

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



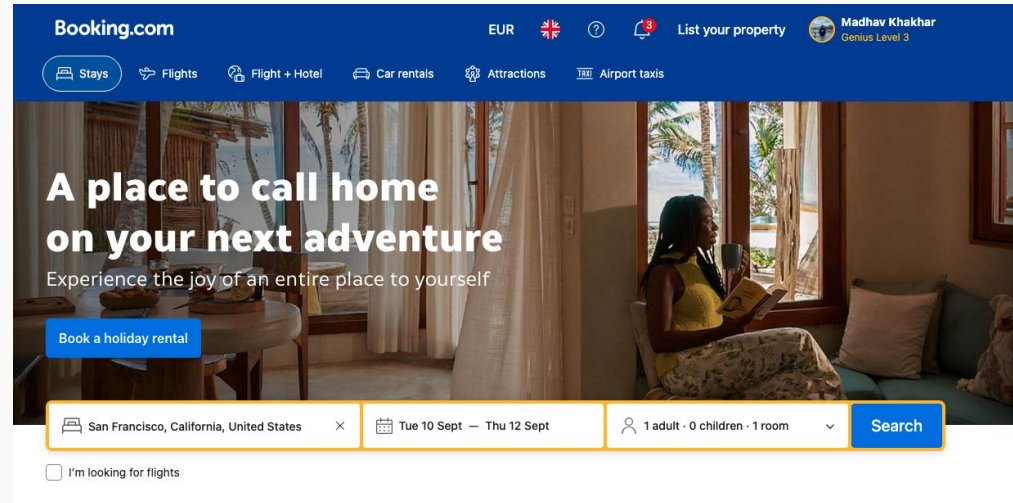
Agenda

- 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

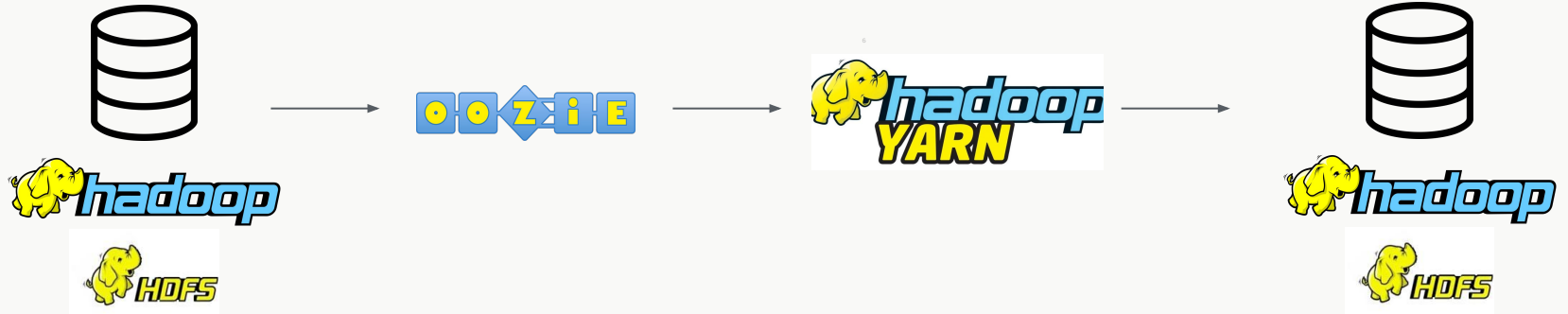
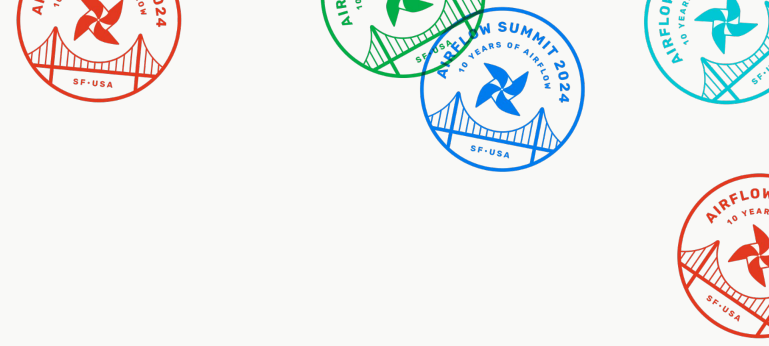
- ~100 TB ML inference events per day
- Many PT of data
- Very LARGE on-prem Hadoop



