ООО «ПОЛЕТ»
ИНН 7730582675 КПП 773301001 ОГРН 1087746610896
125373, Москва, Походный проезд д. 14, цок.эт, пом.I, ком .3, тел: 8 (967) 104-01-09

Retail Suite.Global
Инструкция по установке

Оглавление

1 Установка модуля Retail Suite.Global ERP

Условия лицензирования модуля«RS.ERP»:

- Система лицензирована по количеству одновременно подключенных компьютеризированных рабочих мест фронт-офиса (типы пользователей: POS-терминалы) в одной локальной сети.
- Система лицензирована по количеству одновременно работающих пользователей (компьютеризированных рабочих мест) бэк-офиса.
- Система работает как распределенная сетевая только при наличии минимум одного модуля «Магазин» и модуля «Центр» (или «Региональный Центр»).
- Система защищена электронной лицензией.
- Для расширения количества пользователей бэк-офиса необходимо приобретать лицензии на дополнительные рабочие места.
- Для расширения количества подключений кассовых мест фронт-офиса необходимо приобретать лицензии на дополнительные POS
- Для выполнения различных регламентых задач поставляется отдельная техническая лицензия. Работа пользователей с использованием данной лицензии запрещена.

Компания «ПОЛЕТ» не несет ответственности за противоправные действия третьих лиц в случае продажи последними контрафактных (нелицензионных) версий модуля «RS.ERP».

Любое изменение продукта конечным пользователем приведет к невозможности его автоматического обновления.

1.1 Установка менеджера лицензий

В системе « RS.ERP» реализована защита от нелегального копирования с использованием электронных лицензий.

Взаимодействие с электронными лицензиями выполняется продуктом «RS.E-License». Для установки «RS.E-License» воспользуйтесь документацией на программный продукт.

1.2 Установка системы управления

Перед установкой «RS.ERP» на Вашем компьютере должна быть установлена плафторма «1С:Предприятие». Перед запуском установки все активные программы «1С:Предприятие» должны быть закрыты.

Для установки «RS.ERP» под операционной системой Windows запустите файл установки из комплекта дистрибутива.

Выберите пункт установки «RS.ERP» и укажите каталог для установки продукта.

Внимательно прочитайте текст лицензионного соглашения. Для продолжения установки необходимо принять соглашение.

После завершения установки запустите платформу «1С:Предприятие» и добавьте новую информационную базу.

Для установки под операционной системой Linux воспользуйтесь документацией https://its.1c.ru/db/metod8dev#content:5953:hdoc и https://its.1c.eu/db/metod8dev#content:5979:hdoc

1.3 Запуск программы

Для начала работы с продуктом запустите платформу « 1С:Предприятие» и выберите информационную базу. Выберите имя пользователя из списка.

Для корректной работы продукта при первом запуске введите номер лицензии.

2 Установка модуля Retail Suite.Global ShelfSpace

2.1 Условия лицензирования

Модуль «RS.ShelfSpace» может поставляться в вариантах поставки:

- RS.ShelfSpace Центр – продукт для автоматизации задач мерчандайзинга в центральном офисе торговой сети. Продукт позволяет осуществлять работу с планограммами нескольких магазинов одновременно.
- RS.ShelfSpace StoreBackOffice – продукт для автоматизации задач мерчандайзинга в магазинах торговой сети. StoreBackOffice позволяет получать планограммы из центра и осуществлять работу с мобильными устройствами.
  Дополнительно приобретаются лицензии на расширения количества магазинов и на работу с мобильными устройствами.
- RS.ShelfSpace Mobile App – приложение для доступа персонала магазина к планограммам торгового оборудования с мобильного устройства.
- RS.ShelfSpace Store Portal – web-портал для доступа персонала магазина к планограммам торгового оборудования.

Условия лицензирования модуля «RS.ShelfSpace»:

- Продукт лицензирован по количеству магазинов, порталов и количеству мобильных устройств.
- Для расширения количества используемых магазинов и порталов необходимо приобретать лицензии на дополнительные магазины и порталы.

Компания правообладатель не несет ответственности за противоправные действия третьих лиц в случае продажи ими контрафактных (не лицензионных) версий программного продукта.

Любое изменение продукта конечным пользователем приведет к невозможности его автоматического обновления.

2.2 Установка системы защиты

В модуле «RS.ShelfSpace» реализована защита от нелегального копирования с использованием электронных лицензий.

Взаимодействие с электронными лицензиями выполняется продуктом «RS.E-Licenses». Для установки «RS.E-Licenses» воспользуйтесь документацией на программный продукт.

2.3 Установка модуля

Перед установкой «RS.ShelfSpace» на Вашем компьютере должна быть установлена плафторма «1С:Предприятие». Перед запуском установки все активные программы «1С:Предприятие» должны быть закрыты.

2.3.1 Установка «RS.ShelfSpace»

Для установки «RS.ShelfSpace» под операционной системой Windows запустите файл установки.

Выберите каталог для установки продукта. Внимательно прочитайте текст лицензионного соглашения. Для продолжения установки необходимо принять соглашение. Выберите вид установки: полную или частичную и завершите установку.

Далее запустите платформу «1С:Предприятие» и добавьте новую информационную базу.

Для установки под операционной системой Linux воспользуйтесь документацией https://its.1c.ru/db/metod8dev#content:5953:hdoc и https://its.1c.eu/db/metod8dev#content:5979:hdoc

2.3.2 Установка «RS.ShelfSpace Store BackOffice»

Запустите программу установки «RS.ShelfSpace Store BackOffice».

Выберите каталог для установки продукта и завершите установку.

2.3.3 Установка RS.ShelfSpace Mobile App

Установите мобильное приложение RS.ShelfSpace Mobile App, скопировав файл установки приложения и запустив его на мобильном устройстве.

2.3.4 Публикация информационной базы «Магазин» на web-сервере

Обмен данными между информационной базой «StoreBackOffice» и информационной базой мобильного приложения осуществляется через http-сервисы, поэтому перед началом работы необходимо выполнить следующие настройки:

1. База данных «StoreBackOffice» должна быть опубликована на web-сервере (Apache или IIS). Должны быть опубликованы все сервисы.

Для публикации Web-сервиса необходимо:

1. Запустить 1С:Предприятие в режиме конфигуратора.
2. Создать нового пользователя для работы с мобильным устройством, от имени которого будет происходить обмен данными.

Необходимо проверить, что у этого пользователя не установлен флаг *«Защита от опасных действий»*.

В ролях данного пользователя должны быть указаны «Администратор» и «Веб-сервис».

3. С помощью команды меню *Администрирование → Публикация на веб-сервере* опубликовать веб-сервис «ShelfSpaceMobile», заполнив необходимые поля и нажав на **«Опубликовать»**.

   - *Имя* – имя, по которому будет происходить обращение к базе данных на веб-сервере. Может состоять только из символов латинского алфавита.
   - *Веб-сервер* – выбирается из списка найденных на текущем компьютере веб-серверов. В нашем случае это Internet Information Services.
   - *Каталог* – физическое расположение каталога, в котором будут располагаться файлы виртуального приложения.
   - Флагами указать типы клиентов для публикации, а также указать возможность публикации Web-сервисов. В расположенной ниже таблице можно отредактировать список Web-сервисов, которые будут опубликованы, а также в столбце «*Адрес*» изменить синоним, по которому будет происходить обращение к данному Web-сервису.
   - Также для веб-сервера IIS есть возможность указать необходимость выполнения аутентификации на веб-сервере средствами ОС, установив соответствующий флаг.

Проверить работоспособность публикации можно, перейдя в браузере по пути к файлу публикации, который формируется следующим образом:
http://адрес сервера/имя публикации/ws/имя веб-сервиса?wsdl

Например:

http://192.168.1.63/merch-center/ws/ShelfSpaceMobil?wsdl

При попытке открытия файла публикации продукт попросит ввести имя пользователя веб-сервиса и пароль.

2.3.5 Настройка обмена данными между информационными базами

2.3.5.1 Настройка обмена в «RS.ShelfSpace»

Настройка обмена между «RS.ShelfSpace» и «RS.ShelfSpace Store BackOffice» производится на вкладке «Настройка обмена» (*Администрирование → Настройки системы*)

На вкладке «Параметры обмена» настраивается канал обмена. Предусмотрены следующие способы обмена:

- Файловый через каталоги;
- Web-сервисы;
- Файловый через FTP.

На вкладке «Планы обмена» настраиваются узлы обмена. Узлы обмена настраиваются по типам метаданным.

