

# Тестовое ростелеком №3

## Запускаем Airflow в Docker

Итак, нужно взять скрипт `rostelecom_processing`, который извлекает данные из таблицы №1, обрабатывает и помещает в таблицу №2 (Обозначения условные). Напишем Airflow DAG. Для этого для начала нужно поднять Airflow локально. Пропишем конфигурацию в `yaml` файле:

```
version: '3'
services:
  clickhouse-server-test:
    image: bitnami/clickhouse:latest
    container_name: clickhouse-server_2
    ports:
      - "8123:8123"
      - "9000:9000"
    environment:
      - CLICKHOUSE_ADMIN_PASSWORD=pivanet
    ulimits:
      nofile:
        soft: 262144
        hard: 262144

  postgres:
    image: postgres:13
    environment:
      - POSTGRES_USER=airflow
      - POSTGRES_PASSWORD=airflow
      - POSTGRES_DB=airflow
    volumes:
      - postgres_db:/var/lib/postgresql/data

  airflow-webserver:
```

```

image: apache/airflow:2.2.3
restart: always
depends_on:
  - postgres
environment:
  - AIRFLOW__CORE__EXECUTOR=LocalExecutor
  - AIRFLOW__CORE__SQL_ALCHEMY_CONN=postgresql+psycopg2://a:
  - AIRFLOW__CORE__FERNET_KEY=xDbjnEidiZ8ixmKgR-FpfcMqwg13Y
  - AIRFLOW__CORE__LOAD_EXAMPLES=False
ports:
  - "8080:8080"
volumes:
  - ./dags:/opt/airflow/dags
  - ./logs:/opt/airflow/logs
  - ./plugins:/opt/airflow/plugins
command: ["webserver"]

```

#### airflow-scheduler:

```

image: apache/airflow:2.2.3
restart: always
depends_on:
  - airflow-webserver
  - postgres
environment:
  - AIRFLOW__CORE__EXECUTOR=LocalExecutor
  - AIRFLOW__CORE__SQL_ALCHEMY_CONN=postgresql+psycopg2://a:
  - AIRFLOW__CORE__FERNET_KEY=xDbjnEidiZ8ixmKgR-FpfcMqwg13Y
volumes:
  - ./dags:/opt/airflow/dags
  - ./logs:/opt/airflow/logs
  - ./plugins:/opt/airflow/plugins
command: ["scheduler"]

```







```

volumes:
  postgres_db:

```

Запускаем контейнеры.

```
docker-compose up -d
```

|                          |   |   |  |
|--------------------------|---|---|--|
| <input type="checkbox"/> |  | <a href="#">rotest</a>  | Running (4/4)  |
| <input type="checkbox"/> |  | <a href="#">airflow-scheduler-1</a><br>5da54c086471  | <a href="#">apache/airflow:2.2.3</a><br>Running  |
| <input type="checkbox"/> |  | <a href="#">airflow-webserver-1</a><br>736a0d93751d  | <a href="#">apache/airflow:2.2.3</a><br>Running <a href="#">8080:8080</a>   |
| <input type="checkbox"/> |  | <a href="#">clickhouse-server_2</a><br>93ce69228c72  | <a href="#">bitnami/clickhouse:latest</a><br>Running <a href="#">8123:8123</a> <br><a href="#">Show all ports (2)</a> |
| <input type="checkbox"/> |  | <a href="#">postgres-1</a><br>024b15fcdce3           | <a href="#">postgres:13</a><br>Running   |

Контейнеры успешно запустились. Теперь перейдем в веб-интерфейс Airflow

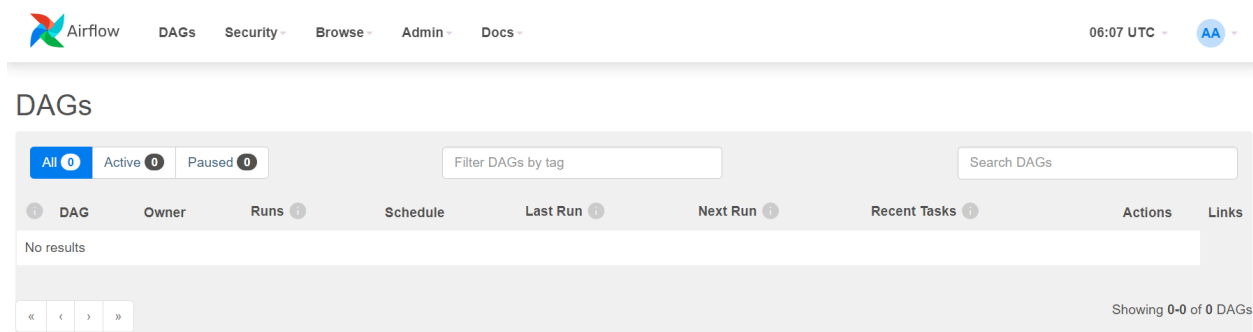
## Пишем тестовый DAG на обработку данных

Чтобы убедиться, что Airflow нормально взаимодействует с ClickHouse, напомним тестовый DAG.

