

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 № 4 Тема:**

«Проектирование сквозного конвейера ETL на Python и Airflow».

Выполнила: Овсепян Милена, АДЭУ-201

Преподаватель: Босенко Т.М.

Москва

2024

Содержание

Постановка задачи	3
Решение задачи	4

Постановка задачи

Задание 4.1. Бизнес кейс «Umbrella»

4.1.1. Развернуть ВМ ubuntu_mgpu.ova в VirtualBox.

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

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

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.

4.1.5. Результаты работы представить в виде файла ФИО.pdf, выгрузить в учебный портал moodle.

Решение задачи

4.1.1. Развернуть ВМ ubuntu_mgpu.ova в VirtualBox. (рисунок 1)

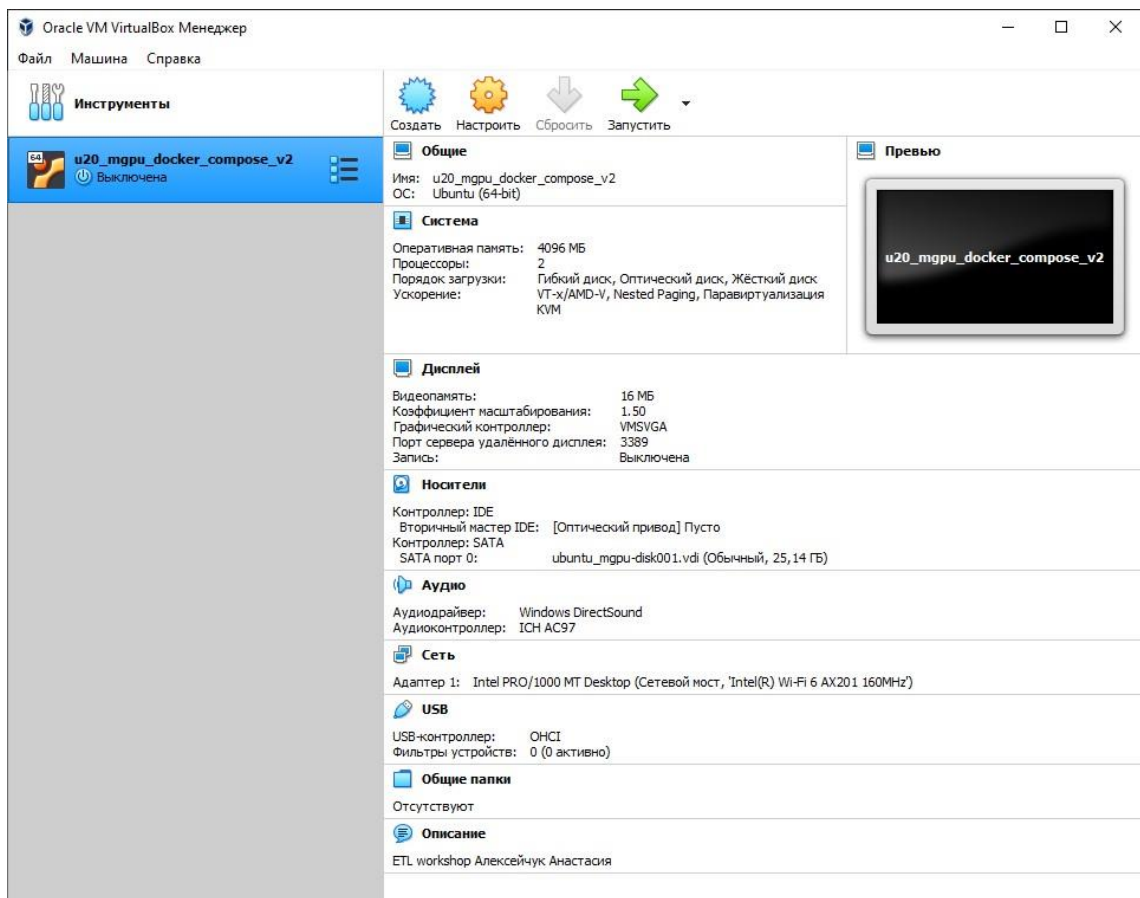


Рисунок 1 – Виртуальная машина в Oracle VM VirtualBox

4.1.2. Клонировать на ПК задание Бизнес кейс Umbrella в домашний каталог ВМ. (рисунок 2)

