How we use Airflow at Booking to orchestrate Big Data workflows

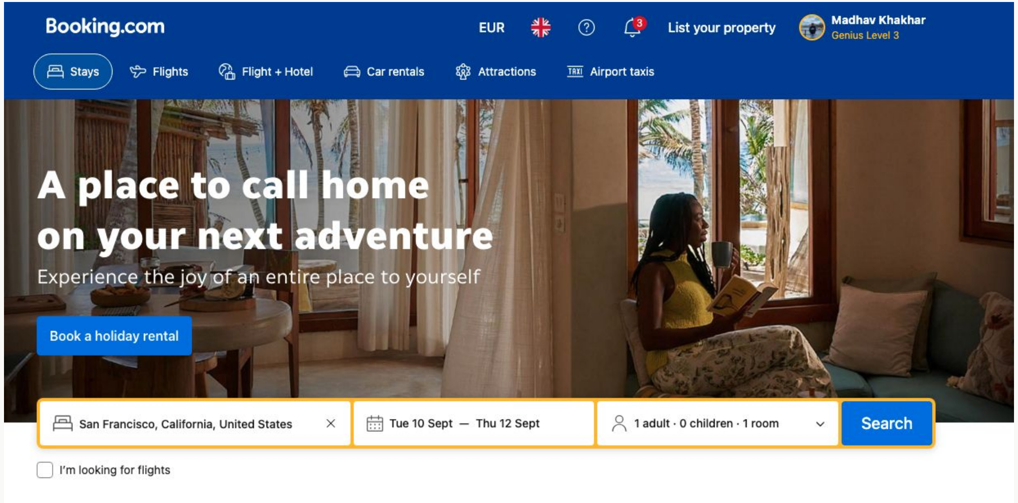
Madhav Khakhar - Senior Software Engineer, Booking.com
Alexander Shmidt - Senior Software Engineer, Booking.com
Mayank Chopra - Senior Technical Product Manager, Booking.com

Agenda

- Booking.com Introduction
- Migration & Modernization
- Workflow Management Platform
- Shifting to Astronomer
- Q&A

Booking.com

- Largest online travel company in the world
- Originally only offered accommodation bookings, currently offering a wide range of travel related services (connected trip).
- To accommodate this, we are in the middle of a data modernization program.



Booking.com

Premiere Online Travel Retailer

100M+

- 100M+ monthly active users
- 24/7 operations

>1500

- 150+ Data Engineers
- 350+ Data Scientists & ML Engineers
- 1000+ Analysts

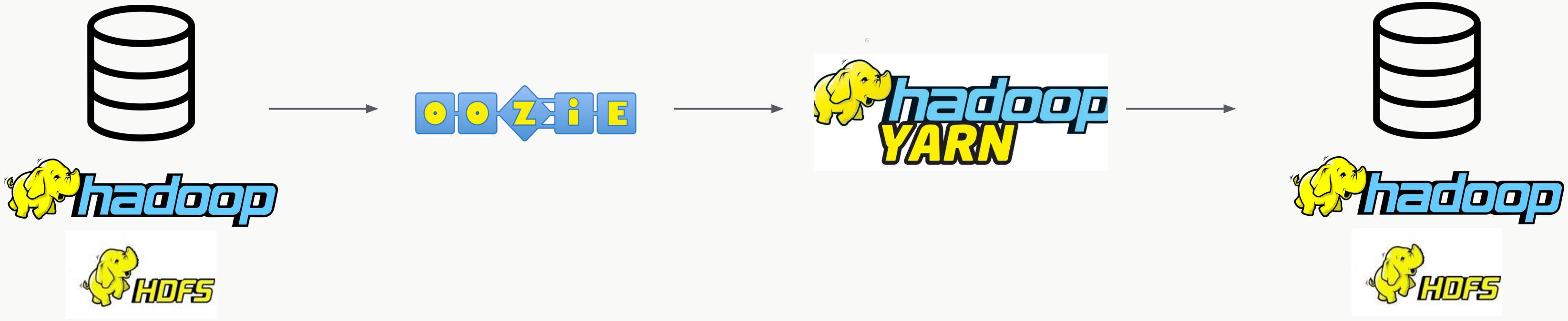
XXX PB

- ~100 TB ML inference events per day
- Many PT of data
- Very LARGE on-prem Hadoop

