Платформа разработки аналитических приложений
Insight low code

INSIGHT

LOW CODE

Подготовка сервера и установка приложений на сервере

ОГЛАВЛЕНИЕ

ОПРЕДЕЛЕНИЯ

Таблица 1 Список определений и сокращений

Термин/Сокращение	Определение
Авторизация	Авторизация (англ. authorization «разрешение; уполномочивание») — предоставление определённому лицу или группе лиц прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий.
Агрегация данных	Сбор информации из баз данных с целью подготовки комбинированных наборов данных для обработки данных
БД	База данных
ВМ	Виртуальная машина
ИС	Информационная система
НСД	Несанкционированный доступ
ОС	Операционная система
ПАК	Программно-аппаратный комплекс
Платформа Insight low code / Платформа	Платформа быстрой разработки аналитических приложений «Инсайт» («Insight low code») – совокупность функциональных и технических модулей
ПО	Программное обеспечение
ППО	Прикладное программное обеспечение
СУБД	Система управления базами данных
ТЗ	Техническое задание
УБД	Система управления базой данных
Фреймворк	От англ. framework — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.
Apache Airflow	Open-source инструмент, который позволяет разрабатывать, планировать и осуществлять мониторинг сложных рабочих процессов
Apache Nifi	Apache Nifi - программный проект от Apache Software Foundation, предназначенный для автоматизации потока данных между программными системами.
CentOS	CentOS — дистрибутив Linux, основанный на коммерческом Red Hat Enterprise Linux компании Red Hat и

Термин/Сокращение	Определение
	совместимый с ним.
CI/CD	Набор методов и практик, отвечающий требованиям современной ПО-разработки
Docker	Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений.
HTML	HyperText Markup Language (англ.) - стандартизированный язык разметки документов в Интернете. Большинство веб-страниц содержат описание разметки на языке HTML. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.
HTTPS	HTTPS — расширение протокола HTTP для поддержки шифрования в целях повышения безопасности. Данные в протоколе HTTPS передаются поверх криптографических протоколов TLS или SSL.
JDK	Java Development Kit — комплект разработчика приложений на языке Java, включающий в себя компилятор Java, стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему Java.
Keycloak	Keycloak продукт с открытым кодом для реализации single sign-on с возможностью управления доступом, нацелен на современные применения и сервисы.
MySQL	MySQL — свободная реляционная система управления базами данных.
Nginx	Nginx — веб-сервер и почтовый прокси-сервер, работающий на Unix-подобных операционных системах.
PHP	от английского Hypertext Preprocessor – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений.
PostgreSQL	PostgreSQL — свободная объектно-реляционная система управления базами данных.
RHEL	Red Hat Enterprise Linux — дистрибутив Linux компании Red Hat. Данный дистрибутив позиционируется для корпоративного использования.

Термин/Сокращение	Определение
SDK	SDK — набор средств разработки, позволяющий специалистам по программному обеспечению создавать приложения для определённого пакета программ, программного обеспечения базовых средств разработки, аппаратной платформы, компьютерной системы, игровых консолей, операционных систем и прочих платформ.
URL	От английского Uniform Resource Locator – единообразный локатор (определитель местонахождения) ресурса.

1. ОБЩАЯ ИНФОРМАЦИЯ

Программный комплекс Insight low code является сложно организованной системой, для нормального функционирования которой требуется стопроцентное удовлетворение внешних зависимостей, настроек операционной системы и корректное указание значений переменных в конфигурационных файлах.

В качестве серверной операционной системы могут использоваться дистрибутивы, основанные на Linux. Рекомендуемая конфигурация Платформы – ALT Linux или Astra Linux.

Ввиду специфики экосистемы Linux и разнообразия сборок даже одноименных дистрибутивов, возможны нюансы развертывания, которые не удалось осветить в данной инструкции.

В случае возникновения сложностей при развертывании, ошибок при запуске и прочих проблемах, необходимо обратиться по адресу электронной почты: devops@goodt.me

Insight low code это продукт, объединяющий в себе **методологию и комплекс программных продуктов** как собственной разработки, так и Opensource решения.

УСЛОВИЯ, ПРИ СОБЛЮДЕНИИ КОТОРЫХ ОБЕСПЕЧИВАЕТСЯ ПРИМЕНЕНИЕ СРЕДСТВА АВТОМАТИЗАЦИИ В СООТВЕТСТВИИ С НАЗНАЧЕНИЕМ

Общесистемное программное обеспечение должно удовлетворять следующим требованиям:

- операционная система на базе RHEL/CentOS 7;
- веб-сервер Nginx версии не ниже 1.16;
- сервер баз данных MySQL версии 5.7;
- сервер баз данных PostgreSQL версии 11;
- язык PHP версии не ниже 7.2;
- JDK 8 и 11;
- необходимые программные библиотеки;
- системные утилиты и другое необходимое программное обеспечение.

Пользовательский интерфейс Системы должен корректно работать при использовании посетителями следующих версий интернет-браузеров:

- Google Chrome версии 74 и выше;
- Microsoft Edge верси 85 и выше;

- Chromium 91 версия и выше;
- Mozilla Firefox версии 68 и выше;
- Safari версии 12.1 и выше;
- Яндекс.Браузер версии 20.3.0.1223 и выше;
- Спутник - версия 5.3.5380.0 и выше.

В программный стек необходимый для работы, входит:

- NGINX;
- MYSQL;
- PostgreSQL;
- Jenkins;
- Keycloak;
- Apache NIFI;
- Docker.

Установка сводится к следующим шагам:

1. Подготовка структуры папок;
2. Установка Docker;
3. Установка PostgreSQL 11;
4. Установка MySQL;
5. Установка Nginx;
6. Установка Apache Nifi;
7. Установка Keycloak;
8. Настройка реверс-прокси для Keycloak;
9. Установка приложений:
    - Установка плагина goodt-dremio-plugin;
    - Установка SDK rtl-dremio-connector;
    - Установка rtl-goodteditor-player;
    - Goodt-editor редактор;
10. Запуск приложений.

Для выполнения приведенной ниже инструкции необходимы права суперпользователя. Рекомендуется создать пользователя, под которым будет выполняться вход в приложение в дальнейшем. Часть действий должна производиться под этим пользователем, часть под суперпользователем.

2. ПОДГОТОВКА СТРУКТУРЫ ПАПОК

Приложение должно быть размещено в строгой структуре папок. Следование этому требованию позволит обеспечить совместимость со скриптами, предназначенными для дальнейшего обновления приложения.

- abc
  - distr
    - scripts
  - data
    - dremio
  - app
    - goodt
      - goodt-dremio-plugin

Для создания структуры выполните следующую команду:

sudo mkdir -p /abc/distr/scripts /abc/data/dremio /abc /app /goodt/ goodt- dremio-plugin

3. УСТАНОВКА И ЗАПУСК DOCKER

Для установки Docker необходимо добавить репозиторий:

sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

После необходимо установить Docker:

sudo yum install docker-ce docker-ce-cli containerd.io

После установки требуется включить автозапуск службы Docker:

sudo systemctl enable docker

Далее, необходимо запустить службу Docker:

sudo systemctl start docker

4. УСТАНОВКА POSTGRESQL 11

sudo yum -y install https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
sudo yum -y install postgresql11-server
sudo /usr/pgsql-11/bin/postgresql-11-setup initdb
sudo systemctl start postgresql-11
sudo systemctl enable postgresql-11

Смените пароль у пользователя postgres:

sudo passwd postgres

Так же установите пароль у пользователя postgres в СУБД:

sudo -u postgres psql
\password

5. УСТАНОВКА MYSQL

wget -P /abc/distr/ https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm && sudo rpm -Uvh /abc/distr/mysql57-community-release-el7-11.noarch.rpm
sudo yum install mysql-community-server -y
sudo systemctl start mysqld.service
sudo systemctl status mysqld.service

Смените пароль для пользователя root в БД со сгенерированного при установке на сгенерированный вами. Для этого выполните следующие шаги:

1. Получение временного пароля, сгенерированного при установке

sudo grep 'temporary password' /var/log/mysqld.log

Пример вывода команды:

2020-05-21T17:30:25.128385Z 1 [Note] A temporary password is generated for root@localhost: MkuWp!/S&0W:

2. Смените пароль из примера выше на сгенерированный вами. Запустите консоль MySQL:

mysql -uroot -p

3. Введите пароль, полученный на предыдущем шаге. В данном примере MkuWp!/S&0W:
4. В консоли MySQL смените пароль на сгенерированный вами:

ALTER USER 'root'@'localhost' IDENTIFIED BY 'НовыйПароль!';

5. Создайте базу данных:

CREATE DATABASE `sup_editor` CHARACTER SET utf8 COLLATE utf8_general_ci;

6. УСТАНОВКА NGINX

sudo yum install -y epel-release
sudo yum install -y nginx

- Конфигурация Nginx (для примера):

1. Создайте файл конфигурации в соответствующей папке и символическую ссылку с него на папку для файлов конфигурации в папке Nginx:

touch /abc/conf/nginx/ssl.conf
ln -s /abc/conf/nginx/ssl.conf /etc/nginx/conf.d

2. Отредактируйте созданный файл и добавьте в него следующее содержимое:

```
server {
listen 443 ssl http2 ;
listen [::]:443 ssl http2 ;
server_name
ssl_certificate /abc/conf/cert/fullchain.crt;
ssl_certificate_key /abc/conf/cert/private.key;
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
ssl_ecdh_curve secp384r1;
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off;
#  ssl_stapling on;
ssl_stapling_verify on;
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;
add_header Strict-Transport-Security "max-age=63072000; includeSubdomains;
preload";
add_header X-Content-Type-Options nosniff;
client_max_body_size 1000M;
proxy_read_timeout 60m;
proxy_send_timeout 60m;

include deploy-app-prod/*.conf;


location / {
proxy_set_header Upgrade $http_upgrade;
```

```
proxy_set_header Connection "upgrade";
proxy_set_header X-Real_IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header Host $http_host;
proxy_set_header X-NginX-Proxy true;
#    proxy_set_header X-Forwarded-Proto $scheme;
real_ip_header X-Real-IP;
proxy_pass http://адрес:порт;
proxy_redirect off;
}

location /dremio-connector/ {
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
proxy_set_header X-Real_IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header Host $http_host;
proxy_set_header X-NginX-Proxy true;
proxy_set_header X-Forwarded-Proto $scheme;
real_ip_header X-Real-IP;
proxy_pass http://адрес:порт;
proxy_redirect off;
}

location /editor/
{
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
proxy_set_header X-Real_IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header Host $http_host;
proxy_set_header X-NginX-Proxy true;
proxy_set_header X-Forwarded-Proto $scheme;
real_ip_header X-Real-IP;
proxy_pass http://адрес:порт;
proxy_redirect off;
}

location /api
{
proxy_set_header Upgrade $http_upgrade;
```

```
proxy_set_header Connection "upgrade";
proxy_set_header X-Real_IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header Host $http_host;
proxy_set_header X-NginX-Proxy true;
proxy_set_header X-Forwarded-Proto $scheme;
real_ip_header X-Real-IP;
proxy_pass http://адрес:порт;
proxy_redirect off;

access_log /abc/logs/nginx/editor-access.log;
error_log /abc/logs/nginx/editor-error.log;
}

location /player
{
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
proxy_set_header X-Real_IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header Host $http_host;
proxy_set_header X-NginX-Proxy true;
proxy_set_header X-Forwarded-Proto $scheme;
real_ip_header X-Real-IP;
proxy_pass http://адрес:порт;
proxy_redirect off;

access_log /abc/logs/nginx/editor-access.log;
error_log /abc/logs/nginx/editor-error.log;
}

location /p
{
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
proxy_set_header X-Real_IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header Host $http_host;
proxy_set_header X-NginX-Proxy true;
proxy_set_header X-Forwarded-Proto $scheme;
real_ip_header X-Real-IP;
```

proxy_pass http://адрес:порт;
proxy_redirect off;

access_log /abc/logs/nginx/editor-access.log;
error_log /abc/logs/nginx/editor-error.log;
}

*}*Разместите файл SLL/TLS сертификата по **пути /abc/conf/cert/fullchain.crt**, файл приватного ключа по пути **/abc/conf/cert/private.key**.

7. УСТАНОВКА NEXUS

Nexus запускается на сервере suo3-agg6p в виде докер контейнера. Конфигурация запуска контейнера описана в файле /abc/bin/ci-cd/docker-compose.yml. С таким содержимым:

```
version: "3.0"
services:
  nexus:
    image: sonatype/nexus3
    restart: always
    environment:
      NEXUS_CONTEXT: /
    volumes:
      - /abc/bin/java/lib/security/cacerts:/etc/pki/ca-trust/extracted/java/cacerts:ro
      - nexus-data:/nexus-data
    ports:
      - 8081:8081
      - 8082:8082
volumes:
  nexus-data:
```

Запуск nexus выполняется командой:

```
docker compose -f /abc/bin/ci-cd/docker-compose.yml up -d
```

Далее на сервере nginx в каталоге cat /etc/nginx/conf.d/ создается файл nexus.conf со следующим содержимым:

```
upstream nexus {
        server 10.206.212.138:8081;
}
upstream registry {
        server 10.206.212.138:8082;
}
server {
  listen 443 ssl http2 ;
  listen 80;
  server_name nexus.XXX.YYY.ru;
  ssl_certificate /abc/conf/cert/fullchain.crt;
```

```
  ssl_certificate_key /abc/conf/cert/private.key;
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
  ssl_ecdh_curve secp384r1;
  ssl_session_cache shared:SSL:10m;
  ssl_session_tickets off;
  ssl_stapling_verify on;
  resolver_timeout 5s;
  add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload";
  add_header X-Content-Type-Options nosniff;
  client_max_body_size 1000M;
  proxy_read_timeout 60m;
  proxy_send_timeout 60m;

location / {
        if ($http_user_agent ~ docker ) {
        proxy_pass http://registry;
        }
        if ($http_user_agent ~ cri-o ) {
        proxy_pass http://registry;
        }
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header X-Real_IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;
        proxy_set_header X-Forwarded-Proto $scheme;
        real_ip_header X-Real-IP;
        proxy_pass http://nexus;
        proxy_redirect off;
        }
}
```

Выполняется проверка конфигурации nginx и перегрузка веб сервера в случае удачной проверки следующими командами:
nginx –t
nginx –s reload
После чего nexus будет доступен по адресу https:\\nexus.XXX.YYYru и при первом входе под учетной записью admin с паролем admin123 будет запрос на изменение пароля.
Далее необходимо создать репозиторий для докер образов в разделе Administration:

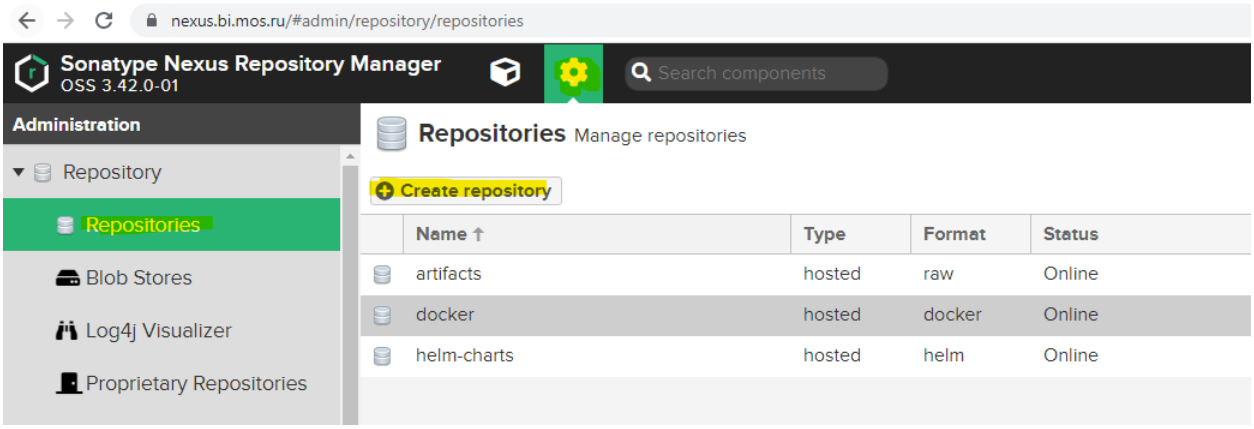


Рисунок 1 – Создание репозитория

8. УСТАНОВКА JENKINS

Официальная документация (https://www.jenkins.io/doc/book/installing/war-file/) сводит установку к запуску war-файла. Однако, в продуктовом окружении это неприменимо. Если выполнить шаги по установке Jenkins с помощью yum, то появляется возможность управлять запуском Jenkins с помощью SystemD.
Данная заметка описывает шаги, которые необходимо выполнить, чтобы добиться такого же результата при ручной установке.
Подготовка окружения

