Implementación de flujos de tareas mediante Airflow

NAME

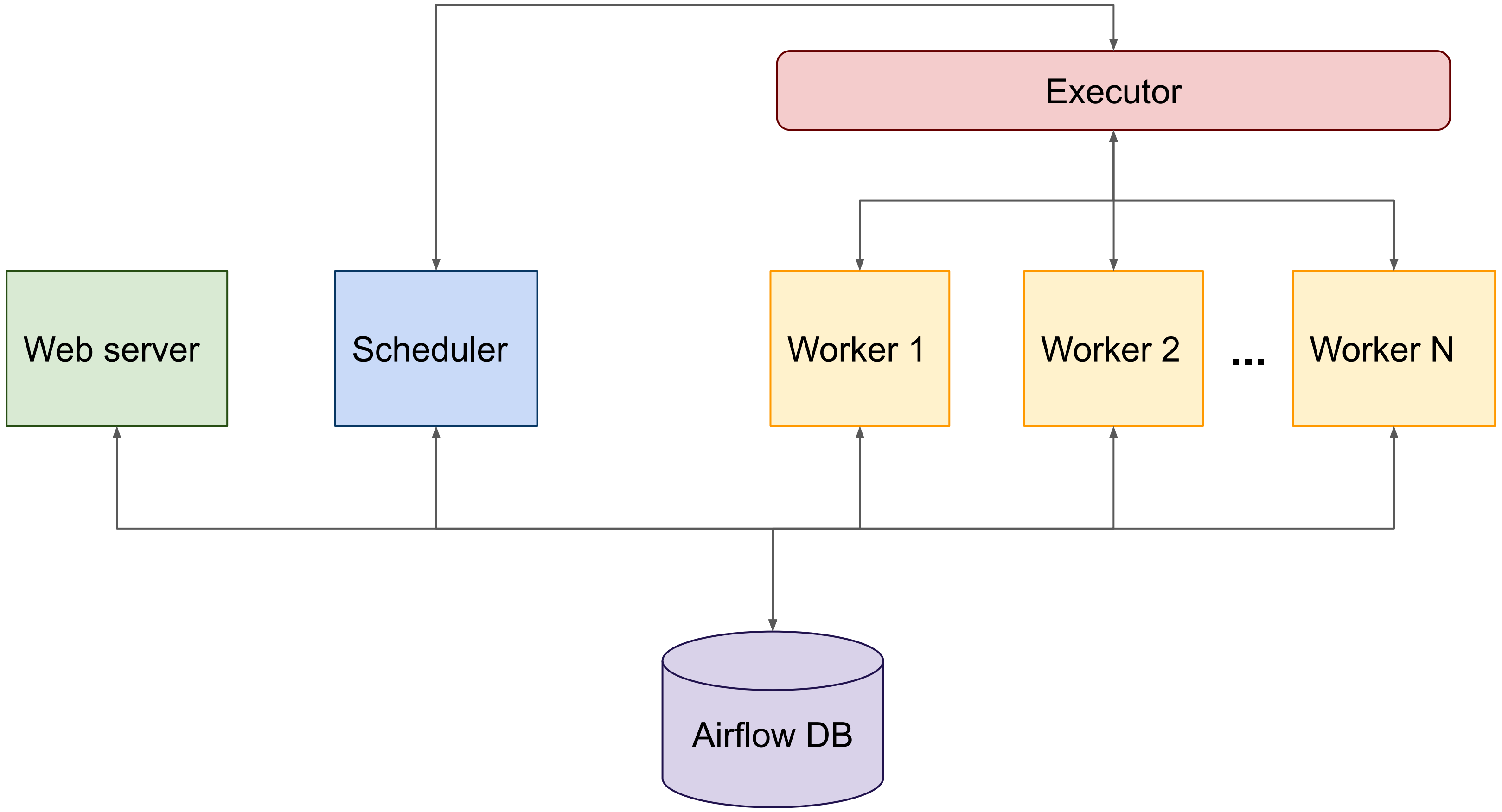
top

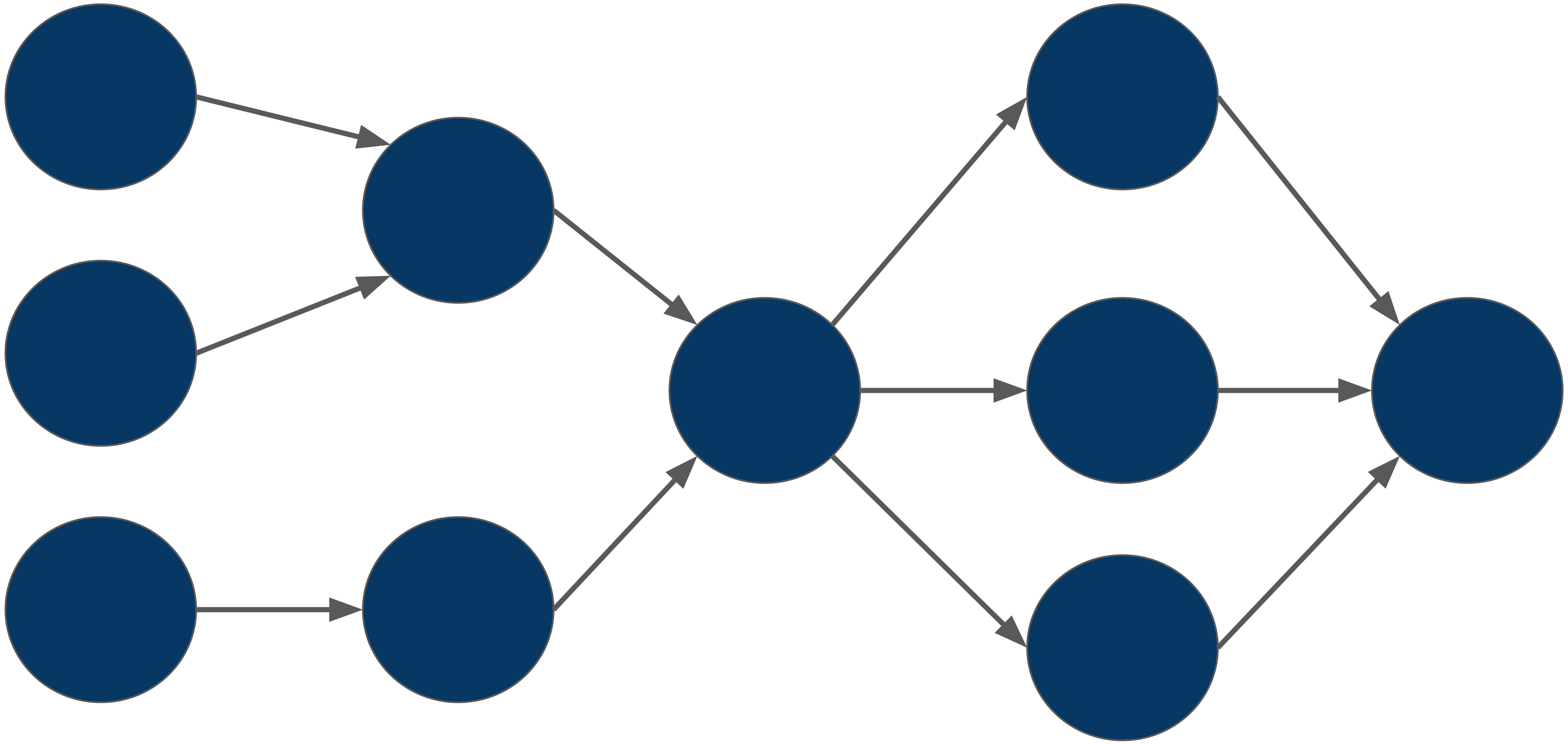
crontab - files used to schedule the execution of programs

K.I.S.S.

Keep It Simple, Stupid!

Apache
Airflow


Executor
Web server
Scheduler
Worker 1
Worker 2
...
Worker N
Airflow DB



amazon S3

Google Cloud

druid

Requests

docker

presto

hadoop

slack

SAMBA

Apache Pig

SQLAlchemy

Microsoft SQL Server

SQLite

JDBC

MySQL

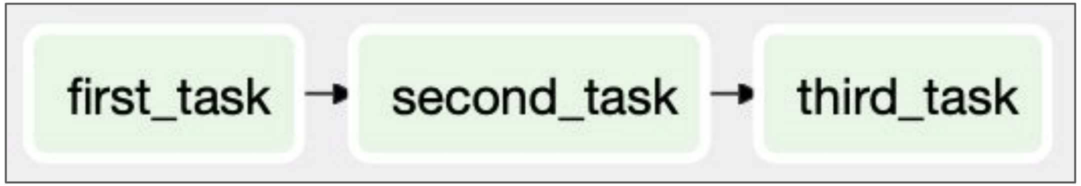
