Dynamic Workflows with Airflow and Python

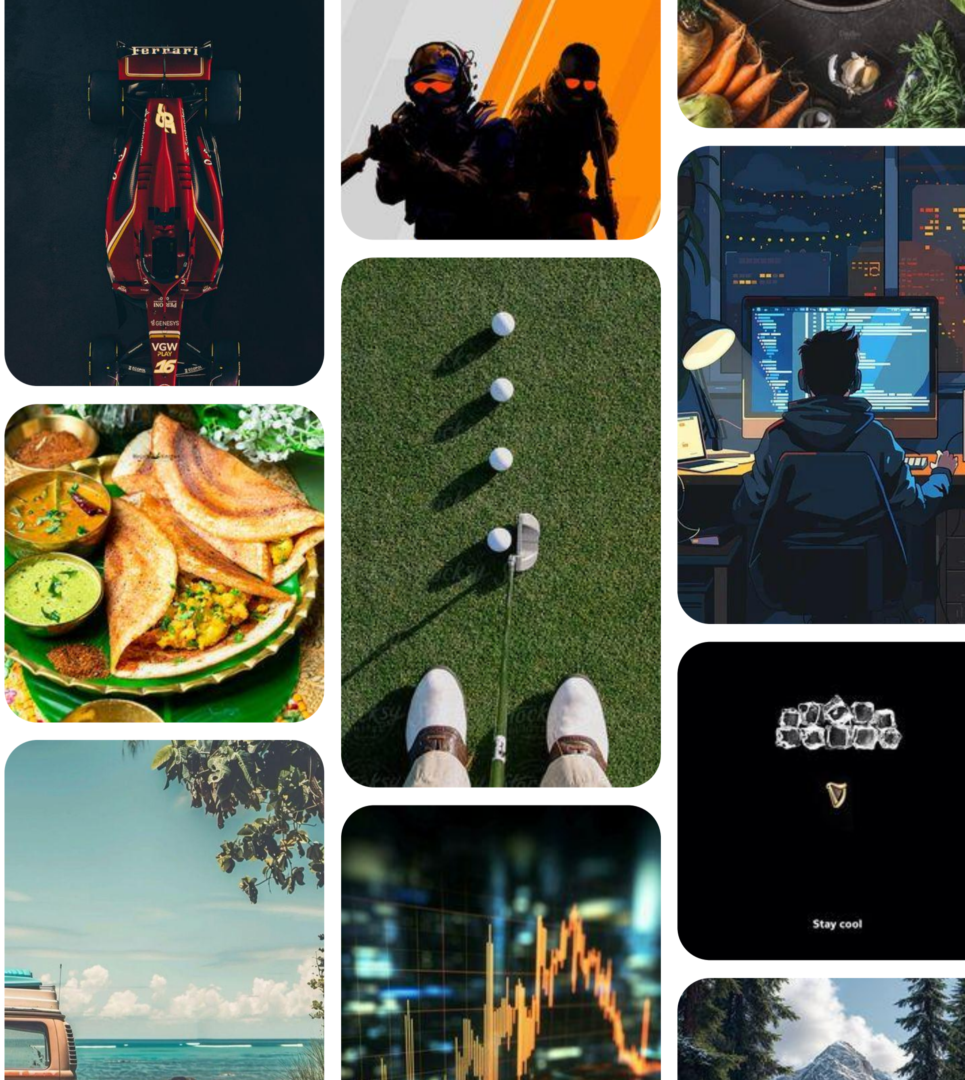
By Thejas Raju

PyLadies Dublin
Date: September 17, 2024

Thejas Raju

He/Him

Software Engineer
