

Лабораторная работа 3. Программные средства консолидации данных из различных источников с использованием Python и Apache Airflow

Цель: научиться работать с Apache Airflow для автоматизации процессов ETL (Extract, Transform, Load). На практике освоить настройку и выполнение DAG в Airflow для извлечения данных из различных форматов (CSV, Excel, JSON), их обработки на Python, загрузки в базу данных SQLite и отправки уведомлений по электронной почте.

Оборудование и ПО:

- Ubuntu (с Docker)
- Apache Airflow
- SQLite
- Python
- Docker
- email сервис (например, Gmail или локальный SMTP-сервер)

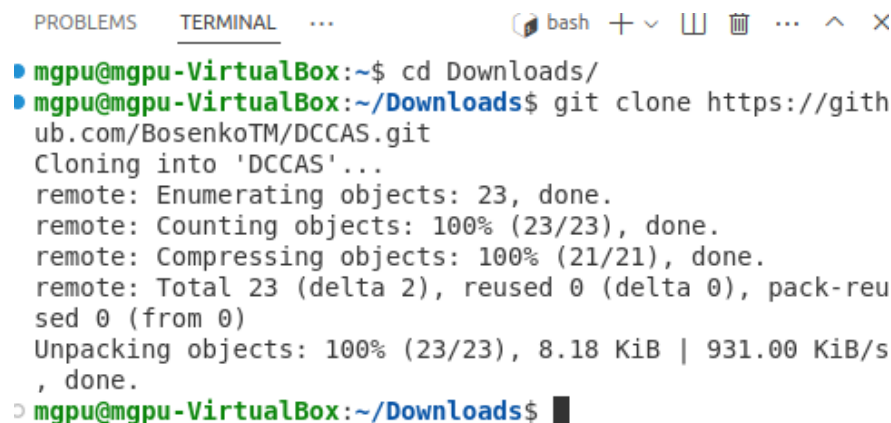
Исходные данные:

- Набор файлов CSV, Excel и JSON, содержащих данные для обработки.
- Конфигурация email для отправки уведомлений.

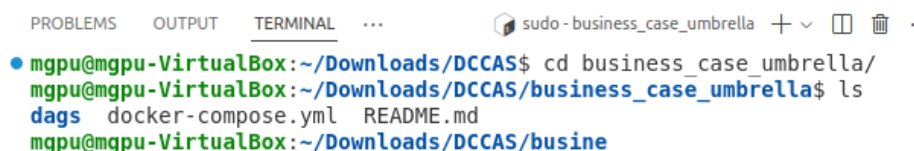
Ход работы

I. Развернуть ВМ *ubuntu_mgpu.ova* в *VirtualBox*. +

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



```
PROBLEMS  TERMINAL  ...  bash + v [ ] [ ] ... ^ x
● mgpu@mgpu-VirtualBox:~$ cd Downloads/
● mgpu@mgpu-VirtualBox:~/Downloads$ git clone https://github.com/BosenkoTM/DCCAS.git
Cloning into 'DCCAS'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 23 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (23/23), 8.18 KiB | 931.00 KiB/s, done.
○ mgpu@mgpu-VirtualBox:~/Downloads$
```



