Airflow

https://airflow.apache.org/docs/apache-airflow/1.10.10/

1. Скачал контейнер из https://github.com/puckel/docker-airflow
2. Распаковал, перешел в папку, где он был распакован и запустил:

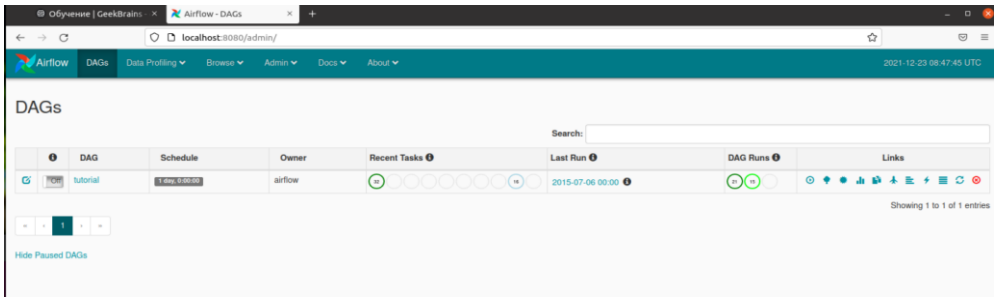
docker-compose -f docker-compose-LocalExecutor.yml up -d

docker exec -it docker-airflow-master_webserver_1 bash

```
max@max:~/Downloads/docker-airflow-master$ docker-compose -f docker-compose-Loca
lExecutor.yml up -d
docker-airflow-master_postgres_1 is up-to-date
docker-airflow-master_webserver_1 is up-to-date
max@max:~/Downloads/docker-airflow-master$ docker exec -it docker-airflow-master
_webserver_1 bash
airflow@daf131963fb3:~$
```

3. В docker-compose-LocalExecutor.yml посмотрел проброшенный порт (8080) для аэйфлоу. Открыл вебморду в браузере

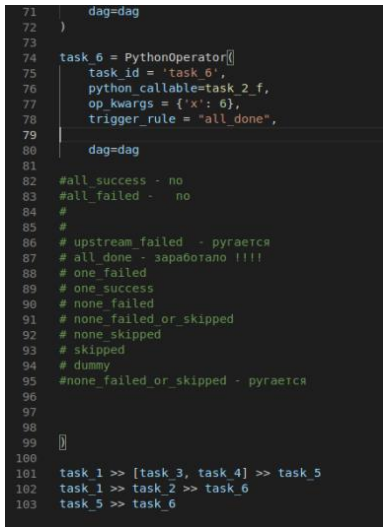
localhost:8080



4. Создаю ПИтоновский скрипт и заливаю его в папку dags

```python
from airflow import DAG
#from airflow.operators.bash_operator import BashOperator
from airflow.operators.python_operator import PythonOperator, BranchPythonOperator
from airflow.models import Variable
from random import randint

from datetime import datetime, timedelta


default_args = {
    "owner": "airflow",
    "depends_on_past": False,
    "start_date": datetime(2015, 6, 1),
    "retries": 1,
    "retry_delay": timedelta(minutes=5),
    "trigger_rule": "none_failed"
}

dag = DAG(
    dag_id='my_hw_7',
    default_args=default_args,
    schedule_interval=None,
    catchup = False
    )

def task_1_f (**kwarg):
    rand_step = randint(1,3)
    rand_step = 3
    if rand_step == 1 or rand_step == 2:
        next_step = eval(Variable.get('hw_etl')).get(rand_step)
    else:
        #next_step = eval(Variable.get('hw_etl')).get(rand_step)
        next_step = ['task_6']
    return next_step

def task_2_f (x, **kwarg):
    print(x)

task_1 = BranchPythonOperator(
    task_id = 'task_1',
    python_callable=task_1_f,
    dag=dag
)
```

```python
)


task_2 = PythonOperator(
    task_id = 'task_2',
    python_callable=task_2_f,
    op_kwargs = {'x': 2},
    dag=dag
)

task_3 = PythonOperator(
    task_id = 'task_3',
    python_callable=task_2_f,
    op_kwargs = {'x': 3},
    dag=dag
)
task_4 = PythonOperator(
    task_id = 'task_4',
    python_callable=task_2_f,
    op_kwargs = {'x': 4},
    dag=dag
)

task_5 = PythonOperator(
    task_id = 'task_5',
    python_callable=task_2_f,
    op_kwargs = {'x': 5},
    trigger_rule = "one_success",
    dag=dag
)

task_6 = PythonOperator(
    task_id = 'task_6',
    python_callable=task_2_f,
    op_kwargs = {'x': 6},
    trigger_rule = "all_done",

    dag=dag

#all_success - no
#all_failed -   no
#
#
```



```python
    dag=dag
)

task_6 = PythonOperator(
    task_id = 'task_6',
    python_callable=task_2_f,
    op_kwargs = {'x': 6},
    trigger_rule = "all_done",

    dag=dag

#all_success - no
#all_failed -   no
#
#
# upstream_failed  - ругается
# all_done - заработало !!!!
# one_failed
# one_success
# none_failed
# none_failed_or_skipped
# none_skipped
# skipped
# dummy
#none_failed_or_skipped - ругается



task_1 >> [task_3, task_4] >> task_5
task_1 >> task_2 >> task_6
task_5 >> task_6
```

5. В результате: