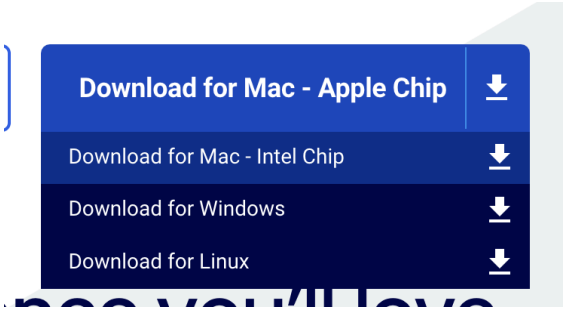
Homework 3: Airflow setup

Installing Airflow using Docker and running simple DAG

Installing Docker

https://www.docker.com/get-started/
Download required version for your platform



Check for successful installation

```
~/Desktop/hse/hse_dc_2024/hw3  main  docker version
Client:
 Cloud integration: v1.0.29
 Version:           20.10.17
 API version:       1.41
 Go version:        go1.17.11
 Git commit:        100c701
 Built:             Mon Jun  6 23:04:45 2022
 OS/Arch:           darwin/arm64
 Context:           default
 Experimental:      true

Server: Docker Desktop 4.12.0 (85629)
 Engine:
  Version:          20.10.17
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.17.11
  Git commit:       a89b842
  Built:            Mon Jun  6 23:01:01 2022
  OS/Arch:          linux/arm64
  Experimental:     false
 containerd:
  Version:          1.6.8
  GitCommit:        9cd3357b7fd7218e4aec3eae239db1f68a5a6ec6
 runc:
  Version:          1.1.4
  GitCommit:        v1.1.4-0-g5fd4c4d
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

Configure Docker

https://airflow.apache.org/docs/apache-airflow/2.3.0/start/docker.html

Downloading template for docker-compose.yaml

`curl -LfO 'https://airflow.apache.org/docs/apache-airflow/2.3.0/docker-compose.yaml'`

I will use LocalExecutor instead of celery. It's more convenient way of running simple test DAGs

```
  AIRFLOW__CORE__EXECUTOR: LocalExecutor
```

Therefore I could remove services airflow-worker, redis and flower because they only work for Celery architecture.

Preparing local environment

initializing required folders and environment variables

```
mkdir -p ./dags ./logs ./plugins
echo -e "AIRFLOW_UID=$(id -u)" > .env
```

Running airflow

Initializing the database

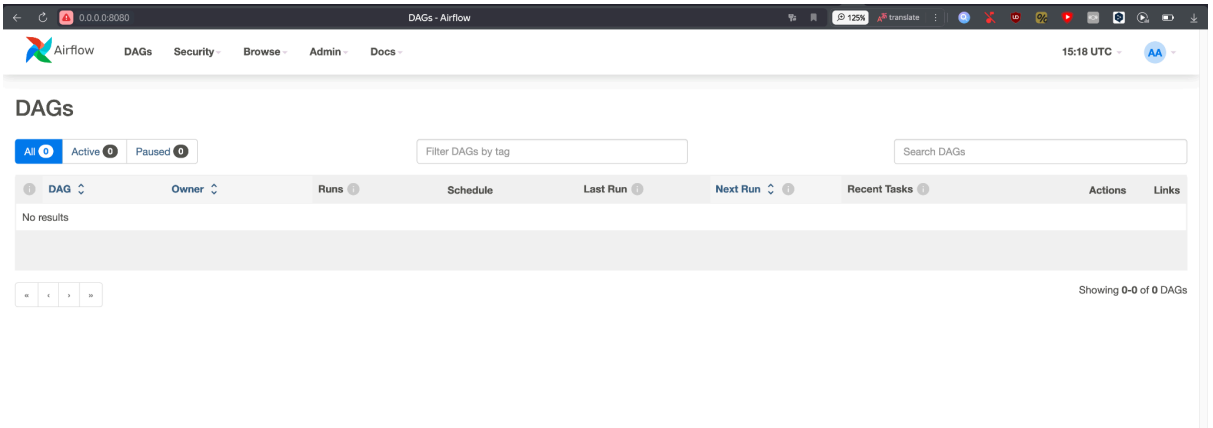
Running airflow init command to perform database migrations and initialize database.

