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**

**Бизнес-кейс «Rocket»**

Выполнила: Сергеева А. И., группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2025

**Цель работы:** разработать и настроить DAG в Apache Airflow для автоматизации процесса загрузки данных о предстоящих запусках ракет и скачивания изображений, связанных с этими запусками. Результаты работы DAG должны быть выгружены с помощью .sh файла, а также построена архитектура аналитического решения и выполнены индивидуальные задания.

**Задачи:**

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

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

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

- Спроектировать верхнеуровневую архитектуру аналитического решения задания Бизнес-кейса «Rocket» в draw.io.

- Спроектировать архитектуру DAG Бизнес-кейса «Rocket» в draw.io.

- Построить диаграмму Ганта работы DAG в Apache Airflow.

- Выполнить индивидуальные задания.

- Оценить экономическую эффективность нового DAG по сравнению со старым.

**Ход работы:**

download\_rocket\_local.py предназначен для загрузки данных о предстоящих запусках ракет, скачивания изображений, связанных с запуском и для подготовки уведомлений о количестве скачанных изображений. Для работы был запущен контейнер с Airflow. Для начала необходимо было ознакомиться с основными элементами интерфейса Apache Airflow. Для перехода в интерфейс необходимо было перейти на адресу <http://localhost:8080/> на рисунке 1. Пароль и логин admin.

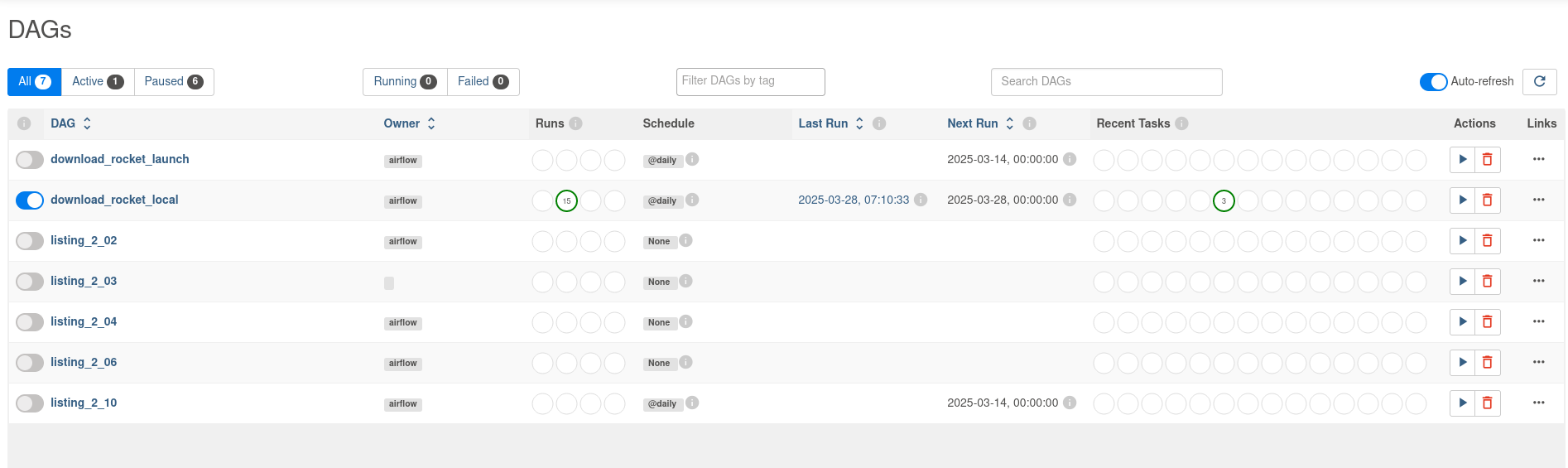


Рисунок 1 – Запуск Airflow

Как видно, DAG уже был запущен, с результатами его выполнения можно ознакомиться на рисунке 2.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 – Выполнение DAG download\_rocket\_local.py

А также есть диаграмма Ганта, по которой также можно отслеживать выполнение DAG и представлена на рисунке 3.

