
# РУКОВОДСТВО
# по эксплуатации
# Программного решения
# для обработки и хранения данных
# ITSumma Data Processing Platform

Москва

2023

# СОДЕРЖАНИЕ

# 1 ВВЕДЕНИЕ

## 1.1 Область применения

Программное решение для обработки и хранения данных ITS DPP (ITSumma Data Processing Platform, далее - программное решение, ПО) предназначено для работы с большими объемами потоков данных. Оно позволяет получать данные из различных внешних источников, обрабатывать их, хранить в структурированном и неструктурированном виде, а также раскрывать данные для внешних систем.

## 1.2 Краткое описание функциональных возможностей

Платформа ITS DPP представляет собой набор автономных модулей, имеющих свое назначение:

- ITS DPP.ETL: реализует полную цепочку ETL- преобразований в рамках потоковой обработки данных, а также предоставление инструментов сбора данных;
- ITS DPP.MPP DB: реализует хранение структурированных данных в кластере Greenplum, который обеспечивает отказоустойчивость и масштабируемость;
- ITS DPP.Analytics DB: реализует хранение структурированных данных с возможностью формировать витрины данных. При необходимости в состав модуля может быть включен бесплатный Add-On с системами аналитики Superset/Redash;
- ITS DPP.MW: реализует полную цепочку ETL- преобразований, включая пакетную обработку данных, а также средства сбора данных;
- ITS DPP.DataLake: реализует хранилище для больших объемов неструктурированных данных, в котором кластерная структура обеспечивает отказоустойчивость и масштабируемость.

## 1.3 Уровень подготовки пользователя

Пользователи платформы ITS DPP делятся на следующие категории:

- пользователи решающие задачи аналитики данных;
- пользователи решающие задачи построения пайплайнов обработки данных.

Минимальные требования к пользователям решающим задачи аналитики данных:

- работа с инструментами аналитики;
- владение SQL и python.

Минимальные требования к пользователям решающим задачи построения пайплайнов обработки данных:

- понимание ETL процессов;
- знание и опыт работы с Airflow/Spark.

## 1.4 Перечень эксплуатационной документации, с которыми необходимо ознакомиться пользователю

Перед началом работы с ПО пользователю необходимо ознакомиться с Руководством по эксплуатации.

# 2 НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ

## 2.1 Виды деятельности, функции, для автоматизации которых предназначено данное средство автоматизации

| | |
|---|---|
| ITS DPP.MW | • Сбор данных с различных источников<br>• Пакетная обработка данных<br>• Реализация ETL цепочек |
| ITS DPP.ETL | • Потоковая передача и обработка данных в режиме реального времени;<br>• Реализация ETL цепочек<br>• Организация корпоративной шины интерактивного обмена данными (data pipeline) между распределенными приложениями;<br>• Мониторинг и управление данными (рабочими процессами); |
| ITS DPP.MPP DB | • Обеспечение отказоустойчивого и безопасного хранения больших объемов структурированных данных;<br>• Обработка больших объёмов структурированных и слабоструктурированных данных в режиме реального времени (для систем предиктивной аналитики, озёр и хранилищ данных, организации регулярной отчетности).<br>• Исследование, нарезка и визуализация данных; |
| ITS DPP.DataLake | • Хранение больших объемов неструктурированных данных; |
| ITS DPP.Analytics DB | • Анализ данных;<br>• Хранение и сортировка данных; |

| | |
|---|---|
| | • Исследование, нарезка и визуализация данных;<br>• Генерация аналитических отчетов по большим данным в режиме реального времени. |

## 2.2 Условия, при соблюдении (выполнении, наступлении) которых обеспечивается применение средства автоматизации в соответствии с назначением

Установка модулей платформы ITS DPP возможна на следующих ресурсах:

- **площадки:**
  - bare-metal серверы;
  - виртуальные серверы;
  - площадки облачных провайдеров (Selectel, Яндекс.Облако, Cloud, VK Cloud);
  - закрытые контуры;
- **операционные системы:**
  - Ubuntu (рекомендуемая версия 20.4)
  - CentOS
  - Astra Linux
  - РЕД ОС.

Для запуска модуля ITS DPP.DSM на площадке должен быть предварительно установлен python 3.10.

Минимальная конфигурация ресурсов, необходимая для запуска каждого модуля:

| **Модуль** | **Количество vCPU** | **Объем RAM** |
|---|---|---|
| ITS DPP.MW (Managed Workflows) | 4 | 16Gb |
| ITS DPP.ETL | 4 | 16Gb |
| ITS DPP.MPP DB | 8 | 32Gb |
| ITS DPP.Analytics DB | 8 | 32Gb |
| ITS DPP.DataLake | 8 | 32Gb |

Входная информация для обеспечения применения модулей:

| **Модуль** | **Входная информация** |
|---|---|
| ITS DPP.MW (Managed Workflows) | источник данных для подключения к модулю |

| | |
|---|---|
| ITS DPP.ETL | источник данных для подключения к модулю |
| ITS DPP.MPP DB | данные для загрузки в хранилище |
| ITS DPP.Analytics DB | данные для загрузки в хранилище |
| ITS DPP.DataLake | данные для загрузки в хранилище |

Требование к подготовке специалистов:

| **Модуль** | **Входная информация** |
|---|---|
| ITS DPP.MW (Managed Workflows) | - опыт работы с ОС Linux<br>- опыт работы с Airflow |
| ITS DPP.ETL | - опыт работы с ОС Linux<br>- опыт работы с Apache Spark |
| ITS DPP.MPP DB | - опыт работы с ОС Linux<br>- владение SQL и опыт работы с Greenplum или Postgresql |
| ITS DPP.Analytics DB | - опыт работы с ОС Linux<br>- владение SQL и опыт работы с Clickhouse |
| ITS DPP.DataLake | - опыт работы с ОС Linux<br>- опыт работы с Apache Hadoop |

# 3 ПОДГОТОВКА К РАБОТЕ

## 3.1 Состав и содержание дистрибутивного носителя данных

Состав дистрибутивного носителя данных отличается и зависит от комплекта поставляемых модулей.

| **Модуль** | **Состав дистрибутивного носителя данных** |
|---|---|
| ITS DPP.MW (Managed Workflows) | - файл playbook.yml содержащий набор инструкций для установки модуля<br>- файл requirements.yml содержащий набор ansible ролей для установки модуля<br>- файлы содержащие примеры конфигураций для установки модуля<br>- инструкция по установке модуля<br>- инструкция по эксплуатации модуля |
| ITS DPP.ETL | - файл playbook.yml содержащий набор инструкций для установки модуля<br>- файл requirements.yml содержащий набор ansible ролей для установки модуля<br>- файлы содержащие примеры конфигураций для установки модуля<br>- инструкция по установке модуля<br>- инструкция по эксплуатации модуля |
| ITS DPP.MPP DB | - файл playbook.yml содержащий набор инструкций для установки модуля<br>- файл requirements.yml содержащий набор ansible ролей для установки модуля<br>- файлы содержащие примеры конфигураций для установки модуля<br>- инструкция по установке модуля<br>- инструкция по эксплуатации модуля |
| ITS DPP.Analytics DB | - файл playbook.yml содержащий набор инструкций для установки модуля<br>- файл requirements.yml содержащий набор ansible ролей для установки модуля<br>- файлы содержащие примеры конфигураций для установки модуля<br>- инструкция по установке модуля |

| | |
|---|---|
| | - инструкция по эксплуатации модуля |
| ITS DPP.DataLake | - файл playbook.yml содержащий набор инструкций для установки модуля<br>- файл requirements.yml содержащий набор ansible ролей для установки модуля<br>- файлы содержащие примеры конфигураций для установки модуля<br>- инструкция по установке модуля<br>- инструкция по эксплуатации модуля |

## 3.2 Подготовка платформы к работе

После установки всех выбранных модулей платформы на выбранную инфраструктуру, запуск компонентов происходит автоматически. При выключении/включении или перезагрузке элементов инфраструктуры работа компонентов платформы также восстанавливается автоматически и не требует дополнительного вмешательства. Подробнее об иных случаях в пункте Аварийные ситуации.