Чтобы залогиниться нужно создать юзера. Сделаем стандартного юзера admin

```
docker exec -it rotest-airflow-webserver-1 bash
```

```
airflow users create --role Admin --username admin --email adm:
```

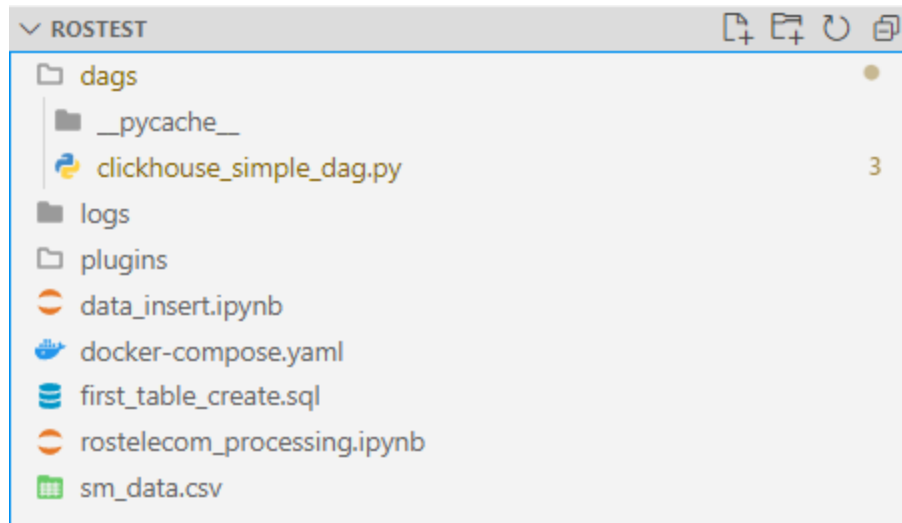


Переходим в Admin → Connections

Создаем новое соединение. Важно указать правильный host, чтобы Airflow обращался к Clickhouse по имени контейнера.

Напишем тестовый DAG, чтобы проверить, что Airflow может взаимодействовать с БД. Поместим DAG в папку dags проекта. После этого нужно установить необходимые библиотеки для работы:

```
pip install apache-airflow==2.2.3
```



Код DAG на выборку данных из таблицы:

```
from datetime import datetime, timedelta
from airflow import DAG
from airflow.providers.http.operators.http import SimpleHttpOperator
from airflow.operators.dummy import DummyOperator

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['your.email@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
    'start_date': datetime(2024, 1, 1),
}

dag = DAG(
    'clickhouse_simple_dag',
```

```

        default_args=default_args,
        description='A simple DAG to query ClickHouse',
        schedule_interval=timedelta(days=1),
        catchup=False,
    )

    start = DummyOperator(
        task_id='start',
        dag=dag,
    )

    query_clickhouse = SimpleHttpOperator(
        task_id='query_clickhouse',
        http_conn_id='clickhouse_default', # ID вашего соединения
        endpoint='', # Здесь может быть ваш endpoint, если требуется
        method='POST',
        data="SELECT * FROM db_test.logss LIMIT 10;", # Пример SQL
        headers={"Content-Type": "application/sql"},
        response_check=lambda response: True if response.status_code == 200 else False,
        dag=dag,
    )

    end = DummyOperator(
        task_id='end',
        dag=dag,
    )

    start >> query_clickhouse >> end

```

Airflow видит DAG:

# DAGs

All **1** Active **1** Paused **0**

**DAG** **Owner**

☒ **clickhouse\_simple\_dag** airflow

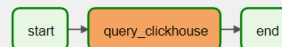
« < **1** > »

**DAG: clickhouse\_simple\_dag** A simple DAG to query ClickHouse **success** Schedule: 1 day, 0:00:00

[Tree](#) [Graph](#) [Calendar](#) [Task Duration](#) [Task Tries](#) [Landing Times](#) [Gantt](#) [Details](#) [Code](#)

