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»

Практическая работа № 5

Тема:

«Airflow DAG»**.**

Выполнила: Алексейчук А.А., АДЭУ-201
Преподаватель: Босенко Т.М.

Москва
2024

Оглавление

Постановка задачи.

Самостоятельная работа

5.1.1. Развернуть ВМ ubuntu_mgpu.ova в VirtualBox.

5.1.2. Клонировать на ПК задание Бизнес-кейс «Rocket» в домашний каталог ВМ.

git clone https://github.com/BosenkoTM/workshop-on-ETL.git

5.1.3. Запустить контейнер с кейсом, изучить основные элементы DAG в Apache Airflow.

Создать DAG согласно алгоритму, который предоставит преподаватель.

Изучить логи, выполненного DAG. Скачать логи из контейнера на основную ОС, используя команду:

docker cp <container_hash>:/path/to/zip/file.zip /path/on/host/new_name.zip

Выгрузить полученный результат работы DAG в основной каталог ОС, используя команду:

docker cp -r <containerId>:/path/to/directory /path/on/host

5.1.4. Создать исполняемый файл с расширением .sh, который автоматизирует выгрузку данных из контейнера в основную ОС данных, полученные в результате работы DAG в Apache Airflow.

5.1.5. 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.

5.1.6. 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.

5.1.7. Построить диаграмму Ганта работы DAG в Apache Airflow.

Проверка ответа URL-адреса с помощью Curl из командной строки (Рисунок 1). Значения в фигурных скобках относятся к одному запуску ракеты. Информация об идентификаторе ракеты, времени начала и окончания окна запуска ракеты, URL-адресе изображения запускаемой ракеты.

Рисунок 1- Проверка ответа URL

Исходный код всех dags

download_rocket_launches.py (Рисунок *2*).

download_rocket_launches.py 4

C: > Users > Nastya > Downloads > download_rocket_launches.py > ...

```python
import json
import pathlib

import airflow.utils.dates
import requests
import requests.exceptions as requests_exceptions
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.python import PythonOperator

dag = DAG(
    dag_id="download_rocket_launches",
    description="Download rocket pictures of recently launched rockets.",
    start_date=airflow.utils.dates.days_ago(14),
    schedule_interval="@daily",
)

download_launches = BashOperator(
    task_id="download_launches",
    bash_command="curl -o /tmp/launches.json -L 'https://ll.thespacedevs.com/2.0.0/launch/upcoming'",  # noqa: E501
    dag=dag,
)


def _get_pictures():
    # Ensure directory exists
    pathlib.Path("/tmp/images").mkdir(parents=True, exist_ok=True)

    # Download all pictures in launches.json
    with open("/tmp/launches.json") as f:
        launches = json.load(f)
        image_urls = [launch["image"] for launch in launches["results"]]
        for image_url in image_urls:
            try:
                response = requests.get(image_url)
                image_filename = image_url.split("/")[-1]
                target_file = f"/tmp/images/{image_filename}"
                with open(target_file, "wb") as f:
                    f.write(response.content)
                print(f"Downloaded {image_url} to {target_file}")
            except requests_exceptions.MissingSchema:
                print(f"{image_url} appears to be an invalid URL.")
            except requests_exceptions.ConnectionError:
                print(f"Could not connect to {image_url}.")


get_pictures = PythonOperator(
    task_id="get_pictures", python_callable=_get_pictures, dag=dag
)
```

```python

notify = BashOperator(
    task_id="notify",
    bash_command='echo "There are now $(ls /tmp/images/ | wc -l) images."',
    dag=dag,
)

download_launches >> get_pictures >> notify

```

Рисунок 2 - Исходный код

Граф DAG в Apache Airflow

После запуска образа на странице localhost:8080 запускаем DAG(Рисунок *3*).

