

# СЛОЖНЫЕ ПАЙПЛАЙНЫ

**KARPOV.COURSES**

# БЛОК ETL

1. Введение в ETL

2. Знакомство с Airflow

3. Сложные пайплайны, часть 1

Вы находитесь здесь

---

4. Сложные пайплайны, часть 2

5. Разработка своих плагинов

6. Установка и настройка Airflow

# ЛЕКЦИЯ «СЛОЖНЫЕ ПАЙПЛАЙНЫ»

1. XCom
2. Taskflow API
3. SubDags
4. TaskGroup
5. Динамическое создание дагов
6. Best Practice

XCOM

# XCOM

- xcom\_push
- xcom\_pull

```
def explicit_push_func(**kwargs):  
    kwargs['ti'].xcom_push(value='Hello world!', key='hi')  
  
def implicit_push_func():  
    return 'Some string from function'  
  
explicit_push = PythonOperator(  
    task_id='explicit_push',  
    python_callable=explicit_push_func,  
    provide_context=True  
)  
  
implicit_push = PythonOperator(  
    task_id='implicit_push',  
    python_callable=implicit_push_func  
)
```



List XComs

Search ▾

Actions ▾



Record Count: 2

<input type="checkbox"/>	Key ↑	Value ↑	Timestamp ↑	Execution Date ↑	Task Id ↑	Dag Id ↑
<input type="checkbox"/> 	return_value	Some string from function	2021-10-10, 01:43:08	2021-10-01, 03:00:00	implicit_push ⚡	dina_examples
<input type="checkbox"/> 	hi	Hello world!	2021-10-10, 01:43:11	2021-10-01, 03:00:00	explicit_push ⚡	dina_examples

```
def print_both_func(**kwargs):
    logging.info('-----')
    logging.info(kwargs['ti'].xcom_pull(task_ids='explicit_push', key='hi'))
    logging.info(kwargs['templates_dict']['implicit'])
    logging.info('-----')

print_both = PythonOperator(
    task_id='print_both',
    python_callable=print_both_func,
    templates_dict={'implicit': '{{ ti.xcom_pull(task_ids="implicit_push") }}'},
    provide_context=True
)
```



```
*** Reading local file: /var/log/airflow/dina_examples/print_both/2021-10-08T00:00:00+00:00/3.log
[2021-10-09 22:45:25,762] {taskinstance.py:903} INFO - Dependencies all met for <TaskInstance: dina_examples.print_both 20
[2021-10-09 22:45:25,800] {taskinstance.py:903} INFO - Dependencies all met for <TaskInstance: dina_examples.print_both 20
[2021-10-09 22:45:25,800] {taskinstance.py:1095} INFO -
-----
[2021-10-09 22:45:25,801] {taskinstance.py:1096} INFO - Starting attempt 3 of 3
[2021-10-09 22:45:25,801] {taskinstance.py:1097} INFO -
-----
[2021-10-09 22:45:25,814] {taskinstance.py:1115} INFO - Executing <Task(PythonOperator): print_both> on 2021-10-08T00:00:00
[2021-10-09 22:45:25,821] {standard_task_runner.py:52} INFO - Started process 616492 to run task
[2021-10-09 22:45:25,831] {standard_task_runner.py:76} INFO - Running: ['airflow', 'tasks', 'run', 'dina_examples', 'print
[2021-10-09 22:45:25,835] {standard_task_runner.py:77} INFO - Job 782: Subtask print_both
[2021-10-09 22:45:25,921] {logging_mixin.py:109} INFO - Running <TaskInstance: dina_examples.print_both 2021-10-08T00:00:00
[2021-10-09 22:45:26,072] {taskinstance.py:1254} INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_OWNER=Karpov
AIRFLOW_CTX_DAG_ID=dina_examples
AIRFLOW_CTX_TASK_ID=print_both
AIRFLOW_CTX_EXECUTION_DATE=2021-10-08T00:00:00+00:00
AIRFLOW_CTX_DAG_RUN_ID=scheduled__2021-10-08T00:00:00+00:00
[2021-10-09 22:45:26,074] {dina_examples.py:110} INFO - -----
[2021-10-09 22:45:26,095] {dina_examples.py:111} INFO - Hello world!
[2021-10-09 22:45:26,096] {dina_examples.py:112} INFO - Some string from function
[2021-10-09 22:45:26,096] {dina_examples.py:113} INFO - -----
[2021-10-09 22:45:26,096] {python.py:151} INFO - Done. Returned value was: None
[2021-10-09 22:45:26,112] {taskinstance.py:1219} INFO - Marking task as SUCCESS. dag_id=dina_examples, task_id=print_both,
[2021-10-09 22:45:26,164] {local_task_job.py:151} INFO - Task exited with return code 0
[2021-10-09 22:45:26,197] {local_task_job.py:261} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

# TASKFLOW API

# Taskflow API