WORKFLOW MIGRATION

Workflows - old stack

Hadoop, Oozie, Apache Spark on Kubernetes



Ingestion

storage

DAM

data Asset Mngmt

permissions

data warehouse

Data Catalog

orchestrator

Data Query

DAS & DVS

Data Availability & Quality

compute

SDK

Read/write library

OpenLineage

Data

Ingestion

storage

DAM

data Asset Mngmt

permissions

data warehouse

Data Catalog

Orchestration & ETL

orchestrator

Data Query

DAS & DVS

Data Availability & Quality

compute

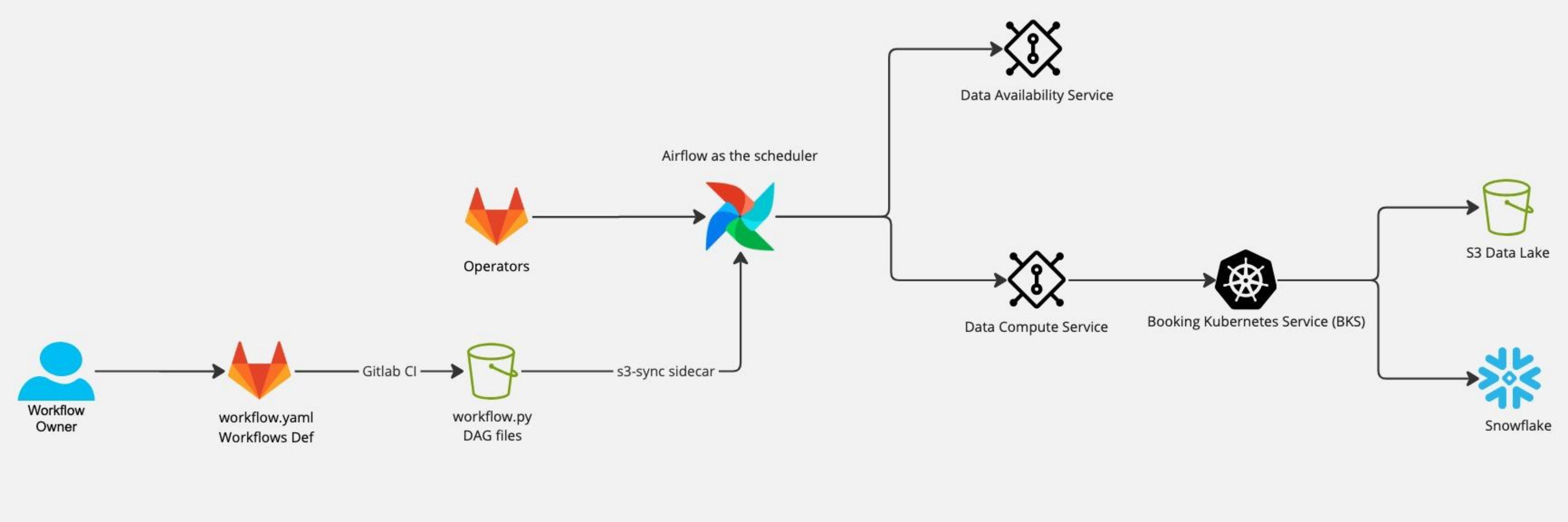
SDK

Read/write library

OpenLineage

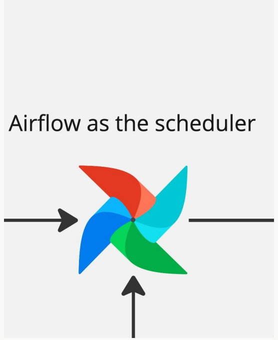
Workflow Management Platform
(Orchestrator Platform)