Изображение выглядит как текст, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 – Диаграмма Ганта DAG download\_rocket\_local.py

Далее был просмотрен лист логов на рисунке 4.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4 – Лист логов

Завершающим этапом необходимо было создать исполняемый файл .sh для автоматизации выгрузки изображений в другую папку на основной ОС. С его содержимым можно ознакомиться на рисунке 5. Создаётся директория, если её ещё нет, где будут храниться изображения, а сами изображения скачиваются из папки data, куда DAG выгружает изображения. Также проверяется проверка, чтобы выгружались только фотографии ракет.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5 - .sh файл для автоматической выгрузки данных

Далее необходимо было сделать созданный файл исполняемым и запустить скрипт на рисунке 6.

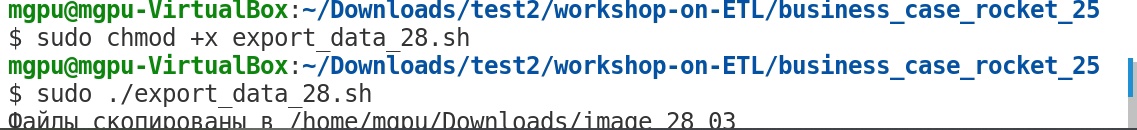


Рисунок 6 – Запуск скрипта по загрузке результатов DAG

Далее была выполнена проверка папки на выгруженные изображения на рисунке 7.

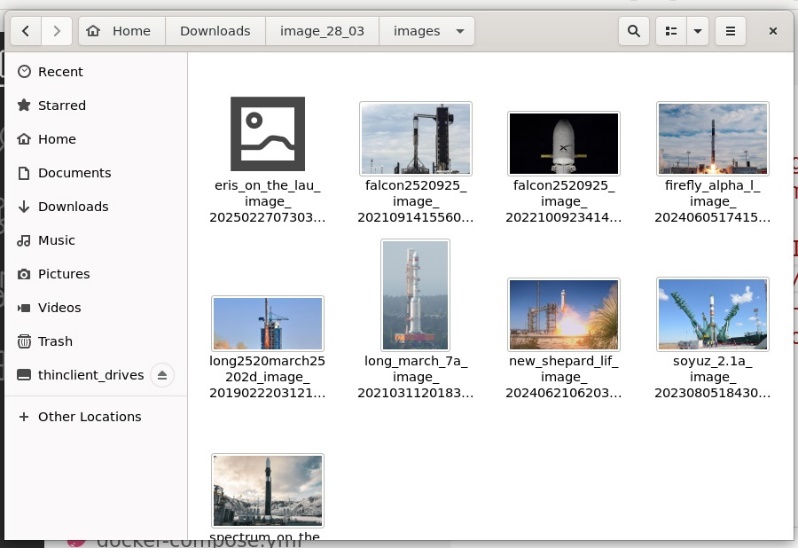


Рисунок 7 – Выгруженные изображения

Данные берутся из API, который предоставляет доступ к базе данных о космических запусках, ракетах, агентствах и событиях. Пользователи могут получать информацию о предстоящих и прошедших запусках, а также сведения о ракетах и космических агентствах, участвующих в миссиях. Дополнительно можно использовать погодные API, данные о стоимости запуска, чтобы делать более точные выводы о днях, когда запуски эффективны, их успешности/неуспешности. Данные хранятся в локальном хранилище, фотографии ракет берутся из интернета и также сохраняются в локальном хранилище. А данные о запусках можно, например, хранить в PostgreSQL, а изображения в MinIO, т. к. изображений много и MinIO позволяет хранить неограниченное количество объектов любого размера. Объём хранилища легко масштабируется добавлением новых серверов, а также к изображениям есть быстрый доступ. В бизнес-слое может быть визуализация для демонстрации карты запусков, уведомления о предстоящих запусках, а также просмотр моделей ракет через Model Viewer от Google с открытым исходным кодом, позволяет отображать 3D-контент в интернете, в данном случае ракеты. Это позволит проводить аналитику и на основе изображений проводить обучающий материал или разбор строения ракет. Также на основе данных о запусках можно выполнять обучение для прогнозирования успешности/неуспешности запусков. Предложенная архитектура бизнес-кейса представлена на рисунке 8.