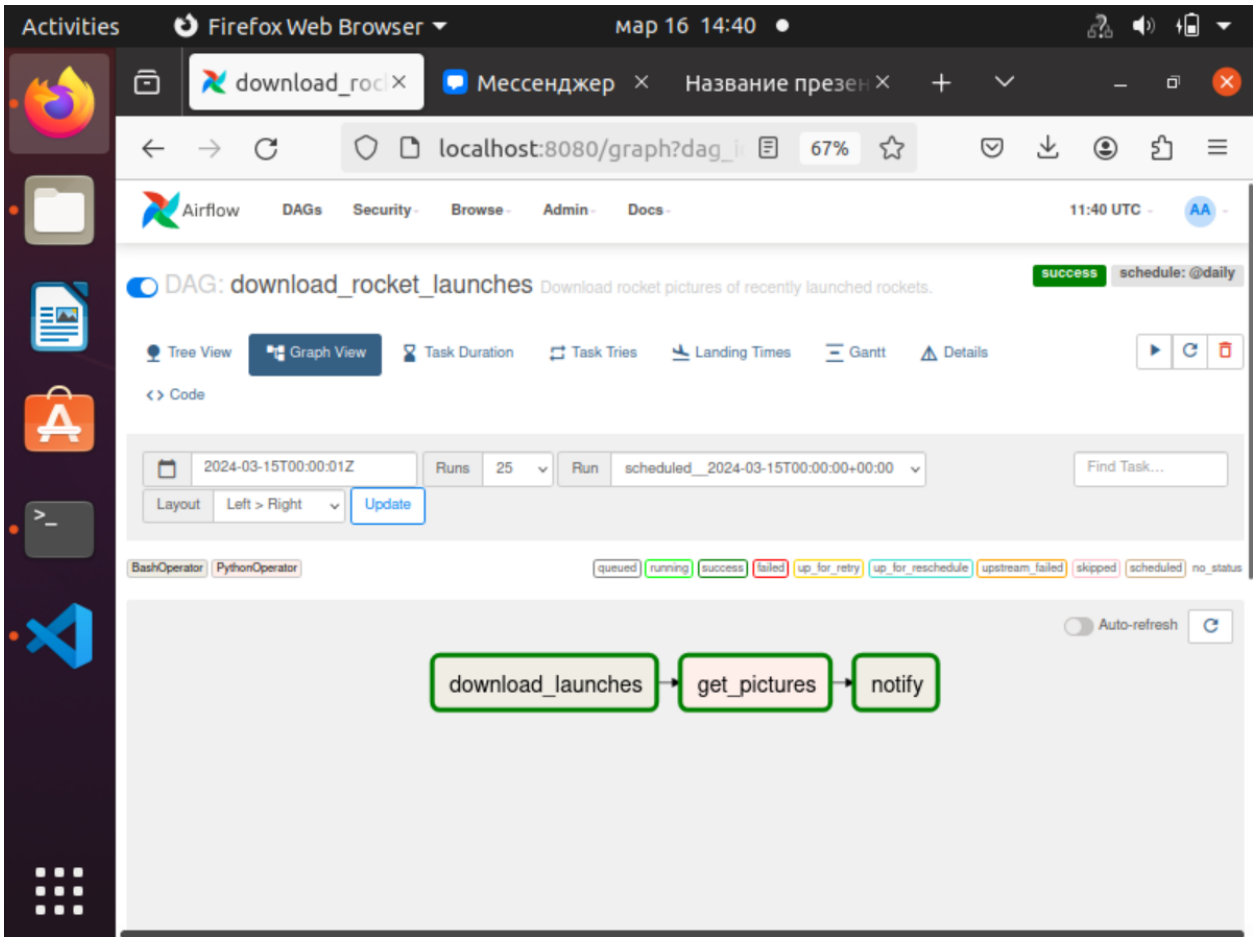

Рисунок 3 - Граф DAG

Верхнеуровневая архитектура задания Бизнес-кейса «Rocket», выполненная в draw.io

https://drive.google.com/file/d/12RS59WPzPVeUAYMhRs-eN78pJXu8fEsH/view?usp=sharing (Рисунок *4*).