```
docker-compose up airflow-init
```

Starting all services

```
docker-compose up
```

Now we can access Airflow UI on http://0.0.0.0:8080/home



Writing simple DAG

DAG difinition

The code is placed inside ./dags/ folder as a separate .py file

```
from airflow import DAG
from airflow.operators.http_operator import SimpleHttpOperator
from airflow.operators.python_operator import PythonOperator
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago
from datetime import datetime, timedelta
```

```python
def process_numbers(ti):
    random_numbers = ti.xcom_pull(task_ids='fetch_random_numbers')
    numbers_list = random_numbers.split()
    total_sum = sum(int(num) for num in numbers_list)
    return total_sum


default_args = {
    'owner': 'airflow',
    'start_date': days_ago(0),
    'depends_on_past': False,
}


with DAG(
    dag_id="fetch_and_process_random_numbers",
    description="Fetch random numbers from random.org and process them",
    schedule_interval="@once",
    catchup=False,
    default_args=default_args,
) as dag:
    fetch_random_numbers = SimpleHttpOperator(
        task_id='fetch_random_numbers',
        method='GET',
        http_conn_id='random_org',

endpoint='integers/?num=10&min=1&max=100&col=1&base=10&format=plain&rnd=new',
        response_filter=lambda response: response.text,
    )

    process_random_numbers = PythonOperator(
        task_id='process_random_numbers',
        python_callable=process_numbers,
        provide_context=True,
    )

    display_result = BashOperator(
        task_id='display_result',
        bash_command='echo "Total sum of random numbers is: {{
task_instance.xcom_pull(task_ids=\'process_random_numbers\') }}"',
    )

    fetch_random_numbers >> process_random_numbers >> display_result
```

This DAG performs a sequence of three tasks:

1. `fetch_random_numbers`:

It sends an HTTP GET request to `random.org` to fetch 10 random numbers ranging from 1 to 100. These numbers are formatted as a plain text list with one number per line.

2. `process_random_numbers`:

   This task involves a Python function `process_numbers` that processes the numbers retrieved by the `SimpleHttpOperator`. The function pulls the numbers from the previous task (using XComs), splits them by newlines, converts them to integers, and calculates their total sum.

3. `display_result`:

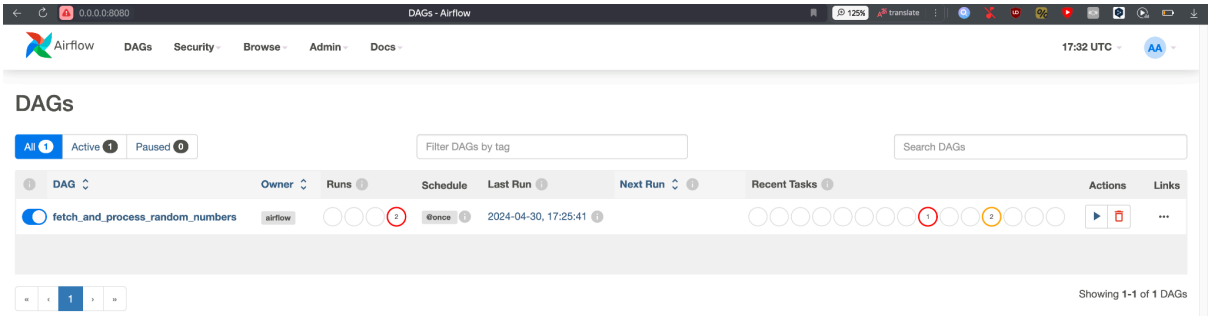
   Function: This task uses a simple Bash command to echo the total sum of the random numbers. The total sum is retrieved using an XCom pull, indicating integration across different tasks within the DAG.

Execute DAG

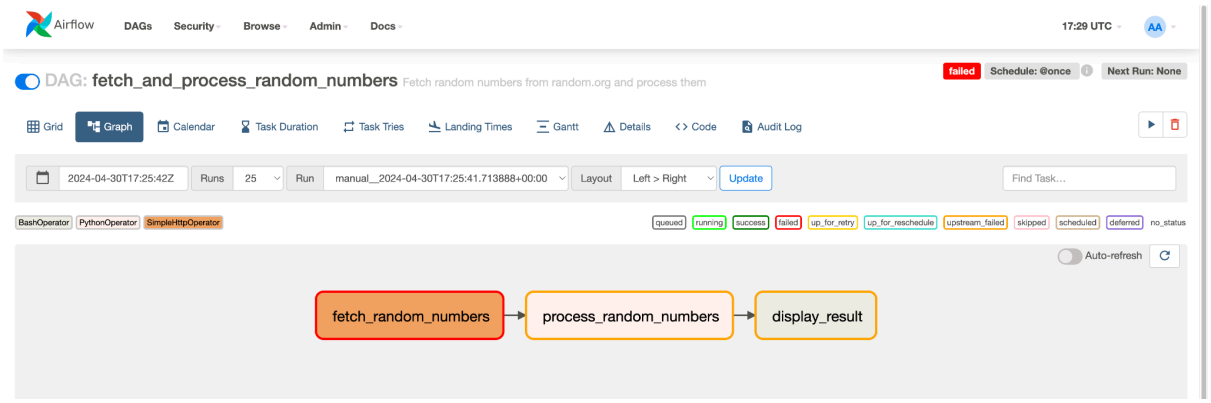
Now we need to restart airflow services.
Stopping current execution using ctrl+c hotkey.