WORKFLOW MIGRATION

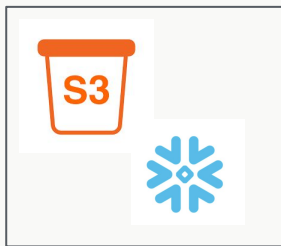
Workflows - old stack

Hadoop, Oozie, Apache Spark on Kubernetes

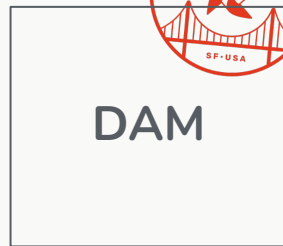




Ingestion



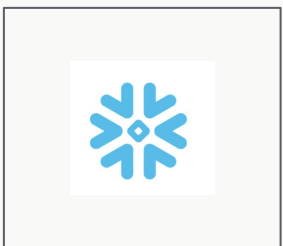
storage



data Asset Mngmt



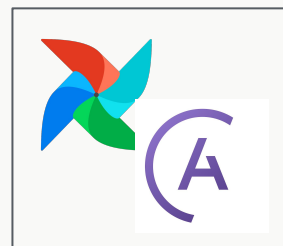
permissions



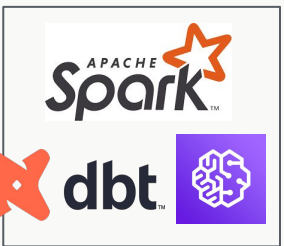
data warehouse



Data Catalog



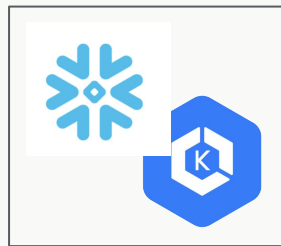
orchestrator



Data Query



Data Availability &
Quality



compute



Read/write library

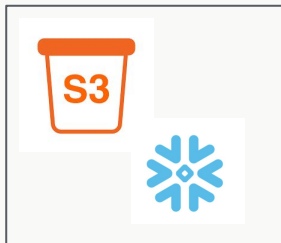


OpenLineage

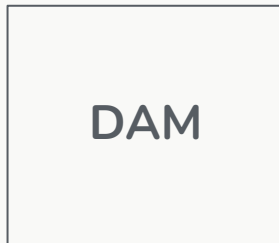
Data



Ingestion



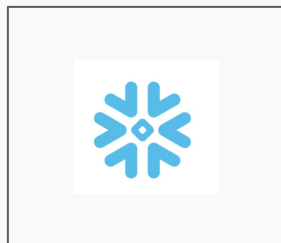
storage