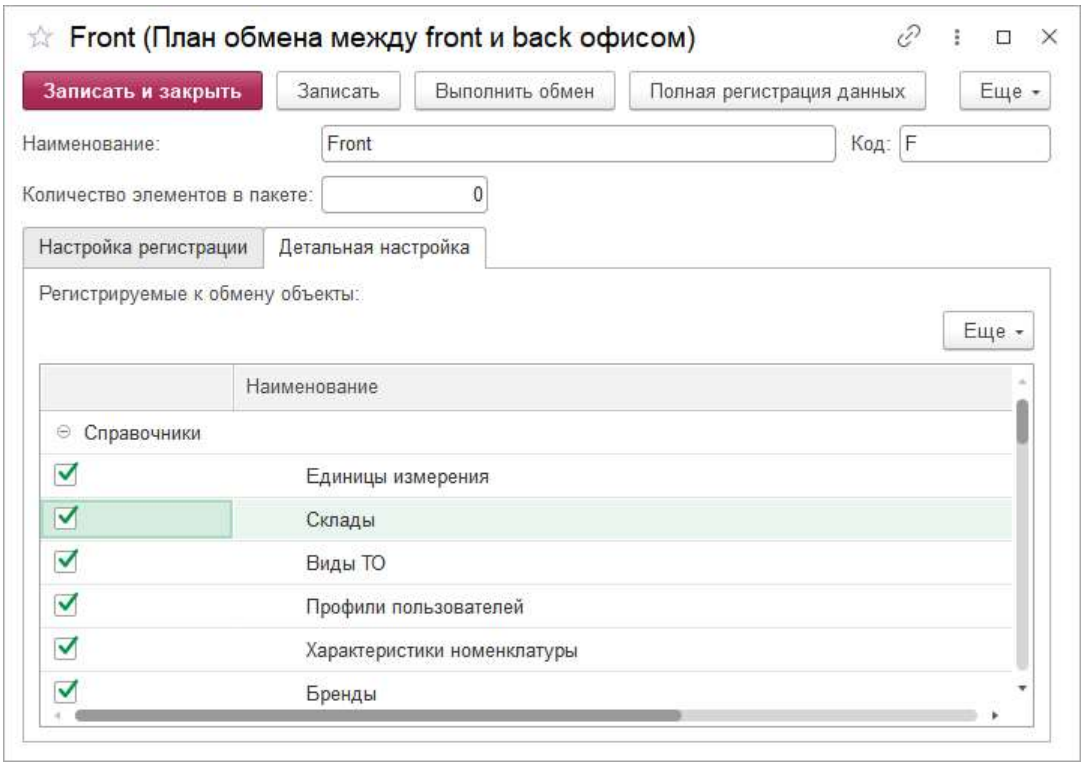
Существуют типы метаданных:

- *Мастер данные* – основные справочники и регистры;

- *Планограммы* – данные по торговому оборудованию;
- *Задания и сообщения* – сообщения и задания для мобильного приложения и фронт-офиса;
- *Статусы номенклатуры* – регистр статусы номенклатуры;
- *Цена номенклатуры* – регистр цены номенклатуры;
- *Выгружать позиции торгового оборудования* – если установлено значение «Да», то при выгрузке в StoreBackOffice будут формироваться картинки плана зала для каждого торгового оборудования. Если установлено значение «Нет», то картинка с планом зала формироваться не будет.

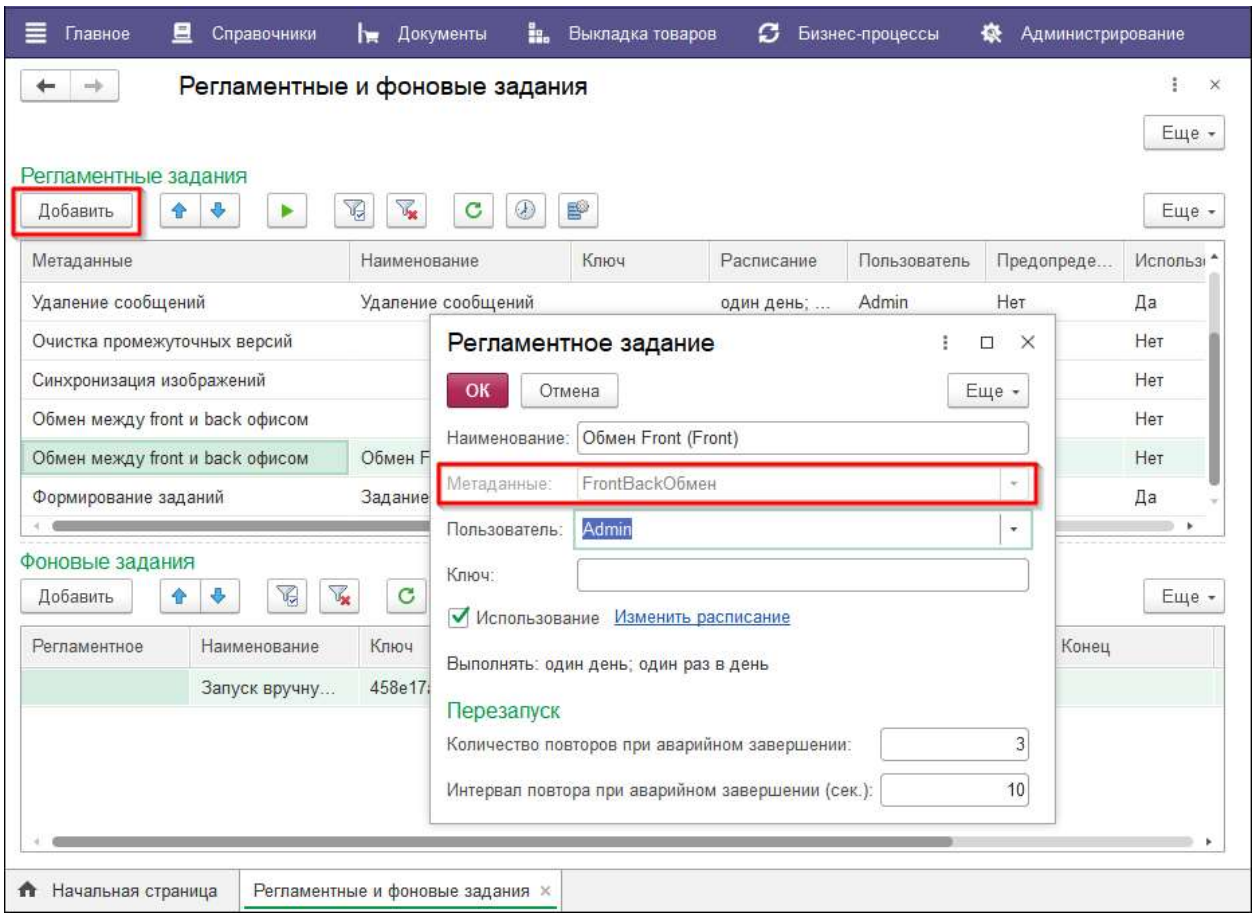
При настройке узла выберите объекты метаданных и нажмите **«Заполнить настройки регистрации»**.

На вкладке «Детальная настройка» указываются выгружаемые данные.



Для автоматического обмена нужно настроить регламентное задание со своим расписанием. Для перехода к консоли регламентных заданий выберите *Администрирование → Сервис → Консоль заданий*.

При создании регламентного задания в поле «Метаданные» выберите *«Обмен между front и back офисом»*.



По ссылке *«Изменить расписание»* необходимо задать периодичность обмена.

2.3.5.2 Выполнение настроек в «RS.ShelfSpace Store BackOffice»

«RS.ShelfSpace Store BackOffice» используется для выполнения настроек обмена данными с «RS.ShelfSpace Store Portal» и мобильным приложением «RS.ShelfSpace MobileApp».

Доступно выполнение настроек:

- Общих настроек;
- Настроек лицензий;
- Настроек обмена;
- Настроек списка структурных единиц;
- Настроек синхронизации.

Общие настройки

На вкладке «Общие настройки» задаются параметры:

- *Путь к каталогу с картинками товаров* – каталог с изображениями товаров (для отображения на планограммах торгового оборудования);
- *Путь к каталогу с картинками товаров URL* – каталог с изображениями товаров (для отображения на планограммах торгового оборудования);
- *Путь к файлу с логотипом организации* – путь к изображению логотипа, параметр указывается для отображения в печатных формах.

Лицензия

На вкладке «Лицензия» указывается лицензия продукта и параметры лицензии (количество лицензий портала и мобильного приложения).

Настройка обмена

Так же, как и для центрального офиса, необходимо выбрать способ обмена и заполнить параметры обмена. Настройка плана обмена для «RS.ShelfSpace Store BackOffice» не производится, он создается автоматически.

Из «RS.ShelfSpace Store BackOffice» выгружается ограниченный объем информации: задания, сообщения и отчеты. Для выполнения обмена необходимо нажать кнопку **«Выполнить обмен»**.

На вкладке «Настройка обмена» указывается web-сервис для обмена с порталом.

Список структурных единиц

На вкладке «Список структурных единиц» указываются структурные единицы, которые участвуют в обмене.

Настройки синхронизации

На вкладке «Настройки синхронизации» задается число объектов в пакетах обмена.