Локаль по умолчанию
Выполните команды:
echo 'export LANG=en_US.UTF-8' >> ~/.bashrc
echo 'export LANGUAGE=en_US.UTF-8' >> ~/.bashrc
echo 'export LC_COLLATE=C' >> ~/.bashrc
echo 'export LC_CTYPE=en_US.UTF-8' >> ~/.bashrc
. ~/.bashrc

Структура папок

Создадим структуру папок в соответствии с принятым корпоративным стандартом и в соответствии с потребностями Jenkins:
- abc
- bin
- jenkins
- war
- logs
- jenkins
- distr

Выполните команду:
mkdir -p /abc/bin/jenkins/war /abc/logs/jenkins /abc/distr

Установка Java 11
Согласно официальной документации, Jenkins полностью поддерживает Java версий 8 и 11 и рекомендованной сборкой является OpenJDK. Установим OpenJDK 11:

```
cd /abs/distr
wget https://XXX.YYY_lts.hb.bizmrg.com/bin/jdk-11.0.9_linux-x64_bin.tar.gz
tar xzfv jdk-11.0.9_linux-x64_bin.tar.gz
mv jdk1.8.0_281 /abc/bin/java
ln -s /abc/bin/java/bin/java /usr/bin
```

Добавим глобальную переменную JAVA_HOME:

```
echo 'export JAVA_HOME=/abc/bin/java' >> ~/.bashrc
. ~/.bashrc
```

Убедимся, что Java успешно установлена

```
java -version
```

Должны получить такой вывод:

```
java version "11.0.9" 2020-10-20 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.9+7-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.9+7-LTS, mixed mode)
```

Установка Jenkins
Получение веб-архива
Скачайте LTS версию Generic Java Package (.war) с официального сайта:
https://www.jenkins.io/download/

```
cd /abc/bin/jenkins
wget https://get.jenkins.io/war-stable/<ver>/jenkins.war
```

Внимание, замените <ver> действительным номером версии!
Конфигурирование SystemD
Создайте файл /etc/rc.d/init.d/jenkins со следующим содержимым:

```
#!/bin/sh

# Check for missing binaries (stale symlinks should not happen)
JENKINS_WAR="/usr/lib/jenkins/jenkins.war"
test -r "$JENKINS_WAR" || { echo "$JENKINS_WAR not installed";
    if [ "$1" = "stop" ]; then exit 0;
    else exit 5; fi; }

# Check for existence of needed config file and read it
JENKINS_CONFIG=/etc/sysconfig/jenkins
test -e "$JENKINS_CONFIG" || { echo "$JENKINS_CONFIG not existing";
    if [ "$1" = "stop" ]; then exit 0;
    else exit 6; fi; }
test -r "$JENKINS_CONFIG" || { echo "$JENKINS_CONFIG not readable. Perhaps you forgot
'sudo'?";
    if [ "$1" = "stop" ]; then exit 0;
    else exit 6; fi; }

JENKINS_PID_FILE="/var/run/jenkins.pid"
JENKINS_LOCKFILE="/var/lock/subsys/jenkins"
```

```
# Source function library.
. /etc/init.d/functions

# Read config
[ -f "$JENKINS_CONFIG" ] && . "$JENKINS_CONFIG"

# Set up environment accordingly to the configuration settings
[ -n "$JENKINS_HOME" ] || { echo "JENKINS_HOME not configured in
$JENKINS_CONFIG";
    if [ "$1" = "stop" ]; then exit 0;
    else exit 6; fi; }
[ -d "$JENKINS_HOME" ] || { echo "JENKINS_HOME directory does not exist:
$JENKINS_HOME";
    if [ "$1" = "stop" ]; then exit 0;
    else exit 1; fi; }

# Search usable Java as /usr/bin/java might not point to minimal version required by Jenkins.
# see http://www.nabble.com/guinea-pigs-wanted-----Hudson-RPM-for-RedHat-Linux-
td25673707.html
candidates="
/etc/alternatives/java
/usr/lib/jvm/java-1.8.0/bin/java
/usr/lib/jvm/jre-1.8.0/bin/java
/usr/lib/jvm/java-11.0/bin/java
/usr/lib/jvm/jre-11.0/bin/java
/usr/lib/jvm/java-11-openjdk-amd64
/usr/bin/java
"
for candidate in $candidates
do
  [ -x "$JENKINS_JAVA_CMD" ] && break
  JENKINS_JAVA_CMD="$candidate"
done

JAVA_CMD="$JENKINS_JAVA_CMD $JENKINS_JAVA_OPTIONS -
DJENKINS_HOME=$JENKINS_HOME -jar $JENKINS_WAR"
PARAMS="--logfile=/var/log/jenkins/jenkins.log --webroot=/var/cache/jenkins/war --daemon"
[ -n "$JENKINS_PORT" ] && PARAMS="$PARAMS --httpPort=$JENKINS_PORT"
[ -n "$JENKINS_LISTEN_ADDRESS" ] && PARAMS="$PARAMS --
httpListenAddress=$JENKINS_LISTEN_ADDRESS"
[ -n "$JENKINS_HTTPS_PORT" ] && PARAMS="$PARAMS --
httpsPort=$JENKINS_HTTPS_PORT"
[ -n "$JENKINS_HTTPS_KEYSTORE" ] && PARAMS="$PARAMS --
httpsKeyStore=$JENKINS_HTTPS_KEYSTORE"
[ -n "$JENKINS_HTTPS_KEYSTORE_PASSWORD" ] && PARAMS="$PARAMS --
httpsKeyStorePassword='$JENKINS_HTTPS_KEYSTORE_PASSWORD'"
```

```
[ -n "$JENKINS_HTTPS_LISTEN_ADDRESS" ] && PARAMS="$PARAMS --
httpsListenAddress=$JENKINS_HTTPS_LISTEN_ADDRESS"
[ -n "$JENKINS_HTTP2_PORT" ] && PARAMS="$PARAMS --
http2Port=$JENKINS_HTTP2_PORT"
[ -n "$JENKINS_HTTP2_LISTEN_ADDRESS" ] && PARAMS="$PARAMS --
http2ListenAddress=$JENKINS_HTTP2_LISTEN_ADDRESS"
[ -n "$JENKINS_DEBUG_LEVEL" ] && PARAMS="$PARAMS --
debug=$JENKINS_DEBUG_LEVEL"
[ -n "$JENKINS_HANDLER_STARTUP" ] && PARAMS="$PARAMS --
handlerCountStartup=$JENKINS_HANDLER_STARTUP"
[ -n "$JENKINS_HANDLER_MAX" ] && PARAMS="$PARAMS --
handlerCountMax=$JENKINS_HANDLER_MAX"
[ -n "$JENKINS_HANDLER_IDLE" ] && PARAMS="$PARAMS --
handlerCountMaxIdle=$JENKINS_HANDLER_IDLE"
[ -n "$JENKINS_EXTRA_LIB_FOLDER" ] && PARAMS="$PARAMS --
extraLibFolder=$JENKINS_EXTRA_LIB_FOLDER"
[ -n "$JENKINS_ARGS" ] && PARAMS="$PARAMS $JENKINS_ARGS"

if [ "$JENKINS_ENABLE_ACCESS_LOG" = "yes" ]; then
   PARAMS="$PARAMS --
accessLoggerClassName=winstone.accesslog.SimpleAccessLogger --
simpleAccessLogger.format=combined --
simpleAccessLogger.file=/var/log/jenkins/access_log"
fi

RETVAL=0

case "$1" in
   start)
   echo -n "Starting Jenkins "
   daemon --user "$JENKINS_USER" --pidfile "$JENKINS_PID_FILE" "$JAVA_CMD"
$PARAMS > /dev/null
   RETVAL=$?
   if [ $RETVAL = 0 ]; then
      success
      echo > "$JENKINS_PID_FILE"  # just in case we fail to find it
         MY_SESSION_ID=`/bin/ps h -o sess -p $$`
         # get PID
         /bin/ps hww -u "$JENKINS_USER" -o sess,ppid,pid,cmd | \
         while read sess ppid pid cmd; do
      [ "$ppid" = 1 ] || continue
      # this test doesn't work because Jenkins sets a new Session ID
            # [ "$sess" = "$MY_SESSION_ID" ] || continue
         echo "$cmd" | grep $JENKINS_WAR > /dev/null
      [ $? = 0 ] || continue
      # found a PID
      echo $pid > "$JENKINS_PID_FILE"
```

```
        done
        touch $JENKINS_LOCKFILE
    else
        failure
    fi
    echo
    ;;
    stop)
    echo -n "Shutting down Jenkins "
    killproc jenkins
    rm -f $JENKINS_LOCKFILE
    RETVAL=$?
    echo
    ;;
    try-restart|condrestart)
    if test "$1" = "condrestart"; then
        echo "${attn} Use try-restart ${done}(LSB)${attn} rather than condrestart
${warn}(RH)${norm}"
    fi
    $0 status
    if test $? = 0; then
        $0 restart
    else
        : # Not running is not a failure.
    fi
    ;;
    restart)
    $0 stop
    $0 start
    ;;
    force-reload)
    echo -n "Reload service Jenkins "
    $0 try-restart
    ;;
    reload)
        $0 restart
    ;;
    status)
        status jenkins
    RETVAL=$?
    ;;
    probe)
    ## Optional: Probe for the necessity of a reload, print out the
    ## argument to this init script which is required for a reload.
    ## Note: probe is not (yet) part of LSB (as of 1.9)

    test "$JENKINS_CONFIG" -nt "$JENKINS_PID_FILE" && echo reload
```

```
    ;;
    *)
    echo "Usage: $0 {start|stop|status|try-restart|restart|force-reload|reload|probe}"
    exit 1
    ;;
esac
exit $RETVAL
```

Создайте файл /etc/sysconfig/jenkins со следующим содержимым:

```
JENKINS_HOME="/abc/bin/jenkins"
JENKINS_JAVA_CMD=""
JENKINS_USER="root"
JENKINS_JAVA_OPTIONS="-Djava.awt.headless=true"
JENKINS_PORT="8080"
JENKINS_LISTEN_ADDRESS=""
JENKINS_HTTPS_PORT=""
JENKINS_HTTPS_KEYSTORE=""
JENKINS_HTTPS_KEYSTORE_PASSWORD=""
JENKINS_HTTPS_LISTEN_ADDRESS=""
JENKINS_HTTP2_PORT=""
JENKINS_HTTP2_LISTEN_ADDRESS=""
JENKINS_DEBUG_LEVEL="5"
JENKINS_ENABLE_ACCESS_LOG="no"
JENKINS_HANDLER_MAX="100"
JENKINS_HANDLER_IDLE="20"
JENKINS_EXTRA_LIB_FOLDER=""
JENKINS_ARGS=""
```

Создайте файл /usr/lib/systemd/system/jenkins.service со следующим содержимым:

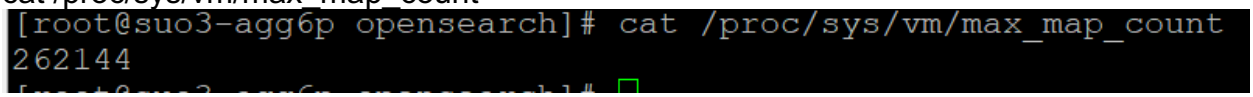
```
[Unit]
Documentation=man:systemd-sysv-generator(8)
SourcePath=/etc/rc.d/init.d/jenkins
Description=LSB: Jenkins Automation Server
Before=runlevel3.target
Before=runlevel5.target
Before=shutdown.target
After=remote-fs.target
After=network-online.target
After=time-sync.target
After=nss-lookup.target
After=time-sync.target
After=sendmail.service
After=network-online.target
Wants=network-online.target
Conflicts=shutdown.target

[Service]
```

```
Type=forking
Restart=no
TimeoutSec=5min
IgnoreSIGPIPE=no
KillMode=process
GuessMainPID=no
RemainAfterExit=yes
ExecStart=/etc/rc.d/init.d/jenkins start
ExecStop=/etc/rc.d/init.d/jenkins stop
ExecReload=/etc/rc.d/init.d/jenkins reload
```

Выполните перезагрузку SystemD:
systemctl daemon-reload
Включите автозапуск службы Jenkins:
systemctl enable jenkins
Запустите Jenkins:
systemctl start Jenkins

9. УСТАНОВКА OPENSEARCH

Установка выполнялась согласно официальной документации https://opensearch.org/docs/latest/install-and-configure/install-opensearch/docker/
Была выполнена подготовка окружения. А именно отключен файл подкачки:
sudo swapoff -a
Так же было увеличено количество памяти доступное OpenSearch:
sudo vi /etc/sysctl.conf
Добавлена строка изменяющая количество выделяемой памяти:
vm.max_map_count=262144
Выполнено обновление конфигурации окружения:
sudo sysctl -p
Выполнена проверка применения изменений:
cat /proc/sys/vm/max_map_count



Запуск стека OpenSearch
Т.к. OpenSearch рекомендовано запускать в режиме кластера, то все контейнеры кластера описаны в едином файле /abc/bin/opensearch/docker-compose.yml, содержимое которого приведено ниже:

```
version: '3'
services:
  opensearch-node1:
    image: opensearchproject/opensearch:latest
    container_name: opensearch-node1
    environment:
      - cluster.name=opensearch-cluster
```

```
      - node.name=opensearch-node1
      - discovery.seed_hosts=opensearch-node1,opensearch-node2
      - cluster.initial_cluster_manager_nodes=opensearch-node1,opensearch-node2
      - bootstrap.memory_lock=true # along with the memlock settings below, disables
swapping
      - "OPENSEARCH_JAVA_OPTS=-Xms2g -Xmx2g" # minimum and maximum Java heap
size, recommend setting both to 50% of system RAM
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536 # maximum number of open files for the OpenSearch user, set to at least
65536 on modern systems
        hard: 65536
    volumes:
      - opensearch-data1:/usr/share/opensearch/data
    ports:
      - 9200:9200
      - 9600:9600 # required for Performance Analyzer
    networks:
      - opensearch-net
  opensearch-node2:
    image: opensearchproject/opensearch:latest
    container_name: opensearch-node2
    environment:
      - cluster.name=opensearch-cluster
      - node.name=opensearch-node2
      - discovery.seed_hosts=opensearch-node1,opensearch-node2
      - cluster.initial_cluster_manager_nodes=opensearch-node1,opensearch-node2
      - bootstrap.memory_lock=true
      - "OPENSEARCH_JAVA_OPTS=-Xms2g -Xmx2g"
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536
        hard: 65536
    volumes:
      - opensearch-data2:/usr/share/opensearch/data
    networks:
      - opensearch-net
  opensearch-dashboards:
    image: opensearchproject/opensearch-dashboards:latest
    container_name: opensearch-dashboards
    ports:
```

```
      - 5601:5601
    expose:
      - "5601"
    environment:
      OPENSEARCH_HOSTS: '["https://opensearch-node1:9200","https://opensearch-
node2:9200"]'
    networks:
      - opensearch-net
  logstash:
    image: opensearchproject/logstash-oss-with-opensearch-output-plugin:7.13.2
    container_name: logstash
    ports:
      - 5044:5044
    volumes:
      - type: bind
        source: ./logstash/pipeline/logstash.conf
        target: /usr/share/logstash/pipeline/logstash.conf
    networks:
      - opensearch-net

volumes:
  opensearch-data1:
  opensearch-data2:

networks:
  opensearch-net:
```

Для запуска кластера OpenSearch выполняется следующая команда:
docker compose up -d -f /abc/bin/opensearch/docker-compose.yml
Краткое описание компонентов

opensearch-node1, opensearch-node2 – ноды OpenSearch (аналог поисковой системы ElasticSearch) утилита полнотекстового поиска и аналитики
opensearch-dashboards – нода отдающая веб-интерфейс для OpenSearch (аналог Kibana), который позволяет взаимодействовать с данными, которые хранятся в его индексах OpenSearch
logstash – нода сбора, преобразования и сохранения в общем хранилище событий из различных источников (файлы, базы данных, логи и пр.) в реальном времени;
Для ноды logstash в файле /abc/bin/opensearch/logstash/pipeline/logstash.conf хранится файл конфигурации для logstash. Содержимое файла приведено ниже:

```
input {
  beats {
    port => 5044
  }
}
filter {
    grok {
```

```
        match => { "message" => "%{COMBINEDAPACHELOG}"}
    }
}
output {
    opensearch {
        hosts => ["https://10.XX.XX.XX:9200"] # OpenSearch cluster IP address
        ssl => true
        ssl_certificate_verification => false
        user  => "admin_user"
        password  => "admin_password"
    }
}
```

На сервере, с которого, необходимо выполнять сбор логов (в нашем случае это веб сервер) устанавливается FileBeat. FileBeat –агент на серверах для отправки различных типов оперативных данных в OpenSearch.
Для установки необходимо выполнить следующие действия:

```
curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-8.6.1-x86_64.rpm
mv filebeat-8.6.1-x86_64.rpm /abc/distr/
rpm -iv filebeat-8.6.1-x86_64.rpm
```