HIVE

PostgreSQL

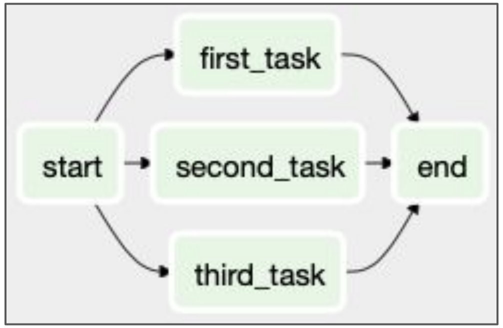
zendesk

ORACLE DATABASE

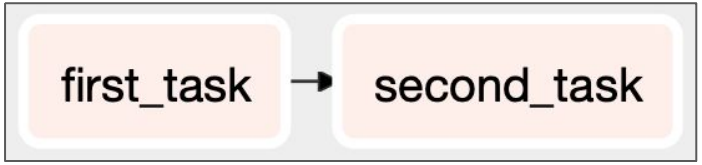
```
default_args = {'start_date': datetime(2019, 1, 1)}

dag = DAG(dag_id='example_serial', default_args=default_args)

with dag:
    first_task = DummyOperator(task_id='first_task')
    second_task = DummyOperator(task_id='second_task')
    third_task = DummyOperator(task_id='third_task')

    second_task.set_upstream(first_task)
    third_task.set_upstream(second_task)
```