2.4 Запуск программы

Для начала работы с модулем запустите платформу «1С:Предприятие» и выберите информационную базу. Выберите имя пользователя из списка.

Для корректной работы продукта при первом запуске введите номер лицензии.

3 Установка модуля Retail Suite.Global BigData&AI

Программный комплекс Retail Suite.Global предназначен для автоматизации и цифровизации оперативной деятельности и управления процессами средних и крупных розничных предприятий (розничных сетей) на основе современных цифровых технологий Искусственного Интеллекта (Artificial Intelligence), Машинного Обучения (Machine Learning), систем хранения данных BigData. Retail Suite.BigData&AI – программный комплекс сбора и хранения аналитических данных из всех модулей Retail Suite.Global, расчета и хранения агрегатов данных и моделей Machine Learning (AI), а также предоставления возможности получения отчетности и аналитических данных во всех Модулях Системы, включая использование регулярных ML-данных в бизнес-процессах Модулей Системы.

Установка ПО Retail Suite.Global Bigdata&AI на сервер предполагается с использованием контейнеризации Docker.

Минимальные требования к стенду
CPU - 8
RAM - 16 Гб
DISK - 50 Гб
ОС – Ред ОС, Астра linux, Debian linux.
СУБД – PostgresPRO, PostgreSQL.

Перед установкой продукта необходимо установить ПО Docker по официальной инструкции к используемому дистрибутиву https://docs.docker.com/engine/install/

2. Авторизоваться в корпоративном docker-registry

```
echo "${REGISTRY_PASSWORD}" | docker login "${REGISTRY_HOST}" -u "${REGISTRY_USER}" --password-stdin
```

3. Загрузить на сервер, где планируется установить программное обеспечение Retail Suite.Business Intelligence, архив с необходимой версией релиза, предварительно скачанный в личном кабинете клиента по адресу lk.supsoft.ru. Архив дистрибутива имеет вид distribution-${RS.BI_VERSION}.zip

4. Переместить полученный архив в /opt, распаковать его и переименовать директорию, полученную в результате распаковки архива

```
mv distribution-${RS.BI_VERSION} /opt && \
cd /opt && \
unzip distribution-${RS.BI_VERSION} && \
mv distribution rs.bi-micro
```

5. Переименовать .env_template в .env

```
cd /opt/rs.bi-micro && mv .env_template .env
```

6. Внести соответствующие изменения в .env-файл. Ниже перечислены переменные, значения которых необходимо установить

POSTGRES_PASSWORD - пароль пользователя postgres. Все микросервисы используют пользователя postgres, при работе с базой данных
KEYCLOAK_ADMIN_PASSWORD - пароль admin-пользователя для доступа в консоль keycloak

KEYCLOAK_HOSTNAME - хостнейм или ip-адрес хоста, который будет ассоциирован с консолью keycloak.
API_GATEWAY_EXTERNAL_URL - URL по которому будет доступен портал.
SHELFSPACE_ADMIN_EMAIL - email admin-пользователя портала
METATRON_DB_USER - пользователь СУБД postgres
METATRON_DB_PASSWORD - пароль пользователя postgres
DISCOVERY_DATASOURCE_USERNAME - пользователь СУБД postgres
DISCOVERY_DATASOURCE_PASSWORD - пароль пользователя postgres
MDX_DATASOURCE_USERNAME -пользователь СУБД postgres
MDX_DATASOURCE_PASSWORD - пароль пользователя postgres

7. Пример .env-файла, в котором заполнены все обязательные переменные:

```
DOCKER_REGISTRY_HOST=registry.supsoft.ru
BI_VERSION=97.1.0
BI_UI_SOURCE=rs

API_GATEWAY_EXTERNAL_URL=test-docker

# SSL
SSL_ENABLED=true
SSL_TRUST_STORE_PASSWORD=password

# Portainer
PORTAINER_EXTERNAL_PORT=9100

# Postgres
POSTGRES_EXTERNAL_PORT=5432
POSTGRES_PASSWORD=password

# Keycloak
KEYCLOAK_HOSTNAME=test-docker
KEYCLOAK_EXTERNAL_PORT=9000
KEYCLOAK_ADMIN_PASSWORD=admin
KEYCLOAK_IMPORT_FILE=rs-bi-realm.json

# Microservices common
EUREKA_INSTANCE_PREFER_IP_ADDRESS=false
EUREKA_REGISTRY_EXTERNAL_PORT=8761
CONFIG_SERVER_EXTERNAL_PORT=8888
GATEWAY_EXTERNAL_PORT=8765

# OAuth
OAUTH_SERVER_URL=http://keycloak:8080
OAUTH_REALM_NAME=rs-bi
OAUTH_CLIENT_ID=oauth
OAUTH_CLIENT_SECRET=PeLRGb4QatijQjKi7DiicJrN6Xb1TWEs
OAUTH_RESOURCE_ACCESS_RESOURCE_NAME=oauth

# Discovery Legacy
```

```
DISCOVERY_LEGACY_VERSION=development-latest
DISCOVERY_LEGACY_SERVER_URL=http://discovery-legacy:8180
METATRON_JAVA_OPTS=$JAVA_OPTS -Xms2g -Xmx4g -
XX:MaxMetaspaceSize=512m
METATRON_ENV_PROFILES=local,postgres-default-db,logging-
file,scheduling,microservice,managements
METATRON_CACHE_PATH=cache
METATRON_SMTP_HOST=<SMTP-узел>
METATRON_SMTP_PORT=<SMTP-порт>
METATRON_SMTP_USERNAME=<SMTP-пользователь>
METATRON_SMTP_PASSWORD=<SMTP-пароль>

METATRON_DB_TYPE=postgres
METATRON_DB_SCHEMA=discovery_legacy
METATRON_DB_URL=jdbc:postgresql://db:5432/rs_bi?currentSchema=$METATRO
N_DB_SCHEMA
METATRON_DB_USER=postgres
METATRON_DB_PASSWORD=${POSTGRES_PASSWORD}

METATRON_MAIL_BASE_URL=test-docker
HADOOP_HDFS_BASE_URL=hdfs://<HDFS_URL>:8020
HADOOP_SPARK_LIVY_URL=http://<SPARK_LIVY_URL>:8999
HADOOP_SPARK_HISTORY_URL=http://<SPARK_HISTORY_URL>:18081
HADOOP_HIVE_HOSTNAME=<HIVE_URL>
HADOOP_HIVE_PORT=10500
HADOOP_HIVE_USER=hive
HADOOP_HIVE_PASSWORD=hive
HADOOP_THRIFT_HIVE_URL=thrift://<THRIFT_HIVE_URL>:9083

# Discovery API
DISCOVERY_DATASOURCE_URL=jdbc:postgresql://db:5432/rs_bi
DISCOVERY_DATASOURCE_SCHEMA_NAME=discovery_api
DISCOVERY_DATASOURCE_USERNAME=postgres
DISCOVERY_DATASOURCE_PASSWORD=${POSTGRES_PASSWORD}

# MDX API
MDX_DATASOURCE_URL=jdbc:postgresql://db:5432/rs_bi
MDX_DATASOURCE_SCHEMA_NAME=mdx_api
MDX_DATASOURCE_USERNAME=postgres
MDX_DATASOURCE_PASSWORD=${POSTGRES_PASSWORD}

# Logging
LOGGING_LOGSTASH_ENABLED=false
LOGGING_LOGSTASH_DESTINATION=localhost:5000
```

8. После подготовки .env-файла, необходимо запустить все микросервисы. Запуск выполняется с использованием docker-compose

```
cd /opt/rs.bi-micro && docker compose --env-file .env up -d
```

9. Статус запущенных контейнеров можно посмотреть через docker
cd /opt/rs.bi-micro && docker compose ps
Либо через portainer, доступный по адресу
http://${SERVER_IP}:9000
Во время первого доступа в portainer, необходимо будет создать admin-пользователя.
10. Для доступа к порталу удобно использовать nginx в качестве прокси.
Пример виртуального хоста для портала:

```
server {
  listen 80;

  server_name test-docker;
  charset utf-8;

  set $gateway 'http://127.0.0.1:7070';

  proxy_http_version 1.1;
  proxy_set_header Host $host;
  proxy_redirect .* $host;
  proxy_cookie_domain .* $host;

  proxy_connect_timeout 3600;
  proxy_send_timeout 3600;
  proxy_read_timeout 3600;

  default_type text/html;
  client_max_body_size 0;

  error_log /var/log/nginx/test-docker/error.log error;
  access_log /var/log/nginx/test-docker/access.log;

  location ^~ / {
    proxy_pass $gateway$request_uri;
  }

  location ^~ /stomp/ {
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_pass $gateway$request_uri;
  }

}
```

11. Просмотр логов контейнера
cd /opt/rs.bi-micro && docker logs {ID_CONTAINER}
12. Остановка стека докер контейнеров
cd /opt/rs.bi-micro && docker compose down --volume

Далее необходимо установить кластер HDP 3.0
Установка всех компонент кластера описана в документации HDP3 по ссылке

Подготовка окружений кластера, установка сертификатов

Зайти под пользователем root на сервер для установки сервиса Ambari

```
ssh-keygen
.ssh/id_rsa
.ssh/id_rsa.pub
cat id_rsa.pub >> authorized_keys
```

.ssh папку скопировать в /root и в home директорию user'a, под которым будем ставить (/home/aloha).