Pinterest



Agenda

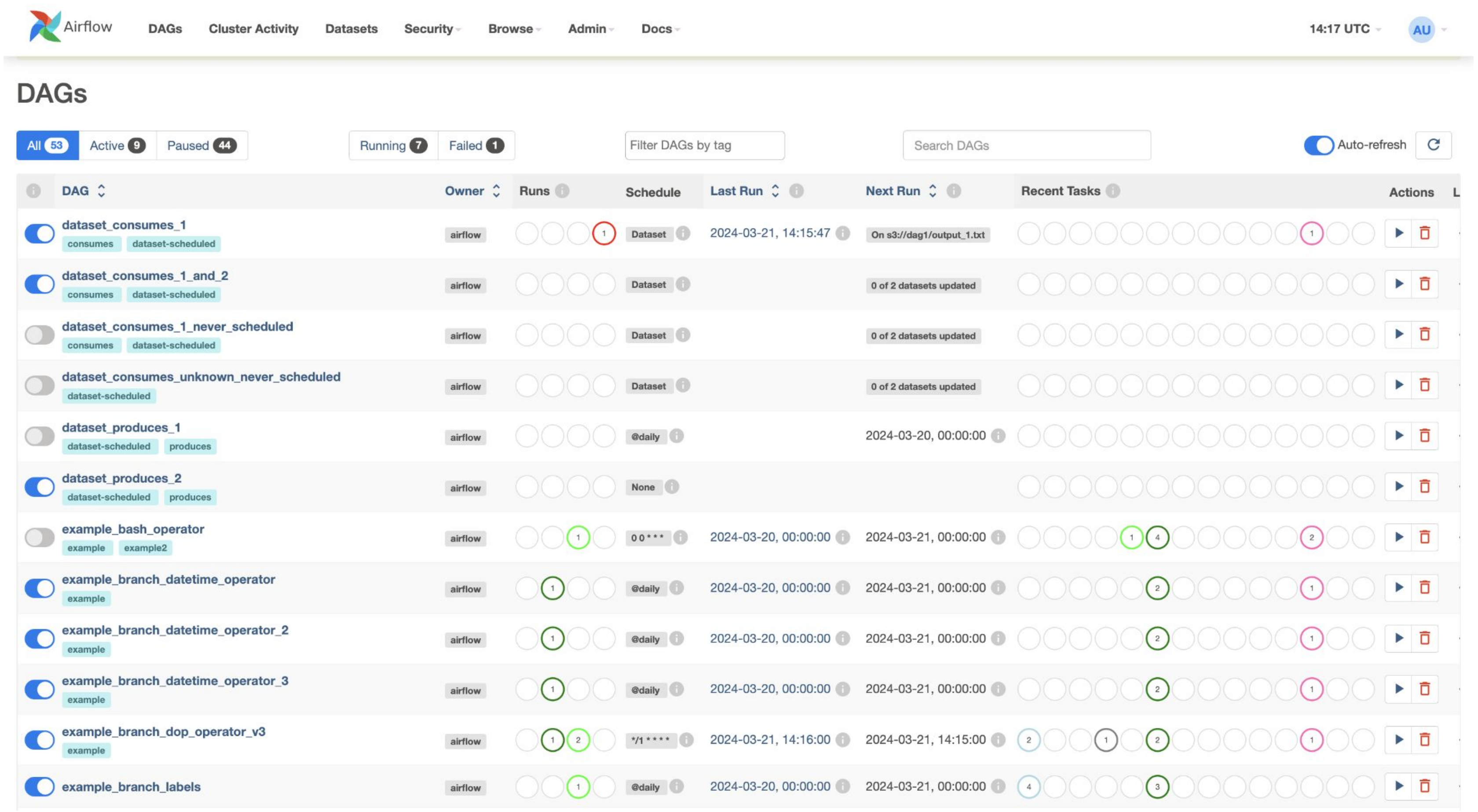
Apache Airflow is a platform created by the community to programmatically author, schedule and monitor workflows.