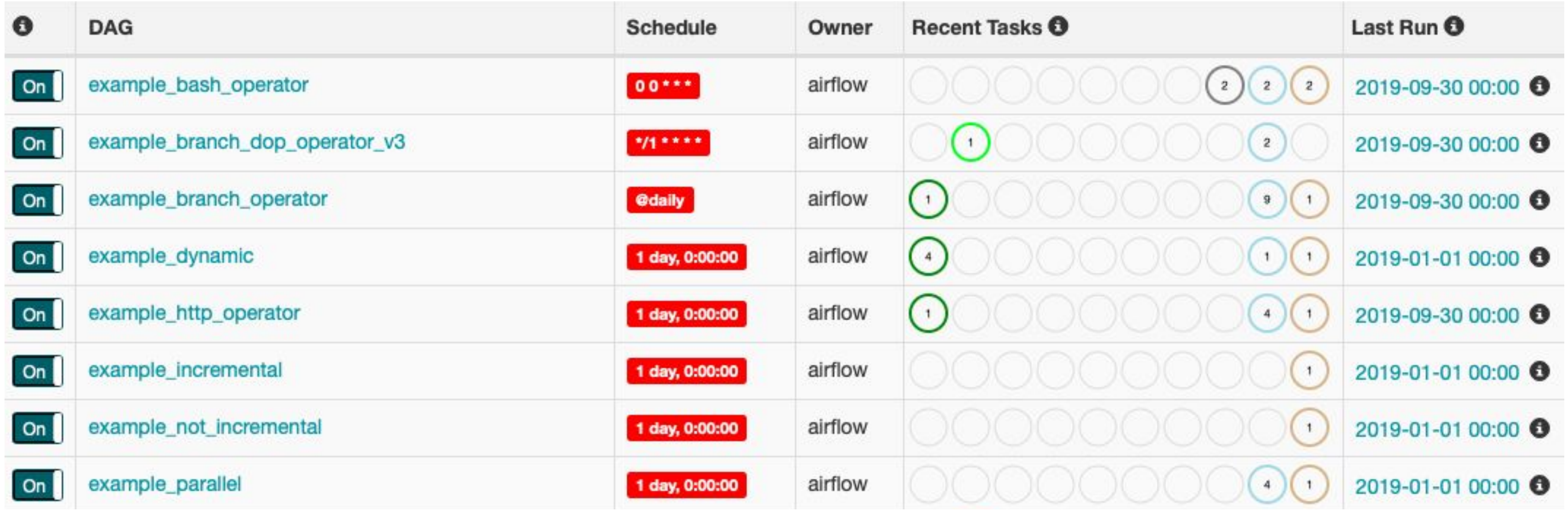


```python
with dag:
    start = DummyOperator(task_id='start')
    first_task = DummyOperator(task_id='first_task')
    second_task = DummyOperator(task_id='second_task')
    third_task = DummyOperator(task_id='third_task')
    end = DummyOperator(task_id='end')

    first_task.set_upstream(start)
    second_task.set_upstream(start)
    third_task.set_upstream(start)
    end.set_upstream([first_task, second_task, third_task])
```