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

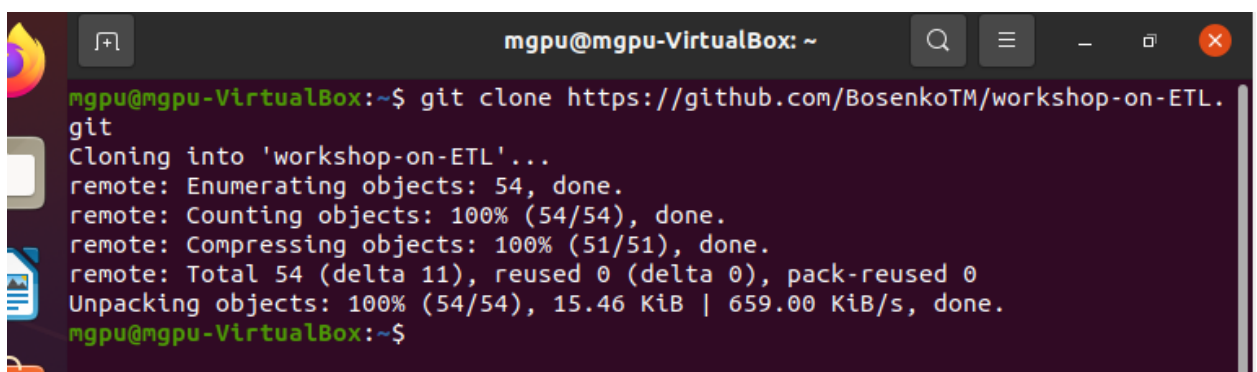


Рисунок 2 – Клонирование репозитория в виртуальную машину

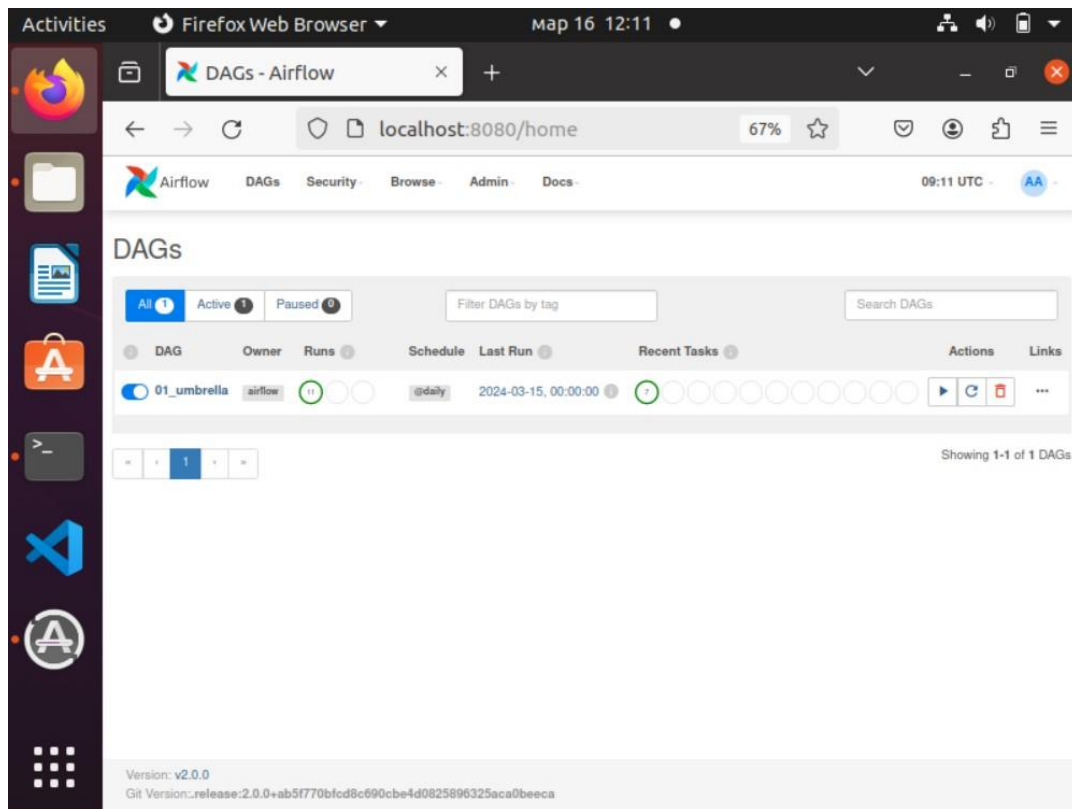


Рисунок 4 – DAG Umbrella

Описание интерфейса:

Переключатель включение/выключение DAG. По умолчанию все новые DAG – остановлены, для запуска DAG необходимо предварительно включить.

Owner — владелец/автор DAG.

Runs — состояние запусков прошлых DAG. У него есть 3 состояния:

- Success: успешно выполнен
- Running: выполняется
- Failed: есть ошибки при выполнении **Schedule**

— периодичность запуска DAG.

Last Run — дата и время последнего запуска DAG.

Recent Tasks — текущее состояние последних запусков DAG

Actions — запуск DAG вручную, обновление или удаление DAG.

Links — список быстрого доступа к просмотру кода DAG, деталей выполнения, просмотру в виде графа или диаграммы Ганта и т.д.