```
docker-compose down
docker-compose up
```

We can see our DAG ready to be executed.



Triggering DAG via “Trigger DAG” button from actions list

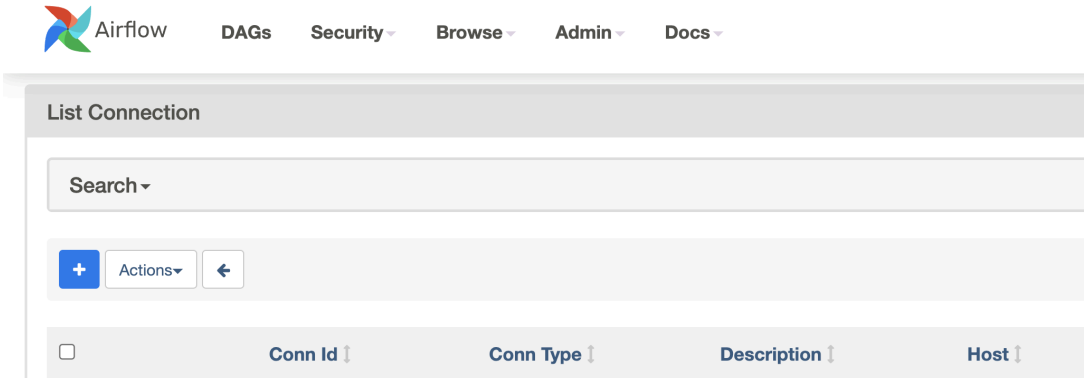


As we can see – DAG failed.

Let's click on the failed task (fetch_random_numbers) and read its logs:

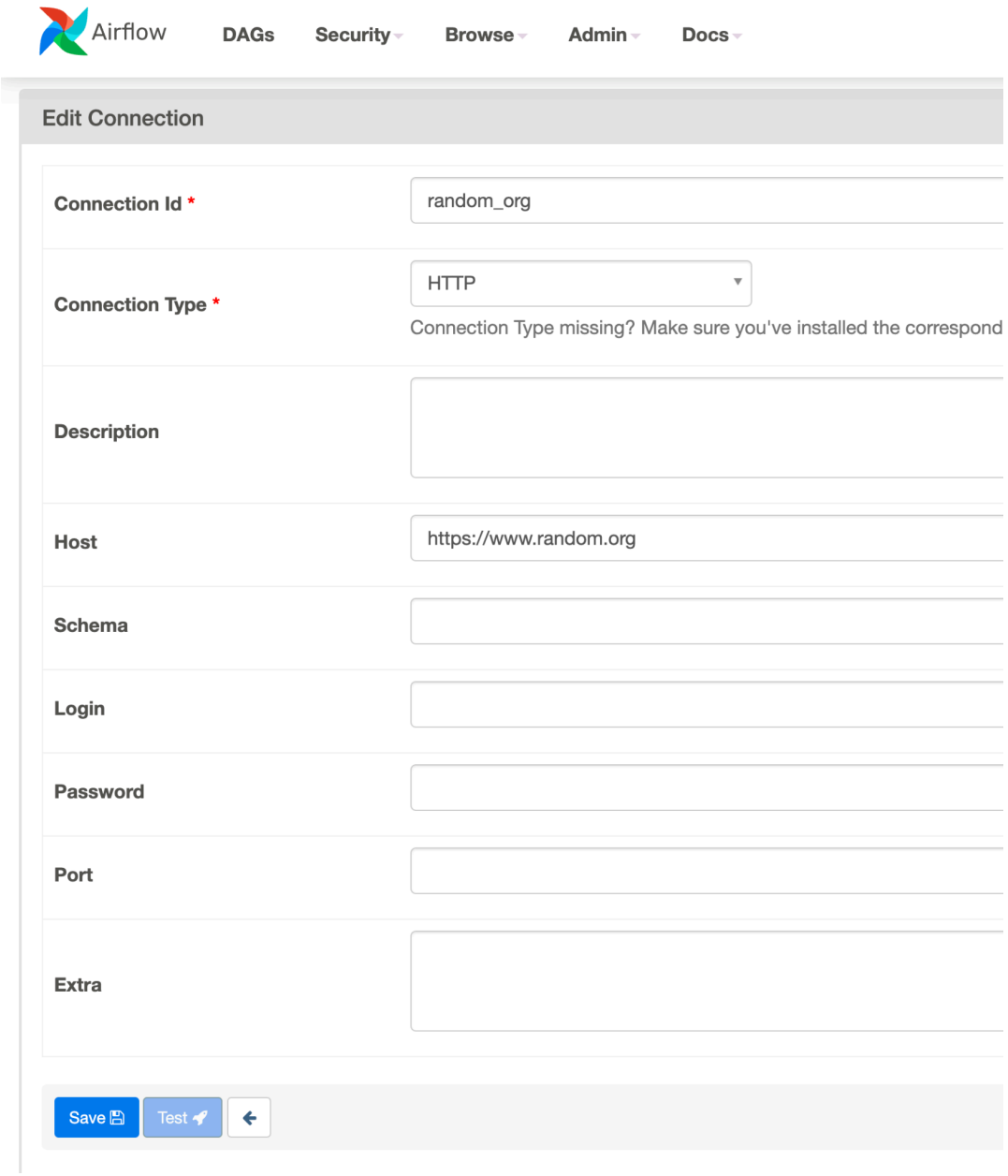
```
[2024-04-30, 17:25:43 UTC] {http.py:102} INFO - Calling HTTP method
[2024-04-30, 17:25:43 UTC] {taskinstance.py:1889} ERROR - Task failed with
exception
Traceback (most recent call last):
  File
"/home/airflow/.local/lib/python3.7/site-packages/airflow/models/connection.p
y", line 430, in get_connection_from_secrets
    raise AirflowNotFoundException(f"The conn_id `{conn_id}` isn't defined")
airflow.exceptions.AirflowNotFoundException: The conn_id `random_org` isn't
defined
[2024-04-30, 17:25:43 UTC] {taskinstance.py:1400} INFO - Marking task as
FAILED. dag_id=fetch_and_process_random_numbers,
task_id=fetch_random_numbers, execution_date=20240430T172541,
