

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

Лабораторная работа 5.1

Проектирование объектной модели данных. Проектирование сквозного
конвейера ETL

Выполнила: Сачкова Г.Г. , группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2025

Задание 4.1. Бизнес кейс «Umbrella»

4.1.1. Развернуть конфигурацию репозитория ВМ в VirtualBox.

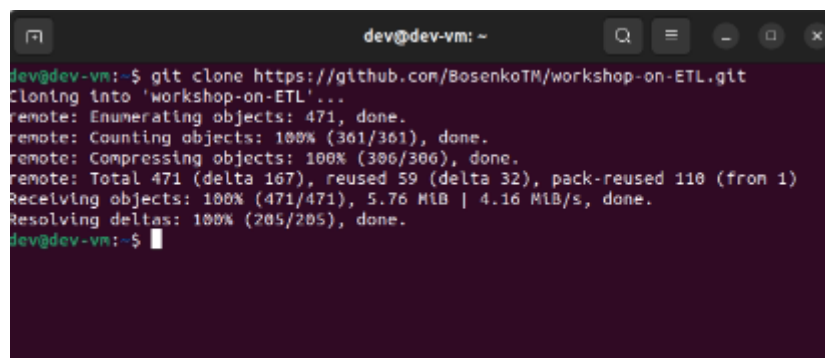
4.1.2. Клонировать на ПК задание Бизнес кейс Umbrella в домашний каталог ВМ.

4.1.3. Запустить контейнер с кейсом, изучить и описать основные элементы интерфейса Apache Airflow.

4.1.4. Спроектировать верхнеуровневую архитектуру аналитического решения задания Бизнес кейс Umbrella в draw.io. Необходимо использовать:

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

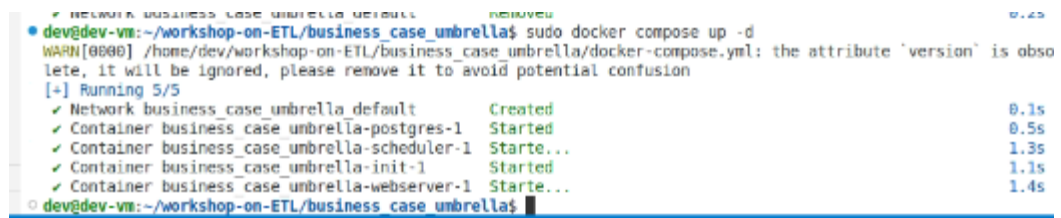
Для начала надо клонировать репозиторий, это продемонстрировано на рисунке 1.



```
dev@dev-vm: ~  
dev@dev-vm:~$ git clone https://github.com/BosenkoTM/workshop-on-ETL.git  
Cloning into 'workshop-on-ETL'...  
remote: Enumerating objects: 471, done.  
remote: Counting objects: 100% (361/361), done.  
remote: Compressing objects: 100% (306/306), done.  
remote: Total 471 (delta 167), reused 59 (delta 32), pack-reused 110 (from 1)  
Receiving objects: 100% (471/471), 5.76 MiB | 4.16 MiB/s, done.  
Resolving deltas: 100% (205/205), done.  
dev@dev-vm:~$
```

Рисунок 1. Клонирование репозитория

Затем нужно запустить контейнер с кейсом. Как можно увидеть на рисунке 2, он успешно загрузился



```
dev@dev-vm:~/workshop-on-ETL/business_case_umbrella$ sudo docker-compose up -d  
WARN[0000] /home/dev/workshop-on-ETL/business_case_umbrella/docker-compose.yml: the attribute 'version' is obso  
lete, it will be ignored, please remove it to avoid potential confusion  
[+] Running 5/5  
✔ Network business_case_umbrella default Created 0.1s  
✔ Container business_case_umbrella-postgres-1 Started 0.5s  
✔ Container business_case_umbrella-scheduler-1 Starte... 1.3s  
✔ Container business_case_umbrella-init-1 Started 1.1s  
✔ Container business_case_umbrella-webserver-1 Starte... 1.4s  
dev@dev-vm:~/workshop-on-ETL/business_case_umbrella$
```

