

Загрузка данных в ClickHouse

1. Разверните инструмент для переливки данных (например, Airflow или Airbyte).
2. Разверните ClickHouse.
3. Выберите любой источник данных (API, сервис, БД и пр.).
4. Настройте подключение инструмента к ClickHouse.
5. Постройте пайплайн для переливки данных в ClickHouse с возможностью регулярного или одноразового выполнения.

1. На локальной машине был поднят доккер с AF.

```
airflow-init-1 exited with code 0
root@NB-R911LJCT:~/otus_airflow# docker compose up -d
[+] Running 7/7
 ✓ Container otus_airflow-postgres-1      Healthy
 ✓ Container otus_airflow-redis-1         Healthy
 ✓ Container otus_airflow-airflow-init-1   Exited
 ✓ Container otus_airflow-airflow-worker-1 Started
 ✓ Container otus_airflow-airflow-scheduler-1 Started
 ✓ Container otus_airflow-airflow-webserver-1 Started
 ✓ Container otus_airflow-airflow-triggerer-1 Started
root@NB-R911LJCT:~/otus_airflow# docker ps
```

В него были добавлены необходимые зависимости – clickhouse-connect

```
root@NB-R911LJCT:~# docker exec -it otus_airflow-airflow-worker-1 /bin/bash
root@cda9c9f35122:/opt/airflow# su - airflow
airflow@cda9c9f35122:~$ pip install clickhouse-connect
Collecting clickhouse-connect
  Downloading clickhouse_connect-0.8.17-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.4 kB)
Requirement already satisfied: certifi in ./local/lib/python3.12/site-packages (from clickhouse-connect) (2025.1.31)
Requirement already satisfied: urllib3>=1.26 in ./local/lib/python3.12/site-packages (from clickhouse-connect) (2.3.0)
Requirement already satisfied: pytz in ./local/lib/python3.12/site-packages (from clickhouse-connect) (2025.1)
Collecting zstandard (from clickhouse-connect)
  Downloading zstandard-0.23.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.0 kB)
Collecting lz4 (from clickhouse-connect)
  Downloading lz4-4.4.4-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.8 kB)
Downloading clickhouse_connect-0.8.17-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.1 MB)
  1.1/1.1 MB 14.5 MB/s eta 0:00:00
Downloading lz4-4.4.4-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
  1.3/1.3 MB 30.9 MB/s eta 0:00:00
Downloading zstandard-0.23.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (5.4 MB)
  5.4/5.4 MB 40.4 MB/s eta 0:00:00
Installing collected packages: zstandard, lz4, clickhouse-connect
Successfully installed clickhouse-connect-0.8.17 lz4-4.4.4 zstandard-0.23.0
```

2. Развернул clickhouse на той же машине, создал в нем базу данных otus и две демонстрационных таблицы «otus.read_from» - таблица ИЗ которой в даге будем читать данные и «otus.write_in» - таблица куда будем данные записывать. «otus.read_from» наполнил случайными данными (150 строк).
3. Для демонстрации решил, что читать буду из того же клика, куда и писать.

4. Подключение было настроено через clickhouse-connect, креды положил в connections AF.

```
# Функция для получения данных из соединения
def write_data_to_clickhouse(**kwargs) -> dict: 1 usage
    # Получаем ID соединения из параметров
    conn_id = kwargs.get('conn_id')

    # Извлекаем соединение
    connection = BaseHook.get_connection(conn_id)

    # Получаем данные соединения
    conn_data = {
        'host': connection.host,
        'port': connection.port,
        'user': connection.login,
        'password': connection.password,
        'database': connection.schema,
    }

    client = get_client(**conn_data)
```

5. Был составлен DAG на чтение из клика из одной таблицы и вставка в другую.

```
cda9c9f35122
  ▶ Log message source details
[2025-04-20, 10:23:37 UTC] {local_task_job_runner.py:123} ▶ Pre task execution logs
[2025-04-20, 10:23:37 UTC] {base.py:84} INFO - Retrieving connection 'clickhouse_local'
[2025-04-20, 10:23:37 UTC] {logging_mixin.py:190} INFO - Таблица из которой мы читаем данные имеет размерность: (150, 2)
[2025-04-20, 10:23:37 UTC] {logging_mixin.py:190} INFO - Таблица в которую данные будут записаны ДО записи имеет размерность: (0, 0)
[2025-04-20, 10:23:37 UTC] {logging_mixin.py:190} INFO - Таблица в которую записались данные ПОСЛЕ записи имеет размерность: (150, 2)
[2025-04-20, 10:23:37 UTC] {python.py:240} INFO - Done. Returned value was: None
[2025-04-20, 10:23:37 UTC] {taskinstance.py:341} ▶ Post task execution logs
```

Из логов AF видно, что сначала мы читаем данные из otus.read_from, которая имеет размерность 150 строк и 2 столбца. Читаем данные из otus.write_in и убеждаемся что она пустая, затем записываем в нее то что прочли из otus.read_from и убеждаемся, что данные записались (размерность стала 150,2)

Код ДАГА:

```
from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.hooks.base_hook import BaseHook
from datetime import datetime
from clickhouse_connect import get_client

# Функция для получения данных из соединения
def write_data_to_clickhouse(**kwargs) -> dict: 1 usage
    # Получаем ID соединения из параметров
    conn_id = kwargs.get('conn_id')

    # Извлекаем соединение
    connection = BaseHook.get_connection(conn_id)

    # Получаем данные соединения
    conn_data = {
        'host': connection.host,
        'port': connection.port,
        'user': connection.login,
        'password': connection.password,
        'database': connection.schema,
    }

    client = get_client(**conn_data)
    df_from = client.query_df("""SELECT * FROM otus.read_from""")
    print("Таблица из которой мы читаем данные имеет размерность: ", df_from.shape)

    df_in = client.query_df("""SELECT * FROM otus.write_in""")
    print("Таблица в которую данные будут записаны ДО записи имеет размерность: ", df_in.shape)

    client.insert_df(
        table="otus.write_in",
        df=df_from
    )

    df_in_after = client.query_df("""SELECT * FROM otus.write_in""")
    print("Таблица в которую записались данные ПОСЛЕ записи имеет размерность: ", df_in_after.shape)

# Определяем DAG
with DAG(
    dag_id='write_data_to_clickhouse',
    schedule_interval='@daily',
    start_date=datetime(year=2023, month=1, day=1),
    catchup=False,
) as dag:
    # Определяем задачу
    task = PythonOperator(
        task_id='write_data_to_clickhouse_task',
        python_callable=write_data_to_clickhouse,
        op_kwargs={'conn_id': 'clickhouse_local'}, # Замените на ваш ID соединения
    )

    task
```