Airflow in Practice: How I Learned to Stop Worrying and Love DAGs

Sarah Schattschneider
PyBay 2019

Who Am I?

- Software Engineer @ Blue Apron
  - Blue Apron - meal kit delivery service to make incredible home cooking accessible to everyone
  - Data Operations Team
- Airflow Interests:
  - Extract Transform Load (ETL) workflows
  - BigQuery Permissioning
  - Testing
- @sschatts

Outline

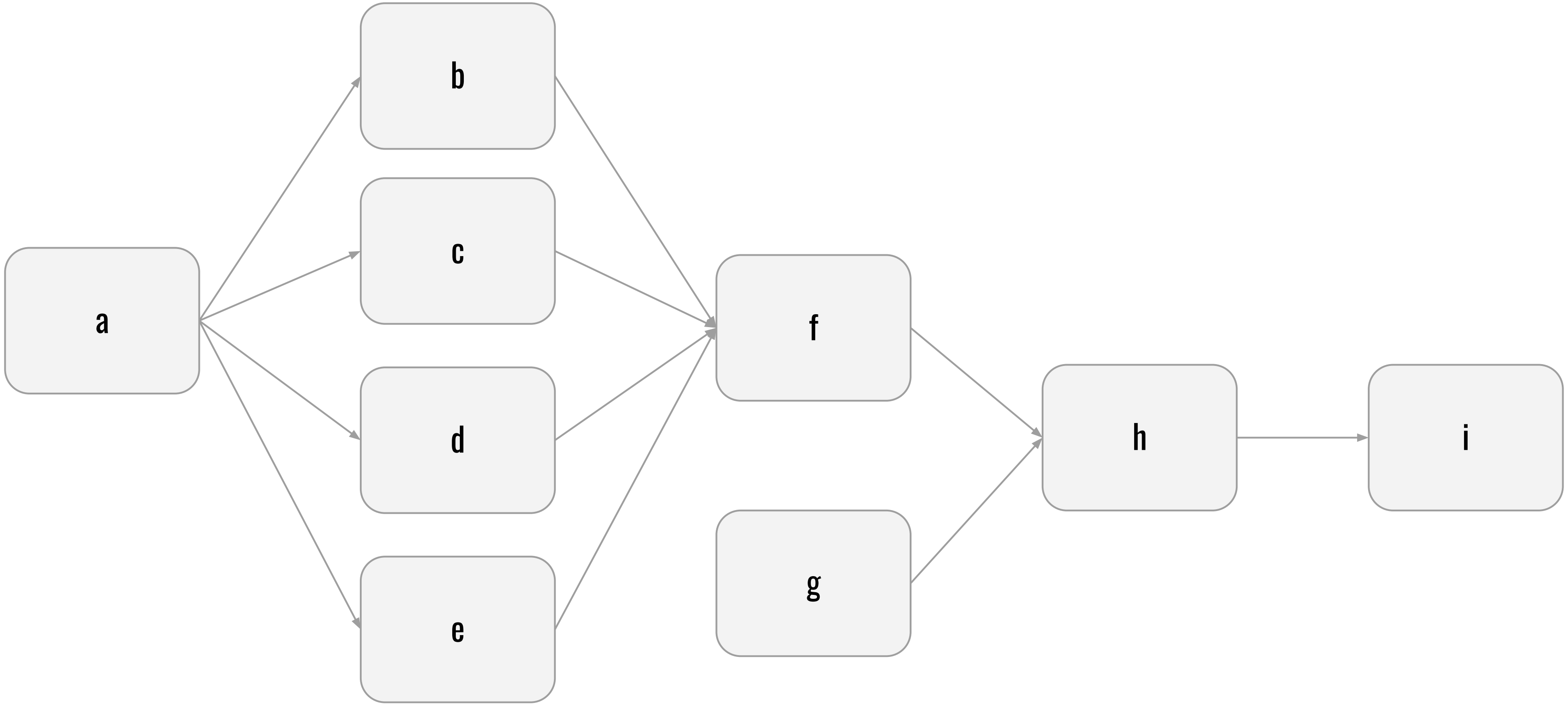
- Background
- Airflow Concepts
- Evaluating Airflow
- Challenges of Airflow
- Common Use Cases
- Best Practices

Background

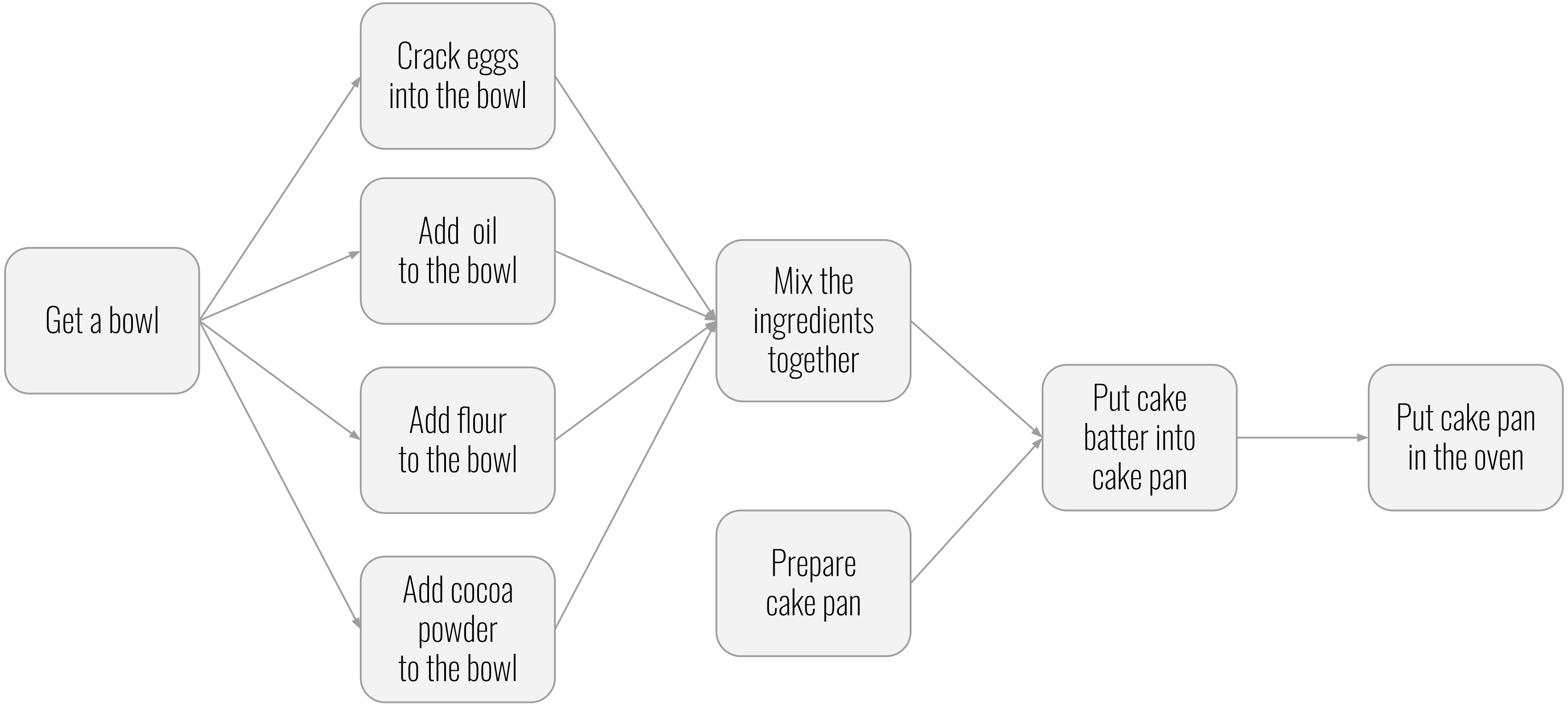
Once upon a time there was cron

```
0 1 * * * /bin/bash echo “hello world”
0 3,15 * * * /scripts/script.sh
0 2 * * mon /scripts/weekly.sh
```

Directed Acyclic Graph (DAG)



Directed Acyclic Graph (DAG)