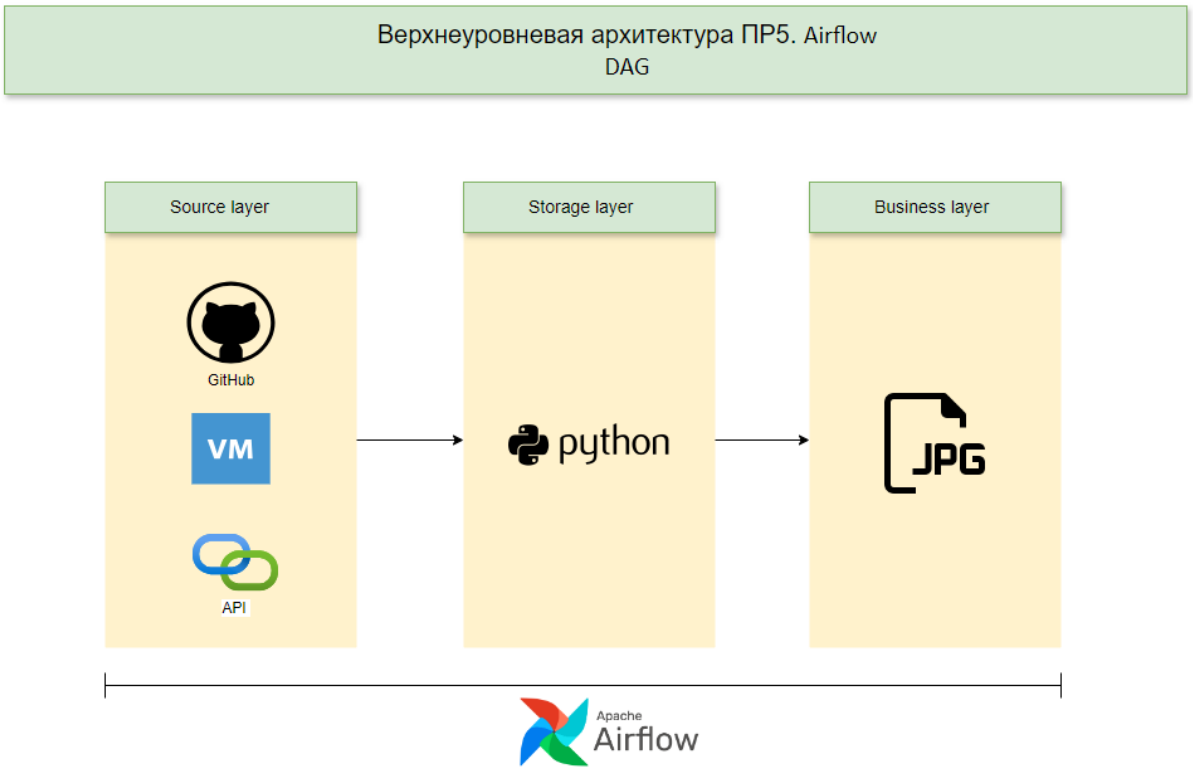

Рисунок 4 - Верхнеуровневая архитектура

Архитектура DAG Бизнес-кейса «Rocket» , выполненная в draw.io

https://drive.google.com/file/d/1hkI594CqEVUDXKf9RxkcCDh2Mx2Hkz9s/view?usp=sharing