Рисунок 2. Контейнер с кейсом запустился

Надо проверить, что airflow действительно работает. На рисунке 3 видно, что он открылся

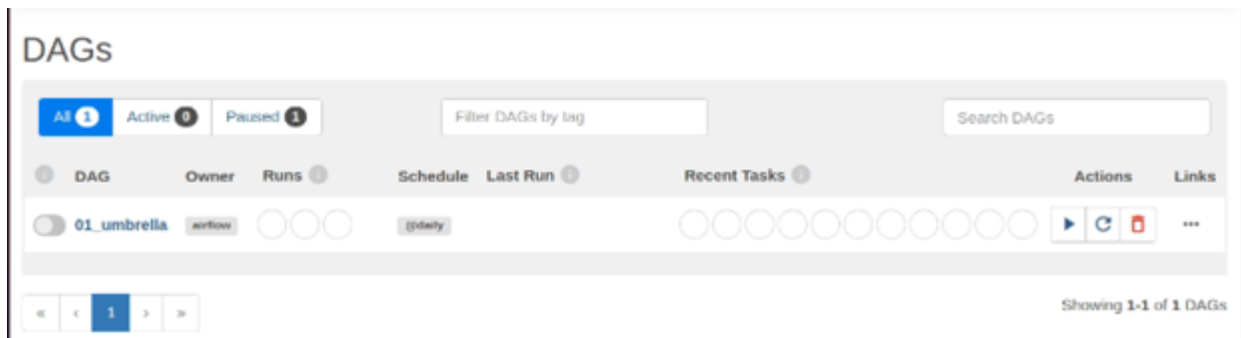


Рисунок 3. Запуск airflow

Дерево дага 01_umbrella изображен на рисунке 4.

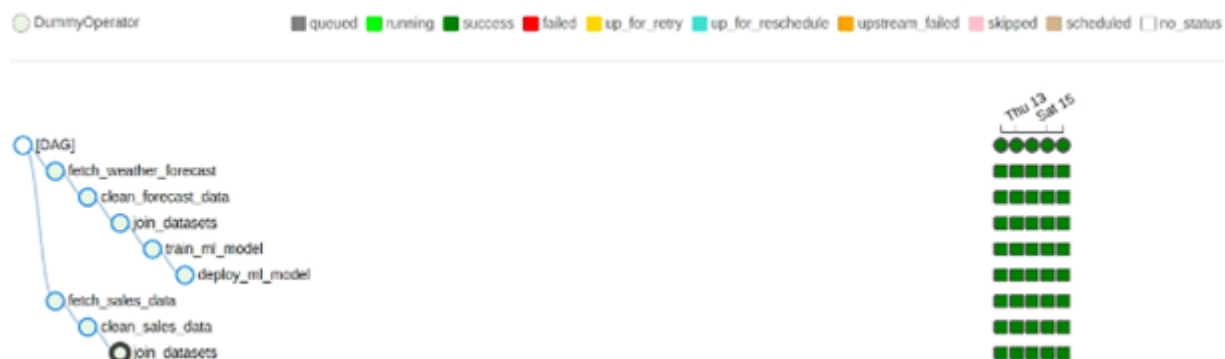


Рисунок 4. Дерево дага 01_umbrella

На рисунке 5 виден график дага, который успешно отработал.

