A startup's data journey
and its growing need for orchestration
Airflow Summit 2022
● 20+ years swimming in data @
● Started Apache Airflow at Airbnb in 2014
● Started Apache Superset at Airbnb in 2015
● Started Preset - The Apache Superset company in 2019
Preset’s Data Journey
● The visualization layer for the modern data stack!
● We offer a freemium + commercial offering for “managed” Apache Superset
● A 3Y old startup, series B company, ~50M raised, about ~75 employees
● SaaS, PLG, freemium, “bottoms-up”, self-serve
● data-native!
Phase 0 - stuff right out of the box
Pase 0.5 - some operational analytics

- Setup DataDog early-on
- Provided some product analytics by proxy
- Mostly downsides
  - Ephemeral
  - Not intended for business intelligence
  - Limited to internal systems
Beta launch -> Product analytics

- **analytics events**: looked into using BigQuery stream ingestion, but decided to wire Superset’s events into **Segment** -> BigQuery - acts as a more solid “transport layer”

- **scrapes**: had to build our own OLTP db -> BigQuery sync as we had unique challenges related to multi-tenancy (thousands of virtual databases)

- **control plane**: copied the pattern we used for Superset

- computing **engagement & growth** metrics (SQL & dbt)
Customer Data

- A huge need to bring CRM and product data together!
- Hubspot (our CRM)
  - Fivetran
  - Hightouch (reverse ETL)
Marketing data

- **Website traffic** - using Segment on our Gatsby site
- **SEO/SEM** - the need for tracking a single fingerprint across the journey
Other data sources vital to Preset

- **GitHub** for velocity and community information
- **Recurly** for our revenue data
- **Pendo** for our onboarding form, guides, in-product survey, ...
- **Sparkpost** for product email campaigns
- **Shortcut** as our Jira-like issue tracker
- **Lever** for our recruiting data
Pros

• simple -> stateless and infra-less (easy to setup & operate)
• incrementally adoptable - small data -> set up incremental load later...
• compatible / complementary with Airflow

Cons

• stateless (no logs!?)
• doesn’t work well with a functional approach
• No sensors (what did the data looked like when it ran!?)
Operational Creep

- Too many trains on their own schedule!
- Trains are leaving the station regardless of whether they are loaded or not
- No sensors!
- Dysfunctional data engineering
- The lazy approach
cacophony

/kəˈkæfənɪ/ noun

a harsh discordant mixture of sounds. "a cacophony of deafening alarm bells"

Similar: din racket noise discord dissonance discordance caterwauling
The case for Airflow

- Staying sane while data flow complexity grows
- Sensors waiting for conditions to be met before doing work
- Logs! Data Ops / rigor / lineage
- The growing need for little scripts that glue things together
- Enabling automating our first steps in ML / data science
A data team!

- Hired our first analytics engineer
- Hired our first data engineer early 2022
- Every team @ Preset is a data team
Thoughts about The Unbundling of Apache Airflow
Some closing statements

- data-native startups have access to world-class data infra through “Modern Data Stack” services!
- services are reasonably priced (dirt cheap really!), and things that used to be very hard are now pretty easy
- edge cases are common! Airflow still a great place to weave in custom things where off-the-shelf comes short
- complexity compounds very quickly!
- with complexity comes the need for an orchestrator!
$ # my most recent commit ->
$ git log --pretty=format:"%h%x09%an%x09%ad%x09%s" | grep Beauchemin | more | head -n 1
$ da76ac72e8  Maxime Beauchemin  Mon Sep 11 15:23:29 2017 +0200  [AIRFLOW-1476] add
$ # checking it out
$ git checkout da76ac72e8
$ # counting lines in the [repo]
$ git ls-files | xarqs cat | wc -l
$ 157011
$ # counting lines on most recent master
$ git checkout master; git ls-files | xarqs cat | wc -l
$ 556810