

Департамент образования и науки 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

Вебинар 28.03.2025

Практическая работа на вебинаре

Выполнила: Шведова С.С., группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2025

Цели:

1. Спроектировать архитектуру
2. Создать даг с индивидуальным заданием
3. Создать файл sh

На рисунке 1 продемонстрирована верхнеуровневая архитектура.

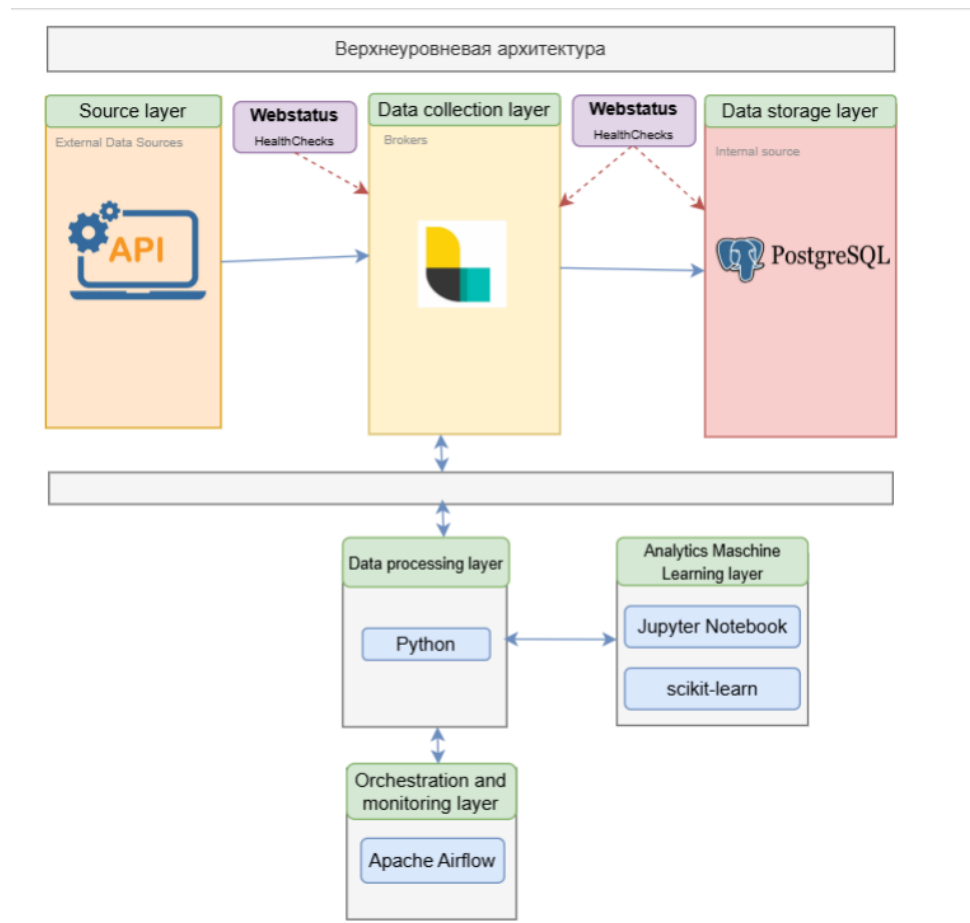


Рисунок 1. Верхнеуровневая архитектура

1. Слой источника данных (API Rocket): Использование API для сбора данных — это распространенный подход.
2. Слой сбора данных (Logstash): Logstash — это мощный инструмент для сбора, обработки и отправки данных. Он хорошо интегрируется с различными источниками данных и может обрабатывать данные в реальном времени.

3. Хранение данных (PostgreSQL): PostgreSQL — это надежная реляционная база данных, которая поддерживает сложные запросы и транзакции. Она подходит для хранения структурированных данных.

4. Обработка данных (Python): Python обеспечивает обработку данных и является простым инструментом с множеством библиотек.

5. Аналитика и машинное обучение (Jupyter Notebook и scikitlearn): Jupyter Notebook предоставляет удобную среду для анализа данных и разработки моделей машинного обучения. Scikit-learn — это мощная библиотека для машинного обучения, которая поддерживает множество алгоритмов.

6. Оркестрация и мониторинг (Apache Airflow): Apache Airflow — это отличный инструмент для оркестрации рабочих процессов. Он позволяет планировать и управлять задачами, а также отслеживать их выполнение.

Эта архитектура позволяет эффективно собирать, хранить, обрабатывать и анализировать данные, обеспечивая гибкость и масштабируемость.

Общая оценка экономической эффективности:

1. Снижение затрат на разработку: Данная архитектура снижает затраты на разработку благодаря использованию открытых технологий и интеграции между компонентами.

2. Улучшение производительности: Применение современного стека технологий позволяет значительно ускорить процессы сбора, обработки и аналитики данных.

3. Масштабируемость: Все компоненты могут быть масштабированы, что увеличивает их функциональность и позволяет приспособиться к изменяющимся требованиям бизнеса.

На рисунке 2 показана архитектура dag.

