

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

Вебинар 28 марта

Выполнил(а): st_105

Москва

2025

Бизнес-кейс «Rocket»

Вариант 15

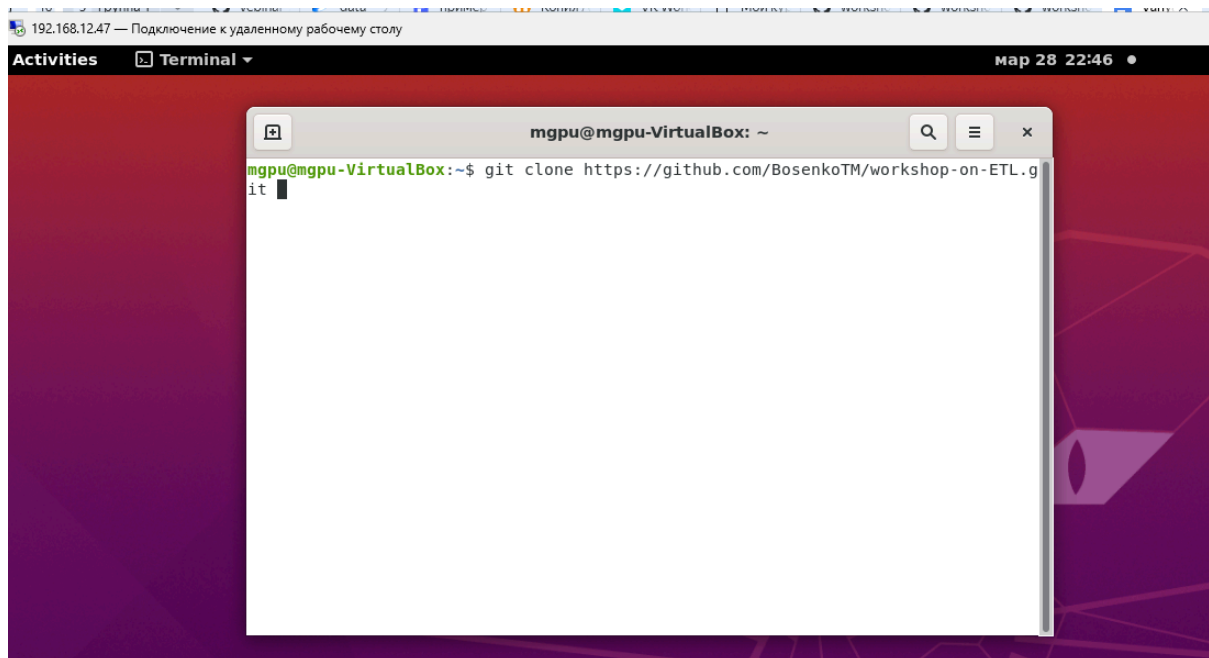
Настроить DAG для
еженедельного
скачивания новых
изображений ракет