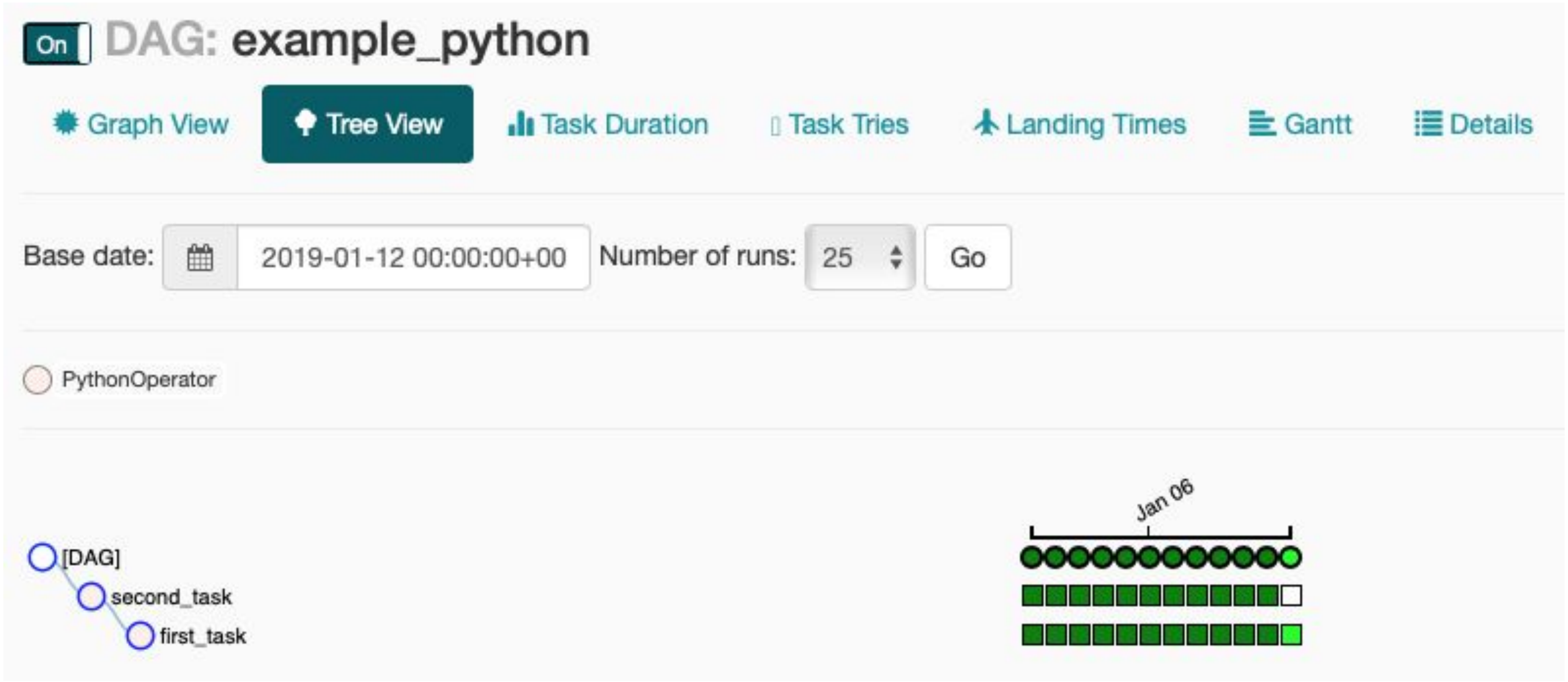


```python
def execute_first_task():
    print('first task')


def execute_second_task():
    print('second task')


with dag:
    first_task = PythonOperator(
        task_id='first_task',
        python_callable=execute_first_task,
    )

    second_task = PythonOperator(
        task_id='second_task',
        python_callable=execute_second_task,
    )

    second_task.set_upstream(first_task)
```





	DAG	Schedule	Owner	Recent Tasks	Last Run
On	example_bash_operator	0 0 * * *	airflow	○ ○ ○ ○ ○ ○ ○ 2 2 2	2019-09-30 00:00
On	example_branch_dop_operator_v3	*/1 * * * *	airflow	○ 1 ○ ○ ○ ○ ○ ○ 2 ○	2019-09-30 00:00
On	example_branch_operator	@daily	airflow	1 ○ ○ ○ ○ ○ ○ ○ 9 1	2019-09-30 00:00
On	example_dynamic	1 day, 0:00:00	airflow	4 ○ ○ ○ ○ ○ ○ ○ 1 1	2019-01-01 00:00
On	example_http_operator	1 day, 0:00:00	airflow	1 ○ ○ ○ ○ ○ ○ ○ 4 1	2019-09-30 00:00
On	example_incremental	1 day, 0:00:00	airflow	○ ○ ○ ○ ○ ○ ○ ○ ○ 1	2019-01-01 00:00
On	example_not_incremental	1 day, 0:00:00	airflow	○ ○ ○ ○ ○ ○ ○ ○ ○ 1	2019-01-01 00:00
On	example_parallel	1 day, 0:00:00	airflow	○ ○ ○ ○ ○ ○ ○ ○ 4 1	2019-01-01 00:00