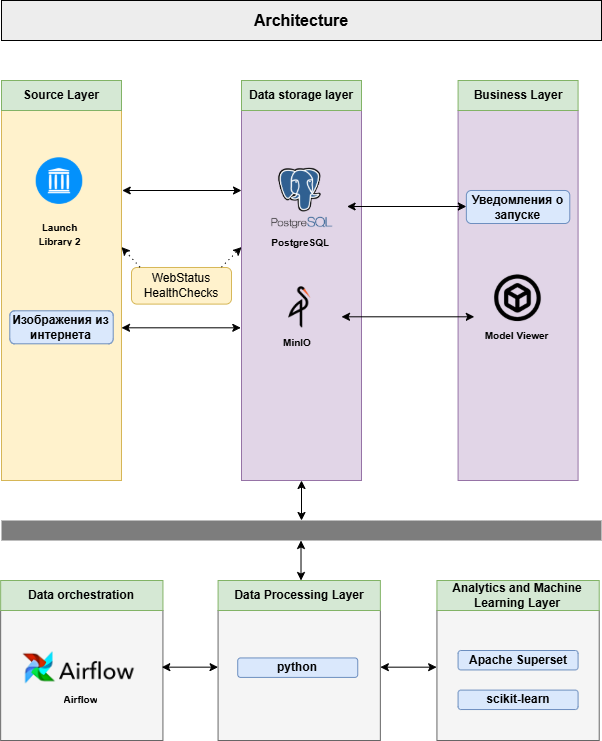


Рисунок 8 – Архитектура бизнес-кейса по ракетам

Архитектура DAG представлена на рисунке 9. Для блока с источниками данных используется download\_launches, который выгружает данные о запусках, фотографии выгружаются через интернет и сохраняются через get\_pictures, а система мониторинга реализована в методе notify.

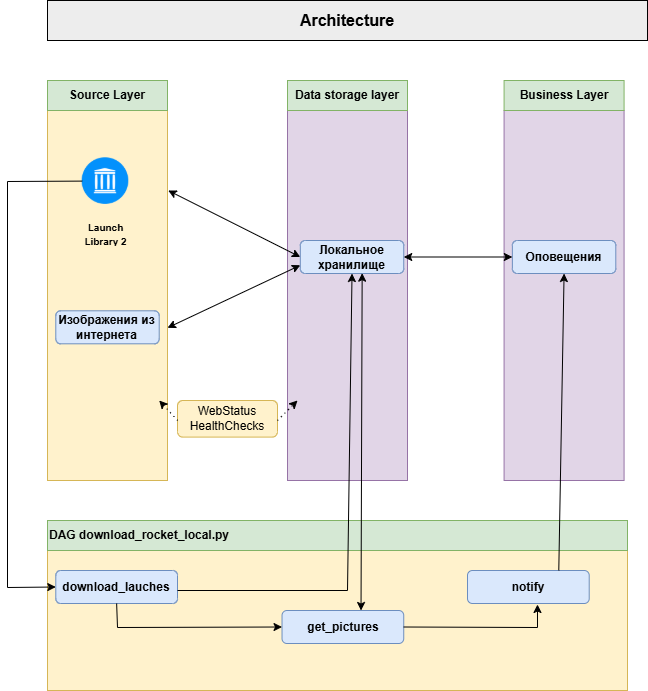


Рисунок 9 – Архитектура DAG

В рамках индивидуального задания необходимо было настроить отчёты по изображениям, которые не удалось скачать, разработать систему мониторинга успешных запусков DAG, а также проанализировать возможные уязвимости. Для данных действий необходимо было добавить сбор информации о неудачных загрузках в get\_pictures, также было выполнено сохранение отчёта в файл для дальнейшей визуализации. Для самого DAG было добавлено отслеживание начала и окончания DAG, были добавлены повторные попытки выполнения для случаев возможных сбоев. При анализе DAG было выявлено, что нет обработки таймаутов для HTTP-запросов, проверки статуса HTTP-ответа, обработки случаев, когда URL изображения отсутствует, механизма повторных попыток при сбоях, централизованного сбора ошибок и статистики. || exit 1 было добавлено для остановки выполнения, если не сработала команда curl. При загрузке изображений была добавлена проверка, что статус 200, то есть всё корректно. Отчёты готовились в формате CSV.

