sha1: e5b8876dec7122ca221f6a17a74d429bdad66a00
      globalserver_instance: '1'
      path: appkit/v1
      state: started
    database_url: jdbc:postgresql://host:5432/database
    database_alias: db_alias
    database_secret: db-user-secret
    appvol:
      type: nfs
      server: 10.10.0.1
      path: /export
    resbooks:
    - type: global_server_excl
      name: global-server-excl
      admin_secret: gs-admin-auth
    - type: global_server_share
      name: global-server-share
      admin_secret: gs-admin-auth
    - type: global_scheduler
      name: global-scheduler
      scheduler_token_secret: scheduler-token-secret
    - type: haproxy
      name: haproxy
      service:
        type: ingress
        ingress_class: nginx
        host: ''
      statistics_secret: secret-haproxy-auth
    - type: grafana
      name: grafana
      service:
        type: ip
        external_ip: 10.10.0.1
      admin_secret: grafana-admin
    - type: rabbitmq
      name: rabbitmq
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    apmq_secret: rabbitmq-global
    admin_secret: rabbitmq-admin
```

Мы готовы к развертыванию Global ERP. Для этого выполним:

```
kubectl apply -f ./config.yaml
```

Начнется инициализация контейнеров, после чего сервер приложений должен быть доступен через браузер, войти однако не получится - требуется проинициализировать базу данных.

Совет

В случае неполадок посмотрите созданное пространство имен на наличие перезагружающихся подов. Также в случае ошибок с конфигурацией nsctl создаст событие на конфигурационный ресурс.

Шаг 8. Запустите обновление базы данных

1. Выполните следующую команду:

   ```
   ./appkit.sh upgrade --config-path ./config.yaml --resgroup gs-cluster-1
   ```

2. Повторно примените ресурс:

   ```
   kubectl apply -f ./config.yaml
   ```

Совет

Для отслеживания статуса обновления и чтения логов, используйте команду:

```
./appkit.sh wait_upgrade --namespace gs-ctk --resgroup gs-cluster-1
```

Следующие шаги

- *Настройте планировщик задач*
- *Настройте взаимодействие серверов приложений через JGroups*
- *Разверните Global ERP через единый Helm-чарт*

4.3 Миграция с предыдущих версий gs-ctk

Вы можете перенести существующую конфигурацию gs-ctk версии 4 или более раннюю на версию 5.0 при помощи утилиты nscli.

1. Скачайте и установите на jump-хост утилиту nscli версии 5.0.0:

   ```
   cd ~
   wget --backups=1 --user=<пользователь> --ask-password "https://repo.global-system.
   ↪ru/artifactory/general/ru/bitec/gs-ctk-nscli/SNAPSHOT-5.0.0/gs-ctk-nscli-SNAPSHOT-
   ```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪5.0.0.zip"
unzip -o gs-ctk-nscli-SNAPSHOT-5.0.0.zip -d nscli
chmod +x nscli/bin/initvenv.sh
./nscli/bin/initvenv.sh
```

2. Выгрузите существующую конфигурацию с кластера версии 4 или ранее при помощи этой утилиты:

```
cd nscli
./configmgr.sh export_legacy --namespace gs-ctk --file old_config.yaml
```

3. Создайте конфигурацию версии 5.0.0

```
./configmgr.sh migrate_legacy --name my-migrated-config --legacy-config old_config.
↪yaml --new-config new_config.yaml
```

Разверните кластер версии 5.0.0 и примените получившийся файл.

4.4 Настройки конфигурации

Конфигурационный ресурс - основной источник информации о системе для gs-ctk.

Документация по доступным настройкам будет формироваться автоматически позже. Пока для ознакомления со всеми опциями предлагаем вам пример максимальной конфигурации с кратким описанием всех настроек.

Пример конфигурационного ресурса

Совет

Обратите внимание, что некоторые опции (admin_secret, send_metrics_to_external_system, external_prometheus_endpoint: external.prometheus.domain, external_loki_endpoint) повторяются в настройках группы и книг ресурсов. В таких случаях вы можете установить только одну настройку, только другую или обе - опция из книги ресурсов переопределяет более общую настройку для группы ресурсов.

Указывать опции для которых указано значение по умолчанию необязательно.

```
apiVersion: global-system.ru/v1
kind: GlobalConfiguration
metadata:
  name: gs-ctk-config
spec:
  # type - пока только advanced
  type: advanced
  # resgroups - список групп ресурсов
  resgroups:
    # name - название группы ресурсов
  - name: gs-cluster-1
    # namespace - название пространства имен, в котором необходимо развернуть группу␣
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪(если развернуто несколько операторов, по умолчанию - пусто)
  namespace: gs-ctk
  # enabled - включена ли группа ресурсов (по умолчанию - true)
  enabled: false
  # appkit - читайте ниже
  appkit:
    applib_sha1: 93c8a473ed8d0b6c20c30b952a8ebcebbad5af9c
    appsrc_sha1:
    globalserver_sha1: d70488cd57b20794be30779dded0c0a86781eb02
    profile_sha1: 15bd21a523102fb41d32b87ffdac8880f3e37427
    globalserver_instance: '2'
    path: appkits/v1
    state: started
  # database_schema_version - читайте ниже
  database_schema_version: '1'
  # groupkit - путь к комплекту группы (по умолчанию - пусто)
  groupkit: groupkits/v1
  # timezone - часовой пояс (по умолчанию - Europe/Moscow)
  timezone: Europe/Moscow
  # database_url - путь к базе данных
  database_url: jdbc:postgresql://host:5432/database
  # database_alias - имя БД в системе
  database_alias: ssng
  # database_secret - секрет с логином и паролем к БД
  database_secret: sng-db
  # search_domains - поисковые домены (по умолчанию - пусто)
  search_domains:
  - mycompany.ru
  # gossiprouters - список хостов/портов gossiprouter (по умолчанию - пусто)
  gossiprouters:
  - host: gr.mycompany.ru
    port: 1100
  # admin_secret - имя единого секрета администратора (по умолчанию - пусто)
  admin_secret:
  # java_version - версия java (по умолчанию - '8')
  java_version: '8'
  # appvol - прикладное хранилище
  appvol:
    type: nfs
    server: nfs.mycompany.ru
    path: /export
  # extra_vols - дополнительные точки монтирования (по умолчанию - пусто)
  extra_vols:
  - name: mp1
    path: mnt/mp1
    volume:
      type: nfs
      server: nfs.mycompany.ru
      path: /export2
  # debug_mode - выключить перезагрузку подов при ошибках (по умолчанию - false)
  debug_mode: true
  # rabbitmq - настройки RabbitMQ
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    # по умолчанию - берутся из настроек книги ресурсов rabbitmq, если такой нет -␣
↪RabbitMQ отключен
    # поставьте `rabbitmq: false`, чтобы выключить
    rabbitmq:
      enabled: true # значение по-умолчанию
      host: rabbitmq.mycompany.com
      port: 5672 # порт по-умолчанию
      vhost: globalerp
      secret: rabbitmq-secret
    # точки выгрузки внутренней Grafana, оставляйте пустым, если в целях отладки не␣
↪требуется другое
    #prometheus_endpoint:
    #loki_endpoint:
    #tempo_endpoint:

    # send_metrics_to_external_system - отправлять данные во внешнюю систему (по␣
↪умолчанию - false)
    send_metrics_to_external_system: true
    # external_prometheus_endpoint - внешняя точка выгрузки данных Prometheus
    external_prometheus_endpoint: external.prometheus.domain:9090
    # external_loki_endpoint - внешняя точка выгрузки данных Loki
    external_loki_endpoint: external.loki.domain:3100

    # название контура (по умолчанию - название пространства имен)
    project_name: gs-ctk
    # селектор нод, как в Kubernetes
    #node_selector: {}

    # книги ресурсов
    resbooks:

      # global_scheduler - планировщик задач
    - type: global_scheduler
      # name - имя книги
      name: global-scheduler
      # enabled - включена ли книга ресурсов (по умолчанию - true)
      enabled: true
      # debug_mode - выключить перезагрузку подов при ошибках (по умолчанию - false)
      debug_mode: true
      # селектор нод, как в Kubernetes
      #node_selector: {}
      # java_xmx - предельный объем кучи виртуальной машины Java (по умолчанию - 800M)
      java_xmx: 800M
      # send_metrics_to_external_system - отправлять данные во внешнюю систему (по␣
↪умолчанию - false)
      send_metrics_to_external_system: false
      # external_prometheus_endpoint - внешняя точка выгрузки данных Prometheus
      external_prometheus_endpoint: external.prometheus.domain:9090
      # external_loki_endpoint - внешняя точка выгрузки данных Loki
      external_loki_endpoint: external.loki.domain:3100
      # resources - запросы и лимиты ресурсов контейнера с планировщиком (указаны␣
↪значения по умолчанию)
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    resources:
      requests:
        memory: 1G
        cpu: '1'
      limits:
        memory: 1G
        cpu: '1'
    # agent_resources - запросы и лимиты ресурсов контейнера с агентом сбора␣
↪телеметрии (указаны значения по умолчанию)
    agent_resources:
      requests:
        memory: 250M
        cpu: '1'
      limits:
        memory: 500M
        cpu: '1'
    # scheduler_token_secret - секрет с токеном шедулера
    scheduler_token_secret: scheduler_token

    # global_server_excl - эксклюзивный под Global Server
  - type: global_server_excl
    # name - имя книги
    name: global-server-excl
    # enabled - включена ли книга ресурсов (по умолчанию - true)
    enabled: true
    # debug_mode - выключить перезагрузку подов при ошибках (по умолчанию - false)
    debug_mode: true
    # селектор нод, как в Kubernetes
    #node_selector: {}
    # java_xmx - предельный объем кучи виртуальной машины Java (по умолчанию - 3500M)
    java_xmx: 3500M
    # send_metrics_to_external_system - отправлять данные во внешнюю систему (по␣
↪умолчанию - false)
    send_metrics_to_external_system: false
    # external_prometheus_endpoint - внешняя точка выгрузки данных Prometheus
    external_prometheus_endpoint: external.prometheus.domain:9090
    # external_loki_endpoint - внешняя точка выгрузки данных Loki
    external_loki_endpoint: external.loki.domain:3100
    # resources - запросы и лимиты ресурсов контейнера с сервером приложений (указаны␣
↪значения по умолчанию)
    resources:
      requests:
        memory: 4G
        cpu: '2'
      limits:
        memory: 4G
        cpu: '2'
    # agent_resources - запросы и лимиты ресурсов контейнера с агентом сбора␣
↪телеметрии (указаны значения по умолчанию)
    agent_resources:
      requests:
        memory: 250M
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        cpu: '1'
      limits:
        memory: 500M
        cpu: '1'
    # admin_secret - имя секрета администратора
    admin_secret: gs-admin
    # service - способ публикации книги ресурсов (по умолчанию - пусто)
    # global_server_excl поддерживает публикацию только через IP
    service:
      type: ip
      external_ip: 10.20.0.5

    # global_server_share - масштабируемые поды Global Server
  - type: global_server_share
    # name - имя книги
    name: global-server-share
    # enabled - включена ли книга ресурсов (по умолчанию - true)
    enabled: true
    # debug_mode - выключить перезагрузку подов при ошибках (по умолчанию - false)
    debug_mode: true
    # селектор нод, как в Kubernetes
    #node_selector: {}
    # java_xmx - предельный объем кучи виртуальной машины Java (по умолчанию - 3500M)
    java_xmx: 3500M
    # send_metrics_to_external_system - отправлять данные во внешнюю систему (по␣
↪умолчанию - false)
    send_metrics_to_external_system: false
    # external_prometheus_endpoint - внешняя точка выгрузки данных Prometheus
    external_prometheus_endpoint: external.prometheus.domain:9090
    # external_loki_endpoint - внешняя точка выгрузки данных Loki
    external_loki_endpoint: external.loki.domain:3100
    # resources - запросы и лимиты ресурсов контейнера с сервером приложений (указаны␣
↪значения по умолчанию)
    resources:
      requests:
        memory: 4G
        cpu: '2'
      limits:
        memory: 4G
        cpu: '2'
    # agent_resources - запросы и лимиты ресурсов контейнера с агентом сбора␣
↪телеметрии (указаны значения по умолчанию)
    agent_resources:
      requests:
        memory: 250M
        cpu: '1'
      limits:
        memory: 500M
        cpu: '1'
    # admin_secret - имя секрета администратора
    admin_secret: gs-admin
    # replicas - количество реплик
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    replicas: 1

    # grafana - под со стеком мониторинга
  - type: grafana
    # name - имя книги
    name: grafana
    # enabled - включена ли книга ресурсов (по умолчанию - true)
    enabled: true
    # debug_mode - выключить перезагрузку подов при ошибках (по умолчанию - false)
    debug_mode: true
    # селектор нод, как в Kubernetes
    #node_selector: {}
    # service - способ публикации книги ресурсов (по умолчанию - пусто)
    # grafana поддерживает публикацию через IP и через Ingress (см. haproxy)
    service:
      type: ip
      external_ip: 10.20.0.6
    # resources - запросы и лимиты ресурсов контейнера (указаны значения по умолчанию)
    resources:
      requests:
        memory: 500M
        cpu: '1'
      limits:
        memory: 1Gi
        cpu: '2'
    # storage_class - класс хранилища для постоянного тома Grafana
    storage_class: local-path
    # storage_size - запрашиваемое пространство (по умолчанию - 1Gi)
    storage_size: 4Gi
    # admin_secret - имя секрета администратора
    admin_secret: grafana-secret

    # haproxy - под с балансировщиком нагрузки
  - type: haproxy
    # name - имя книги
    name: haproxy
    name: grafana
    # enabled - включена ли книга ресурсов (по умолчанию - true)
    enabled: true
    # debug_mode - выключить перезагрузку подов при ошибках (по умолчанию - false)
    debug_mode: true
    # селектор нод, как в Kubernetes
    #node_selector: {}
    # service - способ публикации книги ресурсов (по умолчанию - пусто)
    # haproxy поддерживает публикацию через IP (см. grafana) и через Ingress
    service:
      type: ingress
      ingress_class: nginx
      host: globalerp.mycompany.ru # '' - для публикации на всех хостах
      annotations: {} # дополнительные аннотации
    # resources - запросы и лимиты ресурсов контейнера с балансировщиком (указаны␣
↪значения по умолчанию)
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    resources:
      requests:
        memory: 500M
        cpu: '1'
      limits:
        memory: 1G
        cpu: '2'
    # agent_resources - запросы и лимиты ресурсов контейнера с агентом сбора␣
↪телеметрии (указаны значения по умолчанию)
    agent_resources:
      requests:
        memory: 250M
        cpu: '1'
      limits:
        memory: 500M
        cpu: '1'
    # statistics_secret - секрет для сбора статистики
    statistics_secret: haproxy-secret
    # tls_secret - секрет для ключа и сертификата TLS (по умолчанию - пусто, TLS␣
↪отключен)
    tls_secret: ''
    # replicas - количество реплики при работе через Ingress (по умолчанию - 1)
    replicas: 1
    # send_metrics_to_external_system - отправлять данные во внешнюю систему (по␣
↪умолчанию - false)
    send_metrics_to_external_system: false
    # external_loki_endpoint - внешняя точка выгрузки данных Loki
    external_loki_endpoint: external.loki.domain:3100

    # rabbitmq - под с rabbitmq
  - type: rabbitmq
    # name - имя книги
    name: rabbitmq
    # enabled - включена ли книга ресурсов (по умолчанию - true)
    enabled: true
    # debug_mode - выключить перезагрузку подов при ошибках (по умолчанию - false)
    debug_mode: true
    # селектор нод, как в Kubernetes
    #node_selector: {}
    # service - способ публикации консоли администратора книги ресурсов (по умолчанию -
↪ пусто)
    # rabbitmq поддерживает публикацию через IP (см. grafana) и через Ingress (см.␣
↪haproxy)
    service:
    # resources - запросы и лимиты ресурсов контейнера (указаны значения по умолчанию)
    resources:
      requests:
        memory: 2Gi
        cpu: '1'
      limits:
        memory: 4Gi
        cpu: '2'
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
      # apmq_secret - секрет для соединения между RabbitMQ и GlobalERP
      apmq_secret: apmq-secret
      # admin_secret - секрет администратора (по умолчанию - пусто)
      admin_secret: rabbitmq-admin-secret
      # vhost - название виртуального хоста (по умолчанию - 'globalrabbitmq')
      vhost: globalrabbitmq
      # admin_port - порт консоли администратора (по умолчанию - 15672)
      admin_port: 15672

      # jgroups_dns - включает возможность общения серверов приложений между собой
    - type: jgroups_dns
      # name - имя книги
      name: jgroups-dns

      # gossiprouter - роутер сообщений между серверами приложений, более продвинутое,␣
↪чем jgroups_dns решение
    - type: gossiprouter
      # name - имя книги
      name: gossiprouter
      # enabled - включена ли книга ресурсов (по умолчанию - true)
      enabled: true
      # debug_mode - выключить перезагрузку подов при ошибках (по умолчанию - false)
      debug_mode: true
      # селектор нод, как в Kubernetes
      #node_selector: {}
      # service - способ публикации книги ресурсов (по умолчанию - пусто)
      # gossiprouter поддерживает публикацию только через IP
      service:
        type: ip
        external_ip: 10.20.0.7
      # java_xmx - предельный объем кучи виртуальной машины Java (по умолчанию - 800M)
      java_xmx: 800M
      # send_metrics_to_external_system - отправлять данные во внешнюю систему (по␣
↪умолчанию - false)
      send_metrics_to_external_system: false
      # external_prometheus_endpoint - внешняя точка выгрузки данных Prometheus
      external_prometheus_endpoint: external.prometheus.domain:9090
      # external_loki_endpoint - внешняя точка выгрузки данных Loki
      external_loki_endpoint: external.loki.domain:3100
      # resources - запросы и лимиты ресурсов контейнера с планировщиком (указаны␣
↪значения по умолчанию)
      resources:
        requests:
          memory: 1G
          cpu: '1'
        limits:
          memory: 1G
          cpu: '1'
      # agent_resources - запросы и лимиты ресурсов контейнера с агентом сбора␣
↪телеметрии (указаны значения по умолчанию)
      agent_resources:
        requests:
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        memory: 250M
        cpu: '1'
      limits:
        memory: 500M
        cpu: '1'
    # gossip_router_port - порт GossipRouter (по умолчанию - 12001)
    gossip_router_port: 12001
```

Комплект группы (appkit) и версия схемы

Конфигурации группы ресурсов содержит информацию о комплекте приложения и целевой версии схемы базы данных:

- `appkit` - Описание текущего комплекта приложений. При изменении настроек комплекта приложений, после применения конфигурации сервера приложений будут автоматически перезапущены с новой версией компекта приложений.
  - `applib_sha1`, `appsrc_sha1`, `globalserver_sha1` и `profile_sha1` - хеш-суммы соответствующих компонентов appkit. Из всех только `appsrc_sha1` может не указываться.
  - `globalserver_instance` - версия сервера приложений, желательно обновлять при каждом изменении `appkit`.
  - `path` - путь к комплекту на системном NFS-хранилище.
  - `state` - состояние комплекта приложений. Возможные значения:
    - `started` - запущен, сервера принимают пользователей
    - `stopped` - остановлен, сервера отключены
    - `drained` - осушен, доступ через балансировщик закрыт, непривилегированные пользователи отключены
- `database_schema_version` - версия схемы БД. При изменении целевой версии схемы базы данных - автоматически будет запущено обновление БД. Информация о текущей версии хранится в ресурсе, в поле status.

4.5 Развертывание одним чартом

Вы можете развернуть кластер Системы Global из одного чарта, поместив в него и пространство имен gs-ctk, и конфигурационный ресурс. Для этого:

1. Создайте чарт при помощи команды `helm create`.
2. Добавьте в папку `charts` наш чарт gs-ctk из дистрибутива nsctl
3. Очистите папку `templates` и добавьте ваш конфигурационный файл. Вы можете шаблонизировать его, например, выделив параметры комплекта приложений для удобства или для автоматической их подстановки *вашим CI/CD-скриптом*.
4. Добавьте в values.yaml *параметры развертывания gs-ctk*, например:

```
gs-ctk:
    imageRepository: "docker-register.net"
    imagePullSecret: "docker-secret"
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    systemVolume:
        nfs:
        server: "10.10.0.1"
        path: /export(venv)
```

У вас должен получиться следующий чарт:

```
my-chart
├── charts
│   └── gs-ctk
│       ├── Chart.yaml
│       └── ...
├── Chart.yaml
├── templates
│   └── globalconfigurations.yaml
└── values.yaml
```

5. Создайте и разверните секреты в вашем кластере, или воспользуйтесь одним из решений *ниже*.
6. Разверните его при помощи команды:

```
helm --install upgrade my-release .
```

Весь кластер развернут из одного чарта при помощи одной команды Helm!

Управление секретами

Созданный нами чарт сейчас не разворачивает секреты автоматически. Этот вопрос умышленно выведен из рассмотрения в инструкции выше, поскольку вопрос хранения секретов сложный.

Есть несколько способов автоматизировать развертывание секретов:

1. Сделать шаблоны секретов и передавать значения через Helm. Это допустимый вариант, если секреты внедряются в надежном месте (то есть, например, не хранятся в values.yaml открыто).
2. Использовать helm-secrets и sops для хранения и подстановки секретов.
3. Использовать Sealed Secrets, External Secrets или другой оператор для Kubernetes, который забирает на себя задачу непосредственного развертывания секретов.

Напоминаем, что хранение секретов в открытом виде вместе с самим чартом крайне опасно.

4.6 Хуки

Для каждой книги или целой группы ресурсов gs-ctk есть список доступных хуков. Текст хуков подставлется в шаблон ресурсов книги

Список хуков:

- `tolerations` - добавление Tolerations для пода.

> **Примечание**
>
> Указывайте Tolerations в *values.yaml при развертывании Helm-чарта*

Использование этого хука переопределяет параметры, указанные в values.yaml.

Пример настройки хуков

```
apiVersion: global-system.ru/v1
kind: GlobalConfiguration
metadata:
  name: config
spec:
  type: advanced
  resgroups:
  - name: gs-cluster-1
    hooks:
      tolerations: |
        tolerations:
        - key: "node"
          operator: "Equal"
          value: "first"
          effect: "NoSchedule"
    resbooks:
    - type: global_server_excl
      name: global-server-excl
      hooks:
        # хук книги переопределяет хук группы для данной книги
        tolerations: |
          tolerations:
          - key: "node"
            operator: "Equal"
            value: "second"
            effect: "NoSchedule
    - type: global_server_share
      name: global-server-share
    ...
```

4.7 Обновление

Обновление комплекта приложения с использованием nscli

1. Для обновления *комплекта приложения* (appkit: сервера приложений, прикладного решения) подготовьте его и загрузите на NFS-хранилище:

   ```
   ./appkit.sh push --namespace gs-ctk --source workspace/appkit --destination appkit/
   ↪v2
   ```

   - `--namespace gs-ctk` - пространство имен, в котором развертан gs-ctk
   - `--source workspace/appkit` - путь к папке с комплектом на локальной ФС относительно текущей папки
   - `--destination appkit/v2` - путь к папке, в которую следует поместить комплект, относительно *точки монтирования системного хранилища*

2. Обновите хеши и пути к комплекту приложения (appkit) в конфигурации при помощи команды:

```
# не требует подключения к кластеру, хеши будут посчитаны с комплекта приложений на␣
↪локальном диске
./appkit.sh switch_local --config-path ./config.yaml --resgroup gs-cluster-1 --
↪local-appkit workspace/appkit --remote-appkit appkit/2
```

   - `--config-path ./config.yaml` - путь к конфигурационном ресурсу
   - `--resgroup gs-cluster-1` - название группы ресурсов
   - `--local-appkit workspace/appkit` - путь к папке с комплектом на локальной ФС относительно текущей папки (`--source` из предыдущего пункта)
   - `--remote-appkit appkit/v2` - путь к папке на системном хранилище, в котором находится комплект (`--destination` из предыдущего пункта)

   ИЛИ

```
# хеши читаются с комплекта приложений на хранилище в кластере, следовательно␣
↪требует подключения к кластеру
./appkit.sh switch_remote --config-path ./config.yaml --resgroup gs-cluster-1 --
↪namespace gs-ctk --remote-appkit appkit/v2
```

   - `--config-path ./config.yaml` - путь к конфигурационном ресурсу
   - `--resgroup gs-cluster-1` - название группы ресурсов
   - `--namespace gs-ctk` - пространство имен, в котором развертан gs-ctk
   - `--remote-appkit appkit/v2` - путь к папке на системном хранилище, в котором находится комплект (`--destination` из предыдущего пункта)

3. Переключите *показатель версии базы данных* `database_schema_version`, чтобы спровоцировать обновление базы данных при следующем применении ресурса:

```
./appkit.sh upgrade --config-path ./config.yaml --resgroup gs-cluster-1
```

4. Повторно примените ресурс:

```
kubectl apply -f ./config.yaml
```

Совет

Для отслеживания статуса обновления и чтения логов, используйте команду:

```
./appkit.sh wait_upgrade --namespace gs-ctk --resgroup gs-cluster-1
```

Обновление комплекта приложения без nscli

Вот инструкция, как сделать без nscli то же, что и эти команды nscli:

```
./appkit.sh push --namespace gs-ctk-ns --source workspace/appkit --destination appkit/v1
↪# загрузка комплекта приложений на NFS (требует доступа к кластеру)
./appkit.sh switch_local --config-path ./new_config.yaml --resgroup gs-cluster-1 --local-
↪appkit workspace/appkit --remote-appkit appkit/v1 # переключение комплекта приложений
./appkit.sh upgrade --config-path ./new_config.yaml --resgroup gs-cluster-1 # обновление␣
↪счетчика версии БД
```

1. Сформировать комплект приложения в виде трех zip-архивов:
   - `globalserver.zip`
   - `applip.zip`
   - `profile.zip`

   Внимание

   Обратите внимание, что архивы содержат в себе *содержимое* соответствующих папок комплекта приложений. Так `applip.zip` содержит в себе непосредственно модули, а не папку `applib` с модулями.