После установки стандартный файл конфигурации /etc/filebeat/filebeat.yml необходимо привести к данному виду:

```
filebeat.inputs:
- type: filestream
  id: my-filestream-id
  enabled: true
  paths:
    - /var/log/nginx/error*.log

# ============================== Filebeat modules
==============================
filebeat.config.modules:
  # Glob pattern for configuration loading
  path: ${path.config}/modules.d/*.yml
  # Set to true to enable config reloading
  reload.enabled: false
  # Period on which files under path should be checked for changes
  #reload.period: 10s

# ======================== Elasticsearch template setting
======================
setup.template.settings:
  index.number_of_shards: 1
  #index.codec: best_compression
  #_source.enabled: false
# ================================== General
==================================
```

```
# The name of the shipper that publishes the network data. It can be used to group
# all the transactions sent by a single shipper in the web interface.
#name:

# The tags of the shipper are included in their own field with each
# transaction published.
#tags: ["service-X", "web-tier"]

# Optional fields that you can specify to add additional information to the
# output.
#fields:
#  env: staging

# ---------------------------- Logstash Output ----------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["suo3-agg6p:5044"]

# ================================ Processors
================================
processors:
  - add_host_metadata:
      when.not.contains.tags: forwarded
  - add_cloud_metadata: ~
  - add_docker_metadata: ~
  - add_kubernetes_metadata: ~
```

Так же необходимо включить модуль логирования nginx, для этого необходимо файл конфигурации /etc/filebeat/modules.d/nginx.yml привести к следующему виду:

```
# Module: nginx
# Docs: https://www.elastic.co/guide/en/beats/filebeat/main/filebeat-module-nginx.html

- module: nginx
  # Access logs
  access:
    enabled: false

    # Set custom paths for the log files. If left empty,
    # Filebeat will choose the paths depending on your OS.
    #var.paths:

  # Error logs
  error:
    enabled: true

    # Set custom paths for the log files. If left empty,
    # Filebeat will choose the paths depending on your OS.
```

```
    #var.paths:

  # Ingress-nginx controller logs. This is disabled by default. It could be used in Kubernetes
environments to parse ingress-nginx logs
  ingress_controller:
    enabled: false

    # Set custom paths for the log files. If left empty,
    # Filebeat will choose the paths depending on your OS.
    #var.paths:
```

10. УСТАНОВКА APACHE AIRFLOW

Установка Airfllow выполнена согласно официальной документации производителя: https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html . Была выполнена подготовка окружения. Созданы папки /abc/distr/airflow/ и /abc/bin/airflow/.
В каталог /abc/distr/airflow/airflow_images были скопированы образы, которые необходимы для запуска сервиса AirFlow: apache_airflow_2-5-1.tar, postgres_13.tar, redis.tar. Выполнена загрузка данных образов в окружение docker:
docker load -i ./airflow_images/*.tar
В каталоге /abc/bin/airflow был создан файл docker-compose.yaml со следующим содержимым:

```
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied.  See the License for the
# specific language governing permissions and limitations
# under the License.
#

# Basic Airflow cluster configuration for CeleryExecutor with Redis and PostgreSQL.
#
# WARNING: This configuration is for local development. Do not use it in a production
deployment.
#
```

```
# This configuration supports basic configuration using environment variables or an .env file
# The following variables are supported:
#
# AIRFLOW_IMAGE_NAME           - Docker image name used to run Airflow.
#                                Default: apache/airflow:2.5.1
# AIRFLOW_UID                  - User ID in Airflow containers
#                                Default: 50000
# AIRFLOW_PROJ_DIR             - Base path to which all the files will be volumed.
#                                Default: .
# Those configurations are useful mostly in case of standalone testing/running Airflow in
test/try-out mode
#
# _AIRFLOW_WWW_USER_USERNAME   - Username for the administrator account (if
requested).
#                                Default: airflow
# _AIRFLOW_WWW_USER_PASSWORD   - Password for the administrator account (if
requested).
#                                Default: airflow
# _PIP_ADDITIONAL_REQUIREMENTS - Additional PIP requirements to add when starting
all containers.
#                                Default: ''
#
# Feel free to modify this file to suit your needs.
---
version: '3'
x-airflow-common:
  &airflow-common
  # In order to add custom dependencies or upgrade provider packages you can use your
extended image.
  # Comment the image line, place your Dockerfile in the directory where you placed the
docker-compose.yaml
  # and uncomment the "build" line below, Then run `docker-compose build` to build the
images.
  image: ${AIRFLOW_IMAGE_NAME:-apache/airflow:2.5.1}
  # build: .
  environment:
    &airflow-common-env
    AIRFLOW__CORE__EXECUTOR: CeleryExecutor
    AIRFLOW__DATABASE__SQL_ALCHEMY_CONN:
postgresql+psycopg2://airflow:airflow@postgres/airflow
    # For backward compatibility, with Airflow <2.3
    AIRFLOW__CORE__SQL_ALCHEMY_CONN:
postgresql+psycopg2://airflow:airflow@postgres/airflow
    AIRFLOW__CELERY__RESULT_BACKEND:
db+postgresql://airflow:airflow@postgres/airflow
    AIRFLOW__CELERY__BROKER_URL: redis://:@redis:6379/0
    AIRFLOW__CORE__FERNET_KEY: ''
```

```
    AIRFLOW__CORE__DAGS_ARE_PAUSED_AT_CREATION: 'true'
    AIRFLOW__CORE__LOAD_EXAMPLES: 'true'
    AIRFLOW__API__AUTH_BACKENDS:
'airflow.api.auth.backend.basic_auth,airflow.api.auth.backend.session'
    _PIP_ADDITIONAL_REQUIREMENTS: ${_PIP_ADDITIONAL_REQUIREMENTS:-}
  volumes:
    - ${AIRFLOW_PROJ_DIR:-.}/dags:/opt/airflow/dags
    - ${AIRFLOW_PROJ_DIR:-.}/logs:/opt/airflow/logs
    - ${AIRFLOW_PROJ_DIR:-.}/plugins:/opt/airflow/plugins
  user: "${AIRFLOW_UID:-50000}:0"
  depends_on:
    &airflow-common-depends-on
    redis:
      condition: service_healthy
    postgres:
      condition: service_healthy

services:
  postgres:
    image: postgres:13
    environment:
      POSTGRES_USER: airflow_user
      POSTGRES_PASSWORD: airflow_password
      POSTGRES_DB: airflow
    volumes:
      - postgres-db-volume:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD", "pg_isready", "-U", "airflow"]
      interval: 5s
      retries: 5
    restart: always

  redis:
    image: redis:latest
    expose:
      - 6379
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 5s
      timeout: 30s
      retries: 50
    restart: always

  airflow-webserver:
    <<: *airflow-common
    command: webserver
    ports:
```

```
      - 8080:8080
    healthcheck:
      test: ["CMD", "curl", "--fail", "http://localhost:8080/health"]
      interval: 10s
      timeout: 10s
      retries: 5
    restart: always
    depends_on:
      <<: *airflow-common-depends-on
      airflow-init:
        condition: service_completed_successfully

  airflow-scheduler:
    <<: *airflow-common
    command: scheduler
    healthcheck:
      test: ["CMD-SHELL", 'airflow jobs check --job-type SchedulerJob --hostname
"$${HOSTNAME}"']
      interval: 10s
      timeout: 10s
      retries: 5
    restart: always
    depends_on:
      <<: *airflow-common-depends-on
      airflow-init:
        condition: service_completed_successfully

  airflow-worker:
    <<: *airflow-common
    command: celery worker
    healthcheck:
      test:
        - "CMD-SHELL"
        - 'celery --app airflow.executors.celery_executor.app inspect ping -d
"celery@$${HOSTNAME}"'
      interval: 10s
      timeout: 10s
      retries: 5
    environment:
      <<: *airflow-common-env
      # Required to handle warm shutdown of the celery workers properly
      # See https://airflow.apache.org/docs/docker-stack/entrypoint.html#signal-propagation
      DUMB_INIT_SETSID: "0"
    restart: always
    depends_on:
      <<: *airflow-common-depends-on
      airflow-init:
```

```
      condition: service_completed_successfully

  airflow-triggerer:
    <<: *airflow-common
    command: triggerer
    healthcheck:
      test: ["CMD-SHELL", 'airflow jobs check --job-type TriggererJob --hostname
"$${HOSTNAME}"']
      interval: 10s
      timeout: 10s
      retries: 5
    restart: always
    depends_on:
      <<: *airflow-common-depends-on
      airflow-init:
        condition: service_completed_successfully

  airflow-init:
    <<: *airflow-common
    entrypoint: /bin/bash
    # yamllint disable rule:line-length
    command:
      - -c
      - |
        function ver() {
          printf "%04d%04d%04d%04d" $${1//./ }
        }
        airflow_version=$$(AIRFLOW__LOGGING__LOGGING_LEVEL=INFO && gosu airflow
airflow version)
        airflow_version_comparable=$$(ver $${airflow_version})
        min_airflow_version=2.2.0
        min_airflow_version_comparable=$$(ver $${min_airflow_version})
        if (( airflow_version_comparable < min_airflow_version_comparable )); then
          echo
          echo -e "\033[1;31mERROR!!!: Too old Airflow version $${airflow_version}!\e[0m"
          echo "The minimum Airflow version supported: $${min_airflow_version}. Only use this
or higher!"
          echo
          exit 1
        fi
        if [[ -z "${AIRFLOW_UID}" ]]; then
          echo
          echo -e "\033[1;33mWARNING!!!: AIRFLOW_UID not set!\e[0m"
          echo "If you are on Linux, you SHOULD follow the instructions below to set "
          echo "AIRFLOW_UID environment variable, otherwise files will be owned by root."
          echo "For other operating systems you can get rid of the warning with manually
created .env file:"
```

```
        echo "    See: https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html#setting-the-right-airflow-user"
        echo
      fi
      one_meg=1048576
      mem_available=$$(($$(getconf _PHYS_PAGES) * $$(getconf PAGE_SIZE) / one_meg))
      cpus_available=$$(grep -cE 'cpu[0-9]+' /proc/stat)
      disk_available=$$(df / | tail -1 | awk '{print $$4}')
      warning_resources="false"
      if (( mem_available < 4000 )) ; then
        echo
        echo -e "\033[1;33mWARNING!!!: Not enough memory available for Docker.\e[0m"
        echo "At least 4GB of memory required. You have $$(numfmt --to iec $$((mem_available * one_meg)))"
        echo
        warning_resources="true"
      fi
      if (( cpus_available < 2 )); then
        echo
        echo -e "\033[1;33mWARNING!!!: Not enough CPUS available for Docker.\e[0m"
        echo "At least 2 CPUs recommended. You have $${cpus_available}"
        echo
        warning_resources="true"
      fi
      if (( disk_available < one_meg * 10 )); then
        echo
        echo -e "\033[1;33mWARNING!!!: Not enough Disk space available for Docker.\e[0m"
        echo "At least 10 GBs recommended. You have $$(numfmt --to iec $$((disk_available * 1024 )))"
        echo
        warning_resources="true"
      fi
      if [[ $${warning_resources} == "true" ]]; then
        echo
        echo -e "\033[1;33mWARNING!!!: You have not enough resources to run Airflow (see above)!\e[0m"
        echo "Please follow the instructions to increase amount of resources available:"
        echo "   https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html#before-you-begin"
        echo
      fi
      mkdir -p /sources/logs /sources/dags /sources/plugins
      chown -R "${AIRFLOW_UID}:0" /sources/{logs,dags,plugins}
      exec /entrypoint airflow version
    # yamllint enable rule:line-length
    environment:
```

```
      <<: *airflow-common-env
      _AIRFLOW_DB_UPGRADE: 'true'
      _AIRFLOW_WWW_USER_CREATE: 'true'
      _AIRFLOW_WWW_USER_USERNAME: ${_AIRFLOW_WWW_USER_USERNAME:-
airflow}
      _AIRFLOW_WWW_USER_PASSWORD: ${_AIRFLOW_WWW_USER_PASSWORD:-
airflow}
      _PIP_ADDITIONAL_REQUIREMENTS: ''
    user: "0:0"
    volumes:
      - ${AIRFLOW_PROJ_DIR:-.}:/sources

  airflow-cli:
    <<: *airflow-common
    profiles:
      - debug
    environment:
      <<: *airflow-common-env
      CONNECTION_CHECK_MAX_COUNT: "0"
    # Workaround for entrypoint issue. See: https://github.com/apache/airflow/issues/16252
    command:
      - bash
      - -c
      - airflow

  # You can enable flower by adding "--profile flower" option e.g. docker-compose --profile
flower up
  # or by explicitly targeted on the command line e.g. docker-compose up flower.
  # See: https://docs.docker.com/compose/profiles/
  flower:
    <<: *airflow-common
    command: celery flower
    profiles:
      - flower
    ports:
      - 5555:5555
    healthcheck:
      test: ["CMD", "curl", "--fail", "http://localhost:5555/"]
      interval: 10s
      timeout: 10s
      retries: 5
    restart: always
    depends_on:
      <<: *airflow-common-depends-on
      airflow-init:
        condition: service_completed_successfully
```

```
volumes:
  postgres-db-volume:
```

Этот файл развертывает следующие компоненты:
airflow-scheduler – Планировщик, выполняет мониторинг всех заданий и DAG`ов, и запускает задание, когда его зависимости завершены.
airflow-webserver – Веб интерфейс, который публикуется на порту 8080 - http://localhost:8080.
airflow-worker – исполнитель, выполняет задания полученные от планировщика.
airflow-triggerer - Триггер запускает цикл событий для отложенных задач.
airflow-init – Служба инициализации airflow.
postgres – БД postgres.
redis - The redis - брокер, выполняет пересылку сообщений от планировщика к исполнителю.
Публикация приложения
Для публикации приложения на веб сервере в каталоге /etc/nginx/conf.d/ создаем файл airflow.conf с таким содержимым:

```
server {
  listen 443 ssl http2 ;
  server_name af.XXX.YYY.ru;
  ssl_certificate /abc/conf/cert/fullchain.crt;
  ssl_certificate_key /abc/conf/cert/private.key;
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
  ssl_ecdh_curve secp384r1;
  ssl_session_cache shared:SSL:10m;
  ssl_session_tickets off;
  ssl_stapling_verify on;
  resolver_timeout 5s;
  add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload";
  add_header X-Content-Type-Options nosniff;
  client_max_body_size 1000M;
  proxy_read_timeout 60m;
  proxy_send_timeout 60m;

        location / {
           proxy_pass http://10.206.212.132:8080;
           proxy_set_header Host $host;
           proxy_set_header X-Real-IP $remote_addr;
           proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
           proxy_set_header X-Forwarded-Proto "https";
        }
}
```

Затем выполняется проверка конфигурации nginx и перегрузка веб сервера в случае удачной проверки следующими командами:

```
nginx –t
nginx –s reload
```

После чего Airflow будет доступен по адресу https://af.XXX.YYY.ru/

11. УСТАНОВКА APACHE NIFI

- Создайте структуру папок командой:
  mkdir -p /abc/bin

- Скачайте и распакуйте архив с OpenJDK в папку /abc/bin/java :
  curl -O https://goodt_lts.hb.bizmrg.com/bin/jdk-8u301-linux-x64.tar.gz
  tar xzfv jdk-8u301-linux-x64.tar.gz
  mv jdk1.8.0_301 /abc/bin/java

- Создайте символическую ссылку /usr/bin/java :
  ln -s /abc/bin/java/bin/java /usr/bin/java

- Задайте значение глобальной переменной $JAVA_HOME :
  echo 'export JAVA_HOME=/abc/bin/java' >> ~/.bashrc
  source ~/.bashrc

- Скачайте и распакуйте архив с NiFi, переместите распакованную папку в /abc/bin/nifi :
  curl https://archive.apache.org/dist/nifi/1.15.0/nifi-1.15.0-bin.tar.gz -O
  tar xzfv nifi-1.15.0-bin.tar.gz
  mv nifi-1.15.0 /abc/bin/nifi

- Выполните установку сервиса NiFi для systemd и запустите NiFi как сервис:
  /abc/bin/nifi/bin/nifi.sh install
  systemctl daemon-reload
  systemctl enable nifi
  systemctl start nifi

- Добавьте локальный IP адрес в /etc/hosts
  nano /etc/hosts
  10.x.x.x test-nifi.goodt.me

- Отредактируйте файл nifi.properties
  nano /abc/bin/nifi/conf/nifi.properties

- Отредактируйте блоки nifi.web.https.host=127.0.0.1 nifi.web.https.port=8443
  nifi.web.https.host=test-nifi.goodt.me
  nifi.web.https.port=8080

- Необходимо произвести промежуточную упаковку в контейнер PKCS12:
  openssl pkcs12 -export -out nifi.p12 -inkey private.key -in fullchain.crt -name nifi-key
  Перед созданием контейнера PKCS12 у вас спросят пароль для последующего экспорта.
- Сгенерируйте надежный пароль и введите его.

