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.

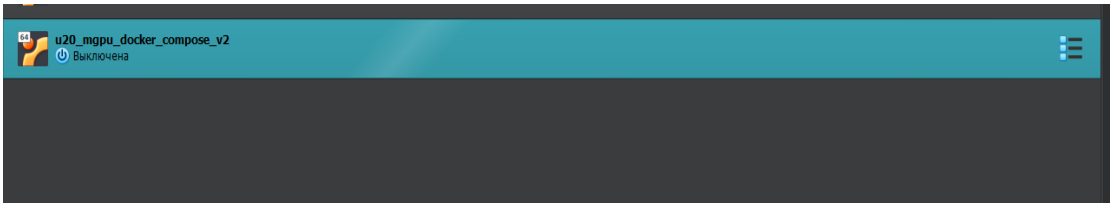
Задание по варианту 23:

Настроить DAG для периодической загрузки данных из CSV, Excel и JSON файлов в базу данных, объединения данных в единый DataFrame и последующей визуализации данных в виде диаграмм, которые сохраняются в виде изображений.

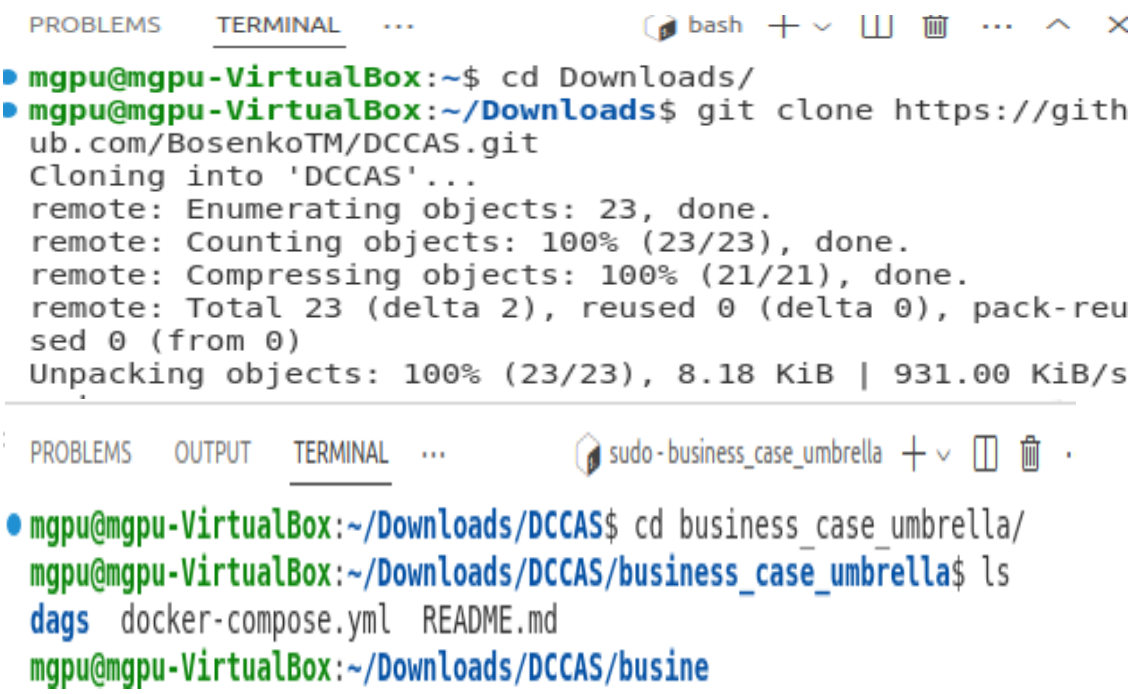
Выполнил: Петров Евгений С., БД-231м.

Ход работы

1. Развернуть ВМ ubuntu_mgpu.ova в VirtualBox.

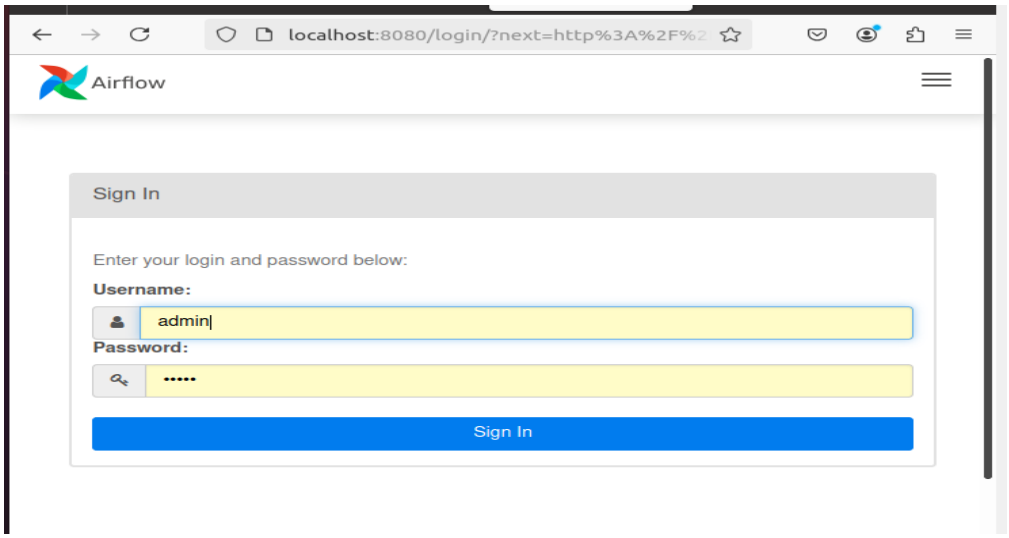


2. Клонировать на ПК задание Бизнес кейс Umbrella в домашний каталог ВМ.


PROBLEMS
TERMINAL
bash
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
PROBLEMS
OUTPUT
TERMINAL
sudo - business_case_umbrella
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

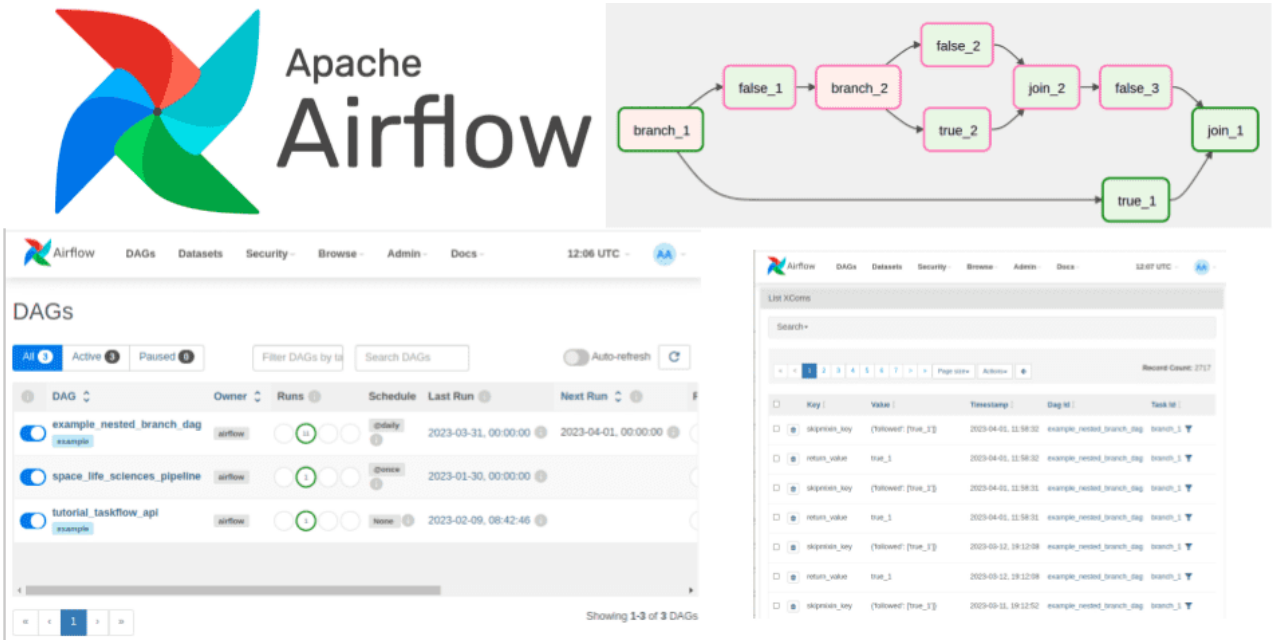
3. Запустить контейнер с кейсом.

```
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$
```



4. Изучить и описать основные элементы интерфейса Apache Airflow.