Features

Pure python & flexibility

Useful UI & integrations

Ease of use & open source community

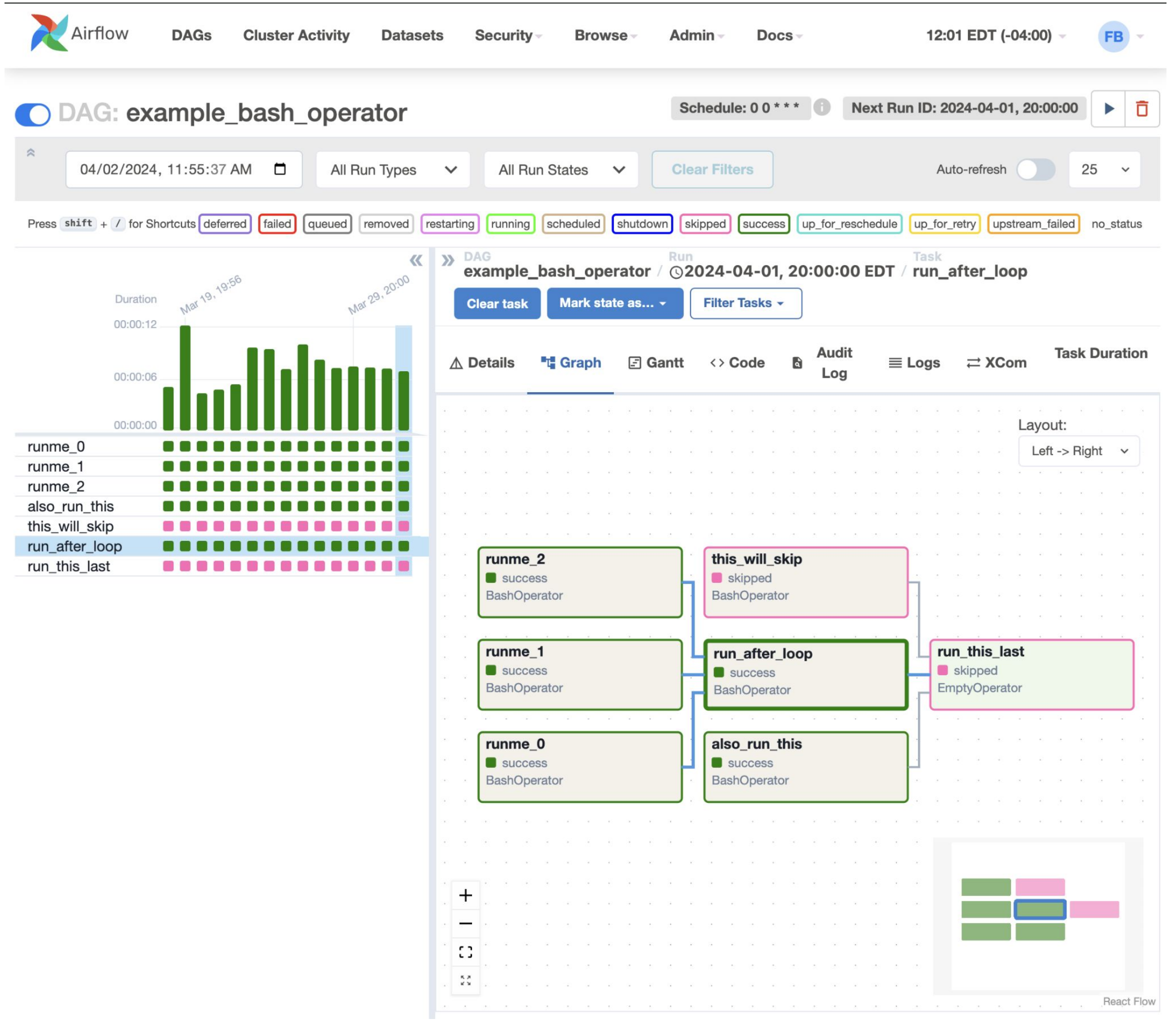


DAGs

All 53 | Active 9 | Paused 44 | Running 7 | Failed 1

Filter DAGs by tag | Search DAGs | Auto-refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions
dataset_consumes_1 consumes dataset-scheduled	airflow	1	Dataset	2024-03-21, 14:15:47	On s3://dag1/output_1.txt	1	
dataset_consumes_1_and_2 consumes dataset-scheduled	airflow		Dataset		0 of 2 datasets updated		
dataset_consumes_1_never_scheduled consumes dataset-scheduled	airflow		Dataset		0 of 2 datasets updated		
dataset_consumes_unknown_never_scheduled dataset-scheduled	airflow		Dataset		0 of 2 datasets updated		
dataset_produces_1 dataset-scheduled produces	airflow		@daily		2024-03-20, 00:00:00		
dataset_produces_2 dataset-scheduled produces	airflow		None				
example_bash_operator example example2	airflow	1	0 0 * * *	2024-03-20, 00:00:00	2024-03-21, 00:00:00	1 4 2	
example_branch_datetime_operator example	airflow	1	@daily	2024-03-20, 00:00:00	2024-03-21, 00:00:00	2 1	
example_branch_datetime_operator_2 example	airflow	1	@daily	2024-03-20, 00:00:00	2024-03-21, 00:00:00	2 1	
example_branch_datetime_operator_3 example	airflow	1	@daily	2024-03-20, 00:00:00	2024-03-21, 00:00:00	2 1	
example_branch_dop_operator_v3 example	airflow	1 2	*/1 * * * *	2024-03-21, 14:16:00	2024-03-21, 14:15:00	2 1 2 1	
example_branch_labels	airflow	1	@daily	2024-03-20, 00:00:00	2024-03-21, 00:00:00	4 3	



DAG: example_bash_operator

Schedule: 0 0 * * *

Next Run ID: 2024-04-01, 20:00:00

04/02/2024, 11:55:37 AM
All Run Types
All Run States
Clear Filters
Auto-refresh
25

Press shift + / for Shortcuts deferred failed queued removed restarting running scheduled shutdown skipped success up_for_reschedule up_for_retry upstream_failed no_status

DAG example_bash_operator / Run 2024-04-01, 20:00:00 EDT / Task run_after_loop

runme_2 success BashOperator
runme_1 success BashOperator
runme_0 success BashOperator
this_will_skip skipped BashOperator
run_after_loop success BashOperator
also_run_this success BashOperator
run_this_last skipped EmptyOperator

2024 Pinterest
Reference

Spinner at Pinterest

1.3k+ active users

14k+ daily DAG runs

131k+ daily task instances

160K+ daily UI/API requests

Ref: Airflow Summit 2022

Workflows as Code

The main characteristic of Airflow workflows is that all workflows are defined in Python code.

Workflows as Code

The main characteristic of Airflow workflows is that all workflows are defined in Python code.