Изменить права доступа на:

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

На каждом сервере проверить подключение к другим серверам под обоими пользователями:

```
[root@apachai1 aloha]# ssh root@apachai4.apm.local ls
The authenticity of host 'apachai4.apm.local (10.10.8.149)' can't be established.
ECDSA key fingerprint is SHA256:HPaVr/HxhhjbrCCrfly5Hz50hMRp5pEsl9jbsxrM0Gc.
ECDSA key fingerprint is MD5:92:ee:a2:cd:07:a1:fd:98:83:93:b2:20:1b:fc:8f:29.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'apachai4.apm.local,10.10.8.149' (ECDSA) to the list of known hosts.
anaconda-ks.cfg
[root@apachai1 aloha]# ssh root@apachai4.apm.local ls
anaconda-ks.cfg
```

Проверить на каждом сервере, что пользователь получает root привеллегии без введения пароля:

```
[aloha@apachai1 ~]$ sudo su -
Последний вход в систему:Пт янв 25 11:28:45 MSK 2019с 10.14.3.89на pts/0
[root@apachai1 ~]#
```

Синхронизировать время на серверах, например утилитой ntp

```
[root@apachai1 aloha]# yum install -y ntp
...
Установлено:
 ntp.x86_64 0:4.2.6p5-28.el7.centos
Установлены зависимости:
 autogen-libopts.x86_64 0:5.18-5.el7 ntpdate.x86_64 0:4.2.6p5-28.el7.centos
Выполнено!
[root@apachai1 aloha]# systemctl enable ntpd
Created symlink from /etc/systemd/system/multi-user.target.wants/ntpd.service to
```

/usr/lib/systemd/system/ntpd.service.
[root@apachai1 aloha]#

Создать БД PostgresPRO\PostgeSQL

user= postgres

```
CREATE DATABASE rangerkms;
CREATE DATABASE rangeradmin;
CREATE DATABASE oozie;
CREATE DATABASE hive;
CREATE USER rangerkms WITH PASSWORD 'rangerkms';
CREATE USER rangeradmin WITH PASSWORD 'rangeradmin';
CREATE USER hive WITH PASSWORD 'hive';
CREATE USER oozie WITH PASSWORD 'oozie';
GRANT ALL PRIVILEGES ON DATABASE rangerkms TO rangerkms;
GRANT ALL PRIVILEGES ON DATABASE rangeradmin TO rangeradmin;
GRANT ALL PRIVILEGES ON DATABASE hive TO hive;
GRANT ALL PRIVILEGES ON DATABASE oozie TO oozie;
```

Даем доступ к подключению с соседних машин (в этом случае вообще всем внешним машинам)

-bash-4.2$ vim /var/lib/pgsql/data/postgresql.conf

listen_addresses = '*'

-bash-4.2$ vim /var/lib/pgsql/data/pg_hba.conf

host all all 0.0.0.0/0 md5

```
-bash-4.2$ psql
psql (9.2.24)
Введите "help", чтобы получить справку.
postgres=# SELECT pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 строка)
postgres=#
```

Добавляем драйвер postgresql-jdbc

find / -name postgresql-jdbc.jar 2>/dev/null

Если не нашли, ставим:
yum install postgresql-jdbc*
ls /usr/share/java/postgresql-jdbc.jar
chmod 644 /usr/share/java/postgresql-jdbc.jar

Регистрируем на сервере
ambari-server setup --jdbc-db=postgres --jdbc-driver=/usr/share/java/postgresql-jdbc.jar

Задаем предварительные настройки Ambari

Hive: hive.strict.managed.tables = false

Hdfs: hadoop.proxyuser.airflow.hosts=*, hadoop.proxyuser.airflow.hosts=*

YARN Queue Manager: Создаем очереди: aloha, etl.

Ranger. Создаем пользователя для схем баз данных.

Создать пользователя и группу: group=rsl, user=rsl
Добавить соответствующие права.

HDFS: Создать каталог user'а на hdfs:

hdfs dfs -mkdir /user/rsl
sudo -u hdfs hadoop fs -chown -R nch /user/rsl

Также нужно проверить права необходимых пользователей: aloha, airflow, hive, spark3

Spark. Создаем структуру проекта

GIT: Репозиторий проекта:
https://git.polet-it.ru/rs.analytics/analytics/-/tree/development/RSL/

Развернуть объекты базы данных с помощью скрипта \scripts\CreateDBObjects.scala,
из папки
https://git.polet-it.ru/rs.analytics/analytics/-/tree/development/RSL/scripts/ddl/rsl.

Для этого на стенде rsl6reg (10.14.8.33)

- *создать папку /home/aloha/scripts,*
- *скопировать содержимое папки RSL/scripts/ из репозитория,*
- *запустить spark-shell (spark v 3.4)*

spark-shell --master local --packages io.delta:delta-core_2.12:2.4.0,com.github.scopt:scopt_2.12:4.0.1 --conf "spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" --conf "spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"

- *запустить CreateDBObjects.scala с соответствующими параметрами:*

"-c", "demo","-p","/home/aloha/scripts/ddl/rsl/stage","-a","rsl","-s"," stage","-t","external"

(аналогично для tmp, dmart),

где demo – имя клиента,
/home/aloha/scripts/ddl/rsl/stage – путь к каталогу со скриптами создания объектов, rsl – название проекта,
stage –схема БД (stage, tmp, dmart)

Spark. Создаем и заполняем служебные объекты БД

Информацию по служебным объектам скопировать в виде csv-файлов в соответствующие нужным таблицам папки в схеме stage: hdfs:///user/aloha/stage/demo (valid_periods, time_periods, calendar, events, и др).
Затем загрузить их в tmp, dmart соответствующими скриптами.

Postgres. Создаем объекты БД

GIT: https://git.polet-it.ru/rs.analytics/analytics/-/tree/development/RSL/source

Создать необходимые объекты БД Postgres с помощью соответствующих скриптов.

Установка интеграционных компонент

Необходимо установить на сервер утилиты java 11 и karaf.

```
wget https://archive.apache.org/dist/karaf/4.4.1/apache-karaf-4.4.1.tar.gz
tar -xvzf ./apache-karaf-4.4.1.tar.gz
wget https://download.java.net/java/GA/jdk11/9/GPL/openjdk-11.0.2_linux-x64_bin.tar.gz
tar -xvzf ./openjdk-11.0.2_linux-x64_bin.tar.gz
```

edit bash_profile

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
<------>. ~/.bashrc
fi

# User specific environment and startup programs

export JAVA_HOME=/home/karaf/jdk-11
PATH=$PATH:$HOME/.local/bin:$HOME/bin:$JAVA_HOME/bin

export PATH
```

Запустить karaf в качестве system сервиса

Запустить karaf с поддержкой консоли

```
$KARAF_HOME/bin/karaf
```

Установить feature Service Wrapper

```
karaf@root()> feature:install service-wrapper
```

Подготовить Apache Karaf для регистрации в качестве сервиса/демона в операционной системе

```
karaf@root()> wrapper:install
```

Выключить karaf

```
karaf@root()> system:shutdown
```

Переключиться на root пользователя

```
sudo su -
```

Отредактировать karaf.service для запуска экземпляра karaf под пользователем karaf

```
vi $KARAF_HOME/bin/karaf.service

# Добавить в раздел [Service]

User=karaf

Group=karaf

# Сохранить
```

Зарегистрировать Apache Karaf в качестве сервиса/демона

```
systemctl enable $KARAF_HOME/bin/karaf.service
```

Запустить karaf

```
systemctl start karaf
```

Настроить репозитории в системе

```
$KARAF_HOME/etc/org.ops4j.pax.url.mvn.cfg
org.ops4j.pax.url.mvn.repositories= \
   https://archiva.polet-it.ru/repository/internal/@id=internal, \
   https://archiva.polet-it.ru/repository/snapshots/@id=snapshots@snapshots@noreleases, \
   https://archiva.polet-it.ru/repository/mirror2/@id=mirror2,
feature:install maven
feature:repo-add mvn:org.apache.activemq/artemis-features/2.20.0/xml/features
feature:repo-add mvn:org.ops4j.pax.jms/pax-jms-features/1.1.1/xml/features
feature:install artemis-jms-client artemis-core-client
feature:install pax-jms-config pax-jms-artemis pax-jms-pool-narayana
feature:repo-add mvn:io.hawt/hawtio-karaf/2.14.3/xml/features
feature:install hawtio
```

CXF Нужно добавить перед camel, для исправления неправильной зависимости