```
@dag(default_args=DEFAULT_ARGS,
      schedule_interval='@daily',
      tags=['karpov'])
def dina_taskflow():

    @task
    def list_of_nums():
        return [1, 2, 3, 4, 6]

    @task
    def sum_nums(nums: list):
        return sum(nums)

    @task
    def print_sum(total: int):
        logging.info(str(total))

    print_sum(sum_nums(list_of_nums()))

dina_taskflow_dag = dina_taskflow()
```

[DAGs](#)[Security](#)[Browse](#)[Admin](#)[Docs](#)

02:14 MSK (+03:00)



## DAG: dina\_taskflow

success

schedule: @daily

[Tree View](#)[Graph View](#)[Calendar View](#)[Task Duration](#)[Task Tries](#)[Landing Times](#)[Gantt](#)[Details](#)[Code](#)

2021-10-07T03:00:01+0

Runs

25



Run

scheduled\_\_2021-09-29T00:00:00+00:00



Layout

Left &gt; Right



Update

Find Task...

[\\_PythonDecoratedOperator](#)

queued

running

success

failed

up\_for\_retry

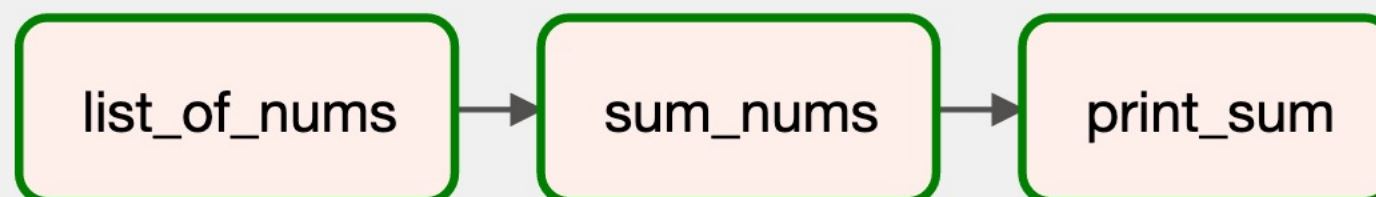
up\_for\_reschedule

upstream\_failed

skipped

scheduled

no\_status

☐ Auto-refresh

# SUBDAGS





DAG: dina

Tree View

Details

< > Co

2021-10-01T

Layout Left > Rig

Find Task...

DummyOperator SubDagOp

ing schedule: @daily

nes Gantt



kipped scheduled no\_status

Auto-refresh

Task Instance: section-1  
at: 2021-09-28T00:00:00+00:00

Zoom into Sub DAG

Instance Details

Rendered

Log

All Instances

Filter Upstream

Download Log (by attempts):

All 1 2

Task Actions

Ignore All Deps

Ignore Task State

Ignore Task Deps

Run

Past

Future

Upstream

Downstream

Recursive

Failed

Clear

Past

Future

Upstream

Downstream

Mark Failed

Past

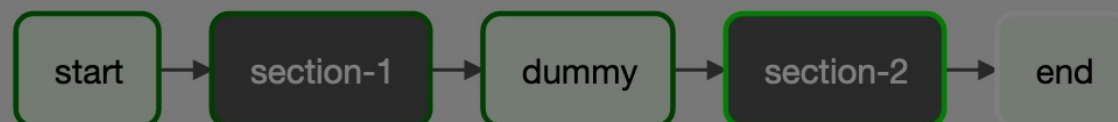
Future

Upstream

Downstream

Mark Success

Close



[DAGs](#)[Security](#)[Browse](#)[Admin](#)[Docs](#)

22:58 MSK (+03:00)



^ DAG: dina\_subdag

## SUBDAG: dina\_subdag.section-1

success

schedule: @daily

[Tree View](#)[Graph View](#)[Calendar View](#)[Task Duration](#)[Task Tries](#)[Landing Times](#)[Gantt](#)[Details](#)[Code](#)

2021-09-28T03:00:01+0

Runs

25

▼

Run

scheduled\_\_2021-09-28T00:00:00+00:00

▼

Layout

Top > Bottom

▼

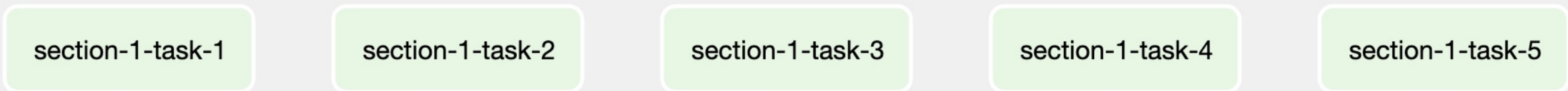
Update

Find Task...

DummyOperator

queued running success failed up\_for\_retry up\_for\_reschedule upstream\_failed skipped scheduled no\_status

☐ Auto-refresh