WFM Platform Design



How we setup the Airflow installation

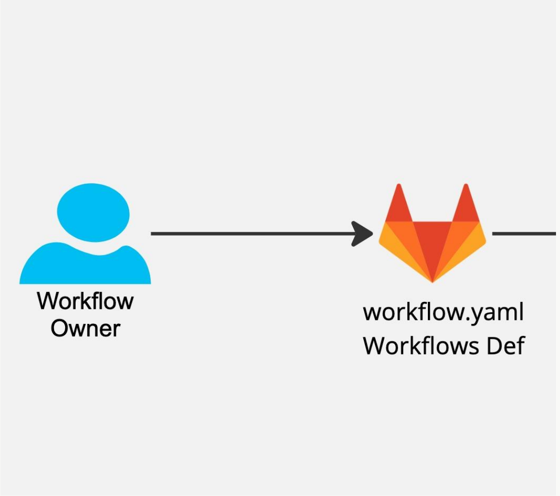
- Used Airflow community helm chart as the base
- Adapted it to deploy on Booking kubernetes service (BKS)
    - Booking Sidecars to support service discovery, S2S authentication and authorization
    - s3-sync sidecar to sync workflow.py DAG files from S3 to Airflow
    - Fluentbit sidecar to ship *triggerer* logs to Opensearch (Kibana)



Workflow definition (first glimpse)

{..} workflow.yaml

```yaml
interval: daily

namespace: traintomigrate

region:
  - bk-eks

dataAssets:
  uses:
    - name: "sample_data_asset"
      version: "1.0"
      materialization: hive.bdx.sample_data_asset.sample_test_data_v1

steps:
  - name: aggregate
    template: pyspark
    config:
      mainPythonFileUri: aggregate.py
      packages:
        pip:
          - bkng-bdx[spark]
          - pendulum
      args:
        - "--nominal_date"
        - "{{ data_interval_start }}"
        - "--filter_date"
        - "{{ macros.ds_add(ds, -1) }}"

    dataAssets:
      publishes:
        - name: "traintomigrate.sparktraining0829.ant"
          version: "1.0"
          materialization: hive.bdx.traintomigrate.sparktraining0829_ant_v1
          production_mode: full_refresh
          period: DAY

```



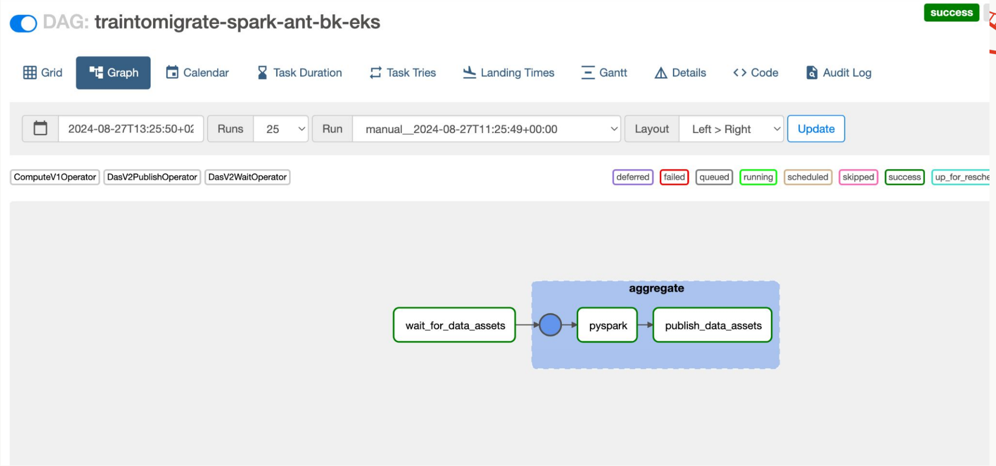
Workflow definition (Why?)

