

Департамент образования города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»

Институт цифрового образования
Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

«Проектный практикум по разработке ETL-решений »

Практическая работа 28.03

«Бизнес-кейс «Rocket»»

Выполнила:

Студентка группы АДЭУ-211

Кравцова Алёна Евгеньевна

Руководитель:

Босенко Т.М

Москва

2025

Цель работы: спроектировать и реализовать верхнеуровневую архитектуру Rocket, а также построить архитектуру DAG. Запустить и разобрать бизнес-кейс Rocket.

Задачи:

1. Сделать файл с расширением .sh для копирования полученных изображений в локальную ОС;
2. Спроектировать верхнеуровневую архитектуру аналитического решения задания Бизнес-кейса «Rocket» в draw.io;
3. Спроектировать архитектуру DAG Бизнес-кейса «Rocket» в draw.io;
4. Выполнить индивидуальное задание.

Перед выполнением задач работы необходимо настроить систему. После клонирования репозитория перейдем в папку business_case_rocket_25 (Рис. 1), в которой расположен интересующий dockerfile и приступим к сборке образа (Рис. 2).

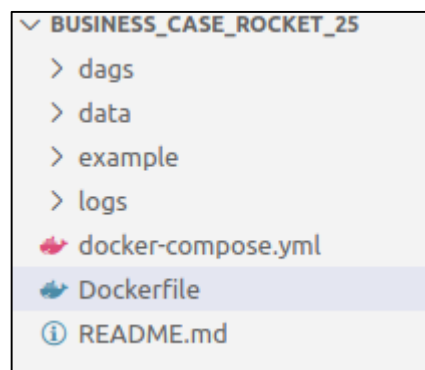


Рис. 1 – Необходимая директория

```
● mgpu@mgpu-VirtualBox:~/Downloads/workshop-on-ETL-main/business_case_rocket_25$ sudo docker build -t custom-airflow
w:slim-2.8.1-python3.11 .
[+] Building 79.5s (7/7) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 568B                                0.0s
=> [internal] load metadata for docker.io/apache/airflow:slim-2.8.1-python3.11 1.8s
=> [internal] load .dockerignore                                    0.0s
=> => transferring context: 2B                                       0.0s
=> [1/3] FROM docker.io/apache/airflow:slim-2.8.1-python3.11@sha256:751babb58a83e44ae23c393fe1552196c25f 40.9s
=> => resolve docker.io/apache/airflow:slim-2.8.1-python3.11@sha256:751babb58a83e44ae23c393fe1552196c25f3 0.0s
=> => sha256:51d1f07906b71fd60ac43c61035514996a8ad8dbfd39d4f570ac5446b064ee5d 3.51MB / 3.51MB 1.0s
=> => sha256:fe87ad6b112e2dffa9a52f49adf5bb70a80d9af2c737f71a947cb5017a12b148 12.87MB / 12.87MB 2.8s
=> => sha256:3ee88b8d122ebb0fbb9be864918a05a7621f1b4e1801154b2a0bd64e9476c333 4.47kB / 4.47kB 0.0s
=> => sha256:751babb58a83e44ae23c393fe1552196c25f3e2683c97db1a6d98b7d15e7a0e8 1.61kB / 1.61kB 0.0s
=> => sha256:e1caac4eb9d2ec24aa3618e5992208321a92492aef5fef5eb9e470895f771c56 29.12MB / 29.12MB 8.7s
=> => sha256:a205efa96734ac8633bf8d388ed9b6cd527835d31ebee070ba1cedfb880b4ca4 25.59kB / 25.59kB 0.0s
=> => sha256:4d8ccb72bbadfe34ab482a41ca4c7c07b97dfa2e523cb6317b4ff5948244765b 244B / 244B 1.3s
=> => sha256:8100581c78ddcf0a8c7cb0eb6b3028d3045eaf4f34d4704a6c72ed0b3f5714a 3.41MB / 3.41MB 2.9s
=> => sha256:93b9b11f045531996f84ad513fddafe1bd8a638797c3cfa2632eabea0bb59df 1.64kB / 1.64kB 0.0s
```

Рис. 2 – Сборка airflow

Далее поднимем все сервисы (Рис. 3).

```
● mgpu@mgpu-VirtualBox:~/Downloads/workshop-on-ETL-main/business_case_rocket_25$ sudo docker compose up --build
[+] Running 12/12
✓ postgres 11 layers [████████████████████] 0B/0B Pulled 17.1s
✓ 1f3e46996e29 Pull complete 0.9s
✓ 47e20ba03731 Pull complete 0.5s
✓ 101b82465a4f Pull complete 0.6s
✓ 319529a7ccb0 Pull complete 1.2s
✓ c2f9392cfd4c Pull complete 1.2s
✓ 4e04446ce95d Pull complete 10.1s
✓ 47bfe778b869 Pull complete 2.6s
✓ b1d66b287aa8 Pull complete 3.9s
✓ 7865e52a4759 Pull complete 5.2s
✓ 7d75f14147c2 Pull complete 8.0s
✓ 11052a5424e7 Pull complete 8.0s
[+] Running 7/7
✓ Network business_case_rocket_25_default Created 0.2s
✓ Volume "business_case_rocket_25_postgres_data" Crea... 0.0s
✓ Volume "business_case_rocket_25_logs" Created 0.0s
✓ Container business_case_rocket_25-postgres-1 Create...
✓ Container business_case_rocket_25-init-1 Created
```

Рис. 3 – Запуск сервисов

Airflow запущен успешно (Рис. 4).