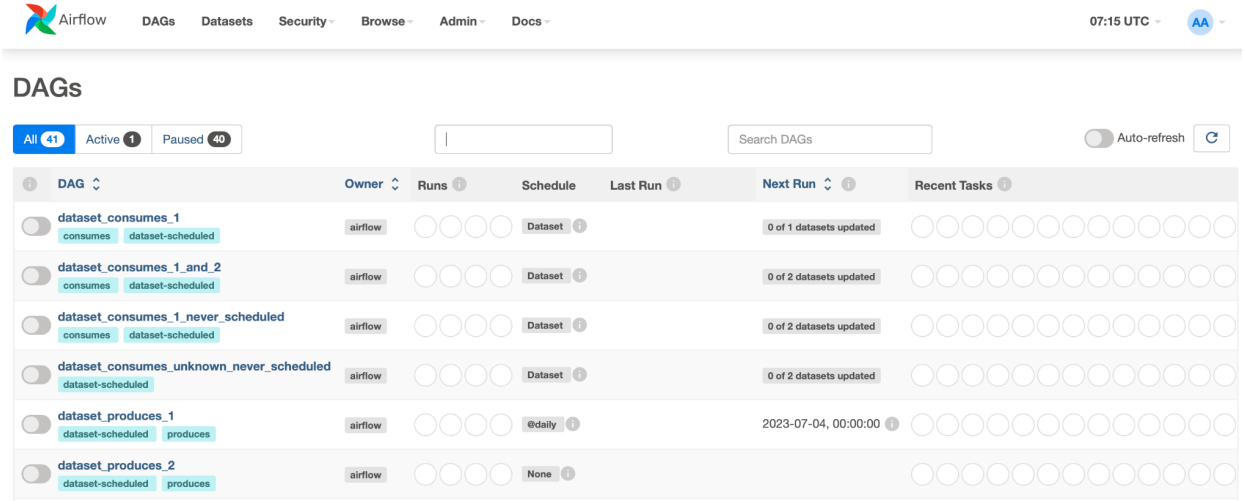
# 4 ОПИСАНИЕ ОПЕРАЦИЙ

## 4.1 Модуль MW

### 4.1.1 Описание интерфейса

Для подключения к модулю необходимо перейти по адресу http://IP:8080, где IP - ip адрес сервера, на котором установлен модуль.

Интерфейс модуля представлен интерфейсом Airflow. Данный интерфейс позволяет отслеживать и управлять сущностями типа DAG и задачами.




Расширенное описание интерфейса доступно в документации для Airflow: https://airflow.apache.org/docs/apache-airflow/stable/ui.html.

### 4.1.2 Работа с DAG файлами

Данная инструкция позволяет запустить DAG для обработки тестовых данных. Данный процесс симулирует процесс сбора и обработки курсов различных валют с сайта cbr-xml-daily.ru и включает в себя следующие этапы:

- сбор данных из источника;
- обработка данных в необходимом формате;
- загрузка данных в базу.

Больше примеров работы с DAG файлами можно найти в официальной документации Airflow https://airflow.apache.org/docs/apache-airflow/stable/tutorial/index.html.

**Шаг 1. Подготовка DAG файла**

Разберем работу с DAG файлами на примере тестового сценария, Для этого используем файл exchange_rate.py.

1. Добавьте файл с кодом в директорию с DAG файлами /opt/airflow/dags

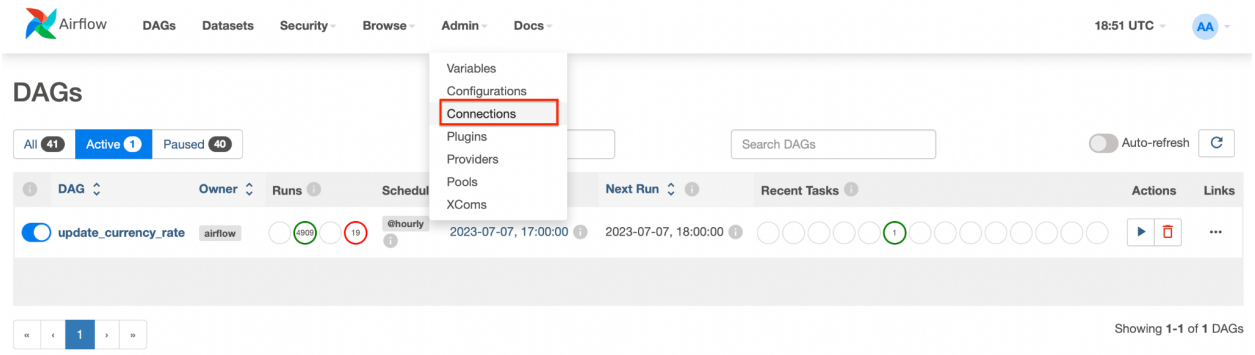
```
from airflow import DAG
from airflow_clickhouse_plugin.hooks.clickhouse_hook import
ClickHouseHook
from clickhouse_driver import Client
from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago
import requests
```

```
import pandas as pd
from datetime import datetime

def insert_df_to_clickhouse(df):
    client =
ClickHouseHook(clickhouse_conn_id='clickhouse_finance').get_conn()
    client.insert_dataframe(f'INSERT INTO currency_rate VALUES',
        df,
        settings={'use_numpy': True}
    )


with DAG(
    dag_id='update_currency_rate',
    start_date=days_ago(2),
    schedule_interval='@hourly'
) as dag:

    def update_currency_rate():
        data = []
        raw_json =
requests.get('https://www.cbr-xml-daily.ru/daily_json.js').json()
        for valute in raw_json['Valute']:
            raw_json['Valute'][valute]['__inserted_time'] =
datetime.now()
            data.append(raw_json['Valute'][valute])
        df = pd.DataFrame.from_records(data)
        insert_df_to_clickhouse(df)

    update_currency_rate = PythonOperator(
        task_id='update_currency_rate',
        python_callable=update_currency_rate,
        dag=dag
    )

    update_currency_rate
```

2. Загрузка обработанных данных выполняется в СУБД Clickhouse, компонент модуля Analytics DB, поэтому стоит убедиться, что данным модуль установлен и запущен.

