Apache Airflow

大数据领域非常热门的开源任务编排框架

以code形式构建和执行调度工作流

https://airflow.ecdataway.com/

主要场景：

- data pipeline： ETL
- data analysis
- data preparing for machine learning

优势：

Web UI 提供

- 任务监控
- 日志查询
- 工作流管理
- 任务依赖管理

对比服务器CRON配置：

可在网页直接观察每日任务执行情况

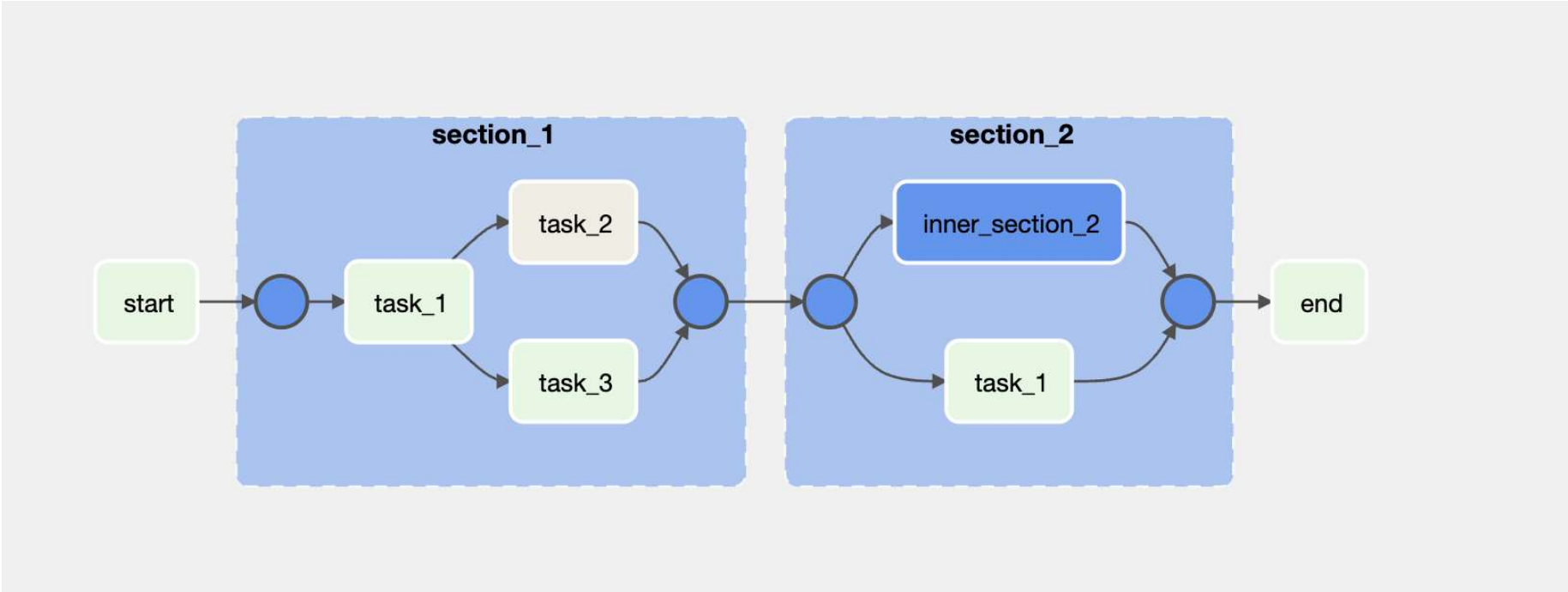
Airflow DAGs Datasets Security Browse Admin Docs 14:38 CST (+08:00) AA