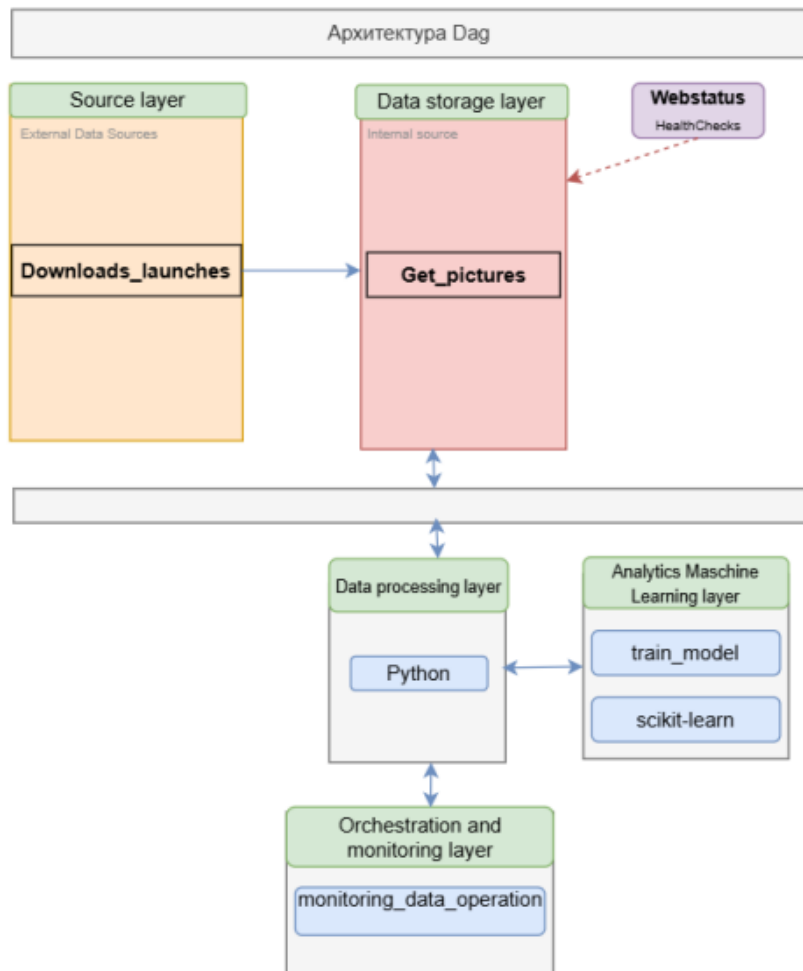


Рисунок 2. Архитектура дага

1. Слой источника данных (**Downloads_launches**) Этот компонент отвечает за сбор данных. Он может представлять собой систему, которая загружает данные о запусках. Данные извлекаются и подготавливаются для передачи в следующий компонент.

2. Хранение данных (**get_pictures**) Этот компонент, вероятно, отвечает за хранение изображений. После извлечения данных из **Downloads_launches**, они сохраняются в **get_pictures**.

3. Обработка данных (**Python**) Python используется для обработки данных, полученных из **get_pictures**. Это может включать в себя очистку, преобразование и подготовку данных для анализа. С помощью библиотек,

таких как Pandas или NumPy, вы можете извлекать данные из `get_pictures`, обрабатывать их и подготавливать для анализа.

4. Аналитика и машинное обучение (`train_model` и `scikit-learn`) Этот компонент отвечает за обучение моделей машинного обучения с использованием `scikit-learn`. Он может включать в себя как обучение, так и оценку моделей. После обработки данных используется `scikit-learn` для создания и обучения моделей.

5. Оркестрация и мониторинг (`monitoring_data_operation`) Этот компонент отвечает за управление и мониторинг всего рабочего процесса. Он может включать в себя инструменты, такие как Apache Airflow, для автоматизации задач и отслеживания их выполнения. Создается DAG, который описывает последовательность задач, начиная с `Downloads_launches`, затем переходя к `get_pictures`, обработке данных с помощью Python, обучению модели и, наконец, мониторингу выполнения всех этих задач.

Все компоненты, использованные в данной архитектуре, являются бесплатными, тем самым снижая затраты компании.

Далее на рисунке 3 надо написать команду, которая вызывает утилиту `cURL`, используемая для передачи данных с использованием различных протоколов (в данном случае HTTP).

```
dev@dev-vm:~/workshop-on-ETL/business_case_rocket_25$ curl -L "https://ll.thespacedevs.com/2.0.0/launch/upcoming" -o data_space.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
  0     0     0     0     0     0     0     0     0     0
100 25992 100 25992     0     0 34919     0     0     0
dev@dev-vm:~/workshop-on-ETL/business_case_rocket_25$
```

Рисунок 3. Команда `curl -L`

На рисунке 4 показан граф дага `download_rocket_local`, который успешно отработал.