2. Рядом с каждым из архивов должен лежать файл с хэш-суммой архива, полученной по алгоритму команды `sha1sum` из GNU Core Utilities. Название файла должно соответствовать названию архива и иметь суффикс `.sha1`.

   Так, например, в комплекте приложений должен находиться архив `applip.zip` и рядом с ним файл `applip.zip.sha1` с текстом `4f3b71ee067f146d68a8795b3bde1e5c0f8956d7`.

   Вы можете сгенерировать эти файлы при помощи следующего shell-скрипта:

   ```
   for archive in appkit/*.zip
   do
    sha1sum < "$archive" | tr -d ' -' | tr -d '\n' > "$archive.sha1"
   done
   ```

3. Загрузите архивы и хеш-суммы на системное NFS-хранилище.
4. Замените в конфигурации поля с хэш-суммами (`applib_sha1`, `appsrc_sha1`, `globalserver_sha1` и `profile_sha1`) и путем к комплекту (`path`) на новые. Обновите значения счетчиков `globalserver_instance` и `database_schema_version`.

Обновление кластерных утилит

1. *Загрузите и распакуйте* новую версию nscli и Helm-чарт
2. Обновите, используя *созданный ранее* `values.yaml`:

   ```
   helm upgrade gs-ctk ~/nscli/helm_chart -f values.yaml
   ```

4.8 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.

Для включения режима отладки для отдельной книги ресурсов или целой группы, добавьте в ее конфигурацию параметр `debug_mode: true`, например:

```
apiVersion: global-system.ru/v1
kind: GlobalConfiguration
metadata:
  name: config
spec:
  type: advanced
  resgroups:
  - name: gs-cluster-1
    debug_mode: true # Включаем режим отладки для группы
    resbooks:
    - type: global_server_excl
      name: global-server-excl
    - type: global_server_share
      name: global-server-share
      debug_mode: false # Выключаем режим отладки для книги
    ...
```

4.9 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-ctk.

Схема отображает типичную схему, которая может отличаться для выбранного вами Ingress-контроллера. Пользователи при подключении к кластеру распределяются балансировщиком нагрузки на нодах 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 (рекомедуем ingress-nginx), укажите при настройке значений 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.

4.10 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. Прямого 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“.

```
./configmgr.sh configure_resbook --config-path config.yaml --resgroup gs-cluster-1 --
↪resbook jgroups-dns --classname jgroups_dns
```

```
Настройка книги ресурсов: jgroups-dns (класс jgroups_dns)
============================================================

JGroups DNS не требует дополнительных настроек.

Книга ресурсов настроена!

Следующие действия:
- Список книг ресурсов можно получить при помощи команды `./configmgr.sh list_resbooks --
↪config config.yaml --resgroup gs-cluster-1`
- Удалить книгу ресурсов: `./configmgr.sh remove_resbook --config config.yaml --resgroup␣
↪gs-cluster-1 --resbook jgroups-dns`
```

```
apiVersion: global-system.ru/v1
kind: GlobalConfiguration
metadata:
  name: config
spec:
  type: advanced
  resgroups:
  - name: gs-cluster-1
    resbooks:
    - type: jgroups_dns
      name: jgroups-dns
    ...
```

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

1. Добавьте книгу ресурсов „gossiprouter".

```
./configmgr.sh configure_resbook --config config.yaml --resgroup gs-cluster-1 --
↪resbook gossiprouter --classname gossiprouter
```

```
Настройка книги ресурсов: gossiprouter (класс gossiprouter)
============================================================

Введите внешний ip(external_ip):10.10.0.1
Введите внешний порт:12001
Введите максимальный размер кучи Java (java -Xmx):800M
Введите запрос CPU для globalserver:1
Введите запрос MEMORY для globalserver:1G
Введите лимиты CPU для globalserver:1
Введите лимиты MEMORY для globalserver: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
Дополнительно отсылать метрики во внешнюю систему[да,нет]:нет

Книга ресурсов настроена!

Следующие действия:
- Список книг ресурсов можно получить при помощи команды `./configmgr.sh list_
↪resbooks --config config.yaml --resgroup gs-cluster-1`
- Удалить книгу ресурсов: `./configmgr.sh remove_resbook --config config.yaml --
↪resgroup gs-cluster-1 --resbook gossiprouter`
```

2. Измените параметр `gossiprouters` группы ресурсов:

```
./configmgr.sh configure_resgroup --config config.yaml --resgroup gs-cluster-1
```

```
Настройка группы ресурсов: gs-cluster-1
=======================================

...
Введите адрес к дополнительному GossipRouter или пропустите поле:10.10.0.1
Введите порт к дополнительному GossipRouter:12001
Добавлен GossipRouter 10.10.0.1:12001
Введите адрес к дополнительному GossipRouter или пропустите поле:cluster2.example.ru
Введите порт к дополнительному GossipRouter:12001
Добавлен GossipRouter cluster2.example.ru:12001
Введите адрес к дополнительному GossipRouter или пропустите поле:
...

Группа ресурсов настроена!

...
```

```yaml
apiVersion: global-system.ru/v1
kind: GlobalConfiguration
metadata:
  name: config
spec:
  type: advanced
  resgroups:
  - name: gs-cluster-1
    gossiprouters:
    - host: 10.10.0.1
      port: 12001
    - host: cluster2.example.ru
      port: 12001
    resbooks:
    - type: gossiprouter
      name: gossiprouter
      service:
        type: 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 наоборот 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.

4.11 Отладка прикладного решения

Предупреждение

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.

Требования

- gs-ctk версии `>=5.1.4` (проверить командой: `cat ~/nscli/version.txt`)
- Минимальные ресурсы: 12 ГБ ОЗУ, 4 ядра CPU
- Настроенная работающая группа ресурсов с включенными книгами ресурсов `global_server_share` и `haproxy`
- Наличие в комплекте приложений архива `appsrc.zip`, содержащего исходные файлы проекта

Инструкция по отладке

Запуск отладчика

1. Администратор включает отладку при помощи команды nscli:

```
~/nscli $ ./cloud_debugger.sh --namespace gs-ctk start
```

```
Укажите группу ресурсов:gs-cluster-1
Выберите книгу ресурсов, которую необходимо взять за основу облачного␣
↪отладчика:global-server-share
Укажите название для отладчика:debugger-5edafd63
Точка во времени в формате эпохи Unix, до которой будет доступен отладчик (или␣
↪укажите от текущего момента указав в качестве значения
"+N", где N - время в секундах):+3600
Облачный отладчик будет доступен до 03:14:08 19.01.2038
Ожидаем готовности облачного отладчика...
Ожидаем готовности облачного отладчика...
Ожидаем готовности облачного отладчика...
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
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%3Dba0211cf-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
```

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%3Dba0211cf-cd8a-4ba5-b2a2-a5d1791e587b%26folder%3D%2Fuser
↪%2Fapplication
```

Управление отладчиками

Для просмотра списка активных отладчиков:

```
./cloud_debugger.sh --namespace gs-ctk list
```

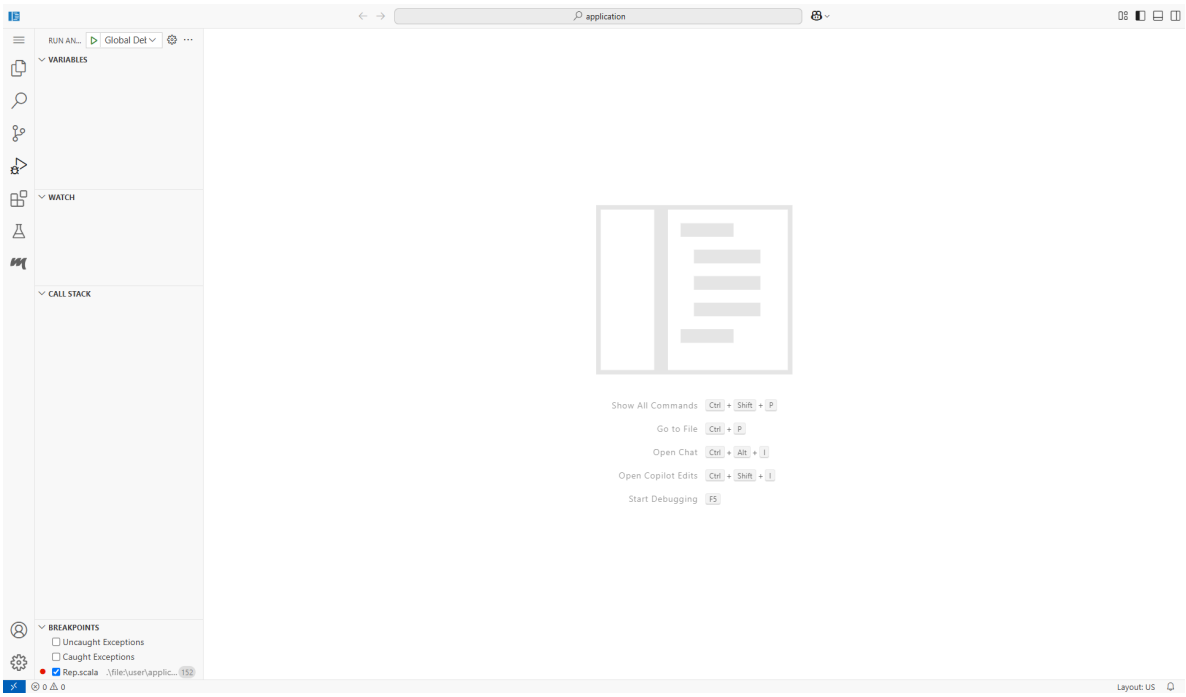
Для удаления отладчика:

```
./cloud_debugger.sh delete --namespace gs-ctk --group my-group --book debugger-mydebugger
```

Подключение к отладчику

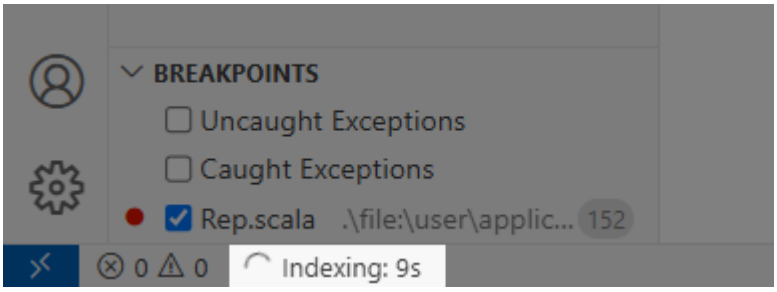
2. Администратор передает ссылки разработчикам (или другим лицам, проводящим отладку).
3. Разработчик переходит в отладчик и на сервер приложений по ссылкам.

   Важно: Используйте HTTPS для подключения. При проблемах с аутентификацией (циклическая переадресация) очистите cookies браузера.

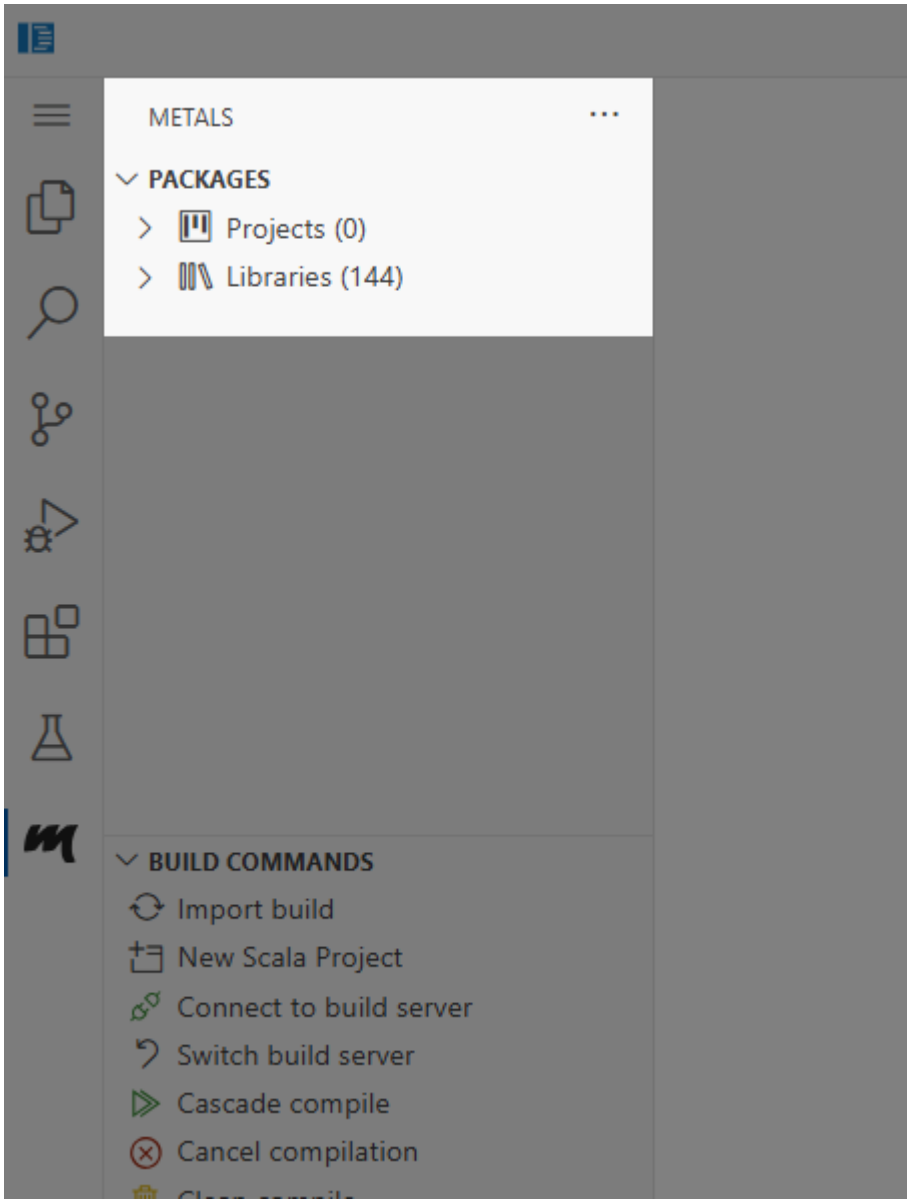


4. Разработчик подтверждает, что он находится в доверенном каталоге `/user/application`, и переходит во вкладку Metals на боковой панели.

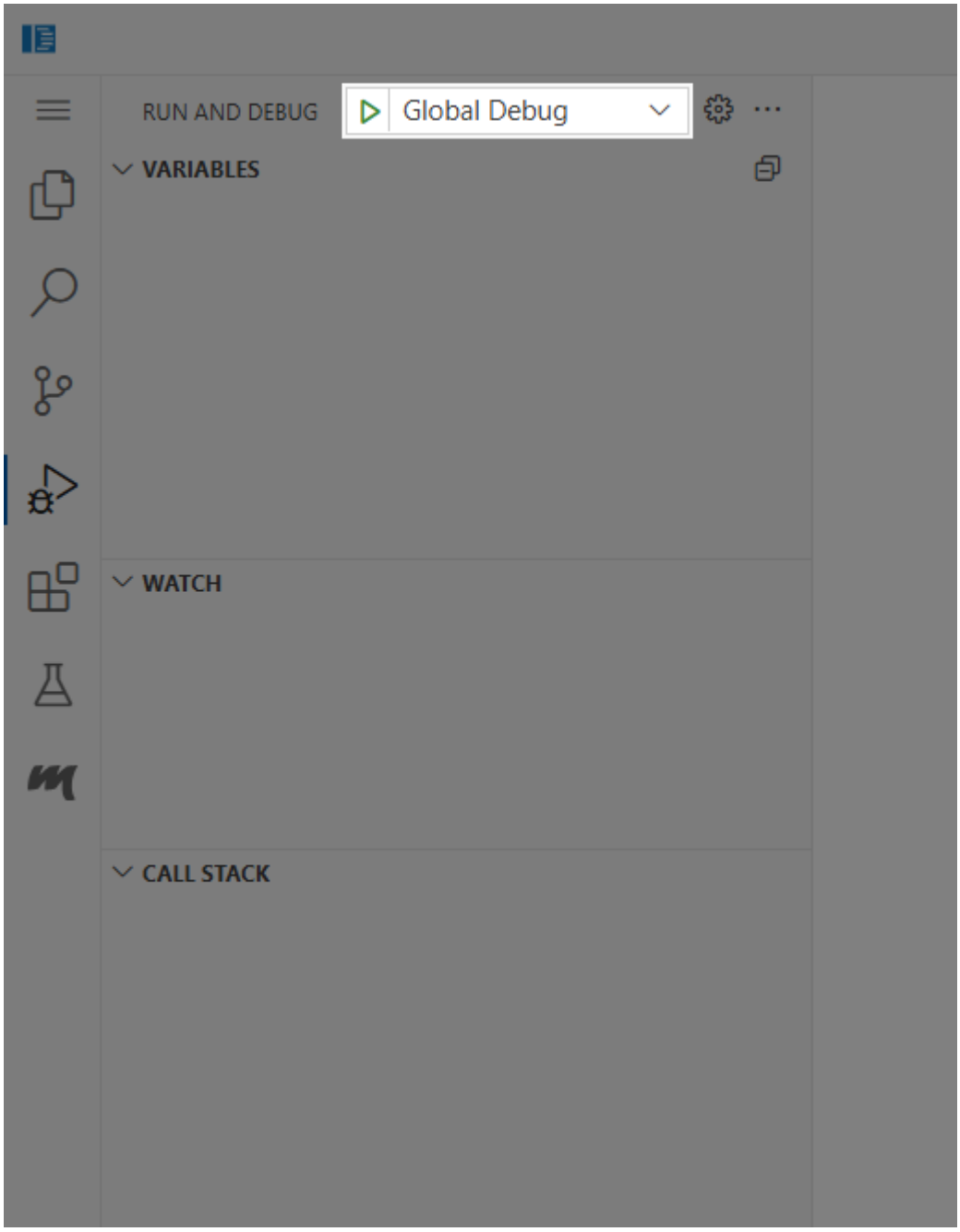
5. В течении 30 секунд должна начаться индексация проекта. Дождитесь завершения. После этого в верхней части вкладки Metals появится список используемых модулей и библиотек.



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 остановы, 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`.

Устранение проблем

Проблема с аутентификацией

Если после успешной авторизации происходит переадресация на страницу входа:

- Очистите cookies браузера для домена отладчика
- Используйте для подключения только протокол HTTPS

Безопасность

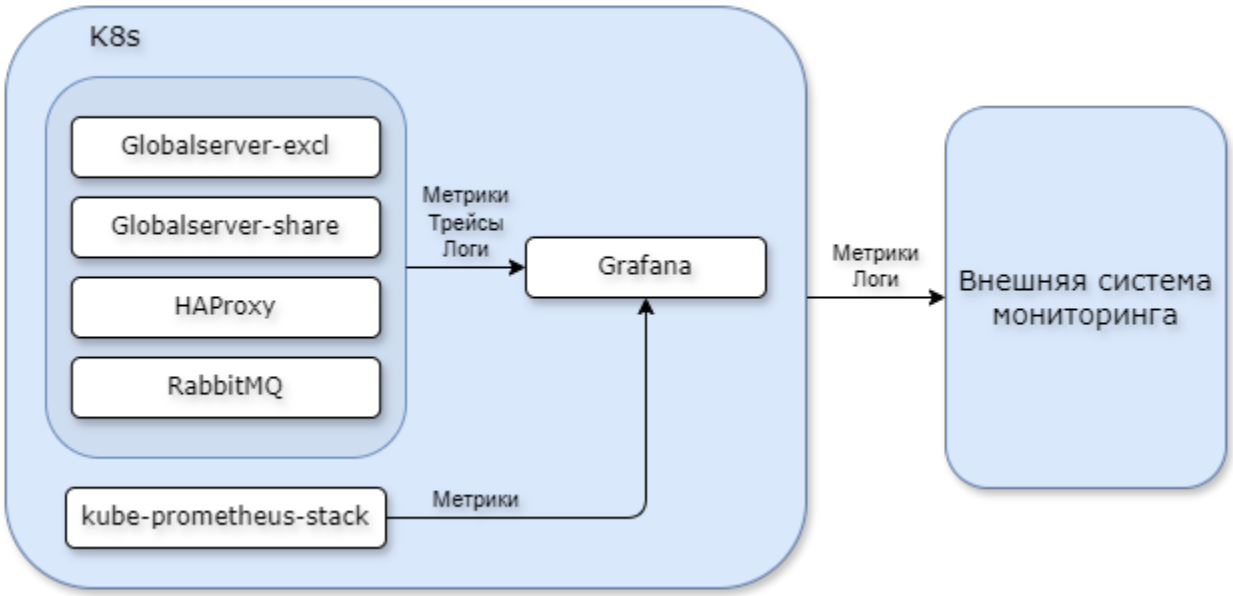
В ссылку для отладчика вписан токен для подключения, который используется для авторизации в OpenVSCode Server. Следовательно, без этой ссылки подключиться к отладчику невозможно.

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.

4.12 Мониторинг кластера Global ERP

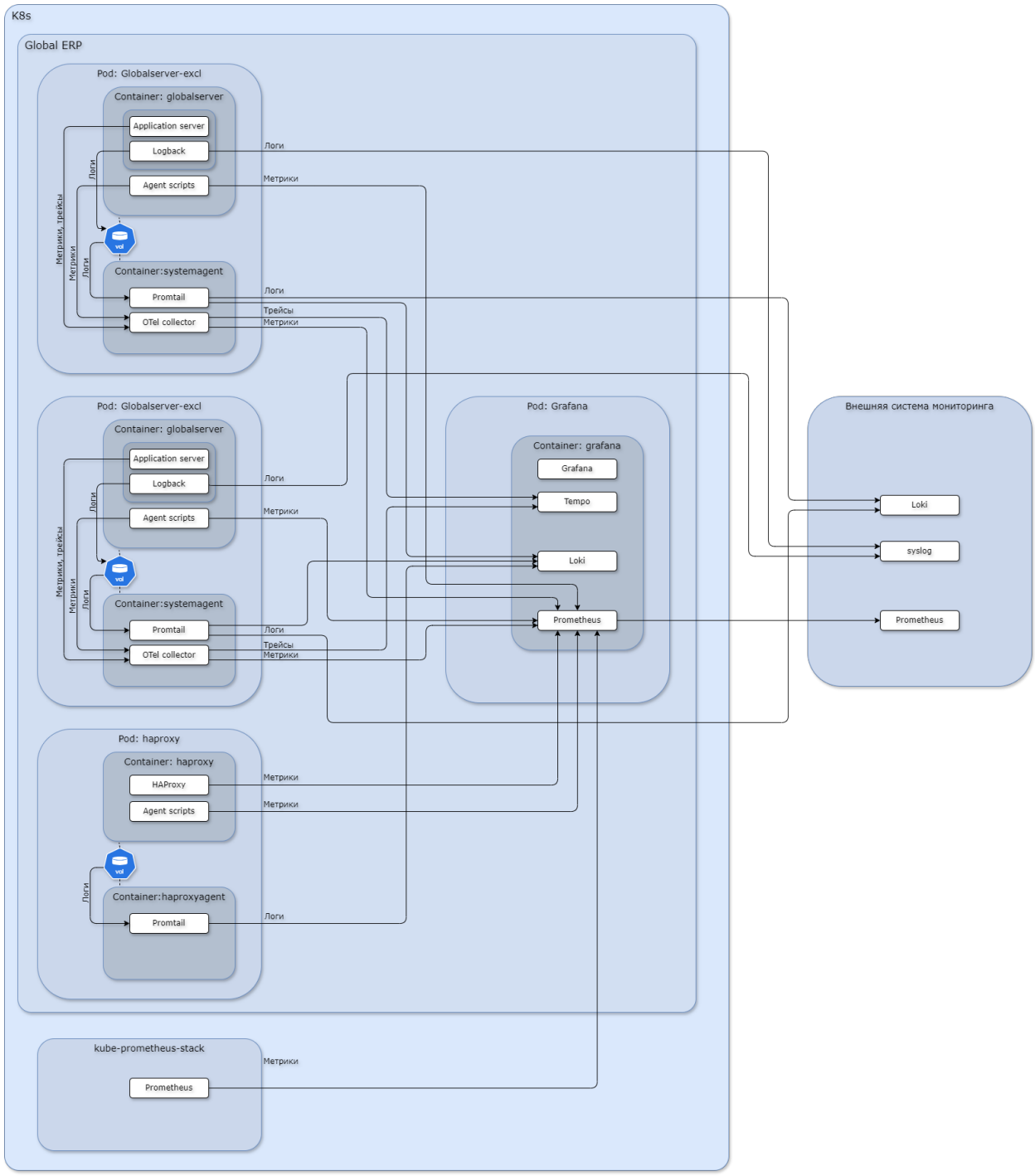


Кластер Global ERP собирает следующую телеметрию:

- Метрики с использованием стека kube-prometheus-stack + Opentelemetry Collector + Prometheus.
- Трейсы пользовательских операций с использованием стека Opentelemetry Collector + Tempo.
- Логи с использованием стека Promtail + Loki.

Вся телеметрия собирается во внутренние инстансы Prometheus, Loki и Tempo, которые запущены в поде grafana. Внутренняя Grafana используется для оперативного мониторинга системы, так как данные удаляются при перезапуске пода.

Для долгосрочного хранения телеметрии требуется отдельная внешняя система мониторинга.



Логи

Логи планировщика заданий, серверов Global и HAProxy обрабатываются агентом Promtail и отсылаются в базу Loki.