Проанализировать
процесс подключения к
серверу для получения
изображений

Разработать
автоматическую
обработку ошибок при
отсутствии
изображений

1. Развертывание ВМ в VirtualBox

2. Скачивание и настройка проекта



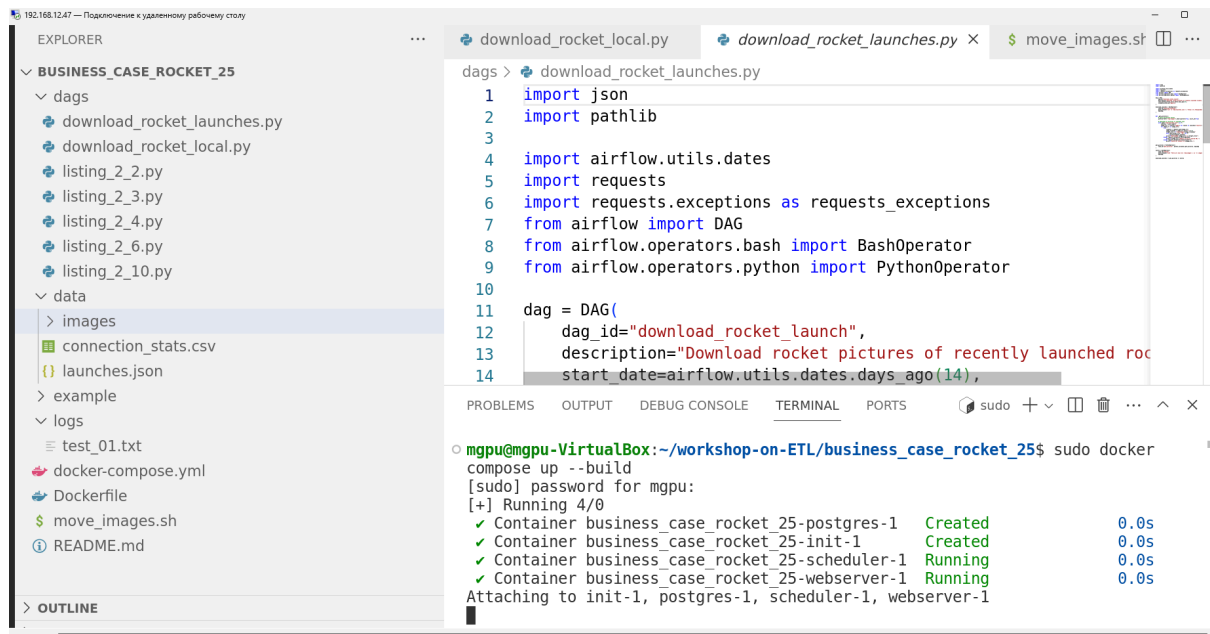
Запуск airflow с помощью команд:

```
sudo docker build -t custom-airflow:slim-2.8.1-python3.11 .
```