DAGs (Directed Acyclic Graphs) - Графы направленного ациклического связывания:

- Список всех определенных и загруженных DAG.
- Возможность управления и контроля за запуском и остановкой DAG.
- Просмотр статуса выполнения каждой конкретной DAG.

Tree View (Представление в виде дерева):

Интерактивное дерево с иерархией задач и их статусом. Позволяет легко наблюдать и управлять задачами и их зависимостями. (рисунок 5)

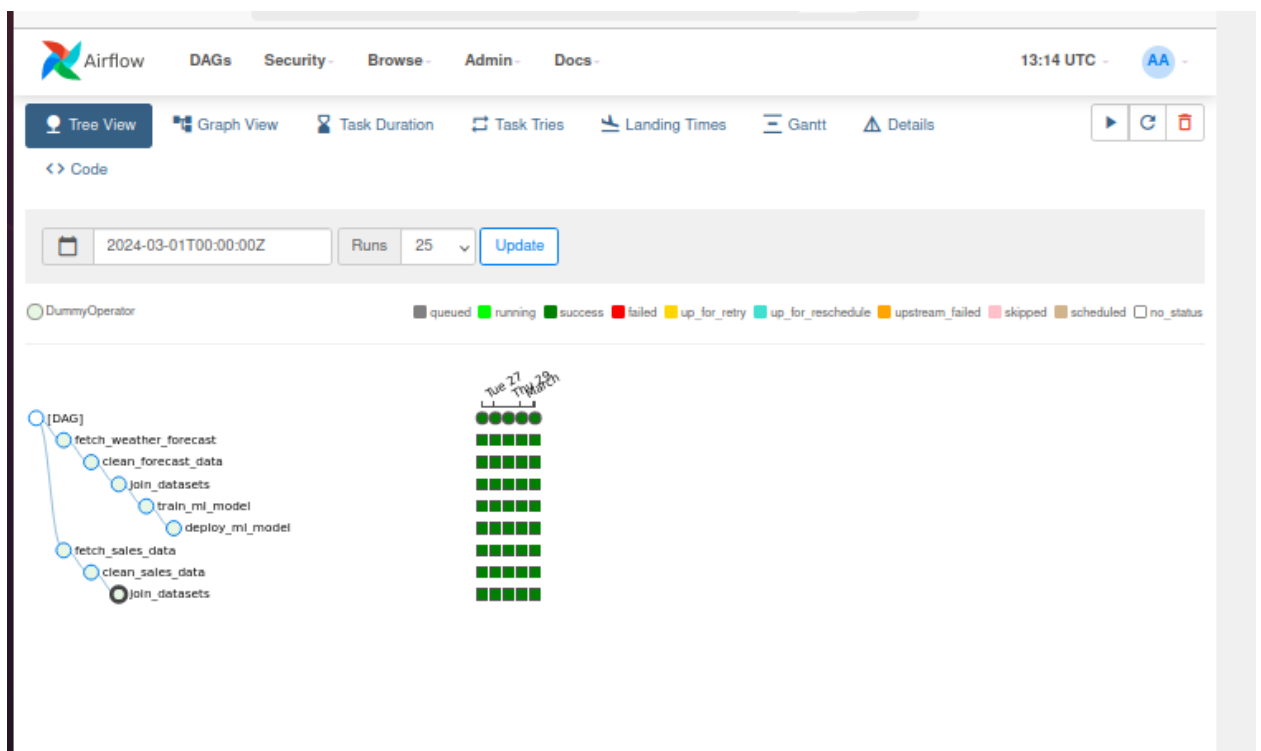


Рисунок 5 – Вкладка Tree View

Graph View (Визуальное представление):

- Визуализация структуры DAG в виде графа с зависимостями между задачами. Позволяет легко отслеживать поток выполнения задач с учетом зависимостей. (рисунок 6)