Airflow Concepts

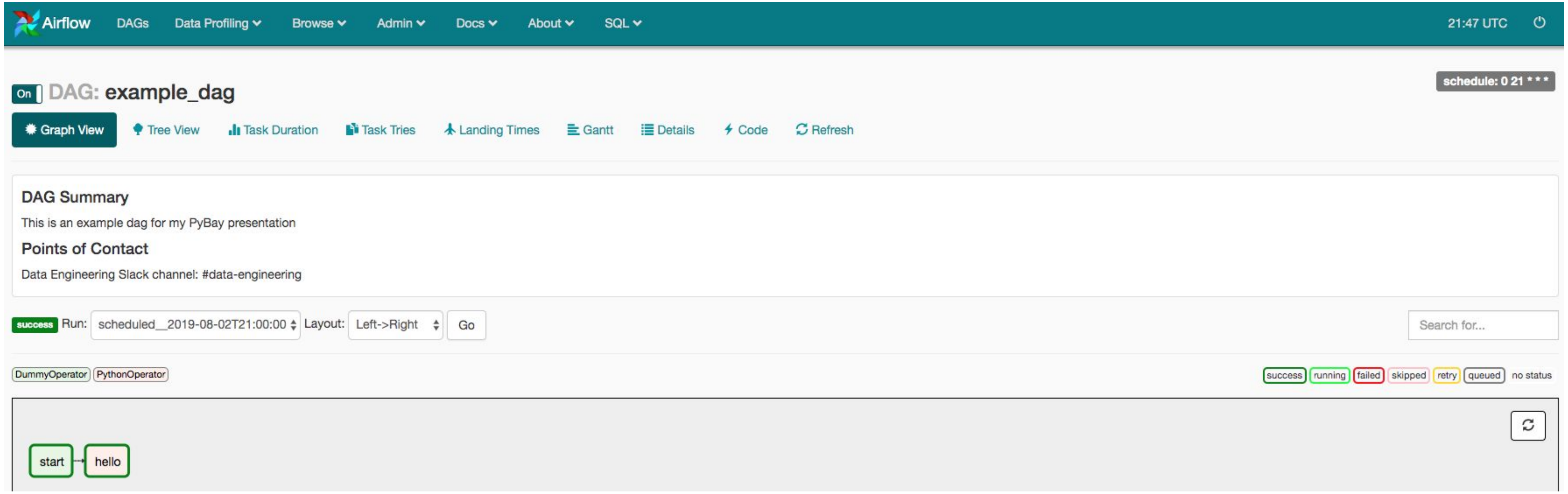
What is Airflow?

- Programmatically author workflows
- Stateful scheduling
- Rich Command Line Interface (CLI) and User Interface (UI)
- Logging, monitoring, and alerting
- Modularity and Testability
- Solves common problems in batch processing

Declare a DAG

```
def hello_world():
   return 'Hello World!'

default_args = {
  'owner': 'airflow',
  'depends_on_past': False,
  'start_date': datetime(2019, 8, 1),
  'email_on_failure': False,
  'retries': 2,
  'max_active_runs': 1
}

dag = DAG('example_dag', default_args=default_args, schedule_interval='0 21 * * *')
dag.doc_md = """
#### DAG Summary
This is an example dag for my PyBay presentation

#### Points of Contact
Data Engineering
Slack channel: #data-engineering
"""

start = DummyOperator(task_id='start', dag=dag)
start.doc_md = """ #start operator """

hello = PythonOperator(
   task_id='hello',
   python_callable=hello_world,
   dag=dag
)
hello.doc_md = """ #hello world operator """
start >> hello
```