Также возможно настроить отправку логов во внешний инстанс Loki или отправку логов напрямую из Logback Globalserver-a.

Внешний инстанс Loki

Для отправки логов во внешний инстанс Loki при *конфигурировании* групп ресурсов и книг ресурсов кластера Global в поле «Дополнительно отсылать метрики во внешнюю систему» требуется выставить «Да» и задать корректный адрес внешнего инстанса 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

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

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        period: 24h

ruler:
  alertmanager_url: http://localhost:9093

analytics:
  reporting_enabled: false
```

Отправка логов из Logback

Для отправки логов во внешнюю систему напрямую из Logback Globalserver-а в профиль аппкита по пути `profile\globalserver\template\config` требуется добавить дополнительные файлы конфигурации:

- `logback-LoggerContext-ext.xml` - дополнительная конфигурация логгера «system».
- `logback-LoggerContext-session-ext.xml` - дополнительная конфигурация логгеров «app» и «session».

Метрики

Телеметрия Globalserver-а через Opentelemetry Collector отправляется во внутренний Prometheus.

Системная телеметрия кластера k8s собирается с помощью kube-prometheus-stack (требует отдельной установки).

Также возможно настроить внешний инстанс Prometheus на получение метрик из внутреннего инстанса.

kube-prometheus-stack в собственном кластере k8s

При разворачивании кластера Global ERP в собственном кластере k8s требуется требуется установить kube-prometheus-stack по официальной документации

kube-prometheus-stack в VK Cloud

При разворачивании кластера Global ERP в VK Cloud требуется установить аддон «kube-prometheus-stack» из панели управления кластером. В конфигурации аддона для компонента prometheus-node-exporter требуется добавить блок «extraArgs» для получения телеметрии ФС нод кластера.

```
prometheus-node-exporter:
  image:
    repository: "prometheus/node-exporter"
    tag: v1.7.0
  namespaceOverride: "monitoring"
  resources:
    limits:
      cpu: 200m
      memory: 50Mi
    requests:
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
      cpu: 100m
      memory: 30Mi
  extraArgs:
    - --collector.filesystem
    - --collector.filesystem.mount-points-exclude=^/(dev|proc|sys|var/lib/docker/.+|var/
↪lib/kubelet/.+)($|/)
    - --collector.filesystem.fs-types-exclude=^(autofs|binfmt_misc|bpf|cgroup2?
↪|configfs|debugfs|devpts|devtmpfs|fusectl|hugetlbfs|iso9660|mqueue|nsfs|overlay|proc|procfs|pstore|rp
↪pipefs|securityfs|selinuxfs|squashfs|sysfs|tracefs)$
```

Настройка внешнего Prometheus

Prometheus устанавливается по официальной документации. Для получения метрик сервера Global Prometheus настраивается на сбор метрик с внутреннего инстанса через федерацию.

Пример config.yaml

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

scrape_configs:
  - job_name: 'federate'
    scrape_interval: 15s

    honor_labels: true
    metrics_path: '/federate'

    params:
      'match[]':
        - '{__name__=~"^[^go_].+"}'

    static_configs:
      - targets:
        - '<prometheus_external_ip>:9090'
```

Настройка внешнего инстанса Grafana

Grafana устанавливается и настраивается по официальной документации.

4.13 Развертывание с использованием gs-ctk 4

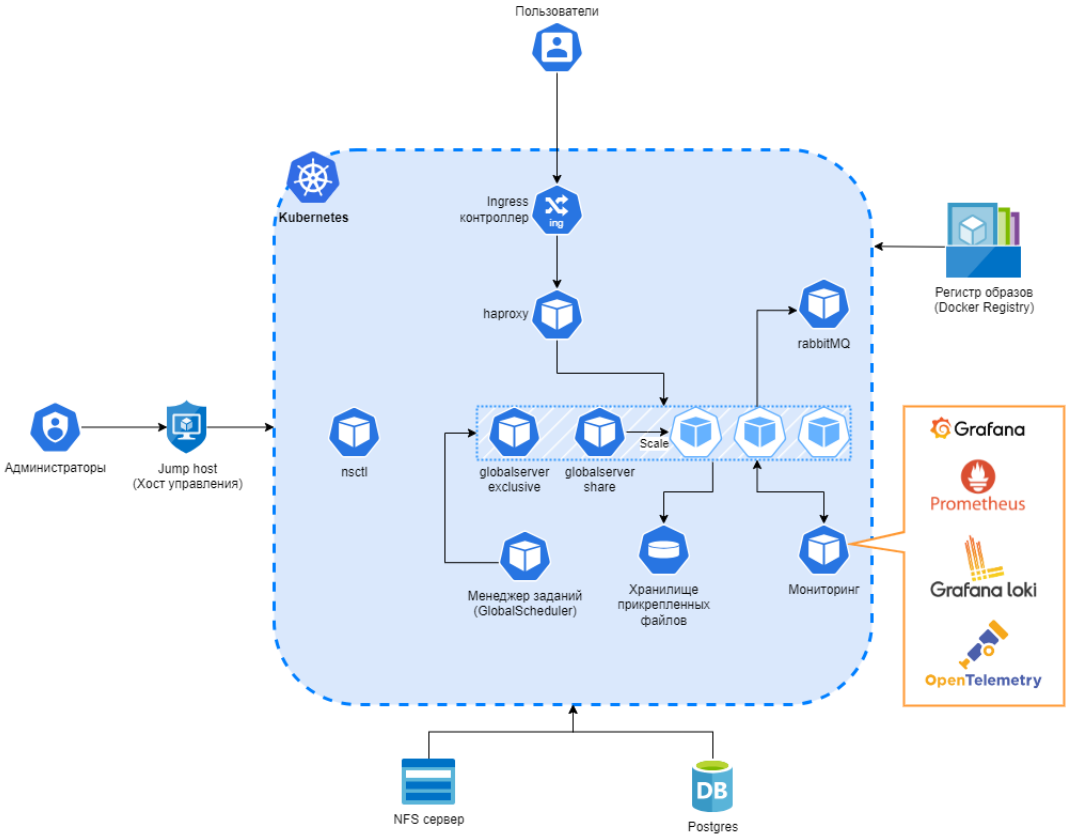
Описание

Предупреждение

На тестовых и предварительных контурах следует подумать об использовании *gs-ctk версии 5.0*. Эта версия проще в развертывании и обслуживании, благодаря использованию Helm и конфигурационных ресурсов, а использование мастеров конфигурации и интерактивных команд необязательно.

На продуктивных средах пока стоит использовать 4.9.0 - она испытана в боевых условиях.

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.

Установка

Предупреждение

На тестовых и предварительных контурах следует подумать об использовании *gs-ctk версии 5.0*. Эта версия проще в развертывании и обслуживании, благодаря использованию Helm и конфигурационных ресурсов, а использование мастеров конфигурации и интерактивных команд необязательно.

На продуктивных средах пока стоит использовать 4.9.0 - она испытана в боевых условиях.

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

Совет

Версия kubectl должна соответствовать версии в платформе kubernetes или быть новее

Настройка работы с docker образами

Если требуется работа с образами из docker регистра

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

Insecure Docker Registry

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

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

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

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

скрипт для создания файла:

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

После чего выполнить перезапуск сервиса с помощью:

```
sudo systemctl restart docker
```

Настройка авторизации kubernetes кластера

Получите конфигурационный файл авторизации для кластера Kubernetes. 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:␣
↪LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJUkMwZFZrYXhBTE13RFFZSktvWklodmNOQVFFT
    server: https://127.0.0.1:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data:␣
↪LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURLVENDQWhHZ0F3SUJBZ0lJR04ydFhtNkEzc2N3RFFZSktvWklodmNOQVFFT
    client-key-data:␣
↪LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBclA1TXlQWlo3YlpoRmo1SldtSGkzT29MSXpWd
```

Настройте авторизацию kubernetes

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

# сохраняем файл
cat <<EOF | tee ~/.kube/config
apiVersion: v1
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:␣
↪LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJUkMwZFZrYXhBTE13RFFZSktvWklodmNOQVFFT
    server: https://127.0.0.1:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data:␣
↪LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURLVENDQWhHZ0F3SUJBZ0lJR04ydFhtNkEzc2N3RFFZSktvWklodmNOQVFFT
    client-key-data:␣
↪LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBclA1TXlQWlo3YlpoRmo1SldtSGkzT29MSXpWd
EOF
```

Проверяем доступ

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

При успешном подключении должны получить вывод:

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

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

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/4.9.0/gs-ctk-nscli-4.9.0.zip"
unzip -o gs-ctk-nscli-4.9.0.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 ~/nscli
```

Запустите мастер создания манифеста рабочего пространства

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

Примечание

В более современных версиях nscli 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 nsctl (читайте подробнее в документации gs-ctk).

Пример работы мастера создания пространства имен:

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

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 распакованном 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/
```

Подготовте и загрузите в этот каталог дистрибутивы.

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

Подготовте 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
          values:
          - k8s-worker01
```

```
cat <<EOF | tee ~/nscli/workspace/local-pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-pv-50
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-storage
  local:
    path: /mnt/disks/disk1
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - k8s-worker01
EOF
```

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

Создание секретов

Секреты используются для безопасного хранения учетных данных

- Создайте секрет для доступа к статистике haproxy

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

```
cat <<EOF | tee ~/nscli/workspace/haproxy-sercret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: secret-haproxy-auth
  namespace: gs-cluster-k8s
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: t0p-Secret
EOF
```

```
kubectl apply -f ~/nscli/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 ~/nscli/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)
  tls.key: |
    $(cat ./ca.key | base64 -w 0)
EOF
```

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

- Создайте секрет с admin аккаунтом globalserver-a

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

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

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

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

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

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

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

- Создайте секрет с 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 ~/nscli/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 ~/nscli/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 ~/nscli/workspace/rabbitmq-admin.yaml
apiVersion: v1
kind: Secret
metadata:
  name: rabbitmq-admin
  namespace: gs-cluster-k8s
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: Secret#123#Pass!
EOF
```

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

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

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

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

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

- Создайте роль

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

```
cat <<EOF | tee ~/nscli/workspace/cadvisor-role.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus-cadvisor
rules:
- apiGroups: [""]
  resources:
  - nodes
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
  - nodes/proxy
  - services
  - endpoints
  - pods
  verbs: ["get", "list", "watch"]
- apiGroups:
  - extensions
  resources:
  - ingresses
  verbs: ["get", "list", "watch"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]
EOF
```

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

- Создайте биндинг роли

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

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

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

Настройка параметров кластера

Параметры кластера настраиваются с помощью пода `nsctl`

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

Получите список подов рабочего простнанства

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

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

Получите доступ по ssh в под `nsctl`

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

Совет

В более новых версиях nscli 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
./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
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
Хотить 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
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
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
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 использвать 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

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 *комплектом 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:~/nscli$ ./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 неуправлемом 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
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
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
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
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
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
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 `~/nscli/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 ресусров:

```
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 `(~/nscli/)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 утилитах, *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
         ...
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    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 неуправлемом 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
```

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`.

Администрирование системы

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

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

Внимание

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

Запуск и остановка комплекта приложений

Команды запуска и остановки комплекта приложений останавливают сервера приложений в подах

Остановка комплекта

Остановить комплект приложений

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

, где

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

Запуск комплекта

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

, где

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

Обновление комплекта группы

На Jump хосте подготовить элементы комплекта группы(наличие всех компонентов не обязательно):

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

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
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

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

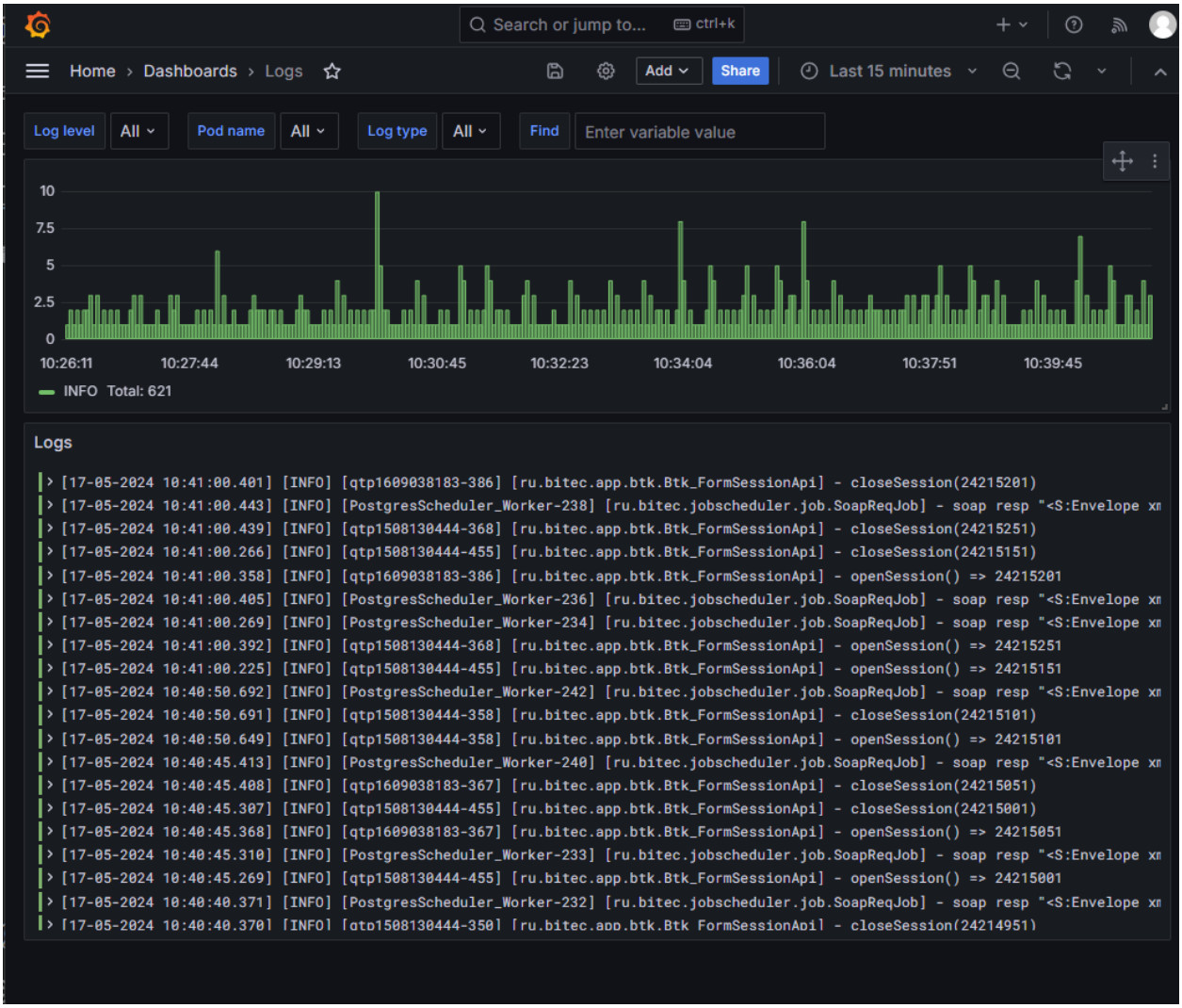
Логи кластера

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



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"
    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
```

Экспорт конфигурации. Восстановление кластера

Скрипт `configmgr.sh` в составе nscli 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 nsctl, 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 ~/nscli
./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 Nscli»* включительно) выглядит следующим образом:

1. Воссоздаем пространство имен при помощи команд:

   ```
   ./namespace.sh create_install_scripts
   chmod +x ~/nscli/workspace/install_scripts/gs-k8s/install.sh
   ~/nscli/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 ~/nscli
   ./configmgr.sh import \
     --namespace gs-k8s \
     --file config.yaml
   ```

   Скрипт сообщит об успешном завершении:

   ```
   Конфигурация импортирована
   ```

5. Запускаем кластер:

   ```
   cd ~/nscli
   ./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"
   ```

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-ctk.

Схема отображает типичную схему, которая может отличаться для выбранного вами Ingress-контроллера. Пользователи при подключении к кластеру распределяются балансировщиком нагрузки на нодах 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.

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. Прямого 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 неполномочного 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
```

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_and_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.

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
```

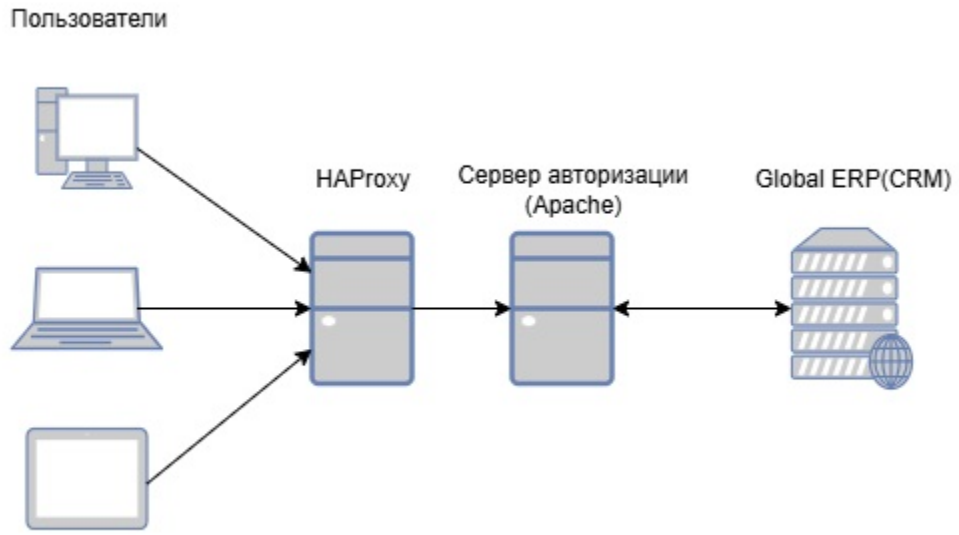
3. Включим книгу ресурсов обратно:

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

5 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) Для корректной работы сервера авторизации на сервере приложением необходим модуль `srm`.

5.1 Схема работы сервера авторизации



Описание элементов:

- Пользователи — компьютеры, смартфоны, планшеты.
- HAProxy — принимает входящие запросы и перенаправляет их либо на сервер авторизации, либо на Global ERP.
- Сервер авторизации:
  - Django + Apache.
  - Принимает запросы от пользователя и отправляет их на Global ERP.
  - Если успешно, генерирует JWT токен.
  - Перенаправление — HAProxy направляет пользователя с токеном на Global ERP.
- Global ERP(CRM):
  - Проверяет JWT токен.
  - Дает доступ к функционалу.

5.2 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;
```

5.3 Развертывание и конфигурация

Скачайте архив дистрибутива с предоставленного ресурса (Предоставляется через контактное лицо, файл gs-authproxy.zip). Создайте временную директорию и загрузите файлы дистрибутива на сервер

```
sudo mkdir -p /tmp/gs-authproxy
```

Распакуйте сервер авторизации

```
sudo mkdir -p ~/gs-authproxy
sudo unzip /tmp/gs-authproxy/gs-authproxy.zip -d ~/gs-authproxy
```

Выдайте разрешение на запуск

```
sudo chmod +x ~/gs-authproxy/set_credentials.sh
sudo chmod +x ~/gs-authproxy/update.sh
sudo chmod +x bin/*
```

Установите нужные серверу авторизации пакеты и настройте окружение:

```
sudo ./bin/installpkg.sh
./bin/initvenv.sh
```

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»:

- Для ключа `jSettingForGenGidUrl`,укажите следующие значения:
  - `"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.
- Для ключа `extUserPublicKey` добавьте публичный ключ подписи в текстовом виде без заголовка и подвала. (Ключ должен быть в одну строку без переносов)

```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAnY1eq7C1PMhnXdvrlM5EcC6B4VgkLheotvPIiLf5vV2ZS+VPDhc2ZyC
↪i4zoFcs8qUwHfCwsTRmdl828YRlzf0rHA/
↪jiT21J1AzkkgMSQmIH9XRxlSS9HmkP6Hgvx7Fe+ir86kU6Pw5OLaCeBnlh0RUrolSnyLNhjPuCqIQ54pfz8YzR/
↪T2jMgxU+hvVjD2vC8OJLNDWv7g0rNzynV4zIWNt6gkiHzkvwIDAQAB
```

Настройка gs-authproxy

- Создайте директорию `/opt/global/gs-ap-config` для конфигурации проекта.

```
sudo mkdir -p /opt/global/gs-ap-config
```

Добавьте необходимые конфигурационные файлы (файл с конфигурацией проекта, иконка для вкладки в браузере, логотип на странице входа).

Структура каталога должны быть следующей:

```
gs-ap-config/
├── static/
│   ├── img/
│   │   ├── logos/
│   │   │   └── customer.svg - логотип на странице входа
│   │   └── icons/
│   │       └── favicon.svg - иконка для вкладки в браузере
└── config.yml - файл с конфигурацией проекта
```

Важно

Система рассчитана на работу только с такой структурой директории Вы можете скопировать структуру из deploy/templates/gs-ap-config

- Сконфигурируйте проект
  В созданной директории заполните файл с конфигурациями проекта `config.yml` в соответствии с требования вашего проекта:
  - подключения к БД установленной в прошлых шагах
  - почтовый сервер
  - настройки Django

        * пути приватных ключей
        * имена для генерации токена
        * url на который будет отправляться запросы
        * путь к файлу с логами
        * сведения об организации
        * время жизни кода подтверждения
        * путь для доступа к cookie
        * включение тихого режима (True/False)
        * информация о компании
    - Задайте переменные окружения с названием `APPROFILEPATH`, где укажите путь к директории с конфигурацией

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

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:
# количество отправляемых писем за лимит времени
  email_limit: 60
# лимит времени по отправки писем
  limit_window: 60
# 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␣
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪(Transport Layer Security)
  email_port: 587
# булевое значение, указывающее, следует ли использовать TLS (Transport Layer Security)␣
↪для защищенного соединения с SMTP-сервером. Установите True, если ваш SMTP-сервер␣
↪поддерживает TLS, и False в противном случае
  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
  # 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
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
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 серевер авторизации пересылает по статичной ссылке).