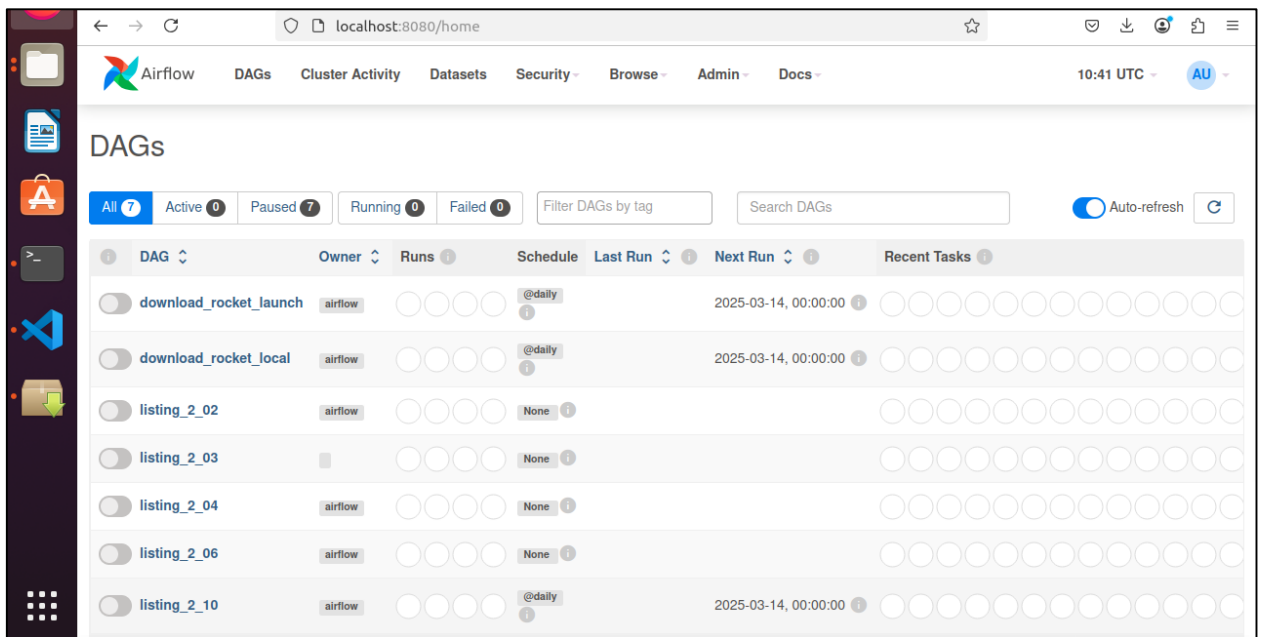


Рис. 4 – Airflow

Проверим работоспособность и запустим DAG, предварительно выдадим все права на папку data (Рис. 5).

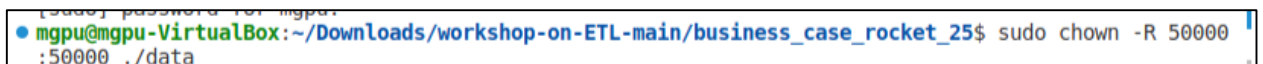


Рис. 5 – Настройка прав доступа

Итак, DAG выполнен успешно (Рис. 6).

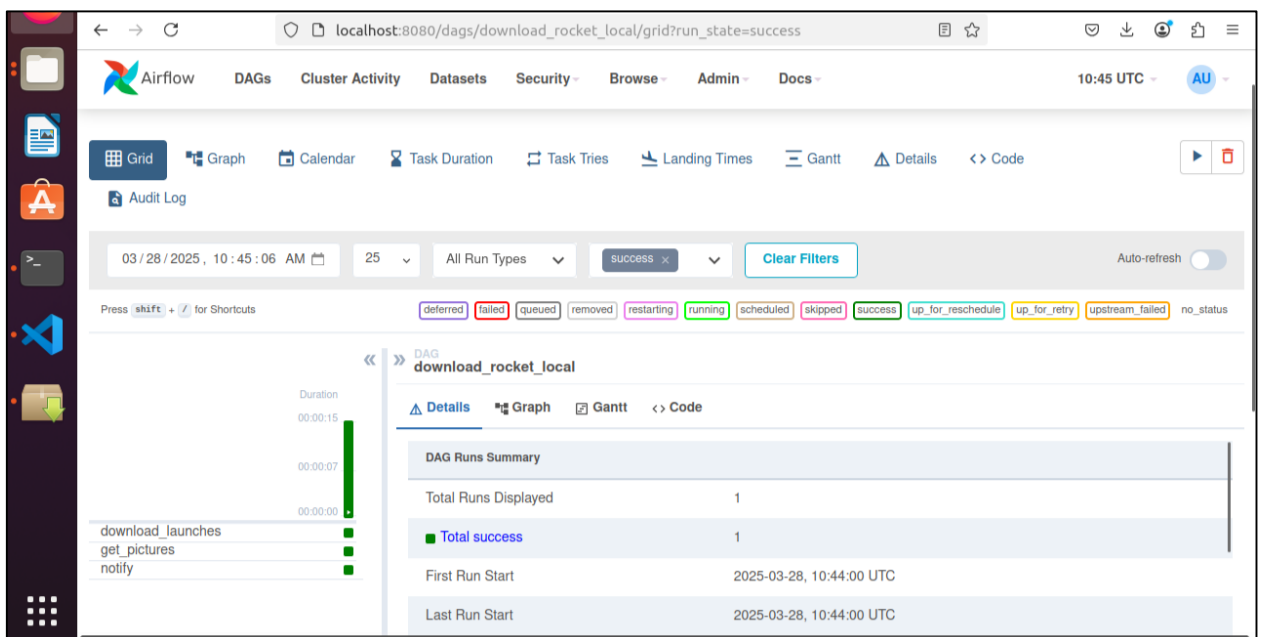


Рис. 6 – Успешный запуск DAG

И в папке data появились изображения (Рис. 7).