По итогам в рамках отказоустойчивости реализованы повторы с задержкой для сетевых операций, обработка ошибок на всех уровнях, в рамках мониторинга логирование в файлы CSV, для повышения производительности внедрены таймауты. Детальные данные в download\_report\_\*.csv, данные за конкретный последний запуск в stats\_\*.csv, исторические данные в summary\_report.csv. Фрагмент DAG представлен на рисунке 10.

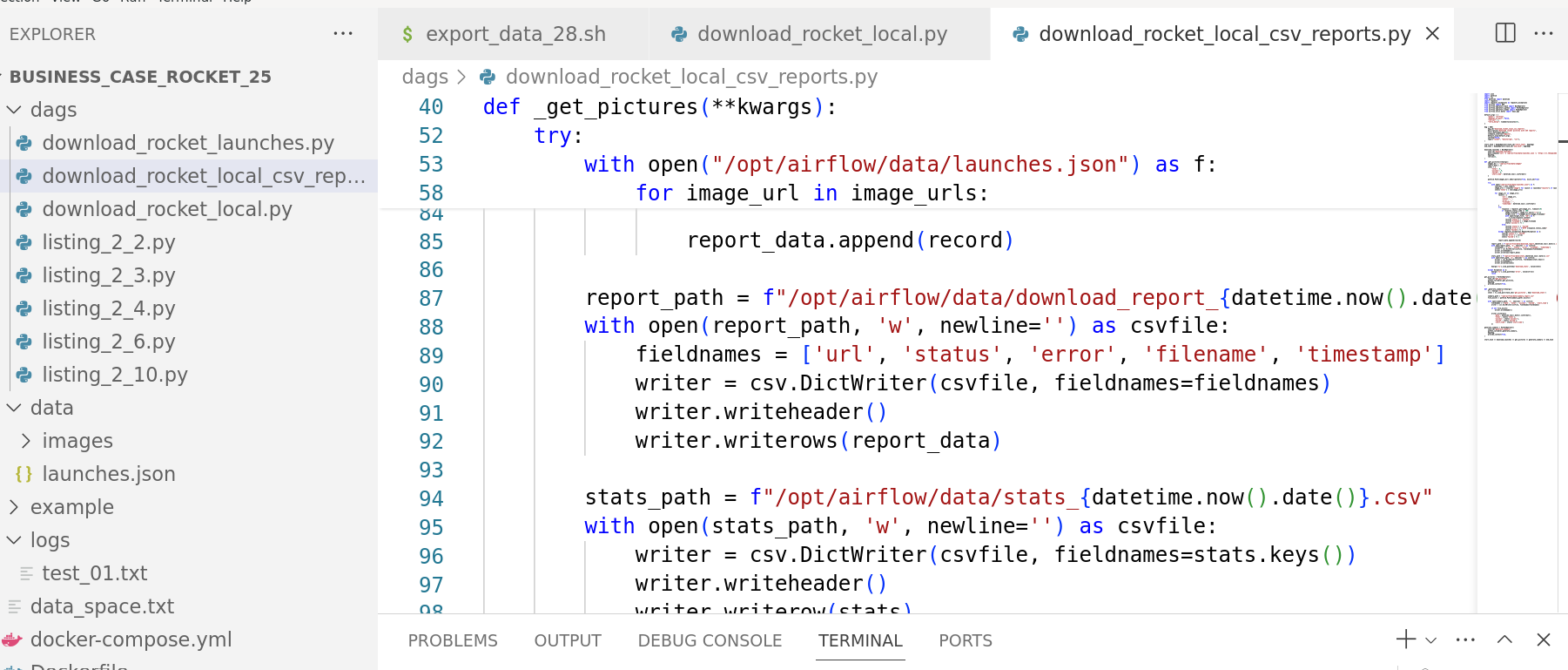


Рисунок 10 – Фрагмент DAG с загрузкой данных в CSV файлы

Далее DAG выполняется на рисунке 11.