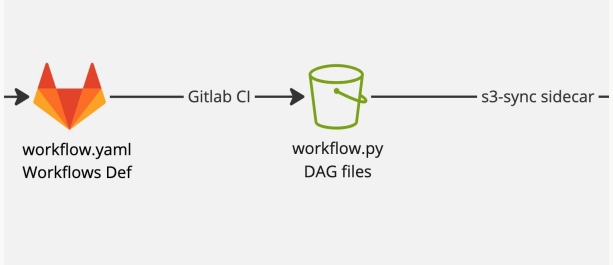
- Abstraction for users
  - Writing a workflow.yaml instead of a python DAG
  - Not having to worry about internals of how a computation job runs
- Standardized templates
  - Platform team owns the templates
- “Pluggable” Airflow Backend
- Ease of enabling governance

Workflow.py on Airflow

{..} workflow.yaml

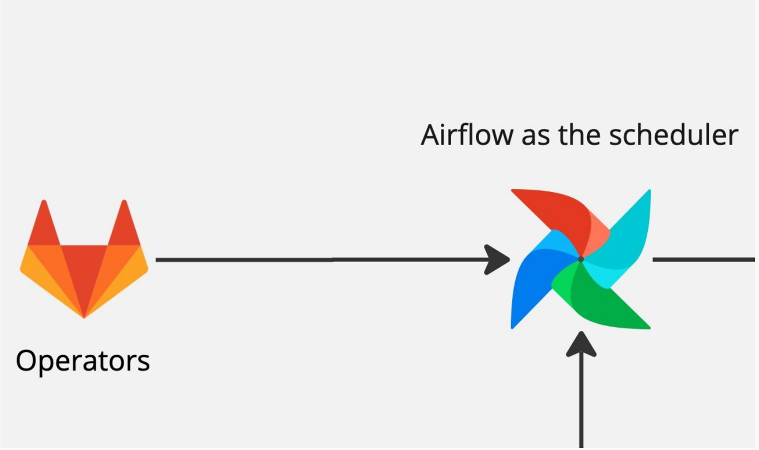
```yaml
interval: daily

namespace: traintomigrate

region:
  - bk-eks

dataAssets:
  uses:
    - name: "sample_data_asset"
      version: "1.0"
      materialization: hive.bdx.sample_data_asset.sample_test_data_v1

steps:
  - name: aggregate
    template: pyspark
    config:
      mainPythonFileUri: aggregate.py
      packages:
        pip:
          - bkng-bdx[spark]
          - pendulum
      args:
        - "--nominal_date"
        - "{{ data_interval_start }}"
        - "--filter_date"
        - "{{ macros.ds_add(ds, -1) }}"

    dataAssets:
      publishes:
        - name: "traintomigrate.sparktraining0829.ant"
          version: "1.0"
          materialization: hive.bdx.traintomigrate.sparktraining0829_ant_v1
          production_mode: full_refresh
          period: DAY

```





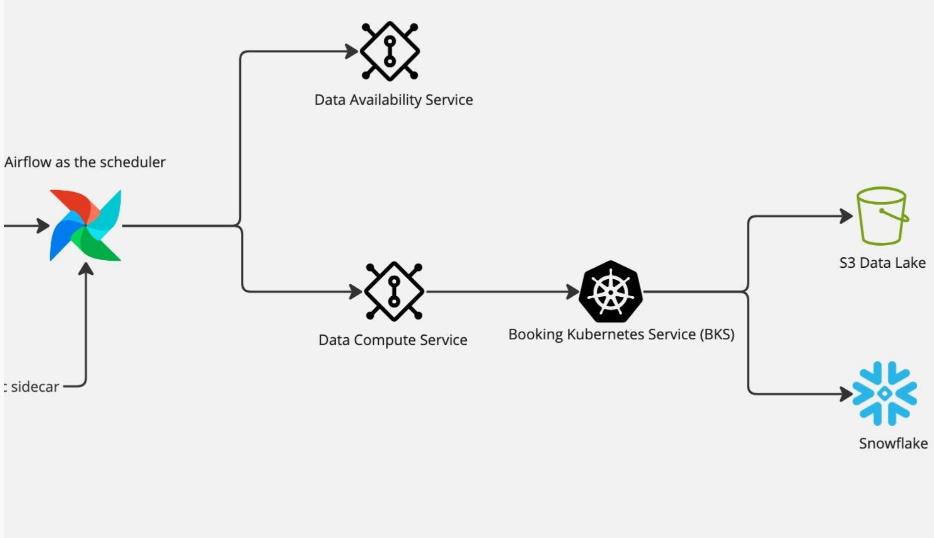
Workflow Steps - Deferrable Operators

- All step templates are Deferrable Operators
  - Typically make API calls to do a POST
  - And then waiting for completion (GET calls)
- Helps us scale better (lightweight workers)
  - Actual polling happens inside the triggerers