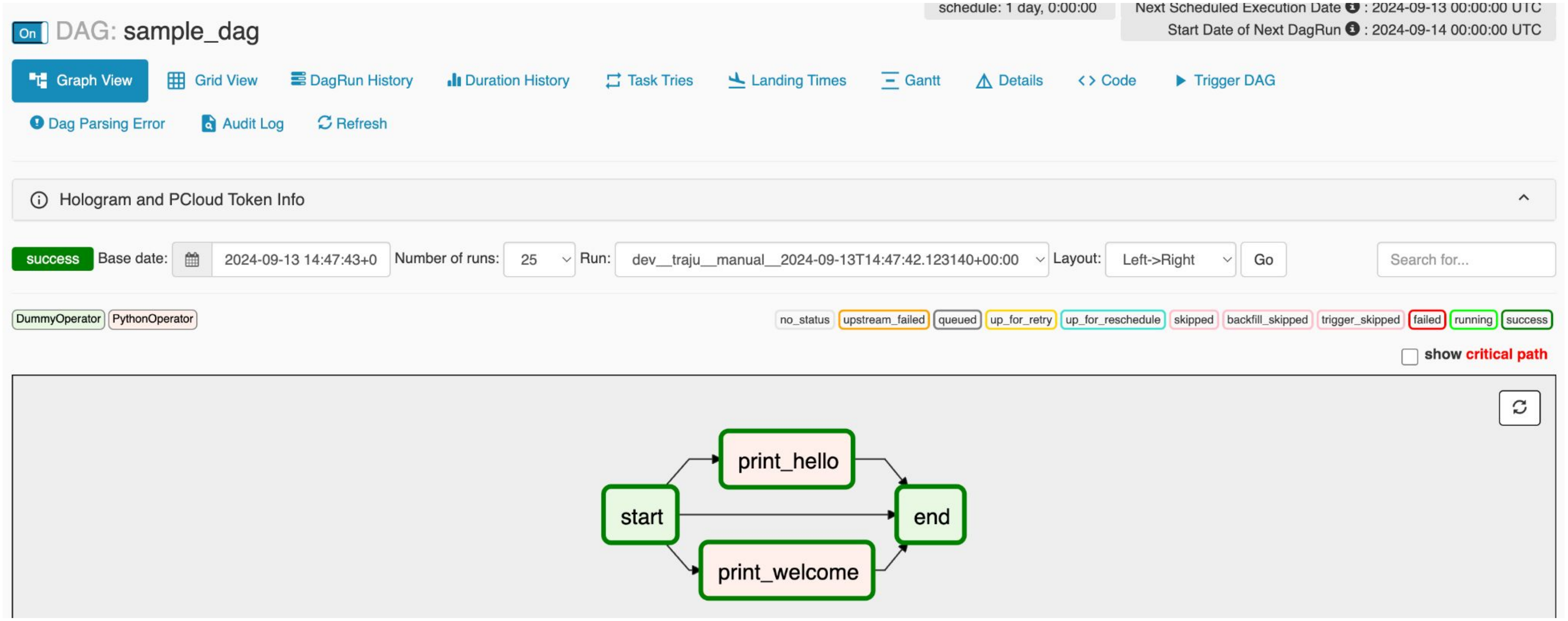
Directed Acyclic Graphs (DAG)

A DAG is a collection of all the tasks you want to run, organized in a way that explicitly lays out their dependencies and execution order in a workflow.

Tasks, Operators, and Dependencies

Tasks are the individual units of work, operators define specific types of tasks, and dependencies are the relationships determining the order in which tasks run.

```
from airflow import DAG
from airflow.operators.dummy_operator import DummyOperator
from airflow.operators.python_operator import PythonOperator
from datetime import datetime

with DAG('sample_dag', start_date=datetime(2024, 9, 10)) as dag:

    start = DummyOperator(task_id='start')

    print_hello = PythonOperator(task_id='print_hello',
python_callable=lambda: print('Hello PyLadies Dublin!'))

    print_welcome = PythonOperator(task_id='print_welcome',
python_callable=lambda: print('Welcome to Pinterest!'))

    end = DummyOperator(task_id='end')

    start >> [print_hello, print_welcome] >> end
    start >> end
```