User Interface



DAG: example_dag

schedule: 0 21 * * *

DAG Summary

This is an example dag for my PyBay presentation

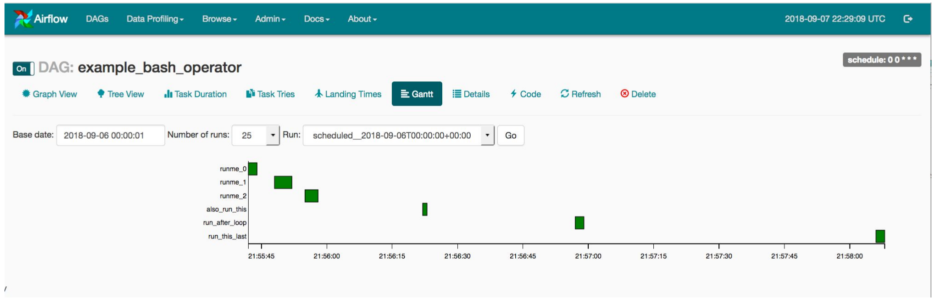
Points of Contact

Data Engineering Slack channel: #data-engineering

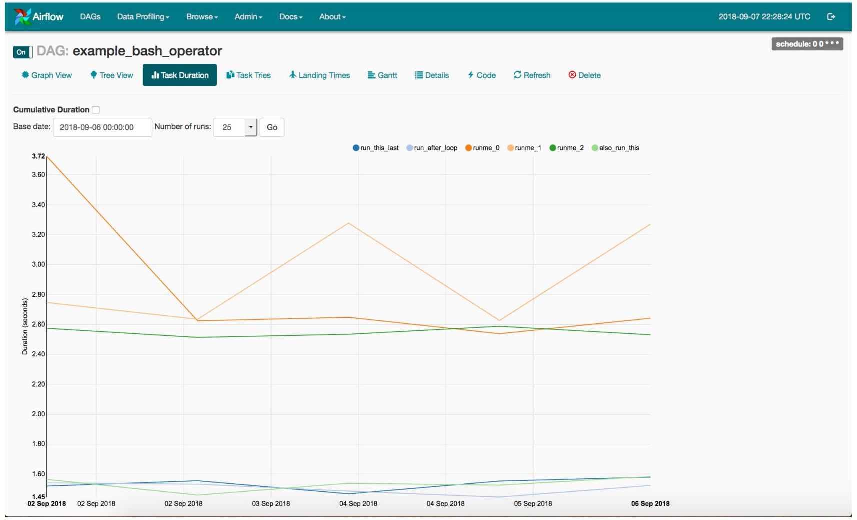
success Run: scheduled__2019-08-02T21:00:00 Layout: Left->Right

DummyOperator PythonOperator

Metrics



Base date: 2018-09-06 00:00:01 Number of runs: 25 Run: scheduled__2018-09-06T00:00:00+00:00



Base date: 2018-09-06 00:00:00 Number of runs: 25

Airflow README

Hooks

Encapsulate all interactions with an API

Operators

Instantiate a hook or two to complete a single goal

Tasks

An instance of an operator

Hooks

Cake Pan

Oven

Operators

Put Cake Pan in Oven

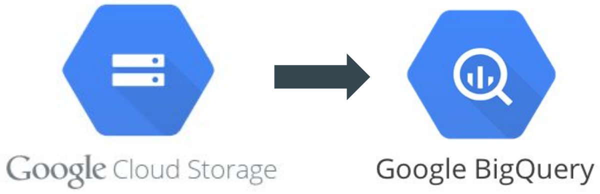
Tasks

```
CakePanToOvenOperator(
  task_id='cake_to_oven',
  cake_ingredients={
      'eggs',
      'oil',
      'cocoa powder',
      'flour'
  },
  oven_conn_id=OVEN_CONN_ID,
  cake _pan_conn_id=CAKE_PAN_CONN_ID,
  dag=dag
)
```

Put cake pan in the oven

Hooks

Operators



Tasks

```
gcs_to_bq =
GoogleCloudStorageToBigQueryOperator(
  task_id='gcs_to_bq',
  bucket='my_bucket’,
  source_objects=['path/to/my_file.csv'],
  destination_project_dataset_table='dataset.table',
  schema=transfer_schema,
  write_disposition='WRITE_TRUNCATE',
  skip_leading_rows=1,
  google_cloud_storage_conn_id=GCS_CONN_ID,
  bigquery_conn_id=BQ_CONN_ID,
  dag=dag
)
```

Airflow Hooks

- AwsHook
- JdbcHook
- GoogleCloudStorageHook
- CassandraHook
- BigQueryHook
- RedisHook
- DatadogHook
- AzureFileShareHook
- GrpcHook
- JiraHook
- SSHHook
- HiveCliHook
- SlackHook
- DockerHook
- S3Hook
- MySqlHook
- PrestoHook

and many more...

Airflow Operators

- PythonOperator
- BashOperator
- EmailOperator
- S3ToRedshiftTransfer
- RedshiftToS3Transfer
- BigQueryToCloudStorageOperator
- GoogleCloudStorageToBigQueryOperator
- SlackAPIPostOperator
- MySqlToHiveTransfer
- SimpleHttpOperator
- DataprocOperationBaseOperator
- JiraOperator
- DockerOperator
- PostgresOperator

and many more...

Evaluating Airflow

What value does Airflow add?

- Retries tasks elegantly
- Alerts on failure
- Re-run specific tasks in the DAG
- Supports distributed execution

Third Party Data Integrations

and many more...