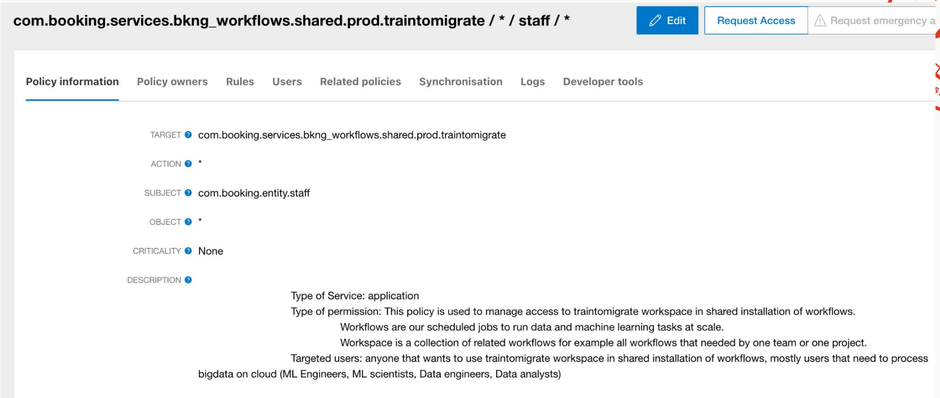
Airflow as pure orchestrator

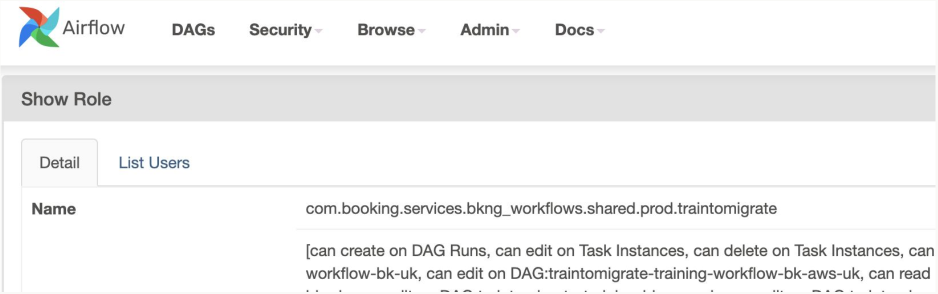
- Integrations
  - Data Availability
  - Data Compute Service
- Actual computation runs on Spark on kubernetes / snowflake (dbt)

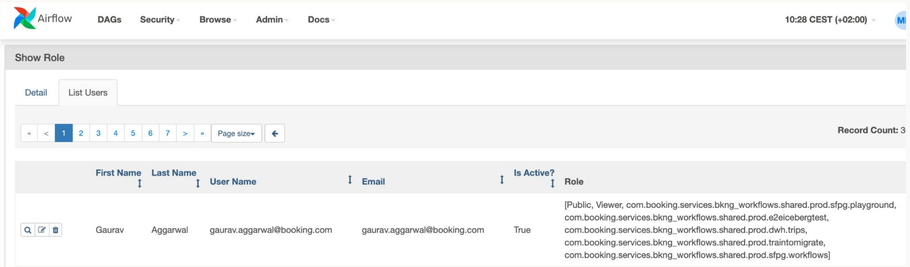


Workflow Access

- One access policy for a collection of workflows
- Users login via Okta, get access to specific workflow DAGs