DAGs

All 20 | Active 16 | Paused 4 | × jd_api | Search DAGs | Auto-refresh

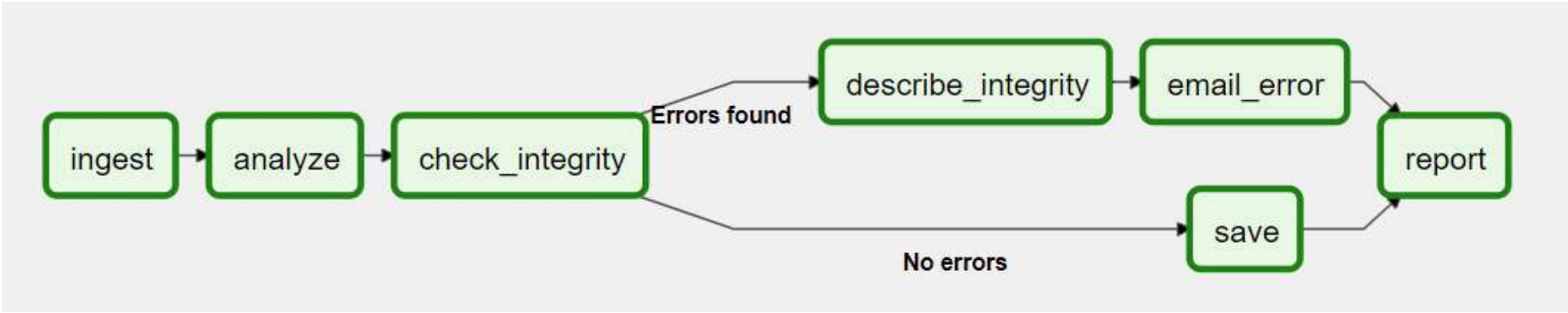
DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
check_shop jd_api jd_api_qdc	yang.jingming	60	0 8 * * *	2022-10-25, 08:00:00	2022-10-26, 08:00:00	1		
check_stock_comment_fav jd_api jd_api_qdc	yang.jingming	80 7	0 12 * * *	2022-10-25, 12:00:00	2022-10-26, 12:00:00	6		
dag_itemsku_move jd_api jd_api_daily	yang.xiaopeng	14	0 3 * * *	2022-10-25, 03:00:00	2022-10-26, 03:00:00	1		
dag_mysql_replication jd_api jd_api_backup	yang.jingming	858 17	*/20 * * * *	2022-10-26, 14:00:00	2022-10-26, 14:20:00	1		
dag_shop_license_parse_tb jd_api	yang.jingming	33	1 day, 0:00:00	2022-10-16, 00:00:00	2022-10-17, 00:00:00	1		
dag_shop_license_tb jd_api jd_api_shop_license	yang.jingming	1 2	0 0 15 * *	2022-09-15, 00:00:00	2022-11-15, 00:00:00	2		
dag_shop_license_tmall jd_api jd_api_shop_license	yang.jingming	1	0 0 15 * *	2022-10-17, 11:25:51	2022-11-15, 00:00:00	2		
dag_sku_linshi adhoc jd_api	yang.jingming	3 3	None	2022-10-10, 13:28:12		1		
dag_sku_source_to_tmp jd_api jd_api_daily	yang.xiaopeng	6 1	0 13 * * *	2022-10-25, 13:00:00	2022-10-26, 13:00:00	1 2		
dag_sku_tmp_to_target jd_api jd_api_daily	yang.xiaopeng	7	0 0 * * *	2022-10-25, 00:00:00	2022-10-26, 00:00:00	3		
dag_yushou_daily jd_api jd_api_daily jd_api_yushou	yang.jingming	9	0 0 * * *	2022-10-25, 00:00:00	2022-10-26, 00:00:00	1		
dag_yushou_hourly								