```
feature:repo-add  mvn:org.apache.cxf.karaf/apache-cxf/3.4.5/xml/features
feature:repo-add mvn:org.apache.camel.karaf/apache-camel/3.14.0/xml/features
// Camel и его модули ставить разными командами
feature:install camel
feature:install camel-jms camel-cxf camel-kafka

feature:repo-add mvn:com.retail/rsl-integration-features/1.1.0-SNAPSHOT/xml/features
```

Repository:

$KARAF_HOME/etc/org.ops4j.pax.url.mvn.cfg

Необходимо установить флаг:

org.ops4j.pax.url.mvn.defaultLocalRepoAsRemote = true

Доступ в Hawtio:

$KARAF_HOME/etc/users.properties

Необходимо раскомментировать:
karaf = karaf,_g_:admingroup
_g_\:admingroup = group,admin,manager,viewer,systembundles,ssh

Для корректного решения зависимостей.

.m2/repository/org/apache/camel/karaf/apache-camel/apache-camel-3.14.0-features.xml
Заменить:

1.

<bundle>mvn:org.codehaus.woodstox/stax2-api/3.1.4</bundle>
<bundle>mvn:org.codehaus.woodstox/woodstox-core-asl/4.4.1</bundle>
на
<bundle dependency="**true**">mvn:org.codehaus.woodstox/stax2-api/3.1.4</bundle>
<bundle dependency="**true**">mvn:org.codehaus.woodstox/woodstox-core-asl/4.4.1</bundle>

2.
На версии apache-cxf 4.3.5 не актуально:
<repository>mvn:org.apache.cxf.karaf/apache-cxf/(3,4]/xml/features</repository>
на
<repository>mvn:org.apache.cxf.karaf/apache-cxf/3.4.5/xml/features</repository>

Настройка:

$KARAF_HOME/etc/ com.retail.rsl6.cfg

Необходимо заполнить следующим содержимым:

```
kafka.bootstrap-servers=vmi250453.contaboserver.net:6667, \
                        vmi250624.contaboserver.net:6667, \
                        vmi250627.contaboserver.net:6667, \
                        vmi250631.contaboserver.net:6667

kafka.topic=rsl6_test
kafka.autoOffsetReset=latest
kafka.groupId=
kafka.repositoryFile=tmp/repo.dat
kafka.offsetRepository=Y
```

$KARAF_HOME/etc/ com.retail.rsl6.cfg

Необходимо заполнить следующим содержимым:

```
osgi.jdbc.driver.name=PostgreSQL JDBC Driver
dataSourceName=rsl6DS
databaseName=rsl6_stage
pool=hikari
user=?
password=?
url=jdbc:postgresql://rsl6reg.corp.local:5432/rsl6_stage?currentSchema=rsl6_test
hikari.maximumPoolSize=20
```

Установить и настроить Apache Airflow

Подготовка:

Установить Python 3.10 с зависимостями

Установить PostgreSQL\ PostgersPRO

Создать базу данных airflow

Установить RabbitMQ

Настроить кластерный режим RabbitMQ

Убедиться, что RabbitMQ демоны не запущены

Определить, какой узел будет MASTER.

На non-MASTER узлах бэкапим .erlang.cookie файл

```
mv /var/lib/rabbitmq/.erlang.cookie /var/lib/rabbitmq/.erlang.cookie.backup
```

Копируем файл “/var/lib/rabbitmq/.erlang.cookie” с MASTER узла на другие узлы и сохраняем по тому же пути.

Устанавливаем права на .erlang.cookie файл

```
chown rabbitmq:rabbitmq /var/lib/rabbitmq/.erlang.cookie
chmod 600 /var/lib/rabbitmq/.erlang.cookie
```

Заупускаем MASTER RabbitMQ Daemon

```
systemctl start rabbitmq-server.service
```

На каждом non-MASTER узле добавляем их в кластер и запускаем

```
# Запускаем сервис
systemctl start rabbitmq-server.service
# Останавливаем приложение
rabbitmqctl stop_app
# Добавляем текущую машину в кластер
rabbitmqctl join_cluster rabbit@{MASTER_HOSTNAME}
# например, где MASTER_HOSTNAME = vmi471363, т.е.
rabbitmqctljoin_cluster rabbit@vmi471363
# Запускаем приложение
rabbitmqctl start_app
# Проверяем статус
rabbitmqctl cluster_status
```

Проверка статуса должна вернуть нечто следующее:
Cluster status of node rabbit@{NODE_HOSTNAME} ...
#
[{nodes,[{disc,[rabbit@{MASTER_HOSTNAME},rabbit@{NODE_HOSTNAME}]}]},
#
{running_nodes,[rabbit@{MASTER_HOSTNAME},rabbit@{NODE_HOSTNAME}]}]

TODO: Setup a load balancer to balance requests between the the Nodes

Port Forwarding
Port 5672 (TCP) → Port 5672 (TCP)
Port 15672 (HTTP) → Port 15672 (HTTP)
Health Check
Protocol: HTTP
Ping Port: 15672
Ping Path: /

TODO: Point all processes to that LB

Установка Apache Airflow

MASTER (airflow-webserver, airflow-scheduler, airflow-worker)
SLAVE (airflow-webserver, airflow-worker)

На каждом узле (MASTER, SLAVE):

Залогиниться под airflow.

Активируем виртуальную среду

```
source ~/software/env3/bin/activate
```

Устанавливаем из PyPI с extras модулями:

```
Pip install "apache-airflow[async,celery,sqlite,postgres,microsoft.mssql,odbc,oracle,rabbitmq,apache.hdfs,apache.hive,apache.livy,apache.spark,apache.sqoop,common.sql,ftp,http,imap,jdbc,ssh]==2.7.1" --constraint "https://raw.githubusercontent.com/apache/airflow/constraints-2.7.1/constraints-3.10.txt"
```

Необходимые модули всегда можно установить по мере надобности

Создаем переменную среды AIRFLOW_HOME

```
echo 'export AIRFLOW_HOME=/opt/airflow' >> ~/.bash_profile
source ~/.bash_profile
```

Создать директорию ${AIRFLOW_HOME}

```
sudo mkdir /opt/airflow
sudo chown -R airflow:airflow /opt/airflow
```

Запускаем команду "airflow", чтобы создался конфигурационный файл в ${AIRFLOW_HOME} .

```
airflow config list --defaults
```

Меняем владельца директории ${AIRFLOW_HOME} на airflow

```
chown -R airflow:airflow /opt/airflow
```

Изменяем настройки {AIRFLOW_HOME}/airflow.cfg

```
cd /opt/airflow
vi airflow.cfg
```

на следующие:

```
[core]
# Для работы в кластерном режиме необходим CeleryExecutor
executor = CeleryExecutor

# Не загружаем примеры
load_examples = False

[database]
# Подключение к БД, созданной ранее
# sql_alchemy_conn = postgresql+psycopg2://<user>:<password>@<host>/<db>
sql_alchemy_conn = postgresql+psycopg2://airflow:airflow@vmi648694.contaboserver.net:5432/airflow

Пул подключений к БД
sql_alchemy_pool_size = 50
# Дополнительное количество подключений
sql_alchemy_max_overflow = 25

[celery]
# Устанавливаем URL брокера RabbitMQ (если используем CeleryExecutor)
# broker_url = amqp://guest:guest@{RABBITMQ_HOST}:5672/
broker_url = amqp://admin:E6fCKIfCDprG@vmi648694.contaboserver.net:5672/

# Устанавливаем ссылку на БД для сохранения результатов работы
result_backend
db+postgresql://airflow:airflow@vmi648694.contaboserver.net:5432/airflow
```

[smtp]
По необходимости.

Создать директорию для DAG:

```
mkdir /opt/airflow/dags
chown -R airflow:airflow /opt/airflow
```

На MASTER узле:

Инициализируем БД

```
airflow db init
```

Создаем пользователя интерфейса admin:

```
airflow users create \
        --username admin \
        --password <password> \
        --firstname Admin \
        --lastname Adminoff \
        --role Admin \
         --email info-test@hexone.ru
```

Настраиваем Systemd для запуска Airflow

```
# Скачиваем архив Airflow
cd /tmp/
wget https://github.com/apache/incubator-airflow/archive/2.7.1.zip

# Разархивируем
unzip 2.7.1.zip
# Переходим в директорию со скриптами
cd airflow-2.7.1/scripts/systemd/
# Обновляем содержимое файлов.
# Устанавливаем AIRFLOW_HOME
# Устанавливаем VENV_HOME
vi airflow
AIRFLOW_HOME=/opt/airflow
VENV_HOME=/home/airflow/software/env3
JAVA_HOME=/usr/jdk64/jdk1.8.0_112
HADOOP_CONF_DIR=/usr/hdp/current/hadoop-client/conf
HIVE_CONF_DIR=/usr/hdp/current/hive-client/conf