Workflow Alerting

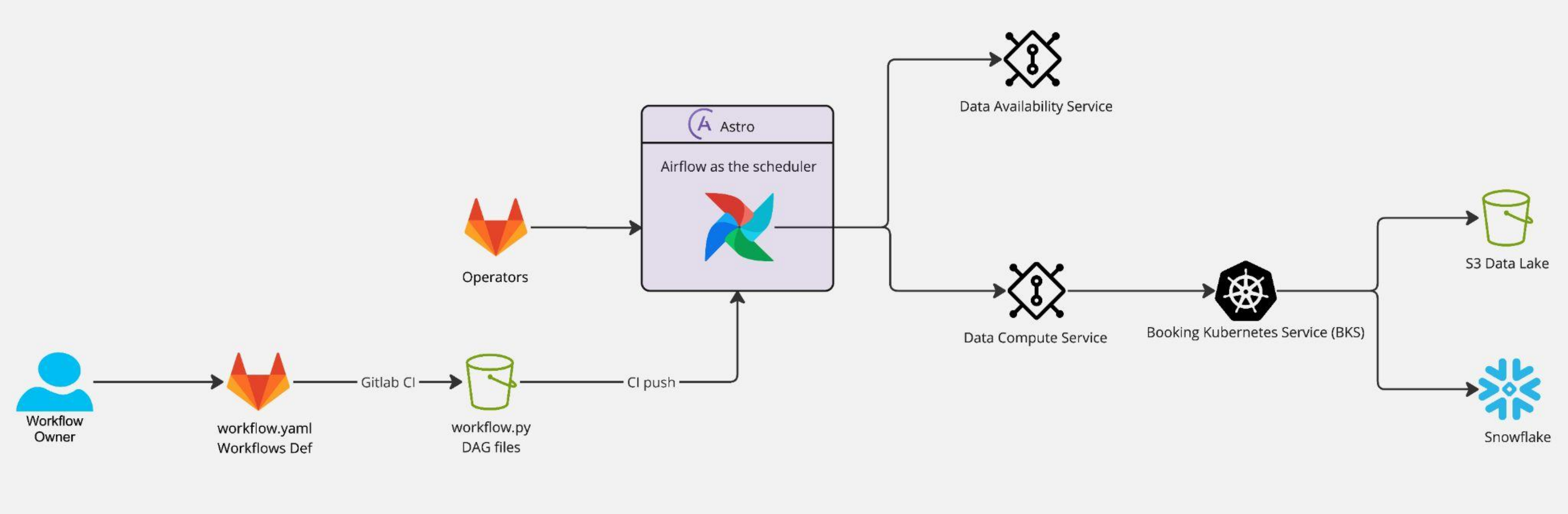
- Integration to AlertAPI (Booking internal tool)
- Boilerplate code to send failure alerts
- Users can subscribe to alerts

```
def send_failure_alert(context):
    dag_id = context['dag'].dag_id
    task_id = context['task'].task_id
    execution_date = context['ts']
    exception = context.get('exception', context.get('reason', ''))
    text = f"Execution failed for workflows: {dag_id} Step: {task_id} Exception: {exception}"
    send_alert(dag_id, task_id, execution_date,text,exception )

def send_alert(dag_id, task_id, execution_date,text,exception=None):
    workflow = dag_id
    logs_url = f"/log?dag_id={dag_id}&task_id={task_id}&execution_date={urllib.parse.quote(execution_date)}"
    requests.post("https://alertapi.booking.com/api/message", json={
        "name": f"persona.b_bkng-workflows_airflow.workflows.{workflow}.{execution_date}",
        "msg_type": 1,  # MSG_TYPE_IN_ALERT
        "msg_text": f"{text} Airflow logs: {logs_url}",
        "refdata" : {
            "workflow": workflow,
            "step" : task_id,
            "execution_date" : execution_date,
            "exception" : f"{exception}",
            "logs_url": logs_url
        }
    })
```

```
success_callback_functions = []
failure_callback_functions = [send_failure_alert]

def generate_dag_callback(callback_functions):
    def execute_callback_functions(context):
        for function in callback_functions:
            function(context)
    return execute_callback_functions

default_args = dict(
    depends_on_past=False,
    retries=1,
    retry_delay=timedelta(minutes=5),
    provide_context=True,
    execution_timeout=timedelta(hours=24, minutes=random.randint(12,20)),
    retry_exponential_backoff=False,
    on_success_callback=generate_dag_callback(success_callback_functions),
    on_failure_callback=generate_dag_callback(failure_callback_functions),
    sla=None,
)
```

Shifting to Astronomer

WFM Platform - Astronomer-powered



Shifting to Astronomer-managed Airflow

Why?