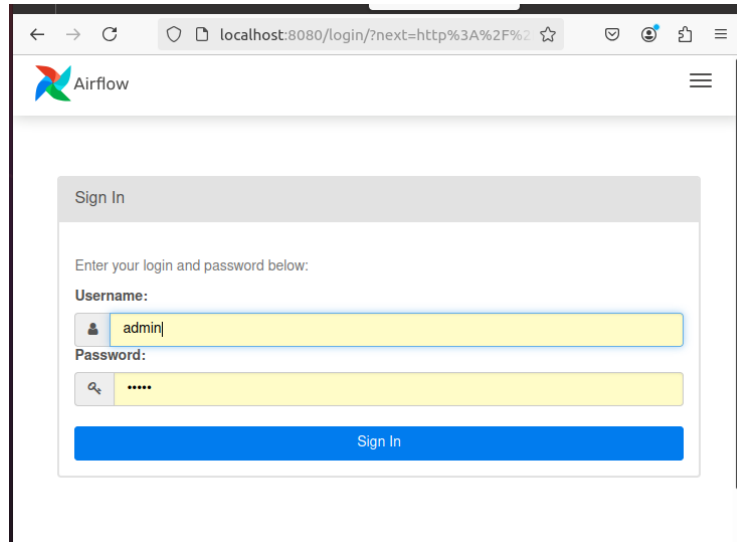
```
PROBLEMS  OUTPUT  TERMINAL  ...  sudo-business_case_umbrella + v [ ] [ ] .
● mgpu@mgpu-VirtualBox:~/Downloads/DCCAS$ cd business_case_umbrella/
mgpu@mgpu-VirtualBox:~/Downloads/DCCAS/business_case_umbrella$ ls
dags  docker-compose.yml  README.md
mgpu@mgpu-VirtualBox:~/Downloads/DCCAS/busine
```

III. Запустить контейнер с кейсом.

```

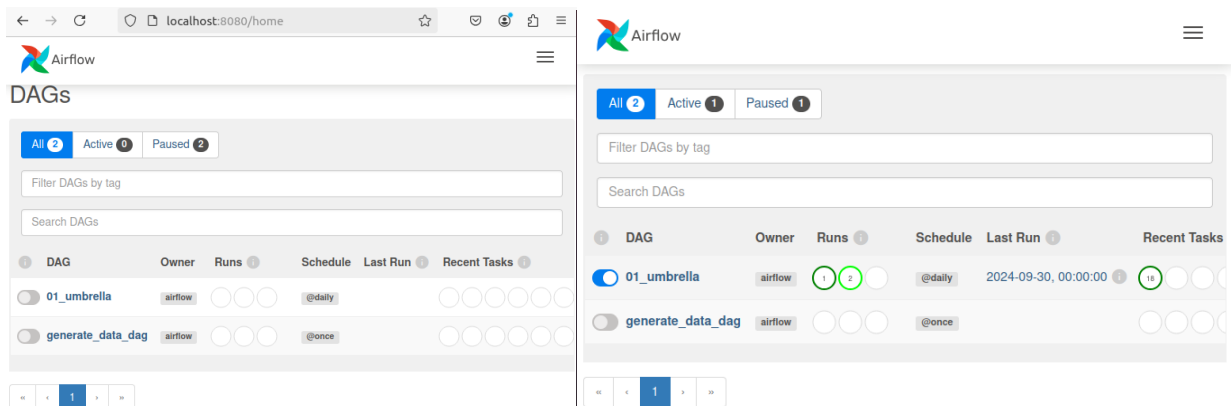
● mgpu@mgpu-VirtualBox:~/Downloads/DCCAS/business_case_umbrella$ sudo docker c
ompose up -d
[+] Running 4/5
  ✔ Network business_case_umbrella_default Created 3.4s
  ✔ Container business_case_umbrella-postgres-1 Started 1.3s
  ✔ Container business_case_umbrella-init-1 Started 2.5s
  ✔ Container business_case_umbrella-scheduler-1 Started 3.0s
  ✔ Container business_case_umbrella-webserver-1 Started 3.0s
○ mgpu@mgpu-VirtualBox:~/Downloads/DCCAS/business_case_umbrella$

```



IV. Изучить и описать основные элементы интерфейса Apache Airflow.

Основные элементы интерфейса **Apache Airflow**, представленные на скриншоте:

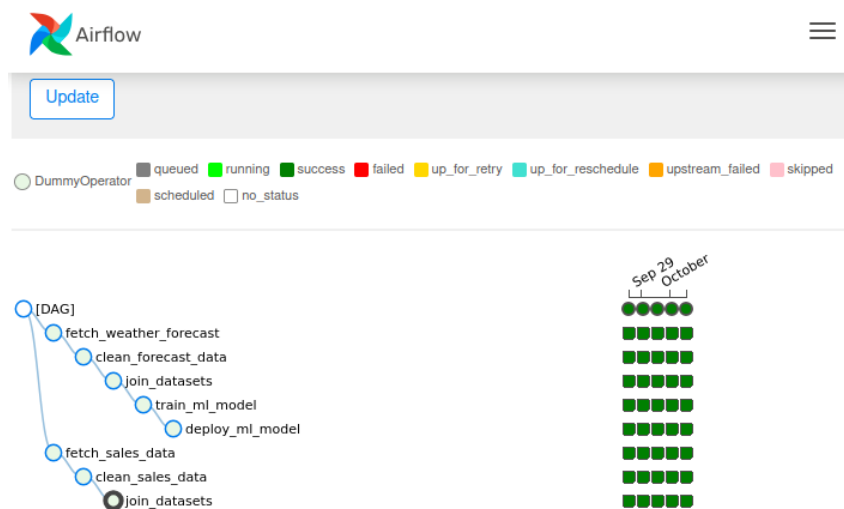


1. **Навигационная панель (с логотипом Airflow):** Расположена в верхней части экрана и служит для переключения между различными разделами интерфейса, такими как список DAGs (Directed Acyclic Graphs), задачи, журналы и т. д.
2. **Фильтры и поиск:**
 - Включает вкладки:
 - All: отображает все DAGs (2 DAG-а на скриншоте).
 - Active: показывает только активные DAGs (1 активный DAG).
 - Paused: показывает DAGs, которые приостановлены.
 - Filter DAGs by tag и Search DAGs: эти поля позволяют фильтровать DAGs по тегам или искать их по имени.
3. **Список DAGs:**
 - Таблица с DAGs включает несколько колонок:
 - DAG: имя DAG. Это 01_umbrella и generate_data_dag.

- Owner: имя владельца DAG. Владелец обоих DAGs — airflow.
- Runs: отображает количество выполнений DAG. В колонке есть индикаторы для текущего и завершенных запусков.
- Schedule: указывает расписание для выполнения DAG. @daily указывает, что DAG 01_umbrella запускается ежедневно.
- Last Run: показывает время последнего запуска DAG.
- Recent Tasks: отображает текущий статус последних задач DAG. Зеленый кружок указывает на успешное выполнение задачи.

4. Переключатель состояния DAG: Справа от имени каждого DAG есть переключатель, позволяющий включать или отключать DAG (запускать или приостанавливать его выполнение).

На скриншоте представлен интерфейс графического представления DAG в Apache Airflow:



Основные элементы интерфейса:

- 1. Кнопка Update:** позволяет обновить статус задач в DAG, чтобы видеть актуальную информацию.
- 2. Легенда:** показывает различные состояния задач, которые отображаются с помощью цветных индикаторов:
 - queued (серый) — задача поставлена в очередь.
 - running (салатовый) — задача выполняется.
 - success (зеленый) — задача успешно выполнена.
 - failed (красный) — задача завершилась неудачей.
 - up_for_retry (желтый) — задача находится в процессе повторной попытки.
 - up_for_reschedule (бирюзовый) — задача будет перенесена.
 - upstream_failed (оранжевый) — не выполнена предыдущая задача.
 - skipped (розовый) — задача была пропущена.
 - scheduled (бежевый) — задача запланирована.
 - no_status (пустой) — нет статуса для задачи.
- 3. Графическая визуализация DAG:**
 - Показана структура DAG, представляющая собой дерево задач.
 - На скриншоте видно две ветки:

- Первая ветка начинается с задачи `fetch_weather_forecast`, затем задачи `clean_forecast_data`, `join_datasets`, `train_ml_model`, и `deploy_ml_model`.
- Вторая ветка начинается с задачи `fetch_sales_data`, затем задачи `clean_sales_data`, и завершается на задаче `join_datasets`.
- Связи между задачами показаны стрелками, указывающими порядок выполнения.