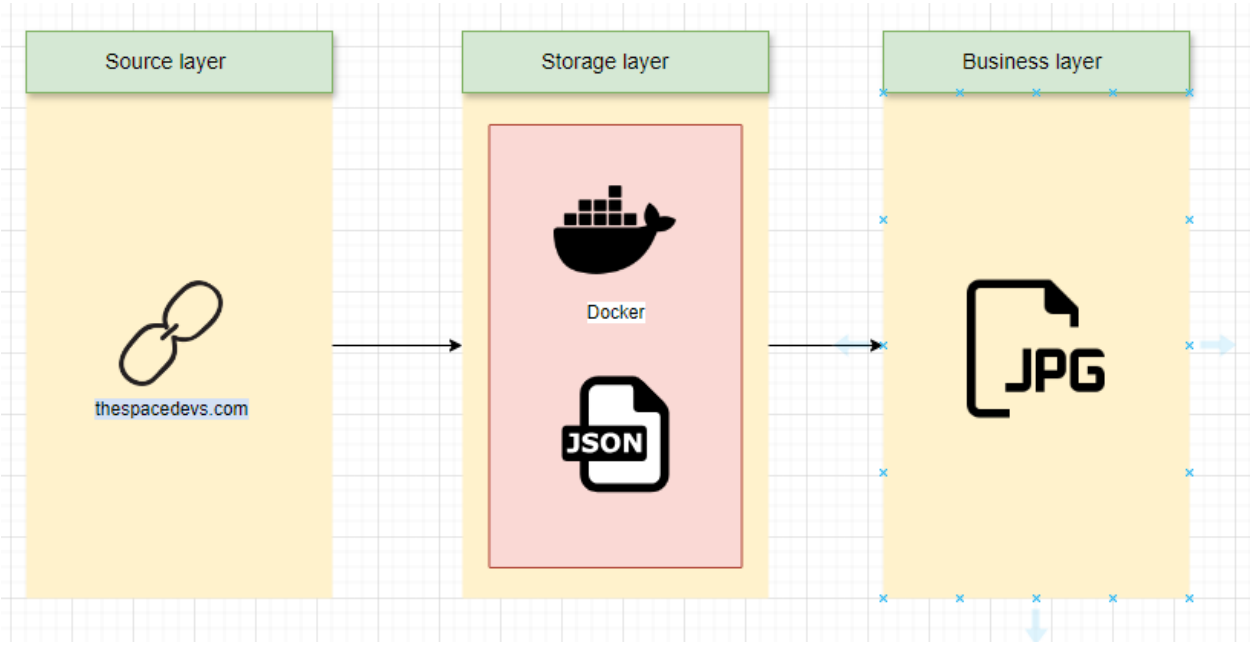

Рисунок 5 - Архитектура DAG

Скрин лог-файла результаов работы dags в Apache Airflow

Проверка файлов логов.

Проверка файла лога download_launches_logs (Рисунок *6*).

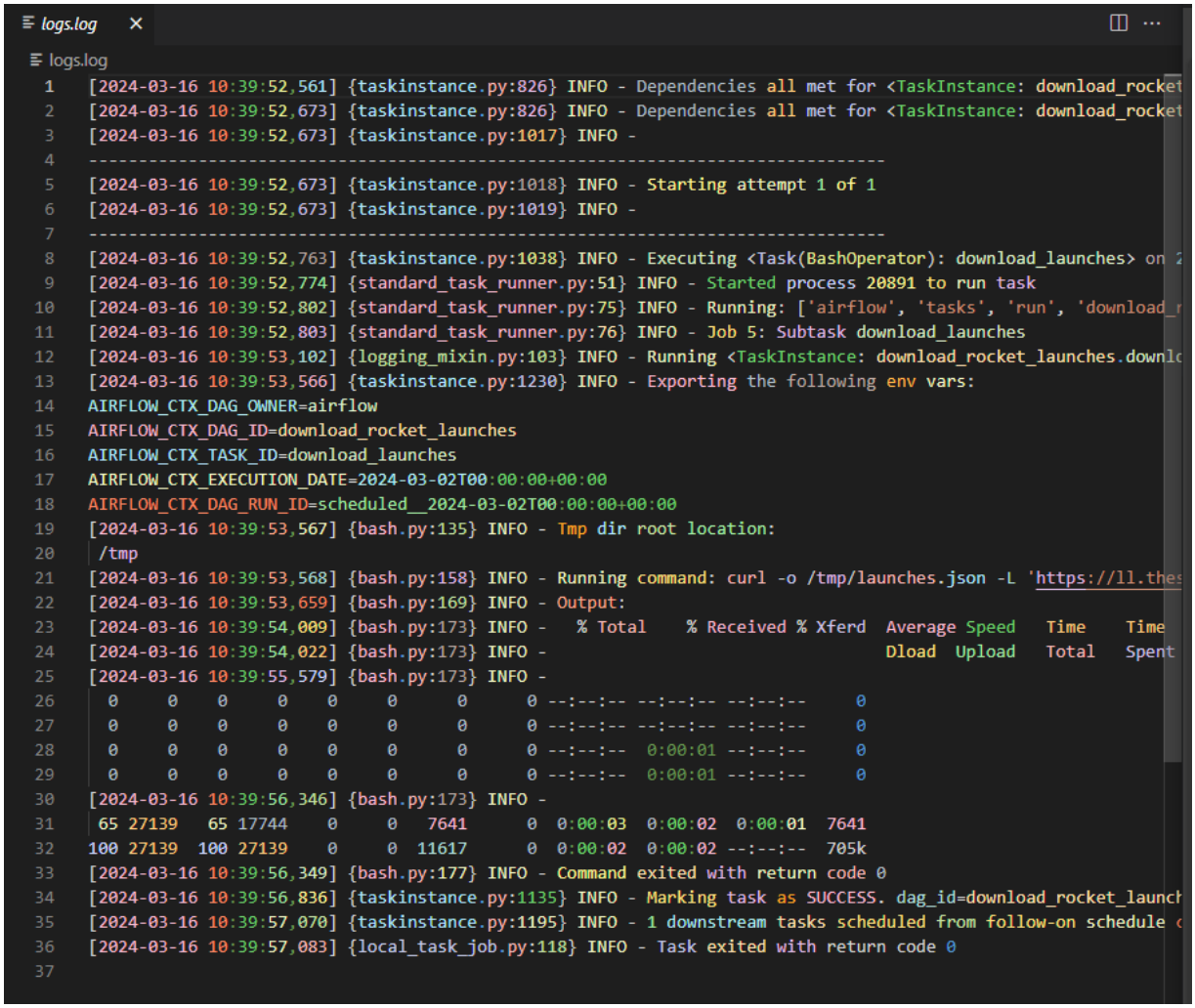

Рисунок 6 - download_launches_logs

Проверка файла лога get_pictures_logs (Рисунок *7*).