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````

Затем инициализируйте БД и соберите статические файлы. Перейдите в директорию gs-authproxy и активируйте виртуальное окружение, затем выполните команды.

```
cd mail_gate_pass
python manage.py migrate
python manage.py collectstatic
```

Настройка Apache2

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

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

Пример заполненного Apache2

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

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

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

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    </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
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    # 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
```

5.4 Обновление

Подготовка

Скачайте релиз и поместите его в рабочий каталог проекта:

`gs-authproxy/workspace/`

Запуск обновления

Для выполнения обновления запустите скрипт:

```
./update.sh
```

Скрипт выполнит все необходимые операции по обновлению файлов.

Резервное копирование

Во время обновления создаётся резервная копия текущей версии проекта в директории:

`../backups/` (на уровень выше каталога gs-authproxy)

При необходимости вы можете восстановить предыдущую версию из этой резервной копии вручную. Для этого перейдите в каталог с резервными копиями и распакуйте необходимую вам:

```
cd ~/backups/
tar -xzf backup_20250725_142846.tar.gz -C ./extracted_files
```

Перед извлечением убедитесь что каталог extracted_files существует.

6 Процедура обновления Global ERP

6.1 1. Общие положения

Процедура описывает порядок, состав, сроки и условия выполнения обновлений программного комплекса Global ERP — как базовой платформы, так и прикладных модулей, поставляемых в рамках проекта внедрения. Процедура утверждена вендором и применяется на всех проектах, где используется Global ERP.

6.2 2. Состав обновления

Обновление может включать следующие компоненты:

- Образы Kubernetes — контейнеры инфраструктурных компонентов;
- Сервер приложений — обновление версии платформы и сервисных модулей;
- Образ прикладного решения — сборка новой версии проектного кода и расширений;
- Релизы конфигурации — поставочные данные и настройки.

Каждый компонент может обновляться независимо или в составе комплексного релиза.

6.3 3. Подход к обновлению

Обновление выполняется по многоступенчатой модели с обязательным прохождением промежуточных сред. Процедура включает следующие этапы:

1. DEV (разработка) – сборка коробочного решения с проектными расширениями и выполнение простейших функциональных тестов;
2. TEST / QA – установка версии на тестовый стенд, проведение регрессионных и интеграционных тестов (автоматизированных и ручных);
3. PRE-PROD / UAT – окончательное тестирование обновлённого решения с учётом проектных расширений на продуктивных данных;
4. PROD – установка в промышленный контур после утверждения заказчиком.

Все обновления проходят предварительную проверку на совместимость версий компонентов (ядро, СУБД, окружение).

6.4 4. Условия и режимы обновления

Поддерживаются два режима обновления:

- Стандартный (Stop-and-Upgrade) — выполняется при отсутствии активных сессий, с монопольным доступом к базе данных;
- Мягкий (Rolling Update) — используется для минимизации простоя системы. Поддерживает перекрытие версий, автоматическое завершение блокирующих сессий и обновление контейнеров без полной остановки кластера.

Обновления базы данных выполняются по принципу **expand/contract** (сначала добавление новых структур, затем удаление устаревших). Файловые ресурсы и контейнеры обновляются средствами CI/CD и Kubernetes.

6.5 5. Периодичность и сроки проведения

- Релизы платформы выпускаются **ежемесячно**;
- Обновления контейнерных образов — **раз в полгода**;
- Прикладные решения обновляются по мере готовности модулей (в среднем раз в квартал).

Сроки внедрения конкретного обновления на стороне заказчика согласовываются индивидуально и фиксируются в плане релизов.

6.6 6. Ответственность и контроль качества

- Обновление выполняется **обученными специалистами заказчика**, а также при необходимости техническими специалистами вендора или сертифицированного партнёра по платформе Global ERP;
- Все действия фиксируются в журналах обновлений;
- Перед установкой каждой версии выполняется резервное копирование базы данных и конфигурации;
- При необходимости возможен откат к предыдущей версии.

6.7 7. Документирование и сопровождение

По каждому релизу формируется комплект сопроводительной документации:

- Release Notes — описание изменений, новых функций и исправлений ошибок;
- Инструкция по установке и обновлению;
- Инструкция по откату (Rollback).

Документация публикуется на внутреннем портале и сопровождается версиями в системе контроля версий (GitLab).

6.8 8. Принятие обновлений

После установки выполняется приёмочное тестирование на стороне заказчика. По итогам подписывается акт приёмки обновления или формируется список замечаний, подлежащих исправлению в ближайшем релизе.

6.9 9. Сроки и порядок проведения обновлений

Длительность и график проведения обновлений зависят от состава релиза и объёма данных:

- Платформенные обновления (сервер приложений, контейнеры, конфигурация) выполняются, как правило, не более 20 минут, некоторые не требуют остановки работы пользователей;
- Обновление образа прикладного решения выполняется в среднем за 4 минуты, включая копирование данных по сети и скрипты обновления базы данных.
- Обновления базы данных (структурные миграции, преобразование данных) занимают до двух минут, долгие скрипты миграции выполняются в двухфазном режиме:
  - Обновление схемы, решение поддерживает работу и старой и новой схемы, фоновая работа скриптов миграции данных без остановки работы пользователей.
  - После завершения скриптов миграции обновление новым решением с окончательным переходом на новую структуру данных.

Обновления планируются заранее и согласовываются с ответственными представителями заказчика. За 10 рабочих дней до установки направляется уведомление с перечнем изменений, планом обновления и оценкой влияния на систему.

Проведение обновлений осуществляется в регламентное окно обслуживания (в вечернее или ночное время) с учётом требований заказчика к доступности системы. При использовании режима Rolling Update обновления могут выполняться без полной остановки сервисов.

7 Дополнительно

7.1 Настройка SSO

Keycloak

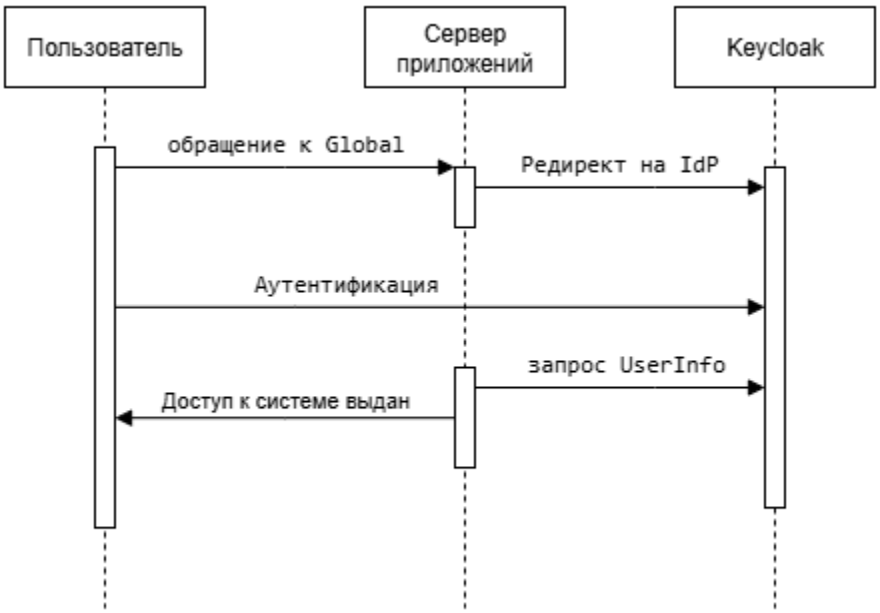
Введение

Эта инструкция описывает настройку единого входа (SSO) по **OpenID Connect (OIDC)** между Keycloak и сервером приложений **Global Server**.

Мы создадим Realm и клиента в Keycloak, подключим федерацию пользователей (например, LDAP/AD), настроим защиту от брутфорса и (опционально) второй фактор аутентификации по e-mail, а затем соединим Keycloak и Global Server через `global3.config.xml`.

Global Server поддерживает SSO начиная с версии `1.25.0-rc33`

Общая схема работы



Предварительные требования

- Развёрнутый Keycloak (версии 21+)
- Доступ администратора к Keycloak и Global AS
- SMTP-сервер для отправки почты (для 2FA по e-mail)

Часть 1. Настройка Keycloak

Создание Realm

1. Зайдите в админ-консоль Keycloak → **Create realm**
2. Укажите имя (например, `dev-test`)
3. Запомните URL realm: `https://<keycloak-host>/realms/dev-test`

Настройка федерации пользователей (LDAP/AD)

1. Realm → **User federation** → **Add provider: ldap**
2. Заполните поля:
   - Connection URL
   - Bind DN / Bind credentials
   - Users DN (Base DN)
   - Use StartTLS/SSL — по необходимости
3. Во вкладке **Mappers** настройте сопоставление атрибутов (`username`, `email`, `firstName`, `lastName`). При использовании нескольких доменов настройте маппинг имени пользователя на **userPrincipalName**
4. Сохраните и выполните импорт пользователей

Создание клиента (OIDC) для Global Server

1. Realm → **Clients** → **Create**
2. Заполните:
   - **Client Type**: OpenID Connect
   - **Client ID**: `global-erp`
   - **Client authentication**: ON (конфиденциальный клиент)
3. **Valid Redirect URIs** — `https://{global.server}/*` (позже уточнить)
4. **Scopes** — `openid email profile`
5. Сохраните и сгенерируйте **Client Secret**

Включение защиты от брутфорса

1. Realm → **Realm Settings** → **Security Defenses** → **Brute Force Detection**
2. Включите и задайте параметры:
   - Max login failures
   - Wait increment
   - Quick login check и т.д.

(Опционально) Настройка 2FA с кодом на e-mail

1. Realm → **Realm Settings** → **Email** — настройте SMTP
2. Установите плагин (например `keycloak-email-otp`)
3. Realm → **Authentication** → **Flows**
4. Скопируйте поток «Browser» и добавьте шаг `Email OTP` после `Username Password Form` → Required

Часть 2. Настройка Global AS (global3.config.xml)

Добавьте новый блок `<openId>` в конфигурацию:

```
<security>
  <authenticators>
    <openId name="dev-test" default="true" autoLogin="false">
      <provider url="https://<keycloak-host>/realms/dev-test"/>
      <client id="global-as"
              secret="*** CLIENT_SECRET ***"
              authMethod="post"
              scopes="openid,profile,email">
      </client>
    </openId>
  </authenticators>
</security>
```

7.2 Синхронизация пользователей (LDAP)

Для настройки синхронизации пользователей нужно в приложении `Настройка системы` перейти в `Общие настройки модулей`.

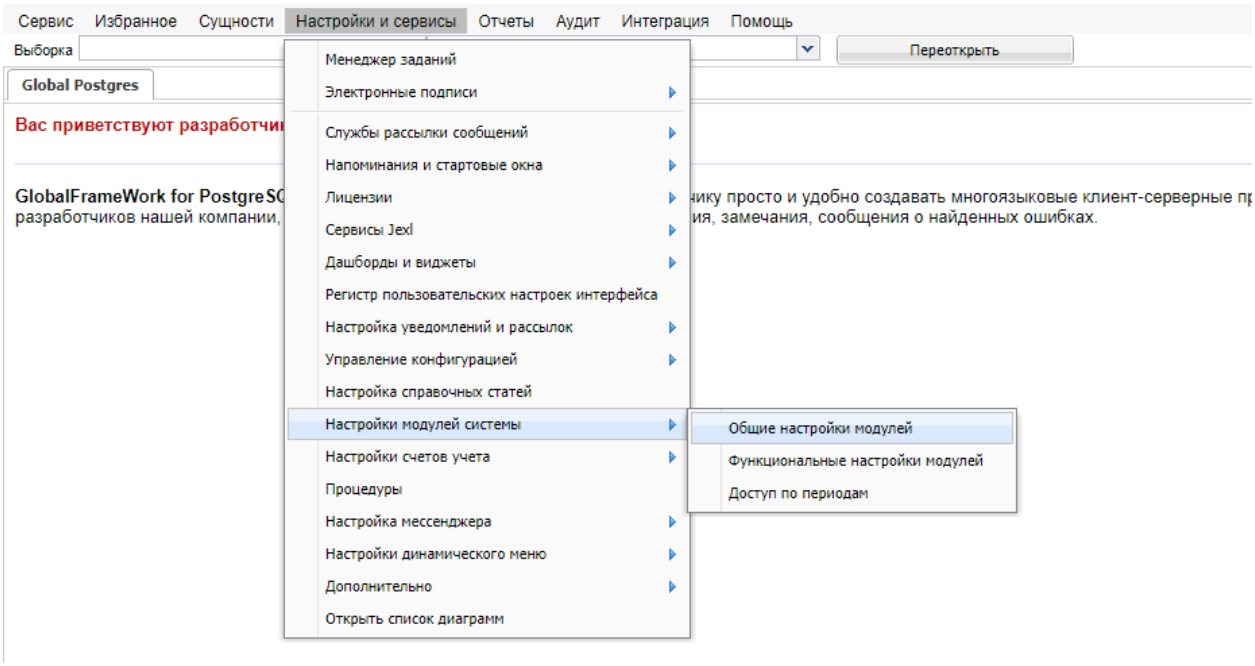


Рисунок 1. Расположение общих настроек модулей

Зайти в настройки модуля Btk и открыть настройку `LDAP-синхронизация`.

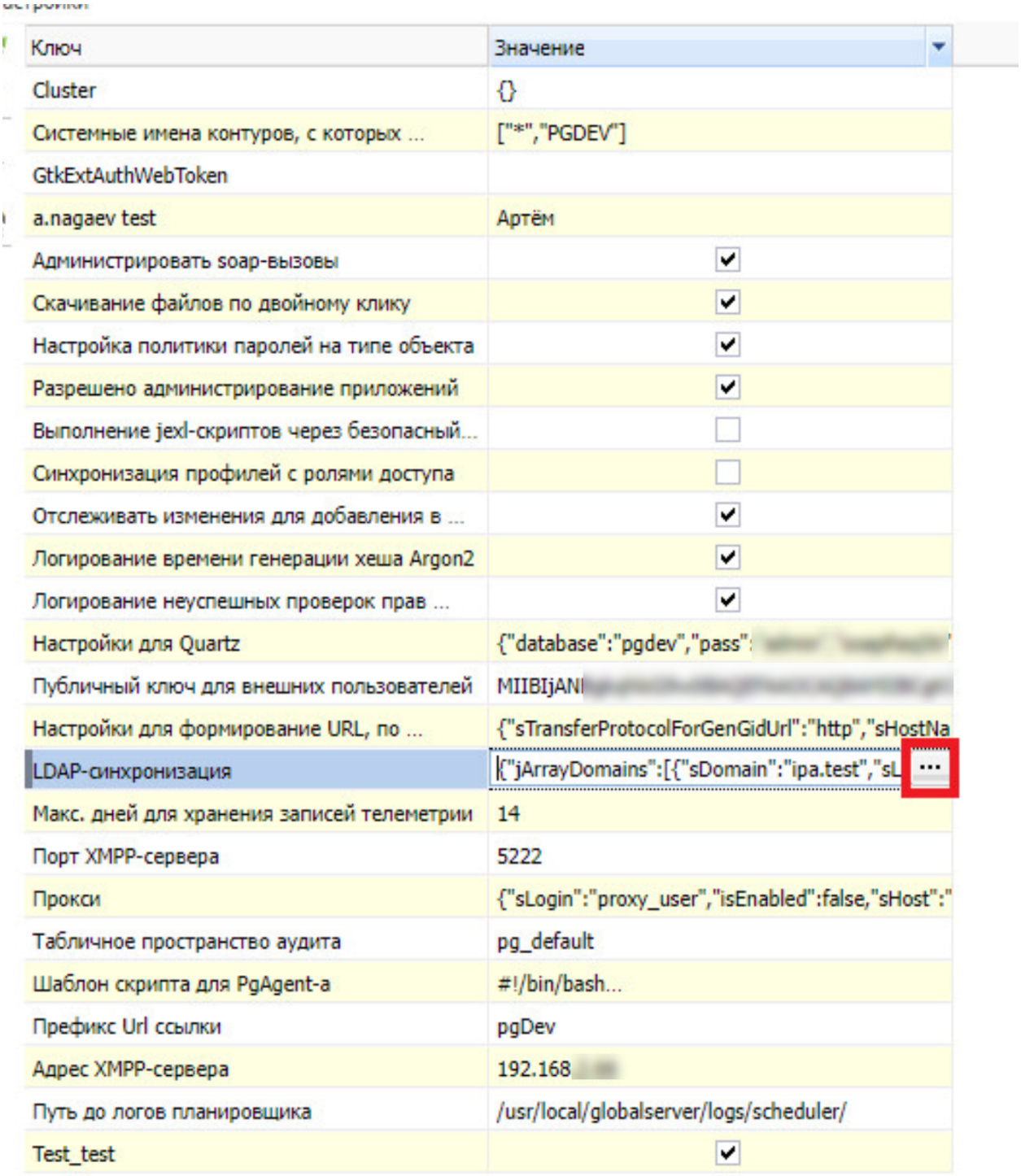


Рисунок 2. Настройки модуля Btk

Пример настройки:

LDAP-синхронизация:

```
{"jArrayDomains":[{"sDomain":"local.local","sLogin":"admin@local.local","sUrl":"ldap://
↪<IP_ADDRESS>:<PORT>","sPass":"EncryptedStorage:<PASS>"}],"sGroupFilter":null,
↪"idSyncType":2}
```

Настройки для Quartz:

```
{"database":"<DATABASE>","pass":"<PASSWORD>","soapReqStr":"/app/sys/soap/sys-service-1.0.
↪0","publicKey":"<PUB_KEY>","login":"<LOGIN>","soapPort":80,"restAcquireJobReqStr":"/
↪app/sys/rest/ss/pkg/Btk_QuartzRestPkg/getactivejobs","soapHost":"pg.local"}
```

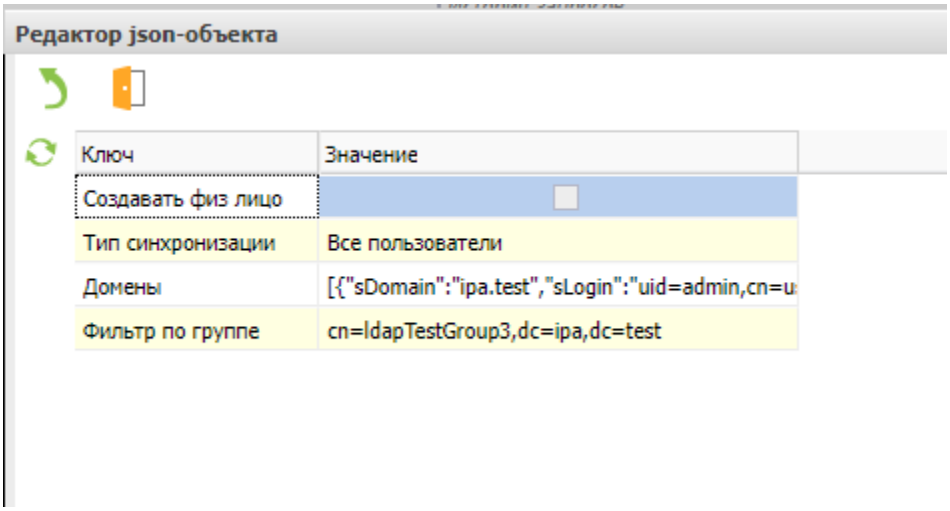


Рисунок 3. Настройки Ldap-синхронизации

Пример настройки:

Домены:

```
[{"sDomain":"local.local","sLogin":"admin@local.local","sUrl":"ldap://<IP_ADDRESS>:<PORT>
↪","sPass":"EncryptedStorage:<PASS>"}]
```

Таблица 1. Настройки Ldap-синхронизации

Наименование	Описание
Создавать физ лицо	При включенной настройке создаётся физ лицо в случае, если оно не было найдено по совпадению ФИО с пользователем
Тип синхронизации	`Пользователи группы`: Переносит в Global все подгруппы заданной группы из указанных доменов и входящих в них пользователей с сохранением изначальной структуры. Необходимо заполнить настройку «Фильтр по группе».`Все пользователи`: Переносит в Global все группы и всех пользователей из заданных доменов с сохранением структуры, если это возможно.
Фильтр по группе	Полное название группы, члены которой должны быть синхронизированы. Используется только в случае, когда выбран тип синхронизации «Пользователи группы»
Домены	Список доменов, в которых будет производиться поиск групп и пользователей

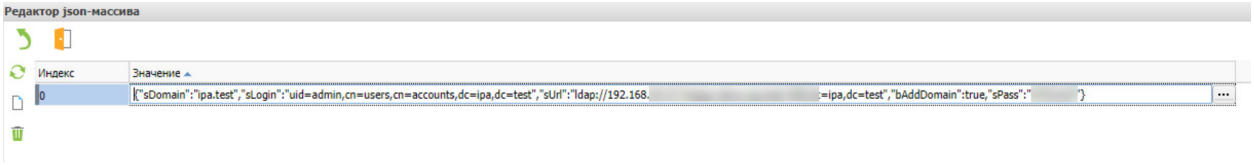


Рисунок 4. Список доменов

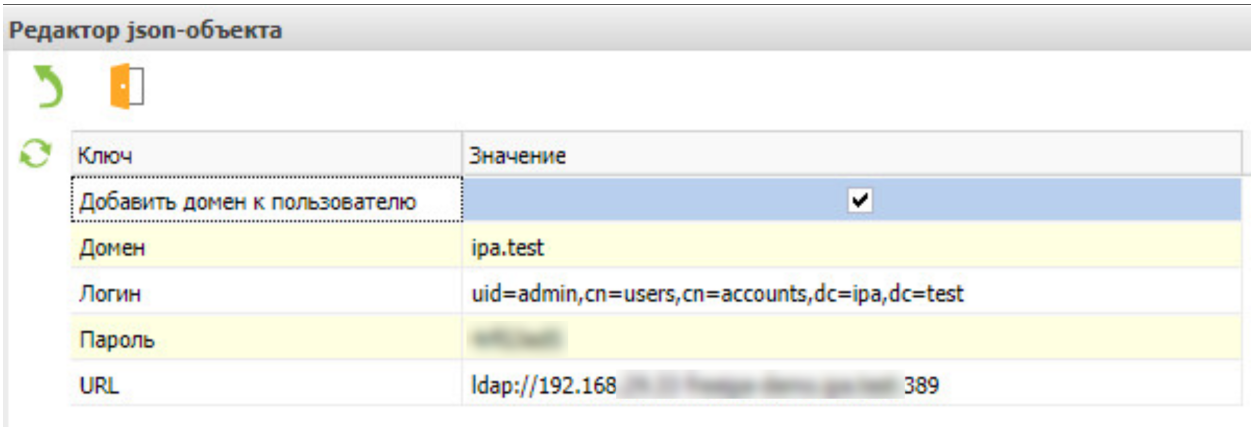


Рисунок 5. Настройки домена

Таблица 2. Настройки домена

Наименование	Описание
Добавить домен к пользователю	Если включено - добавляет к системному имени пользователя домен (User@Domain)
Домен	Домен через точки
Логин	Логин
Пароль	Пароль
URL	URL

После настройки можно выполнять синхронизацию. Для этого нужно зайти в приложение `Администратор` и открыть список пользователей.

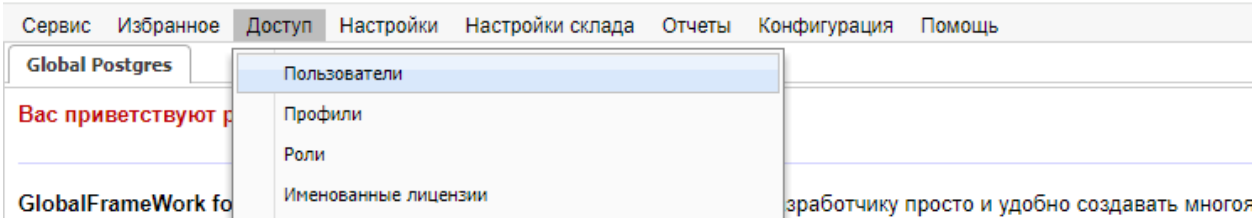


Рисунок 6. расположение списка пользователей

В списке пользователей нужно раскрыть выпадающий список у операции с молоточком и выбрать `Синхронизация пользователей Active Directory`

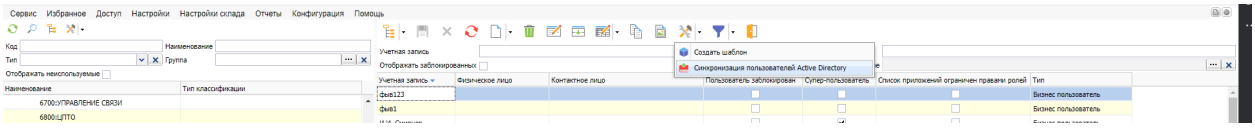


Рисунок 7. Операция синхронизации

В открывшемся окне можно выбрать шаблон для новых пользователей. Для запуска синхронизации нужно нажать на операцию с шестеренкой, после завершения синхронизации выведется отчёт и группы с пользователи в папках соответствующих доменов

- Все пользователи
  - LDAP - Active Directory
    - Домен: `Название первого домена`
    - Домен: `Название второго домена`
    - ...
    - Домен: `Название последнего домена`

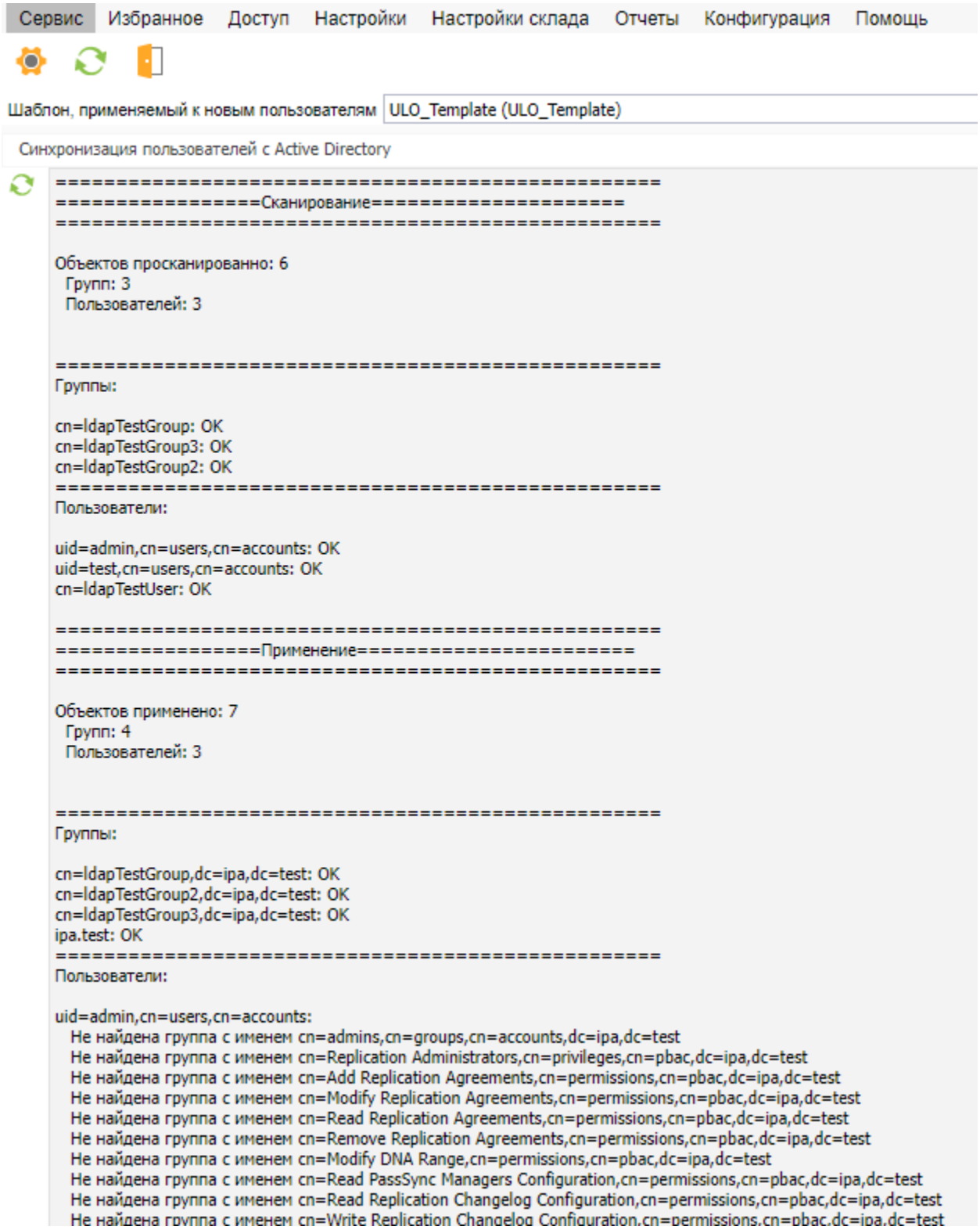


Рисунок 8. Окно синхронизации

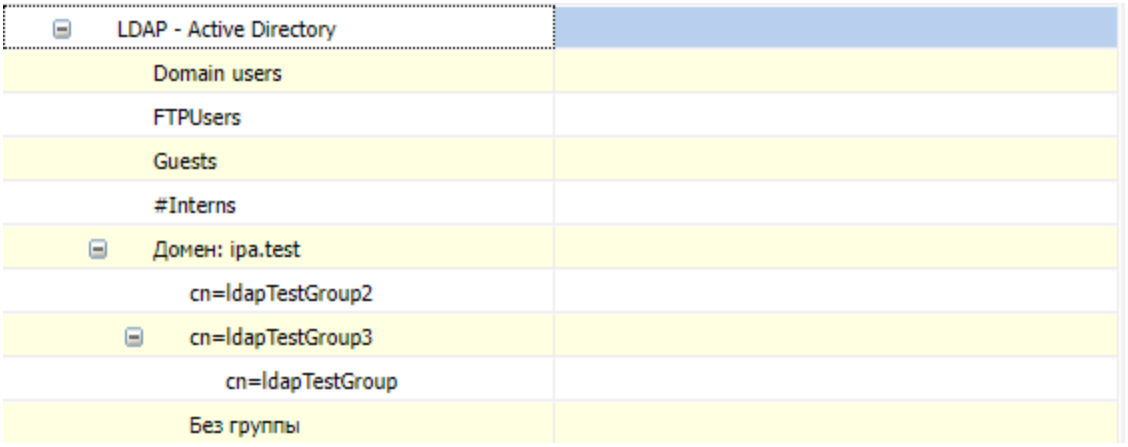


Рисунок 9. Итоговый результат синхронизации

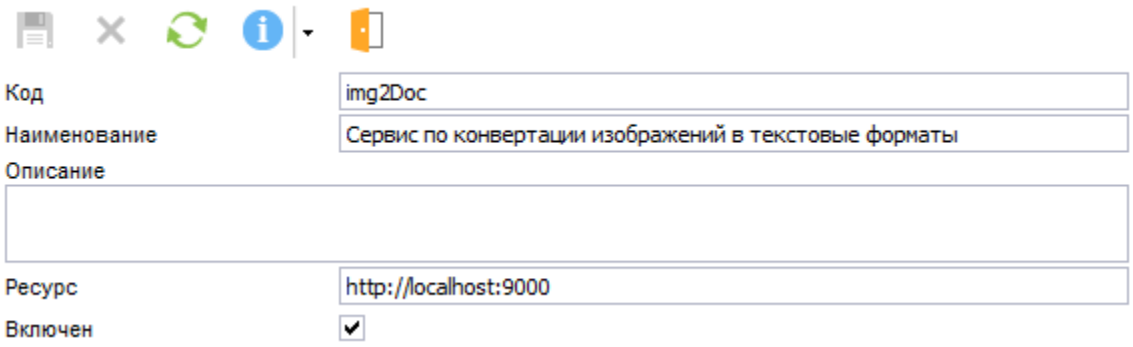
7.3 Установка и настройка сервиса конвертации избр. в документы

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 cpu 3Gb ram (~4 одновременных пользователя).

Интеграция с Global

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



7.4 Генерация ключей шифрования

GlobalScheduler может работать без указания ключей, однако такой режим не желателен. Основными недостатками работы без ключей являются:

- Без добавления ключей не работает выполнение заданий от каких-либо пользователей, кроме указанного в настройке btk;
- Не работает отслеживание ошибочных задач, так как данные по задачам планировщик получает с помощью ключей.

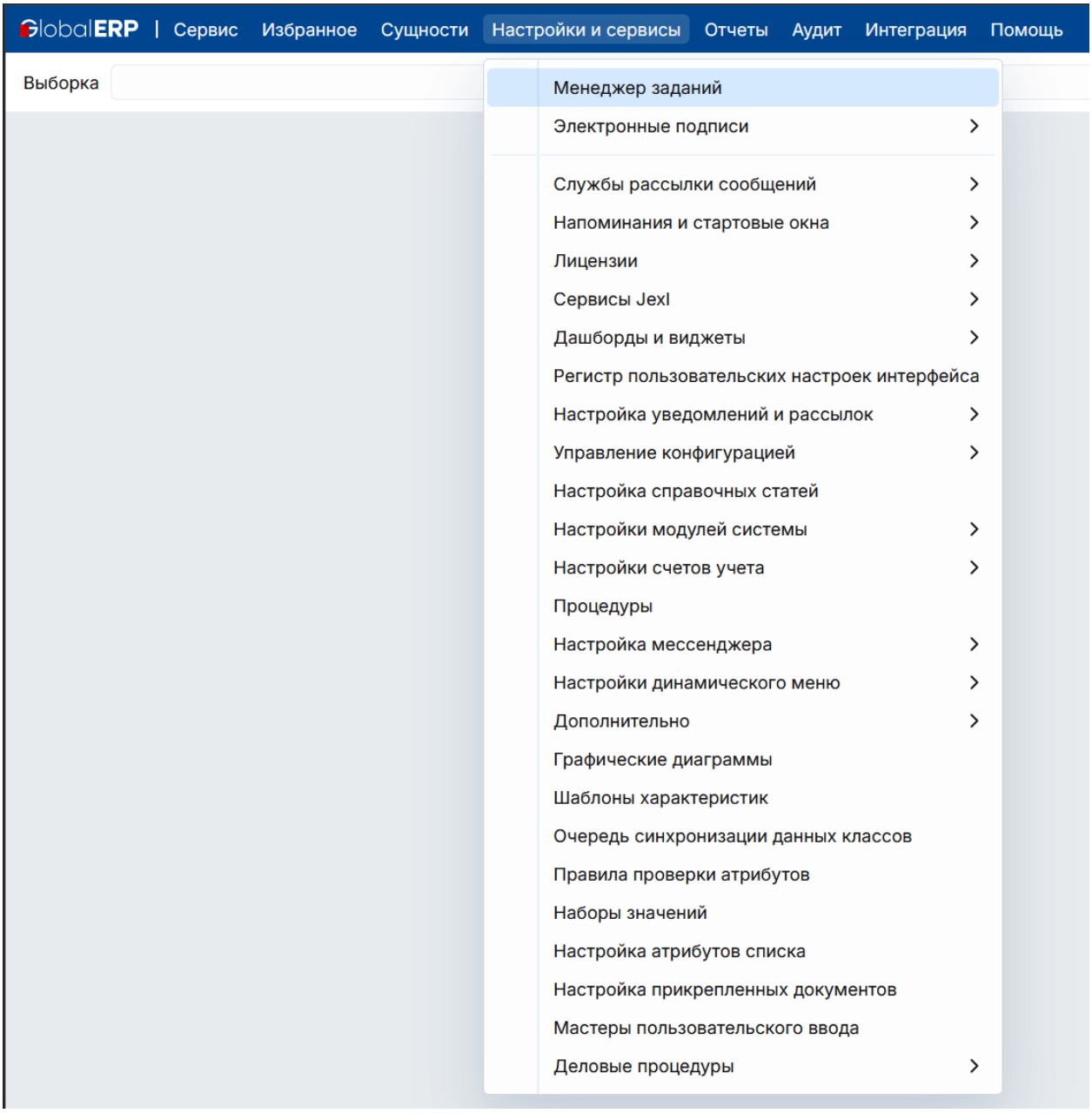
Важно

Основным режимом работы является работа с ключами. Логин, пароль используются на тестовых контурах для отладки работы GlobalScheduler. На проектах, в prod режиме не желательно указывать login, pass в настройках btk.

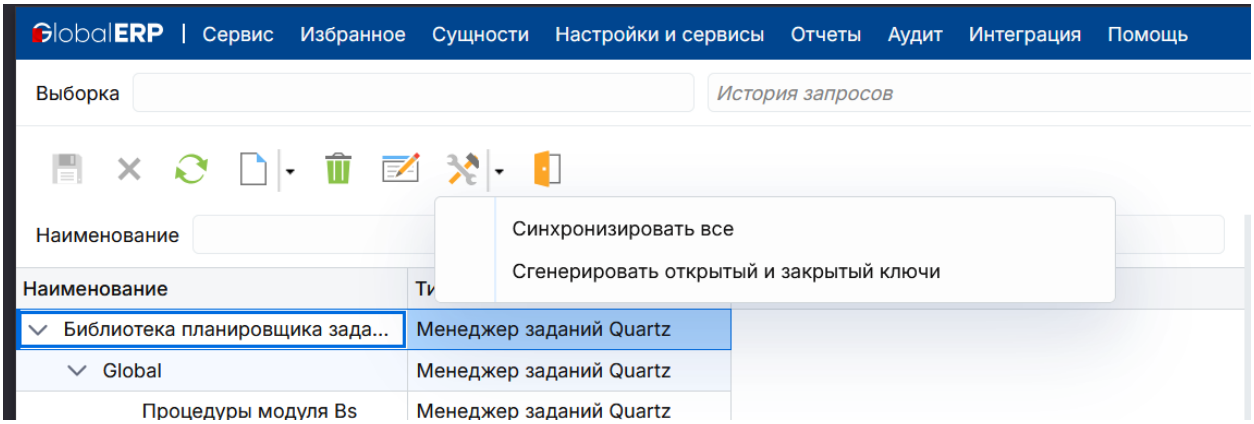
Генерация ключей в системе Global

Генерацию ключей можно выполнить с помощью системы Global.

Для этого в приложении «Настройки системы» откройте меню «Настройки и сервисы» > «Менеджер заданий»



В открывшемся окне выполните 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 сторонними утилитами.

Внимание

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

Пример создания ключей с помощью криптографической библиотеки OpenSSL:

```
# 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; \
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
```

Убедиться, 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"
```

7.5 Конфигурация GlobalScheduler в автономном режиме

Настройка конфигурационного файла

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

Где:

- <DBHOST> — адрес сервера postgres;
- <DBNAME> — имя БД;
- <DBUSER> — пользователь БД;
- <DBPASS> — пароль пользователя БД.

Настройка логирования планировщика

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
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                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/scheduler/jobscheduler.%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"/>
                <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 -->
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        <!--<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 скопируйте файл `/opt/global/globalserver/admin/linux/scheduler/globalscheduler.service.origin` в каталог `/lib/systemd/system` и переименуйте `globalscheduler.service`

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

или создайте 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
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
[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` укажите 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.

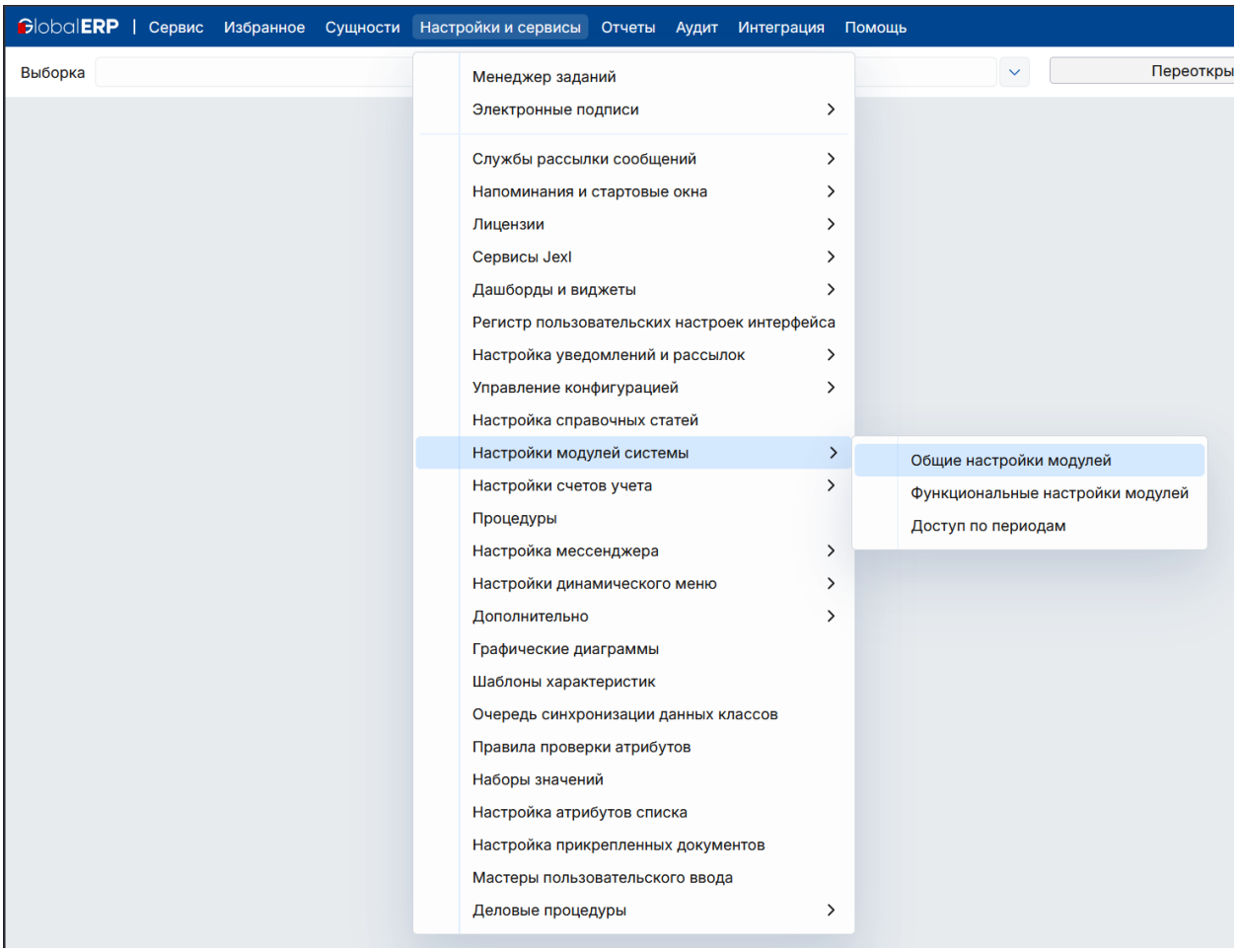
Для применения конфигурации перезапустите сервис globalscheduler:

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

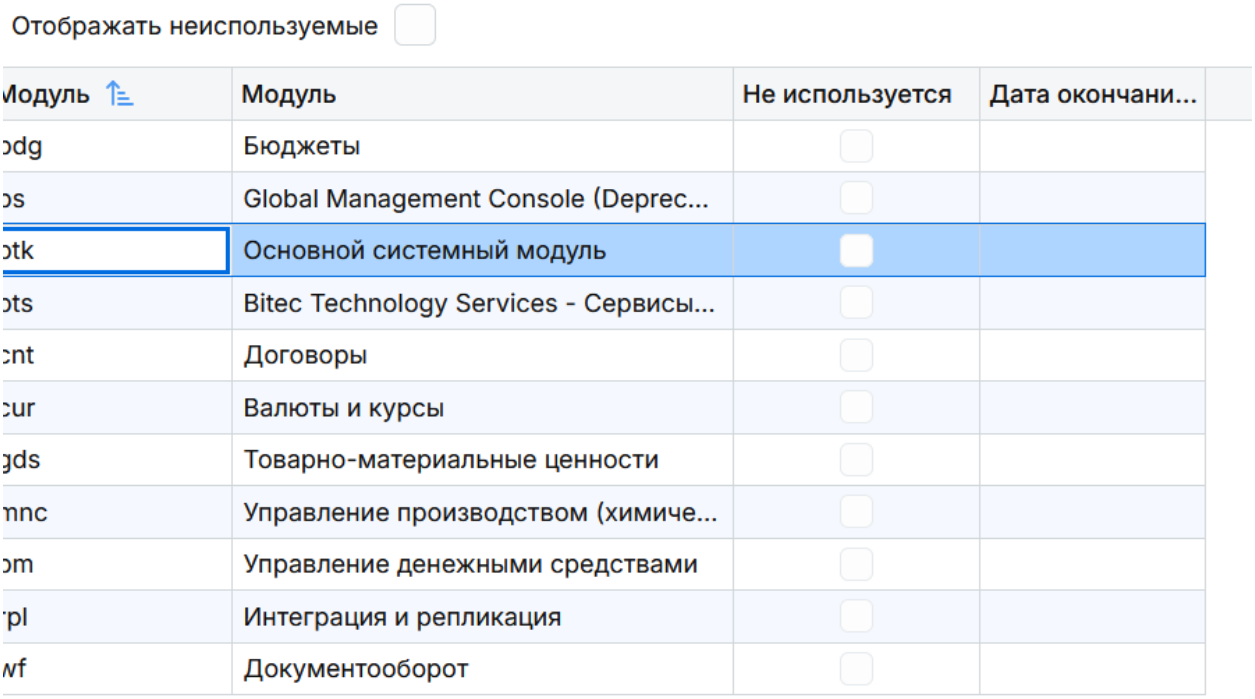
Установка публичного ключа и настроек Quartz

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

Откройте меню: «Настройки и сервисы» > «Настройки модулей системы» > «Общие настройки модулей»



В списке выберите модуль btk и откройте на редактирование



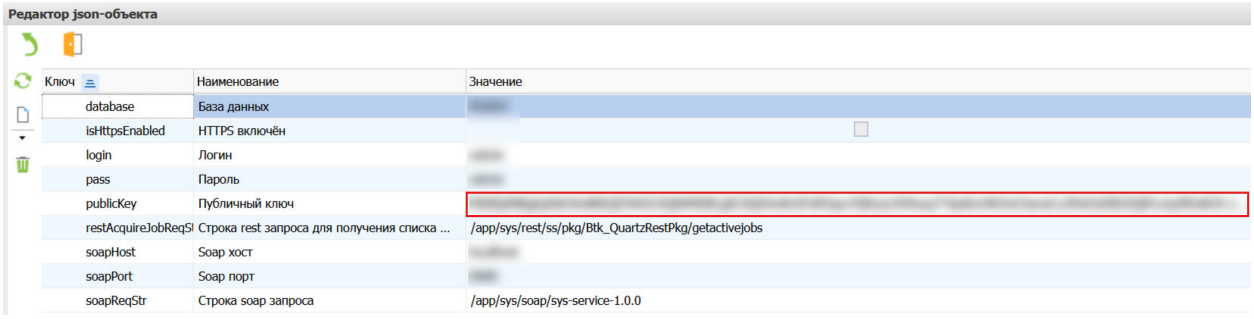
В открывшемся окне выберите «Настройки для Quartz» и нажмите на кнопку с тремя точками

Ключ	Наименование	Значение
proxy	Прокси	
extUserPublicKey	Публичный ключ для внешних пользователей	
sPgAgentScriptTpl	Шаблон скрипта для PgAgent-a	#!/bin/bash...
shedulerLogPath	Путь до логов планировщика	/usr/local/globalserver/logs/scheduler/
bTrackChangesAddToTheConfiguratior	Отслеживать изменения для добавления в ...	☐
bOpenAttachByDC	Скачивание файлов по двойному клику	☐
bSyncProfilesAndRoles	Синхронизация профилей с ролями доступа	☐
bWriteUnSuccessTraceJournal	Логирование неуспешных проверок прав ...	☐
bAdminSoap	Администрировать soap-вызовы	☐
bSafeDialectJexlScript	Выполнение jexl-скриптов через безопасный...	☐
bTextLangEnabledAll	Включить ввод текста на других языках для ...	☐
bPasswordSettingsInOT	bPasswordSettingsInOT	☐
BlockJavaScriptInPdf	Запрет загрузки PDF файлов, содержащих ...	☑
bWriteArgon2HashGenerateTime	Логирование времени генерации хеша Argon2	☑
bUsernameEnglishLettersOnly	Идентификатор пользователя должен ...	☑
bPepmitAdmin	Разрешено администрирование приложений	☑
bNoAdminSettingsDefault	Активность признака "Не распространяются ...	☑
bNoStateSettingsDefault	Активность признака "Не требуется настрой...	☑
bValidateDiscreteAccessBOOnSet	Проверять доступ к БО при установке ...	☑
JexlAccessJournal	Заполнять журнал доступа к объектам ...	☑
nAccessPeriodMonthCount	Количество месяцев для создания периода ...	12
nMaxDaysTelemetry	Макс. дней для хранения записей телеметрии	14
sXmppServer	Адрес XMPP-сервера	[redacted]
JwtParamServiceSecretKey	Ключ для сервиса токенизации параметров	[redacted]
browserCmdAbortDelay	Задержка отображения диалога ответа от ...	5000
nXmppPort	Порт XMPP-сервера	5222
ConfigManagerContoursList	Системные имена контуров, с которых ...	["*"]
sAudTableSpace	Табличное пространство аудита	pg_default
cQuartzProps	Настройки для Quartz	{"database":"[redacted]" ...
ldapSync	LDAP-синхронизация	{"jArrayDomains":[{}]}
QuotaUsageJournal	Журнал использования квот	{"nCellCount":1000,"bEnabled":false}
jSettingForGenGidUrl	Настройки для формирование URL, по ...	{"sTransferProtocolForGenGidUrl":"http","sHostNameForGenGidUrl":""}

В открывшемся окне задайте следующие значения:

- База данных: alias базы данных из файла global3.config.xml;
- HTTPS включён: в зависимости от того, используется ли https в Global, включите данную опцию;
- Логин: логин пользователя 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 нажмите зеленую стрелку, а затем операцию «Сохранить» в окне с настройками модуля btk.

Установка cacerts для java

Для работы `globalscheduler` с использованием `ssl`, добавьте сертификаты в хранилище `java`. Для этого используйте утилиту `keytool`

```
sudo keytool -import -keystore <cacerts path> -alias <Alias_Name> -file <certificate␣
↪file location>
```

Где:

- `cacerts path` — путь до файла `cacerts`, пример пути `/usr/lib/jvm/bellsoft-java21-full-amd64/lib/security/cacerts`;
- `alias name` — название записи;
- `certificate file location` — путь до сертификата.

Пример полной команды:

```
sudo keytool -import -keystore /usr/lib/jvm/bellsoft-java21-full-amd64/lib/security/
↪cacerts -alias gs-ca -file /home/sysadm/gs-CA.crt
```

7.6 Конфигурация GlobalScheduler в Kubernetes

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

```
base64 -w 0 ./quartz.private.txt > ./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
meta
  name: scheduler-token-secret
  namespace: gs-cluster-k8s

  private.key: <b64_key>
```

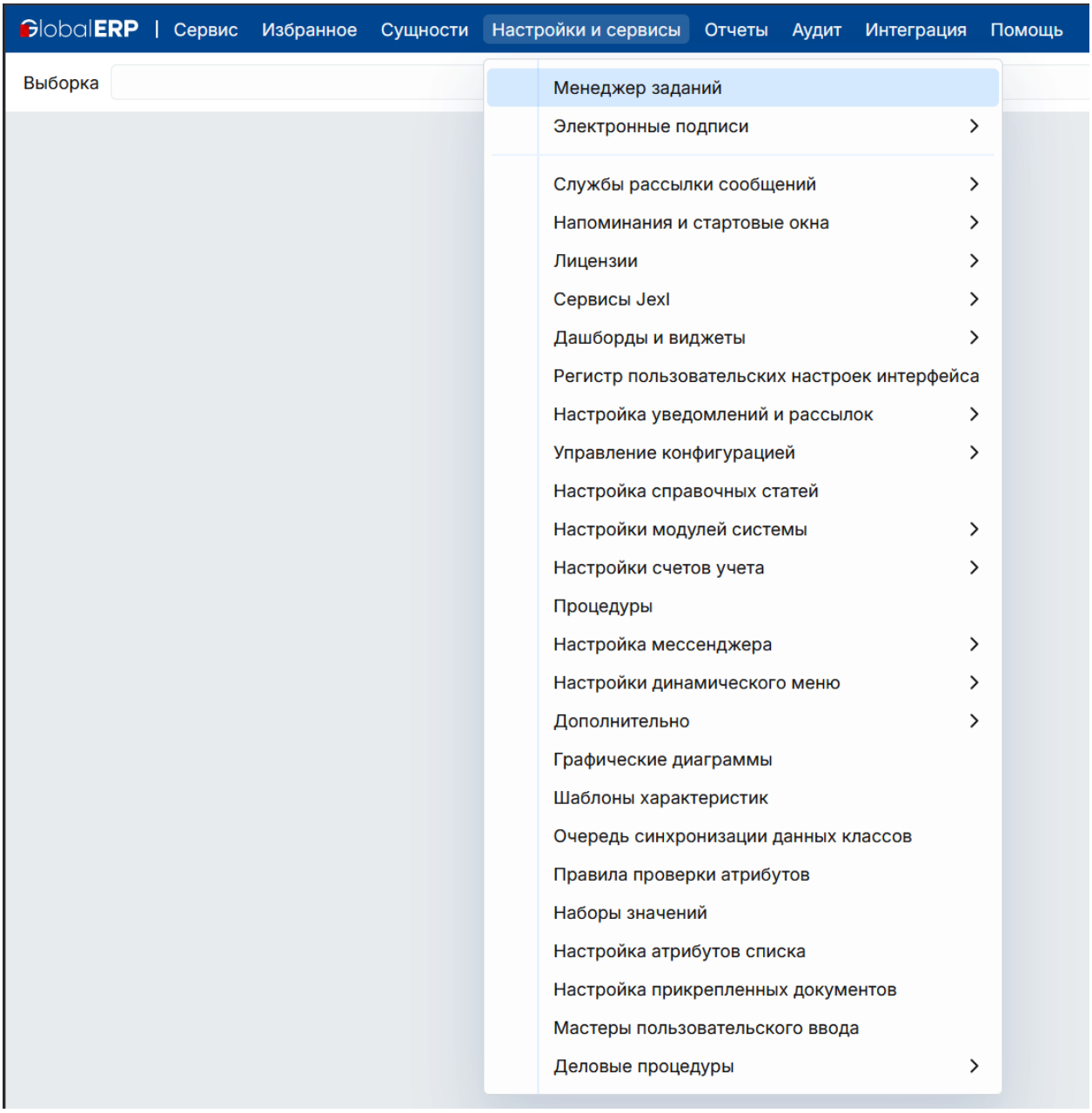
```
cat <<EOF | tee ~/nscli/workspace/scheduler-token-secret.yaml
apiVersion: v1
kind: Secret
meta
  name: scheduler-token-secret
  namespace: gs-cluster-k8s

  private.key: <b64_key>
EOF
```