- Необходимо сформировать Java Keystore из PKCS12 контейнера:
  keytool -importkeystore -srckeystore nifi.p12 -srcstoretype pkcs12 -srcalias nifi-key -destkeystore keystore.jks -deststoretype jks -destalias nifi-key

- Введите пароль для экспорта, который указали на предыдущем шаге. Далее сгенерируйте надежный пароль для самого Java Keystore, введите его.

- Необходимо сформировать Java Truststore:
  keytool -importcert -alias nifi-cert -file cacert.pem -keystore truststore.jks

- Появится запрос пароля для Java Truststore. Сгенерируйте надежный пароль и введите его. Разрешается использовать один и тот же пароль для Keystore и для Truststore.
- Полученные JKS контейнеры необходимо скопировать на серверы с NiFi, где ведется разработка, на продуктовый NiFi и на сервер с NiFi Registry, разместив их по пути:
  /abc/bin/conf/cert/

- Для завершения настройки NiFi на сервере, где ведется разработка, необходимо отредактировать файл /abc/bin/nifi/config/nifi.properties, приведя значения параметров к виду:
  nifi.web.https.host=test-nifi.goodt.me
  nifi.web.https.port=8080
  # security properties #
  nifi.security.keystore=/abc/bin/conf/cert/keystore.jks
  nifi.security.keystoreType=jks
  nifi.security.keystorePasswd=sF9ZCZvX#9rKHQY-Jv1PSx%IXpeGZ4@nwrO
  nifi.security.truststore=/abc/bin/conf/cert/truststore.jks
  nifi.security.truststoreType=jks
  nifi.security.truststorePasswd=sF9ZCZvX#9rKHQY-Jv1PSx%IXpeGZ4@nwrO

- Не забудьте заменить пароль sF9ZCZvX#9rKHQY-Jv1PSx%IXpeGZ4@nwrO на действительный, сгенерированный вами.
- Необходимо добавить пользователя для управления NiFi:
  /abc/bin/nifi/bin/nifi.sh set-single-user-credentials nifi_admin PASSWORD

- После чего, необходимо перезагрузить сервис NiFi:
  systemctl restart nifi
- Проверить работоспособность по адресу https://test-nifi.goodt.me:8080/ введя данные, созданного ранее пользователя nifi_admin.

12. УСТАНОВКА KEYCLOAK

Инструкция описывает следующие шаги:

- Создание базы данных;
- Установка и настройка Keycloak;
- Создание realm через импорт;
- Настройка реверс-прокси в Nginx.

СОГЛАШЕНИЕ О ФОРМАТИРОВАНИИ И ИМЕНОВАНИИ

- Имена объектов, переменных и их значения, параметры, URL, пути в файловой системе, команды и директивы выделяются разметкой "код в строку", например: https://someurl.com , variable = value
- Устанавливаемая версия Keycloak: 15.0.2
- URL узла, на котором установлен Keycloak: http://auth.corp.ru/
- Адрес и порт СУБД: http://db.corp.ru:5432
- Имя базы данных для приложения: keycloak
- Имя пользователя для подключения к базе данных: keycloak_user
- Пароль пользователя keycloak_user : $7r0n6pP@s2WrD
- Файл SSL-сертификата расположен по пути: /abc/conf/cert/fullchain.crt
- Файл приватного ключа к SSL-сертификату расположен по пути: /abc/conf/cert/private.key

12.1. СОЗДАНИЕ БАЗЫ ДАННЫХ

В консоли psql выполните команды:

```
CREATE DATABASE keycloak;
   CREATE USER keycloak_user WITH ENCRYPTED PASSWORD '$7r0n6pP@s2WrD';
GRANT ALL PRIVILEGES ON DATABASE keycloak TO keycloak_user;
\c keycloak;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO keycloak_user
```

12.2. УСТАНОВКА И НАСТРОЙКА KEYCLOAK

1. Подготовьте структуру папок:
   mkdir -p /abc/bin /abc/conf/nginx

2. Для работы требуется Java. Скачайте и распакуйте архив с OpenJDK в папку /abc/bin/java :
   curl -O https://goodt_lts.hb.bizmrg.com/bin/jdk-8u301-linux-x64.tar.gz
   tar xzfv jdk-8u301-linux-x64.tar.gz
   mv jdk1.8.0_301 /abc/bin/java
3. Создайте символическую ссылку /usr/bin/java :
   ln -s /abc/bin/java/bin/java /usr/bin/java

4. Задайте значение глобальной переменной $JAVA_HOME :
   echo 'export JAVA_HOME=/abc/bin/java' >> ~/.bashrc
   source ~/.bashrc

5. Скачайте архив и распакуйте его в папку, откуда будет производиться запуск:
   wget https://github.com/keycloak/keycloak/releases/download/15.0.2/keycloak-15.0.2.tar.gz
   tar xzfv keycloak-15.0.2.tar.gz
   mv keycloak-15.0.2 /abc/bin/keycloak

6. Выполните настройку для работы за реверс-прокси. Предполагается, что Keycloak установлен в /abc/bin/keycloak. Отредактируйте файл /abc/bin/keycloak/standalone/configuration/standalone.xml. Найдите строку:
   /abc/bin/keycloak
   /abc/bin/keycloak/standalone/configuration/standalone.xml
   <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>

7. Добавьте в нее параметр proxy-address-forwarding="true" , чтобы строка приняла такой вид:
   <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true" proxy-address-forwarding="true" />

8. Выполните настройку использования PostgreSQL в качестве источника данных. Скачайте jdbc-драйвер в папку /abc/bin/keycloak/modules/system/layers/keycloak/org/postgresql/main :
   mkdir -p /abc/bin/keycloak/modules/system/layers/keycloak/org/postgresql/main
   cd /abc/bin/keycloak/modules/system/layers/keycloak/org/postgresql/main
   wget https://jdbc.postgresql.org/download/postgresql-42.2.16.jar

9. Создайте файл /abc/bin/keycloak/modules/system/layers/keycloak/org/postgresql/main/module.xml со следующим содержимым:

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.3" name="org.postgresql">
  <resources>
    <resource-root path="postgresql-42.2.16.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

10. Откройте на редактирование файл /abc/bin/keycloak/standalone/configuration/standalone.xml . Найдите секцию <datasources> . Закомментируйте содержимое секции настроек источника данных по умолчанию с именем KeycloakDS :

```
<datasource jndi-name="java:jboss/datasources/KeycloakDS" pool-name="KeycloakDS"
enabled="true" use-java-context="true" statistics-
enabled="${wildfly.datasources.statistics-
enabled:${wildfly.statistics-enabled:false}}">
  <connection-
url>jdbc:h2:${jboss.server.data.dir}/keycloak;AUTO_SERVER=TRUE</connection-
url>
  <driver>h2</driver>
  <security>
    <user-name>sa</user-name>
    <password>sa</password>
  </security>
</datasource>
```

11. Добавьте настройки подключения к PostgreSQL, итоговый вид:

```
<datasource jndi-name="java:jboss/datasources/KeycloakDS" pool-
name="KeycloakDS"
enabled="true" use-java-context="true">
  <connection-url>jdbc:postgresql://db.corp.ru:5432/keycloak</connection-url>
  <driver>postgresql</driver>
  <pool>
    <max-pool-size>20</max-pool-size>
  </pool>
  <security>
    <user-name>keycloak_user</user-name>
    <password>$7r0n6pP@s2WrD</password>
```

```
    </security>
</datasource>
<!--
<datasource jndi-name="java:jboss/datasources/KeycloakDS" pool-name="KeycloakDS"
enabled="true" use-java-context="true" statistics-
enabled="${wildfly.datasources.statistics-
enabled:${wildfly.statistics-enabled:false}}">
    <connection-
url>jdbc:h2:${jboss.server.data.dir}/keycloak;AUTO_SERVER=TRUE</connection-
url>
    <driver>h2</driver>
    <security>
        <user-name>sa</user-name>
        <password>sa</password>
    </security>
</datasource>
-->
```

12. Добавьте в секцию <drivers> подсекцию:

```
  <driver name="postgresql" module="org.postgresql">
    <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
</driver>
```

13. Создайте unit-файл /etc/systemd/system/keycloak.service для systemd :

```
    [Unit]
Description=Keycloak
After=network.target
[Service]

    Type=simple
Restart=always
User=root
Group=root
ExecStart=/abc/bin/keycloak/bin/standalone.sh -b 0.0.0.0
[Install]
WantedBy=multi-user.target
```

14. Перезапустите службу systemd :

```
systemctl daemon-reload
```

15. Включите автозапуск сервиса Keycloak:

systemctl enable keycloak

16. Запустите сервис Keycloak:
systemctl start keycloak

12.3. ПРОВЕРКА РАБОТОСПОСОБНОСТИ

Выполните GET-запрос к URL'у только что запущенного приложения, выполнив команду:
curl localhost:8080

Нижеприведенный ответ говорит об успешном старте приложения:

```
<!--
~ Copyright 2016 Red Hat, Inc. and/or its affiliates
~ and other contributors as indicated by the @author tags.
~
~ Licensed under the Apache License, Version 2.0 (the "License");
~ you may not use this file except in compliance with the License.
~ You may obtain a copy of the License at
~
~ http://www.apache.org/licenses/LICENSE-2.0
~
~ Unless required by applicable law or agreed to in writing, software
~ distributed under the License is distributed on an "AS IS" BASIS,

    ~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
~ See the License for the specific language governing permissions and
~ limitations under the License.
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <meta http-equiv="refresh" content="0; url=/auth/" />
    <meta name="robots" content="noindex, nofollow">
    <script type="text/javascript">
        window.location.href = "/auth/"
    </script>
</head>
<body>
    If you are not redirected automatically, follow this <a href='/auth'>link</a>.
</body>
```

12.4. СОЗДАНИЕ REALM ЧЕРЕЗ ИМПОРТ

1. Скачайте файл импорта: https://goodt_lts.hb.bizmrg.com/txt/rostalent.json
2. Перейдите в панель администрирования Keycloak, выполнив вход под учетной записью с правами администратора.
3. Нажмите на Select realm в левой верхней части окна.
4. Нажмите Add realm.
5. Нажмите Select file.
6. В диалоговом окне выберите скачанный выше файл импорта rostalent.json.
7. В поле Name введите желаемое имя realm'a. Например, rostalent.
8. Нажмите на кнопку Create.
9. Дождитесь сообщения об успешном завершении импорта.

12.5. НАСТРОЙКА РЕВЕРС-ПРОКСИ ДЛЯ KEYCLOAK

1. Добавьте следующие блоки в файл конфигурации Nginx:

```
upstream keycloak {
  server localhost:8080;
}
server {
        listen          9443 ssl http2 ;
        listen          [::]:9443 ssl http2 ;
        ssl_certificate /abc/conf/cert/fullchain.crt;
        ssl_certificate_key /abc/conf/cert/private.key;
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        ssl_ecdh_curve secp384r1;
        ssl_session_cache shared:SSL:10m;
        ssl_session_tickets off;
        ssl_stapling on;
        ssl_stapling_verify on;
        resolver 77.88.8.8 77.88.8.1 valid=300s;
        resolver_timeout 5s;
        add_header Strict-Transport-Security "max-age=63072000;
includeSubdomains; preload";
        add_header X-Content-Type-Options nosniff;
        client_max_body_size 1000M;
        proxy_read_timeout 60m;
        proxy_send_timeout 60m;

        location / {
                proxy_set_header Upgrade $http_upgrade;
```

```
                proxy_set_header Connection "upgrade";
                proxy_set_header X-Real_IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_set_header X-NginX-Proxy true;
                proxy_set_header X-Forwarded-Proto $scheme; #Без этого бэк отдает
все по http
                real_ip_header X-Real-IP;
                proxy_pass http://keycloak;
                proxy_redirect off;
        }
}
```

2. Перезапустите Nginx:
   sudo nginx -s reload
3. После этого Keycloak будет доступен по адресу: ***https://<вашсервер>:9443***

13. УСТАНОВКА ПРИЛОЖЕНИЙ НА ОТДЕЛЬНЫХ ВМ

13.1. НАСТРОЙКА КОННЕКТОРА С RSA КЛЮЧАМИ

При использовании внешних систем авторизации для формируемых в СУП приложениях необходимо учитывать использование RSA ключей - ниже описана процедура настройки коннектора для использования внешних систем авторизации (на примере модифицированного Keycloak с использованием RSA ключа).

-v ключ:/abc/conf/keys/ключ

-e RTL_KEY_RSA=ключ

13.2. УСТАНОВКА SDK RTL-DREMIO-CONNECTOR

Для запуска в Docker выполнить следующую команду:

1. В созданный файл необходимо добавить содержимое:

*docker run -d --name=dremio-connector -p 4400:4400 *
*-e "RTL_DREMIO_USER=dremio_admin" *
*-e "RTL_DREMIO_PASS=$7roNgP@s$w0r9" *
*-e "RTL_DREMIO_HOST=172.20.55.241" *
*-e "RTL_DREMIO_PORT=31010" *
*-e "RTL_APP_PORT=4400" *
*https://XXX.YYY.ru/repository/docker/v2/rtl-dremio-connector/rtl-dremio-connector:164_master *
--bind=0.0.0.0/0

2. Назначаем файлу атрибут "исполняемый":

sudo chmod +x /abc/distr/scripts/dremio-connector.sh

13.3. УСТАНОВКА GOODT-EDITOR РЕДАКТОР

Установка приложения сводится к следующим шагам:

1. Создание базы данных.
2. Создание развертывания.
3. Создание сервиса.
4. Настройка ингресса.

СОГЛАШЕНИЕ О ФОРМАТИРОВАНИИ И ИМЕНОВАНИИ.

- Имена объектов, переменных и их значения, параметры, URL, пути в файловой системе, команды и директивы выделяются разметкой "код в строку", например: *https://someurl.com, variable = value*

- Адрес и порт, на котором располагается приложение: *https://rtl.corp.ru*
- URL, по которому доступно API приложения: *https://rtl.corp.ru/api*
- URL, по которому доступен встроенный player приложения: *https://rtl.corp.ru/player*
- URL, по которому доступно приложение dremio-connector: *https://rtl.corp.ru/dremio-connector*
- URL, по которому доступно приложение Dremio: *http://agg.corp.ru:9047*
- Пароль пользователя для подключения к БД *sup_editor в СУБД MySQL: $7r0n6pP@s2WrD*
- Номер версии приложения: *555_master*
- IP-адрес СУБД MySQL*: 10.0.0.15*
- Номер версии приложения: *555_master*
- Папка на локальном сервере для хранения контента: */abc/data/editor*
- Файл SSL-сертификата расположен по пути: */abc/conf/cert/fullchain.crt*
- Файл приватного ключа к SSL-сертификату расположен по пути: */abc/conf/cert/private.key*
- Файлы конфигурации Nginx расположены по пути: */abc/conf/nginx*

ЗАПУСК ПРИЛОЖЕНИЯ.

Добавьте репозиторий docker-образов Goodt:
docker login https://XXX.YYY.ru

Выполните команду:

```
docker run -d --name=rtl-goodteditor-editor \
      -p 8090:80 \
      -v  /abc/data/editor:/abc/data/upload \
      -e "RTL_DREMIO_URL=https://rtl.corp.ru/dremio-connector" \
      -e "RTL_DREMIO_UI=http://agg.corp.ru:9047" \
      -e "RTL_API_URL=https://rtl.corp.ru/api/" \
      -e "RTL_PLAYER_URL=https://rtl.corp.ru/player/" \
      -e "RTL_SUP_DB_PASS=<пароль>\" \
      -e "RTL_SUP_DB_HOST=10.0.0.15" \
 XXX.YYY.ru /repository/docker/v2/rtl-goodteditor-editor/rtl-goodteditor-editor_b107_f617_1.2.0-101_insight:408
```

13.4. ЗАПУСК ПРИЛОЖЕНИЙ

Необходимо запустить приложения в следующем порядке:

1. NIFI;
2. SDK rtl-dremio-connector;

3. rtl-goodteditor-player;
4. Goodt-editor редактор.

