Building Airflow
Jelle Munk
Amsterdam Area
Software developer
Tech Lead Core Data Infrastructure
Adyen?
Traditional value chain

Merchant → Gateway → Risk management → Processing & Acquiring → Creditcard schemes → Issuers
Adyen Value Chain
One modern platform
Accept payments everywhere

Adyen Acquiring
United States
Canada
Puerto Rico
Brazil
Australia
New Zealand

Partner Acquiring
Europe
Japan
Hong Kong
Malaysia
Singapore
The challenge
Data Warehouse

Our applications scale horizontally, our data store did not..
Let’s fix that!

Build a central data platform where all data can be stored, transformed and turned into value for our customers.
(On prem) Data Platform

Events

Config

ML

Reporting

Insights
Problem solved?

Our new bottleneck was shipping these precious artifacts back to our ‘Application Platform’.
Time to get to business..
The naive approach

Just treat it as yet another file processing problem.
The naive approach

Just treat it as yet another file processing problem..

hdfs://<NAME>/out

hdfs://<NAME>/done

copy

copy

/<NAME>/in

/<NAME>/out
Many downsides

It works but it is far from optimal.

When can I start?
Is it complete?
Where are we stuck?

Can I undo & rerun?
Enter Airflow EventStream

A write-ahead event log for all state changes of airflow tasks.

- Execution date
- DAG Name
- Task Name
- State
- ...
Custom PushOperator

For ‘Shipping’ artifacts and/or other metadata

```json
{
  "dagName": "DAG_NAME",
  "taskName": "TASK_NAME_OR_MODEL_NAME",
  "time": "NOTIFICATION_TIME",
  "executionDate": "AIRFLOW_EXECUTION_DATE",
  "requestId": "REQUEST_ID",
  "start": "START_DATE",
  "end": "END_DATE",
  "metadata": "METADATA",
  "payload": "PAYLOAD_MESSAGE_ONLY_JSON",
  "payloadURI": "PAYLOAD_FILES_LOCATION",
  "files": {
    "FILE_NAME": "MD5"
  }
}
```
Simple Consumer Pattern

Single consumer per DAG

Consumer is responsible for ‘transport’
(updating customer portal, notifying customer or simply ingesting the data)

Consumer is responsible for storing it's state
(i.e. how far have am I in reading the queue)

Eventual consistency, both systems can go down for maintenance without anyone noticing

Free integration with our ‘monitoring’ tooling.
Separation of concern

**Big Data Platform:** keep track of dependencies, splitting the work in tasks and scheduling these (i.e. it manages the factory in which the goodies are produced)

**AirflowConsumer:** represents the customer and is responsible for delivering the final product to the doorstep, once it gets notified the goods are produced. But does not interfere with the production itself.
How did we build it?
Airflow Plugin

To expose REST API with all notifications
(pending a proper event streaming platform that is available on both ends)

For adding automatic notifications on all tasks that do not add a notification (like the PushOperator)

```python
@event.listens_for(sessionmaker, "loaded_as_persistent")
def intercept_loaded_as_persistent(session, object_):
    print("object loaded into persistent state: \$s" % object_)
```
Questions?