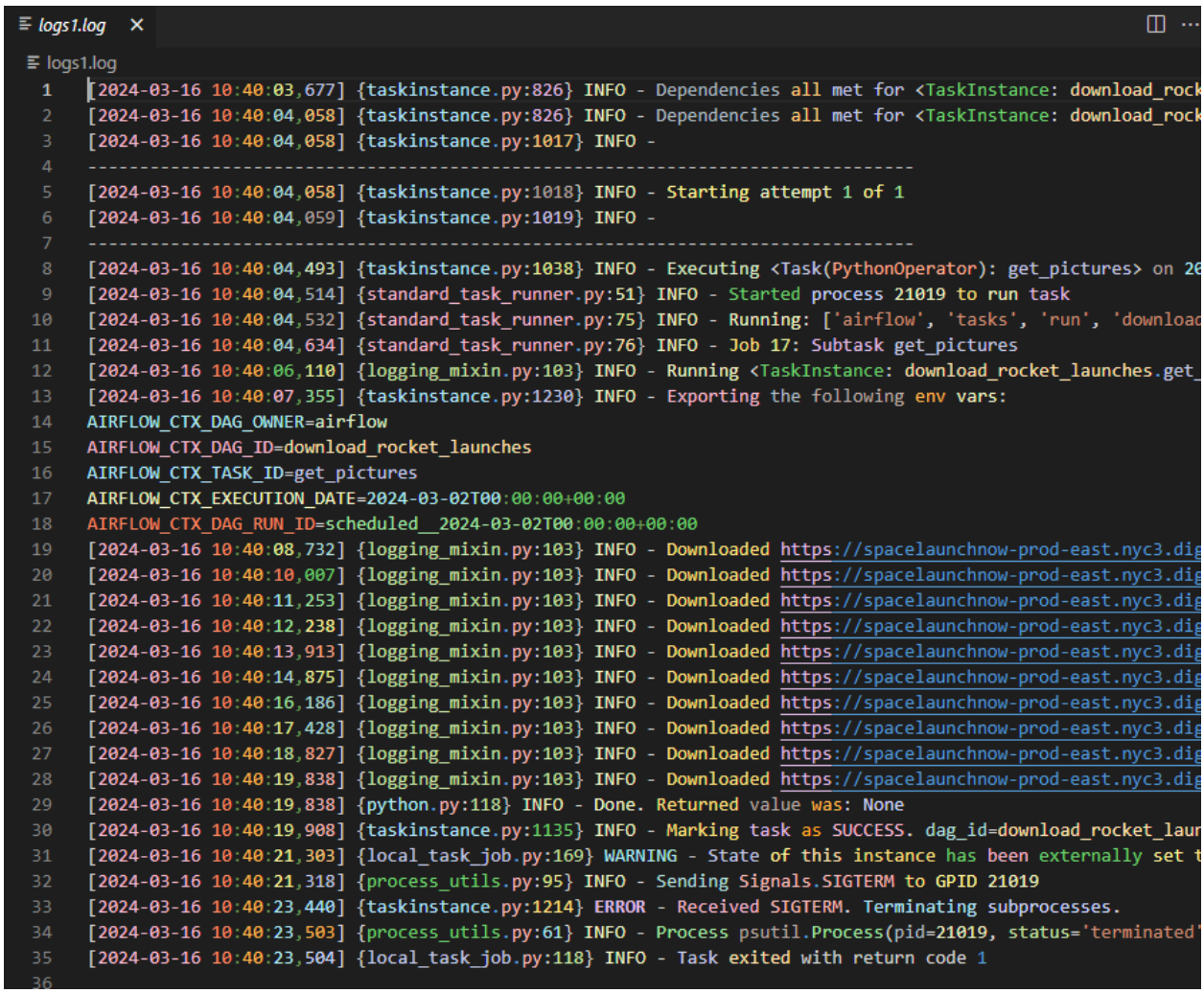

Рисунок 7 - get_pictures_logs

Проверка файла лога notify_logs (Рисунок *8*).