Для это требуется выполнить нижеуказанные команды:
cd /abc/distr/scripts
./nifi.sh
./dremio.sh
./dremio-connector.sh
./editor.sh
./player.sh

14. ДЕПЛОЙ РЕДАКТОРА INSIGHT LOW CODE

14.1. ПОДГОТОВКА К УСТАНОВКЕ

Для установки Insight low code требуется БД postgres, имя базы по умолчанию "supeditor", имя пользователя по умолчанию "supeditor_user"

```
CREATE DATABASE supeditor;
CREATE USER supeditor_user WITH ENCRYPTED PASSWORD '{{ supeditor_user_pass }}';
GRANT ALL PRIVILEGES ON DATABASE supeditor TO supeditor_user;
\c supeditor;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO supeditor_user;
```

Для запуска редактора необходимо наличие следующих файлов конфигураций:
application-customization.yml - содержит настройки бэкенда
production.json - содержит настройки редактора
worker.json - содержит настройки авторизации редактора
production.local.json (плеера) - содержит настройки плеера

Создайте папки:
mkdir -p /abc/conf/ /abc/data/supeditor

14.2. УСТАНОВКА В DOCKER

Необходимо создать 4 файла конфигураций и запустить контерйнер с указанными параметрами.

1. Файл application-customization.yml содержит настройки бэкенда

Расположение: /abc/app/goodt/backend/application-customization.yml
В нем присутствуют переменные:
{{DB_URL}} - jdbc адрес подключения к БД (например, jdbc:postgresql://db.corp.ru:5432/supeditor)
{{DB_USER}} - имя пользователя БД
{{DB_PASS}} - пароль пользователя БД
{{KCLK_URL}} - URL-адрес keycloak
{{KCLK_REALM}} - имя realm
{{KCLK_CLID}} - ID клиента
{{KCLK_CLSECRET}} - secret клиента
{{KCLK_USER}} - пользователь keycloak
{{KCLK_PASS}} - пароль пользователя keycloak
Переменные должны быть заменены вместе с фигурными скобками на настройки стенда.
Создайте в папке /abc/conf/ файл application-customization.yml следующего содержания, заменив переменные:

```
spring:
  profiles: customization
  jpa:
        show-sql: true
        properties:
          hibernate:
                format_sql: true
                enable_lazy_load_no_trans: true
                naming:
                  physical-strategy:
com.goodt.drive.orgstructure.application.utils.SnakePhysicalNamingStrategy
        hibernate:
          ddl-auto: none
        database-platform: org.hibernate.dialect.PostgreSQL9Dialect
  datasource:
        driver-class-name: org.postgresql.Driver
        url: {{DB_URL}}
        username: {{DB_USER}}
        password: {{DB_PASS}}
  liquibase:
        change-log: classpath:db/changelog/changelog.xml
appConfig:
  orgstructure-service:
        host: {{BACKEND_ORGSTRUCTURE_URL}}
  tasksetting-service:
        host: {{BACKEND_TASKSETTING_URL}}
  version: 1.0_alpha
  environment: ${spring.profiles}
  dbDriver: ${spring.datasource.driver-class-name}
  dbUrl: ${spring.datasource.url}
  dbUsername: ${spring.datasource.username}
```

```
  dbPassword: ${spring.datasource.password}
  keyCloak:
            using:
              baseUrl: {{KCLK_URL}}/auth/realms/{{KCLK_REALM}}
              clientId: {{KCLK_CLID}}
              clientSecret: {{KCLK_CLSECRET}}
              serviceUsername: {{KCLK_USER}}
              servicePassword: {{KCLK_PASS}}
      api: {{KCLK_URL}}/auth/admin/realms/{{KCLK_REALM}}
security:
  basic:
            enabled: false
  oauth2:
            client:
              clientId: ${appConfig.keyCloak.using.clientId}
              clientSecret: ${appConfig.keyCloak.using.clientSecret}
              accessTokenUri: ${appConfig.keyCloak.using.baseUrl}/protocol/openid-
connect/token
              userAuthorizationUri: ${appConfig.keyCloak.using.baseUrl}/protocol/openid-
connect/auth
              authorizedGrantTypes: code token
              scope: local
              username: ${appConfig.keyCloak.using.serviceUsername}
              password: ${appConfig.keyCloak.using.servicePassword}
            resource:
              userInfoUri: ${appConfig.keyCloak.using.baseUrl}/protocol/openid-connect/userinfo
app:
  logging:
            path: /abc/logs/supeditor
project:
  path: /abc/data/supeditor
```

2.1 Файл production.local.json (production.local.json при монтировании)
/abc/app/goodt/player/config/production.local.json - содержит настройки плеера
В нем присутствуют переменные:
{{DREMIO_URL}} - URL-адрес дремио-коннектора

Создайте в папке /abc/conf/ файл production.json следующего содержания, заменив переменные:

```
{
"api": {
            "wfm": {
                        "baseURL": "http://localhost:8080"
            },
            "dremio": {
                        "baseURL": "{{DREMIO_URL}}"
```

```
        }
},
"services": [
        {
                "id": "Dremio",
                "options": {
                        "baseURL": "{{DREMIO_URL}}"
                }
        }
],
"assets": {
        "cacheKey": ""
},
"log": {
        "routeMonitor": {
                "enabled": false,
                "url": "https://localhost:3000/ping"
        }
}
}
```

2.2 Файл worker.json
/abc/app/goodt/editor/config/worker.json - содержит настройки авторизации редактора
В нем присутствуют переменные:
{{KCLK_URL - URL}} -адрес keycloak
{{KCLK_REALM}} - имя realm
{{API_URL}} - адрес API бэкенда, указывается без эндпоинта /api
{{KCLK_CLID}} - ID клиента

Переменные должны быть заменены вместе с фигурными скобками на настройки стенда.

Создайте в папке /abc/conf/ файл worker.json следующего содержания, заменив переменные:

```
 {
 "auth": {
        "url": "{{KCLK_URL}}/auth/",
        "realm": "{{KCLK_REALM}}",
        "clientId": "{{KCLK_CLID}}"
        },
 "api": {
        "url": "{{API_URL}}"
        }
 }
```

3. Файл настройки Плеера:
/abc/app/goodt/editor/config/production.local.json - содержит настройки плеера
В нем присутствуют переменные:
{{API_URL}} - адрес API бэкенда, указывается без эндпоинта /api
{{DREMIO_UI}} - URL-адрес веб-интерфейса Dremio
{{DREMIO_URL}} - URL-адрес дремио-коннектора
{{PLAYER_URL}} - URL-адрес плеера

Переменные должны быть заменены вместе с фигурными скобками на настройки стенда.
Создайте в папке /abc/conf/ файл production.local.json следующего содержания, заменив переменные:

```
{
"api": {
"http": {
"baseURL": "{{API_URL}}/api/",
"timeout": 360000,
"withCredentials": false
},
"wfm": {
"baseURL": "http://localhost:8080/"
},
"dremio": {
"uiUrl": "{{DREMIO_UI}}"
}
},
"services": [
{
"id": "Dremio",
"options": {
"baseURL": "{{DREMIO_URL}}"
}
}
],
"player": {
"url": "{{PLAYER_URL}}"
},
"widget-editor": {
"url": "http://localhost:8080/#",
"methods": {
"create": "create",
"edit": "edit?templateId=:id"
}
}
}
```

4. Запуск контейнера редактора

В нем присутствуют переменные:
{{TAG}} - ссылка на Docker-образ редактора в Docker-репозитории
Переменные должны быть заменены вместе с фигурными скобками на настройки стенда.

Запустите контейнер следующей командой, заменив переменные:

```
docker run -d --name=rtl-goodteditor-editor-java \
                    -p 80:80 -p 8097:8097 \
      -v   /abc/data/supeditor:/abc/data/supeditor \
      -v   /abc/conf/application-customization.yml:/abc/app/goodt/backend/application-
customization.yml \
      -v   /abc/conf/production.local.json:/abc/app/goodt/editor/config/production.local.json \
      -v   /abc/conf/worker.json:/abc/app/goodt/editor/config/worker.json \
      -v   /abc/conf/production.json:/abc/app/goodt/player/config/production.local.json \
                    -e "RTL_PROFILE=prod" \
      {{TAG}}
```

14.3. НАСТРОЙКА NGINX

Присутствуют переменные:
{{HOST}} - доменное имя, привязанное к редактору
Переменные должны быть заменены вместе с фигурными скобками на настройки стенда.

Создайте файл editor.conf в папке файлов конфигурации Nginx:

```
server {
  listen 443 ssl http2 ;
  listen [::]:443 ssl http2 ;
  server_name {{HOST}};
  ssl_certificate /abc/conf/cert/fullchain.crt;
  ssl_certificate_key /abc/conf/cert/private.key;
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
  ssl_ecdh_curve secp384r1;
  ssl_session_cache shared:SSL:10m;
  ssl_session_tickets off;
  ssl_stapling on;
  ssl_stapling_verify on;
  resolver 8.8.8.8 8.8.4.4 valid=300s;
  resolver_timeout 5s;
  add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload";
  add_header X-Content-Type-Options nosniff;
  client_max_body_size 1000M;
  proxy_read_timeout 60m;
  proxy_send_timeout 60m;
```

```
location / {
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header X-Real_IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;
    proxy_set_header X-Forwarded-Proto $scheme;
    real_ip_header X-Real-IP;
    proxy_pass localhost:80;
    proxy_redirect off;
}

location /editor/.
{
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header X-Real_IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;
    proxy_set_header X-Forwarded-Proto $scheme;
    real_ip_header X-Real-IP;
    proxy_pass localhost:80;
    proxy_redirect off;
 }

location /api
{
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header X-Real_IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;
    proxy_set_header X-Forwarded-Proto $scheme;
    real_ip_header X-Real-IP;
    proxy_pass localhost:8097;
    proxy_redirect off;

 access_log /abc/logs/nginx/editor-access.log;
 error_log /abc/logs/nginx/editor-error.log;
 }

location /player
{
```

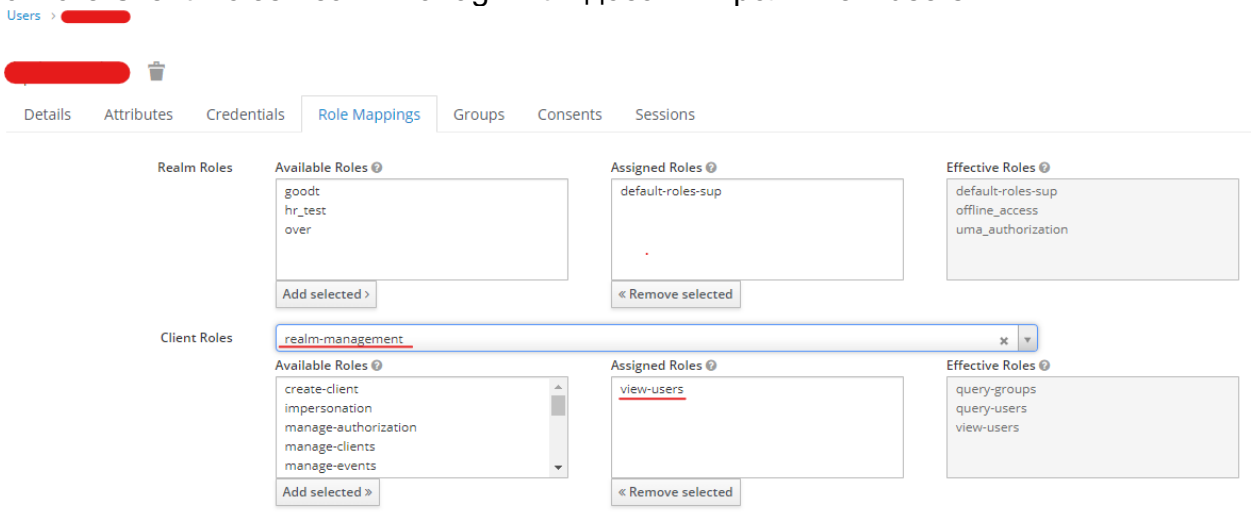
```
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header X-Real_IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;
    proxy_set_header X-Forwarded-Proto $scheme;
    real_ip_header X-Real-IP;
    proxy_pass localhost:80;
    proxy_redirect off;

 access_log /abc/logs/nginx/editor-access.log;
 error_log /abc/logs/nginx/editor-error.log;
 }
}
```

Проверьте работоспособность Nginx после изменения конфигурации:
nginx -t

Вывод, приведенный ниже, говорит о корректной конфигурации:
nginx: the configuration file /abc/conf/nginx/nginx.conf syntax is ok
nginx: configuration file /abc/conf/nginx/nginx.conf test is successful

Перезапустите Nginx:
nginx -s reload

Важно, чтоб у пользователя {{KCLK_USER}}, заведенного в Keycloak были выданы права на просмотр групп пользователей.
Для этого в настройках пользователя в разделе Role Mapping выбрать в выпадающем списке Client Roles: realm-managmint и добавить роль view-users



14.4. УСТАНОВКА В КЛАСТЕРЕ KUBERNETES (K8S)

Необходимо создать 3 файла конфигмапов для передачи конфигураций, а также все остальные манифесты.

1. Файл application-customization.yml содержит настройки бэкенда

Расположение: /abc/app/goodt/backend/application-customization.yml
В нем присутствуют переменные:
{{DB_URL}} - jdbc адрес подключения к БД (например, jdbc:postgresql://db.corp.ru:5432/supeditor)
{{DB_USER}} - имя пользователя БД
{{DB_PASS}} - пароль пользователя БД
{{KCLK_URL}} - URL-адрес keycloak
{{KCLK_REALM}} - имя realm
{{KCLK_CLID}} - ID клиента
{{KCLK_CLSECRET}} - secret клиента
{{KCLK_USER}} - пользователь keycloak
{{KCLK_PASS}} - пароль пользователя keycloak
Переменные должны быть заменены вместе с фигурными скобками на настройки стенда.
Создайте файл application-customization.yml следующего содержания, заменив переменные:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: application-customization-editor
data:
  RTL_PROFILE: "prod"
  application-customization.yml: |
    spring:
      profiles: customization
      jpa:
        show-sql: true
        properties:
          hibernate:
            format_sql: true
            enable_lazy_load_no_trans: true
            naming:
              physical-strategy:
com.goodt.drive.orgstructure.application.utils.SnakePhysicalNamingStrategy
        hibernate:
          ddl-auto: none
        database-platform: org.hibernate.dialect.PostgreSQL9Dialect
      datasource:
        driver-class-name: org.postgresql.Driver
        url: {{DB_URL}}
```

```
      username: {{DB_USER}}
      password: {{DB_PASS}}
    liquibase:
      change-log: classpath:db/changelog/changelog.xml
  appConfig:
    orgstructure-service:
      host: {{BACKEND_ORGSTRUCTURE_URL}}
    tasksetting-service:
      host: {{BACKEND_TASKSETTING_URL}}
    version: 1.0_alpha
    environment: ${spring.profiles}
    dbDriver: ${spring.datasource.driver-class-name}
    dbUrl: ${spring.datasource.url}
    dbUsername: ${spring.datasource.username}
    dbPassword: ${spring.datasource.password}
    keyCloak:
      using:
        baseUrl: {{KCLK_URL}}/auth/realms/{{KCLK_REALM}}
        clientId: {{KCLK_CLID}}
        clientSecret: {{KCLK_CLSECRET}}
        serviceUsername: {{KCLK_USER}}
        servicePassword: {{KCLK_PASS}}
        api: {{KCLK_URL}}/auth/admin/realms/{{KCLK_REALM}}
  security:
    basic:
      enabled: false
    oauth2:
      client:
        clientId: ${appConfig.keyCloak.using.clientId}
        clientSecret: ${appConfig.keyCloak.using.clientSecret}
        accessTokenUri: ${appConfig.keyCloak.using.baseUrl}/protocol/openid-connect/token
        userAuthorizationUri: ${appConfig.keyCloak.using.baseUrl}/protocol/openid-
connect/auth
        authorizedGrantTypes: code token
        scope: local
        username: ${appConfig.keyCloak.using.serviceUsername}
        password: ${appConfig.keyCloak.using.servicePassword}
      resource:
        userInfoUri: ${appConfig.keyCloak.using.baseUrl}/protocol/openid-connect/userinfo
  app:
    logging:
      path: /abc/logs/supeditor
  project:
    path: /abc/data/supeditor
```

2. Файлы настройки самого редактора:
/abc/app/goodt/player/config/production.local.json - содержит настройки плеера и

В нем присутствуют переменные:
{{DREMIO_URL}} - URL-адрес дремио-коннектора

/abc/app/goodt/editor/config/worker.json - содержит настройки авторизации редактора
В нем присутствуют переменные:
{{KCLK_URL - URL}} -адрес keycloak
{{KCLK_REALM}} - имя realm
{{API_URL}} - адрес API бэкенда, указывается без эндпоинта /api

Переменные должны быть заменены вместе с фигурными скобками на настройки стенда.

Создайте файл config-editor.yaml следующего содержания, заменив переменные:
(общий конфигмап на 2 файла)

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: config-editor
data:
  production.json: |
     {
       "api": {
          "wfm": {
             "baseURL": "http://localhost:8080"
          },
          "dremio": {
             "baseURL": "{{DREMIO_URL}}"
          }
       },
       "services": [
          {
             "id": "Dremio",
             "options": {
                "baseURL": "{{DREMIO_URL}}"
             }
          }
       ],
       "assets": {
          "cacheKey": ""
       },
       "log": {
          "routeMonitor": {
             "enabled": false,
             "url": "https://localhost:3000/ping"
          }
       }
```

```
    }
  worker.json: |
    {
    "auth": {
      "url": "{{KCLK_URL}}/auth/",
      "realm": "{{KCLK_REALM}}",
      "clientId": "player"
      },
    "api": {
      "url": "{{API_URL}}"
      }
    }
```