schedule: 1 day, 0:00:00

Next Scheduled Execution Date : 2024-09-13 00:00:00 UTC

Start Date of Next DagRun : 2024-09-14 00:00:00 UTC

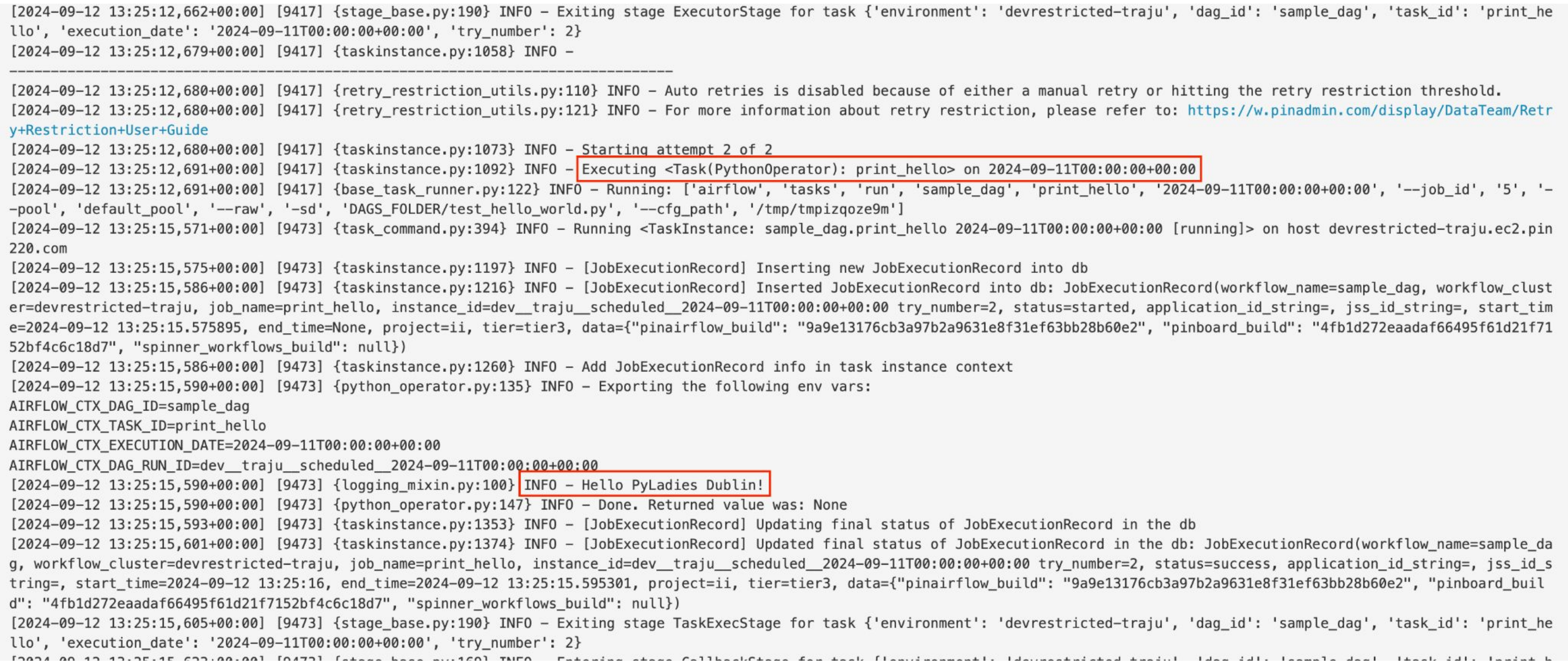
On DAG: sample_dag

Base date: 2024-09-13 14:47:43+0

Number of runs: 25

Run: dev__traju__manual__2024-09-13T14:47:42.123140+00:00

Layout: Left->Right



```
[2024-09-12 13:25:12,662+00:00] [9417] {stage_base.py:190} INFO - Exiting stage ExecutorStage for task {'environment': 'devrestricted-traju', 'dag_id': 'sample_dag', 'task_id': 'print_he
llo', 'execution_date': '2024-09-11T00:00:00+00:00', 'try_number': 2}
[2024-09-12 13:25:12,679+00:00] [9417] {taskinstance.py:1058} INFO -
--------------------------------------------------------------------------------
[2024-09-12 13:25:12,680+00:00] [9417] {retry_restriction_utils.py:110} INFO - Auto retries is disabled because of either a manual retry or hitting the retry restriction threshold.
[2024-09-12 13:25:12,680+00:00] [9417] {retry_restriction_utils.py:121} INFO - For more information about retry restriction, please refer to: https://w.pinadmin.com/display/DataTeam/Retr
y+Restriction+User+Guide
[2024-09-12 13:25:12,680+00:00] [9417] {taskinstance.py:1073} INFO - Starting attempt 2 of 2
[2024-09-12 13:25:12,691+00:00] [9417] {taskinstance.py:1092} INFO - Executing <Task(PythonOperator): print_hello> on 2024-09-11T00:00:00+00:00
[2024-09-12 13:25:12,691+00:00] [9417] {base_task_runner.py:122} INFO - Running: ['airflow', 'tasks', 'run', 'sample_dag', 'print_hello', '2024-09-11T00:00:00+00:00', '--job_id', '5', '-
-pool', 'default_pool', '--raw', '-sd', 'DAGS_FOLDER/test_hello_world.py', '--cfg_path', '/tmp/tmpizqoze9m']
[2024-09-12 13:25:15,571+00:00] [9473] {task_command.py:394} INFO - Running <TaskInstance: sample_dag.print_hello 2024-09-11T00:00:00+00:00 [running]> on host devrestricted-traju.ec2.pin
220.com
[2024-09-12 13:25:15,575+00:00] [9473] {taskinstance.py:1197} INFO - [JobExecutionRecord] Inserting new JobExecutionRecord into db
[2024-09-12 13:25:15,586+00:00] [9473] {taskinstance.py:1216} INFO - [JobExecutionRecord] Inserted JobExecutionRecord into db: JobExecutionRecord(workflow_name=sample_dag, workflow_clust
er=devrestricted-traju, job_name=print_hello, instance_id=dev__traju__scheduled__2024-09-11T00:00:00+00:00 try_number=2, status=started, application_id_string=, jss_id_string=, start_tim
e=2024-09-12 13:25:15.575895, end_time=None, project=ii, tier=tier3, data={"pinairflow_build": "9a9e13176cb3a97b2a9631e8f31ef63bb28b60e2", "pinboard_build": "4fb1d272eaadaf66495f61d21f71
52bf4c6c18d7", "spinner_workflows_build": null})
[2024-09-12 13:25:15,586+00:00] [9473] {taskinstance.py:1260} INFO - Add JobExecutionRecord info in task instance context
[2024-09-12 13:25:15,590+00:00] [9473] {python_operator.py:135} INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_ID=sample_dag
AIRFLOW_CTX_TASK_ID=print_hello