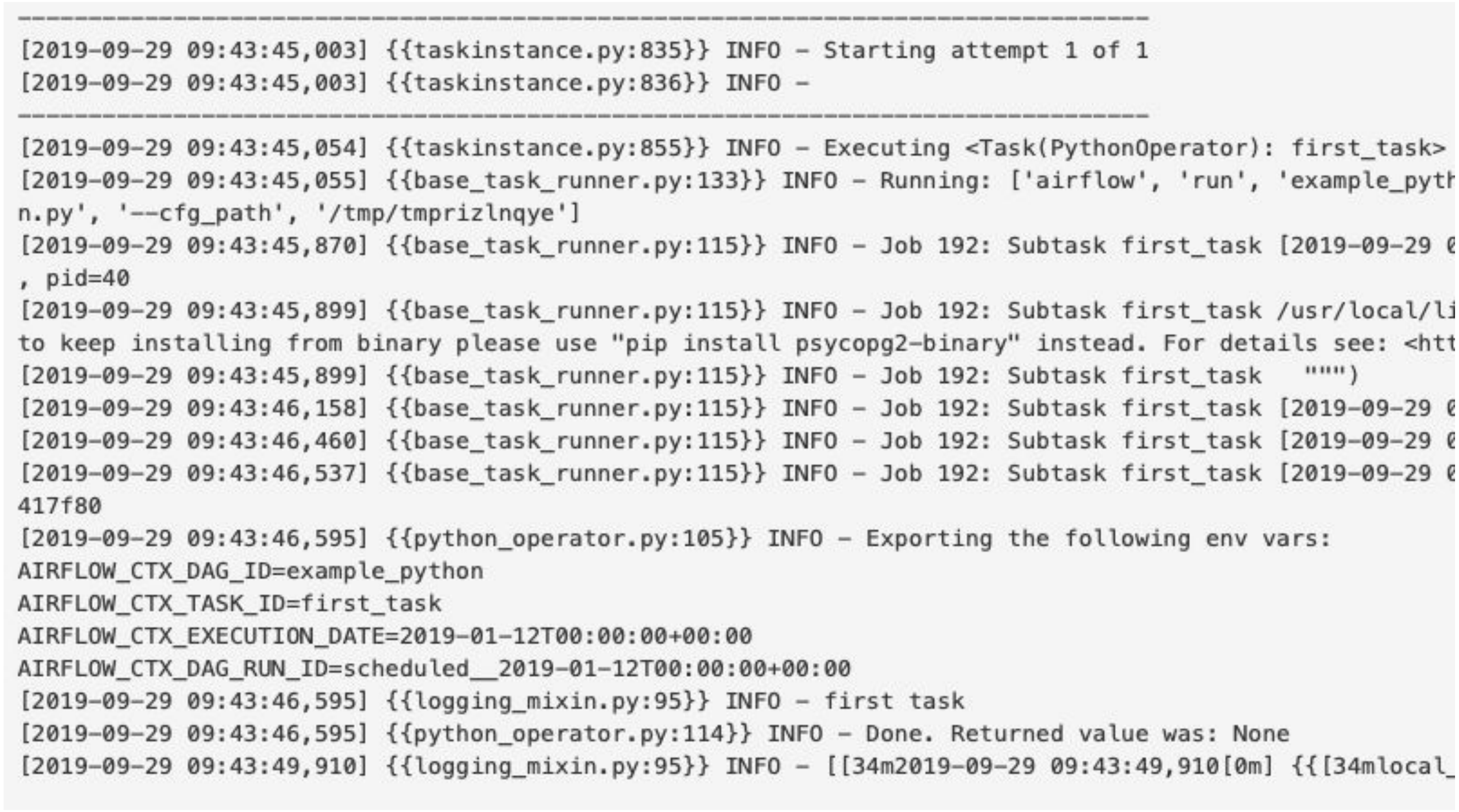


On DAG: example_python

Graph View | Tree View | Task Duration | Task Tries | Landing Times | Gantt | Details

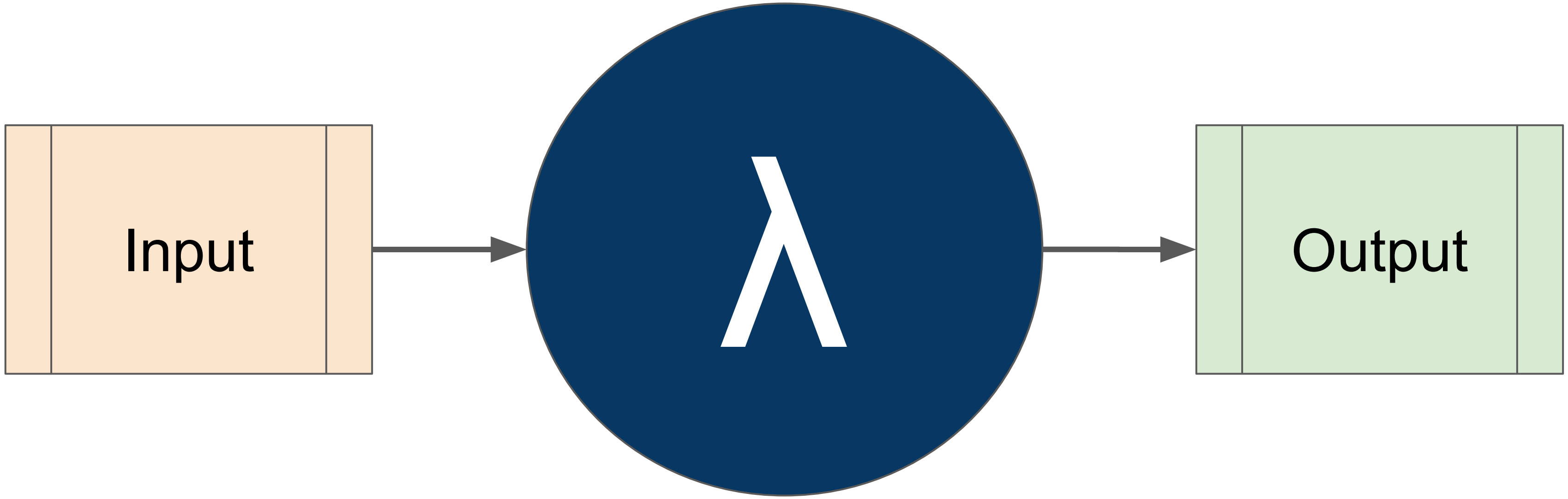
Base date: 2019-01-12 00:00:00+00 Number of runs: 25 Go