# Копируем airflow property файл в целевую директорию
cp airflow /etc/sysconfig/

# Обновляем содержимое airflow-*.service файлов
```

Устанавливаем User и Group значения пользователя и группы под которым будет запускать сервисы airflow
В нашем случае оставляем airflow
vi airflow-*.service

# Исправляем в ExecStart airflow-webserver.service
ExecStart=/bin/bash -c 'source ${VENV_HOME}/bin/activate && ${VENV_HOME}/bin/airflow webserver'
# Исправляем в ExecStart airflow-scheduler.service
ExecStart=/bin/bash -c 'source ${VENV_HOME}/bin/activate && ${VENV_HOME}/bin/airflow scheduler'
# Исправляем в ExecStart airflow-worker.service
ExecStart=/bin/bash -c 'source ${VENV_HOME}/bin/activate && ${VENV_HOME}/bin/airflow celery worker -q default'
# Исправляем в ExecStart airflow-flower.service
ExecStart=/bin/bash -c 'source ${VENV_HOME}/bin/activate && ${VENV_HOME}/bin/airflow celery flower'
# Исправляем в ExecStart airflow-triggerer.service
ExecStart=/bin/bash -c 'source ${VENV_HOME}/bin/activate && ${VENV_HOME}/bin/airflow triggerer'

Удалить из airflow-*.service "--pid /run/airflow/*.pid"
pid всех демонов airflow должны создаваться в AIRFLOW_HOME

Копируем airflow services файлы в целевую директорию
cp airflow-*.service /etc/systemd/system/

Включаем автозапуск во время запуска системы

chkconfig airflow-webserver on
chkconfig airflow-scheduler on
chkconfig airflow-worker on
chkconfig airflow-flower on
chkconfig airflow-triggerer on

Запускаем демоны airflow

systemctl start airflow-webserver
systemctl start airflow-scheduler
systemctl start airflow-worker
systemctl start airflow-flower
systemctl start airflow-triggerer

На SLAVE узле:
Копируем файлы airflow, airflow-*.service в соответствующие целевые директории из MASTER узла.

Включаем автозапуск во время запуска системы

chkconfig airflow-webserver on
chkconfig airflow-worker on

Запускаем демоны airflow

systemctl start airflow-webserver

systemctl start airflow-worker

Установка дополнительных модулей Airflow

На каждом узле Airflow:

Залогиниться под airflow.

Активируем виртуальную среду

source ~/software/env3/bin/activate

Устанавливаем модуль, на примере apache-airflow-providers-oracle:

pip install apache-airflow-providers-oracle

На MASTER узле:

Перезагрузить airflow-webserver

systemctl restart airflow-webserver

Добавление очередей в Worker.

Залогиниться под airflow

Остановить worker

sudo systemctl stop airflow-worker

Добавить новую очередь в скрипт запуска worker'а

cd /etc/systemd/system/
sudo vi airflow-worker.service
# Добавляем в список после флага '-q' через запятую без пробелов новые очереди,
которые будет обрабатывать worker: н-р,
ExecStart=/bin/bash -c 'source ${VENV_HOME}/bin/activate && ${VENV_HOME}/bin/airflow celery worker -q default,etl_loy,etl_scm'
Сохраняем

Запускаем worker

sudo systemctl daemon-reload
sudo systemctl start airflow-worker

Экспорт/Импорт глобальных переменных в Airflow

Экспорт

source ~/software/env3/bin/activate
airflow variables export -v variables_export.json

Импорт

source ~/software/env3/bin/activate
airflow variables import -v variables_export.json

RSL. Airflow. Развертывание цепочки загрузок.

Airflow. Проверить/Установить необходимую версию pyspark:
Активировать environment,
e.g. source ~/software/env3/bin/activate
pip install pyspark==3.4.0
Airflow. Установить дополнительные пакеты (если не установлены):
(см Providers packages reference — apache-airflow-providers Documentation)
pip install apache-airflow-providers-jdbc
pip install apache-airflow-providers-postgres
Hadoop, Ranger: Проверить/создать необходимых пользователей:
airflow:hadoop, aloha:hadoop, rsl, hive
Ambari: Настроить hadoop config:
hadoop.proxyuser.airflow.groups=*
hadoop.proxyuser.airflow.hosts=*
RabbitMQ. Создать необходимые очереди для работы (etl_loy).

Возможно, добавить/изменить очереди:
Исправляем в ExecStart airflow-worker.service
ExecStart=/bin/bash -c 'source ${VENV_HOME}/bin/activate &&
${VENV_HOME}/bin/airflow celery worker -q default,etl_loy'
default оставим для общих задач
Airflow. Создать/импортировать Airflow variables для проекта RSL.
Airflow. Настроить connections типа spark
(Connection Id=spark_etl, Type=Spark, Host=yarn,
Extra={"queue": "etl", "deploy-mode": "cluster"})

Настройка ETL.

Репозиторий: https://git.polet-it.ru/rs.analytics/analytics/-/tree/development/RSL/airflow/rsl

Airflow server (здесь 10.14.8.33).
В папку DAGS_FOLDER (/opt/airflow/dags) (задается в конфигурации) скопировать подготовленные Dags для проекта RSL.

Создать папку (user=demo) /opt/airflow/dags/demo/rsl

Скопировать все DAGs и конфигурационный файл из проекта (Analytics/RSL/airflow/rsl) в выше созданную папку на сервере.

Внести изменения в конфигурационный файл __init__.py:

DAG_USER='demo'

-

HDFS: Скопировать необходимые файлы на HDFS.

Создать папки /user/aloha/spark/lib, /user/aloha/spark/lib/json, /user/aloha/spark/share

Файл properties.json содержит настройки подключения.
При необходимости откорректировать и скопировать в папку /user/aloha/spark/lib/json.

Актуальный jar проекта rsl_2.12-1.0.jar скопировать в /user/aloha/spark/lib

spark3.4.0_jars.zip скопировать в hdfs:///spark3.4.0/spark3.4.0_jars.zip

arima_env.tar.gz скопировать в hdfs:///user/aloha/spark/share/arima_env.tar.gz

Установка компонента Retail Suite.Global Loyalty&CRM MobileApp

Установка компонента происходит на устройстве конечного пользователя из магазинов Android и iOS.

Установка Retail Suite.Global Loyalty&CRM MobileApp Server

Для установки приложения mproxy необходимы следующие сервисы в ОС

- *Java 8*
- Apache Tomcat 9 (>=9.0.80)
- MySQL/MariaDB
- Nginx/Apache

В MySQL/MariaDB создать базу с именем *mproxy_database_2_0* и пользователя для нее.

В Apache Tomcat/Apache (пример настрое указан для Tomcat) опубликовать приложение (WAR-файл), после чего изменить настройки подключения к БД в файле *<apache-tomcat>/webapps/<**mproxy-name**>/WEB-INF/classes/application.properties*:

```
[…]
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/mproxy_database_2_0?useSSL=false&useUnicode=yes&characterEncoding=UTF-8&serverTimezone=Europe/Moscow
# Username and password
spring.datasource.username=mproxy
spring.datasource.password=password
[…]
```

В файле конфиругации Apache Tomcat (<apache-tomcat>/conf/server.xml) изменить перенаправление корня на приложение, добавив в конец секции <Host></Host>:

```
[…]
<Context path="" docBase="<mproxy-name>"></Context>
[…]
```

Пример файла виртуального хоста для nginx:

```
[…]
server {
  listen 80 default_server;

  root /var/www/html;
  index index.html index.htm;

  server_name mproxy;

  location / {
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Forwarded-Server $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass http://127.0.0.1:8080;

    allow all;
  }

  location ~* ^/(mobile|messaging|resources|api)($|\/) {
    proxy_pass http://127.0.0.1:8080;

   allow all;
  }

  ## tomcat manager (block)
  location /manager { deny all; }
  location /host-manager { deny all; }
}
[…]
```

4 Установка модуля Retail Suite.Global Integration

Retail Suite.Integration – программный комплекс (интеграционная шина данных), обеспечивающий взаимодействие всех Модулей Retail Suite.Global в целях передачи данных (интеграции) между Модулями в различных режимах, а также обеспечивающий передачу данных в целях исполнения кросс бизнес-процессов.

Для корректной установки модуля RS.Integration требуется предварительная установка платформы Java 11 по инструкции, прилагаемой к платформе.

Установка ActiveMQ Artemis 2.31.2

Релиз доступен в составе ПО

```
# groupadd -r  -g 995 artemis
# useradd -r -g artemis -d /opt/apache-artemis -s /sbin/nologin -u 997 artemis

# yum install -y libaio



# ARTEMIS_VERSION='2.27.0'
# cd /opt
# wget -P /opt/ \
 https://archive.apache.org/dist/activemq/activemq-artemis/${ARTEMIS_VERSION}/apache-artemis-${ARTEMIS_VERSION}-bin.tar.gz
# tar xvf apache-artemis-${ARTEMIS_VERSION}-bin.tar.gz
# ln -s apache-artemis-${ARTEMIS_VERSION} apache-artemis
# chown -RH artemis: /opt/apache-artemis/.