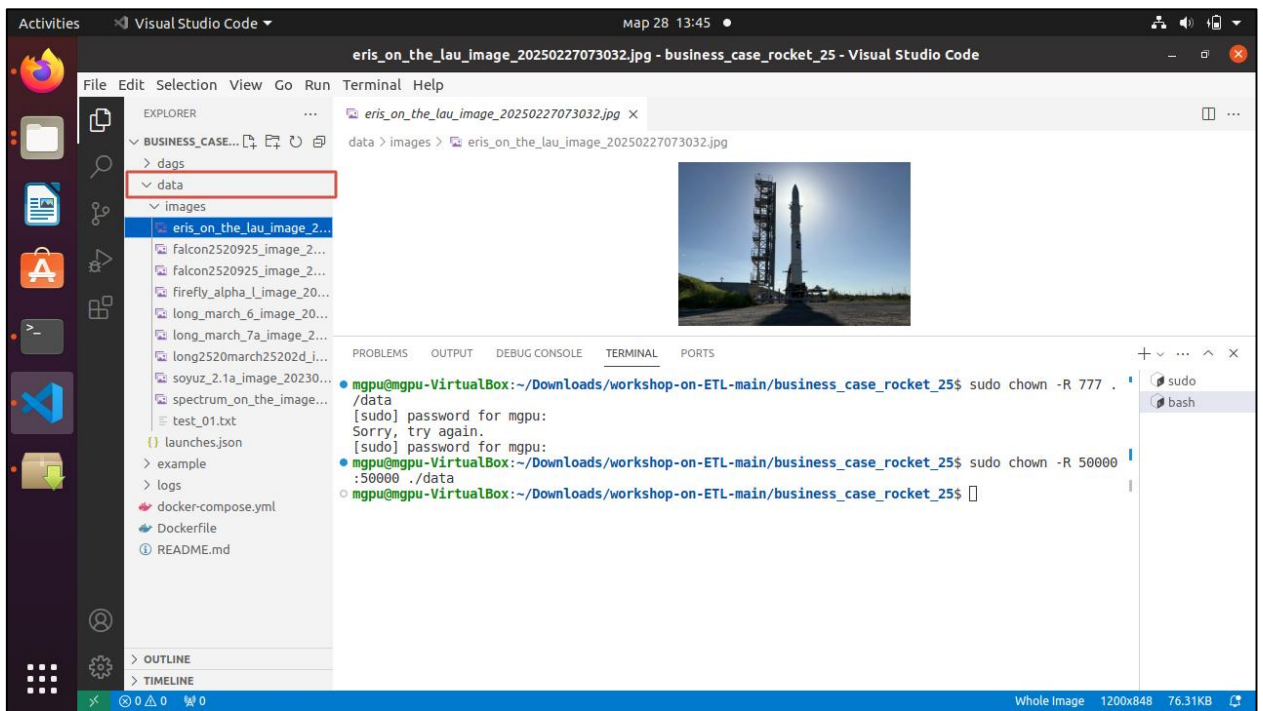


Рис. 7 – Полученные изображения

Задание 1.1. Спроектировать верхнеуровневую архитектуру аналитического решения задания Бизнес-кейса «Rocket» в draw.io. Необходимо использовать:

- Source Layer - слой источников данных.
- Storage Layer - слой хранения данных.
- Business Layer - слой для доступа к данным пользователей.