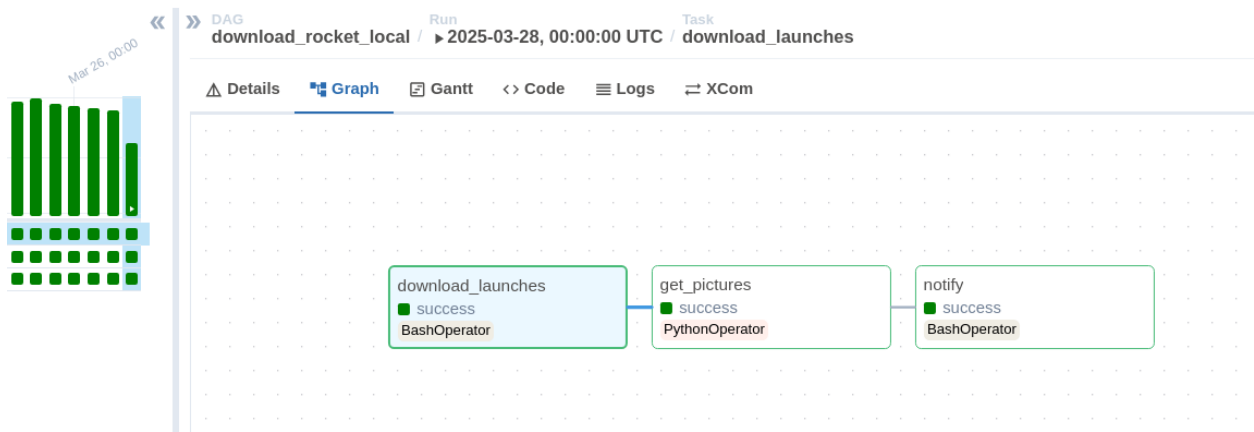


Рисунок 4. Даг отработал

На рисунке 5 показана диаграмма Ганта этого дага

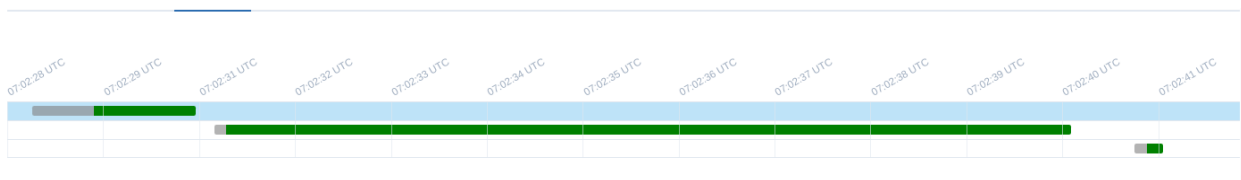


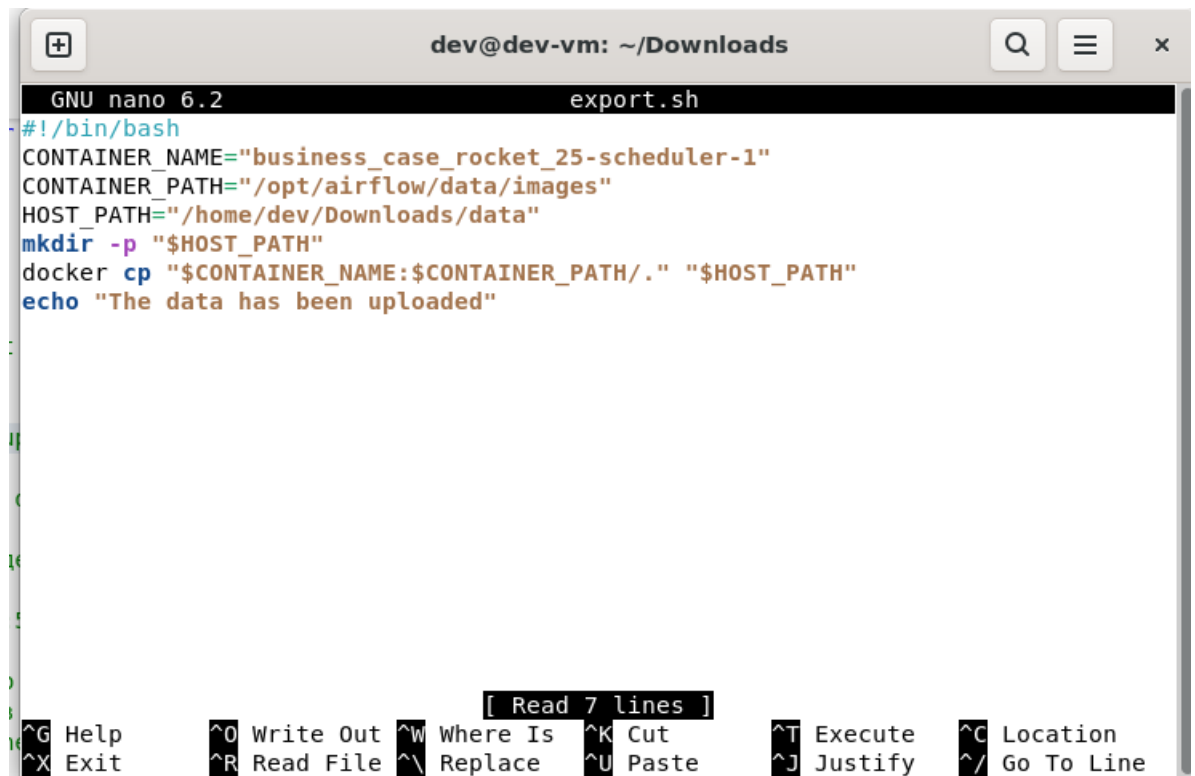
Рисунок 5. Диаграмма Ганта

На рисунке 6 показаны логи

```
3-28T07:02:28.916508+00:00, map_index=-1, run_start_date=2025-03-28 07:02:29.864274+00:00, run_end_date=2025-03-28 07:02:30.985951+00:00, run_duration=1.121677, state=success, executor_state=success, try_number=1, max_tries=0, job_id=60, pool=default pool, queue=default, priority_weight=3, operator=BashOperator, queued_dttm=2025-03-28 07:02:29.197941+00:00, queued_by_job_id=1, pid=1852
scheduler-1 | [2025-03-28T07:02:31.210+0000] {dagbag.py:538} INFO - Filling up the DagBag from /opt/***/dags/download_rocket_local.py
scheduler-1 | [2025-03-28T07:02:31.256+0000] {task_command.py:423} INFO - Running <TaskInstance: download_rocket_local.get_pictures manual__2025-03-28T07:02:28.916508+00:00 [queued]> on host 1685e4639115
webserver-1 | 172.24.0.1 - - [28/Mar/2025:07:02:32 +0000] "GET /object/graph_data?dag_id=download_rocket_local HTTP/1.1" 200 264 "http://localhost:8080/dags/download_rocket_local/grid?tab=graph" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36"
webserver-1 | 172.24.0.1 - - [28/Mar/2025:07:02:33 +0000] "GET /object/grid_data?dag_id=download_rocket_local&num_runs=25 HTTP/1.1" 200 19922 "http://localhost:8080/dags/download_rocket_local/grid?tab=graph" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36"
webserver-1 | 172.24.0.1 - - [28/Mar/2025:07:02:35 +0000] "POST /clear HTTP/1.1" 200 282 "http://localhost:8080/dags/download_rocket_local/grid?tab=graph&dag_run_id=manual__2025-03-28T07%3A02%3A28.916508%2B00%3A00%task_id=download_launches" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36"
webserver-1 | 172.24.0.1 - - [28/Mar/2025:07:02:35 +0000] "GET /confirm?dag_id=download_rocket_local&dag_run_id=manual__2025-03-28T07%3A02%3A28.916508%2B00%3A00%past=false&future=false&upstream=false&downstream=false&state=success&task_id=download_launches HTTP/1.1" 200 2 "http://localhost:8080/dags/download_rocket_local/grid?tab=graph&dag_run_id=manual__2025-03-28T07%3A02%3A28.916508%2B00%3A00%task_id=download_launches" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36"