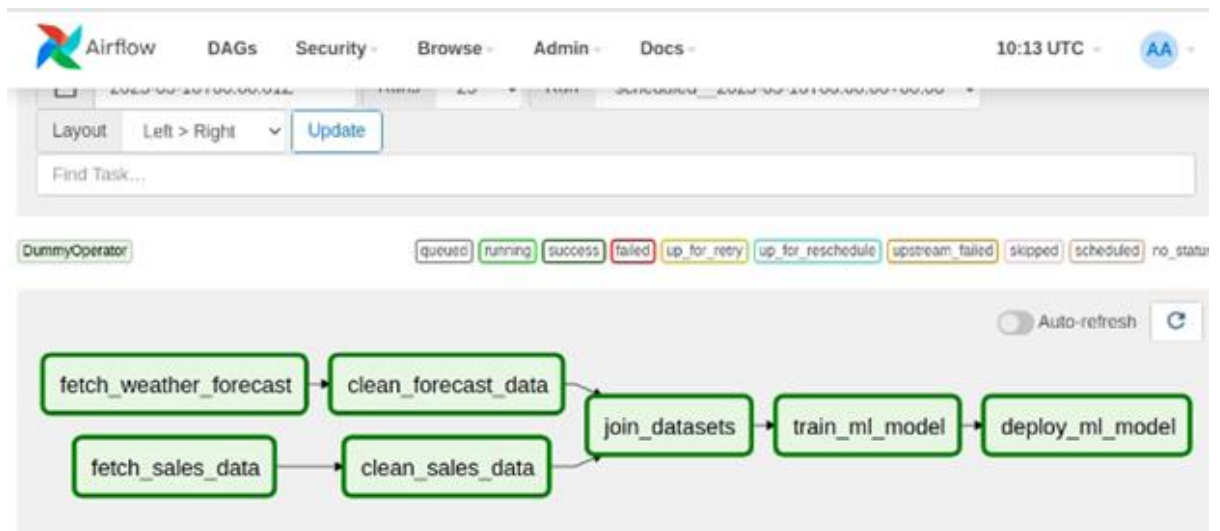


Рисунок 5. График дага

На рисунке 6 продемонстрирован Apache airflow

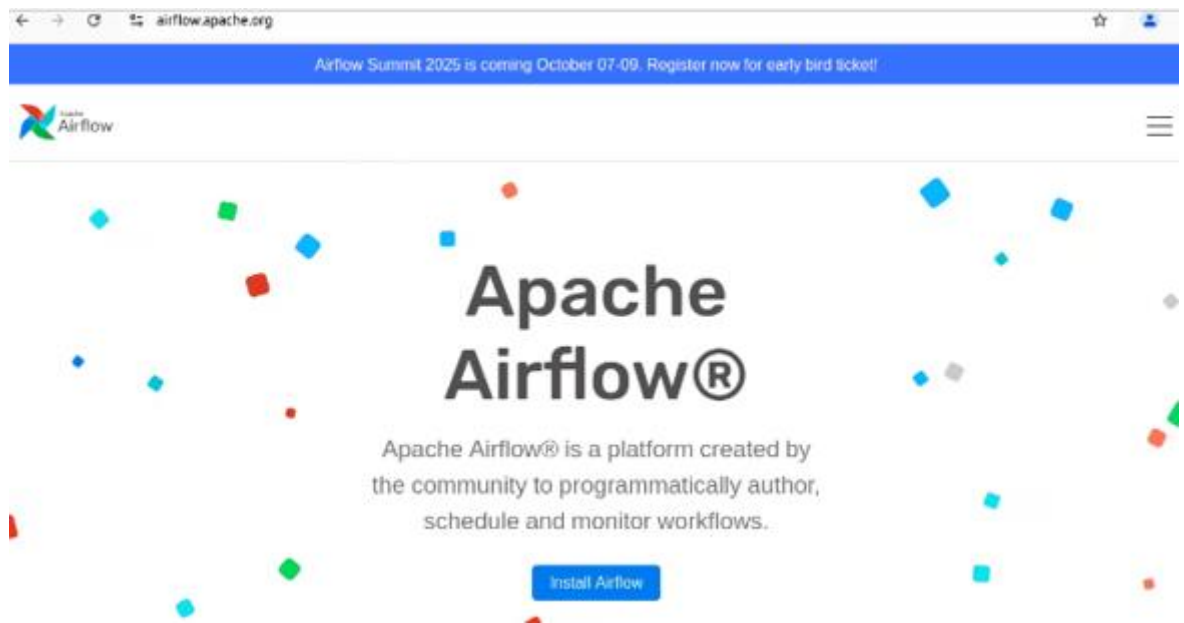


Рисунок 6. Apache airflow

Основные элементы интерфейса Apache airflow:

- **DAG (Directed Acyclic Graph).** Это структура, состоящая из объектов (узлов), которые связаны между собой. DAG описывает логику выполнения задач: какие должны быть выполнены, в каком порядке и как часто.
- **Задача (Task).** Описывает, что делать, например, выборку данных, анализ, запуск других систем. Каждая задача — это экземпляр оператора с определёнными параметрами.
- **Оператор (Operator).** Класс Python, который определяет, что нужно сделать в рамках задачи. Есть операторы для выполнения скриптов Bash, кода Python, SQL-запросов.
- **Веб-сервер.** Предоставляет пользовательский интерфейс для мониторинга, управления и запуска задач. Через веб-интерфейс пользователи могут просматривать список задач, проверять их статус и управлять расписанием выполнения.
- **База метаданных.** Хранит информацию о задачах, их статусе, зависимостях и истории выполнения.
- **Планировщик (Scheduler).** Служба, отвечающая за планирование в Airflow. Отслеживая все созданные Task и DAG, планировщик

инициализирует Task Instance — по мере выполнения необходимых для их запуска условий.

Верхнеуровневая архитектура аналитического решения кейса Umbrella изображена на рисунке 7.

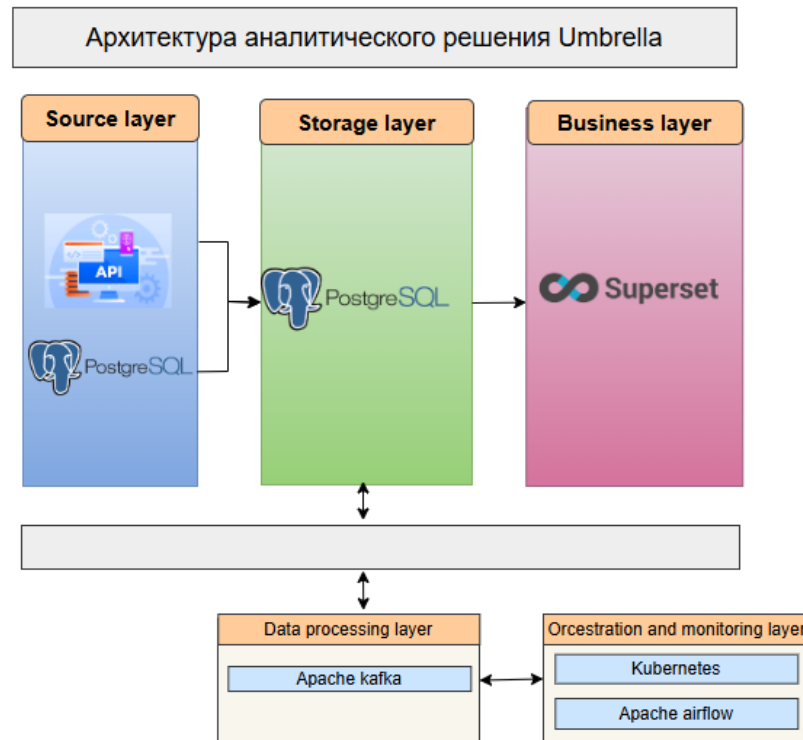


Рисунок 7. Верхнеуровневая архитектура

Схема работы системы:

1. **Источники данных:**
 - Данные загружаются из PostgreSQL.
 - Данные о погоде загружаются с помощью API погоды (каждый час)
2. **Хранение данных:**
 - Хранение данных будет осуществляться с помощью PostgreSQL
3. **Визуализация данных:**
 - Superset подключается к PostgreSQL.
 - Бизнес-пользователи создают дашборды и отчеты в Superset.
4. **Оркестрация и мониторинг:**
 - Airflow управляет всеми ETL-процессами.

○ Kubernetes управляет контейнерами и обеспечивает масштабируемость и отказоустойчивость.

Архитектура обладает масштабируемостью, надёжностью и гибкостью решения для анализа данных.

Задание 4.2. Basic pipeline ETL. Построить конвейер данных на основании Basic pipeline ETL.rar

Для выполнения этого задания необходимо импортировать csv, загрузить файл и прочитать его, а также посмотреть представленную таблицу о поставщиках (рисунок 8).