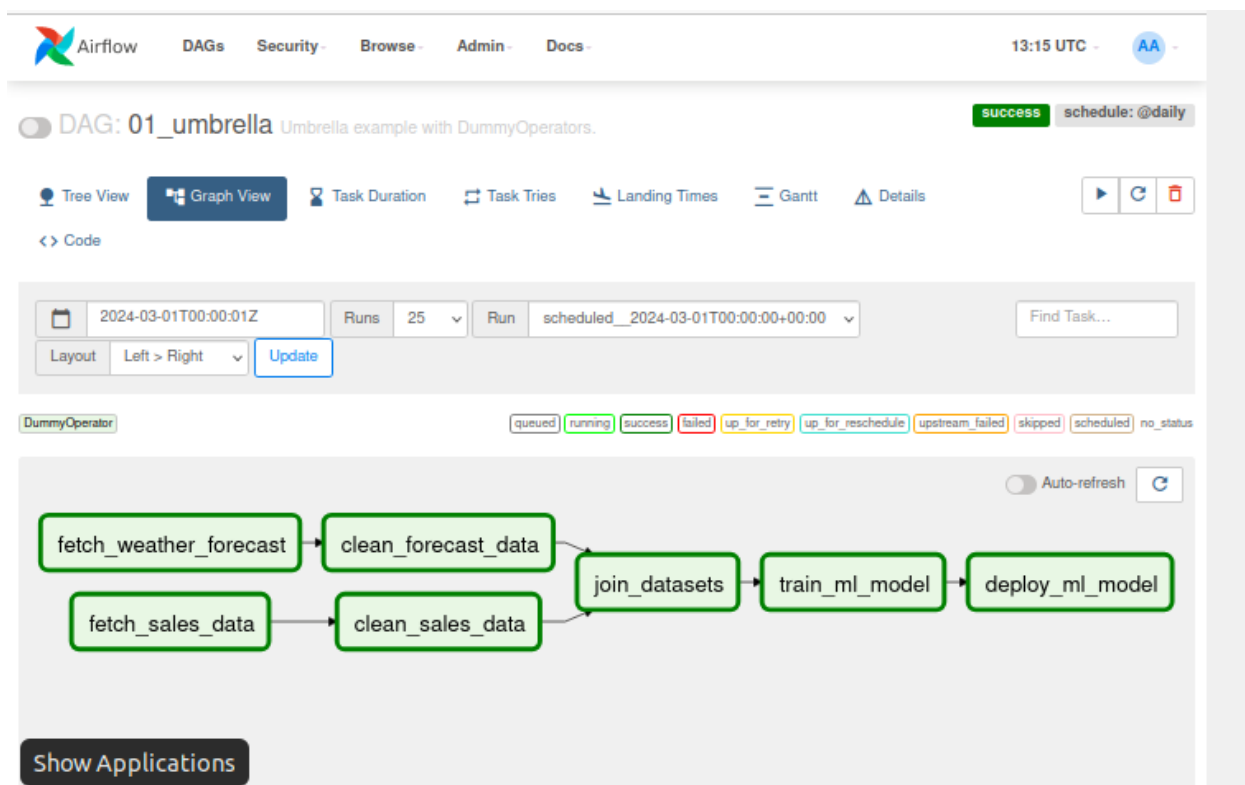


Рисунок 6 – Вкладка Graph View

Task duration (продолжительность задачи)

Продолжительность выполнения различных задач за последние N запусков. Это представление позволяет находить выбросы и быстро понимать, на что тратится время в вашей группе обеспечения доступности баз данных за многие прогоны. (рисунок 7)