# su -l artemis -s /bin/bash

$ vi ~/.bash_profile

[…]

# .bash_profile

# Get the aliases and functions

if [ -f ~/.bashrc ]; then

        . ~/.bashrc

fi

# User specific environment and startup programs

export ARTEMIS_HOME="${HOME}"
```

```
export PATH=${JAVA_HOME}/bin:${HOME}/.local/bin:${HOME}/bin:${PATH}

[…]

$ source ~/.bash_profile

$ java -version

$ mkdir -pv /opt/apache-artemis/devbroker
$ cd /opt/apache-artemis/bin

$ ./artemis create --allow-anonymous --user karaf --password karaf /opt/apache-
artemis/devbroker



$ vi /opt/apache-artemis/devbroker/etc/bootstrap.xml
[…]

<binding uri="http://0.0.0.0:8161">

[…]



$ vi /opt/apache-artemis/devbroker/etc/jolokia-access.xml
[...]
<allow-origin>://</allow-origin>
[...]
```

Тестовый запуск (Ctlr+C для завершения):

```
$ /opt/apache-artemis/devbroker/bin/artemis run
```

Привести файл запуска Systemd к следующему виду:

```
$ vi /opt/apache-artemis/bin/activemq-artemis.service

[…]

[Unit]
Description=Apache ActiveMQ Artemis

[Service]
Type=forking
User=artemis
Group=artemis
PIDFile=/opt/apache-artemis/devbroker/data/artemis.pid
```

```
ExecStart=/opt/apache-artemis/devbroker/bin/artemis-service start
ExecStop=/opt/apache-artemis/devbroker/bin/artemis-service stop
ExecReload=/opt/apache-artemis/devbroker/bin/artemis-service restart
Restart=always

WorkingDirectory=/opt/apache-artemis
Environment=JAVA_HOME=/opt/jdk11

[Install]
Alias=artemis
WantedBy=default.target

[…]
```

```
$ exit

# cp /opt/apache-artemis/bin/activemq-artemis.service /etc/systemd/system/

# systemctl daemon-reload

# systemctl enable activemq-artemis

# systemctl restart activemq-artemis

# systemctl status --no-pager activemq-artemis

# netstat -tulpn | grep 8161
```

Установка Apache Karaf
Релиз доступен в составе ПО

```
# groupadd -r -g 994 karaf
# useradd -r -g karaf -d /opt/apache-karaf -s /sbin/nologin -u 996 karaf
```

```
# KARAF_VERSION='4.4.1'
# cd /opt
# wget -P /opt/ https://archive.apache.org/dist/karaf/${KARAF_VERSION}/apache-karaf-
${KARAF_VERSION}.tar.gz
# tar xvf apache-karaf-${KARAF_VERSION}.tar.gz
# ln -s apache-karaf-${KARAF_VERSION} apache-karaf
# chown -RH karaf: /opt/apache-karaf-${KARAF_VERSION}
```

```
# su -l karaf -s /bin/bash

$ vi ~/.bash_profile
```

```
[…]

# .bash_profile

# Get the aliases and functions

if [ -f ~/.bashrc ]; then

        . ~/.bashrc

fi

# User specific environment and startup programs

export KARAF_HOME="${HOME}"

export
PATH=${JAVA_HOME}/bin:${KARAF_HOME}/bin:${HOME}/.local/bin:${HOME}/bin:${
PATH}

[…]

$ source ~/.bash_profile

$ java -version

$ karaf

  > feature:install service-wrapper
  > wrapper:install
  > system:shutdown



$ cp /opt/apache-karaf/bin/karaf.service /opt/apache-karaf/bin/karaf.service.orig
```

Изменить параметры для Systemd

```
$ vi /opt/apache-karaf/bin/karaf.service

[…]

[Service]
Type=forking
User=karaf
Group=karaf

PIDFile=/opt/apache-karaf/karaf.wrapper.pid
ExecStart=/opt/apache-karaf/bin/karaf-service start
ExecReload=/opt/apache-karaf/bin/karaf-service restart
ExecStop=/opt/apache-karaf/bin/karaf-service stop
```

```
WorkingDirectory=/opt/apache-karaf
Environment=JAVA_HOME=/opt/jdk11

Nice=15

[…]
```

```
$ exit

# cp /opt/apache-karaf/bin/karaf.service /etc/systemd/system/
```

Лимиты

```
# vi /etc/security/limits.conf

[…]

karaf          soft nofile 1000000

karaf          hard nofile 1000000

karaf          soft nproc  16384

karaf          hard nproc  16384

[…]
```

```
# systemctl daemon-reload

# systemctl enable karaf

# systemctl restart karaf

# systemctl status --no-pager karaf

# su -l karaf -s /bin/bash

$ vi ${KARAF_HOME}/etc/org.ops4j.pax.url.mvn.cfg

[…]

#

# if "defaultLocalRepoAsRemote" is set to *any* value, localRepository will be
```

```
# added to the list of remote repositories being searched for artifacts
#
org.ops4j.pax.url.mvn.defaultLocalRepoAsRemote = true
[…]
org.ops4j.pax.url.mvn.repositories= \
    https://archiva.polet-it.ru/repository/internal/@id=internal, \
    https://archiva.polet-it.ru/repository/snapshots/@id=snapshots@snapshots@noreleases, \
    https://archiva.polet-it.ru/repository/mirror2/@id=mirror2
[…]
```

Доступ в Hawtio (раскомментировать):

```
$ vi ${KARAF_HOME}/etc/users.properties
[...]
karaf = karaf,_g_:admingroup
_g_\:admingroup = group,admin,manager,viewer,systembundles,ssh
[...]
```

Ctrl+D для выхода из килента:

```
$ client
maven:
  > feature:install maven
```

artemis и JMS фичи

```
  > feature:repo-add mvn:org.apache.activemq/artemis-features/2.20.0/xml/features
  > feature:repo-add mvn:org.ops4j.pax.jms/pax-jms-features/1.1.1/xml/features
  > feature:install artemis-jms-client artemis-core-client
  > feature:install pax-jms-config pax-jms-artemis pax-jms-pool-narayana
```

hawtio

```
> feature:repo-add mvn:io.hawt/hawtio-karaf/2.14.3/xml/features

> feature:install hawtio
```

cfx

```
> feature:repo-add mvn:org.apache.cxf.karaf/apache-cxf/3.4.5/xml/features
```

camel

```
> feature:repo-add mvn:org.apache.camel.karaf/apache-camel/3.14.0/xml/features
```

```
$ vi ~/.m2/repository/org/apache/camel/karaf/apache-camel/3.14.0/apache-camel-3.14.0-features.xml
[...]
  <bundle dependency="true">mvn:org.codehaus.woodstox/stax2-api/3.1.4</bundle>
  <bundle dependency="true">mvn:org.codehaus.woodstox/woodstox-core-asl/4.4.1</bundle>
[...]
  <repository>mvn:org.apache.cxf.karaf/apache-cxf/3.4.5/xml/features</repository>
[...]
```

```
# systemctl restart karaf
```

```
> feature:install camel
> feature:install camel-jms
> feature:install camel-cxf
> feature:install camel-kafka
```

Доступ через HTTP

http://<HOST>:8181/cxf/transactions?_wadl

http://<HOST>:8161/console/auth/login

```
$ vi ${KARAF_HOME}/etc/karaf-wrapper.conf

[...]

# Initial Java Heap Size (in MB)
```

wrapper.java.initmemory=2048

Maximum Java Heap Size (in MB)

wrapper.java.maxmemory=8192

decanter + elastic

```
# su -l karaf -s /bin/bash

$ client

  > feature:repo-add mvn:org.apache.karaf.decanter/apache-karaf-decanter/2.8.0/xml/features

  > feature:install decanter-collector-camel

  > config:edit org.apache.karaf.decanter.appender.elasticsearch
  > config:property-list

  > config:property-set addresses http://elastic.retailscm.ru:9200
  > config:property-set index.prefix narodnyi-karaf
  > config:update
  > config:property-list

  > log:tail
```

Cron

```
# crontab -e

[...]

30 15 * * 0 systemctl restart karaf >/dev/null 2>&1

0 23 * * 3 systemctl restart karaf >/dev/null 2>&1

[...]
```

SSH

Если SCM и RDF стенды разнесены, нужно добавить RSA-ключ для SSH-авторизации пользователя karaf к rpas1502 на RDF.

SFTP

Настраивается после того, как уже был настроен общий SFTP.