logs2.log

```
[2024-03-16 10:40:29,290] {taskinstance.py:826} INFO - Dependencies all met for <TaskInstance: download_rocket
[2024-03-16 10:40:29,678] {taskinstance.py:826} INFO - Dependencies all met for <TaskInstance: download_rocket
[2024-03-16 10:40:29,678] {taskinstance.py:1017} INFO -
--------------------------------------------------------------------------------
[2024-03-16 10:40:29,678] {taskinstance.py:1018} INFO - Starting attempt 1 of 1
[2024-03-16 10:40:29,679] {taskinstance.py:1019} INFO -
--------------------------------------------------------------------------------
[2024-03-16 10:40:30,193] {taskinstance.py:1038} INFO - Executing <Task(BashOperator): notify> on 2024-03-02T0
[2024-03-16 10:40:30,204] {standard_task_runner.py:51} INFO - Started process 21188 to run task
[2024-03-16 10:40:30,371] {standard_task_runner.py:75} INFO - Running: ['airflow', 'tasks', 'run', 'download_r
[2024-03-16 10:40:30,372] {standard_task_runner.py:76} INFO - Job 31: Subtask notify
[2024-03-16 10:40:31,579] {logging_mixin.py:103} INFO - Running <TaskInstance: download_rocket_launches.notify
[2024-03-16 10:40:32,302] {taskinstance.py:1230} INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_OWNER=airflow
AIRFLOW_CTX_DAG_ID=download_rocket_launches
AIRFLOW_CTX_TASK_ID=notify
AIRFLOW_CTX_EXECUTION_DATE=2024-03-02T00:00:00+00:00
AIRFLOW_CTX_DAG_RUN_ID=scheduled__2024-03-02T00:00:00+00:00
[2024-03-16 10:40:32,362] {bash.py:135} INFO - Tmp dir root location:
 /tmp
[2024-03-16 10:40:32,362] {bash.py:158} INFO - Running command: echo "There are now $(ls /tmp/images/ | wc -l)
[2024-03-16 10:40:32,865] {bash.py:169} INFO - Output:
[2024-03-16 10:40:32,936] {bash.py:173} INFO - There are now 8 images.
[2024-03-16 10:40:32,937] {bash.py:177} INFO - Command exited with return code 0
[2024-03-16 10:40:33,390] {taskinstance.py:1135} INFO - Marking task as SUCCESS. dag_id=download_rocket_launch
[2024-03-16 10:40:35,577] {local_task_job.py:169} WARNING - State of this instance has been externally set to
[2024-03-16 10:40:35,582] {process_utils.py:95} INFO - Sending Signals.SIGTERM to GPID 21188
[2024-03-16 10:40:35,665] {taskinstance.py:1214} ERROR - Received SIGTERM. Terminating subprocesses.
[2024-03-16 10:40:35,665] {bash.py:185} INFO - Sending SIGTERM signal to bash process group
[2024-03-16 10:40:35,928] {process_utils.py:61} INFO - Process psutil.Process(pid=21188, status='terminated',
[2024-03-16 10:40:35,930] {local_task_job.py:118} INFO - Task exited with return code 1
```

Рисунок 8 - notify_logs

Выгрузка папки с изображениями из Apache Airflow (Рисунок 9).