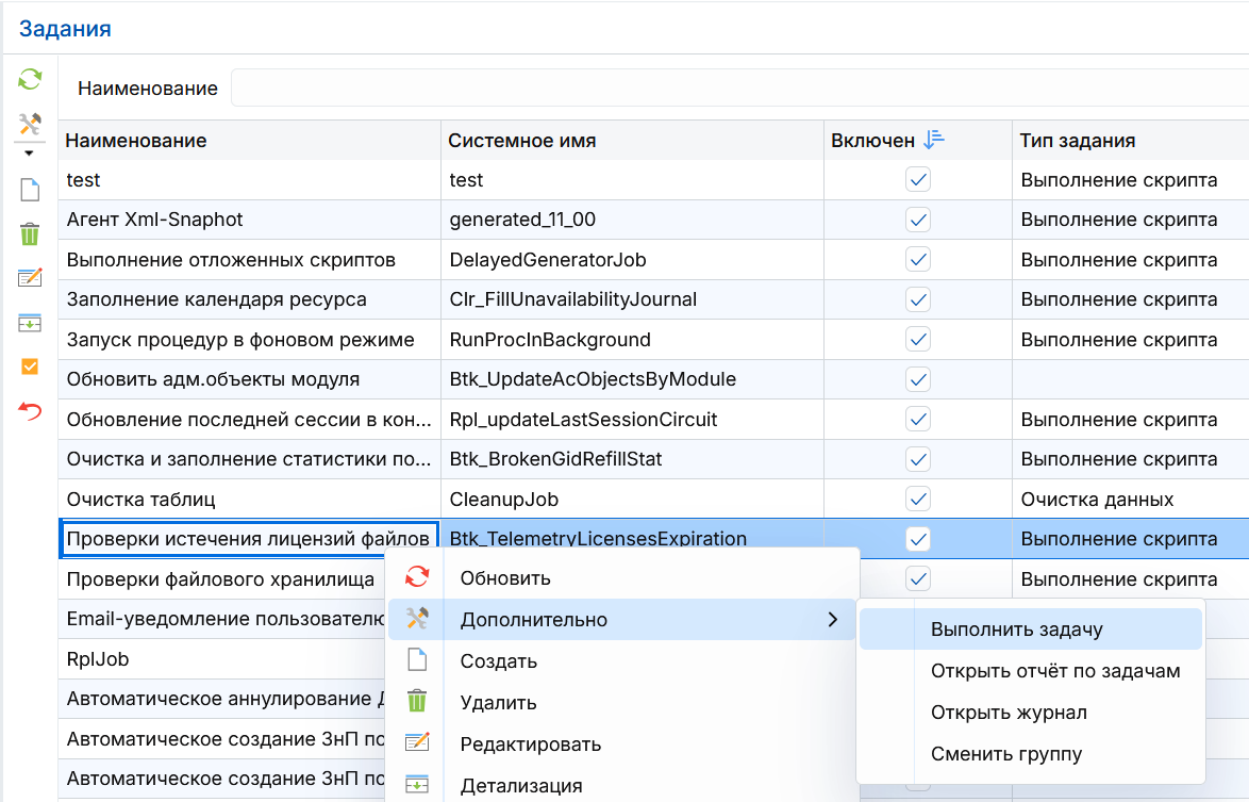
```
kubectl apply -f ~/nscli/workspace/scheduler-token-secret.yaml
```

7.7 Проверка работоспособности сервиса

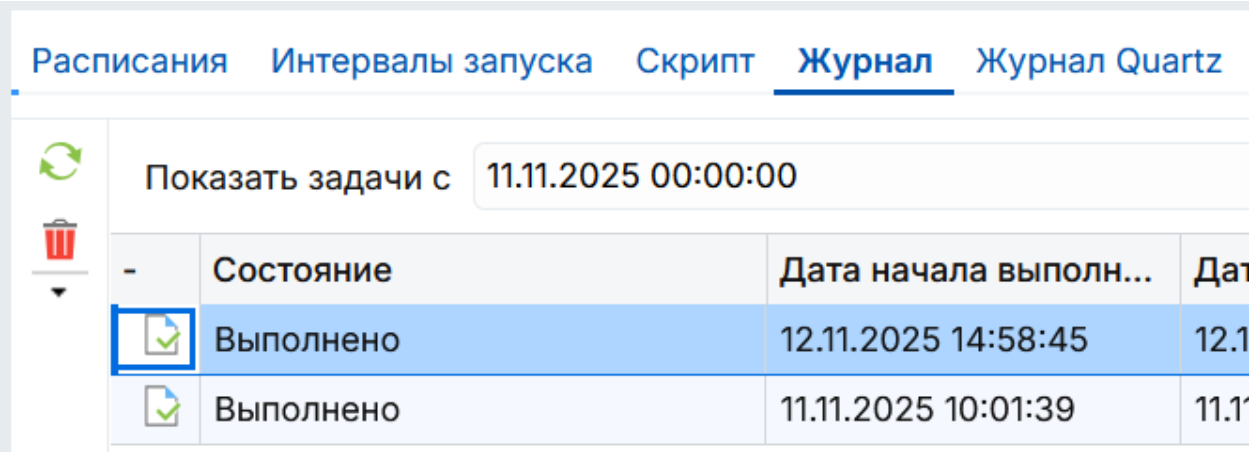
Перейдите в Менеджер заданий:



Правой кнопкой мыши нажмите на активное задание, перейдите в пункт «Дополнительно», выполните команду «Выполнить задачу»:



В нижней части страницы перейдите в окно настроек задачи, перейдите на вкладку «Журнал». В списке будет отображены даты запуска выбранной задачи. Обновите список кнопкой «Обновить», через некоторое время scheduler выполнит задачу и ее состояние переключится на «Выполнено». Если спустя несколько минут состояние не обновилось, необходимо проверить логи сервиса на предмет ошибок, а также перепроверить конфигурацию и настройки.



7.8 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/quarz.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*)

7.9 Руководство по снятию снапшотов GlobalScheduler

Назначение

Процедура снятия снапшотов (thread dump) процесса GlobalScheduler позволяет диагностировать проблемы производительности, зависания потоков и дедлоки в системе Global ERP.

Примечание

Снятие снапшотов требуется при следующих симптомах:

- Зависание задач планировщика.
- Высокая нагрузка на CPU процесса GlobalScheduler.
- Увеличивающееся потребление памяти.
- Ошибки выполнения запланированных заданий.
- Длительное выполнение задач без прогресса.

Общие предварительные требования

- Установленная Java Development Kit (JDK) той же версии, что используется для запуска GlobalScheduler.
- Знание PID процесса Java GlobalScheduler.
- Права доступа к процессу GlobalScheduler.

Часть 1: Снятие снапшотов в Kubernetes

Предварительные требования для Kubernetes

- Доступ к кластеру Kubernetes с правами на выполнение `kubectl exec`.
- Настроенный `kubeconfig` для namespace `gs-ctk`.

Шаг 1.1: Поиск пода GlobalScheduler

```
kubectl get pods -n gs-ctk
```

Пример вывода:

```
NAME                                              READY   STATUS    RESTARTS        AGE
gs-cluster-1-global-scheduler-7d49bffb5f-d9llp    2/2     Running   2 (3m12s ago)   18h
gs-cluster-1-global-server-excl-cc4d9c647-jshzq   2/2     Running   5 (3m29s ago)   18h
gs-cluster-1-global-server-share-95bc95764-lql4x  2/2     Running   6 (3m12s ago)   18h
gs-cluster-1-grafana-0                            1/1     Running   1 (3m12s ago)   18h
gs-cluster-1-haproxy-6d4b44cf94-lzlj2             2/2     Running   2 (3m29s ago)   18h
gs-cluster-1-rabbitmq-0                           1/1     Running   1 (3m29s ago)   18h
nsctl-6f59bdc5c8-b8vsv                            1/1     Running   1 (3m29s ago)   18h
```

Шаг 1.2: Определение PID процесса Java

```
# Подключение к поду и поиск PID
kubectl exec -it gs-cluster-1-global-scheduler-7d49bffb5f-d9llp -n gs-ctk -c␣
↪globalscheduler -- /bin/bash

# Поиск PID процесса Java (обычно дочерний от python3)
ps aux | grep java
# ИЛИ
pstree -p 1
```

Пример вывода:

```
python3(1)—java(27)—{java}(35)
```

В данном случае PID процесса Java = 27.

Шаг 1.3: Снятие thread dump в Kubernetes

```
# Определение переменных
POD_NAME="gs-cluster-1-global-scheduler-7d49bffb5f-d9llp"
NAMESPACE="gs-ctk"
JAVA_PID=27

# Снятие одиночного дампа
kubectl exec $POD_NAME -n $NAMESPACE -c globalscheduler -- /bin/bash -c "/root/
↪globalserver/workspace/local/jdk21/bin/jstack $JAVA_PID" > thread_dump_k8s_$(date +%Y%m
↪%d_%H%M%S).txt

# Снятие нескольких дампов с интервалом
for i in {1..3}; do
    echo "Снятие дампа $i в $(date)"
    kubectl exec $POD_NAME -n $NAMESPACE -c globalscheduler -- /bin/bash -c "/root/
↪globalserver/workspace/local/jdk21/bin/jstack $JAVA_PID" > thread_dump_k8s_${i}_
↪$(date +%Y%m%d_%H%M%S).txt
    sleep 10
done
```

Шаг 1.4: Снятие heap dump в Kubernetes

```
# Снятие heap dump
kubectl exec $POD_NAME -n $NAMESPACE -c globalscheduler -- /bin/bash -c "/root/
↪globalserver/workspace/local/jdk21/bin/jmap -dump:live,format=b,file=/tmp/heap_dump.
↪hprof $JAVA_PID"

# Копирование с пода
kubectl cp $NAMESPACE/$POD_NAME:/tmp/heap_dump.hprof ./heap_dump_k8s_$(date +%Y%m%d_%H%M
↪%S).hprof -c globalscheduler

# Очистка
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
kubectl exec $POD_NAME -n $NAMESPACE -c globalscheduler -- /bin/bash -c "rm -f /tmp/heap_
↪dump.hprof"
```

Часть 2: Снятие снапшотов на автономном сервере

Предварительные требования для автономного сервера

- Права sudo пользователя или доступ под пользователем `global`.
- Знание пути к установке GlobalScheduler (обычно `/opt/global/globalserver`).

Шаг 2.1: Определение PID процесса GlobalScheduler

```
# Определение пода через systemd
sudo systemctl status globalscheduler
PID=$(systemctl show globalscheduler --property=MainPID | cut -d= -f2)

# Проверка PID
echo "PID процесса GlobalScheduler: $PID"
# PID должен совпадать с тем что вывел syst
```

Шаг 2.2: Определение пути к Java

```
# Проверка используемой Java
sudo ls -l /proc/$PID/exe

# Проверка установленных версий Java
update-alternatives --list java
```

Шаг 2.3: Снятие thread dump на автономном сервере

```
# Снятие одиночного дампа
sudo $JAVA_HOME/bin/jstack $PID > /tmp/thread_dump_standalone_$(date +%Y%m%d_%H%M%S).txt

# Снятие нескольких дампов
for i in {1..3}; do
    echo "Снятие дампа $i в $(date)"
    sudo $JAVA_HOME/bin/jstack $PID > /tmp/thread_dump_standalone_${i}_$(date +%Y%m%d_%H
↪%M%S).txt
    sleep 10
done
```

Шаг 2.4: Снятие heap dump на автономном сервере

```
# Снятие heap dump
sudo $JAVA_HOME/bin/jmap -dump:live,format=b,file=/tmp/heap_dump_standalone_$(date +%Y%m
↪%d_%H%M%S).hprof $PID

# Проверка размера
ls -lh /tmp/heap_dump_*.hprof
```

Предупреждение

Снятие heap dump может занять значительное время и потребовать много места на диске (обычно несколько гигабайт). Убедитесь, что в /tmp достаточно свободного места.

Часть 3: Анализ полученных дампов

Быстрый анализ через командную строку

```
# Для анализа перейдите в директорию с дампами
cd /path/to/dumps

# Поиск заблокированных потоков
grep -c "BLOCKED" thread_dump_*.txt

# Поиск deadlock
grep -l "deadlock" thread_dump_*.txt

# Подсчет потоков по состоянию
grep "java.lang.Thread.State" thread_dump_*.txt | awk '{print $2}' | sort | uniq -c |␣
↪sort -nr

# Анализ состояния всех потоков
for file in thread_dump_*.txt; do
    echo "=== $file ==="
    grep "java.lang.Thread.State" "$file" | awk '{print $2}' | sort | uniq -c
    echo
done
```

Часть 4: Интерпретация результатов для GlobalScheduler

Типичные сценарии проблем

Сценарий 1: Дедлок в базе данных

```
Found one Java-level deadlock:
=============================
"Thread-15":
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
  waiting to lock monitor 0x00007f8c4c00b8e0 (object 0x00000000f8a8f8a0, a org.
↪postgresql.core.v3.QueryExecutorImpl)
```

Решение: Проверка длительных транзакций в БД PostgreSQL.

Сценарий 2: Блокировка ресурсов планировщика

```
"QuartzScheduler_Worker-10" #25 prio=5 os_prio=0 tid=0x00007f8c54010800 nid=0x5e3a␣
↪waiting for monitor entry [0x00007f8c4a1fe000]
   java.lang.Thread.State: BLOCKED (on object monitor)
```

Решение: Увеличение размера пула потоков в настройках Quartz.

Сценарий 3: Проблемы с подключением к Global ERP

```
"Thread-8" #20 prio=5 os_prio=0 tid=0x00007f8c5420a800 nid=0x5f1c runnable␣
↪[0x00007f8c4a3ff000]
   java.lang.Thread.State: RUNNABLE
   at java.net.SocketInputStream.socketRead0(Native Method)
```

Решение: Проверка подключения к SOAP-сервису Global ERP.

Часть 5: Устранение распространенных проблем

Общие проблемы и решения

Проблема: «Permission denied» при выполнении jstack/jmap

В Kubernetes: Убедитесь, что у пользователя есть права на выполнение `kubectl exec` в namespace `gs-ctk`.

На автономном сервере: Выполняйте команды с правами пользователя `global`:

```
sudo -u global jstack $PID
```

Проблема: Процесс Java не найден

В Kubernetes: Проверьте PID процесса Java командой внутри пода:

```
kubectl exec $POD_NAME -n $NAMESPACE -c globalscheduler -- ps aux | grep java
```

На автономном сервере: Убедитесь, что GlobalScheduler запущен:

```
sudo systemctl status globalscheduler
```

Проблема: jstack/jmap не найдены

Решение: Укажите полный путь к утилитам JDK:

```
# В Kubernetes
/root/globalserver/workspace/local/jdk21/bin/jstack $PID

# На автономном сервере
/usr/local/jdk-21/bin/jstack $PID
# ИЛИ
/usr/lib/jvm/java-21-openjdk-amd64/bin/jstack $PID
```

Проблема: Недостаточно места для heap dump

Решение: Используйте директорию с достаточным местом:

```
# В Kubernetes
kubectl exec $POD_NAME -n $NAMESPACE -c globalscheduler -- /bin/bash -c "/root/
↪globalserver/workspace/local/jdk21/bin/jmap -dump:live,format=b,file=/opt/global/temp/
↪heap_dump.hprof $JAVA_PID"

# На автономном сервере
sudo -u global jmap -dump:live,format=b,file=/opt/global/temp/heap_dump.hprof $PID
```

Часть 6: Дополнительные диагностические команды

Для автономного сервера

```
# Мониторинг системных ресурсов
top -p $PID
htop -p $PID

# Память процесса
pmap -x $PID | tail -1

# Открытые файлы
lsof -p $PID | wc -l

# Логи GlobalScheduler
sudo journalctl -u globalscheduler -f
sudo tail -f /opt/global/globalserver/logs/globalscheduler.log
```

Для Kubernetes

```
# Логи пода
kubectl logs -n gs-ctk $POD_NAME -c globalscheduler -f

# Мониторинг ресурсов
kubectl top pod -n gs-ctk $POD_NAME

# Описание пода для диагностики
kubectl describe pod -n gs-ctk $POD_NAME

# Проверка состояния контейнера
kubectl get pod $POD_NAME -n gs-ctk -o jsonpath='{.status.containerStatuses[0].ready}'
```

Часть 7: Автоматический мониторинг и снятие дампов

Скрипт для автоматического снятия дампов при высокой нагрузке (Kubernetes)

```
#!/bin/bash
NAMESPACE="gs-ctk"
POD_LABEL="app=globalscheduler"
POD_NAME="gs-cluster-1-global-scheduler-7d49bffb5f-d9llp"
CPU_THRESHOLD=80
MEMORY_THRESHOLD=85

while true; do

    if [ -z "$POD_NAME" ]; then
        echo "$(date): Pod not found"
        sleep 60
        continue
    fi

    # Получаем использование CPU и памяти
    POD_STATS=$(kubectl top pod -n $NAMESPACE $POD_NAME --no-headers 2>/dev/null)

    if [ $? -eq 0 ]; then
        CPU_USAGE=$(echo $POD_STATS | awk '{print $2}' | sed 's/%//')
        MEMORY_USAGE=$(echo $POD_STATS | awk '{print $3}' | sed 's/%//')

        echo "$(date): Pod $POD_NAME - CPU: ${CPU_USAGE}%, Memory: ${MEMORY_USAGE}%"

        # Проверяем пороги
        if [ $CPU_USAGE -gt $CPU_THRESHOLD ] || [ $MEMORY_USAGE -gt $MEMORY_THRESHOLD ];␣
↪then
            echo "$(date): High resource usage detected! Taking emergency thread dump..."

            # Получаем PID Java процесса
            JAVA_PID=$(kubectl exec $POD_NAME -n $NAMESPACE -c globalscheduler -- ps aux␣
↪| grep java | grep -v grep | awk '{print $2}')
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
            if [ -n "$JAVA_PID" ]; then
                TIMESTAMP=$(date +%Y%m%d_%H%M%S)
                kubectl exec $POD_NAME -n $NAMESPACE -c globalscheduler -- /bin/bash -c
↪"/root/globalserver/workspace/local/jdk21/bin/jstack $JAVA_PID" > emergency_dump_$
↪{TIMESTAMP}.txt
                echo "$(date): Emergency dump saved to emergency_dump_${TIMESTAMP}.txt"
            fi
        fi
    else
        echo "$(date): Failed to get pod metrics"
    fi

    sleep 60
done
```

Часть 8: Дополнительные инструменты анализа

```
# Визуальные инструменты (требуют GUI)
jvisualvm
jmc
```

Совет

Для комплексного анализа рекомендуется снимать несколько дампов с интервалом 10–15 секунд и сравнивать их для выявления изменяющихся состояний потоков. Всегда сохраняйте дампы с временными метками для последующего анализа.

При анализе обращайте внимание на:

1. Количество блокированных потоков (BLOCKED).
2. Наличие deadlock.
3. Потоки, находящиеся в состоянии RUNNABLE продолжительное время.
4. Рост количества потоков со временем.
5. Потоки, ожидающие доступа к одним и тем же мониторам.

7.10 Настройка визуализации контуров приложения

1. Назначение

Данный раздел описывает процесс настройки визуального обозначения контуров работы приложения (например, test, dev, prod).
Функция позволяет визуально отличать окружения при работе с интерфейсом — в зависимости от выбранной базы данных цвет рабочего поля изменяется согласно заданной конфигурации.

2. Добавление индикатора контура в конфигурацию БД

Откройте файл конфигурации приложения:

```
nano /opt/global/globalserver/global3.config.xml
```

Для каждой записи о базе данных необходимо добавить атрибут `environmentIndicator`, указывающий на название контура.

Пример записи:

```
<database alias="PGTEST" driver="org.postgresql.Driver" schema="PUBLIC"
          url="jdbc:postgresql://192.168.1.1:6432/postgres"
          connectionType="proxyShared" authenticationType="btk"
          environmentIndicator="TEST_DB"></database>

<database alias="PGDEV" driver="org.postgresql.Driver" schema="PUBLIC"
          url="jdbc:postgresql://192.168.1.2:6432/postgres"
          connectionType="proxyShared" authenticationType="btk"
          environmentIndicator="DEV_DB"></database>
```

После секции `<databases>` необходимо добавить новый тег `<environmentIndicators>`, в котором задаются параметры для каждого контура: название, описание и цвет подсветки интерфейса.

Пример конфигурации:

```
<environmentIndicators activated="true" defaultIndicator="TEST_DB">
    <!-- Включена видимость рамки главной формы приложения и фоновый цвет -->
    <indicator name="TEST_DB">
        <views caption="Запущен контур TEST" color="#87cefa">
            <MainFormBorder/>
            <BackgroundColor/>
        </views>
    </indicator>
    <indicator name="DEV_DB">
        <views caption="Запущен контур DEV" color="#eb1a21">
            <MainFormBorder/>
            <BackgroundColor/>
        </views>
    </indicator>
</environmentIndicators>
```

Пояснения:

- **activated=»true»** — включает отображение визуальных индикаторов.
- **defaultIndicator** — задаёт индикатор по умолчанию.
- **caption** — текст, отображаемый в интерфейсе.
- **color** — цвет рамки и фона рабочего поля (в формате HEX).
- **MainFormBorder** — активирует цветную рамку вокруг главного окна. Необязательный параметр.
- **BackgroundColor** — задаёт цвет фона рабочей области. Необязательный параметр.

3. Применение изменений

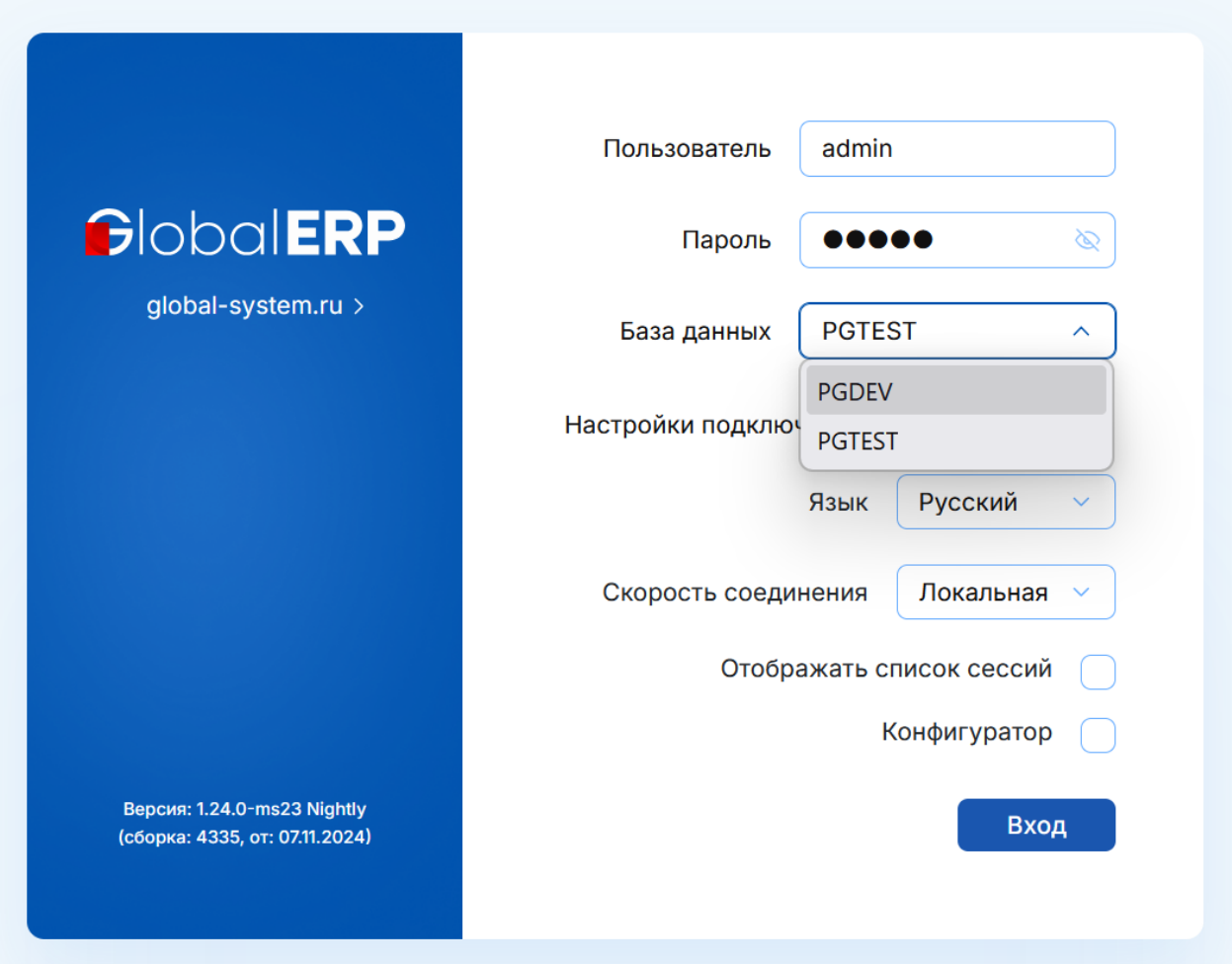
После внесения изменений сохраните файл и **перезапустите сервер**.

```
sudo systemctl restart global3
```

После перезапуска приложение автоматически применит визуальные настройки.
Цвет и подпись интерфейса будут соответствовать выбранному контуру, указанному в конфигурации базы данных.