Изображение выглядит как текст, программное обеспечение, число, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 11 – Выполнение DAG

Все файлы выгружены на рисунке 12.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 12 – Выгрузка CSV файлов

Далее файлы были скачаны и проанализированы в Google Colaboratory для дальнейшего анализа, как видно на рисунках 13–15.

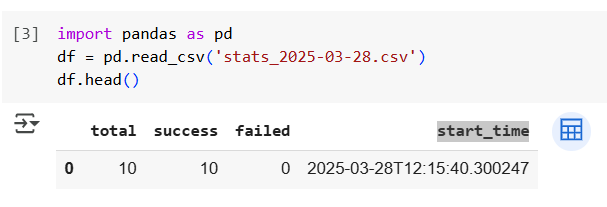


Рисунок 13 – Таблица дневной статистики

Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 14 – Подробная статистика по каждому изображению

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 15 – Историческая статистика

Было загружено 10 изображений успешно, DAG запускался 2 раза за день. Значения с датой были преобразованы в тип даты на рисунке 16, а также был проведён поиск дубликатов.



Рисунок 16 – Предварительная обработка и анализ дублей

Было выявлено 2 дубля. Далее был построен график, показывающий прогресс загрузки, чтобы проанализировать производительность DAG. Код представлен на рисунке 17, а сам график на рисунке 18.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 17 – Построение графика процесса загрузки фотографий

Изображение выглядит как линия, График, диаграмма, скат

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 18 – График прогресса загрузки файлов

Файлы были загружены за 11 секунд, линия растёт равномерно, значит, загрузка стабильная со 2-го файла, при этом разница между 1 и 2 файлом равна 4 секундам. Также был проведён анализ временных задержек между загрузками на рисунке 19. Файлы загружаются в среднем быстро, за 1 секунду, что говорит о высокой производительности DAG.

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 19 – Анализ временных задержек между загрузками

DAGs были проанализированы также с помощью Airflow, где также есть метрики для анализа. Для аналитики построена таблица 1. Новый DAG в 2 раза быстрее, улучшение длительности примерно на 40%, более стабильный. Новый DAG имеет систему отчётности и аналитики. Также новый DAG использует больше ресурсов, но даёт больше аналитических возможностей.

Таблица 1 - Сравнительный анализ DAG: download\_rocket\_local vs download\_rocket\_local\_csv\_reports

|  |  |  |
| --- | --- | --- |
| Метрика | Старый DAG | Новый DAG |
| Среднее время выполнения | 00:00:44 | 00:00:22 |
| Макс. длительность | 00:00:46 | 00:00:27 |
| Мин. длительность | 00:00:25 | 00:00:06 |
| Успешность запусков | 15/15 | 2/2 |
| Генерация отчетов | Нет | CSV-отчеты (3 вида) |
| Мониторинг ошибок | Базово | Есть логи |

Если выполнять запуск 15 раз в день, то выгода будет 15 запусков × (44 сек – 22 сек) = 330 сек/день = 5.5 минут экономии в день. Также автоматические CSV-отчеты и интеграция с Colab экономят примерно 2–3 часа/неделю, ведь теперь не нужно проверять логи и проводить анализ вручную. Следовательно, сокращается время на администрирование. До 90% сокращения времени на поиск и исправление сбоев. По итогу происходит экономия стоимости при хранении логов, предотвращении простоев, трудозатратах сотрудников.

**Выводы:**

В рамках работы был реализован ETL-процесс загрузки данных о запусках ракет и изображений, а также автоматическая выгрузка результатов. Построены 2 архитектуры: для самого DAG и для бизнес-кейса по запуску ракет. .sh-скрипты позволяют автоматизировать процессы и, хотя Airflow поддерживает BashOperator, сложные Bash-команды лучше выносить в отдельные .sh-файлы. .sh-файл для выгрузки результатов DAG был успешно создан. Был создан новый DAG по индивидуальному заданию, который лучше старого, т. к. снижаются затраты на инфраструктуру, мониторинг. Цель и задачи были выполнены.