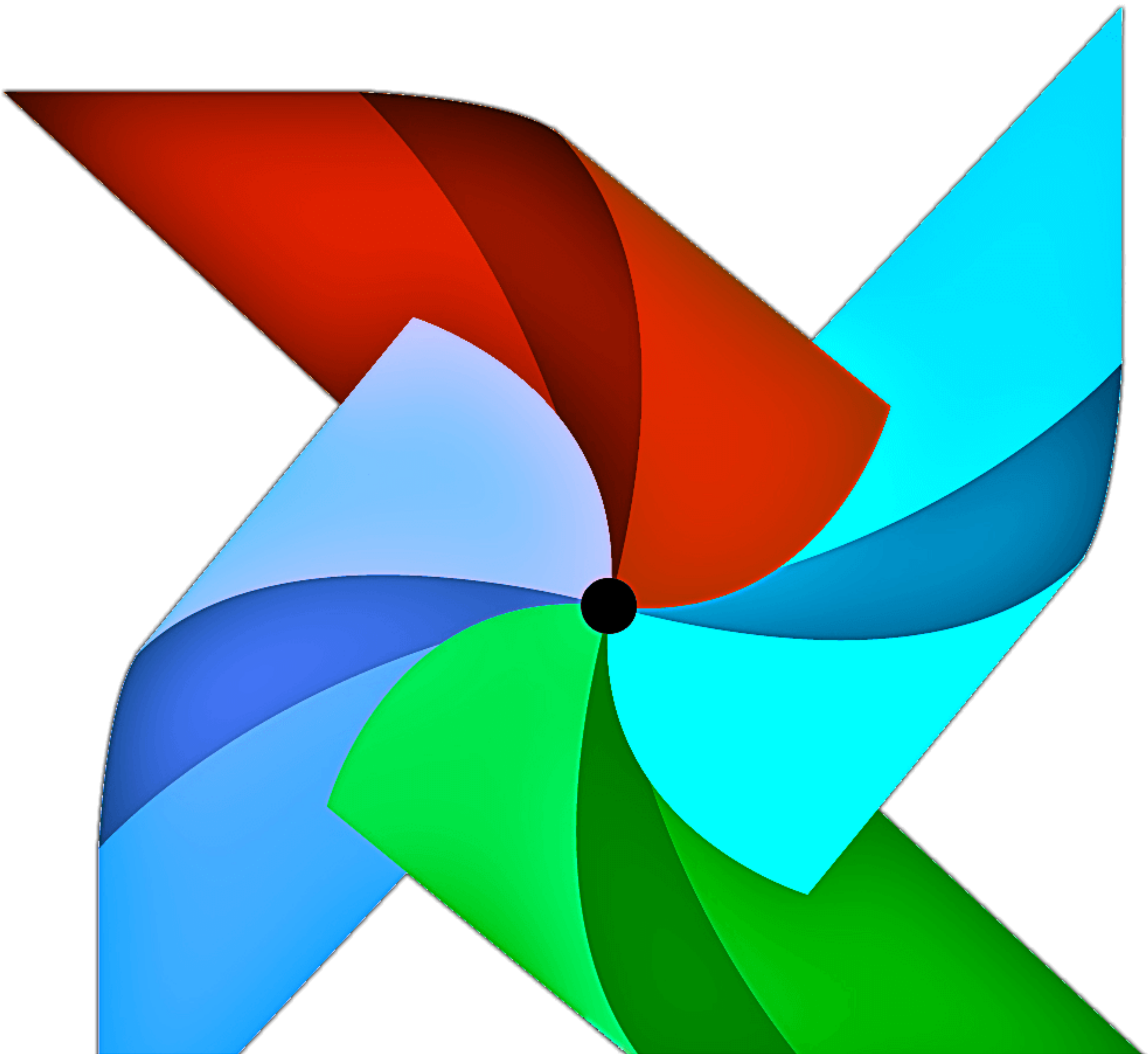
Airflow dynamic DAG graph flow changes at run time.

Craig Godden-Payne
Feb 7 · 2 min read ★



I have recently been working on as project which involved Airflow, and the project required a dynamic workflow.

The project went something along the lines of:

- Remove old files from S3 bucket
- Scan SFTP location to get a list of files
- For each file, upload file into S3
- Remove files from SFTP

One of the complexities with airflow, is that the DAG workflow is created at design time, so cannot be changed at runtime. Since I don’t know how many files I need to upload (I do know that it’s over 100 files, so serial upload is out of the question), I want this to happen in parallel, I needed to find a solution.

I managed to get around this limitation using variables. During setup, the dag does not have access to the execution context, so I was kind of left with the only option being variables. As part of the ‘Scan SFTP location to get a list of files’ task, I also set a variable containing the files, and as part of the DAG setup, I read this variable, creating a seperate task for each of the files that need to be uploaded.

There are a few cons using this method, if you skip the task ‘Scan SFTP location to get a list of files’ the list of files will be the previous iteration. There doesn’t seem to be a way to store information like this against the DAG run, this would be the ideal situation. Also the UI in airflow is a bit buggy using this technique, I noticed sometimes it would show before run and after run, which was weird, and also it means history for the files is not persisted in the Task or Graph view, although the logs are still available. It depends on how much you need this functionality.

Here is an example of how this was achieved:

```
start = DummyOperator(
    task_id='UploadFilesFromSFTPToS3',
    dag=subdag
)

start_upload_files = DummyOperator(
    task_id='StartUploadFiles'
)

end_upload_files = DummyOperator(
    task_id='EndUploadFiles'
)

config = Variable.get(dag_name, deserialize_json=True)
files_to_be_uploaded = config['files_to_be_uploaded']
upload_file_tasks = []
for file_name in files_to_be_uploaded:
    task = S3UploadOperator(
        task_id='SFTPMPUpload_{}'.format(file_name),
        sftp_conn=cfs_ssh_conn,
        default_args=default_args,
        dag=subdag
    )
    upload_file_tasks.append(task)

    # Here we assign the list of upload tasks dynamically (DAG
design time)
    if len(upload_file_tasks) > 0:
        start_upload_files >> upload_file_tasks >> end_upload_files
    else:
        start_upload_files >> end_upload_files

def find_files_function():
    try:
        files =
SFTPHook(ftp_conn_id=cfs_ssh_conn).list_directory(source_data_path)
    except Exception:
        files = []

    config = Variable.get(dag_name, deserialize_json=True)
    config['files_to_be_uploaded'] = files
    Variable.set(dag_name, json.dumps(config))

find_files = PythonOperator(
    task_id="FindFiles",
    python_callable=find_files_function,
    dag=subdag
)
start >> find_files >> start_upload_files >> end_upload_files
```

Written on February 6, 2020.

Originally published on: https://craig.goddenpayne.co.uk/airflow-dynamic-workflow/

Airflow Python Dag