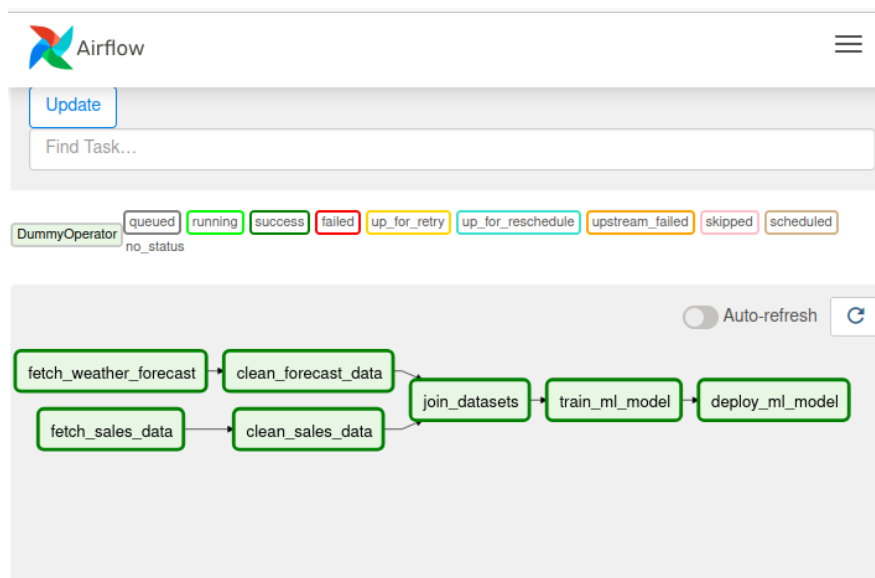
4. Календарь выполнения:

- Справа представлены круговые индикаторы, которые показывают выполнение DAG по дням. Каждый круг символизирует запуск DAG в определенный день.
- Зеленые индикаторы показывают, что все задачи DAG были успешно выполнены в указанный день.

5. Статусы задач:

- На данном графике видно, что большинство задач успешно выполнены (зеленые кружки).

На скриншоте представлен интерфейс графического представления **Graph View** в Apache Airflow:



Вкладка **Graph View** в Apache Airflow предоставляет графическое представление потока выполнения DAG (Directed Acyclic Graph). Она отображает все задачи в DAG, их зависимости и текущий статус выполнения. Основные элементы и функции этой вкладки включают:

1. Графическая схема DAG:

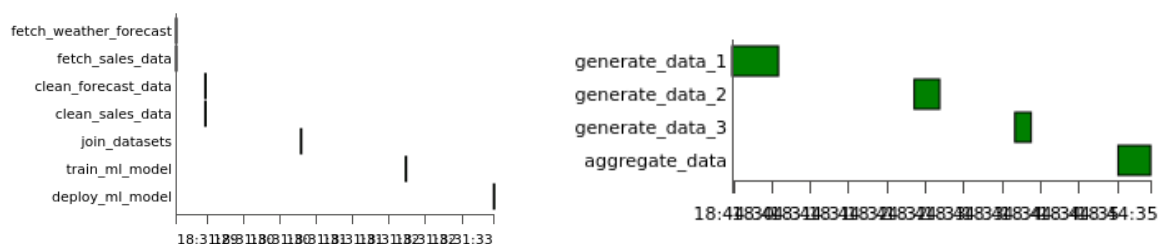
- Задачи представлены в виде узлов (прямоугольников).
- Зависимости между задачами изображаются стрелками, которые показывают, какая задача зависит от выполнения предыдущей задачи.
- Задачи выполняются последовательно, в соответствии с направлением стрелок, начиная с корневой задачи и двигаясь по графу.

2. Цветовые индикаторы: цвет задач указывает их текущий статус:

- `queued` (серый) — задача поставлена в очередь.
- `running` (салатовый) — задача выполняется.
- `success` (зеленый) — задача успешно выполнена.
- `failed` (красный) — задача завершилась неудачей.