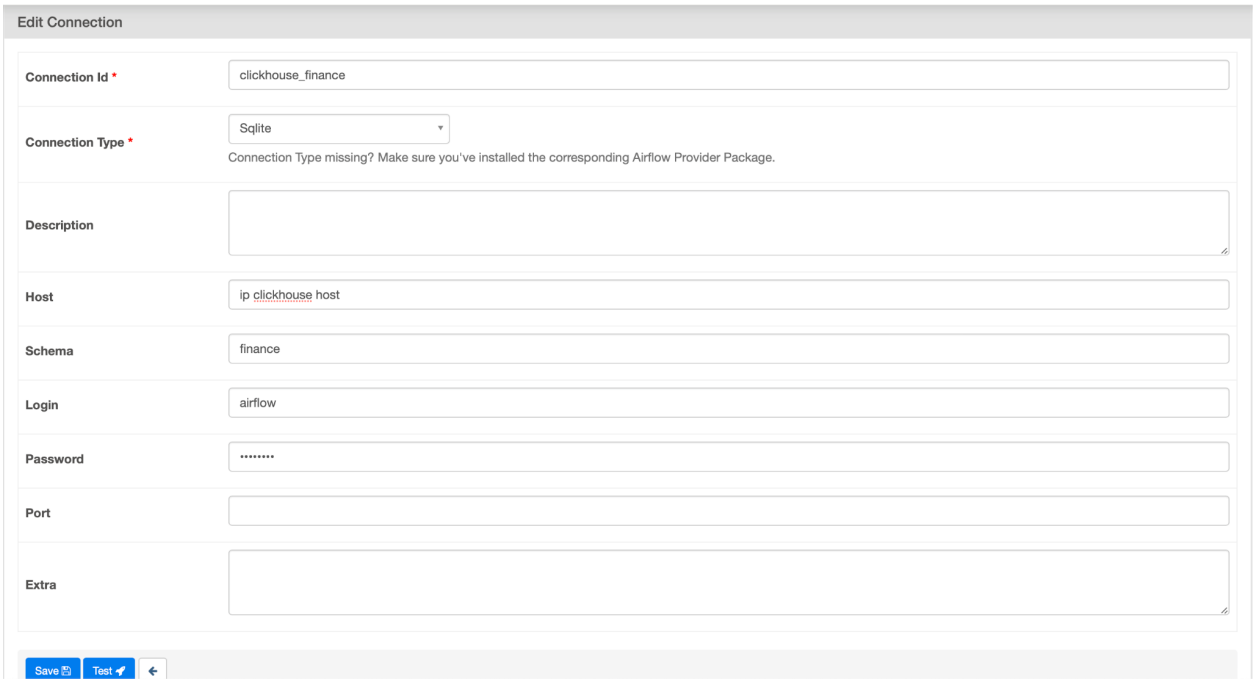
**Шаг 2. Добавление коннектора**

1. Переходим в меню добавления подключений



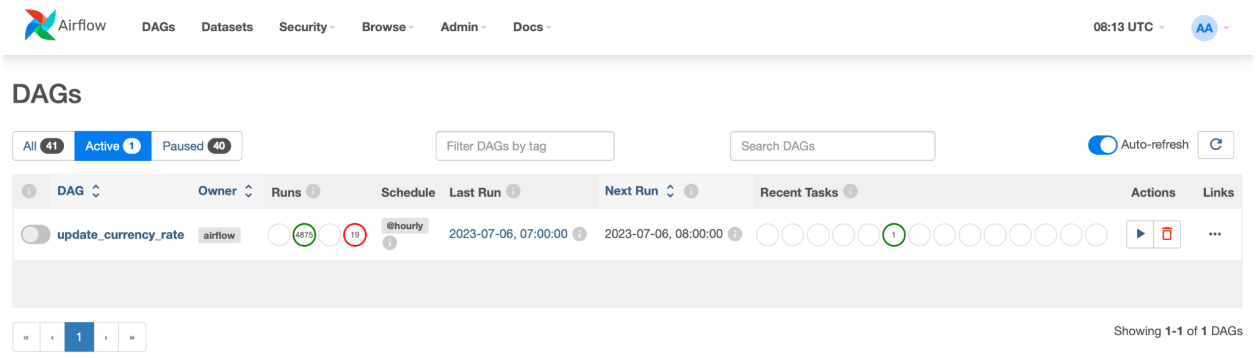

2. Нажимаем на кнопку добавления нового подключения


3. Вводим параметры нового подключения и сохраняем его:
    a. id подключения - `clickhouse_finance` как указано в DAG
    b. тип подключения - Sqlite
    c. host - ip сервера, на котором расположен Clickhouse
    d. schema - имя таблицы в Clickhouse, в которую будем загружать данные
    e. login/password - данные для авторизации в Clickhouse

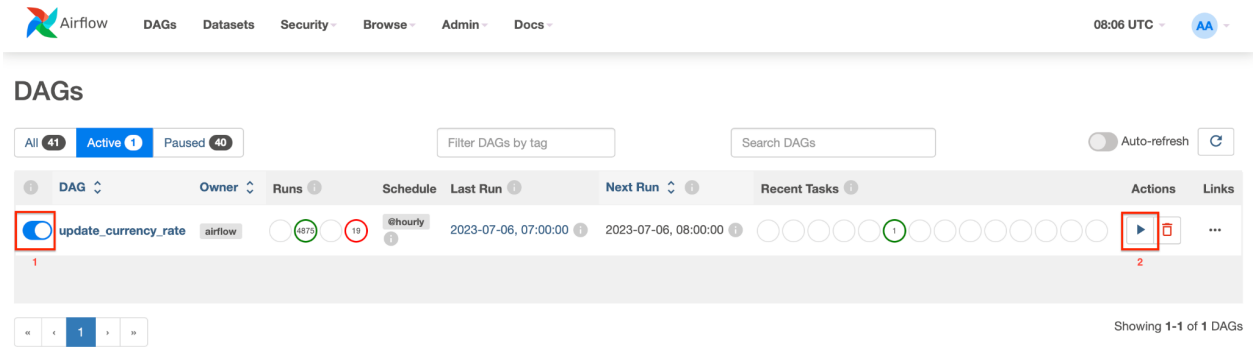



## Шаг 3. Запуск DAG файла

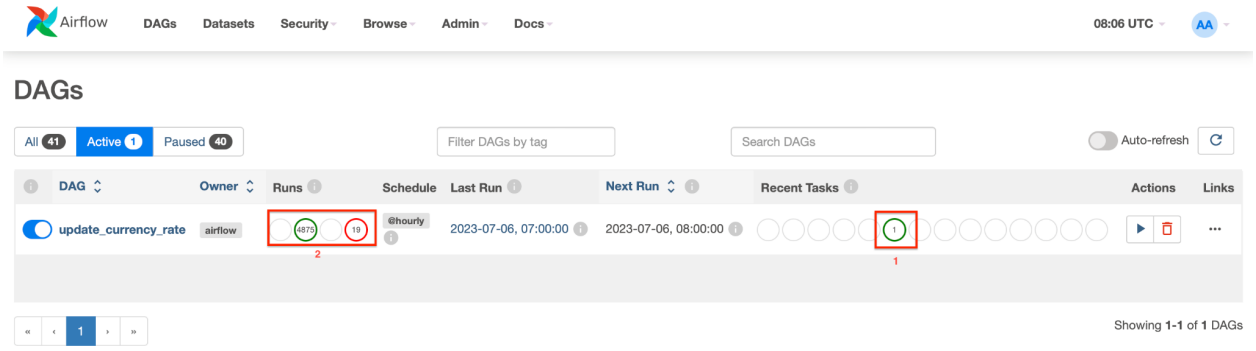
1. Откройте веб-интерфейс компонента Airflow, DAG файл должен отобразиться в списке.



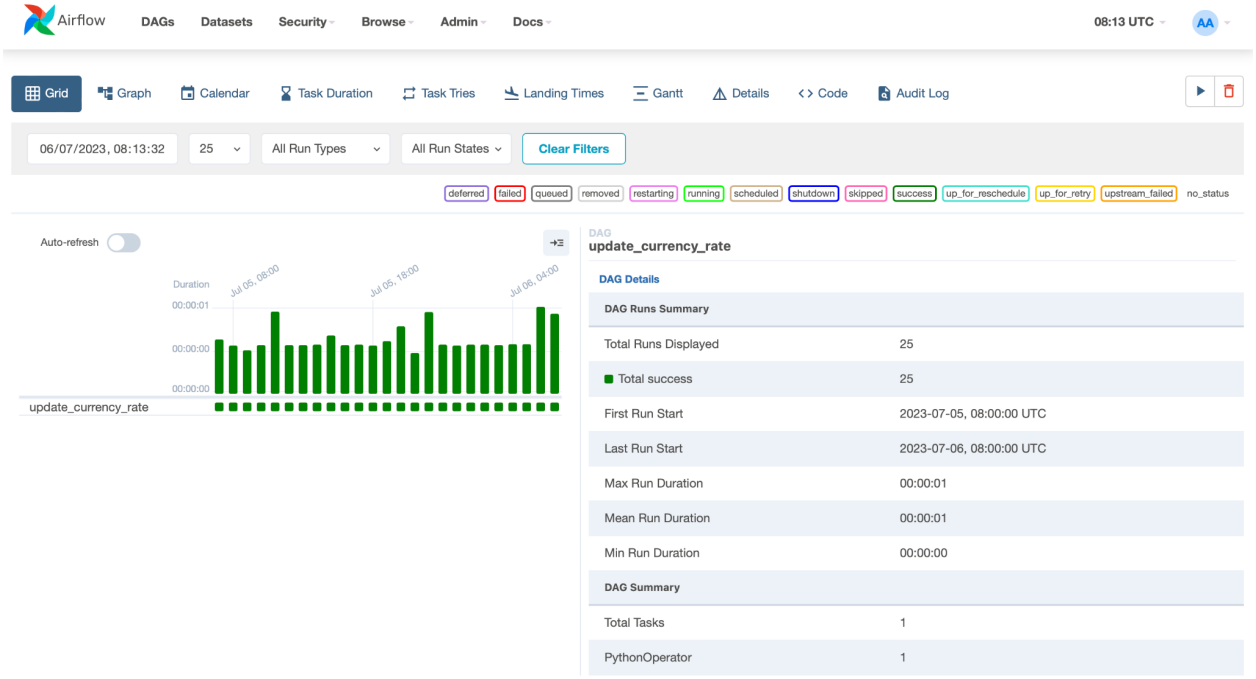

2. Переключите DAG в состояние Unpause (1) и нажмите на кнопку play (2).