```
]def subdag(parent_dag_name, child_dag_name, args):  
    dag_subdag = DAG(  
        dag_id=f'{parent_dag_name}.{child_dag_name}',  
        default_args=args,  
        start_date=days_ago(2),  
        schedule_interval="@daily",  
    )  
  
    for i in range(5):  
        DummyOperator(  
            task_id=f'{child_dag_name}-task-{i + 1}',  
            default_args=args,  
            dag=dag_subdag,  
        )  
  
    return dag_subdag
```



```
with DAG(DAG_NAME,
         schedule_interval='@daily',
         default_args=DEFAULT_ARGS,
         max_active_runs=1,
         tags=['karpov']
    ) as dag:

    start = DummyOperator(task_id='start')
    dummy = DummyOperator(task_id='dummy')
    end = DummyOperator(task_id='end')

    section_1 = SubDagOperator(
        task_id='section-1',
        subdag=subdag(DAG_NAME, 'section-1', DEFAULT_ARGS),
    )

    section_2 = SubDagOperator(
        task_id='section-2',
        subdag=subdag(DAG_NAME, 'section-2', DEFAULT_ARGS),
    )

    start >> section_1 >> dummy >> section_2 >> end
```

# SUBDAGS

- Расписание у дага и сабдага должны совпадать
- Название сабдагов: `parent.child`
- Состояние сабдага и таска `SubDagOperator` независимы
- По возможности избегайте

TASKGROUP

```

with DAG('dina_taskgroup',
        schedule_interval='@daily',
        default_args=DEFAULT_ARGS,
        max_active_runs=1,
        tags=['karpov']
        ) as dag:

    start = DummyOperator(task_id='start')

    with TaskGroup(group_id='group1') as tg1:
        for i in range(5):
            DummyOperator(task_id=f'task{i + 1}')

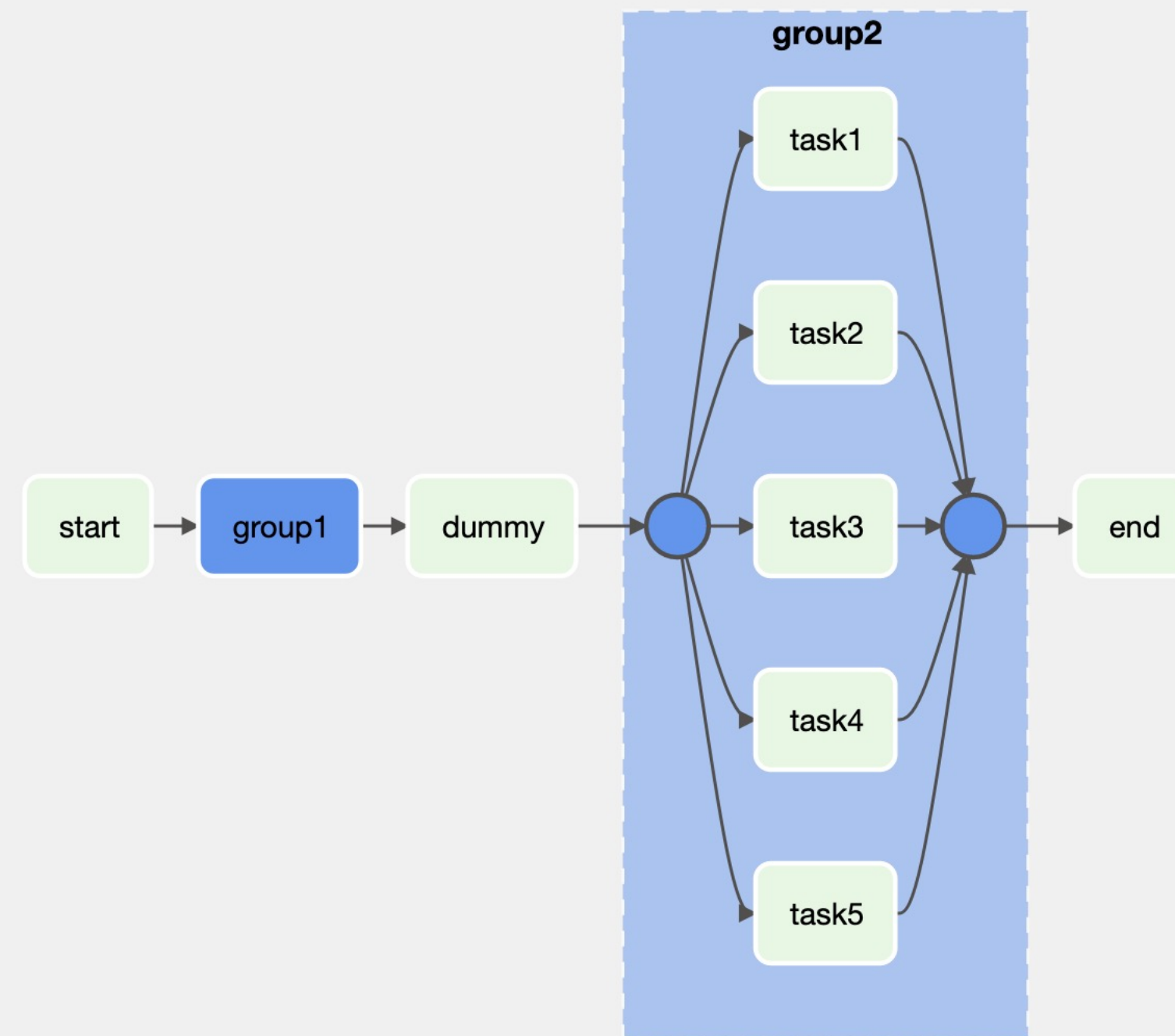
    dummy = DummyOperator(task_id='dummy')

    with TaskGroup(group_id='group2') as tg2:
        for i in range(5):
            DummyOperator(task_id=f'task{i + 1}')

    end = DummyOperator(task_id='end')

    start >> tg1 >> dummy >> tg2 >> end

```



# ДИНАМИЧЕСКОЕ СОЗДАНИЕ ДАГОВ

# ДИНАМИЧЕСКОЕ СОЗДАНИЕ ДАГОВ

- Скрипты должны находиться в DAG\_FOLDER
- dag в globals()

# ДИНАМИЧЕСКОЕ СОЗДАНИЕ ДАГОВ

- Статичная генерация нескольких одинаковых дагов
- Генерация дага из глобальных переменных/соединений
- Генерация дага на основе json/yaml-файла

# AIRFLOW BEST PRACTICE



# AIRFLOW BEST PRACTICE

- Сохраняйте идемпотентность
- Не храните пароли в коде
- Не храните файлы локально
- Убирайте лишний код верхнего уровня
  - Вся логику переносите в код таска
- Не используйте переменные Airflow
  - Загружайте переменные из jinja
  - Загружайте переменные внутри таска
  - Используйте переменные окружения

# СПАСИБО



**ДИНА САФИНА**