可以做复杂的调度依赖管理



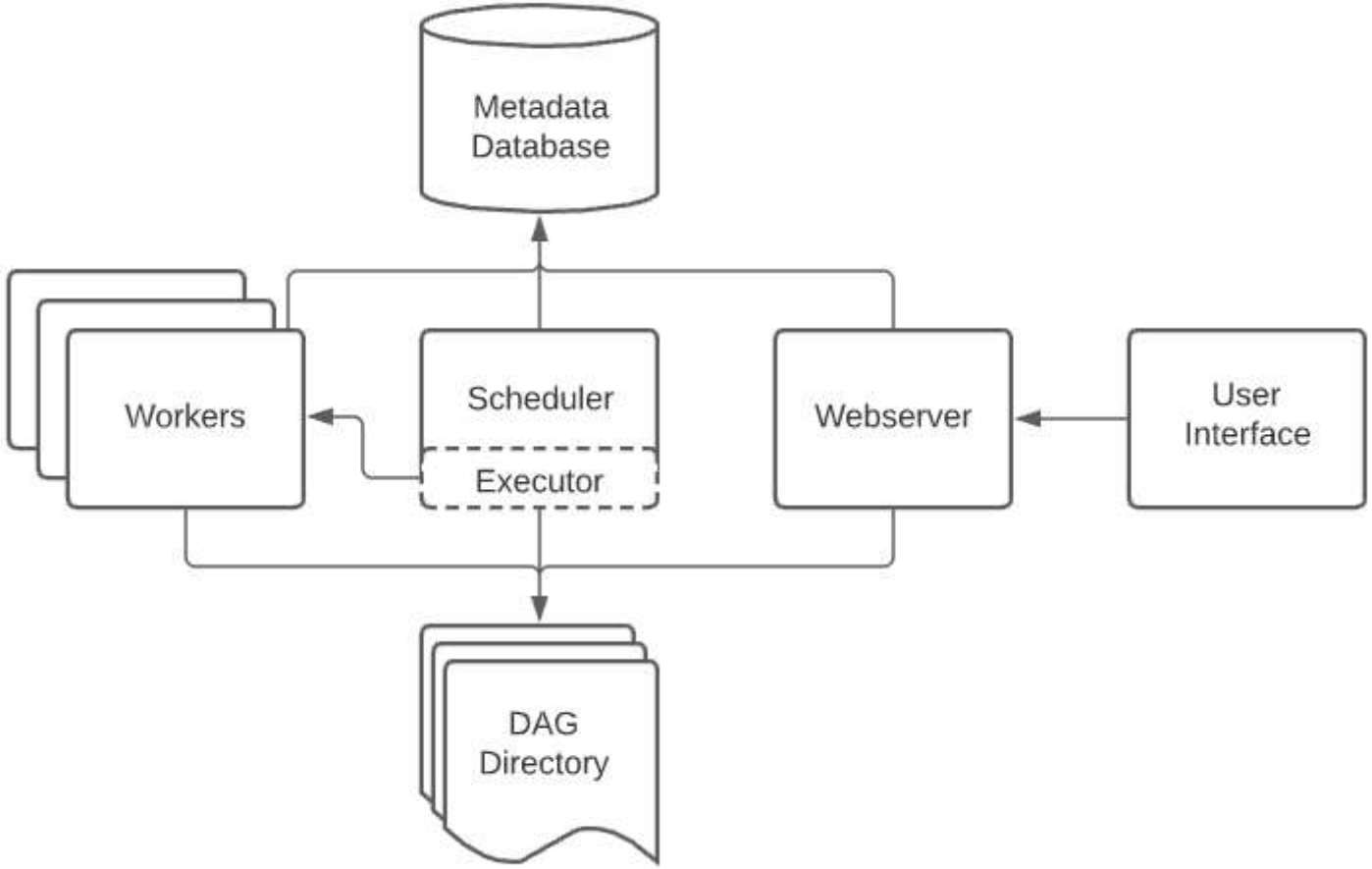
架构与核心概念

- **有向无环图 DAG** （Directed Acrylic Graphs）
- **DAG** 细分成不同的tasks任务



参考

架构


Metadata
Database
Workers
Scheduler
Executor
Webserver
User
Interface
DAG
Directory

定义TASKS

airflow 提供丰富的class

- Operators

  `BashOperator`, `PythonOperator`, `MysqlOperator`, `SparkOperator` ...

- Sensor

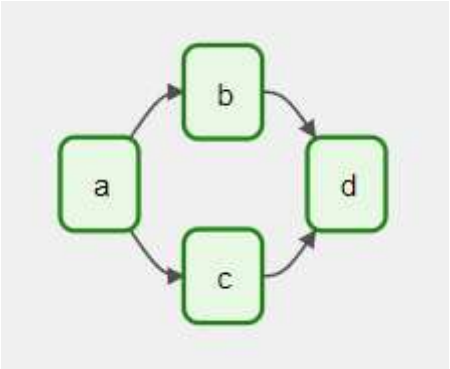
  `FileSensor`, `ExternalTaskSensor` ...

定义控制流

airflow提供的python语法述task之间依赖关系

```
task_a = ...
task_b = ...
task_c = ...
task_d = ...

task_a >> [task_b, task_c] >> task_d
```



example_complex - Graph - Airflow (ecdataway.com)