3. После запуска DAG, в интерфейсе будет отображаться статус текущего запуска (1), а также статистика по предыдущим запускам (2).




4. Для получения более детальной информации необходимо нажать на DAG.



# 4.2 Модуль ETL

## 4.2.1 Подключение к Spark

Есть два способа подключения и загрузки задач в Apache Spark:

- spark-submit: позволяет запускать заранее написанные приложения на языках Scala, Python, или R.
- spark-shell/pyspark: предоставляет интерактивную среду для исполнения команд на языках Scala, Python, или R.

Spark-submit находится в директории /opt/spark/bin/. Если ваш код зависит от других проектов, вам необходимо упаковать зависимости вместе с вашим приложением, чтобы запустить приложение в кластере со Spark. Для этого создайте jar сборку, содержащую ваш код и его зависимости. Можно использовать плагины сборки sbt или Maven. После подготовки кода, используем скрипт bin/spark-submit для запуска проекта. Более подробно об использовании spark-submit в официальной документации https://spark.apache.org/docs/2.0.0/submitting-applications.html.
Для запуска spark-shell необходимо перейти в директорию Spark и выполнить
`./bin/spark-shell`

## 4.2.2 Подключение к Apache Kafka

Подключение к Apache Kafka происходит через командную строку из директории /opt/kafka-3.3.1/bin/. Вся работа с Apache Kafka производится посредством скриптов находящихся в директории /opt/kafka-3.3.1/bin/.

## 4.2.3 Отправка данных в Apache Kafka

**Шаг 1. Создание топика**

Для записи сообщений в Kafka нужно в первую очередь создать сущность topic. Это делается посредством скрипта *kafka-topics.sh.* Для создания topic введите следующую команду:

```
bin/kafka-topics.sh --create --topic new-topic
--bootstrap-server hostname:9092
```

где:

`new-topic` — имя topic;

`hostname` — имя хоста, на котором создается topic.

результат выполнения команды:

```
Created topic new-topic.
```

**Шаг 2. Запись в топик**

Для записи сообщений в используется скрипт *kafka-console-producer.sh.* Для записи сообщения необходимо выполнить следующую команду:

```
bin/kafka-console-producer.sh --topic new-topic
--bootstrap-server hostname:9092
```

где:

`new-topic` — имя topic;

`hostname` — имя хоста, на котором создается topic.

Данная команда открывает запись сообщений в topic. После записи всех необходимых сообщений нажмите Ctrl+C.

**Генерация сообщений скриптом для выполнения тестового сценария**

Для выполнения тестового сценария необходимо сгенерировать фиктивный поток сообщений в Kafka с помощью скрипта *json_faker_to_kafka.py*.

```
import json
import os
import sys
from faker import Faker
from confluent_kafka import Producer, KafkaException,
KafkaError
from confluent_kafka.admin import AdminClient, NewTopic
def create_json_file(data, filename):
    with open(filename, 'w') as f:
        json.dump(data, f, indent=4)
def create_topic_if_not_exist(bootstrap_servers, topic):
    admin_client = AdminClient({'bootstrap.servers':
bootstrap_servers})
    topic_metadata = admin_client.list_topics(timeout=5)
    if topic not in topic_metadata.topics:
        new_topic = NewTopic(topic, num_partitions=1,
replication_factor=1)
        admin_client.create_topics([new_topic])
        print(f"Topic '{topic}' created successfully.")
def write_to_kafka(num_files, bootstrap_servers, topic):
    faker = Faker()
    producer = Producer({"bootstrap.servers":
bootstrap_servers})
    for i in range(num_files):
        name = faker.name()
        age = faker.random_int(min=18, max=80)
        city = faker.city()
        data = {
            "name": name,
            "age": age,
            "city": city
```

```
        }
        # Convert data to JSON string
        json_data = json.dumps(data)
        # Send the JSON data to Kafka
        try:
            producer.produce(topic, key=str(i),
value=json_data)
        except KafkaException as e:
            print(f"Failed to produce message: {e}")
            continue
    producer.flush()
    print(f"{num_files} JSON messages sent to Kafka topic:
{topic}")
def main(num_files, bootstrap_servers, topic):
    create_topic_if_not_exist(bootstrap_servers, topic)
    write_to_kafka(num_files, bootstrap_servers, topic)
if __name__ == '__main__':
    if len(sys.argv) < 4:
        print("Usage: python json_faker.py <num_files>
<kafka_bootstrap_servers> <kafka_topic>")
        sys.exit(1)
    num_files = int(sys.argv[1])
    bootstrap_servers = sys.argv[2]
    topic = sys.argv[3]
    main(num_files, bootstrap_servers, topic)
```

Для запуска скрипта на сервере должен быть установлен python 3.*. Запускаем скрипт командой:
`python json_faker_to_kafka.py 1 dpp-etl-host:9092 transfered`
где `dpp-etl-host` - хостнейм или ip сервера, на котором развернут модуль ETL.

### 4.2.4 Запуск приложения в Spark

Разберем пример запуска готового приложения *stream_kafka_to_gp.py* в Spark через spark-submit. Приложение забирает данные из Kafka и передает их с Greenplum (модуль ITS DPP. MPP DB).

1. Для начала необходимо создать в Greenplum таблицу *transfered.*
2. Далее формируем список переменных для приложения в файле .env

```
GP_URL="jdbc:postgresql://dpp-mpp-db-host:5432/transfered"
- url для подключения к Greenplum
GP_USER="gpadmin" - пользователь в Greenplum
```

```
GP_PASS="" - пароль для пользователя в Greenplum
GP_TABLE="transfered" - имя таблицы в Greenplum
```

3. Запускаем приложение Spark

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, to_json, struct,
from_json
from pyspark.sql.types import StructType, StructField,
StringType, IntegerType
from dotenv import load_dotenv
import os
load_dotenv()
KAFKA_BOOTSTRAP_SERVERS =
os.getenv("KAFKA_BOOTSTRAP_SERVERS")
KAFKA_TOPIC = os.getenv("KAFKA_TOPIC")
GP_URL = os.getenv("GP_URL")
GP_USER = os.getenv("GP_USER")
GP_PASS = os.getenv("GP_PASS")
GP_TABLE = os.getenv("GP_TABLE")
# Define the JSON schema
jsonSchema = StructType([
    StructField("name", StringType(), True),
    StructField("age", IntegerType(), True),
    StructField("city", StringType(), True)
])
# Initialize Spark Session
spark = SparkSession.builder \
    .appName("StreamingApp") \
    .getOrCreate()
# Read messages from kafka
df = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers",
KAFKA_BOOTSTRAP_SERVERS) \
    .option("subscribe", KAFKA_TOPIC) \
    .load() \
    .selectExpr("CAST(value AS STRING)") \
# Write df to console
#query = df.writeStream \
```

```
#      .outputMode("append") \
#      .format("console") \
#      .start()
def write_to_gp(df, epoch_id):
    df = df.select(from_json(df.value,
jsonSchema).alias("data")).select("data.*")
    df.show(truncate=False)
    df.write \
        .format("jdbc") \
        .mode("append") \
        .option("driver", "org.postgresql.Driver") \
        .option("url", GP_URL) \
        .option("user", GP_USER) \
        .option("password", GP_PASS) \
        .option("dbtable", GP_TABLE) \
        .save()
query = df.writeStream \
    .foreachBatch(write_to_gp) \
    .outputMode("append") \
    .start()
query.awaitTermination()
```

Для запуска скрипта на сервере должен быть установлен python 3.*. Запускаем скрипт командой:

```
/opt/spark/bin/spark-submit --master
spark://dpp-etl-host:7077 --packages
org.apache.spark:spark-sql-kafka-0-10_2.12:3.3.0
--driver-class-path lib/postgresql-42.6.0.jar --jars
lib/postgresql-42.6.0.jar stream_kafka_to_gp.py
```

где `dpp-etl-host` - хостнейм или ip сервера, на котором развернут модуль ETL.

4. Теперь необходимо сгенерировать поток фиктивных сообщений в Kafka, как описано в предыдущем пункте.

## 4.3 Модуль MPP DB

### 4.3.1 Подключение к SQL консоли

Подключение к Greenplum осуществляется теми же способами, как оно выполняется для PostgreSQL. Например, через консольную утилиту psql.

Для подключения к Greenplum используем команды:

```
su - gpadmin
bash
```

```
psql postgres
```

### 4.3.2 Создание базы

Осуществляется с помощью команды в SQL консоли:

```
CREATE DATABASE name [ [WITH] [OWNER [=] dbowner]
          [TEMPLATE [=] template]
          [ENCODING [=] encoding]
          [TABLESPACE [=] tablespace]
          [CONNECTION LIMIT [=] connlimit ] ]
```

### 4.3.3 Создание внешней таблицы

Greenplum позволяет подключать таблицы из внешних источников для дальнейшего их использования в процессах преобразования данных.

Пример подключения к внешней clickhouse базе:

```
CREATE READABLE EXTERNAL TABLE
pxf_ch_usd_exchange_rates("date" DATE, "value" REAL)
LOCATION
('pxf://usd_exchange_rates?PROFILE=JDBC&JDBC_DRIVER=com.clic
khouse.jdbc.ClickHouseDriver&DB_URL=jdbc:clickhouse://192.16
8.122.15:8123/demo_db&USER=default') FORMAT 'CUSTOM'
(FORMATTER='pxfwritable_import');
```

```
test_db=# CREATE READABLE EXTERNAL TABLE pxf_ch_usd_exchange_rates("date" DATE, "value" REAL) LOCATION ('pxf://usd_exchange_rates?PROFILE=
JDBC&JDBC_DRIVER=com.clickhouse.jdbc.ClickHouseDriver&DB_URL=jdbc:clickhouse://192.168.122.15:8123/demo_db&USER=default') FORMAT 'CUSTOM'
(FORMATTER='pxfwritable_import');
CREATE EXTERNAL TABLE
test_db=# select * from pxf_ch_usd_exchange_rates limit 5;
    date    |  value
------------+---------
 2021-03-02 | 74.0448
 2021-03-03 | 74.5755
 2021-03-04 | 73.5187
 2021-03-05 | 73.7864
 2021-03-06 | 74.4275
(5 rows)
```

Более подробно об использовании Greenplum в официальной документации https://docs.vmware.com/en/VMware-Greenplum/6/greenplum-database/landing-index.html#differences-compared-to-open-source-greenplum-database.

## 4.4 Модуль DataLake

### 4.4.1 Перенос данных в HDFS

**Шаг 1. Подготовка директории в HDFS**

1. Для создания директорий в HDFS необдходимо работать из под пользователя hdfs, при это имея возможность писать в root директории, поэтому выполняем следующие команды:

```
sudo -s
su - hdfs
```

2. Проверка текущих директорий:
   `hdfs dfs -ls /`
3. Создаем необходимую директорию:
   `hdfs dfs -mkdir /example`
4. Формируем необходимые права доступа для директории:
   `hdfs dfs -chown user /example`
   `hdfs dfs -chmod 777 /example`

**Шаг 2. Копирование файлов в HDFS**

Для копирование файлов можно использовать команду `put` или `copyFromLocal`:

```
hdfs dfs -put <source_path> <destination_path>

hdfs dfs -copyFromLocal ~/file1.txt /hdfsDirectory
```

# 4.5 Модуль Analytics DB

## 4.5.1 Работа с данными в Clickhouse

**Создание таблиц**

В Clickhouse работают стандартный SQL команды, с одним дополнением, для таблиц в Clickhouse требуется добавить параметр ENGINE. По умолчанию лучше использовать значение MergeTree для параметра ENGINE.

Пример создания таблицы:

```
CREATE TABLE my_first_table
(
   user_id UInt32,
   message String,
   timestamp DateTime,
   metric Float32
)
ENGINE = MergeTree
PRIMARY KEY (user_id, timestamp)
```

**Внесение данных в таблицу**

Для внесения данных в таблицу можно использовать SQL команду INSERT, однако важно понимать, каждое применение команды INSERT в Clickhouse вызывает выделение папки в хранилище, чтобы свести в минимуму количество папок, лучше вставлять максимально возможное количество записей сразу.

```
INSERT INTO my_first_table (user_id, message, timestamp,
metric) VALUES
   (101, 'Hello, ClickHouse!',
now(),       -1.0    ),
   (102, 'Insert a lot of rows per batch',
yesterday(), 1.41421 ),
```