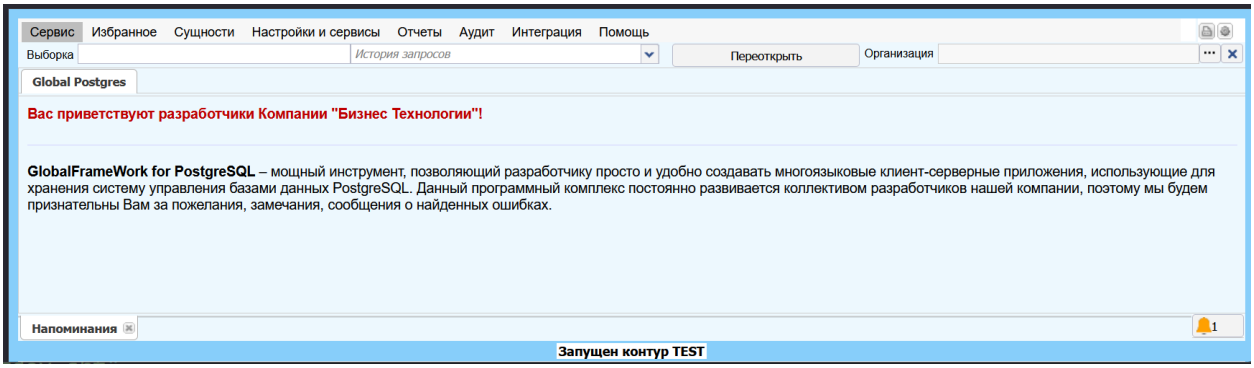
4. Тестирование

На стартовой странице выберете необходимый вам контур:

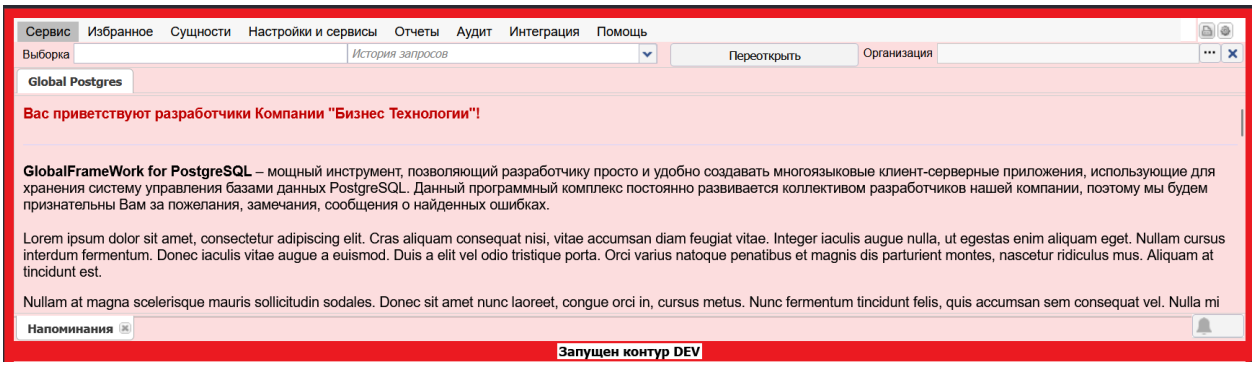


Перейдите в любое приложение. В соответствие с конфигурацией будет добавлена рамка и изменен фон рабочей области.

Для первого контура (Тест):



Для второго контура (Dev):



7.11 Проверка связи улов кластера

Инструкция, как проверить, что узлы кластера общаются между собой на примере сброса кешей приложения.

1. Выполнить скрипт установки кеша на двух разных узлах:

```
var manager = session.sbtClassLoader().loadClass("ru.bitec.app.gtk.eclipse.CacheManager
↪");
var cache = manager.getCache("test_cache", session);
cache.put("test_cache", 1);
if (cache.get("test_cache") == 1) {
  dialogs.showMessage('Кеш установлен')
}
```

2. На одном из узлов выполнить скрипт очистки:

```
var manager = session.sbtClassLoader().loadClass("ru.bitec.app.gtk.eclipse.CacheManager
↪");
manager.clearCaches();
dialogs.showMessage('Кеш сброшен')
```

3. Выполнить скрипт проверки кеша на разных узлах:

```
var manager = session.sbtClassLoader().loadClass("ru.bitec.app.gtk.eclipse.CacheManager
↪");
var cache = manager.getCache("test_cache", session);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
if (cache.get("test_cache") == 1) {
  dialogs.showMessage('Ошибка! Кеш не очищен!')
} else {
  dialogs.showMessage('Кеш очищен')
}
```

8 Лицензирование системы

8.1 Описание

Лицензирование системы привязывается к конкретной базе данных. Уникальный код базы формируется с учётом аппаратно-программного окружения СУБД и параметров самой базы и однозначно соответствует данной БД.

Любые изменения в окружении (например: тип процессора, материнская плата, MAC-адрес сетевого интерфейса, версия СУБД, имя базы данных и т. п.) могут привести к утрате актуальности лицензии.

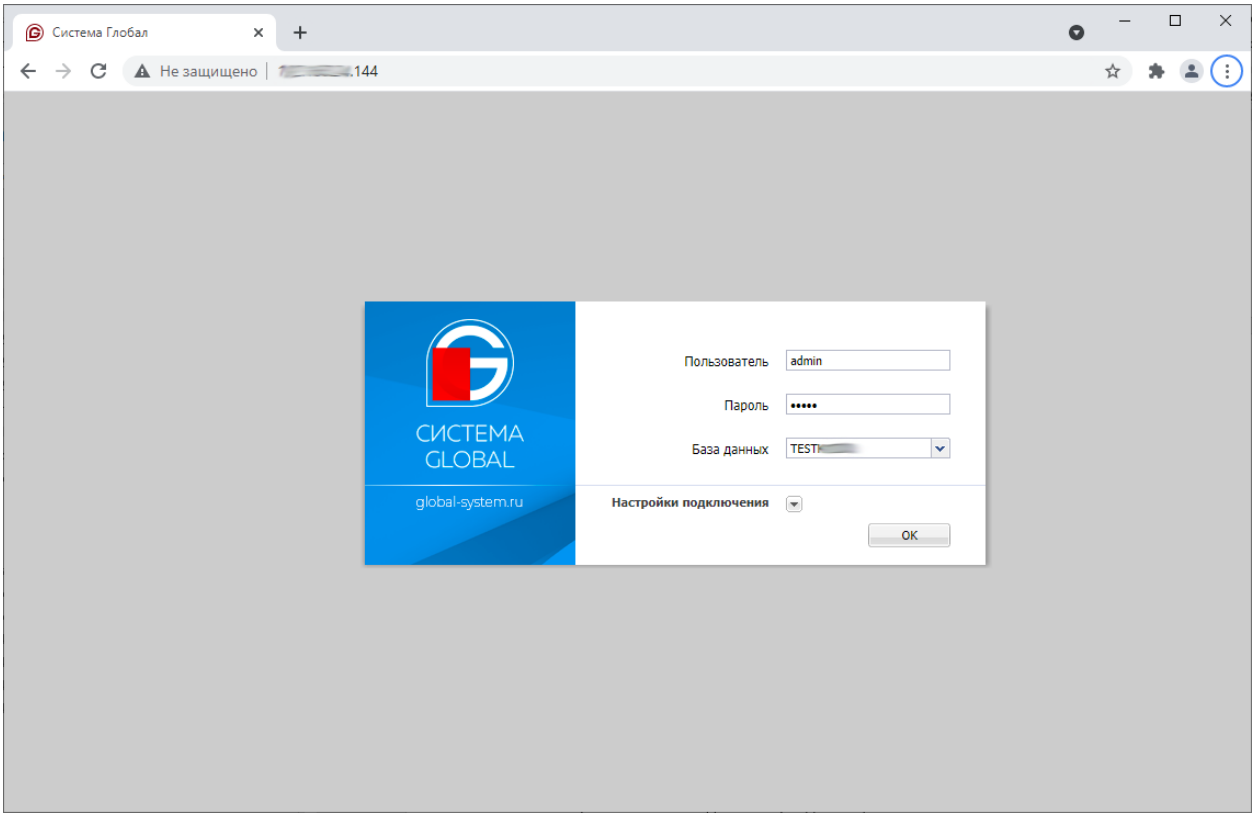
Лицензия устанавливается при первом входе в систему и при необходимости может быть впоследствии изменена или дополнена.

8.2 Первый вход и активация лицензии

Авторизация

Откройте в браузере адрес:

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



На странице входа используйте следующие учётные данные:

- **Логин:** `admin`
- **Пароль:** `admin`

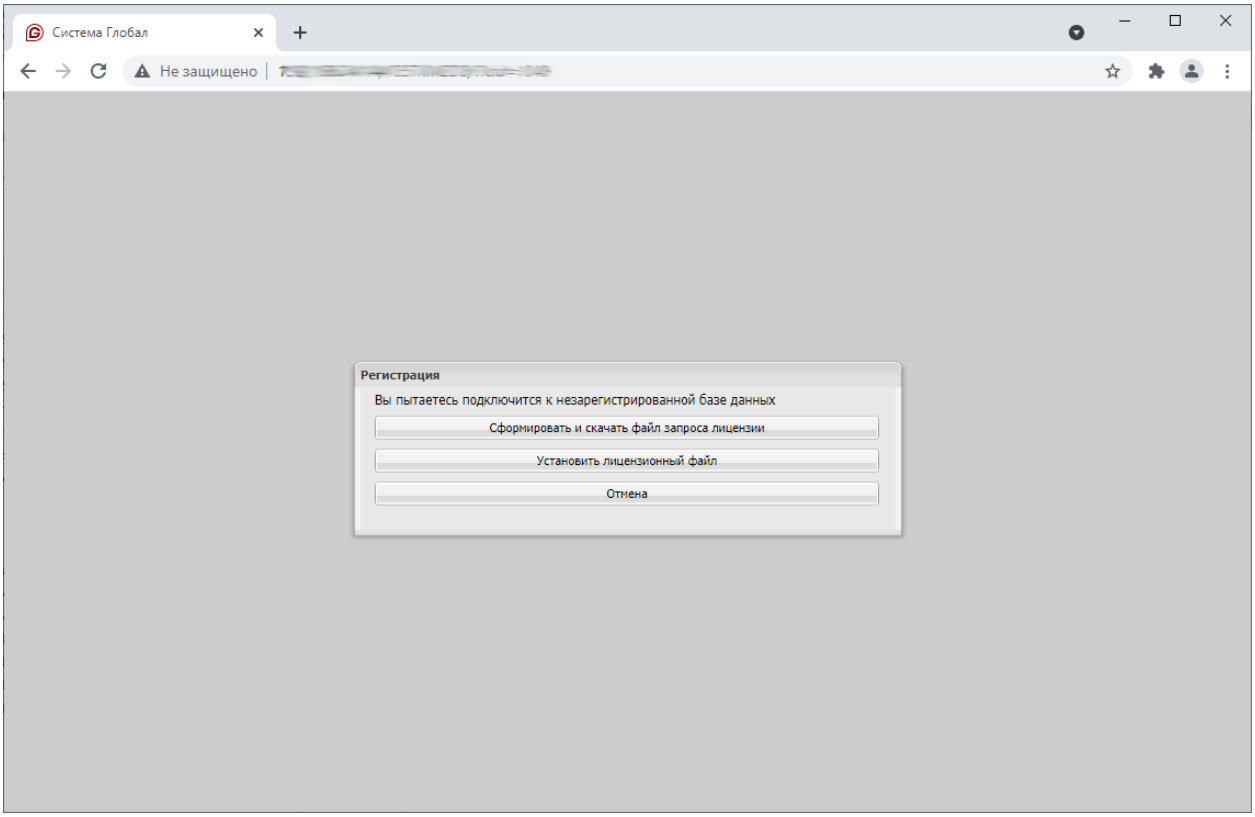
Примечание

Если используется поставочный дамп, учётные данные для входа предоставляет ответственное контактное лицо.

Регистрация базы и получение лицензии

После первого входа система предложит зарегистрировать базу данных и сформировать файл запроса лицензии.

1. Сформируйте файл запроса лицензии на экране регистрации.
2. Передайте полученный файл ответственному контактному лицу в ООО «Бизнес-Технологии».
3. Получив от него лицензионный файл, вновь войдите в систему и нажмите **«Установить лицензионный файл»**.
4. В открывшемся системном окне выберите соответствующий файл и подтвердите выбор.
5. Система проверит корректность лицензии и, при отсутствии ошибок, активирует приложение.



8.3 Управление лицензиями

Откройте приложение **«Настройка системы»** и перейдите по пути: **Настройки и сервисы** → **Лицензии** → **Лицензионные файлы**.

Вкладка «Файлы лицензий»

На вкладке отображаются все загруженные в систему лицензии. Табличное представление содержит основную информацию по каждой лицензии.

В нижней части экрана расположены вкладки **«Приложения»**, **«Модули»**, **«Соединения»**, где можно проверить:

- включённые в лицензию приложения и модули;
- количество одновременных подключений;
- срок действия лицензии.

Слева находятся функциональные кнопки:

- сформировать запрос на новую лицензию;
- добавить лицензию;
- удалить лицензию.

При добавлении лицензии система сравнивает её серийный номер с серийными номерами уже загруженных лицензий:

- при совпадении выполняется **обновление** существующей лицензии;

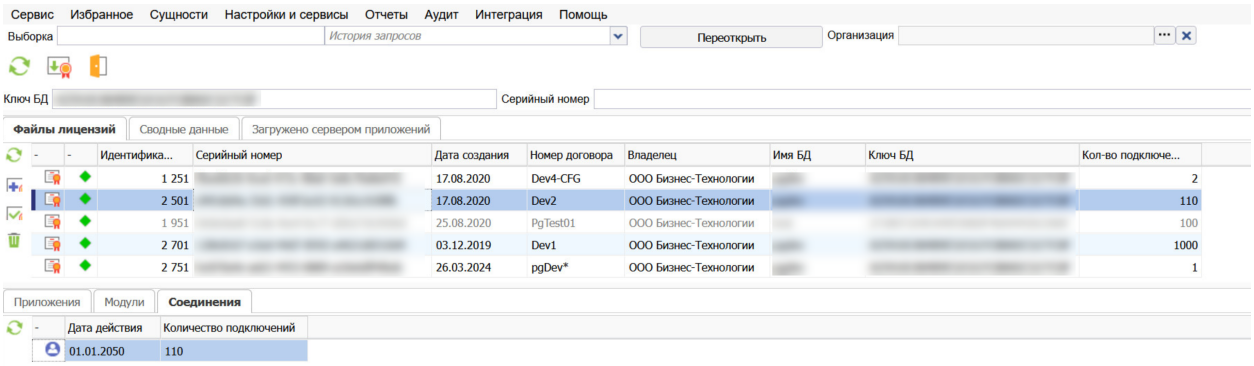
- при отсутствии совпадений выполняется **добавление** новой лицензии.

Важно

Если дата окончания действия лицензии установлена на 01.01.2050, лицензия считается бессрочной.

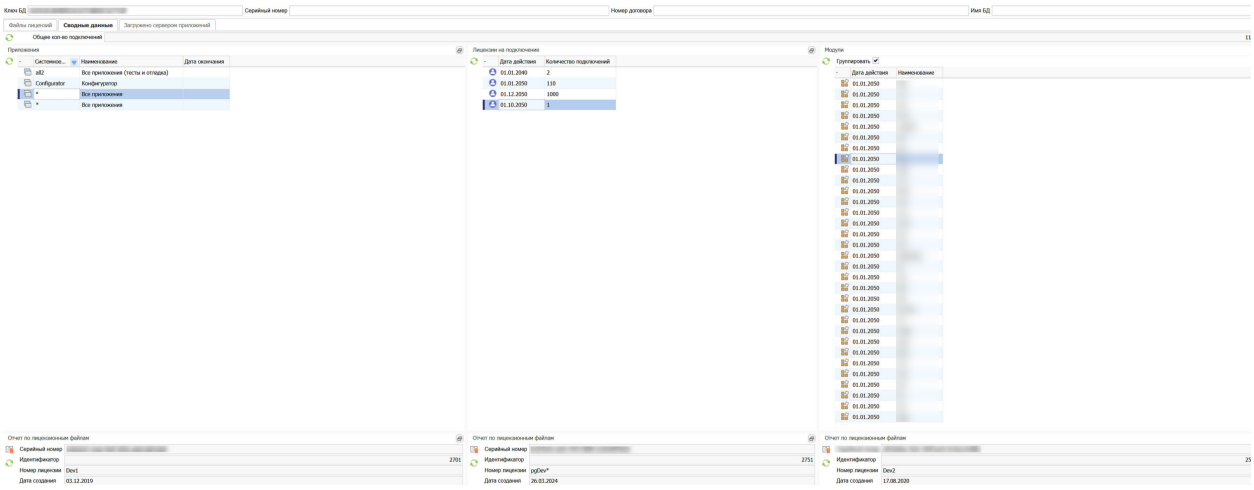
Важно

При наличие у лицензии даты окончания, система начнет процедуру предупреждения об истечение срока действия за 30 дней.



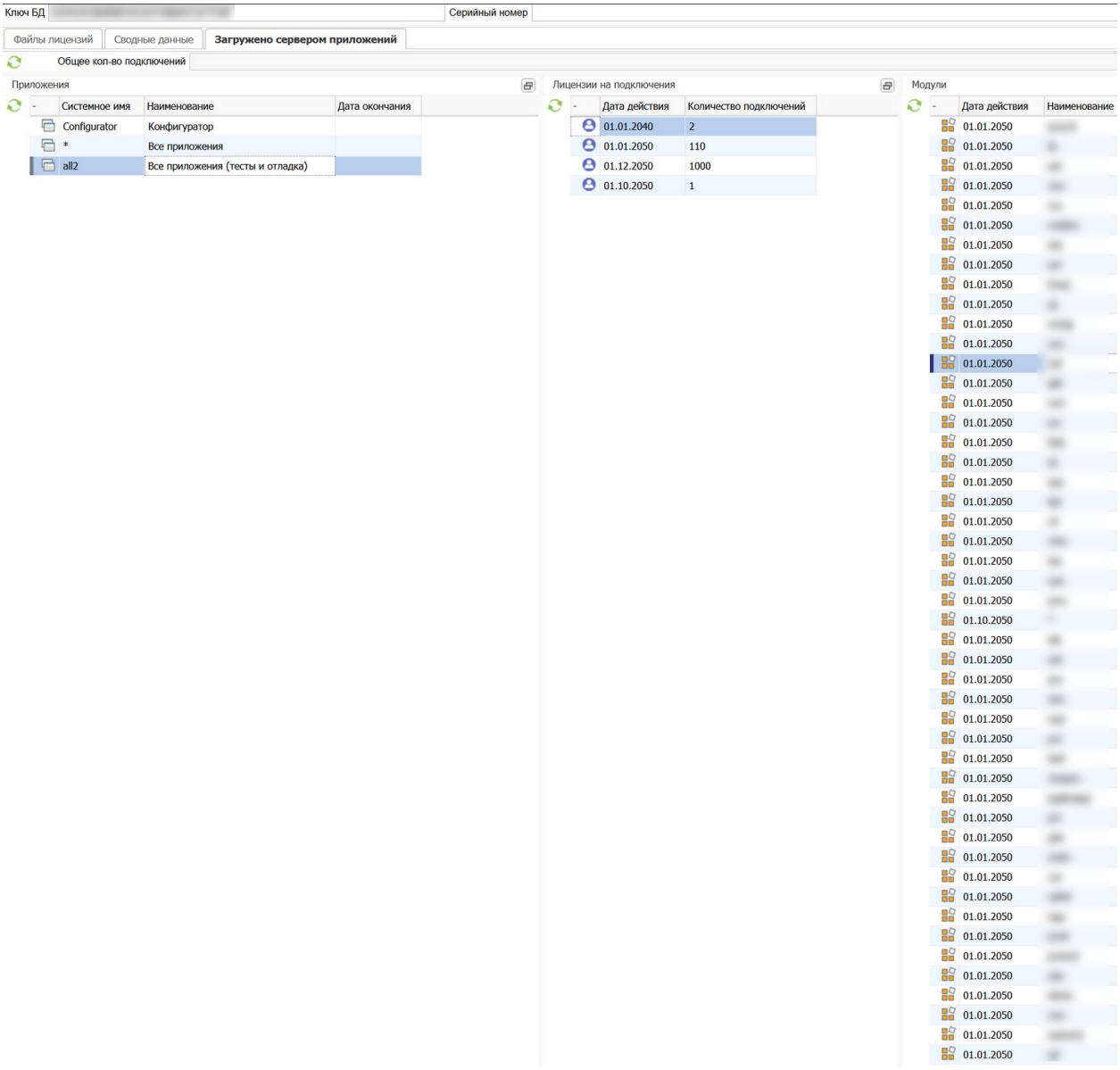
Вкладка «Сводные данные»

Представляет агрегированную информацию по всем загруженным файлам лицензий в удобном, сводном формате.



Вкладка «Загружено сервером приложения»

Отображает процесс чтения модулей из файла лицензии.



8.4 Примечания и рекомендации

- При необходимости переноса сервера СУБД необходимо заранее уведомить контактное лицо тех. поддержки или руководителя проекта для плановой выдачи нового файла лицензии.
- Храните лицензионные файлы и файлы запросов в надёжном месте для оперативного восстановления в случае необходимости.

9 Логирование сервера приложений

9.1 Общий обзор

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

9.2 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.

9.3 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>
    </appender>


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

9.4 Конфигурация логов для HaProxy

Конфигурация находится в /etc/haproxy/haproxy.cfg

```
global
    log /dev/log        local0 debug   # debug - самый подробный уровень
        log /dev/log        local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin
        stats timeout 30s
        user haproxy
        group haproxy
        daemon

        # Default SSL material locations
        ca-base /etc/ssl/certs
        crt-base /etc/ssl/private

        # See: https://ssl-config.mozilla.org/#server=haproxy&server-version=2.0.3&
↪config=intermediate
        ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-
↪SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
↪POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-
↪SHA384
        ssl-default-bind-ciphersuites TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_
↪CHACHA20_POLY1305_SHA256
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

defaults
    option tcplog
    option dontlognull
    option log-health-checks      # Логирует результат health checks
    option abortonclose           # Логирует обрыв соединений
    option dontlog-normal
    option log-separate-errors
    log global
    mode http
    #option httpchk
    option httplog
    timeout connect 5000
    timeout client  50000
    timeout server  50000
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
    errorfile 408 /etc/haproxy/errors/408.http
    errorfile 500 /etc/haproxy/errors/500.http
    errorfile 502 /etc/haproxy/errors/502.http
    errorfile 503 /etc/haproxy/errors/503.http
    errorfile 504 /etc/haproxy/errors/504.http



frontend http-frontend
    bind *:8080
    default_backend http-backend

backend http-backend
    server s1 192.168.200.16:8080 check
```

Команда для просмотра логов:

```
journalctl -u haproxy -f
```

Пример логов

0/0 «GET /login/ing/glow.ong HTTP/1.1“ 09:32:53 haproxy haproxy [44489]: 192.168.200.1:54520 [04/Aug/2025:09:32:38.821] http-frontend http-backend/s1 0/0/1212/834/14706 «GET /view/fonts/inter/inter-regular-variable.ttf HTTP/1.1» 200 121659 - - CD– 6/6/2 200 135559 - - CD– 6/6/

cD - Клиент не отправил и не подтвердил никаких данных, и в конечном итоге время ожидания клиента истекло.

SC - Сервер явно отклонил TCP-соединение.

PC - Прокси-сервер отказался устанавливать соединение с сервером, поскольку при попытке подключения было достигнуто ограничение на сокет процесса.

10 Установка клиентского ПО

10.1 Одиночная установка

Установка в Windows

Установка плагина

1. Скачайте архив с образом
2. Распакуйте архив в удобную папку.
3. Запустите файл `install.cmd` **от имени администратора**:
   - Щёлкните правой кнопкой мыши по `install.cmd`.
   - Выберите пункт «Запуск от имени администратора».
   - Дождитесь завершения установки.

Установка расширения

Для Google Chrome

1. Откройте в Chrome страницу: `chrome://extensions/`
2. Включите «Режим разработчика» (переключатель в правом верхнем углу страницы).
3. Нажмите кнопку «Загрузить распакованное расширение».
4. В открывшемся окне проводника выберите папку: `C:\Program Files\GlobalERP\Gs3-browser-cmd\extension`
5. Убедитесь, что расширение появилось в списке и включено.

   > Важно: файлы расширения должны находиться **строго** в папке `C:\Program Files\GlobalERP\Gs3-browser-cmd\extension`, иначе загрузка может завершиться ошибкой.

Для Yandex Browser

1. Откройте страницу настроек: `browser://tune/`
2. Перетащите папку `C:\Program Files\GlobalERP\Gs3-browser-cmd\extension` в окно настроек браузера.
3. Убедитесь, что расширение появилось в списке и включено.

Установка в Alt Linux

Установка плагина

1. Откройте терминал.
2. Обновите список пакетов:

```
sudo apt-get update
```

3. Установите плагин:

```
cd /tmp
sudo wget https://repo.global-system.ru/artifactory/common/ru/bitec/Gs-browser-plugin-
↪linux/gs3-browser-cmd-0.19.6-alt.x86_64.rpm
sudo apt-get install ./gs3-browser-cmd-0.19.6-alt.x86_64.rpm
```

Установка расширения

Для Chromium

1. Откройте страницу расширений: `chrome://extensions/`
2. Включите «Режим разработчика» (переключатель в правом верхнем углу).
3. Нажмите кнопку «Загрузить распакованное расширение».
4. Выберите директорию: `/lib/gs3-browser-cmd/extension/` (эта папка создаётся при установке RPM-пакета).
5. Убедитесь, что расширение появилось в списке и включено.

Для Yandex Browser

1. Откройте страницу настроек браузера: `browser://tune/`
2. Перетащите в окно настроек директорию: `/lib/gs3-browser-cmd/extension/`
3. Подтвердите установку и проверьте, что расширение активно.