- `up_for_retry` (желтый) — задача находится в процессе повторной попытки.
 - `up_for_reschedule` (бирюзовый) — задача будет перенесена.
 - `upstream_failed` (оранжевый) — не выполнена предыдущая задача.
 - `skipped` (розовый) — задача была пропущена.
 - `scheduled` (бежевый) — задача запланирована.
 - `no_status` (пустой) — нет статуса для задачи.
3. **Увеличение/уменьшение масштаба и перемещение:** можно увеличивать или уменьшать масштаб схемы, а также перемещать граф в пределах окна просмотра.
 4. **Интерактивность:**
 - Нажав на узел задачи, можно увидеть дополнительную информацию о задаче, такую как логи выполнения, расписание, статус и другие метаданные.
 - Можно инициировать запуск задачи, перезапустить или приостановить выполнение прямо из графического интерфейса.
 5. **Структура DAG:** граф дает наглядное представление о том, как задачи взаимодействуют друг с другом. Это помогает понять порядок выполнения задач и логику обработки данных.

На скриншоте представлен интерфейс графического представления **Gantt** в Apache Airflow, которая позволяет отслеживать временные интервалы выполнения каждой задачи:



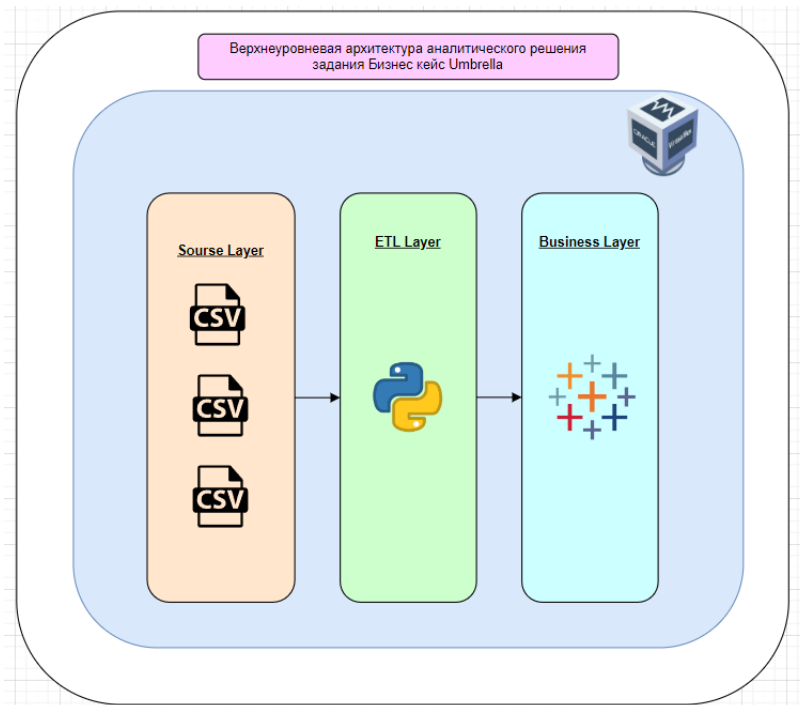
Основные особенности этой вкладки:

1. **Диаграмма Ганта:**
 - Оси времени: горизонтальная ось представляет собой временную шкалу, которая показывает время выполнения задач. Она позволяет визуально оценить, как долго выполнялась каждая задача и в какие моменты времени они начинались и заканчивались.
 - Задачи: Каждая задача представлена горизонтальной полосой. Начало полосы соответствует времени старта задачи, а её длина указывает на продолжительность выполнения.
2. **Цветовые индикаторы:** подобно другим видам в Airflow, цвет полос задач на диаграмме Ганта показывает их статус.
3. **Интерактивность:** нажав на любую полосу на диаграмме Ганта, можно получить более детальную информацию о задаче.

Анализ зависимости времени выполнения: диаграмма Ганта помогает анализировать, как задачи DAG выполняются во времени относительно друг друга. Можно увидеть, какие задачи выполнялись параллельно, какие задачи задержали выполнение других задач, и как оптимизировать граф для более эффективного выполнения.

V. 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.



VI. Выполнить индивидуальное задание.

Задание 16: создать DAG, который извлекает данные из CSV, Excel и JSON файлов, выполняет базовые операции по очистке (удаление дубликатов, обработка пропущенных значений) и сохраняет результат в Google Drive в формате Excel.