На рисунке 8 представлена получившаяся архитектура

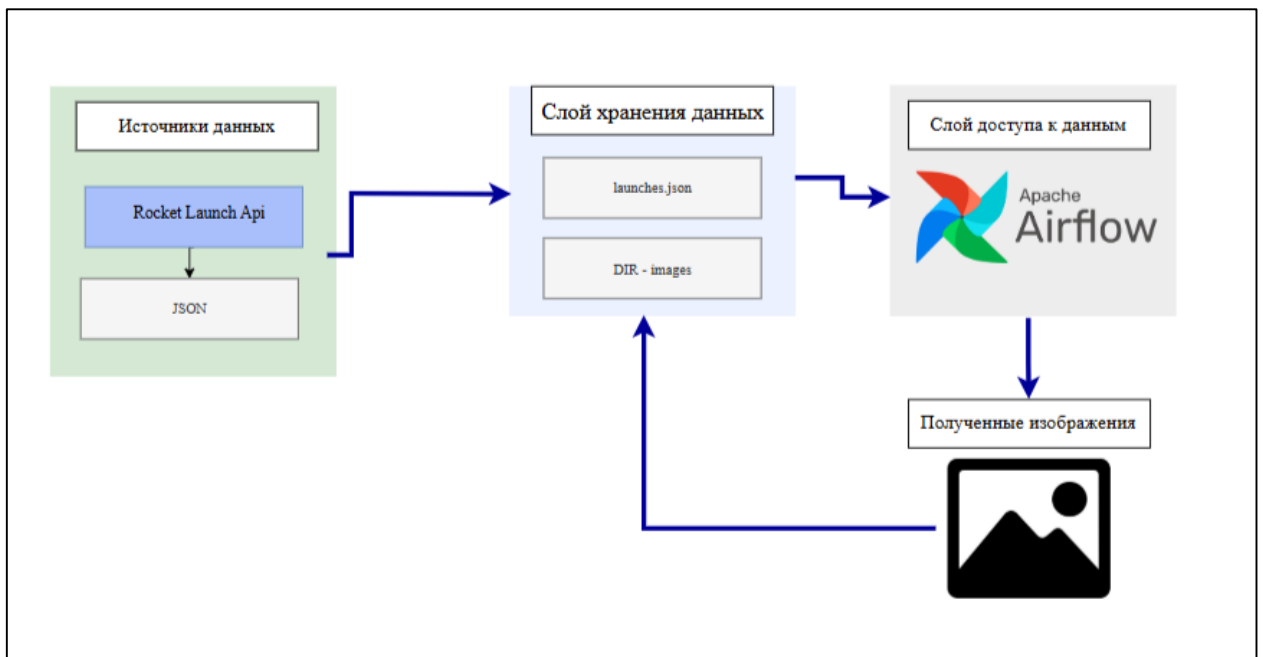


Рис. 8 – Архитектура кейса Rocket

Задание 1.2. Спроектировать архитектуру DAG Бизнес-кейса «Rocket» в draw.io. Необходимо использовать:

- Source Layer - слой источников данных.
- Storage Layer - слой хранения данных.
- Business Layer - слой для доступа к данным пользователей.

Архитектура DAG представлена на рисунке 9.

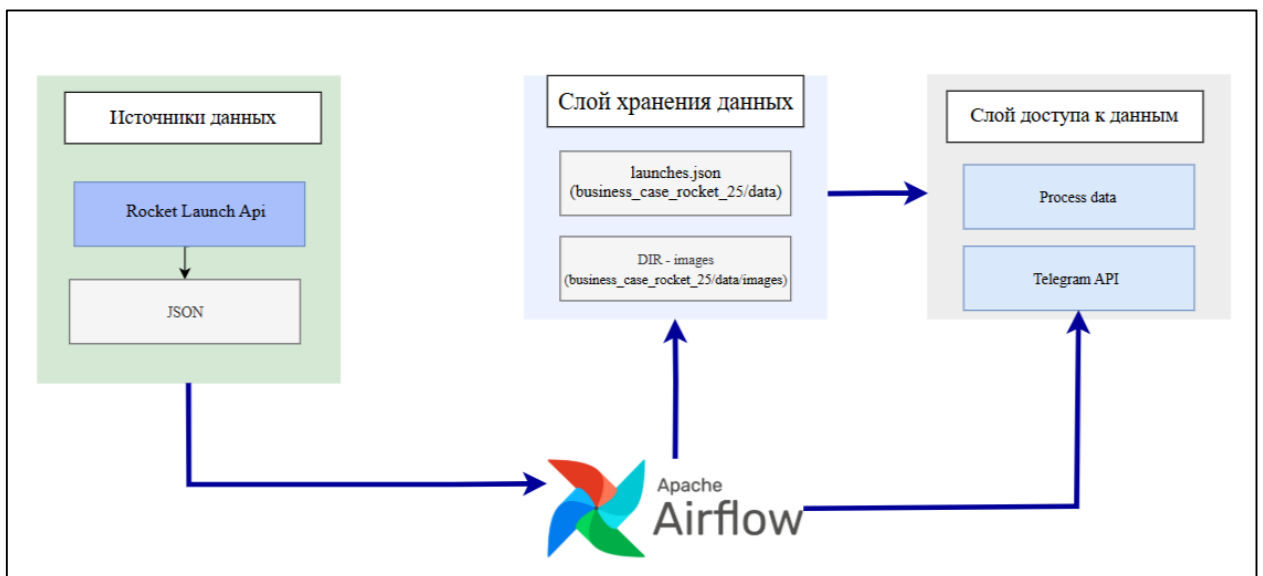


Рис. 9 – Архитектура DAG с учетом задания

Задание 2. Копирование данных в локальную ОС.