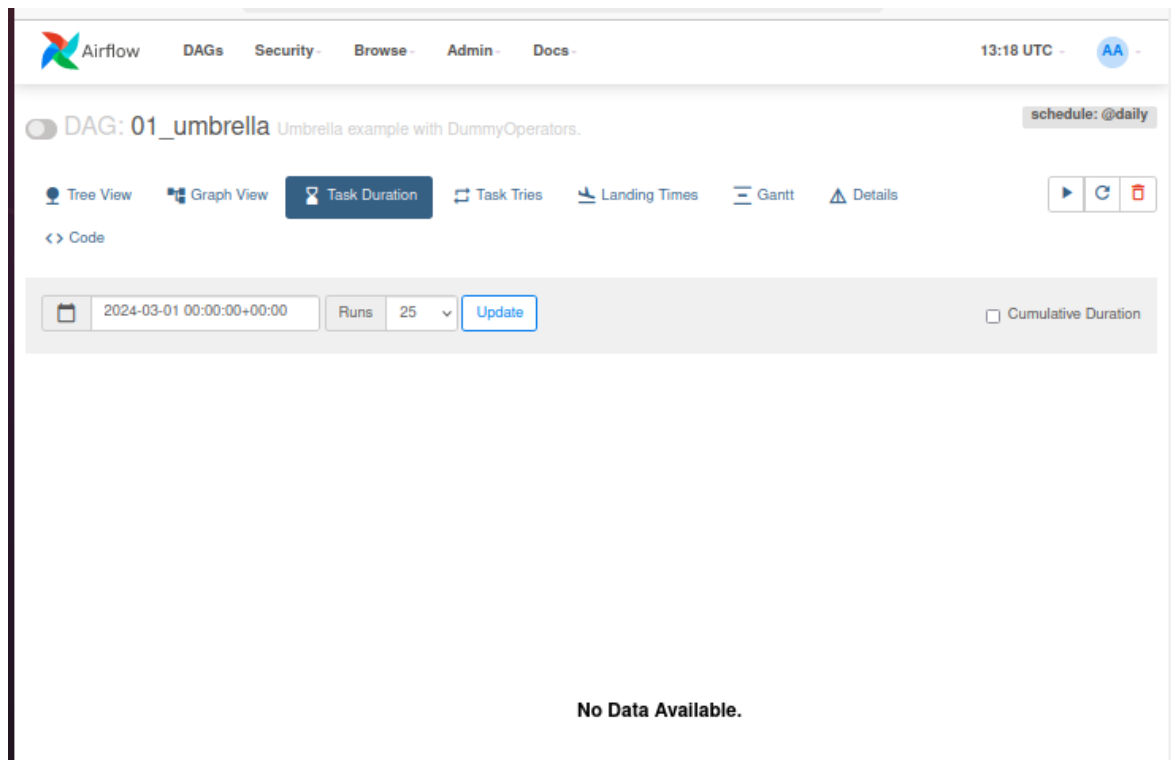


Рисунок 7 – Вкладка Task duration

Task Tries отображает количество попыток выполнения конкретной задачи (Task) в рамках DAG (Directed Acyclic Graph). Каждая строка в этой вкладке представляет одну попытку выполнения задачи. (рисунок 8)

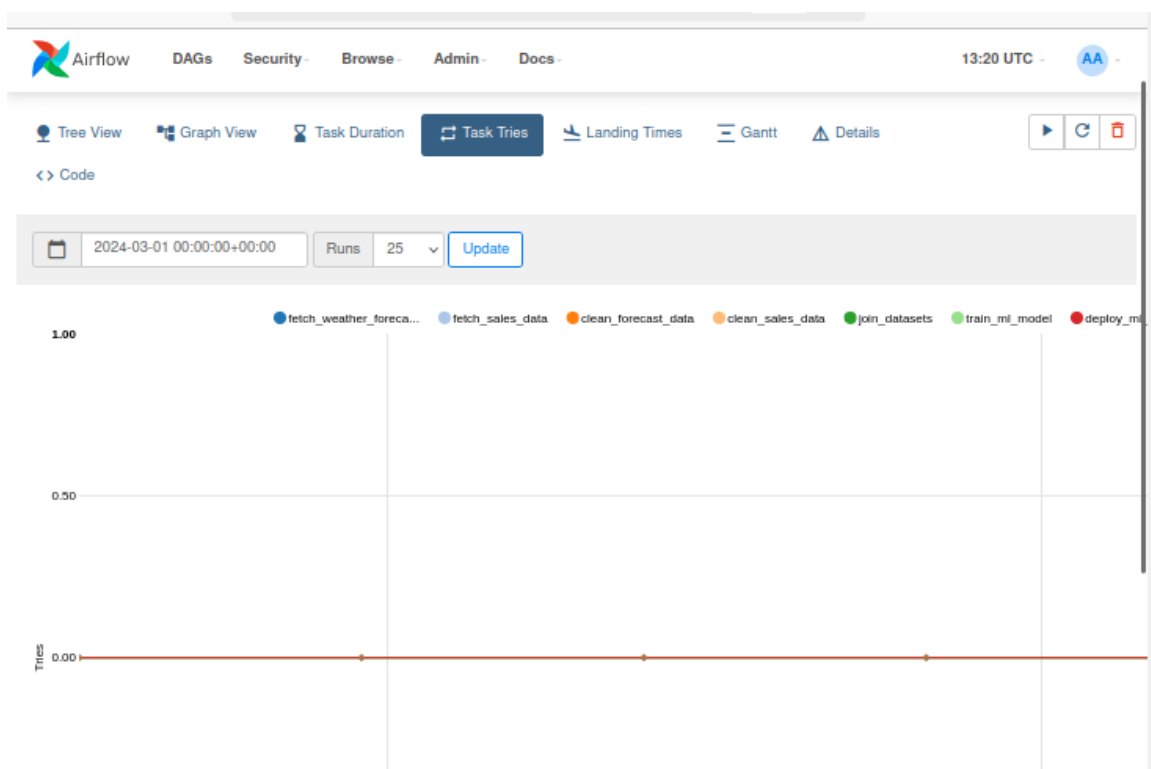


Рисунок 8 – Вкладка Task Tries

Landing Times отображает информацию о времени посадки задачи (Task) в очередь исполнения. Это время указывает на момент, когда задача была добавлена в очередь Airflow и готова к выполнению. (рисунок 9)