1. **Создание файла DAG partners_data_processing_dag:**

- CSV файл (partners_data.csv): содержит ID, Сумма сделок и Категория партнера.
- Excel файл (partners_transactions.xlsx): содержит ID, Количество сделок и Категория партнера.
- JSON файл (partners_average_deals.json): содержит ID, Средняя сумма сделки и Категория партнера.
- Очищенные данные сохраняются в cleaned_partners_data.xlsx.

```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator # type: ignore
from datetime import datetime
import pandas as pd # type: ignore
import random
import json

# Функция для генерации CSV файла с данными о партнерах
def generate_csv_data():
    data_csv = {
```

```

        'ID': [i for i in range(1, 101)],
        'Сумма сделок': [random.randint(10000, 50000) for _ in
range(100)],
        'Категория партнера': [random.choice(['Поставщик', 'Клиент',
'Партнер']) for _ in range(100)],
    }
    df_csv = pd.DataFrame(data_csv)
    df_csv.to_csv('partners_data.csv', index=False, encoding='utf-8')

# Функция для генерации Excel файла с данными о партнерах
def generate_excel_data():
    data_excel = {
        'ID': [i for i in range(51, 151)],
        'Количество сделок': [random.randint(1, 10) for _ in
range(100)],
        'Категория партнера': [random.choice(['Поставщик', 'Клиент',
'Партнер']) for _ in range(100)],
    }
    df_excel = pd.DataFrame(data_excel)
    df_excel.to_excel('partners_transactions.xlsx', index=False)

# Функция для генерации JSON файла с данными о партнерах
def generate_json_data():
    data_json = [
        {'ID': i, 'Средняя сумма сделки': round(random.uniform(1000.0,
20000.0), 2), 'Категория партнера': random.choice(['Поставщик',
'Клиент', 'Партнер'])}
        for i in range(26, 126)
    ]
    with open('partners_average_deals.json', 'w', encoding='utf-8') as
f:
        json.dump(data_json, f, ensure_ascii=False)

# Функция для обработки данных (очистка, удаление дубликатов и
заполнение пропусков)
def clean_data():
    # Загрузка данных
    df_csv = pd.read_csv('partners_data.csv', encoding='utf-8')
    df_excel = pd.read_excel('partners_transactions.xlsx')
    with open('partners_average_deals.json', 'r', encoding='utf-8') as
f:
        data_json = json.load(f)
    df_json = pd.DataFrame(data_json)

    # Объединение всех данных по 'ID'
    merged_df = pd.merge(df_csv, df_excel, on='ID', how='outer')
    merged_df = pd.merge(merged_df, df_json, on='ID', how='outer')