Далее создадим файл copy.sh (Рис. 10), который выгружает данные в ОС.

```
$ cat copy.sh
1  #!/bin/bash
2
3  # Укажите путь к исходной и целевой папкам
4  SOURCE_DIR="/home/mgpu/Downloads/workshop-on-ETL-main/business_case_rocket_25/data/images"
5  DEST_DIR="/home/mgpu/Downloads/photo" # папка для копирования
6  LOG_FILE="/home/mgpu/Downloads/workshop-on-ETL-main/business_case_rocket_25/logs/copy.log"
7
8  # Проверяем существование папок
9  if [ ! -d "$SOURCE_DIR" ]; then
10     echo "$(date): Ошибка! Исходная папка не найдена: $SOURCE_DIR" | tee -a "$LOG_FILE"
11     exit 1
12 fi
13
14 if [ ! -d "$DEST_DIR" ]; then
15     echo "$(date): Папка назначения не найдена, создаем: $DEST_DIR" | tee -a "$LOG_FILE"
16     mkdir -p "$DEST_DIR"
17 fi
18
19 # Копирование файлов
20 echo "$(date): Начинаем копирование файлов из $SOURCE_DIR в $DEST_DIR." | tee -a "$LOG_FILE"
21 cp -r "$SOURCE_DIR"/* "$DEST_DIR"
22
23 # Завершающее сообщение
24 echo "$(date): Процесс копирования завершен." | tee -a "$LOG_FILE"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Пт 28 мар 2025 15:08:09 MSK: Начинаем копирование файлов из /home/mgpu/Downloads/workshop-on-ETL-main/business_case_rocket_25/data/images в /home/mgpu/Downloads/photo.
Пт 28 мар 2025 15:08:09 MSK: Процесс копирования завершен.

mgpu@mgpu-VirtualBox: ~/Downloads/workshop-on-ETL-main/business_case_rocket_25

Рис. 10 – copy.sh

После запуска данные скопировались в указанную папку Downloads/photo (Рис. 11).

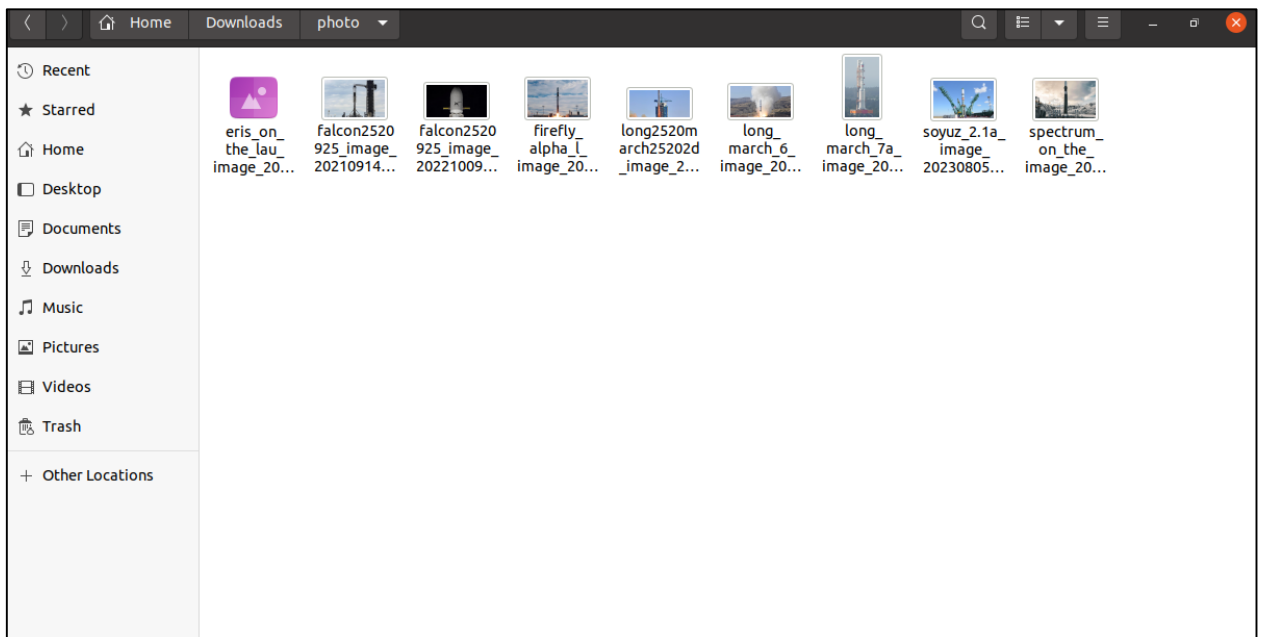


Рис. 11 – Результат копирования файлов

Индивидуальное задание. Вариант 6.

Настроить Airflow для автоматической обработки данных по запуску ракет	Реализовать систему уведомлений при сбоях загрузки изображений	Провести анализ на количество и частоту запусков ракет за последний месяц
--	--	---

1. Настроить Airflow для автоматической обработки данных по запуску ракет.

Для автоматической обработки данных по запуску ракет необходимо в коде DAG прописать параметр `schedule`, например, ежедневно (Рис. 12).