3. Файл настройки Плеера:
/abc/app/goodt/editor/config/production.local.json - содержит настройки редактора и
В нем присутствуют переменные:
{{API_URL}} - адрес API бэкенда, указывается без эндпоинта /api
{{DREMIO_UI}} - URL-адрес веб-интерфейса Dremio
{{DREMIO_URL}} - URL-адрес дремио-коннектора
{{PLAYER_URL}} - URL-адрес плеера

Переменные должны быть заменены вместе с фигурными скобками на настройки стенда.
Создайте файл production-local-json.yaml следующего содержания, заменив переменные:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: production-local-json-editor
data:
  production.local.json: |
    {
    "api": {
    "http": {
    "baseURL": "{{API_URL}}/api/",
    "timeout": 360000,
    "withCredentials": false
    },
    "wfm": {
    "baseURL": "http://localhost:8080/"
    },
    "dremio": {
    "uiUrl": "{{DREMIO_UI}}"
    }
    },
    "services": [
```

```
{
"id": "Dremio",
"options": {
"baseURL": "{{DREMIO_URL}}"
}
}
],
"player": {
"url": "{{PLAYER_URL}}"
},
"widget-editor": {
"url": "http://localhost:8080/#",
"methods": {
"create": "create",
"edit": "edit?templateId=:id"
}
}
}
```

4. Деплоймент редактора

В нем присутствуют переменные:
{{CLUSTER_NAME}} - имя кластера в который производится деплой
{{CLIENT}} - имя неймспейса в который производится деплой
{{BUILD_VERSION}} - Версия редактора
{{TAG}} - ссылка на Docker-образ редактора в Docker-репозитории
Переменные должны быть заменены вместе с фигурными скобками на настройки стенда.

Создайте файл deployment.yaml следующего содержания, заменив переменные:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rtl-goodteditor-editor-java
  namespace: {{CLIENT}}
  labels:
    environment: {{CLUSTER_NAME}}
    version: {{BUILD_VERSION}}
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rtl-goodteditor-editor-java
  template:
    metadata:
      labels:
```

```
    app: rtl-goodteditor-editor-java
    environment: {{CLUSTER_NAME}}
    version: {{BUILD_VERSION}}
spec:
  volumes:
    - name: rtl-goodteditor-editor-java-pv
      persistentVolumeClaim:
        claimName: rtl-goodteditor-editor-java-pvc
    - name: application-customization-editor-cm
      configMap:
        defaultMode: 511
        name: application-customization-editor
    - name: config-editor-cm
      configMap:
        defaultMode: 511
        name: config-editor
    - name: production-local-json-editor-cm
      configMap:
        defaultMode: 511
        name: production-local-json-editor
  imagePullSecrets:
    - name: art-goodt-me
  containers:
  - name: rtl-goodteditor-editor
    image: {{TAG}}
    ports:
    - containerPort: 8097
      protocol: TCP
    - containerPort: 80
      protocol: TCP
    volumeMounts:
    - mountPath: "/abc/data/supeditor"
      name: rtl-goodteditor-editor-java-pv
    - mountPath: /abc/app/goodt/backend/application-customization.yml
      name: application-customization-editor-cm
      subPath: application-customization.yml
    - mountPath: /abc/app/goodt/editor/config/production.local.json
      name: production-local-json-editor-cm
      subPath: production.local.json
    - mountPath: /abc/app/goodt/editor/config/worker.json
      name: config-editor-cm
      subPath: worker.json
    - mountPath: /abc/app/goodt/player/config/production.local.json
      name: config-editor-cm
      subPath: production.json
    env:
    - name: RTL_PROFILE
```

```
      valueFrom:
        configMapKeyRef:
          name: application-customization-editor
          key: RTL_PROFILE
    readinessProbe:
      failureThreshold: 5
      initialDelaySeconds: 60
      periodSeconds: 10
      successThreshold: 1
      tcpSocket:
        port: 8097
      timeoutSeconds: 1
    livenessProbe:
      failureThreshold: 3
      initialDelaySeconds: 90
      periodSeconds: 10
      successThreshold: 1
      tcpSocket:
        port: 8097
      timeoutSeconds: 1
```

5. Сервис редактора

Создайте файл service.yaml следующего содержания:

```
apiVersion: v1
kind: Service
metadata:
  name: rtl-goodteditor-editor-java-svc
spec:
  type: ClusterIP
  selector:
    app: rtl-goodteditor-editor-java
  ports:
    - name: "port-80-to-80"
      port: 80
      targetPort: 80
      protocol: TCP
    - name: "port-8097-to-8097"
      port: 8097
      targetPort: 8097
      protocol: TCP
```

6. Ingress редактора
В нем присутствуют переменные:
{{HOST}} - доменное имя, привязанное к редактору
{{SECRET}} - имя сикрета, содержащего SSL/TLS сертификаты

Создайте файл ingress.yaml следующего содержания:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: editor-2-no-rewrite
  annotations:
    ingress.kubernetes.io/ssl-redirect: "true"
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/proxy-body-size: 1024m
spec:
  tls:
  - hosts:
      - {{HOST}}
    secretName: {{SECRET}}
  rules:
  - host: {{HOST}}
    http:
      paths:
      - path: /api
        pathType: Prefix
        backend:
          service:
            name: rtl-goodteditor-editor-java-svc
            port:
              number: 8097
      - path: /player
        pathType: Prefix
        backend:
          service:
            name: rtl-goodteditor-editor-java-svc
            port:
              number: 80
      - path: /editor
        pathType: Prefix
        backend:
          service:
            name: rtl-goodteditor-editor-java-svc
            port:
              number: 80
      - path: /p
        pathType: Prefix
        backend:
          service:
            name: rtl-goodteditor-editor-java-svc
            port:
```

```
      number: 80
```

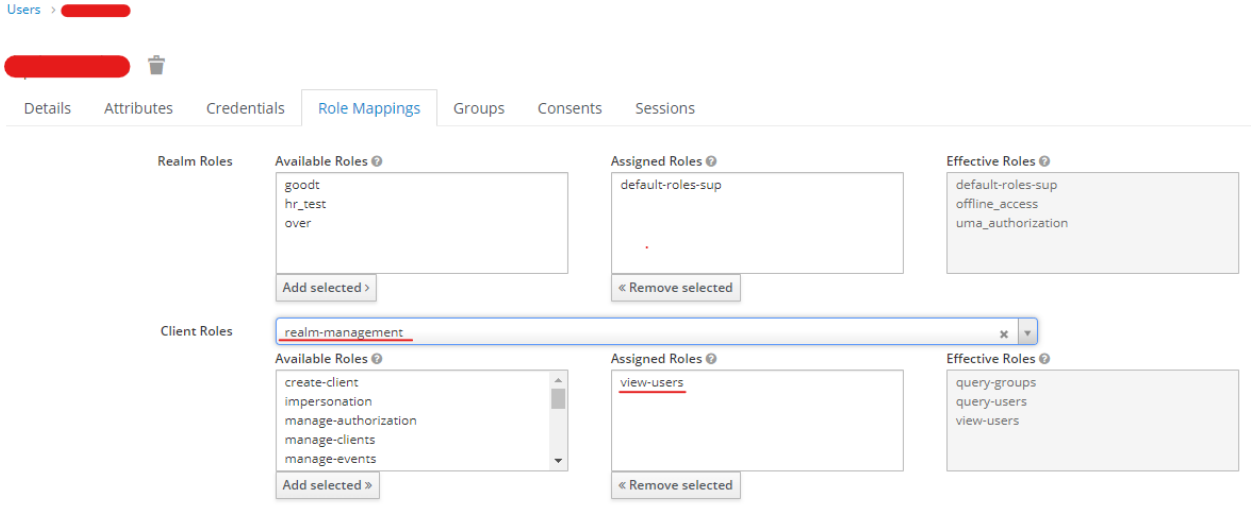
7. PVC редактора

Создайте файл pvc.yaml следующего содержания:
При этом storageClassName: должен быть уточнен в зависимости от представленных в данном кластере классов.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: rtl-goodteditor-editor-java-pvc
  labels:
    app: rtl-goodteditor-editor-java-pvc
spec:
  storageClassName: csi-ceph-hdd-ms1-retain
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
```

8. Передайте в Kubernetes API приведенные выше *.yaml манифесты в соответствии с официальной документацией к Kubernetes.

Важно, чтоб у пользователя {{KCLK_USER}}, заведенного в Keycloak были выданы права на просмотр групп пользователей.
Для этого в настройках пользователя в разделе Role Mapping выбрать в выпадающем списке Client Roles: realm-managmint и добавить роль view-users



15. РАЗВЕРТЫВАНИЕ INSIGHT LOW CODE В KUBERNETES. + COLLABORATE

Раздел посвящен базовому развертыванию программных компонентов Insight low code.

Для успешного выполнения развертывания необходима учетная запись в репозитории Docker-образов Goodt

В базовый Insight low code входят следующие программные компоненты:

1. Приложение Отчеты (rtl-report);
2. Приложение Файловый загрузчик (rtl-fileupload);
3. Приложение dremio-connector;
4. Приложение collaborate-service;

Установка приложения сводится к следующим шагам:

1. Создание базы данных.
2. Создание развертывания.
3. Создание сервиса.
4. Настройка ингресса.

15.1. УСТАНОВКА ПРИЛОЖЕНИЯ ФАЙЛОВЫЙ ЗАГРУЗЧИК (RTL-FILEUPLOAD).

Установка приложения сводится к следующим шагам:

1. Создание базы данных.
2. Создание развертывания.
3. Создание сервиса.
4. Настройка ингресса.

СОГЛАШЕНИЕ О ФОРМАТИРОВАНИИ И ИМЕНОВАНИИ.

- Имена объектов, переменных и их значения, параметры, URL, пути в файловой системе, команды и директивы выделяются разметкой "код в строку", например: https://someurl.com, variable = value
- Доменное имя, на котором располагаются приложения: rtl.corp.ru
- Адрес Keycloak: auth.corp.ru
- Маршрут, по которому располагается приложение: /rtl-api/fileupload
- Адрес хранилища WEBDAV: https://rtl.corp.ru/remote.php/dav/files/corp-user/HCM/rtl-api/fileupload
- Имя пользователя хранилища WEBDAV: corp-WEBDAV-user

- Пароль пользователя хранилища WEBDAV: PaPaPas7@
- Адрес хранилища MINIO: https://rtl.corp.ru/MINIO
- Имя BUCKET хранилища MINIO: MINIO_BUCKETNAME
- Имя пользователя хранилища MINIO: corp-MINIO-user
- Пароль пользователя хранилища MINIO: MiMiMiPas7@
- Пространство имен в Keycloak: insight
- Имя пользователя в Keycloak: api-service
- Пароль пользователя api-service в Keycloak: k3ycl0akU$3r
- Идентификатор клиента в Keycloak: api-service-agent
- UUID клиента api-service-agent в Keycloak: 00000000-0000-0000-0000-000000000000
- Номер версии приложения: 555_master
- Во всех кластерах SSL/TLS сертификат на домен и поддомены (wildcard) хранится в secret'e corp-ru-tls
- Во всех кластерах ключ key.rsa соответствующий пространству имен в Keycloak хранится в secret'e goodt-key-rsa

СОЗДАНИЕ БАЗЫ ДАННЫХ.

В консоли psql выполните команды:

```
CREATE DATABASE fileupload;
CREATE USER fileupload_user WITH ENCRYPTED PASSWORD '$7r0n6pP@s2WrD';
GRANT ALL PRIVILEGES ON DATABASE fileupload TO fileupload_user;
\c fileupload;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO fileupload_user;
```

СОЗДАНИЕ РАЗВЕРТЫВАНИЯ.

Создайте файл deployment.yaml со следующим содержимым:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rtl-fileupload
  labels:
    version: 555_master
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rtl-fileupload
  template:
    metadata:
```

```
    labels:
      app: rtl-fileupload
      version: 555_master
  spec:
    imagePullSecrets:
      - name: art-goodt-me
    containers:
    - name: rtl-fileupload
      image: art.goodt.me/rtl-fileupload/rtl-fileupload:555_master
      ports:
      - containerPort: 8099
        protocol: TCP
      volumeMounts:
      - mountPath: /abc/app/goodt/fileupload/key.rsa
        name: goodt-key-rsa-sec
        subPath: key.rsa
      env:
      - name: RTL_PROFILE
        value: "prod"
      - name: RTL_KCLK_URL
        value: "https://auth.corp.ru"
      - name: RTL_KCLK_REALM
        value: "insight"
      - name: RTL_KCLK_USER
        value: "api-service"
      - name: RTL_KCLK_PASS
        value: "k3ycl0akU$3r"
      - name: RTL_KCLK_CLID
        value: "api-service-agent"
      - name: RTL_KCLK_CLSECRET
        value: "00000000-0000-0000-0000-000000000000"
      - name: RTL_USE_MINIO
        value: "false"
      - name: RTL_USE_FILESYSTEM
        value: "false"
      - name: RTL_USE_WEBDAV
        value: "true"
      - name: RTL_MINIO_URL
        value: "https://rtl.corp.ru/MINIO"
      - name: RTL_MINIO_BUCKETNAME
        value: "MINIO_BUCKETNAME"
      - name: RTL_MINIO_USER
        value: "corp-MINIO-user"
      - name: RTL_MINIO_PASS
        value: "MiMiMiPas7@"
      - name: RTL_WEBDAV_URL
        value: "https://rtl.corp.ru/remote.php/dav/files/corp-WEBDAV-user/HCM"
```

```
      - name: RTL_WEBDAV_USER
        value: "corp-WEBDAV-user"
      - name: RTL_WEBDAV_PASS
        value: "PaPaPas7@"
      - name: RTL_URN
        value: "/rtl-api/rtl-fileupload/api/fileupload"
      - name: RTL_KCLK_AUTH_ENABLED
        value: "false"
    volumes:
    - name: goodt-key-rsa-sec
      secret:
        secretName: goodt-key-rsa
        defaultMode: 256
```

Переменные: RTL_USE_MINIO, RTL_USE_FILESYSTEM, RTL_USE_WEBDAV используются для определения используемого хранилища. Только одна из них может иметь значение "true", остальные должны иметь значение "false"

СОЗДАНИЕ PVC (ТОЛЬКО ПРИ ИСПОЛЬЗОВАНИИ RTL_USE_FILESYSTEM).

Создайте файл pvc.yaml со следующим содержимым:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: rtl-fileupload-pvc
  labels:
    app: rtl-fileupload-pvc
spec:
  storageClassName: nas-hdd
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
```

СОЗДАНИЕ СЕРВИСА.

Создайте файл service.yaml со следующим содержимым:

```
apiVersion: v1
kind: Service
metadata:
  name: rtl-fileupload-svc
spec:
  type: ClusterIP
```

```
  selector:
    app: rtl-fileupload
  ports:
    - name: "port-8099-to-8099"
      port: 8099
      targetPort: 8099
      protocol: TCP
```

НАСТРОЙКА ИНГРЕССА.

Создайте файл fileupload-rewrite.yaml со следующим содержимым:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: rtl-fileupload-rewrite
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
    ingress.kubernetes.io/ssl-redirect: "true"
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/proxy-body-size: 1024m
spec:
  tls:
  - hosts:
      - rtl.corp.ru
    secretName: corp-ru-tls
  rules:
  - host: rtl.corp.ru
    http:
      paths:
      - path: /rtl-api/fileupload(/|$)(.*)
        pathType: Prefix
        backend:
          service:
            name: rtl-fileupload-svc
            port:
              number: 8099
```

Создайте файл fileupload-no-rewrite.yaml со следующим содержимым:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: rtl-fileupload-no-rewrite
  namespace: default
  annotations:
```

```
    nginx.ingress.kubernetes.io/backend-protocol: "https"
    ingress.kubernetes.io/ssl-redirect: "false"
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/proxy-body-size: 1024m
    nginx.ingress.kubernetes.io/preserve-trailing-slash: "false"
spec:
  tls:
  - hosts:
      - rtl.corp.ru
    secretName: corp-ru-tls
  rules:
  - host: rtl.corp.ru
    http:
      paths:
      - path: /static
        pathType: Prefix
        backend:
          service:
            name: rtl-fileupload-svc
            port:
              number: 443
```

ЗАВЕРШАЮЩИЕ ДЕЙСТВИЯ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ.