```
# useradd -d /home/ftpuser2 -g ftpusers -s /sbin/nologin --no-create-home ftpuser2

# usermod -a -G oinstall ftpuser2
```

Пароль, 14 символов:

```
# passwd ftpuser2

# mkdir -pv /home/ftpuser2

# chown root: /home/ftpuser2

# chmod 755 /home/ftpuser2

# mkdir -pv /home/ftpuser2/FileStore

# vi /etc/fstab

[...]

/home/u01/RPASStore /home/ftpuser2/RPASStore none defaults,bind 0 0

[...]

# mount /home/ftpuser2/RPASStore

# df -a | grep ftp
```

5 Установка модуля Retail Suite.Global E-License

Модуль «RS.E-Licenses» предназначен для защиты производимого компанией правообладателем программного обеспечения от несанкционированного распространения и использования.
Модуль состоит из следующих модулей:

- Клиент – универсальная компонента, устанавливаемая на тот же компьютер, где развернуто защищаемое программное обеспечение, и предназначенная для обработки запросов от него к системе.
- Сервер – сервис, отвечающий за обработку запросов клиентов на выполнение лицензионных функций.
- Лицензионный файл – файл (с расширением *.lky), необходимый для корректной работы экземпляра защищаемого программного обеспечения. Имя лицензионного файла совпадает с номером лицензии и устанавливается производителем защищаемого программного обеспечения (например, «15658799.lky»).

Взаимодействие клиента с сервером осуществляется посредством https сервисов. Программный продукт «RS.E-Licenses» с предустановленной периодичностью обращается к серверам правообладателя программного обеспечения для проверки параметров лицензирования и отсутствия нарушения лицензионной политики правообладателя.

Для установки под операционной системой Windows запустите программу setup.exe из поставки дистрибутива.

Изучив приведенную в окне информацию, нажмите кнопку «Далее». Для прекращения процесса инсталляции на любом шаге – нажмите кнопку «Отмена».

Для установки серверной части необходимо отметить все компоненты в списке.

Для установки клиентской компоненты необходимо отметить ее в списке и далее указать параметры подключения к серверу RS.E-Licenses.

Особенности установки RS.E-Licenses

Установка "RS.E-Licenses" должна производиться после установки платформы 1С:Предприятие. Если при установке "RS.E-Licenses" был запущен экземпляр приложения 1С - необходим перезапуск этого приложения.

5.1.1 Для установки под операционной системой Linux (поддерживаются дистрибутивы Astra Linux, RED OS, Debian) выполните следующие шаги:

1. *Выполнить вход на терминал под пользователем с правами root*
2. *Запустить установочный скрипт командой **./install_lm_X.X.X.X.run***
3. *Проверить установку командой **systemctl status lm4.** Статус сервиса должен быть **active**.*
4. *После установки программного продукта веб-интерфейс доступен по адресу **localhost:8001***
5. *Для генерации файлов лицензий необходимо запустить утилиту **remoteburn** из терминала, которая сгенерирует код компьютера. Данный код необходимо передать поставщику лицензий на программные продукты.*

6. *Полученные от поставщика файлы лицензий необходимо разместить в каталог **/opt/lm4/licences***
7. *После установки файлы настройки «RS.E-Licenses» располагаются в каталоге **/opt/lm4/settings.ini***
8. *Файлы логов продукта располагаются в каталоге **/opt/lm4/logs/***

Для настройки параметров используется файл *settings.ini*

В модуле "RS.E-Licenses" существует возможность работы с несколькими серверами, на которых установлена компонента "сервер RS.E-Licenses". На каждом из таких серверов могут находиться лицензии защищаемых программных продуктов

Доступ к веб-интерфейсу распределённой системы серверов осуществляется по адресу http://*имя компьютера*:8001/, где *имя компьютера* - это ip-адрес компьютера с установленным программным продуктом "RS.E-Licenses"

На вкладке "Подключения" выводится текущая информация о подключениях к серверам RS.E-Licenses и клиентах, которые используют ту или иную лицензию.

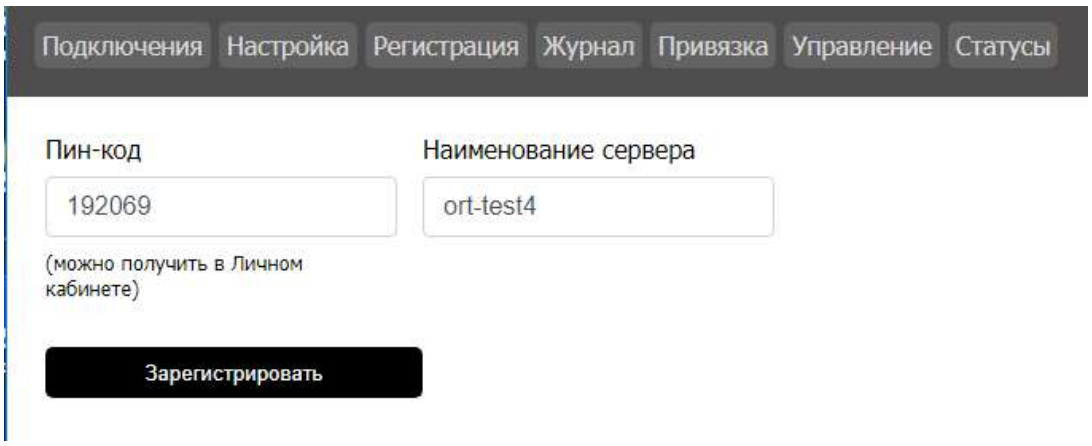
На вкладке "Регистрация" производится регистрация пин-кодов для генерации новых электронных лицензий. Подробнее в разделе **Ошибка! Источник ссылки не найден.** Ошибка! Источник ссылки не найден.

На вкладке "Привязка" производится добавление существующих серверов "RS.E-Licenses" и привязка к ним лицензий.

- *Для использования электронных лицензий необходимо в личном кабинете пользователя создать заявку на новую или измененную лицензию программного продукта.*
- *После создания заявки и ее обработки компанией правообладателем программного обеспечения на электронную почту пользователя поступит письмо с текстом, что в личном кабинете опубликована новая заявка на регистрацию пин-кодов.*
- *Далее пользователь запускает личный кабинет и после авторизации переходит в раздел "Лицензии".*

В табличной части данного раздела опубликованы пин-коды активации программных продуктов.

Если коды активации отображаются, пользователь переходит в веб-интерфейс сервера на вкладку «Регистрация» и последовательно, по одному, регистрирует каждый выданный пин-код.



Для регистрации необходимо указать сам пин-код, ввести наименование сервера и нажать кнопку «Зарегистрировать». На экране появится сообщение о успешной регистрации пин-кода. Регистрацию необходимо проделать для каждого выданного пин-кода.

После регистрации пин-кодов в личном кабинете пользователя также отображается информация о их регистрации.

После обработки компанией правообладателем программного обеспечения зарегистрированных пин-кодов на электронную почту клиента поступит письмо с уведомлением о том, что в личном кабинете опубликована новая лицензия. Пользователю необходимо в личном кабинете перейти на вкладку «Лицензии» и скачать опубликованные файлы лицензий

Скачанные файлы лицензий необходимо скопировать в каталог *\LM4\Licences* (при установке по умолчанию) сервера RS.E-Licenses.

- *В дистрибутиве модуля содержится утилита удаленной прошивки лицензий. Данная утилита предназначена для повторного получения (например, при изменении сервера) электронных лицензий защиты, привязанных к аппаратной части компьютера.*
- *По умолчанию утилита отмечена для установки в составе дистрибутива.*
- *Для получения электронной лицензии необходимо запустить утилиту, полностью скопировать код прошивки на вкладке «Компьютер» и передать его в компанию правообладателя программного обеспечения для генерации лицензии.*
-

- 
Удалённая прошивка ключей
Компьютер
USB Flash
1
Передайте код прошивки (см. ниже) правообладателю ПО и получите лицензионный файл (*.lky)
MmbyxjHdkmT/fD3emte6psTfxJ7ncgDB
fcPdlsWNVQJaIVCkXImwt7lqt/7cmLIK
AqKcF8cv8CHjfwVy+C6e9LD3FaTsyvxm
Bq++1c4kPkot8wm2A7tr/Cqv4+DJFjkY
hoAvEzjYFuz/Nk4rOLFq5Nqa0xds7ZhN
+dHEXEhEyUQ9AwvqnfgjBXNxiIK+QMix
0BiKoC8e64cnR61bOCOXjwnOcTDU6N4o
MbA41bvhP4te5rjjYDbj9PVdsGSyMQZc
kPUd2GBJJBu9Df5ktyG9AxawGSCE4MUL
EU3Uw9o9O0mdfZJZREuqkvvzua459TYs
5vul/Y9ZmHVtl7SuhIWk=
Скопировать в буфер обмена