Cron Scheduling

Monday at 4:00	0	4	*	*	1
	minute	hour	day (month)	month	day (week)
10th day of the month at 10:05	5	10	3	*	*
	minute	hour	day (month)	month	day (week)

Cron Scheduling

- Time zone safety
- Flexibility of determining when to run

Does Airflow Have an Ugly Side?
How to Overcome these Challenges?

Upgrades

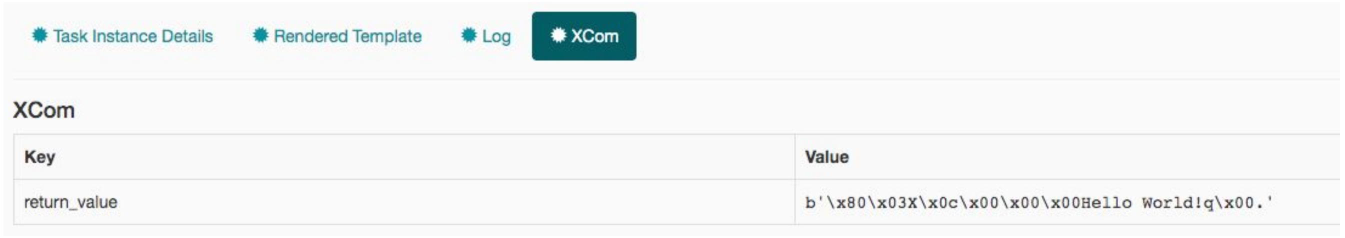
- Upgrades can be more challenging when you have custom hooks and operators

Help with Upgrade Challenges:

- Having DAGs run in staging environment
- Better unit tests
- Contribute to upstream regularly

Data Sharing

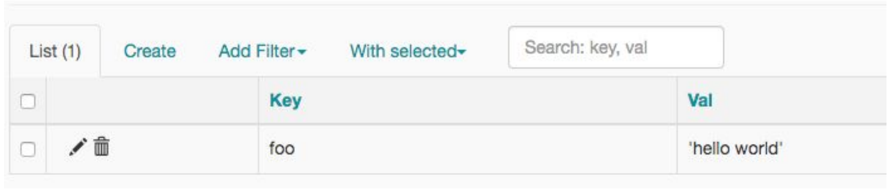
- Xcoms - Airflow cross-communication between tasks



XCom

Key	Value
return_value	`b'\x80\x03X\x0c\x00\x00\x00Hello World!q\x00.'`

- Airflow Variable



		Key	Val
☐		foo	'hello world'

- Python Variable
- Environment Variable

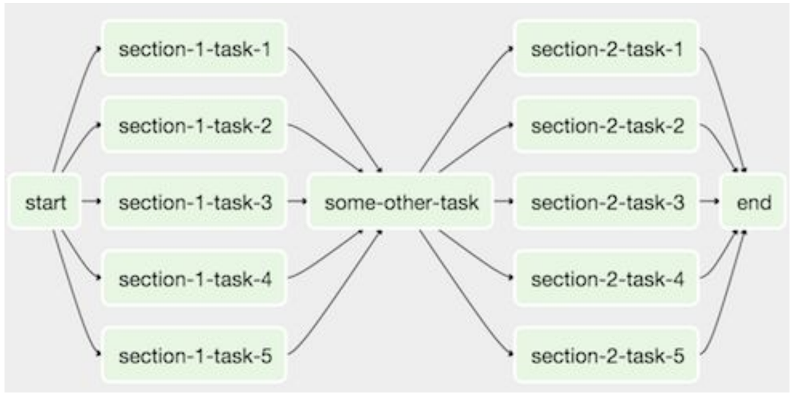
Common and Fun Use Cases

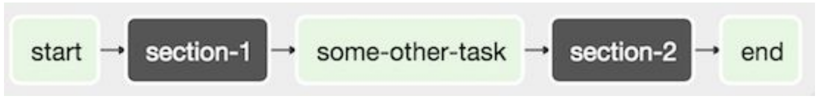
Extract Transform Load (ETL) Jobs

- Airflow enables moving data and transforming data very easily
- Can create custom Hooks and Operators for Third Party APIs

Subdags? How do you use them?