Передайте в Kubernetes API приведенные выше манифесты deployment.yaml, service.yaml, fileupload.yaml в соответствии с официальной документацией к Kubernetes.

Выполните GET-запрос к URL'у только что запущенного приложения, выполнив команду:

```
curl -k https://rtl.corp.ru/rtl-api/fileupload/api/v1
```

Нижеприведенный ответ говорит об успешном старте приложения:

```
Authentication Failed
```

15.2. УСТАНОВКА ПРИЛОЖЕНИЯ ОТЧЕТЫ (RTL-REPORT).

Установка приложения сводится к следующим шагам:

1. Создание базы данных.
2. Создание развертывания.
3. Создание сервиса.
4. Настройка ингресса.

СОГЛАШЕНИЕ О ФОРМАТИРОВАНИИ И ИМЕНОВАНИИ.

- Имена объектов, переменных и их значения, параметры, URL, пути в файловой системе, команды и директивы выделяются разметкой "код в строку", например: https://someurl.com, variable = value
- Доменное имя, на котором располагаются приложения: rtl.corp.ru
- Адрес и порт СУБД: db.corp.ru:5432
- Адрес Keycloak: auth.corp.ru
- Маршрут, по которому располагается приложение: /rtl-api/report
- Имя базы данных для приложения: report
- Имя пользователя для подключения к базе данных: report_user
- Пароль пользователя report_user: $7r0n6pP@s2WrD
- Строка jdbc-подключения к базе данных: jdbc:postgresql://db.corp.ru:5432/report
- Пространство имен в Keycloak: insight
- Имя пользователя в Keycloak: api-service
- Пароль пользователя api-service в Keycloak: k3ycl0akU$3r
- Идентификатор клиента в Keycloak: api-service-agent
- UUID клиента api-service-agent в Keycloak: 00000000-0000-0000-0000-000000000000
- Номер версии приложения: 555_master
- URL приложения Оргструктура: https://rtl.corp.ru/rtl-api/orgstructure
- URL приложения dremio-connector: https://rtl.corp.ru/rtl-api/dremio-connector
- Во всех кластерах SSL/TLS сертификат на домен и поддомены (wildcard) хранится в secret'e corp-ru-tls

СОЗДАНИЕ БАЗЫ ДАННЫХ.

В консоли psql выполните команды:

```
CREATE DATABASE report;
CREATE USER report_user WITH ENCRYPTED PASSWORD '$7r0n6pP@s2WrD';
GRANT ALL PRIVILEGES ON DATABASE report TO report_user;
\c report;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO report_user;
```

СОЗДАНИЕ РАЗВЕРТЫВАНИЯ.

Создайте файл deployment.yaml со следующим содержимым:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rtl-report
  labels:
    version: 555_master
```

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rtl-report
  template:
    metadata:
      labels:
        app: rtl-report
        version: 555_master
    spec:
      imagePullSecrets:
        - name: art-goodt-me
      containers:
      - name: rtl-report
        image: art.goodt.me/rtl-report/rtl-report:555_master
        ports:
        - containerPort: 8089
          protocol: TCP
        env:
        - name: RTL_DB_USER
          value: "report_user"
        - name: RTL_DB_PASS
          value: "$7r0n6pP@s2WrD"
        - name: RTL_DB_URL
          value: "jdbc:postgresql://db.corp.ru:5432/report"
        - name: RTL_PROFILE
          value: "prod"
        - name: RTL_KCLK_URL
          value: "https://auth.corp.ru"
        - name: RTL_KCLK_REALM
          value: "insight"
        - name: RTL_KCLK_USER
          value: "api-service"
        - name: RTL_KCLK_PASS
          value: "k3ycl0akU$3r"
        - name: RTL_KCLK_CLID
          value: "api-service-agent"
        - name: RTL_KCLK_CLSECRET
          value: "00000000-0000-0000-0000-000000000000"
        - name: RTL_BACKEND_ORGSTRUCTURE_URL
          value: "https://rtl.corp.ru/rtl-api/orgstructure"
        - name: RTL_DREMIOCONNECTOR_URL
          value: "https://rtl.corp.ru/rtl-api/dremio-connector"
```

СОЗДАНИЕ СЕРВИСА.

Создайте файл service.yaml со следующим содержимым:

```
apiVersion: v1
kind: Service
metadata:
  name: rtl-report-svc
spec:
  type: ClusterIP
  selector:
    app: rtl-report
  ports:
    - name: "port-8089-to-8089"
      port: 8089
      targetPort: 8089
      protocol: TCP
```

НАСТРОЙКА ИНГРЕССА.

Создайте файл report.yaml со следующим содержимым:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: rtl-report-rewrite
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
    ingress.kubernetes.io/ssl-redirect: "true"
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/proxy-body-size: 1024m
spec:
  tls:
  - hosts:
      - rtl.corp.ru
    secretName: corp-ru-tls
  rules:
  - host: rtl.corp.ru
    http:
      paths:
      - path: /rtl-api/report(/|$)(.*)
        pathType: Prefix
        backend:
          service:
            name: rtl-report-svc
            port:
              number: 8089
```

ЗАВЕРШАЮЩИЕ ДЕЙСТВИЯ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ.

Передайте в Kubernetes API приведенные выше манифесты deployment.yaml, service.yaml, report.yaml в соответствии с официальной документацией к Kubernetes.

Выполните GET-запрос к URL'у только что запущенного приложения, выполнив команду:

curl -k https://rtl.corp.ru/rtl-api/report/api/v1

Нижеприведенный ответ говорит об успешном старте приложения:

Authentication Failed

15.3. УСТАНОВКА ПРИЛОЖЕНИЯ DREMIO-CONNECTOR.

Установка приложения сводится к следующим шагам:

1. Создание развертывания.
2. Создание сервиса.
3. Настройка ингресса.

СОГЛАШЕНИЕ О ФОРМАТИРОВАНИИ И ИМЕНОВАНИИ.

- Имена объектов, переменных и их значения, параметры, URL, пути в файловой системе, команды и директивы выделяются разметкой "код в строку", например: https://someurl.com, variable = value
- Доменное имя, на котором располагаются приложения: rtl.corp.ru
- Маршрут, по которому располагается приложение: /rtl-api/dremio-connector
- Имя пользователя для подключения к Dremio: dc_user
- Пароль пользователя dc_user: $7r0n6pP@s2WrD
- FQDN узла, на котором установлен Dremio: agg.corp.ru
- Порт, на котором Dremio принимает jdbc-соединения: 31010
- Порт, на котором приложение dremio-connector принимает входящие запросы: 4000
- Имя файла с rsa-ключом: key.rsa
- Номер версии приложения: 555_master
- Во всех кластерах SSL/TLS сертификат на домен и поддомены (wildcard) хранится в secret'e corp-ru-tls
- Во всех кластерах ключ key.rsa соответствующий пространству имен в Keycloak хранится в secret'e goodt-key-rsa

СОЗДАНИЕ РАЗВЕРТЫВАНИЯ.

Создайте файл deployment.yaml со следующим содержимым:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rtl-dremio-connector
  labels:
    version: 555_master
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rtl-dremio-connector
  template:
    metadata:
      labels:
        app: rtl-dremio-connector
        version: 555_master
    spec:
      imagePullSecrets:
        - name: art-goodt-me
      containers:
      - name: rtl-dremio-connector
        image: art.goodt.me/rtl-dremio-connector/rtl-dremio-connector:555_master
        ports:
        - containerPort: 4000
          protocol: TCP
        volumeMounts:
        - mountPath: /abc/app/goodt/dremio-connector/key.rsa
          name: goodt-key-rsa-sec
          subPath: key.rsa
        env:
          - name: RTL_DREMIO_USER
            value: "@dc_user"
          - name: RTL_DREMIO_PASS
            value: "$7r0n6pP@s2WrD"
          - name: RTL_DREMIO_HTTP_URL
            value: "http://agg.corp.ru:9047"
          - name: RTL_DREMIO_JDBC_URL
            value: "agg.corp.ru:31010"
          - name: RTL_APP_PORT
            value: "4000"
          - name: RTL_KEY_RSA
            value: "key.rsa"
          - name: RTL_DREMIO_AUTH
            value: "false"
      volumes:
```

```
    - name: goodt-key-rsa-sec
      secret:
        secretName: goodt-key-rsa
        defaultMode: 256
```

СОЗДАНИЕ СЕРВИСА.

Создайте файл service.yaml со следующим содержимым:

```
apiVersion: v1
kind: Service
metadata:
  name: rtl-dremio-connector-svc
spec:
  type: ClusterIP
  selector:
    app: rtl-dremio-connector
  ports:
    - name: "port-4000-to-4000"
      port: 4000
      targetPort: 4000
      protocol: TCP
```

НАСТРОЙКА ИНГРЕССА.

Создайте файл dremio-connector.yaml со следующим содержимым:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: rtl-dremio-connector-rewrite
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
    ingress.kubernetes.io/ssl-redirect: "true"
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/proxy-body-size: 1024m
spec:
  tls:
  - hosts:
      - rtl.corp.ru
    secretName: corp-ru-tls
  rules:
  - host: rtl.corp.ru
    http:
      paths:
      - path: /dremio-connector(/|$)(.*)
```

```
pathType: Prefix
backend:
  service:
    name: rtl-dremio-connector-svc
    port:
      number: 4000
```

ЗАВЕРШАЮЩИЕ ДЕЙСТВИЯ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ.

Передайте в Kubernetes API приведенные выше манифесты deployment.yaml, service.yaml, dremio-connector.yaml в соответствии с официальной документацией к Kubernetes.

Выполните GET-запрос к URL'у только что запущенного приложения, выполнив команду:

curl -k https://rtl.corp.ru/dremio-connector/api/v1

Нижеприведенный ответ говорит об успешном старте приложения:

{"timestamp":"xxxxxxxxxxxxx","status":404,"error":"Not Found","message":"No message available","path":"/api/v1"}

15.4. УСТАНОВКА ПРИЛОЖЕНИЯ COLLABORATE-SERVICE.

Установка приложения сводится к следующим шагам:

1. Создание развертывания.
2. Создание сервиса.
3. Настройка ингресса.

СОГЛАШЕНИЕ О ФОРМАТИРОВАНИИ И ИМЕНОВАНИИ.

- Имена объектов, переменных и их значения, параметры, URL, пути в файловой системе, команды и директивы выделяются разметкой "код в строку", например: https://someurl.com, variable = value
- Доменное имя, на котором располагаются приложения: rtl.corp.ru
- Маршрут, по которому располагается приложение: /socket.io/
- Имя файла с rsa-ключом: key.rsa
- Номер версии приложения: 555_master
- Во всех кластерах SSL/TLS сертификат на домен и поддомены (wildcard) хранится в secret'e corp-ru-tls
- Во всех кластерах ключ key.rsa соответствующий пространству имен в Keycloak хранится в secret'e goodt-key-rsa

СОЗДАНИЕ РАЗВЕРТЫВАНИЯ.

Создайте файл deployment.yaml со следующим содержимым:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rtl-collaborate-service
  labels:
    version: 555_master
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rtl-collaborate-service
  template:
    metadata:
      labels:
        app: rtl-collaborate-service
        version: 555_master
    spec:
      imagePullSecrets:
        - name: art-goodt-me
      containers:
      - name: rtl-collaborate-service
        image: art.goodt.me/rtl-collaborate-service/rtl-collaborate-service:555_master
        resources:
          requests:
            memory: "200Mi"
            cpu: "200m"
          limits:
            memory: "512Mi"
            cpu: "1000m"
        ports:
        - containerPort: 4002
          protocol: TCP
        volumeMounts:
        - mountPath: /abc/app/goodt/collaborate-service/pm2start.sh
          name: goodt-pm2start-cm
          subPath: pm2start.sh
      volumes:
      - configMap:
          name: goodt-pm2start
          defaultMode: 511
        name: goodt-pm2start-cm
```

СОЗДАНИЕ КОНФИГМАПА.

Создайте файл goodt-pm2start.yaml

```
apiVersion: v1
data:
  pm2start.sh: |
   #!/bin/bash
   export PATH=$PATH:~/.nvm/versions/node/v16.17.1/bin
   pm2 start npm --name "collab" -- run start
   pm2 list
   pm2 logs
kind: ConfigMap
metadata:
  name: goodt-pm2start
```

СОЗДАНИЕ СЕРВИСА.

Создайте файл service.yaml со следующим содержимым:

```
apiVersion: v1
kind: Service
metadata:
  name: rtl-collaborate-service-svc
spec:
  type: ClusterIP
  selector:
    app: rtl-collaborate-service
  ports:
    - name: "port-4002-to-4002"
      port: 4002
      targetPort: 4002
      protocol: TCP
```

НАСТРОЙКА ИНГРЕССА.

Создайте файл collaborate-service.yaml со следующим содержимым:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: rtl-collaborate-service-rewrite
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
    ingress.kubernetes.io/ssl-redirect: "true"
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/proxy-body-size: 1024m
spec:
```

```
  tls:
  - hosts:
      - rtl.corp.ru
    secretName: corp-ru-tls
  rules:
  - host: rtl.corp.ru
    http:
      paths:
      - path: /socket.io(/|$)(.*)
        pathType: Prefix
        backend:
          service:
            name: rtl-collaborate-service-svc
            port:
              number: 4002
```

ЗАВЕРШАЮЩИЕ ДЕЙСТВИЯ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ.

Передайте в Kubernetes API приведенные выше манифесты deployment.yaml, goodt-pm2start.yaml, service.yaml, collaborate-service.yaml в соответствии с официальной документацией к Kubernetes.

Выполните GET-запрос к URL'у только что запущенного приложения, выполнив команду:

curl -k https://rtl.corp.ru/socket.io/

Нижеприведенный ответ говорит об успешном старте приложения:

{"code":0,"message":"Transport unknown"}

16. РАЗВЕРТЫВАНИЕ INSIGHT LOW CODE ИЗ DOCKER-ОБРАЗОВ. + COLLABORATE

Раздел посвящен базовому развертыванию программных компонентов Insight low code.

Для успешного выполнения развертывания необходима учетная запись в репозитории Docker-образов Goodt

В базовый Insight low code входят следующие программные компоненты:

1. Приложение Файловый загрузчик (rtl-fileupload);
2. Приложение Отчеты (rtl-report);
3. Приложение dremio-connector;
4. Приложение collaborate-service;

16.1. УСТАНОВКА ПРИЛОЖЕНИЯ ФАЙЛОВЫЙ ЗАГРУЗЧИК (RTL-FILEUPLOAD)

Установка приложения сводится к следующим шагам:

1. Создание базы данных.
2. Запуск приложения.
3. Настройка Nginx.
4. Проверка работоспособности.

СОГЛАШЕНИЕ О ФОРМАТИРОВАНИИ И ИМЕНОВАНИИ

- Имена объектов, переменных и их значения, параметры, URL, пути в файловой системе, команды и директивы выделяются разметкой "код в строку", например: https://someurl.com, variable = value
- Адрес и порт, на котором располагается приложение: rtl.corp.ru:8499
- Адрес и порт СУБД: db.corp.ru:5432
- Адрес Keycloak: auth.corp.ru
- Имя базы данных для приложения: fileupload
- Имя пользователя для подключения к базе данных: fileupload_user
- Пароль пользователя fileupload_user: $7r0n6pP@s2WrD
- Строка jdbc-подключения к базе данных: jdbc:postgresql://db.corp.ru:5432/fileupload
- Пространство имен в Keycloak: insight
- Имя пользователя в Keycloak: api-service
- Пароль пользователя api-service в Keycloak: k3ycl0akU$3r
- Идентификатор клиента в Keycloak: api-service-agent
- UUID клиента api-service-agent в Keycloak: 00000000-0000-0000-0000-000000000000
- Адрес статики: https://rtl.corp.ru/static
- URL хранилища MINIO: https://minio.corp.ru:9002
- Имя BUCKET хранилища MINIO: rtl_corp_bucket
- Имя пользователя хранилища MINIO: minio_user
- Пароль пользователя minio_user в хранилище MINIO: MiMiMiPas7@
- URL хранилища WebDAV: https://rtl.corp.ru/webdav/
- Имя пользователя хранилища WebDAV: webdav_user
- Пароль пользователя webdav_user в хранилище WebDAV: PaPaPas7@
- Номер версии приложения: 555_master
- Файл SSL-сертификата расположен по пути: /abc/conf/cert/fullchain.crt
- Файл приватного ключа к SSL-сертификату расположен по пути: /abc/conf/cert/private.key
- Файлы конфигурации Nginx расположены по пути: /abc/conf/nginx

СОЗДАНИЕ БАЗЫ ДАННЫХ.

В консоли psql выполните команды:

```
CREATE DATABASE fileupload;
CREATE USER fileupload_user WITH ENCRYPTED PASSWORD '$7r0n6pP@s2WrD';
GRANT ALL PRIVILEGES ON DATABASE fileupload TO fileupload_user;
\c fileupload;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO fileupload_user;
```

ЗАПУСК ПРИЛОЖЕНИЯ.

Добавьте репозиторий docker-образов Goodt:

```
docker login https://art.goodt.me
```

Выполните команду:

```
docker run -d --name=fileupload \
                -p 8099:8099 \
                -e "RTL_DB_USER=fileupload_user" \
                -e "RTL_DB_PASS=$7r0n6pP@s2WrD" \
                -e "RTL_DB_URL=jdbc:postgresql://db.corp.ru:5432/fileupload" \
                -e "RTL_PROFILE=prod" \
                -e "RTL_KCLK_URL=https://auth.corp.ru" \
                -e "RTL_KCLK_REALM=insight" \
                -e "RTL_KCLK_USER=api-service" \
                -e "RTL_KCLK_PASS=k3ycl0akU$3r" \
                -e "RTL_KCLK_CLID=api-service-agent" \
                -e "RTL_KCLK_CLSECRET=00000000-0000-0000-0000-000000000000" \
                -e "RTL_URN=https://rtl.corp.ru/static" \
                -e "RTL_USE_MINIO=false" \
                -e "RTL_USE_FILESYSTEM=true" \
                -e "RTL_USE_WEBDAV=false" \
                -e "RTL_MINIO_URL=https://minio.corp.ru:9002" \
                -e "RTL_MINIO_BUCKET=rtl_corp_bucket" \
                -e "RTL_MINIO_USER=minio_user" \
                -e "RTL_MINIO_PASS=MiMiMiPas7@" \
                -e "RTL_WEBDAV_URL=https://rtl.corp.ru/webdav/" \
                -e "RTL_WEBDAV_USER=webdav_user" \
                -e "RTL_WEBDAV_PASS=PaPaPas7@" \
                -e "RTL_KCLK_AUTH_ENABLED=false" \
                -v /abc/conf/cert/key.rsa:/abc/app/goodt/fileupload/key.rsa \
                art.goodt.me/rtl-fileupload/rtl-fileupload:555_master
```

Переменные: RTL_USE_MINIO, RTL_USE_FILESYSTEM, RTL_USE_WEBDAV используются для определения используемого хранилища. Только одна из них может иметь значение "true", остальные должны иметь значение "false".

НАСТРОЙКА NGINX.

Создайте файл fileupload.conf в папке файлов конфигурации Nginx:

```
server {
  listen 8499 ssl http2;
  listen [::]:8499 ssl http2;
  server_name rtl.corp.ru;

  ssl_certificate /abc/conf/cert/fullchain.crt;
  ssl_certificate_key /abc/conf/cert/private.key;
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
  ssl_ecdh_curve secp384r1;
  ssl_session_cache shared:SSL:10m;
  ssl_session_tickets off;
  ssl_stapling on;
  ssl_stapling_verify on;
  resolver 8.8.8.8 8.8.4.4 valid=300s;
  resolver_timeout 5s;
  add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload";
  add_header X-Content-Type-Options nosniff;
  client_max_body_size 1000M;
  proxy_connect_timeout 600;
  proxy_send_timeout 600;
  proxy_read_timeout 600;
  send_timeout 600;

  access_log /abc/logs/nginx/fileupload-access.log;
  error_log /abc/logs/nginx/fileupload-error.log;

  location / {
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header X-Real_IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;
    proxy_set_header X-Forwarded-Proto $scheme;
    real_ip_header X-Real-IP;
    proxy_pass http://localhost:8099;
    proxy_redirect off;
  }
}
```

Проверьте работоспособность Nginx после изменения конфигурации:

nginx -t

Вывод, приведенный ниже, говорит о корректной конфигурации:

nginx: the configuration file /abc/conf/nginx/nginx.conf syntax is ok
nginx: configuration file /abc/conf/nginx/nginx.conf test is successful

Перезапустите Nginx:

nginx -s reload

ПРОВЕРКА РАБОТОСПОСОБНОСТИ.

Выполните GET-запрос к URL'у только что запущенного приложения, выполнив команду:

curl -k https://rtl.corp.ru:8499/api/v1

Нижеприведенный ответ говорит об успешном старте приложения:

Authentication Failed

16.2. УСТАНОВКА ПРИЛОЖЕНИЯ ОТЧЕТЫ (RTL-REPORT)

Установка приложения сводится к следующим шагам:

1. Создание базы данных.
2. Создание развертывания.
3. Создание сервиса.
4. Настройка ингресса.

СОГЛАШЕНИЕ О ФОРМАТИРОВАНИИ И ИМЕНОВАНИИ

- Имена объектов, переменных и их значения, параметры, URL, пути в файловой системе, команды и директивы выделяются разметкой "код в строку", например: https://someurl.com, variable = value
- Адрес и порт, на котором располагается приложение: rtl.corp.ru:8489
- Адрес и порт СУБД: db.corp.ru:5432
- Адрес Keycloak: auth.corp.ru
- Имя базы данных для приложения: report
- Имя пользователя для подключения к базе данных: report_user
- Пароль пользователя report_user: $7r0n6pP@s2WrD
- Строка jdbc-подключения к базе данных: jdbc:postgresql://db.corp.ru:5432/report
- Пространство имен в Keycloak: insight
- Имя пользователя в Keycloak: api-service

- Пароль пользователя api-service в Keycloak: k3ycl0akU$3r
- Идентификатор клиента в Keycloak: api-service-agent
- UUID клиента api-service-agent в Keycloak: 00000000-0000-0000-0000-000000000000
- Номер версии приложения: 555_master
- URL приложения Оргструктура: https://rtl.corp.ru:8480
- URL приложения dremio-connector: https://rtl.corp.ru:4400
- Файл SSL-сертификата расположен по пути: /abc/conf/cert/fullchain.crt
- Файл приватного ключа к SSL-сертификату расположен по пути: /abc/conf/cert/private.key
- Файлы конфигурации Nginx расположены по пути: /abc/conf/nginx

СОЗДАНИЕ БАЗЫ ДАННЫХ.

В консоли psql выполните команды:

```
CREATE DATABASE report;
CREATE USER report_user WITH ENCRYPTED PASSWORD '$7r0n6pP@s2WrD';
GRANT ALL PRIVILEGES ON DATABASE report TO report_user;
\c report;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO report_user;
```

ЗАПУСК ПРИЛОЖЕНИЯ.

Добавьте репозиторий docker-образов Goodt:

```
docker login https://art.goodt.me
```

Выполните команду:

```
docker run -d --name=report\
                    -p 8089:8089 \
                    -e "RTL_DB_USER=report_user" \
                    -e "RTL_DB_PASS=$7r0n6pP@s2WrD" \
                    -e "RTL_DB_URL=jdbc:postgresql://db.corp.ru:5432/report" \
                    -e "RTL_PROFILE=prod" \
                    -e "RTL_KCLK_URL=https://auth.corp.ru" \
                    -e "RTL_KCLK_REALM=insight" \
                    -e "RTL_KCLK_USER=api-service" \
                    -e "RTL_KCLK_PASS=k3ycl0akU$3r" \
                    -e "RTL_KCLK_CLID=api-service-agent" \
                    -e "RTL_KCLK_CLSECRET=00000000-0000-0000-0000-000000000000" \
      -e "RTL_BACKEND_ORGSTRUCTURE_URL=https://rtl.corp.ru:8480" \
      -e "RTL_DREMIOCONNECTOR_URL=https://rtl.corp.ru:4400" \
                    -e "RTL_API_TEMPLATE_CREATE=false" \
                    art.goodt.me/rtl-report/rtl-report:555_master
```

НАСТРОЙКА NGINX.

Создайте файл learning.conf в папке файлов конфигурации Nginx:

```
server {
  listen 8489 ssl http2;
  listen [::]:8489 ssl http2;
  server_name rtl.corp.ru;

  ssl_certificate /abc/conf/cert/fullchain.crt;
  ssl_certificate_key /abc/conf/cert/private.key;
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
  ssl_ecdh_curve secp384r1;
  ssl_session_cache shared:SSL:10m;
  ssl_session_tickets off;
  ssl_stapling on;
  ssl_stapling_verify on;
  resolver 8.8.8.8 8.8.4.4 valid=300s;
  resolver_timeout 5s;
  add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload";
  add_header X-Content-Type-Options nosniff;
  client_max_body_size 1000M;
  proxy_connect_timeout 600;
  proxy_send_timeout 600;
  proxy_read_timeout 600;
  send_timeout 600;

  access_log /abc/logs/nginx/learning-access.log;
  error_log /abc/logs/nginx/learning-error.log;

  location / {
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header X-Real_IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;
    proxy_set_header X-Forwarded-Proto $scheme;
    real_ip_header X-Real-IP;
    proxy_pass http://localhost:8089;
    proxy_redirect off;
  }
}
```

Проверьте работоспособность Nginx после изменения конфигурации:

nginx -t

Вывод, приведенный ниже, говорит о корректной конфигурации:

nginx: the configuration file /abc/conf/nginx/nginx.conf syntax is ok
nginx: configuration file /abc/conf/nginx/nginx.conf test is successful

Перезапустите Nginx:

nginx -s reload

ПРОВЕРКА РАБОТОСПОСОБНОСТИ.

Выполните GET-запрос к URL'у только что запущенного приложения, выполнив команду:

curl -k https://rtl.corp.ru:8489/api/v1

Нижеприведенный ответ говорит об успешном старте приложения:

Authentication Failed

16.3. УСТАНОВКА ПРИЛОЖЕНИЯ DREMIO-CONNECTOR.

Установка приложения сводится к следующим шагам:

1. Запуск приложения.
2. Настройка ингресса.

СОГЛАШЕНИЕ О ФОРМАТИРОВАНИИ И ИМЕНОВАНИИ

- Имена объектов, переменных и их значения, параметры, URL, пути в файловой системе, команды и директивы выделяются разметкой "код в строку", например: https://someurl.com,variable = value
- Адрес и порт, на котором располагается приложение: rtl.corp.ru:4400
- Имя пользователя для подключения к Dremio: dc_user
- Пароль пользователя dc_user: $7r0n6pP@s2WrD
- FQDN узла, на котором установлен Dremio: agg.corp.ru
- Порт, на котором Dremio принимает jdbc-соединения: 31010
- Порт, на котором приложение dremio-connector принимает входящие запросы: 4000
- Имя файла с rsa-ключом: **key.rsa**
- Номер версии приложения: 555_master
- Файл SSL-сертификата расположен по пути: /abc/conf/cert/fullchain.crt

- Файл приватного ключа к SSL-сертификату расположен по пути: /abc/conf/cert/private.key
- Файлы конфигурации Nginx расположены по пути:/abc/conf/nginx

ЗАПУСК ПРИЛОЖЕНИЯ.

Добавьте репозиторий docker-образов Goodt:

docker login https://art.goodt.me

Выполните команду:

```
docker run -d --name=dremio-connector \
                        -p 4000:4000 \
                        -e "RTL_DREMIO_USER=dc_user" \
                        -e "RTL_DREMIO_PASS=$7r0n6pP@s2WrD" \
                        -e "RTL_DREMIO_HTTP_URL=http://agg.corp.ru:9047" \
                        -e "RTL_DREMIO_JDBC_URL=agg.corp.ru:31010" \
                        -e "RTL_APP_PORT=4000" \
                        -e "RTL_KEY_RSA=key.rsa" \
                        -e "RTL_DREMIO_AUTH=false" \
      art.goodt.me/rtl-dremio-connector/rtl-dremio-connector:555_master
```

НАСТРОЙКА NGINX.

Создайте файл dremio-connector.conf в папке файлов конфигурации Nginx:

```
server {
  listen 4400 ssl http2;
  listen [::]:4400 ssl http2;
  server_name rtl.corp.ru;

  ssl_certificate /abc/conf/cert/fullchain.crt;
  ssl_certificate_key /abc/conf/cert/private.key;
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
  ssl_ecdh_curve secp384r1;
  ssl_session_cache shared:SSL:10m;
  ssl_session_tickets off;
  resolver 8.8.8.8 8.8.4.4 valid=300s;
  resolver_timeout 5s;
  add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload";
  add_header X-Content-Type-Options nosniff;
  client_max_body_size 1000M;
  proxy_read_timeout 60m;
  proxy_send_timeout 60m;
```

```
  access_log /abc/logs/nginx/dremio-connector-access.log;
  error_log /abc/logs/nginx/dremio-connector-error.log;

  location / {
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header X-Real_IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;
    proxy_set_header X-Forwarded-Proto $scheme;
    real_ip_header X-Real-IP;
    proxy_pass http://localhost:4000;
    proxy_redirect off;
  }
}
```

Проверьте работоспособность Nginx после изменения конфигурации:

```
nginx -t
```

Вывод, приведенный ниже, говорит о корректной конфигурации:

```
nginx: the configuration file /abc/conf/nginx/nginx.conf syntax is ok
nginx: configuration file /abc/conf/nginx/nginx.conf test is successful
```

Перезапустите Nginx:

```
nginx -s reload
```

ПРОВЕРКА РАБОТОСПОСОБНОСТИ.

Выполните GET-запрос к URL'у только что запущенного приложения, выполнив команду:

```
curl -k https://rtl.corp.ru:4400/api/v1
```

Нижеприведенный ответ говорит об успешном старте приложения:

```
Authentication Failed
```

16.4. УСТАНОВКА ПРИЛОЖЕНИЯ COLLABORATE-SERVICE.

Установка приложения сводится к следующим шагам:

1. Запуск приложения.
2. Настройка ингресса.

СОГЛАШЕНИЕ О ФОРМАТИРОВАНИИ И ИМЕНОВАНИИ

- Имена объектов, переменных и их значения, параметры, URL, пути в файловой системе, команды и директивы выделяются разметкой "код в строку", например: https://someurl.com,variable = value
- Адрес и порт, на котором располагается приложение: rtl.corp.ru:4402
- FQDN узла, на котором установлен Dremio: agg.corp.ru
- Имя файла с rsa-ключом: **key.rsa**
- Номер версии приложения: 555_master
- Файл SSL-сертификата расположен по пути: /abc/conf/cert/fullchain.crt
- Файл приватного ключа к SSL-сертификату расположен по пути: /abc/conf/cert/private.key
- Файлы конфигурации Nginx расположены по пути:/abc/conf/nginx

ЗАПУСК ПРИЛОЖЕНИЯ.

Добавьте репозиторий docker-образов Goodt:

docker login https://art.goodt.me

Выполните команду:

```
docker run -itd --name=rtl-collaborate-service \
                        -p 4002:4002 \
                        --entrypoint /bin/bash \
       art.goodt.me/rtl-collaborate-service/rtl-collaborate-service:555_master
```

Выполните вход в созданный контейнер:

docker exec -it rtl-collaborate-service bash

Выполните команду:

pm2 start npm --name "collab" -- run start

НАСТРОЙКА NGINX.

Создайте файл collaborate-service.conf в папке файлов конфигурации Nginx:

```
upstream websocket {
  server rtl.corp.ru:4002;
}

server {
  listen 4402 ssl http2;
```

```
  listen [::]:4402 ssl http2;
  server_name rtl.corp.ru;

  ssl_certificate /abc/conf/cert/fullchain.crt;
  ssl_certificate_key /abc/conf/cert/private.key;
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
  ssl_ecdh_curve secp384r1;
  ssl_session_cache shared:SSL:10m;
  ssl_session_tickets off;
  resolver 8.8.8.8 8.8.4.4 valid=300s;
  resolver_timeout 5s;
  add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload";
  add_header X-Content-Type-Options nosniff;
  client_max_body_size 1000M;
  proxy_read_timeout 60m;
  proxy_send_timeout 60m;

  access_log /abc/logs/nginx/dremio-connector-access.log;
  error_log /abc/logs/nginx/dremio-connector-error.log;

  location / {
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header X-Real_IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;
    proxy_set_header X-Forwarded-Proto $scheme;
    real_ip_header X-Real-IP;
    proxy_pass http://localhost:4002/socket.io/;
    proxy_redirect off;
  }
}
```

Проверьте работоспособность Nginx после изменения конфигурации:

```
nginx -t
```

Вывод, приведенный ниже, говорит о корректной конфигурации:

```
nginx: the configuration file /abc/conf/nginx/nginx.conf syntax is ok
nginx: configuration file /abc/conf/nginx/nginx.conf test is successful
```

Перезапустите Nginx:

```
nginx -s reload
```

ПРОВЕРКА РАБОТОСПОСОБНОСТИ.

Выполните GET-запрос к URL'у только что запущенного приложения, выполнив команду:

curl -k https://rtl.corp.ru:4402/socket.io/

Нижеприведенный ответ говорит об успешном старте приложения:

{"code":0,"message":"Transport unknown"}