```
    (102, 'Sort your data based on your commonly-used
queries', today(),     2.718   ),
    (101, 'Granules are the smallest chunks of data read',
now() + 5,   3.14159 )
```

**Выполнение запросов в таблицах**

Для выполнения запросов в Clickhouse можно использовать SQL команду SELECT.

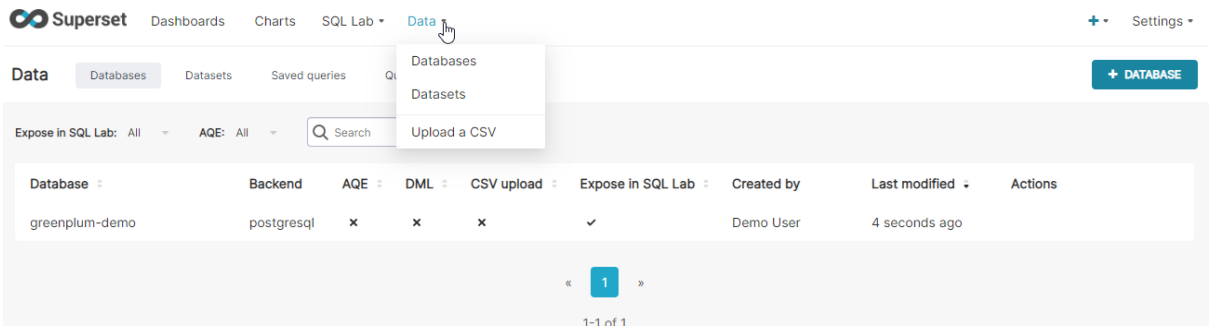
```
SELECT *
FROM my_first_table
ORDER BY timestamp
```

Подробнее о переносе данных из различных хранилищ в Clickhouse можно найти в официальной документации https://clickhouse.com/docs/en/getting-started/quick-start.

### 4.5.2 Формирование дашбордов в Superset

**Создание подключения к БД**

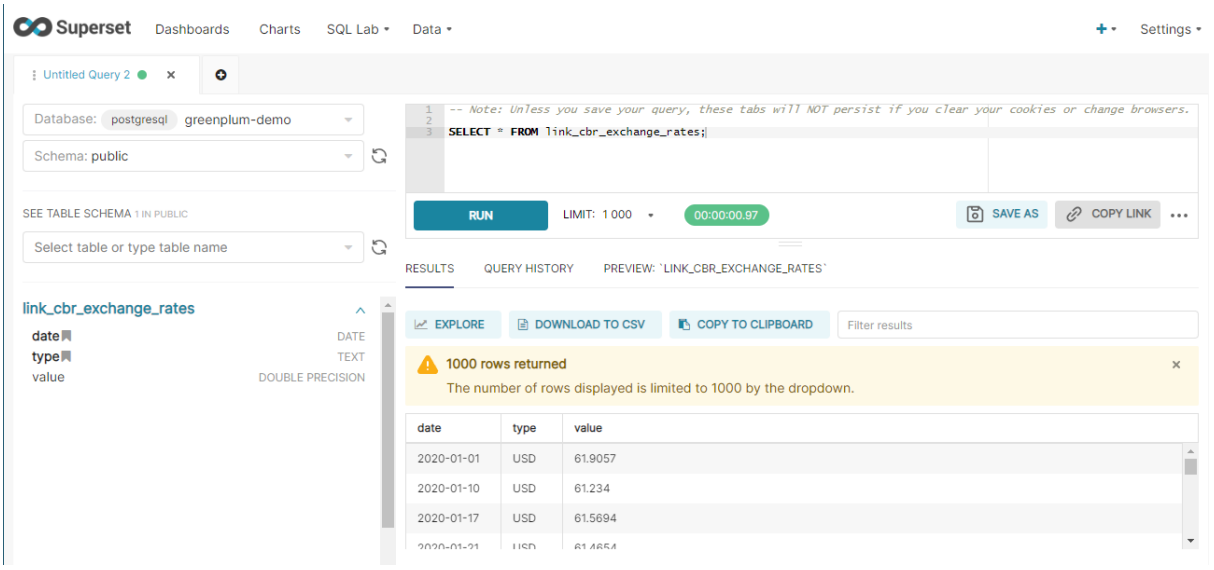
Перед тем как отображать данные необходимо подключить их источник. Это делается с помощью интерфейса Data → Databases




**Редактор запросов**

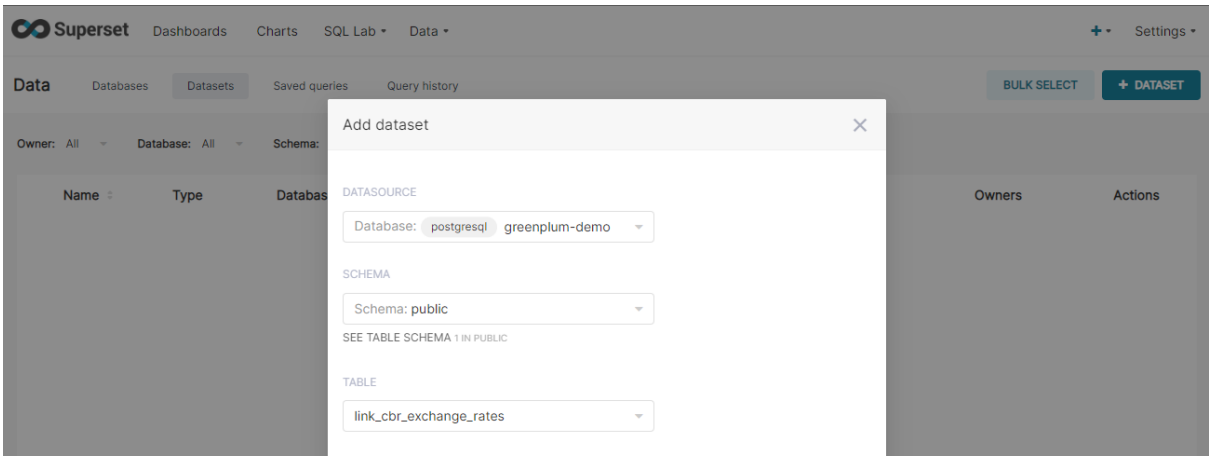
Для построения сложных представлений данных на основе нескольких таблиц в Superset-е есть инструмент “SQL Lab” → “SQL Editor”. Он позволяет максимально

интерактивно строить запросы данных и сохранять их для будущего использования.



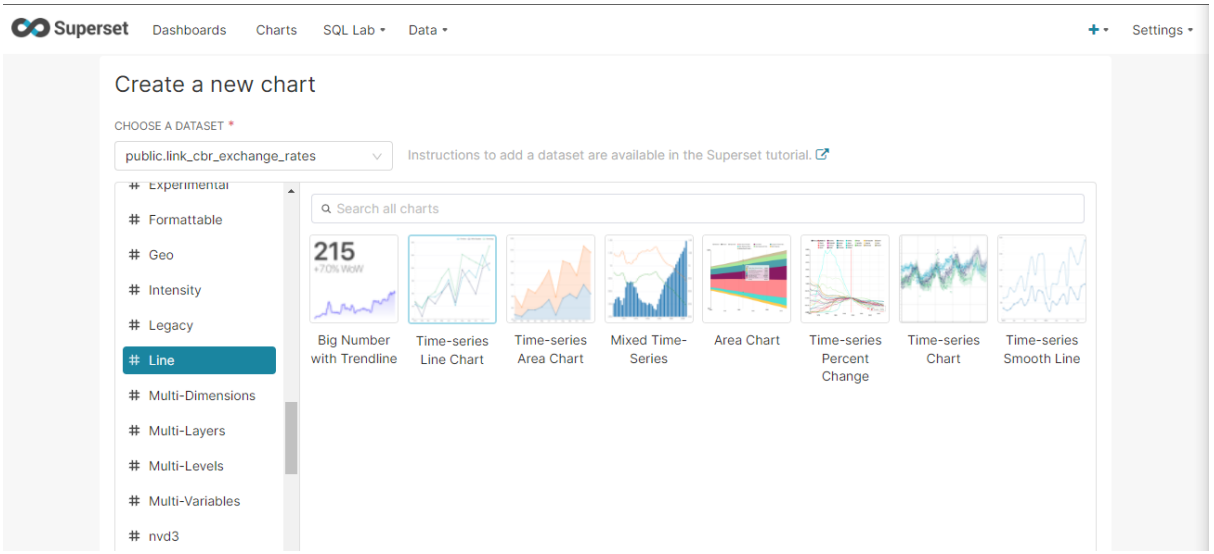

**Создания набора данных**

Перед тем как отобразить данные в superset-е необходимо сформировать Set из них, который будет использоваться в дальнейшем для формирования “разрезов” и графиков. Это делается с помощью интерфейса Data → Datasets.



**Создание Диаграммы**

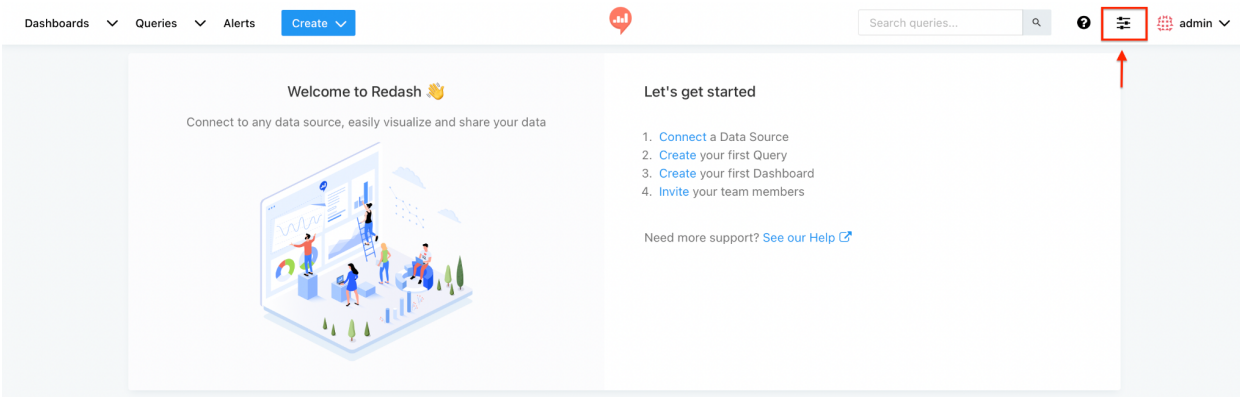
Для отображения данных на основе Dataset-ов используются Chart-ы.



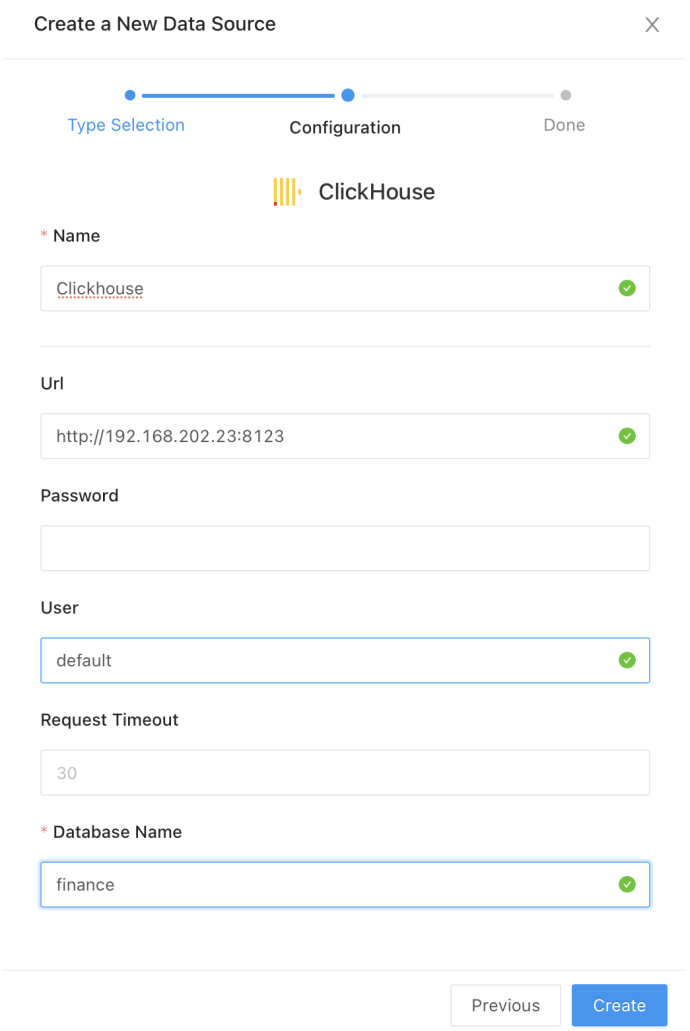

## 4.5.3 Формирование дашбордов Redash

**Создание подключения к БД**

1. Для подключения источника данных необходимо перейти в панель управления, нажав на указанную кнопку.

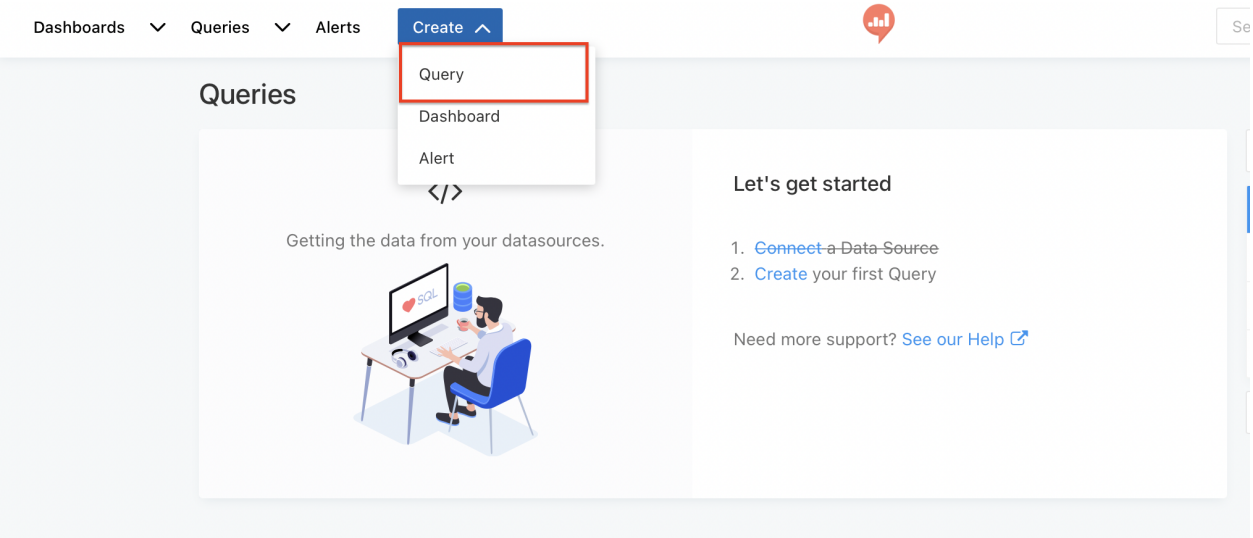


2. Создаем подключение к Clickhouse

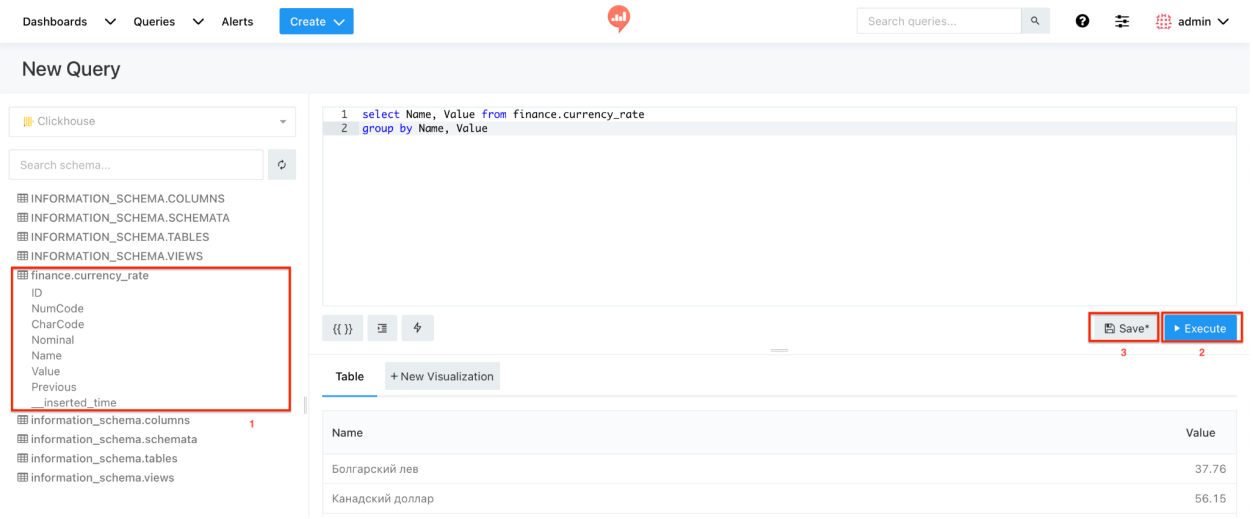



**Создание запросов**

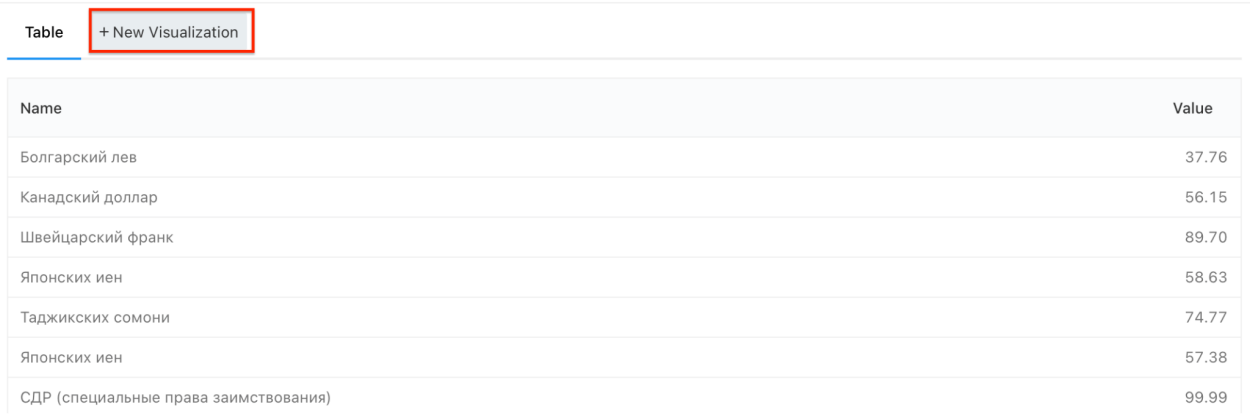
1. Нажимаем на кнопку Create и выбираем в открывшемся списке Query.




2. Выбираем таблицу (1), в которую мы ранее загрузили данные из Airflow, пишем запрос на вывод данных и жмем кнопку Execute (2), для сохранения запроса жмем кнопку Save (3).