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



1. DAGS

DAGS (Directed Acyclic Graphs) – это основной элемент в Airflow, представляющий собой структуру, описывающую последовательность выполнения задач. В интерфейсе вы можете:

- Видеть список доступных DAGS.

- Проверять статус каждого DAG (активен, приостановлен и т.д.).

- Открывать визуализацию графа DAG для просмотра зависимостей между задачами.

- Запускать задачи вручную и просматривать их логи.

2. Datasets

Datasets представляют собой данные, которые потребляются или создаются в процессе выполнения задач. В интерфейсе Airflow можно видеть, какие датасеты используются в DAG, их состояния и метаданные. Это помогает в управлении зависимостями между задачами и в гарантии, что данные используются корректно.

3. Security

Элемент безопасности позволяет управлять доступом к интерфейсу Airflow. Здесь можно:

- Настраивать роли и разрешения для пользователей.

- Ограничивать доступ к определенным DAGS или функционалу на основе ролей.

4. Browse

Этот элемент интерфейса предоставляет доступ к различным данным и информации:

- Вы можете просматривать выполненные экземпляры DAG и их задачи (task instances).

- Доступ к информации о привязках к метаданным (например, логам и результатам выполнения).

- Возможность просматривать состояния выполнения задач.

5. Admin

Раздел администрирования предоставляет инструменты для управления настройками Airflow:

- Управление пользователями и ролями.

- Настройки подключения к базам данных и другим системам.

- Настройка параметров выполнения и конфигурации.

6. Docs

Этот раздел содержит документацию по Airflow, включая:

- Описание доступных операторов и как их использовать.

- Лучшие практики для создания и управления DAG.

- Примеры и справочные материалы.

7. Status: All, Active, Paused

В интерфейсе Airflow вы можете фильтровать DAGs по их статусу:

- **All**: Показывает все DAGs.

- **Active**: Показывает только активные (включенные) DAGs.

- **Paused**: Показывает приостановленные DAGs.

8. Autorefresh

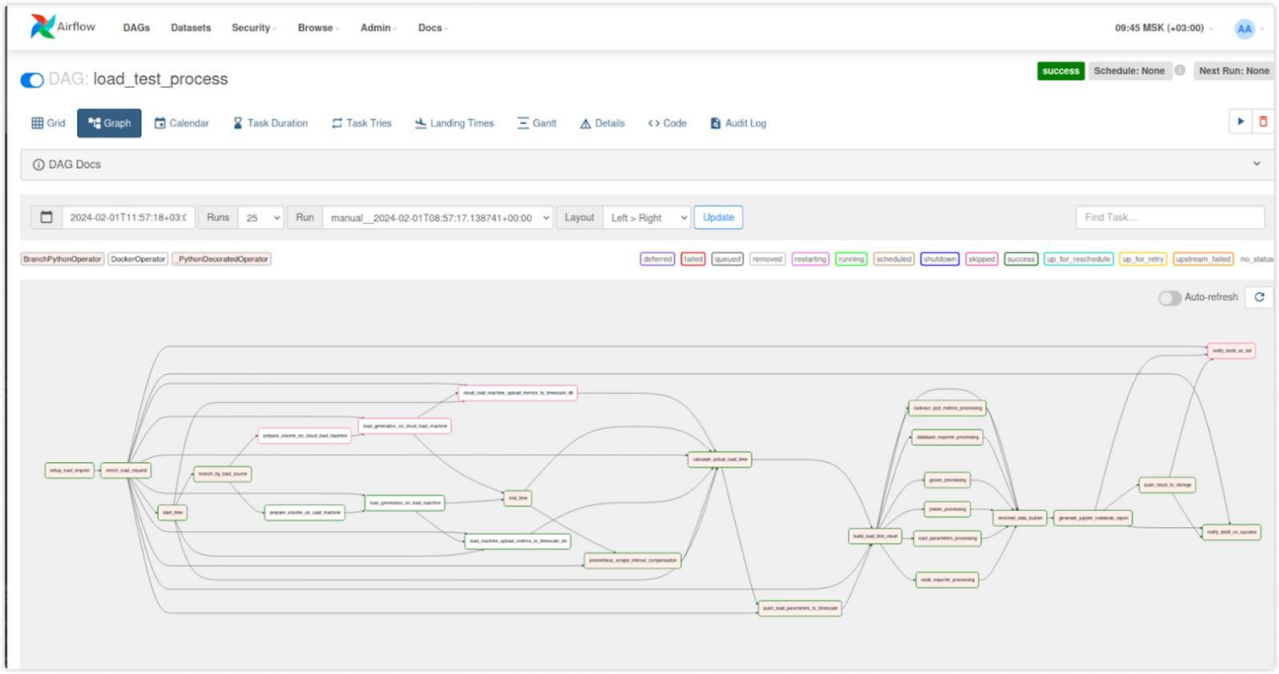
Функция автоматического обновления значения в реальном времени – это удобный инструмент, который позволяет пользователям следить за состоянием задачи и графов без необходимости вручную обновлять страницу.

9. Filter

Элемент фильтрации позволяет пользователям быстро находить конкретные DAGs или задачи по имени, статусу или другим критериям. Это особенно полезно в больших проектах с множеством DAGs.

Эти элементы интерфейса в Apache Airflow помогают пользователям эффективно управлять и мониторить их рабочие процессы, обеспечивая простоту и удобство использования.

На скриншоте представлен интерфейс графического представления **DAG** в Apache Airflow:



Apache Airflow предоставляет мощный интерфейс для графического представления Directed Acyclic Graphs (DAG), что позволяет пользователям эффективно управлять и мониторить задачи. Давайте рассмотрим основные элементы интерфейса:

1. Grid (Сетка):
Это одно из представлений DAG, где отображаются все задачи графа. Каждая ячейка в сетке соответствует конкретной задаче, и пользователи могут видеть состояние задач, такие как «завершено», «в процессе» или «неудачно».

2. Graph (Граф):

Это визуальное представление зависимостей между задачами. Задачи отображаются в виде узлов, а зависимости между ними - в виде рёбер. Это позволяет пользователям легко понять, как задачи связаны друг с другом и в каком порядке они должны выполняться.

3. Calendar (Календарь):

Этот элемент интерфейса позволяет пользователю просматривать, когда были выполнены конкретные задачи и каков их граф выполнения. Календарь может включать отметки для успешных и неудачных запусков задач.

4. Task Duration (Длительность задач):

Этот параметр показывает, сколько времени понадобилось для выполнения каждой задачи. Это может помочь в определении узких мест в процессе и оптимизации производительности.

5. Task Tries (Попытки задач):

В этом разделе отображается количество попыток выполнения каждой задачи. Airflow автоматически повторяет выполнение задач в случае неудачи, и этот элемент позволяет пользователям увидеть, сколько раз каждая задача пыталась выполниться.

