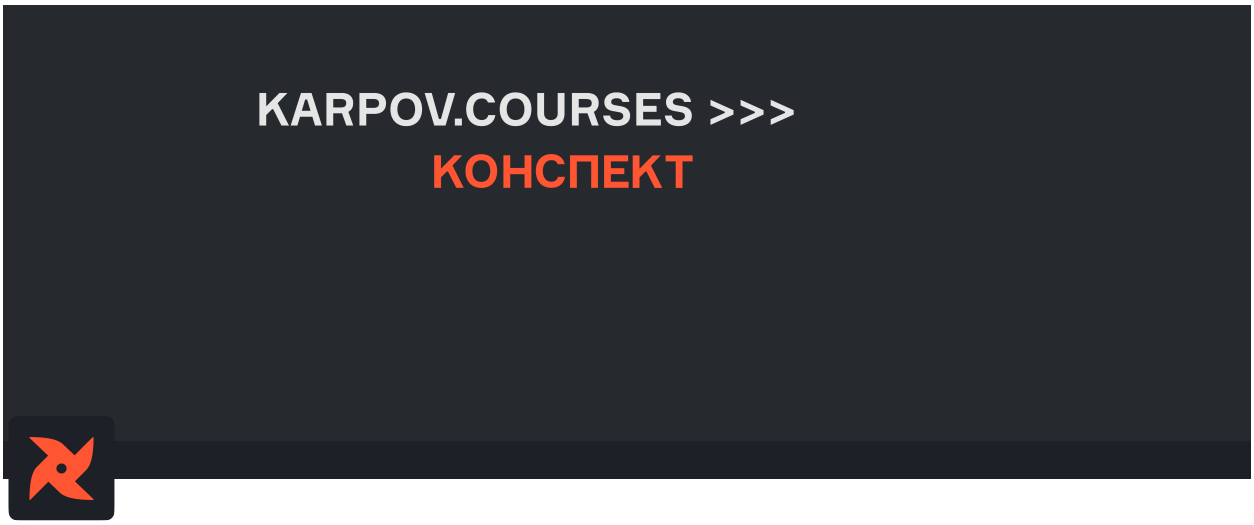

> Конспект > 2 урок > Airflow

> Добавление DAG
> Структура DAG
> Код с лекции

> Добавление DAG

1. Сохранить скрипт в формате **.py**
2. Зайти в GITLAB в проект **airflow-2.0**
3. В папке dags создать папку, назвав ее своим логином
4. Запушить скрипт в свою папку на GITLAB
5. Включить DAG в AIRFLOW

> Структура DAG

Блок импортов

```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator # Так как мы пишет такси в питоне
from datetime import datetime
```

Список операторов можно посмотреть по ссылке

+ необходимые для выполнения кода библиотеки

Блок кода

Здесь вы прописываете ваши функции

```
def foo1():
def foo2():
def foo3():
```

Блок инициализации

Задаем параметры в DAG:

```
default_args = {
    'owner': 'your_name', # Владелец операции
    'depends_on_past': False, # Зависимость от прошлых запусков

    'schedule_interval': '0 12 * * *' # cron выражение, также можно использовать '@daily', '@weekly'
    #'schedule_interval': '@daily' переменные airflow
    #'schedule_interval': timedelta() параметр timedelta

    'retries': 1, # Кол-во попыток выполнить DAG
    'retry_delay': timedelta(minutes=5), # Промежуток между перезапусками

    'email': '', # Почта для уведомлений
    'email_on_failure': '', # Почта для уведомлений при ошибке
    'email_on_retry': '', # Почта для уведомлений при перезапуске

    'retry_exponential_backoff': '', # Для установления экспоненциального времени между перезапусками
    'max_retry_delay': '', # Максимальный промежуток времени для перезапуска

    'start_date': '', # Дата начала выполнения DAG
    'end_date': '', # Дата завершения выполнения DAG

    'on_failure_callback': '', # Запустить функцию если DAG упал
    'on_success_callback': '', # Запустить функцию если DAG выполнился
    'on_retry_callback': '', # Запустить функцию если DAG ушел на повторный запуск
    'on_execute_callback': '', # Запустить функцию если DAG начал выполняться
     # Задать документацию
    'doc': '',
    'doc_md': '',
    'doc_rst': '',
    'doc_json': '',
    'doc_yaml': ''
}
dag = DAG('DAG_name', default_args=default_args)
```