data Asset Mngmt



permissions



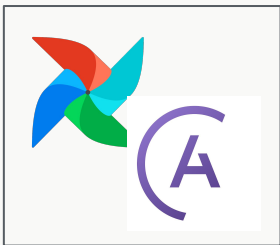
data warehouse



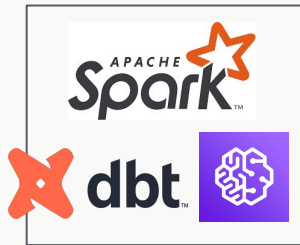
Data Catalog



Orchestration & ETL



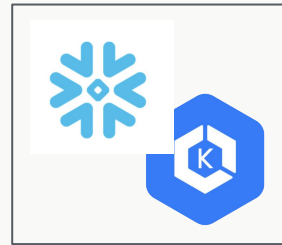
orchestrator



Data Query



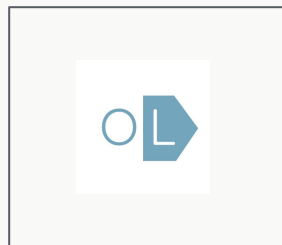
Data Availability &
Quality



compute



Read/write library

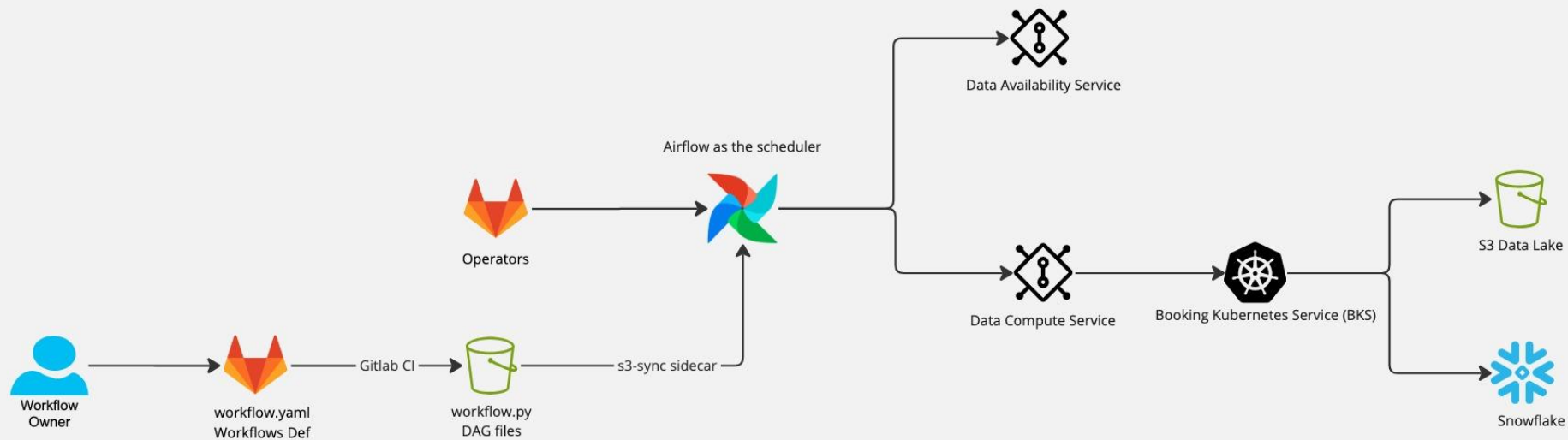


OpenLineage



Workflow Management Platform (Orchestrator Platform)

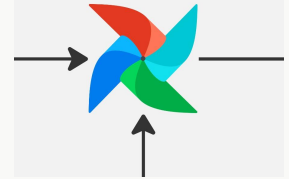
WFM Platform Design



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)

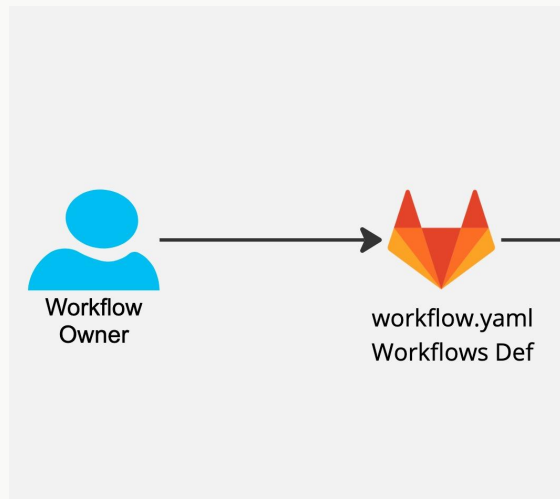
Airflow as the scheduler



Workflow definition (first glimpse)

workflow.yaml

```
1 interval: daily
2
3 namespace: traintomigrate
4
5 region:
6 | - bk-eks
7
8 dataAssets:
9 | uses:
10 |   - name: "sample_data_asset"
11 |     version: "1.0"
12 |     materialization: hive.bdx.sample_data_asset.sample_test_data_v1
13
14 steps:
15 | - name: aggregate
16 |   template: pyspark
17 |   config:
18 |     mainPythonFileUri: aggregate.py
19 |     packages:
20 |       pip:
21 |         - bkng-bdx[spark]
22 |         - pendulum
23 |     args:
24 |       - "--nominal_date"
25 |       - "{{ data_interval_start }}"
26 |       - "--filter_date"
27 |       - "{{ macros.ds_add(ds, -1) }}"
28
29 dataAssets:
30 | publishes:
31 |   - name: "traintomigrate.sparktraining0829.ant"
32 |     version: "1.0"
33 |     materialization: hive.bdx.traintomigrate.sparktraining0829_ant_v1
34 |     production_mode: full_refresh
35 |     period: DAY
36
```



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

```
1 interval: daily
2
3 namespace: traintomigrate
4
5 region:
6 | - bk-eks
7
8 dataAssets:
9 | uses:
10 |   - name: "sample_data_asset"
11 |     version: "1.0"
12 |     materialization: hive.bdx.sample_data_asset.sample_test_data_v1
13
14 steps:
15 | - name: aggregate
16 |   template: pyspark
17 |   config:
18 |     mainPythonFileUri: aggregate.py
19 |     packages:
20 |       pip:
21 |         - bkng-bdx[spark]
22 |         - pendulum
23 |     args:
24 |       - "--nominal_date"
25 |       - "{{ data_interval_start }}"
26 |       - "--filter_date"
27 |       - "{{ macros.ds_add(ds, -1) }}"
28
29 dataAssets:
30 | publishes:
31 |   - name: "traintomigrate.sparktraining0829.ant"
32 |     version: "1.0"
33 |     materialization: hive.bdx.traintomigrate.sparktraining0829_ant_v1
34 |     production_mode: full_refresh
35 |     period: DAY
36
```

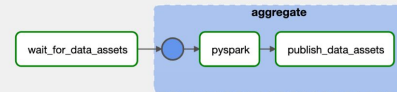
DAG: traintomigrate-spark-ant-bk-eks

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code Audit Log

2024-08-27T13:25:50+00:00 Runs 25 Run manual_2024-08-27T11:25:49+00:00 Layout Left > Right Update

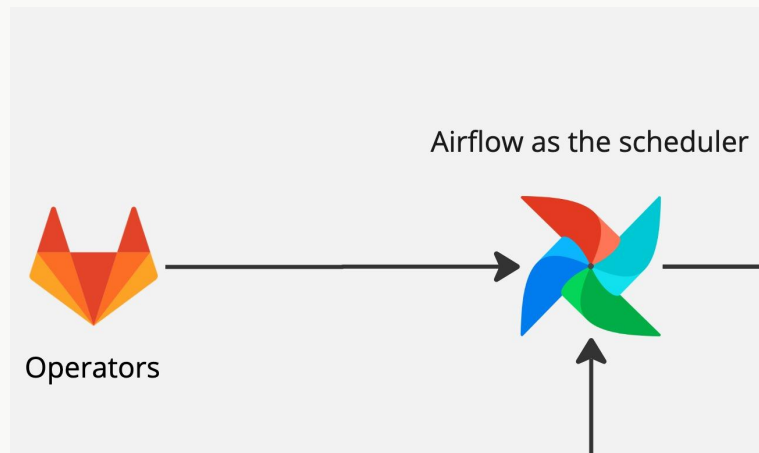
ComputeV1Operator DaaSV2PublishOperator DaaSV2WaitOperator