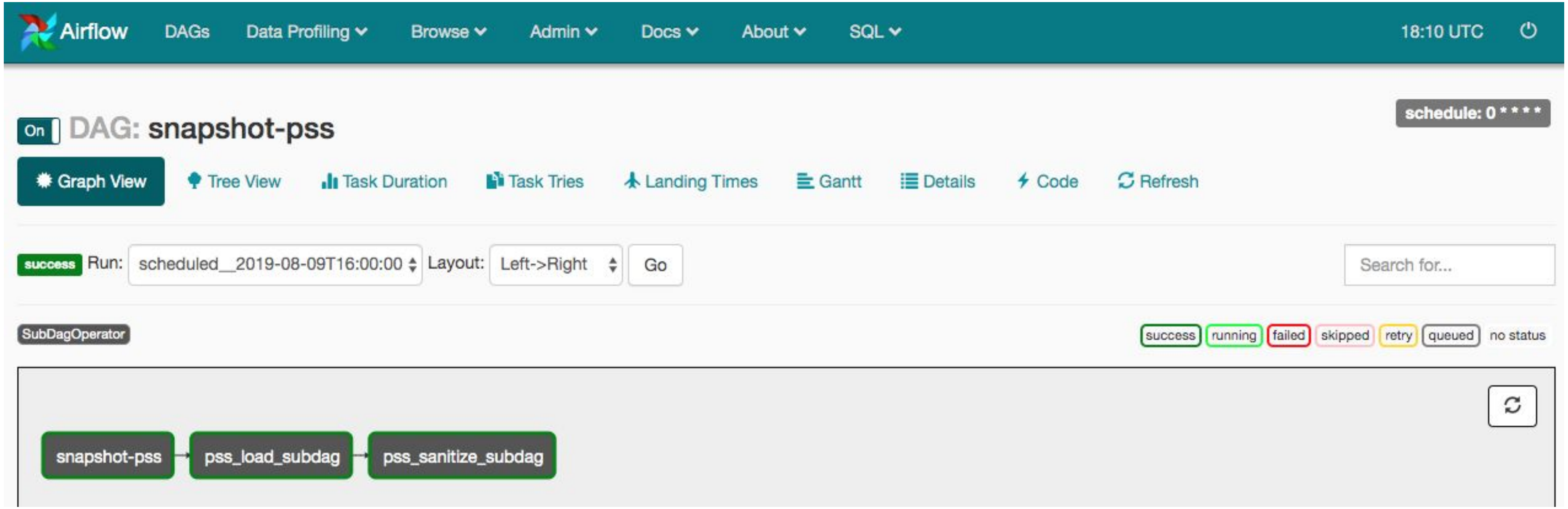
- Used when there are repeating patterns





Airflow Documentation

Subdags Example



DAG: snapshot-pss

schedule: 0 * * * *

success Run: scheduled__2019-08-09T16:00:00 Layout: Left->Right

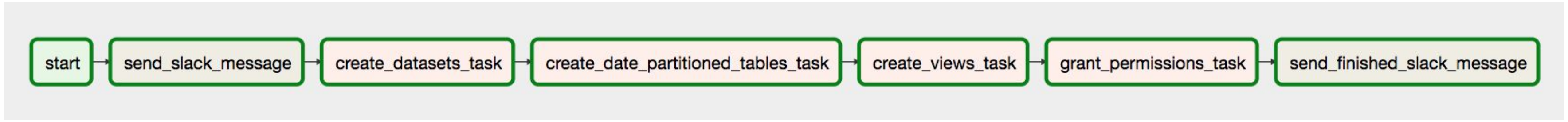
snapshot-pss → pss_load_subdag → pss_sanitize_subdag

Slack Integrations

- Notify different Slack channels

```
slack_notification= SlackAPIPostOperator(
   task_id='slack_notification',
   username='Airflow Alerting User',
   slack_conn_id=SLACK_CONN_ID,
   channel='#airflow-alert-channel',
   text='Hello World',
   dag=dag
)
```

BigQuery Permissioning



ML Feature Extraction

- Export data from our data warehouse
- Use the export data in Spark to train model

Best Practices

Testing

- Unit tests
  - Supporting Python functions
  - Custom hooks and operators
- Acceptance test to run `airflow list_dags`