webserver-1 | 172.24.0.1 - - [28/Mar/2025:07:02:36 +0000] "GET /object/grid_data?dag_id=download_rocket_local&num_runs=25 HTTP/1.1" 200 19922 "http://localhost:8080/dags/download_rocket_local/grid?tab=graph&dag_run_id=manual__2025-03-28T07%3A02%3A28.916508%2B00%3A00%task_id=download_launches" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36"
webserver-1 | 172.24.0.1 - - [28/Mar/2025:07:02:39 +0000] "GET /object/grid_data?dag_id=download_rocket_local&num_runs=25 HTTP/1.1" 200 19922 "http://localhost:8080/dags/download_rocket_local/grid?tab=graph&dag_run_id=manual__2025-03-28T07%3A02%3A28.916508%2B00%3A00%task_id=download_launches" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36"
scheduler-1 | [2025-03-28T07:02:41.256+0000] {scheduler_job_runner.py:424} INFO - 1 tasks up for execution:
scheduler-1 | <TaskInstance: download_rocket_local.notify manual__2025-03-28T07:02:28.916508+00:00 [scheduled]>
scheduler-1 | [2025-03-28T07:02:41.256+0000] {scheduler_job_runner.py:487} INFO - DAG download_rocket_local has 0/16 running and queued tasks
scheduler-1 | [2025-03-28T07:02:41.257+0000] {scheduler_job_runner.py:603} INFO - Setting the following tasks to queued state:
scheduler-1 | <TaskInstance: download_rocket_local.notify manual__2025-03-28T07:02:28.916508+00:00 [scheduled]>
scheduler-1 | [2025-03-28T07:02:41.258+0000] {taskinstance.py:2260} WARNING - cannot record scheduled duration for task notify because previous state change time has not been saved
scheduler-1 | [2025-03-28T07:02:41.258+0000] {scheduler_job_runner.py:646} INFO - Sending TaskInstanceKey(dag_id='download_rocket_local', task_id='notify', run_id='manual__2025-03-28T07:02:28.916508+00:00')
```

Рисунок 6. Логи

Затем нужно создать исполняемый файл `export.sh`, который автоматизирует выгрузку данных из контейнера в основную ОС данных, полученные в результате работы DAG в Apache Airflow (рисунок 7).