deferred failed queued running scheduled skipped success up_for_resch



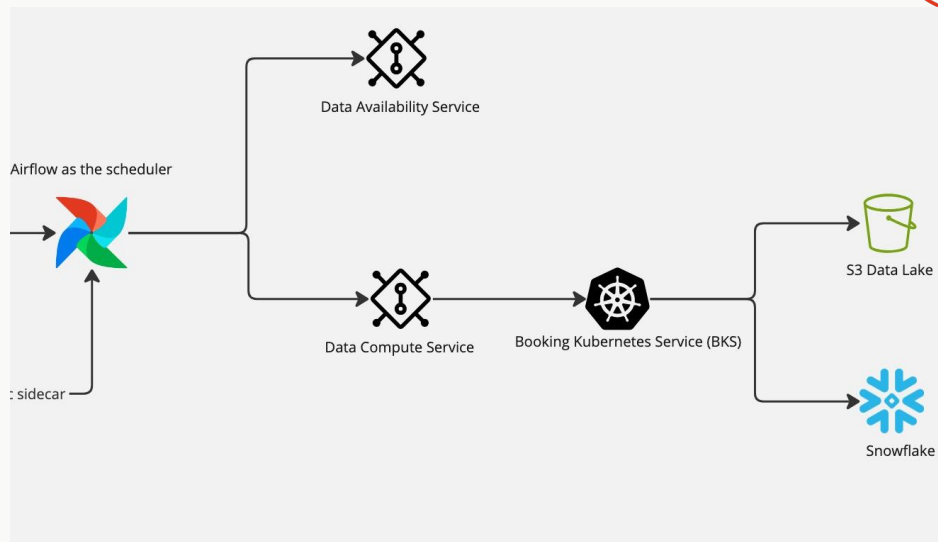
Workflow Steps - Deferrable Operators

- 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



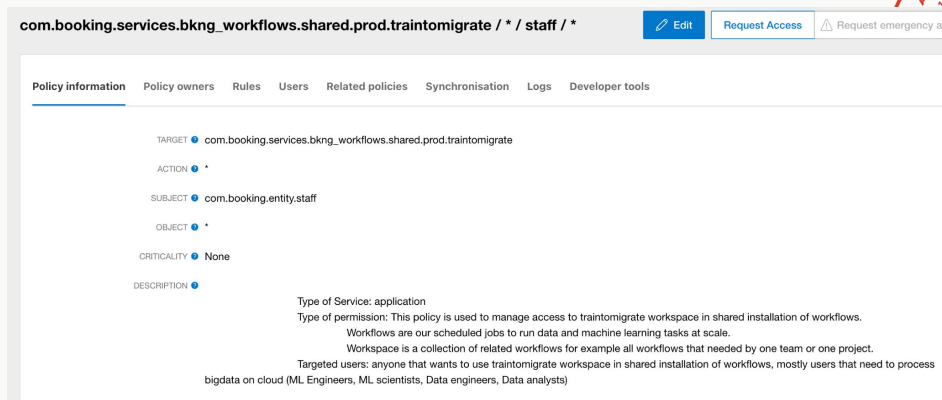
Airflow as pure orchestrator

- 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



com.booking.services.bkng_workflows.shared.prod.traintomigrate / * / staff / *

Policy information | Policy owners | Rules | Users | Related policies | Synchronisation | Logs | Developer tools

TARGET: com.booking.services.bkng_workflows.shared.prod.traintomigrate

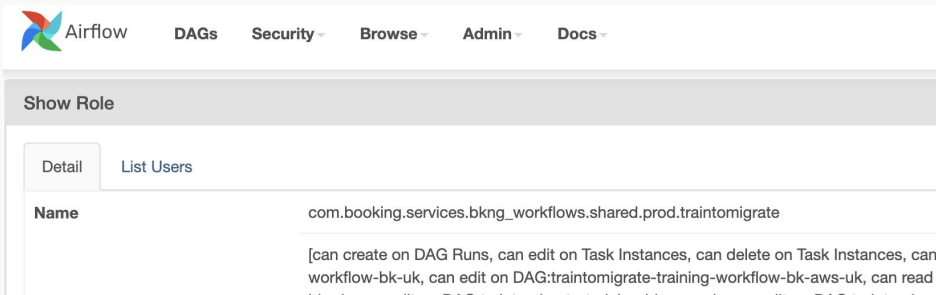
ACTION: *

SUBJECT: com.booking.entity.staff

OBJECT: *

CRITICALITY: None

DESCRIPTION: Type of Service: application
Type of permission: This policy is used to manage access to traintomigrate workspace in shared installation of workflows. Workflows are our scheduled jobs to run data and machine learning tasks at scale. Workspace is a collection of related workflows for example all workflows that needed by one team or one project. Targeted users: anyone that wants to use traintomigrate workspace in shared installation of workflows, mostly users that need to process bigdata on cloud (ML Engineers, ML scientists, Data engineers, Data analysts)