```
schedule_interval="@daily",
```

Рис. 12 – Установка расписания запуска

При данном параметре DAG будет запускаться автоматически каждый день с 00:00 текущего дня до 00:00 следующего дня (Рис. 13).

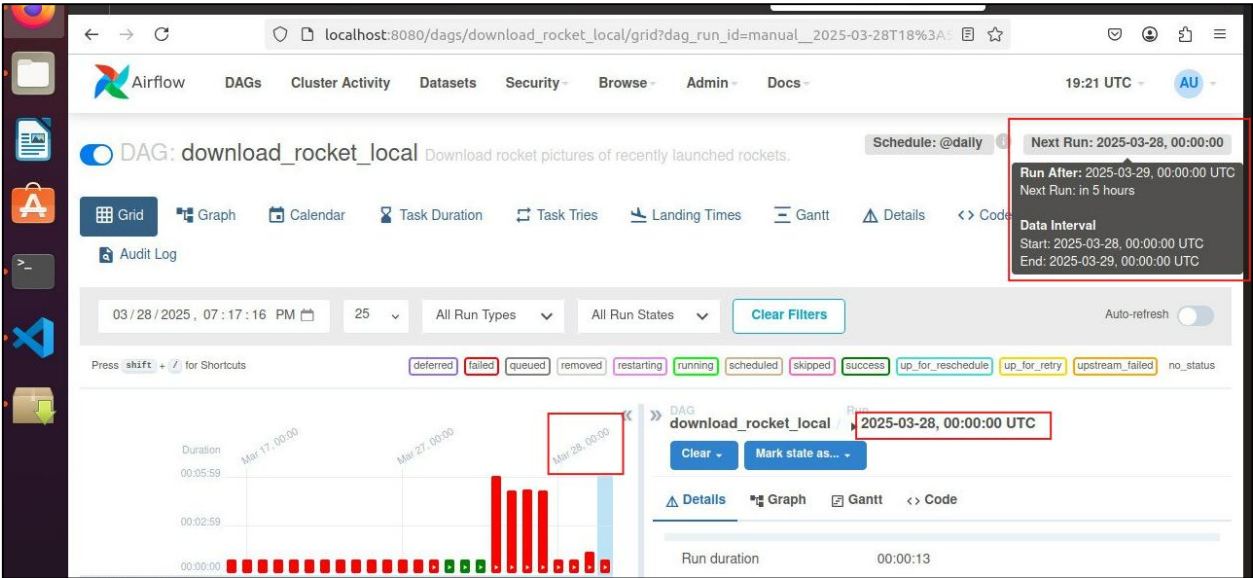


Рис. 13 – DAG запущенный автоматически

2. Реализовать систему уведомлений при сбоях загрузки изображений.

Для отправки уведомлений будет выбран Telegram бот за счет простоты внедрения и повышения удобства.

Сперва необходимо создать тг бота и начать с ним общение, чтобы получить `TOKEN_ID` и `CHAT_ID`. `TOKEN_ID` узнается после создания бота (Рис. 14).

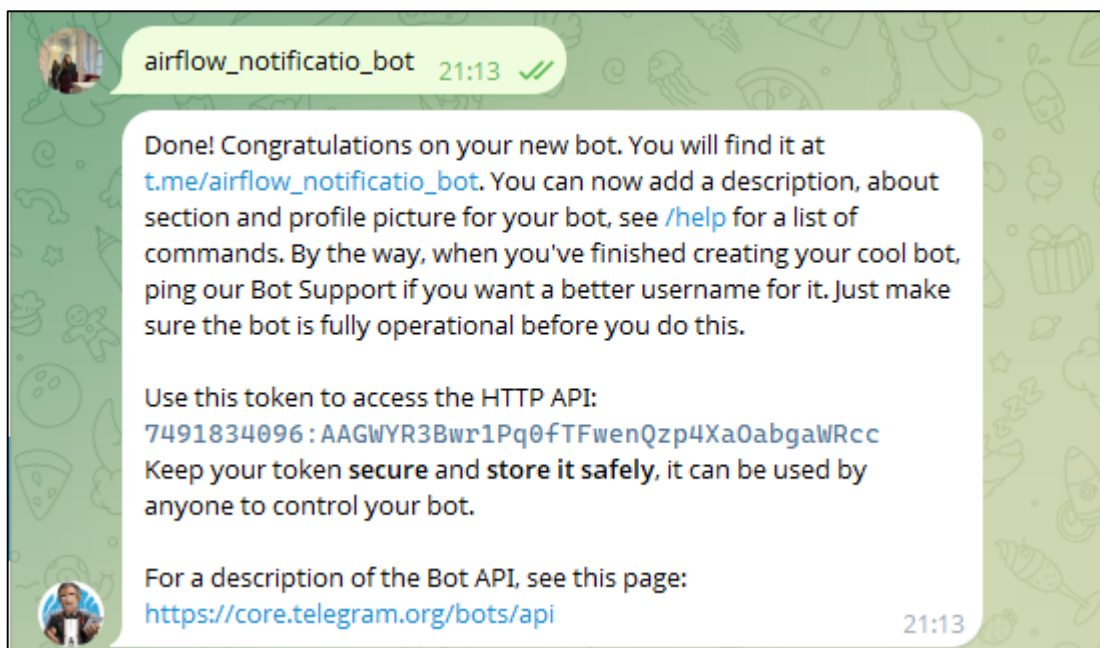


Рис. 14 – TOKEN_ID

Далее введем в браузере api запрос, в ответе которого расположен CHAT_ID (Рис. 15).



Рис. 15 – CHAT_ID

Далее добавим в даг следующий код:

```
import requests
```

```
TELEGRAM_BOT_TOKEN = " 7491834096:AAGWYR3Bwr1Pq0fTFwenQzp4Xa0abgaWRcc " #
Токен бота
```

```
TELEGRAM_CHAT_ID = " 1482158775" # chat_id
```

```
def send_telegram_alert(context):  
    """Функция отправки сообщений в Telegram при ошибке DAG."""  
    message = f"❌ Ошибка в DAG {context['task_instance'].dag_id}\nTask:  
{context['task_instance'].task_id}\nError: {context['exception']}"  
    url = f"https://api.telegram.org/bot{TELEGRAM_BOT_TOKEN}/sendMessage"  
    params = {"chat_id": TELEGRAM_CHAT_ID, "text": message}  
    try:  
        response = requests.get(url, params=params)  
        response.raise_for_status()  
    except requests.exceptions.RequestException as e:  
        print(f"Ошибка отправки в Telegram: {e}")
```

Далее специально допустим ошибку в Dag'е для проверки уведомления, например, в наименовании переменной (Рис. 16).