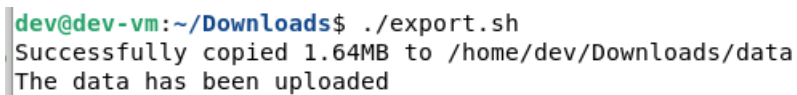
The screenshot shows a terminal window titled "dev@dev-vm: ~/Downloads". Inside, the GNU nano 6.2 editor is open, displaying the script "export.sh". The script contains the following code:

```
#!/bin/bash
CONTAINER_NAME="business_case_rocket_25-scheduler-1"
CONTAINER_PATH="/opt/airflow/data/images"
HOST_PATH="/home/dev/Downloads/data"
mkdir -p "$HOST_PATH"
docker cp "$CONTAINER_NAME:$CONTAINER_PATH/." "$HOST_PATH"
echo "The data has been uploaded"
```

At the bottom of the window, there is a status bar with various keyboard shortcuts: ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^C Location, ^X Exit, ^R Read File, ^\ Replace, ^U Paste, ^J Justify, and ^_ Go To Line. A message "[Read 7 lines]" is also visible.

Рисунок 7. Исполняемый файл export.sh

Как можно увидеть на рисунке 8, все успешно скопировалось



The screenshot shows the terminal output after running the script:

```
dev@dev-vm:~/Downloads$ ./export.sh
Successfully copied 1.64MB to /home/dev/Downloads/data
The data has been uploaded
```

Рисунок 8. Файл sh успешно отработал

На рисунке 9 показан каталог data, где файлы перенеслись.

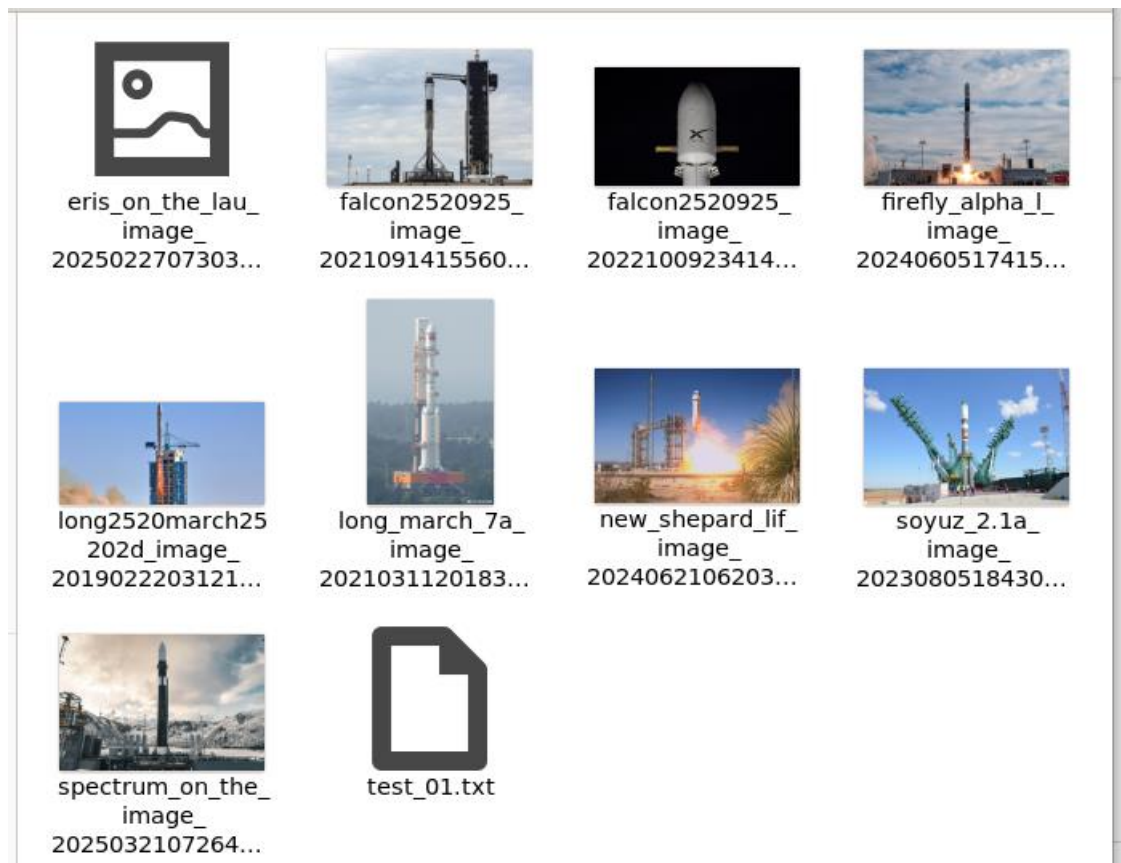


Рисунок 9. Файлы перенеслись

Выполнение индивидуального задания

Вариант 13. Создать отчет по списку ракет и их изображений, используя данные JSON. Настроить загрузку изображений с альтернативных источников. Проанализировать, какие типы исключений нужно обрабатывать для успешной загрузки.