```

```

# Очистка данных: удаление дубликатов и обработка пропусков
merged_df.drop_duplicates(subset='ID', keep='first', inplace=True)
merged_df.fillna({'Сумма сделок': 0, 'Количество сделок': 0,
'Sредняя сумма сделки': 0}, inplace=True)

# Сохранение результата в Excel файл
merged_df.to_excel('cleaned_partners_data.xlsx', index=False)

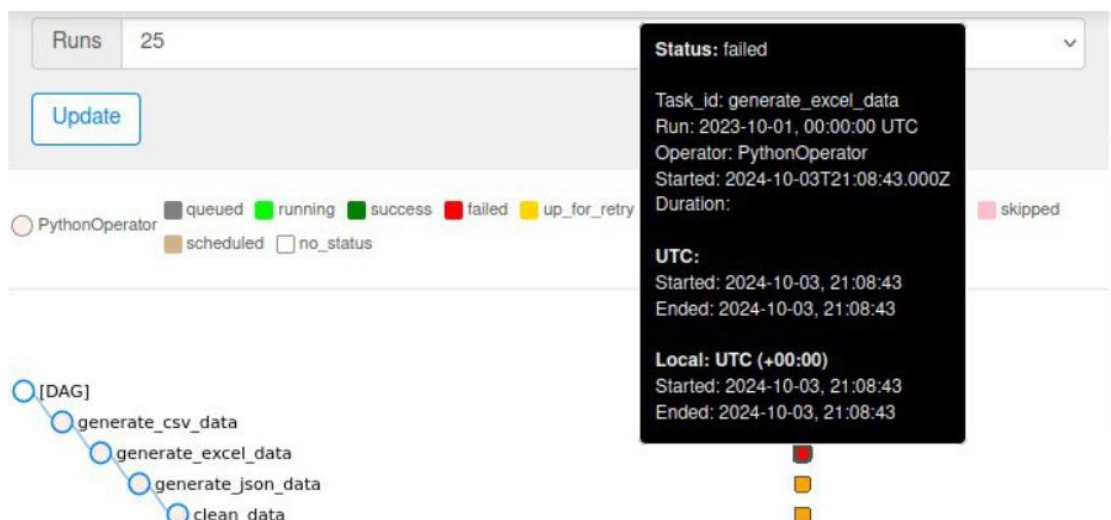
# Определение DAG
dag = DAG('partners_data_processing_dag', description='Генерация и
очистка данных о партнерах',
        schedule_interval='@once', start_date=datetime(2023, 10, 1),
catchup=False)

# Определение задач
task_generate_csv_data = PythonOperator(task_id='generate_csv_data',
python_callable=generate_csv_data, dag=dag)
task_generate_excel_data =
PythonOperator(task_id='generate_excel_data',
python_callable=generate_excel_data, dag=dag)
task_generate_json_data = PythonOperator(task_id='generate_json_data',
python_callable=generate_json_data, dag=dag)
task_clean_data = PythonOperator(task_id='clean_data',
python_callable=clean_data, dag=dag)

# Установка зависимостей между задачами
task_generate_csv_data >> task_generate_excel_data >>
task_generate_json_data >> task_clean_data

```

2. Запустили контейнер с кейсом про партнеров компании:



3. Проверили вкладку "Log" (журнал), чтобы увидеть сообщения об ошибках: Судя по приведённым логам, проблема заключается в том, что модуль **orenpurhl** не установлен в окружении **Apache Airflow**. Этот модуль необходим для работы с Excel файлами через библиотеку **pandas**.

4. Устанавливаем модуль orepuxl:



































```

● mgpu@mgpu-VirtualBox:~/Downloads/DCCAS/business_case_umbrella$ sudo docker p
S
CONTAINER ID        IMAGE                                     COMMAND                  CRE
ATED              STATUS              PORTS                    NAME
S
9dc9ef18a742       apache/airflow:2.0.0-python3.8        "/usr/bin/dumb-init ..." 9 s
econds ago        Up 6 seconds        0.0.0.0:8080->8080/tcp, :::8080->8080/tcp  busi
ness_case_umbrella-webserver-1
e1b9b9a64a6d       apache/airflow:2.0.0-python3.8        "/bin/bash -c 'airfl..." 9 s
econds ago        Up 7 seconds        8080/tcp                busi
ness_case_umbrella-init-1
5954cd02abb0       apache/airflow:2.0.0-python3.8        "/usr/bin/dumb-init ..." 9 s
econds ago        Up 6 seconds        8080/tcp                busi
ness_case_umbrella-scheduler-1
115fede8849d       postgres:12-alpine                   "docker-entrypoint.s..." 9 s
econds ago        Up 8 seconds        0.0.0.0:5432->5432/tcp, :::5432->5432/tcp  busi
ness_case_umbrella-postgres-1

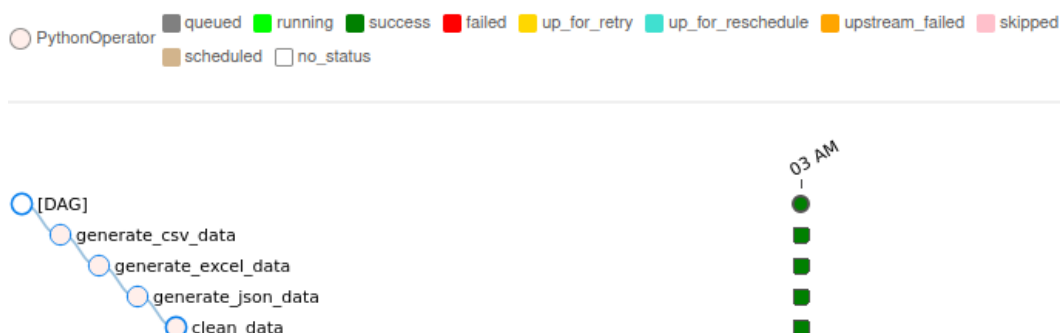
○ mgpu@mgpu-VirtualBox:~/Downloads/DCCAS/business_case_umbrella$ sudo docker e
xec -it 5954cd02abb0 bash
airflow@5954cd02abb0:/opt/airflow$ pip install openpyxl
Defaulting to user installation because normal site-packages is not writeabl
e
Collecting openpyxl
  Downloading openpyxl-3.1.5-py2.py3-none-any.whl (250 kB)
    |████████████████████████████████████████| 250 kB 1.9 MB/s
Collecting et_xmlfile
  Downloading et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Installing collected packages: et_xmlfile, openpyxl
Successfully installed et_xmlfile-1.1.0 openpyxl-3.1.5
WARNING: You are using pip version 20.2.4; however, version 24.2 is availabl
e.
You should consider upgrading via the '/usr/local/bin/python -m pip install
--upgrade pip' command.