```
40 def _get_pictures():  
41     images_dir = "/opt/airflow/data/images"  
42     pathlib.Path(images_dir).mkdir(parents=True, exist_ok=True)  
43  
44     with open("/opt/airflow/data/launches.json") as f:  
45         launches = json.load(f)  
46         image_urls = [launch["image"] for launch in launches["results"]]  
47         for image_url in image_urls:  
48             try:  
49                 response = requests.get(image_url)  
50                 response.raise_for_status()
```

Рис. 16 – Допущение ошибки в наименовании переменной

Далее перезапустим контейнеры и запустим даг. Даг отработал с ошибкой (Рис. 17).

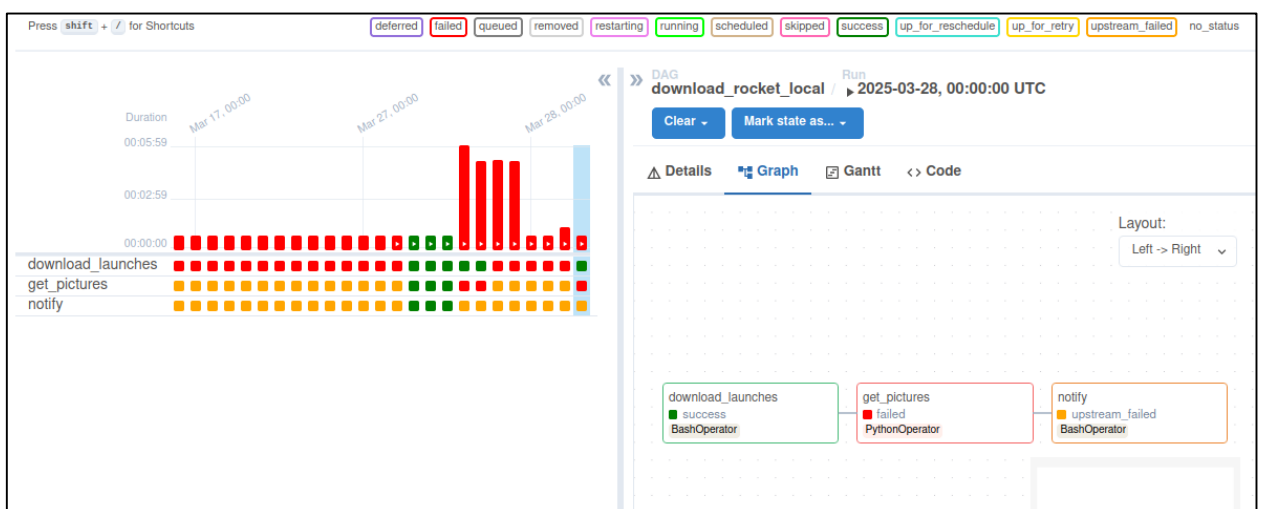


Рис. 17 – Результат отработки DAG с ошибкой

После чего пришло уведомление от тг-бота (Рис. 18).

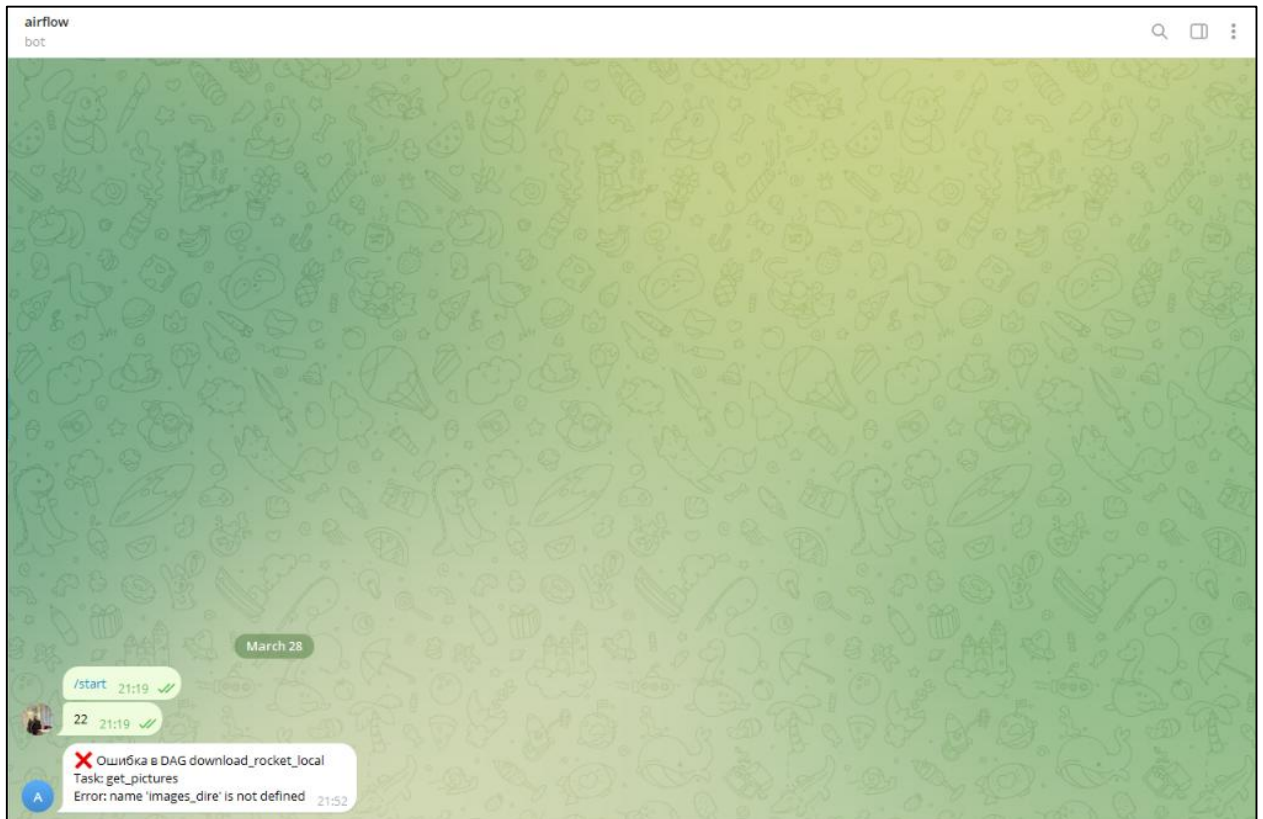


Рис. 18 – Уведомление об ошибке от тг-бота

3. Провести анализ на количество и частоту запусков ракет за последний месяц

Загрузим launches.json в google colab для дальнейшего анализа (Рис. 19).



Рис. 19 – Загрузка json файла

Посмотрим типы данных. Для выполнения задания понадобится параметр даты, который имеет тип object. Необходимо преобразовать в тип datetime (Рис. 20).