Удаление плагина

```
sudo apt-get remove gs3-browser-cmd
```

10.2 Установка групповыми политиками

Установка браузера

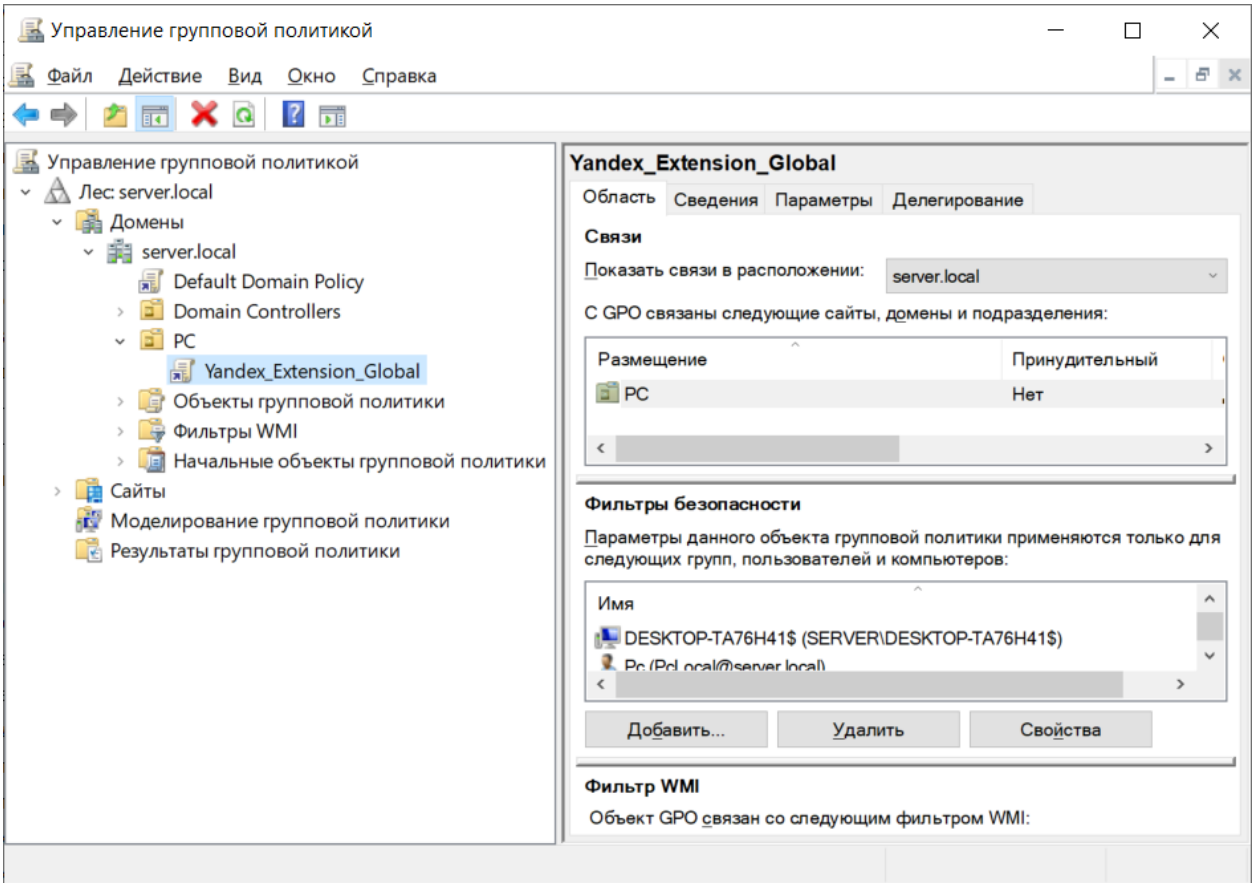
В примерах установки будет использоваться образ инсталлятора корпоративного Яндекс Браузера.

Скачайте инсталлятор корпоративного Янедкс Браузера или соберите собственный дистрибутив через конструктор. Переместите инсталлятор в обще доступную сетевую директорию. Убедитесь, что общая папка доступна на компьютерах пользователей.

Скачайте и разместите файл ADMX в папке «C:\Windows\PolicyDefinitions».

Скачайте и разместите файл ADML (на русском или английском языке) в папке «C:\Windows\PolicyDefinitions<язык_ОС>».

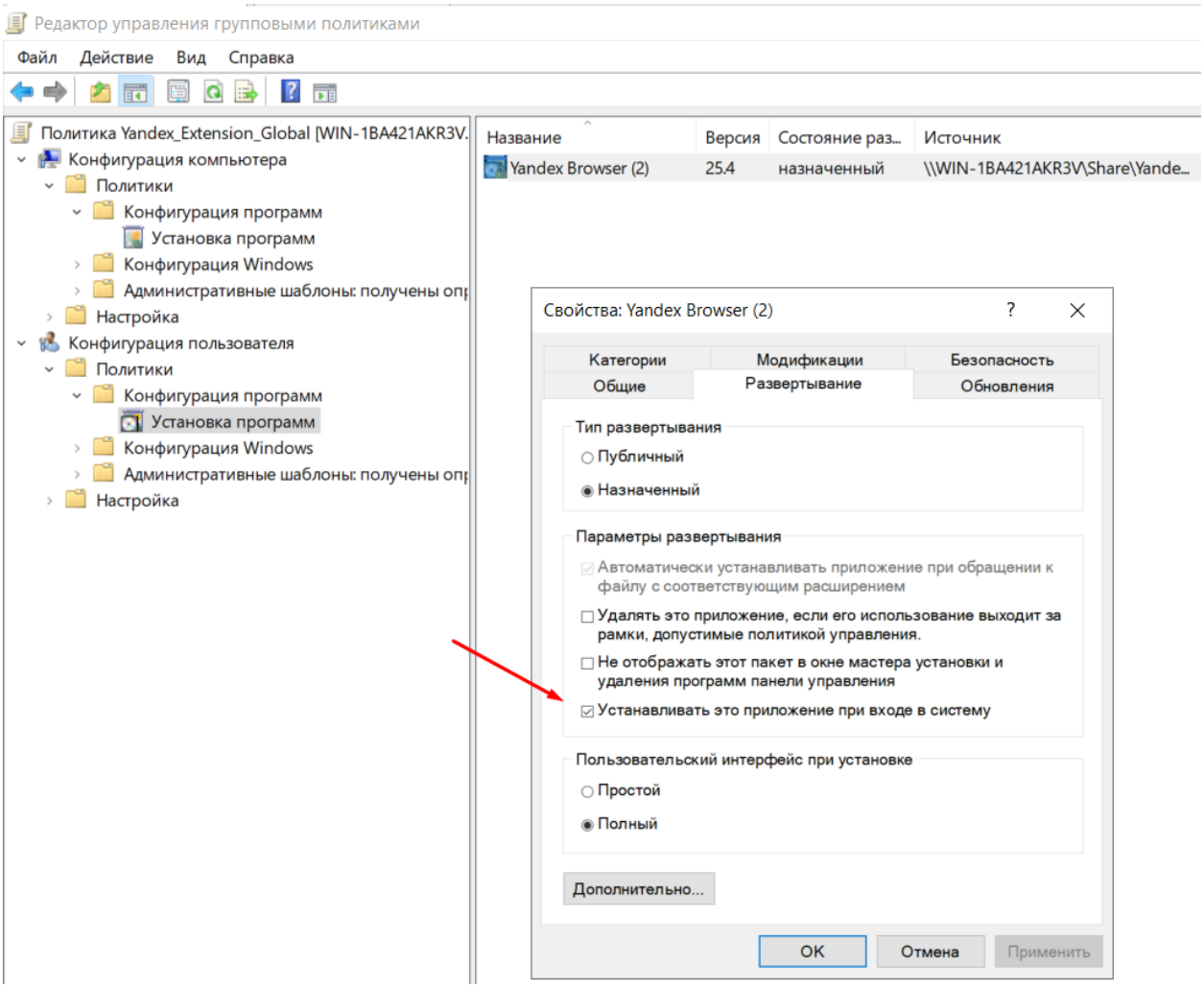
Перейдите в «Панель управления\Система и безопасность\Администрирование». Откройте «Управление групповой политикой». Перейдите в «Лес -> Домены -> Домен -> PC» и создайте объект групповой политики. Добавьте в фильтры безопасности нужные группы пользователей и пк.



В примере используется Windows Server 2019.

Нажмите ПКМ по новой политике -> Изменить...

Перейдите в «Конфигурация польлзователя -> Политики -> Конфигурация программ -> Установка программ». Нажмите ПКМ «Пакет -> Создать... » и выберите инсталлятор браузера в обще доступной папке «YandexBrowserStock.msi». Тип развертывания «Назначенный». В свойствах выберите «Устанавливать это приложение при входе в систему».



Запустите CMD от имени администратора. Введите `gpupdate /force`. После перезапуска пк в домене браузер установится автоматический.

Установка расширения

Установите веб-сервер для раздачи файлов с расширением и xml манифеста обновления. В качестве веб сервера рекомендуется Apache, в примере используется open source сервер HFS. Страница проекта на GitHub.

Загрузите crx файл из репозитория вендора.

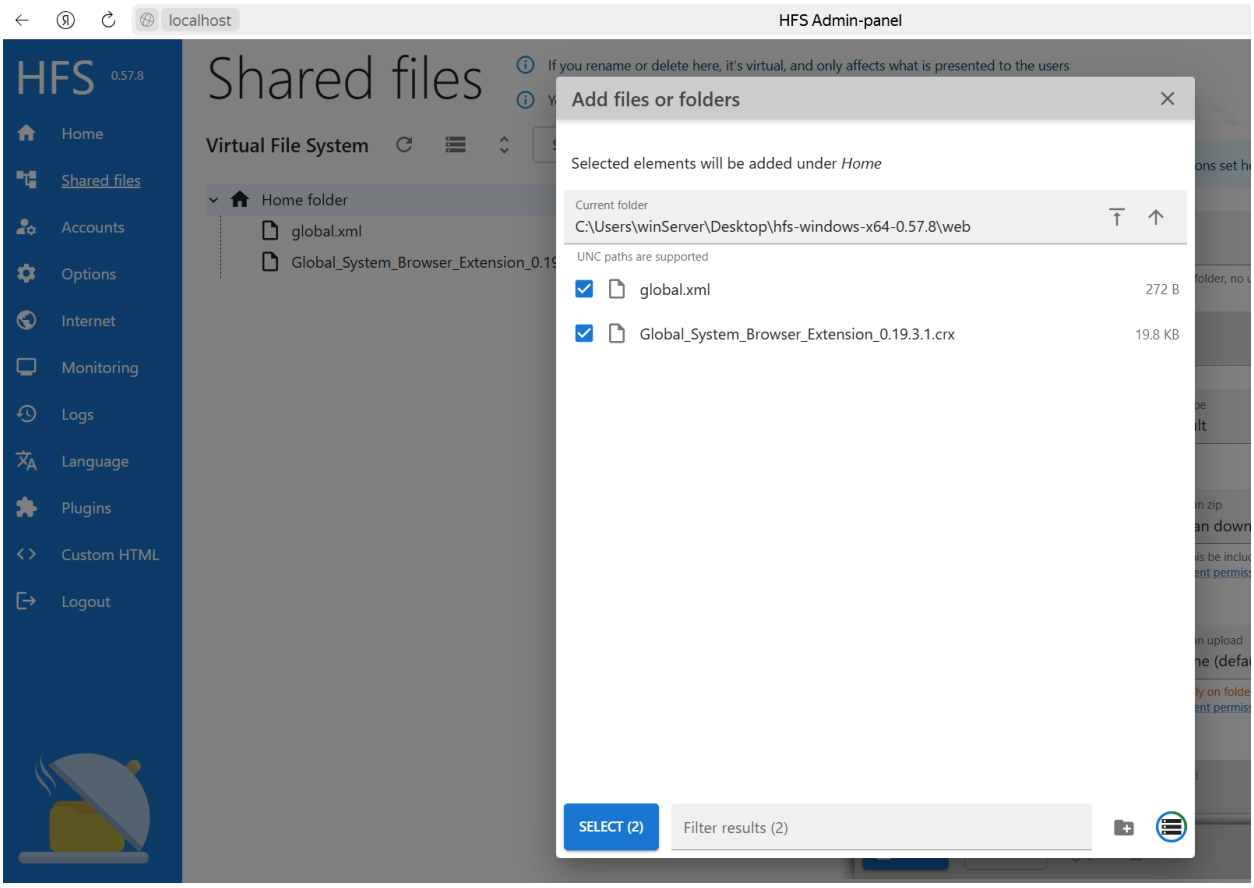
Скачайте xml файл манифеста для обновления update.xml. Внутри файла прописывается ID расширения и ссылка на crx файл. Версия crx файла должна соответствовать указанному в манифесте. Образец:

Примечание

Замените ссылку http://192.168.0.1/ на свой веб-сервер.

Зайдите в админ панель веб-сервера: http://localhost/~/admin/. Перейдите в ме-

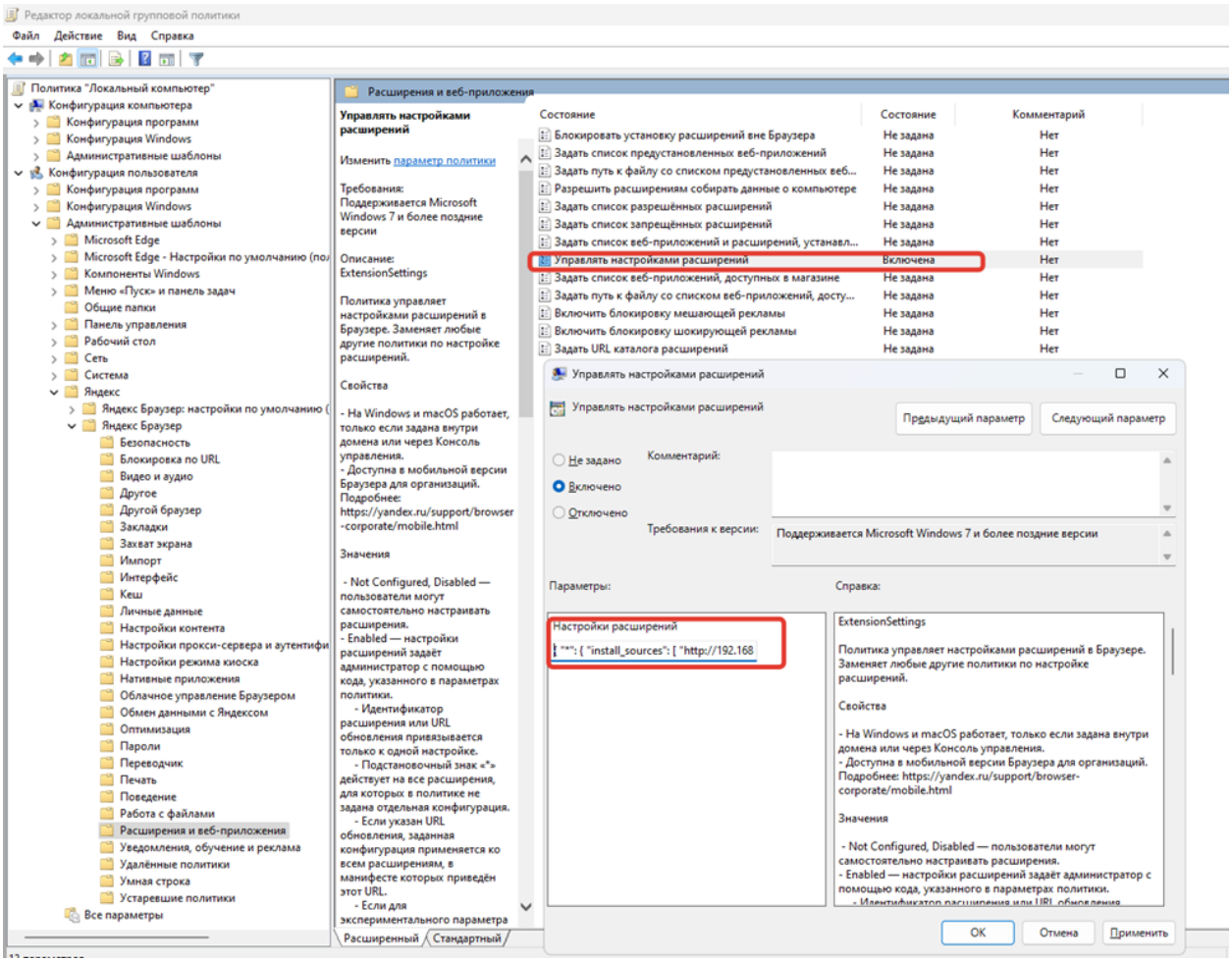
ню «Shared Files -> Add -> From Disk» и загрузите оба файла crx и xml:



Настройка групповой политики

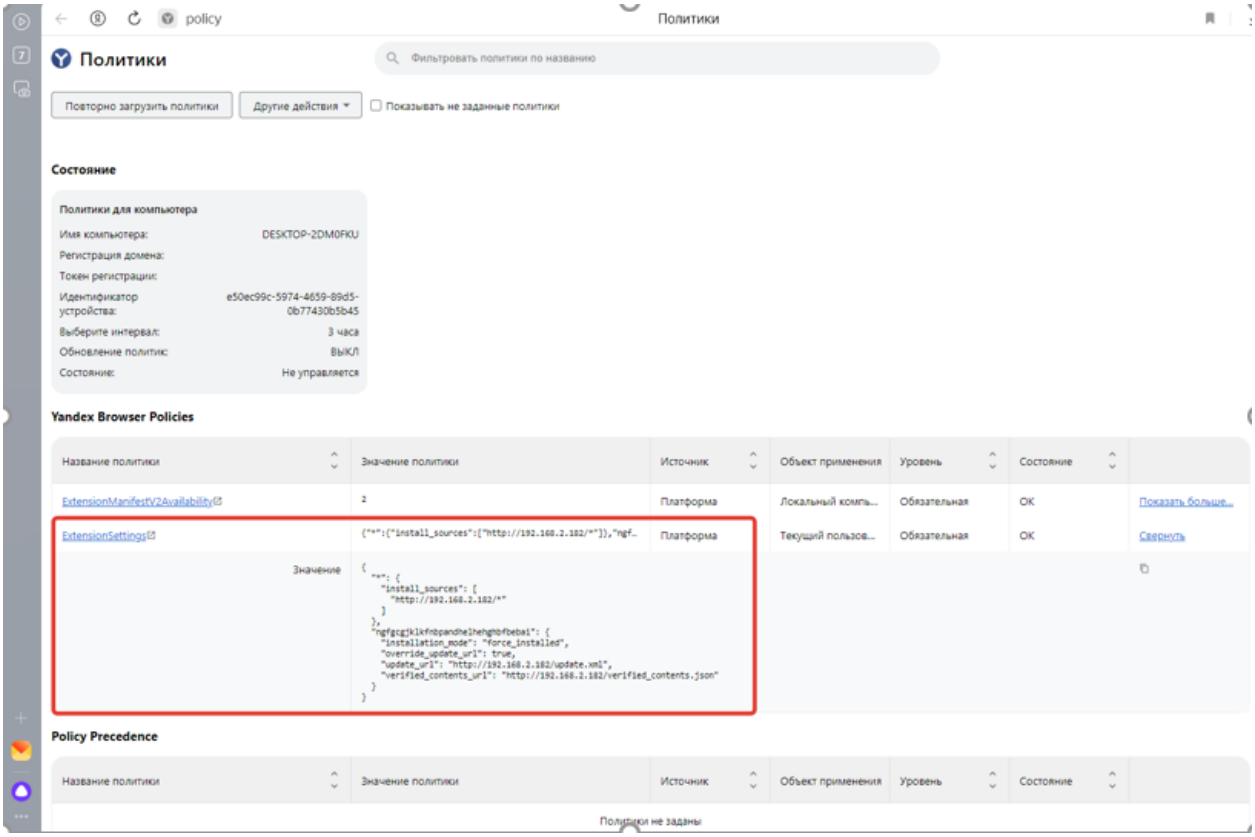
Настроить политику extension-settings, указать следующее значение (адрес сервера, имя файла, ид расширения указать актуальные)

```
{ "*": { "install_sources": [ "http://192.168.2.182/*" ] },
↪"ngfgcgjklkfnbpandhelhehghbfbebai": { "installation_mode": "force_installed",
↪"override_update_url": true, "update_url": "http://192.168.2.182/update.xml" } }
```

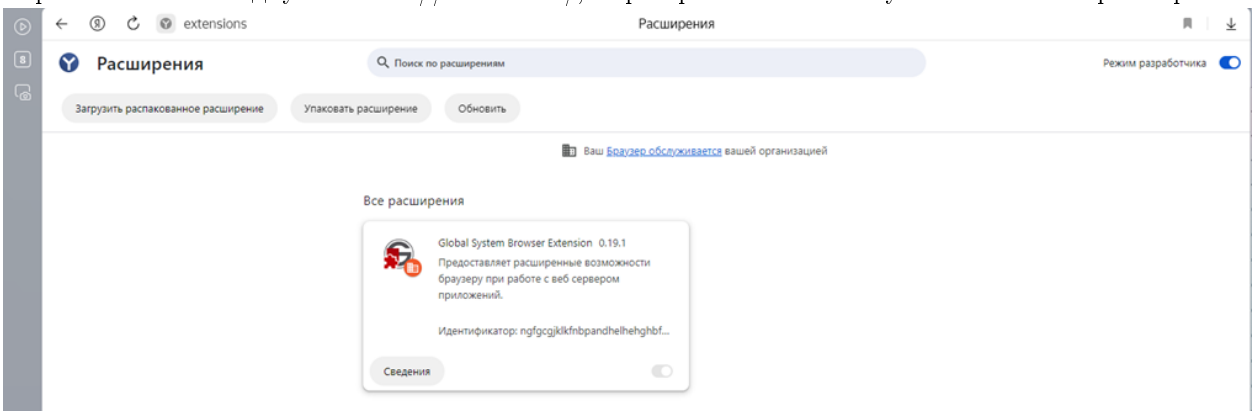


После изменений введите `gpupdate /force` в CMD.

Проверка применения политик. Открыть страницу browser://policy/, нажать кнопку «Повторно загрузить политики» Убедиться, что политики успешно прочитаны из домена и применены к браузеру



Перейти на закладку browser://extensions/, проверить наличие установленного расширения.



11 Конфигурация

11.1 global3.config.xml

База данных

Более подробная информация.

Настройки по умолчанию:

```
        <database alias="PGTEST" driver="org.postgresql.Driver" schema="PUBLIC"
                                            url="jdbc:postgresql://pgDevDB:5432/test"␣
↪connectionType="proxyShared" authenticationType="btk">
                                <users>
                                        <user name="test" password="test"/>
                                </users>
                        <metaManager
                                mode="Xml"
                                defaultNamespace="ru.bitec.app.btk"
                                sbtName="pgtest_sbt"
                        />
                        <eclipseLink
                                persistenceUnitName="pgdev"
                        />
                </database>
```

Пример настроек:

Поменяйте переменные alias, url, name, password на свои.

```
        <database alias="local" driver="org.postgresql.Driver" schema="PUBLIC"
                                            url="jdbc:postgresql://192.168.24.36:5432/test
↪" connectionType="proxyShared" authenticationType="btk">
                                <users>
                                        <user name="AdminDB" password="PasswordDB"/>
                                </users>
                        <metaManager
                                mode="Xml"
                                defaultNamespace="ru.bitec.app.btk"
                                sbtName="pgtest_sbt"
                        />
                        <eclipseLink
                                persistenceUnitName="pgdev"
                        />
                </database>
```

SBTs

Более подробная информация.

Значения по умолчанию:

```
    <sbts>
                <sbt name="pgdev-sbt"
                     sourceMode="Jar"
                     jarFolder="%G3_HOME%/application/applib/"
                     binaryFolder="%G3_HOME%/application/applibBin"
                />
    </sbts>
```

Квоты

Более подробная информация.

Настройки по умолчанию:

```
    <quotas>
        <server maxOldGenMemory="0" maxEdenSpaceMemory="80%" raiseError="true"/>
        <session maxUiCellCount="5M" maxFormCount="10" raiseError="true"/>
        <transaction maxTxCellCount="5M" raiseError="true"/>
    </quotas>
```

Рекомендуемые настройки:

```
    <quotas>
        <server maxOldGenMemory="80%" maxEdenSpaceMemory="80%" raiseError="true"/>
        <session maxUiCellCount="5M" maxFormCount="10" raiseError="true"/>
        <transaction maxTxCellCount="1M" raiseError="true"/>
    </quotas>
```

Безопасность

Более подробная информация.

Настройки данных по умолчанию:

```
    <security>
        <loginDefaults user="admin" password="admin" dataBase="DEVBTK"/>
        <users>
            <user name="admin" password="admin" roles="ssh,http-monitor"/>
            <user name="system" password="system" roles="system"/>
        </users>
    </security>
```

11.2 Логирование

Настройка осуществляется в конфигурационных файлах xml с префиксом logback-, расположенных в директории конфигурации дистрибутива:

```
../
  ├── admin - скрипты для сетевых администраторов
  ├── application - образы решений
  |   └── config - конфигурационные файлы сервера приложений
  |       ├── global3.config.xml
  |       ├── infinispan.config.xml
  |       ├── jgroups.config.xml
  |       ├── logback-LoggerContext.xml
  |       ├── logback-LoggerContext-ext.xml
  |       ├── logback-LoggerContext-session.xml
  |       └── logback-LoggerContext-session-ext.xml
  ├── logs - логи
```

logback-LoggerContext.xml - файл конфигурации логирования уровня сервера приложений, для постоянных настроек, определяемых при установке сервера приложений.

logback-LoggerContext-ext.xml - файл конфигурации логирования уровня сервера приложений, для дополнительных временных настроек.

logback-LoggerContext-session.xml - файл конфигурации логирования уровня пользовательских сессий, для постоянных настроек, определяемых при установке сервера приложений.

logback-LoggerContext-session-ext.xml - файл конфигурации логирования уровня пользовательских сессий, для дополнительных временных настроек.

Документация по Logback.