Инициализируем таски:

```
t1 = PythonOperator(task_id='foo1', # Название таска
                    python_callable=foo1, # Название функции
                    dag=dag) # Параметры DAG

t2 = PythonOperator(task_id='foo2', # Название таска
                    python_callable=foo2, # Название функции
                    dag=dag) # Параметры DAG

t3 = PythonOperator(task_id='foo2', # Название таска
                    python_callable=foo2, # Название функции
                    dag=dag) # Параметры DAG
```

Блок логики

Задаём логику выполнения

```
# Python операторы
t1 >> t2 >> t3
```

```
# Методы таска
t1.set_downstream(t2)
t2.set_downstream(t3)
```

Для параллельного выполнения тасков используется структура Python операторов в виде

```
A >> [B, C] >> D
```

или прописываются зависимости через методы таска

```
A.set_downstream(B)
A.set_downstream(C)
B.set_downstream(D)
C.set_downstream(D)
```

Таким образом, таски B и C будут выполняться параллельно, а D выполнится только после успешного выполнения B и C

Подробнее про параметры DAG в документации

> Код с лекции

Импортируем библиотеки

```
import requests
from zipfile import ZipFile
from io import BytesIO
import pandas as pd
from datetime import timedelta
from datetime import datetime

from airflow import DAG
from airflow.operators.python import PythonOperator
```

Подгружаем данные

```
TOP_1M_DOMAINS = 'http://s3.amazonaws.com/alexa-static/top-1m.csv.zip'
TOP_1M_DOMAINS_FILE = 'top-1m.csv'
```

Таски

```
def get_data():
    top_doms = requests.get(TOP_1M_DOMAINS, stream=True)
    zipfile = ZipFile(BytesIO(top_doms.content))
    top_data = zipfile.read(TOP_1M_DOMAINS_FILE).decode('utf-8')

    with open(TOP_1M_DOMAINS_FILE, 'w') as f:
        f.write(top_data)


def get_stat():
    top_data_df = pd.read_csv(TOP_1M_DOMAINS_FILE, names=['rank', 'domain'])
    top_data_top_10 = top_data_df[top_data_df['domain'].str.endswith('.ru')]
    top_data_top_10 = top_data_top_10.head(10)
    with open('top_data_top_10.csv', 'w') as f:
        f.write(top_data_top_10.to_csv(index=False, header=False))


def get_stat_com():
    top_data_df = pd.read_csv(TOP_1M_DOMAINS_FILE, names=['rank', 'domain'])
    top_data_top_10 = top_data_df[top_data_df['domain'].str.endswith('.com')]
    top_data_top_10 = top_data_top_10.head(10)
    with open('top_data_top_10_com.csv', 'w') as f:
        f.write(top_data_top_10.to_csv(index=False, header=False))


def print_data(ds): # передаем глобальную переменную airflow
    with open('top_data_top_10.csv', 'r') as f:
        all_data = f.read()
    with open('top_data_top_10_com.csv', 'r') as f:
        all_data_com = f.read()
    date = ds

    print(f'Top domains in .RU for date {date}')
    print(all_data)
```

```
    print(f'Top domains in .COM for date {date}')
    print(all_data_com)
```

В Airflow есть свои глобальные переменные, список которых можно посмотреть в документации

Инициализируем DAG

```
default_args = {
    'owner': 'a.batalov',
    'depends_on_past': False,
    'retries': 2,
    'retry_delay': timedelta(minutes=5),
    'start_date': datetime(2021, 10, 7),
    'schedule_interval': '0 12 * * *'
}
dag = DAG('top_10_ru_new', default_args=default_args)
```

Инициализируем таски

```
t1 = PythonOperator(task_id='get_data',
                    python_callable=get_data,
                    dag=dag)

t2 = PythonOperator(task_id='get_stat',
                    python_callable=get_stat,
                    dag=dag)

t2_com = PythonOperator(task_id='get_stat_com',
                        python_callable=get_stat_com,
                        dag=dag)

t3 = PythonOperator(task_id='print_data',
                    python_callable=print_data,
                    dag=dag)
```

Задаем порядок выполнения

```
t1 >> [t2, t2_com] >> t3
```