Созданный даг продемонстрирован на рисунке 10.


```

10 dag = DAG(
11     description="Download rocket pictures from primary and alternative sources with exception handling and reporting.",
12     start_date=airflow.utils.dates.days_ago(14),
13     schedule_interval="@daily",
14     catchup=False,
15 )
16
17
18 download_launches = BashOperator(
19     task_id="download_launches",
20     bash_command="""
21     curl -o /opt/airflow/data/launches.json -L 'https://ll.thespacedevs.com/2.0.0/launch/upcoming'
22     """,
23     dag=dag,
24 )
25
26 def _get_pictures():
27     images_dir = "/opt/airflow/data/images"
28     pathlib.Path(images_dir).mkdir(parents=True, exist_ok=True)
29
30     report = {"successful_downloads": [], "failed_downloads": []}
31
32     with open("/opt/airflow/data/launches.json", "r") as f:
33         launches = json.load(f)
34
35     for launch in launches["results"]:
36         image_url = launch.get("image")
37         alternative_image_url = launch.get("alternative_image")
38         name = launch.get("name", "Unknown Launch")
39
40         def download_image(url, filename):
41             try:
42                 response = requests.get(url, stream=True, timeout=10)
43                 response.raise_for_status()
44                 target_file = f"{images_dir}/{filename}"

```

Рисунок 10. Код созданного дага

На рисунке 11 показано, что созданный даг отработал

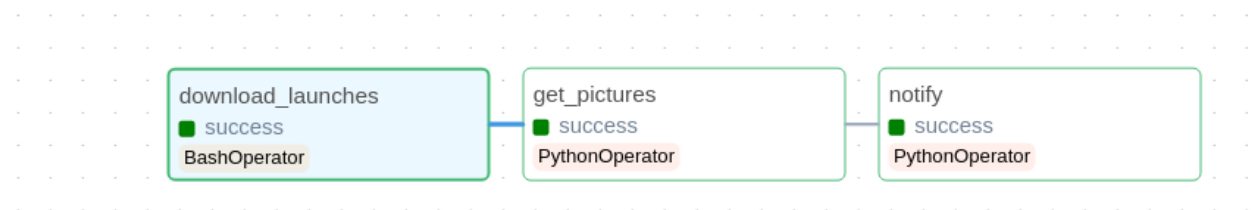


Рисунок 11. Даг отработал

На рисунке 12 продемонстрирована диаграмма Ганта

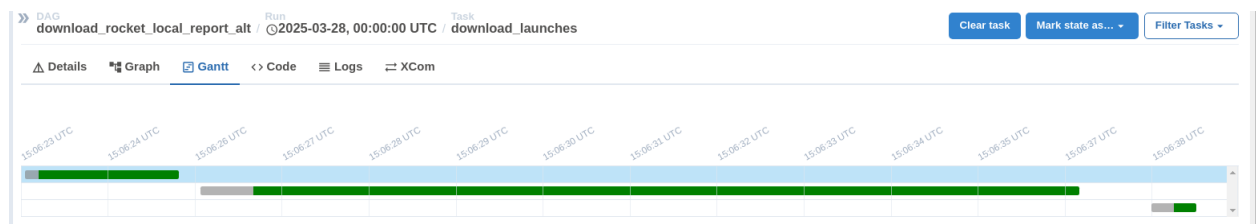


Рисунок 12. Диаграмма Ганта

На рисунке 13 показаны логи отчета по списку ракет и их изображений, используя данные JSON **(1 задание)**.

```

[2025-03-28, 15:06:38 UTC] {logging_mixin.py:188} INFO - Download Report: Successful downloads: 10, Failed downloads: 0. Check /opt/***/data/download_report.json for details.

```

Рисунок 13. Логи отчета

На рисунке 14 показан сам файл json

```

data > {} download_report.json > ...
2      "successful_downloads": [
14          },
15          {
16              "name": "Falcon 9 Block 5 | Starlink Group 6-80",
17              "url": "https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20221009234147.png"
18          },
19          {
20              "name": "Eris-1 | Maiden Flight",
21              "url": "https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/eris_on_the_lau_image_20250227073032.jpg"
22          },
23          {
24              "name": "Falcon 9 Block 5 | Starlink Group 11-13",
25              "url": "https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20221009234147.png"
26          },
27          {
28              "name": "Falcon 9 Block 5 | Fram2",
29              "url": "https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20210914155606.jpeg"
30          },
31          {
32              "name": "Long March 2D | Unknown Payload",
33              "url": "https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/long2520march25202d_image_20190222031211.jpeg"
34          },
35          {
36              "name": "Long March 6 | Unknown Payload",
37              "url": "https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/long_march_6_image_20210709074933.jpg"
38          },
39          {
40              "name": "Soyuz 2.1a | Soyuz MS-27",
41              "url": "https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/soyuz_2.1a_image_20230805184309.jpg"
42          }
43      ],
44      "failed_downloads": []
45  }