6. Landing Time (Время завершения):

Этот элемент показывает, когда была завершена ссылка (проверка) задач. Это может быть важно для анализа временных задержек и повышения эффективности обработки данных.

7. Gantt (График Ганта):

Это элемент, который показывает выполнение задач по временной шкале. Пользователи могут видеть, какие задачи выполняются параллельно, а также их взаимные зависимости, что позволяет оценить общую эффективность DAG.

8. Details (Детали):

Этот раздел предоставляет пользователям детальную информацию о конкретной задаче, включая ее состояние, параметры, лог выполнения и любые другие метаданные, которые могут быть важны для диагностики или мониторинга.

9. Code (Код):

В этом разделе можно просмотреть исходный код DAG. Это полезно для разработчиков, чтобы быстро просмотреть настройки и логику выполнения.

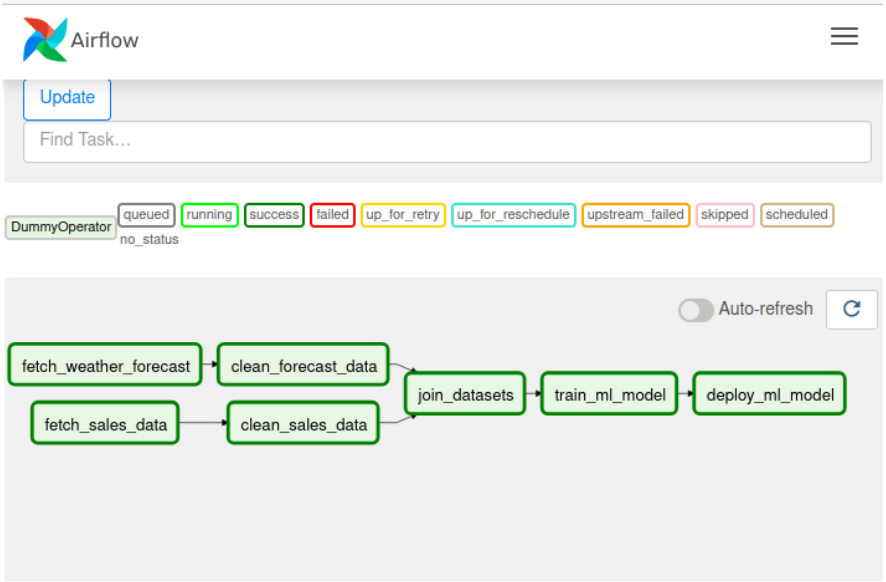
10. Audit Log (Журнал аудита):

Этот элемент сохраняет записи обо всех изменениях и событиях, происходящих в процессе выполнения DAG. Это важно для мониторинга, анализа и обеспечения безопасности.

11. Auto-refresh (Автообновление):

Эта функция автоматически обновляет интерфейс через определенные промежутки времени, что позволяет пользователям оставаться в курсе текущего состояния задач и DAG без необходимости ручного обновления страницы

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



Графический интерфейс представляет собой один из основных способов визуализации рабочих процессов (DAG - Directed Acyclic Graph) в Apache Airflow. Он предоставляет пользователям интуитивно понятный способ отслеживания состояния задач и управления ими. Вот основные элементы интерфейса Graph View:

1. Узлы (Nodes)

Каждый узел в графе представляет собой конкретную задачу (task) в рамках рабочего процесса. Узлы могут отображать различную информацию:

- **Имя задачи**: Отображается в узле.

- **Состояние задачи**: Цвет узла указывает на текущее состояние задачи.

2. Цвета узлов

Цвета узлов помогают быстро оценивать состояние задач:

- **Синий**: Задача ожидает выполнения.

- **Зеленый**: Задача успешно выполнена и завершена.

- **Красный**: Задача завершилась с ошибкой (неудача).

- **Серый**: Задача не была запущена, например, если она отключена.

- **Оранжевый**: Задача находится в состоянии "выполняется" или "обрабатывается".

- **Желтый**: Задача в состоянии "пауза" или "ожидание".

3. Связи (Dependencies)

Стрелки между узлами показывают зависимости между задачами. Это помогает понять, в каком порядке задачи должны быть выполнены. Зависимости могут быть:

- **Последовательные**: Одна задача должна завершиться, чтобы началась следующая.

- **Параллельные**: Несколько задач могут выполняться одновременно.

4. Параметры фильтрации

В графическом интерфейсе Airflow можно переключаться между различными видами представления (например, "Graph View", "Tree View" и "Gantt View"). Также есть возможность фильтровать задачи по статусу, дате выполнения и другим параметрам.

5. Всплывающие подсказки

Наведение курсора на узел задачи для вызова всплывающей подсказки, которая может содержать дополнительную информацию, такую как время выполнения, продолжительность, и другие метрики.

6. Панель инструментов

В верхней части интерфейса находится панель инструментов, позволяющая:

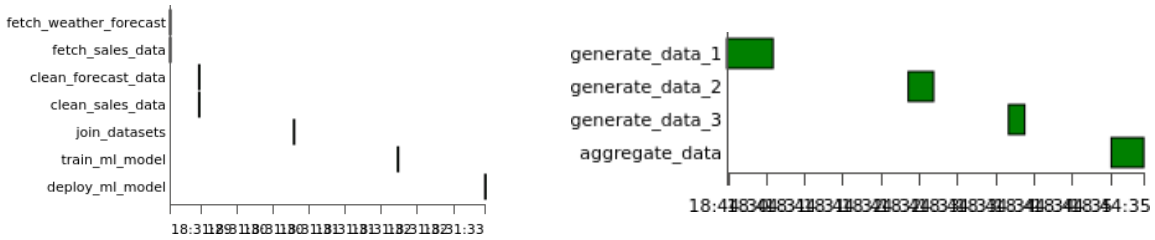
- Перезапустить DAG.

- Задавать параметры выполнения (например, выбрать дату).

- Получить доступ к журналам выполнения задач.

7. Состояние DAG

В верхней части страницы тоже может отображаться общее состояние DAG, включая общее количество задач, количество успешных и завершившихся с ошибками задач. 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 изаканчивались.

- Задачи: Каждая задача представлена горизонтальной полосой. Начало полосы соответствует времени старта задачи, а её длина указывает напродолжительность 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.

5. Выполнить индивидуальное задание.

Задание 23: Настроить DAG для периодической загрузки данных из CSV, Excel и JSON файлов в базу данных, объединения данных в единый DataFrame и последующей визуализации данных в виде диаграмм, которые сохраняются в виде изображений.

1. **Создание файла DAG**

    1. Файл CSV: данные о сотрудниках и их квалификациях (имя, квалификация, стаж).
    2. Файл Excel: данные о проектах (проект, сотрудник, часы работы).
    3. Файл JSON: данные о стоимости часа работы в зависимости от квалификации сотрудника.