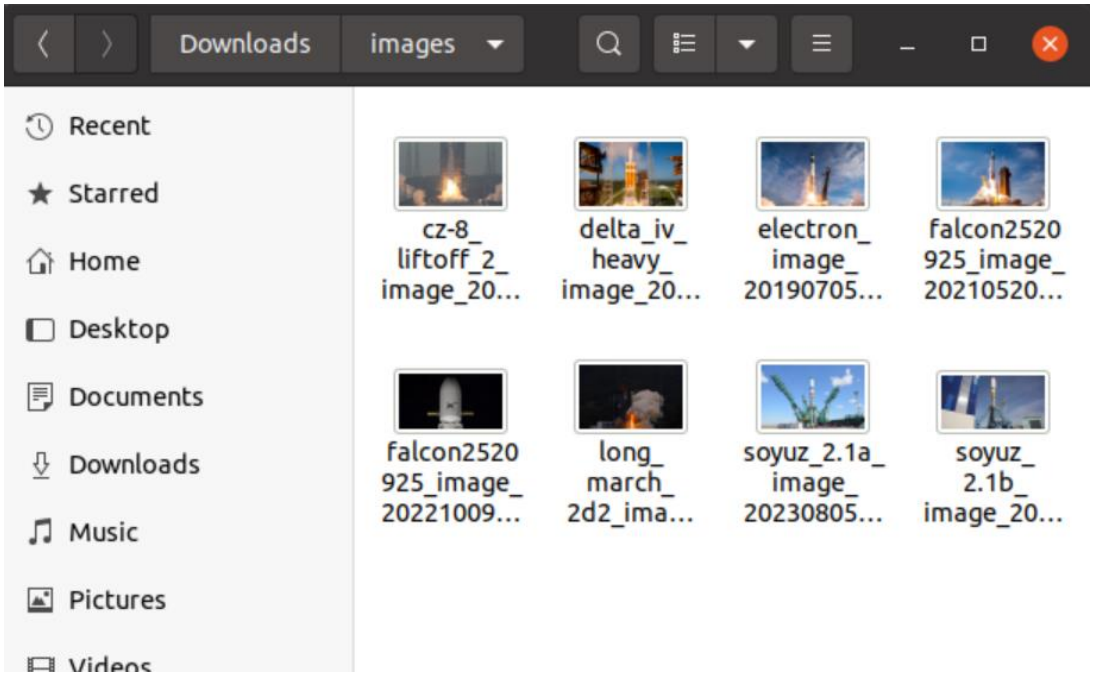

Рисунок 9 - Выгрузка картинок

Диаграмма Ганта DAG в Apache Airflow

Диаграмма Ганта DAG (Рисунок *10*).