```

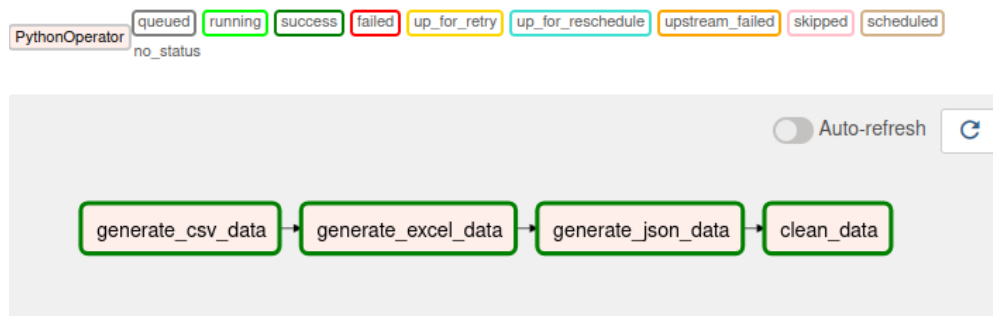
5. Перезапускаем partners data processing dag:

DAG	Owner	Runs	Schedule	Last Run	Recent Tasks	Actions	Links
01_umbrella	airflow	  	@daily		          	  	...
partners_data_processing_dag	airflow	  	@once	2023-10-01, 00:00:00	          	  	...

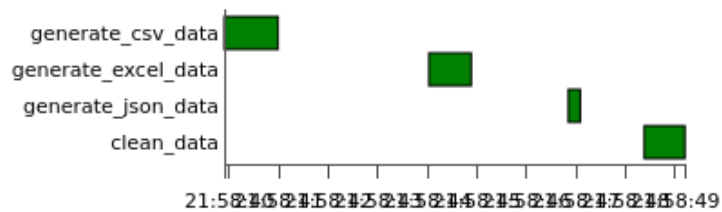
6. На скриншоте представлен интерфейс графического представления DAG в Apache Airflow:



7. На скриншоте представлен интерфейс графического представления Graph View в Apache Airflow:



8. На скриншоте представлен интерфейс графического представления Gantt в Apache Airflow, которая позволяет отслеживать временные интервалы выполнения каждой задачи:



9. Проверим файлы в текущем каталоге:

```
airflow@5954cd02abb0:/opt/airflow$ ls
airflow.cfg          partners_data.csv
cleaned_partners_data.xlsx  partners_transactions.xlsx
dags                 unittests.cfg
logs                 webserver_config.py
partners_average_deals.json
```

10. Как забрать *cleaned_partners_data.xlsx* для из этой директории?
11. Спроектировать верхнеуровневую архитектуру аналитического решения задания Бизнес кейс Задание 16 в draw.io.