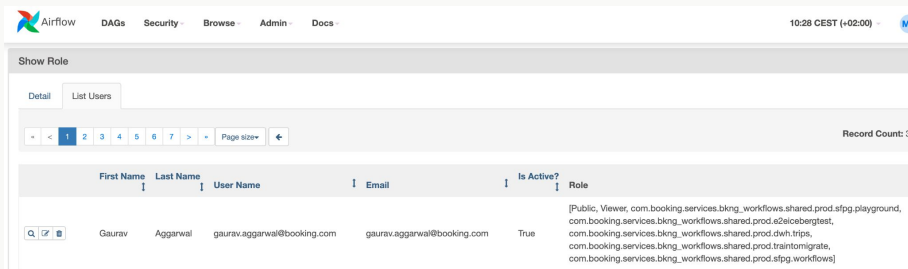
Airflow | DAGs | Security | Browse | Admin | Docs

Show Role

Detail | List Users

Name: com.booking.services.bkng_workflows.shared.prod.traintomigrate

[can create on DAG Runs, can edit on Task Instances, can delete on Task Instances, can workflow-bk-uk, can edit on DAG:traintomigrate-training-workflow-bk-aws-uk, can read



Airflow | DAGs | Security | Browse | Admin | Docs | 10:28 CEST (+02:00)

Show Role

Detail | List Users

Page size: 5

First Name	Last Name	User Name	Email	Is Active?	Role
Gaurav	Aggarwal	gaurav.aggarwal@booking.com	gaurav.aggarwal@booking.com	True	[Public, Viewer, com.booking.services.bkng_workflows.shared.prod.sfgp.playground, com.booking.services.bkng_workflows.shared.prod.a2icebertest, com.booking.services.bkng_workflows.shared.prod.dwh.trips, com.booking.services.bkng_workflows.shared.prod.traintomigrate, com.booking.services.bkng_workflows.shared.prod.sfgp.workflows]

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)

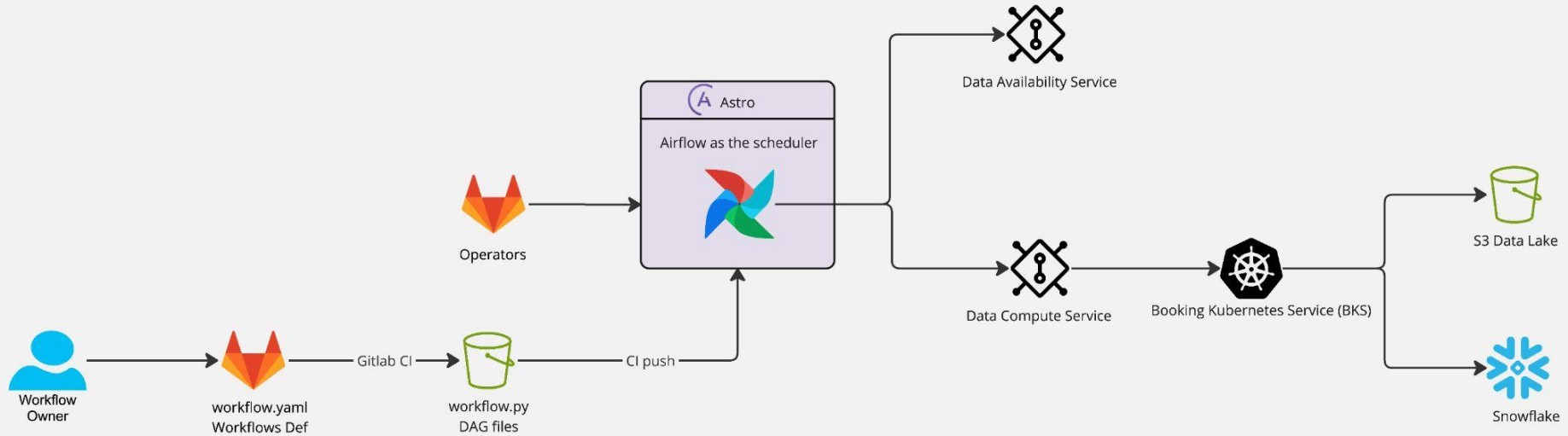
def send_alert(dag_id, task_id, execution_date, text, exception=None):
    workflow = dag_id
    logs_url = f"/log?dag_id={dag_id}&task_id={task_id}&execution_date={urllib.parse.quote(execution_date)}"
    requests.post("https://alertapi.booking.com/api/message", json={
        "name": f"persona.b_kng-workflows_airflow.workflows.{workflow}.{execution_date}",
        "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
        }
    })
```