start_date=20240430T172543, end_date=20240430T172543
[2024-04-30, 17:25:43 UTC] {standard_task_runner.py:97} ERROR - Failed to
execute job 7 for task fetch_random_numbers (The conn_id `random_org` isn't
defined; 207)
```

That's because we need to configure the connection.
Open the Admin tab and select Connections.
Click the + button to add a new connection.

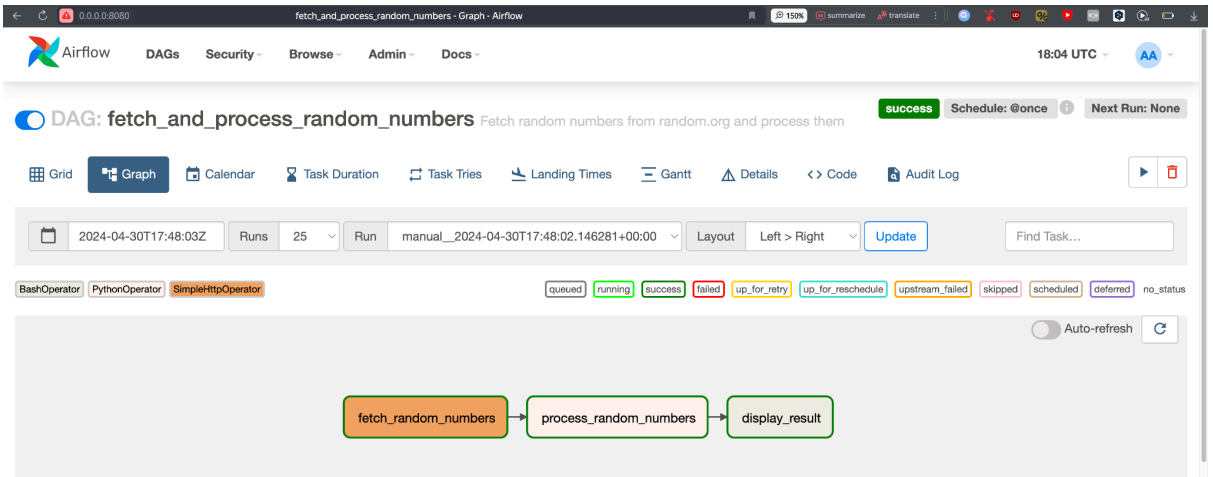


Enter the following information:

Conn Id: random_org
Conn Type: HTTP
Host: https://www.random.org



Lest try to trigger DAG one more time



And this time our DAG executed successfully!
We can see tasks execution results as XCOM outputs in interface

DAG: **fetch_and_process_random_numbers** Fetch random numbers from random.org and process them

Grid | Graph | Calendar | Task Duration | Task Tries | Landing Times | Gantt | Details | Code

Task Instance: **display_result** at **2024-04-30, 17:48:02**

Task Instance Details | Rendered Template | Log | XCom

XCom

Key	Value
return_value	`Total sum of random numbers is: 537`