AIRFLOW_CTX_EXECUTION_DATE=2024-09-11T00:00:00+00:00
AIRFLOW_CTX_DAG_RUN_ID=dev__traju__scheduled__2024-09-11T00:00:00+00:00
[2024-09-12 13:25:15,590+00:00] [9473] {logging_mixin.py:100} INFO - Hello PyLadies Dublin!
[2024-09-12 13:25:15,590+00:00] [9473] {python_operator.py:147} INFO - Done. Returned value was: None
[2024-09-12 13:25:15,593+00:00] [9473] {taskinstance.py:1353} INFO - [JobExecutionRecord] Updating final status of JobExecutionRecord in the db
[2024-09-12 13:25:15,601+00:00] [9473] {taskinstance.py:1374} INFO - [JobExecutionRecord] Updated final status of JobExecutionRecord in the db: JobExecutionRecord(workflow_name=sample_da
g, workflow_cluster=devrestricted-traju, job_name=print_hello, instance_id=dev__traju__scheduled__2024-09-11T00:00:00+00:00 try_number=2, status=success, application_id_string=, jss_id_s
tring=, start_time=2024-09-12 13:25:16, end_time=2024-09-12 13:25:15.595301, project=ii, tier=tier3, data={"pinairflow_build": "9a9e13176cb3a97b2a9631e8f31ef63bb28b60e2", "pinboard_buil
d": "4fb1d272eaadaf66495f61d21f7152bf4c6c18d7", "spinner_workflows_build": null})
[2024-09-12 13:25:15,605+00:00] [9473] {stage_base.py:190} INFO - Exiting stage TaskExecStage for task {'environment': 'devrestricted-traju', 'dag_id': 'sample_dag', 'task_id': 'print_he
llo', 'execution_date': '2024-09-11T00:00:00+00:00', 'try_number': 2}
[2024-09-12 13:25:15,623+00:00] [9473] {stage_base.py:169} INFO - Entering stage CallbackStage for task {'environment': 'devrestricted-traju', 'dag_id': 'sample_dag', 'task_id': 'print_h
```

Principles

Scalable

Elegant

Extensible

Dynamic

Dynamic Workflows

Workflows that can automatically adjust and scale based on varying inputs, conditions, or external triggers.

Input-based Dynamic Workflows

Using environment variables

```
deployment = os.environ.get("DEPLOYMENT",
"PROD")

if deployment == "PROD":
    task = Operator(param="prod-param")
elif deployment == "DEV":
    task = Operator(param="dev-param")

// DAG code
```

Example of an **unsafe** approach:

keeping both the dev and prod data in the same location puts you at risk of overwriting prod data. This behavior will be blocked in the future.

```
prod_database = 'default'
dev_database = os.environ.get('user')

table_name = 'very_important_table'

output_dev_location = f's3://   bucket-name   /project/{dev_database}/{table_name}'
output_prod_location = 's3://   bucket-name   /project/{prod_database}/{table_name}'

...
```

Use Case: S3ResourceSpec

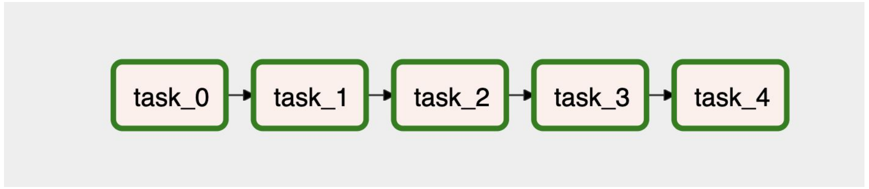
Input-based Dynamic Workflows at Pinterest

S3ResourceSpec

```
from airflow.lineage.pinterest.s3_resource
import S3ResourceSpec

# define the s3 url for the output table
output_s3_resource = S3ResourceSpec(
base_url='s3://bucket-name/retention-30days/
prefix_name/number_of_buckets',
test_resource_retention_days=30,
)

output_s3_resource_entity =
output_s3_resource.get_resource_entity()
```