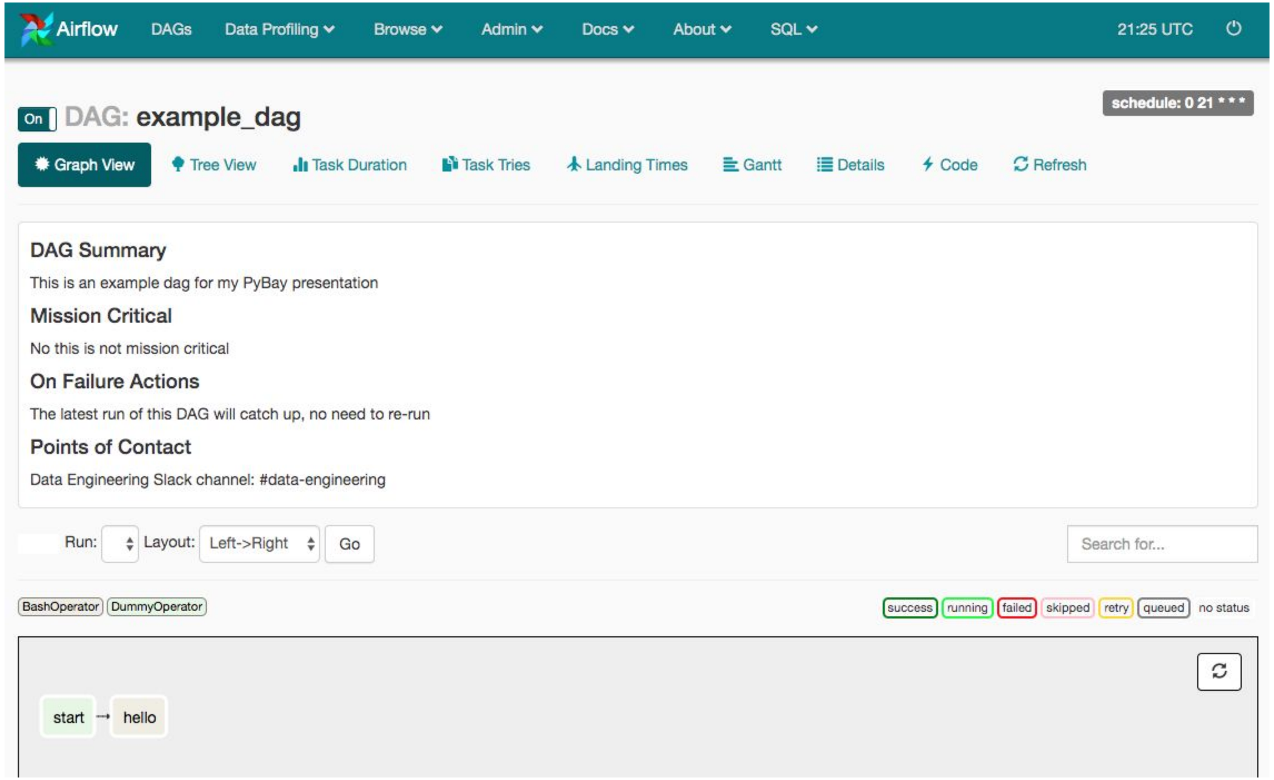
Doc MD for the DAG

```
#### DAG Summary
SHORT 2-3 sentence paragraph of project description goes here.

#### Mission Critical
Yes/No and a short blurb for context

#### On Failure Actions
Rerun DAG? Yes/No and a short blurb for context

#### Points of Contact
Multiple points of contact are preferable with area of responsibility and title if possible. If updating,
make sure to include the date

#### Business Use Case/Process
   https://www.lucidchart.com/documents/edit/12345

#### Prerequisites/Dependencies/Resourcing
    -    Some cool documentation
```

In the Codebase

```
dag = DAG('example_dag', default_args=default_args, schedule_interval='0 21 * * *')
dag.doc_md = """
##### DAG Summary
SHORT 2-3 sentence paragraph of project description goes here.

##### Mission Critical
Yes/No and a short blurb for context

##### On Failure Actions
Rerun DAG? Yes/No and a short blurb for context

##### Points of Contact
Multiple points of contact are preferable with area of responsibility and title if possible. If updating, make sure to include
the date

##### Business Use Case/Process
   https://www.lucidchart.com/documents/edit/12345

##### Prerequisites/Dependencies/Resourcing
    -     Some cool documentation
"""
```

Materialized in the UI



DAG: example_dag

DAG Summary

This is an example dag for my PyBay presentation

Mission Critical

No this is not mission critical

On Failure Actions

The latest run of this DAG will catch up, no need to re-run

Points of Contact

Data Engineering Slack channel: #data-engineering

Thanks!

...

@sschatts
sarah@blueapron.com

Software Engineer - Data Operations
Blue Apron