25  manual\_\_2024-05-27T07:08:57.882972+00:00  Left > Right

[DummyOperator](#) [SimpleHttpOperator](#) [queued](#) [running](#) [success](#) [failed](#) [up\\_for\\_retry](#) [up\\_for\\_reschedule](#) [upstream\\_failed](#)



DAG выполнен успешно:

Логи:

```

[2024-05-27, 07:08:59 UTC] {base.py:79} INFO - Using connection
[2024-05-27, 07:08:59 UTC] {http.py:140} INFO - Sending 'POST' t
[2024-05-27, 07:08:59 UTC] {taskinstance.py:1277} INFO - Marking
    
```

```
[2024-05-27, 07:09:00 UTC] {local_task_job.py:154} INFO - Task <
[2024-05-27, 07:09:00 UTC] {local_task_job.py:264} INFO - 1 dow
```

## Пишем актуальный DAG на обработку данных

Логика будет разделена на два файла. Один (dag\_script.py) содержит процесс обработки, второй (dag\_script\_run\_v1.py) логику DAG.

Когда делаем новый DAG нужно не забыть перекинуть выполняемый скрипт в папку /dags и указать имя файла через нужный путь, который доступен Аирфлоу:

```
/opt/airflow/dags/dag_script.py
```

```
docker exec -it rostest-airflow-webserver-1 pip install clickho
```

(Установка драйвера в контейнер)

```
docker exec -it rostest-airflow-scheduler-1 pip install clickho
```

Далее мы формируем файл dag\_script\_run\_v1.py, который будет выполнять dag\_script.py в DAG. dag\_script.py представляет из себя скрипт обработки, который берет данные из таблицы №1, преобразует и вставляет в таблицу №2 (Обозначения условные)

### DAG скрипт:

```
from datetime import datetime, timedelta
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.operators.dummy import DummyOperator
import json
import pandas as pd
```



```

from clickhouse_driver import Client
from pandas import json_normalize
import re
from dag_script import *

# Функция для запуска Python скрипта
def run_python_script():
    exec(open('/opt/airflow/dags/dag_script.py').read())

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['example@email.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
    'start_date': datetime(2024, 1, 1),
}

dag = DAG(
    'clickhouse_script_execution',
    default_args=default_args,
    description='Run a Python script to interact with ClickHouse',
    schedule_interval=timedelta(days=1),
    catchup=False,
)

start = DummyOperator(
    task_id='start',
    dag=dag,
)

run_script = PythonOperator(
    task_id='run_dag_script',
    python_callable=run_python_script,

```

```

        dag=dag,
    )

    end = DummyOperator(
        task_id='end',
        dag=dag,
    )

    start >> run_script >> end

```

Airflow видит наш новый DAG (clickhouse\_script\_execution):

| DAG   | Owner   | Runs                           | Schedule       | Last Run             | Next Run             | Recent Tasks   | Actions                                      | Links |
|---|---------|--------------------------------|----------------|----------------------|----------------------|----------------|--|-------|
| <input checked="" type="checkbox"/> clickhouse_script_execution | airflow | <span>5</span> <span>10</span> | 1 day, 0:00:00 | 2024-05-27, 08:29:25 | 2024-05-27, 07:51:32 | <span>1</span> | <span>▶</span> <span>⏏</span> <span>⋮</span> |       |
| <input type="checkbox"/> clickhouse_simple_dag                  | airflow | <span>5</span> <span>2</span>  | 1 day, 0:00:00 | 2024-05-27, 07:08:57 | 2024-05-27, 06:41:42 | <span>5</span> | <span>▶</span> <span>⏏</span> <span>⋮</span> |       |

Showing 1-2 of 2 DAGs

Таблица до выполнения DAG:

The screenshot shows the DataLens application window. At the top, there's a header bar with the title "dataset\_new" and a subtitle "Введите SQL выражение чтобы отфильтровать результаты". Below the header, there's a sidebar on the left with icons for "Таблица" (Table) and "Текст" (Text). The main area displays a table with the following columns: "script\_id", "TYPE\_EQUIPMENT", "taskId", "system", "BUSINESS\_KEY", "NO\_CACHE", "HAS\_IN\_CATALOG", and "GET\_CATALOG". The table is currently empty.