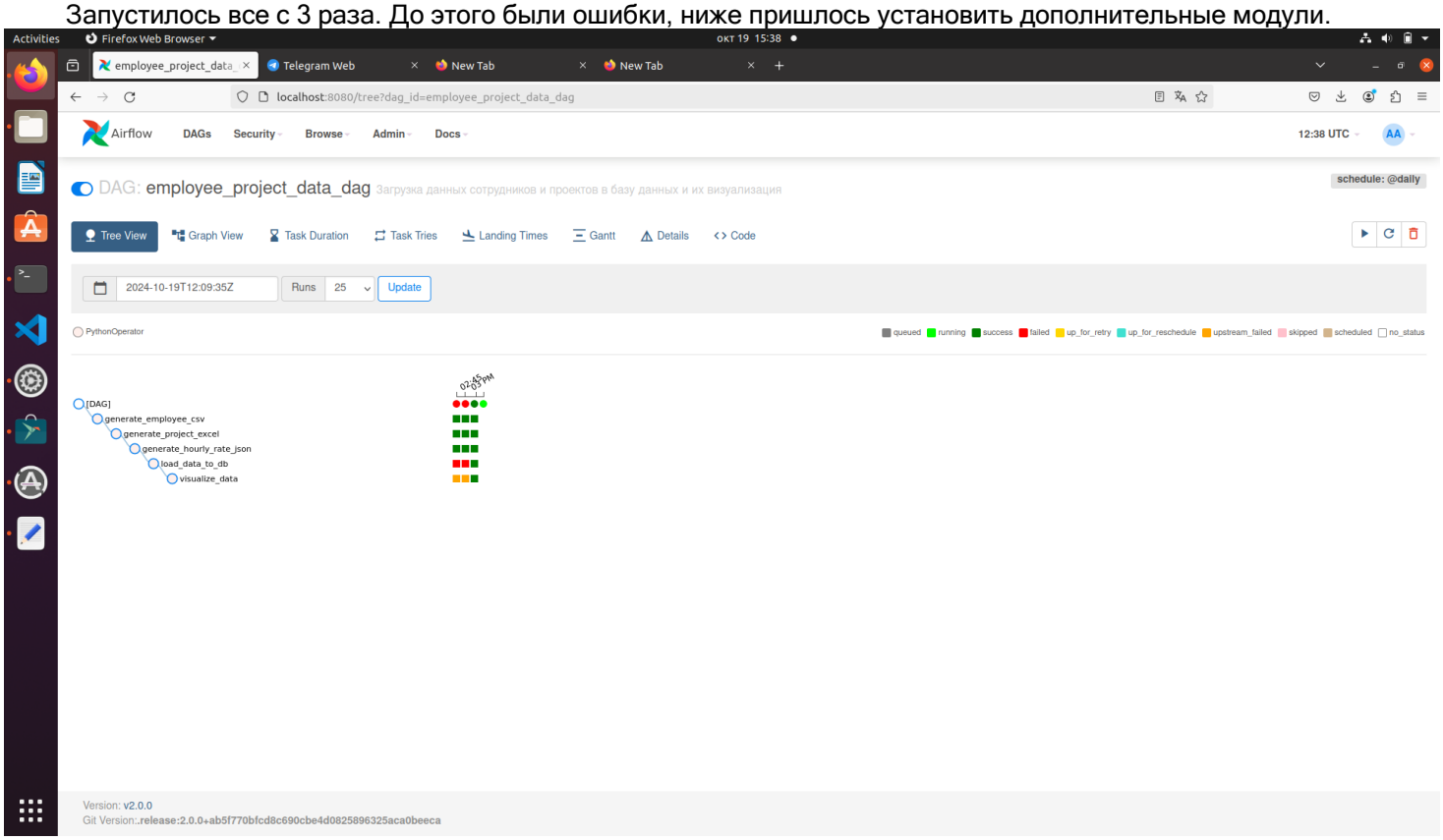
```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from datetime import datetime
import pandas as pd
import random
import json
import matplotlib.pyplot as plt
import os
from sqlalchemy import create_engine

# Генерация CSV файла с данными о сотрудниках
def generate_employee_csv():
data_csv = {
'Имя': [f'Сотрудник {i}' for i in range(1, 101)],
'Квалификация': [random.choice(['Junior', 'Middle', 'Senior']) for _ in range(100)],
'Стаж': [random.randint(1, 15) for _ in range(100)]
}
df_csv = pd.DataFrame(data_csv)
df_csv.to_csv('employees_data.csv', index=False, encoding='utf-8')

# Генерация Excel файла с данными о проектах
def generate_project_excel():
projects = [f'Проект {j}' for j in range(1, 21)]
data_excel = {
'Проект': [],
'Сотрудник': [],
'Часы работы': []
}

for project in projects:
for _ in range(5):  # Каждый проект включает 5 сотрудников
data_excel['Проект'].append(project)
data_excel['Сотрудник'].append(random.choice([f'Сотрудник {i}' for i in range(1, 101)]))
data_excel['Часы работы'].append(random.randint(1, 10))

df_excel = pd.DataFrame(data_excel)
df_excel.to_excel('projects_data.xlsx', index=False)

# Генерация JSON файла с данными о стоимости часа работы
def generate_hourly_rate_json():
data_json = {
'Квалификация': ['Junior', 'Middle', 'Senior'],
'Стоимость часа': [10, 20, 30]
}
with open('hourly_rate_data.json', 'w', encoding='utf-8') as f:
json.dump(data_json, f, ensure_ascii=False)
```

```
# Загрузка данных в базу данных
def load_data_to_db():
# Загрузка данных
df_csv = pd.read_csv('employees_data.csv', encoding='utf-8')
df_excel = pd.read_excel('projects_data.xlsx')
with open('hourly_rate_data.json', 'r', encoding='utf-8') as f:
data_json = json.load(f)
df_json = pd.DataFrame(data_json)

# Объединение данных
merged_df = pd.merge(df_excel, df_csv, on='Сотрудник', how='left')
merged_df = pd.merge(merged_df, df_json, left_on='Квалификация', right_on='Квалификация',
how='left')

# Создание соединения с базой данных
engine = create_engine('sqlite:///employees_projects.db')  # Используем SQLite
merged_df.to_sql('employee_project_data', con=engine, index=False, if_exists='replace')

# Визуализация данных
def visualize_data():
# Загрузка данных из базы
engine = create_engine('sqlite:///employees_projects.db')
df = pd.read_sql('SELECT * FROM employee_project_data', con=engine)

# Группировка данных
df_grouped = df.groupby('Квалификация').agg({'Часы работы': 'sum'}).reset_index()

# Визуализация
plt.figure(figsize=(10, 6))
plt.bar(df_grouped['Квалификация'], df_grouped['Часы работы'], color='blue')
plt.title('Общее количество часов работы по квалификации')
plt.xlabel('Квалификация')
plt.ylabel('Часы работы')
plt.savefig('hours_by_qualification.png')
plt.close()

# Определение DAG
dag = DAG('employee_project_data_dag',
description='Загрузка данных сотрудников и проектов в базу данных и их визуализация',
schedule_interval='@daily',
start_date=datetime(2023, 10, 1),
catchup=False)

# Определение задач
task_generate_employee_csv = PythonOperator(
task_id='generate_employee_csv',
python_callable=generate_employee_csv,
dag=dag
)

task_generate_project_excel = PythonOperator(
task_id='generate_project_excel',
python_callable=generate_project_excel,
dag=dag
)
```