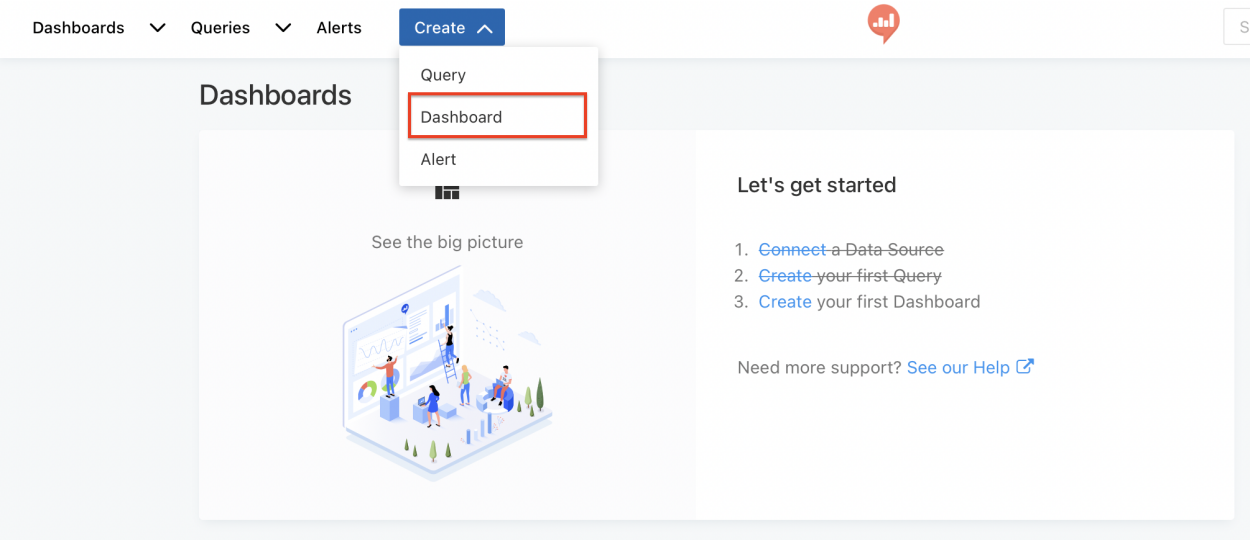



3. По умолчанию запрос визуализируется в виде таблицы, чтоб добавить новую визуализацию нужно нажать кнопку New Visualisation.

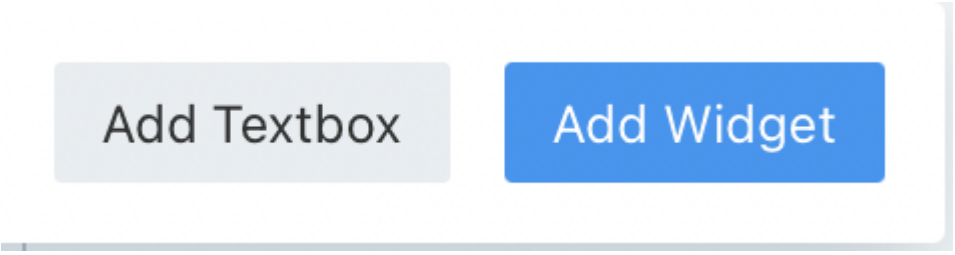



**Создание дашбордов**

1. Нажимаем на кнопку Create и выбираем в открывшемся списке Dashboard




2. Дашборд формируется при помощи кнопки Add Textbox, которая позволяет добавлять описания и кнопки Add Widget, которая позволяет добавлять заранее созданные визуализации.




Больше информации по работе с Redash можно найти в официальной документации по ссылке https://redash.io/help/user-guide/getting-started.

*Блок 4 заполняется для всех функций/задач/процедур/модулей ПО*

# 5 АВАРИЙНЫЕ СИТУАЦИИ

## 5.1 Действия в случае возникновении отказа в обслуживания компонентов платформы

### 5.1.1 Модуль DSM

При перезагрузке сервера с модулем, все компоненты модуля запускаются автоматически.

### 5.1.2 Модуль MW

При перезагрузке сервера с модулем, все компоненты модуля запускаются автоматически.

При отказе одного из серверов входящих в кластер балансировка нагрузки происходит автоматически.

### 5.1.3 Модуль ETL

При перезагрузке сервера с модулем, все компоненты модуля запускаются автоматически.

При отказе одного из серверов входящих в кластер балансировка нагрузки происходит автоматически.

### 5.1.4 Модуль MPP DB

При перезагрузке сервера с модулем, все компоненты модуля запускаются автоматически.

При отказе одного из серверов входящих в кластер балансировка нагрузки происходит автоматически.

### 5.1.5 Модуль Data Lake

При перезагрузке сервера с модулем, все компоненты модуля запускаются автоматически.

При отказе одного из серверов входящих в кластер балансировка нагрузки происходит автоматически.

### 5.1.6 Модуль Analytics DB

При перезагрузке сервера с модулем, все компоненты модуля запускаются автоматически.

При отказе одного из серверов входящих в кластер балансировка нагрузки происходит автоматически.

## 5.2 Действия в случае возникновении проблем при работе с компонентами платформы

### 5.2.1 Модуль MW

При возникновении проблем в работе Airflow перезагрузите сервис командой:

```
systemctl restart airflow-webserver
```

При возникновении проблем в загрузке и выполнении DAG процессов перезагрузите сервис командой:

```
systemctl restart airflow-scheduler
```

Для анализа проблем возникающих при работе с DAG файлами в Airflow используются логи, расположенные в директории /opt/airflow/logs/<*DAG_name*>.

### 5.2.2 Модуль ETL

Для анализа проблем возникающих при работе Apache Kafka используются логи, вызываемые по команде:

```
journalctl -u kafka
```

При возникновении проблем в работе Apache Kafka перезагрузите сервис командой:

```
systemctl restart kafka
```

Для анализа проблем возникающих при работе Apache Spark используются логи, вызываемые по команде:

```
journalctl -u spark-master
```

При возникновении проблем в работе Apache Spark перезагрузите сервис командой:

```
systemctl restart spark-master
```

### 5.2.3 Модуль MPP DB

Для анализа проблем возникающих при работе Greenplum используются логи, вызываемые по команде:

```
su - gpadmin
gpstate
```

При возникновении проблем в работе Greenplum перезагрузите сервис командой:

```
su - gpadmin
gpstop/gpstart/gprestart
gprestart –force
```

### 5.2.4 Модуль Data Lake

Для анализа проблем возникающих при работе Apache Hadoop используются логи, вызываемые по команде:

```
journalctl -u hadoop
```

При возникновении проблем в работе Apache Hadoop перезагрузите сервис командой: `systemctl restart hadoop`

### 5.2.5 Модуль Analytics DB

Для анализа проблем возникающих при работе Clickhouse используются логи, вызываемые по команде:

```
journalctl -u clickhouse
```

При возникновении проблем в работе Clickhouse перезагрузите сервис командой:

```
systemctl restart clickhouse
```

Для анализа проблем возникающих при работе Redash используются логи, вызываемые по команде:

```
journalctl -u redash
```

При возникновении проблем в работе Redash перезагрузите сервис командой:

```
systemctl restart redash
```

Для анализа проблем возникающих при работе Superset используются логи, вызываемые по команде:

```
journalctl -u superset
```

При возникновении проблем в работе Superset перезагрузите сервис командой:

```
systemctl restart superset
```