Импорт csv

```
[1] import csv
```

To deal with files in Python, we use the open() function, it's a built-in Python function. This function accepts two different arguments (inputs) in the parentheses, always in the following order:

- the name of the file (as a string)
- the mode of working with the file (as a string)

The syntax to open a file in python is:

```
file_obj = open("filename", "mode")
```

Чтение файла

```
[2] f = open(r'suppliers.csv')
    reader = csv.reader(f)
    print(reader)
```

<csv.reader object at 0x7f2aab906a40>

id	supplier_name	contact_person	contact_email	contact_phone	product_name	price	purchase_date	quantity
1	Postavshhik A	Ivan Ivanov	ivan@supplierA.com	123-456-7890	Tovar 1	100.00	15.01.2023	10
2	Postavshhik A	Ivan Ivanov	ivan@supplierA.com	123-456-7890	Tovar 2	150.50	20.01.2023	5
3	Postavshhik B	Petr Petrov	petr@supplierB.com	123-456-7891	Tovar 1	95.00	25.01.2023	15
4	Postavshhik B	Petr Petrov	petr@supplierB.com	123-456-7891	Tovar 3	200.00	30.01.2023	8
5	Postavshhik V	Sergej Sergeev	sergey@supplierC.com	123-456-7892	Tovar 2	120.00	01.02.2023	20
6	Postavshhik V	Sergej Sergeev	sergey@supplierC.com	123-456-7892	Tovar 4	250.00	05.02.2023	3
7	Postavshhik G	Anna Annova	anna@supplierD.com	123-456-7893	Tovar 1	110.00	10.02.2023	7
8	Postavshhik G	Anna Annova	anna@supplierD.com	123-456-7893	Tovar 2	140.00	15.02.2023	12
9	Postavshhik D	Oleg Olegov	oleg@supplierE.com	123-456-7894	Tovar 3	130.00	20.02.2023	10
10	Postavshhik D	Oleg Olegov	oleg@supplierE.com	123-456-7894	Tovar 5	220.00	25.02.2023	6
11	Postavshhik E	Dmitrij Dmitriev	dmitry@supplierF.com	123-456-7895	Tovar 1	105.00	01.03.2023	14
12	Postavshhik E	Dmitrij Dmitriev	dmitry@supplierF.com	123-456-7895	Tovar 6	300.00	05.03.2023	4
13	Postavshhik ZH	Elena Elenovna	elena@supplierG.com	123-456-7896	Tovar 4	115.00	10.03.2023	9
14	Postavshhik ZH	Elena Elenovna	elena@supplierG.com	123-456-7896	Tovar 2	125.00	15.03.2023	11
15	Postavshhik Z	Mikhail Mixajlov	mikhail@supplierH.com	123-456-7897	Tovar 7	160.00	20.03.2023	8
16	Postavshhik Z	Mikhail Mixajlov	mikhail@supplierH.com	123-456-7897	Tovar 8	180.00	25.03.2023	5

Рисунок 8. Импорт csv и загрузка файла

На рисунке 9 показана фильтрация строк у столбца supplier_name, а именно вывод только тех строк, которые содержат «Поставщик А», «Поставщик Б», «Поставщик В», «Поставщик Д», «Поставщик Ж». А далее настраивается подключение к sqllite, удаление таблицы с поставщиками, чтобы создать эту таблицу.

```
[1] Code = ['Поставщик А', 'Поставщик Б', 'Поставщик В', 'Поставщик Д', 'Поставщик Ж']

supp_data = []

next(reader, None)
for row in reader:
    if(row[1] in Code):
        row[3] = float(row[3]) * 100
        supp_data.append(row)

print(len(supp_data))
print(supp_data[0:4])

[[('1', 'Поставщик А', 'Маша Иванова', 10000.0, '123-456-7890', 'Товар 1', '100.00', '2023-01-15', '10'), ('2', 'Поставщик А', 'Маша Иванова', 15050.0, '123-456-7890', 'Товар 2', '150.50', '2023-01-20', '5'), ('3', 'Поставщик Б', 'Сергей Сергеев', 12000.0, '123-456-7892', 'Товар 1', '120.00', '2023-02-01', '20')]]

Loading the data into SQL DB

Загрузка в SQL DB

[4] import sqlite3
conn = sqlite3.connect('sessia.db')

[5] try:
    conn.execute('DROP TABLE IF EXISTS `suppliers`')
except Exception as e:
    print(str(e))

[6] try:
    conn.execute("""
CREATE TABLE suppliers (
    id INT,
    supplier_name VARCHAR(100),
    contact_person VARCHAR(100),
    contact_email VARCHAR(100),
    contact_phone VARCHAR(15),
    product_name VARCHAR(100),
    price DECIMAL(10, 2),
    purchase_date DATE,
    quantity INT);""")
    print('Table created successfully');
except Exception as e:
    print(str(e))
    print('Table Creation Failed')
finally:
    conn.close()

Table created successfully
```