定义调度间隔

- cron表达式语法

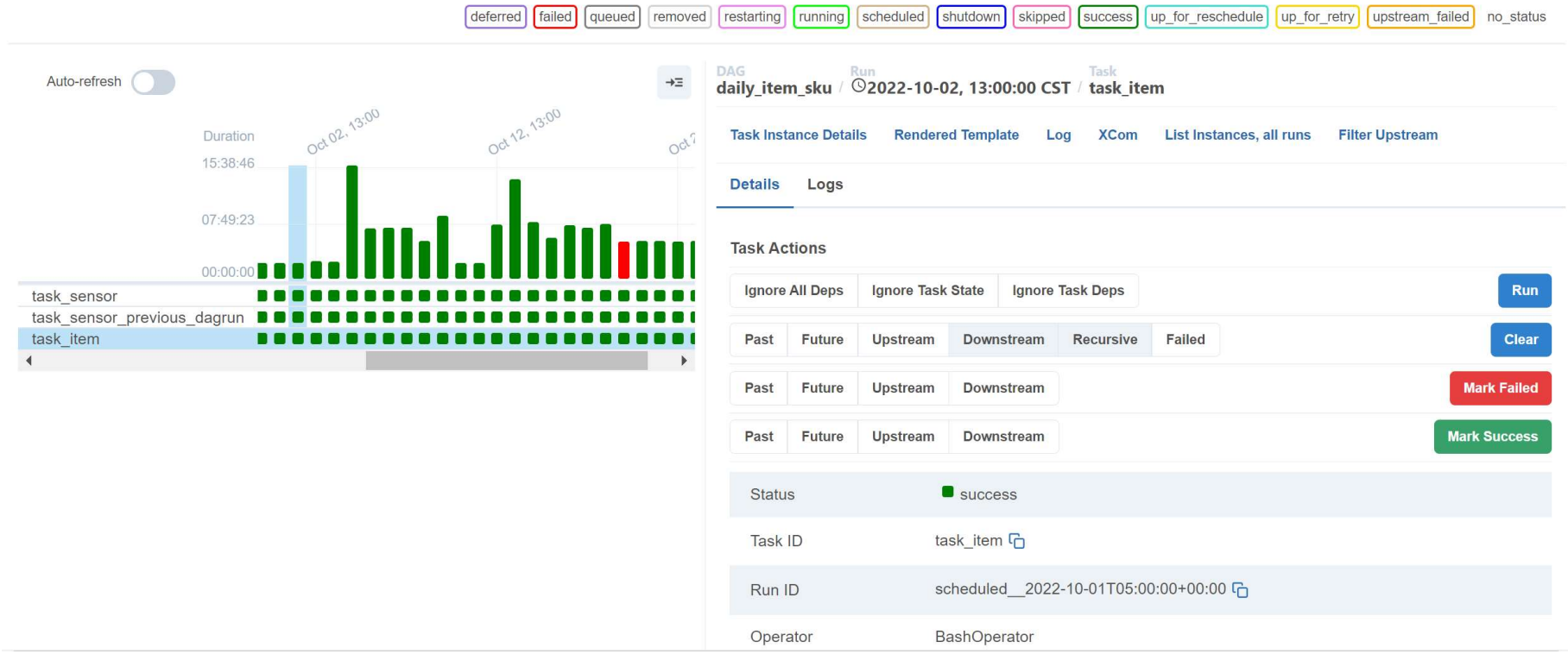
```
"0 1 * * *"
```

- 预设

```
"@once",
"@hourly"
...
```

调度启停

定义DAG的时候要考虑到幂等性和原子性



模板导入宏变量

通过jinja模板引擎传入宏变量

```
templated_command = dedent(
    """
{% for i in range(5) %}
    echo "{{ ds }}"
    echo "{{ macros.ds_add(ds, 7)}}"
{% endfor %}
"""
)

t3 = BashOperator(
    task_id='templated',
    depends_on_past=False,
    bash_command=templated_command,
)
```

AIRFLOW DAG 编写

```
1  from datetime import datetime, timedelta
2  from textwrap import dedent
3  import pendulum
4  # The DAG object; we'll need this to instantiate a DAG
5  from airflow import DAG
6
7  # Operators; we need this to operate!
8  from airflow.operators.bash import BashOperator
9  with DAG(
10     'tutorial',
11     description='A simple tutorial DAG',
12     schedule_interval="0 1 * * *",
13     start_date=pendulum.datetime(2022, 9, 22, tz='Asia/Sha
14     catchup=False,
15     tags=['example'],
```

AIRFLOW DAG 编写

```
16      # 传入给task的默认参数
17      default_args={

18          'depends_on_past': False,
19          'email': ['airflow@example.com'],
20          'email_on_failure': False,
21          'email_on_retry': False,
22          'retries': 1,
23          'retry_delay': timedelta(minutes=5),
24          # 'wait_for_downstream': False,
25          # 'execution_timeout': timedelta(seconds=300),
26          # 'on_failure_callback': some_function,
27          # 'on_success_callback': some_other_function,
28          # 'on_retry_callback': another_function,
29          # 'trigger_rule': 'all_success'
30      },
```