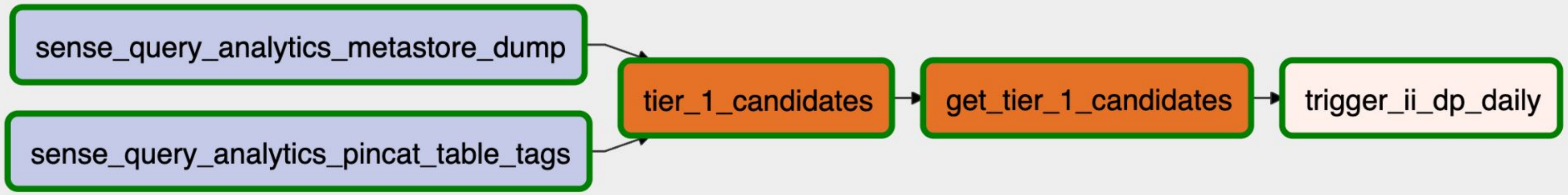
Condition-based Dynamic Workflows

Using iteration to generate tasks dynamically



```
from airflow import DAG
from airflow.operators.python_operator
import PythonOperator
from datetime import datetime

with DAG('sample_dag',
start_date=datetime(2024, 9, 10)) as dag:
    previous_task = None
    for i in range(5):
        task =
PythonOperator(task_id=f'task_{i}',
python_callable=lambda: print(f'Hello from
task {i}'))
        if previous_task:
            previous_task >> task
        previous_task = task
```

Use Case: Data Profiling


sense_query_analytics_metastore_dump
sense_query_analytics_pincat_table_tags
tier_1_candidates
get_tier_1_candidates
trigger_ii_dp_daily

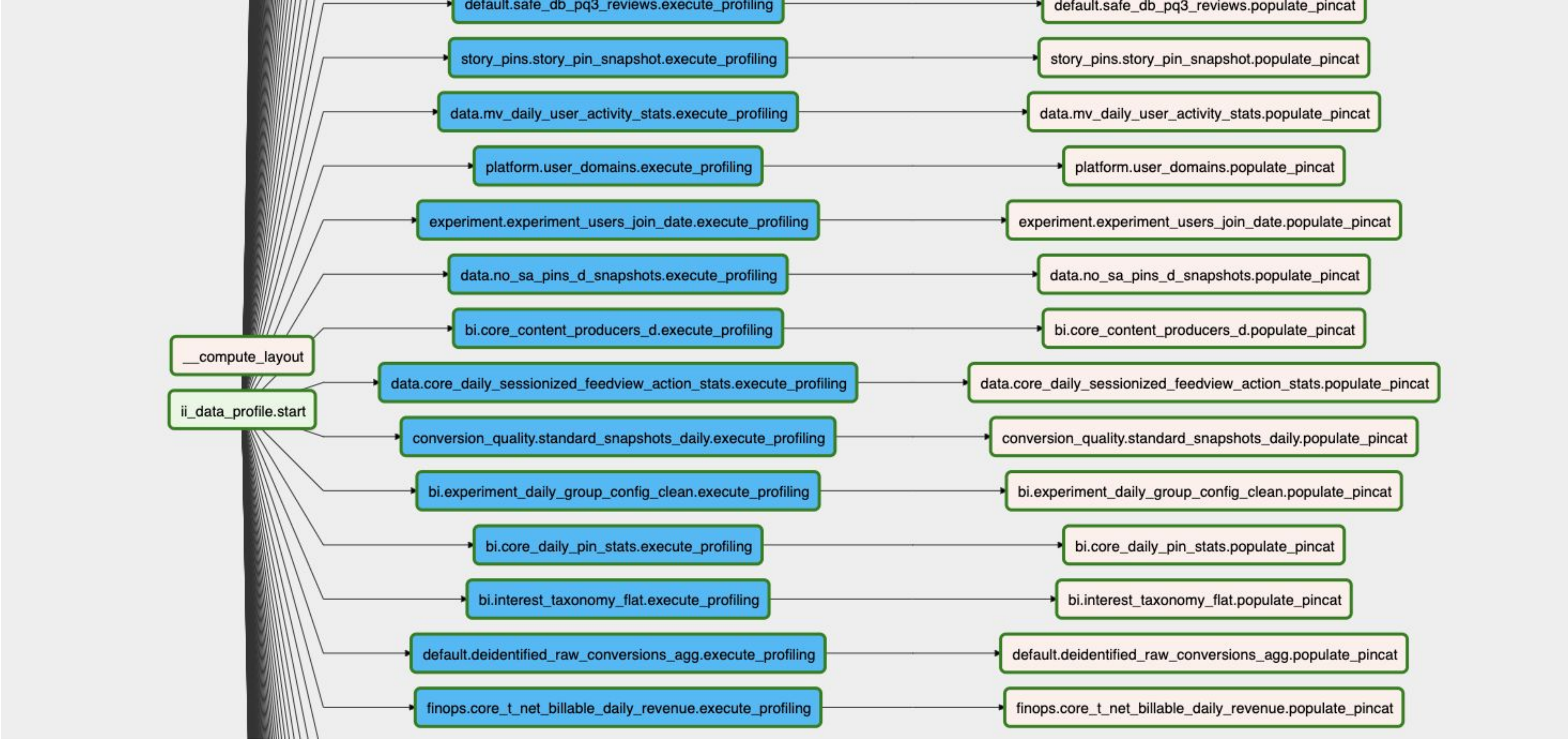
```
def _push_tables_to_xcoms(context):
    resolved = []
    for row in context['job_output']:
        dataset_name = row[0]

        if dataset_name not in resolved:
            resolved.append(dataset_name)

    context['task_instance'].xcom_push(key='datasets',
value=resolved)

context['task_instance'].xcom_push(key='wf_exec_date',
value=context['execution_date'].add(days=6)

.strftime('%Y-%m-%d'))
```

```
def _set_config(context, dag_run_obj):
    """Set the dag_run.config to pass downstream."""
    dag_run_obj.payload = {
        'datasets':
context['task_instance'].xcom_pull(task_ids='get_tier_1
_candidates', key='datasets'),
        'wf_exec_date':
context['task_instance'].xcom_pull(task_ids='get_tier_1
_candidates', key='wf_exec_date')
    }
    LOG.info(dag_run_obj)
    LOG.info(dag_run_obj.payload)
    return dag_run_obj
```