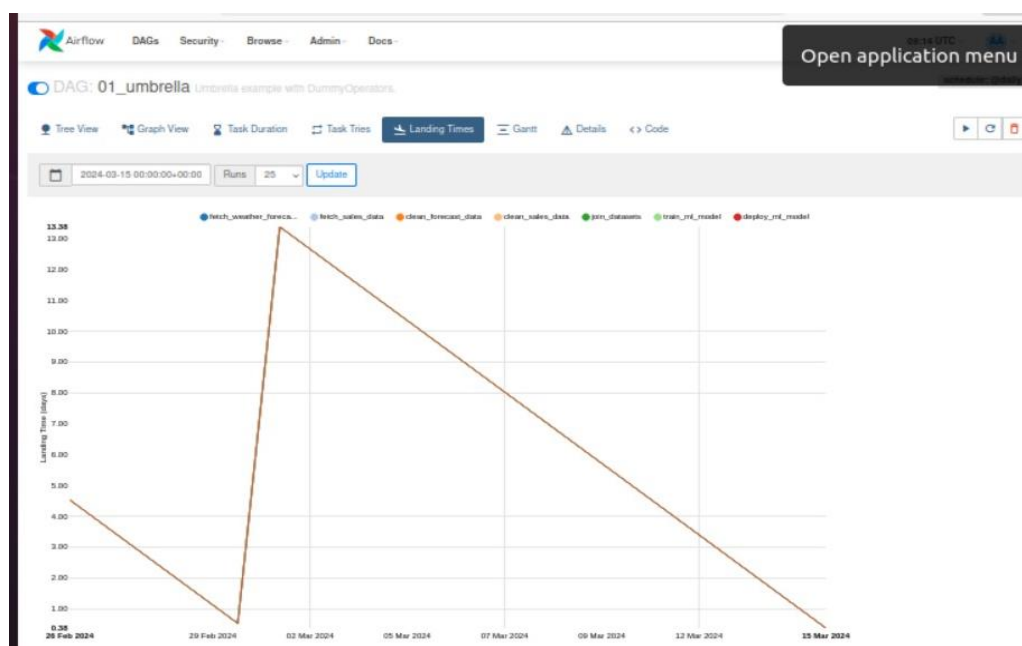


Рисунок 9 – Вкладка Landing Times

Gantt. Диаграмма Ганта позволяет анализировать длительность и перекрытие задач. Вы можете быстро определить узкие места и места, на которые тратится большая часть времени при выполнении конкретных запусков группы обеспечения доступности баз данных. (рисунок 10)

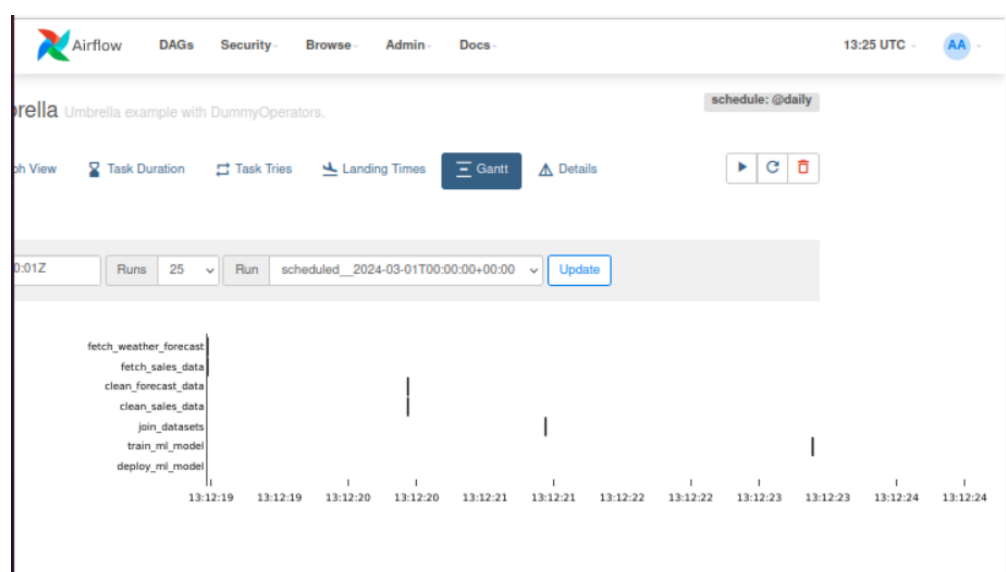


Рисунок 10 – Вкладка Gantt

Details (детали) предоставляет дополнительную информацию о задаче (Task) или даге (DAG). В этой вкладке можно найти различные аспекты и параметры, касающиеся задачи или DAG, что обеспечивает более глубокое понимание и контроль над их выполнением. (рисунок 11)