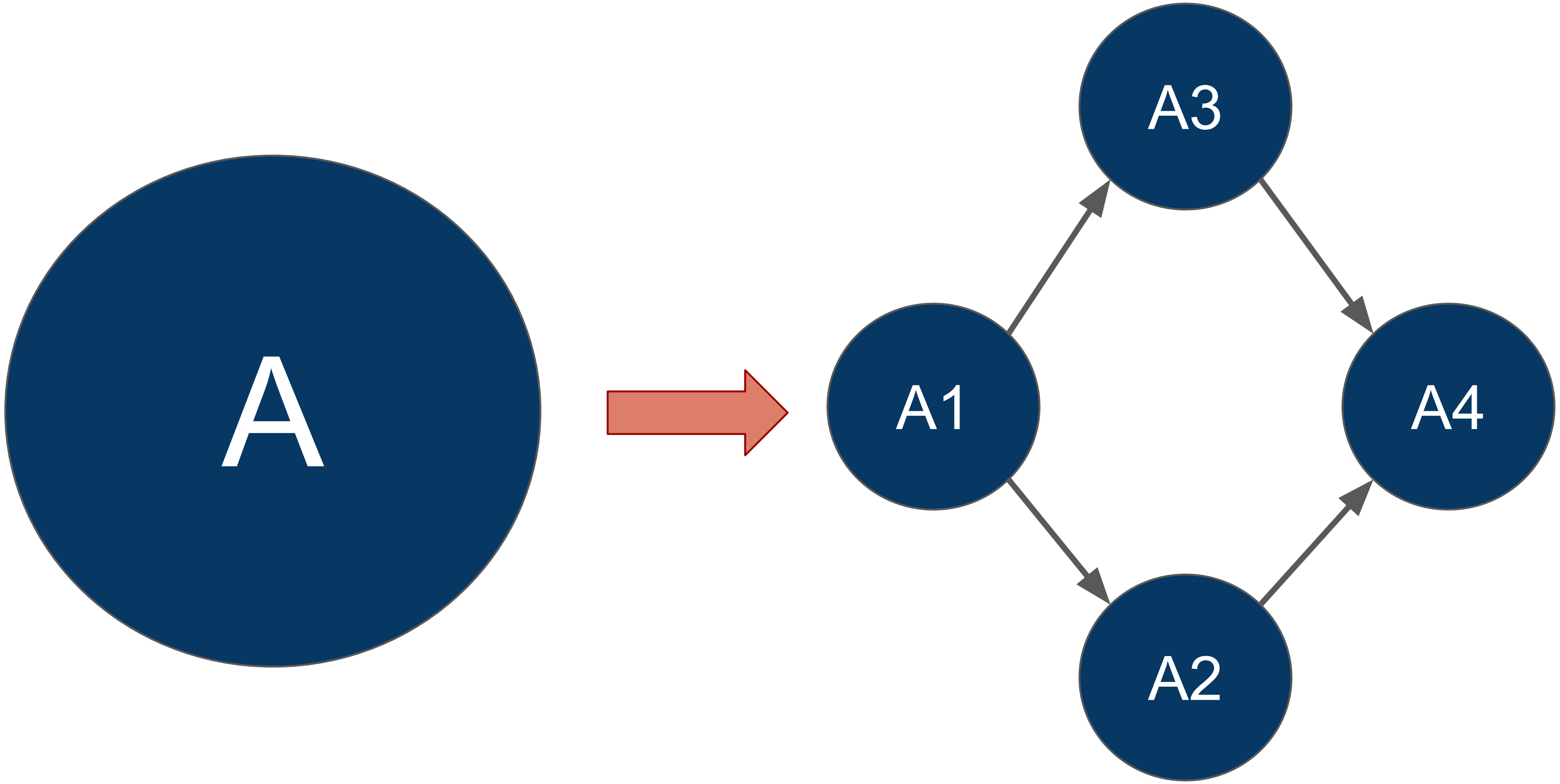
PythonOperator

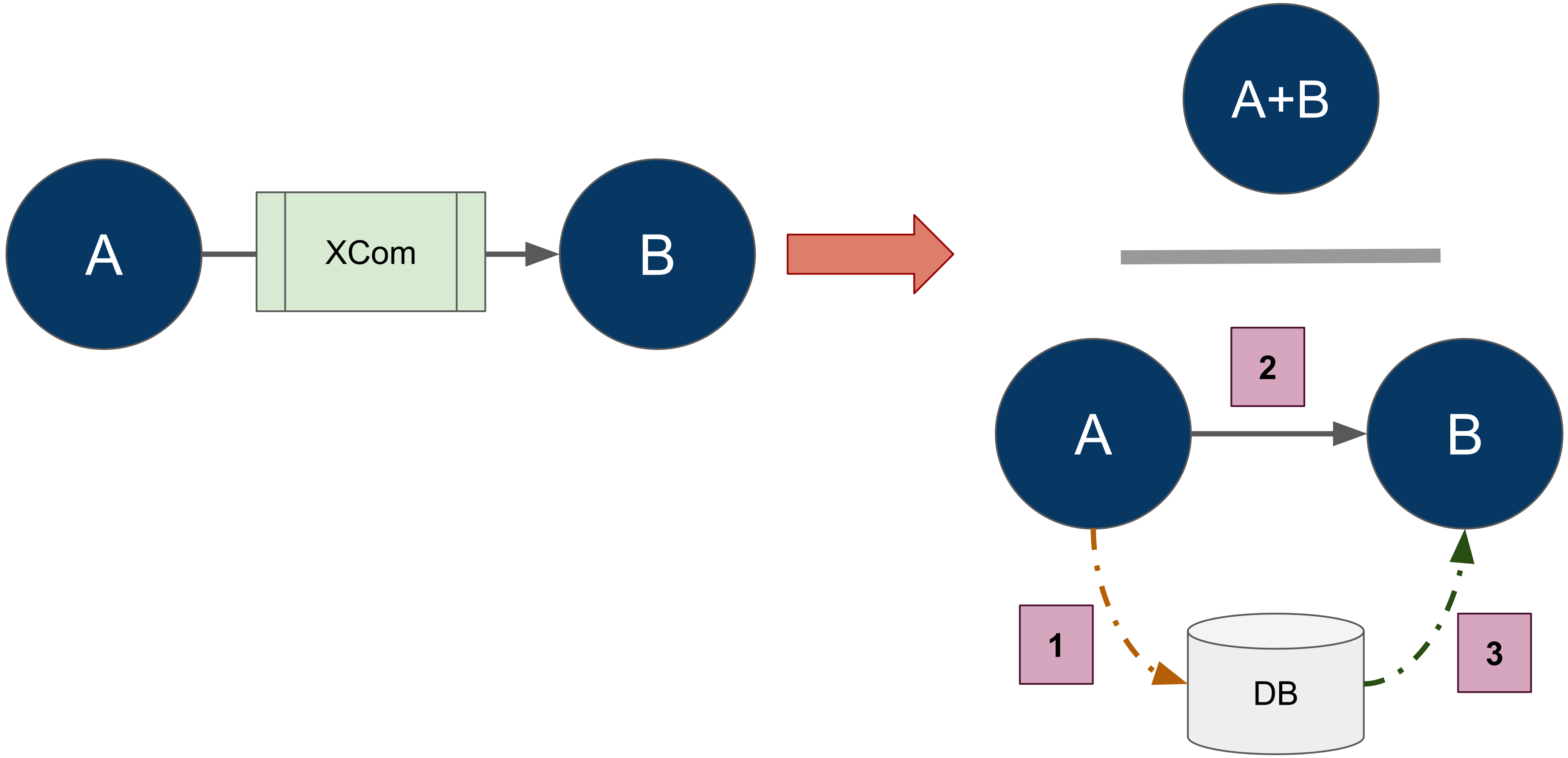


```
--------------------------------------------------------------------------------
[2019-09-29 09:43:45,003] {{taskinstance.py:835}} INFO - Starting attempt 1 of 1
[2019-09-29 09:43:45,003] {{taskinstance.py:836}} INFO -
--------------------------------------------------------------------------------
[2019-09-29 09:43:45,054] {{taskinstance.py:855}} INFO - Executing <Task(PythonOperator): first_task>
[2019-09-29 09:43:45,055] {{base_task_runner.py:133}} INFO - Running: ['airflow', 'run', 'example_pyth
n.py', '--cfg_path', '/tmp/tmprizlnqye']
[2019-09-29 09:43:45,870] {{base_task_runner.py:115}} INFO - Job 192: Subtask first_task [2019-09-29 0
, pid=40
[2019-09-29 09:43:45,899] {{base_task_runner.py:115}} INFO - Job 192: Subtask first_task /usr/local/li
to keep installing from binary please use "pip install psycopg2-binary" instead. For details see: <htt
[2019-09-29 09:43:45,899] {{base_task_runner.py:115}} INFO - Job 192: Subtask first_task   """)
[2019-09-29 09:43:46,158] {{base_task_runner.py:115}} INFO - Job 192: Subtask first_task [2019-09-29 0
[2019-09-29 09:43:46,460] {{base_task_runner.py:115}} INFO - Job 192: Subtask first_task [2019-09-29 0
[2019-09-29 09:43:46,537] {{base_task_runner.py:115}} INFO - Job 192: Subtask first_task [2019-09-29 0
417f80
[2019-09-29 09:43:46,595] {{python_operator.py:105}} INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_ID=example_python
AIRFLOW_CTX_TASK_ID=first_task
AIRFLOW_CTX_EXECUTION_DATE=2019-01-12T00:00:00+00:00
AIRFLOW_CTX_DAG_RUN_ID=scheduled__2019-01-12T00:00:00+00:00
[2019-09-29 09:43:46,595] {{logging_mixin.py:95}} INFO - first task
[2019-09-29 09:43:46,595] {{python_operator.py:114}} INFO - Done. Returned value was: None
[2019-09-29 09:43:49,910] {{logging_mixin.py:95}} INFO - [[34m2019-09-29 09:43:49,910[0m] {{[34mlocal_
```