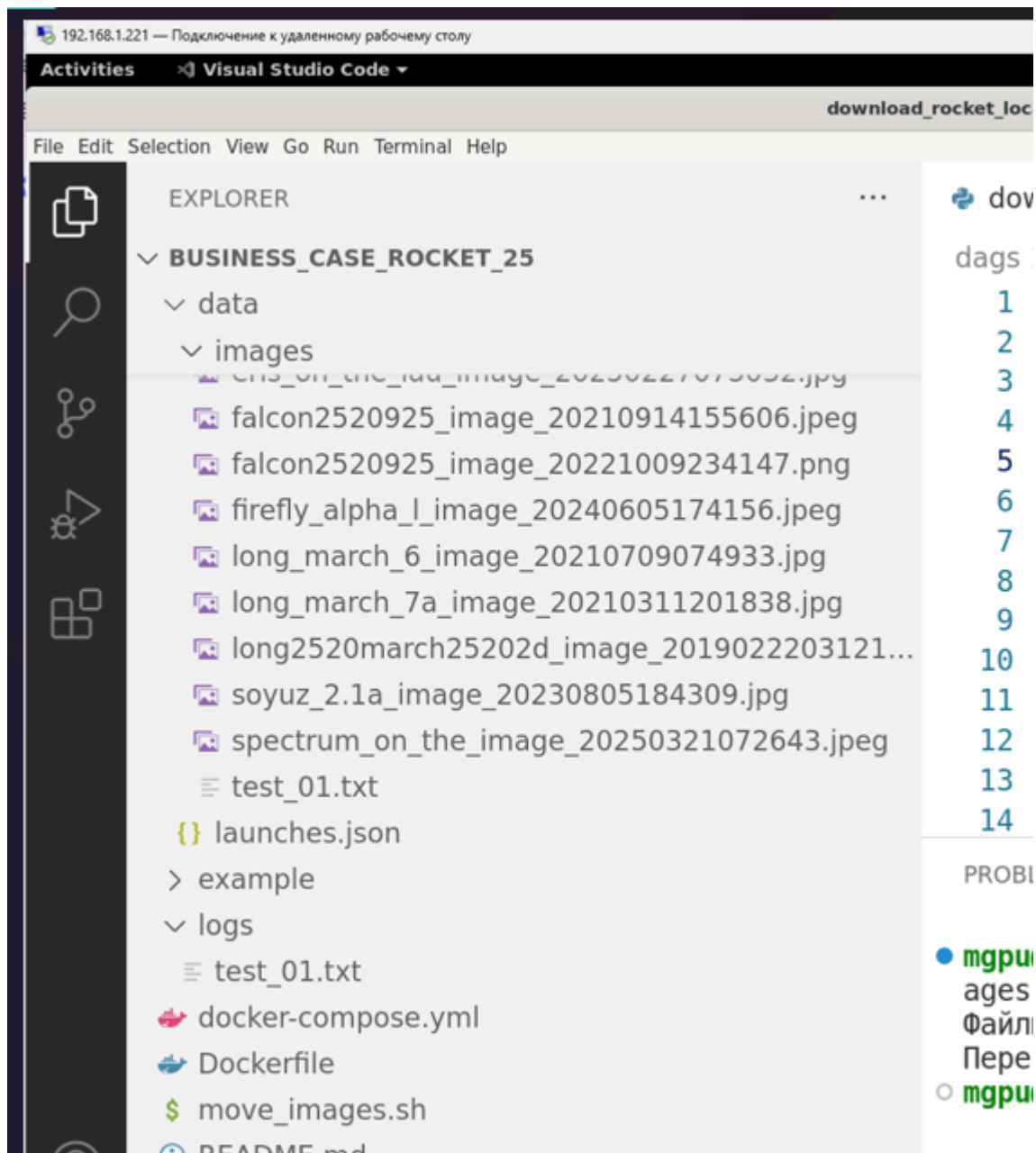
```
sudo docker compose up --build
```

Меняем владельца папки:

```
sudo chown -R 50000:50000 ./data
```



Проверка работоспособности DAG файла, изображения появились в папке images



Индивидуальное задание 1:

Настроить DAG для еженедельного скачивания новых изображений ракет
Вносим изменения в файл DAG

```

dag = DAG(
    dag_id="rocket_images_weekly_download_v2",
    description="Еженедельное скачивание изображений рак
    schedule_interval="@weekly",
    default_args=default_args,
    catchup=False,
    tags=['rocket', 'images', 'download', 'monitoring'],
)

```

Индивидуальное задание 2:

Проанализировать процесс подключения к серверу для получения изображений

Добавляем в DAG возможность записывать в файл csv время подключения к серверу

```

def analyze_and_log_connection():
    test_url = "https://il.thespacedevs.com/2.0.0/launch/upcoming"
    attempts = 3
    timeouts = []
    successful_attempts = 0
    stats = {
        'timestamp': datetime.now().isoformat(),
        'attempts': attempts,
        'successful_attempts': 0,
        'avg_response_time': 0,
        'max_response_time': 0,
        'min_response_time': 0,
        'status': 'FAILED'
    }

    for i in range(attempts):
        try:
            start_time = time.time()
            response = requests.get(
                test_url,
                timeout=(5, 10),
                headers={'User-Agent': 'Airflow DAG tester/1.0'}
            )
            response_time = time.time() - start_time
            timeouts.append(response_time)

            if response.status_code == 200:
                successful_attempts += 1
                print(f"Попытка {i+1}: Успешно. Время ответа: {response_time:.2f}c")
            else:
                print(f"Попытка {i+1}: Ошибка. Код статуса: {response.status_code}")

        except requests.exceptions.RequestException as e:
            print(f"Попытка {i+1}: Ошибка подключения - {str(e)}")
            timeouts.append(10) # Максимальный таймаут

    if successful_attempts > 0:
        stats.update({