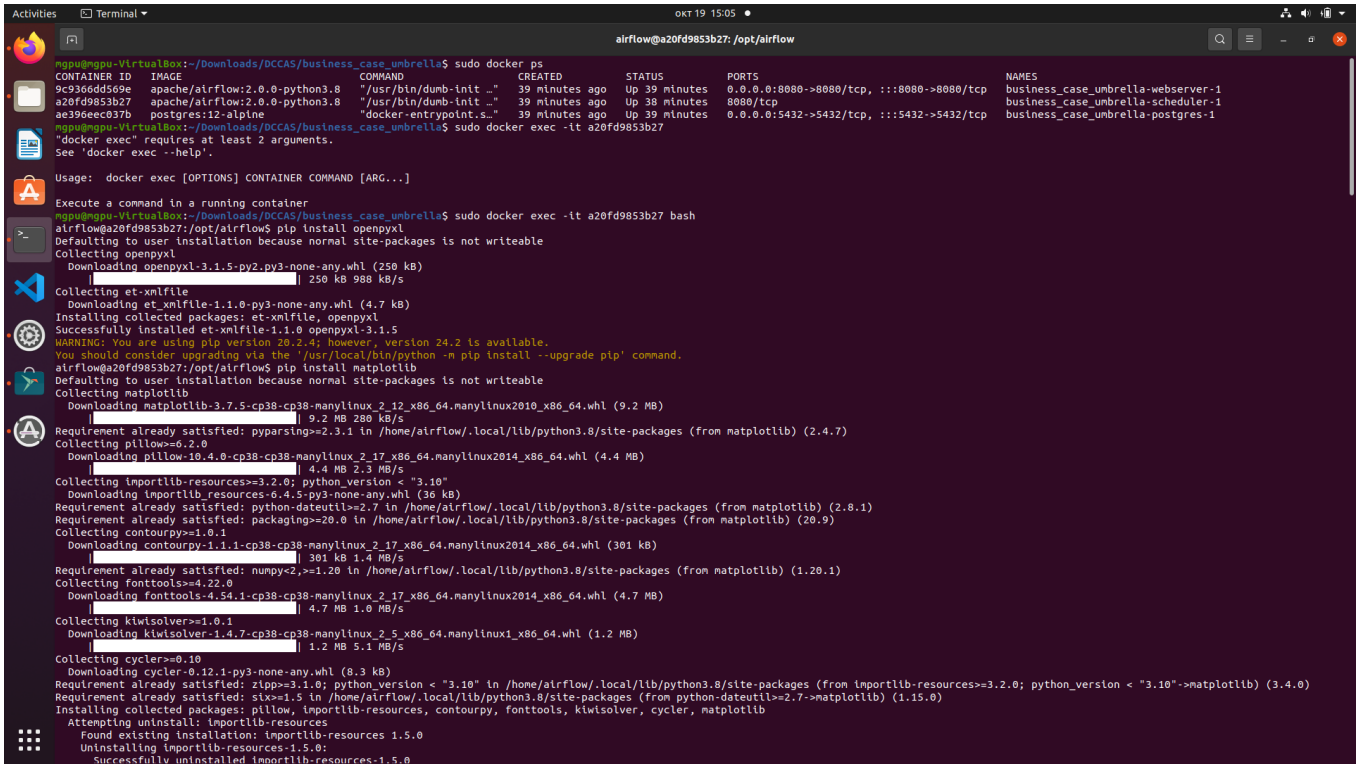
```
task_generate_hourly_rate_json = PythonOperator(
task_id='generate_hourly_rate_json',
python_callable=generate_hourly_rate_json,
dag=dag
)

task_load_data_to_db = PythonOperator(
task_id='load_data_to_db',
python_callable=load_data_to_db,
dag=dag
)

task_visualize_data = PythonOperator(
task_id='visualize_data',
python_callable=visualize_data,
dag=dag
)
```

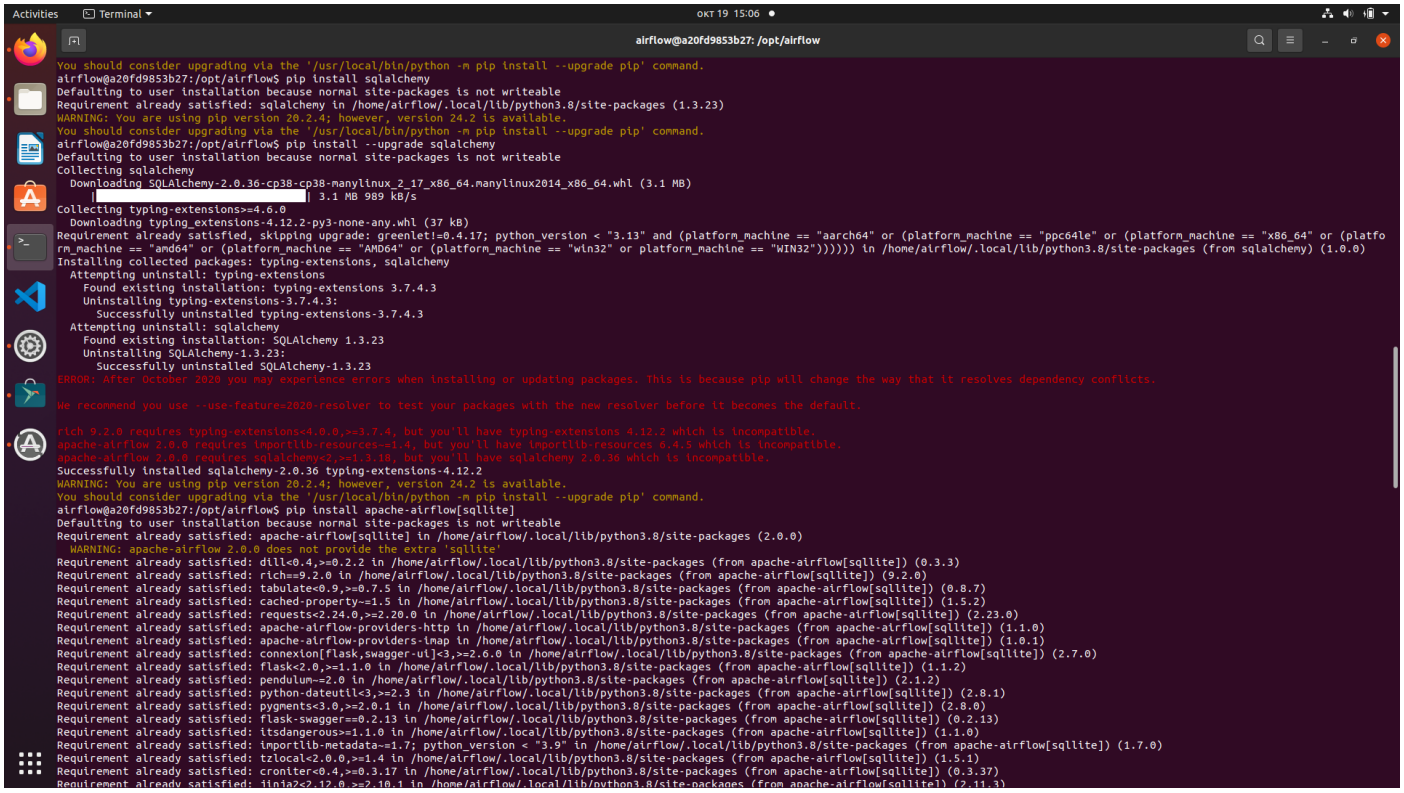
2. Запуск DAG:

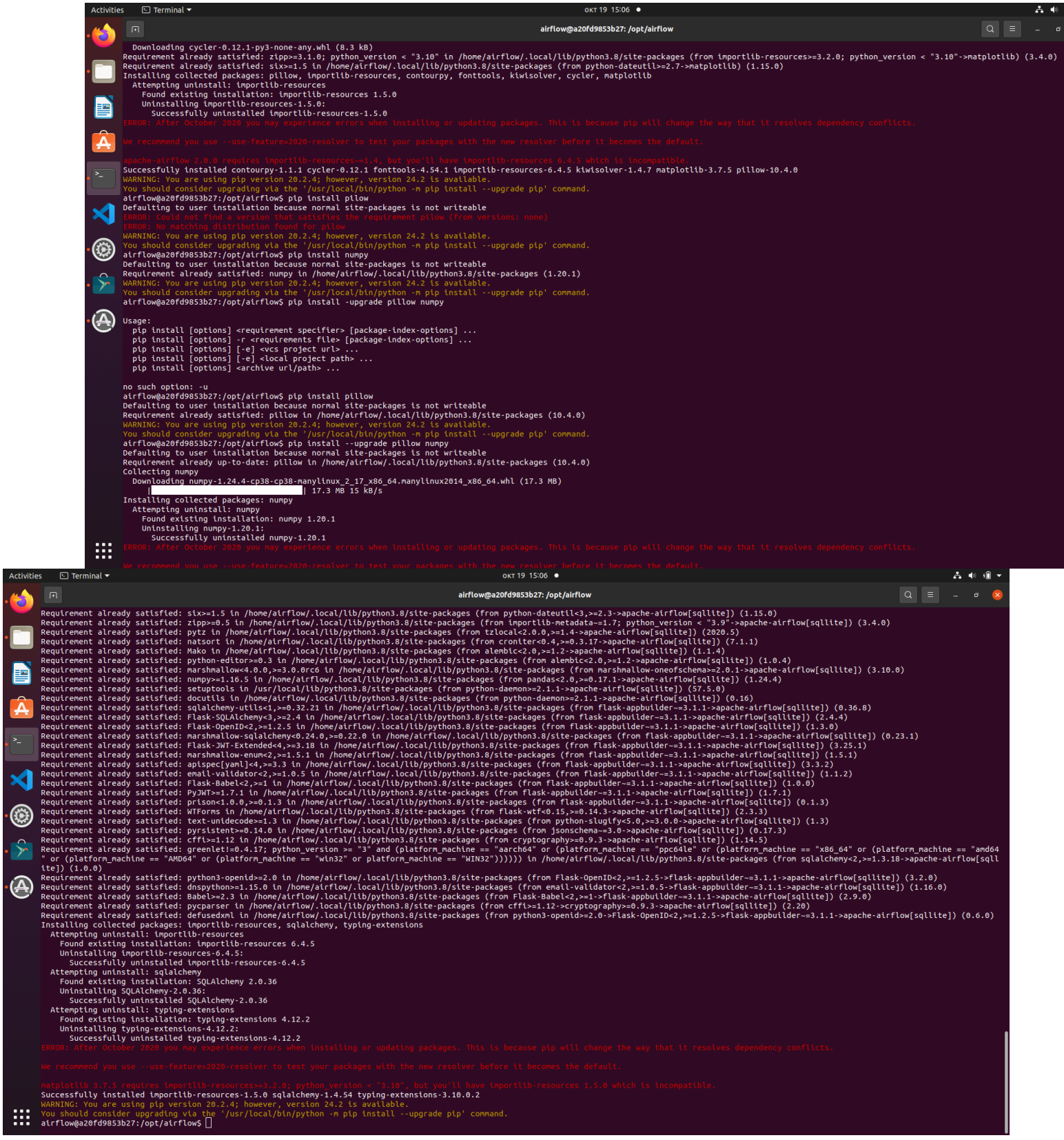
Запустилось все с 3 раза. До этого были ошибки, ниже пришлось установить дополнительные модули.



3. **Предварительно перед этим был установлен модуль openpyxl, maplotlib, sqlalchemy, apache-airflow[sqllite], pillow/numpy:**