Запустим DAG:

Логи:

```

[2024-05-27, 08:37:19 UTC] [standard_task_runner.py:52] INFO - Started process 2563 to run task
[2024-05-27, 08:37:19 UTC] [standard_task_runner.py:76] INFO - Running: ['***', 'tasks', 'run', 'clickhouse_script_execution', 'run_dag_script', 'manual_2024-05-27T08:37:11.636313+00:00', '--job-id', '33', '--raw', '--']
[2024-05-27, 08:37:19 UTC] [standard_task_runner.py:77] INFO - Job 33: Subtask run_dag_script
[2024-05-27, 08:37:19 UTC] [logging_mixin.py:109] INFO - Running <taskinstance: clickhouse_script_execution.run_dag_script manual_2024-05-27T08:37:11.636313+00:00 [running]> on host 5da54c086471
[2024-05-27, 08:37:19 UTC] [taskinstance.py:1426] INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_EMAIL=your_email@example.com
AIRFLOW_CTX_DAG_OWNER=***
AIRFLOW_CTX_DAG_ID=clickhouse_script_execution
AIRFLOW_CTX_TASK_ID=run_dag_script
AIRFLOW_CTX_EXECUTION_DATE=2024-05-27T08:37:11.636313+00:00
AIRFLOW_CTX_DAG_RUN_ID=manual_2024-05-27T08:37:11.636313+00:00
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Call to process_dataframe at depth 0
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Found dictionaries in columns: ['CATALOG_FILTER.filters', 'INTERACTION_TOPICS', 'CASE_TYPES', 'GET_FILTERED_SERVICES_ARRAY', 'PRODUCT_ARRAY', 'TASK_DESCRIPTION.co
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Expanded dictionaries, recursing...
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Call to process_dataframe at depth 1
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Found lists in columns: ['CATALOG_FILTER.filters.templates']
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Column CATALOG_FILTER.filters.templates after processing lists, skipping empty dicts:
0 {'operator': 'eq', 'template': 'Bonra', 'kind'...
1 {'operator': 'include', 'template': 'main_scri...
2 {'operator': 'include', 'template': 'task_feed...
Name: CATALOG_FILTER.filters.templates, dtype: object
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Processed lists, recursing...
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Call to process_dataframe at depth 2
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Found dictionaries in columns: ['CATALOG_FILTER.filters.templates', 'INTERACTION_TOPICS.theme', 'INTERACTION_TOPICS.detail', 'INTERACTION_TOPICS.result', 'CAS
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Expanded dictionaries, recursing...
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Call to process_dataframe at depth 3
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - No more lists or dictionaries to process. Exiting recursion.
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Existing data deleted successfully.
[2024-05-27, 08:37:24 UTC] [logging_mixin.py:109] INFO - Data inserted into db_test.dataset_new successfully.
[2024-05-27, 08:37:24 UTC] [python.py:175] INFO - Done. Returned value was: None
[2024-05-27, 08:37:24 UTC] [taskinstance.py:1277] INFO - Marking task as SUCCESS. dag_id=clickhouse_script_execution, task_id=run_dag_script, execution_date=20240527T083711, start_date=20240527T083719, end_date=20240527
[2024-05-27, 08:37:24 UTC] [local_task_job.py:154] INFO - Task exited with return code 0
[2024-05-27, 08:37:24 UTC] [local_task_job.py:264] INFO - 1 downstream tasks scheduled from follow-on schedule check

```

Всё выполнено успешно.

Таблица после работы скрипта:

dataset\_new

Введите SQL выражение чтобы отфильтровать результаты

|   | asc script_id   | asc TYPE_EQUIPMENT | asc taskid   | asc system   | asc BUSINESS_KEY      | 123 NO_CACHE | 123 HAS_IN_CATALOG | 123 GET_CATAL |
|---|-----------------|--------------------|--------------|--------------|-----------------------|--------------|--------------------|---------------|
| 1 | 17424b97-bf0a-4 | оборудованием      | 159caebd-6ab | CRM_B2C_PROI | 17424b97-bf0a-45f9-9c | 0            | 1                  |               |
| 2 | 7e3cfe7-53a7-4  | nan                | d22b74a9-941 | CRM_B2C_PROI | 7e3cfe7-53a7-40a9-bl  | 0            | 1                  |               |
| 3 | a67dfe4b-a531-4 | nan                | 6660d574-ab1 | CRM_B2C_PROI | a67dfe4b-a531-4366-9  | 0            | 0                  |               |

Обновить

Save

Cancel

Экспорт данных ...

200

3

MSK

ni

## Итоги:

- ClickHouse развернут в Docker контейнере
- Создана БД, таблица
- В таблицу занесены данные из тестового набора
- Сделан скрипт для обработки тестовых данных и создания таблицы.
- Создана таблица для дальнейшей работы с данными специалистами по BI
- Развернут Airflow в Docker контейнере
- Написан DAG, который берет данные из первой таблицы, обрабатывает и заносит во вторую таблицу