```
[6] 1 df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 24 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   id                                    10 non-null     object
1   url                                    10 non-null     object
2   launch_library_id                    0 non-null      object
3   slug                                  10 non-null     object
4   name                                  10 non-null     object
5   status                               10 non-null     object
6   net                                   10 non-null     object
7   window_end                           10 non-null     object
8   window_start                         10 non-null     object
9   inhold                               10 non-null     bool
10  tbddtime                             10 non-null     bool
11  tbddate                               10 non-null     bool
12  probability                           0 non-null      object
13  holdreason                            10 non-null     object
14  failreason                            10 non-null     object
15  hashtag                               0 non-null      object
16  launch_service_provider              10 non-null     object
17  rocket                                10 non-null     object
18  mission                              10 non-null     object
19  pad                                    10 non-null     object
20  webcast_live                         10 non-null     bool
21  image                                 10 non-null     object
22  infographic                           0 non-null      object
23  program                              10 non-null     object
dtypes: bool(4), object(20)
memory usage: 1.7+ KB

[7] 1 df["window_start"] = pd.to_datetime(df["window_start"])
```

Рис. 20 – Преобразование в тип datetime

Далее посмотрим данные, которые есть в файле (Рис. 21).

```
[33] 1 print("Минимальная дата в данных:", df["window_start"].min())
      2 print("Максимальная дата в данных:", df["window_start"].max())
      3 print("Сегодняшняя дата (UTC):", today)
      4 print("Дата 30 дней назад (UTC):", last_month)
      5

Минимальная дата в данных: 2025-03-29 11:30:00+00:00
Максимальная дата в данных: 2025-04-04 22:42:00+00:00
Сегодняшняя дата (UTC): 2025-03-29 05:11:41.462977+00:00
Дата 30 дней назад (UTC): 2025-02-27 05:11:41.462977+00:00
```

Рис. 21 – Минимальная и максимальная дата

Таким образом, данных за последние 30 дней нет, поэтому проведем анализ данных с 29 марта по 4 апреля. Построим bar chart по сгруппированным количествам запусков по дате (Рис. 22).

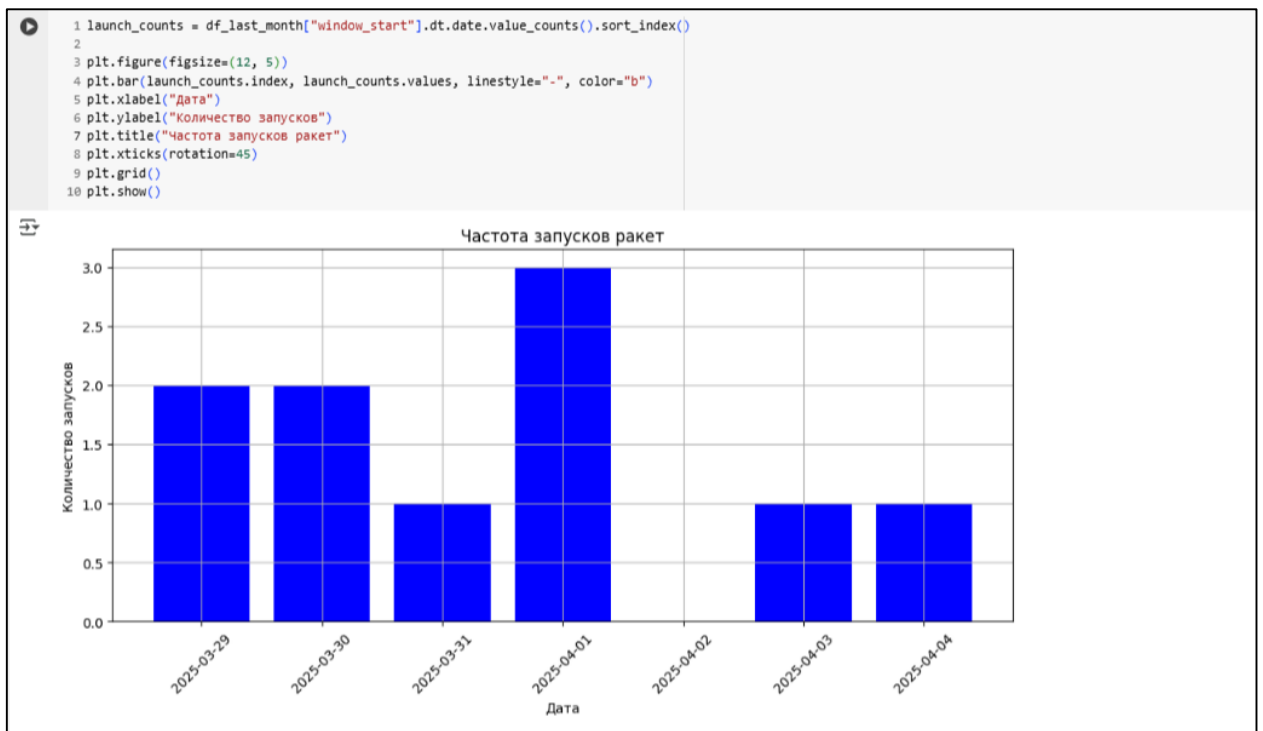


Рис. 22 – Частота запусков ракет

Закключение: таким образом была закреплена работа с airflow, а именно проработка DAG, его запуск и выгрузка данных. Был получен опыт подключения уведомлений через тг-бота. Все задачи были выполнены.