```

```

'successful_attempts': successful_attempts,
'avg_response_time': round(mean(timeouts), 2),
'max_response_time': round(max(timeouts), 2),
'min_response_time': round(min(timeouts), 2),
'status': 'SUCCESS' if successful_attempts == attempts else 'PARTIAL'
})

print(f"Статистика подключения: {stats}")

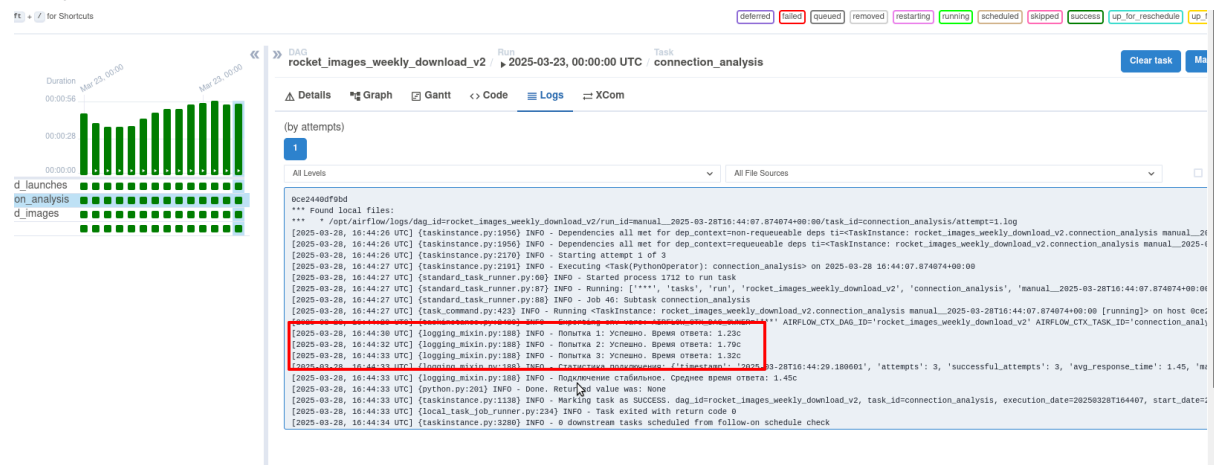
# Сохраняем статистику в CSV
save_connection_stats_to_csv(stats)

avg_time = stats['avg_response_time']
if avg_time > 5:
    print(f"Предупреждение: Высокое среднее время ответа ({avg_time:.2f}c)")
else:
    print(f"Подключение стабильное. Среднее время ответа: {avg_time:.2f}c")
else:
    save_connection_stats_to_csv(stats)
    raise AirflowSkipException("Все попытки подключения провалились, пропускаем загрузку
изображений")

connection_analysis = PythonOperator(
    task_id="connection_analysis",
    python_callable=analyze_and_log_connection,
    dag=dag,
)

```

Результат



Было совершено 14 запусков Dag с помощью триггера

В статистику сохранилось 14 записей

data	connection_stats.csv
7	2025-03-28T16:42:49.040441,3,3,1.09,1.31,0.83,SUCCESS
8	2025-03-28T16:43:25.977577,3,3,1.31,1.46,1.07,SUCCESS
9	2025-03-28T16:43:27.812572,3,3,1.52,2.07,1.23,SUCCESS
10	2025-03-28T16:43:28.235822,3,3,1.53,1.79,1.16,SUCCESS
11	2025-03-28T16:44:25.772624,3,3,1.38,1.62,1.01,SUCCESS
12	2025-03-28T16:44:26.092196,3,3,1.3,1.52,1.03,SUCCESS
13	2025-03-28T16:44:26.012836,3,3,1.92,2.13,1.57,SUCCESS
14	2025-03-28T16:44:29.204723,3,3,1.38,1.53,1.28,SUCCESS
15	2025-03-28T16:44:29.180601,3,3,1.45,1.79,1.23,SUCCESS
16	