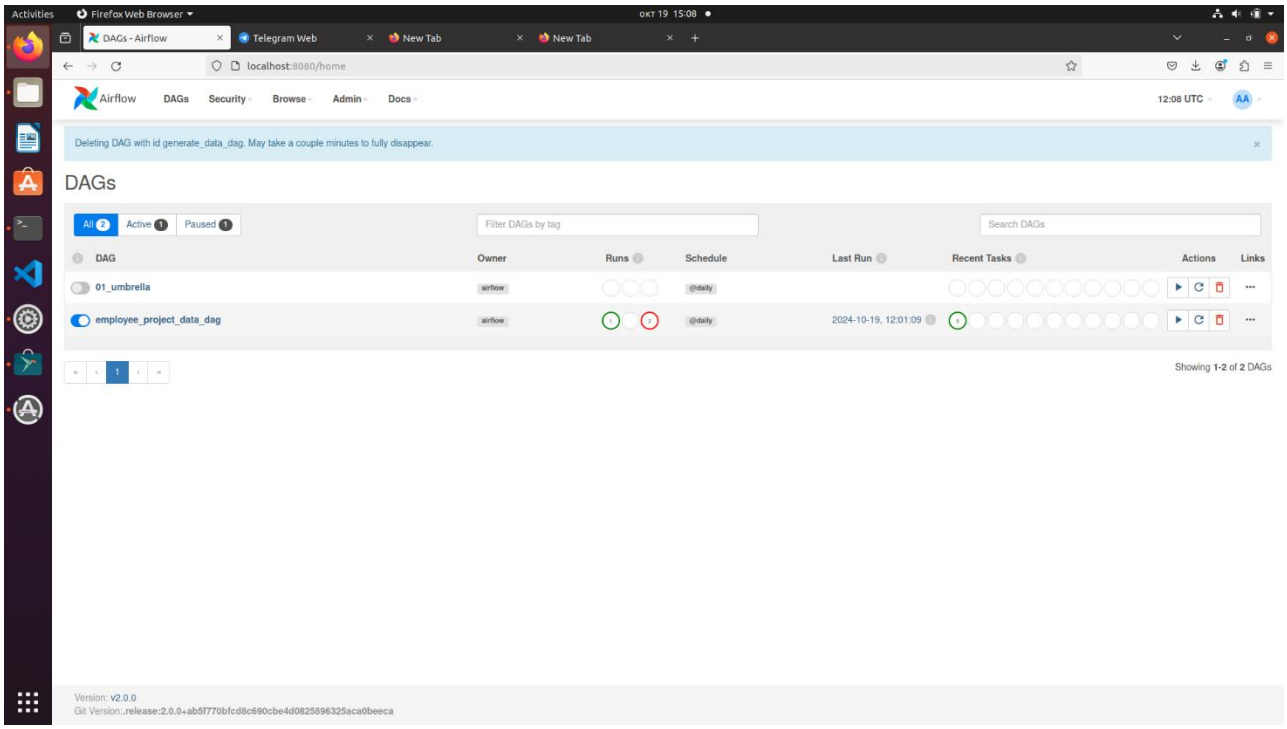






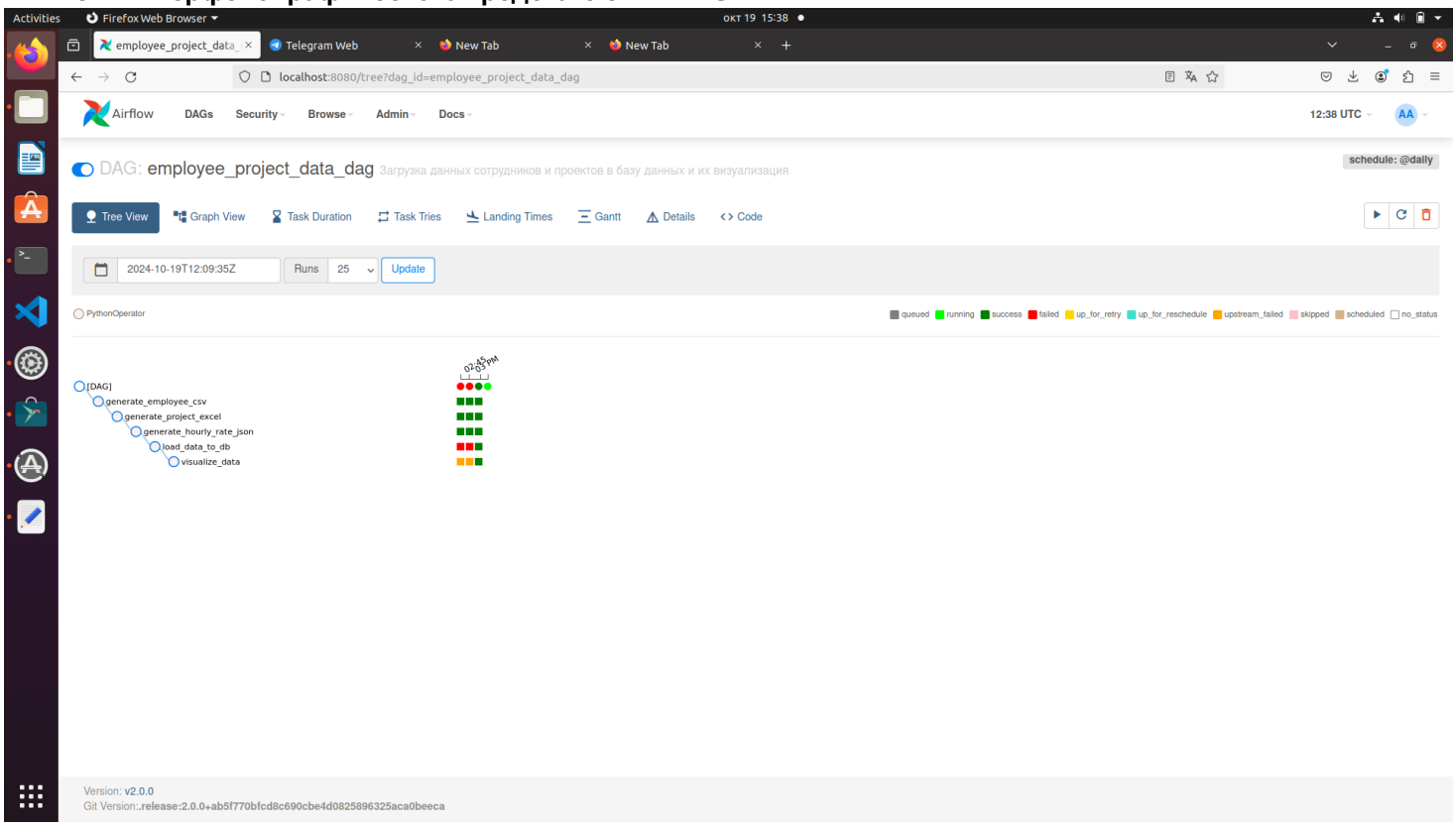
Модули были установлены, так как много раз выскакивали ошибки при запуке dag, анализировал логи, смотрел в каких местах ошибки и тем самым подгружал дополнительные библиотеки чтобы все заработало.

4. **Статус запуска DAG:**

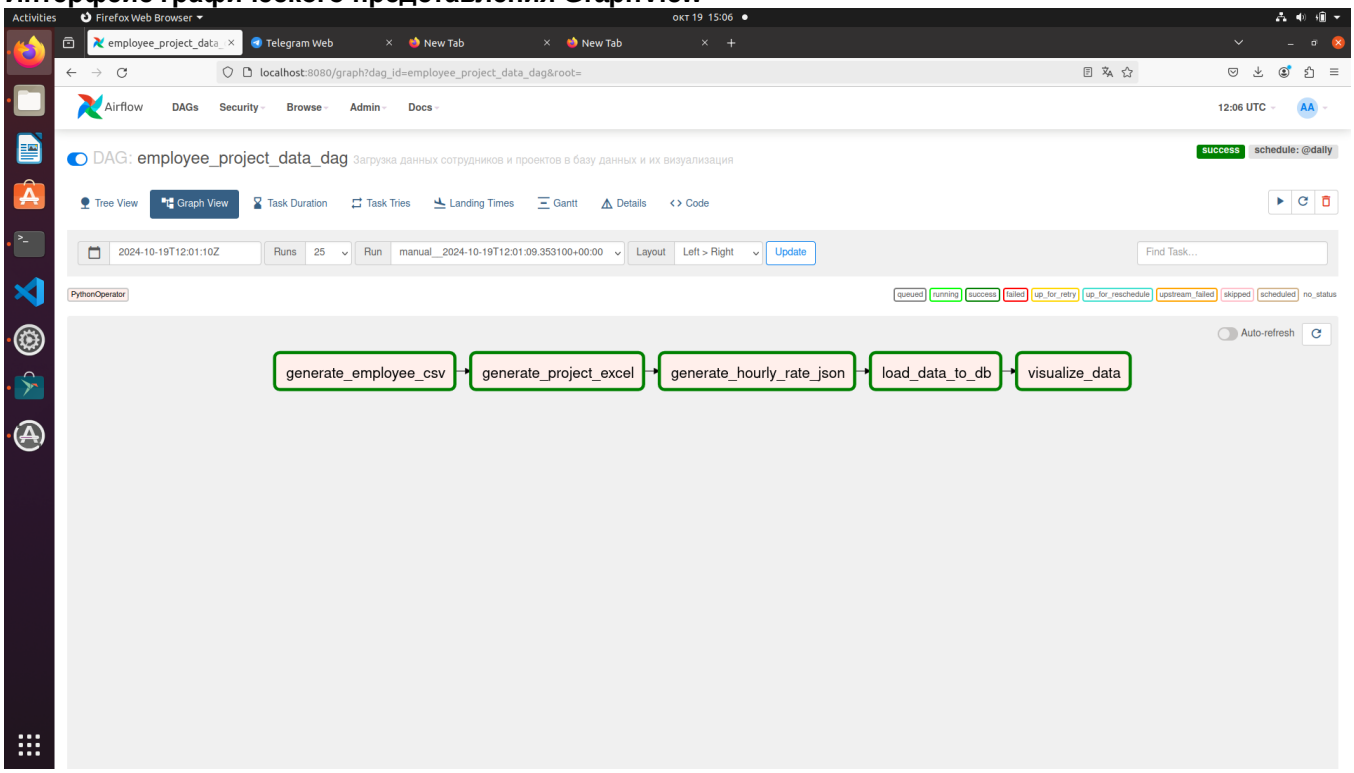


Два раза были неудачные запуски, где после установки дополнительных библиотек, корректировки кода все заработало.