What changes?

Learnings

Shifting to Astronomer: Why?

Internal adoption of Airflow rapidly grows

Takes care of reliability (uptime, on-call)

Easy upgrade to newer Airflow versions

Easy [auto]scaling

Support engineers

Shifting to Astronomer: What changes?

Network integration for accessing internal systems

Service-to-service authentication

DAG deployment flow

User access

Infra-as-code

Shifting to Astronomer: What changes?

Network integration for accessing internal systems

Service-to-service authentication

DAG deployment flow

User access

Infra-as-code

Shifting to Astronomer: What changes?

Network integration for accessing internal systems

Service-to-service authentication

DAG deployment flow

User access

Infra-as-code

Shifting to Astronomer: What changes?

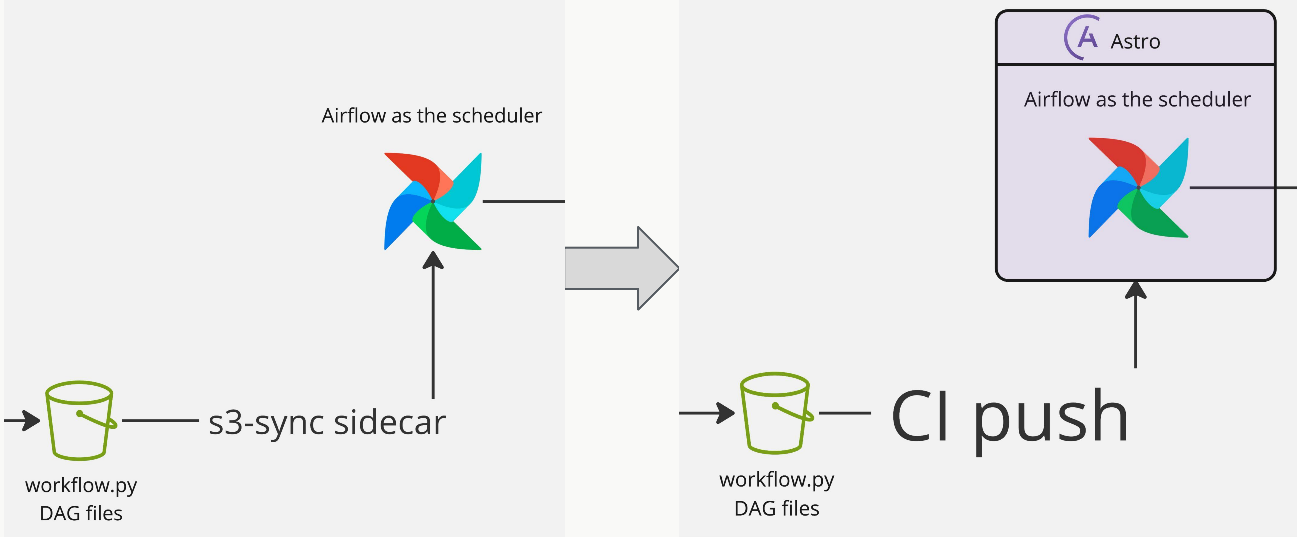
Network integration for accessing internal systems

Service-to-service authentication

DAG deployment flow

User access

Infra-as-code



Shifting to Astronomer: What changes?

Network integration for accessing internal systems

Service-to-service authentication

DAG deployment flow

User access

Infra-as-code

Shifting to Astronomer: What changes?

Network integration for accessing internal systems

Service-to-service authentication

DAG deployment flow

User access

Infra-as-code

Shifting to Astronomer: Learnings

Doing a proof-of-concept of integrating a vendor into your infrastructure helps uncover a lot of small issues you often don't think about after using internally-streamlined deployment and communication processes.

Do cost analysis of different architectural decisions for your use case, as yours might be different from a "typical" one

Questions?

LinkedIn - Madhav Khakhar
LinkedIn - Alexander Shmidt
LinkedIn - Mayank Chopra