Далее Файл csv проанализирован в google colab

https://colab.research.google.com/drive/1gjG-3AD4Rm5Pf7Ii8D9AF02ZD_1s8mLK#scrollTo=oT7Ffje7-2PU

Индивидуальное задание 3:

Разработать автоматическую обработку ошибок при отсутствии изображений

3. Задача для загрузки изображений с обработкой ошибок

```
def download_images_with_error_handling():
    images_dir = "/opt/airflow/data/images"
    pathlib.Path(images_dir).mkdir(parents=True, exist_ok=True)

    try:
        with open("/opt/airflow/data/launches.json") as f:
            launches = json.load(f)

            if not isinstance(launches.get("results"), list):
                raise AirflowSkipException("Некорректный формат JSON: отсутствует список results")

            if not launches["results"]:
                raise AirflowSkipException("Нет данных о запусках в JSON")

            image_urls = []
            for launch in launches["results"]:
                if isinstance(launch, dict) and launch.get("image"):
                    image_urls.append(launch["image"])

            if not image_urls:
                raise AirflowSkipException("В данных о запусках отсутствуют URL изображений")

            success_count = 0
            for idx, url in enumerate(image_urls, 1):
                try:
                    if not url.startswith(('http://', 'https://')):
                        print(f"Некорректный URL: {url}")
                        continue

                    response = requests.get(
                        url,
                        timeout=(10, 30),
                        stream=True,
                        headers={'User-Agent': 'Airflow Rocket Image Downloader'}
                    )
                    response.raise_for_status()

                    content_type = response.headers.get('content-type', "")
                    if not content_type.startswith('image/'):
                        print(f"URL {url} возвращает не изображение: {content_type}")
                        continue

                    filename = f"{images_dir}/{url.split('/')[-1].split('?')[0]}"
                    if not filename.lower().endswith(('.jpg', '.jpeg', '.png')):
                        filename += '.jpg'
```

```

with open(filename, 'wb') as f:
    for chunk in response.iter_content(chunk_size=8192):
        if chunk:
            f.write(chunk)

success_count += 1
print(f"Успешно загружено изображение {idx}/{len(image_urls)}: {filename}")

```

```

except requests_exceptions.SSLError:
    print(f"Ошибка SSL для {url}")
except requests_exceptions.ConnectionError:
    print(f"Ошибка подключения для {url}")
except requests_exceptions.Timeout:
    print(f"Таймаут при загрузке {url}")
except requests_exceptions.HTTPError as e:
    print(f"HTTP ошибка {e.response.status_code} для {url}")
except Exception as e:
    print(f"Неожиданная ошибка при загрузке {url}: {str(e)}")

```

```

print(f"Итого загружено: {success_count}/{len(image_urls)} изображений")

```

```

if success_count == 0:
    raise AirflowSkipException("Не удалось загрузить ни одного изображения")