```
112 success_callback_functions = []
113 failure_callback_functions = [send_failure_alert]
114
115 def generate_dag_callback(callback_functions):
116     def execute_callback_functions(context):
117         for function in callback_functions:
118             function(context)
119     return execute_callback_functions
120
121 default_args = dict(
122     depends_on_past=False,
123     retries=1,
124     retry_delay=timedelta(minutes=5),
125     provide_context=True,
126     execution_timeout=timedelta(hours=24, minutes=random.randint(12,20)),
127     retry_exponential_backoff=False,
128     on_success_callback=generate_dag_callback(success_callback_functions),
129     on_failure_callback=generate_dag_callback(failure_callback_functions),
130     sla=None,
131 )
```

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?

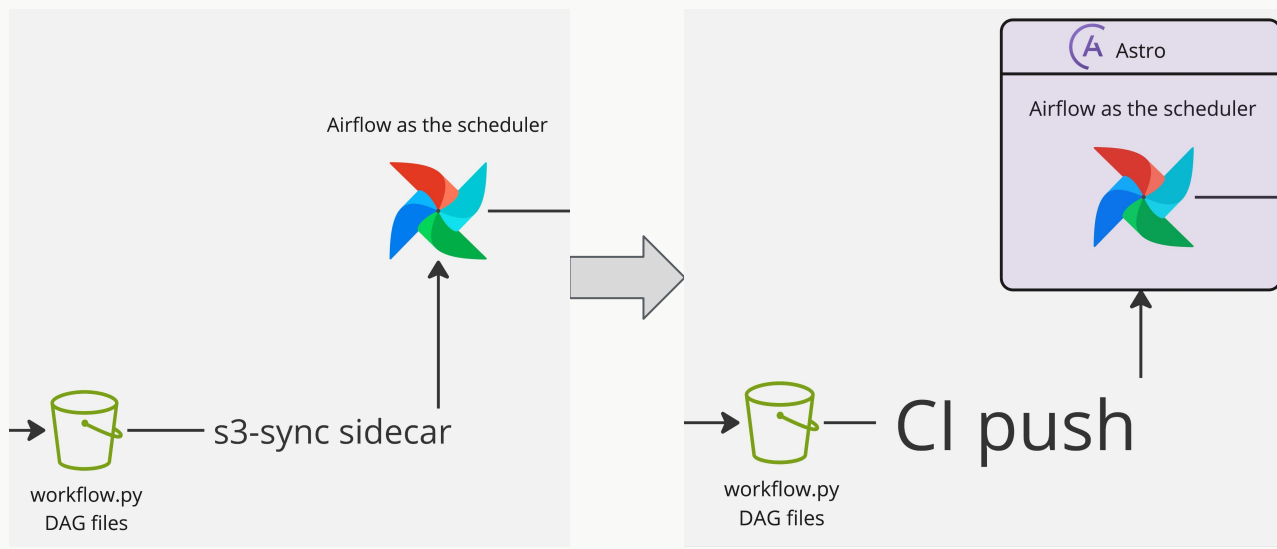
Network integration for accessing internal systems

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](#)