Рисунок 9. Фильтрация строк, создание таблицы в базе данных о поставщиках

На рисунке 10 показана выборка столбцов, загрузка значений в таблицу supplier в базу данных, вывод строк всех и выгрузка csv файла

```
[7] print(supp_data[4])

['5', 'Поставщик В', 'Сергей Сергеев', 12000.0, '123-456-7892', 'Товар 2', '120.00', '2023-02-01', '20']

Выбор столбцов

[8] data_s = [(row[0], row[1], row[3], row[5], row[6], row[7], row[8]) for row in supp_data]
data_s[:2]

[('1', 'Поставщик А', 10000.0, 'Товар 1', '100.00', '2023-01-15', '10'), ('2', 'Поставщик А', 15050.0, 'Товар 2', '150.50', '2023-01-20', '5')]

[13] conn = sqlite3.connect('sessia.db')
cur = conn.cursor()
try:
    cur.executemany("INSERT INTO suppliers(id, supplier_name,contact_person, product_name, price, purchase_date, quantity) VALUES (?, ?, ?, ?, ?, ?, ?)", data_s)
    conn.commit()
    print('Data Successfully')
except Exception as e:
    print(str(e))
    print('Data Failed')
finally:
    conn.close()

Data Successfully

Вывод строк

[14] conn = sqlite3.connect('sessia.db')
rows = conn.cursor().execute("select * from suppliers")
for row in rows:
    print(row)

(1, 'Поставщик А', '10000.0', None, None, 'Товар 1', 100, '2023-01-15', 10)
(2, 'Поставщик А', '15050.0', None, None, 'Товар 2', 150.5, '2023-01-20', 5)
(3, 'Поставщик Б', '9500.0', None, None, 'Товар 1', 95, '2023-01-25', 15)
(4, 'Поставщик Б', '20000.0', None, None, 'Товар 3', 200, '2023-01-30', 8)
(5, 'Поставщик В', '12000.0', None, None, 'Товар 2', 120, '2023-02-01', 20)
(6, 'Поставщик В', '25000.0', None, None, 'Товар 4', 250, '2023-02-05', 3)
(9, 'Поставщик Д', '13000.0', None, None, 'Товар 3', 130, '2023-02-20', 10)
(10, 'Поставщик Д', '22000.0', None, None, 'Товар 5', 220, '2023-02-25', 6)
(13, 'Поставщик Ж', '11500.0', None, None, 'Товар 4', 115, '2023-03-10', 9)
(14, 'Поставщик Ж', '12500.0', None, None, 'Товар 2', 125, '2023-03-15', 11)

Write data in a csv file

csvfile = open('itog_suppliers.csv', 'w')
csv_writer = csv.writer(csvfile, lineterminator='\n')
csv_writer.writerow(['id', 'supplier_name', 'contact_person', 'product_name', 'price', 'purchase_date', 'quantity'])
csv_writer.writerows(data_s)
csvfile.close()
```

Рисунок 10. Выборка столбцов, вставка данных и выгрузка в csv файл

Выводы:

1. Был построен конвейер данных;
2. Спроектирована верхнеуровневая архитектура кейса Umbrella;
3. Запущен контейнер с кейсом;
4. Описан интерфейс Apache Airflow.