```

Рисунок 14. Json файл

На рисунке 15 показана загрузка изображений с альтернативных источников (**2 задание**) при условии, что какое-то изображение не будет загружено.

```

[2025-03-28, 15:06:26 UTC] (standard_task_runner.py:87) INFO - Running: ['***', 'tasks', 'run', 'download_rocket_local_report_alt', 'get_pictures', 'scheduled_2025-03-27T00:00:00:00', '...', '115', '...ra
[2025-03-28, 15:06:26 UTC] (standard_task_runner.py:88) INFO - Job 115: Subtask get_pictures
[2025-03-28, 15:06:27 UTC] (task_command.py:423) INFO - Running <taskinstance: download_rocket_local_report_alt.get_pictures scheduled_2025-03-27T00:00:00:00 [running]> on host ebffd50c031b
[2025-03-28, 15:06:27 UTC] (taskinstance.py:2480) INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='***' AIRFLOW_CTX_DAG_ID='download_rocket_local_report_alt' AIRFLOW_CTX_TASK_ID='get_pictures' AIRFLOW_CTX_EXECUT
[2025-03-28, 15:06:28 UTC] (logging_mixin.py:188) INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/spectrum_on_the_image_20250321072643.jpeg to /opt/***/data/images/spectrum_on
[2025-03-28, 15:06:29 UTC] (logging_mixin.py:188) INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/long_march_7a_image_20210311201838.jpg to /opt/***/data/images/long_march_7a_
[2025-03-28, 15:06:30 UTC] (logging_mixin.py:188) INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/firefly_alpha_l_image_20240605174156.jpeg to /opt/***/data/images/firefly_alp
[2025-03-28, 15:06:31 UTC] (logging_mixin.py:188) INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20221009234147.png to /opt/***/data/images/falcon2520925_
[2025-03-28, 15:06:32 UTC] (logging_mixin.py:188) INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/eris_on_the_lau_image_20250227073032.jpg to /opt/***/data/images/eris_on_the_
[2025-03-28, 15:06:33 UTC] (logging_mixin.py:188) INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20221009234147.png to /opt/***/data/images/falcon2520925_
[2025-03-28, 15:06:34 UTC] (logging_mixin.py:188) INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20210914155606.jpeg to /opt/***/data/images/falcon2520925_
[2025-03-28, 15:06:35 UTC] (logging_mixin.py:188) INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/long2520march25202d_image_20190222031211.jpeg to /opt/***/data/images/long252
[2025-03-28, 15:06:36 UTC] (logging_mixin.py:188) INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/long_march_6_image_20210709074933.jpg to /opt/***/data/images/long_march_6_in
[2025-03-28, 15:06:37 UTC] (logging_mixin.py:188) INFO - Downloaded https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/soyuz_2.1a_image_20230805184309.jpg to /opt/***/data/images/soyuz_2.1a_image_

```

Рисунок 15. Загрузка изображений с альтернативных источников

Функция работает следующим образом:

- Если функция `download_image` вернула `False` (т.е., основное изображение не удалось загрузить), и если при этом в JSON-файле указан `alternative_image_url`, то функция `download_image` вызывается повторно, но уже для альтернативного URL.
- Если загрузка альтернативного изображения прошла успешно, информация добавляется в отчет об успешных загрузках.

- Если загрузка альтернативного изображения также не удалась, выводится сообщение о том, что не удалось загрузить ни основное, ни альтернативное изображение.

Для обеспечения успешной загрузки изображений из различных источников **(3 задание)** необходимо обрабатывать следующие типы исключений:

1. requests.exceptions.RequestException:

- Описание: Этот класс является базовым для большинства исключений, возникающих при использовании библиотеки requests. Он включает в себя исключения, связанные с проблемами сети, HTTP-ошибками и другими общими проблемами при выполнении HTTP-запросов.

- Почему нужно обрабатывать: Перехват этого исключения позволяет обрабатывать широкий спектр проблем, связанных с отправкой и получением HTTP-запросов.

2. OSError (или IOError в старых версиях Python):

- Описание: это исключение возникает при ошибках, связанных с операционной системой, таких как проблемы с файловой системой, недостаток прав доступа и т.д.

- Почему нужно обрабатывать: при сохранении загруженных изображений в файлы могут возникать проблемы с доступом к файловой системе или нехваткой места на диске.

3. json.JSONDecodeError:

- Описание: это исключение возникает, если возникают проблемы с декодированием JSON-данных из файла или URL.

- Почему нужно обрабатывать: если формат JSON недействителен.

4. Exception:

- Описание: это общий класс исключений, который перехватывает все остальные, более специфичные исключения.

- Почему нужно обрабатывать: В блоке except имеет смысл обрабатывать исключение, чтобы перехватывать любые другие непредвиденные исключения.

На рисунке 16 продемонстрирована загрузка json файла в Google Colab

```
[1] import json
with open('download_report.json') as json_file:
    data = json.load(json_file)
data
```

```
{'successful_downloads': [{'name': 'Spectrum | Maiden Flight',
'url': 'https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/spectrum_on_the_image_20250321072643.jpeg'},
{'name': 'Long March 7A | Unknown Payload',
'url': 'https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/long_march_7a_image_20210311201838.jpg'},
{'name': 'Firefly Alpha | FLTA006 (Message in a Booster)',
'url': 'https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/firefly_alpha_1_image_20240605174156.jpeg'},
{'name': 'Falcon 9 Block 5 | Starlink Group 6-80',
'url': 'https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20221009234147.png'},
{'name': 'Eris-1 | Maiden Flight',
'url': 'https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/eris_on_the_lau_image_20250227073032.jpg'},
{'name': 'Falcon 9 Block 5 | Starlink Group 11-13',
'url': 'https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20221009234147.png'},
{'name': 'Falcon 9 Block 5 | Fram2',
'url': 'https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/falcon2520925_image_20210914155606.jpeg'},
{'name': 'Long March 2D | Unknown Payload',
'url': 'https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/long2520march25202d_image_20190222031211.jpeg'},
{'name': 'Long March 6 | Unknown Payload',
'url': 'https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/long_march_6_image_20210709074933.jpg'},
{'name': 'Soyuz 2.1a | Soyuz MS-27',
'url': 'https://thespacedevs-prod.nyc3.digitaloceanspaces.com/media/images/soyuz_2.1a_image_20230805184309.jpg'}],
'failed_downloads': []}
```

```
[13] failed_downloads = pd.DataFrame(data['failed_downloads'])
failed_downloads
```

```
[9] successful_df = pd.DataFrame(data['successful_downloads'])
successful_df
```