```

```

except json.JSONDecodeError:
    raise Exception("Ошибка: Некорректный JSON в файле launches.json")
except Exception as e:
    raise Exception(f"Критическая ошибка: {str(e)}")

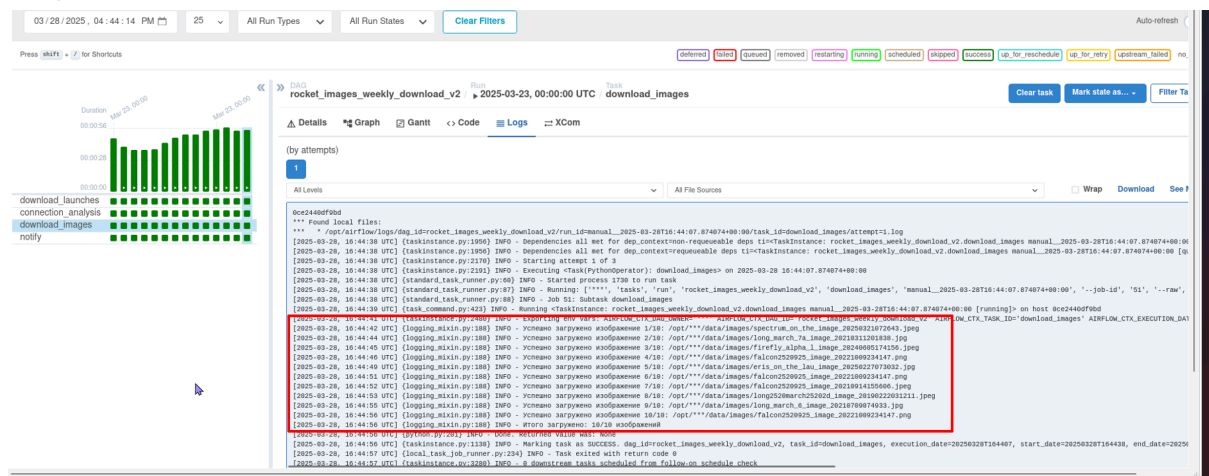
```

```

download_images = PythonOperator(
    task_id="download_images",
    python_callable=download_images_with_error_handling,
    dag=dag,
)

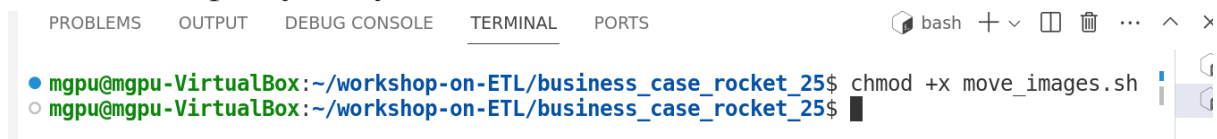
```

Результат:



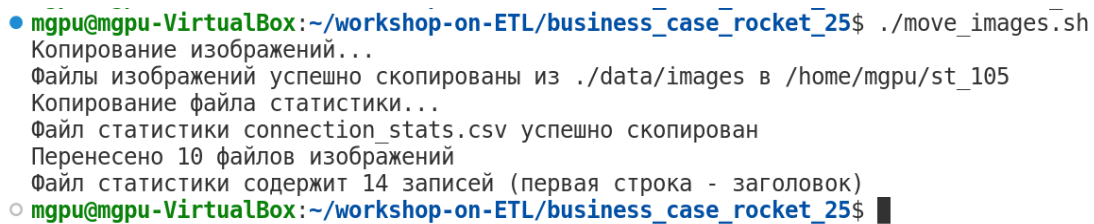
Для перекачки файлов изображений и статистики был создан файл `move_image.sh`