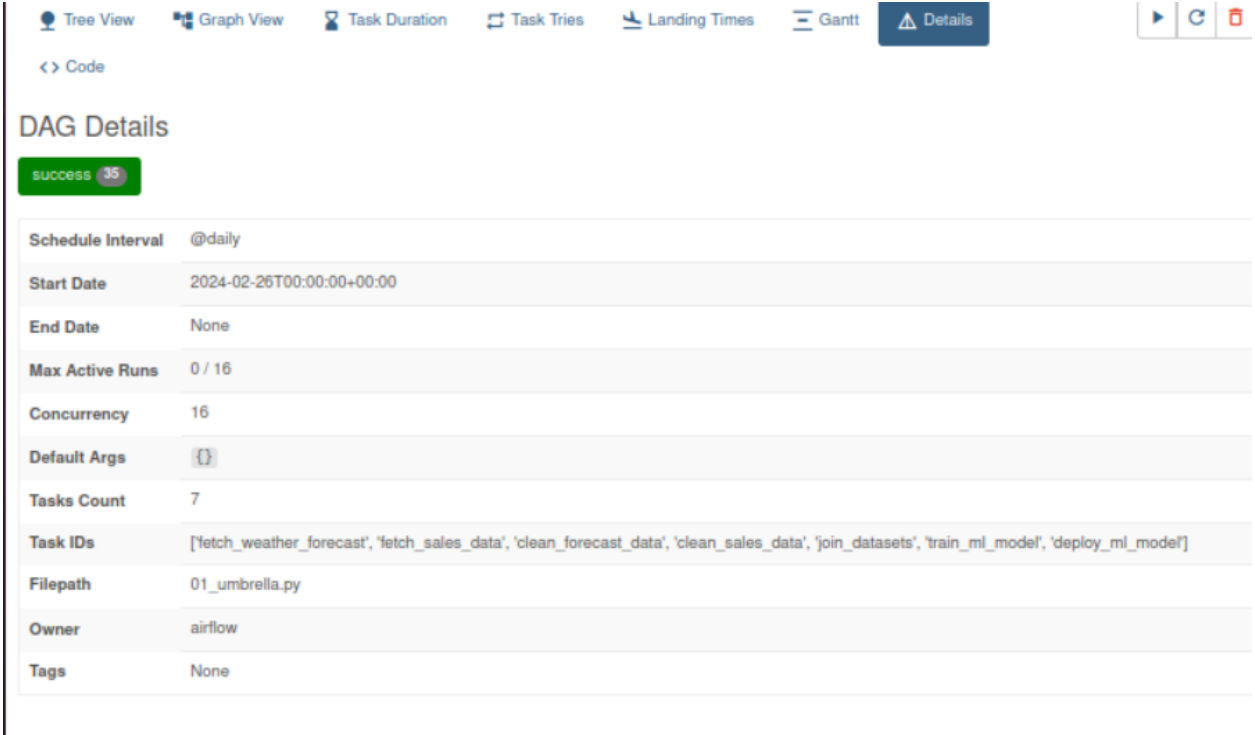
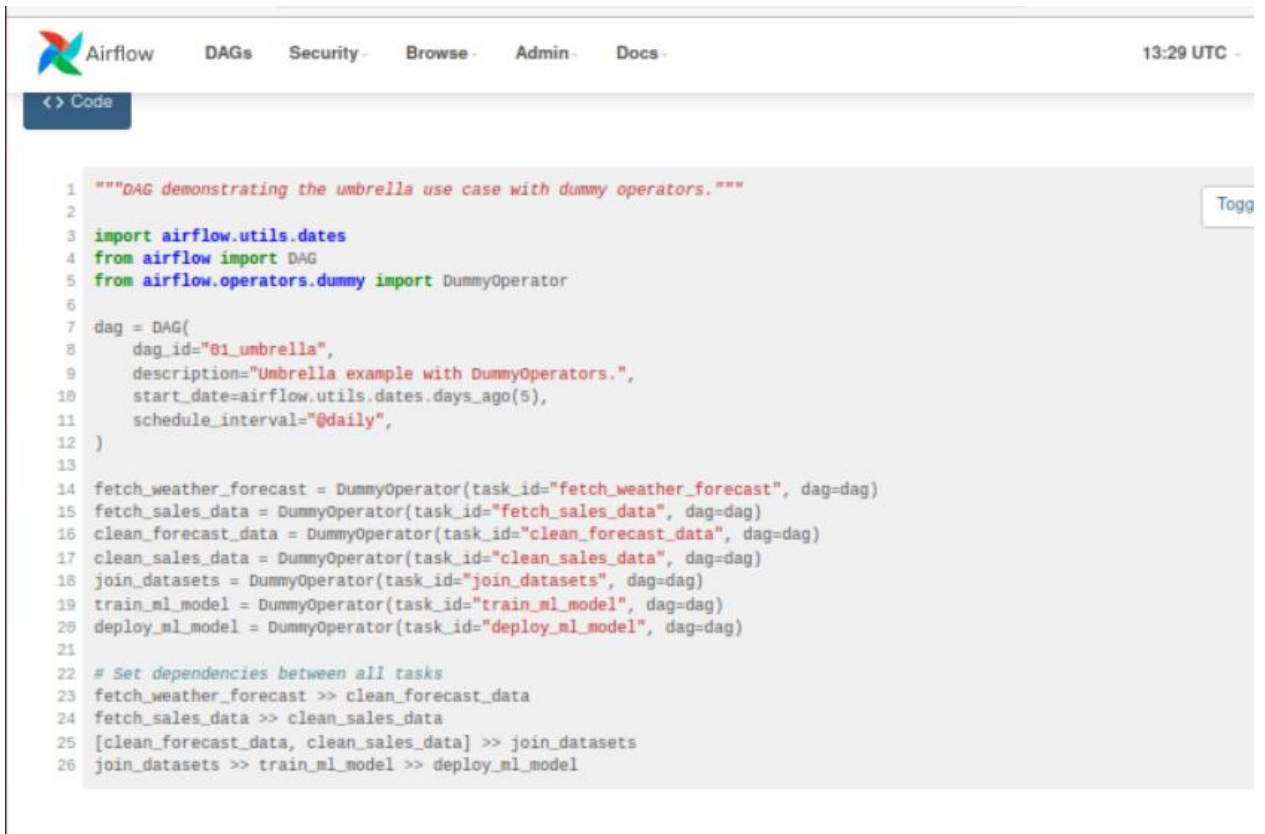