	name	year
0	Spectrum Maiden Flight	2025
1	Long March 7A Unknown Payload	2021
2	Firefly Alpha FLTA006 (Message in a Booster)	2024
3	Falcon 9 Block 5 Starlink Group 6-80	2022
4	Eris-1 Maiden Flight	2025
5	Falcon 9 Block 5 Starlink Group 11-13	2022
6	Falcon 9 Block 5 Fram2	2021
7	Long March 2D Unknown Payload	2019
8	Long March 6 Unknown Payload	2021

Рисунок 16. Загрузка json файла

На рисунке 17 показано график распределения успешных запусков по годам. Как можно заметить, среди всех успешных запусков больше всего было в 2021 году.

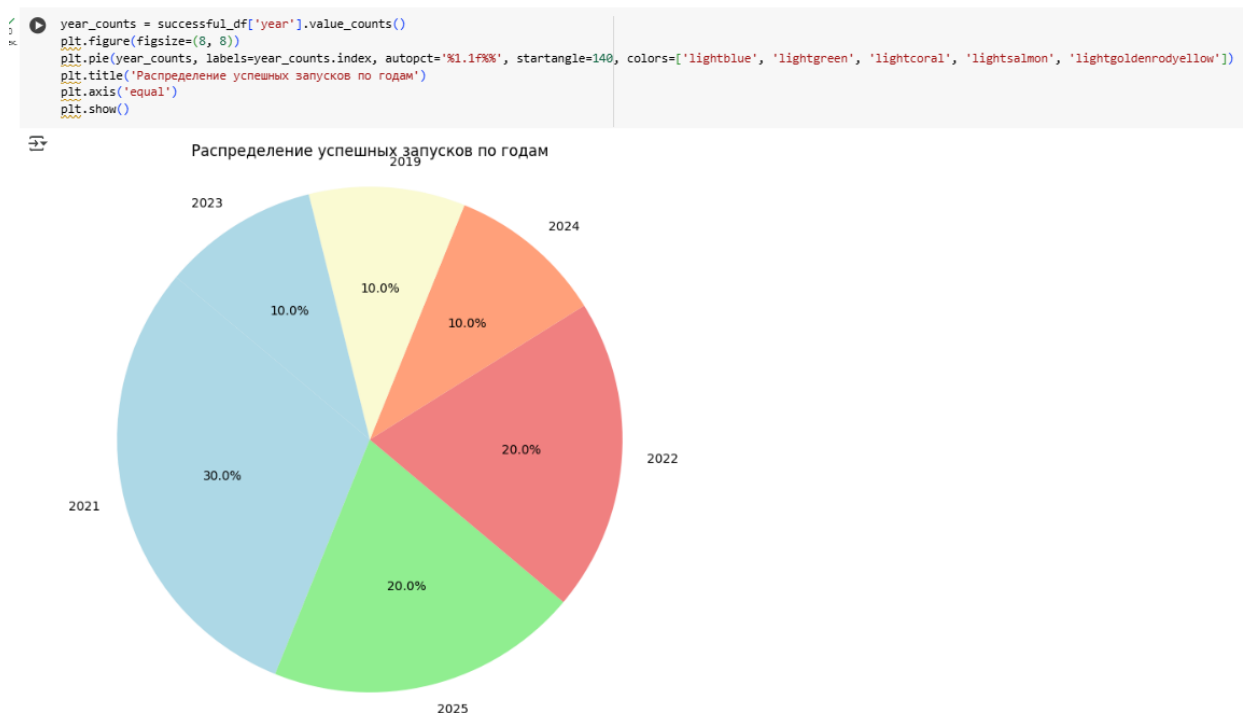


Рисунок 17. График распределения успешных запусков по годам

Выводы:

1. Автоматизация загрузки и обработки изображений (и других типов данных) позволяет снизить затраты на ручной труд. Вместо того, чтобы сотрудники вручную загружали и обрабатывали каждое изображение, автоматизированная система может выполнять эту задачу быстрее и с меньшими затратами.
2. Эта автоматизированная система работает круглосуточно, что позволяет значительно повысить производительность и сократить время выполнения задач.
3. Автоматизированная система настроена на выполнение определенных правил и проверок, что позволяет снизить количество ошибок, связанных с человеческим фактором.