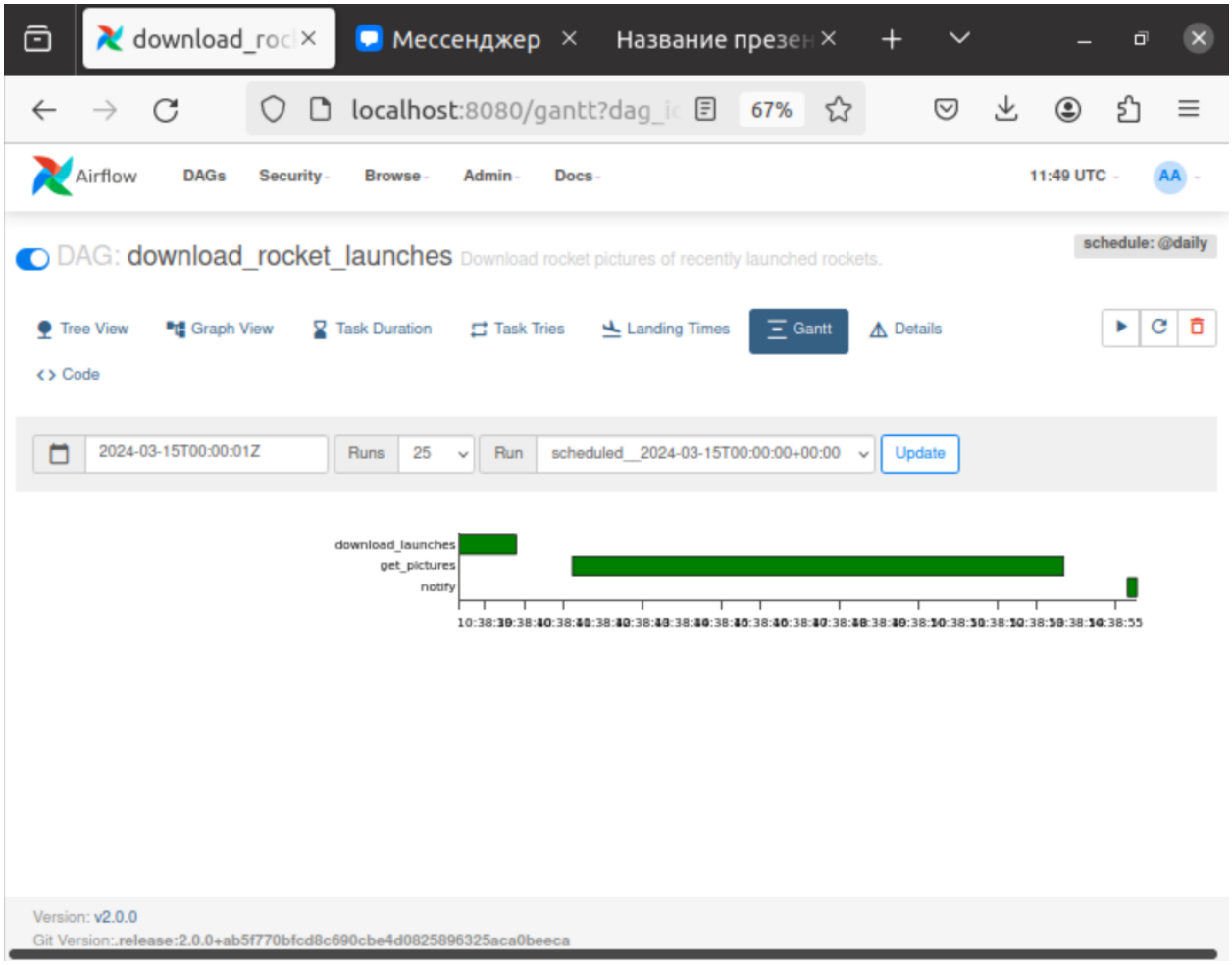

Рисунок 10 - Диаграмма Ганта

Исполняемый файл с расширением .sh, который автоматизирует выгрузку данных из контейнера в основную ОС данных, полученные в результате работы DAG в Apache Airflow:

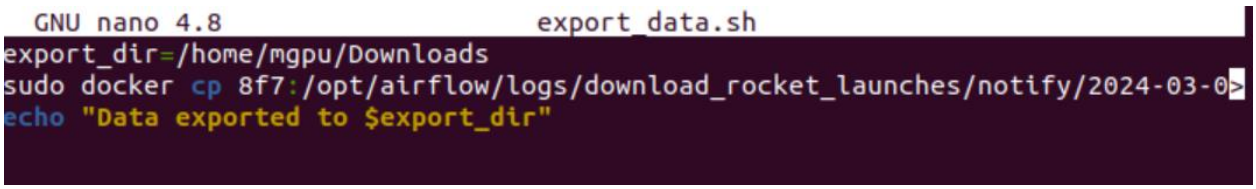

Рисунок 11 - Исполняемый файл .sh

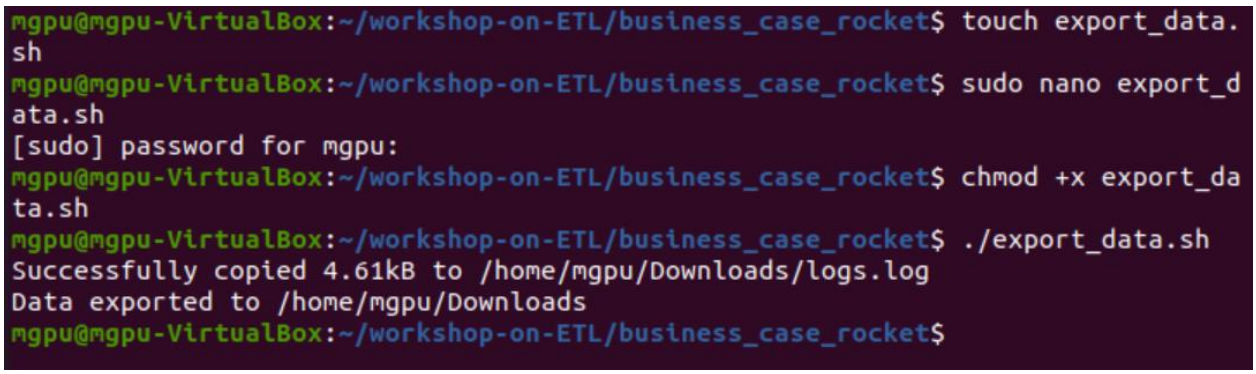

Рисунок 12 - Успешный запуск файла

Заключение

После выполнения работы было освоено:

1. Запуск контейнера и создание DAG в Apache Airflow:

Были изучены основные элементы и функционал Apache Airflow, создали и запущены Directed Acyclic Graph (DAG), изучили логи выполнения и скачали их для анализа.

2. Проектирование верхнеуровневой архитектуры аналитического решения:

Была спроектирована верхнеуровневая архитектура для аналитического решения задания «Rocket», включая слои источников данных, хранения данных и доступа к данным пользователей.

3. Проектирование архитектуры DAG:

Была разработана архитектура DAG для кейса «Rocket», учитывая слои источников, хранения и бизнес-логики доступа к данным.

4. Построение диаграммы Ганта работы DAG:

Была построена диаграмма Ганта для работы DAG в Apache Airflow, что позволяет визуализировать и планировать последовательность выполнения задач в графическом виде.

В итоге, выполнение приведенных шагов позволило освоить процессы создания и анализа DAG в Apache Airflow, а также разработки архитектуры данных и построения диаграмм работы DAG.