Рисунок 11 – Вкладка Details

Code. Дает возможность посмотреть код конвейера данных. Прозрачность – это все. Хотя код конвейера находится в системе контроля версий, это быстрый способ получить доступ к коду, который генерирует группу обеспечения доступности баз данных, и предоставить еще больше контекста. (рисунок 12)



The screenshot shows the Apache Airflow web interface. At the top, there is a navigation bar with links for DAGs, Security, Browse, Admin, and Docs. The current time is 13:29 UTC. Below the navigation bar, there is a tab labeled 'Code'. The code editor displays the following Python code:

```
1 """DAG demonstrating the umbrella use case with dummy operators."""
2
3 import airflow.utils.dates
4 from airflow import DAG
5 from airflow.operators.dummy import DummyOperator
6
7 dag = DAG(
8     dag_id="01_umbrella",
9     description="Umbrella example with DummyOperators.",
10    start_date=airflow.utils.dates.days_ago(5),
11    schedule_interval="@daily",
12 )
13
14 fetch_weather_forecast = DummyOperator(task_id="fetch_weather_forecast", dag=dag)
15 fetch_sales_data = DummyOperator(task_id="fetch_sales_data", dag=dag)
16 clean_forecast_data = DummyOperator(task_id="clean_forecast_data", dag=dag)
17 clean_sales_data = DummyOperator(task_id="clean_sales_data", dag=dag)
18 join_datasets = DummyOperator(task_id="join_datasets", dag=dag)
19 train_ml_model = DummyOperator(task_id="train_ml_model", dag=dag)
20 deploy_ml_model = DummyOperator(task_id="deploy_ml_model", dag=dag)
21
22 # Set dependencies between all tasks
23 fetch_weather_forecast >> clean_forecast_data
24 fetch_sales_data >> clean_sales_data
25 [clean_forecast_data, clean_sales_data] >> join_datasets
26 join_datasets >> train_ml_model >> deploy_ml_model
```

Рисунок 12 - CODE

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.

Верхнеуровневая архитектура аналитического решения задания Бизнес кейс Umbrella представлена на рисунке 13:

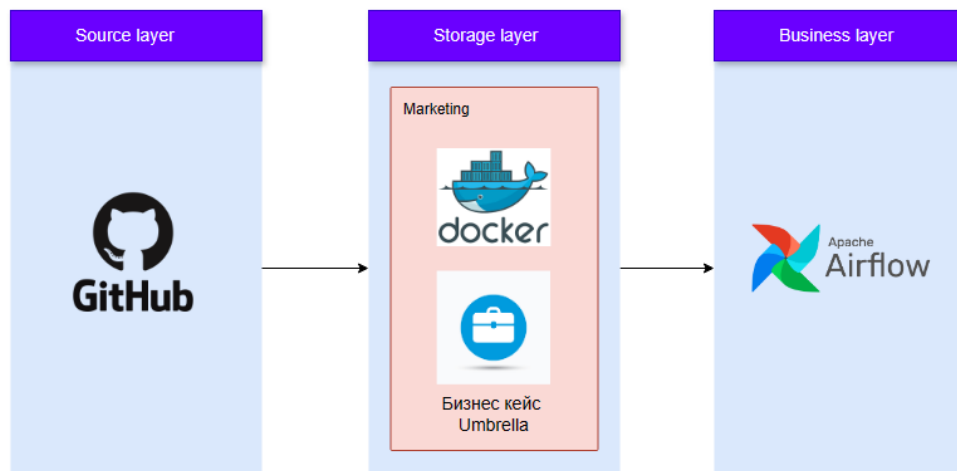


Рисунок 13 – Верхнеуровневая архитектура аналитического решения

Более подробно с архитектурой можно ознакомиться по ссылке:

https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=_blank&layers=1&nav=1&title=%D0%9E%D0%B2%D1%81%D0%B5%D0%BF%D1%8F%D0%BD_%D0%BF%D1%804#Uhttps%3A%2F%2Fdrive.google.com%2Fuc%3Fid%3D1vl1OxoYmLrCCBcOKC5aRjCOplztwuJOe%26export%3Ddownload