Назначений файлу статуса исполнительный



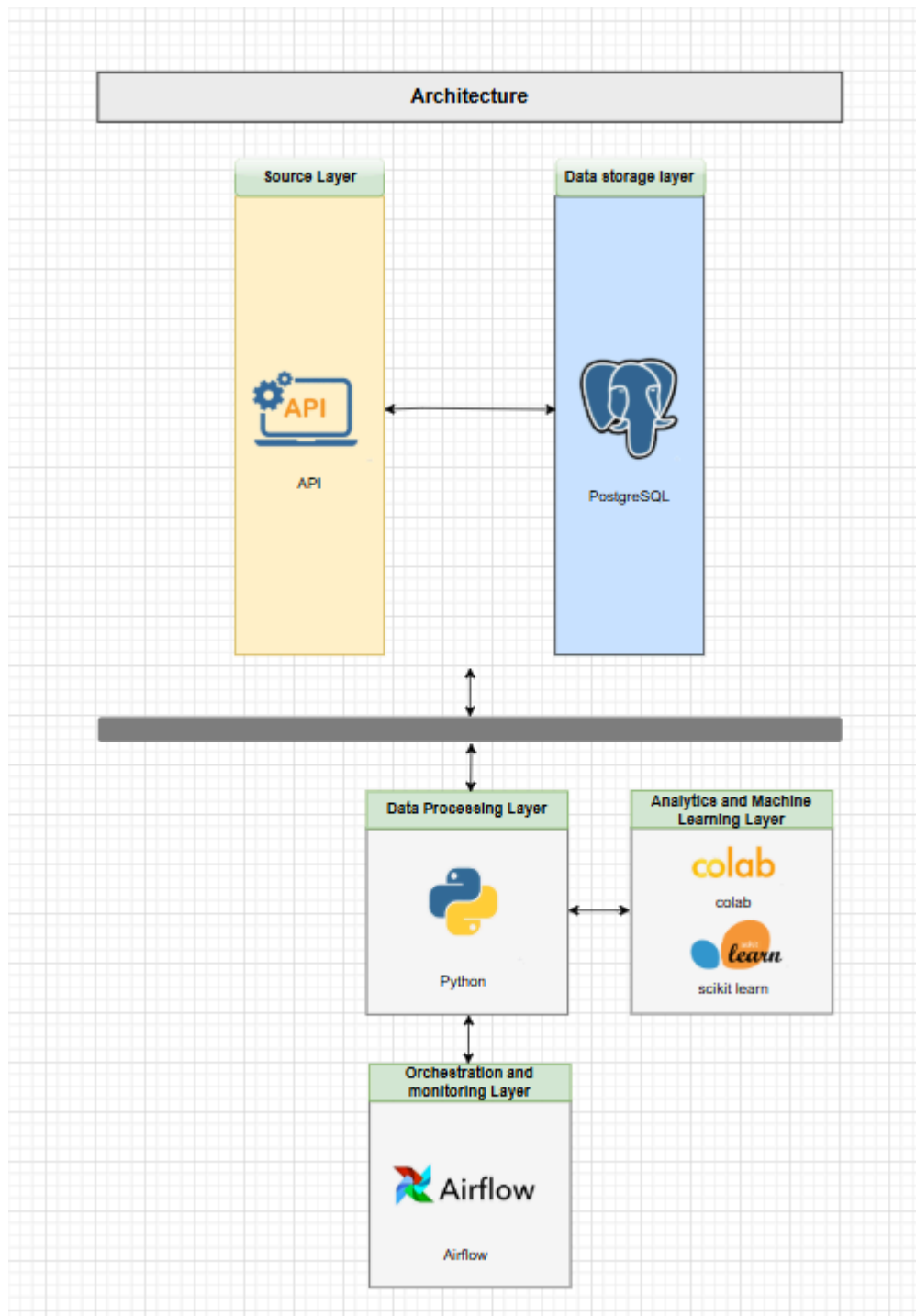
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
mgpu@mgpu-VirtualBox:~/workshop-on-ETL/business_case_rocket_25$ chmod +x move_images.sh
mgpu@mgpu-VirtualBox:~/workshop-on-ETL/business_case_rocket_25$
```

При исполнении скрипта выводится сообщение:



```
mgpu@mgpu-VirtualBox:~/workshop-on-ETL/business_case_rocket_25$ ./move_images.sh
Копирование изображений...
Файлы изображений успешно скопированы из ./data/images в /home/mgpu/st_105
Копирование файла статистики...
Файл статистики connection_stats.csv успешно скопирован
Перенесено 10 файлов изображений
Файл статистики содержит 14 записей (первая строка - заголовок)
mgpu@mgpu-VirtualBox:~/workshop-on-ETL/business_case_rocket_25$
```

Верхнеуровневая архитектура:



Выводы по работе:

Был изменен файл даг в соответствии с индивидуальным заданием, а также составлен исполняемый файл для перекачки данных с одной папки в другую, проанализирована статистика в google colab https://colab.research.google.com/drive/1gjG-3AD4Rm5Pf7Ii8D9AF02ZD_1s8mLK#scrollTo=oT7Ffje7-2PU по данным подключения к серверу.