Triggered DAG: ii_dp_daily

```
from
airflow.providers.pinterest.models.dynamic_dag
import DynamicDAG


dag = DynamicDAG(
    dag_id='ii_data_profile',
    schedule_interval=None,
    default_args=default_args,
    concurrency=20,
    compute_layout=compute_layout,
    max_active_runs=3,
)
```

```
def compute_layout(dynamic_dag: DynamicDAG,
                   execution_date: datetime = None,
                   dagrun_conf: dict = None) -> None:

    dag_name = dynamic_dag.dag_id

    start = dynamic_dag.add_task_into_dynamic_dag(DummyOperator,
task_id='{}.start'.format(dag_name))

    data_profiling_task = {}
    populate_pincat_task = {}

    if isinstance(dagrun_conf, dict) and "datasets" in dagrun_conf and
isinstance(dagrun_conf["datasets"], list):

        exec_dt = dagrun_conf["wf_exec_date"]

        for dataset_name in dagrun_conf["datasets"]:

             start >> data_profiling_task[dataset_name] >>
populate_pincat_task[dataset_name]
```



default.safe_db_pq3_reviews.execute_profiling
default.safe_db_pq3_reviews.populate_pincat
story_pins.story_pin_snapshot.execute_profiling
story_pins.story_pin_snapshot.populate_pincat
data.mv_daily_user_activity_stats.execute_profiling
data.mv_daily_user_activity_stats.populate_pincat
platform.user_domains.execute_profiling
platform.user_domains.populate_pincat
experiment.experiment_users_join_date.execute_profiling
experiment.experiment_users_join_date.populate_pincat
data.no_sa_pins_d_snapshots.execute_profiling
data.no_sa_pins_d_snapshots.populate_pincat
bi.core_content_producers_d.execute_profiling
bi.core_content_producers_d.populate_pincat
__compute_layout
ii_data_profile.start
data.core_daily_sessionized_feedview_action_stats.execute_profiling
data.core_daily_sessionized_feedview_action_stats.populate_pincat
conversion_quality.standard_snapshots_daily.execute_profiling
conversion_quality.standard_snapshots_daily.populate_pincat
bi.experiment_daily_group_config_clean.execute_profiling
bi.experiment_daily_group_config_clean.populate_pincat
bi.core_daily_pin_stats.execute_profiling
bi.core_daily_pin_stats.populate_pincat
bi.interest_taxonomy_flat.execute_profiling
bi.interest_taxonomy_flat.populate_pincat
default.deidentified_raw_conversions_agg.execute_profiling
default.deidentified_raw_conversions_agg.populate_pincat
finops.core_t_net_billable_daily_revenue.execute_profiling
finops.core_t_net_billable_daily_revenue.populate_pincat

Advantages

Flexibility

Scalability

Efficiency

Best Practices

Best Practices

Optimize DAG parsing

Manage number of tasks and parallelism

Leverage XComs strategically

Conclusion

Conclusion

- Airflow's capabilities provide extensive control over workflow automation and scheduling.
- Dynamically generating workflows can enhance adaptability and operational efficiency.
- Following best practices is essential for optimizing performance and maintainability.

Questions?

Thank you!

Thejas Raju

Data Engineering at Pinterest

www.linkedin.com/in/thejas-raju

Appendix

- Spinner: Pinterest’s Workflow Platform
- Airflow at Pinterest: Airflow Summit 2022
- Apache Airflow: Quick Start