Recomendaciones


Input
λ
Output


A
A1
A2
A3
A4

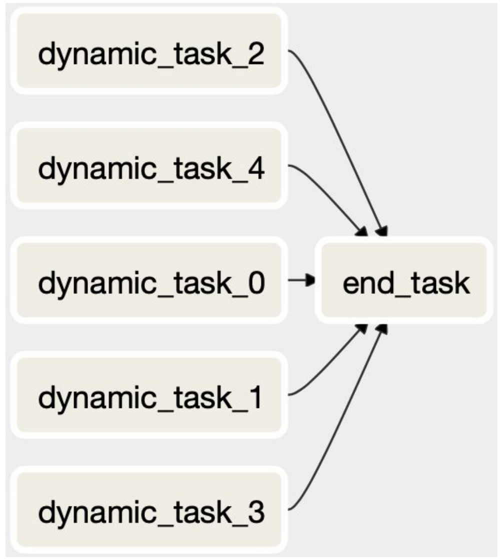

A
XCom
B
A+B
2
A
B
1
DB
3

Jinja

```python
with dag:
    task = BashOperator(
        task_id='task',
        bash_command='echo {{ ds }} {{ params.param }}',
        params={'param': 'value'},
    )
```

```
INFO - Temporary script location: /tmp/airflowtmpp3mwola8/taskqtvtld1h
INFO - Running command: echo 2019-01-11 value
INFO - Output:
INFO - 2019-01-11 value
INFO - Command exited with return code 0
```

```python
with dag:

    dynamic_tasks = []
    for task_id in range(5):
        dynamic_task = BashOperator(
            task_id='dynamic_task_' + str(task_id),
            bash_command='echo {{ task.task_id }}',
        )
        dynamic_tasks.append(dynamic_task)

    end_task = BashOperator(
            task_id='end_task',
            bash_command='echo {{ task.task_id }}',
    )
    end_task.set_upstream(dynamic_tasks)
```



```
with dag:

    sql = '''
        DELETE FROM daily_totals;

        INSERT INTO daily_totals
        SELECT
            day,
            COUNT(*) AS total
        FROM
            table
        GROUP BY
            day;
    '''

    non_incremental_task = PostgresOperator(
        task_id='non_incremental_task',
        sql=sql,
    )
```

```
with dag:

    sql = '''
        DELETE FROM daily_totals
        WHERE day = '{{ ds }}';

        INSERT INTO daily_totals
        SELECT
            day,
            COUNT(*) AS total
        FROM
            table
        WHERE
            day = '{{ ds }}';
    '''

    incremental_task = PostgresOperator(
        task_id='incremental_task',
        sql=sql,
    )
```

```yaml
# YAML file
task_id:
  incremental_task
sql:
  DELETE FROM daily_totals
  WHERE day = '{{ ds }}';

  INSERT INTO daily_totals
  SELECT
    day,
    COUNT(*) AS total
  FROM
    table
  WHERE
    day = '{{ ds }}';
```

```python
with dag:
    for filename in glob.glob(YAML_DIR + '/*.yaml'):
        with open(filename, 'r') as stream:
            yaml_data = yaml.safe_load(stream)

            incremental_task = PostgresOperator(
                task_id=yaml_data['task_id'],
                sql=yaml_data['sql'],
            )
```

Jordi Contestí

habitissimo DATA LABS

¡Gracias!