AIRFLOW DAG 编写

```
58      )
59
60      t1 >> [t2, t3]
61
62      t1.doc_md = dedent(
63          """\
64      #### Task Documentation
65      You can document your task using the attributes `doc_m
66      `doc` (plain text), `doc_rst`, `doc_json`, `doc_yaml`
67      rendered in the UI's Task Instance Details page.

68      ![img](http://montcs.bloomu.edu/~bobmon/Semesters/2012
69      **Image Credit:** Randall Munroe, [XKCD](https://xkcd
70      """
71      )
72      dag.doc_md = __doc__  # providing that you have a docs
```

AIRFLOW DAG 编写

```
62      t1.doc_md = dedent(
63          """\
64      #### Task Documentation
65      You can document your task using the attributes `doc_r
66      `doc` (plain text), `doc_rst`, `doc_json`, `doc_yaml`
67      rendered in the UI's Task Instance Details page.

68      ![img](http://montcs.bloomu.edu/~bobmon/Semesters/2012
69      **Image Credit:** Randall Munroe, [XKCD](https://xkcd
70      """
71      )
72      dag.doc_md = __doc__  # providing that you have a docs
73      dag.doc_md = """
74      This is a documentation placed anywhere
75      """ # otherwise, type it like this
```

AIRFLOW DAG 编写

```
62      t1.doc_md = dedent(
63          """\
64      #### Task Documentation
65      You can document your task using the attributes `doc_r
66      `doc` (plain text), `doc_rst`, `doc_json`, `doc_yaml`
67      rendered in the UI's Task Instance Details page.

68      ![img](http://montcs.bloomu.edu/~bobmon/Semesters/2012
69      **Image Credit:** Randall Munroe, [XKCD](https://xkcd
70      """
71      )
72      dag.doc_md = __doc__  # providing that you have a docs
73      dag.doc_md = """
74      This is a documentation placed anywhere
75      """ # otherwise, type it like this
```

AIRFLOW DAG 编写

```
62      t1.doc_md = dedent(
63          """\
64      #### Task Documentation
65      You can document your task using the attributes `doc_r
66      `doc` (plain text), `doc_rst`, `doc_json`, `doc_yaml`
67      rendered in the UI's Task Instance Details page.

68      ![img](http://montcs.bloomu.edu/~bobmon/Semesters/201
69      **Image Credit:** Randall Munroe, [XKCD](https://xkcd
70      """
71      )
72      dag.doc_md = __doc__  # providing that you have a docs
73      dag.doc_md = """
74      This is a documentation placed anywhere
75      """ # otherwise, type it like this
```

查看DAG

一些细节

PYTHON OPERATOR

1. 基础 API（1.0版本）
2. taskflow API（2.0版本）

基础PythonOperator定义

```
from airflow.operators.python import PythonOperator

def insert_category_mysql(month):
    pass

task_category = PythonOperator(
        task_id='task_category',
        python_callable=insert_category_mysql,
        op_kwargs={
            'month': '{{data_interval_start.in_timezone("Asia
        }
    )
```

taskflow定义

```
from airflow.decorators import task

@task
def task_category(data_interval_start=None):
    month = data_interval_start.in_timezone("Asia/Shanghai").
    pass

task_category = task_category()
```

渐进式整合PYTHON脚本

不耦合	部分耦合	完全耦合
`BashOperator` 调用Python脚本	`PythonOperator` 调用Python函数	TaskFlow API 装饰器封装Python函数

最佳实践

将业务逻辑和调度定义逻辑分开

在调度逻辑中使用业务函数

- `dag_some_task.py`：调度脚本。定义airflow相关的DAG、Tasks等。
- `some_task.py`：业务逻辑脚本。将业务封装成不同的函数。

DAGS仓库

Airflow需要的DAGs仓库

```
- src/
  - utils/
  - dags/
    - yjm/
      - dags_1.py
      ...
    - yxp/
```

服务器自动同步仓库代码

TOOLKIT仓库

解决不同Python仓库都要使用的公司工具类

在各个仓库不一致的问题

```
- config
  ...
- lib
  - PyMysqlPool.py
  - MyCkClient.py
  ...
```

nint_tookkit仓库地址

安装

```
pip install \
git+ssh://git@git.sh.nint.com/yang.jingming/nint_toolkit.git@
```

使用

```
from nint_toolkit.lib.MyCkClient import MyCkClient
```

谢谢

Q&A