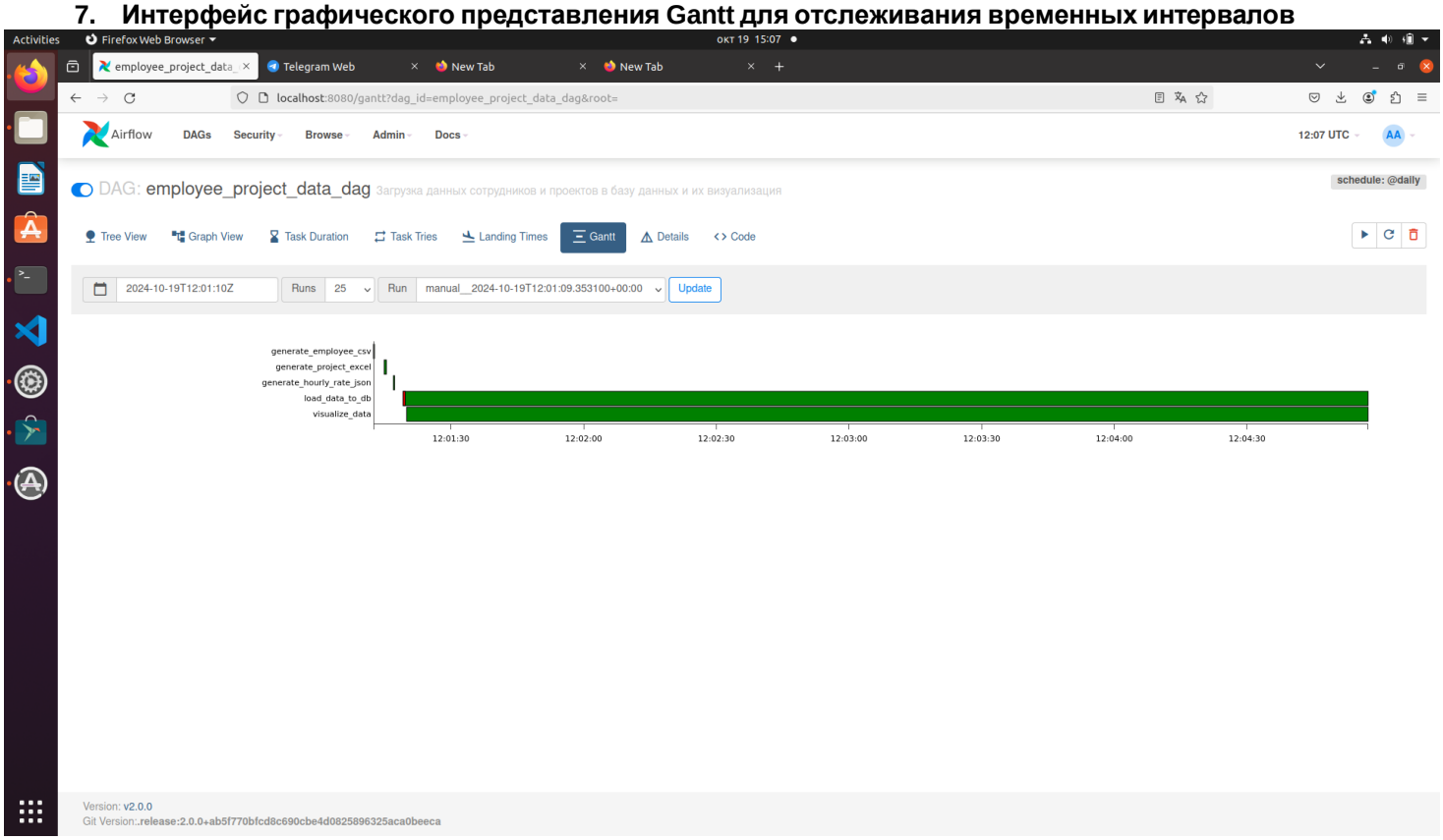
5. Интерфейс графического представления DAG :



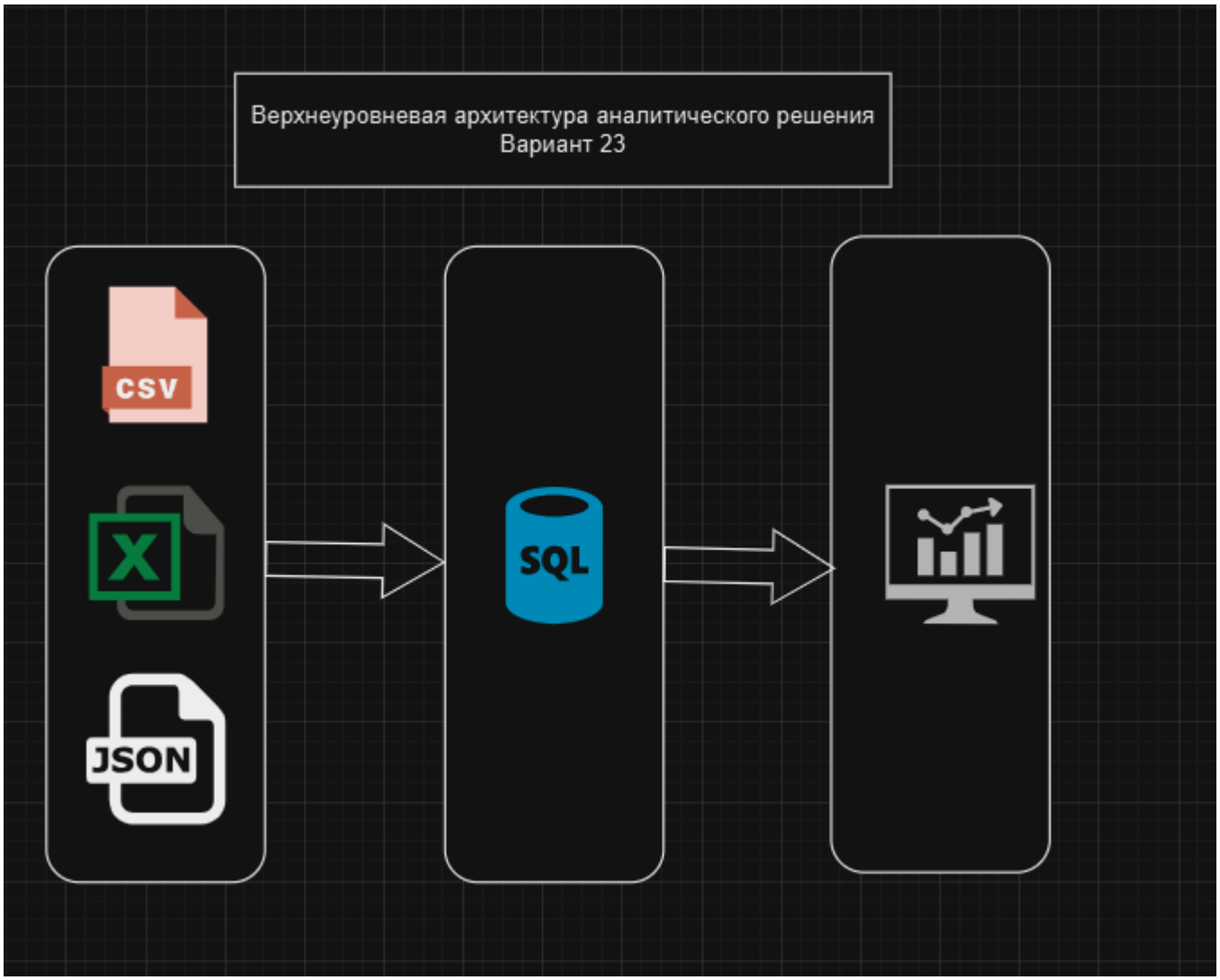
6. Интерфейс графического представления GraphView



7. Интерфейс графического представления Gantt для отслеживания временных интервалов



8.Спроектировать верхнеуровневую архитектуру аналитического решения задания 23



Вывод о проделанной работе и что получилось на выходе:

1. CSV файл с данными о сотрудниках**: Функция `generate_employee_csv` генерирует файл с данными о 100 сотрудниках, включая их имена, квалификацию и стаж. Данные варьируются с использованием `random`, что делает каждый запуск уникальным.
Excel файл с данными о проектах**: Функция `generate_project_excel` создает Excel файл с данными о 20 проектах, где каждый проект включает 5 сотрудников и случайное количество часов работы.
JSON файл с данными о стоимости часа работы**: Функция `generate_hourly_rate_json` создает JSON файл, содержащий ставки за час работы для разных уровней квалификации.

2. Функция `load_data_to_db` загружает данные из всех трех форматов (CSV, Excel, JSON), объединяет их по сотрудникам и квалификациям, а затем записывает объединенные данные в базу данных SQLite. Это обеспечивает удобное хранение и доступ к данным.

3.Функция `visualize_data` извлекает данные из базы данных, группирует их по квалификации и подсчитывает общее количество часов работы для каждого уровня квалификации. Затем она создает барграф для визуализации этих данных, сохраняет его в виде изображения и закрывает фигуру.

4.Главный DAG `employee_project_data_dag` настроен для выполнения каждый день, начиная с 1 октября 2023 года. Он состоит из пяти задач, связанных между собой по порядку их выполнения.

5.Airflow используется для автоматизации процесса загрузки и обработки данных. Это особенно полезно для периодических заданий, где нужно генерировать, загружать и визуализировать данные на регулярной основе.